SemNet: the knowledge representation of lolita

Baring-Gould, Sengan

How to cite:

Use policy
The full-text may be used and/or reproduced, and given to third parties in any format or medium, without prior permission or charge, for personal research or study, educational, or not-for-profit purposes provided that:

- a full bibliographic reference is made to the original source
- a link is made to the metadata record in Durham E-Theses
- the full-text is not changed in any way

The full-text must not be sold in any format or medium without the formal permission of the copyright holders.
Please consult the full Durham E-Theses policy for further details.
SemNet: The Knowledge Representation of LOLITA

THESIS BODY
(Volume I/II)

The copyright of this thesis rests with the author. No quotation from it should be published without the written consent of the author and information derived from it should be acknowledged.

Sengan Baring-Gould

Ph.D. Thesis

University Of Durham
Department of Computer Science

2000

18 OCT 2000
Abstract

Many systems of Knowledge Representation exist, but none were designed specifically for general purpose large scale natural language processing. This thesis introduces a set of metrics to evaluate the suitability of representations for this purpose, derived from an analysis of the problems such processing introduces. These metrics address three broad categories of question: Is the representation sufficiently expressive to perform its task? What implications has its design on the architecture of the system using it? What inefficiencies are intrinsic to its design? An evaluation of existing Knowledge Representation systems reveals that none of them satisfies the needs of general purpose large scale natural language processing. To remedy this lack, this thesis develops a new representation: SemNet. SemNet benefits not only from the detailed requirements analysis but also from insights gained from its use as the core representation of the large scale general purpose system LOLITA (Large-scale Object-based Linguistic Interactor, Translator, and Analyser). The mapping process between Natural language and representation is presented in detail, showing that the representation achieves its goals in practice.

© Sengan Baring-Gould 2000

The copyright of this thesis rests with the author. No quotation from it should be published without his prior written consent and information derived from it should be acknowledged.
Contents

1 Methodology .................................................. 1
  1.1 Introduction .............................................. 1
  1.2 Program Complexity ...................................... 2
    1.2.1 Architecture: Large-Scale System .................. 2
    1.2.2 Efficiency .......................................... 3
    1.2.3 Robustness: a Possible Source of Complexity ...... 3
  1.3 Conceptual Complexity ................................... 4
    1.3.1 Integration of Conceptual Systems ................. 4
    1.3.2 Depth ............................................. 5
    1.3.3 Robustness: avoiding failure ...................... 6
  1.4 LOLITA .................................................. 7
  1.5 Problem to solve ....................................... 8

2 Foundations of Representation .......................... 10
  2.1 Assumptions ............................................. 10
  2.2 Knowledge: What are concepts and relations? .......... 11
  2.3 The "world": a hidden assumption ..................... 12
  2.4 Meaning space ......................................... 13
  2.5 Building relations ..................................... 14
  2.6 Good and Bad Partitions ................................ 15
  2.7 Properties forming concepts and hierarchies .......... 16
  2.8 Individuals and universals, and meta-properties ...... 19
  2.9 Fuzzy concepts .......................................... 21
  2.10 Recapitulation and Practical Consequences ........... 23
    2.10.1 The basis: links .................................. 23
3 Representation Requirements

3.1 Requirements for the K.B. ........................................... 27
  3.1.1 Large KB .................................................. 27
  3.1.2 The importance of search .................................. 28
  3.1.3 Requirements particular to NL systems .................. 28

3.2 Requirements for an NL system: Partitioning ............. 29
  3.2.1 Naturalness ................................................ 29
  3.2.2 Richness .................................................. 30
  3.2.3 Language-Independence .................................. 30
  3.2.4 Compositionality ......................................... 31

3.3 Requirements for a Large System .......................... 31
  3.3.1 Development issues ....................................... 31
    3.3.1.1 Correctness ....................................... 32
    3.3.1.2 Ease of use ....................................... 34
  3.3.2 Representation for a large project .................... 34
    3.3.2.1 Uniqueness ....................................... 34
    3.3.2.2 Cohesion .......................................... 35

3.4 Requirements for a large KB: Efficiency .................. 36
  3.4.1 Topological Properties .................................. 36
    3.4.1.1 Topological distance ............................. 36
    3.4.1.2 Determinism of Search ............................ 37
  3.4.2 Assumed properties of the interpretation ............ 37
    3.4.2.1 Non-linearity ..................................... 38
    3.4.2.2 Distributedness .................................. 38
    3.4.2.3 Distributedness and Non-linearity .............. 40
    3.4.2.4 Distributedness, Non-linearity, Compositionality and Cohesion ........................................ 41

3.5 Ranking the different requirements ........................ 42

3.6 Conclusion ...................................................... 43

4 Literature Review .................................................. 44
4.1 FOPL .............................................................. 44
4.2 CLE and CLARE .................................................. 46
  4.2.1 CLE's representations ..................................... 46
    4.2.1.1 Particularities of QLF ............................... 47
    4.2.1.2 LF syntax ............................................. 49
    4.2.1.3 QLF syntax ........................................... 52
  4.2.2 Separated knowledge in CLE .............................. 53
  4.2.3 Evaluation .................................................. 53
4.3 Conceptual Dependency ........................................ 55
  4.3.1 The representation ......................................... 56
  4.3.2 Inference .................................................... 57
  4.3.3 Conclusion .................................................. 58
    4.3.3.1 Primitives ............................................ 58
    4.3.3.2 Expectation-driven Inferencing ....................... 58
4.4 CGT .............................................................. 60
  4.4.1 Conceptual graphs ......................................... 60
  4.4.2 Arcs .......................................................... 60
  4.4.3 Nodes ........................................................ 61
  4.4.4 Types ........................................................ 62
  4.4.5 Complex statements ...................................... 64
  4.4.6 Evaluation .................................................. 66
4.5 SNePS ............................................................ 69
  4.5.1 SNePS: Philosophy .......................................... 69
  4.5.2 SNePS: Implementation .................................... 70
    4.5.2.1 Network elements .................................... 70
    4.5.2.2 Standard elements .................................. 73
  4.5.3 ANALOG ....................................................... 76
  4.5.4 Applications ................................................ 77
    4.5.4.1 The SNePS standard distribution ...................... 77
    4.5.4.2 CASSIE ............................................... 78
    4.5.4.3 Applications ......................................... 79
4.5.5 Evaluation ........................................... 80
  4.5.5.1 Insufficient richness of SNePSp ............... 80
  4.5.5.2 Variable nodes .................................. 81
  4.5.5.3 Uniqueness ..................................... 83
  4.5.5.4 Cohesion ....................................... 83
  4.5.5.5 Needs of an NL representation ................. 83
  4.5.5.6 Distributedness and Non linearity ............. 84
  4.5.5.7 Topological distance and Determinism of search .. 85
  4.5.5.8 Conclusion ..................................... 86

4.6 CYC ..................................................... 86
  4.6.1 The Representations ................................ 86
  4.6.2 Conclusion and Evaluation ......................... 87

4.7 KL-ONE .................................................. 88
  4.7.1 Representation .................................... 89
  4.7.2 Terminological Language ........................... 89
  4.7.3 Automatic classification and subsumption .......... 91
  4.7.4 Evaluation ....................................... 95

4.8 Defeasible Inheritance .................................. 95

4.9 LOLITA’s Semantic Network ............................. 97
  4.9.1 Philosophical assumptions ......................... 97
  4.9.2 Basis: nodes and arcs ............................. 98
  4.9.3 Quantification .................................... 99
    4.9.3.1 Individual quantification ..................... 99
    4.9.3.2 Universal quantification ..................... 99
    4.9.3.3 Existential quantification ................... 100
    4.9.3.4 Quantification of events .................... 101
  4.9.4 Sorts ............................................ 102
  4.9.5 Extended representation .......................... 105
  4.9.6 Evaluation ....................................... 106
    4.9.6.1 Insufficient Richness of LOLITA 92 ........ 106
    4.9.6.2 Low Distributedness .......................... 108
5 Basic Representation

5.1 Expressing information as a graph

5.1.1 Fundamentals

5.1.1.1 Graphs

5.1.1.2 Labelled edges

5.1.1.3 Arrows

5.1.1.4 Nodes

5.1.2 Events

5.1.2.1 The need for events

5.1.2.2 The solution: Events

5.1.2.3 Demonstration of Properties: some examples

5.1.2.4 The direction of events

5.2 Grammar

5.3 Introducing sets

5.3.1 Representation of sets

5.3.1.1 Elements of a set

5.3.1.2 Subsets and Supersets

5.3.2 Quantification

5.3.2.1 Structural Description

5.3.2.2 Use
5.3.3 Cardinality and Antonyms ........................................... 134
5.3.4 Inheritance ......................................................... 135
5.4 Defining concepts ................................................... 135
  5.4.1 Sorts ............................................................. 136
    5.4.1.1 Intensional Representation .................................. 136
    5.4.1.2 Introducing Sorts ............................................ 137
  5.4.2 Interaction of Sorts and Previous features .................... 137
    5.4.2.1 Quantification ............................................... 137
    5.4.2.2 Inheritance and the conceptual hierarchy .................. 138
    5.4.2.3 Sorts and Cardinality ....................................... 140
  5.4.3 Using sorts .................................................... 142
    5.4.3.1 Sorts and entities .......................................... 142
    5.4.3.2 Sorts for events ............................................. 144
    5.4.3.3 Expressing implicature ..................................... 152

6 Reasoning ................................................................. 156
  6.1 Common Element Rule .............................................. 156
    6.1.1 Description .................................................. 156
    6.1.2 Non-Linearity ................................................. 158
    6.1.3 Quantificational dependency ................................... 159
  6.2 Quantification and Cardinality ................................... 160
  6.3 Synonym Tracing .................................................. 161
    6.3.1 Sorts .......................................................... 161
    6.3.2 Synonym events can be qualified by events .................. 161
    6.3.3 Synonym events and the Belief control ....................... 162
    6.3.4 Synonym events and Believed events ......................... 163
    6.3.5 Copying Inferred Events ..................................... 163
  6.4 Inheritance ...................................................... 164
    6.4.1 Definition of Inheritance .................................... 165
    6.4.2 spec. events can be qualified by other events ............... 165
    6.4.3 spec. events and the Belief control ........................ 166
    6.4.4 spec. events and Believed events ........................... 167
6.4.5 Inheritance and Sorts .......................... 167
6.4.6 Inheritance and Quantification .................. 170
  6.4.6.1 Algorithm Outline .......................... 170
  6.4.6.2 Partial arcs .............................. 174
  6.4.6.3 Crossed quantification dependencies ........ 175
6.5 Type-Checking and Semantic Integration Assumptions .......... 176
  6.5.1 Depth ........................................ 176
    6.5.1.1 The desired type of depth ................. 176
    6.5.1.2 Annotating depth .......................... 178
    6.5.1.3 Maintaining depth .......................... 181
    6.5.1.4 Dealing with depth degeneration ............ 181
  6.5.2 Data-structures ................................ 182
    6.5.2.1 Shifted Circular Lists ..................... 182
    6.5.2.2 BIB Trees ................................. 184
6.6 Type-Checking .................................... 188
  6.6.1 Simple compatibility .......................... 188
  6.6.2 Underdefined $s_p$ ............................ 190
  6.6.3 Type-Checking is not the only consistency check ....... 192
6.7 Semantic Integration ................................ 193
  6.7.1 Introduction .................................. 193
    6.7.1.1 The need for classification ................. 193
    6.7.1.2 Problem Breakdown .......................... 194
    6.7.1.3 Disclaimer and Reading Conventions ........ 197
  6.7.2 Single Node Integration ........................ 199
    6.7.2.1 Downwards Integration ...................... 199
    6.7.2.2 Upwards Integration ......................... 206
  6.7.3 Real time integration .......................... 231
    6.7.3.1 Problem: Semantic Integration is inherently expensive .. 231
    6.7.3.2 The light at the end of the tunnel .......... 232
    6.7.3.3 ... is not an incoming train! ............... 233
    6.7.3.4 Implementation issues ...................... 235
6.7.3.5 Conclusion ................................................. 237
6.7.4 Multiple Node Integration ................................. 237
  6.7.4.1 Motivation ............................................. 237
  6.7.4.2 Nodes that must be integrated simultaneously ... 238
  6.7.4.3 Integrating subgraphs ................................. 241
  6.7.4.4 Opportunities for improving efficiency ............ 243
6.7.5 Subsumption in the light of KL-ONE’s results ............. 244
6.7.6 Conclusion .................................................. 246

7 Extended Representation ...................................... 247
  7.1 Representation of Values ................................... 247
    7.1.1 The need for Values .................................. 247
    7.1.2 Unquantifiable and Quantifiable Values ............. 248
    7.1.3 Values Representation .................................. 250
    7.1.4 Use ..................................................... 250
      7.1.4.1 Quantifiable Values ............................... 251
      7.1.4.2 Unquantifiable Values ............................. 251
      7.1.4.3 Order .............................................. 252
      7.1.4.4 Value Types ....................................... 252
      7.1.4.5 External Multiplication and Units .............. 252
      7.1.4.6 Uses .............................................. 253
    7.1.5 Conclusion ............................................. 255
  7.2 Stative Events ............................................. 255
    7.2.1 Belief ................................................ 256
      7.2.1.1 The belief event .................................. 256
      7.2.1.2 Belief chains ..................................... 256
      7.2.1.3 The belief control ................................ 258
      7.2.1.4 The belief control and the inheritance hierarchy 259
      7.2.1.5 Belief and certainty values ...................... 261
      7.2.1.6 Logical connectives ................................ 261
      7.2.1.7 Homogeneity ...................................... 262
    7.2.2 Communication ......................................... 263
7.2.3 Causal events .................................................. 264
7.2.4 Conclusion .................................................. 265
7.3 Negation ......................................................... 265
  7.3.1 The Antonym form of negation ......................... 266
  7.3.2 Absence of occurrence .................................. 267
  7.3.3 Negation: Conclusion ................................... 268
7.4 Parts ............................................................ 269
  7.4.1 Parts Representation ..................................... 269
  7.4.2 Intrinsic properties and Substances .................. 270
  7.4.3 Parts of events .......................................... 271
  7.4.4 Conclusion ................................................ 271
7.5 Time ............................................................ 271
  7.5.1 A time paradigm for N.L. ............................... 272
  7.5.2 The Time representation .................................. 273
    7.5.2.1 Times as concepts, represented by nodes ....... 273
    7.5.2.2 Associating events with times ..................... 273
    7.5.2.3 Instants and Intervals .............................. 274
    7.5.2.4 Representing chronology ............................ 276
    7.5.2.5 Composite events .................................... 278
    7.5.2.6 Times of template events ......................... 282
    7.5.2.7 Times and the Inheritance Hierarchy ............. 283
  7.5.3 Expressing Dates ......................................... 284
    7.5.3.1 Representing a calendar ............................ 284
    7.5.3.2 Multiple calendars .................................. 285
    7.5.3.3 Linguistic analysis .................................. 285
    7.5.3.4 Dates: Conclusion ................................... 286
  7.5.4 Tense ...................................................... 286
    7.5.4.1 The nature of tense ................................. 286
    7.5.4.2 Using a representation of tense ................... 288
  7.5.5 Conclusion ................................................ 289
7.6 Representations for Language ................................ 289
7.6.1 Linguistic and Conceptual Nodes ........................................ 289
7.6.2 Intension and Frames of Existence ................................. 290  
  7.6.2.1 Intension .................................................. 290  
  7.6.2.2 Frames of Existence ...................................... 291  
7.6.3 Textual References .................................................. 293  
  7.6.3.1 The idea .................................................. 293  
  7.6.3.2 The Representation ........................................ 293  
7.6.4 Ambiguity .......................................................... 294

8 Evaluation ........................................................................... 297

8.1 LOLITA: an Application of SemNet ....................................... 297  
  8.1.1 Lolita’s textual analysis ........................................ 298  
    8.1.1.1 Overview .................................................. 298  
    8.1.1.2 LOLITA’s role in SemNet’s design ..................... 300  
  8.1.2 Natural Language Generation ..................................... 301  
8.2 Semantic Mapping .......................................................... 303  
  8.2.1 Basic formalism .................................................. 303  
  8.2.2 Meta-properties .................................................. 304  
    8.2.2.1 Similar representation of Meta-properties and Pro- 
             perties .................................................... 304  
    8.2.2.2 Propositionality of meta-properties .................... 306  
  8.2.3 Example using Definitionals Sorts and Anaphoric Reference 
  8.2.4 Conclusion ...................................................... 309  
8.3 Comparison with other representations .............................. 310  
  8.3.1 Quantification .................................................... 310  
    8.3.1.1 Richness ................................................ 310  
    8.3.1.2 Cohesion ................................................ 324  
  8.3.2 Sorts and the Inheritance Hierarchy ............................ 329  
    8.3.2.1 Definition of concepts ................................ 330  
    8.3.2.2 Non-separative knowledge base ....................... 337  
    8.3.2.3 Inheritance hierarchy .................................. 343  
    8.3.2.4 Qualifiable Inheritance Hierarchy .................... 343
8.3.3 Values ........................................... 346
  8.3.3.1 Richness .................................. 346
  8.3.3.2 Reasoning .................................. 347
  8.3.3.3 Cardinality ................................ 348
  8.3.3.4 Negation .................................. 349

8.3.4 Belief ........................................... 351
  8.3.4.1 Logical connectives ....................... 351
  8.3.4.2 Stative events ............................. 352
  8.3.4.3 Belief and certainty ....................... 353

8.3.5 Ambiguity ....................................... 353

8.3.6 Time ............................................ 354
  8.3.6.1 Richness versus Allen’s Theory ........... 354
  8.3.6.2 Efficiency versus Allen’s Theory ........ 356
  8.3.6.3 Cohesion .................................. 359
  8.3.6.4 Textrefs .................................. 359

8.3.7 Parts ............................................ 360

8.3.8 Language ........................................ 361

8.3.9 Intensionality and Extensionality .......... 362

8.3.10 Frames of Existence ......................... 363

8.4 Essential Features ............................... 364
  8.4.1 Naturalness .................................. 364
  8.4.2 Richness .................................... 365
  8.4.3 Language Independence ...................... 366
  8.4.4 NL Analysis .................................. 369
  8.4.5 Reasoning .................................... 369
    8.4.5.1 Standard Inferences ..................... 369
    8.4.5.2 Building Inferences ...................... 370
    8.4.5.3 Other Inferences ......................... 370
    8.4.5.4 Conclusion ................................ 371

8.5 Scalability ....................................... 371
  8.5.1 Uniqueness .................................. 371
CONTENTS

8.5.1.1 Uniqueness of the Representation .................. 372
8.5.1.2 Maintenance of uniqueness ...................... 376
8.5.1.3 Uniqueness: Conclusion ......................... 377

8.5.2 Cohesion .............................................. 377

8.6 Efficiency .............................................. 381
8.6.1 Topological Distance ................................. 381
8.6.2 Determinism of Search ............................... 384
8.6.3 Distributedness ..................................... 384
8.6.4 Non-Linearity ....................................... 386

8.7 Features for Development ............................... 389
8.7.1 Ease of Use ......................................... 389
8.7.2 Compositionality .................................... 390
  8.7.2.1 High-level Compositionality .................... 390
  8.7.2.2 Low level Compositionality ...................... 391
  8.7.2.3 Conclusion ...................................... 392

8.8 Weaknesses ............................................ 393
8.8.1 Disjunctive Concept definitions .................... 393
8.8.2 Trading Efficiency for Uniqueness ................. 394

8.9 Conclusion ............................................ 394

9 Conclusion .................................................. 397
9.1 Review .................................................. 397
9.2 Future Work ........................................... 398

APPENDICES

A Assumptions ................................................. A-1
  A.1 Reasoning, and hence the symbolic paradigm ........ A-2
    A.1.1 Why is reasoning important? ..................... A-2
    A.1.2 Means of reasoning ................................ A-3
      A.1.2.1 The symbolic paradigm ...................... A-4

(VOLUME II)
A.1.2.2 Successes of the symbolic paradigm: Science and Mathematics ........................................... A-4
A.1.2.3 Lack of equally successful alternatives ................................................................. A-7

A.2 Symbolic, and other forms of Knowledge ................................................................. A-7
A.2.1 Scientific Knowledge ................................................................. A-8
A.2.2 Scientific knowledge: Too limitative? Alternatives? ............................................. A-11
A.2.3 Limitations of the Scientific model of Knowledge .................................................. A-12
A.2.3.1 Belief ................................................................. A-12
A.2.3.2 Inconsistencies ................................................................. A-14
A.2.3.3 Certainty ................................................................. A-15
A.2.4 Common Sense Symbolic Models ................................................................. A-16
A.2.4.1 Common Sense Knowledge Base: Difficulties and Requirements ............................ A-16
A.2.4.2 Mutual Dependencies ................................................................. A-16
A.2.4.3 New common sense models ................................................................. A-17
A.2.4.4 Implicit and Explicit Models: Important differences ........................................ A-18
A.2.4.5 Example ................................................................. A-19
A.2.4.6 Requirements particular to LOLITA ................................................................. A-21
A.2.5 CYC: a case study ................................................................. A-22
A.2.5.1 CYC's functionality ................................................................. A-22
A.2.5.2 CYC's knowledge ................................................................. A-24
A.2.6 Conclusion ................................................................. A-27

A.3 Information extraction? ................................................................. A-28
A.3.1 Understanding ................................................................. A-28
A.3.2 Interpretation ................................................................. A-32

B Basic Representation: Detailed Issues ................................................................. B-1

B.1 Quantification ................................................................. B-1
B.1.1 Attempt 1: Quantification on the node ? ................................................................. B-1
B.1.2 Attempt 2: Simplified Quantification on the arc ................................................ B-2
B.1.2.1 Three basic quantifications ................................................................. B-2
B.1.2.2 Common element rule ................................................................. B-4
B.1.2.3 Shorthand quantifications ................................................................. B-4
B.1.2.4 The use of quantification on the arcs ....... B-6
B.1.2.5 Problems ...................................... B-9
B.1.3 Final Quantification Scheme .................. B-10
  B.1.3.1 Quantification Dependencies ............... B-11
  B.1.3.2 Sets of sets of ................................ B-13
B.1.3.3 Quantification and Intensionality ............ B-13
  B.1.3.4 Logical existential quantification ........ B-14
B.2 Proof that $F$-$F$ events are bijective ........... B-17
B.3 Implementation of the basic representation ....... B-19
  B.3.1 Events ........................................ B-19
  B.3.2 Templates and action of arcs ................. B-21
  B.3.3 Families ..................................... B-21
B.4 Sorts .............................................. B-23
  B.4.1 The need for sorts at each end of each arc .... B-23
    B.4.1.1 A Control on Events ....................... B-23
    B.4.1.2 Problems with Sorts on Events: Transitive Events .... B-24
    B.4.1.3 Associating a sort to each arc ............ B-25
    B.4.1.4 Problems with associating only one sort to each arc B-25
  B.4.2 Why sorts? ................................... B-26

C Further Reasoning Results ......................... C-1
  C.1 A depthless algorithm to determine the orange node C-1
    C.1.1 Ensuring all paths reach the orange node .... C-2
    C.1.2 A data-structure to determine the common node C-3
    C.1.3 Finding the common node ..................... C-5
    C.1.4 Optimising the search ........................ C-6
    C.1.5 Overall complexity .......................... C-7
    C.1.6 Minimising the worst case .................... C-9
    C.1.7 Finding the orange node: a conclusion ....... C-11
  C.2 A Formal Examination of the Complexity of Semantic Integration C-11
    C.2.1 A model ..................................... C-12
    C.2.1.1 The virtual inheritance lattice ............ C-12
D Extended Representation: Detailed Issues

D.1 Values

D.1.1 Representation Design and Basic Richness

D.1.1.1 Assumptions

D.1.1.2 Design

D.1.1.3 Phenomena’s ranges

D.1.2 Network realization

D.1.2.1 Basic design

D.1.3 The use of values

D.1.3.1 Set Cardinality

D.1.3.2 Sensations

D.1.3.3 The use of values in language

D.1.3.4 Belief and certainty

D.2 Negation

D.2.1 Use of Absence of Occurrence

D.2.1.1 Naturalness of Absence of occurrence

D.2.1.2 Defined Absence and Uniqueness

D.2.2 Problems in Performing Negation

D.2.2.1 Conversion to antonym depends on quantification and sorts

D.2.2.2 Preservation of event arity

D.2.2.3 The treatment of defined individuals

D.2.2.4 Difficulties associated with unique existentials

D.2.2.5 Effect of the network being propositional
D.2.2.6 Implications of negating a definitional event ........ D-37
D.2.2.7 Conclusion: conversion to antonym ............... D-39
D.2.2.8 Negating events by absence of occurrence ....... D-39

D.3 Relating concepts to their possible parts .............. D-41
  D.3.1 Parts of concepts ................................ D-41
    D.3.1.1 Motivation .................................. D-41
    D.3.1.2 Properties of division ....................... D-42
    D.3.1.3 The need for a new relation ................. D-43
    D.3.1.4 Substances .................................. D-43
    D.3.1.5 Granularity of division ...................... D-43

D.3.2 The has_part event ................................ D-44
D.3.3 Using has_part .................................. D-45
    D.3.3.1 Representing substances and intrinsic properties D-45
    D.3.3.2 has_part for events .......................... D-47

D.4 The representation of Time ............................. D-50
  D.4.1 Time and language: A new paradigm ............... D-50
    D.4.1.1 Scientific model ............................. D-50
    D.4.1.2 Forcing the Distinction between Intervals and Instant... D-50
    D.4.1.3 Precise times ................................ D-51
    D.4.1.4 Time as an axis ................................ D-53
    D.4.1.5 Perceptual and Real Instants ................. D-54
    D.4.1.6 Event density ................................ D-55
    D.4.1.7 The need for instants and for intervals ....... D-56
    D.4.1.8 The ends of intervals ......................... D-56
    D.4.1.9 Representing the time of an event ........... D-57

D.4.2 Example Templates ................................ D-58

D.4.3 Time and the Inheritance Hierarchy ................. D-59
  D.4.3.1 Time and size ................................ D-60
  D.4.3.2 Time and the inheritance hierarchy ........... D-60
  D.4.3.3 The singular role of time ..................... D-61
  D.4.3.4 Representing the Gregorian calendar .......... D-62
D.4.4 Aspect ...................................................... D-66
D.4.4.1 Aspect and Tense ..................................... D-66
D.4.4.2 Aspect and Classification of Verbs .............. D-67
D.4.4.3 Language dependence of Aspect .................. D-69

D.5 Linguistic and Conceptual Nodes ...................... D-69
D.5.1 Linguistic nodes ....................................... D-70
D.5.1.1 The problem ......................................... D-70
D.5.1.2 The solution: linguistic nodes .................... D-70
D.5.1.3 The use of linguistic nodes ....................... D-70

D.5.2 Expressing language dependent information .......... D-74
D.5.2.1 Controls on linguistic nodes ...................... D-74
D.5.2.2 Relations between linguistic nodes .............. D-74

D.5.3 Sorts and linguistic nodes .......................... D-75

D.6 Intension, Extension and Frames of Existence ........ D-75
D.6.1 Intensionality and Extensionality .................. D-75
D.6.2 Intensional and Extensional equality ............... D-76

D.6.3 Synonyms and Uniqueness ............................ D-78
D.6.3.1 Definitional and Observational Sorts .......... D-78
D.6.3.2 Set Union: union_ ................................ D-78
D.6.3.3 Antonyms: antonym_ .............................. D-80
D.6.3.4 Powerset: powerset_ and infinite_powerset ...... D-81
D.6.3.5 No Uniqueness Break for spec_ .................. D-82

D.6.4 Frames of existence .................................. D-83
D.6.4.1 The need for frames of existence ................ D-83
D.6.4.2 The use of frames of existence ................... D-85
D.6.4.3 Intension as pan-frame intension ................. D-89
D.6.4.4 The choice of frames of existence ............... D-93

D.6.5 The idea of a concept ............................... D-97
D.6.6 Definitional and observational sorts: a matter of expression ? D-99

D.7 Textual References ..................................... D-100
D.7.1 Textual references: The problem .................... D-100
D.7.1.1 Text Structure ............................................. D-100
D.7.1.2 Statements involving the text’s representation .... D-105
D.7.1.3 Linking Concepts to Text Structure .................. D-106
D.7.2 Textrefs: A Representation of Text Structure .......... D-110
  D.7.2.1 Shared aspects with time ............................ D-110
  D.7.2.2 Differences to time ................................. D-111
D.7.3 The use of textrefs ........................................ D-114
  D.7.3.1 Interpretations ...................................... D-114
  D.7.3.2 Keeping track of the words forming LOLITA’s facts D-116
  D.7.3.3 Coreference tasks .................................... D-116
  D.7.3.4 Information Retrieval ............................... D-117
D.7.4 Textrefs: Conclusion ...................................... D-118
D.8 Ambiguity ................................................... D-118
  D.8.1 The type of ambiguity to represent .................... D-118
    D.8.1.1 Ambiguity in Natural Language .................... D-118
    D.8.1.2 A simple attempt: Global ambiguity ............... D-119
    D.8.1.3 Sharing and alternative arcs: Local ambiguity ... D-120
    D.8.1.4 Reflecting the locality of ambiguity in the represen-
           tation .................................................. D-121
  D.8.2 The nature of ambiguity modelled ...................... D-121
  D.8.3 A general representation of ambiguity ............... D-122
    D.8.3.1 Representing interpretation ........................ D-123
    D.8.3.2 Variable nodes ..................................... D-124
  D.8.4 Packed Structures ..................................... D-128
    D.8.4.1 Correlations in ambiguous statements ............. D-128
    D.8.4.2 Belief and Interpretation ........................ D-130
    D.8.4.3 Dismissing interpretations ........................ D-130
    D.8.4.4 Structuring ambiguity ............................. D-132
  D.8.5 Hidden Assumptions ................................... D-135
  D.8.6 Features of the representation for disambiguation .... D-136
    D.8.6.1 “The shortest path...” ............................ D-136
    D.8.6.2 A detailed example: “The astronomer married the star” D-137
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>D.8.6.3</td>
<td>Maintaining underspecified variable nodes</td>
<td>D-138</td>
</tr>
<tr>
<td>D.8.6.4</td>
<td>Node count</td>
<td>D-138</td>
</tr>
<tr>
<td>D.8.7</td>
<td>Ambiguity: Conclusion</td>
<td>D-142</td>
</tr>
</tbody>
</table>
List of Figures

4.1 CGT - version of the Donkey Sentence ........................................ 64
4.2 ANALOG - version of the Donkey Sentence ................................. 77
4.3 Every Farmer who owns a donkey beats it .................................. 104

5.1 Examples of Events ....................................................................... 120
5.2 Observed features ......................................................................... 143
5.3 The person who kissed Mary was Jacob ....................................... 145
5.4 Template Buying event .................................................................. 147

6.1 Every person's father loves that person's mother. .......................... 159

7.1 There are no men who can read and write ..................................... 267
7.2 I did not eat any sandwich in Paris between the 12th and 15th of April ................................................................. 268

8.1 SemNet is at the core of LOLITA ................................................. 298
8.2 Cohesion ....................................................................................... 380

C.1 A company employs employees at different factories .................. C-11
C.2 Quantification Examples .............................................................. C-16
C.3 Quantification Examples .............................................................. C-17

E.1 Frames of existence ..................................................................... E-85
List of Tables

4.1 Summary of Surveyed Representations' features .......................... 113

5.1 Notation: Event Symbols ........................................ 125

6.1 Types of nodes in Semantic Integration ............................... 199

8.1 Representations' Richnesses ......................................... 366
8.2 Language Independence .............................................. 367
8.3 Standard Inferences .................................................. 370
8.4 Breaks of uniqueness ................................................. 372
8.5 Topological distance of SemNet events ............................... 383
8.6 Conceivable Breaks of Distributedness .............................. 386
8.7 Types of SemNet events .............................................. 388
8.8 Summary of All Representations' features .......................... 395

E.1 References using structure ......................................... E-103
Acknowledgements

This thesis would not have been possible without all the people who contribute free software, or free Internet access to papers. In particular, our work relies heavily on \LaTeX, the Glasgow Haskell Compiler, the Chalmers University Haskell Compiler, software from the FSF, MIT (X11R6), LINUX and many others. I also used the E-Print and DFKI Internet paper archives extensively, and hope other archives will become freely available.

It would also not have been possible without the existence of LOLITA, and hence Prof. Garigliano, her initiator, and the rest of the LOLITA team, to whom she also owes a lot. All of the above, my family and Simone my wife, have helped me through good times and bad, and without them, this thesis would also not exist.
Statement

This work builds on the semantic network implemented in LOLITA when I joined the project in 1992. This network was developed by Dr Derek Long and Dr Roberto Garigliano, but is substantially different from that presented in this thesis:

- Quantification was expressed on the nodes.
- Belief status was represented as an arc.
- Many of the concepts now expressed as events were expressed as arcs making it impossible to assign them belief (size, set inclusion etc)
- Most of the extensions discussed in this thesis did not exist (time, values, ambiguity, negation by absence, etc)
- There was no notion of semantic integration as a basic operation on the network, although there was the notion of actions (rather than events) forming an inheritance hierarchy of a sort.
- There was no notion of cohesion being a guiding principle in the construction of a large representation.

I shall refer to their network as LOLITA 92, as it appears in pre-1992 revisions of LOLITA. My work to later revisions of the network make it bare more resemblance to this thesis, although they do not fully implement it in all its details, due to lack of resources.

Similarly, although the ideas of non-linearity and distributedness were first introduced by Drs Long and Garigliano, they were not as well-defined or as rigorously
applied to the representation as has been done in this thesis. Again, although there is an idea of nodes representing concepts rather than sets, the consequence that the representation is intensional (the morning star and the evening star are different concepts – different nodes) was not realized in LOLITA 92.
Reading conventions

Some of the assumptions of this thesis are little known or unpublished. They either may be taken as given or can be read in the Appendices. Similarly, demonstrations of certain properties of the representation, such as richness, are quite lengthy due to the need to demonstrate the wide range of expression that can be achieved using combinations of the representational constructs. These experimental results of a sort, are therefore also presented in the Appendices. For ease of reference, a detailed index is included at the end of the second volume.

It proves convenient to consider any entity capable of reasoning to be an "agent". Similarly it proves convenient to use different personal pronouns to improve the ease of expression. Thus the pronoun "she" will be assigned to the LOLITA tool. The pronoun "he" will be assigned to the agent being discussed. This leaves the neutral pronoun "it" free as a general purpose referent.

Because graphs cannot always be projected clearly onto two-dimensional paper, if two nodes of a figure have the same label, they refer to the same node of the graph.

It will be assumed that for any agent, remembering a fact incurs a cost \(^1\).

Recently, the author changed his surname from Short to Baring-Gould. References to his papers are cited under his former surname.

\(^1\)This is because the final goal is to produce a working tool, and that currently there is no known method of storing an infinite amount of knowledge. Note though, that no assumption is being made that this is possible either. Since a working tool is the final goal, if one wishes to start building it now, one must assume that the resulting system will have a limited memory.
Chapter 1

Methodology

1.1 Introduction

The difficulty in building a large-scale general-purpose Natural Language (NL) system stems from program and conceptual system complexity. Large-scale general-purpose NL systems use a wide range of types of processing: complex lexing, string parsing, tree-manipulations, a wide variety of reasoning methods, etc. They also use a wide range of conceptual systems that must work together in a transparent manner: for instance the grammar must use the terms the lexer produces, which means the grammar must be tailored to the conceptual system the lexer pre-assumes. The overall complexity and functionality of the system depends on how well its components take into account each other’s needs.

This research investigates Knowledge Representation (KR) for large-scale general-purpose NL systems: KR that addresses the difficulties outlined above. A KR expresses statements in terms of a (possibly implicit) conceptual system. This thesis details a set of requirements that the form and the conceptual system of a KR adapted to large-scale general-purpose NL systems must satisfy. As no KR is found that satisfies these requirements, a new KR is proposed: SemNet.

This chapter is organized as follows: 1.2 analyzes why large-scale general-purpose NL systems are complex from a programming standpoint. 1.3 delves into the con-
ceptual system complexity to be tackled. And 1.4 presents LOLITA, the research system used to develop this work.

1.2 Program Complexity

A large-scale general-purpose NL system must grapple with program complexity. One must determine how to break the problem into independent components, ensuring each has access to the information it requires. Moreover one must do so efficiently since a large-scale system must process vasts amounts of information.

1.2.1 Architecture: Large-Scale System

Because a general purpose system must be able to deal with information from many domains, it must perform deep analysis. This requires it to handle a large number of discriminating rules. These rules create cross dependencies as the result from one rule may be used by another. As the number of cross dependencies increases, the maintainability and flexibility of the system decreases. As the scale increases, so does the cost of maintenance and testing.

The complexity of cross-dependencies can be mastered by breaking the problem into components. Each component performs a specific task, leading to information that can be used by other components. The input and output of a component are clearly constrained to a particular task. A component should be expressed as simply as the task it describes allows. No two components should perform similar tasks: minimising the number of components reduces the cost of maintenance and testing while ensuring each improvement to a component improves the system's overall performance. In NL, there is agreement on the initial text-processing tasks and the associated components: tag a text, parse it, map it into a semantic representation.

The architecture of toy systems inherently limits their potential growth. Extending toy systems requires adding ad hoc extensions as the need arises, becoming unmaintainable as the cross-dependencies become too complex to follow, and usu-
ally resulting in a complete change of architecture\(^1\). However toy systems are cheap to build since they can involve more shortcuts, where the general situation is not dealt with.

The hallmark of large scale systems is the ease with which new rules can be added to extend their coverage. The effort invested in providing a general purpose foundation shared by many rules can be reused by new extensions, subsidising later development and reducing overall complexity. Because components are shared by many rules, they are more general, reducing the likelihood an extension will need to change them. Similarly, the representation of a large scale system is able to express more types of information since similar tasks are processed by one component manipulating one representation. The difficulty is to divide the problem into the smallest number of simple components performing well defined independent tasks.

Notice that high coverage does not imply a large-scale system.

1.2.2 Efficiency

General-purpose systems must process information as expected in a wide variety of situations. This requires them to cope with multiple-domains, resulting in vasts amounts of information. For instance, the knowledge base of a NL system may encode hundreds of properties known of each of hundreds of thousands of concepts. Much of this information is irrelevant at any one point of processing, but needed to provide a wide coverage. Similarly, processing may result in vasts amounts of intermediary results, for instance to allow a highly modularised system. Dealing efficiently with vast amounts of information, organising it for fast retrieval, and so on, is a research subject in itself.

1.2.3 Robustness: a Possible Source of Complexity

Robustness is often thought of as a final polishing activity: ensuring a wide enough coverage for a domain, testing extensively, and debugging any problems that crop

\(^1\)for much the same reason as explained in 2.6 (p. 15)
up. But this assumes that every possible input to the system has been considered in advance, a weak assumption. Indeed, ensuring robustness requires designing it into the system.

Most systems attempt to recover from errors (e.g.: Verbmobil [Wahlster et al. 92] [Görz et al. 96] and Pangloss [Nirenburg 95] [Brown 96]). This involves either extracting information from the intermediary results of the component that failed, or reprocessing the component’s input in another way, often resulting in lower precision or certainty. This increases program complexity and requires the standard and error recovery components to agree.

1.3 Conceptual Complexity

A large-scale general-purpose NL system uses many conceptual systems to express different stages of the transformation of NL into its knowledge representation. These conceptual systems must be designed so that information expressed in one can easily be used by another. They must also allow a wide range of expression to distinguish the wide of phenomena that occur in NL.

1.3.1 Integration of Conceptual Systems

The transformation of NL into a system’s KR involves distinct steps. Each step derives a particular type of information from another type of information. For instance, morphology determines each word’s root form and features (number, gender, case, etc). This derived information expresses choices within a particular conceptual system. Each step uses input information, which is also expressed in terms of choices within a different conceptual system. For the intended behaviour to occur, every component that uses information expressed in terms of a particular conceptual system must agree on it having the same meaning. Most NL systems do not agree on the conceptual systems they use, precluding directly exchanging knowledge between different systems, such as grammars, semantic rules, and so on.
Similarly, within a given conceptual system, different components must agree on
the meaning of the choices the conceptual system makes. For instance different
reasoning algorithms that use the cause primitive must agree on its meaning and
intended usage. Not doing so leads to very subtle and hard-to-find bugs which only
surface once an inconsistency is detected.

Because it is so important not to confuse the semantics of NL system’s conceptual
systems, minimising the complexity and number of the conceptual systems used
improves maintainability and flexibility. For instance a system built of specialised
reasoning components which each manipulates a different KR adds three additional
requirements: all relevant information must be mappable from one representation
to another so that no reasoning capabilities are lost; all the transformations from
information in one KR to another must be correct; relationships between informa-
tion expressed in different KRs must be maintained.

1.3.2 Depth

The depth of a system depends on the range of behaviours it is (or will be) capable
of displaying. Certain differences in input information lead to differences in the
behaviour that the system is expected to display. If a system is incapable of
recognising these differences, it will not be able to display the full range of desired
behaviours. Increasing the depth of a system leads to an increased initial cost of
development: the conceptual systems must be able to express a wider range of
choices, and the rules that implement them must be more discriminating. As in
the case of program complexity, the increased initial cost offsets a higher later cost:
to extend a less deep system to the range of behaviours of a deeper system requires
rewriting all the existing rules to be more precise, and adding further rules. The
question is not one of the actual coverage of possible forms of input information,
but is rather the notion of how many behaviours a system could display without
changing all the rules that produce the systems various behaviours\textsuperscript{2}.

The conjunction of a large scale and deep system leads to a general purpose system.

\textsuperscript{2}in contradistinction to the rules used to transform input information itself
The coverage of a general purpose system must span many domains so that it can process information as expected in a wide variety of situations. This requires it to be deep: it must distinguish every information difference in its input that will lead to a difference in its expected behaviour. Moreover a large coverage involves the maintenance of a lot of domain-specific information and rules, requiring a large-scale architecture.

1.3.3 Robustness: avoiding failure

The conceptual systems of a NL system are usually not designed to consider robustness. Instead they limit the representation to only those forms which are expected to be produced by the processing algorithms. Thus, there is no way for the algorithm to express the error within the representation. Instead it must abort. An example of such a system is the attribute grammar of a C-compiler which can only produce a parse-tree for a valid C program, and must abort otherwise.

However this is not the only option: one can ensure that each feature in the input data is detected independently of all others by a given component. Furthermore each feature is expressed independently from all others in the component’s output in such a manner that later processes can test for the existence of any feature independently from all others. Each feature can also be assigned a certainty and/or precision, which further processes can use to estimate these values for their conclusions. Additional processes may hypothesise the existence of some likely feature from the existence of others, if it is missing but needed for later processing, at the cost of decreased certainty. Independent features, certainty values, and an order of evaluation dictated by the available information go a long way to achieving robustness. Determining independent features, separating out the way in which each is detected and determining good hypotheses involves a lot of work, and may involve more steps than a more efficient but less robust algorithm.
1.4 LOLITA

LOLITA is a research tool built to investigate the program and conceptual complexity of large-scale general-purpose NL systems, which tends to increase exponentially [Boguraev et al. 95]. LOLITA research belongs to the field of N.L.E., or Natural Language Engineering. N.L.E. is a sub-field of A.I. which discusses methods of mastering the complexity inherent in large-scale general-purpose NL projects, just as software engineering addresses the complexity inherent in large projects with tools such as Design Patterns [Gamma et al. 95]. [Boguraev et al. 95] provides a detailed overview of N.L.E. Artificial Intelligence (A.I.) in this thesis is considered to be concerned with reproducing successful human behaviour.

This research was performed using the LOLITA system as an experimental vehicle. Since LOLITA is a research tool, flexibility is essential: LOLITA must enable different approaches to be tried out as experiments. Indeed, since the theory of NL is relatively young and every NL problem cannot be solved in an established way, radically different solutions may be attempted. For instance, strategies for pronoun resolution can be based on reasoning, on discourse, or on syntactic information. Implementing many different strategies is simplest when the information is easily accessible in the same knowledge base. Thus, LOLITA’s design requires a single knowledge base that all components can access.

Requiring a single knowledge base for a NL system makes the design of the KR a difficult research task, as the KR’s design impinges heavily on the design of all other components: as it is common to all components, any severe design errors could force a rewrite of large parts of the system. To maximise flexibility, components should use the knowledge representation as a native language in which to reason, rather than translating information to and from process specific languages. If all components share the same representation, they can be combined efficiently to produce complex reasoning behaviour as discussed in A.3.1 (p. A-28): this encourages system integration. This means that different algorithms should not require a lot of preprocessing before they can be used. This is particularly true

3Large-scale Object-based Linguistic Interactor, Translator and Analyser
if different algorithms must share the same preprocessing, as it can indicate that
a concept that algorithms wish to refer to explicitly is implicit in the representa-
tion, indicating a bad partition. The representation must strike the right balance
between conflicting needs of different algorithms to maintain high efficiency and
sufficient richness.

Thus the first key question was whether a suitable representation already existed.
To answer this, the notion of suitable must be defined clearly. This is done in 3 (p.
27). Then a survey of existing representations is carried out in 4 (p. 44).

To ensure the representation addressed the needs of different system components I
worked with the LOLITA team members responsible for LOLITA's different com-
ponents. In this respect, I collaborated with Mark Smith (Generation [Smith 95]),
Simon Shiu (Reasoning [Shiu 96]), Sanjay Poria (Reasoning [Poria 97]), Agnieszka
Urbanovitz (Discourse processing [Urbanovitz 95]), Roberto Garigliao (Grammar,
Reasoning...[Long et al. 93]) and Rick Morgan (Template Extraction [Long et al. 93]).
In this way, it was possible to understand problems their algorithms faced when
processing the knowledge representation. Furthermore I wrote the semantic anal-
ysis component of LOLITA that builds the KR from grammatical parses. It is
crucial for this component, unlike reasoning, that the representation be suitable
for NL. Indeed it is quite clear whether or not the KR can express a given sentence
if your component must build an expression for it. Thus building this component
is a particularly good way of testing the NL credentials of the representation.

1.5 Problem to solve

A research-worthy problem was identified: to find a knowledge representation for
a large-scale general-purpose natural language system. An experimental vehicle to
ground the research was also determined: the LOLITA NL research system.

The rest of the thesis is organized as follows: Chapter 2 investigates the foundations
of representation. Chapter 3 determines a set of requirements for a representation
used by a large-scale general-purpose NL system. Chapter 4 tests existing KRs
against Chapter 3's requirements, and finds that none of them are satisfactory. Chapters 5 and 7 present a new representation: SemNet. Chapter 6 presents the reasoning machinery assumed by SemNet. Chapter 8 evaluates SemNet in light of Chapter 3's requirements and Chapter 9 concludes.
Chapter 2

Foundations of Representation

Having assumed that symbolic knowledge suffices for N.L.E., this chapter discusses the inner nature of representation: partitioning the "world" into a set of distinguishable concepts. The resulting conclusions will determine the general shape of LOLITA’s representation.

2.1 Assumptions

There are many opportunities for design error in ambitious projects such as LOLITA. As shown in many studies, such as those by [Sommerville 92], design errors prove the most difficult and expensive to correct, often requiring large parts of an implementation to be rewritten.

In order to avoid such frustrating and wasteful setbacks, a principled approach is preferred where possible: If at an abstract level, it is possible to show that large classes of possible solutions present inherent problems rendering them unsuitable for use in a system such as LOLITA, many costly mistakes can be avoided. Appendix A (p. A-1) examines in detail the ideas central to the rest of the thesis. In this manner, it avoids hidden assumptions and clarifies the discussion. The main assumptions follow:

A general purpose system such as LOLITA can be designed following a two part model: a natural language text is first analysed, and the results are added to
a knowledge base. The thus obtained knowledge can then be processed. The key resulting feature is that all processing manipulates knowledge of a single knowledge base.

To merit the term "understanding", the "natural language analysis phase" must express the information of the text in terms of the existing concepts in the knowledge base. This involves complex reasoning, and is better described as interpretation.

It is assumed that decisions affecting the design of the knowledge base and benefiting reasoning will also benefit many of the other forms of knowledge processing that LOLITA must perform. Because of their success and agent independence, symbolic reasoning techniques are adopted to implement LOLITA's reasoning. As a result, LOLITA’s knowledge base is symbolic. Features and examples of symbolic common sense models, needed for text interpretation and inspired from scientific models are discussed in A.2 (p. A-7). The main difference with scientific models, is that LOLITA’s knowledge base must support different beliefs, certainties and have the means to cope with inconsistencies.

Finally, partly inspired from the following discussion, no ontology is considered absolute: the choice of ontology depends only on the reasoning capabilities of LOLITA and the type of behaviours she is expected to display.

2.2 Knowledge: What are concepts and relations?

Appendix A (p. A-1) mentions vague ideas such as "relation", "concept" and "meaning" being the basic units of knowledge, but it did not define them precisely. Only their effects have been considered: predictive ability. The lack of any definition bodes ill in a scientific or engineering discussion. Thus it is important to dedicate some space to resolving these ideas. Indeed, they will prove crucial to the design of the whole system.
2.3 The "world": a hidden assumption

The assumptions have taken for granted the possibility that prediction is possible and is the task of any reasoning system. However, such an assumption is based on the idea that there are relations that somehow hold in the world. This in turn relies on the notion that the world is homogeneous, or more strictly that the sensory perception of the agent delivers a homogeneous view of whatever is going on in the "external world", whether or not such a thing exists. The reason for this is that a relation is a homogeneous pattern assuming certain fixed reference points to be detected. Without any such reference points, no pattern could be detected, and thus no relation formed, which precludes any prediction.

It should be noted that references to "the world" in this thesis refer to that which is perceived through the agent's senses, rather than to any "external" reality. This means no claims are made about the nature or existence of such a thing as "an external reality" or "a world". However, the discussion becomes quickly clouded by verbose formulations such as "the agent's interpretation of his sensory perception", so the concept of "world" will be used as a convenient shorthand for the sum of the agent's perception and interpretation of this perception.

Within the requirements of homogeneity of perception is also that of persistence. Reasoning occurs over a certain time lapse, and uses a set of relations built up over time. If sensory perception were homogeneous at every instant, but patterns detected the previous instant could not be used the next, reasoning would achieve nothing since its conclusions would be useless by the time they were derived. Thus, the requirement of persistence is strengthened to the extent that it is possible to build up a stable pattern of relations believed to hold in the ("world"), and reason with them.

So what is homogeneity? Homogeneity is the condition for relations to be detected from perceptual input, or patterns to be derived from sensory data. Sensory data as such is not useful. It is only when it is organised into some pattern that it can be used for reasoning, making choices, and thus gives a degree of control. Thus it must be interpreted. In order for patterns or relations to be detected, some fixed
points must first be recognised within the sensory data. This corresponds to being able to identify concepts before being able to detect relations between them. The question is therefore to determine potential fixed points.

At this point, the only information available is the sensory data, so any fixed points can only be detected through properties applied to this data. These will provide a basis for sorting the data. For instance, consider a property which detects sharp changes in colour over the full set of data provided by the visual sense. Such a property could be useful for detecting potential edges of objects. Complex combinations of such properties may thus be used to identify various apparently fixed patterns, such as tables, people, and so on. These will serve as concepts. Thus concepts would appear to be a means of identifying patterns within sensory data, and could be understood as functions mapping sensory data to a value True or False, depending on whether the pattern was detected or not.

It should be noted that the properties considered to restrict the scope of concepts may involve behaviours. Indeed, the notion of sensory perception includes anything that can be felt by the agent. Thus it includes emotions, memory, sensing his body such as his own breathing.

2.4 Meaning space

The picture that emerges is of a set of sensory data which can be partitioned into subsets of the sensory data using a set of combinations of these basic properties. Since the "external world" is not the subject of study, the full range of possible sensory data must be considered. This means that the subsets recognised by the conceptual functions will include possibilities never before encountered. In this sense, the combination of properties acts as a filter stating which parts of the sensory data are relevant to the concept and which are not. Those that are not, are left unspecified, unconstrained, allowing the concept to be recognised in varying circumstances.

Building on this picture, the view is that of a meaning space consisting of all the
combinations of sensory inputs that can be generated using the full range of each sense. This space is partitioned into chunks, each corresponding to a concept, by the concept functions. Any experience must fit within this space, since the senses are taken to be the means by which experiences are perceived. Since this space is unconstrained, it shall be referred to as typeless. Similarly constrained spaces can be viewed as typed. The whole process can be seen as partitioning the sensory input, i.e. partitioning the world into discrete objects.

2.5 Building relations

Once fixed points, i.e. concepts have been built, relations between them may be established. These again are derived from the sensory data and the capacity to identify single concepts. They correspond to meta-patterns involving concepts rather than particular sensory data, and thus are not limited to defining particular concepts. Instead they allow concepts to be joined one to another. However these meta-patterns do also depend on the sensory data. For instance, the position in the visual data where one concept is identified with respect to another will determine whether the corresponding object is on top, or next to the other. Similarly, a subjective understanding of time often plays a role in this process.

Hence it is assumed that the same process is involved in the building of relations as is in the building of simple concepts. According to this point of view, relations and simple concepts are both concepts. To avoid confusion, the former will be called "events", in connection with the fact they were derived from or represent a particular occurrence or process. The latter will be called "entities", to remind us that they correspond to some form of object.

Once some of these meta-patterns or relations are in turn defined, they may themselves be used to define new concepts. For instance, the difference between a teacher and a miner lies not in the properties one can detect directly from the sensory data, but from their behaviour: the behaviour of someone teaching should be very different from that of someone digging into a coal-face.
Another example of partition playing the role of a constraining property is that of time. The understanding of time is derived from sensory perception. Sensory perception is in constant flux, with various cycles, such as the rates of breathing and heart beat. In most situations, noise is not constant, and what is perceived visually changes. By associating sensory perception with a memory of the last processed instant, the changes can be noticed. This in turn leads to a notion of time, based on internal and external changes perceived by the agent. However the partitioning “time” is so basic, it also plays the role of a constraining property for many other concepts, as seen above.

2.6 Good and Bad Partitions

The picture does not include determining which concepts to build or how to build them. Obviously there are an infinite number of ways of building such concepts. However only some of them may prove useful. The problem is therefore one of determining useful concepts given a particular task or motivation. The main task considered here is that of prediction leading to control. This means that the constraints determined in A (p. A-1) for any predictive system also apply here.

Consider how a bad partitioning can adversely affect the building of a knowledge base following the principles previously discussed based on the notion that maintaining a concept incurs a cost. An analogy will provide a useful illustration: The problem is to build a large square by sticking either small squares or small circles on a piece of paper. It is easy to produce a perfect square using small squares. The number of small circles one needs tends to infinity as the required precision is increased. In the same way, the number of relations required to compensate for initially poor partitions increases as the required degree of accuracy from a prediction increases. Ultimately, the set of additional relations create a new implicit concept: that of the good partition, to some degree of precision. But they do so at the cost of a higher memory requirement, the lack of an explicit concept, and a needless increase to the cost of reasoning: more relations need to be considered.
Hence the choice of partitioning is very important. Obviously in an adaptive system, this is not so problematic, as when a particular partition proves expensive to maintain, the partitions may be reconsidered and recomputed, and the knowledge base can be appropriately restructured. However for static systems, which are unable to adapt in this way, the weight imposed by an inappropriate partition will increase with the size of the knowledge base. It should be noted that unless restructuring is performed by some automatic process, it can be very costly. This is an important point, in that it forms the basis of the claim that much attention must be payed to the design of the representation and that of LOLITA beforehand, where possible, in order to avoid later expensive restructuring of the whole system.

2.7 Properties forming concepts and hierarchies

The discussion has focused on concepts being determined by a conjunction of properties. The simplest such model is that of a concept uniquely restricted by a property formed by conjoining a set of simpler properties. For instance one such property would be “anything that can produce a tactile sensation within a certain range is a solid object”. For more constrained properties, a larger number of simpler properties would be involved, so that the concept of an animal would involve many properties simply to identify a single part of it, such as its head. Since an animal must have all these properties, it will require a very large total number of properties. The concept for a particular animal, such as a bear will be even further restricted, so that it only identifies bears. However since it is also an animal it will have all the properties required for animals. Hence, many concepts will share the same constraining properties.

This observation suggests a useful way of organising concepts: if a set of concepts have as some of their restricting properties all those of another, they can all be related to this other. This link is an “is a” relation, corresponding to the fact all these more defined concepts are specialisations (or more restricted versions) of this other. For instance, the concept of “bear” could be related to that of “animal” by an “is a” relation in the knowledge base. In this manner, every concept known
to the system is connected by such relations to the concept representing the full possible universe: typeless. This set of relations is an inheritance hierarchy. For ease of reference, the most general concepts will be said to lie at the top of this hierarchy, whereas the most specialised will lie at its bottom. Hence, typeless is the topmost concept. Notice that all concepts are more specialised than typeless, so all will be included in the hierarchy.

What does such an organisation provide as benefits? The first obvious benefit is that it allows restrictive information to be expressed only once for a whole class of concepts. For instance the information defining animals need only be expressed at the level of the animal concept, rather than also for each concept which is an animal. Similarly once a fact is learnt about all animals, only one relation need be built to express it, rather than one for each animal concept. This corresponds to the picture of a meaning space developed earlier: if a relation or property holds for a restriction of meaning space, i.e. a chunk of meaning space, it also holds for any more restricted chunk within it. Every concept can be assigned the full list of properties and relations in which it is involved by a simple form of reasoning which considers only "is a" relations.

Another benefit comes from the observation that problems can often find simple solutions at the right level of "granularity". This was also mentioned in the the discussion of scientific models (A.2 (p. A-7)) where granularity was considered to be an issue of tautologies: statements consisting of complex combinations of others. Thus, if certain such statements were used as primitives for reasoning, the result would be obtained more simply, than if their parts were considered. This observation stems from the fact that the most primitive statements in the knowledge base, or most detailed models are too precise: they provide too much detail in which to describe the situation, thus introducing irrelevant information to the reasoning process, confusing it. Over-general models, on the other hand provide too little detail with which to express the situation clearly which means that even to hope to reach the right conclusion, they must be shored up using additional information. An example of the former case would be trying to solve a billiard ball game using General relativity, and of the latter would be to understand semiconductors using
classical mechanics. In both cases the problem is that of information: too much or too little. The degree to which a concept is restricted corresponds to the amount of information that is known about it. Thus, the inheritance hierarchy provides an easy mechanism with which the precise amount of information needed for some reasoning task can be varied: concepts further down the inheritance hierarchy have a higher granularity.

Each concept corresponds to a level of granularity particularly useful for some types of reasoning: most of the restrictive properties of any concept are shared by its ancestors. These ancestors must be useful for prediction to be worth remembering. Thus, they must express relations believed to hold in the world at their particular level of granularity which is useful to reason with.

Problems expressed at some level of granularity can often be solved at a coarser level. Indeed, most of the restrictions defining any concept usually are also true of its ancestors. Take, for example, that one wants to know whether a lecturer pays taxes. It is known that a lecturer is employed, so is an employee. Moreover it is known that employees get a salary. Finally it is known that people who get a salary must pay taxes. Hence one can deduce that all lecturers, and in particular the lecturer concerned, pay taxes. Notice that the level of granularity at which the reasoning took place was coarser than that of a single lecturer: it occurred at the level of employees. Reasoning at a coarser level of granularity is cheaper. At any level of granularity, only statements with same granularity or coarser need to be considered: only a subset of the knowledge base need be considered. Indeed, at finer levels of granularity, concepts are more restricted than those considered in the question, so conclusions that hold for them need not hold for all of the cases expressed in the question.

This idea of solving the problem at a coarser level of granularity can be further exploited by using the fact that what is true for a general concept must be true for all its specialisations. Thus although what may be deduced will also most often be general (it is rare to have a model which gives specific information from general data), it will provide an approximate answer to the problem. This information may
in turn be used as a basis to reduce the part of the knowledge base that must be considered in more precise reasoning. In this way, a locality for reasoning can be constructed which is limited to relevant information, either directly by considering only information of coarser granularity or indirectly by considering information related to that used when deriving a general conclusion from the coarser data. Obviously if absolute certainty of the results is required, more of the knowledge base can be considered to ensure nothing relevant is left out, but when this condition is relaxed, the proposed scheme allows for cheaper reasoning.

In a similar vein, unsafe reasoning methods which make assumptions, or approximations may prove useful for providing lower certainty results, or sketching out the shape of a proof. An example of this principle being exploited is the analogy algorithm: It is known that miners' jobs are threatened, but one wants to know whether this is also the case for printers. Because miners and printers share a large number of relatively specialised properties at a low granularity, such as working in a labour-intensive industry with old technology and heavy machinery, printers' jobs can be deduced also to be at risk\(^1\). Although the restrictions that miners and printers do not share could invalidate this conclusion, they are few and, for the majority, not specific to miners or printers. Thus they are discounted. Overall the conclusion is less certain than one deduced from all the available information and performed at a finer level of granularity. But, given the information it used, and thus the speed it achieved, if the analogy algorithm provides an answer which is not far from that achieved by a valid method using more information and more time, it is useful.

### 2.8 Individuals and universals, and meta-properties

The description given so far of properties is that they are used to detect patterns in the sensory data, and thus to create concepts recognising that pattern and using it as a basis for recognising other patterns: relations. Now, this works very well

---

\(^1\) Example taken from [Long et al. 93]
for trees, about which one knows a certain number of facts which one can use for reasoning. But does it work so well for individuals, such as one's dog? There seems to be a large difference between individuals and these universal type concepts. In fact, although this difference seems very strong, it still fits within the framework of concepts as restricted by properties: if one knows an individual, such as a friend, one is capable of recognising him. Hence, there is a set of properties that can be applied to one's sensory input data, and which can detect his presence. However, if one suddenly saw two of him, one would get a shock. Hence, one's friend has associated with him the information that he cannot be present twice in one's sensory data.

More generally, one may associate any concept with a certain size, corresponding to the maximum number of instances one believes could be present simultaneously in one's sensory data. This condition may be extended somewhat, by extending one's concept locality according to one's experience: one would not expect someone to be phoning from New York and being due to give a lecture in Durham within a few minutes. Moreover, it would seem from experience that one learns a set of properties that one believes is sufficient to distinguish between any instance of a concept one believes to be universal. For instance, consider the following situation: one is looking at a eagle and someone next to one says "Oh, isn't it great?", and then leaves. But if then he reappears, this time wearing different clothes, and says something similar, one would be entitled to wonder at his apparent short memory. Yet, this disconcerting experience could happen if you had just met a pair of twins, one at a time. Similarly, to people who have not learnt to recognise European, African or Chinese faces, they may all seem the same.

This observation is important in the design of the knowledge base, since whereas most of the properties discussed so far have involved recognising an instance of a concept, or express some fact about such instances, this property is expresses something about the concept itself. In this sense it is a meta-property. Although it has not been explicitly stated, another such meta-property has been discussed: the "is a" relation, which allows the inheritance hierarchy to be built.
2.9 Fuzzy concepts

The discussion has only considered the simple model of concept restriction where the property restricting the concept is the conjunction of several other simpler properties. Although this model has the advantage of simplicity, and thus ease of reasoning, it proves unfortunately too naive for many uses in practice. To see this, consider the example of an animal standing behind a tree. Although some of its body is hidden by the tree, a human agent may still be able to identify that it is a horse. However, with the conjoined model, all the properties must be satisfied in order for the concept to be recognised. This can simply be rectified by allowing the restricting property also to consist of disjoined simpler properties. However, such a solution is somewhat unsatisfactory, since it requires explicitly writing down all possible situations in which the concept will be recognised.

This is precisely the problem that [Garigliao 89], addressed in his homogeneity theory. Its main argument is that a concept may be restricted by a large set of partial restrictive properties, without the requirement for all of them to hold within the whole of its space. Instead, one requires each to hold more within the concept’s space, than without it. This is specified by using the notion of a property’s density, corresponding to the amount a property holds within a certain amount of meaning space. Hence, the properties can distinguish between the concept’s space and the rest of the world. The difference in the density of a property within the concept’s space and the rest of the world is called its fracture value. The full restrictive property at each point of the meaning space is defined by the conjunction of the subset of the partial properties that hold at that point. The amount that the density of a property varies over the concept’s space is called its homogeneity with respect to that concept. By specifying a minimal bound for the fracture value and the homogeneity, the area of meaning space occupied by the concept defined by the partial properties can be specified.

Homogeneity theory provides an alternative solution to the problem that some concepts appear difficult to express in terms of a unique restriction. Examples are the concepts “game”, and “energy”, which are defined by their usage and the similarity
of their occurrences. The classical solution is that proposed by [Wittgenstein 53] of setting up a prototype of this concept. This corresponds to a set of family resemblances which are used to form the concept. In Homogeneity theory however, this is expressed by a low homogeneity and low fracture value of the partial properties forming the full restrictive property. The fact that homogeneity and fracture values can be expressed explicitly allows concepts to be ordered in terms of fuzziness if so required.

Although this model deals with real world situations, it implies that some of the properties restricting a concept do not apply to all of its instances. In this sense the concept is fuzzy. This adds an element of uncertainty requiring an adapted type of reasoning algorithm. Since a need for this type of processing has not yet emerged for practical applications of LOLITA, at LOLITA's current level of development, this thesis will not concern itself further with this problem. However it will be required that the information needed for such processing be expressible in the knowledge base. This fits within the requirement for the design of the system to minimise the impact of known future developments.

A similar issue is that of the evolution of concepts with time. The concepts considered here are static. That is to say, once the agent has defined them he cannot change them, although he can learn new facts about them. This raises the issue of inappropriate partitioning and the increasing cost in order to maintain it in a growing knowledge base, which has already been discussed. Indeed the whole principle of completely static concepts is rather naive, since it is only when you know something about a concept that you can define it clearly. Instead an adaptive mechanism would be preferable, including a measure of how approximative the current definition of the concept is. Again, homogeneity theory's notions of fracture and homogeneity provide the required tools. Note that there is no assumption that the definition of the concept should converge to "the truth", but that the concept should represent a better partition, in that it provides a greater degree of control. This also applies for cases where the world changes slowly enough for concepts to be built and used, but over a shorter time than the agent's lifetime. This for instance is the case of people whose appearance changes over the years.
Again however, there is no need for this type of processing in current practical applications of LOLITA, given the extent of her development. Thus, this thesis will not concern itself further with this issue.

2.10 Recapitulation and Practical Consequences

After this discussion of the formation of concepts, it is useful to summarise the major points.

2.10.1 The basis: links

Concepts are defined either with respect to properties mapped onto sensory perception, or with respect to other concepts. The important point here is that these definitions, or the relations in which concepts participate form links between the concepts. It is these links that are fundamental to the knowledge base. It is they that define what the agent knows about each concept. Thus only the links to a concept express its participation in the agent’s model of the world. However, these links lead to other concepts, which in turn are determined by their links. Hence, a cyclic model emerges, where there are many links between symbols (the concepts), underpinned by a primitive level of properties mapped onto the sensory level.

Consider now the importance of this to reasoning. The motivation for creating concepts was to allow reasoning. Thus, the motivation of not only the definitions of concepts but also the facts about them is uniquely the ability to reason about the world. All these relations specify how any given concept relates to the rest of the world. Thus relations include the notion of what its behaviour should be, or facts useful to determining this. Reasoning is about determining what the agent expects the future behaviour of (instances of) some concepts to be. It must therefore use the facts available about the concepts. Since facts are expressed as links, only those links involving the concepts need be considered. Thus the search-space for reasoning can be reduced: only facts linked to the concepts under scrutiny need be considered.
Although the scheme just discussed has the advantage of a very small search space, it is very limited. Often it is not just the relations involving the concept under investigation that are important, but also the concepts involved in these relations. However these too are connected to other concepts, and so on, until eventually all the concepts of the knowledge base are considered. For large knowledge bases, as found in large scale systems, such searches are prohibitively expensive, and therefore would render reasoning impossible. It therefore becomes important to consider the issue of relevance of any information during the reasoning process. This relevance has two components.

The first type of reasoning-relevance is dependent on the type of reasoning process considered: temporal reasoning only deals with temporal information. Therefore, only temporal information is relevant to it. It might however be possible to use a general reasoning method to solve a question which only involves a certain kind of information. For instance, a general mathematic reasoning method might be used to calculate possible share prices. But it should only consider economic information when solving this question. It might also be able to solve other problems, such predicting the weather from the variation of temperatures over the last few months, but it should constrain itself to information relevant to the type of problem being solved.

The second type of reasoning-relevance is dependent on the concept under investigation. Consider the following situation: a murder has been committed, and the alibi of a suspect must be checked. Various witnesses have given the approximate time at which he was seen at various places, and their statements must be checked for consistency. Moreover, approximate durations are known to determine whether the various scenarios are temporally possible, for instance whether the suspect could have left the scene of the crime and arrived at his office within the time left. Many things may be known to have occurred at a similar time, such as the inspector’s children having been to school, a rock concert happening the same evening, the sun setting and so on. However not all these facts will be relevant to solving the problem. Thus the issue here is to determine which concepts are most related to the concept under investigation. Since the expression “more related to”
is clumsy, the term "associativity" will be used for this notion. Hence the second type of relevance for reasoning is associativity.

Are some concepts more related to a particular concept than others? This is one of the assumptions of the model of concepts proposed above: concepts are defined with respect to some properties, and are linked by relations to others. It is a valid assumption as no concept will be linked directly to every other\(^2\). A direct link here is one which links a concept to another by a single relation. The reason for this is that only those relations that prove useful to the agent can justify the cost required to maintain them or create them. If the agent needed to know all these relations in order to predict sufficiently well to fulfil his tasks, but was unable to store all these relations, he would not succeed at his tasks. From this perspective the whole process of predicting, including the creation of concepts, would have failed. Hence if one is dealing with limited agents for whom remembering something incurs a cost, one has no choice but to assume that some behaviours are more important to the agent, and will be conceptualised as a relation, and other are not and will not.

### 2.10.2 Cyclicity and Relativism

The model of concept making presented here agrees with the notion that concepts are not absolute. Indeed, here each agent is considered as partitioning the space of his sensations into concepts. Although these concepts may not vary over time, in cases where there is a need for them to do so, they can. Thus concepts correspond to hypothetical reference points, which help create a model of the world.

Just as reasoning and memory incur a cost, so does the partitioning process. Thus a concept need only be defined to the extent that it proves cost-beneficial: a concept must simply be adequately defined for the tasks the agent will undertake. Again the concept of adequacy is the key, as it was in the case of communication (see A.3.2 (p. A-32)). The tasks of interest to the agent will in turn depend on the agent's motivation. For instance, people interested in computers will have a more

\(^2\)Here it is assumed that a different relation is being considered for each such link, since otherwise the problem would be solved by a link to the least specified concept, typeless
precise definition of it than others: whereas the notion of programming may be more crucial to my definition of computers, to my mother computers appear more like physical boxes with a keyboard and a screen.

In the previous discussion, it was mentioned that relations may be defined in terms of restricting properties, but in turn may participate in the definition of concepts. An example of this was the notion of time. This is an important point since it will partly determine the choice of the knowledge base representation. Indeed, it gives an idea of the internal structure of knowledge: relations defining or expressing facts about concepts, which in turn either define or express facts about other concepts, and so on... Only the properties directly operating on the sensory input are not defined by other relations of the knowledge base. Hence, a picture emerges of a cyclic structure, ultimately reducible to properties operating on sensory data. Thus these properties correspond to the ultimate primitives. However, it is the organisation into a super-structure of mutually related concepts at many levels that gives the reasoning power considered in A.2 (p. A-7). The notion to remember is one of cyclicity underpinned by some primitives defined externally from the knowledge base.

One question remains: what should be done about the agent’s sensory primitives if the agent has a very limited sensory input. For an agent only lacking one sense, such as a blind person, colours may to a certain extent be perceived through another sense, such as that of heat. However LOLITA is an agent with only one sense, textual input. In this case, it is not reasonable to expect any primitive defining concepts to be based on the available sensory input. Thus the question becomes how to cope without this sensory basis. Again the question boils down to behaviour: if the agent is able to reason about the world, and if most of his predictions hold, he will be successful at controlling his environment and fulfilling his tasks. Since no two agents need share the same concepts, or indeed the same senses, the only question is whether they fulfil their tasks. If so, they will be considered to have understood what they needed to.
Chapter 3

Representation Requirements

An analysis of the requirements of a large scale NLE system's Knowledge Base leads to various criteria that the knowledge representation must satisfy. These criteria not only provide a basis for the evaluation of other NLE representations, but also guide the design of LOLITA's representation.

3.1 Requirements for the K.B.

This section discusses requirements LOLITA's K.B. must satisfy.

3.1.1 Large KB

In order to be useful in many situations, the LOLITA system should have a wide vocabulary. Since one of its tasks is translation, this vocabulary must include words from more than one language. As words may be ambiguous, each word may correspond to many concepts. Moreover, each of these needs to be defined, and used to express some facts. Indeed, only useful concepts are worth their cost in memory. The K.B. under consideration is therefore huge. This scale alone leads to engineering problems.
3.1.2 The importance of search

In a language processing system such as LOLITA, many forms of processing involve searching for information, which to a certain extent is unknown. For instance for a template extraction tool, particular types of information (temporal, spatial, etc...) are to be found, rather than precise questions answered. Similarly even during the interpretation phase, or for reasoning, facts relevant to the task being performed must be found. Since search is so important for so many tasks of LOLITA, it is essential that it should be as efficient as possible.

3.1.3 Requirements particular to NL systems

In natural language one can refer directly to concepts. For instance, “John saw five fish.” Here “five” is a meta-property as defined in 2.8 (p. 20) and qualifies the set of fish that John saw. As a result, the fish John saw must be represented by an explicit reference in the K.B.

Explicit references to all concepts are also needed for NL phenomena such as anaphoric reference, or ellipsis. For instance, “One of the people I work with, is John. John’s tall friends are loathsome people. They enjoy torture, betray each other regularly, and cause no end of trouble. I really don’t understand why he likes them.” Here reference has been made to the set of people I work with, John, and John’s friends. Thus, each should be associated with a single symbol.

Thus, all concepts that can be referred to in NL, must be expressed in the K.B. in such a way that they can easily be qualified or referred to.

Further, there should be no representational difference between meta-properties and other relations in which a concept is involved: these two kinds of relations are expressed in the same way in NL. For instance, “John believes there are small men in that room” and “John believes there are five men in that room”. In one case, a meta-property is the object of belief, in the other a property is.
3.2 Requirements for an NL system: Partitioning

In section 2.4 (p. 13), the importance of a good partitioning was stressed: A bad partitioning led to a multiplication of additional facts to patch it over. In the K.B., this partitioning not only corresponds to the definition of concepts, but also to the primitives used to define them. Thus the choice of the available primitives will determine not only how good the partitioning is, but also what can be expressed with the representation. These two aspects are respectively the naturalness and the richness of the representation.

3.2.1 Naturalness

Naturalness corresponds to the isomorphism of the representation to the domain it is trying to model. This englobes two important aspects:

- the quality of the partitioning, and therefore of the concepts expressed in the K.B.;
- the structure of the K.B. itself.

The first point has already been discussed, and the need for it established.

However the second point requires further discussion: The K.B. is trying to model the type of knowledge described in 2.2 (p. 11). Why should it reflect any structure this knowledge might exhibit? In pragmatic terms the question asks in what way such an isomorphism would improve control, or in this case the processing LOLITA is to perform. This in turn depends on the type of processing involved.

As discussed in 2.10.2 (p. 25), the structure of knowledge involving the definition and use of concepts is primarily cyclic: concepts are either defined by properties mapped onto sensory input or by relations with other concepts; and the facts they express were also conveyed by relations between them.
It was established in 2.10.1 (*p. 23*) that if reasoning is to be performed about a particular concept, only the links and concepts associated with this concept need be considered. All the information known about any concept is expressed by a set of links involving it. Therefore the cyclic structure of the knowledge considered provides a scheme where information associated with a concept is topologically near, in the sense that only a few links need be traversed to reach it. By further exploiting the way in which concepts are defined, it was shown that concepts could also be organised into an inheritance hierarchy which allowed reasoning to be performed at various levels of granularity: topologically, it separated generic information about the class of concepts to which a particular concept belongs, and information specific to that particular concept.

By having a K.B. organised topologically in the way just described, the search-space required to find information about any given concept is dramatically reduced. This is important, because tasks such as reasoning do not involve looking up a particular fact, but searching for facts relevant to the problem they are to solve. Thus reflecting the structure of the knowledge determined from its nature in the K.B. proves pragmatically useful.

### 3.2.2 Richness

As observed previously, the partitioning determines what can expressed in LOLITA’s K.B. This effectively determines the limits of what she can know, understand and reason about. For LOLITA to be a useful tool, it is essential that she be able to represent at the very least, a large proportion of the information in the natural language texts she is to process. The richness of a representation corresponds to the number of phenomena that it can express.

### 3.2.3 Language-Independence

One of LOLITA’s tasks is translation. This means that the representation used must be independent from any particular language. This applies not only to the
choice of concepts in the K.B., but particularly to the choice of primitives. Thus the representation for any phenomenon should not depend on the particular language it is formulated in. This is difficult in that each culture has its own particular way of conceptualising different phenomena. For instance, the linguistic use of tense and aspect conceptualises time differently from the way the Gregorian calendar does. Even such simple notions as hours may differ from culture to culture: the Roman day was divided into ten variable hours of sunlight. The representation should allow these diverse phenomena to be expressed in the K.B. naturally.

3.2.4 Compositionality

A representation is compositional if the meaning of any statement expressed in it is\(^1\) a systematic function of the meaning of its syntactically well-formed parts. This definition does not preclude byzantine meaning assigning functions which can cope with representations where the meaning of each part depends on what other parts are in the statement. However, such functions reflect a worse partitioning, or modularity than simple functions combining parts each expressing independent pieces of information. Thus they suffer from all the problems already associated with poor partitions, such as low flexibility. It is therefore worthwhile to ensure that the meaning assigning function combines parts of statements which express consistently simple independent pieces of information.

3.3 Requirements for a Large System

3.3.1 Development issues

Two representation issues are involved in the development of systems. The first, correctness, is important to systems performing valid reasoning. It is classed here as a development issue since proofs of correctness provide some guarantee that the

\(^1\)Strictly speaking, it is compositional if it does not preclude the interpretation function from being compositional.
system will work as expected. The second, ease of use, determines how easy the representation is to understand for its developers.

3.3.1.1 Correctness

If the representation is to be used for reasoning, and one wishes to be sure that the results one expects will be derived, it is necessary to prove some properties of the representation. These include syntactic properties such as a well-defined language (with known morphology and grammar), some well formed formulas chosen as axioms, explicitly defined rules of inference, and consistency. Consistency is the property that there is no statement such that both the statement and its negation are provable. Another such property is decidability. This is the property that there exists a terminating algorithm that will determine for any statement whether or not it is provable. It has been shown that any representation at least as strong as First Order Logic cannot be decidable [Davis 89].

Notice that many of these properties assume that rules of inference have already been defined. Once these have been defined, there is little distance to go before obtaining a reasoning algorithm. Thus, these rules do not appear to be of general interest for all processing in the LOLITA system, but of particular interest to those people implementing reasoning tools: it will be advantageous to them to see whether the inference rules they decide upon will not produce the desired results before they start implementing their solution. Hence this particular aspect of correctness is outside the scope of this PhD.

Other requirements falling under the notion of correctness are soundness and completeness. These deal with the difference between the representation, and what it is expressing, between the syntax and the semantics or interpretation. Soundness is the property that every statement that can be proved syntactically is true in the semantics. That is to say that one never can prove a statement, if what it means is clearly false. Completeness is the property that every statement considered true in the semantics is provable in the syntax. Note that this requires the meaning to be formalised itself in another syntactic form. Since there is no such formalisation of
all the things expressible in natural language, these properties are of little help in the design of a general representation. They may be of more use to some particular task, such as deriving a formal notion of inheritance against which inheritance algorithms can be tested. But again, this means that these issues are outside the scope of this PhD, and of more relevance to those studying particular reasoning methods.

It might turn out through either of these forms of analysis that the representation cannot express what it seemed to intuitively. This might lead to some discomfort that this analysis is not performed before the implementation of the tool starts. However, this is an issue of costs versus benefits. These forms of analysis are expensive to conduct. The question to ask is therefore which of correcting the system in case of error, or performing the full analysis will be the cheapest. Another issue affecting this decision is how strong the requirement is for the reasoning methods to be completely correct.

It should be remembered that LOLITA is designed to support non-valid forms of reasoning, such as analogy. Therefore if a valid reasoning algorithm is known to make mistakes from time to time, it can be treated within the same framework as a non valid reasoning algorithm with the appropriate level of certainty. Moreover, some tasks use reasoning rarely and when they do, do not depend on high certainty. These include information extraction tasks such as template extraction. Thus useful tools could be produced even without the availability of valid reasoning. They could then be sold in a commercial environment, or in an academic environment could demonstrate the group's grant worthiness, in both cases helping to sustain the research effort.

The alternative is to analyse the representation deeply. This runs the risk of developing very thoroughly a representation which later will turn out incapable of representing expressions of natural language which had not been thought of at the design stage. Dealing with such expressions might require redesigning the relevant representation. Moreover if cohesion is important, this may affect to a large extent the rest of the representation. This would mean reanalysing all the representation,
which would prove expensive and make the system inflexible to change. Instead if
analysis is postponed until really useful, the initial stages of the representation can
follow the prototyping development process outlined earlier. Once the representa-
tion seem stable enough to warrant using it as the basis of reasoning tools, effort
can be expended in analysing it.

3.3.1.2 Ease of use

Ease of use corresponds to how easy it is for developers to understand the repre-
sentation intuitively: although the representation need not be understandable in
the sense that is hidden away from the end user, it helps developers if it is. This
in turn has the advantage of reducing the likelihood that they get confused, and
that bugs creep into the end product. This requirement is obviously of lesser im-
portance than those directly having an effect on the performance of the tool, such
as efficiency or richness, but still should be borne in mind.

3.3.2 Representation for a large project

The representation for a large project such as LOLITA, must be designed as such:
uniqueness and cohesion are properties which determine how difficult it is to scale
the system up to real world large scale applications. Bad decisions at this stage are
expensive later, when the amount of code to write and test increases unnecessarily
quickly.

3.3.2.1 Uniqueness

If a representation is unique, there is only one way in which any fact can be ex-
pressed with it. This means that there is only one way to recognise a particular
type of statement. If there is more than one way to express a fact, there must
be different ways of recognising it when searching through the K.B. This leads to
an increase in the number of possibilities to consider, that is to say it increases
the potential search-space. Since searching for information in the K.B. is a very
frequent activity for a large scale system, it must be optimised. The requirement for uniqueness is one such optimisation. It also benefits the clarity of the implementation: not adopting any uniqueness measures what-so-ever results in a large proportion of the implementation checking for alternative expressions of the same fact. Moreover it can lead to multiple reasoning tools to solve what is essentially the same problem.

The real uniqueness just described is preferable to the weaker “normalised” forms. In this case, although there may be many ways in which a particular piece of information can be represented, only one of these is considered legal. Normalisation is computationally costly, since for a system to be robust, every statement entering the K.B. should be checked, and if necessary converted to its legal form. This process can be quite costly. However without it, the risk is run that illegal statements are entered into the K.B. and “lost” there, if they are not recognised by any of the processing code, or worse, misidentified and lead to processing errors.

3.3.2.2 Cohesion

Cohesion is related to uniqueness. The idea is to use as much as possible the full expressiveness of the existing levels of a representation in order to reduce as much as possible the extensions needed to express new phenomena. This results in a layered representation where each level corresponds to a particular type of reasoning which applies to a wide range of phenomena. Thus many phenomena which have the same abstract behaviour can be grouped together and represented in a uniform manner. This allows the same reasoning tools to be used to reason about all of them.

This design process may show insufficiencies of the basic levels which are then often found to have effects on the representation of many other phenomena. Thus by improving and re-using the basic levels, a multitude of ad-hoc extensions are avoided, and the resulting representation is more thoroughly tested.
3.4 Requirements for a large KB: Efficiency

In a K.B. where concepts (including relations and defining properties) are expressed as symbols, statements correspond to some combination of these symbols. The precise way in which these symbols are combined is determined by the design of the representation. Although this factor does not affect what can be expressed by the representation, it substantially limits the efficiency with which the representation can be processed.

Very large K.B.s do not fit in computer RAM. They must therefore be swapped from disk to memory, which is an extremely expensive operation. Depending on the way in which the knowledge base is structured, the following properties can dramatically influence the amount of swapping performed. For instance, graphs typically have as unit a vertex. The K.B. would therefore consist of a set of files, each consisting of a set of vertices annotated with the edges that connect to them. The more edges an algorithm traverses, the more likely different files of the K.B. must be accessed.

3.4.1 Topological Properties

3.4.1.1 Topological distance

Topological distance refers to the number of steps that must be made through the K.B. in order to reach a relevant piece of information. For a K.B. which divides information up into statements this would correspond to the cost of locating the particular statement in terms of steps through the relevant data structures, and then the number of steps through the statement required to determine the desired piece of information. For example, a K.B. using some form of hashing would include the lists of pointers dealing with the cases where two pieces of data are assigned the same index. The steps through these would be included in the calculation of the topological distance. For a graph, it would be the number of edges to be traversed which would be counted. Since topological distance plays a role in determining the efficiency of retrieval, it is important to reduce it.
3.4.1.2 Determinism of Search

The determinism of the search is determined by the number of paths through the K.B. that would be investigated and would potentially lead to the information being searched for. Thus it corresponds to search space. In a graph this could potentially be equal to the number of paths through the graph starting at a particular vertex, assuming a depth-first search. However it would be unreasonable to make such an assumption, if the topological distance in the K.B. had been minimised. Therefore, a breadth-first search is assumed. Under these conditions, only paths of the same or lesser length than the minimal path to the relevant information need be considered. Maximising the determinism of search reduces this search space, and therefore increases the efficiency of retrieval.

3.4.2 Assumed properties of the interpretation

A representation is interpreted, assigned a meaning, by a mapping or “interpretation”: what was called a “meaning assigning function” in 3.2.4 (p. 31). This “meaning” is the representation’s semantic model. A syntactic representation, if it is to be useful, assumes some semantic model and interpretation into that model, whether or not this is defined formally.

Since the interpretation is that which assigns meaning to statements in the representation, it limits the efficiency of access to the encoded information. In this section, properties of the interpretation, which affect efficiency are discussed. To enjoy the resulting efficiency, the representation must not prevent the interpretation having these properties.

Both properties are formulated in terms of atomic symbols. These are the symbols that the interpretation combines. They are thus indivisible symbols for the interpretation, such as “man” in “man(x)” in FOPL. Clearly, each representation will define the atomic symbols it uses.
3.4.2.1 Non-linearity

If a sequence of symbols forming a statement must be read in a predefined order, for meaning to be assigned to it, then the statement is linear. A representation is non-linear if it does not require statements to be read in any pre-defined order.

If the representation is linear, each time a piece of information is to be extracted from a statement, the whole statement must be read from its start. This is inefficient when a lot of information must be accessed randomly. On the other hand, if the representation is non-linear, information may be searched for from any symbol of the statement.

Formally, a representation is said to be non-linear if reading it from any atomic symbol and in any direction gives information which is sound with respect to its semantic model.

3.4.2.2 Distributedness

• Definition

The distributedness of a representation determines how much of any statement expressed in it must be read to be assigned an interpretation. Since statements in the K.B. may relate many concepts in a complex way, it is advantageous not to have to read them fully if only part of their meaning is relevant. For instance the statement “John brought his son to school yesterday” involves many relations:

• John is someone’s father.

• the occurrence of John taking this person to school

• that occurrence happened yesterday

• ...

If this statement were in a K.B., it would have to be read fully before the fact that something involving John occurred yesterday could be ascertained. This fact
may be needed in some reasoning. Requiring the full statement to be read is an unnecessary expense which can be avoided.

What condition is required so that only the interesting parts of any statement need be read? Examine the case in which it fails. If the interpretation combines any set of atomic symbols of the representation in such a way that the interpretation of any subset of the symbols is not expressed by the interpretation of the full set of symbols, all symbols of the full set must be read if the correct meaning is to be assigned to the statement. Thus, in a fully distributed representation, the interpretation would never combine any set of atomic symbols into a meaning which is not expressed by the interpretation of any subset of the symbols.

Formally, the degree to which a representation is distributed depends on the proportion of possible sections of the representation that give information which is sound with respect to the K.B.'s semantic model. Sections are groups of atomic symbols, and never divide atomic symbols. Non-distributed sections are those that yield information unsound with respect to the K.B.'s semantic model, when divided.

The granularity of distributedness is an informal notion of the size of non-distributed sections in a representation. It is informal, since it is possible that most of a representation's non-distributed sections are atomic symbols, but a few exceptions involve dozens of atomic symbols each.

- **Distributedness in practice**

Distributedness measures to what extent independent pieces of information are encoded independently in the representation. This notion of independence means that information must be encoded from where it will be accessed. For instance, to simplify the NL interpreter, one might decide that a representation for NL should represent tense and the order in which propositions were stated. Since tense and order implicitly encode time of occurrence, annotating propositions with their time would break uniqueness. To avoid this, one might decide time should always be determined by reasoning. In effect, this would add an interpretation function that gives a meaning to a subset of symbols which is not expressed by the interpretation of the full set of symbols: it would break distributedness. The problem is that time
is accessed from the proposition, and is a concept local to the proposition, but it
has been encoded as a concept global to the full text from which the statements
were derived.

In a fully distributed representation, any segment of the K.B. is sound with respect
to the full K.B.'s semantic model. In particular, for graphs, this means that any
piece of the graph can be cut away, and it will be sound with respect to the full
graph. Trying this out in practice provides an easy method to test a graph's
distributedness.

Although distributedness improves efficiency by avoiding having to read irrelevant
information, it must be applied carefully. For instance, over-zealous use may in-
crease topological distance without increasing the efficiency due to distributedness.
The clearest case of this is where a distinction is made between two concepts, so
that distributedness allows them to be read independently, when in fact they would
always be used together.

3.4.2.3 Distributedness and Non-linearity

Distributedness and non-linearity are complimentary properties in a distributed
K.B.: a representation is non-linear if reading it from any atomic symbol and in
any direction gives information which is sound with respect to its semantic model.

In a distributed K.B., it is convenient to take the sections of the K.B. which are not
distributed as the atomic symbols considered by Non-linearity. Since these non-
distributed sections of the K.B. must be read entirely to be assigned a meaning, it
matters little to efficiency in which order elements within these sections are read.
The point about linearity was that it forced additional irrelevant information to
be read: a non-linear representation within the non-distributed section would not
prevent anything from being read, so would not increase efficiency. Thus, in a
distributed K.B., the atomic symbols considered by non-linearity are the sections
of the K.B. below the granularity of distributedness.

Note that in non-distributed K.B.s, the representation determines the atomic sym-
bols chosen for Non-linearity. Furthermore, a separate meaning assignment function will be given to the non-distributed sections in the K.B. Internally each may be non-linear with respect to the full section.

3.4.2.4 Distributedness, Non-linearity, Compositionality and Cohesion

Distributedness and non-linearity assume some degree of compositionality. Indeed, their requirements could be trivially satisfied if each atomic symbol were assigned no meaning: no information is sound with respect to the full interpretation of the K.B. However, such a solution would be counterproductive since each atomic symbol would give no clue to direct the search through the K.B.: obtaining information out of the interpretation function would become an essentially random event. At the other end of the spectrum, each atomic symbol could encapsulate all its meaning, and the only composition function would be addition of information. In this case, no additional meaning would be given to groups of atomic symbols. Between these two extremes lies an area where each atomic symbol read delivers some meaning, but certain combinations of atomic symbols have a meaning which is not only given by combining the information present only in the atomic symbols.

Clearly, distributedness and non-linearity are more worthwhile concepts towards the latter extreme. But this extreme contradicts cohesion, where representations were to build on each other, reducing the number of necessary reasoning engines: models incorporating the behaviour of a complex phenomenon may involve the interaction of sub-phenomena, which each involve their own specific reasoning. Thus complex representations may need to build on more than one existing representation, and may require more than one atomic symbol. A balance between the two extremes is required. Moreover, efficiency is maintained if the interpretation function which gives additional meaning to the groups of atomic symbols can return some of the additional meaning early (before having read the whole group). This is equivalent to intermediary (or partial) interpretation functions which can interpret smaller groups of atomic symbols than occur in each complete representational symbol of the complex representation. The full interpretation of a complex
Chapter 3: Representation Requirements

representational symbol would combine the results of the partial interpretation functions. To maintain non-linearity, the partial interpretation functions should give information as soon as it is available whatever the subset of atomic symbols corresponding to the complex representational symbol. This leads to a sort of hierarchy of interpretation functions. If such a hierarchy can be built, the scope for byzantine rules in the interpretation functions is minimised, so the representation is likely to be more compositional, as defined by 3.2.4 (p. 31).

3.5 Ranking the different requirements

It may not be possible to satisfy all of the requirements outlined above simultaneously. To obtain the best overall compromise, the requirements must be ranked by importance.

Of paramount importance are issues pertaining to whether or not the K.B. can express the information needed. Unless this condition is satisfied, the tool will not satisfy its requirements. These requirements include naturalness, richness and language independence.

The issues determining whether or not the tool will scale up to a large scale domain, without requiring an unnecessarily high amount of resources poured into it are of great importance whether in an academic environment, or in a commercial one. As previously mentioned, the cost of maintaining large programs can absorb 2 to 4 times the resources required for development. [Sommerville 92]. In a prototyping model of development, it is also important to encourage wide re-use. For instance reasoning modules should be re-used. In this way effort is put into the development and testing of a few reasoning methods. Otherwise one risks channelling effort into many similar reasoning methods for different representations of similar phenomena. Therefore uniqueness and cohesion rank second.

Efficiency ranks third, since an unbearably slow system will be unusable in practice. This includes reducing topological distance, increasing determinism of search and distributedness, and requiring non-linearity.
Finally come the requirements of use to developers. Ease of use is useful to all developers storing and retrieving information in the K.B. Thus people developing the analysis phase, natural language generation, or particular tools such as template analysis benefit from improvements in the ease of the representation's use. However this aspect is less critical since the K.B.'s interface will hide obscure features they need not know about if they do not need them: most developers only need to deal with the information they need rather than obscure features of the representation. The second requirement is correctness which already has been considered to lie outside the scope of this work. Thus these issues will be the least considered in the choice or design of the representation.

It is of interest to note that the first three sets of requirements parallel the requirements of NLE for a working system that is integrated, maintainable and feasible in hardware terms.

### 3.6 Conclusion

Now that the set of requirements for LOLITA's K.B. have been established, the problem is to find a K.B. that satisfies them. This can be achieved by examining known representations. Since LOLITA is to process natural language, representations designed for natural language are of particular interest: they should include the features needed for NL.
Chapter 4

Literature Review

In the quest for a representation satisfying LOLITA's requirements, existing representation schemes are reviewed. Attention is concentrated on representations satisfying some of LOLITA's requirements, and on empirical evidence supporting the understanding of knowledge assumed in 2.2 (p. 11).

Since KR started in the 1950s, a vast amount of work has been done in this area. Thus this survey cannot hope to be comprehensive. Instead, only well-known families of knowledge representation which have some of the features deemed necessary for LOLITA will be considered. This explains the lack of reference to other known representations such as NETL [Fahlman 79]; Preference Semantics [Wilks et al. 92]; Econet, Episodic Logic and derivatives [Schubert 79], [Hwang et al. 93]; Frame language [Hirst 87]. These are either rare, not devoted to natural language, do not fit any of the required briefs, or do not add substantially to the knowledge provided by the analysed cases.

4.1 FOPL

Many systems are based on first order predicate logic. A naive implementation would represent this either as linear statements (as written mathematically) or as a syntax tree (making clear what is applied to what). In both cases the representation is packed, linear and non-distributed. Indeed, \textit{FOPL} expressions are not non-
linear since they cannot be assigned meaning when read backwards. Similarly, they are not distributed, since an arbitrary slice of an $\mathcal{FOPL}$ statement is not sound with respect to its full reading: $a$ is not sound with respect to $a \Rightarrow b$.

$\mathcal{FOPL}$ is packed. This is to say, if one wishes to read a fact within a statement, the whole statement must be searched for it, and the relevant fact “extracted” taking into account the context around it:

$$\forall x \forall y (a(x, y) \Rightarrow b(x)) \neq \forall x (\forall y a(x, y) \Rightarrow b(x))$$

This also causes difficulties in the representation of NL sentences. Indeed, different NL sentences may refer to what is expressed in $\mathcal{FOPL}$ by the same variables. This requires them to be amalgamated into one very long $\mathcal{FOPL}$ statement. This packing is very different from the ideal representation for LOLITA where all the facts about things are in their proximity and accessed without regard for irrelevant information.

A second feature of $\mathcal{FOPL}$ is that it makes it difficult to reference concepts which are built of others. For instance, “One of the people I work with, is John. Johns’ tall friends are loathsome people. They enjoy torture, betray each other regularly, and cause no end of trouble. I really don’t understand why he likes them.” Here reference has been made to the set of people I work with, John, and John’s friends. These concepts are not represented in $\mathcal{FOPL}$ by a single symbol such as a set, but by a conjunction of properties required of a variable for it to refer to them: $\forall x ((\text{friend.of}(\text{John}, x) \land \text{tall}(x) \Rightarrow \text{people}(x) \land \text{loathsome}(x)) \land \cdots$. Thus sets are created on the fly implicitly by a set of conditions, rather than explicitly which would allow them to be referred to. This is a major stumbling block for a NL system since reference of this sort is common in NL.

Of course, it is possible to express the information of $\mathcal{FOPL}$ less naively, by using a more computationally efficient internal representation. But since this thesis is concerned with the internal representation used, such a change is no less than a change of representation. The problem with $\mathcal{FOPL}$ is not that it cannot express the sort of information that one would like to express, but that it does not provide a means of organising it to allow efficient access to the knowledge.
\( \mathcal{FOPL} \) does have advantages for formal analysis. However this is not one of the requirements of this thesis, since it assumes that the representation can be modelled in terms of relevant mathematics to enable its properties to be analysed. This alleviates the requirement for a formal semantics to be presented for it\(^1\).

### 4.2 CLE and CLARE

This section discusses the representations used by the Core Language Engine CLE and derived systems. Two generations of CLE will be considered in this section: CLE-3 and CLE-6 (also known as CLARE) are large-scale natural language processing systems, mainly developed by SRI International Cambridge (UK).

#### 4.2.1 CLE’s representations

CLE uses a series of representations. Their names, and the claims made for them follow:

- **LF** (Logical Form): LFs are completely disambiguated language: Alternative readings of N.L. expressions give rise to different logical forms. They express the literal meanings of N.L. expressions: they specify the truth conditions of particular readings of these N.L. expressions. They are rich enough to represent knowledge expressed in N.L., and are suitable for reasoning.

- **TRL** (Target reasoning language): A language similar to LF in what it expresses but which is used for reasoning and which is converted into database languages such as SQL for database access.

- **QLF** (Quasi Logical Form): QLF is an extension of LF allowing unresolved quantifications, unresolved references and implicit relations to be expressed.

---

\( ^1 \)Other researchers also make this assumption: Shapiro talks of translation of SNePS into \( \mathcal{FOPL} \) as a “model of the network, rather than another syntax for the same network.” Indeed \( \mathcal{FOPL} \) does not incorporate certain properties of the SNePS network, such as the uniqueness of concepts: if \( n_1 \) and \( n_2 \) are constants (concepts), then in \( \mathcal{FOPL} \) one can write \( \forall(P, z)[P(n_1, z) \iff P(n_2, z)] \) which is illegal in SNePS.
- **RQLF** (Resolved Quasi Logical Form): A RQLF is a QLF with additional information stating what are the proposed resolutions of the unresolved or implicit parts of the QLF.

LF is a higher order logic, consisting of first-order logic and extensions motivated by the requirement for richness. These include generalised quantifiers; tense, aspect and modality; lambda abstraction; types of logical form (statements, questions, commands)... Both QLF and RQLF are intermediary representations used during the interpretation phase.

CLE-3 uses LF, QLF and RQLF as different representations through the analysis phase. LF is the final form of the knowledge which can be stored in a database, and reasoned with.

CLE-6 only uses QLFs through the analysis phase. However QLFs are translated into an unambiguous fully resolved form, TRL, once the analysis has been performed. TRL appears to be similar to LF. This is used as an interface to a database which stores some of CLE's knowledge. TRL is used for reasoning, which appears to be limited to abduction.

All forms of the representation are written as lines of symbols. This results in particularly complex statements, which can include many events embedded one into another.

### 4.2.1.1 Particularities of QLF

QLF was designed as an internal representation for natural language analysis, and includes much information specifically relevant to this task: the representation may simultaneously include the original sentence, lexical, syntactic, and various levels of semantic information. All these pieces of information are derived at different phases through the analysis, and are particularly useful in that any analysis rule has access to any information that may be relevant to it. This makes the system very flexible to change, since rules may be adjusted to use information, which in other systems, may already have been discarded.
Another particularly important feature of QLF is that it is monotonic. This is more the case for QLF-6 than QLF-3. The idea behind this is that the form of each statement stays much the same throughout the analysis, but the detail gets filled in as the analysis proceeds. In this manner, rules can match the general form of the statement, and determine whether they should be applied or not by matching against specific information at some point of the representation. If the required information appears there, the rule can be applied. This differs from most representations where any subset of the set of symbols forming the representation of the expression can arbitrarily be swapped around, deleted or added to: even if the required information is expressed in the statement, it must be searched for in many places. In this respect monotonicity resembles uniqueness. However, monotonicity differs from uniqueness, in not allowing information to be deleted.

Monotonicity is implemented by using meta-variables which can be constrained, and then instantiated, as only means of modifying QLFs. In terms of processing this monotonicity allows rules to be applied in an order-independent manner. This significantly simplifies building the analyser, in that the order of analysis does not need to be specified by the developer. It also leads to more flexibility, where rules can easily be improved by using information which might only be available at a later phase of the analysis. The lack of order means that the right new place need not be determined by the programmer. Because constraints are used, only information that is known at a particular time is added to the QLF, so no tentative conclusions are made, later to be retracted.

Although QLFs contain information from the various phases of analysis, the representation overall expresses independent information separately: one could argue that the information from the syntactic phase and the semantic phase “means” the same thing, and therefore breaks this criteria. But it should be remembered that each has a different level of associated ambiguity, and each such level is associated with different forms of processing, which reduces the dependence between these parts of the representation. The only processing which accesses the different types of information is precisely that which creates the dependencies between them: the analysis.
4.2.1.2 LF syntax

LF will be discussed first since it is easier to understand. QLF is an extension of LF. LF extends the language of first order predicate logic in three main ways. First, use is made of lambda abstraction for the formation of higher order expressions. Second, the language is extended with generalised quantifiers. Finally, tense operators, intentional operators, and other higher order operators are included.

\[ \text{dcl, quant(exists, A,} \]
\[ \text{[college_place, A],} \]
\[ \text{[past,} \]
\[ \text{quant(exists, B, [event, B],} \]
\[ \text{[and, [design1, B, joh1, A],} \]
\[ \text{[in_location, B, cambridge1]]])]] \]

This first example is a declaration. It corresponds to the sentence that "John designed a college in Cambridge", where Cambridge is the location of the designing. The dcl stands for declaration, and is one of the 4 types of statement that can be expressed. The others are whq (wh-question), imp (imperative), and ynq (yes-no question). Declarations are facts.

In this example the concepts are fully disambiguated: college_place, design1, joh1 and cambridge1 stand for particular concepts which are defined in sense entries. These may include the conditions in which they apply, for instance semantic selection criteria:

\[ \text{sense(design,} \]
\[ \text{v:[arglist=([(B,np:[semGaps=Gaps])],} \]
\[ \text{vform=inf, eventvar=E, subjval=A, semGaps=Gaps],} \]
\[ \text{[design1, qterm(<t=quant,n=sing, l=ex>,E,[event,E]),A,B])} \]

This specifies that when the subject is eventually instantiated, it will be referred to by the variable A. The variable E must be of type event because design1 is only involved in events. The arglist specifies that design takes another argument of syntactic category np (noun-phrase) as object, thus specifying that design1
is transitive. This example is quite simple, but the information given allows the
behaviour of different meanings in complex situations to be expressed.

It should be noticed that the notions of location (and time) are quite simple in that
there appears to be no model of location where locations themselves are discussed.
This would seem to be a limitation of the representation which could not allow
locations such as "half way between the lamp-post and the post-box" to be expressed.

• Embedded events

Statements may include more than one event. An example of such embedded events
is "John reads every interesting book that Mary buys", represented by:

\[
\text{quant}(\text{forall}, B, \\
[\text{and}, [\text{and}, [\text{book1}, B], [\text{interesting1}, B]], \\
[\text{pres}, \text{quant}(\text{exists}, E, [\text{event}, E], \\
[\text{buy}, E, \text{mary1}, B)]]], \\
[\text{pres}, \text{quant}(\text{exists}, D, [\text{event}, D], [\text{read}, D, \text{john1}, B]])]
\]

• General quantifiers, values and measures

LF uses a generalised notion of quantifiers, which unifies the ideas of cardinality
and quantification. All forms of quantification are understood to be intersection of
the set being quantified upon (A) and some restrictive set (B). The choice of B
depends on the quantifier. Thus if the quantifier is \(\forall\), B equals A, and \(A \cap B = A\).
However the choice of B is expressed in terms of the cardinality of A \(n = |A|\)
and \(A \cap B (m = |A \cap B|)\). This allows complex quantifiers such as "at least
three" to be expressed, by stating that \(m \geq 3\) and so on. Some examples:

\(\text{every: } N^*M^*[\text{eq}, N, M] \) (also written as \(\text{forall}\))
\(\text{not every: } N^*M^*[\text{not, eq}, N, M] \) (also written as \(\text{exists}\))
\(\text{most: } N^*M^*[\text{ratio}, M, N, 1, 2] \) (i.e. \(n \geq m/2\))
\(\text{at least four and at most ten: } N^*M^*[\text{and, geq, M, 4}, [\text{leq, M, 10}]]\)

These quantifiers are expressed as the first argument of a quant predicate.
"N\^\*" is a lambda expression. In this case it is the second argument of the eq predicate.

Similarly, QLF allows measures to be expressed using the predicate amount. This allows any quantity to be expressed in the appropriate unit. For instance, "John bought at least five pounds of apples" is expressed as:

```plaintext
quant(amount(R^-I^-[geq, I, 5], pound), A,
    [apple, A],
    [past, quant(exists, E, [event, E], [buy, E, john1, A]))])
```

In the same way the unit cardinality is used to express the cardinality of a set.

Adjectives provide another form of measurement: those adjectives that have comparative and superlative forms are considered gradable. Thus notions such as "John is tall" would be represented as [tall1_degree, john1, 176]. Comparatives are expressed using one of two four place higher-order predicates more and less as in [more, P, Item1, Item2, Degree]. This latter expression signifies that Item1 has a degree with respect to P greater by Degree than that of Item2 with respect to P. Thus "Mary is 2 inches taller than John" is expressed as:

```plaintext
[pres, quant(exists, A, [state, A],
    [be, A,
     quant(amount(B^-C^-[eq, C, 2], inch), D,
       [degree, D],
       [more, E^-F^-[tall1_degree, E, P],
        mary1, john1, D))])]
```

"Mary is taller than John" is expressed similarly except that [eq, C, 2] becomes [eq, C, 0] and the unit inch is replaced by unit. However, sentences such as "John is as tall as Bill" is expressed as:

```plaintext
[pres, quant(exists, A, [state, A],
    [be, A,
     [more, B^-C^-[tall1_degree, B, C], mary1, bill1, 0]])]
```
"Mary is taller than John" is expressed similarly except that Superlatives are expressed using the \([\text{order}, \text{Item}, \text{Type}, \text{P}, \text{N}]\), which signifies that \text{Item} is the \text{Nth} concept of type \text{Type} with respect to type \text{P}. Thus the statement "The third oldest building" would be represented as:

\[ X^[\text{order}, X, Y[\text{building}, Y], X^L[\text{old_degree}], 3] \]

Note that for values, LF appears neither to have a high degree of uniqueness (use of \text{leq} and \text{geq}, of \text{more} and \text{less}, use of quantifiers or \text{amount}(R^I[\text{geq}, I, X], \text{cardinality}) to express cardinality), nor of cohesion (use of \text{more} and of \text{geq} when the former could be expressed in terms of the latter).

### 4.2.1.3 QLF syntax

The QLF notation includes lexical information (1), and linguistic information such as n(number) and p(syntactic type).

\[
\begin{align*}
&[\text{and},] \\
&[\text{jump1},] \\
&q\text{term}(<t=\text{quant}, n=\text{sing}, l=\text{ex}>, E1_1, [\text{event}, E1]), \\
&q\text{term}(<t=\text{quant}, p=\text{det}, n=\text{sing}, l=\text{every}>, X_2, [\text{child}, X])], \\
\end{align*}
\]

\[
\begin{align*}
&[\text{shout1},] \\
&q\text{term}(<t=\text{quant}, n=\text{sing}, l=\text{ex}>, E2_3, [\text{event}, E2]), \\
&a\text{\_index}(X)]
\end{align*}
\]

- **An example of monotonic resolution**

This example is given in QLF-6 since this is the more monotonic variant of QLF. Notice that now every term is given an index of the form "+x", where \(x\) is some letter. This replaces the previous use of \text{a\_index} in QLF-3.

\[
\begin{align*}
&s: \text{meet}(\text{term}(+b, <\text{type}=q, \text{lex}=\text{every}>, \text{boy}, _{-q}, _{-x}), \\
&\text{term}(+g, <\text{type}=q, \text{lex}=q>, Y^\text{and}(\text{girl}(Y), \text{tall}(Y)), r, -y))
\end{align*}
\]
Y is a restriction on the term *g. Similarly the restriction boy stands for *X*boy(X).

The analysis phase instantiates the meta variables, written as "_x", where x is any alphabetic character:

_s becomes [+b,+g], signifying that +b outscopes +g.

_q becomes forall, and _r becomes exists.

_x becomes +b to indicate that the term +b is defined only in terms of its own restrictions. Similarly, for _y to +g.

\[[+b,+g]: \text{meet}(\text{term}(+b,<\text{type}=q, \text{lex}=\text{every}>, \text{boy}, \text{forall}, +b),
\text{term}(+g,<\text{type}=q, \text{lex}=q>,
\ Y^*\text{and}(\text{girl}(Y), \text{tall}(Y)), \text{exists}, +g))\]

### 4.2.2 Separated knowledge in CLE

CLE uses purely linguistic information to analyse natural language text. For this reason, the domain model consisting of the world knowledge, and the linguistic information consisting of the rules, the lexicon, and the context are separated. The context contains QLFs corresponding to the previous sentences which were analysed, for purposes including anaphora resolution.

QLF is not used as the representation scheme for CLE-6's world knowledge base. Instead a normal database, in this case using an SQL language, is used. This is accessed by converting QLFs to and from TRLs, which in turn can be transformed to and from SQL. For the CLARE project, the database contained about 500 axioms, for both examples of application: a project management database, and the ATIS domain\(^2\).

### 4.2.3 Evaluation

CLE-3 suffers from 3 levels of representation, which was argued against in the definition of the requirements for the representation. However, in CLE-6 only QLFs

\(^2\)Richard Crouch, personal communication
are used: this development is encouraging in that it provides empirical evidence supporting the argument for cohesion. Indeed most of the changes of representation between CLE-3 and CLE-6 were motivated by pragmatic reasons, such as convenience.

All the forms of the representation are packed: statements expressing events can include other events, as in “John reads every interesting book that Mary buys”. This packed representation is linear and has a low distributedness since to access some events, one must read others. As such the representation would be of little use for searching for facts. For instance “John designed a college in Cambridge” can be understood to state that one of the colleges in Cambridge was designed by John. If a system using QLFs as the representation for its knowledge base were asked whether there are any colleges in Cambridge, it would potentially have to read through all the statements of its knowledge base to find the relevant fact.

The cohesion appears weak in the manipulation of values, where there appears to be different ways of representing the size of a set through general quantifiers, measures, comparatives and superlatives, and finally ordering. Similarly it would seem that time and tense in CLE-6 is transformed into time points or intervals. These can be understood as extensions of the notion of values, and so should also be treated in a cohesive manner. Thus, CLE appears to suffer at the level of cohesion.

Although the generalized quantification appears to unify many issues in a nice way, it introduces certain assumptions. For instance, it models “most” as “more than half” which is true but not necessarily the full meaning. This is due to the model which requires a precise value.

Lambda abstraction is used very often in LF. This is probably related to monotonicity and meta-variable instantiation in that it is easier to instantiate one argument of a predicate with a lambda abstraction, than it is to build the corresponding formula around the relevant arguments. However, it requires additional effort on the part of algorithms using LF to instantiate the lambda expressions with the relevant arguments of each predicate. This is inefficient, and somewhat awkward in that these are additional steps. It can also be seen as affecting uniqueness, although its
benefits during the analysis stage may outweigh these issues.

QLF lacks any structure to help in the search for all the information involving a particular concept, such as John. This lack includes any form of inheritance hierarchy. This is because it is assumed that QLFs will not be used for searches required for instance for reasoning. Instead, a database is to be used.

However QLF is a representation specifically geared towards the analysis phase of natural language processing, and it is in this domain that it excels. Its monotonicity allows for an extremely flexible analysis architecture, where further processing, whether evaluation functions or linguistic rules, may use all the information derived from previous analysis. QLF also supports ambiguity allowing unresolved quantifications, unresolved references and implicit relations to be expressed. Moreover these ambiguous forms may be constrained, thereby allowing disambiguation to proceed incrementally without requiring a multitude of explicit alternatives to be represented.

To conclude, QLF satisfies very well its task as a representation for analysis, but is not the sort of general formalism needed for LOLITA.

4.3 Conceptual Dependency

Conceptual Dependency is a theory of conceptual processing, and the data/mental structures used in this processing. Its purpose was to account for this processing without considerations of computational efficiency. It does not focus on a representation of knowledge which allows knowledge to be organised, accessed and built efficiently. Instead it concentrates on the concepts to represent: what partitions of meaning space (see 2.4 (p. 13)) should be associated with each structural component? In other words this asks by what basic terms concepts should be expressed. Or in its own words, it attempts to provide a “representation of the conceptual base that underlies all languages” [Schank 79]. For this reason, it does not address issues such as its distributedness or cohesion, and from the outset its representation can be deemed insufficient for LOLITA’s needs. Some of its ideas have proved
influential, so some space will be devoted to them.

4.3.1 The representation

Conceptual dependency theory was based on the assumptions that:

1. All implicit information in the sentence should be made explicit in its representation.

2. If two sentences have the same meaning, their representations should be identical, regardless of the words used.

The first assumption emphasises the notion of some form of reasoning, since inferences must be made during the interpretative process to make explicit the implicit information. For this to be done by general inference rules, the representation must be canonical. In other words, there must be rules which determine how one states a term in the representation, so that the inference rules can always find the features they need for reasoning in well defined locations of the representation of a statement.

The second assumption leads to the notion that the representation of similar words or sentences should be similar. In other words semantic similarity is translated into syntactic similarity in the semantic representation. The requirement of a canonical representation leads to a representation of concepts based on features derived from a componential analysis. This is similar to the work done in semantics [Leech 81], where for instance man and woman share most features (adult, human...) but differ in their gender feature. The difference lies in that this representation must be good for reasoning. CD's representation consists of:

- a set of between ten to twelve primitives which represent the type of action being performed.

- a set of states, representing the preconditions and results of actions
• a set of dependencies or conceptual relationships, which could exist between the primitives, states and objects involved.

Statements in this representation form conceptual dependency graphs.

4.3.2 Inference

The other important idea of conceptual dependency theory is the notion that knowledge should guide inference. Thus specific representations to facilitate inferencing were developed: scripts, plans, goals, MOPs and TOPs. "Conceptual dependencies captured many inferences that can be made about physical actions. (...) Scripts captured knowledge about the most likely inferences to make in stereotypical situations. Plan and goal representations provided inferential capabilities at this level of understanding. TOPs combined goal/plan knowledge with contextual features, to provide more specific predictions useful in planning or plan understanding" [Lytinen 92].

In CD, knowledge includes expectations for inference. Scripts, and the later more general MOPs include things that are expected of various situations. For instance, when entering a restaurant one expects to be served by a waitress, at a table, to get a menu, then eat food, and finally pay. This sequence of expected actions is encompassed by a script. Obviously not all possible situations can be scripted explicitly, but the idea is that situations similar to the current problem can be found in the knowledge base, and modified to suit it. This reduces the amount of search required to solve problems since inference does not solve its problems by working everything out from first principles, but by taking previous solutions and fitting them to the problem. This approach was adopted by various semantic parsers (Riesbeck's analyser [Schank 75], MOPTRANS, Direct Memory Access Parsing) which did not separate syntactic, semantic and pragmatic processing, but integrated these tasks. This allowed each to inform the whole interpretation task as to its expectations and constraints. Similarly various planners used a similar strategy with relevant expectations encoded in TOPs. (CHEF, PERSUADER, MEDIATOR and CABARET.)
4.3.3 Conclusion

4.3.3.1 Primitives

Using a fixed set of meaning primitives has a major disadvantage. Only being able to represent primitives results in arbitrarily complex expressions in order for simple concepts like “walking” to be expressed. Moreover the requirement that all that is implicitly understood must be expressed explicitly in the statement can result in difficulties for ambiguous statements. Finally the foundation is the assumption that all concepts can be represented in terms of a small set of fixed conceptual primitives, in contradistinction to reasoning primitives. This is highly disputable.

The problem is twofold. First, as explained in A.2 (p. A-7), the ability to reason at many levels of granularity has many advantages and exploits a many levelled structured organisation of knowledge. Expressing everything in terms of primitives flattens this. Secondly, setting a fixed level of primitives does not allow any structure to appear within the primitives themselves. For instance, various primitives express the notion of movement: MOVE is the movement of a body-part of an agent by that agent, whereas PTRANS is the transfer of location of an object. These two notions have a lot in common, namely the moving something. However this similarity cannot be expressed in the representation.

It is of empirical interest to note that the notion of primitive has virtually disappeared [Lytinen 92]. Most implementations and representations descended from CD, have a virtually unlimited number of vocabulary items.

4.3.3.2 Expectation-driven Inferencing

Expectation driven inferencing is an interesting idea. However it depends on a large knowledge base of relevant situations to be used. This dependency makes of it a good tool for reasoning by first approximation. It does not however replace the richness of a good understanding. For instance, if one knows how to bake a loaf of bread, one can change the recipe in order to make a cake, knowing only that cakes must be sweet, sugar is sweet, and ingredients determine the taste of the final
product. Although this might succeed, it could just as well fail, say because the sugar burnt, and will not be the basis for a deep mastery of cooking. In a sense it is a form of tinkering, without a deep understanding of the processes underlying the situations encoded in the knowledge base. Only a deep understanding of the behaviour of the ingredients will allow radically new recipes to be created.

Two forms of processing emerge from this discussion. The first is knowledge intensive inference which relies on being able to adapt knowledge of expectations to a problem. This is not very flexible, but is efficient since the expectations constrain the search space. It also does not rely on a deep understanding of the phenomena concerned. The second is processing intensive inference which builds on understanding causes and effects, and a deep understanding corresponding to a model of the problem. This is not constrained by expectation so can derive radically new solutions to problems. However it is less efficient for the same reason.

One might wonder which a large scale system such as LOLITA should use. In A.2 (p. A-7), an approach based on deep understanding at various levels was suggested. In general however, not all the information available to LOLITA will be based on some model of the problem. For instance, new information obtained from analysing texts about a new field may be very shallow at first. Thus, processing intensive algorithms based on a deep understanding may fail in these circumstances. Moreover, there is little advantage in inferring something from first principles if the same result could be obtained from existing knowledge about past situations. Thus the answer is that both should be used. Indeed, the choice is not limited to these two extremes, but to any combination in between, forming a broad spectrum of possibilities to choose from. Similarly, in an ideal system, effort would be devoted to make new models from knowledge about similar situations.

This additional requirement does not impose any changes to the representation, since a representation able to express specific facts and general models at any level of granularity can express scripts (simple models at a high level of granularity) and instances of situations (all the times I’ve been to the restaurant).
4.4 CGT

John F. Sowa published an influential book [Sowa 84] introducing his conceptual graph theory. This built on the work of philosopher and logician C.S. Peirce. Peirce developed in 1883 the linear notation for logic which is used today, with only a few changes of notation. He also developed his existential graphs in 1896, a graphical notation which he believed would form the “logic of the future” [Sowa 91a]. His graphs formed a complete system of first-order logic, with extensions to modal and higher-order logic, something the semantic networks of A.I. didn’t have until the 1970s.

The first semantic networks were implemented in machine translation systems in the early 1960s [Sowa 91a]. The first such system was Nude [Lehmann 92] used in 1956 as an interlingua for machine translation at the Cambridge Research Unit. [Quillian 66] also investigated semantic networks as a means of associative memory. But it was Sowa’s book that provided the first clear definition of a semantic network representation geared towards natural language. It succitated much interest among researchers, and has been adopted as the basis of many NLP systems.

4.4.1 Conceptual graphs

Conceptual graphs are a form of semantic network with nodes and arcs. The nodes correspond to concepts, or occurrences of concepts, whereas the arcs express relations between the concepts.

4.4.2 Arcs

In the most basic form of the theory, only one arc is used: LINK. This connects nodes expressing particular concepts and nodes representing the relations between them. Thus, the following expression signifies “A bird is flying”:

[BIRD] <- (LINK) <- AGENT <- (LINK) <- FLY

Note that the use of the LINK arc allows monadic relations. However, the relation can also be expressed on the arcs, thus:
[BIRD] <- (AGENT) <- FLY
This is the form generally used. The contraction can however be pursued:
[BIRD] <- (FLYING)
Notice that nodes are written between square brackets as in [NODE], whereas arcs are written in round brackets as in (ARC). The arrows give the direction of the arc.

4.4.3 Nodes

Nodes are complex objects containing two pieces of information: a type and a referent. The type identifies the general concept to which the node refers. For instance if the node refers to a particular cat, its type is CAT. Types are discussed further in section 4.4.4 (p. 62). The referent field contains various information of relevance to the node. In the basic notation, only three kinds of referent are supported:

- Existential: This is denoted by "⁺" and indicates the existence of some individual of the appropriate type. It corresponds to the existential quantifier "∃" in logic.

- Individual marker: The symbol "#" followed by an integer identifies a unique individual and corresponds to a constant in logic.

- Literal: a quoted string or a number identifies an individual by its form.

Names, quantifiers and sets are extended referents, and can be expanded into the basic notation. These correspond to pieces of graph with nodes and arcs, but which are written for convenience as the referent of an expression. For instance, the concept [PERSON: John] is expressed as the graph:

[PERSON: #42109] -> (NAME) -> [WORD: "John"]

Notice that John is assigned a unique identifier, so that if two Johns are known, each will be identified uniquely.

Sets are in general expressed by curly brackets. In this manner, it is possible to predicate that a set of a given type exists by {∗}. This expresses a general plural
referent. Similarly size and some of the elements of the set can be expressed: [PERSON:{Norma, Frank, *}] @ 4] expresses the phrase “Norma, Frank, and two others”. There are four ways in which sets can participate in any relationship:

- Collective sets are used when all its elements participate in some relationship together: “Pat and her husband own the estate”.
- Disjunctive sets are used when only one of the elements of the set participates in the relation, as in “The elephant Clyde lives in either Africa or Asia”.
- Distributive sets are used when each element of the set satisfies some relation, but they do so separately: “Betty and Jerry are laughing”.
- Respective sets are used when each element of an ordered sequence bears a particular relationship to a corresponding element of another sequence: “Dick, Jerry, and Jimmy are married to Pat, Betty, and Rosalynn respectively”.

[STUDENT: {*}$\Phi$2] <- (AGNT) <- [READ] -> (OBJ) -> [BOOK: {*}$\Phi$3]

[STUDENT: Dist{*}$\Phi$2] <- (AGNT) <- [READ] -> (OBJ) -> [BOOK: {*}$\Phi$3]

The first line does not distinguish whether each student read every book, or whether one read two of them, and one read one. The second line states that both students each read three books. This is the distributive reading.

Quantification includes the generic plural {*}, existential quantification * and universal quantification expressed as $\forall$ in logic, and expressed in CGT using the relation $\neg \exists x (\neg P(x)) \equiv \forall x P(x)$. Concepts which are not counted, but measured, such as mass nouns, can also be expressed. For instance, the expression MONEY:@$\$$10 of type MONEY and of referent @$\$$10 denotes 10 dollars. It is written in a graph as:

[MONEY] -> (MEAS) -> [MEASURE] -> (NAME) -> [WORD: "$10"]

4.4.4 Types

Concepts in CGT are defined by a type hierarchy, which forms a lattice. These are general concepts, not referring to particular occurrences of concepts, like “Tabby, the
cat”, but to all its (potential) occurrences, as in cats in general. There is a set $T$ of such basic types. Each occurrence of a concept is expressed as a separate node, but all the occurrences of the same concept share the same type.

Thus types are separated from the rest of the semantic net. Sowa argues that this is necessary since the type hierarchy is a higher-order relation not between individual concepts, but between types of individuals. If “is-a” relations were to be used to define the conceptual hierarchy within the semantic net, these would be relations over sets. Sowa argues that types are similar to sets, but statements about types are analytic, and must be true by intension. Statements about sets are synthetic and are verified by observing the extensions. Thus if the intersection of the set of cats with the set of dogs is empty, it means that at the moment no individual happens to be both a dog and a cat. But if the intersection of the types cat and dog is ⊥, this means that it is logically impossible for an entity to be both a dog and a cat.

Types can be defined within the semantic net in two ways: by genus and differentiae, or by prototypes. The former define a restriction which all occurrences of a concept must satisfy. The latter specify sets of family resemblances forming a concept. They are needed since some terms are difficult to express in terms of a unique restriction. Examples are the concepts “game”, and “energy”, which are defined by their usage and the similarity of their occurrences\(^3\).

The first kind of definition expresses the difference between the new concept, and some concept serving as basis for the definition, the genus. For instance:

```
type CIRCUS-ELEPHANT(x) is
  [ELEPHANT:*x] <- (AGNT) <- [PERFORM] -> (LOC) -> [CIRCUS]
type ELEPHANT-CIRCUS(y) is
  [ELEPHANT] <- (AGNT) <- [PERFORM] -> (LOC) -> [CIRCUS:*y]
```

The first type is defined with respect to the genus elephant, and corresponds to an elephant performing in a circus. The second, defined with respect to the genus circus, signifies a circus where elephants perform. Such definitions can be used

\(^3\)See [Wittgenstein 53] for further details.
to expand instances of complex relations such as "to buy" into a form expressing them in terms of more primitive concepts. The topmost unrestricted type is \( T \).

The second type of definition allows occurrences of a concept to be exceptions to the statements that define them. A concept is defined by many schemata which express concepts and relations that are commonly associated with a particular concept type. For instance, a schemata for bus could include the fact it is driven by a driver, contains about 50 people and is used for travelling. This is not restrictive since a bus may appear in a museum. On the other hand a restaurant is restricted to places where people buy and consume food. Prototypes specialise the concepts in one or more schemata to show the form of a typical individual. Prototypes specify defaults that are true of a typical case. For instance, a prototypical elephant weighs around 5400 kilograms, is grey, has two ears...

Sowa argues that both these forms of definition can be used in order to understand stories in the same way as Schank's scripts. However, each script describes only one particular situation, whereas Sowa's definitions can be assembled into scripts according to the situation described. In this manner, he claims his system is more flexible.

### 4.4.5 Complex statements

Negation poses a problem within this representation since one might not only want to negate a single proposition, but an arbitrary structure in the graph. For instance,
if one can negate complex relations, such as "to buy", one should also be able to negate the graph corresponding to their definitions. This is achieved through the use of contexts: a whole piece of net can be referred to as a new concept. This is done using a node with GRAPH as type, and expressing the piece of net as a literal in the referent field:

\[
\text{[GRAPH: [PERSON: Mary] <- (AGNT) <- [MARRY] -> (PTNT) -> [SAILOR]]}
\]

The type label is GRAPH, so the meaning of the graph inside is irrelevant. It is treated purely as a literal, or is "quoted". In order for the meaning to be accessed, the arc STMT is used. The following states that there is a proposition with the nested graph as its statement:

\[
\text{[PROPOSITION] -> (STMT) -> [GRAPH:}
\]

\[
\text{[PERSON: Mary] <- (AGNT) <- [MARRY] -> (PTNT) -> [SAILOR]]}
\]

Propositions are statements of belief, and correspond to normal statements in the semantic net. They can be negated, through the use of the NEG monadic relation (using contraction):

\[
\text{(NEG) -> [PROPOSITION:}
\]

\[
\text{[PERSON: Mary] <- (AGNT) <- [MARRY] -> (PTNT) -> [SAILOR]]}
\]

Propositions are statements and can be believed in, whereas the objects of desires are situations:

\[
\text{[PERSON: Tom] <- (EXPR) <- [BELIEVE] -> (PTNT) -> [PROPOSITION:}
\]

\[
\text{[PERSON: Mary | x] <- (EXPR) <- [WANT] -> (PTNT) -> [SITUATION]
\]

\[
\text{ -> (DSCR) -> [PROPOSITION:
\]

\[
\text{[*x] <- (AGNT) <- [MARRY] -> (PTNT) -> [SAILOR]]}
\]

This states that Tom believes that Mary wants to marry a sailor. In general, negation, modalities, and the patients of verbs like think and know are linked to contexts of type PROPOSITION. Whereas times, locations and the patients of verbs like want and fear are linked to contexts of type SITUATION. Other such information includes tense and aspect, and intersentential relations such as cause, consequence and method.

It is through these nested contexts that the scope of quantifiers is determined. Since PROPOSITIONS can include more than one graph, they can also be used to build AND, OR and other logical relations. Implication is represented using the rule

\[
-(p \land \neg q) \equiv (p \Rightarrow q).
\]

This allows sentences such as "If a farmer owns a donkey, then he beats it" to be represented:
(NEG) -> [PROPOSITION:
[FARMER: *x] -> (STAT) -> [OWN] -> (PTNT) -> [DONKEY: *y]
(NEG) -> [PROPOSITION:
[*x] <- (AGNT) <- [BEAT] -> (PTNT) -> [*y] ]

The statements [*x] and [*y] are unrestricted concepts of type T. This means that they may be coreferent with any concept of any type. To avoid this, special coreference links are used to link the appropriate concepts with farmer and donkey respectively. This is shown in the linear formalism above by using the variables x and y. In the diagram 4.4.5 (p. 64), dotted lines are used.

4.4.6 Evaluation

○ Topological Distance and Determinism of search

CGT is difficult to evaluate in terms of efficiency since it is a theory, rather than an implementation, and therefore leaves out some of the detail necessary for an in-depth evaluation. For instance, the existence of monadic relations could seem to imply that all the conceptual graph is built with only the LINK arc, and all other arcs are notational abbreviations for examples. This would adversely affect topological distance, and determinism of search since more steps would be required through the knowledge base either to attain the desired information, or to determine whether the search path is correct.

In general there appear to be many links involved in expressing relations. For instance a situation is expressed as “the descriptions of propositions stated as graphs”, involving a list of 2 arcs and 3 nodes, where only one is actually refered to.

Moreover, it would appear from the donkey sentence example, that the same concept may be expressed by different nodes linked together by coreference links. This further lengthens topological distance when searching for all known information about any particular concept.

○ Non-linearity

It is unclear whether CGT requires linear reading or not. Indeed this depends
largely on the implementation of the GRAPH primitive: if only one concept of the
nested graph is linked to the graph primitive, it may be necessary to read the
whole of every statement to determine whether or not it is a nested graph, and
then whether it is negated. The rest of the notation appears to be non-linear as
far as statements in the graph are concerned.

Many expressions are written as literals in the referent field of nodes, such as
monetary values, descriptions of sets, and so on. This means that they are not
represented directly in the graph, and cannot enjoy the advantages that such a
notation provides. Thus for instance, their representation is linear.

○ Distributedness

It is clear that CGT has a very low distributedness. This is most easily seen by
the fact negation is applied to a whole proposition. Thus if only the proposition is
considered, the negation of the statement in the semantic net will be obtained. Thus
reading a part of the graph does not give a meaning sound with respect to the whole
knowledge base. This problem extends to the understanding of quantifiers, since the
scope of quantifiers is given by nesting contexts. Thus, in order to determine what
quantification one is dealing with, all the contexts in which the statement under
consideration is, must be traversed. If GRAPH is linear, this can be very expensive!

Another example of non-distributedness is the use of coreference links to give un-
restricted nodes in contexts a conceptual type. This was illustrated by the donkey
sentence, where the entity beating the donkey could be of any type, and was only
restricted by the coreference link. This requires all coreference links to be read,
if the type of a concept is required, which reduces the benefit of expressing the
concept type on the node.

○ Nodes

Many nodes may express the occurrence of the same concept in CGT. To an extent
this is due to the expression of quantification on the node. For instance if the ele-
ments of a set participate in one relation collectively, but in another distributively,
two separate nodes must be specified for the node. They are linked by coreference
links.

However, other concepts such as verbs which participate in many relations are not linked together in this way. It could be argued that this is because each such verb corresponds to a different occurrence of the relation, and thus corresponds to a different occurrence of the concept.

Because the type lattice is separated from the rest of the semantic net, there is no means to find all determining all occurrences of any particular concept type. This is problematic for reasoning tasks such as inheritance.

○ **Uniqueness and Cohesion**

Overall the uniqueness of relations appears relatively high. Indeed, contexts and negation are used even to build up logical connectives. However the conceptual graph does not encode all relations, such as values. This means that cohesion cannot be checked over a wide range of relations.

○ **Richness**

CGT would appear to represent in some manner most natural language statements, even if through the use of literals. However it does not appear to make any provision for handling ambiguity.

On the other hand, the availability of multiple ways of defining concepts, either through genus and differentiae, or though schemata and prototypes allows a large variety of concepts to be defined. This is further enhanced by allowing complex relations to be defined in terms of more primitive relations. The graph expansion feature thus allows reasoning to be performed at whatever level of granularity is required without requiring extensions to the inference machinery. Similarly the potential for scripts to be formed is supported by the variety of ways in which concepts can be defined.

○ **Correctness**

The basic logic of CGT is isomorphic to $\mathcal{FOPL}$. However, the concept hierarchy and system of contexts do not seem to have been modelled formally.
○ Conclusion

Although CGT has many attractive features, it is obviously not built with efficiency as a requirement.

4.5 SNePS

SNePS, the Semantic Network Processing System, “is a system for building, using and retrieving from propositional networks” [Shapiro 94]. As such, it does not commit all aspects of its representation to a precise definition, leaving them open for further experimentation. Its intended goal is “a system for representing the beliefs of a natural-language-using intelligent system (a “cognitive agent”)” [Shapiro et al. 92]. This means that particular attention has been payed to intention, and belief.

Its representation evolved over many years from its first incarnation as SAMEN-LAQ, through MENTAL, SNePS 79, SNePS 2.0 to SNePS 2.3. An offshoot, specifically developed for Natural language processing will be considered separately: ANALOG.

4.5.1 SNePS: Philosophy

The SNePS semantic network is designed to be the mental representation of a cognitive agent: CASSIE. It is therefore intensional, representing what CASSIE believes the state of the world to be rather than the state of the world itself. As such, it can represent concepts which can be expressed intensionally, but may not have any extension: trivially mythical beasts such as unicorns, but also new concepts such as the “Golden Mountain” or a “square circle”. This intensionality forms the basis of the uniqueness principle: “There is a one-to-one correspondence between nodes and represented concepts” ([Shapiro et al. 87], [Rapaport 95]). Thus intensionally different concepts will be expressed as distinct nodes even if they have a common extension: the morning star and the evening star constitute such
a case. The philosophical basis of this intensionality is Alexius Meinong's theory of intensional objects.

In SNePS the meaning of any statement is given by its location with the network. The meaning of every concept thus depends on the whole of the network, and every change to the network affects all concepts. However to avoid complete symbolic circularity, some of the concepts are "grounded" by being associated with certain internal representations of the agents' perception. For instance, the concept "cat" will be linked to the internal representation of what the agent perceives to be a cat in his environment. These internal perceptual representations are also referred to in the network (as "sensory nodes"), but are not described in it.

4.5.2 SNePS: Implementation

4.5.2.1 Network elements

As in other semantic networks, the SNePS network consists of nodes and arcs. Nodes express concepts, "anything about which information can be stored and/or transmitted" [Shapiro 79]. Arcs express relations between these concepts which are non-conceptual in the sense given above. A formal semantics for SNePS is given by [Hill 94] in terms of Aczel's non well founded set theory. Because of its clarity, this discussion shall follow its account of the meaning of the SNePS network elements.

- Arcs

In the SNePS network, arcs are considered to be "grammatical punctuation like the parentheses and commas in the standard syntax of predicate calculus". (see [Shapiro et al. 92]) However this does not mean that they are indistinguishable. Just as there are various kinds of punctuation, different arc labels determine the legal reasoning on the network.

Arcs have two kinds of labels which determine whether they are descending or ascending (or inverse) arcs. Descending arcs express the meaning dependencies, where the meaning of the source of the arc is determined by its targets. Although implemented as ascending arcs, the converse relations to descending arcs are viewed
theoretically as “following descending arcs backwards” [Shapiro et al. 92].

In [Hill 94], (descending) arcs indeed express structure which determines the meaning of each concept. However, arc labels are also considered essential to this task, and are seen in a similar light to sensory nodes in forming the “atoms”, or primitive terms with respect to which the meaning of every concept in the network is given.

In [Shapiro 79] a third kind of arc, an “auxiliary arc” was discussed. This was an implementation device for efficiency, but was eliminated with the advent of SNePS 2.0.

- **Node Types**

There are four functional categories of node in SNePS:

1. **Proposition Nodes:** are the kind of entities which an agent may or may not believe.

2. **Act nodes:** Entities an agent may or may not intend to perform.

3. **Rule nodes:** “are, in some ways, like both propositions and acts. In order for a rule to “fire”, it must be believed, the agent must intend to apply it, and its (appropriate) antecedents must be believed. When a rule fires, the agent forms the intention of believing its consequents”. [Shapiro 91]

4. **Individual nodes:** Everything that is neither an act or a proposition: “Thus individuals include not only traditional individuals, but also classes, properties, relations, etc.” [Shapiro 91]

There are two topological types of nodes in SNePS:

1. **Atomic Nodes:** are nodes which have no (descending) arcs emanating from them⁴.

---

⁴In other words, they only have ascending arcs emanating from them: “isolated nodes cannot be created” [Shapiro et al. 92]
2. **Non Atomic Nodes**: are nodes which have (descending) arcs emanating from them.

Atomic nodes are further subdivided into:

1. **Sensory nodes**: provide syntactic grounding of the network, avoiding circularity of meaning. They correspond to recognizable states that the agent perceives the environment in which it lives to have: "they represent interfaces with the external world" [Shapiro et al. 87].

2. **Base nodes**: represent individual concepts of SNePS. Their meaning depends on the molecular nodes that dominate them.

3. **Variable nodes**: To express propositions, SNePS uses variable nodes, which like those of logic "represent arbitrary individuals or propositions"[Shapiro 94].

Non atomic nodes are subdivided into

1. **Molecular nodes**: represent propositions, including rules, or structured individuals (see 4.5.3 (p. 76)). Their meaning is given by the nodes that they dominate.

2. **Pattern nodes**: "are similar to open sentences or functional terms with free variables in standard predicate logic"[Shapiro 79]. They are like molecular nodes, but they dominate variable nodes.

Molecular nodes and base nodes depend on each other, effectively producing a circular semantics. Despite this, they have been proven to have a unique decoration (or meaning) if they are modeled in terms of Aczel's non well-founded set theory. [Hill 94] only provides semantics for SNePS_p, which excludes variable nodes. This makes unclear whether variable nodes have any meaning, and if so, in what way it determines and is determined by the molecular nodes that it adjoins.

Other node types existed in earlier incarnations of SNePS (SNePS 79, and before) which were implementation devices either to allow efficient inferencing, or to remedy insufficiencies of the representation.
4.5.2.2 Standard elements

SNePS represents beliefs of a cognitive agent as logical statements. Where in logic a statement is true, it is believed in SNePS. Similarly where it is false in logic, it is not believed in SNePS\(^5\).

- Sensory nodes

Sensory nodes are connected to the corresponding SNePS concepts by LEX arcs. For instance, if apple is the class of all apples, and APPLE is the sensory node for apples:  apple: LEX : APPLE;

- Class hierarchy

"A conscious decision was made not to use class membership or any other particular relation for the basic structure of the network, but to allow rules as representationally complete as the predicate calculus to be represented and used" [Shapiro et al. 92].

SNePS uses classes rather than sets to represent concepts such as "animal". Molecular nodes with SUBCLASS and SUPERCLASS arcs relate the classes, so that statements such as "a dog is a kind of animal" to be expressed. Class membership is used to state that what kind a concept belongs to, as in "Rover is a dog":

\[
\begin{align*}
m_1: & \text{ MEMBER : } m_0; \text{ CLASS : Rover;} \\
m_0: & \text{ LEX : DOG;} \\
m_2: & \text{ SUBCLASS : } m_0; \text{ SUPERCLASS : } m_3; \\
m_3: & \text{ LEX : ANIMAL;}
\end{align*}
\]

- Logical Connectives, or “rule nodes”

SNePS 2.1 includes a set of six non-standard logical connectives, which are "as adequate as standard connectives (...) and express common modes of human reasoning simply" [Shapiro 94]. Standard connectives result in large networks because of their binary nature, which is avoided in SNePS by connectives taking "arbitrarily large sets of arguments".

- and-entailment: \(\{A_1, \ldots, A_n\} \land \Rightarrow \{C_1, \ldots, C_m\}\) is true only if each \(C_j\) is

---

\(^5\)See [Shapiro 93] for further information
entailed by the conjunction of all \( A_i \). It is represented as a node with \&ANT arcs to each \( A_i \) and a CQ arc to each \( C_j \).

- **or-entailment**: \( \{A_1, \ldots, A_n\} \lor C_j \) is true only if each \( A_i \) entails each \( C_j \). It is represented as a node with ANT arcs to each \( A_i \) and a CQ arc to each \( C_j \).

- **numerical-entailment**: \( \{A_1, \ldots, A_n\} \Rightarrow \{C_1, \ldots, C_p\} \) is true if the conjunction of any \( n \) of the antecedents \( A_i \) implies the conjunction of the consequents \( C_j \).

- **and-or**: \( \land^i_j \{P_1, \ldots, P_n\} \) is true if at least \( i \) and at most \( j \) of the \( P \) are true. \( i = j = 0 \) is a generalisation of NOR, and \( i = j = 1 \) is a generalisation of XOR\(^6\). It is represented as a node with ARG arcs to each of the \( P \), MIN arc to \( i \) and MAX arc to \( j \).

- **thresh**: \( _\land^i_j \{P_1, \ldots, P_n\} \) is true if either fewer than \( i \) or more than \( j \) of the \( P \) are true. It is represented as a node with ARG arcs to each of the \( P \), THRESH arc to \( i \) and THRESHMAX arc to \( j \).

**default** and **non-derivable** are also used, but less often.

SNIP, the SNePS Inference Package uses these connectives to perform forwards and backwards inference on any information using them.

- **Quantification**

SNePS uses special nodes to represent variables. These are connected to the proposition in which they occur by forall\(^7\) or exists\(^8\) arcs. In logic, quantifiers may appear either at the outermost level of a proposition or can occur within the proposition. For instance, in

\[
\forall x (Man(x) \Rightarrow \exists y (Woman(y) \land Loves(x, y)))
\]

\( \forall \) occurs at the outermost level, whereas \( \exists \) occurs within the proposition. SNePS

---

\(^6\)eXclusive OR

\(^7\)or AVB

\(^8\)also written EVB
mirrors this by allowing quantification arcs to be linked to any of the propositions' rule nodes.

As of version 2.1, existential quantification had not yet been implemented in SNePS, and was expressed instead by skolem functions. The example would be written somewhat like:

\[
\begin{align*}
m_1: & \quad \text{ANT} : m_0; \text{FORALL} : \text{man}_1; \text{CQ} : m_2; \text{CQ} : m_3; \\
m_0: & \quad \text{MEMBER} : \text{man}_1; \text{CLASS} : \text{man}; \\
m_2: & \quad \text{MEMBER} : \text{thiswoman}; \text{CLASS} : \text{woman}; \\
m_3: & \quad \text{AGENT} : \text{man}_1; \text{ACT} : \text{loves}; \text{OBJECT} : \text{thiswoman}; \\
\text{thiswoman}: & \quad \text{Skf loved-by}; \text{A1 man}_1
\end{align*}
\]

"SNePS 2 uses restricted quantification, which means that every quantified expression must have a restriction as well as a scope". [Shapiro 94]. This is represented using and-entailment where the restrictions are the antecedents of the entailment. Notice that in sentences such as "Every farmer that beats a donkey owns it", it is the restriction that ensures that the sentence is equivalent neither to "Every farmer that owns a donkey beats it" nor to "Every farmer beats and owns a donkey".

- **Assertions and belief**

Propositions in SNePS may or may not be asserted, which states whether or not they are believed by the cognitive agent. This allows the full proposition to be distinguished from its parts that are also represented by molecular nodes. Belief follows the standard rules of logic, where belief corresponds to true. Thus from the example in 4.5.2.2 (p. 74), one can infer that all men love someone, but that someone may be loved by a being other than a man.

Molecular nodes dominate the nodes they are connected to by descending arcs. This means that their meaning is determined by the nodes they dominate. Their meaning is not determined by the nodes that dominate them. So, the meaning of a molecular node "John ate an apple" is not changed when qualified by another molecular node as in "Mary believes John ate an apple".

- **Attributes**

Attributes are connected to concepts by molecular nodes taking as \text{OBJECT} the
concept, and as PROPERTY the property being attributed to the concept:

\[ m_1! : \text{OBJECT } m_0; \text{PROPERTY } m_2; \]

\[ m_2 : \text{LEX } \text{YELLOW} \]

means that \( m_0 \), say a canary, is yellow.

### 4.5.3 ANALOG

ANALOG is designed as a more "Natural Logic" for NLP. It argues that the way SNePS represents restrictions on its variables as antecedents of entailments is clumsy and unnatural for any process that must build expressions in SNePS representation from statements in NL.

It introduces as solution "structured variables": instead of linking the quantification arcs to the main molecular node of the proposition, which is usually the entailment taking the outermost variable restrictions as antecedents, it links the arcs to the class membership relations that type the variable. The asserted propositions become the consequences of the entailment. Now the molecular node expressing the entailment can be deleted. The operation is repeated, so that the entailments expressing variable restriction are all removed.

\[ m_0 : \text{MEMBER } \text{man}_1; \text{CLASS } \text{man}; \]

\[ \text{man}_1 : \text{ANY } m_0; \]

\[ m_1 : \text{MEMBER } \text{thiswoman}; \text{CLASS } \text{woman}; \]

\[ \text{thiswoman} : \text{SOME } m_1; \text{DEPENDS } \text{man}_1 \]

\[ m_2! : \text{AGENT } \text{man}_1; \text{ACT } m_3; \]

\[ m_3 : \text{ACTION } \text{loves}; \text{OBJECT } \text{thiswoman}; \]

Because the variables are connected to the class membership molecular nodes, one always knows the restrictions of the variables implicit in any asserted proposition. The main lesson of ANALOG is that it is preferable to express a variables' quantification and restrictions locally to the variable, rather than globally by involving the whole proposition.
Figure 4.2: ANALOG - version of the Donkey Sentence

4.5.4 Applications

SNePS is a general purpose tool for investigating semantic networks: "The set of arc labels used to structure the network is determined by the user, so that SNePS can be used to experiment with different conceptual structures" [Shapiro 79]. However, the motivation for building SNePS is "to be a system for representing the beliefs of a natural-language-using intelligent system."

4.5.4.1 The SNePS standard distribution

The SNePS standard distribution comes with a set of tools: Belief revision, to ensure network consistency, Inferential abilities, and a GATN grammar for NL input. The latter will be discussed with CASSIE.

- SNeBR

"SNeBR (the SNePS system for Belief Revision) is an assumption-based truth maintenance system ensures that SNePS' belief space is always consistent" [Shapiro et al. 92]. "SNeBR is an implementation of an abstract belief revision system called the Multiple Belief Reasoner (MBR), which in turn is based on a relevance system called SWM. ..."
SWM contains the rules of inference of MBR and defines how contradictions are handled”\(^\text{10}\).

- **SNIP**

SNIP, the SNePS Inference Package, allows path based and node based forms of inference on SNePS. Inheritance rules can be input to SNIP, which it will then apply, for instance to allow inheritance. The rules are not written in SNePS representation, but in a special SNIP language. [Shapiro 94, Shapiro et al. 87] The use of a special language allows many different representations to be tested. However SNIP does limit logical connectives to the set of rule nodes discussed in 4.5.2.2 (p. 73).

Because SNIP is a general purpose tool, it cannot be expected to be as fast as a dedicated reasoner, for instance for inheritance. But work has been conducted on SNIP’s efficiency. This has led to good improvements by the use of a learning mechanism that allows SNIP to reuse previous inferences, or form specific rules from the general ones it encountered. [Choi 93]: “Normally ... representations that are expressible are slow, and representations that are fast to execute are not expressible. ... We bridge the gap between performance and expressibility in deductive reasoning systems, especially in natural deduction systems”. All the examples given were hard, but were conducted in a small database and required few facts, which might not be relevant to the type of inference required for NL applications.

### 4.5.4.2 CASSIE

The main project behind SNePS is to build a cognitive agent “CASSIE”\(^\text{11}\) not far dissimilar from LOLITA [Shapiro et al. 87].

CASSIE is a use of SNePS. SNePS provides all the tools used for CASSIE’s implementation: the semantic network, the reasoning tools SNIP and SNeBR, the Natural language analysis and so on. Natural language analysis in SNePS is provided by morphological analysis, followed by parsing by a GATN grammar, and semantic

\(^{10}\text{Shapiro et al. 87}\)

\(^{11}\text{Cognitive Agent of the SNePS System — an Intelligent Entity}\)
rules that transform the input sentence into SNePS representation. The examples of the natural language processed by SNePS are very simple. The lack of deep analysis such as disambiguation, analysis of anaphoric reference and pragmatics shows that SNePS’ natural language component is not very developed. Similarly, all the examples of the natural language analysed by SNePS are very simple, reduced to constants, or explicit descriptions of the quantification as in “For every p and d if p is a person and d is a pet then p loves d”. [Shapiro et al. 87]

It is unclear whether CASSIE has its own knowledge base prior to any NL analysis. If it does, there is no information as to what size it has. Such prior knowledge is essential for any NL interpretation.

SNACTor [Kumar 93] is a model of rational cognitive agents who can also plan and act. Plans are treated as mental objects, which “enables the modelled agent to discuss, formulate, use, recognise, and reason about acts and plans, which is a major advance over operator based descriptions of plans” [Shapiro et al. 92]. The formalism of SNePS was extended to be isomorphic to the OK formalism, to allow the representation of plans, goals, etc. It is now part of the standard distribution, so will probably be used in CASSIE.

4.5.4.3 Applications

Although SNePS is primarily designed to be the basis of a rational agent, it has been applied to many specific domains. The resulting systems use representations very specific to the problems they are tackling. These are in general unsuitable to represent the variety of expression occurring in NL. However, they show SNePS to be a very versatile tool for building applications requiring reasoning and knowledge representation. Descriptions are given in [Shapiro et al. 92, Shapiro et al. 87, Shapiro 79].

Because SNePS is freely available, it has been used by researchers at different universities to study representations suitable for specific problems. Again, for further information, refer to [Shapiro et al. 92, Shapiro et al. 87]. For instance, work was conducted by Michael Almeida into a representation of time suitable for short
narratives.

4.5.5 Evaluation

The philosophy underlying SNePS is similar to that argued for in 2.2 (p. 11). SNePS is intensional, refers to the belief of the cognitive agent whose mind it models, and even has the notion of concepts being grounded by the agent’s perception of its environment. In our terms, sensory nodes can be taken to be reference points in meaning space. Thus concepts which are not directly associated with a sensory node are given a meaning, or area of meaning space, by the manner in which they are associated to concepts which are directly associated to sensory nodes. In these respects, it would seem, SNePS is an ideal representation for LOLITA.

4.5.5.1 Insufficient richness of SNePS

Because molecular nodes always dominate base and variable nodes, it is not possible to express certain statements in SNePS. For instance, “I know what John believes” cannot be expressed. The node JB representing John’s belief is the OBJECT of the molecular node M₁ expressing “Sengan knows JB”, and is the OBJECT of the molecular node M₂ expressing “John believes JB”. The node JB should depend on M₂, since it is the concept “what John believes”. If it does, as in the case of a base node, it depends on all molecular nodes which dominate it. But M₁ also dominates it! So JB is not “what John believes”, but “what Sengan knows and John believes”, which is not the intended concept. The only other possibility is that JB is not defined by M₂, in which case it is “something that John believes and Sengan knows”. Again, this does not express “I know what John believes”.

This proves limiting to the representation, as it is no longer possible to represent the intentional distinction between “The accident that took place at five o’clock involved John Bull”, and “The accident involving John Bull occurred at five o’clock”. In SNePS, if the representation of time is a time arc connected to the molecular (event) node, all the times events occur and the events themselves must be mu-
tually defining. However, if the representation of time is given by molecular node dominating the molecular node that expresses the event, all events define the time at which they occurred. Only by allowing two representations of time can this be solved. The same argument applies to all other representations which apply to molecular nodes: location, cause, belief, source... which in SNePS would require two forms.

4.5.5.2 Variable nodes

• Meaning?

It is unclear exactly where variable nodes fit into SNePS: do they represent concepts? or are they rather a mechanical device to allow quantification to be applied to the basic propositional network SNePS. "The representation and semantics of variable nodes remains a research topic to this day"[Shapiro et al. 92].

The name "pattern nodes" suggests that variables are part of a mechanism that allows the proposition expressed to be derived for any concept that can be substituted into the pattern. Indeed, [Shapiro et al. 87] says of a rule node \( r \) that dominates variable nodes \( v_i \) to which it is connected by AVB arcs, "\( r \) is the Meinongian objective corresponding to the proposition that the rule that would be expressed by \( r \) without the AVB arcs holds after replacing each \( v_i \) by any Meinongian object in its range".

The use of variables seems to solve some of the limitations of richness encountered in SNePS. For instance, if the statement "I know what John believes" were represented in the network as the rule

\[
\forall x(\text{believes}(John, x) \Rightarrow \text{knows}(Sengan, x))
\]

each thing that John believes would fit the pattern, so it could be inferred that Sengan knows it too. There is however no concept of what John believes, but a collection of facts that SNePS believes John to believe.

• Lack of Structure

The use of variables also results in a lack of structure: SNePS and ANALOG use
restrictions on variables rather than building explicit sets. Thus “Every man loves someone” is represented by:

$$\forall x \text{Man}(x) \Rightarrow \exists y (\text{Person}(y) \land \text{love}(x, y))$$

instead of:

$$\forall x \in \text{Man} \exists y \in \text{Person} \text{ love}(x, y)$$

This difference means that a variable restricted by a predicate $p$, and another restricted by the same predicate $p$ and another $q$ cannot be related. For instance, the men in the proposition “All rich men own a car” will not be related explicitly in the network to the men of “Every man loves someone”.

In a set based representation, a subset relation would usually be used to relate these sets so that the additional structure can be exploited by many forms of reasoning, such as inheritance of properties from sets to their subsets, which proves important to natural language analysis, as for instance in disambiguation.

The inability to relate variables in the way sets can be related, makes it impossible to define new concepts and then talk about them within the network. For instance, the departmental football team is constituted by those people in the department who play football for the department. Without the ability to define the concept, it is not possible to make statements such as “The departmental football team won the trophy every year since 1979.” Similarly, it is impossible to represent statements such as “The departmental football team is also the university’s golf team.”

- **An unnatural distinction between variables and constants**

The distinction between variables and constants in SNePS$_{P}$ seems particularly unnatural, as it results in distinctions between pattern nodes and molecular nodes, base nodes and variable nodes. Base nodes can be related by some relations, such as equivalence, but not variable nodes, and so on. These limitations prove unnatural, since it means things can be said about John’s cat which cannot be said about John’s cats. The resulting differences in structure also require two different treatments for building network structure, depending on the existence or not of variables. ANALOG has gone some extent towards solving these problems, but still retains an asymmetry.
4.5.5.3 Uniqueness

Uniqueness is hard to ascertain. According to [Shapiro 79], "Since SNePS is not a particular semantic network but a system for building, operating on, and experimenting with semantic networks, the user is responsible for choosing arc labels". For instance, set membership can be represented in three different ways illustrated in [Shapiro 79]. Uniqueness does not appear to be a high priority, as shown by the use of two different ways of representing set membership in the same example in [Shapiro et al. 92]: by default, every car has an engine.

4.5.5.4 Cohesion

During the course of its long history, SNePS has evolved a great deal, making it difficult to know what exactly constitutes its current status. Even such a simple issue as quantification has changed substantially [Shapiro et al. 92]: in MENTAL, quantifiers were represented as nodes; in SNePS79, they were expressed as arcs, but any rule node could only have AVB or EVB arcs attached to its variables, but not both; in SNePS 2.1, skolem functions implement the dependency of existential quantification on universal quantification; finally in ANALOG, quantification is expressed by structured variables, and DEPENDS arcs.

The same pattern appears for the other subjects that SNePS represents, many of which choose very different representations for specific problems, without concern for general cohesion of the representation.

4.5.5.5 Needs of an NL representation

Despite being geared towards a cognitive agent able to understand NL, the SNePS representation provides no means of expressing ambiguity. In fact, it would seem from the available documentation, that no account at all is taken of ambiguity, at the level of the network.

\[12\text{[Shapiro et al. 92]}\]
Similarly, notions such as values which are very common in NL are lacking from SNePS' formalism. Others, such as a reasonable representation of time are not part of the standard distribution. If one wanted to use the temporal representation of [Almeida 1986] and ANALOG's representation of quantification, substantial re-implementation of the SNePS code would be required. A representation of multilevelled quantification is not defined, although used in NL: "Every general commands a platoon of 30 men", although the idea was mentioned once in [Ali 93].

4.5.5.6 Distributedness and Non linearity

SNePS relies on knowing the types of arcs emanating from and connected to each node, to know its type: variable, pattern, molecular, or base node. This reduces its distributedness, as variable and pattern nodes do not seem to have the status of concepts. However SNePS 2.0 is closer to having a high distributedness than SNePS79, as it includes an assertion tag, which had previously been expressed using the and-or rule node: each molecular node would have to be checked to see whether it was dominated by another node, to see whether it was believed or not. Overall SNePS has good non-linearity, although this may be affected by the need to check the configuration of arcs off a node to determine its type (variable or not).

ANALOG has good distributedness. Although it relies on the existence of the ANY or SOME arc to distinguish between propositions that restrict the variable from those that make statements about it, not reading the arc does not render the information read unsound with respect to the rest of the network. If the ANY or SOME arc is read after the molecular node to which the variable node is connected, one learns that the information expressed by the molecular node was a restriction, but that does not change the information itself. ANALOG's non-linearity is the same as SNePS 2.1.

However, the use of a DEPENDS arc to express quantification scoping prevents the distributedness of both ANALOG and SNePS from being maximal: the statements expressed with or without the DEPENDS arc are very different.
The use of rule nodes to express restrictions on variables does however dampen the usefulness of SNePS’ distributedness, since to be acted upon, information must be believed. Because SNePS uses rule nodes whenever it uses variables, many nodes in the network are not directly believed. Inference must determine from the nodes that dominate them how to apply them to base nodes known in the network, and thus deduce propositions that are believed. This reliance on inference is computationally expensive.

4.5.5.7 Topological distance and Determinism of search

Topological distance in SNePS and ANALOG is high. One reason is the distinction between variables and base nodes: an additional membership proposition must be traversed with respect to a solution which represents the members of the concept and the concept itself in one single node. Another reason is that subtle distinctions seem to be made between concepts which do not bring much benefit. For instance, in Michael Almeida’s representation of time, there appears to be a distinction between an event and the proposition that expresses the event. Again, in ANALOG, some transitive events are represented by two nodes, where one would do: see 4.5.3 (p. 76).

Determinism of search is not good in SNePS or ANALOG, because of the distinction between variables and base nodes. This distinction means that statements about, say, all animals, will be represented by variables. Suppose one knows that Tweety is a bird, and birds are animals, and one wants to infer from what is known about all animals, additional facts about Tweety. The inference algorithm searches up the class hierarchy until it reaches the concept “animal”. In non-trivial circumstances, this will be connected to many different variables, corresponding to different kinds of animals. At this point SNePS must search to determine which variables correspond to animals whose restrictions are compatible with birds: this requires a search through all the variable nodes which are members of the class animal. On the plus side, however, the use of labelled arcs helps improve the determinism of search.
4.5.5.8 Conclusion

Despite its many pluses, SNePS is not the representation needed for LOLITA. This is mainly due to its lack of cohesion, and its inability to represent many of the things necessary for NL. Currently it is still more a system for experimenting with semantic nets than a semantic net suitable for NL processing.

Unlike most other knowledge representations, a formal semantics is provided for a significant fragment $\text{SNePS}_P$ of the representation. Its' most interesting result is that mutually defining concepts may nevertheless be given a unique meaning, ultimately expressed in terms of sensory nodes.

4.6 CYC

CYC is an ambitious project to capture the common sense people share. More details about its motivations are given in A.2.5 (p. A-22). Its interest to LOLITA is its brief: build a large K.B. associated with efficient reasoning tools, and fast access to facts of the K.B. Thus is satisfies many of the requirements of 3 (p. 27). Started in 1984, and still “in business” it provides empirical knowledge of paths not to follow: it is one of the first projects that attempts to deal with the problems that arise in actually representing knowledge in the large. ([Guha et al. 90a]).

Over such a long period the knowledge representation has changed substantially. Unfortunately, little information is available about it. However, in contrast to most other systems is that CYC has two representations: an internal and an external one (CYCL).

4.6.1 The Representations

Initially (before 1991), CYC's representation language (CYCL) was primarily a frame-based language. The CYC K.B. was thought of as a set of unit/slot/entry triples, and inferencing was done pretty much by inheritance. A frame-based system was chosen as initial tests appeared to show that it would be sufficient to express
90% of the statements needed to represent common sense knowledge. However, as time passed, a series of increasingly baroque add-ons and workarounds had to be devised to express statements such as higher-arity predicates. For instance, higher-arity predicates were encoded by tuples, and predicates had many variant forms in which the only difference was the order of the arguments.

[Lenat et al. 90] presented a second version of CYCL. It was still essentially a frame based, but it was overlaid by a form of First Order Predicate Calculus (FOPC) with equality, augmentations for default reasoning, skolemization, and some second-order features (e.g., quantification over predicates is allowed in some circumstances). Like FOPC, CYCL allows using ForAll (universal quantification), ThereExists (existential quantification), and LogImplication (material implication), as well as the other common ways of combining variables and logical expressions such as LogAnd (conjunction), LogOr (disjunction), and LogNot (negation). It uses a form of circumscription, includes the unique names assumption, and can make use of the closed world assumption where appropriate.

According to Cycorp, the current third version is again different.

Little is known about the internal representation, other than it is propriety. However [Whitten 95]\(^\text{13}\) suggests that it is now a semantic network.

### 4.6.2 Conclusion and Evaluation

Little technical information is available as to the features of CYC's internal representation, so the level of such aspects as distributedness cannot be ascertained. However, CYC does provide us with some important and very relevant empirical knowledge. As discussed in A.2.5 (p. A-22), 400000 assertions in the K.B. were found insufficient to encode much of common sense knowledge. This underlines the importance of a representation capable of supporting a very large scale K.B., and efficient search techniques.

Another "historical lesson", was that the second revision of CYCL was motivated

\(^{13}\)And personal communication with Dr. Whitten
by a lack of cohesion. This demonstrates the importance of cohesion. Indeed, even CYCL's second incarnation was far from cohesive. For instance, it associated people with various amounts of money: networth, liquidworth (including bank accounts), spendable money (cash plus credit cards), and actual cash-on-hand. Each of these relations are expressed by a separate slot, rather than being expressed in terms of simpler concepts expressing their meaning.

Technically, the description of the internal representation after the change from a frame based system appears similar to the notion of semantic nets. If this is indeed the case, the fact a mainly empirically motivated effort has opted for a semantic net as basic representation is of particular interest.

4.7 KL-ONE

Under the title "KL-ONE" comes a whole family of knowledge representations derived from Brachman's Ph.D. thesis (1978), a response to [Woods 75]. What binds them together is the separation of terminological and assertional knowledge. The terminological knowledge provides the definitions of the K.B.'s concepts, from which an inheritance hierarchy expressing which concepts subsume which others can be automatically inferred. To achieve this, the terminological language was well-defined by a formal \( \mathcal{FOPC} \) model, which was a novelty at the time. The assertional knowledge is then expressed separately in terms of the defined concepts.

Although defining concepts does not serve much purpose unless one is prepared to use them, the first implementations of lacked an assertional component\(^\text{14}\), and later implementations all differed in the representation they used for assertions: SPHINX allowed Prolog-like rules to be asserted, but KRYPTON used First Order Predicate Calculus. Since the different systems are similar in their terminological components, they are separative knowledge bases. This survey will thus limit itself to the common issues of terminological representation and classification.

\(^{14}\) [MacGregor 91]
4.7.1 Representation

KL-ONE was one of the first representations to emphasise the semantics of the primitives used to define concepts: until then, many systems did not define the meaning of the primitives they used. This "principled" approach meant that the same KL-ONE primitives could be used to express concepts of very different domains and supported reasoning in a general and extensible way. This allowed a new operation to be performed even with concepts new to KL-ONE: automatic classification, which deduces from the definition of a given concept, which concepts of the K.B. it subsumes, and which subsume it.

"Initially, the KL-ONE project set out to develop a set of representational conventions that would be sufficient to express any concept expressible in natural language. (...) These primitives dealt with basic conceptual relationships such as a concept having an attribute, satisfying a constraint, being defined by a set of properties, being more specific than another concept, etc. Concept structuring primitives are contrasted with "primitive" domain concepts such as "ship", "tank", "bagel", "transistor". Concept structuring primitives (perhaps together with some logical primitives for things like sets and sequences) should be the only primitives on which the reasoning algorithms depend." [Woods et al. 92]

4.7.2 Terminological Language

Each of KL-ONE's descendents introduce a new terminological language, to achieve the best compromise between efficiency and richness for their particular purpose. This variety, [MacGregor 91] argues, is to be welcomed and reflects an engineering attitude. Instead of summarising all of the languages of the KL-ONE family, a daunting task addressed in [Woods et al. 92], a taste of the problem is given by the family of languages $\mathcal{AL}$ [Donini et al. 95]:
\[ C, D \rightarrow A \text{ (atomic concept)} \]
\[ \top \text{ (universal concept)} \]
\[ \bot \text{ (empty concept)} \]
\[ \neg A \text{ (atomic negation)} \]
\[ C \cap D \text{ (intersection)} \]
\[ \forall R.C \text{ (universal role quantification)} \]
\[ \exists R.T \text{ (restricted existential role quantification)} \]

where \( R \) denotes a role, which in \( \mathcal{AL} \) is always atomic.

The language is most easily understood in terms of its formal model: An interpretation \( \mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I}) \) consists of a set \( \Delta^\mathcal{I} \) (the domain of \( \mathcal{I} \)) and a function \( \cdot^\mathcal{I} \) (the interpretation function of \( \mathcal{I} \)) that maps every concept to a subset of \( \Delta^\mathcal{I} \) and every role to a subset of \( \Delta^\mathcal{I} \times \Delta^\mathcal{I} \) such that [Donini et al. 95]:

\[
\begin{align*}
\top^\mathcal{I} &= \Delta^\mathcal{I} \\
\bot^\mathcal{I} &= \emptyset \\
(C \cap D)^\mathcal{I} &= C^\mathcal{I} \cap D^\mathcal{I} \\
(\neg A)^\mathcal{I} &= \Delta^\mathcal{I} \setminus A^\mathcal{I} \\
(\forall R.C)^\mathcal{I} &= \{ a \in \Delta^\mathcal{I} | \forall b, (a, b) \in R^\mathcal{I} \Rightarrow b \in C^\mathcal{I} \} \\
(\exists R.T)^\mathcal{I} &= \{ a \in \Delta^\mathcal{I} | \exists b, (a, b) \in R^\mathcal{I} \}
\end{align*}
\]

Thus, one can define the set of men as: \( \text{Men} = \text{Person} \cap \neg \text{Female} \). Similarly, \( \text{Men} \cap \forall \text{has\_child}\text{.Female} \) defines the set of all men whose children are female. And, \( \text{Men} \cap \exists \text{has\_child}.\top \) defines the set of all men who have children.

\( \mathcal{AL} \) can be extended by further constructs [Donini et al. 95]:

- Union of concepts (indicated by the letter \( \mathcal{U} \)), written as \( C \sqcup D \) and interpreted as:

\[
(C \sqcup D)^\mathcal{I} = C^\mathcal{I} \cup D^\mathcal{I}
\]
• Full existential Quantification (indicated by the letter \( \mathcal{E} \)), written as \( \exists R.C \) and interpreted as:

\[
(\exists R.C)^\mathcal{I} = \{ a \in \Delta^\mathcal{I} | \exists b, (a, b) \in R^\mathcal{I} \land b \in C^\mathcal{I} \}
\]

• Complement of the non atomic concepts (indicated by the letter \( \mathcal{C} \), written as \( \neg C \), and interpreted as:

\[
(\neg C)^\mathcal{I} = \Delta^\mathcal{I} \setminus C^\mathcal{I}
\]

• Number restrictions (indicated by the letter \( \mathcal{N} \)), written as \( (\geq nR) \) and \( (\leq nR) \), where \( n \) ranges over the nonnegative integers, and interpreted as, where \( \text{card} \) denotes the cardinality of sets:

\[
(\geq nR)^\mathcal{I} = \{ a \in \Delta^\mathcal{I} | \text{card}\{b | (a, b) \in R^\mathcal{I}\} \geq n \}
\]

\[
(\leq nR)^\mathcal{I} = \{ a \in \Delta^\mathcal{I} | \text{card}\{b | (a, b) \in R^\mathcal{I}\} \leq n \}
\]

• Intersection of roles (indicated by the letter \( \mathcal{R} \), written as \( Q \cap R \), where \( Q \) and \( R \) are arbitrary roles, and interpreted as:

\[
(Q \cap R)^\mathcal{I} = Q^\mathcal{I} \cap R^\mathcal{I}
\]

This defines a family of languages \( \mathcal{AL}[\mathcal{U}][\mathcal{E}][\mathcal{N}][\mathcal{R}] \), where for instance \( \mathcal{ALER} \) stands for \( \mathcal{AL} \) augmented by full existential quantification and role intersection.

More complex statements such as \( \text{Woman} \cap (\leq 2 (\text{has\_child} \cap \text{has\_female\_relative})) \), signifying “women having at most 2 daughters” or \( \exists \text{has\_child} . \text{Female} \) signifying “individuals having a female child” can now be built.

An important difference to other representations is that using the role-values in role restrictions (e.g. \( \forall R.C \) is a role restriction with role-value \( C \)) does not change them: In \( \text{Woman} \cap (\leq 2 \text{has\_child} . \text{Female}) \), no subconcept of \( \text{Female} \) will be constructed to correspond to the female children of women.

### 4.7.3 Automatic classification and subsumption

In KL-ONE the operation of determining a new concepts’ place within an inheritance hierarchy is called classification. The fundamental operation classification
uses is subsumption. This decides whether or not a concept $C$ is subsumed by another $D$ from their definitions. [Hollunder et al. 90] Automatic classification provides the bedrock of KL-ONE’s efficiency. The 1984 paper [Brachman et al. 84] discovered an unexpected complexity cliff: subsumption in a language $\mathcal{FL}^-$ takes in the worst-case polynomial time, but in the language $\mathcal{FL} - \mathcal{FL}^-$ plus the role restriction operator – subsumption is in the worst case Co-NP-hard. It also argued that for a K.B. server to be useful, the worst-case complexity of the classification algorithm should be polynomial.

Since [Brachman et al. 84] identified these problems, the complexity of subsumption has been extensively studied. [Schmidt-Schauf 89] showed that subsumption in $\mathcal{ACR}$, a subset of KL-ONE’s original language is undecidable, so KL-ONE’s language is also undecidable. That is to say that KL-ONE’s language’s syntax allows statements to be written which cannot be proven or disproven from KL-ONE’s axioms. Thus, no algorithm can be produced which states for any pair of concepts written in KL-ONE’s language whether or not one of them subsumes the other. Another important result was [Nebel 91]’s proof that expanding terminological definitions is inherently Co-NP-complete.

The first subsumption algorithms were incomplete [Donini et al. 95]. They were structural algorithms [Baader et al. 94] which basically compare the strings corresponding to the definitions of $C$ and $D$, after normalising them to make their syntax similar. Although this works for descriptions containing conjunction, value restrictions, and number restrictions in only polynomial time, it fails to cover all cases when other features are introduced into the language.

The first complete subsumption algorithm appeared in [Schmidt-Schauf et al. 91]. It introduced a new paradigm of subsumption algorithms by converting the subsumption problem into a satisfiability problem: $C$ is subsumed by $D$ if and only if $C \cap \neg D$ is unsatisfiable. Proving satisfiability is achieved using a scheme similar to tableaux calculus for $\mathcal{FOPCL}$ using rules to control the search space and to ensure termination [Hollunder et al. 90]. The resulting algorithms are complete, so reflect the complexity of the problem, without adding complexity for internal
book-keeping. [Donini et al. 95]

The tableaux based system considers the meaning of the definitions by constructing all possible abstract classes of meaning allowed by the two definitions, and seeing whether all of C’s fit in D. This is equivalent to seeing if any meaning can be built for C ∩ ¬D. The algorithm uses constraint systems to express the interpretations it considers: each constraint system expresses an abstract class of meanings. In practice, the algorithm uses symbols to express an instantiation of some concept (be it explicit, or implicit as in a restriction). Thus, for each solution, the concepts’ domains are actually constructed. If the interpretation building rules of a language suggest more than one possible meaning, the number of possible meanings is exponential. To prove unsatisfiability, every possible interpretation must be checked, which renders unsatisfiability potentially exponential in the number of its terms. It is because they consider the meaning domain, rather than the syntax, that one can be sure that the algorithms are complete.

Since [Schmidt-Schauß et al. 91], a flurry of papers appeared on the complexity of satisfiability in a variety of terminological languages. [Donini et al. 95] recapitulates the main worst-case results\textsuperscript{15}:

<table>
<thead>
<tr>
<th>$\mathcal{AC}$</th>
<th>polynomial</th>
<th>NP-complete</th>
<th>Co-NP-complete</th>
<th>PSPACE-complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C \cup D$</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>$\exists R . C$</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$\geq nR$</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$\leq nR$</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>$R \cap R'$</td>
<td>-</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

Until recently, many implementations used weakened terminological languages to achieve almost complete inference. However, most users prefer more expressive languages. The recent results that only the weakest languages have polynomial satisfiability complexities[Donini et al. 95][Woods 91], increase the trend towards more

\textsuperscript{15}PSPACE-complete means the algorithm is NP-complete timewise, but only requires a polynomial amount of memory.
complete languages at the expense of completeness. However, [Doyle et al. 91] criticises incompleteness by pointing out that if an application need to make inferences the KBMS is unable to make, the inferences will be implemented somewhere else, destroying the conceptual coherence of the knowledge base. CLASSIC [Borigida 92] answers this by allowing users to define their own subsumption algorithms for any language extensions the users define: CLASSIC’s structural subsumption algorithm is complete for the weak language it provides. For its part, KRIS attempts to provide an optimised yet general complete non-structural subsumption algorithm that has exponential behaviour in the worst case, but is as efficient as implementations using smaller languages when the database contains information that could be expressed only in those languages: Results of empirical tests described in [Baader et al. 94] show good behaviour on realistic knowledge bases, and average behaviour on randomly generated knowledge bases.

KL-ONE’s complexity results are interesting in that they show that the complexity of subsumption is bad in any useful language: that there is no point getting hung-up about this in the design of the representation. However, the complexity results are not all bad news: [Wilf 86] proves that the number of steps using backtracking required to decide whether a graph of $L$ vertices can be coloured with $K$ colours\textsuperscript{16} is bounded below: $\sum_{L=0}^{\infty} K^{L} \cdot 2^{L/2} 2^{-L^{2}/2K}$, an infinite series which converges to a finite value!. Thus, instead of considering the worst complexity, which need not be indicative of the behaviour of real data, it is worth considering the average case.

This is what Woods did in his paper [Woods 91], by showing by “back of the envelope calculations” that determining whether a concept intensionally subsumes another can be achieved on average in $O(\log(N))$ steps, so the classification of a concept can be achieved in $O(N \log(N))$ steps. The key point is that $N$ is the size of the database, so instead of achieving exponential complexities in terms of the size of the terms’ definitions, Woods achieves near-linear complexity in terms of the database’s size. To achieve this result, he argues that extensional subsumption – that is to say a concept $x$ subsumes a concept $y$ if and only if every instance of

\textsuperscript{16}a well known NP-complete problem.
Chapter 4: Literature Review

$y$ will necessarily be an instance of $x$ – is the cause of the intractability problems. Instead, he suggests defining an intensional subsumption, which is required to be sound – i.e. if it states that $x$ subsumes $y$, then every instance of $y$ must be in $x$, BUT if it does not state this, it is still possible that every instance of $y$ is in $x$. Thus the intensional subsumption is incomplete, but its choice may be motivated by psychological grounds or efficiency (e.g. the use of polynomial structural algorithms). Woods then takes a simple example of intensional subsumption which cannot determine that “person whose limbs are hairy” subsumes “person whose arms are hairy and whose legs are hairy”. Extensional subsumption can determine this case, but Woods maintains that people would not without some conscious effort.

4.7.4 Evaluation

Although KL-ONE was implemented to support Natural Language concepts, the fact no explicit subsets are built means that sentences such as “Every man’s wife’s mother irritates him” cannot be expressed. Thus, KL-ONE cannot be used for the type of processing, LOLITA requires. KL-ONE’s uniqueness is low, because many of the language’s constructs allow one to express the same operation – for instance full existential role quantification can be obtained by role intersection and concept union [Donini et al. 95]. However, the concepts that are defined are unique by virtue of the automatic classification.

The two main results from KL-ONE are, for this thesis, the discovery that to be tractable a language must be so weak that it is virtually unusable, and the idea of automatic classification: that the definition of a concept must be used to determine it’s place in the concept hierarchy.

4.8 Defeasable Inheritance

Not truly a system of representation in itself, defeasible inheritance is a tool used by many representation systems. Its essence lies in Marvin Minsky’s comment
“What I’m getting at is that there is a problem with exceptions. It is very hard to find things that are always true”. This is because, it is claimed, the real world contains exceptions to almost every generalisation. “Although most people’s ideal elephant is a grey, four-legged, jungle dweller, there are non-grey elephants, three-legged elephants, elephants who don’t live in jungles” [Touretzky 86]. Defeasible inheritance claims to be a solution to these problems.

[Touretzky 86] presented the first formalisation of defeasible inheritance, which corrected the flaws existing in previous systems. In particular the addition of multiple inheritance ("Clyde is an elephant", "Clyde is a circus performer" and "Circus performers are performers" so Clyde should inherit all the properties of elephants and of circus performers) creates problems for defeasible inheritance ("Elephants are shy" and "Performers are not shy": is Clyde shy or not?). Many previous implementations of defeasible inheritance would conclude that Clyde is shy because Elephant is one step away from Clyde (Clyde is an Elephant) whereas Performers is two steps away (Clyde is a Circus Performer is a Performer). Touretzky solves this by replacing distance by a partial order: Shy Thing is above Performer and Elephant, so Performer < Shy Thing > Elephant. Similarly Performer > Circus Performer > Clyde < Elephant. Note that no ordering is stated between Elephant and Circus Performer (or Performer). This allows him to conclude that one cannot conclude that Clyde is either shy or not-shy.

However, in [Selman et al. 91] Touretzky’s inheritance algorithm was shown to be NP-hard. Similarly the conditioning measures he proposed to render accesses to the knowledge base efficient (\(O(N.log(N))\)) prove NP-hard. Although alternatives have since been found [Horty et al. 87], it is clear that defeasible inheritance is computationally more expensive than non-defeasible inheritance.

Furthermore, the need for defeasible inheritance does not appear water-tight. A white three legged elephant is not an exception if the K.B. is told that elephants are born with four legs and that a large subset of elephants is the set of grey elephants. Instead, it is not an elephant of the grey majority, and it has only three legs. Even if the elephant was born with three legs, and a deeper model is
not used, homogeneity theory can cope with this type of exception. Thus one can maintain that exceptions are unnecessary, most of them arising from imprecise use of language, and the rest from expecting all defining properties to apply to all of the concepts’ instantiations. Indeed, in a defeasible system “birds fly” means all birds fly unless some birds are known not to fly, “ostriches don’t fly” means all ostriches do not fly unless an example of an ostrich that does fly is known: even if the properties apply to an exception (ostriches), they apply to all instances of that exception. Homogeneity theory, however, allows for penguins that fly or penguins with long necks, but starts becoming less convinced for penguins with long necks that fly – perhaps a heron?

4.9 LOLITA’s Semantic Network

When the author joined the LOLITA project in 1992, LOLITA already had an internal representation, which shall be referred to as LOLITA 92. Most of the information in this section comes either from [Garigliano et al. 92] (the only document produced at that stage) or the source code and Dr. Roberto Garigliano, the project lead.

4.9.1 Philosophical assumptions

LOLITA 92 assumes that the meaning of any node is represented by that node together with the whole of the semantic network. Nodes which are close to a specific node will contribute more to its meaning that those further away. It is impossible to define the meaning of a node by considering only those nodes within an arbitrary distance of it. Distributedness and Non-linearity are recognised as being necessary to ensure that the whole network need not be read for useful information to be derived. In LOLITA 92, a distributed knowledge base is defined to be one of which “any section gives information which is sound with respect to the knowledge base’s semantic model”. There is no notion of gradation or of atomic symbols. LOLITA 92 defines non-linearity in the same way this thesis does. However the relationship
between distributedness, non-linearity, compositionality and cohesion has not been realised.

LOLITA 92 considers normalisation an acceptable alternative to uniqueness and cohesion to be unnecessary. Indeed in internal discussions, a certain amount of redundancy (multiple mechanisms) was sometimes argued necessary to cope with the complexity of LOLITA’s problem. Naturalness and language independence are assumed.

LOLITA 92 does not equate concepts with sets, understanding the impracticability of computer implementations based on the mathematical definition that sets are defined by the elements they contain. However, it does not take the next step understanding that the network is intensional so the evening star and the morning star should be represented by different nodes, being different concepts.

The network is assumed to be formed of descendents of typeless, as this reduces inheritance’s search-space, but the implication that this requires some form of automatic classification has not been realised.

4.9.2 Basis: nodes and arcs

LOLITA 92 is semantic network with arcs and nodes.

Every arc has a label and a direction (a source and a target). The label gives it its type. Some arcs specify the internal structure of an event (subject_, object_, and action_). The others express relations between concepts.

Every node represents a concept. It is associated with a label, a set of controls and a quantification. The label states the word to which it corresponds. The controls state what type of node it is: an event, or an entity. Events are relations between concepts, have a type specified by the target of their action_ arc, and take two arguments connected by a subject_ and an object_ arc respectively. Entities are all other nodes.

All nodes are built into an inheritance hierarchy, which is assumed to be strict, but lacking automatic classification is not: every property inferred from an explicit
ancestor is sound with respect to the full semantic net, but not every one of a node’s ancestors are necessarily above it in the inheritance hierarchy.

Events and entities have further subtypes specified by their controls. An important event subtype are prototypical events (in SemNet template events) which specify the types of arguments an event of a given type may take. Every event is built as a child of its prototypical event. Similarly, entities’ types (such as human, animate, etc) are specified by the controls, allowing common decisions to be made without searching the inheritance hierarchy.

4.9.3 Quantification

LOLITA 92 expresses quantification as a control on each node. Quantification states in which way an event applies to a concept or its instances. There are 3 types:

4.9.3.1 Individual quantification

Individual quantification on a node \( n \) is used to state that the events connected to \( n \) consider \( n \) to be an individual concept. Nodes expressing sets cannot have individual quantification. An example is "Francis ate an apple": the nodes "Francis", "eat", "apple" and the eating event all have individual quantification.

If an event is viewed as a predicate of \( \mathcal{FOPL} \), whose name is given by the event’s action, whose subject is the first argument, and whose object is the second, individual quantification can be seen to correspond to constants and to existential quantifiers which are not dependent on any universal quantifier, as illustrated by:

\[
\exists x \in \text{Apples} . \text{eat}(\text{Francis}, x)
\]

4.9.3.2 Universal quantification

Universal quantification states that the relation applies to all elements of the set expressed by the node. An example would be "Francis ate every apple". Here the
nodes "Francis" and "eat" have individual quantification whereas "apples" has universal quantification. "Every saint shares all things" would be represented with both the nodes saint and things being universally quantified.

LOLITA's universal quantification corresponds to $\mathcal{FOPL}$ universal quantification:

$$\forall x \in\text{Apples} \ . \ \text{eat}(\text{Francis},x)$$

$$\forall x \in\text{Saints} \ \forall y \in\text{Things} \ . \ \text{share}(x,y)$$

### 4.9.3.3 Existential quantification

Existential quantification allows the choice of an element of one set to be bound to the choice of the element of another. For instance, "every mother gives birth to a child" means that each mother gives birth to a different child. Here the child is existentially quantified and depends on the choice of mother, a universally quantified set.

If a node has existential quantification, it must be a set. It must also be connected to at least one event $e$ by a subject- (or object-) event, which itself has as object- (or subject- respectively) a universal node $u$. If the action of $e$ is $a$, then for every element $x$ of $u$, there is an element $y$ of $n$ such that $y$ as $x$ (or $x$ as $y$ respectively).

The choice of $x$ determines the choice of $y$. In "every farmer owns a donkey", the subject- node (farmers) has universal quantification, and the object- node (owned donkeys) has existential quantification.

An existentially quantified node corresponds to the set of all elements that participate in the relation In $\mathcal{FOPL}$, if a relation involves a term quantified existentially over a set, it does not imply that the set is limited to elements participating in that relation. For instance,

$$\forall x \in\text{Farmers} \ \exists y \in\text{Donkeys} \ . \ \text{own}(x,y)$$

does not imply that the set of Donkeys includes only donkeys that are owned by farmers. However, if the corresponding existentially quantified donkeys node $d$ were to follow the same scheme various problems would arise. The first is that $d$ would refer to the same set as its superset, the set of all donkeys, which breaks uniqueness. The second is that it provides no means to refer to the set of donkeys.
owned by farmers: for instance, one might want to specify that there are five such donkeys. LOLITA 92's solution is to require that nodes corresponding to existentially quantified sets include only elements that participate in the relation. Thus in the following example, Donkeys$\_1$ refers to the set of all donkeys that are owned by Farmers$\_1$.

\[ E_0: \{ \text{subject.: (Farmers}_1, \forall); \text{action.: (own, I); object.: (Donkeys}_1, \exists) \} \]

Since the node corresponds to the whole set, LOLITA 92 existential quantification is not directly assimilable to FOPC existential quantification.

Existential quantification does not imply that for every instance of the universally quantified set(s), there is a different instance of the existentially quantified set. For instance "every child has a mother" is represented:

\[ E_0: \{ \text{subject.: (Children, \forall); action.: (have, I); object.: (Mothers, \exists)} \} \]

Existential quantification in LOLITA 92 therefore corresponds to the following in FOPC, if the event corresponding to the relation $r$ has the universal set $A$ as subject., and the existential set $B$ as object.:

\[
(\forall x \in A \exists y \in B. \ r(x, y)) \land (\forall y \in B \ \exists S \subseteq A \ \forall x \in S. \ r(x, y))
\]

or more simply,

\[
(\forall x \in A \ \exists y \in B. \ r(x, y)) \land (\forall y \in B \ \exists x \in A. \ r(x, y))
\]

4.9.3.4 Quantification of events

Events are represented by nodes, so are also assigned a quantification. If only one event occurred, then it will be given individual quantification. An example would be "Caesar was born”. However sentences such as "every man ate an apple” involve many events, one for each man eating an apple. This would be therefore be given a universal quantification.

Although the event may have a quantification, the dependency of existential quantifications on universal quantifications occurs only between the subject. and object. of any given event. The sentence "every man eats an apple” is represented by the universally quantified event:

\[ 17 \] The nodes are paired with their quantifications
(Event_0, ∀): { subject_: (Men, ∀), action_: (eat, I), object_: (Apples, ∃) }

If the quantification dependency were between the event and the targets of its subject_, action_, and object_arcs, the event above would state that all men of Men participated in every one of the events, and for each of these events, there was an apple being eaten. This is not the intended meaning: for every man of Men there is an apple being eaten (and a corresponding event). Equally, if the event were given an existential quantification, the existentially quantified target of the event’s object_ would depend on an existentially quantified event, which is not possible since existential sets depend only on universal sets.

Despite the lack of dependency postulated between events and the targets of their arcs, quantification on events is still used in some sentences. For instance, “John wakes up every morning” involves a set of events. However the event “John wakes up” only involves an individual subject_ and action_. It is therefore possible to have sets of events when the subject, action, and object (for transitive events) are all individuals. In this case the event is existentially quantified and the subject_ of another event, with action_ time and object_ the universally quantified set of mornings.

4.9.4 Sorts

Unlike KL-ONE, LOLITA 92 does not separate a concept’s definition from the things said about it into an A-Box and a T-Box. Instead, an event defines a concept if the concept has no ancestor which is also of that event type. Non-definitional events (observational events) are those explicitly connected to a concept but could be inherited from the concept’s ancestors. Definitional events can only be inherited virtually down the inheritance hierarchy – copying them into the network would change the concept’s definition. Thus, in the following example, the event of action a is observational for Y but definitional for X:
Chapter 4: Literature Review

\[(X, U): \{\text{spec.: } (X,U)\}\]
\[(Y, U): \{\text{spec.: } (Y,U)\}\]
\[(E_0, R, U): \{\text{subject.: } (X,U); \text{action.: } (a,I); \text{object.: } (Y,U); \text{spec.:}(E_1,U)\}\]
\[(E_1, R, U): \{\text{subject.: } (X,U); \text{action.: } (a,I); \text{object.: } (Y,U)\}\]

To simplify the building of the representation, there is another rule: if a set is qualified by a size event of size less than all, it is maximal with respect to the events it is connected to. This can however be transformed into the standard representation by copying all the set’s parents’ definitional events down to it. Thus, “all people who live in tents live in deserts” can be built as:

\[(X,U): \{\text{spec.}(Y,U)\}\]
\[(E_2, R, I): \{\text{subject.: } (X,U); \text{action.: } (\text{live.in.deserts},I)\}\]
\[(E_3, R, U): \{\text{subject.: } (Y,U); \text{action.: } (\text{live.in.tents},I)\}\]
\[(E_4, R, U): \{\text{subject.: } (Y,U); \text{action.: } (\text{size.},I); \text{object.: } (\text{less.than.all},I)\}\]

Determining whether an event is observational or definitional is not simply determining whether it could be inherited from an ancestor. For instance, consider the statement “every farmer who owns a donkey beats it”. This can be represented by stating that the set of farmers who own donkeys is a subset of those that beat donkeys, and that for each farmer of this set the donkey that the farmer beats is the same as the one he owns: figure 4.3 (p. 104). \(F_1\) is the set of farmers who beat their donkeys, \(F_2\) is the set of farmers who own and beat their donkeys, \(D_1\) is the set of donkeys beaten by some farmer, \(D_2\) is the set of owned and beaten donkeys. This will be referred to as scheme A.

From this excerpt of network, \(E_1\) would appear to be observational, since it can be inherited from \(E_0\). This is indeed the case: \(E_0\) restricts \(F_1\) to farmers who beat donkeys, and restricts \(D_0\) to those donkeys that are beaten. \(E_2\) restricts \(F_2\) to farmers who own a donkey, and \(D_2\) to donkeys that are owned by farmers. \(E_1\) would restrict these farmers who beat donkeys, but this has already been done by \(E_0\), so \(E_1\) is observational. However, what exactly \(E_2\) means depends on whether there is a statement restricting a superset of \(F_1\) or \(D_1\):
Figure 4.3: Every Farmer who owns a donkey beats it

1. If there are no inherited or explicit statements involving farmers owning donkeys, then $D_2$ is the biggest set restricted by being a donkey and being owned by farmers, so $D_2$ is the set of donkeys being owned by farmers.

2. If there is some statement that all farmers own a donkey, attached to Farmers, then $D_2$ is not the biggest set restricted by being a donkey and being owned by a farmer, so $D_2$ is a subset of donkeys owned by farmers.

The first case corresponds to “all farmers who own a donkey beat it”. The second case however throws back into question the interpretation of $E_1$ as observational: $E_2$ would now be considered observational too since it is inherited from the statement “all farmers own a donkey”. Thus, $F_2$ and $D_2$ would appear only to be connected to observational events, and not to be defined by any definitional ones, which is not allowed since it breaks uniqueness. Moreover, this is not the meaning of $F_2$ or $D_2$: assume $E_3$ is the event “all farmers own a donkey”, and $D_3$ the set of such donkeys; $D_1$ is now the set of donkeys that are beaten by farmers, but are not
necessarily owned by any farmer; $F_1$ is the set of farmers who beat donkeys, and they would inherit $E_3$ so that there is a (virtual) set of donkeys that they own; $F_2$ is then the set of farmers who own the donkey that they beat: this is a different set to all the others, so is defined by $E_2$ and $E_3$.

This source of complexity also occurs in more subtle examples such as "People own donkeys" or "All farm animals are owned by farmers" if donkeys are defined as farm animals. Additional intermediate sets must also be taken into account as in: "some farmers of the farmers who beat their donkeys, and all the farmers who own their donkeys are rich", where being rich is an observational feature of donkey owning farmers. Since the question is whether an event says more, restricts further than its ancestor (i.e. is definitional), it depends not only on the event and the inheritance hierarchy but also on other events qualifying its subject and object. Building statements is also rendered more complex. For instance, if the KB contains "all farmers who own a donkey beat it", adding "all people own donkeys" changes the first statement's meaning: $F_2$ changes from all farmers who own a donkey to some farmers who own a donkey. This change in meaning must be rectified by changing the previously existing statement to "all farmers own a donkey and beat it": this is what the two statements "all people own a donkey" and "all farmers who own a donkey beat it" imply. This is built in Scheme B of 4.3 (p. 104), where there is a spec. event from $P_1,D_1$ to $F_0,D_2$ respectively. $P_1$ is the set of people who beat the donkey that they own (since $P_0$ is the subject of an "all people own a donkey" event); and $D_1$ are these donkeys. $F_0$ is the set of all farmers, and all these farmers are observed to own and beat their donkeys ($D_2$).

4.9.5 Extended representation

LOLITA 92 expresses belief with a belief. event and a status. arc connected to the event being believed. If an event has no status. arc, then it is believed by default.

LOLITA 92 expresses time with a time. arc. This connects either to a tense node or a date. The tense nodes are: Pres. (present), Past. (past), Fut. (future), Instant.
(present progressive), PastInst_ (past progressive), PastPresInst_ (present perfect progressive), PastPastInst_ (past perfect progressive), PastPres_ (present perfect), PastPast_ (past perfect), FutPast_ (future perfect).

LOLITA 92 expresses source information with a set of arcs: day_, month_, year_, source_ [Bokma et al. 92].

LOLITA 92 lacks representations of ambiguity, intension, parts, values.

4.9.6 Evaluation

4.9.6.1 Insufficient Richness of LOLITA 92

• Existentials

A difficulty arises with existentially quantified sets. It is best illustrated by an example. One wants to represent the sentence "every mother has a child, and each of these children loves a toy". Representing the first part is simple enough:

\{ subject.: (Mothers, \forall), action.: (have, I), object.: (Children, \exists) \}

In order to represent the second part, one would like to write:

\{ subject.: (Children, \forall), action.: (love, I), object.: (Toys, \exists) \}

However, this would be assigning Children two quantifications, an existential one and universal one. This is not possible. If one chooses the existential quantification for Children, Toys' existential will not have a universal on which to depend. If one chooses universal quantification, the first part of the statement will state that all mothers have all children. To understand this, keep in mind distributedness allows one to read any part of a statement independently from the rest of it, and still obtain information which is sound with respect to the knowledge base. A similar problem arises when one wishes to state that every mother has a child, and each of these children loves all toys.

The problem is that quantification is represented on the nodes, whereas in fact it specifies how the relation specified by any given event applies to the relation's arguments.
• Quantification of Events

The rule that the targets of events’ arcs are not quantificationally dependent on the event makes it impossible to express certain sentences. For instance, "Every day I buy an apple" Here the choice of the apple depends on the day, as illustrated by the analogous sentence: “every day I water an apple tree”. The way one would try to express it is:

\((Event_0, V)\): \{subject_: \((Event_1, \exists)\), action_: \((time, I)\), object_: \((Days, V)\)\}

\((Event_1, V)\): \{subject_: \((Sengan, I)\), action_: \((water, I)\), object_: \((AppleTrees, \exists)\)\}

However, there since there no dependency is allowed between \(Event_1\) and the target of its object_, the statement cannot be expressed. Moreover, in the example \(Event_1\) has been given both universal and existential quantification which is not possible.

• Quantification is intrinsic to relations, not concepts

This section has uncovered two problems. The first is the rule that restricts existential dependency only to arguments of some event in order to enable statements such as “every man loves a woman” to be represented, is too restrictive. The second is that where a dependency between the event and the targets of its arcs is wanted, the event may be existentially dependent on the object of a time event of which it is the subject, yet be the universal set on which its object depends. This means that quantification dependency is intrinsic to relations, in this case either the time event or the object arc, but is not intrinsic to nodes.

• Arcs used for events

LOLITA 92 expresses many relations as arcs which prohibits other events from qualifying them. For instance, the cardinality of a set is given by the size arc, so statements such as “John thinks there are 5 apples in the basket” cannot be expressed. Similarly, despite the above example there is no time event, but a time arc which prohibits statements of belief about about an event’s time.
4.9.6.2 Low Distributedness

• Belief

The fact that events are believed by default unless they are attached to a status arc means that any section of the network which discards the status arc is unsound with respect to the network’s semantic model.

• Quantification

The fact that some events apply to sets rather than to their elements breaks distributedness: for one to know whether an event refers to the elements of any set, or the set itself, one must first read its action.

• Lack of sorts

The scheme to simplify building events breaks distributedness since without the less than all size arc, the event inherited to the lower set is definitional. Moreover inserting an event between the two sets changes the higher set’s events from being inherited as observational to being inherited as definitional.

The scheme of using the inheritance hierarchy to distinguish definitional events from observational events breaks distributedness: an event may appear to be definitional if the events of the ancestor of the concept to which it applies are not fully read.

4.9.6.3 Uniqueness

• Arcs and events expressing the same concept

Due to hardware constraints, LOLITA 92 represented many relations by arcs. In particular, the set relations were implemented as spec. and inst. arcs (subset and instance respectively). However, this meant certain statements could not be expressed: “John says Berny is a horse”. This was fixed by adding an is-a event as follows:

\[(Event_0, I): \{\text{subject.: (Berny, I), action.: (is-a, I), object.: (Horse,V)}\}\]
\[(Event_1, I): \{\text{subject.: (John, I), action.: (say, I), object.: (Event_0,I)}\}\]
This leads to a break in uniqueness which substantially complicated the code for semantic integration, semantic selection rules and pragmatics. Indeed many parts of LOLITA ignore is_a events completely.

- No values

LOLITA 92 lacks a representation of values. This results in many different ways of expressing the same concept.

4.9.6.4 Cohesion

- Predicates on the set

Not all events connected to a node corresponding to a set make statements about its elements. For instance, events specify whether sets have subsets and instances. These relations consider the sets as constants, yet the sets have universal or existential quantification. For such relations to be possible one must postulate that events correspond to two types of relation, one which is subject to quantification and the other which is not. For instance, this renders the treatment of negation more complex: the process of negating an event depends on the quantification of the concepts that serve as its arguments.

- Lack of sorts

The scheme to simplify building events reduces cohesion since it requires additional rules to interpret it.

The scheme of using the inheritance hierarchy to distinguish definitional events from observational events results in a fragile inheritance hierarchy where repeating an event further down the hierarchy renders it observational. The changes in the network due to adding simple statements such as “all people own a donkey” can be complex, especially when combined with issues such as belief and cause. For instance, if John believes that “all people own a donkey”, and LOLITA and he believe that “all farmers that own a donkey beat it”, then LOLITA’s belief in the statement will be expressed differently from that of John’s. One will involve scheme A, the other will involve scheme B. This is because scheme A assumes that the
second statement also means that not all farmers own a donkey, whereas scheme B assumes that they do.

- **Events**

Although events form their own hierarchy, action types also form an inheritance hierarchy. Thus an event may have one parent connected to it by a `spec.` or `inst.` arc, and another parent whose `action.` is connected to its `action.` by a `spec.` arc. This converts a search up the event hierarchy from linear complexity to greater than linear complexity in the number of actual ancestors. Moreover, all action types are individually quantified, requiring the addition of special rules to handle their `spec.` hierarchy.

- **Time**

Source control uses a different representation to the time representation to express time.

The representation also represents tense directly in the knowledge base. But tense is an anaphoric relation, dependent on the order of the sentences in the text, and the time the text was uttered. Neither of these are recorded. Expressing it as an absolute in the knowledge base means either that every reasoning tool using time must be able to infer the order of events from the order of the text (supposing it were recorded), or that mistakes will happen. For instance, this could result in “John was awarded a medal. He had been seen stealing an apple.” being a possible summary of “John fought against the invader. He was awarded a medal. But a few months later he was shunned. He had been seen stealing an apple.”

### 4.9.6.5 Topological distance

The scheme of using the inheritance hierarchy to distinguish definitional events from observational events requires a lot of search to determine whether events are observational or definitional. Moreover, for large-scale applications, the repetition of observational events, and the sets created in order to express them as observational will result in a very large network, as most events within it will be observational.
Indeed, the purpose of defining concepts is to be able to make statements about them. As an example of a set built simply to state an event is observational, consider the statement "there are only 3 farmers in New York". This must be built as a subset of the set of sets of three elements. Since one is unlikely ever to wish to refer to this set, it involves a cost of four nodes for no need, and of two more per concept which cannot inherit the observational event.

4.9.6.6 Language Independence, Non-Linearity, Determinism of search, and Packedness

The direct representation of tense in the network, renders LOLITA 92 language dependent. It has good non-linearity, which is affected only by the lack of distributedness of predicates on sets, and the status arc. Its determinism of search is relatively good, but is reduced by the need to check for less than all size arcs. LOLITA 92 is not a packed representation.

4.9.7 Conclusion

LOLITA 92 does not satisfy the requirements for LOLITA: Its richness is insufficient. It has low distributedness. It has various breaks of uniqueness and little understanding of the problems normalisation can cause. It has weak cohesion and no understanding that multiple mechanisms are undesirable. It has bad topological distance and poor language independence.

Despite these failings, LOLITA 92 did introduce restricted notions of distributedness and non-linearity, and the idea that concepts' definitions and observational facts could be expressed in the same knowledge-base. It also does not fair too badly in terms of non-linearity, determinism of search, and is not packed. LOLITA 92 was also accompanied by a large-scale implementation to which the author had full access. This was invaluable, since while other researchers were confronted by the problems involved in setting up a new framework, the author could bypass this and gain direct experience of the problems encountered in a large scale system.
Unlike SNEPS, which comes with a tiny grammar, LOLITA had a wide coverage grammar, some semantic analysis, a natural language generator and some reasoning algorithms. The need to build on existing work meant that LOLITA 92 was selected as the starting point for this work.

4.10 Conclusion

4.10.1 Summary

Table 4.1 (p. 113) summarizes the features of the different representations reviewed. Each row scores systems qualitatively by functionality. Not all terms used in the table have been defined so far. The next sections remedy this lack.

4.10.1.1 Primitives

There are two kinds of primitives: meaning primitives, and structural primitives. For an algorithm to reason with information, a limited number of structural primitives is required. These structural primitives direct the inference, for instance stating to which arguments a predicate is applied. The number of such primitives must be limited if a fixed interpreter is to be constructed. However, if the number of meaning primitives is limited, the potential richness of the system is limited. Indeed, meaning primitives limit the actual concepts the representation can express, whereas the structural primitives only limit the classes of concept the representation can express. For further discussion of the different types of primitives see [Brachman 79]

4.10.1.2 Separative and Non-Separative K.B.s

One might think K.B.s which separate statements (separative K.B.s) and those that do not (non-separative K.B.s) could not be compared in terms of distributed-

---

18M-Primitive: Meaning Primitive
19Probably: see 4.4.6 (p. 66)
<table>
<thead>
<tr>
<th>Feature</th>
<th>FOPL</th>
<th>QLF</th>
<th>CD</th>
<th>CGT</th>
<th>SNePS</th>
<th>ANALOG</th>
<th>KLONE</th>
<th>L92</th>
</tr>
</thead>
<tbody>
<tr>
<td>Richness</td>
<td>1/3</td>
<td>2/3</td>
<td>1/3</td>
<td>2/3</td>
<td>2/3</td>
<td>1/3</td>
<td>1/3</td>
<td>2/3</td>
</tr>
<tr>
<td>Cohesion</td>
<td>Weak</td>
<td>Weak</td>
<td>Weak</td>
<td>Weak</td>
<td>Weak</td>
<td>Weak</td>
<td>Weak</td>
<td>Weak</td>
</tr>
<tr>
<td>No M-Primitive Limit(^{18})</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Language Independence</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>NL Naturalness</td>
<td>1/4</td>
<td>3/4</td>
<td>2/4</td>
<td>1/4</td>
<td>2/4</td>
<td>1/4</td>
<td>2/4</td>
<td>2/4</td>
</tr>
<tr>
<td>Define Explicit New Concepts</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Scripts</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Packed</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Uniqueness</td>
<td>0/3</td>
<td>1/3</td>
<td>2/3</td>
<td>1/3</td>
<td>1/3</td>
<td>1/3</td>
<td>1/3</td>
<td>1/3</td>
</tr>
<tr>
<td>Non-linearity</td>
<td>0/3</td>
<td>0/3</td>
<td>0/3</td>
<td>1/3</td>
<td>1/3</td>
<td>0/3</td>
<td>1/3</td>
<td>0/3</td>
</tr>
<tr>
<td>Distributedness</td>
<td>0/3</td>
<td>0/3</td>
<td>1/3</td>
<td>1/3</td>
<td>2/3</td>
<td>2/3</td>
<td>1/3</td>
<td>1/3</td>
</tr>
<tr>
<td>Topological Distance</td>
<td>0/2</td>
<td>0/2</td>
<td>1/2</td>
<td>1/2</td>
<td>1/2</td>
<td>1/2</td>
<td>0/2</td>
<td>1/2</td>
</tr>
<tr>
<td>Determinism of Search</td>
<td>0/2</td>
<td>0/2</td>
<td>1/2</td>
<td>1/2</td>
<td>1/2</td>
<td>1/2</td>
<td>1/2</td>
<td>1/2</td>
</tr>
<tr>
<td>Monotonic</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Non-Separative</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Single Representation</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Used in a K.B.</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Semantic Network</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Ease of Use</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Correctness</td>
<td>2/2</td>
<td>?</td>
<td>?</td>
<td>1/2</td>
<td>1/2</td>
<td>0/2</td>
<td>2/2</td>
<td>0/2</td>
</tr>
</tbody>
</table>

Table 4.1: Summary of Surveyed Representations' features

ness: Since sections in non-separable K.B.s can include pieces of what would be different statements in a separative K.B., there are more possible sections in non-separable sections. However, distributedness measures the proportion of sections that give information which is sound with respect to the full K.B. This proportion is independent from the number of sections unless the representation of the non-separable K.B. introduces cross-dependencies between the atomic symbols of different statements.

In non-separable K.B.s it is more critical to obtain good distributedness – to avoid reading the whole K.B. – but it is also more difficult to obtain good distributedness in non-separable K.B.s. However, this is counterbalanced by the fact topological distance, determinism of search and non-linearity can better be exploited in non-separable K.B.s, since each statement can be entered from more than one point.
4.10.2 Conclusion

None of the representations proved to be simultaneously sufficiently rich, natural, cohesive, unique and efficient for LOLITA. However, the examination of the available alternatives proved a valuable exercise as it illustrated many features of Natural Language, and analysed critically a variety of representational solutions proposed to date. It also provided the following ideas of particular interest to this thesis:

- \textit{FOPC}: Explicit sets are required for NL-expression.

- \textit{QLF}: Representations can be updated monotonically, by adding constraints to the eventual choice of a concept. NL requires a representation of ambiguity.

- \textit{CD}: Scripts can provide plans of usual events.

- \textit{CGT}: Whole sections of network can be inherited to instances: Schemata.

- \textit{SNePS/ANALOG}: Based on ideas similar to 2.2 (p. 11), its formal semantics are circular, but are shown to have a unique meaning as they are “grounded” by the sensory nodes (states of the agents perceptions).

- \textit{KL-ONE}: The position of a concept in the hierarchy can be determined from its definition: Automatic Classification.

- \textit{LOLITA 92}: A single knowledge base can both define and express facts about concepts.

A more detailed examination of the features of the different representations is given in 8 (p. 297), where the representation developed for LOLITA is compared with the existing systems reviewed in this chapter.
Chapter 5

Basic Representation

As no existing representation was found satisfying LOLITA's requirements, this chapter discusses the basic representation developed for LOLITA: SemNet. This concept-forming representation is domain-independent, allowing many classes of concepts and different phenomena to be represented. It forms an independent representational layer which can be built upon, as discussed by [Brachman 79].

A simple representation scheme will be adopted initially, and extended gradually throughout the chapter to achieve a representation that can express the information LOLITA needs.

5.1 Expressing information as a graph
5.1.1 Fundamentals
5.1.1.1 Graphs

Graphs consist of a collection of edges and vertices. Edges are unordered pairs of vertices. Each vertex is associated with a unique identifier allowing it to be distinguished from all others. By associating each vertex with a concept, relations between these concepts can be expressed using edges. For instance, one may wish to express which cities are near each other. This can be done by associating a vertex with each of the cities. Edges can then be used to express proximity between the cities. In such a schema, there will be an edge between Durham and Newcastle signifying that they are close.
Each vertex may appear in more than one edge. For instance, one edge may state that Durham is near to Newcastle, whereas another states that it is far from Inverness. When this occurs, the intended meaning is that all the relations hold: Durham is near to Newcastle AND it is far from Inverness.

### 5.1.1.2 Labelled edges

The example developed so far is quite restrictive: as edges are pairs of vertices there appears to be no way of distinguishing between them. This means that only one type of relation can be expressed by the graph. This limitation is removed by associating each edge with a label, expressing the type of the relation it represents. Thus our example can be extended to state that Birmingham and Bristol are moderately close, whereas Penzance and Inverness are distant.

### 5.1.1.3 Arcs

The relations that have been considered so far are all symmetrical: if Durham is close to Newcastle, Newcastle is also close to Durham. However not all the relations one may wish to express will be. For instance, “Francis ate an apple” is not a symmetrical relation. “Francis” and the “apple” can each be represented as a vertex, and connected by an “eats” edge. However there is no way to know from the graph whether it is Francis who eats the apple, or the apple which eats Francis. This difference in meaning can be represented by using directed edges. These are ordered pairs of vertices\(^1\). Henceforth, directed labelled edges will be referred to as arcs. They are the only form of edges used in SemNet.

At this point a set of definitions and conventions are needed to simplify the rest of the text. Arcs are ordered triples \((l, a, b)\), where \(l\) is the arc’s label, \(a\) is its source, and \(b\) is its target. In the example, the source would correspond to “Francis”, and the target to the “apple”. Where an arc is said to connect \(a\) and \(b\), or is said to be between \(a\) and \(b\), where \(a\) and \(b\) stand for any two vertices, \(a\) is the source vertex and \(b\) the target. An expression of the form “label arc”, such as “subject

\(^1\)an ordered pair is one for which \((a, b) \neq (b, a)\) for \(a \neq b\)
arc" signifies that the arc has the label label. Finally, it should be noted that arcs stem from source vertices. This means that there is no way of knowing of which arcs a vertex is the target, only of which arcs it is the source.

5.1.1.4 Nodes

Just as edges were extended into arcs, it will prove convenient to extend vertices into nodes. In the example, the vertices Francis and apple have been discussed as if these names were their unique identifiers. However, such a scheme would only allow LOLITA to know about one particular Francis and one particular apple. Thus, it is convenient for vertices not only to be associated with a unique identifier – its noderef; but also with a natural language identifier – its label. In the example, both Francis and apple would be node labels. Note that unlike noderefs, each node label can be associated with many nodes.

Although nodes all correspond to concepts, it proves useful to distinguish them into particular types for some algorithms. These types control the behaviour of the algorithms that manipulate them. Thus each type will be called a control, and the set of these controls associated with each node, the controls of that node. Finally, as noted above, each node is associated with the arcs of which it is the source.

5.1.2 Events

Expressing relations as arcs has two disadvantages: arcs can only express relations between pairs of objects, and arcs cannot be referred to in the same way nodes can. Each problem is considered in turn, and then a solution is proposed: events.

5.1.2.1 The need for events

• Arcs can only express relations between pairs of objects

Arcs are pairs of vertices. Thus relations can only be expressed between two concepts. This is natural for transitive relations involving two such concepts: "Fran-
"cis ate an apple" is readily expressed as an eat arc between Francis and apple. However many statements only involve one concept: these are intransitive statements such as "John run". There is no unique way of modeling this information with relations. Two alternatives spring to mind. Either one could use an "action" arc between John and run, or a "run" arc between John and a "no object" node which would then be ignored. The first alternative is asymmetrical from the transitive case as the label does not correspond to the verb. This breaks cohesion, resulting in more cases for the algorithms to distinguish, and rendering them unnecessarily complex. The second alternative is preferable, as the special case that an algorithm has to deal with is reduced to "no object" as target. However this is unsatisfying, as intuitively an arc connects two things, whereas here it is connecting something to nothing in a particular way.

Arcs also prove unnatural when one wishes to express a relation between more than two objects. Consider for instance, the sentence "John and Mary killed Bill". It could be expressed as 2 "kill" arcs, one between John and Bill and the other between Mary and Bill. However this is the way the two statements "John killed Bill" and "Mary killed Bill" would be expressed. Thus the joint responsibility conveyed by the first sentence would have been lost. Another representation must be found to express the distinction. An alternative would be to create a node corresponding to the pair "John and Mary" which would be the source of the kill arc, and have two has_part arcs stemming from it with John and Mary as respective targets. Although this has its advantages, it does require further search from any node to ensure whether it is the real source/target of the arc, or a dummy pair, the parts of which really participate in the relation.

- **Arcs cannot form part of a relation themselves**

Arcs are the only method of relating things in a graph, but they can only connect nodes. Thus the representation developed so far renders it impossible to express information about the relation itself. However this would be something useful to do. Let us consider two examples.

Consider the sentence "John bought the tree in the park". It could be interpreted
in one of two ways: the tree is in the park or the buying occurred in the park. It is easy to see how the first alternative can be expressed: some arc expressing location between the tree and the park. However expressing the location of the buying is not so easy, since the buying is represented as an arc, and there is no way to associate an arc with a node. The possibility left would be to associate the source node, John, with the location of the buying. This corresponds to stating that occurrences, such as people buying things, only happen where the agent is. Although this may seem reasonable, it is not infallible. Consider for instance “Bill in Washington telephoned Helmut in Bonn”. Where does the telephoning occur? Obviously neither in Washington nor in Bonn, but along some path between Washington and Bonn. As this example shows, there may be up to three locations for a relation.

Consider the sentence “Jane said that Tim ate an apple”. It is the occurrence of Tim eating the apple that was reported, not John or the apple. Again, some method of relating a node, Jane, to an arc, eat, seems to be needed.

5.1.2.2 The solution: Events

The solution is to associate a relation with a node rather than with an arc. Two arcs are needed to associate the relation’s node with its arguments: these are the subject_ and object_ arcs. The target of the subject_ arc corresponds to the source node in our previous example, and the target of the object_ arc corresponds to the target node. Two types of arc are needed so that relations preserve the notion of direction explained in section 5.1.1.3 (p. 116). Let us reconsider the example “Francis ate an apple”. Now it will be represented as 3 nodes, Francis, eat, and apple; a subject arc between eat and Francis; and an object arc between eat and apple

Although it is sufficient to use two arcs and a node to express a relation, it proves far more useful to add a third action_ arc to provide the type of the relation.

2Note that the use of subject_ and object_ arcs is independent from the syntactic form the sentence an event can be expressed as in English: both “Francis ate an apple” and “An apple was eaten by Francis” have the same representation.
Previously, the node eat in the "Francis ate an apple" example was associated with the word eat, thus determining the type of the relation. Now, each relation type is expressed explicitly in the net: each is associated with a unique node. This allows information to be expressed about the relation type itself. Thus our example is represented as 4 nodes: Francis, a node representing "Francis"; apple, a node representing an "apple"; eat, a node representing the relation type "to eat"; eating, a node representing the particular occurrence of the "to eat" relation: that of "Francis ate an apple". The eating node is connected to the Francis node by a subject arc, to the apple node by an object arc, and to the eat node by an action arc.

Henceforth nodes representing relations\(^3\) shall be called events. Nodes representing relation types shall be called actions. All other nodes shall be called entities. As these differences correspond to different behaviours of LOLITA's algorithms, each node will be associated with a control specifying to which of these three types it belongs.

For convenience, the following convention will be adopted: the subject of an event e will be the target of the subject arc stemming from the node corresponding to e. The object and action of the event e are similarly defined.

The picture developed so far lacks one final ingredient. Consider the case when one wishes to know all the relations in which Francis figures. As stated previously, arcs stem from the source nodes. Thus there will be a subject arc between the event

\(^3\)note: not relation types
and Francis. However there will not be any arc between Francis and the event. For most applications, finding the relations in which a particular node figures is a necessity. This requirement is fulfilled by adding the arcs subject_of, object_of, and action_of which are the inverse\textsuperscript{4} of the subject_, object_, and action_ arcs respectively.

It might be considered expedient, for practical reasons, to limit the size of the semantic net. An obvious way of doing this is to allow a hybrid representation of relations. When a relation is not described by another, it could be represented as an arc, and as an event otherwise. However this will confuse the discussion of algorithms. Thus all relations shall be considered to be events henceforth. The only arcs discussed in the text are \{subject_, object_, action_, subject_of, object_of, action_of\}. Note however that in pictorial representations, it may be convenient for clarity to represent an event as an arc.

5.1.2.3 Demonstration of Properties: some examples

To illustrate the use of events, the examples demonstrating their need will be reconsidered. One such set of problems was associated with arcs connecting two concepts. The first case was that of an intransitive action, such as "John ran". This can easily be expressed as an event with subject_ John and action_ run. There will be no object_ arc. The second problem was relations between two entities and a third "John and Mary killed Bill". Again the solution is natural: an event with two subject_ arcs to John and Mary, action_ kill and object_ Bill. Events appear to resolve the problems associated with arcs connecting two concepts.

The second set of problems was those related to not being able to refer to a relation explicitly in the representation. Two examples were considered. The first was "John bought the tree in the park", where the event of John buying the tree occurred in the park. Now that there is a node e\textsubscript{1} which corresponds to this event, it can be connected to the node expressing the location "in the park" with the relevant locative event e\textsubscript{2}. This is an example of an event (e\textsubscript{1}) being described or qualified

\textsuperscript{4}the inverse of an arc \((l,s,t)\) is the arc \((l,t,s)\)
by another ($e_2$). In this case, $e_1$ will be the subject of $e_2$. The second example was "Jane said that Tim ate an apple". In this case, the event of Tim eating the apple $e_1$ is the object of the saying event $e_2$. Thus events can either be the subject or the object of another event.

5.1.2.4 The direction of events

- Directional events

So far the notion of direction has been limited to the idea that relations may be asymmetrical. Many relations are asymmetrical, and therefore have a direction. However no convention has been established to determine what direction a relation should be given. For instance, it seems to have been implicitly accepted that "Francis ate an apple" is a relation from Francis (the source/subject node) to the apple (the target/object node). It is worth considering whether any advantage would be gained by adopting such a convention.

If there is no convention to decide the direction of events, then the program will need to process each relation specifically to determine the role played by the subject and object. This lack of cohesion is, in a large scale system, a large undertaking. However one must be aware of the implications of adopting such a convention: It would mean assigning a general meaning to direction, which all directed relations must obey. The question is therefore whether there is such a general meaning implicit in all directed relations which could be expressed by the direction of each event.

Two questions are of relevance to agents, such as LOLITA, which want to reason about the world (as they believe it to be). The first question is what the state of the world is, and the second is what changes to the state of the world occur. In this light, there are two types of events: static and dynamic events.

Stative events describe some entity's state. Examples are "The car is red" or "A sheep is an animal". Thus there are two elements to a stative event: the entity being described, and the state itself. The convention will be adopted that the entity being described will be the subject of the stative event. In some cases the state
will be expressed by an event, such as for "John lives" in the sense "John is alive". In other cases, the state is expressed as a relation between an entity and another entity, such as in "John weighs 80kgs". In this latter case, the other entity will be connected to the event by an object arc to maintain directionality.

Dynamic events describe some change to the world. Usually, such changes are limited to a particular entity. Dynamic events will therefore limit themselves to describing the changes of such an entity. Either this entity is changing by itself, or is being changed by another entity. In both cases one entity performs the change. This entity will be the subject of the dynamic event. In the second case there is also an entity being changed. This will constitute the object of the event. An example of the first case is "John got up", where the subject of the event will be John and the action get.up. An example of the second case is "Danielle cut the log" where Danielle is the subject, cut is the action and log is the object of the event. Although the examples show cases where the subject performs the change because s/he wants to, it should be noted that this is not necessarily the case. Thus John will be the subject of the event corresponding to "John grew".

- Adirectional events

No mention has been made of symmetrical relations. The distinction between subject and object arcs was needed to provide a direction to the event. However no direction is needed for symmetrical relations, so they will only require the use of one of these two types of arcs. Further insight can be provided by considering that symmetrical relations are stative where both entities considered are being described: not only is Newcastle near Durham, but also Durham is near Newcastle. Thus both members of the relation are being described and should be assigned the subject arc according to the rules above.

When an event must be connected to more than one subject arc or more than one object arc to express its meaning, each of the subject or object arcs respectively are called partial. Events involving partial arcs cannot be fully understood before all their partial arcs are read: "Durham is near" cannot be understood except in relation to another location. This means that the granularity of dis-
tributedness does not extend beyond full arcs to partial arcs.

- Cohesive Directionality is easier to use

Let us consider how these rules help in practice. The first problem in a large scale natural language system is obviously to transform natural language into the internal representation and vice-versa. For active sentences, the subjects and objects of the sentence tend to follow the classification established for the subject.s and object.s of events. For passive sentences, subjects correspond to object.s and vice-versa. This eases the transformation to and from natural language. This similarity is also useful in that it makes it easier for a wide range of developers to determine the correct internal representation. However the raison d'être of the internal representation is to allow processing of the information expressed in natural language text. An example of such processing is reasoning about all the things John did in order to determine whether his actions indicate he was involved in some crime, or was an innocent bystander: the things John did are represented by events which have John as subject. Similarly all the things done to John may play an important part in determining his motivation: these correspond to events of which John is the object.
### 5.2 Grammar

This section presents the textual notation of the network that will be used henceforth.

\[
\text{event} = \text{event:} \{ \text{Arc.clearations}_{\text{opt}} \} \\
\text{event} = (E_{\text{NUM}}, \text{event.status}) \\
\text{event.status} = \text{R} | \text{H} \\
\text{Arc.declarations} = \text{arc.declaration} | \text{arc.declaration Arc.declarations}_{\text{opt}} \\
\text{arc.declaration} = \\
\{ \text{arc.status Quantifications - arc.type - Quantifications arc.status : target} \} \\
\text{arc.status} = \text{O} | \text{O} \\
\text{Quantifications} = \\
\text{quantification Quantifications}_{\text{opt}} | \text{Quantifications Quantifications}_{\text{opt}} \\
\text{quantification} = \text{I} | \text{V} | \text{E} | \text{F} \\
\text{arc.type} = \text{subject.} | \text{action.} | \text{object.} | \text{subject.of} | \text{action.of} | \text{object.of} \\
\text{target} = \text{Node} | [\text{Nodes}] \\
\text{Nodes} = \text{Node} | \text{Node}, \text{Nodes}
\]

The meaning of the atomic symbols is detailed in table 5.1 (p. 125).

The terminal \text{Node} is a unique string\(^8\) of any characters other than ‘’, ‘]’ or ‘}’, such as \text{farmer} or \(E_{\text{NUM}}\).

---

\(^{5}\)see 7.2.1.3 (p. 258)  
\(^{6}\)see 5.4 (p. 135)  
\(^{7}\)see 5.3.2 (p. 127)  
\(^{8}\)For readability, the examples in this thesis will not each be assigned a unique string.
From the grammar, it is clear that each event node (event) has a unique number \( NUM \) and a belief status. Each event may take zero or more arcs (Arc_declarations).

Each arc (arc_declaration) has a definitional or observational status (arc_status) with respect to its source (event) and its target (target). Similarly quantification (Quantifications) is expressed with respect to source and target. Finally, each arc has a type (arc_type). An example would be “John hates Jane”:

\[ E_6: \{ \Delta I\text{-subject}\_IO: John; OI\text{-action}\_IO: hate; \Delta I\text{-object}\_IO: Jane \} \]

## 5.3 Introducing sets

So far all the examples developed consist of individual entities such as “an apple” or “Francis”. However groups of entities, such as apples, or men must also be represented. These groups can be represented as sets. This section will discuss how sets can be expressed in the semantic net, and what the resulting benefits are.

### 5.3.1 Representation of sets

Informally, a set is a collection of distinct objects. The distinct objects in the set are called members, elements or instances of the set. Sets are objects or concepts in their own right. It is therefore appropriate to associate them with nodes, just as was done for “Francis”. Thus each set is associated with a label, and a unique identifier. This allows many distinct sets of apples to be associated with the label “apple” but to be distinguished by their noderefs.\(^9\)

#### 5.3.1.1 Elements of a set

Simply representing a set as a node is not terribly informative. It would be useful to be able to state that an object \( o \) is an element of a set \( S \). For instance the statement “This apple is a fruit” states that a particular apple is an element of the

\(^9\)Mathematically, two sets are the same (or equal) if they have the same elements. SemNet does not actually have sets, but concepts, as will be discussed in 5.4 (p. 135). However, the discussion is simplified if concepts are temporarily approximated to sets.
set of fruits. Since this is a relation between two objects, an event will be used to represent it. The chosen action is "inst." signifying that the subject node has as instance the object node. Thus the statement will be represented as an event with the set of "fruits" as the subject "inst." as action and the "apple" as object. It will be convenient to use the standard mathematical notation \( o \in S \) to represent the fact the element \( o \) is an instance of the set \( S \).

There is no obligation for any element of a set to be specified. This is important as often these elements will not be known. For instance one may know that one's neighbour has a lot of apples in his cellar without knowing each of them, or even knowing how many of them there are.

5.3.1.2 Subsets and Supersets

It is also useful to represent relations between sets, such as "all apples are fruits". This example corresponds to the set of apples \( A \) being the subset of the set of fruits \( B \). This means that every element of \( A \) is an element of \( B \), but not vice-versa. This is written \( A \subset B \). It should be noted that it is possible for sets \( A \) and \( B \) to be the same. This is represented as an event with subject "B", action "spec.", and object "A". \( B \) is the superset of \( A \).

If a node \( o \) is the object of an inst. or spec. event with subject \( s \), then \( s \) is \( o \)'s parent, and \( o \) is \( s \)'s child. Similarly if a node is the parent of a node, or the parent's parent, or the parent's parent's parent..., it is an ancestor of that node, and that node is its descendent.

5.3.2 Quantification

The representation of quantification in SemNet is reached through a richness argument which presents a series of quantification schemes, each of which is shown to be insufficiently expressive to represent Natural Language examples. The failures of each are analysed, and corrected, leading to the final representation, which this section summarizes. The full argument is given in B.1 (p. B-1), which also justifies
the claims of this section.

5.3.2.1 Structural Description

• Types of Quantification

  o Basic Types

  • **Individual** Quantification (written \( I \)) binds the concept itself into the relation: "John ate an apple"

    \[ (E_0,R): \{ \Delta I\text{-subject.-}IO: \text{John}; \Delta I\text{-action.-}IO: \text{eat}; \\
    \Delta I\text{-object.-}I\Delta: \text{apple} \} \]

    The concept can also be a set, in which case individual quantification applies to the set, rather than to its elements: "Jack ate five apples"

    \[ (E_1,R): \{ \Delta \forall\text{-subject.-}IO: \text{Jack}; \Delta \forall\text{-action.-}IO: \text{eat}; \\
    \Delta F\text{-object.-}F\Delta: \text{apples}, \} \]

    \[ (E_2,R): \{ \Delta I\text{-subject.-}IO: \text{apples}_1; \Delta I\text{-action.-}IO: \text{size.}; \\
    \Delta I\text{-object.-}IO: 5 \} \]

    In this manner, statements can also be made about groups: "The team won the game" (where \text{Team} is a set with as elements the people forming the team).

    \[ (E_3,R): \{ \Delta I\text{-subject.-}IO: \text{Team}; \Delta I\text{-action.-}IO: \text{win}; \\
    \Delta I\text{-object.-}IO: \text{game} \} \]

  • **Universal** Quantification (written \( \forall \)) binds all the elements of the concept into the relation: "Every person has a nose" (\text{has\_part} is the action of every event in the set of events \( E_4 \))

    \[ (E_4,R): \{ \Delta F\text{-subject.-}FO: \text{Person}; \Delta \forall\text{-action.-}IO: \text{has\_part}; \\
    \Delta F\text{-object.-}F\Delta: \text{Nose} \} \]

  • **Existential** Quantification (written \( \exists! \)) chooses an element of the concept depending on the choice of element of the other concept participating in the relation: "Every student has one supervisor."

    \[ (E_5,R): \{ \Delta \forall\text{-subject.-}\exists!\Delta: \text{Student\_Supervisors}; \\
    \Delta \forall\text{-action.-}IO: \text{supervise}; \Delta F\text{-object.-}F\Delta: \text{Students} \} \]

    Because \( \exists! \) chooses only one element of \text{Students\_Supervisors} for each element of the counterpart concept \( E_5 \) chosen by \( \forall \), \( E_5 \) may have more elements than \text{Student\_Supervisors}. 
Existential Quantification does not correspond to its namesake in $\mathcal{FOL}$ in that it always depends on the relation's other quantification which must contain a corresponding $\forall$. Further, unlike a variable qualified by $\exists!$ in $\mathcal{FOL}$ which is an element of a given set, the node referred to by $\exists!$ is a set, which may even be empty.

- **Shorthands**

  - **Framed Universal** Quantification (written $F$) states that the relation is a bijection. Framed universals only occur in pairs on either end of a relation as they are a shorthand for two identical relations, one with $\forall - \exists!$ quantification and the other with $\exists! - \forall$: "Every car has a steering wheel"\textsuperscript{10}

    $$(E_6,R): \{ \Delta F\text{-}subject\text{-}IFO: \text{Cars}; \Delta \forall\text{-}action\text{-}IO: \text{has part};$$

    $$\Delta F\text{-}object\text{-}F\Delta: \text{Steering. Wheels}_1 \}$$

    For every element of $\text{Cars}$ there is an event of $E_6$ and for every element of $E_6$ there is an element of $\text{Steering. Wheels}_1$, so there are as many steering wheels as cars. Because Framed Universal quantification occurs often, the alternative form is normalized out of the network:

    $$(E_7,R): \{ \Delta \forall\text{-}subject\text{-}\exists!O: \text{Cars}; \Delta \exists!\text{-}subject\text{-}\forallO: \text{Cars};$$

    $$\Delta \forall\text{-}action\text{-}IO: \text{has part};$$

    $$\Delta \forall\text{-}object\text{-}\exists!\Delta: \text{Steering. Wheels}_1;$$

    $$\Delta \exists!\text{-}object\text{-}\forall\Delta: \text{Steering. Wheels}_1 \}$$

  - **Arbitrary** Quantification (written $A$) binds to an arbitrary element of the concept. Each relation connected to a concept by an arbitrary quantification chooses its own arbitrary concept. Thus, $A$ is a shorthand to avoid building instances of concepts connected to only one event\textsuperscript{11}. For instance, "A dog howled": $E_8$ instead of $E_9$ and $E_{10}$

    $$(E_8,R): \{ \Delta I\text{-}subject\text{-}A\Delta: \text{Dogs}; \Delta I\text{-}action\text{-}IO: \text{howl} \}$$

    $$(E_9,R): \{ \Delta I\text{-}subject\text{-}I\Delta: \text{Dogs}; \Delta I\text{-}action\text{-}IO: \text{inst};$$

    $$\Delta I\text{-}object\text{-}I\Delta: \text{Dog}_1 \}$$

    $$(E_{10},R): \{ \Delta I\text{-}subject\text{-}I\Delta: \text{Dog}_1; \Delta I\text{-}action\text{-}IO: \text{howl} \}$$

\textsuperscript{10}Proof in B.2 (p. B-17).

\textsuperscript{11}Other than the inst...
• Quantification at both ends of each arc

Quantification describes the way in which a relation binds two concepts, so is intrinsic to relation, and not to the concepts themselves. Thus quantification is placed on the arcs. The relation may apply differently to the two concepts that it binds, so two quantifications are involved: one with respect to each concept. Thus, quantification is expressed on each end of each arc. For instance, “Every day, Sengan waters a different Apple Tree”:

\((E_{11}, R): \{ \Delta F\text{-subject.} - F\Delta: E_{12}; \Delta V\text{-action.} - IO: at\text{.} time; \\
\Delta F\text{-object.} - FO: Days \}\)

\((E_{12}, R): \{ \Delta V\text{-subject.} - IO: Sengan; \Delta V\text{-action.} - IO: water; \\
\Delta F\text{-object.} - F\Delta: Apple\_Trees \}\)

Every event of the set \(E_{12}\) has as subject. Sengan, as action. water, but chooses a different object.: a different apple tree. Each event of \(E_{11}\) has as action. at.time, but has as subject. an event of \(E_{11}\) and as object. the corresponding day.

• Many Levelled Quantification

SemNet’s concepts may correspond to sets of sets: For instance, the First Division is a set of football teams. Each team is a set of players, so the First Division is a set of sets. To allow statements to be made about such concepts, quantification is many levelled in SemNet: a relation can not only refer to sets, to set elements, but also to elements of sets of sets, to elements of sets of sets of sets, and beyond.

This is achieved by extending the quantification to a string of quantification symbols \(s_1 s_2 \cdots s_i\), where the \(j\)th symbol \(s_j\) refers to the \(j\)th level of elements of the set referred to. The first such level of a set \(S\) is its elements, the second is the elements of its elements and so on. Because the choice of each symbol is free between \(\forall, \exists, I\), \(F\) and the shorthands \(F\) and \(A\), complex relations can be built. Each end of each arc is associated with the string of symbols expressing the way in which the arc relates to the node to which it is connected at that end.

Although in theory an infinite number of quantification levels can be built, in practice only three appear to be needed to express most statements in natural language,
discounting artificially contrived examples\textsuperscript{12}. For instance "Every shepherd owns some sheep":

\[(E_{13}, R): \{ \Delta FV\text{-}subject\text{.}_FO: \text{Shepherds}; \Delta \forall V\text{-}action\text{-}IO: \text{owns}; \Delta FF\text{-}object\text{.}_FF\Delta: \text{Sheep\_Herds} \}\]

To every element \(s\) of \text{Shepherd} corresponds a set of elements \(e\) of \(E_{13}\), such that \(s\) is the \text{subject\text{.}} of every element of the set \(e\). Every element of every set of \(E_{13}\) has as \text{action\text{.}} \text{owns}. And to every element \(e\) of \(E_{13}\) corresponds an element \(h\) of \text{Sheep\_Herds}, such that every element \(e'\) of \(e\) is the \text{object\text{.}} of a corresponding element \(h'\) of \(h\).

The \(n\)th quantification level of a many levelled quantification \(Q\), corresponds to the \(n\)th symbol of the string \(Q\), after removing any prefixed \(I\)'s. The zeroth quantification of any many levelled quantification is \(I\). The depth of a many levelled quantification \(Q\) is the number of quantification symbols in the string \(Q\), after removing any preceding \(I\) quantifications. Thus \(IV\exists!F\) and \(FFF\) are of depth 3.

\subsection{5.3.2.2 Use}

\textbullet \ Common Element Rule

This rule states that when a set quantification chooses a particular element of a set, all the other set quantifications off that node chose that element too. A set quantification is a universal or existential quantification\textsuperscript{13}, in other words a quantification that refers to elements of the set represented by the node to which it refers. "The set quantifications off a node" means "All the quantifications that refer to that node, that is to say all the quantifications on an arc connected to the node on the node's end".

Thus, in "Every shepherd owns some sheep", when a given shepherd is chosen, the chosen element of \(E_{13}\) is the set of owning events between him and his sheep, and the chosen element of \text{Sheep\_Herds} is that shepherd's sheep. This allows statements such as "The number of sheep every shepherd owns is given by his age":

\textsuperscript{12}At most five seem needed for the most complex of template events
\textsuperscript{13}and hence also Framed Universal
(E_{13},R):  \{ \Delta F\forall\text{-subject.-FO}: \text{Shepherd}; \Delta \forall\text{-action.-IO}: \text{owns}; \\
\Delta FF\text{-object.-FF}\Delta: \text{Sheep.Herds} \}

(E_{14},R):  \{ \Delta F\text{-subject.-FO}: \text{Shepherd}; \Delta \forall\text{-action.-IO}: \text{has.age}; \\
\Delta \forall\text{-object.-}\exists!\Delta: \text{Shepherd.Age} \}

(E_{15},R):  \{ \Delta F\text{-subject.-FO}: \text{Sheep.Herds}; \Delta \forall\text{-action.-IO}: \text{size.}; \\
\Delta \forall\text{-object.-}\exists!\Delta: \text{Shepherd.Age} \}

Statements that refer to two different elements of the same concept c must refer to a second node, defined to be an individual synonym\textsubscript{r} of c (7.6.2.1 (p. 291)). The individual quantification implies no quantificational dependency between c's two nodes. In order to guarantee that c's two nodes will always refer to different instantiations, a size-zero (7.3.2 (p. 267)) synonym\textsubscript{r} event is added expressing the quantificational dependency required for the two concepts to refer to the same instantiation. For instance: "Everybody likes someone else"

(E_{16},R):  \{ \Delta F\text{-subject.-FO}: \text{People}; \Delta \forall\text{-action.-IO}: \text{like}; \\
\Delta \exists!\text{-object.-}\forall\text{IO}: \text{People'} \}

(E_{17},R):  \{ \Delta I\text{-subject.-IO}: \text{People}; \Delta I\text{-action.-IO}: \text{synonym\textsubscript{r}}; \\
\Delta I\text{-object.-}\exists!\Delta: \text{People'} \}

(E_{18},R):  \{ \Delta F\text{-subject.-FO}: \text{People}; \Delta \forall\text{-action.-IO}: \text{synonym\textsubscript{r}}; \\
\Delta F\text{-object.-FO}: \text{People'} \}

(E_{19},R):  \{ \Delta I\text{-subject.-IO}: E_{18}; \Delta I\text{-action.-IO}: \text{size.}; \\
\Delta I\text{-object.-IO}: 0 \}

6.1.2 (p. 158) discusses reasoning with the common element rule, and shows that the common element rule does not break non-linearity.

- Quantification Dependencies

Many Levelled Quantification introduces ambiguity into the dependencies between \forall and \exists!\textsuperscript{14}: in the arc (\forall\exists!\text{-subject.-}\forall\forall), on which \forall(s) does the \exists! depend? Two rules establish the correct dependency:

1. Quantification dependencies do not occur within a many levelled quantification.

\textsuperscript{14}And hence also for Framed Universals
2. If the dependency is ambiguous: The default reading is that quantification dependencies occur between the same level of the two quantifications. If another reading is meant, it must be expressed explicitly using arrows in figures or boxes in text grouping the levels involved in the dependencies.

Thus, in \((\forall \exists \cdot \text{subject}. \cdot \forall \forall)\), the \(\exists!\) depends on the underlined \(\forall\). Other examples include \(\forall \forall \leftarrow \exists!\), \(\forall \exists \leftarrow \exists!\), and \(\forall \forall \leftarrow \exists!\) where the \(\exists!\) depends on the boxed \(\forall\).

- **Partial Arcs**

Events can take as subject (or object) more than one element. Such arcs are called partial arcs. They may occur explicitly, as in “1 + 1 + 1 = 3”:

\[(E_{20}, R): \{ \Delta I\cdot \text{subject}. \cdot IO: [1, 1, 1]; \Delta I\cdot \text{action}. \cdot IO: \oplus; \Delta I\cdot \text{object}. \cdot I\Delta: 3 \}\]

But they may also occur implicitly as in “The shareholders own the company”

\[(E_{21}, R): \{ \Delta I\cdot \text{subject}. \cdot \forall O: \text{shareholders}_1; \Delta I\cdot \text{action}. \cdot IO: \text{own}; \Delta I\cdot \text{object}. \cdot I \Delta: \text{company}_1 \}\]

Note that the arcs of \(E_7\) are not partial as they refer to the same elements.

- **Intensionality and Quantification**

Unlike Set Theory, SemNet does not consider equality to be extensional\(^\text{15}\). That is to say, that a set is not equivalent to its elements. This allows the dispossessed to be defined as the set of people who own nothing:

\[(E_{22}, R): \{ \Delta F\cdot \text{subject}. \cdot F \Delta: \text{Dispossessed}; \Delta \forall \cdot \text{action}. \cdot IO: \text{own}; \Delta F\cdot \text{object}. \cdot F \forall \Delta: \text{Dispossessed\_Possessions} \}\]

\[(E_{23}, R): \{ \Delta F\cdot \text{subject}. \cdot F \Delta: \text{Dispossessed\_Possessions}; \Delta \forall \cdot \text{action}. \cdot IO: size.; \Delta \forall \cdot \text{object}. \cdot IO: 0 \}\]

In set theory, there is only one empty set, so all elements of \(\text{Dispossessed\_Possessions}\) would be the same empty set. This implies that there is only one element of \(\text{Dispossessed\_Possessions}\), hence only one element in \(E_0\), and hence only one \(\text{Dispossessed}\). Furthermore, it assumes that a set can have \(\emptyset\) as a element.

\(^{15}\text{See 5.4 (p. 135).}\)
From the intensional viewpoint, all of John's possessions is a different concept to all of Jack's, even if both turn out to own nothing. Rather than being considered an intrinsic feature of sets, cardinality is considered incidental: a property like any other. Because equality is intensional, two concepts are only equal in SemNet if they have the same definition. I.e. SemNet does not have the rule that any two empty sets are equivalent: a concept can be a set of many 'empty groups', where each such 'empty group' has a type given by its definition (in particular, the type of the arguments of the events defining it and their quantification). In practice this means that $E_{22}$ and $E_{23}$ do not imply that there is only one dispossessed.

5.3.3 Cardinality and Antonyms

Two important events to the manipulation of sets are size and antonym. The first states that the cardinality of its subject set is given by its object:

$$(E_0,R): \{ \Delta I\text{-subject}_{-}IO: A; \Delta I\text{-action}_{-}IO: size; \Delta I\text{-object}_{-}IO: B \}$$

As shown by the $I$-subject $I$ quantification, size takes a set as subject and not its elements. However size can participate in relations on elements of sets, if these elements are sets:

$$(E_1,R): \{ \Delta F\text{-subject}_{-}FO: A; \Delta V\text{-action}_{-}IO: size; \Delta V\text{-object}_{-}\exists!\Delta: B \}$$

$B$ is the set of sizes corresponding to the sets of $A$.

antonym is used to partition sets into subsets. The subsets are disjoint but their union forms the set. The subsets are the subject s of the antonym event, and the set is the object. Again, the sets are referred to by individual quantification. Although logically unnecessary, the subsets are also connected to the set by specific events. This simplifies processing for algorithms which use the subset relation, making the representation more cohesive. antonym is used for instance to state that mammals can be divided into two mutually exclusive types, males and females. Similarly, the chemical elements form mutually exclusive types, but all together form the set of known chemical elements.

$$(E_2,R): \{ \Delta I\text{-subject}_{-}I\Delta: [\text{male,female}]; \Delta I\text{-action}_{-}IO: antonym_; \Delta I\text{-object}_{-}IO: \text{animals} \}$$
5.3.4 Inheritance

Once the set relations have been defined, a hierarchy can be built using them. The features and advantages of this are discussed in 2.7 (p. 16). All concepts, other than the \texttt{inst.} and \texttt{spec.} relations themselves are members of this hierarchy since it helps organize knowledge. If an event is known to apply to all the elements of a set, one knows that it applies also to any subset of these elements. Inheritance is the process which applies this rule to the knowledge base, thereby allowing concepts only to be expressed at the relevant level of granularity. More details of the inheritance algorithm are given in 6.4 (p. 164).

5.4 Defining concepts

As seen above the \texttt{spec.} and \texttt{inst.} events can be used to create a hierarchy of concepts. But so far no difference has been made between those events which restrict the sense-domain of a concept, and those that express facts about all objects fitting this restriction.

Consider the example of a telephone: telephones could be defined to be devices with which people can communicate vocally. Such a definition sets the bound between those things which can be considered to be a telephone and those which cannot. One might wish to state that telephones are made of plastic and metal. Encountering a telephone made of wood however would not require a revision of the definition of telephones, and would not lead it to not being considered a telephone. However if something was encountered which did not allow people to communicate vocally but allowed them to boil tea, it would not considered to be a telephone: it violates the telephone’s definition. If one were convinced it was a telephone, one would need to revise one’s definition.

There is thus a difference between those statements which restrict the instances of the concept, and those that express facts about the instances of the concept once it has been defined. This difference must be expressed in the representation for the
required difference in behaviour to be obtained.\textsuperscript{16}

5.4.1 Sorts
5.4.1.1 Intensional Representation

SemNet’s representation is intensional. That is to say that every concept is defined directly by its defining properties. Naive set theory is extensional. That is to say that a set is defined only by its elements. It can however be defined indirectly by some defining properties:

\[ X := \{ x : P(x) \} \]

Here the set \( X \) is defined as all elements \( x \) that satisfy the property \( P \). However the critical difference is that the set is defined by the elements it contains. For instance, in set theory, two sets are equal if they contain the same elements. Thus, the set’s defining properties \( P(x) \) build the elements which define the set, but being not involved in equality, they are not the set’s definition themselves.

In contrast to this, in SemNet, a concept is defined by its defining properties. Two concepts are only equal if they have the same defining properties. In set theory, sets are defined by their elements. This means that their elements are assumed to be known – for instance for equality tests. However, in SemNet, it is the other way round: anything which satisfies the defining properties of the concept is an instantiation of that concept, but no element that actually satisfies the defining properties of the concept need be known. A concept thus corresponds to a region of meaning-space (2.4 (p. 19)), in which there may or may not be instantiations. A set on the other hand corresponds to the instances it contains.

A concept can either be full or partial with respect to a relation\textsuperscript{17}. A concept is \textbf{full} with respect to a relation \( r \) if it must include all the instantiations of the concept defined by \( r \) alone. A concept is \textbf{partial} with respect to a relation \( r \), if it need not include all the instantiations of the concept defined by \( r \) alone. This difference is critical, as it allows statements to be made about the concepts that one defines.

\textsuperscript{16}Woods also discusses this difference [Woods 75].
\textsuperscript{17}or set of relations
All the properties constituting the definition are full with respect to the concept they define: every instantiation which has these properties is an instantiation of the concept. All the other properties of the concept, which do not define it, but are statements about it, or use it once it has been defined, are partial with respect to it: instantiations of other concepts may have these properties.

The representation must allow these two kinds of statements to be distinguished.

5.4.1.2 Introducing Sorts

The difference between relations that are partial with respect to a concept, and those which are full with respect to it is represented by two sorts on each arc: a sort may either be observational if the arc is partial with respect to the concept to which it is connected, or definitional if it is full with respect to the concept. Just as for quantification, a richness argument (B.4.1 (p. B-23)) shows that an arc may independently be partial or full with respect to its target, or with respect to its source. Thus, each arc has a sort at each end. Definitional sorts are written $\Delta$ and observational sorts $O$ on arcs:

$$(E_0,R): \{ OV\text{-}subject.\exists!\Delta: Hairy\_things; \Delta\forall\text{-}action.\neg O: is\_hairy \}$$

$E_0$ is defined the set of all events with action. is_hairy. It has as subject. the concept it defines: Hairy\_things. $E_0$ is not defined by its subject. arc, as its action. arc is sufficient to restrict it to all is_hairy events.

5.4.2 Interaction of Sorts and Previous features

Since sorts change the way in which relations are understood, they could be thought to interact with the previously defined notions of quantification, and set relations. This section investigates these possibilities.

5.4.2.1 Quantification

Definitional sorts may be used in conjunction with existential quantification dependency. In this case, the meaning is as before that every instantiation of the
concept satisfies the relation including its quantification. For instance, this means that if an event is expressed as:

\[(E_0,R): \{\Delta \forall -subject.-\exists!O: X; \Delta \forall -action.-\forall O: Y; \Delta \forall -object.-\exists!O: Z\}\]

then any instantiation of \(E_0\) must satisfy the three relations: \(\{\forall -subject.-\exists!: X\}\), \(\{\forall -action.-\forall I: Y\}\), and \(\{\forall -object.-\exists!: Z\}\). Thus, if \(e\) is an instantiation of \(E_0\), there must be an \(x\) of \(X\) and a \(z\) of \(Z\), such that \(e\) has as \(subject\_x\), as \(action\_Y\), and as \(object\_z\).

Observational sorts might be thought to contradict quantification for existential dependencies. Indeed, observational sorts have been said to mean that a concept is partial with respect to a particular relation. How can a concept be partial with respect to a relation, if for that relation to be true, the concept must satisfy certain conditions, such as having enough elements for it to be quantified \(\forall\) in a \(\forall - \exists!\) relation? This problem holds even more acutely for \(F - F\) quantification, where the concept must be full! The solution lies in the notion that concepts may be full with respect to their observational relations, but are not necessarily so. What is certain is that concepts are full with respect to their definitional relations. If an observational relation \(r\) involves a quantification which implies that the concepts it involves is full with respect to it, it means that the relations that define the concept restrict it to the full concept \(r\) would define.

For completeness, note that a relation \(r\) can be definitional with respect to a concept \(c\) it quantifies arbitrarily over \((A): it is equivalent to a concept \(i\) which is an instance of \(c\) but which is connected to no other relation but \(r\).

5.4.2.2 Inheritance and the conceptual hierarchy

Since every event is qualified by sorts, inheritance must build the events it infers with sorts. Since sorts change the meaning of relations, the events built must be qualified the right sorts.

- **Meaning of inst. and spec. definitional wrt their subject.**

The notation allows all events to be definitional with respect to their subject.
This includes the spec. and inst. events:

\[(E_0,R): \{\Delta I\text{-subject.-}I\Delta: S; \Delta I\text{-action.-}IO: \text{spec.}; \Delta I\text{-object.-}IO: O\}\]

This states that \(S\) is required to include \(O\). This requirement does not influence the choice of elements other than \(O\), so all other elements are included in \(S\). This means that if \(S\) is not restricted in any other way, it is the maximal set to include \(O\): **typeless**. If it is otherwise restricted, only these restrictions apply, so \(S\) is a copy of \(S'\) which has all of the restrictions on \(S\) other than \(E_0\). Either way, uniqueness is broken as \(S\) refers either to **typeless** or to \(S'\): two nodes may not refer to the same concept.

spec. and inst. events are therefore not allowed to be definitional with respect to their subject.. The only exception to this rule could be for subject. **typeless**, but in practice it is simpler and more robust to recognise the **typeless** node itself.

* Meaning of inst. and spec. definitional wrt their object.*

As mentioned in section 2.4 (*p. 13*), the universe of all concepts (and instantiations of those concepts) is expressed as the top node **typeless**. This means that all concepts must be in the set hierarchy. Since it is impossible for a set not to be in the set hierarchy, the statement that a concept is in the hierarchy cannot in itself be considered restrictive.

The place of a concept within the set hierarchy is restrictive. This place corresponds to the restrictions on the superset of the set’s scope, which is determined by the definitional events this set has or inherits. Thus if a spec. or an inst. event is definitional with respect to its object. it means that the events defining its subject. should be inherited as definitional by its object. Similarly if the spec. or inst. event is observational with respect to its object., it means that all the events defining its subject. should be inherited as observational by its object. In both cases, observational events are inherited to the spec. or inst. event’s object. as observational.

* Implications for the set hierarchy*

Concepts which are only defined by one spec. or inst. event are illegal since they
break uniqueness. Indeed, if a spec. or inst. event is definitional with respect to its object. o, inheritance means that its object. o is restricted by the events that define its subject. s: o does not differ from s, unless it is further restricted by another definitional event. In this respect, the treatment of spec. and inst. events differs from that of all other events, since any other event can restrict a concept uniquely. Notice that if a concept is the definitional specialization (or instance) of more than one concept, it can be unique since the combination of definitional properties it inherits from its parents can be unique.

**Rule 5.4.2.1:** "All concepts other than typeless must be the definitional object. of at least one spec. or inst. event".

All concepts are restrictions of typeless, since typeless is that which is not restricted. However, since typeless is not restricted, there may not appear to be any benefit in connecting concepts to it by a definitional spec. or inst.: there are no definitional events to inherit from typeless. The problem is that this makes it particularly difficult to find the most generic concepts. These are the concepts which would have as only parent typeless, and can now only be found by examining the whole semantic network. To avoid this expensive search, the above rule is adopted.

inst. and spec. events with observational object. targets prove useful in compressing and organizing concepts' observational characteristics.

### 5.4.2.3 Sorts and Cardinality

- **Constants, a notational variant**

The representation allows a definitional event to be attached to an instance. Does this make a sensible statement? Suppose that the definitional events restrict the concept they define to a unique element. For this singleton concept, the requirement that definitional events restrict the domain of the concept to all those elements which satisfy them will be fulfilled. Writing this singleton as a set concept A, with observed size. 1, and stating all the observed events about it with universal quantification, or writing it as an instance i defined by definitional events therefore
expresses the same information. The question is reduced to one of convenience. Choosing the latter alternative has the advantage of reducing the size of the net. However rare, sets of size. 1 with universally quantified events attached to them will occur: for instance “a one man team” requires the notion of a set. Sometimes in this thesis it will prove simpler to discuss instances in terms of size 1 sets, as it enables one to consider the subset relation independently from the cardinality, both intrinsic to inst. Since i is uniquely defined, it corresponds to a constant.

- **Cardinality, Sorts and Extension**

A concept’s cardinality may be defined or observed.

If a concept is connected to a definitional size. which qualifies the concept’s elements, any of its instantiations must have that size.: each of its elements is a set concept, with the specified number of elements. However, there is no need for the concept to have any instantiations, unless explicitly written. Notice that if the definitional size. relation has I quantification with respect to the concept, the concept c is equivalent to a singleton concept s with observed size. 1, to which all the relations of c are connected, with the proviso that their quantification is suitably adjusted with respect to s. Thus such concepts (c) correspond to the instantiation of an implicit concept (s).

If a concept is connected to an observational size. which qualifies the concept’s elements, there are no concepts satisfying its definitional properties that have other size.s: The concept includes all instantiations fitting its definitional properties, and every one of them has the stated size. Again, the concept need not have any instantiations. However, if the definitional size. has I quantification with respect to the concept, the concept has the specified number of instantiations. In other words, size. introduces the concept’s extension: the number of instantiations of a concept an agent believes to exist corresponds to the number he believes he could perceive in the “external world” he conceives from his perception.
5.4.3 Using sorts

Only examples of their use can give a real understanding of sorts. Entities will be discussed first, as they are simpler to understand, being only subject to events. Events will then be considered, as they are not only subject to their own subject-, action-, and object- arcs, but also to other events.

5.4.3.1 Sorts and entities

- Defining entities

Entity concepts in the inheritance hierarchy correspond to all instantiations with certain properties. This corresponds to precisely the situation in which events which are definitional with respect to a node should be used.

Take as example the concept of cars. Suppose that an essential feature of cars is to have wheels, and no car can lack wheels. This is represented as a “have_part” event $E_0$ which has as subject_ the concept of all cars, and as object_ some subset of wheels. The event is definitional with respect to the cars, so there will be a definitional sort associated with the subject_ arc from “cars” to $e$. The next question is whether the event should be definitional with respect to its object_. Since the subset of wheels concerned must be defined (Rule 5.4.2.1 (p. 140)), and the only distinguishing feature is that the set refers to the wheels that are part of cars, the answer is yes.

As the previous example shows, features defining a set need not be intrinsic properties of their elements, but may be extrinsic and refer to their current situation for instance. This means that an object may be an element of a particular set at one time, and not at another. See 7.5.2.7 (p. 283) for further discussion.

For convenience, an event $E_1$ will be said to define a concept $c$ if the subject_of or object_of arc connecting the concept $c$ to it, $E_1$, has a definitional sort.

- Observed features

Once a concept has been defined, events which are observational with respect to it are used to express its features. To illustrate, consider the phrases “Three of the
**Figure 5.2: Observed features**

*stock brokers who live in New York ...* (Y), and *"The three stock brokers who live in New York ..."* (X). The first states that there are more than three stock-brokers in New York, whereas the second states that there are only three. This difference is expressed by definitional and observational sorts (see figure 5.2 (*p. 143*)).

- Aside: Under-defined concepts

In many cases, concepts appear insufficiently defined in order for them to be defined uniquely. For instance, *"a dog sneezed"* does not appear to refer to any particular dog. Similarly arbitrary sets seem possible: *"some people laughed"*. However, these seemingly undefined concepts are defined by the statements they are involved in. The dog is defined at the very least by the fact it barked. But not only is it known that it barked, the time it barked is known within a certain accuracy, as is the area: the dog that barked was in the neighbourhood of the narrator, and not on another planet. During the course of a normal conversation, a lot of information is left unsaid but assumed by the conversants. Part of the interpretative task is to fill in such information from the context.

This contextual information is important, since without it statements that were not meant are produced. For instance *"The fat philosopher kicked my cat"*, does not imply that there is only one fat philosopher in the universe at a particular time. Instead it refers to the fat philosopher that I have met, and that I believe you to know at this particular time and location... In other cases, the contextual information is not needed. *"The fat philosopher who employs me kicked my cat"* may contain sufficient information to determine the philosopher uniquely. Discussion of
the many meanings of "the" is pursued in [Garigliano 92].

5.4.3.2 Sorts for events

- **Sorts on the sources of arcs**

As stated in section 5.1.2.3 (p. 121), events may participate in other events. This means that the scope of one event may be restricted by another, just as the scope of entities is restricted by definitional events. Since this is the case, the role that subject., object. and action. play in this matter must be determined: does it make sense to talk of definitional and observational subject., object. and action. arcs, or have they all an implicit sort?

As discussed in section B.1.2.4 (p. B-6), the subject., object. and action. arcs can be thought of as facts about the event, just as events can be considered as facts about other events or about entities. Thus these arcs may either be observed characteristics of the event, or may define it. Their behaviour is identical to the behaviour of events to entities, described above: Events referring to other events, behave in the same way. A few examples will illustrate the point:

\[ E_5: \{OV\text{-subject.}\exists!\forall\Delta: \text{owners}; \forall\forall\text{-action.}-IO: \text{own}; \]
\[ OV\text{-objec}t.\exists!\forall\Delta: \text{possessions}\} \]

The subject. and object. arcs are observational, whereas the action. arc is definitional. Because the scope of the event is only restricted by the action. arc, all events which have an "owns" action are known to have as subject an owner, and as object the owner's possessions.

\[ E_6: \{\forall I\text{-subject.-}IO: \text{John}; OI\text{-action.-}IO: \text{hate}; \]
\[ \forall I\text{-object.-}IO: \text{Jane}\} \]

The subject. and object. arcs are definitional, whereas the action. arc is observational. Because the event is defined by the subject. and object. arcs, and it is an instance, it states that the only event which exists or could exist involving John as the subject. and Mary as the object. is that of John hating Mary.

- In figure 5.3 (p. 145), \( E_7 \)'s subject. arc is observational, whereas its object.
and action arcs are definitional. Moreover the event $E_7$ is further restricted by a temporal event $E_8$. Because of the restrictions and the fact $E_7$ is an instance, it states that the only entity which kissed Mary at the specified time $t$ was Jacob. Note that in this example $E_7$, $E_8$ and $t$ are all mutually defining, but $t$ would be further restricted to having occurred “before now”.

Usually, an event will have definitional subject, object and action arcs. All three are needed at least to distinguish it from all other possible events.

- **The four sort combinations**

Since an arc may be observational or definitional with respect to its source and its target, there are four combinations. In the following examples, $S-T$ will be taken to mean $S$ states the sort of the arc with respect to its source, and $T$ with respect to its target, for the arcs subject, action and object.

$\Delta-O$ states that the source event is restricted by the arc’s target, but the arc’s target is not restricted by the event. For instance, if one knew that all of the children at a school failed their examinations, the event of failing the exams would have this sortal combination for its subject. Most action arcs have this type of sorts, since most events use actions rather than define them. This is a very common combination.

$O-\Delta$ states that the choice of the source event is determined by other arcs than this one, but that this one restricts its target. For instance, “Fred’s only child” might be the target of such an arc, since it is sufficient to define the event to know that its subject is Fred and its action is parenting. This combination is rare.
Δ-Δ states that the choice of both source and target are restricted by this arc. This might seem problematic, but source and target will be further restricted by some other arcs: if either is to perform any restriction on the other, it must itself be a restriction of typeless. Thus, such arcs can be understood as imposing a type restriction: they state that the source is restricted to having arcs of the relevant kind (subject-, object- or action-) connected to elements of the target’s type, and vice versa. The type of source and target are given by the other definitional arcs they are connected to. For instance,

\[(E_9,R): \{ΔF\text{-}subject\_\text{-}FΔ: Farmer_1; ΔV\text{-}action\_\text{-}IO: beat; ΔV\text{-}object\_\exists!ΛΔ: Donkey_1\}\]

states that Farmer_1 is the set of farmers who beat a donkey, and Donkey_1 is the set of donkeys beaten by farmers. Because Farmer_1 inherits all the definitional events of Farmers, these restrict the subject of E_9 to farmers. Similarly, E_9 is restricted to having as object donkeys. Finally, the action_ arc restricts it to beating, which is sufficient to determine the elements of E_9, and thus also of Farmer_1 and Donkey_1 in the organization determined by the quantification.

O-O states that neither the source nor the target restrict each other, but that this arc is relates them. This is extremely rare, but an example is described in 5.4.3.3 (p. 153).

- Template events
  - Definition

Events restricted only by their action-s are called template events\(^\text{18}\) because they form a template for all events of a given action type.\(^\text{19}\)

An event E_{10} can be defined uniquely by an action_ arc and its target action. This means that E_{10} includes all events with that action. Thus, the subject_ and object_ arcs which are observational with respect to the event, must refer to all the subjects the event has (or could have). In other words, this allows the type of the event to be defined. In some cases, this allows new useful concepts to be

\(^{18}\text{In other LOLITA literature they are sometimes also referred to as prototypical events.}\)

\(^{19}\text{An inheritable transitivity control specifies whether or not they take an object_, to ensure distributedness.}\)
defined, as in owners and possessions (see figure 5.4 (p. 147)). In other cases, the concepts thus produced are of lesser interest. For instance,

\[(E_{11}, R): \{\text{OV}\text{-subject}_-\exists \forall \Delta: \text{animals}_1; \text{\Delta \text{-action}_-\exists \forall \text{-O}: eat}; \]

\text{OV\text{-object}_-\exists \forall \Delta: food}_1

where both \text{animals}_1 and \text{food}_1 are subsets of the set of animals and food respectively. These subsets are necessary, for instance to avoid all animals being believed to eat at all times: they would all inherit \(E_{11}\) if \text{animals}_1 were replaced by \text{animals}^{20}.

- **Type-checking**

Once the range of \text{subject}s and \text{object}s an event can take has been defined in this way, type-checking can be performed with respect to this event. This is done by ensuring that the concept which is to be the \text{subject} or \text{object} of an event is a specialization of that event’s template’s \text{subject} or \text{object}, respectively. If it is not a specialization, but a generalization, it can be restricted to a specialization, by building the appropriate observational \text{spec} or \text{inst} event. For instance, John is a human, and the sentence “John owns a car” is processed. The set \text{owners} is a specialization of the set \text{humans}, so by adding an \text{inst} event from the set of \text{owners} to John, John may be an owner. It is not however possible for Clyde the elephant to own a car, since elephants and humans are antonyms: building the relevant \text{inst} or \text{spec} event is either impossible, or results in an empty set^{21}

^{20}For a discussion of inheritance, and time, see 7.5.2.7 (p. 283)

^{21}The process of ‘rejecting’ events that violate the definition of their templates is quite complex.
A sentence such as "everybody owns an animal" might be thought to cause problems with this scheme. Indeed, everybody refers to the set of all people, whereas owners are defined as a subset of people, and is a restriction by virtue of the definitional role the template event of "owning" plays with respect to the concept owner. However, what the sentence states that all people are owners. This is indeed what is built by the above rules: If $H \subset O \subset H$ then $H = O$. The new spec. relation is observational with respect to its object., so humans are not defined as owning animals.\(^\text{22}\)

In practice this type information is useful for disambiguation. An example task is to disambiguate the word star in the sentence "The president married the star". The word star has two meanings: astronomical entity, and human celebrity. A naive definition of the "astronomical star" could include an event "emits electromagnetic radiation", whereas that of "human celebrity" could include an "is human" event. Assuming that only humans can be married, the word star can be found to have referred to the meaning "human celebrity" by considering the arcs (and corresponding events) which defined each of the possible meanings: only the "human celebrity" meaning has the required human characteristic. This search is constrained to definitional attributes, or in other words to the inheritance hierarchy between the superset of "married humans" and the possible meaning.

- Non-literal control

The strict type checking copes well with literal concepts which obey their definitions. However, concepts are often used loosely. For instance in a financial text, one might state that "share prices soared today", while the soaring event can only take as subject objects that fly such as birds or airplanes. In this case, soaring is used as a metaphor which should be interpreted if LOLITA is to be successful at processing real-world financial texts.

\(^{\text{22}}\)In practice, a synonym. event would be used to avoid building a cyclic spec. structure, since acyclic spec. structures allow more efficient inheritance algorithms. The synonym. event would also be observational. See 7.6.2.1 (p. 291) for more information on synonym. events.
As discussed previously, it is advantageous to incorporate information into the central knowledge base so that all reasoning algorithms can process it. In this case, the metaphor resolution algorithms may rely on other forms of reasoning, such as causal reasoning. Thus for it to be able to use the available reasoning algorithms, one would like to express non-literal statements in the network before they are interpreted into the literal statements most reasoning algorithms process. The problem is therefore twofold. First, non-literal concepts must be recognised by processing algorithms that only treat literal concepts, to avoid errors. Secondly, they must convey the meaning expressed in the text.

Conveying the meaning expressed in the text means that the amount of information which is lost with respect to what was said is minimized. This ensures that the metaphor resolution algorithms are not denied any chance of successful interpretation. This condition is satisfied by using the representation of the statement that caused the type-clash, as this contains no more or less information that the representation of a valid statement would have had.

The problem is to ensure that the statements thus added to the network are recognised as non-literal to the processing algorithms. Indeed, adding the non-literal statement to the network is adding a contradiction to the network if taken literally. For instance, all owning events take as subject some owner which is a human. However if Caligula gave a farm to his horse, then his horse owned the farm which contradicts the statement that all owners are humans. However by adding a non-literal control to the horse owning the farm, the statement can be considered as the interpretation of a statement, without for that matter being believed literally true. It requires further interpretation for the properties it might be believed to have to be determined. For instance a round square could be interpreted as a closed line which forms a convex figure from the common properties of circles and squares. Further interpretation might lead to the plausible assumption it looks something like a square with rounded corners.

The non-literal control occurs on events, including \texttt{inst..} and \texttt{spec..}. In the example of Caligula's horse, the horse is a non-literal \texttt{inst..} of \texttt{owners} as well as being the
subject of a non-literal owning event. It is however a literal inst. of horses, and
is literally owned by Caligula himself. The non-literal control does not interfere
with definitional sorts: a statement may both define a concept and yet be non-
literal. For instance, “Dumbo is a flying elephant”, or “Mickey is a talking mouse”
contradict the definition of the meaning of flying or talking referred to, yet are
defined by them.

○ Place in the conceptual hierarchy

Since events restricted only by their action.s correspond to the full set of all events
with a particular action type a, they are the parent in the inheritance hierarchy
of all events with that particular action type. Similarly, they are specializations of
more general event types. However, unlike entity concepts which inherit restrictions
from their ancestors, template events are self-sufficient in that only their action.
restricts them. This means that a template event is connected to its parent(s)
by an observational spec. relation. Since it must be connected to a definitional
spec. relation, it is also connected to typeless23 by a definitional spec. event.
Non-template events are often connected to their parents by definitional spec. or
inst. relations, so that should they lack an action. they inherit the template’s.

Since template events correspond to the full set of all events of a particular action
type, they correspond to the concept expressed by the action. They are therefore
connected to all the observed features of that concept, not only what subject and
object it can take, but also the location where it can occur, the time and duration
it can occur, the facts required for it to occur, and the facts it results in, and so
on. This information is then inherited to all its instantiations, as described in 6.4
(p. 164).

○ Partial arcs and templates

Events which have partial arcs, such as “near”, must be connected to all these
partial arcs to be understood (5.1.2.4 (p. 123)). For instance, an object cannot

23 Actually, it is connected to a synonym of typeless so that search down from typeless itself
leads to the conceptual hierarchy, rather than to concepts which are children of typeless for
technical reasons. For further information on synonyms, see 7.6.2.1 (p. 891)
be near alone, but must be near to something else, so its event always expects more than one partial subject. This is expressed by defining the set that has as

subject the event, and stating that the cardinality of each element of it must be
some value greater than one, in this case two:

\[(E_{12}, R): \{\OV\text{-subject-} \exists!\forall\Delta: \text{locations}; \ \DA\text{-action-}IO: \text{near}\}\]

\[(E_{13}, R): \{\DA\text{-subject-} \exists!\forall\Delta: \text{locations}; \ \DA\text{-action-}IO: \text{size};
\ \DA\text{-object-}IO: 2\}\]

The partial subject arcs need not always refer to elements of the same set, as
shown in the previous example. In this case, the template event will have the
partial arcs connected to their different sets:

\[(E_{14}, R): \{\OV\text{-subject-} \exists!\forall\Delta: \text{unquantifiable}_1;
\ \OV\text{-subject-} \exists!\forall\Delta: \text{values_powerset}_1;
\ \DA\text{-action-}IO: \emptyset; \ \OV\text{-object-} \exists!\forall\Delta: \text{unquantifiable}_2\}\]

It is possible for some events to entail that their relation also holds between a
subset of their partial subjects. For instance, "Macbeth met three witches" also
implies that Macbeth met each witch, as shown by the sentence "You have already
met Mandy, haven't you?" uttered some time later. However, one reading of the
statement implies one event ("Macbeth met three witches. The meeting had been
arranged weeks earlier..."), whereas another does not ("The meetings took place in
a series of hovels"). These entailments are expressed using inheritance:

\[(E_{15}, R): \{\OV\text{-subject-} \exists!\forall\Delta: \text{meetings};
\ \DA\text{-action-}IO: \text{meet}; \ \OV\text{-object-} \exists!\forall\Delta: \text{meetees.}\}\]

\[(E_{16}, R): \{\DA\text{-subject-} \exists!\exists!\forall\Delta: \text{meetings};
\ \DA\text{-action-}IO: \text{meet}; \ \DA\text{-object-} \exists!\exists!\forall\Delta: \text{meetees.}\}\]

○ Antonyms of templates

Template events can be declared antonyms, allowing events such as "like" and
"dislike" to be related. Although there may not be any linguistic term for the gen-
eral event that is partitioned in this way, it must be included in SemNet. Indeed,

\footnote{\OV\text{-subject-} \exists!\forall\Delta: \text{locations}_s is used instead of \OF\text{-subject-} \exists!\forall\Delta: \text{locations}_s because one may wish to have more than one event of \(E_{12}\) which has the same subject.s of locations.s for instance, for locations one does not wish to distinguish by time but which were close at different times, such as two ships meeting at different times – see 7.5 (p. 271).}
antonym partitions a concept into parts, such that every instance of the concept is an instance of one and only one of its parts. In this case, the generic event is associated with the area of human emotions to do with attachment and aversion. It could therefore be associated with or even defined by a model of human emotions, enabling LOLITA to make inferences from statements such as “John likes his horse”.

Examples of the use of template antonyms include conversion to antonym (see D.2.2.1 (p. D-31)) and natural language generation. For instance, this allows LOLITA to generate either “I like Margaret” or “I do not dislike Margaret”, improving the richness of her expression (see [Smith 95] for more details.)

5.4.3.3 Expressing implicature

Often in logic one wishes to state that one statement implies another, purely by virtue of it being true. For instance:

$$\forall x.\text{man}(x) \Rightarrow \text{mortal}(x)$$

In pure set notation this can be seen as saying the set of men is a subset of the set of mortal things. But this can be expressed more simply by sorts.

- Simple implicature

Simple implicature can be expressed by stating the events corresponding to $a$ in $a \Rightarrow b$ as definitional, whereas $b$ are expressed as observational. For instance:

$$(E_{17}, R): \{\forall \text{-subject.-蚓}!: \text{animal}_1; \forall \text{-action.-IO}: \text{eat};$$

$$\forall \text{-object.-蚓}!: \text{food}_1\}$$

$$(E_{18}, R): \{\Delta \text{-subject.-FO}: \text{animal}_1; \forall \text{-action.-IO}: \text{is\_hungry}\}$$

corresponds to

$$\forall x\forall y.\text{animal}(x) \land \text{food}(y) \land \text{eats}(x, y) \Rightarrow \text{hungry}(x)$$

The advantage of expressing implicature in this manner is that all reasoning involving it can use inheritance, instead of a dedicated reasoning algorithm. Moreover,
implementing it in this way is efficient because all instances satisfying the left hand of the implication are explicitly stated in the network. This should be contrasted with the FOL formula where variables are quantified over the whole universe. A naive implementation using FOL would need to test every known concept in its knowledge base to determine to which concepts the rule applied. Instead many FOL-based systems such as CYC state the set over which quantifiers quantify in order to reduce the processing effort required when dealing with such statements. See [Lenat et al. 90] for more details.

• Bi-implicature

Bi-implicature can also be expressed by sorts, despite the fact that in set notation, it corresponds to two sets defined in different ways having the same elements. This section gives an example: sibling. The sibling action relates any two people having a common parent. Thus if one knows two people are related by a sibling event, then one knows they share a common parent. But what one also knows is that it suffices for two people to have the same parent for them to be related by a sibling event. This is rare, as often events’ templates define their possible subject.s or object.s to prevent them being inferred from the existence of what is known about them.

\[
(E_{19}, R): \{ \Delta F \forall \text{-subject.}_F \Delta: \mathcal{P}_1; \Delta \forall \forall \text{-action.}_I; \text{O: parenting}; \\
\Delta \forall F \text{-object.}_F \exists \forall \Delta: \mathcal{C}_1 \}
\]

\[
(E_{20}, R): \{ \text{OF} \text{-subject.}_F \forall \text{O}: \mathcal{C}_1; \Delta \forall \text{-action.}_I; \text{O: sibling} \}
\]

\[
(E_{21}, R): \{ \Delta F \text{-subject.}_F \Delta: \mathcal{C}_1; \Delta \forall \text{-action.}_I; \text{O: size}; \\
\Delta \forall \text{-object.}_F \exists \forall \Delta: \mathcal{S}_1 \}
\]

\[
(E_{22}, R): \{ \text{OF} \forall \text{-subject.}_F \Delta: \mathcal{P}_0; \Delta \forall \forall \text{-action.}_I; \text{O: parenting}; \\
\text{OF} \forall \forall \text{-object.}_F \forall \Delta: \mathcal{C}_0 \}
\]

where \( \mathcal{P}_1, E_{19}, C_1 \) are definitional spec.s (\( E_{29}, E_{30}, \) and \( E_{31} \) respectively) of \( \mathcal{P}_0, E_{22}, \) and \( C_0 \) respectively; all elements of \( S_1 \) are defined to be any value greater than 1. This fragment of network above states that \( \mathcal{P}_1 \) is the set of parents who

\[25\text{This can also be expressed directly using the synonym event, see 7.6.2.1 (p. 291)}
\[26\text{For further information see 7.1.1 (p. 247)}\]
parent more than one child, $C_1$ is the set of such children grouped by parent. $E_{20}$ is the template event for states that all events with action sibling have as subject elements of $C_1$, defined by $E_{19}$ and $E_{21}$, and the definitional events on $C_0$, its parents. Notice $E_{20}$'s bi-observational OFV-subject-FO arc

$$(E_{23}, R): \{\Delta V\text{-subject-}I\Delta: P_2; \Delta V\text{-action-}I\Delta: \text{parenting};$$
$$\Delta F\text{-object-}I\Delta: C_2\}$$

$$(E_{24}, R): \{\Delta I\text{-subject-}I\Delta: [\text{Fred, Mary}]; \Delta I\text{-action-}I\Delta: \text{sibling}\}$$

(where $E_{23}$ is inferred as follows:)

First, consider whether knowing that instantiations are siblings implies that they have the same parent. Because $E_{20}$ is defined only by its action arc, it is the template for the sibling action. Thus all sibling events will be subsets or instances of it. Without loss of generality, consider the event $E_{24}$ "Fred sibbles Mary". Because $C_1$ is the subject of the sibling template, Fred and Mary must be instances of an instance ($C_2$) of it. Since $C_2$ is an instance of $C_1$, it inherits the event $E_{19}$ as $E_{23}$. This states that all elements of $C_2$ share a parent. Thus, if one knows that instantiations are siblings, then one knows they share a parent.

$$(E_{25}, R): \{\Delta I\text{-subject-}I\Delta: \text{Judy}; \Delta I\text{-action-}I\Delta: \text{parenting};$$
$$\Delta I\text{-object-}I\Delta: \text{John}\}$$

$$(E_{26}, R): \{\Delta I\text{-subject-}I\Delta: \text{Judy}; \Delta I\text{-action-}I\Delta: \text{parenting};$$
$$\Delta I\text{-object-}I\Delta: \text{Jane}\}$$

$$(E_{27}, R): \{\Delta V\text{-subject-}I\Delta: \text{Judy}; \Delta V\text{-action-}I\Delta: \text{parenting};$$
$$\Delta F\text{-object-}I\Delta: C_3\}$$

$$(E_{28}, R): \{\Delta I\text{-subject-}V\Delta: C_3; \Delta I\text{-action-}I\Delta: \text{sibling}\}$$

(where $E_{28}$ is inferred as follows:)

Second, consider whether knowing that instantiations share a parent implies that they are siblings. Because $E_{22}$ is defined only by its action arc, it is the template for the parenting action. Thus all parenting events will be subsets or instances of it. Without loss of generality, consider the events $E_{25}$ and $E_{26}$ stating that John and

\[^{27}\text{This fragment is a simplification of the full scheme, since every sibling event need not involve all children so a powerset relation is needed from every element of } C_1, \text{ to elements sets in } C_1' \text{ which would be the subject of } E_{20}. \text{ However an explanation of the full scheme would be unnecessarily complicated.}\]
Jane have as parent Judy. They are instances of the instance $E_{27}$ of $E_{22}$. Because $E_{22}$ is a template, John and Jane are instances of an instance $C_3$ of $C_0$, the set of sets of children grouped by parent. Because $E_{27}$ and $C_3$ have two instances that are known to be different, and may have more, they fit the definitional requirements of $E_{19}$ and $C_1$, given by $E_{19}$ and $E_{21}$ (and the relevant spec. events). Thus they are instances of $E_{19}$ and $C_1$. Once this is made explicit, they inherit the observational $E_{20}$, thus stating that John and Jane and all other elements of $C_3$ are siblings: $E_{28}$.

This can be further inherited down to John and Jane, in the general case explained in footnote 27 (p. 154). Thus, if one knows that instantiations share a parent, one knows that they are siblings.

Thus, the network fragment provides bi-implicature. The bi-observational subject arc states that $E_{20}$ is sibling's prototypical event, yet it does not define its subject, since $E_{19}$ and $E_{21}$ do that. Thus, something is said about the sibling action, although it is not defined. Indeed, it must retain the notion of being different to other events, without which its template event would include any symmetrical event involving someone's children, for instance being near each other.
Chapter 6

Reasoning

A KB's data can be structured to take advantage of the reasoning techniques its representation supports. Since SemNet's extended representation builds on the basic representation without adding new primitive constructs, the basic representation's reasoning methods apply to any SemNet expression. They can therefore be used to structure the whole KB – as is assumed throughout this thesis. This chapter defines the forms of basic reasoning SemNet assumes: the common element rule, synonym tracing, inheritance, type-checking, and semantic integration.

6.1 Common Element Rule

6.1.1 Description

The common element rule specifies how instances of concepts are chosen by the relations that apply to them. Once an instance $i$ is chosen for a particular set quantification level $l$ of a concept $n$, all arcs referring to $l$'s level of quantification refer to $i$. A set quantification is a quantification that refers to elements of the set represented by the node to which it refers: universal or existential quantification\(^1\). In other words, when a set quantification chooses a particular element of a set, all the other set quantifications off that node chose that element too.

\(^1\)and hence also Framed Universal; see B.1.2.3 (p. B-4)
In practice this means that if there are four arcs referring to a node, two by universal quantification, one by existential quantification and one by individual quantification, the three arcs involving a set quantification at the node refer to the same element. Thus the statements:

\[ E_0: \{I\text{-subject.}-\forall : [N]; \ldots \} \]
\[ E_1: \{\exists !\text{-object.}-\forall : [N]; \ldots \} \]
\[ E_2: \{\forall \text{-subject.}-\exists !: [N]; \ldots \} \]
\[ E_3: \{I\text{-subject.}-I: [N]; \ldots \} \]

refer to the following in $\mathcal{FOL}$:
\[
\forall n \in N.(\text{Subj}(n, E_0) \land (\exists ! e_1 \in E_1.\text{Obj}(n, e_1)) \land (\exists e_2 \subseteq E_2.\forall e \in e_2.\text{Subj}(n, e))) \\
\land \text{Subj}(N, E_3) \land ...
\]

Notice that $n$ is shared by all the relations corresponding to the arcs.

The part left out ("...") in the statements corresponds to the other arcs connected to the various events. The notation is as follows:

\[ N: \{\text{quantif}_1 - \text{arc\_label} - \text{quantif}_2: \text{target\_node}, <\text{other arcs}> \} \]

The node $N$ is the source of the arcs expressed in the curly brackets. Every arc is of the form $q_1\text{-label} - q_2$ where $q_1$ is the quantification of the arc with respect to $N$ and $q_2$ is the quantification of the arc with respect to the target node.

The part left out in the $\mathcal{FOL}$ expression corresponds to the part left out in the statements, and to the rest of the $\mathcal{FOL}$ statements for relations involving $\forall - \exists !$ quantification.

This rule is of little consequence if the events are quantificationally independent. However, it is important for events which share an existentially quantified object depending on their universally quantified subject such as: "every man loves his mother", where one event expresses the fact that there is a "mothering" relation between each man and his mother, and the other event states that each man loves a mother of the set of all the men’s mothers. It is only the common element rule that forces the two men chosen by each event to be one and the same, thereby forming the connection that "each man loves the mother that mothers him". Mothering is
now used to choose each man’s mother, rather than just to define the set of all the men’s mothers. The “mothering” relation consists of all the behaviours normally performed by mothers with respect to their children, such as caring for them.

Since the common element rule always applies (to all quantifications other than $A$), sentences such as “Every person’s mother loves someone’s mother” require the introduction of a second mother concept: an individual $\text{spec.}$ or $\text{synonym.}$ of the first. The individual quantification breaks the quantificational dependency between the mothers:

$\begin{align*}
(E_0,R) & : \{\exists F, \exists \text{-subject.-FO: people}; \forall \text{-action.-IO: love;} \\
& \quad \forall \text{-object.-∃!∆: mothers}\} \\
(E_1,R) & : \{\forall \text{-subject.-∃!O: mothers}; \forall \text{-action.-IO: mother;} \\
& \quad \exists F, \exists \text{-object.-FO: people}\} \\
(E_2,R) & : \{\exists F, \exists \text{-subject.-FO: mothers}; \forall \text{-action.-IO: love;} \\
& \quad \forall \text{-object.-FO: mothers}_2\} \\
(E_3,R) & : \{\forall I, \forall \text{-subject.-IO: mothers}; \forall I, \forall \text{-action.-IO: synonym.}; \\
& \quad \forall I, \forall \text{-object.-I∆: mothers}_2\}
\end{align*}$

or

$\begin{align*}
(E_4,R) & : \{\exists F, \exists \text{-subject.-FO: mothers}; \forall \text{-action.-IO: love;} \\
& \quad \forall \text{-object.-∃!∆: mothers}_2\} \\
(E_5,R) & : \{\forall I, \forall \text{-subject.-IO: mothers}; \forall I, \forall \text{-action.-IO: spec.}; \\
& \quad \forall I, \forall \text{-object.-I∆: mothers}_2\}
\end{align*}$

6.1.2 Non-Linearity

Note that no explicit element of the set corresponding to a node is actually chosen. The rule states rather that if an element were chosen, all arcs involving a set quantification would refer to it. Thus if the arcs connected to the set were inherited down to one of its explicit instances, they would all refer to the same instance. Therefore the common element rule does not break non-linearity. If the first arc with set quantification traversed to a set were to choose the element of that set non-linearity would be broken: The first arc may not involve any dependency of
quantification, but another arc off that node might. Since the choice of element depends on the arcs involving quantification dependency, they would have to be traversed first which would impose an order in which the net had to be read. Instead the rule does not expect the first arc traversed to choose the element, but only that the element referred to by all the arcs is the same. This means that if a choice were to be made it would only be made once all the arcs off the node involving a quantification dependency had been traversed. However no choice is actually made, except when inheriting.

6.1.3 Quantificational dependency

The common element rule extends quantificational dependency beyond the two quantifications of every arc. For instance, "every person's father loves that person's mother" is shown in 6.1 (p. 159).

Quantificational dependencies affect reasoning:

Inheritance must ensure that the instances it builds obey the common element rule. For instance, a single donkey must be built when "Every farmer who owns a donkey beats it" is inferred to donkey-owning farmer Giles.

*synonym* events can extend quantification dependencies by connecting the instances of two concepts. This creates a secondary level of dependency. Such *synonym* events can be negated, requiring additional reasoning to ensure that
the chosen instantiations are indeed different: an inconsistency should be detected if the chosen instantiations are proven equal.

Cardinality reasoning must build only one size event per set. However each quantificational dependency may indicate a relationship between the cardinalities of the sets it relates. Cardinality reasoning uses these relations to infer set cardinalities.

### 6.2 Quantification and Cardinality

Quantification introduces a new form of reasoning since it has implications on cardinality – implications which are used by inheritance for instance. Quantification reasoning is relatively simple:

1. It makes explicit the equivalence between individual quantification and cardinality one.

2. It makes explicit the cardinality relation of $\forall - \exists!$ which states that the cardinality of the left hand set is greater or equal to that of the right hand set.

3. If the quantification relation at an end of an arc involves more than one level of quantification (such as the left hand side of $\forall \forall - \exists!$), it can be treated as a unit whose cardinality is the sum of the cardinalities of the instances at the deeper quantification level.

The cardinality relations of $F - F$ and $\exists! - \forall$ can be inferred from the second rule. The cardinality relations of $\forall \forall - \exists!$, or $FF - F$ or $\forall - \exists!\exists!$ can be inferred from rules 2 and 3.

Because the resulting cardinality relationships are derived from quantificational relationships, they express in graph form the quantificational dependencies between different levels of quantification of different concepts. Consider instances of depth $d$ of a multileveled concept $c$. All the concepts these instances quantificationally
depended on can be reached by traversing the graph inferred from the arcs which apply to them.

Values reasoning can use the inferred cardinality relationships to deduce further information. However this is a separate process, as is the unification of cardinality relationships inferred by the above process with pre-existing cardinalities expressed in the KB. Unification and further reasoning break the inferred graph's property of reflecting quantification dependencies.

6.3 Synonym Tracing

SemNet uses synonym events to state equality between two concepts $S$ and $O$ (or their instantiations at any given level). Events known of $S$ are also true of $O$. This applies to all events including cardinality relations, ensuring that if $S$ is known to have a given cardinality, it is inferred over to $O$ too. This kind of inference is called synonym tracing.

6.3.1 Sorts

The definition of synonym gives its behaviour with respect to sorts (7.6.2.1 (p. 291)): If the synonym is definitional with respect to $S$ or $O$ respectively, definitional events of $O$ or $S$ respectively will be inferred as definitional for $S$ or $O$ respectively. In all other cases the events will be inferred to be observational.

6.3.2 Synonym events can be qualified by events

SemNet extends on classical equality by allow synonym events to be qualified by an event. For instance, the synonym connecting Pluto to the concept the furthest planet is qualified by an at time event specifying which years Pluto was the furthest planet. Since the furthest planet of the solar system is Pluto some of the time, events inferred to it from it being Pluto only apply to it for that time.
If $S$ is connected to $O$ by a path (more than one) of synonym events, all of the events qualifying any of the synonym events of the path apply to the inferred events: This is what would result from inferring $S$'s events synonym by synonym down the path. With full semantic integration enabled, the resulting set of events qualifying the inferred events should not be self contradicting: semantic integration makes sure that concepts are connected to concepts they subsume or are subsumed by. But with only partial semantic integration, self-contradictions may occur. If the events qualifying an inferred event contradict each other, the inheread event does not apply.

In general, $S$ and $O$ can be connected by more than one path of synonym events, each with its own set of events $E_1, E_2...$ qualifying it. Tracing the events of $S$ to $O$ requires tracing them over each path from $S$ to $O$. Multiple copies of $S$'s events are attached to $O$, each qualified by a set of events $E_1, E_2...$ For instance the furthest planet may be a synonym of Pluto for some period of time, and a synonym of Neptune for the rest. It would therefore inherit two is.planet events, one applying to the Pluto period, and the other applying to the Neptune period. It is up to domain specific reasoning, in this case time reasoning, to combine the resulting events at $O$'s level into the minimum number of events. This results in the the events is.planet to be correctly inferred to apply all the time.

6.3.3 Synonym events and the Belief control

Due to the necessary assymetry of the belief event (7.2.1.3 (p. 258)), the property that LOLITA believes an event is specified as a control. A synonym event can therefore have a hypothetical or a real belief control, corresponding to the property that LOLITA holds no belief in the synonym event or not (i.e. LOLITA believes the synonym event). The property of LOLITA not believing an event is treated as if it were an event: all events inferred over a synonym event qualified by event $e$ are built qualified by $e$. Thus, any real or hypothetical event inferred over a hypothetical synonym event is built hypothetical, and any hypothetical event inferred over a real synonym is built hypothetical. The lack of the property of LOLITA not believing
an event corresponds to a real event: any real event inferred over real synonym event is built real. Since a hypothetical synonym event must be qualified by a belief event $b$, the inferred events will also be qualified by $b$.

### 6.3.4 Synonym events and Believed events

If knowledge not known to be believed by an agent is used to infer something about what that agent believes, the reasoning is plausible. Thus if an event believed by an agent is traced over a synonym event not known to be believed by the agent, the synonym tracing is plausible, requiring the event’s belief and certainty values to be adjusted [Bokma et al. 92].

### 6.3.5 Copying Inferred Events

Events built by synonym tracing are simply a copy $c$ of the original event $o$:

- The target sort of the arc connected to $S$ or $O$ is as specified in 6.3.1 (p. 161).
- The other arcs are connected to the same nodes as they are for $o$.
- The other sorts are exact copies of those of $o$.
- The quantifications are exact copies of those of $o$.

Note that events built in this way are actually synonyms, and must be built associated by a definitional synonym event: any event qualifying $o$ must also qualify $c$, since if it defines $o$ and $c$ is a copy of $o$ then it must define $c$. Thus if a copied event is itself qualified by another event, synonym tracing can be performed on it as well.
6.4 Inheritance

Once the set relations have been defined, a hierarchy can be built using them. The features and advantages of this are discussed in 2.7 (p. 16). All concepts, other than the inst. and spec. relations themselves are members of this hierarchy since it is an advantageous way of organizing knowledge. This section discusses the process which ensures that concepts only need be expressed at the relevant level of granularity: inheritance.

If an event is known to apply to all the elements of a set, one knows that it applies also to a subset formed of these elements. The task of the inheritance algorithm is to determine all the events known to apply to a particular node, even if they are not explicitly connected to this node, but are to its ancestors. Although this sounds simple, in practice a lot is expected of the inheritance algorithm. For instance, if one knows that all creators who make something are proud of it, and one knows that Naborkov wrote the book “Lolita”, and one knows that to write is a form of making, then the inheritance algorithm should deduce that Naborkov is proud of it by inheriting the event to the appropriate place. Similarly, if LOLITA knows that every person’s father loves that person’s mother, and that Sacha’s father is Nikolai, then she should inherit Nikolai loved Sacha’s mother.

In general, large gaps in a sparse structure between nodes at the lower end of the hierarchy (typeless being at the top) can be filled in this way. More complex examples involve inheriting facts known about concepts. For instance, the statement “John bought a book at the bookshop” involves a lot of knowledge which can be inherited: every shop has a shop attendant; shop employees sell things to customers; people who buy things from shop attendants are customers; buying means that the seller gives the thing to the buyer and the buyer pays the seller; paying is giving money to someone; to give something one must own it first... Thus the statement gets filled in by nodes corresponding to the shop attendant, the money payed, the fact that John owned the money before he gave it, and so on... Such information proves useful during interpretation since the statement could be followed by “He had stolen the money from his father”, which refers to the money implicit in the
buying.

More details of the inheritance algorithm are given below. The advantage of building such a powerful inheritance algorithm lies not only in the ability to structure knowledge at different levels of granularity, but also to understand statements in terms of others which may be associated with certain behaviours. For instance, in the John buying a book example, LOLITA need not have a special procedure to treat the concept “to buy”, but can infer that it corresponds to a small number of simpler statements that are associated with special forms of reasoning, such as cause, time and location. Nothing prevents buying to be associated with its own special type of reasoning, if required, say for efficiency reasons. However, this is a matter of choice, not of necessity. Inheritance thus provides a significant degree of flexibility, allowing one to develop reasoning methods for specific types of concepts, yet not needing to express all information in terms of those primitives.

### 6.4.1 Definition of Inheritance

Each spec. event states that every instantiation of its object. \( O \) is an instantiation of its subject. \( S \). This means that every property of \( S \)’s instantiations also applies to \( O \)’s instantiations. Inheritance’s task is to build (possibly temporary) explicit events inferred from \( S \) and connect them to \( O \).

### 6.4.2 spec. events can be qualified by other events

SemNet differs from classical set-inheritance, in that the set relation spec. can be qualified by an event, in the same way as the synonym. event can. For instance, the spec. connecting Jack the fireman to firemen is qualified by a at time event specifying which years Jack was a fireman. Since Jack’s membership of the concept firemen only applies for certain years, the events inherited down to Jack must also only apply for those years.

If \( S \) is connected to \( O \) by a path (more than one) of spec. events, all of the events qualifying any of the spec. events of the path apply to inferred events:
This is what would result from infering $S$'s events \texttt{spec.} by \texttt{spec.} down the path. With full semantic integration enabled, the resulting set of events qualifying the inferred properties should not be self-contradicting: semantic integration makes sure that concepts are connected to concepts they subsume or are subsumed by. But with only partial semantic integration, self-contradictions may occur. If the events qualifying an inferred event contradict each other, the inferred event does not apply.

In general, $S$ and $O$ can be connected by more than one path of \texttt{spec.} events, each with its own set of events $E_1, E_2...$ qualifying it. Inheriting the properties of $S$ to $O$ requires inheriting them over each path from $S$ to $O$. Multiple copies of $S$'s properties are attached to $O$, each qualified by a set of events $E_1, E_2...$ For instance Jack may be a \texttt{spec.} of child for the first eighteen years of his life, and a \texttt{spec.} of adult for the rest. He would therefore inherit two \texttt{is.human} properties, one applying to the first 18 years of his life, and the other applying to the rest. It is up to domain specific reasoning, in this case time reasoning, to combine the resulting events at $O$'s level into the minimum number of events. This results in the property \texttt{is.human} to be correctly inferred to apply to the whole of Jack's life.

### 6.4.3 \texttt{spec.} events and the Belief control

Due to the necessary assymetry of the belief event (7.2.1.3 (p. 258)), the property that LOLITA believes an event is specified as a control. A \texttt{spec.} event can therefore have a hypothetical or a real belief control, corresponding to the property that LOLITA holds no belief in the \texttt{spec.} event or not (i.e. LOLITA believes the \texttt{spec.} event). The property of LOLITA not believing an event is treated as if it were an event: all events inferred over a \texttt{spec.} event qualified by event $e$ are built qualified by $e$. Thus, any real or hypothetical event inferred over a hypothetical \texttt{spec.} event is built hypothetical, and any hypothetical event inferred over a real \texttt{spec.} is built hypothetical. The lack of the property of LOLITA not believing an event corresponds to a real event: any real event inferred over real \texttt{spec.} event is
built real. Since a hypothetical spec. event must be qualified by a belief event b, the inferred events will also be qualified by b.

6.4.4 spec. events and Believed events

If knowledge not known to be believed by an agent is used to infer something about what that agent believes, the reasoning is plausible. Thus if an event believed by an agent is inherited over a spec. event not known to be believed by the agent, the inheritance is plausible, requiring the event’s belief and certainty values to be adjusted [Bokma et al. 92].

6.4.5 Inheritance and Sorts

Inheritance ensures conservation of definition. If a node d is a definitional descendant of a node a, it inherits all of a’s definitional events as definitional: a’s definitional events also define it. It inherits all of a’s observational events as observational, and inherits all events from observational ancestors as observational. An ancestor is definitional if there is an uninterrupted path of definitional set relations from it to the descendent, otherwise it is observational.

However, inheritance infers new nodes, which are not yet integrated into the inheritance hierarchy. So unlike for the direct descendents, the inheritance hierarchy can not guide the choice of sorts for the arcs of these nodes. These nodes are built as definitional descendents of the corresponding ancestor nodes: an observational is one that just happens to be true (not by definition), so an automatic inferencing process cannot deduce this fact, it has to be told. This might lead one to assume that the inherited arcs should have the same sortal restrictions as those from which they were inherited. But, this is not the case. For instance, take “Fish fed by machines are owned by people”:

\[(E_0,R): \{\Delta V\text{-subject.}-\exists!\Delta: \text{machines}_1; \Delta V\text{-action.}-\text{IO: feed};
\\Delta V\text{-object.}-\exists!\Delta: \text{fish}_1\}\]
(E1,R): \{\Delta F\text{-}object\_FO: fish_1; \Delta \forall\text{-}action\_JO: own; \\
\Delta \forall\text{-}subject\_\exists!\Delta: \text{people}_1\}\}

Now take John, an owner of some of these fish. Conservation definition implies that if he is a definitional/observational inst. of people_1, he will be defined/observed as the subject_ of the event E3 inferred from E1. The event E3 is defined by John for the same reason as E1 is defined by people_1. Similarly E3’s action_ and object_ define it: John cannot be assumed to only own some fish. So far the sorts of the ancestor level have been mirrored at the descendent’s level. But if we do this for John’s fish John\_fish, the statement produced is illegal:

(E2,R): \{\Delta \forall\text{-}subject\_\exists!\Delta: \text{machine}_1; \Delta \forall\text{-}action\_JO: feed; \\
\Delta \forall\text{-}object\_\exists!\Delta: \text{John\_fish}\}\}

(E3,R): \{\Delta F\text{-}object\_FO: \text{John\_fish}; \Delta \forall\text{-}action\_JO: own; \\
\Delta \forall\text{-}subject\_\exists!\Delta: \text{John}\}\}

This states that John’s fish are the fish fed by some automatically inferred machine_1, itself not uniquely defined. The problem is that the point of reference is John since facts are being inferred about him. Inheritance must take this into account.

John is uniquely defined. He is also an instance of those people owning fish fed by machines. Unless there is an explicit non-inferred event about these fish, they are purely intensional in the sense that their posited existence depends completely on him and the fact that he is believed to be a person owning fish fed by machines: there is no extensional confirmation of their existence that could serve as their definition to render them unique. Since this fish concept owes its inferred existence to John, John must define it: nothing else can define it uniquely. The resulting structure is:

(E2,R): \{\Delta \forall\text{-}subject\_\exists!\Delta: \text{machine}_1; \Delta \forall\text{-}action\_JO: feed; \\
\Delta \forall\text{-}object\_\exists!\Delta: \text{John\_fish}\}\}

(E3,R): \{\Delta F\text{-}object\_FO: \text{John\_fish}; \Delta \forall\text{-}action\_JO: own; \\
\Delta \forall\text{-}subject\_\exists!\Delta: \text{John}\}\}

John’s fish must remain defined by E2 or the inferred event would state that all of John’s fish were fed by machines, something which is not known.

In general, quantification requires certain nodes to be mirrored down at the descendent’s level. Simple mirroring of the sortal restrictions at the ancestor’s level does
not guarantee that all these nodes are defined uniquely. This must be guaranteed, so inheritance must work out a way of achieving this. The only uniquely defined concept inheritance is dealing with is the concept \( c \) it is inheriting properties about, so all nodes that would be inherited non-uniquely must be tied back to \( c \). This is achieved by maintaining an is_defined flag while traversing the arcs. is_defined stays true until an arc is traversed from \( a \) to \( b \) such that the arc qualifies \( b \) observationally. If no other path leading to \( b \) is definitional, then \( b \) must be made to depend on \( c \). There may be more than one ways of doing this. In a cyclic structure like:

\[(E_0, R): \{\forall \text{-subject} \neg \exists \Delta: \text{machines}_1; \forall \text{-action} \neg \text{-IO}: \text{feed};
\forall \text{-object} \neg \exists \Delta: \text{fish}_1\}\]

\[(E_1, R): \{\forall \text{-subject} \neg \exists \Delta: \text{fish}_1; \forall \text{-action} \neg \text{-IO}: \text{own};
\forall \text{-object} \neg \exists \Delta: \text{people}_1\}\]

\[(E_4, R): \{\forall \text{-subject} \neg \exists \Delta: \text{people}_1; \forall \text{-action} \neg \text{-IO}: \text{pay};
\forall \text{-object} \neg \exists \Delta: \text{corporations}_1\}\]

\[(E_5, R): \{\forall \text{-subject} \neg \exists \Delta: \text{corporations}_1; \forall \text{-action} \neg \text{-IO}: \text{own};
\forall \text{-object} \neg \exists \Delta: \text{machine}_1\}\]

there are 3 solutions:

- Changing the object arc of \( E_3 \) to \( \Delta - \Delta \).
- Changing the object arc of the event inferred from \( E_4 \) to \( \Delta - \Delta \), and changing the object arc of the event inferred from \( E_5 \) to \( \Delta - \Delta \)
- Doing both of the above.

Each of these solutions is valid, but vary according to what they state is observed and what is defined. For instance, the first solution states that it is known of John's fish that they are fed by a machine, that this machine is owned by a corporation, and that John pays this corporation. In the second option, the only observational statement is that John owns the fish defined as the fish fed by the machine of the corporation John pays. In the third option nothing is observational.

Since all of the solutions are valid, they could all be built. For sets, only some of
them need to be built, since the other solutions are subsumed by these few. For instances all need to be built. Quantification may allow synonym relations to be deduced between the concepts inherited from the same node.

Sorts therefore can introduce ambiguity to the inherited properties. At present, it is unclear whether this will be a problem or not in a real system. If in practice this results in an exponential number of inherited concepts, it will always be possible to choose the largest common denominator: the least restricted solution which is either subsumed or subsumes every other solution. At worst this results in some information loss (observational facts cannot be deduced so represent information), but this is only loss of information that is uncertain at best since if an arc is not observational in the highest common denominator then it is not in some of the alternative solutions.

6.4.6 Inheritance and Quantification

6.4.6.1 Algorithm Outline

Quantification determines which properties are inherited. The common element rule adds some complexity: if 2 events at the level of the ancestor refer to the same instantiation of a node, they must also refer to the same instantiation of the node's descendent. Since instantiations are not explicit, the inheritance algorithm must keep track of them itself.

An ancestor $a$ is known to be connected to its descendent $d$ by a path of $\text{spec.}$ (and $\text{inst.}$) events. These state either by their $\text{action.}$ or their quantification that the top level quantification of $d$ refers to the $n$th level quantification of $a$. For instance, if $d$ is a subset of $a$, $n = 0$. If $d$ is an $\text{inst.}$ of $a$, $n = 1$. If $d$ is an element of an element of $a$, $n = 2$. Only nodes at a's level that apply to $d$ must be inherited: nodes with set quantifications ($\forall$, $\exists$, $F$, $A$) at level $\geq n-1$ of $a$. The $n-1$ comes from the fact that set quantification at level $n-1$ refers to instances of the $n$th level.

The inheritance algorithm searches laterally (not traversing the inheritance hierar-
through the net from $a$. It crosses only arcs involving a quantification related
to one of $a$'s of depth greater than $n - 1$. It builds at $d$'s level a copy of every arc and
node that it traverses. It needs to keep track of which quantifications of a new node
correspond to which quantifications of $a$, since they can be swapped over an arc: $[F]F$-subject.-$F[F]$. One way of achieving this is to annotate each node reached
with a map stating which of its quantifications refer to which quantifications of $a$. Another is to use cardinality to express the quantification dependencies.

The search along a particular path is terminated by termination conditions, some
sortal, and others quantificational. These termination conditions correspond to
cases where building a new node at $d$'s level breaks uniqueness. For instance,
when inheriting "The king owns his possessions" to an instance $d$ of the king's
possessions: the king's crown, building a new node to represent the king would
break uniqueness.

Thus only certain types of quantification affect concept uniqueness. Generally, if
two connected nodes are quantificationally independent, uniqueness is not affected
and no new node needs be build. If the nodes are quantificationally dependent,
then a new node will need to be built:

- If two connected nodes are quantificationally independent:
  
  - If the node reached from $a$ is an individual, then it cannot be inherited:
a new cat is not built for the mice John, Jack and Jacob, when inheriting
  from "The cat caught the mice".

  - If the node reached from $a$ can only be reached traversing arcs involving
    no quantificational dependency (such as $\forall - \forall$, but not $\forall - \exists$), and the
    node reached is not defined by any of these arcs (uniqueness: see below),
    the node should not be duplicated, and the search should terminate.

- If two connected nodes are quantificationally dependent:
  
  - If two nodes at $a$'s level are connected by a $F - F$ quantification, then
    this connection (which may span more than one arc) must be inherited
down to $d$'s level: if the first node encountered was inherited down, then
it it is different to its ancestor (uniqueness), so has a different number of possible instantiations. The $F - F$ implies that the second node has as many instantiations as the first, which is true both at $a$’s level and at $d$’s level.

- If two nodes $x$ and $y$ are connected by a $\forall - \exists!$ quantification, every instantiation of $x$ corresponds to a single instantiation of $y$, but every instantiation of $y$ may be associated with more than one instantiation of $x$. Although this could mean that every instantiation of $u$, $x$’s decendent, is associated with an instantiation of $y$, this is not known for sure. Therefore a new descendent $v$ of $y$ must built if $x$ is mirrored down to $d$’s level. Similarly, although the nature of the $\forall - \exists!$ relation could mean that every instantiation of $v$, $y$’s descendent, is associated with at least one instantiation of $x$, this is not known for sure. Therefore a new descendent $v$ of $y$ must built if $x$ is mirrored down to $d$’s level.

But these general rules depend on other factors, such as whether, in a pair of quantificationally dependent nodes, the instantiations of one of the nodes are of a level less than $n - 1$, while the dependent instantiations of the other node are of a level greater than $n - 1$.

Using cardinality to model quantification dependencies proves convenient. The first step is to build a subset $s$ of $a$ with as unique instantiation $d$: $s$ is a $\text{spec}_-$ of $a$ where the $\text{spec}_-$ is qualified by $\mathcal{E}$, the set of events qualifying the $\text{spec}_-$ path from $a$ to $d$. $s$ is qualified by $\text{size}_-$ events with $\text{object}_-$ values equal to one or zero, for every quantification level $< n - 1$. Inheritance will occur first with respect to $s$, and its conclusions will be copied back to $d$.

Inheritance proceeds as a two level search, where every move at the ancestor’s level is accompanied by a move at $s$’s level. So at every point in the search there are two foci: $f_a$ and $f_s$ at $a$ and $s$’s level respectively. The search is somewhat greedy: rather than decide whether an arc is worth being reflected at $s$’s level, it decides whether a node that has already been reflected at $s$’s level should have been (and therefore whether the search should proceed onwards from this node).
An arc from \( f_a \) that has not yet been visited is copied down to \( f_s \)’s level. The copy only reflects quantification precisely since the treatment of sorts must ensure that all reflected concepts are uniquely defined. If the target \( t \) of this arc at \( f_a \)’s level has already been mirrored, the new arc is attached to the mirrored node, otherwise a new node is built for it. The new node is connected to \( t \) by a specific event qualified by \( \mathcal{E} \), and if the new node is an event, it is qualified by \( \mathcal{E} \) itself. Every time a new arc is reflected down to \( s \)'s level, all the cardinality implications of its quantifications are worked out.

The cardinality relations inferred from the arcs quantifications express explicitly the quantification dependencies. Only quantification dependencies such as \( \forall - \exists! \) or \( F \rightarrow F \) result in a values relations being inferred: the cardinality of \( A \) will be inferred as higher than that of \( B \) if \( A \) is linked to \( B \) by a \( \forall - \exists! \) arc; but the two cardinalities will not be linked if the arc is \( \forall - \forall \) or \( \forall - I \). If the cardinality of some instantiation of a node is known, then any instantiation known to have the same cardinality can be inferred to have that particular cardinality. This applies both to cardinality one (which \( s \)'s < \( n - 1 \)-level instantiations are assigned), and to cardinality zero (applied if the events \( \mathcal{E} \) contradict existing properties of the nodes being inherited).

Once all arcs of \( f_a \) have been mirrored to \( f_s \), a new focus must be found for the search to continue. Only nodes that have a quantification dependency related to one of \( s \)'s \( \geq n - 1 \)-level instantiations should be expanded. Nodes that are quantificationally related to \( s \)'s < \( n \) level-instantiations will be inferred to have cardinality one. Nodes that result in a contradiction will be have cardinality zero, and will not be inferred to \( d \). Therefore only nodes which have at least one level of quantification associated with a \( \geq n - 1 \)-level instantiation of \( s \) which has not a cardinality of one or zero, can be considered as a new focus. A node may not be a potential focus at one point of the search, but become one as the result of a different path reaching it from \( s \).

The search continues until no potential focus points are left. Once this happens, \( s \)'s structure must be mapped over to \( d \), and the resulting network’s sortal restrictions
must be computed. Mapping s’s structure over to d involves:

- Setting the cardinality of s’s n – 1 level to one: the quantifications of level
  n – 1 involved the n-level instantiations so their dependencies also determined
  the subnet to inherit to s. But, there is only one instantiation of level n – 1
  being considered.

- Determining which nodes of s’s subnet should be copied to d: all the nodes
  which obeyed the potential focus expandability test of above, and which
  have no level of quantification qualified by cardinality zero. d’s subnet will
  be connected to the ancestors of those nodes of s’s subnet which did not obey
  the focus expandability test but which do not have zero cardinality.

- Determining for each of s’s subnet’s nodes, which is the highest level of quan-
  tification related to any of s’s ≥ n level-instantiations. The concept built in
  d’s inherited subgraph will be of this level.

- Building the relevant set relation between each node reflected at d’s level and
  its ancestor at a’s level. These set relations are qualified by the events of E.

- Applying semantic integration to d’s new subnet to ensure that none of the
  inherited concepts existed previously, and to ensure the network is well inte-
  grated.

Given that quite amount of network can be inherited down to d’s level an imple-
mentation should ensure that it is able to inherit smaller chunks lazily.

6.4.6.2 Partial arcs

Partial arcs complicate inheritance somewhat. The method of inheriting given
above works for partial arcs.

\[(E_0,R): \{\Delta F\text{-subject}_{-FO}: \text{children}; \Delta V\text{-action}_{-IO}: \text{squashed};
\]
\[\Delta F\text{-object}_{-F\forall O}: \text{ants}\}

If Mary is an instance of the children in \(E_0\), then \(E_1\) is inherited to her:
\((E_1,R)\): \{\Delta I\text{-}subject\_IO: Mary; \Delta I\text{-}action\_IO: squashed; \\
\Delta I\text{-}object\_\forall O: Mary\_ants\}\n
Assume that something is known about ant\_27, an instance of Mary\_ants. Only 
\(E_1\)'s object\_ arc will be copied to s, since mapping arcs to s is greedy. Mapping 
s's subnet to d will keep this arc, resulting in 
\((E_1,R)\): \{\Delta I\text{-}subject\_IO: Mary; \Delta I\text{-}action\_IO: squashed; \\
\Delta I\text{-}object\_\forall O: Mary\_ants\} \Delta I\text{-}object\_\forall O: ants\_27\}\n
This is the correct solution since one can never assume in SemNet that the insts 
of a set are all its elements.

Similarly the algorithm copes with more complex cases where the arcs connected to 
s express a quantificational dependency from s's \(n - 1\) level and s's \(\geq n\) level. Ex-
amples are \(\exists! \forall!\) – \(\forall\) and \(\forall\forall - \exists!\) respectively, where the first \(\exists!\) (\(\forall\) respectively) 
is at level \(n - 1\).

6.4.6.3 Crossed quantification dependencies

Certain quantification dependencies such as \([F]F - F[F]\) cross over each other. 
Even in the case where the first \([F]\) is at level \(n - 1\), the algorithm outlined above 
produces the correct result. However in such cases, building set relations between 
the nodes built in s's subnet and their ancestors becomes more complex. For in-
stance:

\((E_0,R)\): \{\Delta F[F]\text{-}subject\_FO: x; ...\}\n
is inherited to x's one-instance subset s as:

\((E_1,R)\): \{\Delta F[F]\text{-}subject\_FO: s; ...\}\n
\((E_2,R)\): \{\Delta F\text{-}subject\_FO: E_0; \Delta\forall\text{-}action\_IO: spec.; \Delta F\text{-}object\_F\Delta: E_2\}\n
\((E_3,R)\): \{\Delta I\text{-}subject\_I\Delta: s; \Delta I\text{-}action\_IO: size.; \Delta I\text{-}object\_IO: 1\}\n
Notice that the spec. event \(E_2\) has \(F - F\) quantification! This comes from the 
fact \(d\) will be an inst. of \(x\), and the quantification of the relation from \(x\) to \(E_0\) 
crosses over itself \((F[F] - F[F])\): \(E_1\) has as many elements as its parent \(E_0\), but 
each of \(E_1\)'s elements is a subset of the corresponding element of \(E_0\). Because of 
\(E_3\), and the \([F]F - F[F]\) quantification, each element of \(E_1\) is known to have size. 
1. When mapped to \(d\)'s subnet, \(E_1\) becomes \(E_4\):
(E₄,R): \{\Delta F\text{-subject} \_ FO: d; \ldots\}

6.5 Type-Checking and Semantic Integration Assumptions

Semantic Integration and Type-Checking rely on additional constructs to achieve computational efficiency: depth and certain data-structures.

6.5.1 Depth

Type-checking and semantic integration benefit from a notion of the distance of each concept from typeless: depth. The rest of this section discusses how SemNet can be annotated by depth and how depth can be maintained.

6.5.1.1 The desired type of depth

Before deriving an algorithm to annotate depth from typeless, the desired type of depth must be discussed.

Assume first that the depth of a child is one plus the maximum of the depths of its parents. This achieves the desired effect, since each parent has a smaller depth than its children. However, consider the case when a new set is inserted between two previously existing sets. For instance, assume that the concept small\_car is to be inserted between existing sets car and small\_british\_car. small\_british\_car has as instances Mini and Robin\_Reliant. Assume further that depth\((n)\) gives the depth of concept \(n\). Then \(x = \text{depth}(\text{small\_british\_car}) = 1 + \max(a, b, c)\) and \(\text{depth}(\text{Mini}) = \text{depth}(\text{Robin\_Reliant}) = 1 + x\), where \(a = \text{depth}(\text{small\_thing})\), \(b = \text{depth}(\text{british\_thing})\), and \(c = \text{depth}(\text{car})\). When small\_car is inserted, its depth should be \(x + 1\): one plus the maximal depth of its parents. However this is the same depth as its new child small\_british\_car has. Thus the depth of
small_british_car must be recalculated, transferring the problem to Mini and Robin_Reliant. These in turn must then recalculate their depth. The insertion of a new set can thus, in the worst case, cause all its descendents recalculate their depth, which high up in the hierarchy can be disastrous!

A solution could be to add a much larger value, say 1000, as step between parent and child. Although this appears appealing, it can lead to a dramatic increase in the size of SemNet. Indeed, the price for associating each node of SemNet with a 2 byte digit, giving the depth range 0 to 65535 is around 2Mb if SemNet contains 2 million nodes. A depth of 1000 would limit the longest path from any node to typeless to 65 nodes! Increasing the range to 4 bytes, would give a range of 4 billion depths, allowing the longest path to be 4 million nodes, but would cost 4Mb for 2 million nodes.

The best compromise is needed, so that the range of the depth is not too expensive on the one hand, and the likelihood that depth calculations propagate to a node’s children are low on the other hand. If one assumes that a new node may be inserted at any level of the hierarchy with equal likelihood, sequences of concepts in the inheritance hierarchy should be spaced equally in depth. Further the likelihood that a node’s depth calculations propagate to its children is minimized by increasing the depth distance between the node and its children. Thus for a given range of values 0..R, if the longest path from a node to typeless is \( N \), the best spacing is \( R/N \). The value \( R \) should be set to depend on the number of intermediate sets likely to be introduced between any two sets: a network with two byte depths, will have problems with more than 8 intermediate sets being introduced if the longest depth from a node to typeless is 2048 nodes.

The range of the depth depends on the size and topology of the network. Although 65 thousand values seems small and 4 billion rather large, in practice only the first may be needed: a certain amount of compression is possible, since not all 4 billion values are likely to be needed. For instance, each node could be associated with 2 byte-key which indexes into a table encoding the 4 byte depth.
6.5.1.2 Annotating depth

Since most concepts are in the inheritance hierarchy, it is essential for an algorithm annotating depth from `typeless` to have not worse that linear complexity: SemNet is to express gigantic quantities of information. Less than linear complexity is clearly unachievable since every node must be reached to annotate it with its depth.

In order to annotate the depth of each node, it is necessary to know the length of the longest path \( N \) and the depth range \( R \). \( R \) is set when the KB is built, but \( N \) depends on the contents of the KB, and must be measured. This can be achieved using a breadth first search from `typeless`, since the length of the longest path is simply the number iterations required to traverse the whole graph, if each iteration replaces a set of nodes by their children starting at `typeless`.

For simplicity, in the following discussion `spec.` and `inst.` events will be called arcs, and the concepts they link nodes.

After determining \( N \) it is necessary to traverse the network again, so that each node is given as depth, the depth of the parent with maximal depth plus \( R/N \). Since the length of every path to a given node from `typeless` need not be equal, a breadth first search from `typeless` cannot perform this annotation: it would need to delay at every node which had a not yet annotated parent. In practice, breadth first search in a graph is augmented by a set of already visited nodes \( \mathcal{V} \) which ensures that previously processed nodes are not reprocessed. In this case, a node would only be added to \( \mathcal{V} \) when it is depth annotated: when all its parents are annotated, their maximal depth plus \( R/N \) is written to the node.

The above scheme is expensive as the network has to be traversed twice: SemNet is recorded as files each containing hundreds of nodes. Since there is no guarantee that connected nodes appear in the same file, in the worst case, a traversal of the network will require every file to be swapped as many times as it contains nodes. An algorithm allowing SemNet to be traversed in the order in which it is recorded would be ideal, but does not seem possible. In critical situations, a skeleton of the network could be extracted, expressing only `spec.` and `inst.` events as arcs to the
relevant nodes. This would alleviate the problem, as the skeleton could be in one pass reading of SemNet. It would however likely to be too big to fit in memory and also need be swapped – albeit less because of its reduced size. Such schemes are beyond the scope of this thesis. Instead, this thesis will restrict itself to the following algorithm which reduces the problem by only performing one breadth first search on the network.

The algorithm uses an array $T$ for temporary storage associating each node of the network with a value, and indexing via the noderef. In SemNet the bottom part of the noderef range is always used first, so the count of current free nodes is saved in SemNet’s data files. This count can be used to build the relevant size array. Each element has a range sufficient to represent twice the number of all noderefs plus one (currently 4 bytes are sufficient). If $n$ is a noderef, $T[n + 1]$ is the value associated with it. Initially all values of $T$ are set to zero. $N$, the length of the longest path is initially set to 0. The final data-structure is $P$, a shifted circular list of noderefs, used as a queue, and forming the heart of breadth first search. The value zero will be inserted between each generation of the search. So, initially $P = \{\texttypeless + 1, 0\}$. The algorithm then iterates:

1. A noderef $n$ is taken from $P$.
2. If $P = \emptyset$, the iteration terminates.
3. If $n = 0$, it is added to $P$, and $N$ is incremented.
4. If $n \neq 0$, the node for noderef $(n - 1)$ is looked up in SemNet. The noderefs of all of its children $C$ are found:
   - For each $i$ in $C$, if $T[i + 1] = 0$ add $i$ to $P$ ($i$ has not previously been visited).
   - For each $i$ in $C$, set $T[i + 1] = 2 \ast n$ ($2\ast$ makes the entry even: see next page).

Upon termination $N$ is the length of the longest path. Each element $T[n]$ ($n \neq 0$) references the node $n - 1$’s parent which is on the longest path to $\texttypeless$. If
two paths are of the same length, one of them at random is chosen. The reason for
this is that each element of \( T[n] \) is overwritten as many times as a node \((n - 1)\)
has parents. The last time it is overwritten will be by the path last to reach it.
Because the search is breadth first, this will be the longest path from the search
origin: typeless.

\( T \) is then processed again, to calculate the depths. Evaluating the depth of a node
\( n \) proceeds as:

- If \( T[n + 1] \) is zero, \( n \) is not a node of SemNet.

- If \( T[n + 1] \) is odd, it is already a depth, so the algorithm returns \( T[n + 1] \).

- If \( x = T[n + 1] \) is even: it is a reference to another depth. It is evaluated by
evaluating \( T[x] \), setting \( T[n + 1] \) to \((T[x] + 2R/N)\), and returning \( T[n + 1] \).

\( T \) is processed by evaluating every one of its entries. This is a simple one pass
operation: for each node \( n \) of SemNet.

- If \( T[n] \) is zero or odd, the algorithm iterates.

- Otherwise, \( T[n] \) is evaluated.

This is basically a single pass, involving only \( V \) calculations: \( V \) is the number
of nodes in SemNet. It can be conducted simultaneously to saving the SemNet
files, and compressing the depth values into fewer bytes for instance: whether this
compression is possible will depend on the longest path \( N \).

The complexity is thus \( O(M) \) for the breadth first search, since each arc is traversed
only once; and \( O(V) \) for the processing of \( T \) and recording it in SemNet files:
\( O(M + V) \) total complexity, where \( M \) is the number of inst. or spec. events, and
\( V \) is the total number of nodes in SemNet. In the really large, swapping may prove
to be a bad problem requiring additional work. Currently, however, the packed
SemNet easily fits into 8Mb.
6.5.1.3 Maintaining depth

Each time a new node is built, it must be given a depth. The range from which it can be given a depth is determined by the difference between its children's depth and its parents' depth. To be sure that it is given a correct depth, all its parents and children must be known. This means depth assignment can only be performed after semantic integration. In the mean-time it is given an ignore value, which effectively means all algorithms, semantic integration itself, cannot rely on its depth value, and must use either that of its known children to limit downward searches, or that of its known parents to limit upward searches.

Once the node's parents and children are known, the value assigned to it is the maximal depth of any of its parents plus half the depth range between it and its children. If a depth compression scheme is in force (see 6.5.1.1 (p. 177)), if a value closely approximating the new depth exists, it may be used instead, provided it does not exceed the depth of the node's children, and the node's parents depth does not exceed it.

6.5.1.4 Dealing with depth degeneration

Despite a high depth spacing, the introduction of intermediate sets can still cause problems. There are various ways of dealing with this problem.

The range of the depth values can be increased by using family types. Depth is then a pair \((f, d)\) where \(f\) is the node's family type and \(d\), the depth from the top family node. A complete order must be established between all family types to be used by depth calculations: if a node is the intersection of two others \(a\) and \(b\), where there is no order between the family of \(a\) and the family of \(b\), the algorithm cannot choose in a consistent way. Any cases where an order between family types already exists is used for the depth order, since the existing order follows the hierarchy.

The ignore value can be used, which states that the node does not have an explicit depth. Its depth is slightly greater than that of its parents, and slightly less than that of its children. This corresponds to locally reverting to the depth-
annotationless network. It speeds up node construction, but destroys the efficiency gains made by algorithms using depth. This type of solution can improve the robustness of a production quality system, allowing it to cope with the odd occurrence of range spacing overflow when the system is used intensively.

Finally, part or all of SemNet could be depth-annotated again. This is likely to produce a pause in processing, similar to that of garbage collection. The exact length will depend on the size of network chunk being reannotated, and may involve non-obvious costs. For instance, most of the network is likely to be swapped from disk, adding swapping costs. Similarly, in an implementation such as LOLITA, using a packed StaticNet and a DynamicNet for changes, remapping may result in an excessive growth of the DynamicNet, slowing all future accesses to the network. The ideal case for reannotation is during dead periods, such as at night.

6.5.2 Data-structures

This section describes two data-structures that are heavily used in the rest of this chapter: the shifted circular list, and the BIB tree.

6.5.2.1 Shifted Circular Lists

- Shifted Circular Lists: The idea

Linked lists provide cheap flexible storage to sequences of data. They provide $O(N/2)$ average access cost, where accessing the first elements is substantially cheaper than accessing the final elements. However, some algorithms benefit from fast access to both initial and final elements. Shifted circular lists provide such access at a lower access cost, both in terms of memory and processing, than doubly linked lists.

A circular list lacks a final "NULL" element. Instead the last element is linked to the first element. In standard circular lists, references to the list encode the address in memory of its first element. If the list is empty, they encode an invalid address, typically zero. An element is the first of the list, if its address equals the
value of the list reference.

A shifted circular list differs to standard circular lists in that the list reference gives the address of the last element of the list rather than the first. As the last element is linked to the first, the first is only one step further away.

- **Shifted Circular Lists: Two applications o Queues**

Queues are useful data structures, used for instance to ensure breadth first traversal of a search space. In a queue elements are taken from the front of the list and added to its end. If the queue is implemented with a classical list, the cost of these operations depends on the number of elements in the list. With a shifted circular list, both operations are of constant complexity:

- to add element $e$ to the end of the list, insert $e$ between the last element of the list and the first, and change the list reference to $e$'s address.

- to remove element $e$ from the beginning of the list, make the last element of the list refer to the element following $e$.

- **Largest element of a sorted list**

Lists can be sorted with respect to a value associated with their elements. In particular consider the case of lists that must always be kept sorted, but for which the values of elements may vary. If the value of a given element may increase (but never decrease) during processing, as more evidence is gathered, say, it possible to avoid resorting the list at each value change. Since it is cheaper to traverse a singly linked list from its first element to its last, it is cheaper to sort the list from its smallest value to its largest: the element whose value has changed only needs to be moved along the list. Even if references are kept to the list's elements, rather than looking them up each time, the cost of resorting is not independent from the length of the list. However, snooping on the current maximum value is of constant access complexity, which is important if the termination condition of the client algorithm depends on the current maximal value. For example, a reasoning client could be searching for evidence of a minimal certainty level to support some proposition.
6.5.2.2 BIB Trees

Binary trees provide a cheap way of associating data with indexes which have a wide range of possible values. To maintain maximal efficiency, these trees must always be balanced: Data is accessed in trees by searching the unique path associated with the index of each piece of information. Each node of a tree dominates a search space constituted of all the information below it in the tree. In a binary tree, each node is associated with two arcs. Traversing an arc corresponds to choosing the search space associated with the node at the arcs' destination. If the search space of each node is divided equally by its two arcs, the number of search space reduction steps, i.e. the path through the tree, will be the smallest possible for every element of the tree. In such balanced trees, the length of the path to traverse is proportional to $\log(N)$, where $N$ is the number of elements in the tree.

The key to maintaining efficiency is therefore is finding some way of organizing information in such a way that each node of the tree dominates only half the search space its parent dominates. This is where balancing schemes come in: AVL trees, splay trees, and so on, are all rebalanced each time a piece of information is added. I.e. they do not assume that the data has any implicit order that they could exploit. Instead they impose their own ordering scheme on the data, regardless. Skip lists [Pugh 90] do not assume any order, but use a probabilistic scheme to create an order. In all cases, this is an overhead. If the data does have an implicit order that can be used, trees can be built in such a way that no overheads are involved. This is the basic idea behind BIB trees.

- **BIB Trees: The idea**

IB trees, the class of trees to which BIB trees belong is first discussed, followed by a presentation of BIB trees.

- **Building IB trees**

Indexed binary (IB) trees store data indexed by an integer. Since the index is an integer, it is a binary string of some fixed length. Binary strings implicitly form a binary tree, where each digit corresponds to a new layer of branches. A binary tree
can mirror the implicit binary tree of the binary string, by building a branch for each digit in the binary string: a left branch for a zero digit and a right branch for a one. However the depth of the resulting tree is not proportional to the amount of data in the tree, but to the length of the binary string. This is because many of the nodes in the resulting tree only have one descendent. But the purpose of having left and right branches at different levels was to repeatedly divide the search space of the data into two. Where the search space cannot be divided into two, arcs are useless.

The tree's depth can be reduced if no nodes with one child are built. Sequences of such nodes occur when all the indexes of the elements constituting the search space they dominate share a common bit string. Such bit strings result in the same path being built for all of them. The first branch of this common path does distinguish the indexes from others, but all the subsequent ones serve no purpose so may not be built. Since IB trees are random-access structures, a new index may be added breaking up what was until then a common path. Thus, the information about compressed common paths must be retained. This is achieved by associating a binary string to each node of the IB tree, which expresses the part of the path from the node's parent to the node which was compressed out of the branching structure. Thus, if a tree contains only two indexes "a1cd" and "a0cd", the root node would have the string "a", and both leaf nodes would have the string "cd" attached to them.

- **BIB trees**

The IB tree does not guarantee that all of its branches are have \( \log(N) \) depth. It only reflects the distribution of search space associated with each digit of the index binary string: an implicit order in the data.

Since the branches near the root of the IB tree divide most of the search space up, they are the most critical: they will affect the indexing of all of the data. They are therefore required to divide the search-space as equally as possible. The question is therefore whether any digits of the index binary string divides the search space equally. These digits should be associated with the branches near the root node.
To determine this implicit order, one needs to know how the data is constructed and deleted.

The indexes of BIB trees will be the noderefs of SemNet. When new nodes are allocated in SemNet, their noderefs are usually consecutive numbers. Thus only the lower part of the range of possible noderefs is occupied, meaning that the most significant bit which distinguishes noderefs will distinguish between a fully used range and a partially used range. Since the second range will on average, only be half full, it will have on average 66% of the time zero as value and 33% of the time one as value. Since the noderefs are allocated as a consecutive series of numbers, the least significant bit will have zero as value 50% of the time, and one 50% of the time, assuming deletions are not biased to either even or odd noderefs, for which there is no evidence. Similarly, for the least but one significant bit, and so on with a bias\(^2\) slowly increasing as the chosen digit tends to the most significant bit. Thus, by using as index the reversed noderef binary string, one obtains a much more balanced tree than by directly using the noderef's binary string as index. This forms the backwards indexed binary (BIB) tree.

**BIB tree complexity**

BIB trees do not guarantee low tree depth, and therefore logarithmic access time. However, because non-leaf nodes always express a split of the search space, there are at most \(N - 1\) of them. This means that at worst, the path depth will be \(\min(N - 1, R)^3\) where \(R\) is the length of all binary string indexes, say 32. The worst average case is \(\frac{N+1}{2}\) for \(N \leq 2R - 1\). At best there will be \(2^{\log_2(N)+1} - 1\) non-leaf nodes, and path depth of \(\log_2(N)\) as for fully balanced trees.

The question is what the average depth is. This can be answered by looking at very similar structures. Binary radix ([Sedgewick 93]) tries differ only slightly from IB trees: rather than maintaining a copy of the skipped common sections of indexes due to path compression, they associate every node with a bit index. The bit index states on which bit of the index the decision of which child to choose (left or

---

\(^2\)Bias due to contiguous ranges of nodes being deleted after some operations.

\(^3\)The worst case is when no compression is possible, and no balancing: each non-leaf node has a left arc to a leaf node and a right arc to a non-leaf node.
right) is made. The advantage is less memory usage per node: only a bit index to
maintain rather than a common section of index and its length. This is particularly
useful in the case of very long indexes, such as strings, since maintaining copies
of common sections is very expensive. In binary radix tries the complexity of
access is, for random input, $O(ln(N))$. The number of nodes required for storage
is $N/ln(2) \approx 1.44N$. IB trees, being little different, should have similar behaviour.
Even better trees based on radix methods exist, such as Patricia ([Sedgewick 93]),
but unfortunately they are not easily implemented in Haskell. Patricia maintains
the $O(ln(N))$ storage requirement, but achieves a storage requirement of only $N$
nodes.

Why then use IB trees rather than binary radix tries? BIB trees may be deeper
than self-balancing trees such as AVL trees, so one might think that testing the
tree to see if it contains some index would take more steps for BIB trees than
for self-balancing trees. However, the opposite is true. The BIB tree is traversed
following the instructions expressed by the binary string index. As soon as the
instructions fail to match the topology of the tree, one knows that the index has
no entry in the tree. In self-balancing trees, information is ordered according to
some ordering relation such as less than, which means that to find that an index
is not in the tree, a path from the root to a leaf must be traversed. Thus, failing
to find a match takes $log(N)$ operations in self-balancing trees, where it takes less
than $\approx log(N)$ for BIB trees. In binary radix tries, the fact the compressed com-
mon parts of the binary index string are not recorded means that the search may
continue beyond the divergence point. BIB trees were tailor designed to imple-
ment LOLITA’s DynamicNet. Initially the semantic net is a read-only compressed
data-structure accessed from C. Any changes to the network are recorded in the
DynamicNet so lookups for nodes it does not contain should fail early. Since this
is performance critical, experiments were run on AVL trees, binary radix trees and
BIB trees. BIB trees won.
6.6 Type-Checking

Type checking ensures that every instantiation of an event satisfies the definition of its template event. It also ensures there is an explicit set relation between the instantiation's subject and object and its template's subject and object respectively. Because type-checking's results constrain semantic integration's search space, it is performed first. Type checking thus provides a basic level of consistency, and some level of disambiguation.

Type-checking compares the proposed subject and object of an event with those of the event's template event. Finding the template event is easily achieved, since only the template event is connected to the action node by an action of arc. The treatment of subject and object arcs is the same, so only one shall be discussed: the subject arc.

The template event specifies the type of subject events of its type can take. The issue is therefore to check whether the subject $s_p$ of the proposed event $e_p$ fits this requirement. The template event $e_t$ defines its subject set $s_t$, so the proposed subject must fit all other restrictions on $s_t$ given by $s_t$'s definitional events, be they inherited or directly connected. In practice, $s_t$ will only be defined by $e_t$ and will have only one definitional parent, since the set of characteristics a concept must have to be the subject of an event of type $e_t$ may occur without the event actually being the subject of an $e_t$ type event. Thus, the restrictions on $s_t$ will usually be given by $s_t$'s parent.

6.6.1 Simple compatibility

The simplest case occurs when $s_p$ is a descendant of $s_t$'s parent. For instance, if John is a descendent of Humans and $e_p$ is "John owns a car". The template event for owning is:  

\footnote{Ignoring issues such as multilevel quantification, that the current implementation does not support.}
(e, R): \{ OV\text{-subject}. ∃!\Delta: Owners; \Delta V\text{-action}.IO: own;

OV\text{-object}. ∃!\Delta: Possessions \}

where Owners is a definitional specialization of Humans, and Possessions is a
definitional specialization of inanimate entities. Since the concept John is a
descendant of the concept with all of s_t's restrictions (Human), it fits all the re-
quirements to be a subject of an event of type e_t. Thus the proposed event e_p is
accepted, and a spec. (or inst.) event is built between s_t and s_p. Another case
fitting within this category is if s_p is already involved in an event of type e_t: s_p will
in this case be a descendant of s_t, trivially satisfying the requirements.

Since in SemNet, nodes tend to have more children than parents, it is more efficient
to search up the inheritance hierarchy than down it. Thus, the search to establish
this first case starts at s_p and proceeds upwards. A breadth search is used, asso-
ciated with a set \( \mathcal{V} \) of previously visited nodes. Initially \( \mathcal{V} = \emptyset \), \( \mathcal{P} = \{ s_p \} \). The
search then iterates: a node \( n \) is taken out of \( \mathcal{P} \). If no node was available, the
search fails with a failure. Otherwise, if \( n \) is in \( \mathcal{V} \), the search iterates. If it was not
in \( \mathcal{V} \), \( n \) is checked to see if it is \( s_t \)'s parent. If it is the search terminates with a
success. Otherwise \( n \) is checked to see if it violates the node condition: if it does,
the search iterates. If \( n \) satisfies the node condition, \( n \) is placed in \( \mathcal{V} \), and the set
of its parents are determined and added to \( \mathcal{P} \). The node condition ensures that the
search does not wander up to typeless: if \( n \) is of a family type compatible, and
not above the family of \( s_t \), \( n \) satisfies the node condition. The search space can be
further limited by adding the condition that \( n \)'s depth is not smaller than that of
\( s_t \).

\( \mathcal{P} \) can be implemented as a shifted circular list allowing its initial elements to be
easily removed and elements added to its end, providing the breadth first element of
the search. Because \( \mathcal{V} \) ensures that nodes are only traversed once, the complexity
of this algorithm is \( O(N) \) where \( N \) is the number of arcs\(^5\) between \( s_p \) and all
nodes that are its ancestors of a compatible family type (and not above the region
determined by \( s_t \)'s family, or with depth not above \( s_t \)).

\(^5\)In fact spec. or inst. events
6.6.2 Underdefined \( s_p \)

The second case is if \( s_p \) is a descendent of one of \( s_t \)'s ancestors. Unless \( s_p \) belongs to a set explicitly stated to be disjoint from \( s_t \)'s parent, \( s_p \) can have all the features of \( s_t \). Thus the question is to determine whether \( s_p \) is disjoint from \( s_t \).

Two sets can be disjoint in three ways, rendering this computation complex:

- They, or their ancestors, can be subject.s of an antonym. event.
- Their definitional intersection can be empty.
- They can differ in a definitional event: one of them can have the event, and the other can have its negation by absence.

Why is the negation by antonym case not listed? Negation by antonym only states that the events are antonyms: one cannot both like and dislike the same person at the same time. It does not state that the same person cannot like one person and dislike another at the same time. Thus the subject.s and object.s of antonym relations are neither antonyms, nor disjoint.

The negation by absence case reduces to the antonym case. Consider owning. The set of owners has an antonym \( a \) with respect to typeless: all things that do not own something. Every concept known not to own something is a subset/element of \( a \). No membership is predicated of either set, if it is not known whether or not the concept is an owner. Thus, only concepts for whom an absence of every type of owning has been stated, are spec. or inst.s of \( a \).

In effect, solving the underdefined \( s_p \) case adds information to \( s_p \). For instance the "she" of the sentence "She owns a motorbike" will be interpreted by semantic analysis as "a female creature". In SemNet's ontology, each of the following concepts is a superset of the subsequent concept: sexed.creatures, mammals, primates, apes, humans, owners. sexed.creatures is also the superset of male.creatures and female.creatures, which are antonyms with respect to sexed.creatures. humans is also antonymous to all other types of apes with respect apes. Because in this ontology, female.creatures is not disjoint from owners, it is possible for \( s_p \) to
be a descendent of both. As a result, the ‘‘she’’ inherits all the characteristics of owners (such as being human). Semantic integration will make this effect explicit, connecting \( s_p \) to its implicit ancestor \textit{women} \(^6\).

At this point, LOLITA does not include any reasoning about empty sets, very little work on \textit{antonym}s, and does not include the representation of negation by absence since the representation has only single levelled quantification. As a result, only the flavour of an algorithm can be given. Since it is the differences between \( s_t \) and \( s_p \) which will determine whether \( s_p \) can be a subset of \( s_t \), and since nodes inherit characteristics from their ancestors, all the ancestors \( s_t \) and \( s_p \) do not share must be examined. Let those unshared ancestors of \( s_t \) be \( \mathcal{U}_t \) and those of \( s_p \) be \( \mathcal{U}_p \). If \( x \in \mathcal{U}_t \) and \( y \in \mathcal{U}_p \) have a definitional intersection of observational size zero, and are not qualified by another definitional event (including set intersection with another concept\(^7\)), then the check fails. Similarly, if \( x \) and \( y \) are subjects of any antonym relation, the check fails. Otherwise it succeeds. Failure means that \( s_p \) cannot be a subset of \( s_t \), and \( e_p \) cannot be built as a literal event.

This type of algorithm can be built as a lowest first algorithm using depth information. Using a set of visited nodes associated with the node from which originated the path that visited them, simultaneously achieves detection of intersection of paths from \( s_t \) and \( s_p \) (both at common ancestors, and at sized zero subsets) and ensuring each \textit{spec.} or \textit{inst.} event is traversed only once. The resulting complexity is on average \( O(N \cdot \text{log}(V)) \) where \( N \approx (c + a)N' \) is the number of events to traverse (\textit{inst.}, \textit{spec.}, and \textit{antonym} \(^8\)), \( c \) is the average number of children each node of SemNet has, and \( a \) is the average number of antonyms to which each node is connected. \( N' \) is the number of \textit{spec.} and \textit{inst.} events connecting the nodes of \( \mathcal{U}_t \cup \mathcal{U}_p \). Similarly \( V \approx (c + a)V' \) where \( V' \) is the number of elements of \( \mathcal{U}_t \cup \mathcal{U}_p \).

The \( \text{log}(V) \) factor comes from the access time to a tree implementing \( \mathcal{V} \), and this access occurs each time a \textit{inst.}, \textit{spec.} or \textit{antonym} event is traversed: \( N \) times.

The algorithm is therefore of low complexity.

\(^{6}\)where \textit{women} is defined as female people, with no consideration of age.

\(^{7}\)Assuming a network without redundant information such as \textit{woman} is a subset of \textit{typeless!}

\(^{8}\)In fact antonyms are not traversed, just added as common nodes to \( \mathcal{V} \).
If no depth information is available, tricks of the type described in C.1 (p. C-1),
can help reduce the search space.

As an interim solution, by encoding knowledge explicitly, the first case of type-
checking can achieve the desired behaviour: "she owns a motorbike": women is the
intersection of female.creature and humans, and an explicit women.owners set
was added to SemNet as the intersection of owners and women.

6.6.3 Type-Checking is not the only consistency check

Type checking is by no means the only type of consistency checking that should be
performed.

For instance, the statements "John owns a sheep" and "John owns no animals"
will not be flagged as inconsistent. Such statements are easiest to detect during or
after semantic integration, when John's sheep will be built as an instance of John's
animals. What type-checking will determine are inconsistencies of the type "John
is deaf" and "John heard ...".

Similarly, the statement "John and Bill ate my meal" should invoke a quantification-
clash with its template if the eating template is defined as:

\[(E_0,R): \{OV\text{-subject.}\exists!\Delta: \text{animal}_1; \Delta V\text{-action.}\text{-}I\Delta: \text{eat};\]
\[\text{OF\text{-}object.}\text{-}F\Delta: \text{meals}_1\}\]

which states that there is only one meal per eating event. The type-checking al-
gorithms will not fail, so the building of the inst. event between John and Bill's
eating events and the template should be checked and fail.

\[\text{\textsuperscript{9}Hence LOLITA should infer that John and Bill each ate part of my meal, and none was left for me.}\]
6.7 Semantic Integration

6.7.1 Introduction

6.7.1.1 The need for classification

SemNet requires that every algorithm builds the concepts it introduces as a spec. of some other concepts: every concept must have an ancestor or a descendent. However this does not guarantee that every concept has been explicitly connected to all the concepts which are its ancestors and descendents according to its definition.

Take for instance, the concepts "Plastic bags" and "Blue plastic bags" built as intersections of the concepts blue things, plastic things and bags. If "Plastic bags" is added to a KB containing "Blue plastic bags" one would like to see a spec. from "Plastic bags" to "Blue plastic bags". However, since the concepts were built as intersections of other concepts, it will not be there. Algorithms, such as inheritance, which rely on traversing a path of explicit spec. relations in a unique direction (up or down) will fail to make all the expected inferences. This means that every concept added to the semantic network must be integrated so that all its ancestors and all its descendents are reachable by the unique direction inheritance hierarchy search. Classification ensures this criterion is met. The cost of this expensive one-off operation is amortized by the speed gained by all algorithms which use the inheritance hierarchy. Since these algorithms are used very often, the cost is recovered quickly.

SemNet also requires that every concept be unique. I.e. every new concept must be a new combination of restrictions\(^{10}\) of typeless. Since building a new concept as a subset of another does not guarantee that all the concepts' ancestors and descendents are connected to it, it also cannot guarantee that the concept is unique. Classification's other task is to guarantee conceptual uniqueness.

\(^{10}\)Restrictions here is a generic term referring to events connected to a concept, but also for events, the subject., object. and action. arcs.
6.7.1.2 Problem Breakdown

When a new concept is added to SemNet, its definition might entail that other concepts in the KB are its ancestors or descendants. A search that considers only one direction (up or down) the inheritance hierarchy might not be able to reach some of these concepts. The concepts that cannot be reached are the new concept’s implicit ancestors or implicit descendents. Classification’s task is to determine the minimal set of its ancestors and descendents which ensure that it is connected explicitly to all of them by one or more set relations. A minimal set is required since every relation in SemNet costs memory.

Classification deals only with concepts’ definitions: facts known about a concept do not restrict it. Thus, one cannot take the fact that a concept a is qualified by observational arcs which could apply to the new concept n, to mean that a is an implicit ancestor of n.

Classification can be divided into two problems: semantic integration and subsumption. Subsumption decides whether a concept is subsumed by another, or is disjoint from it. Semantic Integration roams the semantic network to find the implicit set relations and make them explicit, from time to time asking subsumption to determine whether a concept is subsumed by another. Subsumption is thus a reasoning task, whereas semantic integration is a network building task.

Subsumption involves more complex reasoning than might at first appear. For instance, take “Every person who rents a flat in Paris is rich” and “Jacques has rented a flat in the 5th arrondissement of Paris”: Jacques’ renting event must be integrated as a descendendent of the former type of event, even if it is represented as “Jacques has rented a flat which is in a place (which is the 5th arrondissement) which is in another place (Paris)”. This can be extended, “Every person who rented a flat in Paris in the 1980s was very rich”: now Jacques’ renting event may not be a descendendent. Specific reasoning is required for each type of event meta-property, such as time, cause, source, size and other values (belief, certainty), equality (synonyms), logical events (and, or) and so on. By requesting the restriction information from the reasoning algorithms dedicated to each type of meta-property, in
effect semantic integration encodes their knowledge into the inheritance hierarchy. This ensures agreement between the results of semantic integration (and thus inheritance) and the expectations of the various types of reasoning. Since subsumption depends on every form of reasoning used by a SemNet based system, it is beyond the scope of this thesis. Instead, algorithms for subsumption will be assumed to exist.

The semantic integration algorithms presented in this section rely on the inheritance hierarchy being depth annotated, although (less efficient) versions of the algorithms are described in C.1 (p. C-1). All complexities discussed here assume depth annotation. The complexity of the subsumption algorithm is not included in the calculation, although it may prove the dominant term. For this reason, much effort is devoted to limiting as much as possible the search space considered by the semantic integration algorithms. The algorithms also assume that they start off with only one node to integrate in an otherwise perfectly integrated network.

There are two kinds of classification: upwards and downwards. Upwards classification connects the concept to its implicit ancestors, while downwards to its descendents.

The downwards semantic integration algorithm determines a set of nodes which must be the ancestors of all implicit descendents of the node to integrate: the red nodes. It then performs a highest first search downwards to find every node which is the descendent of all the red nodes. Because there can be many of them, the search down a particular set of paths actually stops as soon as a descendent of all the red nodes is found, since all this node's descendents are trivially those of the node to integrate. Thus downwards integration makes no call to the subsumption algorithm, if the node to integrate has no non-inherited definitional restrictions. This means that its complexity is known: \( O(N_0) \), where \( N_0 \) is the number of set relations that are traversed. i.e. possibly less than the number of set relations connecting every node which has a red node as ancestor, depending on the topology of the network. C.2 (p. C-11) shows this to be less than \( O(\log(N)) \) in the average case, where \( N \) is the number of concepts in the K.B. However, since the
worst case is \( O(N) \) – clearly a problem if the K.B. contains millions of concepts – an alternative is presented for real time operation. This alternative is based on the fact the inheritance hierarchy is mostly searched upwards, which allows such algorithms (as the inheritance algorithm) to detect implicit ancestors due to downwards inheritance not being performed, with no increase in the time complexity: only a constant increase in the processing time per concept visited, depending on the number of non-integrated concepts. The penalty can be removed by performing full integration during the dead period each night.

Upwards semantic integration divides its search space into three components, which each need to be handled differently: the nodes above the orange node, the beige nodes and the khaki nodes. These three types of nodes are possible ancestors of implicit ancestors of the node to be integrated. The search thus proceeds downwards from these nodes through uncoloured nodes until the subsumption checks with the node to integrate fail. The nodes before the failures are the node to integrate's most specific implicit ancestors. The number of subsumption checks can be minimized to only the occasions when they are actually needed, for instance by using a highest first search downwards and more conditions. The complexity of the algorithms determining the search space and exploring it accumulate to \( O(N \log(N)) \)\(^{11}\) excluding the cost of subsumption, where \( N \) is the number of set relations they traverse, which depends on the topology of the network.

In NL single node integration does occur (e.g. "I like all white sheep", where white sheep had previously been introduced), but is relatively rare. For instance, in "Poor men love their wives more than rich men", there are two set of wives: those of poor men and those of rich men. These are defined by "wifing" events which are themselves defined by their subject., object., and action.: the sets and the events cannot be added independently to SemNet so must be integrated simultaneously. Multiple node integration proves more complex than single node integration because of the possible mutual definitional dependencies between concepts. The adopted solution is to isolate the groups of concepts which are mutually definition-

\(^{11}\)Excluding above orange node integration – for which no complexity result is available, see 6.7.2.2 (p. 221).
ally dependent, so that the groups of concepts to integrate simultaneously is as small as possible. These groups are then added to the network, and each concept is integrated singly – ignoring the fact the others are not yet correctly integrated. This results is a sort of partial integration, which has added information which may mean that integrating the concepts again would result in further integration. Thus the concepts are integrated again and again, until no change occurs. The whole process is a fixed point iteration.

Despite the complexity of the algorithms discussed, they actually deal with a subset of the full representation. In particular, only a single level of quantification is considered, synonyms, antonyms, belief issues and more are not considered. This is either because these aspects do not affect the main thrust of the argument, or because not enough experience is available since the current implementation lacks multiple levels of quantification, for instance.

Although the semantic integration algorithms have good time complexity, this depends critically on the complexity of subsumption. Much research into KL-ONE has shown that subsumption is non polynomial (or worse) for many representation languages. Reasons why SemNet does not share the same causes of bad complexity as KL-ONE are discussed in 6.7.5 (p. 244). However, it is quite possible that SemNet has other such causes, or even – with the introduction of cardinality, for instance – is in some cases undecidable. This need not be as critical as it sounds, as the task is to simulate successful human behaviour. It is not clear that people are always able to relate sets and their supersets, without prior hints. Hindsight – i.e. checking the validity of a statement – is not the problem, the problem is foresight. It may therefore prove acceptable to time out if semantic integration takes too long, so that real time response is still possible.

6.7.1.3 Disclaimer and Reading Conventions

The algorithms that will be discussed here are partial with respect to the full representation: the implemented representation does not include multiple levels of quantification, and the algorithm does not consider this issue. It does however
consider sets and instances, so it is hoped that the principles at work here will also apply to the full problem. For simplicity, the algorithms treat instances as sets of observed size 1, following their definition 5.4.2.3 (p. 140): hence, only integration of sets will be discussed. Similarly, the role of arbitrary quantification, synonyms, non-literal concepts, hypothetical or otherwise qualified set relations will not be discussed: these issues only complicate the presentation of the problem, but do not present any inherent difficulties. Partial arcs are also not considered here.

To simplify discussions of positions in the inheritance hierarchy, the following conventions are adopted: The term “set relation” refers to a spec. event or an inst. event. typeless is at the top of the hierarchy. Instances are at the bottom of the hierarchy. A concept a is said to be above a concept b if the hierarchy distance from typeless to a is lesser than that to b. The hierarchy distance is the minimal number of set relations that must be traversed to reach a given concept from typeless. Similarly traversing the network upwards is to traverse one or more set relations such that each step leads to a concept above the previous one. Below and downwards are the converse relations of above and upwards. A parent is a concept above another and explicitly linked to it by a set relation. A concept’s ancestor is a concept which is either its parent or the parent of one of its ancestors. Child and descendent are the converse relations of parent and ancestor respectively.

Many of the algorithms process path statuses. A path with definitional status is a path consisting purely of definitional set relations linking a concept to one of its ancestors. A path with observational status is a path of set relations, one or more of which have observational status, linking a concept to one of its ancestors. When comparing two statuses, a definitional status will be said “better” than an observational status. Two definitional statuses or two observational statuses will be said “equal”.

Discussions of integration easily become very complicated. To try to simplify communication, the different types of nodes will be discussed as if they had a colour. The table 6.1 (p. 199) provides an easy reference to the correspondences between colours and types.
<table>
<thead>
<tr>
<th>Colour</th>
<th>Abbreviation</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>White</td>
<td></td>
<td>Untouched</td>
</tr>
<tr>
<td>Black</td>
<td>bn</td>
<td>To integrate</td>
</tr>
<tr>
<td>Red</td>
<td>$\mathcal{R}$</td>
<td>First key nodes on definitional paths from $bn$ to typeless</td>
</tr>
<tr>
<td>Grey</td>
<td></td>
<td>First key nodes on paths from $bn$ to typeless</td>
</tr>
<tr>
<td>Orange</td>
<td></td>
<td>Intersection of all Paths of $bn$ to typeless</td>
</tr>
<tr>
<td>Brown</td>
<td></td>
<td>Nodes between Orange Node and $bn$</td>
</tr>
<tr>
<td>Beige</td>
<td></td>
<td>Brown nodes above the grey nodes</td>
</tr>
<tr>
<td>Khaki</td>
<td></td>
<td>Brown nodes below the grey nodes</td>
</tr>
</tbody>
</table>

Table 6.1: Types of nodes in Semantic Integration

At the initial stage of semantic integration, there is a node to be integrated and an integrated semantic net. The node to be integrated will be coloured black and all other nodes white (also known as uncoloured).

For clarity, downward integration is presented first as it introduces concepts needed for the discussion of upwards integration. In practice however, upward integration would be performed first.

### 6.7.2 Single Node Integration

Semantic integration can be substantially optimized if the notion of depth from typeless, discussed in the type checking section, is used. The optimization comes from the substantial simplification of the problem of integration afforded by depth. Because this simplification makes semantic integration easier to explain, the rest of this section will assume that SemNet is annotated by depth. However equivalent slightly more expensive algorithms that do not require depth are presented in C.1 (p. C-1).

#### 6.7.2.1 Downwards Integration

- **Downwards integration: sketching an algorithm**

The problem of downward integration is to find all implicit descendents of the black node, that is to say all the nodes that could be the black node’s descendents. All descendents of the black node must have the same definitional restrictions as the
black node. Thus, in a fully integrated network, they must be explicit descendents of all of the black node's ancestors. Since they will be further restricted than the black node, any node which is a descendent of all of the black node's ancestors is an implicit descendent of the black node.

Thus, detecting the implicit descendents involves testing all the descendents of all the black node's ancestors. As soon as a descendent is disjoint from the black node, neither it nor its descendents can be implicit descendents of the black node. If it is decided that a descendent is an implicit descendent of the black node, then all its descendents are also implicit descendents of the black node: the downwards search can stop at this level.

Since the black node has many ancestors, limiting the number of that must be searched down from will improve efficiency. Such a limited set can be found: red nodes.

What are red nodes? Every concept in SemNet is defined by definitional restrictions: some introduce a new kind of restriction for this definition, whereas others build a restriction by combining the definitional restrictions of their parents. Consider the first kind of nodes and call them key nodes. All non-key nodes are formed by combining (i.e. intersecting) key nodes. This means that along every path upwards from the black node to typeless there will at least one key node: since all concepts are unique, unless the black node is typeless, it must be restricted. Take the first key node encountered along a path from the black node towards typeless, and colour it red, if the path from it to the black node is constituted of an uninterrupted chain of definitional set relations.

Why do only red nodes need to be considered? All definitional paths upwards to typeless from the black node go through a red node. Thus every definitional event the black node inherits from its ancestors is inherited through a red node. Since every implicit descendent of the black node must have all of the definitional restrictions of the black node's ancestors, every implicit descendent must be a descendent of a red node. But the red nodes are fully integrated, so all the implicit descendents of the black node will be explicit descendents of the red nodes. More-
over the descendents of the black node must all have all of its restrictions, so they
must be the descendents of all of the red nodes. Thus only the descendents of all of
the red nodes can be implicit descendents of the black node.\textsuperscript{12} This substantially
reduces the number of nodes that must be tested to see whether they are or not
more restricted than or disjoint from the black node.

The algorithm for downward integration is thus:

1. Find the red nodes.

2. Find the implicit descendents.

• Finding the red nodes

The precise algorithm starts with a set of nodes to expand \( E \), a set of parents \( P \),
and a set of red nodes \( R \). Initially, \( P = R = \emptyset \), and \( E = \{bn\} \) where \( bn \) is the
black node. The algorithm then iterates:

1. All of the definitional parents of each element of \( E \) are found, and those which
   are not elements of the set of red nodes \( R \) are placed in the set of parents \( P \).

2. Each element of the parent set \( P \) is taken out and tested to see if it is a key
   node: if it is, it is placed in the set of red nodes \( R \), otherwise it is placed in
   the set of nodes to expand \( E \).

The process iterates until no new elements are added to \( P \).

Just as determining whether a concept is more restricted than another, or is disjoint
from it, the detection of key nodes requires the use of many forms of specific
reasoning. If a concept is a key node, it is connected to at least one definitional
restriction that cannot be inherited from any of its ancestors. Thus, determining
whether a node is a key node involves testing its definitional restrictions to see if
they can be inherited.

\textsuperscript{12}This argument would also hold for all non-key nodes below the red nodes, but their number
can be combinatorially greater than the number of red nodes since they can correspond to the
intersection of any combination of red nodes. The nice thing about red nodes are that they
provide a clearly identifiable small set of nodes which limit the search.
If the node being tested is an entity, determining whether it is a key node is not as expensive as it might seem: if the node being tested is integrated into SemNet and if the event is simply inherited, the event will have been inherited from one of its ancestors. This ancestor will be relatively close to it since it will be below their common template event: all events have as ancestor their template. There is no such guarantee of proximity when searching up the hierarchy from the node being tested.

If the node being tested is an event, things are slightly more complex. Events are currently rarely built as intersections of events, so this case does not occur in the implementation. However it is conceivable for them to be built as such. As far as being a key node is concerned, the same arguments apply to restrictions on events by other events as do for entities. However for restrictions due to the subject_, object_ and action_ arcs, this is not the case, since all events can be thought of as inheriting their arcs from their template event. Thus, the red nodes of an event are constituted by the event’s template event \(^{13}\) and by red nodes determined in the same way they are for entities.

Later stages of semantic integration also need to test whether events are inherited or not. Thus an implementation may choose to boost efficiency by requiring of inheritance that it mark every event it builds by an inherited control. Similarly, the above search would be performed after semantic integration to check whether an NL utterance built an event that could have been inherited.

- Finding the implicit descendents

If there are more than one red nodes, downward integration can take place. As argued in 6.7.2.1 (p. 200), finding the black node’s implicit descendents is simply a matter of traversing the set relations downwards from from the red nodes and recording every node reached in this manner from all red nodes. These are the implicit descendents. Not all the descendents may need to be connected to the black node: if an implicit descendent has an ancestor which is also an implicit

---

\(^{13}\)Note that any better solution will improve efficiency, but not correctness; Further, most events are built as children, rather than descendents of their template event so in these cases, no additional cost is involved.
descendent, it may not need to be connected to the black node. The algorithm
aims to find the minimal set of such descendents.

The black node may either be a definitional or an observational parent to each
of the minimal set of descendents. If there is an uninterrupted path downwards
of definitional set relations (i.e. whose object is definitional with respect to its
target), from every red node to a particular implicit descendent, then the explicit
set relation built from the black node will be definitional. Otherwise it will be
observational.

In determining the minimal set of descendents it is important to ensure that if an
implicit descendent would be connected directly as a definitional child of the black
node, the path to it remains definitional even if this path passes through one of its
ancestors directly connected to the black node.

More formally, the algorithm processes a set of paths to expand $E$, a set $V$ of
currently visited nodes, and the set $I$ of detected implicit descendents of the black
node. Each path consists of a tuple $(o, s, n)$ where $o :: Node$ is the path’s origin,
$s :: \{Obs, Def\}$ is the path’s status, and $n :: Node$ is the current end of the path.$V$ associates all sets currently visited so far with the origin of the path that visited
them, and the path’s status. $V$’s elements are thus tuples $(v, l)$, where $v :: Node$
is the node and $l$ is a list of paths’ origins ($:: Node$) and statuses ($:: \{Obs, Def\}$)
expressed as tuples $(o, s)$. $I$ is a set of tuples $(n, s)$ where $n :: Node$ is an implicit
descendent, and $s :: \{Obs, Def\}$ is the status of the path to $n$.

Initially, $E$ contains a path for each of the red nodes: each path has definitional
status, their origin and current point are equal, both set to the path’s red node.
Initially, also $V = I = \emptyset$. The algorithm then iterates:

1. For each path $p = (o, s, n)$ taken out of $E$, a new set of paths $P'$ is found:
each new path $(o, s', n')$ has as current end node $n'$ one of the children of $n$;
the status $s'$ of each new path $(o, s', n')$ is the worse of the status $s$ of $p$ and
the status of the set relation connecting $n$ to $n'$.

2. Each path of $P'$ is added to $V$: any current end nodes not in $V$ are added to
it, and all nodes of \( \mathcal{V} \) are updated to include any new paths of \( \mathcal{P}' \) leading to them. If a node \( n \) of \( \mathcal{V} \) already registers a path \((o, s)\) from an origin \( o \), but \( \mathcal{P}' \) includes a path \((o, s', n)\) where \( s' \) is better than \( s \), the path status associated with \( n \) in \( \mathcal{V} \) is updated to \( s' \). (i.e. to definitional).

3. Every node of \( \mathcal{V} \) which had information added to it at this iteration, is checked to see if it has all red nodes as origins of all the paths it is associated to. If so, it is added to \( \mathcal{I} \), the set of the black node’s implicit descendents. If all the paths have definitional status, the implicit node is associated with a definitional status, otherwise with an observational status. Any update to a node \( n \) of \( \mathcal{V} \) also appearing in \( \mathcal{I} \), is accompanied by an update of \( n \)’s status in \( \mathcal{I} \).

4. Only paths of \( \mathcal{P}' \) that resulted in a change of \( \mathcal{V} \) put into \( \mathcal{E} \). \(^{14}\)

The process iterates again until \( \mathcal{E} \) is empty.

\( \mathcal{I} \) now contains the set of all the implicit descendents, each associated with the status the path from the black node to it should have. \( \mathcal{I} \) must now be reduced to the minimal set of descendents \( \mathcal{D} \). If the black node is not restricted by a non-inherited definitional event, \( \mathcal{D} \) is easily determined: Every element \( e \) of \( \mathcal{I} \) which does not have a parent \( p \) in \( \mathcal{I} \) of equal or better status is placed in \( \mathcal{D} \), with the additional constraint if \( e \)’s status is definitional, that the set relation from \( e \) to \( p \) must be definitional. Otherwise, \( \mathcal{I} \) must be preprocessed to remove all non-subsumed nodes. A highest first downwards search, minimises the number of nodes to test for subsumption: if a node is disjoint or is subsumed, so are all of its descendents.

This algorithm can be made more efficient in various ways, for instance by ensuring that when two paths with different origins reach a same node, all paths from that node are annotated by both origins. A highest first search, expanding \(^{15}\) nodes with lesser depth first, ensures that nodes are reached by all paths before being

---

\(^{14}\) This ensures that only a new path, or a change in path status from observational to definitional is searched.

\(^{15}\) I.e. finding all the node’s children and placing them into \( \mathcal{P}' \)
themselves expanded. By annotating the search paths with a list of origins and the status of each, shared paths only need to be traversed once, and the minimal set of nodes can be determined simultaneously to the search. However, even if the order of time complexity is \( O(N_0) \), where \( N_0 \) is the number of set relations connecting every node with a red node as ancestor, little can be done about the main cause of inefficiency: \( N_0 \) is immense, since all descendents of the red nodes must be investigated, as at any point any of these descendents may have an intersection which should be connected to the back node.

- **Another downwards integration scheme**

Another scheme deserves a quick mention. Semantic distance [Short et al. 94a], [Short et al. 94b], uses the number of descendents concepts have to determine associativity and similarity. Traversing the network to count the number of descendents a concept has is highly inefficient, for the same reason as downwards integration is. Since this operation is key to the operation of semantic distance, efficiency is obtained by associating each concept in the hierarchy with the number of descendents it has.

Using this number can dramatically reduce the number of descendents the downwards integration algorithm has to consider. If a concept is an implicit descendent of the black node, it is the descendent of every red node. Not all the red nodes have the same number of descendents. Consider the set \( \mathcal{F} \) of all the descendents of the red node with the fewest descendents: all the implicit descendents are within \( \mathcal{F} \). Searching upwards is often cheaper than searching downwards since nodes in SemNet tend to have more children than parents. To determine whether each of the nodes of \( \mathcal{F} \) has a red node ancestor, search upwards can be used. Only nodes that can still be below the red nodes should be investigated: the depth of a node must be greater than the smallest depth of any of the red nodes. When its depth was lesser, the search can be cut, since there is no chance that further search upwards will reach any red node. Similarly, only nodes with families subset or equal to the families of the red nodes should be searched. Finally, if the node encountered is a red node there is no need to search beyond it.
By adopting a mixed algorithm determining \( \mathcal{F} \) and \( \mathcal{I} \) the set of implicit descendants simultaneously, search can be limited to \( O(N_1) \) where \( N_1 \) is the number of different set relations traversed: the highest node first scheme is augmented by searching upwards all of the node's parents, each time a new node is encountered by the downwards search. Thus, the search downwards can maintain a list of all red nodes the path it is traversing leads up to. The hope is that this search will be cheaper than the previous, as \( N_0 \) should be greater than \( N_1 \) since nodes tend to have more children than parents in SemNet.

### 6.7.2.2 Upwards Integration

- **The orange node**

Defining the implicit ancestors is one thing (6.7.1.2 (p. 194)), finding them is quite another: testing the definition of every node of SemNet would be prohibitively expensive. Only intersections of the black nodes' ancestors can be its implicit ancestors, since all other concepts are either its descendents or are disjoint from it. Further, all the paths from the black node to *typeless* intersect at one node, be it at the very least *typeless* itself. All implicit ancestors of the black node will be descendents of this node, since this node is itself fully integrated upwards and all paths to *typeless* go through it: this is the orange node.

All the restrictions the black node does not inherit are accompanied by set relations to the relevant part of the restrictions' template. If the restriction is an *action*., the black node is connected to the *action*.'s template event through set events. If the restriction is an event, the black node is a descendendent of the *subject* (or *object*) of that event's template.\(^{16}\) This means that only descendents of the orange node may be the black node's implicit ancestors:

1. Consider first a black node with no additional restrictions of its own. Only the intersections of its ancestors can be implicit ancestors, since all other nodes must add a restriction that the black node does not have, if uniqueness

\(^{16}\)The black node cannot be an *action*., since *actions* are not in the inheritance hierarchy.
is to be preserved. Such intersections can only occur between ancestors below the orange node. Indeed, this could only be violated in two cases:

(a) if there was an intersection of a node above the orange node and a node below it, but this would mean that the orange node is not the node through which all paths lead to typeless;

(b) if two ancestors of the orange node had an intersection which was not further restricted by another event\textsuperscript{17} and yet was not above the orange node. This is not possible as the orange node is perfectly integrated.

Since neither case can occur, only the intersections of the black node’s ancestors can be implicit ancestors, if the black node has no additional restrictions.

2. Now consider the black node’s restrictions. Type-checking will have added set relations from the black node to the relevant part of their templates. The \texttt{subject\_}, \texttt{object\_} or \texttt{action\_of} of any instance of a restriction must occur below the template’s \texttt{subject\_}, \texttt{object\_} or \texttt{action\_of} respectively. Because all paths to typeless must pass through it, the orange node is above (or equal to) this \texttt{subject\_}, \texttt{object\_} or \texttt{action\_of} node. The only additional nodes to those considered by the previous case, are those that have a restriction the black node has, but its ancestors do not. Additional nodes to the set of ancestors not due to restrictions on the black node itself, must have some additional restriction because of uniqueness. There are two types of such restrictions that maintain compatibility with the black node:

(a) Those the black node is restricted by. Since these restrictions must occur below their templates, the implicit ancestors must occur below the templates’ \texttt{subject\_s}, \texttt{object\_s} or \texttt{action\_ofs}. Thus they must occur below the orange node.

\textsuperscript{17}Why not further restricted? Since restrictions on the black node are not considered here, the only restrictions still compatible with the black node are those on the black node’s ancestors. But these ancestors are fully integrated, so would be connected to this intersection. This would mean there would a path that did not pass through the orange node to typeless, violating its definition.
(b) Restrictions that are parents or ancestors of those the black node is restricted by: since actions are not in the inheritance hierarchy, this only applies to restricting events. An example would be “All car traders are dishonest” and “John sells Range Rovers”.\(^\text{18}\) Since these restrictions are ancestors of those of the black node, they can apply to any of the black node’s ancestors, even those above the orange node, and yet remain compatible with the black node.

So in all but case 2.b, the orange node provides a good restriction on upwards integration. In particular, if the black node does not have any definitional restrictions of its own, only the search space below the orange node need be investigated.

The orange node thus constitutes a good articulation of the problem:

1. If the black node has no additional restrictions of its own, only intersections of ancestors of the black node below the orange node which are not further restricted by additional restrictions can be the black node’s implicit ancestors.

2. If the black node has additional restrictions of its own, only definitional descendants\(^\text{19}\) of the ancestors of the black node below the orange node may be implicit ancestors of the black node. Among these, only those which are restricted by the restrictions that the black node has can be its implicit ancestors.

3. Children of the orange node’s ancestors can only be compatible with the black node if they are defined only by ancestors of the black node’s own restrictions. If the black node has no such additional restrictions, then no children of the orange node’s ancestors can be compatible with the black node.

As such, the orange node proves useful in the following arguments. The first two points will be considered first.

\(^{18}\) Trading is an ancestor of buying and selling, car traders are people who trade cars a Range Rover is a car, so John is a car trader, and thus dishonest.

\(^{19}\) 6.7.1.2 (p. 194) argued for the need for only definitional descendents.
• The Brown nodes

Colour all explicit ancestors of the black node which are also (explicit) descendents of the orange node brown.

Brown nodes may be either definitional or observational. The need for definitional brown nodes is obvious. However, the children of brown nodes which are observational with respect to the black node may also be its implicit ancestors, since they may be less restricted than the black node. This depends only on their definitional restrictions.

When finding a brown node, the search should also record whether it is an observational or a definitional ancestor of the black node. Indeed, only intersections of brown nodes which are definitional with respect to the black node can be definitional ancestors of it. Indeed they have do not have definitional restrictions that the black node does not also have. An intersection having a brown node which is observational with respect to the black node has definitional restrictions from the brown node which are observational for the black node. Thus intersections with brown nodes which are observational with respect to the black node can only be implicit observational ancestors of the black node.

• Detecting the brown and orange node: an algorithm

The orange node is the one through which all paths to typeless pass. Thus for a naive implementation, detecting it is simply a matter of searching up all the paths to typeless. Unfortunately, this leads to a combinatorial time complexity in graphs with a lot of sharing. This section presents an algorithm achieving near-linear time complexity.

○ Brown and orange node detection: The ideas

Since the orange node is the only one through which all paths from the black node to typeless pass, when searching for the orange node, one wants to be sure that all paths to it have been traversed. But, one does not want to traverse each independently, since that causes high time complexity. Thus one wants to traverse each arc of any path only once, while somehow maintaining tally of which paths
that arc corresponded to. To do this, one needs an identifier for each path.

\( p :: PathId \) is a unique identifier for each path. It is represented as a list of
traversed nodes \((PathId = [Node])\). The full list of traversed nodes is unnecessary, since retaining only the node at which a path split, and the direction the path took is sufficient to uniquely identify any path. Thus, every time a node \( n \) is traversed which has more than one parent, the node \( n \) is appended to the path \( p \), and then each of the parents is appended to a copy of \( p \). Each copy of \( p \) uniquely identifies a path through one of the parents. The path is also compressed so that if a node \( n \) is to be appended to a list, the last element of which is also \( n \), \( n \) is not duplicated.

The idea is thus to maintain a list of the paths going through each arc that is traversed as the search proceeds up the hierarchy. Since the search is a gradual process, the path list only contains the paths from the black node to the just expanded nodes. Different paths may share the same arcs, since they may diverge, only to converge later. To avoid maintaining many paths when each arc is traversed, an arc’s different paths can be packaged together. If a set of paths converge on \( n \), all paths above \( n \) will have as first element a packet containing all paths that reached \( n \), followed by the usual list of nodes expressing the path from \( n \). In effect the packet allows paths sharing the node \( n \) not to be multiplied out.

To know whether a node is orange, one needs to know that all paths from the black node converge on it. Clearly the simplest case is if all the paths from the black node converge on it simultaneously during the search. This will not happen in a breadth first search, since some paths will be longer than others. However, if the depth of each node is used, a lowest first search will provide the desired effect. The search upwards expands nodes by taking a node from the search space to investigate \( S \), finding its parents and adding them to the search-space \( S \). If, at each step, the node it takes is the lowest of those in \( S \), it is a lowest first search.

A lowest first search ensures that all the children of any given node will be reached before the node itself is. As a result, all paths converge simultaneously on all nodes. This satisfies the requirement for a simple way of detecting the orange node, but also means that each set relation is guaranteed only to be traversed once.
The resulting algorithm therefore performs a lowest node first search upwards from the black node. As it searches upwards it maintains a set of paths, each associated with a status stating whether every set relation of the path was definitional, or whether one or more of them were observational. It colours every uncoloured node that it meets brown, unless all paths converge at one node, in which case it colours it orange. It also associates with every node that it meets the best status of set of paths reaching that node.

- **The algorithm**

The algorithm uses 2 sets:

- $\mathcal{N}$ is the set of visited nodes. Its elements are nodes associated with the best status of the paths that reached it, and with the colour associated to it:
  \[(n :: \text{Node}, s :: \{\text{Def}, \text{Obs}\}, c :: \text{Colour})\]

- $\mathcal{S}$ is the search space to expand. Its elements are paths augmented to include packeting, associated with the best status of all alternatives of the packet and with the node to expand next: \[(p :: \text{Path}, s :: \{\text{Def}, \text{Obs}\}, n :: \text{Node})\]

Initially, $\mathcal{N} = \{(bn, \text{Def}, \text{Black})\}$ and $\mathcal{S} = \{([], \text{Def}, bn)\}$, where $bn$ is the black node. The algorithm then iterates:

1. The set of paths $\mathcal{P}$ with the lowest node to expand are taken out of $\mathcal{S}$.

2. For every node $n$ to expand of $\mathcal{P}$, the set of paths in $\mathcal{P}$ associated with $n$ is found: $\mathcal{P}_n$. For every node $n$:

   a) If $n$ is in $\mathcal{N}$, it is already coloured with the colour $c$. Otherwise if $\mathcal{S} = \emptyset$, $c$ is set to orange. Otherwise $c$ is set to brown.

   b) $n$ is added to $\mathcal{N}$: it is annotated with the best status $s$ of the path set $\mathcal{P}_n$ and with the colour $c$.

   c) If $c = \text{Orange}$ the algorithm terminates.

   d) $\mathcal{Q}$ is the set of the first elements of the tuples of $\mathcal{P}_n$. $q$ is the path-packet that combines all $\mathcal{Q}$'s elements. The set of parents $\mathcal{A}(n)$ of $n$ are found.
Each parent \( p \) of \( \mathcal{A}(n) \) is associated with a status which is the worst of \( s \) and the status of the set relation connecting \( n \) to \( p \). Each parent \( p \) is also associated with a path which is \( q \) if \( \mathcal{A}(n) \) contains one element, or is \( q \) appended by \([n, p]\) (putting aside the issue of compression).

(e) The algorithm iterates.

3. The algorithm iterates.

In step 2, \( n \) is “a lowest node” since there may be more than one node at any particular depth. Because at each step, only the lowest nodes are added to \( \mathcal{N} \), nodes are only added to \( \mathcal{N} \) when all paths leading up to them have been investigated. This means that the status a node is annotated with is the result of all the paths leading to it from below. The test \( \mathcal{S} = \emptyset \) detects the orange node, since the orange node is the one through which all paths flow. In a lowest first search, all the paths leading upwards to a node will be traversed before the node itself is traversed. This is also true for the orange node, so if the orange node is the one through which all paths pass, removing the paths leading to the orange node from \( \mathcal{S} \) empties \( \mathcal{S} \).

- Implementation issues, and order of time complexity

\( \mathcal{N} \) can be implemented either by adding information directly to the nodes giving \( O(1) \) access performance, or as a BIB tree, giving near \( O(\log(N)) \) access performance.

\( \mathcal{S} \) can be implemented using a skip list: this sorts the list, costing an average \( \log(N) \) access used for insertion. Elements are always taken from the front of the list. Since the first element includes more than one reference to next elements, this reference must be passed along to the next front element. Always deleting elements from the front of the list would be a little detrimental to the \( \log(N) \) behaviour, since it corresponds to unbalancing the virtual binary tree the skip list represents. However, this only affects the first step made through the virtual tree: all further jumps access complete trees, resulting \( O(\log(N) + 1) \) type behaviour (\( \approx O(\log(N)) \)).

Since elements are sorted by the depth of the node to expand, and two or more
such nodes may share the same depth, each element of $S$’s skip list may refer to more than one set of paths $P_n$.

Rather than calculate the real worst case time complexity, an upper limit will be established. The number of steps taken through SemNet is $N$, where $N$ is the number of set relations between the orange node and the black node. The worst case is for $N$ and $S$ to contain all these $N$ elements, at each access. An access to $S$ is made each time a set relation is traversed, and at most $N$ accesses are made to $N$. Thus at worst $O(2N\log(N)) \approx O(N\log(N))$ access would be made. Since many overestimations have been made, the worst case is better or equal to $O(N\log(N))$.

- Beige and Khaki nodes

Since testing whether a node is compatible with another, and whether it is less or more restricted is expensive, reducing the search space is advantageous. The set of brown nodes can be divided into two sets: beige and khaki nodes. Downwards integration introduced red nodes. The idea underlying red nodes also proves useful for upwards integration. Let all the first key nodes encountered when traversing the set relations (both definitional and observational) up from the black node to typeless be coloured grey\(^{20}\). All beige nodes are ancestors of the grey nodes, and all khaki nodes are some of their descendents.

Implicit ancestors of the black node will either be formed by intersections of khaki or beige nodes. Because beige and khaki nodes are explicit ancestors, the implicit ancestors are not coloured. Since all nodes other than the black node are integrated, only the relations to the black node are unknown. It may thus seem that they may only be implicit ancestors to the black node, since upwards integration is being performed.

However, upwards integration may identify nodes as potential implicit ancestors which will turn out to be equivalent or even descendents of the black node. For instance the black node may be the child of three red nodes $A$, $B$ and $C$. The network may contain the nodes $D = A \cap C$ and $E = B \cap D$. $E$ is equivalent to the

\(^{20}\)Grey nodes thus include the red nodes: do not take the colouring too literally!
black node, but would not be detected when the black node is built because of the intermediate \( D \) node: \( D \) and \( E \) are the intersections of red nodes. Similarly, what appears to be an implicit ancestor may be the black node's child. For instance, \( F = A \cap B \cap C \), if \( F \) is further restricted by a non-inherited definitional event.

Although grey nodes are key-nodes, another node may have some of a grey node's definitional restrictions, and yet not be the grey node's child or ancestor. Because of uniqueness, this only occurs if such nodes are restricted by another definitional event. This event could be one of another grey node's restrictions, thus producing a node which is neither a subset nor a superset of any grey node, but is an ancestor of the black node. In general, such semi-grey nodes can be obtained as combinations of subsets of grey nodes' restrictions. They cannot have all of a grey node's restrictions, as that would make them (integrated) grey nodes' descendents. As a result, only intersections of khaki and/or grey nodes may be equivalent to or an implicit child of the black node: the black node has all the restrictions of the grey nodes. This means intersections of beige nodes need only be tested to see if their restrictions are less strong than those of the black node. Intersections of khaki and/or grey nodes must be tested to see if they are equivalent or more restricted but compatible with the black node.

If an intersection of khaki and/or grey nodes proves equivalent or more restricted, but compatible with the black node, it provides all the black nodes' implicit ancestors. Indeed, the intersection is perfectly integrated, so it is explicitly connected to all of its ancestors. Since it has all the restrictions of the black node, it has all its ancestors too. However not all its ancestors need be the black nodes', since it is more restricted than the black node. Determining which ancestors are also the black node's can be achieved by searching upwards until an orange, grey, beige, or khaki node is reached. Upwards search is cheaper than downwards search if it can be limited so as not to continue past a certain point up to typeless. If the intersection is equivalent to the black node, it provides all the latter's implicit descendents too, Otherwise, it provides a subset of them.

The difference in testing of khaki and beige nodes warrants that they be processed
differently. Note that a brown node cannot simultaneously be assigned the colours beige and khaki since all paths to typeless pass through a grey node, by definition. Thus, it is easy to change the brown and orange node identification algorithm to mark nodes beige or khaki instead of brown: it colours every node it traverses upwards khaki, until it reaches a key node, marks it grey, and then continues marking the following nodes beige. The lowest first search means that the colour of all a node’s children is known before a colour is assigned to it. Since the colour of a node depends on that of its children, no additional cost is incurred to colouring them all brown.

Khaki nodes are processed first because they may unify the black node with an existing node, thereby eliminating the need for any further integration. However the processing beige node is simpler to explain, so will be presented first.

- Processing Beige Nodes
  - Descendents of beige nodes may only be implicit definitional ancestors

Implicit ancestors of the black node are formed by intersecting beige nodes. They cannot be beige, since all beige nodes are explicit ancestors. Thus the search proceeds downwards from each beige node, considering only non-beige and non-grey nodes. The implicit ancestors of the black node, formed by intersecting beige nodes, only lack direct connections to the black node, since it is the only new node. Thus any node which is more restricted than the black node is also rejected.

Each beige node is associated with its status: observational or definitional with respect to the black node. Even if full, the intersection of beige nodes can only be an observational implicit ancestor of the black node if any of the beige parents are associated with an observational status: otherwise the definitional restrictions of the observational beige node will be inherited as definitional by the black node, effectively changing its meaning. Implicit ancestors that are observational with respect to the black node must be connected to it only by observational paths of set relations. Implicit ancestors that are definitional should be connected by at least one definitional path to the back node.

One question remains: are all descendents of beige nodes possible implicit ancestors,
or are some ineligible a-priori? The search space can be reduced if some are. For a node to be an ancestor of the black node it must be less restricted than and not disjoint from the black node. If it is to satisfy the latter condition, it must not be subject to restrictions the black node does not have. If it could be a descendent of the black node, it could have restrictions the black node does not have, but it cannot be a child of the black node since it is not a descendent of a red node. To have at least some of the restrictions of the black node, an implicit ancestor must be a definitional subset of at least one beige node.

○ Beige node algorithm

The algorithm manipulates the set of active paths \( \mathcal{A} \), and the set of discovered implicit ancestors \( \mathcal{I} \). Each active path \( a \) is a tuple \( (p :: \text{Node}, c :: \text{Node}, so) \) where \( so = (s :: \{\text{Def}, \text{Obs}\}, o :: \text{Node}) \). \( p \) is the path’s previous node, \( c \) is the current node the path is investigating, and \( s \) is the status of the path’s origin \( o \) with respect to the black node. Each implicit ancestor \( i \) is a tuple \( (n :: \text{Node}, s :: \{\text{Def}, \text{Obs}\}) \) of the ancestor \( n \) and its status with respect to the black node. Every time a newly discovered implicit ancestor \( i = (n, s) \) is added to \( \mathcal{I} \), all \( n \)’s parents in \( \mathcal{I} \) which have as status \( s \) are removed from \( \mathcal{I} \).

Initially \( \mathcal{I} = \emptyset \), \( \mathcal{A} = \emptyset \), and \( \mathcal{N} \) is the set of nodes coloured by the orange node searching algorithm. Each element of \( \mathcal{N} \) is a tuple \( (n :: \text{Node}, s :: \{\text{Def}, \text{Obs}\}, c :: \text{Colour}) \).

1. Filter out all elements of \( \mathcal{N} \) that have not beige as colour.

   (a) Take an element \((n, s, c)\) out of \( \mathcal{N} \).

   (b) Find the set \( \mathcal{D} \) of \( n \)’s definitional children not in \( \mathcal{N} \).

   (c) For each \( d \) of \( \mathcal{D} \), add \((n, d, (s, n))\) into \( \mathcal{A} \).

   (d) Go to (a) unless \( \mathcal{N} = \emptyset \).

\[21\text{If it were, the inheritance graph would be cyclic, which is illegal; or it would be coloured, and this algorithm deals with the uncoloured descendents of beige nodes.}\]

\[22\text{6.7.1.2 (p. 194) argued for the need for only definitional descendents.}\]
2. The path(s) $P$ of the set $A$, which have a current node of the least depth, are taken out of the set $A$.

3. $P$ is divided into a set of paths $P_c$, where each set $P_c$ contains all paths of $P$ with current node $c$.

4. For each set of paths $P_c$,
   
   (a) If not all of $c$'s definitional parents appear in the previous nodes $p$ of the paths $P_c$, or if $c$ is restricted by a definitional event, it must be tested for subsumption of the black node. If $c$ does not subsume the black node, the algorithm skips to processing the next $P_c$. Otherwise it continues:

   (b) If all the statuses of the paths of $P_c$ are observational, go to (e).

   (c) The worst status $x$ of all paths in $P_c$ is found.

   (d) If $c$ is not coloured, it is added to $I$ associated with $x$.

   (e) The set $D$ of $c$’s uncoloured definitional children is found. For each node $d$ of $D$, a set of paths $N(d)$ is added to $A$, where

   $\forall(p_1, p_2, p_3) \in P_c \exists!(n_1, n_2, n_3) \in N(d). (n_1 = c \land n_2 = d \land n_3 = p_3)$

5. If $A \neq \emptyset$, go to 2. Otherwise terminate.

The search is highest node first since nodes of the least depth are examined first. This means that the parents of any encountered node will have already been traversed, if they can be reached by downwards traversal from any beige node. A node’s parent that was not traversed will thus not appear in any path of $P$ leading to the node. Such nodes are tested in step 4.a: only those that subsume the black node are retained as implicit ancestors.

The paths need only include the origin’s status, and not its node since if two paths come from the same beige node they cannot have different statuses: status is not modulated as only definitional set relations are traversed downwards.

- Optimisations to the beige node algorithm

The main cost in the beige node integration algorithm comes from the subsumption tests. The main concern is therefore to reduce their cost.
One way of doing this is if all the parents of a candidate implicit ancestor are
descendants of beige nodes, is to check whether the event restricting it is inherited.
If it is, there is no need to check it for subsumption: either it was inherited from an
ancestor above the orange node – so the black node has it too –; or it was inherited
from one of the node’s ancestors which is a descendent of a beige node – so it has
already been investigated.

Another way is to create a set of the definitional events restricting the black node
or its ancestors below the orange node. This set can be built by the brown and
orange node detecting algorithm, without incurring any additional cost. Events
restricting candidate implicit ancestors can then be tested with respect to this set
to determine whether they are less restrictive than some event of the set. If they
are, the subsumption test with respect to that restriction succeeds. This improves
on normal subsumption testing, since it removes the need to gather the black node’s
ancestors at each test of subsumption. The set can further be extended if the search
has to proceed beyond the orange node to the black node’s more distant ancestors.

The complexity of beige node integration should not exceed $O(N \log(N))$ excluding
the cost of subsumption, where $N$ is the number of beige nodes, since $P_c$ can be
implemented as a BIB tree and there are $N$ nodes to process – hence $N$ iterations.

- **Processing Khaki Nodes**

Although each khaki node is perfectly integrated, intersections of khaki nodes that
subsume or are subsumed by the black node may not be connected to it. It is these
nodes that khaki node processing must detect.

The problem is not much different to the integration of beige nodes, except that
intersections of khaki or grey nodes that are further restricted by a definitional
event or concept intersection may be subsumed by the black node. The algorithm
is therefore much the same:

The algorithm processes a set of active paths $A$, a set of implicit ancestors $I$, an
implicit descendent $d$. Each active path $a$ is a tuple $(p :: Node, c :: Node, so)$ where
$so = (s :: \{Def, Obs\}, o :: Node)$. $p$ is the path’s previous node, $c$ is the current node
the path is investigating, and \( s \) is the status of the path’s origin \( o \) with respect to the black node. Each implicit ancestor \( i \) is a tuple \((n :: Node, s :: \{Def, Obs\})\) of the ancestor \( n \) and its status with respect to the black node. Each implicit ancestor \( i \) is a tuple \((n :: Node, s :: \{Def, Obs\})\) of the ancestor \( n \) and its status with respect to the black node. Every time a newly discovered implicit ancestor \( i = (n, s) \) is added to \( I \), all \( n \)'s parents in \( I \) which have as status \( s \) are removed from \( I \).

Initially \( d = (\text{typeless}, Def), I = \emptyset \) and \( A = \emptyset \), and \( N \) is the set of nodes coloured by the orange node searching algorithm. Each element of \( N \) is a tuple \((n :: Node, s :: \{Def, Obs\}, c :: Colour)\).

1. Filter out all elements of \( N \) that have not grey or beige as colour.
   
   (a) Take an element \((n, s, c)\) out of \( N \).
   
   (b) Find the set \( D \) of \( n \)'s definitional children \(^{24}\) not in \( N \).
   
   (c) For each \( d \) of \( D \), add \((n, d, (s, n))\) into \( A \).
   
   (d) Go to (a) unless \( N = \emptyset \).

2. The path(s) \( P \) of the set \( A \), which have a current node of the lowest depth, are taken out of the set \( A \).

3. \( P \) is divided into a set of paths \( P_c \), where each set \( P_c \) contains all paths of \( P \) with current node \( c \).

4. For each set of paths \( P_c \),
   
   (a) If not all of \( c \)'s definitional parents appear in the previous nodes \( p \) of the paths \( P_c \), or if \( c \) is restricted by a definitional event, it must be tested for subsumption of the black node. If \( c \) neither subsumes nor is subsumed by the black node, the next \( P_c \) is processed. Otherwise the algorithm continues:
   
   (b) If all the statuses of the paths of \( P_c \) are observational, go to (f).
   
   (c) The worst status \( x \) of all paths in \( P_c \) is found.

\(^{24}\)6.7.1.2 (p. 194) argued for the need for only definitional descendents.
(d) If $c$ was subsumed by the black node, $d$ is set to $(c, x)$, and the algorithm terminates.

(e) If $c$ is not coloured, it is added to $\mathcal{I}$ associated with $x$.

(f) The set $\mathcal{D}$ of $c$'s uncoloured definitional children is found. For each node $d$ of $\mathcal{D}$, a set of paths $\mathcal{N}(d)$ is added to $\mathcal{A}$, where

$$\forall(p_1, p_2, p_3) \in \mathcal{P}_c \exists!(n_1, n_2, n_3) \in \mathcal{N}(d). (n_1 = c \land n_2 = d \land n_3 = p_3).$$

5. If $\mathcal{A} \neq \emptyset$, go to 2. Otherwise terminate.

The main difference stems from step 4.d: if a node $d$ is detected which unifies with the black node, or is subsumed by it, $d$ is placed in $\mathcal{D}$, and the algorithm terminates: as discussed in 6.7.2.2 (p. 214), all the black node's implicit ancestors will be included in $d$'s ancestors.

If $d = (\text{typeless, Def})$, $\mathcal{I}$ is the set of all the black node's implicit ancestors that are subsets of khaki or grey nodes, and khaki node integration terminates after their connection to the black node$^{25}$.

If $d \neq (\text{typeless, Def})$, $\mathcal{I}$ is a subset of the black node's implicit ancestors. The black node is connected to them$^{25}$, and processing continues: all the black node's implicit ancestors included in $d$'s ancestors must be found:

This is simply achieved by a breadth first search upwards$^{26}$, expanding nodes that are subsumed by the black node. Nodes that subsume the black node are collected into a new set $\mathcal{T'}$. The algorithm terminates when no nodes are available for expansion.

This search is more cost-effective than the first: as soon as the set of implicit parents $\mathcal{I}$ is found, there is no need for any other form of upwards integration, since every node of $\mathcal{I}$ is perfectly integrated: no beige node or above-orange node integration is required. It should also be more efficient than the first as nodes tend to have more children than parents, so fewer nodes should need be traversed. This is critical, since at every step it tests subsumption, an expensive operation.

---

$^{25}$The status associated with each node $i$ of $\mathcal{I}$ determines whether $i$ will be connected to the black node by an observational or definitional set relation

$^{26}$Traversing both definitional and observational set relations
Before each node $i$ of $T'$ can be connected to the black node, its relation to the black node must be determined: observational or definitional. Indeed, subsumption does not state how a node's ancestors should be inherited. The essence of the problem is thus to discover whether $i$ has any definitional events (inherited or not) which are observational for the black node. This is easy to detect if all $i$'s ancestors are coloured orange, beige, grey or khaki, since nodes of these colours are associated with their status with respect to the black node. In this case, the status determining algorithm is a breadth first search upwards, expanding each node to its definitional uncoloured parents. All encountered coloured nodes are collected into a set $C$. The algorithm aborts if any node available for expansion is above the orange node – determined using node depth. Otherwise it terminates when no nodes are available for expansion. If the algorithm did not abort, $i$ is connected to the black node by a set relation which has as status, the worst status of the nodes of $C$. Only definitional parents need be considered, since only from them can definitional events be inherited.

If any of these paths upwards does not lead to a coloured node, and goes higher than the orange node – detected using node depth –, then the ancestor discovered is an above-orange node, which is discussed in the next section. This case only occurs if the black node has additional definitional events, as argued in 2b (p. 208).

The same optimizations as discussed for beige node integration can be applied here too. The complexity of khaki node integration should not exceed $O(N \cdot \log(N))$ excluding the cost of subsumption, where $N$ is the number of khaki and grey nodes, since $\mathcal{P}_c$ can be implemented as a BIB tree and there are $N$ nodes to process – hence $N$ iterations.

- **Processing the Above-Orange Nodes**
  - **The idea**

If the black node is further restricted by definitional events, their ancestors may define concepts which subsume the black node. These concepts too, should be connected to the black node. The black node's observational events may also invoke such concepts: their ancestors may define concepts which subsume the black node.
The first step is to determine the black node’s events’ templates. All the ancestors of these templates may introduce concepts which subsume the black node. Take for instance, “All car traders are dishonest” and “John sells Range Rovers”. If the black node is John, then not only must John be correctly integrated under the subject_ of the selling template event, but also must be identified as a car trader. Thus the search must not only proceed upwards to the black node’s events’ templates, and beyond to the ancestors of these templates, but also downwards from these ancestors’ subject_ or object_, to all their specialisations which subsume the black node – like people who trade cars. The number of template ancestors of any event is of $O(\log(N))$ where $N$ is the number of different template events in SemNet: event intersection is rare, so template events tend to be defined by their action_ only. Thus they constitute a tree.

For each event $e_i$ restricting the black node, let $a_i :: \{ \text{subject}_-, \text{object}_- \}$ be the type of the arc connecting $e_i$ to the black node. The set $T(e_i)$ of template event ancestors of each $e_i$ is found. For each $t_{i,j} \in T(e_i)$, let $n_{i,j}$ be the node connected to it by an arc of type $a_i$. For each $n_{i,j}$, the algorithm proceeds downwards testing each definitional descendent to see if it subsumes the black node. The set of such nodes $\mathcal{N}_{i,j}$ is then reduced to the minimal set of implicit parents, by removing every node whose child is also in $\mathcal{N}_{i,j}$. The remaining nodes are connected to the black node by set relations with as status the worst of $e_i$’s status with respect to the black node, and $t_{i,j}$’s status with respect to $e_i$.

- An improved algorithm

Clearly the previous algorithm involves many subsumption tests, rendering it very costly if subsumption turns out to be expensive. Many of these tests can be avoided by exploiting the fact that beige and khaki node integration have already been performed. Recall that for every event $e_i$ to which the black node is connected by an arc of type $a_i$, type-checking made the black node into a subset of the node connected by $a_i$ to $e_i$’s template event $t(e_i)$. Call this node $a_i(e_i)$, and the nodes below it in the hierarchy $a_i(e_i)$’s clan: when John is the black node in “John sells Range-Rovers”, the set of all subject_ of selling events is the relevant
clan. Further call an ancestor clan of \( a_i(e_i) \), each hierarchy of nodes appearing below the node connected to the template of one of \( e_i \)'s ancestors by \( a_i \): the set of subjects involved in trading events constitutes just such an ancestor clan. Khaki and beige node integration will have connected the black node to the minimal set of implicit parents of \( a_i(e_i) \)'s clan. These parents will be perfectly upwards integrated, so they will be connected\(^{27}\) to any nodes of \( a_i(e_i) \)'s ancestor clans that subsume them. Call these nodes of \( a_i(e_i) \)'s clan that are connected to \( a_i(e_i) \)'s ancestor clans, clan connection points. These clan connection points (C.C.P.) can be exploited to drastically reduce the space to be tested by subsumption: the black node is already known to be subsumed by the C.C.P.s, so there is no need to test nodes above them for subsumption. Only the nodes below need be considered.

Nodes of \( a_i(e_i) \)'s clan may be connected to more than one ancestor clan. This raises the question of which clan should be processed first. Choosing to process the lowest ancestor clan first ensures that additional information gathered during its processing, can be used in the processing of the next clans. Indeed, every ancestor clan may be itself connected to clans that are its ancestors. Through these connections, these latter ancestors also become the black node's, and so must be searched. The processing of the lowest ancestor clan may reveal new C.C.P.s to higher clans above the minimal set of implicit parents. These clan connection points will be connected to lower nodes of the ancestor clans, providing a reduction of search space.

The problem of above orange node integration can thus be divided into two steps. First a skeleton of the search space is extracted, expressing the topology of the ancestor clans. This skeleton then guides the search for clan connection points. The skeleton is derived from the hierarchy of the event connected to the black node, whereas the C.C.P.s are derived from the hierarchy above the black node itself.

\( \diamond \) The skeleton algorithm

The skeleton of the search space has as task to reduce the amount of information

\(^{27}\) Possibly indirectly
that must be considered by the search for C.C.P.s. In particular its result states which events signal C.C.P.s, and under what conditions the C.C.P. search upwards has detected all explicit ancestors of the lowest clan. The result is therefore equivalent to a list of commands stating what the C.C.P. search algorithm should search for next. The events that signal C.C.P.s are the ancestors of the event connected to the black node. Because event intersection, although rare, is legal, the search for such events must traverse the whole event hierarchy, rather than hopping upwards from one template event to the next.

Each template event $e$ thus found is associated with a $a_i(e)$ which has a depth $\text{depth}(a_i(e))$. This depth, the "termination-depth", can be used to detect when all explicit ancestors of the current lowest clan have been detected. To simplify the explanation, assume for the moment that no event intersections occur. Consider a node $e_2$, the ancestor of the event $e_1$ connected to the black node. If $e_2$ and $a_i(e_2)$ are perfectly integrated, then any of their ancestors $e_3$ and $a_i(e_3)$ must be above them in the hierarchy. This means that if a node $x$ is a child of $a_i(e_2)$, then any path from $x$ to $a_i(e_3)$ will pass through $a_i(e_2)$. I.e. if $x$ is a subset of $a_i(e_2)$ and if $x$ is also a subset of $a_i(e_3)$, then any subset of $a_i(e_3)$ which includes $x$ is either an ancestor or a descendent of $x$. Thus, limiting the search to nodes below the depth of $a_i(e_2)$ will ensure that an upwards searching process will reach all and only nodes connected to events of $e_2$'s type, and not events of $e_2$'s ancestors' types. This means that the upward search of the C.C.P. algorithm need only test for one event type at a time to detect C.C.P.s. The command list returned by the skeleton algorithm thus only needs to be event types associated with their termination-depths.

What then of the event intersections? Event intersections change the event hierarchy from a tree into a graph. This means that any given depth there may be more than one ancestor of the event connected to the black node. This is reflected in the black node's explicit hierarchy: at a given depth there may be two of the black node's ancestors, each connected to a different ancestor of the event connected to the black node. Thus the depth scheme above does not work unmodified in this case. However, what is true in the case of no event intersections is also true for each top-down path through the black node's ancestor hierarchy. This means that
each new event intersection encountered by the skeleton algorithm on its search upwards introduces only \( n - 1 \) new event types that the C.C.P. search algorithm must consider, where \( n \) is the number of sets involved in the intersection. It also means that the termination-depth associated with each event type remains unchanged. Thus the commands of the command list returned by the skeleton algorithm must only be augmented by a set of event types to record.

To ensure that the C.C.P. algorithm processes the lowest clans first, as argued in 6.7.2.2 (p. 223), the command list returned by the skeleton algorithm must be sorted by depth: greatest depth first. Each command contains three pieces of information: the type of event of the clan to find, the termination depth of that search, and the event types that must be detected and recorded for the processing of higher clans. The type of event of the clan to find is its action., and ensures that the downwards C.C.P. search stays within the clan.

The only other new problem is to ensure that each command is associated with the correct event types that the C.C.P. algorithm must detect and record. This is achieved by recording a range for each new event type encountered during the search up the hierarchy of the event connected to the black node. For a given event \( e \), the range is \( \text{range}(e) = [\text{depth}(a_i(e)), \text{depth}(a_i(f))] \), where \( f \) is \( e \)'s descendent on the path from \( e \) to the event connected to the black node. The C.C.P. algorithm must detect events of \( e \)'s type when searching nodes which have a depth within this range: \( e \) only applies to nodes below \( \text{depth}(a_i(e)) \) and nodes of \( e \)'s type above \( \text{depth}(a_i(f)) \) are not ancestors of the black node for the reasons explained in 6.7.2.2 (p. 224). Since each end of the range is a depth associated with a template event ancestor of the event connected to the black node, the range will not require any command of the command list to be split into two, but will correspond to an integer number of commands of the command list.

The next step is to map the set of ranges thus determined onto the command list structure. This is easily achieved using a key-sorted list \( \mathcal{L} \) of tuples associating a key (depth) with an event type. The list is sorted according to decreasing depth. Each event thus has two entries in the list corresponding to the start and end of its
depth((n, d) :: Path) = d :: Depth
node((n, s) :: Path) = n :: Node
depth(n :: Node) = d :: Depth returns n’s depth.
act(n :: Node) = a :: Node returns the action a of n.

depth range. Once the skeleton algorithm has completed its search, the command list is built by traversing L and maintaining a set S of currently active events. At each step through L the event of the current element of L is added (if not in S) to, or deleted (if in S) from S. A new command is added to the command list each time an event is deleted from S: each command will complete the upwards C.C.P. search for that particular type of event, and trigger the search within its particular clan downwards for implicit ancestors of the black node. It is therefore important to ensure that additions to S appear after deletions: since additions correspond to the beginning of a range, they should appear on the next command, since the command is an expression of the end of a range. This is easily achieved by recording the lesser depth d₁ as 2d₁ + 1 in L, and the greater depth d₂ as 2d₂ in L.

First the hierarchy above the event connected to the black node is searched: L is the key-sorted list. insert(e :: Node, d₁ :: Depth, d₂ :: Depth) inserts the event type e into the list between depths d₁ and d₂. For reference:

```plaintext
> insert (e,d1,d2) list
> = gmerge (\(\_,a\) \(\_,b\) -> a >= b) [(e,2*d2),(e,2*d1+1)] list
```

E is an ordered set of paths to investigate above the event e connected to the black node. Each path is a tuple (n :: Node, d :: Depth), and the ordering depends on the depth. Initially L = [] (is empty), E = {(e, depth(a₁(e)))}.

1. A path x is taken out of E, where ∀y ∈ E,(depth(node(x)) ≥ depth(node(y)))

2. If node(x) is a template event, insert( act(node(x)) , depth(aᵣ(node(x))) , depth(x) ) L
   is executed. Then, depth(x) is set to depth(aᵣ(node(x))).

3. The set P of parents of node(x) is determined. Each node of P is associated with depth(x) and added to E. If E contained a path p with node node(x), the new path is (x, max(depth(p), depth(x)))
4. The algorithm iterates to 1 until $\mathcal{E} = \emptyset$.

The next stage builds the command list. This is achieved by maintaining a state, $\mathcal{S}$, expressing the currently active event types. The command list $\mathcal{C}$ is a queue to allow quick tail insertion. Initially $\mathcal{S} = \emptyset$, and $\mathcal{C} = []$.

1. The head $x$ of $\mathcal{L}$ is removed from $\mathcal{L}$.

2. If $\text{node}(x) \notin \mathcal{S}$ then $\text{node}(x)$ is added to $\mathcal{S}$. Otherwise, $(\text{node}(x), \text{depth}(x)/2, \mathcal{S})$ is appended to $\mathcal{C}$, and $\text{node}(x)$ is removed from $\mathcal{S}$.

3. The algorithm iterates to 1 until $\mathcal{L} = []$.

\textbf{The clan connection point algorithm}

The C.C.P. performs a lowest first upwards search from the black node. Trivially, this will pass through all the black node’s explicit ancestors. The command list extends this search by requiring that a set of nodes of interest are recorded as the search proceeds upwards. Once a condition given by the command list is satisfied, the search proceeds downwards from these nodes of interest, looking for possible implicit ancestors of the black node. Their relation to the black node is confirmed using subsumption. The minimal set of such nodes is determined and connected to the black node. The process then iterates, searching upwards from the minimal set just determined.

$\mathcal{E}$ is a set of Paths to investigate above the black node. $\mathcal{V} :: \text{Nodes}$ is the set of previously visited nodes. Each $\mathcal{N}_y :: \text{S-nodes}$ is a set of nodes associated with event type $y$ ($:: \text{Action-Node}$). Initially $\mathcal{E} = \{(bn, \text{Def}, \{\})\}$, $\mathcal{V} = \emptyset$, and $\forall y \mathcal{N}_y = \emptyset$. $\mathcal{C}$ is the command list less any command referring to the action of an event connected to $bn$: Beige and Khaki node integration dealt with those events and connected $bn$ to all its implicit ancestors below the orange node.

1. **The head $c$ of $\mathcal{C}$ is removed from $\mathcal{C}$: the first command.**

2. **The Lowest First Upwards Search:**
better\((s_1 :: \{\text{Def, Obs}\}, s_2 :: \{\text{Def, Obs}\})\) returns the better status of \(s_1\) and \(s_2\).

\[\text{depth}(n :: \text{Node}) = d :: \text{Depth}\] is the depth of node \(n\).

\[\text{node}((n, s) :: \text{S-node}) = n :: \text{Node}\]

\[\text{status}((n, s) :: \text{S-node}) = s :: \{\text{Def, Obs}\}\]

\[\text{node}((n, s, S) :: \text{Path}) = n :: \text{Node}\] Current point of search.

\[\text{status}((n, s, S) :: \text{Path}) = s :: \{\text{Def, Obs}\}\] Status of path from \(n\) to \(bn\).

\[\text{detected}((n, s, S) :: \text{Path}) = S :: \text{Action-Nodes}\]\n
Set of actions corresponding to the events that appeared along the path and that are to detect.

\[\text{node}((n, d, S) :: \text{Command}) = n :: \text{Action-Node}\]

\[\text{depth}(n, d, S) :: \text{Command}) = d :: \text{Depth}\]

\[\text{to\_detect}((n, d, S) :: \text{Command}) = S :: \text{Action-Nodes}\]

\[\text{previous}((p, c, (s, o)) :: \text{D-path}) = p :: \text{Node}\] Previous node on path.

\[\text{current}((p, c, (s, o)) :: \text{D-path}) = c :: \text{Node}\] Current node of path.

\[\text{status}((p, c, (s, o)) :: \text{D-path}) = s :: \{\text{Def, Obs}\}\] Status of the path's origin with respect to \(bn\).

\[\text{so}((p, c, (s, o)) :: \text{D-path}) = (s :: \{\text{Def, Obs}\}, o :: \text{Node})\] Status \(s\) and origin \(o\) of the path with respect to \(bn\).

(a) \(p\) is a path with a deepest node of \(E\). (Two nodes may share the same depth.) Let \(ln = \text{depth}(\text{node}(p))\).

(b) \(p\) is a path with a deepest node of \(E\). If \(\text{depth}(\text{node}(p)) \leq \text{depth}(c)\), the algorithm goes to 3. Otherwise, \(p\) is taken out of \(E\), and the algorithm continues.

(c) \(I\) is the set of the actions of the events attached to \(\text{node}(p)\) and in \(\text{to\_detect}(p)\).

\[\forall i \in (I \cup \text{detected}(p)), (\text{node}(p), \text{status}(p))\] is added to \(N_i\).

(d) \(\text{node}(p)\) is added to \(V\).

(e) The set of \(\text{node}(p)\)'s parents not in \(V\) is found. Each parent \(x\) is annotated with the worst of \(\text{status}(p)\) and the status of the set relation between \(x\) and \(\text{node}(p)\); and with \(\text{detected}(p) \cup I\). The thus annotated parent is \(\alpha(x)\). Each parent \(x\) is looked up in \(E\):

- If no path of \(E\) has as node \(x\), then the annotated parent \(\alpha(x)\) is added to \(E\).
- Otherwise the path \(p(x)\) which has \(\text{node}(p(x)) = x\) is replaced by \((x, \text{better}(\text{status}(\alpha(x)), \text{status}(p(x))), \text{detected}(\alpha(x)) \cup \text{detected}(p(x)))\).

(f) The algorithm iterates to (b).
3. Find the highest descendents of \( a_i(t(node(c))) \) which are not explicit ancestors of the black node.

\( \mathcal{X} \) is the set of nodes to expand, and \( \mathcal{H} \) is the set of highest descendents. Initially, \( \mathcal{X} = \{a_i(t(node(c)))\} \) and \( \mathcal{H} = \emptyset \).

(a) A least deep node \( x \) of \( \mathcal{X} \) is taken out of \( \mathcal{X} \).

(b) The set \( D \) of \( x \)'s children is determined. For each \( d \in D \) such that

\[
\text{depth}(d) < \text{ln}:
\]

- If \( \exists d' \in \mathcal{N}_{node(c)} \) such that \( node(d') = d \), then add \( d \) to \( \mathcal{X} \).
- If \( d \not\in \mathcal{V} \), and if \( d \) is a definitional\(^{28}\) child of \( x \), then add \( d \) to \( \mathcal{H} \).

(c) The algorithm iterates to (a) until \( \mathcal{X} = \emptyset \).

4. The Highest First Search Down:

\( \mathcal{H} \) is the set of highest descendents of \( a_i(t(node(c))) \) which are not explicit ancestors of the black node, and was determined by the previous stage. \( \mathcal{X} \) is the set of D-paths representing paths to expand. \( \mathcal{A} \) is a set of S-nodes expressing the set of implicit ancestors to connect to \( bn \), and their status with respect to \( bn \). Initially, \( \mathcal{A} = \emptyset \) and \( \mathcal{X} = \emptyset \).

(a) For each \( h \) in \( \mathcal{H} \):

i. Find the set \( \mathcal{P}(h) \) of \( h \)'s definitional parents.

ii. For each s-node \( n \) of \( \mathcal{N}_{node(c)} \), such that \( node(n) \in \mathcal{P}(h) \), add \( (node(n), h, (\text{status}(n), node(n))) \) to \( \mathcal{X} \).

(b) \( \mathcal{Y} \) is a set of D-paths, such that all D-paths of \( \mathcal{X} \) with with current node \( c \) are in \( \mathcal{Y} \), and such that

\( \forall y \in \mathcal{Y} \forall x \in \mathcal{X}. (\text{depth}(\text{current}(y)) \leq \text{depth}(\text{current}(x))) \)

(c) If all the statuses of the D-paths of \( \mathcal{Y} \) are observational, goto (h)

(d) If all of \( c \)'s definitional parents appear in the previous nodes of the D-paths of \( \mathcal{Y} \), and if \( c \) is not connected to a definitional event, \( c \) subsumes \( bn \). Otherwise, test \( c \) for subsumption. Goto (h) unless \( c \) subsumes \( bn \).

---

\(^{28}\)6.7.1.2 (p. 194) argued that only definitional descendents were needed.
(e) The worst status \( s \) of all D-paths of \( Y \) is found. \((c, s)\) is added to \( A \).

(f) Find \( P(c) \), the set of all of \( c \)'s definitional parents. For each s-node \( a \) of \( A \), such that \( \text{node}(a) \in P(c) \), if \( \text{status}(a) = s \), \( a \) is removed from \( A \).

(g) Find \( D(c) \), the set of \( c \)'s definitional children. For each node \( d \) of \( D \), a set of D-paths \( Z(d) \) is added to \( X \), where
\[
\forall y \in Y \exists ! z \in Z(d). \text{previous}(z) = c \land \text{current}(z) = d \land \text{so}(z) = \text{so}(y).
\]

(h) The algorithm iterates to (b) until \( X = \emptyset \).

5. **Connect detected implicit ancestors to the black node:**
   \( \forall a \in A, \) a spec. event of status \( \text{status}(a) \) is built between \( \text{node}(a) \) and \( bn \).

6. **Iterate:**

   To reduce memory consumption, \( N_{\text{node}(c)} \) is set to \( \emptyset \).

   \( \forall e \in E, e \) is replaced by \( (\text{node}(e), \text{status}(e), \text{detected}(e) - \{\text{node}(c)\}) \).

   \( \forall a \in A, (\text{node}(a), \text{status}(a), \emptyset) \) is added to \( E \).

   The algorithm iterates to 1 until \( C = [\] \).

The formal algorithm differs a little from the algorithm given in the introduction of this section. Indeed, a new third stage is introduced which starts not at the nodes of interest, but at the node connected to \( y \)'s template event \( t(y) \) \((y = \text{node}(c))\). Also, the second stage records in \( N_y \) all nodes (below \( a_i(t(y)) \)) that have as descendent a node connected to \( t(y) \). This change deals with the fact that because of inheritance, not all nodes qualified by an event of type \( z \) are actually connected to one. The nodes in \( N_z \) below \( a_i(t(z)) \) are known to be qualified by an event of \( z \)'s type and are on paths above the lowest node connected to an event of \( z \)'s type. Thus they are integrated with respect to the black node, but their descendents which are not in \( N_z \) may not be. This allows the third stage to determine the fourth stage's starting nodes.

\( V \) is necessary because the regular downward searches break the guaranty of the lowest first upwards search that a node will not be traversed twice. No node is added to \( V \) during the downwards search since each such node may have parents that are connected to a parent clan. To minimise \( V \)'s cost, nodes from the clan
below the current one (node(c)) can be flushed at 6.

Unlike the other algorithms, no complexity result will be given. Because all events connected to the black node can be integrated simultaneously (by initialising $E$ of the skeleton algorithm with all events connected to the black node, instead of one of them at a time), the search space considered is the same as that of upwards inheritance from the black node, but with additional space due to the successive downwards searches. The real complexity value will be determined mostly by the cost of subsumption testing. In fact, even the question of whether above-orange node integration should be performed is by no means certain: the cost of above-orange node integration may be prohibitive, and the information it brings unnecessary to many successful human activities. For instance, it may prevent real-time dialogue or translation. This goes against the precept that A.I. is concerned with the simulation of successful human behaviour. There is no need for it to better human behaviour. And if the question is asked whether people always relate every new fact with all the facts of their knowledge that subsume it, the answer is likely to be no. For instance one may be told at some point that all objects that fly must be light. Some time later I am told the Space Shuttle took off. In hindsight the conclusion that the Space-Shuttle is light is trivial, but in foresight it may not be quite so obvious. If incompleteness is the price required for successful human behaviour such as communication in real-time, as long as its lack does not also prohibit communication, it must be paid. Thus above-orange node integration may be worth discarding or performing at a later time.

6.7.3 Real time integration

6.7.3.1 Problem: Semantic Integration is inherently expensive

Full semantic integration is expensive. The cost stems mainly from the child determination algorithm, which must consider all the descendents of the new concept's parents to make all the concept's implicit children explicit. Search techniques, more sophisticated than those presented above, can reduce search space significantly by
restricting the concepts considered to those most likely to be descendents of all
of the concept’s parents. However, in the worse case, when the concept’s parents
are very generic, the search space may be most of the semantic network: potential
customers may be anywhere in it.

Semantic integration is therefore a serious concern for an NLP program which
aims to be efficient, and responsive to the user. The inheritance hierarchy was
introduced as a useful means of structuring knowledge allowing efficient implementa-
tions reasoning algorithms such as inheritance, necessary for large scale K.B.s.
However, it has just been shown that building the inheritance hierarchy is also very
expensive, rendering NL analysis very slow. In effect the burden has been moved
from reasoning to building the representation. It would therefore seem that one
must pay either during NL analysis or during reasoning: One cannot have one’s
cake and eat it.

6.7.3.2 The light at the end of the tunnel...

There is however a solution to this apparent conundrum. It lies in four observations:

- Most searches along the inheritance hierarchy are upwards. Downward searches
  are avoided since they can be very expensive.

- Each night, there is an idle period when office computers are left unused. This
  is an ideal opportunity for intensive data processing which is unresponsive,
  such as semantic integration.

- Each day, the new information added to the K.B. will constitute a small
  fraction of the knowledge already in it. Suppose that the data added to the
  network during the day is integrated at night. The data added each day
  constitutes a small fraction of the K.B. so most of the K.B. will be correctly
  integrated. Only a small fraction will not. Thus most reasoning can take
  advantage of the efficiency gains afforded by integrated knowledge.

- It is rare to need to refer or reason about any descendant of a newly introduced
  concept. Thus, the cost-benefits of integrating new nodes are rarely realized
immediately after data is added to the network.

In this light, it would be possible to avoid integrating new data until the evening if reasoning can exploit the speed-gains given by the fact most of the K.B. is integrated, while not suffering too harsh a penalty to also use un integrated information.

6.7.3.3 ... is not an incoming train!

- Inheritance: Upward search and Unification

Complete inheritance algorithms involve two separate processes: finding information from a concept's ancestors, and unifying different expressions of the same concept at different levels of the inheritance hierarchy. For instance, suppose that one knows that "All people who own animals love them", and "all farmers own a donkey". Finding information from farmers' ancestors involves finding the first statement: All farmers own a donkey, and love the animals they own. Unifying multiple expressions of the same concept is inferring: All farmers love the donkey that they own.\(^\text{29}\).

An integrated semantic network is used to find information, rather than to unify different expressions of the same concept. Thus the discussion will focus on how information is found. Finding all the facts about a concept's ancestors involves finding the set of all the concept's ancestors. Every fact true of a concept of that set is also true of the concept, since SemNet is not defeasible. In a completely integrated network, finding the set of all of a concept's ancestors is simply an issue of traversing the subnetwork of spec_ events between the concept and typeless from the concept and from each spec_'s object_ to its subject_. In a partially integrated network, this process will miss out some of the concept's ancestors, and thus fail to make some expected inferences. A means of ensuring these inferences are made must be found.

- Partial Integration: Upward Integration

As was stated earlier, the main cost of semantic integration comes from downward

\(^{29}\)(and all other animals that they might happen to own.)
integration: determining a concept’s implicit children. Determining its implicit ancestors (upward integration) is much cheaper involving a limited part of the semantic network. Further, a lot of reasoning about new concepts is performed during NL analysis, so efficient access to all information inherited from its ancestors is important. Thus, it is well worth performing upward integration. 30

• The extension to Upward Search

The remaining problem is determining all the parents of concept to which it ought to be, but is not connected. Recall that the descendents that downward integration of a node finds, are the descendents of all of the nodes definitional parents. Thus, an upward search from a concept which has an implicit parent will traverse all of this parent’s definitional parents. Detecting the full set of definitional parents of any unintegrated node during the upward search, provides the key to efficiently finding the implicit parent: only if all the definitional parents have been reached by the upward search is the concept an implicit child of the unintegrated node.

The upward search must recognise definitional parents as different from other nodes. Moreover, since there will be more than one unintegrated node in the network, each definitional parent must be associated with the unintegrated node(s) of which it is a definitional parent. Finally, to determine whether all definitional parents of an unintegrated node have be reached, a count of the number of definitional parents each unintegrated node has must be maintained. This count can either be associated with each definitional parent, or in a separate data-structure available to all upward searching processes.

By maintaining a record of already encountered definitional parents (EDPR), the upward search can easily detect unintegrated nodes that should be visited. Each time a new definitional parent is encountered, the unintegrated nodes of which it is the definitional parent are entered into the EDPR. When an unintegrated node

30Upward integration performed without downward integration may not be complete: Upward integration assumes that all of the network, other than the new concept, is completely integrated. Since downward integration is not performed, a concept’s ancestor may lack explicit connections to its children, which should become the concept’s parents. The lack of connection may mean that a concept is not connected to all of its parents. However reasoning can deal with this in the same way as it does the lack of downward integration: all the implicit children of a concept also lack a connection to all of their parents.
is entered into the EDPR for the first time, the number of its definitional parents is looked up, and inserted into the record less one. One is subtracted since one definitional parent has already been found. Every subsequent time a definitional parent of the unintegrated node is found, the number of definitional parents to find in its EDPR-entry is decremented. When zero is reached, all the unintegrated node's parents have been found, and the unintegrated node is known to be one of the ancestors the upward search must find. Further all of the unintegrated node's ancestors are also ancestors to be found by the upward search. Thus, the unintegrated node is simply inserted into the set of nodes to consider next by the upward search.

Since the inheritance hierarchy is a graph, it is possible for an upward search to reach the same node at different points of its search, having traversed different paths. To avoid considering such nodes' parents many times, the list of already encountered nodes (AEN) states the nodes not to expand. This AEN has to be built anyway, since when completed it will be the result of upward search: the set of ancestors. It is because this data-structure ensures that no node will be reached more than once, that the strategy of decrementing the number of definitional parents to find in the EDPR, each time a new definitional parent is reached works.

6.7.3.4 Implementation issues

- The need for an efficient data-structure

Since the EDPR is key to the efficiency of this approach it must be as efficient as possible. Looking up only once the number of definitional parents an unintegrated node has helps a little, but the main concern is that the unintegrated node's entry can be accessed quickly. A simple array would provide constant time access but would require one entry for every node of SemNet, which is prohibitively expensive. Balanced trees such as AVL trees provide lookup speeds of order $\log(N)$ where $N$ is the number of elements in the tree, but in situations where data is inserted more often than it is looked up, the balancing cost becomes significant: $i_1.f(N)$ insertion and $l_1.f(N)$ lookup where $f(N) \approx \log(N)$ may prove faster on the ground
than a $i_2 \cdot \log(N)$ insertion and $l_2 \cdot \log(N)$ lookup, if $i_2 \gg i_1$\textsuperscript{31} even if $l_1 \gg l_2$ and if insertion is performed far more than lookup. More data is inserted than read from the EDPR since on average most of the unintegrated nodes encountered when searching upwards will not be implicit parents of the starting node. Similarly the AEN data-structure will be more often written to than read, since every node traversed will be added to it, while only nodes reachable by separate paths will be read. Skip lists [Pugh 90] provide a possible solution, since on average they are balanced while not requiring any time spent on rebalancing.

The BIB tree presented earlier provides an ideal data-structure, since it has a near $\log(N)$ access efficiency, but avoids the costs of rebalancing. Lazy evaluation makes this is particularly important since the fact the path assignment mechanism is deterministic: the path through the tree depends on the bit-string of the index, rather than the history of rebalancing. As a result, only nodes of the BIB-tree along the path dictated by the index’s bit-string need be evaluated. Schemes involving tree rebalancing will require other nodes of the tree to have been partially evaluated to determine in which direction the rebalancing should rotate them.

- **The overall time complexity of extended upward search**

Searching upwards is a linear operation in the number of spec_ or inst_ events from the concept upwards, despite the fact that multiple paths may reach the same concept. However this is due to the maintenance of the AEN data-structure which has $\log(N)$ access efficiency. Since this access must be repeated at least once for every node the upward search visits, time complexity is bounded by $MN.\log(N)$ where $M$ is the maximal number of children any concept in SemNet has. Extending the search to include unintegrated children requires the additional EDPR data-structure which has the same access efficiency as the AEN (being the same algorithm) but is invoked for fewer nodes since only a small section of the K.B. is changed in a session. Thus, extending the search does not increase the time complexity of upward searches.

\textsuperscript{31}E.g.: if $0 < N < 1024$ then $i_2 = 1024 \cdot i_1$
6.7.3.5 Conclusion

A means has been found to reason efficiently even with an unintegrated network. It is useful as it allows one to delay downwards integration, an expensive operation, until a more convenient time. It does not however enable downward searches. Indeed, a full downward search means finding all descendents which is the task of downward integration. Hence any full downward search on an unintegrated network can only be as efficient as the best algorithm for downward integration.

6.7.4 Multiple Node Integration

Only single node integration has been considered so far. However not all statements can be integrated in this way. This section determines the class of statements that can be integrated by single node integration, and those requiring multiple node integration. The additional processing required for multiple node integration is then discussed. Finally the ways in which nodes can be combined so that much of the search space they must explore can be shared are considered.

6.7.4.1 Motivation

Why should nodes ever have to be integrated simultaneously? Recall the essence of integration: to find the implicit ancestors and descendents of a concept. Whether a node is an implicit ancestor or descendent clearly depends on the new concept's definitional restrictions. What if these restrictions are also new to the network? Clearly these restrictions should be integrated first, since the integration algorithm assumes that the network is always perfectly integrated. This is all well and good if the restrictions do not themselves depend on the new concept. However, it is possible to have definitional cycles in SemNet. For instance:

\((E_0, R): \{\Delta F_{subject}\_F\Delta: married\_poor\_men; \Delta V_{action}\_I\Omega: husbanding; \\
\Delta F_{object}\_F\Delta: poor\_mens\_wives\} \)

*married\_poor\_men* restricts the event *E_0* by the characteristics it inherits from *poor\_men*, and is restricted by *E_0*. *poor\_mens\_wives* restricts the event *E_0* by
the characteristics it inherits from women, and is restricted by $E_0$. $E_0$ restricts married_poor_men by its action_husbanding and its object_poor_mens_wives, and it restricts poor_mens_wives by its action_, and its subject_married_poor_men.

Adding any of the three concepts to the network independently from the other would lead to an incomplete definition, which would not be correctly integrated, and could possibly violate uniqueness. Since these concepts cannot be added separately, only two avenues are left: integrate a partially integrated network, or integrate the concepts simultaneously. The first option is unattractive since many of the optimizations on search space take advantage of the fact the network is fully integrated. Instead, the second option is chosen.

### 6.7.4.2 Nodes that must be integrated simultaneously

The problem illustrated so far lies with definitional cycles, where some of the restrictions on a concept are themselves restricted by it. In SemNet, this trivially occurs for the $\Delta - \Delta$ arcs: All nodes that are connected by $\Delta - \Delta$ arcs forming a single (unbroken) subgraph must be integrated simultaneously. Such sections of graph are called integration subgraphs.

When creating a new concept, node addition must add all its definitional restrictions simultaneously to the network: once a concept has been given a set of restrictive events, none more can be added since that would change its meaning. What is true for the new concept also applies to its restrictions, if they are new: to be added they must be fully defined.

Because integrating large subgraphs incurs a speed penalty, it is best to add the minimum necessary at a time. Adding concepts in any order cannot achieve this, since all the concepts on which they depend definitionally must also be added. The task is thus to determine the best order in which the smallest groups of mutually dependent concepts can be added to SemNet.

This is simply achieved by associating each integration subgraph formed by $\Delta - \Delta$ arcs, as described above, with a vertex of a dependency graph: the dependencies
are expressed by directed edges from a vertex to the vertices of the subgraphs on which its definition depends. Each subgraph associated with a vertex that depends on no other, can be added separately. When it is added to SemNet, its vertex in the dependency graph is deleted, possibly leaving new vertices depending on no other. The algorithm can then iterate.

So what about the cases in which vertexes are left, but all of them depend on another? Clearly they express further mutual dependencies, but where do they come from? All the \( \Delta - \Delta \) arcs were "consumed" when the subgraphs formed by \( \Delta - \Delta \) arcs described above were built and associated with vertices of the dependency graph. The \( O - O \) arcs do not express any dependency so are not built into the dependency graph. Remain the \( O - \Delta \) arcs, which are forming a mutual dependency. Indeed, this is possible, as the sentence "Every person who made the sandwich he ate for lunch, please stand up" illustrates.

\[(E_0,R): \{\Delta F\text{-subject}.-F\Delta: \text{sandwich.makers}; \Delta V\text{-action}.-IO: \text{make}; \Delta F\text{-object}.-FO: \text{eaten.sandwich}\}\]

\[(E_1,R): \{\Delta F\text{-subject}.-FO: \text{sandwich.makers}; \Delta V\text{-action}.-IO: \text{eat}; \Delta F\text{-object}.-F\Delta: \text{eaten.sandwich}\}\]

Detecting such cycles is easier using the dependency graph. The failure to deliver an independent vertex performs the first part of the problem: cheaply detecting cycles. The vertices of the dependency graph represent subgraphs of SemNet, thus reducing the search space to consider.

The next problem is to find the cycle just detected. Call the direction of edges expressing dependency as down. A cycle in the graph will mean that a sequence of vertices will appear again above it. Conversely it will also appear below it. Thus by searching the graph consistently in one direction, the cycle will be identified. The question remains as to which direction should be chosen. Searching downwards does not guarantee that vertices potentially in cycles will be traversed. Indeed, it will will lead to vertices on which no vertex depends. Since these vertices correspond to the subgraphs that must be added last, visiting them serves no useful purpose. Searching upwards will always visit the subgraphs on which the current vertex depends. Thus, it will visit any cycle of dependency above the start vertex. There
must be at least one such dependency, since otherwise there is a vertex which depends on no other, contradicting the fact none was found.

Finding cycles can thus be achieved by a search upwards. Instead of the usual breadth first search, a depth first search is employed. Indeed, a depth first search ensures that cycles are found first, since none of the vertices depend on no other: the only way the first explored path of the search can terminate is when it has reached a previously encountered vertex. Previously encountered vertices are only along the single path explored so far, so a cycle must have been encountered. The cycle is simply the set of nodes traversed by the path upwards from the previously encountered vertex, to the previously encountered vertex. The detected cycle is collapsed into a single vertex of the dependency graph. If the resulting graph has a vertex independent of all others, the normal adding of information to SemNet resumes. Otherwise another cycle is detected in the same manner.

The efficiency of finding a cycle clearly depends on the vertex chosen to start the search. Since choosing a good starting vertex requires a good idea of the solution, it appears to require many computations. Instead a cheap heuristic is used, so that the time is spent on finding the cycle, rather than a good starting vertex. Indeed, NL sentences tend to be relatively short and self-contained: definitions do not span pages, but are expressed in terms of others, which will previously have been exposed. Thus, the choice will rarely be critical. The chosen vertex is one of those that depended on a vertex that has recently been deleted, to avoid having to search the graph. Hopefully the fact it depends on a subgraph recently added to SemNet will make it likely to be the next to be deleted\(^\text{32}\). Of those vertices dependent on recently deleted nodes, the choice will be a vertex that depends on few others (so that the depth first search has a higher chance of choosing the path leading to the cycle faster... if the vertex is in it), and is depended on by many (at least one of them might then depend on no other at the next iteration).

Determining a cycle is bound by \(O(N)\), where \(N\) is the number of vertices in the dependency graph: the search is depth first. Complex cycles are dealt with by

\(^{32}\)This is not wholly superstition: people tend to talk about a small set of concepts at a time
more than one pass: when a cycle is collapsed, the subgraphs to which its vertices correspond are merged.

6.7.4.3 Integrating subgraphs

The previous section determined the sets of nodes that must be integrated simultaneously. This section explains how the subgraphs they form are integrated.

The difference between integrating single nodes and subgraphs results from the guarantee that all nodes except the single node itself are fully integrated for single nodes. For subgraphs, none of the nodes of the subgraph are integrated. This means that many of the improvements in search space that have relied on a fully integrated network are thrown into question.

Subgraphs are only added when there is some mutual definitional dependency. Since concepts are defined by their ancestors as well as the new events that further restrict them, each concept which has more than one parent can be partially integrated upwards with respect to its parents. This integration can be performed on each of the nodes separately since the relation between them is being ignored: these relations further restrict the concepts concerned, and the restriction of a concept has the same ancestors as the full form of the concept. 33 This step will thus establish red nodes for each node, and perform the usual khaki and beige node manipulations for each. However, unlike for single node integration, subsumption tests that fail must be reconsidered when the relations between elements of the subgraph are processed. Thus the minimal set of beige parents which is determined is temporary: The nodes of the temporary minimal parent set are connected by temporary set relations to the new concepts. Similarly, the khaki node integration may have determined that the new concepts are descendents of existing khaki nodes, in which case the set of each new node’s parents is suitably amended. The set of subsumption failures for both beige and khaki nodes is retained.

Now comes the really expensive step: fixpoint iteration. Once all the information

---

33 This is not the case for descendents, hence the partial integration is only upwards.
that could be derived ignoring the relations between the nodes has been found, the
relations have to be considered. They will probably affect some of the subsumption
cases of the nodes which failed the previous subsumption test. When all of these
failed nodes have been rechecked, those which now succeed are processed, leading
to a new temporary minimal set of parents. This is again, if necessary, connected by
temporary set relations, and the process iterates until no change in the number of
failed nodes occurs. This iteration process is necessary since new information about
some part of the subgraph may affect its neighbours in subsequent iterations. The
information then may propagate through the subgraph. For instance the object
of an event $e$ may become marked as the descendend of a concept, which happens
to be connected to an event, which can now be the ancestor of $e$, because of the
object.'s integration... affecting the event's subject., which now can be... Once
the fixed point calculation terminates, downwards integration can occur as for
single nodes.

Clearly, clever techniques must be employed to ensure that failed nodes are only
retested for subsumption if the status of their potential descendend changes in a way
making this feasible. For instance, if information about an event's location changes,
there no point rechecking a node's potential subsumption of it, if the failure was
due to insufficient temporal information on the new node. Similarly, if the failure
was due to disjointness, there is no point retesting it, since further restrictions will
not change disjointness. This imposes a requirement on the subsumption reasoning,
namely that it provides types of failure; if the failure is due to lack of information,
the type of information needed; and potentially some record of salvageable work
completed so far, so that this work need not be repeated.

Nevertheless, clever techniques do not remove the fact that testing for subsumption
is not cheap, and doing it many times is definitely ruinous. Since more iterations
are required for information to propagate through a large subgraph, the smaller the
subgraph the better. This explains the effort devoted in the last section to finding
the smallest subgraphs that can be added at a time.

No time complexity results are available, since the behaviour depends on the topol-
ogy of the existing information in SemNet rather than the input data. It also
depends strongly on how much additional work a new call to the subsumption al-
gorithms really involve. Only empirical studies on a large database incorporating
a lot of information and with subsumption determiners for each of a large range of
types of reasoning, can establish whether this fixed point calculation is on average
usable in realistic systems. LOLITA is unfortunately not yet up to this.

6.7.4.4 Opportunities for improving efficiency

The previous sections discussed cases where multiple node integration is detrimen-
tal to efficiency. However this section discusses cases where exploration of search
space can be shared, reducing the overall cost. Unlike integration of subgraphs,
these cases would actually be implemented as single node integration, but the re-
results of searches would be carried over to subsequent processing.

Integrating multiple nodes downwards can be more efficient per node than inte-
grating each node singly. This is because the main cost of downwards integration
is the search down the hierarchy to find all of a black node’s implicit descendent.
This search can be shared by any black nodes which share red nodes in common:
if there are two black nodes $a$ and $b$, such that $\mathcal{A}$ is the set of $a$’s red nodes, and $\mathcal{B}$
is $b$’s red nodes; and if $\mathcal{A} \subset \mathcal{B}$, then all of $a$’s implicit descendent will be traversed
when searching for all of $B$’s implicit descendent.

Integrating multiple nodes requires some changes to the algorithm for integrating
a single node. In particular, $\mathcal{I}$ is replaced by the sets $\mathcal{I}_{bn}$ where $bn$ is a black node,
and $\mathcal{I}_{bn}$ is that black node’s implicit descendent. Similarly, each black node $bn$ is
associated with the set of its red nodes $\mathcal{R}_{bn}$, found in the same way as in 6.7.2.1 (p.
201). Now, when information is added to $\mathcal{V}$, the set of origins of each manipulated
node is compared with each of $\mathcal{R}_{bn}$: if the set of origins is equal or greater than
$\mathcal{R}_{x}$, the information to add to $\mathcal{I}$ of 3 (p. 204) is added to $\mathcal{I}_{x}$. The resulting sets
$\mathcal{I}_{bn}$ are then each processed to produce the minimal sets of descendants $\mathcal{D}_{bn}$. 
6.7.5 Subsumption in the light of KL-ONE's results

In 4.7.3 (p. 91), the worst case time complexity of subsumption for KL-ONE was shown to be NP for most (useful) terminological languages. This may seem rather worrying as NLE requires reasonable response times. The line followed here follows that of [Woods 91]: "the subsumption should be sound – i.e. every instance of the subsumed concept should be an instance of the subsumer. However the converse need not be the case – i.e. that is structural subsumption should entail extensional subsumption, but not vice-versa". In practice, this means that if two concepts are not connected by a spec. or inst. event, it does not mean that no such relation exists between them, only that none is known/has been determined. In essence, then, at worst, the subsumption algorithm can be incomplete.

Clearly, if a new representation is introduced for which reasoning in the worst case is intractable or undecidable, the subsumption algorithm may either be NP or incomplete. For instance, this may well be the case for the values representation: the constraints expressed by a graph stating the size of a set, or stating the time of an event may make subsumption undecidable in some cases. However, because many constructs of KL-ONE languages make subsumption in them NP does not mean that this must also be the case for SemNet. For instance [Donini et al. 95] lists the causes of non-polynomiality:

- \( \mathcal{ALU} \) (Union of concepts): Does not exist in SemNet. The closest is to define a concept by a disjunction of definitional events, e.g.: "a man under 15 or over 30 years of age". However, unlike a union of concepts, all disjunctive events must be expressed at the level of the concept. This breaks the combinatorial growth union introduces: let \( C_n = C_{2n} \cup C_{2n+1} \) and \( \forall n > m \ C_n = A_n \) where all \( A_n \) are atomic concepts. Then the expression of \( C_0 \) in terms of atomic concepts is of exponential length in the depth of concept union. Because the unsatisfiability algorithms consider each term of a union individually, their overall complexity becomes NP.

- \( \mathcal{AL+E} \) (Full existential quantification): Does not exist in SemNet in the way it is used in KL-ONE. In SemNet, existential quantification refers to all
elements of a concept, thus corresponding to universal role quantification in KL-ONE terms: The role of quantification is limited in SemNet to the choice of elements within the concepts: the concepts may even lack any instances! In KL-ONE, quantification also determines the number of instances (at least one).

- $\mathcal{AC}$ (Complement of non-atomic concepts): These are rarely defined in SemNet, corresponding to the antonym of the concept and typeless. Only explicit antonyms in the network need be considered, reducing the search space to the template antonym's descendent events: intensionally, a human may be male and female in some frame of existence. Only an explicit statement of antonymity may change this.

- $\mathcal{AR}$ (Role intersection): Does not exist in SemNet. The closest thing is the events whose action.s are defined by others. All instances of such events inherit all the events that define them, so they are no worse than shorthands for sections of network. Role intersection's bad complexity comes from the possibility it introduces of simulating full existential quantification in KL-ONE: $\exists \tilde{R}.D = \exists(R \cap P_D).T \cap \forall(R \cap P_D).\tilde{D}$ Ignoring the details, one can see that existential quantification is being achieved by the back door using role intersection.\(^{35}\)

That SemNet does not appear to have the same causes of complexity as KL-ONE is encouraging in that it suggests the basic representation is a solid basis on which to build. If no other source of complexity is found in the basic representation, the majority of the network operations can be performed by complete algorithms, reducing the information lossage.

\(^{34}\)It may be subject to the danger of terminological cycles of the form discussed by [Nebel 91], but these do not appear to occur in practice.

\(^{35}\)See [Donini et al. 95] for more details
6.7.6 Conclusion

This section introduced the problem of semantic integration, discussed algorithms to solve it, and presented future problems to resolve. Clearly, the baseline depends on the average time complexity of subsumption. Reasons why one can be hopeful for it to be cheap were discussed, but only empirical investigation or some formal calculation will tell. In both cases, this can only be performed when the relevant reasoning algorithms have been defined.

Many aspects have been ignored, for instance the treatment of synonym events, which are used to deal with "cyclic" subsumption which can occur when one concept has the definitional restrictions of the other as observational and the other's observational information as definitional. Similarly many optimizations have been ignored, principal among which the reduction of average complexity using the fact that all subsumption tests are performed with respect to the black node. Thus often parents and children are tested subsequently for subsumption, so many results can be reused if memoized.

No claim is made for the completeness of the above algorithms. They appear sufficient for the reasoning undertaken in LOLITA so far, but only formal modelling could justify such a claim, and it was decided in 3.3.1.1 (p. 32) not to consider this issue in this thesis.
Chapter 7

Extended Representation

Natural Language involves many domain-dependent phenomena. These must be modelled if the representation is to be rich enough to express the meaning of a wide variety of texts. The representation is built upon the domain-independent representation discussed in the previous chapter.

A wide range of phenomena are discussed: Not only are phenomena explicitly discussed in texts modelled (Values, Belief, Negation, Parts) but also implicit phenomena within the text that are needed for interpretation: the mapping from words to concepts, issues of Intension, Textrefs and Ambiguity.

7.1 Representation of Values

7.1.1 The need for Values

The discussion has so far centered on two-valued propositions: either someone is a man or he is not. Either an object is owned or it is not, and so on. However, such a view is rather limited. Many phenomena involve some form of gradation or intensity. Indeed, the whole class of comparative and superlative adjectives requires such a notion: "John is nicer than Mary", or "Alfred is heavier than Margaret". Similarly notions such as time or money involve such orderings: "John arrived after Tony", "This coat is more expensive than the other". In natural language, many adverbs involve gradation: "John ran faster than Andrew", or "Mary plays the
violin better than Alexandra". Even verbs can refer to notions of intensity: "Alfred weighs more than Margaret".

7.1.2 Unquantifiable and Quantifiable Values

This general notion of gradation, or intensity is represented by a new class of concepts: value types. It is traditional in computer science to reach for numbers as soon as a notion involving some form value is considered. However, this is not the approach taken here. People express certain values as numbers, and others not. This indicates the way in which they find it useful to conceptualize these values. Since numbers are an additional abstraction away from reality, they can be considered to involve more effort. In particular in order to express some phenomenon in terms of numbers a unit value must be determined.

In some cases, determining a unit value for a particular value type is obvious, for instance when dealing with discrete entities\(^1\): There are five cows on the meadow. In other cases, it is less obvious, for instance when dealing with continuous phenomena: the room is five metres by six. In this case, a unit must be determined. Often many units may exist for such partitionings of continua: pounds, stones, kilograms, ounces; metres, yards, feet, Ångströms, light years... Finally there are cases where determining such a unit value does not appear to be worth the effort for human purposes. For instance there is no unit for anger so one would not say "John is fourteen angries angry", but one could still say that "John is angrier than Russell". It should be noticed that in both the many unit case, and the no unit case, there is still some underlying notion of intensity.

One could wonder why one should not have two different representations, one for values of quantifiable type (those to which a numerical value can be assigned), and one for values of unquantifiable type. The problem with such an approach is that it does not reflect the great similarity in the way people talk about both kinds of value: "John is very happy" and "John is very heavy" are expressed identically.

\(^{1}\)or more precisely entities that were partitioned as discrete parts of the world by the partitioning process described in 2.4 (p. 19)
even though the latter refers to a quantified value type. Indeed, it is possible for a value of quantifiable type to be described in such a manner that no numerical value is ever assigned to it: "John is heavier than Mary and Ann, but he is lighter than Jacob". Unless Jacob and Mary or Ann weigh the same weight, John's weight cannot be determined. This sort of statement is no different to a statement involving unquantifiable value types, so there is no reason to postulate an additional representation for it.

Again one could postulate that all values correspond to numbers. But this corresponds to an additional unnecessary assumption, an entity without necessity in Ockham's terms. It is useful to remember at this point that LOLITA is to simulate human behaviour accurately. In particular, behaviours that humans would not produce are not wanted. Representing numerically concepts which people do not conceptualize as numbers can lead to such undesirable behaviour: reasoning might start referring to the precise numbers assumed leading to unforeseeable errors. Similarly representing notions such as "very hot" by numbers picked out of the air might lead to aberrant behaviour such as "Today was very hot, it was 104° F. Yesterday was hot, it was 103.5° F".

The solution appears therefore to express all value types with some basic notion not involving numbers to express intensity, but also to allow numbers to be used when a unit of measure has been defined for the particular value type. Quantifiable value types are therefore an extension of unquantifiable value types. The classification of values into quantifiable and unquantifiable values will follow that of natural language. The existence of a unit of measurement of the value shows whether it is quantifiable or not. So for instance the concept of niceness which has no unit, and is never associated with numbers, will be classified as unquantifiable (One never says "He is 5 nices nice."). However the concept of weight is associated with numbers and a unit of measurement, hence is quantifiable, as in "He weighs 70 kgs." When a value is quantifiable, so is the difference between two such values.
7.1.3 Values Representation

The values representation is reached through an abstract richness argument in D.1 (p. D-1) which discusses the needs of the value representation and demonstrates that these needs are satisfied by the following representation.

The values representation is based on five events:

- **has_value**: Takes as **subject** the concept associated with a value, and as **object** that value.

- **real_positive**: All elements of a concept connected to this event are real positive numbers.

- **⊕**: Internal addition: \( \sum s_n = o \)

- **⊗**: Internal multiplication: \( \prod s_n = o \)

- **⊙**: External multiplication: \( \prod s_n = o \)

where \( s_n \) are all the partial **subject**s of the event, and \( o \) is its **object**.

Internal operations only take arguments corresponding to the same phenomena, and if there are units, expressed in the same unit. External operations allow different types of values and/or units to be mixed: at least one of the event's **subject**s must be of a different type to the others.

The results of the addition and multiplication events are not defined in the network, as it is more efficient to have an appropriate "reasoning" engine which calculates the results of additions and multiplications.

7.1.4 Use

This section provides a brief overview of the capabilities of the values representation to express a wide range of phenomena, in a manner useful to N.L. processing. D.1 (p. D-1) includes full examples and examines many cases to show the representation to be very rich for N.L. processing.
7.1.4.1 Quantifiable Values

Units can be defined for quantifiable values using:

- \( \forall v \ v + 0 = v \)
- \( \forall v \ v + \infty = \infty \)
- \( \forall v \ v \times 1 = v \)

In this manner, the values 0, 1 and \( \infty \) can be defined. From then onwards, all numbers can be defined (\( 2 = 1 + 1, 3 = 1 + 2 \ldots 21 = 2 \times 10 + 1 \ldots \)). For instance, \( E_0 \) defines 0:

\[
(E_0, R): \{ \Delta F\text{-subject\,-}FO: \mathcal{V}; \ \Delta \forall\text{-subject\,-}I \Delta 0; \ \Delta \forall\text{-action\,-}IO: \Theta; \\
\Delta F\text{-object\,-}FO: \mathcal{V} \}
\]

where \( \mathcal{V} \) is the set of all values, defined as a restriction of \texttt{typeless} by \( \Theta \)'s template event.

Thus, internal addition and multiplication are sufficient to define any number, thus making the class of quantifiable values those which have both internal addition and multiplication. Note that although it may be possible to express numerically the value of quantifiable value, that does not mean one has to: this is particularly useful for N.L., where values may be left unstated: "John is heavier than Mary".

7.1.4.2 Unquantifiable Values

Without the unit value (1), values are unquantifiable. It is therefore sufficient to remove any means of defining the unit value to achieve unquantifiable values. Only internal multiplication allows the unit value to be defined, so all values for which there is no internal multiplication are unquantified.

Unquantifiable values are assumed without loss of generality to be bijectively mapped onto \( \mathbb{R}^+ \), as this simplifies the expression of total order. As internal addition is still allowed, the values 0 and \( \infty \) can be defined for unquantifiable values. However this is not a problem, as 0 and \( \infty \) then represent the boundaries of the unquantifiable range: the absolute maximum and minimum of the range.
7.1.4.3 Order

Order can be represented using $\oplus$, by the equivalence: $a < b \equiv \exists \delta \in \mathcal{V}. a + \delta = b$ as long as $\mathcal{V}$ includes only elements greater than zero. Similarly, the equivalence $a \leq b \equiv \exists \delta' \in \mathcal{V}'. a + \delta' = b$ defines $\leq$ if $\mathcal{V}'$ includes only positive elements. The introduction of order also allows intervals to be expressed: $x \in [a, b] \equiv a \leq x \leq b$.

The set of real numbers can be obtained by restricting typeless by real.positive. The set of real numbers, less zero, can be achieved by partitioning the set of real numbers into the set of null numbers and the set of non-null numbers.

For unquantifiable values, a partial order can be established by connecting values by these events. For quantifiable values, a total order can be established for values which either have a known numerical value, or which are expressed as the sum of a value with no known numerical value and another with a known numerical value: "John is two pounds heavier than Paul, and Mary is one pound less". The total order Mary.Weight < Paul.Weight < John.Weight can be inferred.

The representation of order is the same for quantifiable values – for which a unit was defined – and for unquantifiable values – for which no value can be defined.

7.1.4.4 Value Types

The different types of values do not get mixed up, as they are defined by different specializations of has.value. Each range of values is defined by the definitional subject_ of the relevant specialization’s template event (eg has.weight). It is declared to be an observed antonym of all other values subsets, with respect to the set of all values. Thus weights cannot be confused with distances, as attempts to build such events be rejected by the type-checker.

7.1.4.5 External Multiplication and Units

The external multiplication operator is used to express the value of some continuous phenomena for which no natural unit exists. Counting has a natural unit:
the thing being counted. However distance has no such natural unit, leading to
different units: metres, yards... Having defined a unit, say a metre, a distance can
be expressed as a product of the unit metre times the distance measured in me-
tres. Metre itself is a value defined by an appropriate additional definitional event.
In essence, values for which no natural units exist are modeled as unquantifiable
values (no internal multiplication) for which a numerical value can be expressed
externally, using external multiplication. For instance,

\[(E,R): \{ \Delta I\text{-subject-}IO: [1.\text{metre},5]; \Delta I\text{-action-}IO: \odot; \Delta I\text{-object-}IO: v \} \]

states that \(v\) is 5 metres. \(v\) can also have its numerical value determined in terms
of other units such as yards.

This means there is no predetermined unit to which all values are converted but
any unit may be used naturally – an essential feature for a general purpose N.L.
system: a text may include a unit unknown to the readers. Another important issue
is values may be used vaguely in N.L.: Just as "The house is 100 yards to your left"
does not mean "The house is 91.44 metres to your left", so does "Paul weighs 85
Kg" not mean "Paul weighs 85000 g". The values representation represents these
values differently, thus preserving the implied precision, which can be interpreted
by a specific interpretation engine.

It is also used in sentences involving numerical quantities and unquantifiable values:
"John is twice as happy as Mary". Here John's happiness value is Mary's times
two.

Finally, it can be used for phenomena which can be quantified, but which have no
natural zero-point, such as temperatures: \(0^\circ C \neq 0^\circ F\). This avoids sentences such
as "It is twice as hot as yesterday" to be literally interpreted: yesterday was \(10^\circ C\)
so today is \(20^\circ C\)!

7.1.4.6 Uses

The values representation underlies the representation of set cardinality, belief
value, and certainty value. In other words, the operators \texttt{size}, \texttt{belief.value}
and \texttt{certainty.value} are specializations of \texttt{has.value}, and enforce their own se-
mantics using type-checking.

Other ranges of values can be defined using the 5 value events: negative numbers (using \( \forall p \in \mathbb{R}^+ \exists m \in \mathbb{R}^- . p + m = 0 \)), natural numbers \( \mathbb{N} \) - used as the argument of \text{size}_- \), for instance, and complex numbers (\( i \ast i = -1 \)).

In natural language, values can model many phenomena such as sensations. They prove particularly flexible in terms of N.L., modeling with ease extensional adjectives ("A \underline{red} ship"), relational adjectives ("a \underline{big} ship")\(^2\), subjective adjectives ("A \underline{beautiful} ship"), comparatives, superlatives (including such forms as "The third \underline{oldest} ship", and adverbs of intensity "A \underline{very} big ship".

For instance, "John is older than Mark" is represented as:

\((E_0,R)\): \{ \Delta I\text{-subject}_-IO: John; \Delta I\text{-action}_-IO: has\_age; \Delta I\text{-object}_-I\Delta: x \} \)

\((E_1,R)\): \{ \Delta I\text{-subject}_-IO: Mark; \Delta I\text{-action}_-IO: has\_age; \Delta I\text{-object}_-I\Delta: y \} \)

\((E_2,R)\): \{ \Delta I\text{-subject}_-IO: y; \Delta I\text{-subject}_-_A\Delta: \mathbb{R}_0^+; \Delta I\text{-action}_-IO: \Theta; \Delta I\text{-object}_-IO: x \} \)

The arbitrary quantified \_subject connected to \( \mathbb{R}_0^+ \) corresponds to the positive \( \delta \) in \( a < b \equiv \exists \delta \in \mathcal{V} . a + \delta = b \)

Along similar lines, one obtains "John is tall"

\((E_0,R)\): \{ \Delta I\text{-subject}_-IO: John; \Delta I\text{-action}_-IO: has\_height; \Delta I\text{-object}_-I\Delta: h \} \)

\((E_1,R)\): \{ \Delta F\text{-subject}_-FO: Men; \Delta V\text{-action}_-IO: has\_height; \Delta F\text{-object}_-F\Delta: \mathcal{H} \} \)

\((E_2,R)\): \{ \Delta I\text{-subject}_-IO: Men; \Delta I\text{-action}_-IO: inst.; \Delta I\text{-object}_-I\Delta: John \} \)

\((E_3,R)\): \{ \Delta I\text{-subject}_-IO: Men; \Delta I\text{-action}_-IO: size.; \Delta I\text{-object}_-I\Delta: s \} \)

\((E_4,R)\): \{ \Delta I\text{-subject}_-IO: x; \Delta I\text{-subject}_-_A\Delta: \mathbb{R}_0^+; \Delta I\text{-action}_-IO: \Theta; \Delta I\text{-object}_-IO: h \} \)

\((E_5,R)\): \{ \Delta I\text{-subject}_-I\Delta: x; \Delta I\text{-subject}_-_IO: s; \Delta I\text{-action}_-IO: \Theta; \Delta I\text{-object}_-IO: y \} \)

\((E_6,R)\): \{ \Delta I\text{-subject}_-\forall O: \mathcal{H}; \Delta I\text{-action}_-IO: \Theta; \Delta I\text{-object}_-I\Delta: y \} \)

which states that John is taller than the average of every person's height. Note that the subset of tall people can be built, so that if John and Mary both happen

\(^2\)Using averages expressed in terms of \( \Theta, \otimes, \odot \)
to be tall, these events need not be repeated for each. The cost of building a new
tall person is thereby limited to one instance event.

7.1.5 Conclusion

The values representation is rich enough to express a wide variety of problems in
a cohesive way, provides a natural mapping to natural language, and yet allows
standard reasoning to be performed on it (8.3.3.2 (p. 347)).

7.2 Static Events

Some events refer to relations without for that matter requiring that the relations
be believed. An example of such an event is "to simulate": "to simulate" an
explosion does not mean that the explosion is believed to have occurred, but that
somehow the process the simulation event is describing is related to the idea of the
occurrence of an explosion. The set of such events involving the idea a concept
expresses, without requiring that the concept it expresses is believed is the set of
static events.

Various kinds of static events are of particular interest. The first kind are belief
events, which allow agents to believe statements other agents do not. A second
kind involves diverse forms of communication, such as saying something another
may not believe, but can nevertheless understand. Other forms, such as simulation,
exist. Since they are more of interest to particular applications, rather than to a
general purpose tool such as LOLITA they are of lesser interest to this thesis.

Some static events change the behaviour of the program with respect to their
statements in standard ways. Indeed, although the meaning of their statements is
changed in that the program’s behaviour with respect to them is changed, the stan-
dard treatments need not even consider the statements themselves. For instance,
belief and source events will determine the program’s behaviour with respect to
statements independently of their meaning.
7.2.1 Belief

In A.2.3.1 (p. A-12), the notion that statements are not either true or false, but believed or not by some agent was discussed. Since this idea can influence reasoning, it must be captured by the representation.

First, the discussion will focus on two aspects A.2.3.1 (p. A-12) brought to light. The first was that every statement must be believed by an agent to be of use to practical reasoning. The second introduced an exception to this rule where some statements are not believed, but statements that depend on them are.

7.2.1.1 The belief event

Since a statement which is not believed cannot be used for any practical reasoning (A.2.3.1 (p. A-12)) all statements must be attached to the agent which believes in them. A belief event is used, taking as subject the agent concerned and as object the event(s) the agent believes in. For instance, "John believes all horses like him":

$$(E_{0},R): \{ \Delta I\text{-subject} - IO: \text{John}; \Delta I\text{-action} - IO: \text{believes}; \Delta I\text{-object} - IO: E_{1} \}$$

$$(E_{1},H): \{ \Delta F\text{-subject} - FO: \text{Horses}; \Delta \forall\text{-action} - IO: \text{like}; \Delta \forall\text{-object} - IO: \text{John} \}$$

7.2.1.2 Belief chains

In some cases, one does not wish an event to be believed itself, but to depend on another event, which itself may be believed. This event might for instance be a belief event, as in "Mary believes John believes all prisoners are criminals". This statement does not imply that Mary herself believes that all prisoners are criminals. A solution is to state that if an event is the target of a belief event, it depends on that belief event: it is not believed independently from that belief event. Thus, if an event is attached to a chain of belief events, where each is the target of another

---

3Or to some other stative event, see 7.2.2 (p. 263)
belief event of the chain, the event depends on the whole belief chain. Each belief event states that its object is believed by its subject. Hence the belief chains are read from the outermost to the innermost. Only statements connected to an outermost belief event will be acted on by an agent. Belief chains require a special form of reasoning since statements do not depend only on one agent, the one who will act on them. Since agents may share beliefs, there may be many belief chains per event.

Chains express what an agent believes another to believe. They can be used by the first to make predictions about what the second will do. This is done by considering what he believes the other to know⁴, stepping so to speak into the other's shoes. This process results in more beliefs about the other agent's belief, which in turn may prompt some action by the agent. For instance, if LOLITA believes Mary believes John believes all prisoners are criminals, and LOLITA believes Mary's brother Peter is in prison, she might assume that Mary believes John believes Peter is in prison, and she might explain to John why Mary is so touchy when he inquires after Peter's health. It should be noticed that the reasoning process about another's belief can be recursively performed at every step through the chain. This allows LOLITA not only to reason about (her beliefs of) Mary's beliefs, but also what she thinks Mary would believe John to believe.

Belief events are not the only event that can form chains. There is a class of chain events which share the property that inner events have some form of dependency on outer ones. Although chain events often form chains, they can form graphs. Thus an event a can depend on b both directly, and via a third event c. This cannot be simplified to a depending only on b, as the following example illustrates: assume a belief event $E_0$ has as object both belief events $E_1$ and $E_3$, and $E_1,E_2,E_3$ have as subject.s John, Mary, Jacob and as object.s $E_2,E_3$ and $E_4$ respectively. Assume further that $E_4$ states "Peter is in prison". The chain states not only that John believes that Mary believes that Jacob believes that Peter is in prison, but also

---

⁴Note that if an agent assumes that another agent knows facts he knows, he is performing plausible invalid reasoning. Such reasoning can be useful, but corresponds to adding new information to the network with new certainty and belief values.
that John believes Jacob believes that Peter is in prison.

7.2.1.3 The belief control

Although this scheme appears to work, there is a problem: the outermost belief event is not believed by anyone. As discussed in A.2.3.1 (p. A-12), since the knowledge base does not model some absolute truth, all events must be believed by some agent either directly, or indirectly. Events that are not believed are meaningless since no agent can act on them. This leads to the uncomfortable conclusion that the whole knowledge base is meaningless.

A solution might appear to be simply to require the outermost belief event to have as subject, LOLITA. Thus, for a chain of belief events to be meaningful, it must have as outermost event a "LOLITA believes" event. The solution fails however on two counts. Firstly it requires the "LOLITA believes" event to have a different semantics to all other events: unlike all others this event is special, in that it is meaningful if not qualified by another belief event. Moreover, it breaks distributedness: the "special" status of the "LOLITA believes" event can result in parts of the knowledge base being unsound with respect to the whole knowledge base. Take for instance statements such as "LOLITA believes John believes she believes the earth is flat". The last part of the statement "she believes the earth is flat" is unsound with respect to the full statement.

The problem is that the outermost event is treated as if it were a statement made in a knowledge base expressing absolute truths about the world. But the knowledge base actually only reflects LOLITA's beliefs about the world. In such a knowledge base, the outermost event is redundant! It was introduced to allow the knowledge base to include statements which LOLITA did not believe, but which formed parts of statements she did. These were distinguished by their dependency on a non-outermost belief event. However, if a means other than an event of stating whether an event is believed by LOLITA herself is found, the "outermost event" problem will disappear. The solution adopted for SemNet is to add to each event a control stating whether LOLITA believes it or not. Since controls are part of the node,
they are not affected by distributedness. Since they are independent from other information, they have their own semantics. The belief control has two values: hypothetical \((H)\) and real \((R)\). Real signifies LOLITA believes the event, whereas hypothetical signifies she holds no belief about it.

\[(E_0, R): \{\Delta I\text{-subject.-IO}: \text{John}; \Delta I\text{-action.-IO}: \text{believes}; \Delta I\text{-object.-IO}: E_1\}\]

\[(E_1, H): \{\Delta I\text{-subject.-IO}: \text{Lolita}; \Delta I\text{-action.-IO}: \text{believe}; \Delta I\text{-object.-IO}: E_2\}\]

\[(E_2, H): \{\Delta I\text{-subject.-IO}: \text{Earth}; \Delta I\text{-action.-IO}: \text{is flat}\}\]

Even if \(E_0\) were cut out of the net, \(E_1\) would not be believed since it would still be hypothetical.

The belief control can be understood as a notion of belief internal to LOLITA: either she believes something or she doesn’t. The belief event refers an external notion of belief, such as that she might believe an external agent to have. Thus the introduction of a belief control does not break uniqueness. However, if LOLITA is the subject of a real belief event, this can be deleted and its’ target rendered real. Keeping it would be like believing that one believes something, rather than just believing it.

7.2.1.4 The belief control and the inheritance hierarchy

Just like all other events, hypothetical events are elements of the inheritance hierarchy: all forms of reasoning involving inheritance should work when applied to them too. Even if they are hypothetical, they still are concepts and as such defined by their place in the hierarchy. However, LOLITA does not believe them. This means she also does not believe that they are descendents of their real ancestors. This means that they are joined by a hypothetical inst. or spec. relation to their real ancestors. The hypothetical inst. or spec. relation must be an object. of a belief event, having as subject. the same agent as the belief event connected to the hypothetical event. Thus, the previous example should be extended:
\[(E_0,R): \{\Delta V\text{-subject-}IO: \text{John}; \Delta V\text{-action-}IO: \text{believes};
\Delta A\text{-object-}IO: E_1; \Delta A\text{-object-}IO: E_3\}\]
\[(E_3,H): \{\Delta I\text{-subject-}IO: E_4; \Delta I\text{-action-}IO: \text{inst.};\}
\Delta I\text{-object-}IO: E_1\}\]
where \(E_4\) is the belief template event, and similarly for \(E_2\).

Since sets may have differing hypothetical elements, they may also have differing hypothetical cardinalities: "John believes there are 53 million French, but Mary believes them to number 54 million."

The inheritance rules must take into account these hypothetical inst. and spec. relations. If they did not, the inherited events would be real indicating that LOLITA believes the inherited events although she does not believe the event they describe. What inheritance should produce is the set of events which an agent would believe, if he believed some event. This is achieved by requiring that all events inherited via a hypothetical inst. or spec. event \(e\), must have hypothetical status, and be connected to the belief chains\(^5\) of which \(e\) is the object.

The inheritance rules must also take into account the inheritance of hypothetical events. It might seem clear that if "John believes all mammals drink their mother's milk", John should also believe that kittens drink cat milk. However, if John does not know that platypusses are mammals, then he will not know that a baby platypus drinks its mother's milk. In other words, LOLITA's concept hierarchy need not be the same as John's, and any inheritance of events believed by others over LOLITA's hierarchy is plausible reasoning.

Rarely, a real event \(e_r\) is a descendent of a hypothetical one \(e_h\). This could happen for instance as a result of semantic integration (see 6.7 (p. 193)). In order for \(e_r\) to inherit its real ancestors properties, without them being mapped down as hypothetical, it must be connected by a real spec. or inst. relation to the parent it would have been connected to had the network contained only real events. But it must also be connected by a hypothetical spec. or inst. relation to \(e_h\), so that it inherits all the properties \(e_h\) adds. This latter requirement allows \(e_r\) to inherit

\(^5\)or a copy of them
all the properties that the agent who believes in \( e_h \) believes \( e_h \) to have.

### 7.2.1.5 Belief and certainty values

As explained in D.1.3.4 (p. D-27), belief and certainty values are subjective values, depending on the agent which assigns them. Thus, their treatment follows the standard subjective value treatment. However, one might think that they should qualify the belief \textit{event} rather than the event being believed, in a manner analogous to the homogeneity \textit{event}.

Connecting belief and certainty values to the belief \textit{event} works well when there is a belief event. However, when LOLITA believes an event, there is none. As a result, the belief and certainty values must be connected to the event she believes. This leads to a problem when one wishes to express simultaneously the certainty an agent has about an event in which he believes, and the certainty LOLITA has of his belief in that event: both would be modelled as belief and certainty values connected to the same belief event. The method would have succeeded had there been a belief event for LOLITA rather than a belief control. However the standard treatment of subjective values is more appropriate anyway since it underlines the fact that the values are indeed subjective.

### 7.2.1.6 Logical connectives

The idea of dependency can be generalised to events other than the belief event itself. This is of particular interest since it allows the introduction of logical connectives. As discussed in A.2.3.1 (p. A-12), it is not the notion of truth that matters in LOLITA’s knowledge base, but the notion of belief. In a sense belief replaces truth, and as a result the standard logical connectives have their usual meanings, if one replaces “truth” by “believed in”.

The logical connectives are and, or, and xor.\(^6\) They take as subject the events they connect, and these subject events need not be attached to belief events or be

---

\(^6\)Although a concept can be defined by a set of hypothetical events which are the subject of a real or xor event, this construction is deprecated as it seems likely to result in NP reasoning
real themselves. Just as for logical connectives, some combinations of connectives cannot be believed. For instance, it is not possible to believe in more than one of the subject.s of a xor_. event and in the xor_. event simultaneously. Similarly, once one knows that an agent believes an alternative a of a xor_. event and the xor_. event itself, one can substitute that he only believes a and disbelieves all of the other alternatives. Unlike classical logical connectives however, the number of subject.s of these events is not limited to two.\footnote{\textit{xor} does not break uniqueness as defining a set by its instances and a given \textit{size} is illegal.}

For example, "\textit{John believes either Jack or Mary ate Fred}" would be expressed as:

\[(E_0, R): \{\Delta I\text{-subject}._{\text{-} IO}: \text{John}; \Delta I\text{-action}._{\text{-} IO}: \text{believes};\]
\[\text{\hspace{1cm}} \Delta I\text{-object}._{\text{-} IO}: E_1\}\]

\[(E_1, H): \{\Delta I\text{-subject}._{\text{-} IO}: E_2; \Delta I\text{-subject}._{\text{-} IO}: E_3;\]
\[\text{\hspace{1cm}} \Delta I\text{-action}._{\text{-} IO}: \text{xor.}\}\]

\[(E_2, R): \{\Delta I\text{-subject}._{\text{-} IO}: \text{Jack}; \Delta I\text{-action}._{\text{-} IO}: \text{eat};\]
\[\text{\hspace{1cm}} \Delta I\text{-object}._{\text{-} IO}: \text{Fred}\}\]

\[(E_3, H): \{\Delta I\text{-subject}._{\text{-} IO}: \text{Mary}; \Delta I\text{-action}._{\text{-} IO}: \text{eat};\]
\[\text{\hspace{1cm}} \Delta I\text{-object}._{\text{-} IO}: \text{Fred}\}\]

The real status of \(E_2\) shows that LOLITA believes Jack is guilty.

\subsection{7.2.1.7 Homogeneity}

In 2.9 (p. 21), it was stated that it would be required of the representation that it be able to represent full restrictive properties formed of partial properties and their homogeneity and fracture values as described in [Gariglano 89]. This is achieved analogously to logical connectives. Instead of a set of unbelieved or hypothetical events being the subject.s of a believed or real logical connecting event, they are the subject.s of a real or believed homogeneity_. event.

The homogeneity_. event is always qualified by a has\_homogeneity\_value and a has\_fracture\_value event: if one of these events is not explicitly attached to a homogeneity_. event, it will be inherited down from the template event, identically
to the way subjective value events must always be associated with the agent that
made the value judgement: see D.1.3.2 (p. D-20). Both has_fracture_value and
has_homogeneity_value events are specialisations of the has_value event. Since
the fracture and homogeneity values can be represented as nodes, an order can
be built to state that the use of the term "game" is more fuzzy and the term
"execution" is more precise. Thus "game" is viewed somewhat like syndromes in
Biology: it need need not have every one of the features/symptoms of the syndrome,
only some suffice.

No research has yet been conducted into subsumption algorithms for concepts
defined by partial properties using the homogeneity representation. This means
semantic integration (6.7 (p. 193)) is not yet possible for these concepts.

7.2.2 Communication

Belief is only one of a set of stative events, including "to say", "to write", "to
think⁸", "to pretend", "to dream" and so on. All these events are forms of com-
mutation and share the particularity that although one may not believe the
statement they express, one can believe the statement has been made. Indeed the
definition of "to pretend" includes the notion that what it states is not to be be-
lieved. Thus although the statement itself may not be believed, it should not be
purged from the net if it is the object of such an event.

For an agent like LOLITA, with as only sense textual input, the notion of infor-
mation’s source is very important: she has no means to verify independently what
she has been told, but must rely on others. By maintaining a model of each of her
sources, she can check how reliable a source has been previously on various topics,
and how much she should trust him. ([Bokma et al. 92]) Many events can express
source: "to say", "to tell", "to write"... so they are all instances of a general notion
of source, which can be reasoned with appropriately. The date the source stated
the information is of importance, since a source’s reliability may vary with time.

⁸in the sense to hear in one’s head, rather than to believe
This is expressed as by the standard time representation: a time event attached to
the source event — see 7.5 (p. 271).

Just as events can be qualified by belief chains, so they can be by chains of
source events\(^9\). Since each source event states that its object\(\_\) was reported by its
subject\(\_\), it will be the subject\(\_\) of the outermost event that is LOLITA’s source.
The general treatment of source events means that statements such as “Jan wrote
that John told her that he was the first to report the missing jewels” are processed
as so many levels of source.

7.2.3 Causal events

There are two kinds of causal events: necessary\(_{\text{cause}}\) and sufficient\(_{\text{cause}}\).
They correspond to the scientific notion of causality discussed in [Long et al. 93]:
if \(a\) causes \(b\), then \(b\) will occur after \(a\).

Cause should not be confused with material implication, in that material implica-
tion is derived logically from the definition or knowledge about the concepts in
the system: it is intensional. Thus material implication is time independent: as
soon as \(a\) is true, then \(b\) is true necessarily if \(a \Rightarrow b\). Cause is not implicit in
the concepts, but is something stated about the “external world”: it is something
time-dependent and extensional.

The subject\(\_\) of a necessary\(_{\text{cause}}\) event must be present if its object\(\_\) event is
to occur: it does not state that the object\(\_\) event will occur, but that it can occur,
if the subject\(\_\) is satisfied. The subject\(_s\) of a sufficient\(_{\text{cause}}\) event will cause
the object\(\_\) event to occur assuming the conditions of the necessary\(_{\text{cause}}\) events
are satisfied. For instance, it is a necessary cause that the gun be loaded for it to
be fired, but a sufficient cause for its trigger to be pulled: without the necessary
cause, the gun will not fire.

Template events may be associated with preconditions — using necessary\(_{\text{cause}}\),
and postconditions — using sufficient\(_{\text{cause}}\). It is not rare that an event changes

\(^9\)which are members of the class of chain events
some state of the world by changing the arguments of a relation type (such as owning) that is its precondition and its postcondition. For instance, selling has as precondition and as postcondition different people owning something:

\[(E_0, R): \{O\forall\exists\text{-subject}_-\forall\exists\forall\Delta: \text{sellers}; \Delta\forall\forall\cdot\text{-action}_-\cdot\text{-IO}: \text{sell};\]
\[O\forall\forall\cdot\text{-object}_-\forall\forall\Delta: \text{sellees}\}\]

\[(E_1, R): \{\Delta\forall\forall\cdot\text{-subject}_-\forall\forall\forall: E_2; \Delta\forall\forall\cdot\text{-action}_-\cdot\text{-IO}: \text{necessary}\_\text{cause};\]
\[\Delta\forall\forall\cdot\text{-object}_-\forall\forall\forall: E_0\}\]

\[(E_2, R): \{\Delta\forall\exists\text{-subject}_-\forall\exists\forall\forall\forall: \text{sellers}; \Delta\forall\forall\cdot\text{-action}_-\cdot\text{-IO}: \text{own};\]
\[\Delta\forall\forall\cdot\text{-object}_-\forall\forall\forall: \text{sellees}\}\]

Causal events may take hypothetical events as subject_ and/or object_, and yet be real themselves: "If England has won the test series I'll eat my hat!" In this case the causal event is real, but both the winning and the eating are hypothetical.

Causality in LOLITA is also discussed in [Poria 97], which gives common-sense (i.e. plausible) reasoning methods by causality.

7.2.4 Conclusion

A few stative events have been discussed. In particular, the representation’s sufficient richness for representing belief, logical connectives, and source have been demonstrated, while not breaking distributedness. The use of homogeneity theory copes with concepts difficult to represent. Communication in general and causality were also discussed. However this survey is far from complete. For instance "to know" has not been considered. It resembles "to believe", but as the example "The field Jane crossed this morning had a lion in it. I told her before she crossed it, but she didn’t believe me" shows, it differs a little: one cannot say that Jane did not know about the lion, simply that she did not believe in it.

7.3 Negation

LOLITA recognises two forms of “negation”. The first is conversion to antonym and the second is absence of occurrence. Each will be considered in turn.
Conversion to antonym corresponds to cases where we wish to replace a relation with a relation that precludes it. For instance, say we have the relation “Francis likes Durham”. Conversion to antonym would result in “Francis dislikes Durham”. Note that an active relation “liking” gets replaced by an equally active relation “disliking”. For convenience, we will write the antonym of a relation \( r \) as \( \neg r \).

In contrast, absence of occurrence corresponds to cases where a relation which previously held is said not to have occurred anymore. An example is the conversion of “I hit him” into “I did not hit him”. The latter does not postulate the existence of a relation of “not-hitting”, but the absence of the “hitting” relation. For convenience, if a relation \( r \) did not occur, it will be written as \( \sim r \).

Negation in natural language may correspond to either case. For instance sentences such as “I do not love him” do not imply “I hate him”. Other sentences such as “I do not trust him” imply “I distrust him”. Unfortunately this tends to confuse the issue.

### 7.3.1 The Antonym form of negation

The representation for relations and their converted to antonym forms is quite obvious: The action of the event \( a \) is replaced by that corresponding to the relation’s antonym \( \neg a \). So the events “John likes Mary” and “John dislikes Mary” only differ in that their actions are “like” and “dislike” respectively. An antonym event joins the template event of a relation to the template event of its antonym. (see 5.4.3.2 (p. 151)). Thus there is an antonym event which has as subjects the template events of “like” and “dislike”.

Clearly, not every relation has an antonym. For instance, \textsc{spec.} and \textsc{inst.} lack an antonym form.

For instance “Sengan likes a fish”,

Figure 7.1: There are no men who can read and write

\((E_0,R)\): \{ΔI-subject-IO: Sengan; ΔI-action-IO: like; ΔI-object-IO: fish\_1\}

\((E_1,R)\): \{ΔI-subject-IO: fish\_0; ΔI-action-IO: inst; ΔI-object-IO: fish\_1\}

has as negation by antonym "Sengan dislikes all fish":

\((-E_0,R)\): \{ΔV-subject-IO: Sengan; ΔV-action-IO: dislike; ΔF-object-FO: fish\_0\}

7.3.2 Absence of occurrence

The representation of an event which did not occur requires some thought. It follows from the representation of absence of entities. For instance, "There are no men who read and write" is represented in figure 7.1 (p. 267). Here, the relevant men are selected by the two definitional events \(E_1\) and \(E_2\). Their non-existence is expressed by an observational size zero event \(E_3\): if there is no object fitting all of the restrictions, then the intersection between the sets is empty.

Like entities, events also can be restricted by other events. For instance, the sentence "I ate sandwiches in Paris between the 12th and 15th of April" corresponds to the "eating" event being restricted by temporal and locative events. Consider the statement "I did not eat any sandwich in Paris between the 12th and 15th of April". Suppose that I ate a sandwich in Paris on the 1st of December, and that I ate a sandwich in London on the 13th of April, then the sets of "eating" events
Figure 7.2: I did not eat any sandwich in Paris between the 12th and 15th of April restricted by the temporal and locative events are not empty, but their intersection is. Thus the the number of observed instantiations of the "eating" event is zero.

It might be argued that empty sets of events is unintuitive in some respect. However this idea has been implicit from our definition of events and the non-existence of entities. To see this, notice that in men reading and writing example, the subject arcs of $E_1$ and $E_2$ have $F - F$ quantification. This indicates they are bijections. Hence the number of instantiations of the "reading" and "writing" events is the same as the number of men: zero. This corresponds to the intuition that if there are no men to perform the events, then the events could not have occurred! The zero size treatment therefore follows from expressing events as sets of their instantiations. The natural representation within this framework of absence of occurrence is the same as that of non-existence of entities: it uses a size zero event.

D.2.1.1 (p. D-28) demonstrates the naturalness of negation by absence of occurrence, and shows that concepts such as "The people John never met" can be defined without breaking uniqueness.

7.3.3 Negation: Conclusion

SemNet has two forms of negation, allowing it to account not only for antonym events, but also for events that did not occur. This distinction provides a clear difference between notions often confused in other representations. Because nega-
tion introduces no new operators, uniqueness is maintained. Distributedness is also maintained, trivially in the case of conversion to antonyms, but also in the case of absence of occurrence: that a set can be defined by no means states that it has any instantiations. Only a non size zero event can make any claim of the existence of instantiations. D.2 (p. D-28) introduces some problems involved in negating actual events in the network, making clear that the reasoning involved is non-trivial.

7.4 Parts

inst. and spec. cannot express the parts of an object. Indeed, inst. and spec. do not distinguish between intrinsic and extrinsic properties, resulting in all properties being inherited. Only intrinsic properties should be inherited over the parts relation. Another difference is that a concept can be divided in many different ways. For instance, Europe can be divided up into countries or into mountains and plains. Set relations cannot express this naturally. These points are discussed in more detail in D.3.1 (p. D-41).

7.4.1 Parts Representation

All the parts of a concept can be expressed as partial object.s of a has_part event, which has as subject. the concept:

\[(E_0, R): \{\Delta I\text{-subject.}-\text{IO}: \text{Europe}; \Delta I\text{-action.-IO}: \text{has\_part}; \Delta I\text{-object.}-\forall \text{O}: \text{European\_Countries}\}\]

\[(E_1, R): \{\Delta I\text{-subject.-IO}: \text{Europe}; \Delta I\text{-action.-IO}: \text{has\_part}; \Delta I\text{-object.}-\forall \text{O}: \text{Geographic\_Features}\}\]

Notice that the use of sets avoids all parts having to be enumerated explicitly, allowing LOLITA to cope with partial knowledge.

If the has_part event has only one object., then the object. is only one of the parts of the subject.'s concept – indeed if the partial arc interpretation were maintained, the object. is all the parts of the concept, i.e. the concept itself, which breaks uniqueness. This causes no decrease of distributedness since the granularity
of distributedness is full arcs. The decrease in non-linearity and the break of compositionality is avoided by having two different has.part actions (has.part₁ and has.part₂), although, for simplicity, the distinction is not maintained throughout this thesis:

\[(E₂, R): \{\Delta I\text{-subject-Io}: \text{France}; \Delta I\text{-action-Io}: \text{has.part}; \Delta I\text{-object-Io}: \text{Roussillon}\}\]

### 7.4.2 Intrinsic properties and Substances

Some concepts have the property that some of their properties are inherited by their parts. Such properties are called intrinsic, and the concepts are called substance-like. Metal is an example: its intrinsic properties such as heat conduction, electrical conductivity do not vary from the object to its parts.

\[(E₀, R): \{\Delta FV\text{-subject-Fo}: \text{metals}; \Delta VV\text{-action-Io}: \text{has.part}; \Delta [VV]\text{-object-}\exists I\Delta: \text{metals'}\}\]

\[(E₁, R): \{\Delta I\text{-subject-Io}: \text{metals}; \Delta I\text{-action-Io}: \text{synonym}; \Delta I\text{-subject-Io}: \text{metals'}\}\]

\[(E₂, R): \{\Delta F\text{-subject-F}\Delta: \text{metals}; \Delta V\text{-action-Io}: \text{is.metal}\}\]

where is.metal corresponds to the definitional properties of metals.

This states that any piece of a piece of metal is also a piece of metal: it is metallic. To illustrate the difference, pieces of a table are not also tables.

Not only entities can display substance-like behaviour. Parts of some events retain the full event’s properties: parts of breathing events are also breathing events: if John was breathing for a whole hour, he was also breathing during any minute of that hour.

There is limit to how much some concepts can be divided before the intrinsic properties are lost. For instance, if a match-stick is divided into two, the result is two pieces of wood. However, if a wood cell is divided into two, the result is not two pieces of wood. This granularity can be specified either by the object’s size, or for events by the actual event’s process. The granularity of walking is limited to taking a single step. The time taken for this depends on the animal considered: a cheetah and a sloth walk at different speeds!
7.4.3 Parts of events

The parts of events can be stated with the \texttt{has.part} event, which allows the streams of actions corresponding to the event to be stated. For instance, eating is part cutting food, part putting it in the mouth, part chewing it, and part swallowing it. The various subpart events have preconditions and postconditions which allow them to be ordered in time: some have postconditions which satisfy the preconditions of others. The resulting causal model explains why small pieces of food do not need to be further cut before being put into the mouth, or why yogurt and soup do not need to be chewed.

Parts of events are also used to distinguish between the two events "John only slept three hours last night" and "I walked in when John was sleeping" where one discusses the full extent of time between when John fell asleep and woke up, and the other only discusses his state at some point in time. These "aspectual" differences occur in many natural language sentences and thus must be represented.

7.4.4 Conclusion

SemNet can model parts, intrinsic and extrinsic properties, and complex actions such as eating or driving by using the \texttt{has.part} event. These issues are discussed in more detail in D.3.2 (p. D-44), which also introduces the notion of event density, and how it can be modelled with homogeneity theory. In D.3.2 (p. D-44), \texttt{has.part} is shown not to break uniqueness, and its richness and naturalness are further demonstrated. It is also shown not to break distributedness or non-linearity.

7.5 Time

The representation of time is particularly important to an NL system. Indeed, time is present in every sentence as the tense of verbs. Furthermore, all of an agent's interactions with its environment involve time to some extent.
7.5.1 A time paradigm for N.L.

The scientific model of time is of an independent axis, where events occur either at a time point (instantaneous events), or at time points within a range (durations). This model does not prove natural for expressing statements in N.L. systems. Indeed, in the scientific model, all times are represented as dates, and are thus are assumed to be subject to some origin and to a total order – which is rarely known. Durations are expressed as the differences between two dates, and intervals as the time between two dates. This means that the expression of sentences generates vast numbers of variables with constraints on them, such as $t_1 < t_3$. The expression of unknown ranges as pairs of variables doubles the problem. This also requires every verb LOLITA is to encounter to be known, so that the representation of its time can be built either as an instant or as a range. This difference in representation between time ranges and instants is precisely what is not needed: In N.L., statements can be made in a manner indicating the event was instantaneous, although it was not by nature. This form of “perceptual instant” is most naturally mapped to N.L. if the representation does not differ significantly between instants and ranges.

These points lead to a time representation which is natural for N.L.: each time associated with an event is a concept. Such times may be instants or may be intervals: only if the duration of the time is specified does one know. There is no assumed time unit – just as for values – and any time system can be represented. Because each time has a duration, it is possible to order the times: time $a$ occurred after time $b$. But unlike the scientific model, there is no need for an origin or a total order: the representation represents all times as durations, and dates are expressed only in terms of durations. This inverse paradigm to the scientific model’s results in a far more flexible representation, able to represent a time in any time system in a very similar manner. It also results in far fewer variables, constraints, and assumptions about events’ natures, proving far more natural to N.L. expression.

These points are discussed in far more detail in D.4.1 (p. D-50). Other issues are also considered, such as the need for the representation to be able to state beliefs about the times at which events occurred.
7.5.2 The Time representation

7.5.2.1 Times as concepts, represented by nodes

The previous section concluded that time should be represented in a uniform manner: the representation of time should not change significantly when dealing with instants or intervals, with known or unknown times, with precise or imprecise times. These variations of information do not warrant radically different representations. Instead more information should be provided by additional events qualifying the time concepts. In SemNet, this results in concepts being represented as nodes.

7.5.2.2 Associating events with times

Most times are associated with an event. This association must be represented. As was discussed in D.4.1.9 (p. D-57), there may be quantificational and sortal dependencies between times and events. Similarly, it makes sense to discuss the time of an event, although the event may be believed. Thus the relation between the event and the time is a concept, and should be represented by an at_time event. If it were represented by an additional arc from the event, it would be impossible to discuss the certainty or belief an agent has in it.

The time of an event $a$ will be expressed by another event at_time $b$, which only takes one subject_ ($a$) and one object_ ($a$'s time).

In common with all other events, the at_time event allows quantificational dependencies to be expressed. Thus, sentences such as "every day the sun rises" can be represented as:

$(E_0, R): \{\forall \text{subject}_-\cdot \text{IO}: \text{sun}; \forall \text{action}_-\cdot \text{IO}: \text{rise}\}$

$(E_1, R): \{\forall \text{subject}_-\cdot \exists! \text{O}: E_0; \forall \text{action}_-\cdot \text{IO}: \text{at}\_\text{time};$

$\Delta F\cdot \text{object}_-\cdot F\Delta: \text{sunrise}_\text{times}\}$

where sunrise_times is further defined to only occur once a day. Indeed, because the at_time takes the events as subject_ it can quantify over them, and associate each one with its time.

In common with all other events, the at_time event allows its subject_ to be
defined by its object\textsubscript{1}, vice-versa, or both. For instance, sentences such as \textit{"Every time the FTSE drops, my boss screams"} can be represented as:

\((E_0,R)\): \{\forall\text{-subject}\textsubscript{-IO}: \text{FTSE}; \forall\text{-action}\textsubscript{-IO}: \text{drop}\}\)

\((E_1,R)\): \{\Delta F\text{-subject}\textsubscript{-FO}: E_0; \forall\text{-action}\textsubscript{-IO}: \text{at.time}; \Delta F\text{-object}\textsubscript{-FA}: \text{times}\}\)

\((E_2,R)\): \{\Delta F\text{-subject}\textsubscript{-FA}: E_3; \forall\text{-action}\textsubscript{-IO}: \text{at.time}; \Delta F\text{-object}\textsubscript{-FO}: \text{times}\}\)

\((E_3,R)\): \{\forall\text{-subject}\textsubscript{-IO}: \text{boss}\textsubscript{1}; \forall\text{-action}\textsubscript{-IO}: \text{scream}\}\)

where \text{boss}\textsubscript{1} is defined to be my boss. \(E_0\) defines the set of times \text{times}, and \text{times} defines \(E_3\).

In common with all other events, the \text{at.time} event can be qualified by the standard representations of belief, source, certainty, or absence of occurrence. Thus, \textit{"The crime did not happen at 5 o'clock"} is represented as:

\((E_0,R)\): \{\forall\text{-subject}\textsubscript{-IO}: \text{crime}; \forall\text{-action}\textsubscript{-IO}: \text{at.time}; \forall\text{-object}\textsubscript{-IO}: \text{time}_{5\text{o'clock}}\}\)

\((E_1,R)\): \{\Delta I\text{-subject}\textsubscript{-IO}: E_0; \Delta I\text{-action}\textsubscript{-IO}: \text{size}; \Delta I\text{-object}\textsubscript{-IO}: 0\}\)

### 7.5.2.3 Instants and Intervals

In section D.4.1.5 (p. D-54), it was argued that intervals and instants should not differ substantially in the representation. Indeed, in N.L. intervals could be expressed as if they were instants, so many of the algorithms in LOLITA would treat them the same way, unless forced not to by differing representations.

This means that a single time node may represent either an interval or an instant. However, one still wants to retain the ability to state that a time is an interval or an instant if it is thought to be one or the other. Two ways of representing this spring to mind:

- Associating the endpoints of an interval with two values. These would be stated to be equal for instants. For intervals of known durations, the addition of the duration to the value of the beginning of the interval gives the value
of its end.

- Associating the intervals with durations. An instant has nil duration, whereas an interval has a non-nil duration.

The first solution is a direct mapping of the scientific model. One disadvantage is that it assumes a unique origin of time. This contradicts the requirement of 7.5.3.2 (p. 285) that the representation of time should not impose one calendar system into which all others must be converted. Another disadvantage is that to represent an interval, one is forced to represent the start and end points of the interval, even if one does not know what they are. Indeed, one even has to do this for instants, where the start and end points are represented by the same node.

The second solution separates out as much as possible the notions of date and duration. This is important as NL sentences often express only one or other of these quantities. For instance, "John saw the film on Sunday" makes no mention of the duration of this activity. Conversely, "Leave to simmer for 5 minutes" makes no reference of date. However, if the times an interval started and ended are known, this can clearly be represented by the combination of the date of the event's start and its duration. Moreover, this solution does not impose any calendrical system, since durations can be expressed in any unit, using SemNet's standard values system.

Clearly the second solution is preferable for LOLITA. It can simply be represented by a has_duration event, which is a specialization of the standard has_value event. The has_duration takes as subject a time node, and as object a value expressing a duration. The durations are typed by the has_duration event, and may only be obtained by external multiplication of a temporal unit by a positive real value.

As was discussed previously, there is a zero value, and a set of non-zero values which simplify the representation of many properties, such as absence of occurrence. This same set structure can be used to simplify statements that a time node is an instant or an interval. For instance:
(E₀,R): {ΔF-subject-FΔ: instants; ΔV-action-IO: has_duration;
ΔV-object-IO: zero-duration}
where instants is a subset of the set of all times. Any time node can be stated to
be an instant simply by making it an instance of instants.

Further, because has_duration is a specialization of has_value all the expressions
allowed by the standard values system may also be used for durations: "Cook the
salmon for two to three minutes", or "Less than two hours ago, president Yeltsin an-
nounced his candidature for the forthcoming Russian presidential elections". Similarly,
the standard values system copes with many units, and the notion that times
may be used in a fuzzy manner, so five minutes need not be precisely 300 seconds.

A point further discussed in D.4.1.8 (p. D-56), is the issue of whether or not
intervals include their endpoints. This problem has been of much interest in the
literature. The position adopted in SemNet is conservative: they do not, unless
explicitly stated. In practice, the question is usually artificial in that people tend
not to talk precisely about time.

7.5.2.4 Representing chronology

• The need for events to relate times

Not only does one want to represent the duration of times, one also wants to
relate them, for instance to say that one precedes another. In the scientific model,
each time is assigned a numerical value, which is implicitly ordered. However this
assumes a unique origin of time¹⁰. It contradicts the requirement of 7.5.3.2 (p.
285) that the representation of time should not impose one calendar system into
which all others must be converted. Another solution is required.

If no unique origin can be assumed, each date must be expressed with respect to
one of many times serving as origin. Thus all dates in the Judeo-Christian calendar
would be expressed with respect to the supposed birth date of Christ, whereas the
dates of the Jewish and Muslim calendars would be expressed in terms of different

¹⁰Or that the origin is specified in terms of another system of time
origins. A date is thus the duration of the interval between the date's origin and the time being dated.

What is therefore needed is a means of relating two times: the time being dated \( dt \) must be related to the time \( it \) representing the interval between the origin time \( ot \) and \( dt \), and \( it \) must be related to \( ot \). The relations all concern the start and end of the intervals of the time \( ot \), \( it \) and \( dt \). \( ot \) is the instant \( it \) starts, and \( dt \) starts at the instant \( it \) ends. Thus what is needed is a relation between times that states that one starts the other, and another that states one ends the other.

- **The starts.. and ends.. events**

The starts.. event states that the start points of the times it takes as subject.s coincide. If any of these times is an instant, then it is taken as its start point.

The ends.. event states that the end points of the times it takes as subject.s coincide. If any of these times is an instant, then it is taken as its end point.

- **A break of Uniqueness!**

starts.. and ends.. cause a break of uniqueness: if all the times the starts.. event takes as subject.s are instants, the starts.. event becomes equivalent to synonym.. for times. Similarly, if all of the times the ends.. event takes as subject.s are instants, the ends.. event is equivalent to synonym.. for times. Further, the break of uniqueness can also occur for intervals: if both times are both subject.s of starts.. and ends.. events, then they are synonyms.

It might seem that requiring the starts.. and ends.. events to join intervals of different durations would avoid this problem. However, this is not the case, since it is still possible to build a set of events equivalent to a synonym event: \( a, b, c, \) and \( d \) are intervals: \( \text{starts.}(a,b) \land \text{starts.}(b,c) \land \text{ends.}(a,d) \land \text{ends.}(c,d) \) states that \( a \) and \( c \) are synonymous.

The situation is similar to that encountered for addition and internal multiplication of values. Addition of a nil value, or multiplication by one are both equivalent to synonym.., so constitute a break of uniqueness. However, this is intrinsic to the nature of addition and multiplication so cannot be avoided if one wants to represent
them. Thus cases where starts_ and ends_ events combine to state synonymity will have to be detected by reasoning.

7.5.2.5 Composite events

Many general notions, such as one interval being within another, can be expressed with the starts_ and ends_ events. However this leads to complex clusters of events which are hard for many algorithms, such as NL generation, to recognise. It proves therefore convenient to represent the notions by composite events. These have template events stating their equivalence to the appropriate combination of starts_ and ends_ events, allowing standard reasoning.

• is_in

The is_in event states that the time expressed by its subject_ is included in the time expressed by its object_, unless both times are instants, in which case it is equivalent to synonym_. It corresponds to "during" or "while" in NL. More formally, it states that the start time of its object_ follows the start time of its subject_ and that the end time of its object_ precedes the end time of its subject_.

For instance, "During the meeting, John presented Siemens’ projects":

\((E_1,R)\) \:
\{ \Delta I\text{-subject}_-\text{-IO}: \text{John} ; \Delta I\text{-action}_-\text{-IO}: \text{present} ; \\
\quad \Delta I\text{-object}_-\text{-IO}: \text{projects}_1 \} \\

\((E_2,R)\) \:
\{ \Delta I\text{-subject}_-\text{-IΔ}: \text{E}_1 ; \Delta I\text{-action}_-\text{-IO}: \text{at\_time} ; \Delta I\text{-object}_-\text{-IΔ}: \text{t}_1 \} \\

\((E_3,R)\) \:
\{ \Delta I\text{-subject}_-\text{-IΔ}: \text{t}_1 ; \Delta I\text{-action}_-\text{-IO}: \text{is\_in} ; \\
\quad \Delta I\text{-object}_-\text{-IO}: \text{t}_0 \} \\

where \(t_0\) is the interval during which the meeting took place, \(t_0\) and \(t_1\) are both times, projects\(_1\) are Siemens’ projects.

The template of is_in is \(E_0\) in:

\((E_0,R)\) \:
\{ \text{OV\_subject}_-\exists!\Delta: \text{A} ; \text{ΔV\_action}_-\text{-IO}: \text{is\_in} ; \\
\quad \text{OV\_object}_-\exists!\Delta: \text{B} \} \\

\((E_1,R)\) \:
\{ \Delta F\text{-subject}_-\text{-FΔ}: \text{D} ; \Delta F\text{-subject}_-\text{-FO}: \text{A} ; \\
\quad \Delta V\text{-action}_-\text{-IO}: \text{ends} \}
(E₂,R): {ΔF-subject.ₚ-FΔ: C; ΔF-subject.ₚ-FO: A;
    ΔV-action.ₚ-IO: starts.}

(E₃,R): {ΔF-subject.ₚ-FΔ: G; ΔF-subject.ₚ-FO: B;
    ΔV-action.ₚ-IO: ends.}

(E₄,R): {ΔF-subject.ₚ-FΔ: F; ΔF-subject.ₚ-FO: B;
    ΔV-action.ₚ-IO: starts.}

(E₅,R): {ΔF-subject.ₚ-FΔ: F; ΔF-subject.ₚ-FΔ: C;
    ΔV-action.ₚ-IO: ends.}

(E₆,R): {ΔF-subject.ₚ-FΔ: G; ΔF-subject.ₚ-FΔ: D;
    ΔV-action.ₚ-IO: starts.}

where F and G are instants, and A, B, C and D are times.

• after_

In NL, the use of common prepositions such as “after” or “before” is ambiguous. This ambiguity stems from events being associated with intervals, which have start and end points. The issue then, is what stating that one interval is after the other means in terms of time points.

The problem is circumscribed to intervals of durations similar to the time separating the events. Indeed, the question simply does not arise for instants since there is no ambiguity: “John died five minutes after Jack”. Similarly, if the earlier and later events are relatively short with respect to the interval between them, they can be treated as instantaneous: the imprecision of the duration of the interval between events will eclipse any differences due to a reference to the startpoint or endpoint of its interval.

• Examples of the problem

The sentences “Haydn was alive before Mozart was alive”¹¹ and “Mozart was alive after Haydn was alive” are not equivalent. Clearly “after” and “before” are not strict converses. Again “Haydn was alive after Mozart died” is not equivalent to “Mozart died before Haydn was alive”. In these cases, “after” places the later event

¹¹Examples taken from [Galton 87]
after the endpoint of the earlier event. In contrast, "before" places the earlier event before the startpoint of the later event. Hence the second sentence states that Mozart was alive after Haydn died. Similarly the forth sentence states that Haydn was born after Mozart died.

In the sentence "John borrowed a book 5 minutes after Jack did", "borrowed" refers to the start of the borrowing event. Similarly for "John went shopping 5 minutes after Jack did". This differs from the previous analysis of "after".

In the sentence "John revised his course-notes before the examination", the endpoint of the revision occurred before the startpoint of the examination.

In the sentence "John published a book two weeks after Jack did", it is to the endpoints of both publishing events that the "after" refers.

In many cases, people are uncertain, or give different interpretations. In an informal survey of ten people, simple sentences such as "Jack ate an apple 5 minutes after John did" and "John played a game 5 minutes after Jack did", produced a wide variety of interpretations. However many people said that only the context could decide the issue.

○ Solutions

Clearly, "after" can have a variety of meanings, such as:

1. There is an instant in the second event which occurs after the start of the first event.

2. The start of the first event occurred before the start of the second event.

3. There is an instant in the second event which occurs after the end of the first event.

4. The start of the second event is after the end of the first event.

\footnote{One could also argue that there are two different meanings of the verbs in these sentences, one referring to the process and the other to some important instant within it, such as its completion. The ambiguity would then be at the level of the event's action, rather than the "after". This displaces the problem from the temporal model to the data and disambiguation algorithms. Here no assumption is made as to the solution, and the worst case is considered.}
5. The end of the second event is after the end of the first event.

Since "after" is ambiguous, it can be represented using the representation of ambiguity. This requires various meanings of "after" to be defined with template events. By structuring the template events appropriately, it is possible to ensure that at each stage of disambiguation the information common to the meanings of "after" can be used. For instance, all the interpretations have in common the first meaning of "after". As usual with disambiguation, there may be no need to disambiguate the meaning fully (see D.8.6.3 (p. D-138)).

The disambiguation process itself is expected to combine a wide variety of sources of information. Linguistic rules for instance appear to constrain references to states to the third or forth meanings. Reasoning and knowledge of the meaning of events may also be needed to eliminate some meanings. In the Haydn and Mozart examples, the fact that one is only alive once is used to conclude that the third and forth meanings are equivalent. Similarly, knowledge will determine whether the events can be considered as instants with respect to the time separating them. Note that instants can always be specified verbally in a system with little knowledge as in "John started reading the book five minutes after he had bought it".

It proves useful to allow the after. event to take a third argument: the duration between the start of the first time and that of the second time. This allows easy modeling of statements such as "After 5 minutes, my head began to droop". This third argument is represented by a second event qualifying the after.: a duration. event. The representation for "Ten minutes after being elected, John was deposed by the military" is:

\[(E_0,R): \{ \Delta I\text{-subject.-}\forall \Delta: \text{People}_1; \Delta I\text{-action.-}IO: \text{elect} ;
\Delta I\text{-object.-}IO: \text{John} \}\]

\[(E_1,R): \{ \Delta I\text{-subject.-}I\Delta: \text{Military}_1; \Delta I\text{-action.-}IO: \text{depose} ;
\Delta I\text{-object.-}IO: \text{John} \}\]

\[(E_2,R): \{ \Delta I\text{-subject.-}IO: E_0; \Delta I\text{-action.-}IO: \text{at.time};
\Delta I\text{-object.-}I\Delta: t_0 \}\]
(E₃,R): \{ \Delta I\text{-subject} \Delta \Delta: E_{1}; \Delta I\text{-action} \Delta \text{-IO}: \text{at.time}; \\
\Delta I\text{-object} \Delta \text{-IO}: t_{1} \} \\
(E₄,R): \{ \Delta I\text{-subject} \Delta \Delta: t_{1}; \Delta I\text{-action} \Delta \text{-IO}: \text{after} \Delta \text{-}4; \\
\Delta I\text{-object} \Delta \text{-IO}: t_{0} \} \\
(E₅,R): \{ \Delta I\text{-subject} \Delta \Delta: E_{4}; \Delta I\text{-action} \Delta \text{-IO}: \text{duration}; \\
\Delta I\text{-object} \Delta \text{-IO}: 10 \text{minutes} \} \\
Here, the 4th meaning of after\text{.} is used: The start of the second event is after the end of the first event.

Note that the after\text{.} event may take an instant as subject\text{.,} object\text{.,} or both. Moreover, there is no obligation for all after\text{.} events to be connected to a duration\text{.} event: if they are not connected to one explicitly, they will inherit one from after\text{.'s} template.

follows\text{.} is one of "after"'s meanings. It is also used to express continuation in narrative texts, by LOLITA's semantic analysis. Formally, it states that the start point of the time expressed by its subject\text{.} precedes the start point of the time expressed by its object\text{.} It also includes duration and is used in German sentences such as "Heidi aß einen Apfel 5 Minuten später als Claudia" which both refer to the start of the two events. Template events for after\text{.}₄ (the fourth meaning of after above) and follows\text{.} are given in D.4.2 (p. D-58).

7.5.2.6 Times of template events

Events can have a minimal, a maximal, and an average duration. These times can be expressed on the template event, by stating that each element of the template event set is observed to define a time. This time is observed to have a duration within the specified range. For instance, "sleeping" has as minimum duration a minute, and as maximum a day.

The average duration is expressed using the standard average mechanism provided by values, see D.1.3.1 (p. D-16). This average information is essential for planning, but also for temporal reasoning about texts, since it is usually assumed. For instance, the average time for swallowing is a second. Average times may be
expressed in terms of the amount of effort involved. For instance, the average duration can be related to the number of objects the event takes. In this way, the fact that washing up takes about a minute per item can be expressed.

It is also possible to state that the event only has meaning for a range of dates. For instance, the event "typing" could be defined as only being valid for times since the advent of the typewriter. D.4.3.3 (p. D-61) details how template events can restrict the number of occurrences events of a particular type may have.

Most events are associated with a time, either inherited or connected to them. Indeed, a time-less event is not an event which holds at all times, but an event holding at some unknown time. This is necessary to preserve distributedness: if the lack of a time event were to add meaning, an event would be assigned different meaning depending on whether the whole network were read or not. Thus, even such eternal truths as "1 + 2 = 3" must be qualified by a time event with as time an infinite interval – which can be specified on Θ's template event. This requirement is particularly important for events that do not have template events, such as spec. and inst.. Since they cannot inherit a time, these events must themselves be connected to a time event. This expense is reduced using arbitrary quantification, as is done in D.6.4.2 (p. D-86).

7.5.2.7 Times and the Inheritance Hierarchy

The set relations expressing the inheritance hierarchy can be assigned times. Just as for 7.2.1.4 (p. 260), the events inherited over set relations restricted by times gain the time restrictions themselves. Thus one can state that Sam was a fireman in 1989 by joining Sam to Firemen by an inst. event restricted by an at.time event with time 1989. If a concept is defined only by events occurring during a particular range of time, then the concept cannot be used at other times: Sengan's features are defined for all times after 1971. Sengan cannot be qualified by any event with a time before 1971\textsuperscript{13}. These points are discussed further in D.4.3 (p. D-59).

\textsuperscript{13}In the frame of existence in which it is defined for all times after 1971
7.5.3 Expressing Dates

As discussed in 7.5.2.4 (p. 276), if many calendar systems are to be supported, no unique origin for all dates can be assumed. The date of each time must be expressed with respect to the origin of the calendar system in which it is expressed. This means that a date is the duration of the interval between the calendar's origin and the time being dated.

7.5.3.1 Representing a calendar

This means that no special extension of the representation is needed to express dates. Instead the components of the calendar system can be built with the events developed so far. A full example is given in D.4.3.4 (p. D-62). This section focuses on the main ideas.

A calendar is a model of time, which can be expressed in terms of times since a particular point in time, such as the supposed date D of Jesus Christ's Birth in the Gregorian Calendar. Thus a set of times which have as durations centuries, years, months (etc) can be built. These times start centuries, years, months (etc) after D, and serve as startpoint for further times of durations years, months, days (etc). The construction may continue further recursively. Thus, "4th May 1993" is the third day after the beginning of the fifth month after the beginning of the 92cd\textsuperscript{14} year after the beginning of the 19th century\textsuperscript{15} after D. As D.4.3.4 (p. D-62) explains, this model copes with all the irregularities of the calendar system by using fuzzy units: months may be 28, 29, 30 or 31 days long. Days may be 23, 24 or 25 hours long, and so on. The fact that the unit "month" is used to state the date for the term May, rather than, say, the number of days since the beginning of the year, makes the expression far simpler. In effect, "months" are fuzzy units, which vary in length. Further reasoning is required to convert — when possible — a date into number of days since some event, but since this operation is rarely needed, it is useful that the representation does not require it. D.4.3.4 (p. D-62) discusses

\textsuperscript{14}There is no year 0!
\textsuperscript{15}1900 years = 19*100
how side-effects of the use of fuzzy units can be countered, and how the calendar’s representation can be rendered more efficient.

7.5.3.2 Multiple calendars

Just as the Gregorian calendar can be defined in terms of duration with respect to some origin, so can other calendars, such as the Jewish and Arab ones. By allowing different units to be used, notions such as lunar months can be captured.

Fuzzy durations allows the Roman calendar to be represented. The Roman day was divided into ten hours of sunlight as measured by a sundial. Converting this time into our units would require knowing the location (latitude) of the event. It also requires the exact date to be known: the year, month, and day. Requiring conversion would make it virtually impossible to express partially undefined times such as “the fifth hour of some day during the forth lunar month 434”. However it is made possible by explicitly modeling the calendars in SemNet.

When calendars support identical systems of units which can be precisely converted, and if the time between their origins is known, it is possible to determine what the date in one corresponds to in the other. Similarly it is possible to determine which times occurred earlier. However specific reasoning to determine the accuracy of such conversions is required: a date in one calendar may be converted into a range of days in the other.

7.5.3.3 Linguistic analysis

A linguistic analysis of the way date is expressed in NL reveals that times shorter than a day are expressed by referring to the time elapsed since the beginning of the day. Thus seconds, minutes and hours start at zero (no time elapsed). However days, months, years and centuries are counted by position: counting starts at one – there is no 0th day, 0th month, 0th year, or indeed 0th century.

All times are converted into time elapsed since an origin. In the case of the Gregorian Calendar the origin is the instant 00:00:00 on the 1st January 1 AD. Thus
there is no difference between the treatment of usual date information and such common statements as "I'll see you on the forth day of the conference." As NL statements about time in calendars other than the Gregorian often specify the method of referring to time "The forth hour of the day", the correct representation will be built for them: in this case the forth hour is referred to by counting from one, so refers to the hour starting 3 hours after the beginning of the day.

7.5.3.4 Dates: Conclusion

The representation of date uses the standard notions of duration, while allowing efficient access to date if required. Because of this, many different ways of expressing dates are supported, and the standard reasoning methods for times work. However, the representation does not require a complete model of every calendar used, allowing tasks that do not require such information to be performed.

7.5.4 Tense

Many systems\(^\text{16}\) represent tense separately from time, as an independent quantity. This section discusses the nature of tense, and the implications of such a strategy.

7.5.4.1 The nature of tense

Classically, tense has been assigned a dual role: the expression of the chronology of events in sentences, and the aspect of events.

• Ordering of events

The classic treatment of the ordering of events by tense is in [Reichenbach 47], which models tense as a combination of three times: the speech time (when the event was recounted), the reference time (a secondary time used to form complex tenses), and the event time. This accounts for the event "John had drunk a pint"

\(^{16}\text{Such as CGT, CLE-3, and many forms of DRT}\)
occurring before "John fell off the pavement" and before "John is feeling better now".

This representation fails however to account properly for tenses in other languages. For instance, French has a richer system of tenses that English or German, and requires no less than four reference times to capture two of its tenses: "le futur du passé" and "le futur antérieur du passé". Chinese on the other hand does not have morphological variation of its words to express tense. It uses a system three particles following the verb: roughly, one expresses the simple past, one that of an event already having been experienced in the past, and one denoting the present progressive. Auxiliaries are used to denote the future, and all other information about the chronology of events is expressed by prepositions such as "before" and "after". Separate representations of tense and time cannot hope to capture the equivalence between the Chinese use of prepositions and the English use of tense to express, say, the past perfect.

Moreover, modeling tense simply in terms of three times is insufficient. Tense refers to times implicit in surrounding phrases. Assume for a moment that tense were not dependent on the surrounding phrases. If this were true, a summary can be produced by pasting the key sentences of a statement together. For instance, "John was awarded a medal. He had been seen stealing an apple." would be a possible summary of "John fought against the invader. He was awarded a medal. But a few months later he was shunned. He had been seen stealing an apple." The use of a particular tense in a particular phrase therefore depends on the phrases around it, making tense an anaphoric notion, crossing sentence boundaries. Just as all other anaphoric phenomena, the references it makes must be made explicit.

Tense must thus be analysed into the time associated with each event but also into a set of cross-phrase dependencies. These may relate the speech time, reference time, the event time, or any other times used to model tense. Since these times may each be referred to, they should be represented as independent times, and be related with the normal time relations.

- Aspect
Aspect in the literature seems to be a cover-all term, dealing with many of the features of tense other than temporal ordering: perfective aspect, progressive aspect, and classification of verbs. It is discussed in detail in D.4.4 (p. D-66). Here, the main points are summarized.

There is no difference in order between statements in the past and the perfect tenses: they both have already occurred prior to the reference time. However they are not equivalent. The difference lies in the truth of the events’ postconditions: “I have eaten” suggest that the postcondition of eating – namely not being hungry – still holds, whereas this is not so for “I ate”. For instance, the former is a natural answer to “Would you like to join us for lunch”, in distinction to the latter.

Similarly, statements in the progressive refer some interval within an event. For instance, “While John was making breakfast, Mary got dressed”. It does also not state whether the event reached completion. The exact behaviour of verbs with respect to the progressive has been classified – see D.4.4 (p. D-66) for more details.

Aspect varies from language to language, more than the implicit ordering of events in time given by tense. For instance, the present in English has a habitual meaning “I eat an apple” (every day), but does not in French: “Je mange une pomme” (“I am eating an apple”).

Aspect’s multi-faceted nature suggests that it should not be accorded its own specific representation, as does its variance with respect to different languages. Instead it should be analysed into constituent parts. This removes it from the specific area of the representation of time, and rephrases it as a set of global problems, some of which are addressed in 7.4 (p. 269) and 7.2.3 (p. 264).

7.5.4.2 Using a representation of tense

Since the number and meaning of the tenses vary from language to language, tense does not appear to be a basis for conceptual partitioning of the agent’s environment. Instead it appears to be a linguistic unit of communication, whose meaning depends on the language.
This has practical repercussions for systems that use separate representations of time and tense. Two representations require two sets of algorithms for temporal reasoning, leading to trouble maintaining consistency, and decreasing efficiency. Further, a good model of tense and aspect is required for any but the most trivial of tasks. Simply generating in N.L. the summary of a text would require sentences in the summary to be expressed in different tenses from those appearing in the original document. Similarly translation between languages would require remapping of all tenses. Thus if tense is to be analysed into independent concepts, it is best to do so where all algorithms can benefit from it: in SemNet.

7.5.5 Conclusion

The representation of time is cohesive, using the standard sorts, quantification and values representations for much of its expressive power, and by not giving tense its own representation. The same representation expresses dates, as well as durations and instants. It proves natural to N.L. expression, representing such statements as "The fifth day of the conference" and "The fifth of March" similarly. It is also particularly suitable for reasoning 8.3.6.2 (p. 357). It is rich, allowing for instance dates in many different calendar systems to be expressed, without breaking efficiency features such as distributedness and non-linearity. Determinism of search is high, and topological distance is low for most purposes, although it is higher for dates than alternative simpler representations which are less flexible.

7.6 Representations for Language

7.6.1 Linguistic and Conceptual Nodes

As LOLITA is to be a multi-lingual tool, it must be possible to relate words in different languages to her concepts. This is achieved by introducing a new kind of nodes: linguistic nodes.

Linguistic nodes are linked to their corresponding concepts by in_language events.
Different languages have different specialisations of this event: \texttt{in.english}, \texttt{in.french}...
Each linguistic node is uniquely determined by its label – the word in the relevant language – and its controls – which determine its linguistic features in the given language. Thus, for instance "\textit{la lune}" (French) has feminine grammatical gender, but "\textit{der Mond}" (German) has masculine gender, and "\textit{the moon}" has neuter gender. Exceptions can be expressed between linguistic nodes, using such linguistic events as \texttt{has.root}, plural "\textit{men}" for instance between the irregular and the singular "\textit{man}".

The lack of an \texttt{in.language} event for a particular language connected to a concept indicates the lack of a word for the concept, which must then be expressed by a paraphrase. If two different linguistic word nodes in the same language are linked to the same concept, they are synonyms. If more than one \texttt{in.language} event stem from a linguistic word node, the word is ambiguous in that language. By qualifying the different \texttt{in.language} events by \texttt{has.linguistic.frequency} events, the raw frequencies of the different meanings can be expressed – a help to disambiguation.

Because linguistic nodes are uniquely defined by their label and linguistic controls they can define conceptual nodes: For words unknown to LOLITA, such as perhaps \textit{phemenology}, the corresponding concept ("the meaning of \textit{phemenology}") is uniquely defined, and hence can be entered into SemNet without breaking uniqueness. Similarly, one new meaning of a known word can be defined using linguistic nodes.

All these issues are discussed in more detail in D.5 (p. D-69) which demonstrates the richness of the resulting representation.

\section*{7.6.2 Intension and Frames of Existence}

\subsection*{7.6.2.1 Intension}

SemNet is an intensional representation. Thus, the concepts "\textit{Morning Star}" and "\textit{Evening Star}" are different, even if they refer to the same extensional object. In order to express such extensional equality, the \texttt{synonym} event is introduced in
D.6.2 (p. D-76). It states its subject.s refer to the same entity. Sorts affect its modality:

- If both subject.s are defined by the synonym., then they restrict each other: the definitional events of both concepts apply to both concepts.

- If the synonym. is observational with respect to both subject.s, then the two concepts are observed to be the same. This is not due to their definitions, so is a quirk of the world: extensional. Thus each concept inherits all the other's events as observational.

- If the synonym. is observational with respect to one of its subject.s, but definitional with respect to the other, the latter is restricted by the definitional events of the former. The former is observed to be synonymous with the latter as the result of this, and thus inherits the events of the latter as observational.

synonym. also allows events such as antonym., powerset., and infinite.powerset.to be defined. This, and synonym.'s implications for uniqueness are discussed in D.6.2 (p. D-76).

The ability to conceptualise the same object in different ways proves essential to understanding many texts. For instance, flowers can be thought of as beautiful arrangements of petals on top of a green stem, or as the reproductive organs of plants. Choosing the right conceptualisation is important in trying to understand why people give each other flowers, or in giving a biology lecture.

7.6.2.2 Frames of Existence

Frames of existence allow SemNet to represent alternative possible realities. For instance, to express the fact that "Sherlock Holmes lives in London" does not imply that Sherlock Holmes is real. Similarly "I need a hammer" does not indicate that there is any hammer: "I need a Dragon". The ability to represent alternative possible realities is essential for planning which considers different possible futures.
These different problems are addressed by the same representational device: the in_Frame event.

\((E_0,R)\): \{\(\Delta I\)-subject\_\(-I\Delta\): \text{Sherlock.Holmes}; \(\Delta I\)-action\_\(-I\Delta\): live\_\text{in}\;^{17};
\(\Delta I\)-object\_\(-I\Delta\): London\}

\((E_1,R)\): \{\(\Delta A\)-subject\_\(-I\Delta\): \text{Sherlock.Holmes}; \(\Delta A\)-subject\_\(-I\Delta\): \(E_0\);
\(\Delta A\)-action\_\(-I\Delta\): in\_\text{Frame}; \(\Delta A\)-object\_\(-I\Delta\): fictional\}

Each in_Frame event associates all instantiations of a concept within a frame of existence with a frame marker representing that frame. Thus the restriction of typeless by an in_Frame event with a given frame marker corresponds to all instantiations of all concepts within that frame. Such restrictions allow structure to be expressed within the frames using the standard set relations: two frames of existence share no instantiations if they are antonyms. Similarly, the cartoon frame of existence has as specialisations cartoons from Walt Disney and those from Warner Brothers. Thus, frames of existence can be used to further structure information.

For instance, different micro-theories can be declared to apply to different ranges of a given phenomenon: for precise modelling of small phenomena, quantum mechanics is useful; for precise modelling of fast bigger phenomena, general relativity is useful; for less precise modelling about normal phenomena at normal speeds, naive physics may be better. These issues are discussed in detail in D.6.4 (p. D-83).

Adding the in_Frame events not only allows imaginary concepts – from books or films for instance – to be distinguished from "real world" concepts, but it also allows observational statements to be made about intensional concepts themselves: statements that would apply to any instantiation of the term, whether in "our world" or in any other. This is for instance useful for defining events as combinations of other events, such as the definition of powerset\_ in D.6 (p. D-75).

\(^{17}\text{This example is simplified as no representation of location is presented in this thesis.}\)
7.6.3 Textual References

7.6.3.1 The idea

Texts are structured to convey meaning in a clearer manner. In particular texts tend to focus on one idea at a time. This structure can be used in the interpretation phase, only if LOLITA has some idea of the text's structure. Because Natural Language texts often refer to other sections of the text ("See D.7.1.1 (p. D-100) for more details"), it proves convenient to represent the text structure in SemNet. In particular, this allows different interpretations of the same text to be represented, and thus to be reasoned about.

D.7.1.1 (p. D-102) distinguishes two forms of text structure: logical and physical. Logical text structure is independent of the medium conveying the text. It is formed of words, sentences, paragraphs, sections, chapters and so on. Physical text structure depends on the medium of the text: pages and volumes for books, Internet addresses for the world wide web, scrolls for Chinese classics, and so on.

Both types of structure can be modelled using the Textual References representation. This links concepts resulting from semantic analysis, to a conceptualisation of the structure of the text from which they were derived. The representation of the text structure is very similar to the representation of time, exploiting the fact that segments of text map well onto intervals of time, by replacing duration by length. This means only one reasoning engine is needed for both phenomena. The concepts expressing the text structure are called textrefs.

7.6.3.2 The Representation

The representation of text structure mirrors the representation of time very closely. There are therefore textref variants of starts_, ends_, is_in, and follows_: T_Starts, T_ends, T_is_in and T_follows. Similarly, a specialisation of has_value mirrors the effect of has_duration: T_segment_length. Just as follows_ could be qualified by duration_, so T_follows can be by T_segment_length.
Chapter 7: Extended Representation

Structure is expressed relative to the text in which it occurs. The beginning of the text serves as origin. This mirrors calendars for time for which dates are expressed as a duration since an origin. For instance:

\[(E_0,R)\]: \{\Delta I\text{-subject} - I\Delta: \text{page4}; \Delta I\text{-action} - IO: \text{follows};
\]
\[\Delta I\text{-object} - IO: \text{sengan.thesis.start}\}\]

\[(E_1,R)\]: \{\Delta I\text{-subject} - I\Delta: E_0; \Delta I\text{-action} - IO: \text{T.segment.length};
\]
\[\Delta I\text{-object} - IO: \text{4.pages}\}\]

\[(E_2,R)\]: \{\Delta I\text{-subject} - IO: 4; \Delta I\text{-subject} - IO: 1.page;
\]
\[\Delta I\text{-action} - IO: \emptyset; \Delta I\text{-object} - I\Delta: 4.pages\}\]

\[(E_3,R)\]: \{\Delta I\text{-subject} - I\Delta: \text{page4}; \Delta I\text{-action} - IO: \text{T.segment.length};
\]
\[\Delta I\text{-object} - IO: 1.page\}\]

defines the fourth page of this thesis.

Just as for calendars, a model of the text must be built in SemNet for it to know that the forth page follows the third and precedes the fifth: all the arguments about the fuzzy nature of units still apply. See D.4.3.4 (p. D-62) for more details. Two models can be created for the same text, one corresponding to the logical and one to the physical structure of the text.

The textrefs are connected to the corresponding concepts by two events: \text{words\_used} and \text{phrase\_means}. The \text{phrase\_means} event states that the concepts it links to textrefs are an interpretation of the text they represent. The \text{words\_used} event links the words from which LOLITA’s interpreter derived a fact to the fact itself.

D.7 (p. D-100) and D.8.3.1 (p. D-123) give further details of Textrefs and comprehensively illustrate their uses, which include information retrieval, co-reference in the MUC’95 competition, discussion of different people’s interpretations of a text, and the representation of ambiguity.

7.6.4 Ambiguity

Ambiguity in SemNet stems from the interpretation process: words and sentences in natural language can correspond to a vast number of concepts. Each word of the sentence may correspond many different meanings; there may be more than one
set of functional (grammatical) dependencies between the words of different sub-
phrases of the sentence; and morphological variations of the words themselves, such
as number or tense, may correspond to different meanings that would be expressed
differently in SemNet’s more precise terms. To make matters worse, ambiguity is
combinatorial. That is to say that the numbers of alternatives for each locus of
ambiguity must be multiplied, rather than added, to obtain the ambiguity of the
sentence. For instance, if each word of the sentence has two meanings, and the
sentence contains \( n \) words, the number of interpretations is \( 2^n \) and not \( 2n \). The
two key requirements of the ambiguity representation are thus cohesion – as usual
to avoid multiplying the reasoning tools – and minimising the effects of ambiguity’s
combinatorial nature.

To achieve conciseness of the expression of ambiguity, one must avoid expanding all
the alternatives, but instead somehow express them in a packed form: instead of
multiplying out a product, its terms are retained. As D.8.1.4 (p. D-121) discusses,
ambiguity can be packed in different ways, but schemes which a fixed locality
of ambiguity either prove incapable of expressing all the possible alternatives, or
end up expanding out all the ambiguity. However, D.8.3 (p. D-122) achieves a
representation which satisfies the requirements.

The representation introduces no new events, but uses instead the representations
of belief, textual references and intensional equality. Its cohesion is thus guaran-
teed. The key idea for packing is the concept of variable nodes. Variable nodes are
concepts which are defined to be intensionally equal to one of a set of alternative
concepts using synonym events grouped by a xor event. The variable node can
then be built into standard events, so that the structure they provide is shared
by all of its alternatives. For instance, a fragment of “The frog found the bug”:

\[
(E_0,H): \{ \Delta I-\text{subject}.-I\Delta: \ \forall N_4; \ \Delta I-\text{action}.-IO: \text{find}; \\
\Delta I-\text{object}.-I\Delta: \ \forall N_2 \} ... \\
(E_6,H): \{ \Delta I-\text{subject}.-I\Delta: \ \forall N_3; \ \Delta I-\text{subject}.-IO: \text{frog}_{\text{amphibian}}; \\
\Delta I-\text{action}.-IO: \text{synonym} . \} \\
(E_7,H): \{ \Delta I-\text{subject}.-I\Delta: \ \forall N_3; \ \Delta I-\text{subject}.-IO: \text{frog}_{\text{frenchman}}; \\
\Delta I-\text{action}.-IO: \text{synonym} . \} 
\]
\((E_8,H): \{\Delta I\text{-}subject\_IO: \forall N_3; \Delta I\text{-}action\_IO: \text{inst.}; \\
\Delta I\text{-}object\_I\Delta: \forall N_4\}\)

\((E_9,H): \{\Delta I\text{-}subject\_IO: [E_6, E_7]; \Delta I\text{-}action\_IO: \text{xor.}\} \ldots\)

\((E_{11},H): \{\Delta I\text{-}subject\_IO: [... , E_9, \ldots , E_6, E_9, \ldots ]; \Delta I\text{-}action\_IO: \text{and.}\}\)

\((E_{12},R): \{\Delta I\text{-}subject\_IO: "The frog found the bug"; \\
\Delta I\text{-}action\_IO: \text{phrase}\_\text{means}; \Delta I\text{-}object\_IO: E_{11}\}\)

The belief event is connected to a phrase\_\text{means} event, signifying that it is the interpretation of the sentence represented by the textref. Variable nodes are integrated into the concept hierarchy, to the extent that the features common to all the alternatives are available directly (or via inheritance) on the variable node. This aspect of variable nodes resembles under-specified concepts, and avoids applying many disambiguation algorithms to each of the alternatives.

Variable nodes also allow different levels of grouping to be expressed, so that rough disambiguation algorithms can eliminate whole classes of possibilities using such information as family controls. The anticipated disambiguation process – and that currently implemented – is of refining the meaning via a series of better approximations, i.e. refining the concept granularity considered. If the process achieves the desired logarithmic behaviour, each step will divide the search space by more than one, making it possible to investigate an exponential search space of \(O(e^N)\) in only \(O(N)\) steps.

As explained in D.8.4.4 (p. D-132) the logical events grouping the alternatives allow single interpretations of the whole sentence to be manipulated and even discounted, while maintaining sharing as far as possible. D.8.6 (p. D-136) goes on to show that the representation copes many different types of ambiguity, such as quantification or prepositional attachment.

SemNet is novel in its usage of the standard representation to express ambiguity, resulting in a higher level of flexibility without suffering a loss of efficiency. An example of the gain in flexibility is the ability to express LOLITA's belief in different interpretations with the standard belief\_value events, allowing standard belief reasoning to be applied. Many more issues are discussed in D.8 (p. D-118).
Chapter 8

Evaluation

Evaluation is as crucial in Science as Experimentation, as it determines whether or not the state of our knowledge has advanced or not. This chapter presents an overview of the LOLITA system, a demonstration of SemNet in action in semantic mapping, and an in-depth evaluation of SemNet versus its competitors.

8.1 LOLITA: an Application of SemNet

SemNet is not just a theory. A previous version of SemNet\(^1\) was implemented in a large natural language system: LOLITA. The lessons learnt from that implementation helped in this new revision.

SemNet lies at the core of LOLITA. All new information is input to LOLITA as natural language, which is interpreted and added to a single SemNet network. Every reasoning algorithm takes SemNet as input and produces SemNet as output. Only the final stage of output produces something other than SemNet. This architecture guarantees flexibility since the results of every reasoning algorithm may be used by any other part of the system.

Since SemNet is touted for use in Natural Language applications, this section will focus on the natural language analysis phase.

\(^1\)not LOLITA 92, but a descendent of it
8.1.1 Lolita’s textual analysis

8.1.1.1 Overview

Lolita’s current textual analysis is a 9 stage in-order process whatever the language:

The first step is morphological analysis. The features and root form of each word of the text are determined. For instance “dogs” becomes plural, noun or verb and dog. SemNet’s linguistic layer encodes exceptions and feature information for the rules applying to regular cases.

The second step is grammatical analysis. Functional groups of words are determined from rules governing the ordering of sentences’ words according to their morphological features. The result is a parse-graph expressing the various possible word groupings and the grammatical categories they can have in each grouping. For instance in “John dogs her often”, “dogs” may only be a verb, while “John” must be a subject. This step introduces some ambiguity.

The third step is to convert the parse graph into a set of parse trees ordered by syntactic and discourse preferences: trees that violate syntactic requirements of agreement of number, gender and tense are dispreferred; while trees that correspond to common constructions are preferred. This is done because the next stages
originally were written to manipulate trees.

The fourth step is normalisation. Different word orders may express the same semantic relation. For instance, "You can not have your cake and eat it" is converted into "You can \{not have your cake\} or \{not eat it\}" so that negation is only applied to single verb phrases. Normalisation converts parse trees into a subset of equivalent parse trees to ensure that words are grouped together semantically, which reduces the number and complexity of the semantic mapping rules. Some of the rules also perform disambiguation on the basis of syntactic information alone.

The fifth step is semantic mapping which maps the parse tree into a SemNet expression. After normalisation the parse tree groups together words that are related semantically. Semantic Mapping is applied (mainly) compositionally to the parse tree, starting at the leaves and combining larger and larger branches to produce the final SemNet graph. Anaphoric references are included into SemNet at this stage.

The sixth stage are semantic transformations which are applied to the SemNet produced by semantic mapping. This step does the non-compositional transformations that semantic mapping cannot perform.

Type checking ensures that the events built by semantic mapping are legal, and provides some degree of disambiguation by rejecting interpretations that violate event templates. For instance, in "The astronomer married the star" type checking ensures that "the star" referred to is not an astronomical object.

Semantic integration ensures for every node that it is a descendent/ancestor of all the nodes of which it can be inferred to be a descendent/ancestor from its definition.

Discourse and pragmatics analysis constitute the final stage. Discourse analysis uses the way people generally speak to choose a particular interpretation. For instance, people usually concentrate on a context of a few related concepts. This context is tracked using semantic distance and associativity ([Short et al. 94a, Short et al. 94b]). Pragmatic analysis uses plausible inferences such as causal explanation for disambiguation.
While presuming an order of analysis is inflexible and sub-optimal, it is simple and sufficient for LOLITA's current level.

8.1.1.2 LOLITA's role in SemNet's design

The most important lesson learned from LOLITA is the paramount importance of ambiguity. Initially, it was assumed that the discourse rules determining which parse to choose would remove any ambiguity: the semantic network would never contain any ambiguity. The addition of pronoun resolution led to the introduction of alternative links, for which they proved mainly sufficient.

In 1994 LOLITA's knowledge base was made Wordnet compatible. It was anticipated that the additional Wordnet data would improve analysis. In practice, however, it compounded the ambiguity problems already faced. Furthermore, Wordnet only includes various hierarchies, (such as spec., has.part) and various linguistic relations, which did not have the same meanings as those expected by LOLITA. Alternative links made disambiguation algorithms difficult to program mainly because ambiguous concepts were implicit. For instance, it is very inefficient to connect all the possible referents of a pronoun to the corresponding Textref, and then to delete the connections when possible referents are discounted. When the alternative links were replaced by a subset of SemNet's representation, processing was improved significantly.

LOLITA's treatment of ambiguity was too greedy: rather than allowing ambiguity to be shared, LOLITA's design resulted in statements involving many distinct possibilities, rather than more general statements corresponding to a range of possibilities. For instance, "with a kite" can either be a verb phrase or a prepositional phrase in "A dog saw a man with a kite". Of the five corresponding events, only one is ambiguous, but all must be rebuilt by semantic mapping. Similarly, pronouns were replaced by all their alternatives, prepositional phrases were replaced with their various constructions, rather than leaving them underspecified until they could be disambiguated.

The introduction of Textrefs allows future generations of LOLITA to break from
8.1.2 Natural Language Generation

A three-phased approach is used. Abstract transformations provide a means of producing a wide variety of paraphrases for use by the output stage. They take as input SemNet and produce SemNet as output, acting as another form of inference. For instance, anonymize a generalized expression with a specific one: "I like her" → "I do not dislike her," or replacing "hit with a gun" → "hit." These transformations rely on SemNet's event and entity inheritance hierarchies to reduce their search space and to remain abstract.

The practical benefits of other aspects of SemNet's design were confirmed by the implementation. The explicit use of a node for each concept, including sets rather than quantified expressions in ZCPOD - greatly simplified the mapping rules and the processing of referents.

The implementation strongly influenced the design of the representation. Practical experience of the ill-adjacency of using both a link and an event to express membership of the set hierarchy underlined the importance of uniqueness. Similarly, the need for two kinds of negation emerged from attempts to build a general negation algorithm for an industrial contract. Negation, and other extended representations brought to light the need for multi-levelled quantification at both ends of each arc and observational definitional sorts. Language independence of the representations was enforced by the building of an automatic translation system which unearthed the language dependence of tense, varying vastly from Chinese to Classical Greek.

Chapter 8: Evaluation
A planner determines which nodes the generator should talk about, which it must not mention, and the style to generate at a node level. Although the planner's instructions were not implemented in SemNet, they could have been since they operate on single nodes. This stage was only partially implemented in [Smith 95].

A plan-realiser generates the natural language expression for the chosen nodes. Since not all concepts have a corresponding word in a given language, the plan-realiser searches up the inheritance hierarchy to find concepts that do correspond to words (language isomorphic). For instance "big motorbikes" is the intersection of "big things" and "motorbikes". A similar search for template events occurs to determine whether an event should be generated as a noun (the explosion) or an event (an explosive device exploded). The network guides the plan-realiser since relative clauses correspond to events attached to the concept being realised. Unlike other systems, the plan-realiser is not given a section of the knowledge base to realise. Instead it determines what to say. Distributedness guarantees that any portion of the net that it chooses to realise is sound with respect to the whole network. Non-linearity allows the plan-realiser to traverse the network in the order best suited to generation (which varies according to the chosen language). Finally, as for textual analysis, the lack of implicit concepts simplifies the maintenance of anaphoric references.

SemNet proved critical to [Smith 95]'s approach. All quotes in this paragraph are from [Smith 95] unless otherwise stated. Unlike Conceptual Dependency Theory [Schank 79], the fine granularity of SemNet's concepts simplified the generator "The granularity means that there are many language isomorphic concepts (...) This is in contrast to systems that use primitive concepts or concepts that have a larger grain size than words: in these cases the lexicalisation process is more complicated as for each concept more than one possible word or phrase could be used. For non-language isomorphic concepts, it is easy to follow round the SemNet representation in order to decompose these concepts into language isomorphic ones". Indeed, SemNet's unpacked representation not only guides the plan-realiser's search but also ensures relevant information is "close at hand and where it is needed" (low topological distance and good determinism of search). "The rich knowledge in the semantic
network allows knowledge intensive generation. Prototypical events\(^2\) (...) for example allow paraphrasing via abstract transformations. (...) each SemNet node contain important information for generation (both semantic, e.g. the rank, and linguistic e.g. the presence of an irregular verb". Finally, the "(...) internal representations have been useful. This is not least because they have been built in tandem with the generator's needs as well as the needs of other LOLITA subcomponents"

8.2 Semantic Mapping

This section shows how SemNet helps in the construction of the representation from parse-trees. The examples given here are simplified for reasons of space: ambiguity is not shown.

8.2.1 Basic formalism

LOLITA produces a binary grammatical parse-trees. Semantic mapping rules are applied to these to produce a first pass at an interpretation represented in SemNet.

Semantic mapping are rules that are applied to the parse-tree compositionally, traversing the tree in-order from left to right, following the order of the processed sentences. Each may add new nodes to SemNet (\texttt{graph\_built\_in\_semnet}). Each rule applied to a leaf of the parse-tree produces a semantic pipe, and each rule applied to a branch of the parse-tree combines the semantic pipes of its children. A semantic pipe is a list of semi-arcs and additional information, containing a label and a target, but no source.

Semantic mapping rules are of the form:

\begin{verbatim}
leaf( node\_of\_parse\_tree)
    \rightarrow graph\_built\_in\_semnet
    = resulting\_semantic\_pipe
\end{verbatim}

\(^2\)Template events in our terminology
\texttt{branch( node\_of\_parse\_tree, left\_semantic\_pipe, right\_semantic\_pipe) \\
\quad \rightarrow graph\_built\_in\_semnet \\
\quad = resulting\_semantic\_pipe}

The \texttt{graph\_built\_in\_semnet} section is optional for a given node of the parse-tree.

### 8.2.2 Meta-properties

#### 8.2.2.1 Similar representation of Meta-properties and Properties

SemNet represents meta-properties such as cardinality in the same way as it represents the other properties of a concept. Therefore when building a statement such as ‘‘I have five cats’’ which reaches semantic mapping with the parse:

\begin{verbatim}
  sen (defpronoun I [Sing,Sexed,Nom,Per1])
  (transvp (transvb HAVE [Pres] *19)
    (snouncl (adj FIVE) (commoun CATS [Plur,Sexed,Per3] *4) ))
\end{verbatim}

"I have red cats" would have had the parse:

\begin{verbatim}
  sen (defpronoun I [Sing,Sexed,Nom,Per1])
  (transvp (transvb HAVE [Pres] *19)
    (snouncl (adj RED *4) (commoun CATS [Plur,Sexed,Per3] *4) ))
\end{verbatim}

In both cases the resulting representation consists of a cat node qualified by an event (size or red), and only the quantification changes.

To simplify the discussion, the automatic order of evaluation is not followed. Semantic mapping applied to the non-anaphoric leaves produces semantic pipes containing the generic concepts.

\begin{verbatim}
  leaf(transvb HAVE) = [(Act, have,)]
  leaf(commoun CATS) = [(Unknown, cats)]
  leaf(adj five) = [(Unknown, 5)]
\end{verbatim}

The pronoun "I" is resolved by looking up the current speaker in a datastructure called the Context.

\begin{verbatim}
  leaf(defpronoun I) = [(Unknown, Sengan)]
\end{verbatim}

The \texttt{snouncl} rule builds a subset \texttt{cats}_{22} of \texttt{cats}_{3}, and attaches it to a size.. 5 event. The resulting pipe contains only the \texttt{cats}_{22} concept, making the difference
in parses between "I have five cats" and "I have cats" local:

\[
\text{sen} \quad \text{(defpronoun I [Sing, Sexed, Nom, Per1])}
\]

\[
\text{transvp} \quad \text{(transvb HAVE [Pres] *19)}
\]

\[
\text{(missing.det (commnoun CATS [Plur, Sexed, Per3] *4) ()))}
\]

\[
\text{branch( snouncl ...), [(Unknown, 5)], [(Unknown, cats_3)] )}
\]

\[
(E_{48428}, R): \{ \Delta I\text{-subject.-I0: cats_3}; \Delta I\text{-action.-I0: spec.;}
\]

\[
\Delta I\text{-object.-I}\Delta: \text{cats}_{32} \}
\]

\[
(E_{48429}, R): \{ \Delta I\text{-subject.-I0: cats}_{32}; \Delta I\text{-action.-I0: size.;}
\]

\[
\Delta I\text{-object.-I0: 5 }\}
\]

\[
= [(\text{Unknown, cats}_{32})]
\]

In the "I have red cats" case, \(E_{48429}\) would be is_red with appropriate quantifica-

The \textit{transvp} rule simply labels and merges both pipes:

\[
\text{branch( transvp ...), [(Act, have_7)], [(Unknown, cats_{32})] )}
\]

\[
= [(\text{Act, have}_7), (\text{Obj, cats}_{32})]
\]

The \textit{sen} rule builds the final event:

\[
\text{branch( sen ...), [(Unknown, Sengan)], [(Obj, cats_{32}), (Act, have_7)] )}
\]

\[
(E_{48430}, R): \{ \Delta V\text{-subject.-I}\Delta: \text{Sengan}; \Delta V\text{-action.-I0: have}_7;
\]

\[
\Delta F\text{-object.-F}\Delta: \text{cats}_{32} \}
\]

\[
= [(\text{Unknown, } E_{48430})]
\]

The fact meta-properties are represented in the same way as normal properties also
simplifies the treatment of anaphoric references, such as "There were 5 of them": a size_ event need only be added to referenced node. QLF represents cardinality
by generalised quantifiers, which would require it to change the statement in which
the anaphoric referent first appeared.
8.2.2.2 Propositionality of meta-properties

Another example is "Jack believes John is a soldier" which has as parse:

\[
\text{sen (full.propernoun (propernoun JACK [Sing, Sexed, Nom, Per3]) () )}
\]

\[
\text{(transvp (transvb BELIEVE [Pres] *3))}
\]

\[
\text{(sen (full.propernoun (propernoun JOHN [Sing, Sexed, Nom, Per3]) () )}
\]

\[
\text{(is.a (is.a IS [Pres] *1))}
\]

\[
\text{(detph (det A))}
\]

\[
\text{(commoun SOLDIER [Sing, Per3]) ) ) )}
\]

The semantic mapping proceeds as follows. The propositional nature of SemNet makes the treatment of believing meta-properties or believing properties follow the same path. The only difference being that noun phrases of the form "a x" generate an individual which must be rectified into the generic concept for is_a branches.

\[
\text{leaf(propernoun JACK) = [(Unknown, Jack)]}
\]

\[
\text{leaf(transvb BELIEVE) = [(Act, believe_4)]}
\]

\[
\text{leaf(propernoun JOHN) = [(Unknown, John)]}
\]

\[
\text{leaf(is.a BE) = [(Act, spec.)]}
\]

\[
\text{leaf(det A) = [(Unknown, a)]}
\]

\[
\text{leaf(commoun SOLDIER) = [(Unknown, soldier_1)]}
\]

\[
\text{branch ( detph ..., [(Unknown, a)], [(Unknown, soldier_1)] )}
\]

\[
\text{→ (E_{48432}, R): \{ΔI-subject.-I0: soldier_1; ΔI-action.-I0: inst.; ΔI-object.-IΔ: soldier_3 \}}
\]

\[
\text{= [(Unknown, soldier_3)]}
\]

\[
\text{branch ( is.a ..., [(Act, spec.), [(Unknown, soldier_3)] )}
\]

\[
\text{→ delete([E_{48432}, soldier_3])}
\]

\[
\text{= [(Act, spec.), (Obj, soldier_1)]}
\]

\[
\text{branch ( full.propernoun ..., [(Unknown, John)], [] )}
\]

\[
\text{= [(Unknown, John)]}
\]

\[
\text{branch ( sen ..., [(Unknown, John)], [(Act, spec.), (Obj, soldier_1)] )}
\]

\[
\text{→ (E_{48433}, R): \{ΔI-subject.-I0: John; ΔI-action.-I0: spec.; ΔI-object.-IΔ: soldier_1 \}}
\]

\[
\text{= [(Unknown, E_{48433})]}
\]

\[
\text{branch ( transvp ..., [(Act, believe_4)], [(Unknown, E_{48433})] )}
\]

\[
\text{= [(Act, believe_4), (Obj, E_{48433})]}
\]
branch( full.propernoun ..., [(Unknown, Jack)], [] )
    = [(Unknown, Jack)]

branch( sen ..., [(Unknown, Jack)], [(Act, believe), (Obj, E48433)] )
    → make_hypothetical(E48433)

    (E48434, R): { ΔI-subject-IO: Jack; ΔI-action-IO: believe; 
                   ΔI-object-IO: E48433 }

    = [(Unknown, E48434)]

8.2.3 Example using Definitionals Sorts and Anaphoric Reference

The following example "The man who caught the dog likes it" shows that anaphoric reference is simplified by having every concept as a node, and how the representation of ambiguity is used in semantic mapping. It also illustrates the ease of use of definitional sorts in semantic mapping.

    sen (relcl (detph (det THE)
               (commoun MAN [Sing,Sxed,Per3] *8) )
               (transvp (transvb CATCH [Pres] *13)
                     (detph (det THE)
                     (commoun DOG [Sing,Sxed,Per3] *8) )
               (transvp (transvb LIKE [Pres])
                     (defpronoun IT [Sing,Neutral,Per3] )

    leaf(det THE)  = [(Unknown, the)]
    leaf(commoun MAN) = [(Unknown, man)]
    leaf(transvb CATCH) = [(Act, catch10)]
    leaf(det THE)  = [(Unknown, the)]
    leaf(commoun DOG) = [(Unknown, dog)]
    leaf(transvb LIKE) = [(Act, like2)]

Usually expressions of the form "The X" refer to a concept mentioned in the context. [Gariglione 92] presents a full discussion. Semantic mapping achieves this by using the representation of ambiguity: man2353 is the variable node, man2249 is a man previously mentioned in the context, while man2354 is a man not referred to in the context in case those in the context are discounted. Notice that while man2353
defines \texttt{man}_{2354}, it is observational with respect to \texttt{man}_{2249}. This guarantees that a previously referred to man is not redefined, while a newly introduced concept has a definition. The \texttt{xor} event (\texttt{E}_{48438}) is connected to a \texttt{phrase.mends} event, textrefs not being included for simplicity.

\texttt{branch( detph \ldots, [((Unknown, the)], [((Unknown, man7)]) )}

\texttt{(E}_{48435}, R) : \{ \Delta I\text{-subject.-I0: man7; } \Delta I\text{-action.-I0: inst.; } \Delta I\text{-object.-IΔ: man2353 } \}

\texttt{(E}_{48435}, R) : \{ \Delta I\text{-subject.-I0: man7; } \Delta I\text{-action.-I0: inst.; } \Delta I\text{-object.-IΔ: man2354 } \}

\texttt{(E}_{48436}, H) : \{ \Delta I\text{-subject.-IΔ: man2354; } \Delta I\text{-action.-I0: synonym.; } \Delta I\text{-subject.-I0: man2353 } \}

\texttt{(E}_{48437}, H) : \{ \Delta I\text{-subject.-I0: man2249; } \Delta I\text{-action.-I0: synonym.; } \Delta I\text{-subject.-IΔ: man2353 } \}

\texttt{(E}_{48438}, H) : \{ \Delta I\text{-subject.-I0: E}_{48436}; \Delta I\text{-subject.-I0: E}_{48437}; \Delta I\text{-action.-I0: xor.} \}

= [((Unknown, man2353)]

A similar process occurs for "The dog". However assume that this time there is no pre-existing referent in the context, so the variable node structure is garbage-collected away.

\texttt{branch( detph \ldots, [((Unknown, the)], [((Unknown, dogs)]) )}

\texttt{(E}_{48439}, R) : \{ \Delta I\text{-subject.-I0: dogs; } \Delta I\text{-action.-I0: inst.; } \Delta I\text{-object.-IΔ: dog485 } \}

= [((Unknown, dog485)]

\texttt{branch( transp \ldots, [((Act, catch10)], [((Unknown, dog485)]) )}

= [((Act, catch10), (Obj, dog485)]

After the relcl is built, the belief value of any of the synonym events qualified by \texttt{E}_{48438} and known to violate \texttt{E}_{48440} is reduced. This and other disambiguation steps will reduce the number of possible referents.

\texttt{branch( relcl \ldots, [((Unknown, man2353)], [((Act, catch10), (Obj, dog485)] )}

\texttt{(E}_{48440}, R) : \{ \Delta I\text{-subject.-IΔ: man2352; } \Delta I\text{-action.-I0: catch10; } \Delta I\text{-object.-IΔ: dog485 } \}

= [((Unknown, man2352)]

The anaphoric reference "it" can refer to entities and events alike. For instance, in "The man who swims every morning likes it", the "it" refers to swimming ev-
very morning. Although there is a default meaning of "it" if no referent exists as in "Penguins do it on land", the referent "it" is always anaphoric (unlike "the" above).

\[ \text{leaf( defpronoun IT )} \]

\[ \rightarrow (E_{48441}, R): \{ \Delta I\text{-subject.-I0: typeless; } \Delta I\text{-action.-I0: inst.; } \Delta I\text{-object.-I}\Delta: it_{4826} \} \]

\[ (E_{48442}, H): \{ \Delta I\text{-subject.-I}\Delta: it_{4826}; \Delta I\text{-action.-I0: synonym.; } \Delta I\text{-subject.-I0}: E_{48440} \} \]

\[ (E_{48443}, H): \{ \Delta I\text{-subject.-I}\Delta: it_{4826}; \Delta I\text{-action.-I0: synonym.; } \Delta I\text{-subject.-I0: dog}_{485} \} \]

\[ (E_{48444}, H): \{ \Delta I\text{-subject.-I0: E}_{48442}; \Delta I\text{-subject.-I0: E}_{48443}; \Delta I\text{-action.-I0: xor.} \} \]

\[ = [(\text{Unknown, it}_{4826})] \]

\[ \text{branch( transvp ... , [(Act, like}_2), [(\text{Unknown, it}_{4826})])} \]

\[ = [(\text{Act, like}_2), (\text{Obj, it}_{4826})] \]

Because man\textsubscript{353} is already defined, the new event \( E_{48445} \) does not define it. A discourse rule stating that if an anaphoric reference is ambiguous between an event and its subject or object, then the subject or object is chosen would make "it" refer to "dog".

\[ \text{branch( sen ... , [(\text{Unknown, man}_{2353})], [(\text{Act, like}_2), (\text{Obj, it}_{4826})])} \]

\[ \rightarrow (E_{48445}, R): \{ \Delta I\text{-subject.-I0: man}_{2352}; \Delta I\text{-action.-I0: catch}_{10}; \Delta I\text{-object.-I}\Delta: it_{4826} \} \]

\[ = [(\text{Unknown, E}_{48445})] \]

### 8.2.4 Conclusion

The fact every concept, including events and meta-properties, are nodes simplifies the treatment of anaphoric references. The many representations that required expressions of the form \( \forall x \in X' ; f(x) \rightarrow g(x) \) to define concepts dynamically would not be able to refer to a single symbol corresponding to the implicit concept ranged over by \( x \). The representation of cardinality as a meta-property (rather than a generalised quantifier as in QLF) makes it possible to simply add cardinality after the concept has been defined. The use of definitional sorts on arcs simplifies not changing existing concepts’ definitions while allowing new ones to be defined in
ambiguous situations. LOLITA 92, and KL-ONE would require duplicate events

to achieve the same effect.

8.3 Comparison with other representations

This section compares SemNet with the other major representations discussed in
the literature review.

8.3.1 Quantification

This section contrasts SemNet’s use of quantification to that of other representa-
tions, showing the sentences or operations other representations have difficulties
with.

8.3.1.1 Richness

• Existentials
  ○ Reference to Existentials

SemNet enables statements to be made about existentially quantified elements of
concepts. For instance, “Every person has a secret he does not want revealed. We
can learn those secrets (for blackmail).”.

In CGT, the co-reference link allows the existential to be connected to a new
node serving as object for the second sentence. SNePS and ANALOG both build
an explicit node to represent the existentially quantified variable. LOLITA 92
expresses existentials like SemNet in this respect.

In $\mathcal{FOPL}$, there is no explicit set of the secrets people have$^3$, so the second sentence
must rebuild the set implicitly:

\[ \forall p \in \text{people} \ \exists s \in \text{secrets}.\text{has}(p, s) \land \text{not\_want\_revealed}(p, s) \land \text{learn}(we, s) \]

$^3$Some secrets are companies', not people's.
This renders the processing of natural language more difficult since instead of replacing the anaphoric "*those secrets*" by a single node, it must be replaced by a complex expression.

QLF is based on LF, an extension of first order predicate logic which adds lambda abstraction, generalised quantifiers and higher order operators. Lambda abstraction allows anaphoric references to be easily built at QLF's level, but this will be resolved into LF by substituting the first sentence into the second.

KL-ONE cannot express the second sentence, as it provides no reference-symbol to the existentially quantified variable:

\[
\text{Person} \quad \exists (\exists (\text{has not want revealed}).\text{secrets})
\]

<table>
<thead>
<tr>
<th>(\mathcal{FOL})</th>
<th>QLF</th>
<th>CGT</th>
<th>SNePS</th>
<th>ANALOG</th>
<th>KLONE</th>
<th>LOLITA 92</th>
<th>SemNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

○ Unique existential and Full existential

SemNet can represent both sentences involving unique existentials and full existentials with \(\exists!\) and \(\exists!\forall\) respectively.

KL-ONE can express sentences involving unique existentials using number restrictions: \((\leq 1 \text{ R}) \quad \exists (\exists \text{ R.C}).\) It also can express full existential quantification \((\exists \text{ R.C}).\)

CGT can represent full existential quantification: [\text{STUDENT}: \ast]. It can also represent unique existentials through the use of distributed sets. "Every man is married to a unique wife"

\[
[\text{MAN}: \{\ast\}] \quad \text{AGNT} \quad [\text{IS MARRIED TO}] \quad \text{OBJ} \quad [\text{WIFE: Dist\{\ast\}}]
\]

QLF uses generalised quantifiers. Thus, every is represented \(N^M^*[\text{not}, [\text{eq}, N, M]]\) while unique existential would be represented \(N^M^*[\text{eq}, M, 1]\).

SNePS, ANALOG, and LOLITA 92 cannot express sentences involving unique existentials.

<table>
<thead>
<tr>
<th>(\mathcal{FOL})</th>
<th>QLF</th>
<th>CGT</th>
<th>SNePS</th>
<th>ANALOG</th>
<th>KLONE</th>
<th>LOLITA 92</th>
<th>SemNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>
• Quantification dependencies
  ○ Bijection

SemNet can represent sentences such as “Every day, my chicken lays an egg”, which express a one-to-one relation:

\[(E_0, R): \{\Delta \forall \text{-subject.-}IO: \text{my.chicken}; \Delta \forall \text{-action.-}IO: \text{lays}; \Delta \forall \text{-object.-}\exists!\Delta: \text{egg}\}\]

\[(E_1, R): \{\Delta F \text{-subject.-}F\Delta: E_0; \Delta \forall \text{-action.-}IO: \text{at.time}; \Delta F \text{-object.-}F\Delta: \text{daily.egg.laying.time}\}\]

where the time is further defined as only occurring once within each day’s interval.

CGT can express such sentences with respective sets. LOLITA 92 is identical to SemNet in this respect, using framed universals.

QLF needs two statements to express this: one statement for the \( \forall - \exists! \) and another for the \( \exists! - \forall \). Note that this increases the complexity of the natural language analysis since it must now build two statements every time a bijection is expressed. The same goes for \( \mathcal{FOLP} \), SNePS and ANALOG.

KL-ONE has no means of expressing bijections.

<table>
<thead>
<tr>
<th>( \mathcal{FOLP} )</th>
<th>QLF</th>
<th>CGT</th>
<th>SNePS</th>
<th>ANALOG</th>
<th>KLINE</th>
<th>LOLITA 92</th>
<th>SemNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

○ Branching Quantifiers

Certain natural language sentences cannot be expressible in any linear notation [Barwise 79, Fauconnier 75, Quine 69, Quine 70]. These sentences require a non linear quantifier prefix, such as the Henkin prefix [Henkin 61]. For example “Some relative of each villager and some relative of each townsman hate each other.” An attempt in \( \mathcal{FOLP} \) would be:

\[\forall v \exists r_1 \forall t \exists r_2 \text{Hates}(r_1, r_2)\]

However here \( r_2 \) depends both on \( t \) and on \( v \), when the sentence requires it to depend only on \( t \).
SemNet can represent such sentences:

\[(E_0, R): \{\Delta F\text{-subject\_FO: townsman}; \Delta V\text{-action\_IO: related}\_to; \Delta \forall\text{-object\_∃!Δ: t}\_relative\}\]

\[(E_1, R): \{\Delta F\text{-subject\_FO: villagers}; \Delta V\text{-action\_IO: related}\_to; \Delta \forall\text{-object\_∃!Δ: v}\_relative\}\]

\[(E_2, R): \{\Delta F\text{-subject\_FΔ: t}\_relative; \Delta V\text{-action\_IO: hate}; \Delta F\text{-object\_FΔ: v}\_relative\}\]

\[(E_3, R): \{\Delta F\text{-subject\_FΔ: v}\_relative; \Delta V\text{-action\_IO: hate}; \Delta F\text{-object\_FΔ: t}\_relative\}\]

SNePS and ANALOG are non-linear representations which do not express quantification on their nodes, so are able to express this statement [Ali 93].

CGT is able to represent this sentence using the co-reference link, and the fact that 

**PROPOSITION:** may take more than one argument:

\[
\begin{align*}
\text{[PROPOSITION:} & \text{[Villager: {※}] } \leftarrow \text{ (AGNT) } \leftarrow \text{ [RELATED\_TO] } \rightarrow \text{ (PTNT) } \rightarrow \text{ [V\_relative: ※x]} \\
\text{[Townsman: {※}] } \leftarrow \text{ (AGNT) } \leftarrow \text{ [RELATED\_TO] } \rightarrow \text{ (PTNT) } \rightarrow \text{ [T\_relative: ※y]} \\
\text{[{※x}] } & \leftarrow \text{ (AGNT) } \leftarrow \text{ [HATE] } \rightarrow \text{ (PTNT) } \rightarrow \text{ [※y]} \\
\text{[{※y}] } & \leftarrow \text{ (AGNT) } \leftarrow \text{ [HATE] } \rightarrow \text{ (PTNT) } \rightarrow \text{ [※x]} \]
\end{align*}
\]

LOLITA 92 cannot represent this sentence. "some relative of each villager" and "some relative of each townsman" are modelled by "Each villager (universal) has a relative (existential)" and "Each townsman (universal) has a relative (existential)." But the "(they) hate each other" part of the proposition would require the two sets of relatives to be quantified as framed universals, not existentials (in LOLITA 92, an event cannot take two existentials as arguments).

**FOLP** is a linear representation and cannot express this sentence. QLF is a linear representation, which uses the ordering of terms to express quantification dependencies implicitly like FOPL so cannot represent this sentence. KL-ONE does not represent any quantification dependencies so cannot express this sentence.
\[
\begin{array}{|c|c|c|c|c|c|c|c|}
\hline
\text{FOPL} & \text{QLF} & \text{CGT} & \text{SNePS} & \text{ANALOG} & \text{KLONE} & \text{LOLITA 92} & \text{SemNet} \\
\hline
0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\
\hline
\end{array}
\]

- **Donkey Sentences**

In special cases such as the so called donkey sentences, an anaphoric term may be resolved to a quantified variable \(v\) outside the scope of the quantifier that binds \(v\) [Geach 62].

KL-ONE cannot represent this sentence since it does not provide a reference-symbol for the existentially quantified variable.

\(\text{FOPL}\) cannot represent this satisfactorily since variables are always quantified before they are referred to.

QLF can represent this sentence as:

```latex
quant(forall, X
    [and ,
    [farmer, X],
    quant(exists,Y,[donkey,Y],[own,X,Y]),
    [beats,X,a_index(Y)])]
```

However, the \text{a_index}(Y) term is special in that it is not fully resolved into LF, thereby requiring special interpretation rules.

CGT expresses this as a rule: "If a farmer owns a donkey, then he beats it", which does not involve any quantification dependency. CGT is claimed to be isomorphic with \(\text{FOPL}\) [Sowa 84], so should not be able to represent this sentence. However, its use of co-reference links might be equivalent to QLF's \text{a_index}(Y) term.

SNePS can represent this sentence by using entailment: "Every farmer who owns a donkey" is the ANT while "... beats it" is the CQ. ANALOG can also represent it as shown in 4.5.3 (p. 77).

LOLITA 92 can express this sentence, but with some difficulty (4.9.4 (p. 103)) using set relations.

SemNet expresses this sentence using sorts, rather than unbound dependencies.
<table>
<thead>
<tr>
<th>FOPL</th>
<th>QLF</th>
<th>CGT</th>
<th>SNePS</th>
<th>ANALOG</th>
<th>KLONE</th>
<th>LOLITA 92</th>
<th>SemNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

- Higher order aspects
  - Higher order statements

SemNet can express higher order statements about propositions such as "(1) Every day I eat an apple":

\[(E_0, R): \{\Delta\forall\text{-subject.}-\text{IO}: \text{Sengan}; \Delta\forall\text{-action.}-\text{IO}: \text{eat};
\Delta F\text{-object.}-F\Delta: \text{apple}\}\]

\[(E_1, R): \{\Delta F\text{-subject.}-\text{FO}: E_0; \Delta\forall\text{-action.}-\text{IO}: \text{at.time};
\Delta F\text{-object.}-F\Delta: \text{daily.}\text{apple.eating.time}\}\]

where daily.\text{apple.eating.time} is further defined to occur only once a day. SemNet can also express "(2) Not everything you want you will get":

\[(E_4, R): \{\Delta\forall\text{-subject.}-\text{IO}: \text{Paul}; \Delta\forall\text{-action.}-\text{IO}: \text{want};
\Delta F\text{-object.}-F\Delta: E_2\}\]

\[(E_5, R): \{\Delta I\text{-subject.}-\text{IO}: E_2; \Delta I\text{-action.}-\text{IO}: \text{spec.};
\Delta I\text{-object.}-I\Delta: E_3\}\]

\[(E_6, R): \{\Delta\forall\forall\text{-subject.}-\text{IO}: \text{Paul}; \Delta\forall\forall\text{-action.}-\text{IO}: \text{get};
\Delta F\forall\text{-object.}-F\Delta: E_3\}\]

\[(E_7, R): \{\Delta F\text{-subject.}-F\Delta: E_6; \Delta\forall\text{-action.}-\text{IO}: \text{size.};
\Delta\forall\text{-object.}-I\O: 0\}\]

\[(E_8, R): \{\Delta I\text{-subject.}-\text{IO}: E_3; \Delta I\text{-action.}-\text{IO}: \text{size.};
\Delta I\text{-object.}-A\Delta: E_3\text{-non_zero.size}\}\]

The difficulty here lies in fact to want takes an event as object (wanting an object is to want an event with as object an object: "I want a carrot" could mean I want a carrot to own, I want a carrot to eat)

QLF's events and states are quantified over, allowing events to depend quantificationally on their arguments or not as required. This makes it possible to represent (2). CLE-3 does not have a representation of time, but CLE-6 does and allows quantification over times which should allow (1) to be represented.

\text{FOPL} does not quantify its relations, but could express (1) by making time into
an argument of the eat function:

\[ \forall d \in \text{days} \exists a \in \text{itapples.eat}(sengan, a, d) \]

The way in which \( \text{FOPL} \) models sentence (1) could exacerbate the branching quantifier problem. Likewise CGT can express (1) as:

\[
[\text{Day: *}] \leftarrow [\text{TIME}] \leftarrow [\text{SITUATION}:
(\text{NEG}) \rightarrow [\text{PROPOSITION: [PERSON: SENGAN] \leftarrow (\text{EAT}) \rightarrow [\text{APPLE: *}]}}]
\]

SNePS and ANALOG represent propositions but do not quantify over them, so cannot express (2). However, they appear able to represent (1) in the same way as \( \text{FOPL} \) and CGT.

LOLITA 92 cannot express (1) as discussed in 4.9.6.1 (*p. 107*). It can express (2) however:

\[
(E_0, \forall): \{ \text{subject.: (Paul, I); action.: (wants, I); object.: (Things}_1, \forall) \}
\]

\[
(E_1, \forall): \{ \text{subject.: (Paul, I); no.action.: (gets, I); object.: (Things}_2, \forall) \}
\]

\[
(\text{Things}_1, \forall): \{ \text{spec.: (Things}_2, \forall); \}
\]

\[
(\text{Things}_2, \forall): \{ \text{size.: (not.zero, I);} \}
\]

KL-ONE neither expresses quantificational dependency between universal and existentials, nor has any notion of propositions.

<table>
<thead>
<tr>
<th>( \text{FOPL} )</th>
<th>QLF</th>
<th>CGT</th>
<th>SNePS</th>
<th>ANALOG</th>
<th>KLONE</th>
<th>LOLITA 92</th>
<th>SemNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

○ Number of degrees of quantificational freedom

SemNet has 2 degrees of quantificational per argument and event-type: every arc has quantification on both ends. This enables it to represent all the sentences in this section. For instance, SemNet can represent distributed readings such as "Jane bought all the clothes separately" without adding any additional mode of quantification:
Chapter 8: Evaluation

\[(E_0, R): \{\Delta V\text{-subject-IO}: \text{Jane}; \Delta V\text{-action-IO}: \text{buy}; \Delta F\text{-object-FA}: \text{clothes}_1\}\]

QLF has \(n + 1\) degrees of freedom for \(n\) arguments: every variable has one quantification, as in higher order logic. This means statements such as "There are some people who own properties together" interpreted as follows cannot be expressed:

\[(E_2, R): \{\Delta \exists!\text{-subject-VA}: \text{people}_{43}; \Delta V\text{-action-IO}: \text{own}; \Delta \exists!\text{-object-VA}: \text{properties}_9\}\]

\(\text{FOPL}, \text{SNePS}\) and \(\text{ANALOG}\) have 1 degree of freedom per argument: events are not quantified over, so sentences such as "Every day Sengan waters a different tree" cannot be expressed.

\(\text{CGT}\) and \(\text{LOLITA 92}\) have only one shared degree of freedom per concept: quantification is on the node. Sentences such as "Every mother has a child, and each such child loves a toy" cannot be expressed. In practice this scheme proves so limited that co-reference or synonymy links are used to extend it, at the price of greater topological distance, and lower determinism of search.

\(\text{KL-ONE}\) has only one degree of quantificational freedom: the only means of expressing quantification are existential and universal roles which only quantify over the second argument. Sentences such as "Every man's wife's mother irritates him" cannot be expressed.

<table>
<thead>
<tr>
<th>(\text{FOPL})</th>
<th>QLF</th>
<th>CGT</th>
<th>SNePS</th>
<th>ANALOG</th>
<th>KLONE</th>
<th>LOLITA 92</th>
<th>SemNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>(n)</td>
<td>(n + 1)</td>
<td>(n)</td>
<td>(n)</td>
<td>(n)</td>
<td>1</td>
<td>(n)</td>
<td>2(n)</td>
</tr>
</tbody>
</table>

○ Collective Readings

SemNet can express collective readings such as:

1. "Pat and her husband own the estate (together)"
\[(E_0, R): \{\Delta I\text{-subject-IO}: \text{Pat}; \Delta I\text{-subject-IO}: \text{Pat's husband}; \Delta I\text{-action-IO}: \text{own}; \Delta I\text{-object-IA}: \text{estate}_{22}\}\]

2. "The people who own the estate":
\[(E_1, R): \{\Delta I\text{-subject-V0}: \text{People}; \Delta I\text{-action-IO}: \text{own}; \Delta I\text{-object-IA}: \text{estate}_{33}\}\]
(3) "Two companies bought 5 computers (in all)"

\( (E_2, R): \{\exists ! \text{-subject} \cdot \forall \Delta: \text{companies}; \forall \forall \text{-action} \cdot \text{-IO: buy; }\)
\( \exists ! \text{-object} \cdot \forall \Delta: \text{computers} \}

where companies has size 2, and computers has size 5.

CGT can represent all these statements as:

(1) \([\text{PEOPLE: } \{\text{Pat, Pat's Husband}\}] \leftarrow (\text{AGNT}) \leftarrow (\text{OWN}) \rightarrow (\text{PTNT}) \rightarrow [\text{ESTATE}]\)

(2) \([\text{PEOPLE: } \{\ast\}] \leftarrow (\text{AGNT}) \leftarrow (\text{OWN}) \rightarrow (\text{PTNT}) \rightarrow [\text{ESTATE}]\)

(3) \([\text{COMPANY: } \{\ast\}\{\ast\}] \leftarrow (\text{AGNT}) \leftarrow (\text{BUY}) \rightarrow (\text{PTNT}) \rightarrow [\text{COMPUTER: } \{\ast\}\{\ast\}]\)

QLF and LOLITA 92 can represent (1) and (2) but not (3). QLF represents them as:

(1) \(\text{quant(forall,A,U(Pat,Pat's Husband)},\)
\(\quad [\text{pres,quant(exists,E,[event,E],[vote1,E,A])}]\)

(2) \(\text{quant(forall,A,People},\)
\(\quad [\text{pres,quant(exists,E,[event,E],[vote1,E,A])}]\)

while LOLITA 92 represents them as:

\( (E_0, I): \{ \text{subject.}: (\text{Pat}, I); \text{subject.}: (\text{Pat.husband}, I); \text{action.}: (\text{own}, I); \)
\( \text{object.}: (\text{Estate}_1, I) \} \)

\( (E_1, I): \{ \text{subject.}: (\text{People}_2, \forall); \text{action.}: (\text{own}, I); \text{object.}: (\text{Estate}_1, I) \} \)

\( \mathcal{FOPC}, \text{SNePS and ANALOG can represent (1), but not (2) or (3), because they do not quantify over their molecular nodes/predicates. In (1), the molecular node is qualified by two arguments (\( \mathcal{FOPC} \) equivalent to own(Pat, PH, estate)), but in the second the people are qualified by a } \forall \text{ which always entails an event per argument:} \)

\( \forall p \in \text{people.own}(p, \text{estate}) \)

KL-ONE lacks any notion of event. Thus while it is possible to state that Pat owns an estate and pat.husband owns an estate, there is no means of stating that they both do. Furthermore there is no cardinality applied to concepts being defined, it is only possible to express "companies that buy 5 computers".
Distributive readings

SemNet can express distributive readings such as:

1. "The two representatives voted" (2 separate events):

\[(E_3, R): \{\Delta F\text{-subject.-}FO: \text{representatives}_{51}; \Delta \forall\text{-action.-}IO: \text{voted}; \\}
\]
\[\Delta \forall\text{-object.-}∃\forall\Delta: \text{outcome}_{22}\}\]
where representatives_{51} has size 2.

2. "Two companies each bought 5 computers":

\[(E_4, R): \{\Delta F\text{-subject.-}FΔ: \text{companies}; \Delta \forall\text{-action.-}IO: \text{buy}; \Delta F\text{-object.-}F\forallΔ: \text{computers}\}\]
where companies has size 2, and each element of computers has size 5. SemNet can also represent the interpretation (3) in which each of the two companies bought a possibly different set of 5 computers:

\[(E_5, R): \{\Delta F\text{-subject.-}FΔ: \text{companies}; \Delta \forall\text{-action.-}IO: \text{buy}; \Delta \forall\text{-object.-}∃\forallΔ: \text{computers}\}\]
where companies has size 2 and the elements of computers have size 5. Note that if one really wanted to ensure no interpretation where the two companies bought the same computers one would need to add a further antonym event.

\(\mathcal{FOPC}\) can represent (1), (2) and (3):

1. \(∀r ∈ \text{representatives} \exists o ∈ \text{outcome vote}(r, o)\)
2. \(∀c ∈ \text{companies} \exists! b ∈ \text{computer batches} ∀i ∈ b . \text{buy}(c, i)\)
3. \(∀c ∈ \text{companies} \exists b ∈ \text{computer batches} ∀i ∈ b . \text{buy}(c, i)\)

QLF can represent (1) and (3) but not (2) because the generalised quantifiers are not guaranteed to select different elements of a set.

(1) quant(forall, A,
\[
[\text{subset,A,representatives},
\text{[past,quant(exists,E,[event,E],[vote1,E,A])]])}\]

(3) quant(N*M*[eq,M,2],A,[company,A],
\[
\text{quant(set(K*L*[eq,L,5]),B,[computer,B]}
\text{[past,quant(exists,E,[event,E],[order,E,A,B])]])}\]
CGT, SNePS, ANALOG, KL-ONE, and LOLITA 92 can represent (1) but not (2) or (3). CGT cannot represent it because it has no predicate to range over sets. LOLITA 92, SNePS and ANALOG lack multilevelled quantification. KL-ONE cannot express anything beyond "companies that buy 5 computers" (which may or may not refer to the same computers).

<table>
<thead>
<tr>
<th>FOPL</th>
<th>QLF</th>
<th>CGT</th>
<th>SNePS</th>
<th>ANALOG</th>
<th>KLONE</th>
<th>LOLITA 92</th>
<th>SemNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

- Partial arcs

SemNet represents many individual concepts participating in an event by partial arcs. This allows statements such as (1) "The people who carried the piano":
\[(E_0, \mathbf{R}): \{\Delta I\text{-subject-}\mathbf{\forall} \Delta: \text{people}_3; \Delta I\text{-action-}\mathbf{\forall} \Delta: \text{carry}; \]
\[\Delta I\text{-object-}\mathbf{\forall} \Delta: \text{piano}_1\}\]
to be made about single individuals (2) "John and Mary carried the piano":
\[(E_0, \mathbf{R}): \{\Delta I\text{-subject-}\mathbf{\forall} \Delta: \text{John}_5; \Delta I\text{-subject-}\mathbf{\forall} \Delta: \text{Mary}_2; \]
\[\Delta I\text{-action-}\mathbf{\forall} \Delta: \text{carry}; \Delta I\text{-object-}\mathbf{\forall} \Delta: \text{piano}_1\}\]
Partial arcs avoids in many cases the need to build sets corresponding to unions of individuals. This is important because it reduces the search-space of finding all the events in which a concept participated; it reduces the amount of semantic integration required; and should reduce the complexity of reasoning algorithms since unions are disjunctive concepts (as witnessed by inheriting any of union_'s definitions to the level of the concepts forming the union will show).

QLF requires all cases equivalent to partial arcs to be converted to unions, but can represent (1) and (2). Similarly, CGT can represent (1) and (2) using collective sets.

SNePS may be able to represent (1) using rule nodes, and can represent (2). ANALOG lacks rule nodes so cannot cannot represent (1) but could represent (2).

LOLITA 92 has some notion of partial arcs connecting an individual event to individual concepts (such as the and_ and or_ events). However no rules were defined as to how such partial arcs behaved if attached to universal or existential concepts. Quantification cannot take over from partial arcs for this reason, making
LOLITA 92 incapable of representing (1), but able to represent (2).

\( \mathcal{FOPC} \) does not allow the same predicate to be applied to a variable number of arguments. But, it would allow (2) to be represented with a different predicate. KL-ONE has no notion of partial arcs, having no notion of events. LOLITA 92

<table>
<thead>
<tr>
<th>( \mathcal{FOPC} )</th>
<th>QLF</th>
<th>CGT</th>
<th>SNePS</th>
<th>ANALOG</th>
<th>KLONE</th>
<th>LOLITA 92</th>
<th>SemNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>4</td>
<td>3(^4)</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

- Multilevelled quantification
  - In sentences

SemNet can express multilevelled quantification such as "The number of sheep every shepherd owns, is given by his age" or "Two companies bought a possibly different set of 5 computers".

\((E_4, R): \{ \Delta F\text{-subject} - F \Delta: \text{companies}; \Delta \forall \text{-action} - IO: \text{buy}; \Delta \forall \text{-object} - \exists ! \forall \Delta: \text{computers}\}\)

where companies has size 2 and the elements of computers have size 5.

\( \mathcal{FOPC} \) can express multilevelled quantification as in

\[ \forall c \in \text{companies} \exists s \in \text{computers} \forall x \in s. \text{buy}(c, x) \]

Although CGT is claimed to be isomorphic with \( \mathcal{FOPC} \), two differences make multilevelled quantification unlikely. Firstly, CGT requires that each concept have a type, so not only the type "elephants", but also the type "elephant groups" (etc) would be needed. How this would be expressed in the type-lattice is unclear. Secondly, it is unclear how the relation between \( s \) and \( c \) in the \( \mathcal{FOPC} \) statement could be expressed in a CGT statement which relates \( c \) and \( x \).

QLF has some notion of 2-levelled quantification, so that "Two companies each ordered 5 computers" can be represented:

\[
\text{quant}(N^M^+[\text{eq}, M, 2], A, [\text{company}, A], \text{quant}(K^L^-[\text{eq}, L, 5], B, [\text{computer}, B] ),
\]

\(^3\)SNePS is awarded less than QLF because its rule nodes may not express (1) correctly.
But QLF cannot refer to groups. So it could not state that the two companies bought the same batch of computers.

\( \mathcal{FOPL} \), SNePS, ANALOG, KL-ONE and LOLITA 92 have no notion of multi-leveled quantification.

<table>
<thead>
<tr>
<th>( \mathcal{FOPL} )</th>
<th>QLF</th>
<th>CGT</th>
<th>SNePS</th>
<th>ANALOG</th>
<th>KLONE</th>
<th>LOLITA 92</th>
<th>SemNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

\* Predicates on sets

SemNet uses individual quantification to refer to sets. This makes sets into first class citizens that can be quantified over. This goes beyond multilevelled quantification which allows the grouping of elements to be stated and statements to be made about the atomic elements of the sets. Predicates on sets allows statements to be made about any level of grouping within a given set. Thus, SemNet can represent "Jane has 3 apples in each of her piles of apples":

\( (E_0, R): \{ \Delta V\text{-subject}.-\text{IO}: \text{Jane}; \Delta V\text{-action}.-\text{IO}: \text{own}; \Delta F\text{-object}.-\text{F\Lambda}: \text{apple.piles}_2 \} \)

\( (E_2, R): \{ \Delta F\text{-subject}.-\text{FO}: \text{apples.piles}_2; \Delta V\text{-action}.-\text{IO}: \text{size.}; \Delta V\text{-object}.-\text{IO}: 3 \} \)

This and SemNet’s ability of referring to existentially quantified elements of a set, enable SemNet to represent statements about existentially quantified sets. For instance "Every farmer that owns a donkey beats it. There were 5 such donkeys in Italy in 1992."

Even though QLF has a notion of 2-levelled quantification, it cannot refer to sets: only their elements, and therefore cannot represent this statement.

\( \mathcal{FOPL} \), SNePS, ANALOG, KL-ONE, and LOLITA 92 cannot express this. SNePS, ANALOG and LOLITA 92 do have some relations that apply to sets, but this must be derived from the relation’s type, which reduces cohesion and their distributedness.
Events as first class citizens

In the main, SemNet treats events like any other concept. First, events can be qualified by cardinality. This corresponds to their use in natural language, where "5 wins for Arsenal" and "Arsenal won 5 times" mean the same thing. Second, events are quantified over not only by other events, but by the event's own arcs. This allows an event set to be structured by time into groups:

\[(E_1, R): \{\Delta F\text{-subject.}-F\forall\Delta: E_0; \Delta \forall\text{-action.} - IO: at\text{.time};\]
\[\Delta F\text{-object.}-F\Delta: \text{times}_2\}\]

Quantification by an event's own arcs is what enables the collective readings above to be expressed, such as the statement: "All people who own individually or as a group a theatre":

\[(E_2, R): \{\Delta \exists\text{-subject.}-\forall\Delta: \text{people}_43; \Delta \forall\text{-action.}-IO: own;\]
\[\Delta F\text{-object.}-F\Delta: \text{theatre}_7\}\]

In QLF, events are quantified over and can be given a cardinality so QLF treats its events as first class citizens.

SNePS and ANALOG do not allow quantification over events or cardinality to be applied to them. In fact SNePS and ANALOG do not have any notion of cardinality, but I am assuming that if one were available, it would take the form of an event. However, SNePS and ANALOG do not allow a given event to be defined by other events, which would reduce cardinality to apply only in the observational case.

CGT does not quantify over its events, or allow cardinality to be applied to them. \(\mathcal{FOPL}\) is not a higher order logic so cannot quantify over events. KL-ONE has no notion of events.

While LOLITA 92 did have quantification on its events, adding a size event to them was not considered legal. Similarly, events' quantification was not related to that of their subject\_s and object\_s, reducing richness.
8.3.1.2 Cohesion

- Extensibility
  - Events as first class citizens

In the main, SemNet treats events like any other concept. The fact that events can be qualified by cardinality, enables SemNet to express absence of occurrence without extension: "John ate no apples" and "John didn't eat any apples" are represented by an event identical to $E_0$ except that in the first case, the apples are qualified by \texttt{size. 0}, whereas in the second case the event is qualified by \texttt{size. 0}. Because of the $F - F$ quantification, the equivalence of the two statements can be inferred trivially by a quantification algorithm.

$$(E_0, R): \{\Delta V\text{-subject.-IO: John; } \Delta V\text{-action.-IO: eat; } \Delta F\text{-object.-F}A: \text{apples}_2\}$$

The fact that SemNet's arcs quantify over their events makes it easy for algorithms that consider only quantification to use arcs as their basic unit. The fact that the same quantification rules apply to SemNet's arcs and events simplifies SemNet's interpretation rules, only requiring an arc level interpretation function. Thus an algorithm deducing cardinality relationships from quantification need not concern itself with events. Similarly, inheritance algorithms determining the section of graph to be inherited, need only operate on arcs to determine the quantificational dependencies they must respect.

The fact that SemNet's events are quantified over by other events and by events' own arcs makes SemNet very easy to extend: any quantificational dependency that can be expressed by an event's arc can also be expressed by an extension event. This includes enabling the choice of an event's subject. to quantificationally depend on the extension event.

QLF does not distinguish between its events and arguments as far as quantification
is concerned. Although its structure would allow it to express absence of occurrence using zero-cardinality of events, the examples of [Alshawi 92] p.155 "John did not see anyone" use a not predicate. This indicates QLF does not use zero-cardinality to express absence of occurrence. While CLE-3 does not allow events to be extended, CLE-6 would appear to.

SNePS and ANALOG do not allow quantification over events, and their extensibility is limited (4.5.5.1 (p. 80)). LOLITA 92's extensibility was limited in the same way; algorithms had to take full events as quantificational units; and cardinality could not be applied to events.

CGT does not quantify over its events, but does allow events to be referred to as individual propositions, which is sufficient for many epistemological statements. However this does allow the representation to be extended by external events, creating problems such as 4.5.5.1 (p. 80). $\mathcal{FOPL}$ is not a higher order logic so cannot quantify over events; it does not allow events to be referred to; and extending it is not easy since it requires new arguments to be added predicates. KL-ONE has no notion of events.

<table>
<thead>
<tr>
<th>$\mathcal{FOPL}$</th>
<th>QLF</th>
<th>CGT</th>
<th>SNePS</th>
<th>ANALOG</th>
<th>KDONE</th>
<th>LOLITA 92</th>
<th>SemNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

\* Multilevelled Quantification

Multilevelled quantification is a very important part of SemNet's cohesion and modularity. Without it, many statements which rely on quantificational dependencies between groups of things cannot be expressed. For instance, "John eats 5 apples a day" requires multilevelled quantification if one is to be able to represent that each apple is eaten during a distinct eating event:

$(E_0, R)$: \{$\Delta \forall \cdot \text{subject}.-\text{IO}: \text{John}; \Delta \forall \cdot \text{action}.-\text{IO}: \text{eat};$

\hspace{1em} $\Delta \forall \cdot \text{object}.-\text{FF} \Delta: \text{apples}_{28}$\}

$(E_1, R)$: \{$\Delta \forall \cdot \text{subject}.-\text{FFO}: E_0; \Delta \forall \cdot \text{action}.-\text{IO}: \text{at.time};$

\hspace{1em} $\Delta \forall \cdot \text{object}.-\text{FF} \Delta: \text{times}_{124}$\}

$(E_2, R)$: \{$\Delta \forall \cdot \text{subject}.-\text{FFO}: \text{times}_{124}; \Delta \forall \cdot \text{action}.-\text{IO}: \text{is.in};$

\hspace{1em} $\Delta \forall \cdot \text{object}.-\text{FF} \Delta: \text{days}_{37}$\}
$(E_3, R): \{\Delta F\text{-}subject\text{-}FO: \text{apples}_{28}; \Delta V\text{-}action\text{-}IO: \text{size}_1; \\
\Delta V\text{-}object\text{-}IO: 5\}$

In general, it is necessary to be able to structure events’ grouping according to many extended events: location, time, frame of existence, etc.

Multilevelled quantification is also essential in enabling negation by absence of occurrence. To use absence of occurrence with any event which would normally involve multiple occurrences, a new level of quantification must be built. For instance, "All the people that John has not met" requires multilevelled quantification.

$(E_4, R): \{\Delta VV\text{-}subject\text{-}IO: \text{John}; \Delta VV\text{-}action\text{-}IO: \text{meet}; \\
\Delta \exists! V\text{-}object\text{-}\exists! \Delta: \text{people}_2\}$

$(E_5, R): \{\Delta F\text{-}subject\text{-}F\Delta: E_4; \Delta V\text{-}action\text{-}IO: \text{size}_1; \Delta V\text{-}object\text{-}IO: 0\}$

QLF’s lack of multilevelled quantification reduces the extensibility of CLE-6 and may explain its use of the not predicate instead of 0-cardinality. LOLITA 92’s lack of dependency between the quantification of events and their arguments prevents any increase in extensibility that multilevelled quantification would give it.

<table>
<thead>
<tr>
<th>$FOPC$</th>
<th>QLF</th>
<th>CGT</th>
<th>SNePS</th>
<th>ANALOG</th>
<th>KLONE</th>
<th>LOLITA 92</th>
<th>SemNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

- Quantification
  - Full Existential Quantification

Multilevelled quantification in SemNet provides a natural extension to unique existential and framed universal quantification allowing them to range over sets of arbitrary size. For instance “Every unicycle has one wheel” is represented:

$(E_6, R): \{\Delta F\text{-}subject\text{-}F\Delta: \text{unicycle}; \Delta V\text{-}action\text{-}IO: \text{has\_part}; \\
\Delta F\text{-}object\text{-}F\Delta: \text{wheels}_{20}\}$

It could also be represented:

$(E_2, R): \{\Delta F\text{-}subject\text{-}F\Delta: \text{unicycle}; \Delta V\text{-}action\text{-}IO: \text{has\_part}; \\
\Delta F\text{-}object\text{-}F\forall\Delta: \text{wheels}_{21}\}$

$(E_3, R): \{\Delta F\text{-}subject\text{-}FO: \text{wheels}_{21}; \Delta V\text{-}action\text{-}IO: \text{size}; \\
\Delta V\text{-}object\text{-}IO: 1\}$
"Every bicycle has 2 wheels" is represented by changing the size of the second representation to 2:

\[(E_4, R): \{\Delta F\text{-subject-}-F\Delta: \text{bicycle}; \Delta\forall\text{-action-}-IO: \text{has part};\]
\[\Delta F\text{-object-}-F\forall\Delta: \text{wheels}_{22}\}\]

\[(E_5, R): \{\Delta F\text{-subject-}-FO: \text{wheels}_{22}; \Delta\forall\text{-action-}-IO: \text{size};\]
\[\Delta\forall\text{-object-}-IO: 2\}\]

\(\mathcal{FOPL}\), QLF, and KL-ONE have equally cohesive solutions which do not use multilevelled quantification. CGT requires an additional quantification mode (Distributed sets) to express unique existentials. SNePS, ANALOG, and LOLITA 92 cannot represent unique existentials.

<table>
<thead>
<tr>
<th>(\mathcal{FOPL})</th>
<th>QLF</th>
<th>CGT</th>
<th>SNePS</th>
<th>ANALOG</th>
<th>KLONE</th>
<th>LOLITA 92</th>
<th>SemNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>-</td>
<td>2</td>
</tr>
</tbody>
</table>

○ Partial arcs

SemNet's treatment of partial arcs is cohesive in that partial arcs take over where quantification stops. This allows any statement about some concept to be inherited down to that concept. For instance if an event is connected to many instances of a set by a \(I - \forall\) quantification, it can be inherited down to a known instances using partial arcs using (many) \(I - I\) arcs. This means that the interpretation function that returns all the facts known about a given concept can express them SemNet, allowing higher level interpretation functions to take SemNet as input.

QLF requires all cases equivalent to partial arcs to be converted to unions. While this does build on an existing (union) representation, it does not allow events to be directly connected to the concepts which participate in them. This means that the interpretation function that returns all the known facts about a given concept cannot express them in QLF... either requiring all higher level interpretation functions to take something other than QLF as input, or to know how to search for all information about a concept themselves.

CGT has no notion of partial arcs. Instead the equivalent for entities can be expressed using collective sets. For propositions, only conjunction and disjunction
exist, expressed by nested propositions. This is even less cohesive than QLF since two mechanisms exist for the same task.

SNePS and ANALOG do not quantify over molecular nodes, but do connect their various types of entailment operators to other molecular nodes. Because predicates are not quantified, quantification cannot express partial arcs applied to sets. However, SNePS' rule nodes may make up for this. ANALOG has no rule nodes. The existence of two mechanisms for SNePS makes it less cohesive than QLF. LOLITA 92's treatment of partial arcs is similar to ANALOG's: no way of expressing partial arcs applied to sets, rendering cohesion minimal.

\texttt{FOPL} is not cohesive since a different number of arguments requires a different predicate. KL-ONE has no notion of partial arcs, having no notion of events.

<table>
<thead>
<tr>
<th>\texttt{FOPL}</th>
<th>QLF</th>
<th>CGT</th>
<th>SNePS</th>
<th>ANALOG</th>
<th>KZONE</th>
<th>LOLITA 92</th>
<th>SemNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>3</td>
<td>2\textsuperscript{5}</td>
<td>1</td>
<td>-</td>
<td>1</td>
<td>5</td>
</tr>
</tbody>
</table>

- **Number of modes of quantification**

Some other representations do not share SemNet's multilevelled quantification, high degree of quantificational freedom and partial arcs. Instead they add different modes of quantification to improve their richness. This is less cohesive than SemNet's unique system of quantification.

CGT has four modes of quantification: Collective, Distributive, Disjunctive and Respective. The two latter modes are shorthands, so shall not be considered. Relations applied to distributive sets are true for each of their elements separately. Effectively, this is a means of ensuring that there is one event per element of the argument distributive set. All the elements of a collective set participate in the relations that qualify them together. Effectively, this means there is one event for all the elements of the set. While this enables CGT to express many concepts, the use of a collective set makes it impossible to express $1 + 1 = 2$ without two different links to the two ones.

QLF has three modes of quantification:

\[^{4}\text{SNePS is awarded less than QLF because if rule-nodes are not appropriate, it is less cohesive.}\]
Chapter 8: Evaluation

- The variable in normal quantification ranges over the elements of sets.
- The variable in set quantification ranges over the sets themselves.
- The variable in subset quantification ensures an event per element.

<table>
<thead>
<tr>
<th>FOPL</th>
<th>QLF</th>
<th>CGT</th>
<th>SNePS</th>
<th>ANALOG</th>
<th>KLONE</th>
<th>LOLITA 92</th>
<th>SemNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

- **Common element rule**

The common element rule states that were an instantiation of a concept to be chosen for reference by one event, all other events applied to that concept also choose that instantiation. The common element rule is necessary to express "Every person loves his/her mother" rather than "Every person loves someone's mother". SemNet's common element rule applies not only to atomic elements of concepts, but also any level of quantification within a concept quantified by multilevelled quantification. Thus, the choice of an element of locations2 determines which group of events of E0 is chosen:

(E1, R): \{ΔF-subject-F∀O: E0; Δ∀-action-JO: in.loc; ΔF-object-FΔ: locations2\}

This type of choice of events dependent on location, etc, is possible in SemNet because quantification treats events like any other concept.

FOPL, SNePS, ANALOG and QLF use explicit variables, so they share the common element rule with SemNet. CGT and LOLITA 92 express quantification on the nodes making the common element rule implicit. However, the rule is more powerful in SemNet because quantification is richer in SemNet.

KL-ONE does not share this feature.

<table>
<thead>
<tr>
<th>FOPL</th>
<th>QLF</th>
<th>CGT</th>
<th>SNePS</th>
<th>ANALOG</th>
<th>KLONE</th>
<th>LOLITA 92</th>
<th>SemNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

8.3.2 Sorts and the Inheritance Hierarchy

The use of sorts is intimately entwined with the inheritance hierarchy in SemNet: concepts are defined so that their position in the inheritance hierarchy can be
uniquely ascertained, leading to benefits such as the guaranty of uniqueness.

8.3.2.1 Definition of concepts

Most representations can define the concepts they use.

In SemNet each arc has a sort at each end which specifies whether the arc is definitional with respect to the concept at that end. Furthermore, linguistic nodes are assumed unique if their label and controls are unique, enabling atomic concepts to be defined. The definitions of concepts are used to automatically place them within SemNet’s inheritance hierarchy, a process known as semantic integration. No human hand-crafted the hierarchy.

KL-ONE defines concepts in terms of other concepts and concepts’ fulfilling of certain roles. Concepts may also be defined as atomic. The definitions of concepts are used to automatically place them within KL-ONE’s inheritance hierarchy, a process known as classification. No human hand-crafted the hierarchy.

LOLITA 92’s concepts are defined by their position in the inheritance hierarchy which is hand-crafted by a knowledge-enterer. A (non-atomic) entity is defined by the events which are connected to it and which cannot be inherited from any of its ancestors. The same rule applies to defining events by events that qualify them (all events that occurred today). However, in LOLITA 92, an event’s arcs are considered to always define it – unless the event has a prototypical control in which case it defines its arguments. Actions and entities marked as named-individuals are assumed atomic.

CGT can define both entity and relation concept types as monadic abstractions \( \lambda a \ u \) where there is only one variable: \( a \). The resulting types are placed in a type-lattice by knowledge enterers. Although CGT does not discuss atomic concepts, it introduces them implicitly as those undefined concepts that are used in the definition of others.

SNePS and ANALOG define their classes of entities, properties and actions with LEX arcs to sensory nodes. Thus the definition of named concepts is considered
outside the scope of the KB. Classes are concepts with many instantiations. Like LOLITA 92, molecular nodes are always defined by the nodes they dominate, but unlike LOLITA 92, there is no support for a template event equivalent.

QLF and FOPL have no notion of defining named concepts: concepts are assumed different if their identifiers are different, but no definition is given. They will therefore not be discussed in this section.

- Richness of definitional representational devices

SemNet uses the same language to make definitions and assertions guaranteeing that the richness of definitions is no lower than that of assertions. Similarly, anything can be defined: events, entities, concepts with many instantiations as well as individuals. Because SemNet has two sorts per arc, each arc can define or state things about their events or their arguments. This flexibility enables the representation of:

(1) "Who ate my caramel?"

(2) "It was John who spilled the juice on the carpet"

\[(E_0, R): \{OI\text{-subject-}I\Delta: \text{person}_{4989}; \Delta I\text{-action-}IO: \text{eats}_3; \\\n\Delta I\text{-object-}IO: \text{caramel\_chunk}_8\}\]

where \(E_0\) and \(\text{caramel\_chunk}_8\) are further defined by elements extracted from the context, such as time, whose caramel, and the like.

\[(E_1, R): \{OI\text{-subject-}IO: \text{John}_6; \Delta I\text{-action-}IO: \text{spill}_3; \\\n\Delta I\text{-object-}I\Delta: \text{juice}_7\}\]

LOLITA 92 uses the same language to define entities as to express facts about them. Because an event's arcs are considered to always define it, LOLITA 92's assertional capabilities are reduced: it is impossible to state (2), while (1) requires \(\text{person}_{4989}\) to be marked with a \text{status} arc with target \text{wh\_question}. LOLITA 92 can define entities, events to some extent, concepts with many instantiations as well as individuals.

CGT can define both entity and relation concept types: these are concepts with many instantiations. Since definitions are expressed with the same constructs as assertions, no richness is lost due to differing representations. CGT cannot define individual entities or events making (1) and (2) unrepresentable.
SNePS and ANALOG can define classes of entities, properties and actions. It also defines molecular and base nodes – individuals in statements. This latter type of definition would be similar to a SemNet network in which all arcs are $\Delta - \Delta$, but it is used to make statements rather than define concepts. Therefore this evaluation does not count it as being a form of definition. Thus SNePS and ANALOG cannot define complex classes of entities (people who kill elephants) or individuals.

Members of the KL-ONE family use different representations to define concepts (T-box) and to make assertions about the concepts thus defined (A-box). In most cases [Woods et al. 92] (a notable exception being CLASSIC), the T-box language is weaker than the A-box language. The use of two languages increases the cost of reasoning which must translate facts from the T-Box to the A-Box to infer facts from combinations of definitions and assertions. Unlike most other representations, KL-ONE’s T-boxes cannot relate concepts, only subsets of concepts: \texttt{Men} $\sqcap \forall \texttt{has\_child}\texttt{Female}$ specifies the set of men with female children (a subset of female). Thus it cannot express “Every man’s wife’s mother irritates him”. Similarly it cannot define events. Role intersection would correspond in SemNet to two events restricting a concept, which is not equivalent to defining events. KL-ONE can only define entities with many instantiations.

<table>
<thead>
<tr>
<th>QLF</th>
<th>CGT</th>
<th>SNePS</th>
<th>ANALOG</th>
<th>KLONE</th>
<th>LOLITA 92</th>
<th>SemNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

- Extensions to improve definitional ability

The representational devices listed above are those which are only used to define concepts. For many representations, they prove too limited for practical usage. They are either insufficiently rich to express complex concepts (SNePS, ANALOG, QLF), or cannot be used when new information is added automatically to the KB (CGT). Adding information to KBs is necessary for NLP since sentences such as “People who own pets, usually like them” define new concepts (“People who own pets”).

To solve this, SNePS, ANALOG, CGT use an extension: expressions of the form $\forall x \in X', f(x) \rightarrow g(x)$. 
QLF introduces variables by a term of the form
\[
\text{quant}((\text{quantifier}), (\text{variable}), (\text{restriction}), (\text{formula}))
\]
where the restriction is a truth statement the variable must obey. For instance,

\[
\text{quant}(\text{forall}, B, \\
\quad [\text{and}, [\text{and}, [\text{book1}, B], [\text{interesting1}, B]], \\
\quad [\text{pres}, \text{quant}(\text{exists}, E, [\text{event}, E], \\
\quad \quad \quad [\text{buy}, E, \text{mary1}, B])]), \\
\quad [\text{pres}, \text{quant}(\text{exists}, D, [\text{event}, D], [\text{read}, D, \text{john1}, B])])
\]

restricts the variable B to being an interesting book that Mary bought.

The concepts thus defined are implicit, expressed by the range over which the restricted variable can range. While this does allow facts about the concept's instantiations to be expressed, it does not allow properties of the concept itself to be discussed: the concepts are not explicit, no symbol is introduced that corresponds to all the concept's instantiations. Thus, the cardinality of a concept cannot be represented. Similarly, the following statements cannot be represented:

1. "The departmental football team won the trophy every year since 1979."
2. "The departmental football team is also the university's golf team."
3. "Nobody plays both in the departmental football team and in the cricket team"

SNePS, ANALOG, and CGT are further limited since their representations are not higher order. They cannot use this extension to define events, further reducing richness. For instance, they could not dynamically add (4) to the KB: "bomb explosions" are events with subject_s bombs:

4. "Yesterday's bomb explosions broke killed at least 5 people"

QLF is slightly richer than SNePS, ANALOG and CGT. Its generalised quantifiers allow it to state how many instantiations of a given concept are involved in a statement. However generalised quantifiers do not state how many instantiations a concept has. Being higher order, QLF can define events.

Beyond the limitations of implicit concepts discussed above, the concepts defined with this extension have the full range of expressiveness that their representations afford observational statements.
Because SemNet, LOLITA 92 and KL-ONE do not resort to this extension mechanism, they are richer.

<table>
<thead>
<tr>
<th>QLF</th>
<th>CGT</th>
<th>SNePS</th>
<th>ANALOG</th>
<th>KLONE</th>
<th>LOLITA 92</th>
<th>SemNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

- **Definition maintenance**

KL-ONE and SemNet use classification to maintain their inheritance hierarchies. This cuts search costs since determinism of search is improved: a concept’s definition decides its place in the hierarchy, and there is only one way to define a concept. This moves the cost of multiple non-reusable searches to organising the KB at concept-creation time: a reusable result since this organisation not only cuts search costs, but also reduces the search space considered by classification of concepts later added to the KB [Baader et al. 94].

Since LOLITA 92 lacks any automatic means of classification, new concepts are simply added existing hand-crafted inheritance hierarchy. This means the information classification would find is not expressed in the inheritance hierarchy. As long as this information is not needed, determinism of search is unaffected, but recovering it requires a non-reusable expense of the same order as classification’s.

CGT’s hand-crafted type-hierarchy does not allow new type definitions to be added to it automatically. [Sowa 84] mentions type-contraction, a means to determine whether a statement implicitly includes a given type. But it does not explicitly mention an automatic means of maintaining the type hierarchy.

The definition extension mechanism used by CGT, QLF, SNePS and ANALOG does not support either classification or even introducing the concepts into the inheritance hierarchy. This means that the rich men in “All rich men own a car” will not be related to the men in “Every man loves a woman”. Determining such observational facts due to the implicit ancestors of a dynamically defined concept is expensive: for SNePS, ANALOG and CGT the class hierarchy must be searched up, and any entailment involving a concept of that type or class must be checked to see whether it applies to the dynamically defined concept. QLF has no inheritance hierarchy.
• Additional forms of definition

Beyond definition of concepts by genus and differentiae, CGT provides an additional means of definition: by prototype or schema. Rather than defining what the concept is and is not, prototypes state properties the concept usually has. However this advantage is offset by the lack of an automatic means of maintaining the concept type lattice.

SemNet provides one way of defining concepts (genus and differentiae), but is open to being expanded by another method based on Homogeneity theory [Garigliano 89]. This is left for a later date, since a means of maintaining the inheritance hierarchy when concepts are defined by Homogeneity Theory has yet to be determined. It also allows atomic concepts to be introduced via the linguistic nodes which are uniquely defined.

LOLITA 92 and KL-ONE provide only one way of defining concepts (genus and differentiae). So do SNePS and ANALOG (class definitions in terms of atomic sensory nodes).

• Cohesion

SemNet introduces two forms of definition: atomic language nodes that need no definition, and concepts defined by events and arcs. It is clear what arcs define a concept: the arcs connected to it (or inherited to it) with a definitional sort. Similarly the arcs that assert facts about a concept are those connected to it (or inherited to it) with an observational sort. However SemNet adds a new representational device to distinguish them: sorts.

SemNet also allows for a third form of definition to be added cohesively: The introduction of homogeneity theory is expected to be as cohesive as the introduction
of any other form of reasoning: the only change expected to semantic is the calculation of satisfiability. The semantic integration algorithm itself is not expected to change. A full application of homogeneity theory may require qualifying the spec. and inst. events that form the inheritance hierarchy by some density. This is no different from the requirement that time introduced: concepts that only exist for a certain range of time, or that are qualified by an event for that range of time, must be connected to the inheritance hierarchy by spec. and inst. events qualified by the time range during which they exist. This ensures that Jack, the fireman, could have had other jobs previously.

KL-ONE introduces two forms of definition: genus and differentiae, and atomic concepts. Atomic concepts are those that are not defined. It is clear what defines a concept: anything about it in the T-box. Similarly any information about it in its A-box are assertions about it. However, the use of two different representations to express definitions and assertions reduces cohesion. Cohesion is further reduced by the lack of any extension mechanism which could allow KL-ONE to provide other forms of definition.

LOLITA 92 introduces two forms of definition: genus and differentiae, and atomic concepts. LOLITA 92's cohesion is reduced by its ideosynchracies. LOLITA 92 uses an existing representation (set relations) to define or make assertions about concepts. This makes it unclear what is defined and what is asserted since the inheritance hierarchy must be traversed to determine whether an event is definitional or observational. Furthermore, additional rules apply: if the concept has a Named control, all events connected directly to it are observational by definition, since it is an atomic concept. Again, cohesion is lowered by the prototypical control added to distinguish observational events from prototypical events. Actions can neither be defined as atomic concepts nor as restrictions of other concepts. Indeed, actions are implemented in a spec. hierarchy, but are individually quantified (but not as named individuals). All these ideosynchracies require specific rules to be added to the inference machinery, for instance in the case of actions to allow events' ancestors to include the events connected to the ancestors of the event's action. The lack of classification further reduces cohesion, since two mechanisms are necessary
to determine all the facts a concept can derive from its ancestors.

CGT introduces four types of definition: genus and differentiae, prototypes, atomic concepts (those not defined by the previous two), and concepts using the implicit concept extension. It is clear what is defined: the definition of concepts's types, or the left-hand-side of the implicit concept extension. Any other statements about sets are asserted. Since the prototypes add richness, CGT will be scored as having 3 types of definition.

SNePS and ANALOG introduce three forms of definition: classes are defined by LEX arcs connected to sensory nodes; implicitly defined concepts are expressed in an entailment; and molecular nodes are defined by the nodes they dominate. Correspondingly there are three ways of making assertions about a concept: statements about members of classes; the consequents of entailments implicitly defining concepts; and nodes dominating molecular nodes. As discussed in 4.5.5.1 (p. 80), the last condition implies that to preserve richness, two representations would be needed for each type of event that in SemNet would apply to another event (time, belief, cause, etc.) This further reduces cohesion.

QLF neither assumes a type hierarchy nor defines its concepts, although it does allow implicit concepts to be defined.

<table>
<thead>
<tr>
<th>QLF</th>
<th>CGT</th>
<th>SNePS</th>
<th>ANALOG</th>
<th>KHONE</th>
<th>LOLITA 92</th>
<th>SemNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>6</td>
</tr>
</tbody>
</table>

8.3.2.2 Non-separative knowledge base

In the following section, CGT will be treated as a separative KB because [Sowa 84] argues that separating definitions from assertions is necessary. However, CGT’s definition extension is non-separative, allowing it to behave like the other non-separative KBs.

• Non-separative KBs are required by NLP

Unlike the other representations, SemNet and LOLITA 92 are non-separative KBs. Indeed, [Sowa 84] goes into great detail explaining the theoretic reasons why CGT
must keep types and sets separate: Sets are by definition extensional, so if the
intersection of two sets is said to be empty, all that means is that there happen
not to be any elements common to both sets. Keeping types separate, he argues,
ensures it is possible to state Cats ∩ Dogs is empty by definition. Despite his
argument, SemNet is able to represent both statements in a non-separative KB: by
stating that the definitional intersection of Cats and Dogs is observed to be empty,
one is stating that there is no animal which is both a cat and a dog by definition;
by stating that the definitional intersection of Cats and Dogs in the "real world"
frame of intension is observed to be empty, one is stating that no animal has been
observed to be both a cat and a dog.

Indeed, non-separative KBs are necessary for practical KBs: As the LILOG team
discovered [Beierle 92], separating definitional and observational statements proves
unnatural to natural language analysis, where new information comes and refines
previously introduced concepts. Such "changes of definition" have to be propaga-
gated back from the observational KB to the definitional KB, or the interpretation
function of the two KBs combined has to be changed to infer all the relations that
are not propagated back on the fly. This is the route taken by [Beierle 92], who
use a KB derived from KL-ONE. While this avoids semantic integration during the
natural language analysis, the equivalent operations must be done each time a fact
about a non-integrated concept is accessed. This is effectively the same as implicit
concept definitions.

The following sentences illustrate the need for a non-separative KB:
(1) "This lesson we will be learning about triangles. Triangles are three
sided polygons. The sum of their angles is 180°."

(1) introduces a new concept, and only later goes on to define it. This is problem-
atic for KL-ONE and CGT which need a concept to be defined once, rather than
allowing its definition to be refined over time. QLF requires no definition of any
of its concepts. SNePS, ANALOG, LOLITA 92 and SemNet are non-separative
KBs. While definition refinement over time may require truth-maintenance, this
difficulty stems from reasoning, not the representation.
(2) "John thinks an apple is a juicy red fruit, while Jack thinks it is a type of computer"

(2) illustrates how definitions can be talked about. This cannot be done in a separative KB where definitions are independent from assertions, since "John thinks" is an assertion.

<table>
<thead>
<tr>
<th>QLF</th>
<th>CGT</th>
<th>SNePS</th>
<th>ANALOG</th>
<th>KLONE</th>
<th>LOLITA 92</th>
<th>SemNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

• Locality of definition

Natural language introduces many new concepts in every sentence. However, these concepts are only of local interest, making sense within their context. Because context is lost once a new text is analysed, all the information required from the context must be added to the KB. This results in the KB’s representation of a text being a highly interconnected set of statements forming a locality. In a non-separative KB many of the facts derived from the context are shared between multiple assertions and definitions. In a separative KB the context must also be made explicit. However, many facts cannot be shared because definitions are independent one-from-another and from-each-other. This breaks up locality and requires facts to be repeated.

For instance, "John said that every one of Lolita’s husbands had a fast car". In a separative KB, such as KL-ONE or CGT, Lolita’s husbands and Lolita’s husbands’ cars must be defined in the definitional part of the KB. However, to represent John’s statement, the fact that Lolita’s husbands each own a car must be repeated. SNePS, ANALOG, QLF, LOLITA 92 and SemNet can represent this without repetition.

The number and granularity of the definitions introduced by NL appears unsuited to separative KBs: "The accident that took place at five o’clock involved John Bull" defines a new concept the accident that x knew about on day y, and which occurred on day w at five o’clock. This concept is specialised to one particular conversation, probably never to appear in other discussions. The effort of defining it independently from the locality of the rest of the conversation, and making it a concept that can be used in any context appears excessive. Indeed, given the number of
new concepts per sentence, the requirement that each be defined independently increases KB size dramatically. Furthermore, placing definitions in a separate KB adds an extra level of indirection: algorithms that need to use concepts' definitions must look them up in the other KB. This applies even to algorithms that only use truth conditions, and not definitions, such as NL generation. Similarly, NL analysis must work harder since a small difference in sentences results in a very different final representation: "The accident that took place at five o'clock involved John Bull" and "The accident involving John Bull occurred at five o'clock" will be separated between KBs differently.

SNePS, ANALOG, QLF and LOLITA 92 require non-local transformations on the KB to express sortal information. This requires all algorithms to know how sorts are represented. SemNet on the other hand uses the same structures to represent assertions and definitions: only the sortal information changes, and algorithms that do not need it can ignore it. Even algorithms such as inheritance which must ensure that information derived from definitional events is added as definitional to the KB need only pass sortal information to standard rules that will determine the correct sorts.

<table>
<thead>
<tr>
<th></th>
<th>QLF</th>
<th>CGT</th>
<th>SNePS</th>
<th>ANALOG</th>
<th>KLINE</th>
<th>LOLITA 92</th>
<th>SemNet</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

- **Mutually defined concepts**

Most of the representations only allow concepts to be defined in terms of other pre-defined concepts. However, certain concepts are mutually defining. For instance, "Every shepherd owns some sheep" defines the shepherd and the sheep that he owns (and even the events of owning those sheep).

SemNet and LOLITA 92 represent this with one event that is definitional with respect to both shepherd and sheep. This circular semantics should not be a problem given [Hill 94]'s model of equally circular SNePS.

---

6 This guarantees that any instances fitting the new definitional description will be correctly integrated and will thus inherit all the information available about them.
\[(E_0, R): \{\Delta Fv\text{-}subject\_FF: \text{shepherd}_5; \Delta v\text{-}action\_FF: \text{own}_3; \\
\Delta FF\text{-}object\_FF: \text{sheep}_7\}\]

Although SNePS' base nodes and molecular nodes are mutually defining, they do not define generic concepts that are used in multiple statements. Because they are statement specific, they are not instances of SemNet's meaning of mutually defining concepts. Similarly, SNePS' class concepts are defined by sensory nodes, not each other.

KL-ONE defines each of its concepts by the attributes (predefined concepts) it has, which does not allow for mutually defining concepts.

CGT defines its statically defined concepts as monadic abstractions \(\lambda a \ u\) where there is only one variable: \(a\). This prohibits mutually defining concepts.

QLF defines each of its concepts implicitly as a restriction on a variable. While this could allow the definition of an inner quantification to depend on an outer variable, since the representation is linear mutual definitions are impossible.

All the definition extensions of the form \(\forall x \in X'; f(x) \rightarrow g(x)\), define the implicit concept \(f(x)\) to have the observational characteristics \(g(x)\). Mutually defining concepts would have to imply each other to define each other. While this might be possible, it'd be difficult to deal with.

<table>
<thead>
<tr>
<th>FOPCL</th>
<th>QLF</th>
<th>CGT</th>
<th>SNePS</th>
<th>ANALOG</th>
<th>KLONE</th>
<th>LOLITA 92</th>
<th>SemNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- Granularity of definition

SemNet has a very fine granularity of definition: every one of SemNet's concepts is defined. This is how SemNet guarantees conceptual uniqueness: each concept has a definition, which can be verified to be unique. Semantic integration thus guarantees uniqueness. The other result is that every SemNet entity and event is in the inheritance hierarchy.

Like SemNet, each of LOLITA's concepts is defined, but are not unique because LOLITA has no semantic integration.
KL-ONE's T-box statements are all uniquely defined, as guaranteed by classification. However the A-box statements which make observational statements about the T-box concepts are neither guaranteed to be unique nor to have a definition. Only T-box concepts are in the type hierarchy.

Like KL-ONE, CGT's type statements are all defined, and should be uniquely defined by the knowledge enterer places them into their static type-lattice. However most of the knowledge takes the form of assertions involving no definitions.

SNePS and ANALOG do not define their classes except in terms of sensory nodes assumed to correspond to their agent's recognition of the class type. Like CGT and KL-ONE, most of their statements are not defined.

<table>
<thead>
<tr>
<th>FOPL</th>
<th>QLF</th>
<th>CGT</th>
<th>SNePS</th>
<th>ANALOG</th>
<th>KLONE</th>
<th>LOLITA 92</th>
<th>SemNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

- **Definition limited to not whole graph**

Many representations such as LOLITA 92 [Shiu 96] and SNePS [Hill 94] claim that the meaning of a concept is given by its position in the semantic net. However if this were the full story, any concept added to the KB would change the meaning of every concept – even if the concept added had been inferred by a sound reasoning rule. Practically, semantic integration would be impossible.

By using definitional and observational sorts, SemNet ensures that only definitional information changes the meaning of concepts. Even then, if a definitional fact is inferred from some others, the meaning of the concept is not considered to have changed: essentially the question is not whether two graphs are identical, but after everything that could be inferred from them by the reasoning algorithms has been explicitly added to them, whether they are identical.

In some sense the two positions are not so different: ultimately the meaning of the concepts will show up in the run-time behaviour of the system. The existence of inferred events might change a little certain system behaviours, as does having something fresh in your mind versus having to work it out again. However the major difference is that SemNet's definitions help ensure consistency via semantic
integration, type checking, and the like. These need clear finite definitions.

8.3.2.3 Inheritance hierarchy

An inheritance hierarchy reduces search-space and the size of KB. It also simplifies various algorithms, such as reasoning by analogy [Long et al. 93] and semantic distance [Short et al. 94a, Short et al. 94a] by structuring the KB.

SemNet, KL-ONE, CGT, SNePS, ANALOG, and LOLITA 92 all have some form of inheritance, class or type hierarchy. FOPL and QLF lack such a hierarchy.

- Every concept is in the Inheritance Hierarchy

SemNet is an integrated KB which performs semantic integration not only on a concept’s definitional events, but also on its observational events. This makes the concept into an observational descendent of the observational events' template events’ arguments. This means that any facts known to apply to the arguments of a particular type of event will be inherited to that event, even if the event qualifies a concept observationally. For instance, if it is known that all rich people own a car, the mere fact of stating “John is rich” will make him an observational descendent of rich people, and hence able to inherit “owns a car”.

In KL-ONE only T-box statements are classified, so assertions cannot be inherited in this way. LOLITA 92, CGT, SNePS, ANALOG, FOPL, and QLF do not perform semantic integration.

<table>
<thead>
<tr>
<th>FOPL</th>
<th>QLF</th>
<th>CGT</th>
<th>SNePS</th>
<th>ANALOG</th>
<th>KLOSE</th>
<th>LOLITA 92</th>
<th>SemNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

8.3.2.4 Qualifiable Inheritance Hierarchy

SemNet allows the events of its inheritance hierarchy to be qualified by other events. Any event inherited through a spec. or inst. event qualified by events Q, will be built qualified by (descendents of) the events of Q. This enables SemNet to properly integrate believed statements “John thinks Jack is a fireman”: Jack inherits events
qualified by a "John thinks" belief event, ensuring only John believes that Jack has the properties of a fireman, not LOLITA herself. Similarly, qualifying the inheritance hierarchy by time events ensures that concept instantiations only inherit the properties of the class to which they belonged while they belonged in it. This scheme is very natural since it applies to many phenomena (time, belief, stative events, frames of existence, homogeneity theory, certainty, etc.). It also improves extensibility and cohesion since all that inheritance need know is how to copy events qualifying the inheritance hierarchy down to the events being inferred. An event inherited through two specs with incompatible qualifying events will gain both, and then be rejected by semantic integration's satisfiability calculation. Since satisfiability is used here, there is no representation dependent reasoning complexity added to perform this operation.

LOLITA 92 had a similar scheme for stative events only: "John said Susan is a soldier": because LOLITA 92 uses arcs for the spec_ and inst_ relations, a new event is_a had to be introduced for the inheritance hierarchy to be qualifiable. Having two schemes proved problematic as algorithms only one or the other.

KL-ONE, SNePS, ANALOG, CGT do not have an equivalent. QLF and FOPC do not have an inheritance hierarchy.

<table>
<thead>
<tr>
<th>FOPC</th>
<th>QLF</th>
<th>CGT</th>
<th>SNePS</th>
<th>ANALOG</th>
<th>KLONE</th>
<th>LOLITA 92</th>
<th>SemNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

- Events as first class citizens

SemNet treats events like any other concept: not only can their own arcs define them or make statements about them, but so can other events. This enables the representation to be easily extended for instance by the time or frame of existence representation. Furthermore they are integrated into the inheritance hierarchy enabling statements about them to be inherited to their descendents. Thus statements such as "Designing software properly is hard" can be inherited to any instance of designing a particular piece of software.

\footnote{For optimisation, satisfiability calculation attempts to combine all the events of a given type (each type of values, time, etc) into a unique strong restriction.}
The molecular nodes of SNePS and ANALOG are defined by the nodes they dominate. They are not defined by other molecular nodes, and their arcs cannot make observational statements about them. As discussed in 4.5.5.1 (p. 80), one cannot distinguish between "The accident that took place at five o'clock involved John Bull" and "The accident involving John Bull occurred at five o'clock". The same applies to LOLITA 92 and to CGT.

KL-ONE has no notion of events.

<table>
<thead>
<tr>
<th></th>
<th>CGT</th>
<th>SNePS</th>
<th>ANALOG</th>
<th>KLINE</th>
<th>LOLITA 92</th>
<th>SemNet</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

○ Template events

SemNet events define the arguments they take: events that are defined only by an action arc are template events. Any events of that action type will be subsumed by the template event, and therefore will be integrated under it. Statements made about a template event apply to all events of its type – thanks to inheritance. Thus one can state that waking up takes a certain amount of time, or that walking is composed of many steps. Indeed, SemNet even allows a template event to define what arguments an event connected to it may take (see "to buy" 5.4 (p. 147)). Type-checking enforces these restrictions, possibly adding new information: "She owns the house" results in "She" being re-classed as a female human.

Events with LOLITA 92's prototypical control are prototypical events: their subject and object specify the arguments events of the action's type take. Like SemNet, type-checking enforces these restrictions possibly adding information. Similarly, statements about a prototypical event apply to its descendents, and special rules ensure that semantic integration integrates events of a prototypical event's action type below it.

Being a separative KB without higher order quantification, CGT cannot express facts about types of relations. It can however define relation types. Type expansion enables facts from a relation's definition to be added to a conceptual graph, but the resulting graph does not distinguish which information was definitional and
which was observational. Although relation types' definitions specify what types of arguments the relation takes, this specification is not enforced. Instead a separate canon of legal conceptual graphs is used: any graph of a CGT KB must be derivable from any combination of copying, restricting, joining, and simplifying graphs of the canon. This use of a different system to enforce selectional restrictions, could lead to graphs that do not follow a relation type's definition. [Sowa 84] gives no algorithm to determine whether a conceptual graph is canonical or not.

SNePS and ANALOG do not define their relations.

<table>
<thead>
<tr>
<th></th>
<th>CGT</th>
<th>SNePS</th>
<th>ANALOG</th>
<th>LOLITA 92</th>
<th>SemNet</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

- **Actions**

SemNet's actions can be defined. This improves determinism of search by grouping a certain combination of events under one action type. It also improves extensibility, since it enables a new conceptual level to be defined from an existing simple representation. For instance in the time representation, higher-level relationships such as after, before, is-in were built from a minimal initial set of operators. CGT is able to define relations in a similar manner.

SNePS, ANALOG and LOLITA 92 cannot define actions.

<table>
<thead>
<tr>
<th></th>
<th>CGT</th>
<th>SNePS</th>
<th>ANALOG</th>
<th>LOLITA 92</th>
<th>SemNet</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

### 8.3.3 Values

#### 8.3.3.1 Richness

\(FOPC\), CGT, KL-ONE, SNePS, ANALOG and LOLITA 92 do not support values. They are able to refer to values as constants, but not to reason with them or relate them.

QLF has a limited system of values based on three primitives. An expression of the form \( [\text{value type}, \text{(concept), (amount)}] \) associates the concept with the
numerical amount of the particular value type, as in [tall_degree, john, 176]. Since no unit is specified, each value type has an implicit unit type, requiring multiple value types for measurements expressed in feet, yards, light years, metres, etc. Expressions of the form [more, (Predicate), (Item1), (Item2), (Degree)] are used to express comparatives, where the Degree is the difference between the value associated with Item1 and Item2. This allows comparatives to be represented. Finally expressions of the form [order, (Item1), (Item2), (value type), (nth)] allow superlatives of the form “The third oldest building”. Because the two means of relating values actually relate the items having the relevant value, values in QLF are very restricted: one cannot say “My sister is three times more beautiful than yours!”.

SemNet has a fully developed representation of values which can represent unquantifiable values, values with no natural unit, values with no agreed zero point and quantifiable values in a very similar manner: phenomena as diverse as durations, cardinality, degree of anger, belief and certainty, temperature, quantity of money (etc) are all represented with the same three operators. Thus sentences such as “My sister is three times more beautiful than yours!” and “Bill Gates is ten times richer than his closest competitor” are represented by the same external multiplication operator. Values are concepts in their own right, allowing expression such as “Don’t you know what the temperature was yesterday?” to be represented. Strict-typing ensures that different units are not confused. Values are also used in the representation of natural language expressions such as comparatives “John is taller than Matthew”, superlatives “Bill Gates is the richest man”, relative adjectives “John is tall”, subjective adjectives “What a beautiful ship!” and adverbs.

<table>
<thead>
<tr>
<th>FOPFL</th>
<th>QLF</th>
<th>CGT</th>
<th>SNePS</th>
<th>ANALOG</th>
<th>KLONE</th>
<th>LOLITA 92</th>
<th>SemNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

8.3.3.2 Reasoning

The great degree of similarity between all forms of values allows the same reasoning algorithms to be applied to all. Unquantified variables behave little differently
from quantified variables whose value is not known. This means that many known techniques for reasoning about variables and simultaneous equations (including inequalities) can be applied to a generalised framework. For instance, much research has been invested into the constraint satisfaction problem. In this work, graphs are often used to express the equalities considered, since the graph formalism captures the structure of the problem and often complex manipulation of equations proves easier in this form. The work by [Dechter et al. 91] is an example. Similarly, MAPLE, a well known symbolic mathematics tool [Char et al. 91] uses graphs to express problems it must solve.

A single representation of all types of values allows reasoning algorithms devised for them to be used for any phenomenon that is represented using them. This means that any effort devoted to building these algorithms will improve the system as a whole, and that any new phenomenon added as a value has a wealth of readily available reasoning tools immediately at its disposal. The same tools can be used in the calculation of interest rates, in determining the cardinality of a set given constraining information, in reasoning about the certainty values of statements John believes in, and even solving riddles involving unquantifiable values, such as how nice people are.

8.3.3.3 Cardinality

\( \text{FOPC} \) and ANALOG have no notion of cardinality.

KL-ONE has no notion of cardinality, although it allows the number of concepts satisfying a role to be specified.

SNePS has no notion of cardinality although it allows the number of true propositions connected to a thresh or and-or logical connective to be specified within a given range.

CGT’s notion of cardinality is expressed as part of set nodes’ labels. Thus it is not a concept within the CGT graph, making statements such as "John does not know how many apples he picked today" impossible to express. No relations between
values can be represented "I have more apples than you".

QLF uses generalised quantifiers to express cardinality, a scheme separate from its values system, making it impossible for it to represent "John has the third largest number of apples". Equality and inequality Relations between values can be represented, as can statements treating cardinality as a concept in its own right (theoretically – the author has not seen any such examples). However statements involving arithmetic are beyond QLF: "John has five times as many apples as I do".

LOLITA 92 allows sets to be attached to a integer value or a term such as some, most, all, etc by a size arc. However this cardinality is not a concept in its own right as is demonstrated by the use of a how many status to represent sentences such as "How many apples did John pick?" Similarly, the cardinality is not based on values, requiring the analogy algorithm to use size restriction arcs such as large restriction, medium restriction and small restriction [Long et al. 93].

SemNet's cardinality representation is based on values, making cardinality into first-class concepts, which can be related one to another by any arithmetic operation. Moreover size can be applied to events, unlike any other representation, introducing a consistent system of negation which recognises that if "Every dragon has wings" and there are no dragons, then there are no events of dragons having wings. This symmetry is particularly relevant to SemNet which allows events to be treated as subjects or objects to other events "Three loud explosions woke the city of Londonderry this morning", following their use in Natural Language.

<table>
<thead>
<tr>
<th>FOPL</th>
<th>QLF</th>
<th>CGT</th>
<th>SNePS</th>
<th>ANALOG</th>
<th>KLONE</th>
<th>LOLITA 92</th>
<th>SemNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

8.3.3.4 Negation

ANALOG has no notion of negation. KL-ONE has no negation, but can refer to the complement of a concept.

FOPL, QLF and CGT use standard FOPL negation. CGT's use of a negation
context to achieve this reduces its efficiency (footnote 8 (p. 352)).

LOLITA 92 represents negation by replacing the action. arcs of its events by no_action. arcs. This requires the negated event to be duplicated if both positive and negative versions are wanted: "Jane believes that John didn't hit Henry, but he did" requires both a "John hit Henry" and a "John did not hit Henry" event.

SNePS uses the and-or operator with $n = m = 0$ to express the fact a molecular node is not true. The separation of the molecular node from the and-or operator allows statements such as "Jane believes that John didn't hit Henry, but he did" to be represented with only one molecular node.

SemNet can represent a negation by antonym which converts statements such as "A man likes a fish" to their opposite such as "All men dislike all fish".

Unlike other representations, SemNet can also represent negation by absence of occurrence which states the event described never occurred. It is cohesive since it states the negated events has no instantiations. This allows "Jane believes that John didn't hit Henry, but he did" only to require one "John hit Henry" event. This fits in well with semantic integration since any event that is integrated as a descendant of a negated event will be deduced to be of size. 0. For instance, if Mary believes that John didn't hit Henry she must also believe that John didn't punch him or smack him. It also allows a more symmetrical treatment of statements such as "John thinks there are five apples in the basket, but Jane thinks there are none": each believes in a different size. event connected to apples_in_basket, rather than one believing in the basket containing apples and the other believing in the basket not containing apples. Finally, SemNet's treatment of absence of occurrence simplifies the treatment of quantification: "No penguins fly" and "Penguins do not fly" are equivalent because whether the size. 0 is placed on the subject or the event, the subject.'s $F - F$ quantification ensures that the event or the subject (respectively) are inferred to also be of size. 0:

$$(E_0,R): \{\Delta F{\text{-subject.}} - F{\text{-O}: \text{Penguins; } \Delta V{\text{-action.}} - I{\text{-O}: \text{fly;}}}$$
8.3.4 Belief

8.3.4.1 Logical connectives

KL-ONE has no logical connectives. Instead it uses roles, role intersection, concept intersection and union to define its concepts.

QLF and $\text{FOPL}$ use the standard $\text{FOPL}$ logical connectives.

It is standard in semantic nets, for all (believed) propositions qualifying a given node to be true without the need for an explicit and connective. Thus the and connective is only used if it need be referred to.

CGT essentially uses first order logic. It improves uniqueness by using only two primitives: the PROPOSITION concept containing conjoined statements, and the NEG monadic relation providing negation. Because PROPOSITION may affect non-linearity or distributedness\(^8\), efficiency is reduced.

ANALOG’s logical connectives are “and-entailment” and “or-entailment”, which allow statements of the forms $(m_1 \land m_2 \land ...) \Rightarrow (m_9, m_8, ...)$ and $(m_1 \lor m_2 \lor ...) \Rightarrow (m_9, m_8, ...)$. They do not allow complex concepts to be built such as “Either Mary loves him or she loves me” to be built.

SNePS adds to ANALOG’s logical connectives “and-or” and “thresh”, whose truth depends on the number $i$ of their arguments which are true: if $n \leq i \leq m$ then and-or is true, and if $i < n \lor m < i$ then thresh is true, where $n$ and $m$ are specified on the logical connectives. The and-or subsumes the and case ($n = m$, the number of arguments), the or case ($n = 1, m$ is the number of arguments) and the exclusive or case ($n = m = 1$). Although thresh should increase richness, the author has never seen it used.

LOLITA 92’s four logical connectives (and, or, e.or, implication) are belief based. They apply only to individual events, being undefined for sets of events, and may
each only take a pair of arcs. There is no separate negation operator. Instead the action arc of the negated event is replaced by a no.action event.

SemNet extends LOLITA 92 by giving meaning to logical connectives connected to sets of events, and to multiple arcs connected to a logical connective. Like LOLITA 92 there is no negation operator, but either the antonym action or a size_zero attached to the event.

<table>
<thead>
<tr>
<th>FOPL</th>
<th>QLF</th>
<th>CGT</th>
<th>SNePS</th>
<th>ANALOG</th>
<th>KLONE</th>
<th>LOLITA 92</th>
<th>SemNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

8.3.4.2 Stative events

KL-ONE, QLF and FOPL have no representation of belief or statements.

CGT represents statements and beliefs as propositions, which are nodes representing statements which correspond in some undefined manner to CGT graphs. This affects not richness but efficiency. Truth and belief are separate in CGT with belief only applying to external agents while truth is the belief of the KB’s agent.

LOLITA 92, SNePS and SemNet are propositional networks: Propositions can be qualified by other propositions. All three use this feature to represent stative and belief events.

LOLITA 92 uses a status arc to state whether or not an event is believed in by LOLITA or whether it is a constituent of a stative or belief statement. Since LOLITA 92 defaults the event to believed if the arc is absent, distributedness is reduced. LOLITA can represent statives, belief events and one form of cause. LOLITA uses a separate representation for source control to state the source of information in her KB, reducing cohesion.

ANALOG and SNePS 2.0 mark events which are believed as asserted (on the node), preserving its distributedness. However they cannot represent forms of cause or

---

If these graphs are somehow encapsulated in the node, as constants are in CGT, the representation suffers from low non-linearity. Otherwise, there is no marker to state that the believed facts are not believed by the CGT agent, hence distributedness is lowered since every fact must be checked to see whether it is believed by some other agent.
statives such as to simulate or to say.

SemNet uses the belief control (hypothesis or real) to state whether events are believed of not in a distributed manner. It also allows stative and belief events to be chained together, thereby allowing such statements as "John said Jack believes Peter plays Shakespeare." While ANALOG, SNePS and LOLITA 92 could allow this chaining, no such use has been demonstrated. SemNet's spec. and inst. events can be qualified by a stative event and be hypothetical: "John thinks she was a firewoman." SemNet also distinguishes between necessary and sufficient causes, "The gun must be loaded to fire" and maintains cohesion by using the same representation to express information source for source control and standard statives.

<table>
<thead>
<tr>
<th>FOPL</th>
<th>QLF</th>
<th>CGT</th>
<th>SNePS</th>
<th>ANALOG</th>
<th>KLONE</th>
<th>LOLITA 92</th>
<th>SemNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

8.3.4.3 Belief and certainty

SNePS, ANALOG, KL-ONE, CGT, LOLITA 92, QLF and FOPL have no representation of gradation of certainty or belief, reducing their richness: "I think John didn't leave his briefcase in your room, but I'm sure that he had it with him on the train" cannot be expressed. SemNet does.

<table>
<thead>
<tr>
<th>FOPL</th>
<th>QLF</th>
<th>CGT</th>
<th>SNePS</th>
<th>ANALOG</th>
<th>KLONE</th>
<th>LOLITA 92</th>
<th>SemNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

8.3.5 Ambiguity

CGT, SNePS, ANALOG, KL-ONE, FOPL and LOLITA 92 have no representation of ambiguity.

While QLF does pack its ambiguities at the syntactic and semantic-mapping levels, when it comes to resolving lexical ambiguities it unpacks the QLFs and treats each of the resulting logical forms (LFs) individually. This is because "it turned out more efficient, on average, to apply sorts to QLFs after extracting them from
the packed structure. This was because as sort expressions became larger and more refined the cost of maintaining sorted packed semantic records increased, especially since differences between word sense sorts often prevented the packing of local alternatives” [Alshawi 92] (page 145). Indeed, CLARE’s approach is to have fewer meanings per word than LOLITA’s (Cooper et al.95): CLARE’s core lexicon has 2500 entries, versus LOLITA’s 100,000 nodes).

SemNet’s approach to ambiguity, in particular allowing underspecified concepts, defined only by the features the alternatives share, makes reasoning with ambiguous concepts natural. Its use of belief improves cohesion and uniqueness, allowing standard belief reasoning to be applied to disambiguation: analyses which are assigned low certainty or belief due to contradictions with knowledge in the KB are dispreferred.

<table>
<thead>
<tr>
<th>$\mathcal{FOP!L}$</th>
<th>QLF</th>
<th>CGT</th>
<th>SNePS</th>
<th>ANALOG</th>
<th>KLONE</th>
<th>LOLITA 92</th>
<th>SemNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

8.3.6 Time

8.3.6.1 Richness versus Allen’s Theory

Allen’s Interval Logic [Allen et al.85] is the most well known theory of Time in AI. Because KL-ONE, CGT, ANALOG, LOLITA 92, $\mathcal{FOP\!L}$ and CLE-3 lack a theory of time, SemNet’s time representation will be compared to Allen’s.

- Time Points

Allen’s theory of time does not support points of time. While it admits that time intervals can be thought of as points (and are thus point like), the theory itself leaves no space for points, as every interval contains another:

$$\text{HOLDS}(p, T) \leftrightarrow (\forall t)[\text{IN}(t, T) \Rightarrow (\exists s)[\text{IN}(s, T) \& \text{HOLDS}(p, s)]]$$

As [Galton 90] points out, this creates problems when one wishes to speak about continuously varying quantities. For instance, if a ball is thrown into the air, there is a instant of time when it is stationary at the top of its trajectory. Indeed, any object in continuous movement along a given trajectory is only instantaneously at
a given place on that trajectory. For instance, Allen's robot which must pile up blocks with an arm would be incapable of picking a block off a turn-table, since it uses Allen's theory of time. In general the lack of time points makes it impossible to express the state of a continuous changing quantity at a given time – something essential in what could be called an analog world.

Philosophically Allen argued that in physics, no limit has been found to the divisibility of intervals [Allen 83]. However, a more practical reason for abolishing time-points completely is that Allen could not answer the question "Are the endpoints of an interval in the interval or not?" Take a race. There are two intervals: the race (RR) and after the race (AR). Running holds during interval RR, and not running holds during interval AR. If endpoints are included in the intervals, then running and not running both hold at the end-point e ending RR and starting AR. If they are not, then neither running nor not-running holds at e. Allen dismisses intervals open at one end as artificial.

Simply because it is unknown whether the endpoints of intervals should be included does not mean that there is no place for time-points in the representation. Indeed, the question may be unanswered because there is no answer that applies to all cases. Galton solves this by introducing different types of properties (states of position and states of motion), and different reasoning rules for each. However the problem is rarely of any concern to real world applications, so does not need to be solved. Instead SemNet just states that it is unknown whether an event applies during the endpoints of an interval unless it is explicitly stated.

• Durations

Allen's theory of time does not natively include reference to durations. Instead "a logic for reasoning about durations, which is separate from, but integrates nicely with, the basic interval logic" [Allen et al.85] is used.

SemNet's representation of duration is integral to the rest of the representation. Since durations are expressed as values they can be within certain ranges, precise, stated in terms of multiple units, etc. Allen cannot represent durations which are known to lie within one of many ranges of time, whereas SemNet can with standard
logical connectives.

In both cases, interval relations can constrain duration information. For instance in SemNet, if $a$, $b$, $c$ and $d$ are all intervals, \( \text{starts} \_ (a,b) \land \text{ends} \_ (b,c) \land \text{ends} \_ (a,d) \land \text{starts} \_ (c,d) \) states that the sum of the durations of $a$ and $c$ is equal to the sum of the durations of $b$ and $d$. The same example can be built in Allen's representation. In SemNet these relations can easily be translated into values relations between durations so that values reasoning can solve the resulting set of simultaneous equations.

[Allen et al.85] shows how duration information can constrain its interval relations. This is not true for SemNet since it only records known relations: either the duration information contradicts the interval relations or it does not.

8.3.6.2 Efficiency versus Allen's Theory

- Allen

Allen's theory of time uses disjunctions of up to 13 time operators to express the relations between two intervals. Essentially, it states all the possible relations between the two intervals, those not stated being known not to hold. Intervals are represented as nodes of a directed graph, the arcs of which are labelled by the disjunction of relations. This means there can be up to $2^{13}$ possible labels. A transitivity relation expresses how to combine single relations of the arcs from intervals $a$ to $b$ and $b$ to $c$. The result is a disjunction of between one and four relations. Since each arc can carry more than one relation, the cost of transitivity is $m \cdot n$ where $m$ is the number of relations on one arc, and $n$ that on the other. An incremental constraint propagation algorithm traverses the graph and builds the relations between every pair of nodes of the graph. Clearly, this proves inefficient, so reference intervals[Allen 81] are introduced which group intervals together. Relations are only built between intervals of a reference interval, and it is the user who determines which intervals to group together.

The use of disjunctions renders the problem of determining consistency of the
graph NP-hard [Vilain et al.89]. Similarly it renders the problem of determining a consistent scenario given by the graph NP-hard. [Drakengren et al.91] The problem is that every one of the 13 operators makes a statement about both the endpoints of the intervals. Therefore, if information is known only about one of the endpoints, disjunctions must be used. Similarly, the only relations allowed between endpoints are $<$, $>$ and $=$, meaning that 4 of Allen's relations ($=$ s d f) must be considered to state X is entirely within Y (expressed as X ($=$ s d f) Y). This NP-hardness prohibits the semantic integration of Allen's intervals, a key tool needed to organise any large KB.

- SemNet

In comparison, SemNet only expresses information known to be true. Since each piece of information is true, there are not multiple reasoning paths (caused by disjunctions) to follow to determine whether a given graph is consistent, or to find a scenario consistent with it. Indeed, SemNet can be mapped to [Dechter et al. 91]'s Simple Temporal Problems. Only when a duration has more than one possible values (expressed with the logical or event), does the problem become a general temporal constraint satisfaction problem (TCSP). According to [Knight et al. 93], [Dechter et al. 91] is the only major attempt at reasoning about time that addresses the problem of interval duration.

[Dechter et al. 91] uses a constraint network for temporal reasoning and consistency checking. In these networks, the nodes represent time points, and the arcs represent ranges of the duration of the intervals between the time points. This forms a constraint satisfaction problem. Reasoning is determining the times that the time points could take. The implicit origin is the earliest time point. Consistency checking ensures that the constraints do not prevent the time points from being assigned any values.

Certain classes of this problem can be solved cheaply. One such case is the Simple Temporal Problem (STP), when each arc is only assigned one range, which can be solved in $O(n^3)$ steps where $n$ is the number of time points in the graph. If an arc is assigned more than one range, it means that two time points may be separated by
a time within only one of the ranges. This might occur if one knew that something happened either between 8am and 10am or between 1pm and 3pm. This general problem is NP hard, although [Dechter et al. 91] provides indications on how this can be addressed.

The network used by [Dechter et al. 91], differs to the one formed by SemNet’s \text{starts}_, \text{ends}_, and \text{has\_duration} events:

- In SemNet, intervals are represented by nodes qualified by \text{has\_duration} events. The associated duration is either a single value, or can be associated with a range of values to which it belongs. In [Dechter et al. 91], arcs are used to represent the intervals between time points.

- In [Dechter et al. 91], nodes represent time points, whereas in SemNet there is no such notion. The closest is the \text{starts} and \text{ends} events which specify that a set of times share a common start or end point.

However deep these differences may appear, it is in fact possible to find a mapping between [Dechter et al. 91]'s graph, and SemNet's.

If one forgets for an instant what the events in SemNet mean, and simply looks at the topology of the two graphs, one can see a parallel between the time nodes of SemNet and the labelled arcs of [Dechter et al. 91]. If all SemNet times that share a common start point are connected to the same \text{starts} event, and similarly for the \text{ends} event, a parallel can also be seen between SemNet's \text{starts} and \text{ends} events and [Dechter et al. 91]'s time points. Expressing these parallels explicitly, an implementation of [Dechter et al. 91]'s STP reasoning system could consider the \text{starts} and \text{ends} events as representing time points, and the time nodes as representing arcs labelled by their durations. The solution it would derive would be possible values for the start points of those intervals connected to a \text{starts} event, and for the end points of those intervals connected to a \text{ends} event. Consistency checking would also ensure that the graph does not express a contradiction, that there exists some set of values that can be assigned to the time points, and which satisfy the constraints on them.
Chapter 8: Evaluation

There might appear to be a failure in this scheme, if one of the times of a \texttt{starts} or \texttt{ends} event is an instant. In this case, there is a time representing one of [Dechter et al. 91]'s time points. But if the instant is connected to both a \texttt{starts} and an \texttt{ends} event, it will be read as an arc labelled with nil duration by the STP algorithm. This will then correctly conclude that the time points represented by the \texttt{starts} and \texttt{ends} events are equal. Thus, the scheme also works in this case.

This possible mapping to a graph representation for which reasoning has been derived and a reasoner implemented, shows that SemNet's representation of time can be used for reasoning, and in the usual case (STP), reasoning efficiently.

\subsection{Cohesion}

Allen's representation is not as cohesive and SemNet since it uses 13 relations where only one is needed [Allen et al.85, Allen et al.89]. Indeed, SemNet itself could be more cohesive as the only relation that is needed is a transitive \texttt{meets} relation which states that the \texttt{subject} interval's ending-point is the same as the \texttt{object} interval's starting-point. This will be addressed in the next revision of SemNet. By allowing more complex events such as \texttt{before} to be defined and their equivalent representation in terms of \texttt{starts} and \texttt{ends} events to be inherited, SemNet guarantees uniqueness while allowing algorithms such as Natural Language Processing to enter information at a lower level of granularity.

SemNet's representation of time is also more cohesive as it sits into the general framework of SemNet, using values, logical events, and the like. Both are used for the representation of other intervals: textrefs for SemNet, and DNA strands for Allen [van Beek et al.96].

\subsection{Textrefs}

KL-ONE, CGT, SNePS, ANALOG, LOLITA 92 and \texttt{FOPL} cannot associate text with its interpretation.

QLF maintains a syntactic record of the text through its analysis, but discards it
when sortal restrictions processing reached.

SemNet can represent the structure of a text and associate it with its interpretation using textrefs. Textrefs use the same representation as time, segments of text being a similar phenomenon to intervals of time. Similarly, Allen's interval logic has been applied to DNA strands to determine whether they are linear [Golumbic et al. 93]. Textrefs allow such statements as "John thinks that 'Blessed are the poor in spirit: for theirs is the kingdom of heaven' means that mentally handicapped people will go to heaven after they die. But Jack thinks that it means that people who are not attached to objects or people are in a state of mental peace and contentment".

<table>
<thead>
<tr>
<th>FOPL</th>
<th>QLF</th>
<th>CGT</th>
<th>SNEPS</th>
<th>ANALOG</th>
<th>KLONG</th>
<th>LOLITA 92</th>
<th>SemNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

8.3.7 Parts

FOPL, KL-ONE, ANALOG, SNEPS, and QLF do not have a representation of parts.

CGT represents parts with an arc relating an object's part to it. Thus "A brick is part of an arch" ([Sowa 84] p 71) can be represented. However, there is no way of expressing multiple systems of division "Europe consists of 12 countries. Europe consists of lakes, rivers, mountains, plains and seas." or of expressing a complete system of division "A floppy-disk has two parts: a casing and a magnetic disk" (and not more).

LOLITA 92's representation of parts is limited to a has_part event which states the subject has as parts the object_s. Like CGT the resulting event does not represent a system of division.

[Lenat et al. 90] discusses parts. It is able to represent multiple systems of division. It is unclear whether it can express a complete system of division. It discusses the difference between intrinsic and extrinsic properties as being the difference between properties that are inherited to parts and those that are not, although no representation is shown. Similarly, it discusses the notion of a granularity beyond
which the notion of substance collapses, but no representation is given.

SemNet is able to represent all the cases discussed above. It represents the inheritance relation of intrinsic properties explicitly in the network, requiring no additional representational machinery. It is also able to represent the parts of an event, and relations between the event and its parts: the relation between the part-events’ pre- and post-conditions and the pre- and post-conditions of the whole event D.3.3.2 (p. D-48); and the relation between the time of an event and the time of the events’ parts. Similar relationships for entities can be expressed such as for size or weight. Finally it can represent the minimal granularity beyond which substance-like behaviour breaks down.

<table>
<thead>
<tr>
<th>( \mathcal{FOL} )</th>
<th>QLF</th>
<th>CGT</th>
<th>SNePS</th>
<th>ANALOG</th>
<th>KLONE</th>
<th>LOLITA 92</th>
<th>CYC</th>
<th>SemNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

### 8.3.8 Language

\( \mathcal{FOL} \), KL-ONE and LOLITA 92 make virtually no provisions for representing any relationship between concepts and the words used to refer to them in natural languages. The only concession is that a concept’s identifier can be an English word. However this does not allow for ambiguous words, or multiple languages.

SNePS and ANALOG associate concepts with their sensory nodes using the \texttt{Lex} arc. Sensory nodes are assumed to be associated with a representation of the agent’s perception of that concept. This representation is expressed outside of the network. In practice, the \texttt{Lex} is bound to a node with the English word for the concept. This allows concepts to be associated with single words. Thus SNePS and ANALOG cannot directly map concepts to multiple languages.

CGT is able to represent the name of concepts using the dyadic relation (\texttt{NAME}). This enables a concept to be associated with multiple names, for instance the concept four with the French “\textit{quatre}” and the Roman numeral “\textit{IV}” (Sowa 84 p 88). However the (\texttt{NAME}) predicate does not state of which language the word is,
or qualifying the relation by a belief. [Sowa 84] also does not discuss ambiguous words.

QLF's sense entries map words satisfying semantic and syntactic constraints to QLF formulas. They are therefore more than a simple association between a concept and the corresponding word but a single purpose tool. They do provide mapping from multiple languages to concepts, and can implicitly express their language by additional constraints. They provide no means of discussing the relationship between the words and the concepts at the conceptual level, so cannot represent "Some people call them freedom fighters but others call them terrorists".

SemNet goes far beyond the other representations. Multiple words can be associated with the same concept (synonyms), words from each language are distinguished (in english, in french, etc. events) and can be discussed ("John thinks "man" is "homme" in French"), the frequency a word is used to express a particular meaning can be expressed, concepts such as "the meaning of phenomenology" can be discussed, the root form of words (men is the irregular plural of man) can be represented or discussed ("Jack thinks "sheeps" is the plural of "sheep""), and statements such as "Some people call them freedom fighters but others call them terrorists" can be made. Because grammatical features are expressed on the linguistic nodes, SemNet can represent the fact that "Der Mond" is masculine, "la lune" is feminine, and "the moon" is neuter.

<table>
<thead>
<tr>
<th>FOPL</th>
<th>QLF</th>
<th>CGT</th>
<th>SNePS</th>
<th>ANALOG</th>
<th>KLONE</th>
<th>LOLITA 92</th>
<th>SemNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
</tbody>
</table>

8.3.9 Intensionality and Extensionality

FOPL and QLF do not define their concepts so have no notion of intensionality and extensionality.

While LOLITA 92 does define its concepts, equality is based on set theory: two concepts are assumed equal if they have the same instantiations. Therefore it is an extensional-only representation.
SNePS and ANALOG are purely intensional representations and cannot express any extensional facts. "However, if SNePS is used just to represent a mind — that is, a mind’s model of the world — then it does not need to represent any extensional objects." [Shapiro et al. 87].

KL-ONE defines concepts intensionally. Intensional facts are expressed in the A-box, while extensional ones are expressed in the T-Box. There may or may not be equality at the T-Box level, depending on the representation.

Similarly, CGT defines its concepts as a type lattice which forms CGT’s intensional level. CGT’s extensional level are the statements about the sets in the set hierarchy. By keeping the types separate from the concept’s set hierarchy [Sowa 84] can distinguish between sets that happen to be empty extensionally and types that are empty intensionally. However CGT cannot express equality between two extensional concepts since “No concept may belong to more than one line of identity” ([Sowa 84] p 142) where lines of identity are undirected graphs built of co-reference links, the CGT equivalent to equality.

SemNet’s representation is essentially intensional: concepts are defined by their properties and equality is equality of definition. However, SemNet also allows extensional equality to be expressed via observational synonym events. Intensionality is essential to represent differences of view, such as an astronomer’s and an astrologer’s conceptions of the planet Venus.

<table>
<thead>
<tr>
<th></th>
<th>FOPL</th>
<th>QLF</th>
<th>CGT</th>
<th>SNePS</th>
<th>ANALOG</th>
<th>KLONE</th>
<th>LOLITA 92</th>
<th>SemNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

### 8.3.10 Frames of Existence

FOPL, QLF, CGT, SNePS, ANALOG, KL-ONE and LOLITA 92 have no concept of frames of existence.

SNePS claims that because it is intensional, it can represent non-existent concepts such as the golden mountain. However, SNePS’ agent cannot represent (and thus reason about) the fact it does not believe some concepts have instantiations in the
real world.

SemNet has a representation of frames of existence and can therefore represent fictional characters such as Sherlock Holmes, the not-necessarily existing hammer of "I need a hammer", the non-existent "carnivorous cow", the dream world of "I dreamt John killed my wife last night", etc.

<table>
<thead>
<tr>
<th>FOPL</th>
<th>QLF</th>
<th>CGT</th>
<th>SNePS</th>
<th>ANALOG</th>
<th>KLONE</th>
<th>LOLITA 92</th>
<th>SemNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

### 8.4 Essential Features

Section 3.5 (p. 42) states that for SemNet to even attempt its task, it must satisfy the requirements of naturalness, richness, usability for NL, and language independence. This section presents SemNet's evaluation for these aspects.

#### 8.4.1 Naturalness

Naturalness was divided into two components by 3.2.1 (p. 29): the quality of the partitioning, and therefore of the concepts in the K.B., and the structure of the K.B. itself.

The quality of a partitioning is dependent on what it will be used for. SemNet is to be used for N.L. analysis, which means the expression of N.L. concepts must be natural and the representation must help rather than hinder the process of interpretation. SemNet satisfies the first requirement as it does not force N.L. concepts to be expressed in terms of some limited set of meaning primitives. Indeed, LOLITA differs from other systems in that there may be many concepts per word, and every concept need not correspond to a word in every language. Instead, as many concepts as is useful can be represented. Although this increase in the number of concepts adds another difficulty to the task of N.L. analysis, disambiguation, it also helps in that precise concepts are better described. This helps the reasoning processes which must integrate the text's information into the rest of LOLITA's
knowledge, and later reason with it, for it to be useful. Indeed, SemNet has shown itself natural to many reasoning processes, as witnesses the wide variety of reasoning algorithms developed for it described in 8.4.5 (p. 369). This can attributed to the natural structure of the K.B., discussed below, and to the choice of structural primitives which maintain the independence of reasoning processes by maintaining a separate representational unit for each process: quantification, sorts, hypothetical control, arcs and nodes. Thus the independent reasoning processes can be combined in many different ways leading to great flexibility.

The structure of the knowledge SemNet was to capture is discussed in 2.10.2 (p. 25). First of all it was to be "a cyclic structure": SemNet is indeed such a structure, it is a graph. Secondly it was to be "ultimately reducible to properties operating on sensory data. Thus these properties correspond to the ultimate primitives (...) The notion to remember is one of cyclicity underpinned by some primitives defined externally from the knowledge base.". This corresponds directly to concepts being constrained by events, themselves eventually constrained by their actions and those of the events constraining them. The properties operating on sensory data correspond to the action nodes of SemNet which are not defined by any arc. Finally, "it is the organisation into a super-structure of mutually related concepts at many levels that gives the reasoning power considered in A.2 (p. A-7)." This corresponds directly to the inheritance hierarchy into which every concept of SemNet is integrated: different concepts at different levels of granularity participate in relations appropriate for their level of granularity with other concepts.

8.4.2 Richness

Although SemNet uses very few structural primitives, reflecting a high degree of cohesion, this in no way undermines richness. Indeed, as table 8.1 (p. 366) and the survey in 8.3 (p. 310) show, SemNet has more available meaning primitives than other representations making it very rich. Furthermore, SemNet is open ended, allowing the representation to be extended not only by the addition of new

---

9 Only useful relations are remembered, see 2.7 (p. 17)
<table>
<thead>
<tr>
<th>Feature</th>
<th>FOPC</th>
<th>QLF</th>
<th>CGT</th>
<th>SNePS</th>
<th>KLONE</th>
<th>SemNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quantification</td>
<td>N-N</td>
<td>Y-N</td>
<td>N-Y</td>
<td>N-N</td>
<td></td>
<td>Y-Y</td>
</tr>
<tr>
<td>Sorts</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Set Hierarchy</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Cardinality</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>(\frac{1}{2})</td>
<td>Y</td>
</tr>
<tr>
<td>Antonyms &amp; Synonyms</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Values</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Belief</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Logical Events</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Communication</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Causality</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Negation</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Parts</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Time</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Linguistic Nodes</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Intension</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Frames of existence</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Textrefs</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Ambiguity</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Location (partially)</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2(^{14})</td>
</tr>
<tr>
<td>Tenses</td>
<td>N</td>
<td>D</td>
<td>D</td>
<td>I</td>
<td>N</td>
<td>I</td>
</tr>
<tr>
<td>Scripts</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N(^{16})</td>
</tr>
</tbody>
</table>

Table 8.1: Representations' Richnesses

reasoning methods, but also by allowing existing reasoning methods to be used for new phenomena while keeping information separate by strict typing. For instance, the values reasoning methods may be applied to school class marks.

### 8.4.3 Language Independence

\(^{10}\)x→y: x stands for higher order quantification, and y stands for multi-levelled quantification.

\(^{11}\)KL-ONE only has crude quantification.

\(^{12}\)FOPC, CGT, and SNePS do not have sorts in that they do not define explicit concepts but rather conditions/rules: \(\forall x \in X f(x) \rightarrow g(x)\)

\(^{13}\)Not independent Type Lattices

\(^{14}\)Locative prepositions of other languages have been modelled in terms of the location representation, such as Spanish [Fernandez 95]

\(^{15}\)N: No; D: Directly; I: Indirectly. The indirect representation maps tense into time, thus coping with tenses in languages that a tense specific representation could not express. For instance, the classic [Reichenbach 47] representation cannot express the French “futur antérieur du passé” and the “futur du passé” tenses as these require 4 reference times.

\(^{16}\)No, but the extended notion of inheritance takes on some of the burden.
Table 8.2 (p. 367) compares the language independence of SemNet to that of other representations. [Morgan et al. 94] gives further examples of SemNet’s language independence.

The distinction between language and concept nodes separates many of the language-dependent aspects out from the conceptual level. For instance the grammatical gender may vary ("the table" (neuter), "la table" (feminine), "der Tisch" (masculine)) whereas the conceptual gender does not: non-sexed. Because the conceptual level is not a dummy (i.e. does (should) not rely on the spelling of the words to derive its meaning), but defines its concepts in terms of others, thus resulting in differing reasoning behaviours according to the concept, it expresses the meaning of its concepts in terms of internal phenomena-models (such as time or values). This means, that should a word not exist in some language, such as "drizzle" in Italian, a paraphrase can be built for it ("pioggia rada", i.e. "sparse rain"). In this case, this comes from the fact drizzle is a spec. of raining events, defined to have a low intensity. It also means that each word can correspond to more than one concept, and words of different languages may share some meanings but not others. For instance "star" and "der Stern" share the meaning of astronomical object meaning but not of celebrity.

As 4.2.1.2 (p. 51) shows, QLF’s value representation is language dependent: it follows the English syntactic construction "John is 2 meters tall" by the use of be in its meaning representation. SemNet on the other hand, uses the specific physical

---

17 Not in CLE-3, but apparently in CLE-6: 4.2.3 (p. 54)
18 only with [Almeida 1986]'s non-standard extensions
19 LEX arcs refer to sensory perceptions rather than words... but they are used as words. They do not incorporate the linguistic information associated with linguistic nodes in LOLITA
size concept **has . height**. This corresponds directly to the French "Jean mesure 2 mètres", although its original motivation was reasoning efficiency. Modelling concepts with values also allows concepts from different languages to be related - despite their referring to different ranges of the value. For instance, "tall" in Italian refers to a different height to "tall" in Dutch. Similarly if colours are modelled using values, the two Russian words for blue could be captured, an approximation to which is in English the difference between light and dark blue.

The deep analysis of tense into temporal ordering, and representing (incompletely as yet) aspect in terms of other phenomena, is another example of language independence: aspect and temporal ordering differ according to language, so representations that express them directly, such as CGT and CLE-3, cannot expect to be language independent even for languages with tenses. For languages in which the order of events is expressed explicitly, such as Chinese (see 7.5.4.1 (p. 287)) a direct representation of tense would be useless. The same applies to the calendrical system that varies from culture to culture: the modelling in terms of time operators allows any calendar to be expressed.

Language independence has also been tested in practice: An experimental Italian-English translator was easily produced, requiring a hundred line Italian morphology module, seven extensions to LOLITA’s English grammar, one new normalisation rule, and one new semantic rule to be added. All other changes were to the interface, or the data (adding new labels for the words) [Morgan et al. 94].

Overall SemNet appears to score highly for language independence, and although the author's European culture might have biased his viewpoint, the emphasis on models for reasoning can be hoped to have kept the language dependence to a minimum.

---

20 See 7.1.4.6 (p. 254), D.1.3.3 (p. D-25) and D.1.3.3 (p. D-29).

21 See 7.5.4 (p. 286) and 7.5.4.1 (p. 287)
8.4.4 NL Analysis

SemNet not only supports NL analysis – as proven by LOLITA’s very existence – but supports it well, as shown by LOLITA’s participation in many events and projects detailed in 8.1 (p. 297). Indeed, many features of the representation were specifically designed for analysis, such as the representation of ambiguity, without breaking overall cohesion.

The representation also satisfies the requirements particular to NL systems detailed in 3.1.3 (p. 28). SemNet represents all concepts explicitly as nodes, which can be picked up as referents (see 4.1 (p. 45) and 8.1 (p. 297)). Similarly there is no representational distinction between meta-properties and properties, mirroring the behaviour of NL.

Overall, in comparison to other representations such as ANALOG or SNePS claimed to support NL analysis, SemNet’s position is very strong.

8.4.5 Reasoning

Even for NL systems\textsuperscript{22}, an important feature of a representation is the ability to support reasoning. This has been shown in practice for the implemented SemNet, both in this thesis and in other publications.

8.4.5.1 Standard Inferences

LOLITA incorporates a wide range of reasoning techniques summarised in table 8.3 (p. 370) and used throughout the system.

Inheritance, Analogy and Semantic Distance owe their efficiency to SemNet’s use of a set hierarchy in which every concept is integrated – in contrast to SNePS,
Inheritance  This currently performs a sort of virtual copy by listing the events connected to a concept’s ancestors. Thus it is less powerful than the scheme discussed in 6.4 (p. 164) advocating building the inferred events. It also currently requires a different interpretation for the returned events. Further details are given in [Shiu 96].

Epistemological  Finding out whether LOLITA believes something is currently equivalent to finding out whether something (or its negation) is in SemNet. Similarly, finding out whether someone else believes something is currently equivalent to finding out whether he said it.

Personal Closed World Assumption (Plausible) Assumes that if an event involves LOLITA she should know about it. Also, in absence of other information, LOLITA believes nice things about herself (like her being healthy).

Logical Connectives, Basic Causality, Synonyms & Antonyms  A variety of simple inference rules used to test whether concepts are equivalent or not.

Analogy  The implementation of [Long et al. 93].

Semantic Distance  An approximation of [Short et al. 94a, Short et al. 94b] is used by pragmatics for disambiguation. It addresses the issue of similarity (A.3.1 (p. A-28)) and of associativity (2.10.1 (p. 85)).

Table 8.3: Standard Inferences

ANALOG or CGT for instance. This in turn is only permitted by the explicit representation of every concept by its own node – in contrast to $\mathcal{FOPL}$.

8.4.5.2 Building Inferences

Inference is also used extensively during the building of the representation: new nodes are checked to see if there is an equivalent old one; relate (which is used to join words such as composite nouns (cod liver oil) or possessives) is disambiguated by determining plausible relations between the concepts from existing knowledge in SemNet; type-checking; semantic integration; and the resolution of metaphors and metonymies (see [Heitz 96]).

8.4.5.3 Other Inferences

For the D.E.A.R. project (using LOLITA as an NL-interface to a database, – see 8.1 (p. 297)) the SQL version of the NL query is inferred from SemNet by a specially
built inference engine which deals with problems such as: The user inquires whether John runs. If the database contains "John does not move", the NL-interface must deduce the answer No. This can only happen if the interface also asks the database the SQL equivalent of "Does John move?".

In 8.3.3.2 (p. 347), the high similarity of the values representation to the internal representations of problem solving systems such as [Dechter et al. 91] was underlined. This provides some degree of confidence that the design of the values representation – underlying all values – will prove particularly suitable for reasoning. Similarly a transformation from the time representation (and hence Textrefs) to the internal representation of [Dechter et al. 91], provided in 8.3.6.2 (p. 357), shows that the time representation can be used for reasoning. Furthermore, Drs M.Fox and D.Long of the LNLE (Durham University) are using SemNet and its time representation in their planning system. Finally, [Poria 97] discusses an algorithm for building explanations using causality, pre- and post-conditions and the template events of SemNet.

8.4.5.4 Conclusion

A wide range of reasoning methods have been implemented for SemNet, and many others appear easily applicable to it. SemNet's thus seems to support reasoning well.

8.5 Scalability

Scalability ranked second in 3.5 (p. 42)'s ranking.

8.5.1 Uniqueness

For SemNet, uniqueness can be divided into two aspects. Uniqueness of the representation analyses whether it is possible for two different statements to express the
Chapter 8: Evaluation

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>DL-N₁</td>
<td>No²⁵</td>
<td>No</td>
<td>DL₂</td>
<td>L/C₃</td>
<td>L-R₄</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>No</td>
<td>No₅</td>
<td>L-R₆</td>
<td>No</td>
<td>S-R₇</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No²⁶</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DP-N₈</td>
<td>C₉</td>
<td>L-N₁₀</td>
<td>C-R₁₁</td>
<td>No</td>
<td>No²⁶</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No²⁶</td>
</tr>
<tr>
<td>No</td>
<td>L-N₂¹₂</td>
<td>P-R₁₃</td>
<td>Is one!</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>No</td>
<td>S-D₁₄</td>
<td>L₁₅</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>No</td>
<td>No₁₇</td>
<td>L₁₈</td>
<td>No</td>
<td>No₁₉</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No²⁶</td>
</tr>
<tr>
<td>Type</td>
<td>Remedy</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D: Deliberate</td>
<td>P: Partial</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S: Superset</td>
<td>C: Contradiction</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N: Normalisation</td>
<td>R: Rule</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L: Limited</td>
<td>D: Definition</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 8.4: Breaks of uniqueness

same fact. But in a representation such as SemNet, new concepts can be defined. This introduces a second aspect: maintenance of uniqueness.

8.5.1.1 Uniqueness of the Representation

• Investigation

It is particularly difficult to show the uniqueness of a representation. It corresponds to stating that for a given set of lego-brick shapes, no object can be built using two different sets of shapes. Disproving uniqueness is simply a matter of finding a counter-example. To overcome this problem, the representations developed throughout this thesis will be tested for uniqueness in pairs. The task is simplified by asking oneself whether it is possible to build a contradictory statement from the two representations. If it is, the combination of the representations might break uniqueness. Only the tests which are not obvious are justified. The comments are associated with a note, such as "(P)".

There are two additional breaks of uniqueness for efficiency:

²⁴Time refers to time-like reasoning: currently time and textrefs
²⁵See 5.4.2.1 (p. 137).
²⁶has_part and inst. express different concepts: see 7.4 (p. 269)
1. Framed universal quantification is equivalent to $\forall \rightarrow \exists!$ and $\exists! \rightarrow \forall$. This break of uniqueness is deliberate as framed universals occur often and are the only case of two arcs combining to restrict the relation expressed by the arc itself. By adopting a special representation the need to test for multiple arcs of same type between every two nodes is avoided, and the bijective nature of the relation the arc expresses can be dealt with directly. The cost of normalising to framed universals is offset by the overall gains of efficiency. (E)

2. Arbitrary quantification is a shorthand to avoid building instance nodes and inst. events, resulting in a smaller network that need not be swapped in and out of memory. It soak up the memory burden that is imposed by expressing concepts such as partial order in terms of $\ominus$, thus improving overall uniqueness.\(^a\) (E)

3. Cardinality and quantification interact in three different ways:

- A set of size 1 or an instance with individual quantification are almost the same concepts. However one can also refer to the former as a set ("A one man team") which is impossible for the latter.

- Cardinality can contradict quantification. For instance, if two sets $A$ and $B$ are joined by an arc of quantification $\forall \rightarrow \exists!$, then quantification implicitly states there are more elements in $A$ than there are in $B$. Cardinality can contradict this. Reasoning is required to detect this situation. (RV)

- If two sets $A$ and $B$ are joined by an arc of quantification $\forall \rightarrow \exists!$, and are of equal cardinality, then there is one $a$ in $A$ for every $b$ in $B$. The same is true if the relation between $A$ and $B$ had had a $F \rightarrow F$ quantification, and no cardinality information. In this sense, there appears to be a break of uniqueness, but since size. is not inherited, and can be hypothetical, whereas quantification is always inherited, and never hypothetical, one could argue that a (trivial) reasoning step is involved.

4. The use of arbitrary quantification and synonym. is illegal, and must be tested for since arbitrary quantification is a shorthand for inst. D.6.3.5 (p. D-82) (C)

5. Sorts can express implicature (5.4.3.3 (p. 152)). But so, it would seem, can belief, using $a \rightarrow b \equiv \neg a \lor b$. For instance, $\forall x (\neg \text{man}(x)) \lor \text{mortal}(x)$. Since $x$ is quantified over the universe, it corresponds to typeless. The action corresponding to $\neg \text{man}(x)$ can be defined from the antonym of the set of men with respect to typeless. Type-checking cannot unify a proposed subject with a direct ancestor other than the parent of the relevant template event: the direct ancestor would lack properties that subjects of that type of event must have – for instance being human for owners. In this case, type-checking can unify neither not-men things nor mortal things with typeless. Thus, the statement cannot be built in SemNet in terms of belief. Since sorts cannot express any other uses of implication, there is no further scope for breaking uniqueness. For further details, see 6.6 (p. 188). (C)

6. Set relations spec. and inst. may not be definitional with respect to their subject.: 5.4.2.2 (p. 189) (C) (P)

\(^a\)It corresponds to the sharing used in the fly-weight pattern in O-O programming. [Gamma et al. 95]
7. *synonym.s* could be used to break the uniqueness of sorts: If a *synonym._ event stated that concepts X and Y share the same instantiations, but the definitional child Y of X has some more definitional restrictions, Y's definitional restrictions would effectively be observational: see D.6.3.1 (p. D-78). This means it might be possible to replace sorts completely by synonyms, but this would probably break distributedness (see 4.9.4 (p. 103)) (E)

8. The use of *spec._ breaks the uniqueness of *inst._ deliberately. The resulting increase of determinism of search improves efficiency despite the additional cost of normalisation: B.3.1 (p. B-20) (E)

9. Cardinality could state that a subset has more elements than its superset. This type of contradiction has to be detected by reasoning. (RV)

10. *synonym._ does not directly break uniqueness with *inst._ and *spec._ (D.6.3.5 (p. D-82)), but cycles of *spec._ are equivalent to equality: if $\mathcal{H} \subset \mathcal{O} \subset \mathcal{H}$ then $\mathcal{H} = \mathcal{O}$. To avoid the efficiency costs of a cyclic inheritance hierarchy, such cycles are replaced by *synonym._ events (5.4.3.2 (p. 148)). (E)

11. *antonym._ assumes its *subject.s are subsets of its *object.s. If this is not true, the *antonym._ event is meaningless. (C)

12. Two sets are trivially extensionally synonymous if both have 0 cardinality. This situation can only be detected by reasoning. (RV)

13. Any observational *antonym._ can be expressed purely in terms of cardinality: the cardinalities of the partitions must add up to the cardinality of the superset, and the cardinalities of the intersections of the partitions must be 0. However, *antonym._ can define one of the superset's partitions with respect to the other partitions and the superset itself, something only cardinality augmented by *synonym._ can achieve. (P)

14. Antonyms can be expressed in terms of *synonym._ belief events, and quantification. D.6.3.3 (p. D-80)

15. Addition of zero and multiplication by one are equivalent to *synonym._ for values: (P) (RV)

16. Two times that *starts._ and *ends._ at the same instant are synonymous: 7.5.2.4 (p. 277) (P) (RV)

17. Because one can state that an object's parts have smaller values than the object itself, for all objects of some type, it would appear that parts and values break uniqueness: a car part is lighter than the car. However, if in a specific statement, an object part has the greater value, the contradiction is between the inferred value from the general statement and the value in the specific statement -- not between the part and value events.

18. Values and time break uniqueness as far as interval durations go: it is possible to build an interval of a given duration with *starts._ and *ends._ and a set of unit intervals or by using values (*has_duration*). The first choice is however very clumsy and inefficient. (E)

19. Because one can state that all events' parts (sub-events) are shorter than the full event (super-event), it would appear that parts and time break uniqueness. However, if in a specific statement, a sub-event takes longer than its super-event, the contradiction is between the super-event's time and the sub-event's time -- not between the part and time events.
C This rule is enforced by rejecting the building of such events.

E This break of uniqueness improves efficiency.

P This cannot be avoided since it is part and parcel of the concepts modelled.

RV Since it might not be known whether a value is zero or not, it is not normalisable, so values reasoning engines have to know this.

- Delocalization of events' actions to subject- and object- nodes for better
determinism of search (B.3.1 (p. B-19)). This is necessary due to the imperfect mapping of a graph onto a linear address space.

- Family controls to avoid searching up the inheritance hierarchy (B.3.3 (p. B-21)).

Finally, it might appear that partial arcs break uniqueness of quantification: An event may have many subjects, expressed either as partial arcs, or by a quantification such as \( I - \text{subject}. - \forall \). However, in the first case the individual participants are directly referred to, whereas in the second, all the members of the concept referred to participate. Because every concept is defined intensionally in terms of a set of properties to satisfy, no concepts are (intensionally) the union of some others. This means that intensionally, the two cases refer to different concepts, and uniqueness is not broken.

The other representations (intension, frames of existence, linguistic nodes, textual references and ambiguity) do not appear to further break uniqueness.

- Analysis

The two main causes of breaks of uniqueness are the synonym- event and trade-offs to improve efficiency. Of the 19 breaks, 7 were due to synonym- and 8 to efficiency, leaving 6 due to other causes\(^{27}\).

The reason SemNet was to have a high degree of uniqueness was to decrease the search space and thus improve efficiency. It is therefore of no great concern when uniqueness is broken to further heighten efficiency. However, the other main factor,

\(^{27}\)Two breaks due to synonym- improve overall efficiency.
the synonym event, must be considered. Its introduction was necessary to express extensional equality, and proved useful in the representation of ambiguity. Five of the breaks due to synonym cannot be avoided, coming from the nature of the concepts (7,10,12,15,16) considered. Of the two left, one is due to arbitrary quantification and is hence possible to eliminate (8.8.2 (p. 394)). The other main cause is the use of antonym, union, and other such events which can be expressed by large combinations of other events (see D.6.3.2 (p. D-78)). They were introduced to improve efficiency by reducing network size, but are rarely used in practice.

Despite this thorough check, other breaks of uniqueness cannot be guaranteed not to exist. For instance, it is conceivable that two alternative expressions of the same statement might be representable, using two combinations of many representations. However the work on maximising compositionality, non-linearity and distributedness should reduce the chances of uniqueness being broken: these three properties are best satisfied when each atom of the representation encapsulates tightly a particular meaning which is independent from all other representational atoms. This is best achieved when each atom represents a completely independent idea. If the ideas really are independent, then different combinations of atoms cannot express the same concept. I.e., the uniqueness of the representation cannot be broken.

8.5.1.2 Maintenance of uniqueness

Because SemNet is concerned with the building of new concepts, and building concepts is essential to most of the processing it supports, concepts are bound to be re-occur at some point. Not only re-occur, but be expressed purely in terms of their restrictions. This means, that unless some means of maintaining uniqueness is devised, multiple copies of each concept will appear in the K.B.

To avoid such copies, a means is needed to find all of the potential copies of the concept within a very large database – no mean task. However, this means already exists: the inheritance hierarchy organises concepts according to their restrictions so two copies of an intentionally identical concept should appear in the same place in the hierarchy. Thus, when concepts are integrated into SemNet by semantic
integration, repeated concepts will be automatically found.

Thus, the inheritance hierarchy and sorts provide a structure for maintaining the uniqueness of new concepts, and this uniqueness is guaranteed by the same tools that maintain them. This feature is unavailable to all representations which do not maintain every concept in the inheritance hierarchy.

### 8.5.1.3 Uniqueness: Conclusion

The overall conclusion is thus mixed. SemNet includes 9 breaks of uniqueness which decrease efficiency for a total of 55 entries in table 8.4 \( (p. \ 372) \) (18\%). But table 8.4 \( (p. \ 372) \) does not include other parts of the representation which do not cause any breaks – which if added lead to 9\% failure-rate. Furthermore SemNet ensures the uniqueness of new concepts by semantic integration, something only KL-ONE also attempts. The overall uniqueness is therefore good, but may be improved a little. Given the richness of SemNet in comparison to its nearest rival in the uniqueness league (CGT), and given the fact most breaks of uniqueness were limited to special cases, SemNet’s overall performance is very honourable: better than the others, although not perfect.

### 8.5.2 Cohesion

Cohesion is achieved by ensuring that every representation uses the full expressive power of existing representations. This means that the interpretation functions of later levels use the results of earlier ones. And indeed, SemNet’s interpretation process can consist of various layers, since each representation presented in this thesis, beyond nodes and arcs, is built upon another: each representation only assumed the services of the representations upon which it was built. The final picture is of five broad layers or levels, which each may encompass further layers.

The first level of interpretation deals with the building blocks of events: Arc types, Quantification, Sorts and various controls such as the belief control (Real versus Hypothetical). The rules of this level apply to all of SemNet’s concepts. For in-
stance, one such rule is that the type of an event is always given by its action event. The results of these interpretation rules can be further interpreted by rules pertaining to higher-level representations. This first level thus constitutes the framework within which everything else in SemNet is defined. In essence it defines a simple language that can express events.

The second level of interpretation deals with simple events: its rules combine the interpretation resulting from the first interpretation layer, the interpretation of arcs, to produce new relations between concepts: events.

The third level deals with groups of events of particular types: it consists of independent sets of rules which each deal with interpreting the combination of events of a particular type. For instance, they interpret that if $a$ is $b + 2$, and $b$ is $c * 5$, then $a$ is $c * 5 + 2$. Thus they are each limited to a particular conceptual area. The sets of rules are independent as they operate completely internally to a particular representation and do not concern any other: the representations were presented as clearly building upon – but not changing the meaning of – other representations in this thesis. antonym_, cardinality, values, causality, parts, textrefs, and the language layer are included in this level.

The fourth level processes the interpretation of the third. Its events can either be considered simply as independent information ("The set of dogs is a subset of the set of animals") or as an extension of the search-space of information associated with concepts ("Dogs are furry animals, so are furry"). In this second case, the existing interpretation rules are not actually changed, but the search process that precedes interpretation is extended to look in additional places for information: if the information searched for is found in some other place, meaning preserving transformations are applied to it to move it to the concept being interpreted. This is the case of the inheritance hierarchy (and inheritance) and the synonym_ event (and the transfer of events from synonym to synonym). A key feature is that the meaning preserving transformations is that they use no knowledge beyond the event language (level one) and the relevant fourth type representation: inst_ and spec_ for inheritance, synonym_ for synonym event transfer. The search exten-
sions are perfectly encapsulated within the relevant representation and no further interpretation functions are added.

The fifth level also processes the interpretation of the third. It creates new interpretation modes and extends the transformation rules used by the fourth level. In the normal interpretation mode, the events of this type just give additional information about the given concept: "John believes in concept x", "Event x occurred at time t". However, most tasks assume some range of their values: reasoning about events considers some range of time, some frame of existence... Thus new interpretation modes are used which incorporate a state specifying a range of time, and/or frame, and/or belief for the concepts to be interpreted. No additional interpretation combination rules are added, but the search that precedes interpretation filters out much of the search space it considers. The transformation rules used by the fourth level are also extended assuming knowledge about the event language (level one) and the relevant fifth type of representation, and the existence of a mechanism for extending the transformation rules of the fourth level. Frames, Belief and Time fit into this category.

Strictly speaking the transformation rules discussed in the fourth and fifth layers are reasoning. However, many tasks simply wish to ignore the inheritance hierarchy, or the availability of synonym events. It simplifies them considerably if dealing with these problems is delegated to the interpretation function they use. Thus further levels could be considered which hide other forms of simple reasoning.

Depending on the representation considered, the third level may have further internal layers as illustrated by figure 8.5.2 (p. 380):

- Negation, Time, Ambiguity and potentially union (etc), rely exclusively on existing reasoning methods: negation uses antonym and cardinality; time uses time-like reasoning; ambiguity uses synonym, textrefs and belief; and union (etc) could be implemented purely in terms of synonym and belief.

- Cardinality, Belief and certainty values, and Textrefs rely heavily on existing reasoning methods: cardinality, belief and certainty values rely on values (and less on belief); and textrefs rely on time-like reasoning.
Figure 8.2: Cohesion

- Parts, Language and Time-like use to some extent other representations: Parts, Language and Time-like all use values.

- All representations are event based so use events, arc-types, sorts, quantification, and controls.

Figure 8.5.2 (p. 380) illustrates how all but the most basic representations build on the others. The preceding analysis of uniqueness shows this to be achieved with no two representations covering the same conceptual ground. Furthermore, the intense layering demonstrated by 8.5.2 (p. 380) shows an intense reuse of reasoning methods:

- Negation, Time, and Ambiguity rely exclusively on existing reasoning methods.

- Cardinality, Belief and Certainty values, textrefs, potentially union (etc) rely heavily on existing reasoning methods.

Despite the argument being abstract, the result is of real benefit. For instance,
it means that sorts are used at many levels: at the level of single events, they provide a representation for template events, at the level of groups of events, they restrict events or their dependencies (time, location...), as well as restricting other concepts. Thus they represent different level-specific concepts, in such a uniform manner that whatever new representation is added to SemNet, it will already have sorts functionality implemented. The cohesion is such, that some representations take over precisely where others end. For instance, partial arcs take over where quantification ends. Had this not been so, the representation’s richness would have been seriously curtailed, since statements which could not be made in all circumstances where they should apply.

Reasoning rules usually also follow the layering discussed, concentrating on particular representations (see table 8.3 (p. 370)) – although some encode the dependencies between representations, such as the fact that subsets are smaller or equal to their supersets. Thus, with the layered interpretation function, with the way in which representations cooperate where one is lacking without breaking uniqueness, and the way in which there is a unified way of expressing many relations such as sorts, one can conclude that SemNet has a very high cohesion, better than all others which were evaluated “Weak”.

8.6 Efficiency

Efficiency ranked third in 3.5 (p. 42)'s ranking.

8.6.1 Topological Distance

Table 8.5 (p. 383) lists the topological distance of the events used by SemNet’s basic and extended representations. The distances in the table assume the action arc has to be checked, but not traversed. The action arc must always be checked, even with the scheme described in B.3.1 (p. B-19): the action may be ambiguous, in which case the delocalized action is the most specific action which encompasses all the alternatives. However, it need not be traversed if the arc and its target
are recorded on the node: the information recorded on the action node is rarely of interest (what the template event is).

Where there are two distances in the table, for instance "3 / 2+n":

- If the event is transitive, it may have one subject_ or object_ for many object_s or subject_s respectively. The first value is the distance from one of the multiple arcs’ targets to the target of the unique arc. The second value is the distance from the target of the unique arc to all of the targets of the multiple arcs.

- If the event is intransitive, the first value is the number of steps to reach the event node, and check its action_ (traversal used for analysing belief chains for instance). The second value is the number of steps to reach the event’s other subject_s.

The table does not cover the representation fully. In particular, the use of controls such as family controls reduces the topological distance further. This proves an important optimisation for tasks such as semantic type-checking.

Overall topological distance is low: all information is only a few arcs away. However, it might seem high for particular tasks. For instance, determining the date or a numeric value requires analysing a network of time or values relations. That such simple tasks are penalised might seem awkward, but it should be remembered that the gain in cohesion avoids a multitude of reasoning techniques. Since the point is reasoning, and not printing dates, this compromise is the correct one. Moreover, SemNet easily beats its competitors CGT, SNePS, ANALOG and KL-ONE. For instance, ANALOG uses two nodes for each event, while CGT implements all arcs in terms of multiple simple LINK arc primitives and encourages multiple copies of concepts.
### Table 8.5: Topological distance of SemNet events

<table>
<thead>
<tr>
<th>Relation</th>
<th>Type</th>
<th>Distance</th>
<th>N.p.c.</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>inst_.</td>
<td>Inherit H.</td>
<td>2</td>
<td>Many</td>
<td></td>
</tr>
<tr>
<td>spec_.</td>
<td>Inherit H.</td>
<td>2</td>
<td>Many</td>
<td></td>
</tr>
<tr>
<td>powerset_</td>
<td>Set rel.</td>
<td>2</td>
<td>Many</td>
<td></td>
</tr>
<tr>
<td>infinite.powerset</td>
<td>Set rel.</td>
<td>2</td>
<td>Many</td>
<td></td>
</tr>
<tr>
<td>union_.</td>
<td>Set rel.</td>
<td>2 / 1+n</td>
<td>Many</td>
<td></td>
</tr>
<tr>
<td>size_.</td>
<td>Card.</td>
<td>2</td>
<td>One</td>
<td></td>
</tr>
<tr>
<td>belief_.</td>
<td>Bel.</td>
<td>2</td>
<td>Many</td>
<td></td>
</tr>
<tr>
<td>and_.</td>
<td>Bel.</td>
<td>1/n</td>
<td>Many</td>
<td>and_. has n arguments</td>
</tr>
<tr>
<td>or_.</td>
<td>Bel.</td>
<td>1/n</td>
<td>Many</td>
<td>or_. has n arguments</td>
</tr>
<tr>
<td>xor_.</td>
<td>Bel.</td>
<td>1/n</td>
<td>Many</td>
<td>xor_. has n arguments</td>
</tr>
<tr>
<td>source_.</td>
<td>Source</td>
<td>2</td>
<td>Many</td>
<td></td>
</tr>
<tr>
<td>cause_.</td>
<td>Cause</td>
<td>2</td>
<td>Many</td>
<td></td>
</tr>
<tr>
<td>has_value</td>
<td>Values</td>
<td>2</td>
<td></td>
<td>One per value type per concept</td>
</tr>
<tr>
<td>⊕</td>
<td>Values</td>
<td>1+m</td>
<td>Many</td>
<td>m is the number of object_. arcs.</td>
</tr>
<tr>
<td>⊗</td>
<td>Values</td>
<td>1+m</td>
<td>Many</td>
<td>m is the number of object_. arcs.</td>
</tr>
<tr>
<td>◯</td>
<td>Values</td>
<td>1+m</td>
<td>Many</td>
<td>m is the number of object_. arcs.</td>
</tr>
<tr>
<td>synonym_</td>
<td>Syn.</td>
<td>1/1+n</td>
<td>Many</td>
<td>The search must also include the concept’s synonyms’ synonyms...</td>
</tr>
<tr>
<td>antonym_</td>
<td>Ant.</td>
<td>2 / 1+n</td>
<td>One</td>
<td></td>
</tr>
<tr>
<td>has_part</td>
<td>Parts</td>
<td>2 / 1+n</td>
<td>Many</td>
<td></td>
</tr>
<tr>
<td>starts_.</td>
<td>Time like</td>
<td>2</td>
<td>Many</td>
<td></td>
</tr>
<tr>
<td>ends_.</td>
<td>Time like</td>
<td>2</td>
<td>Many</td>
<td></td>
</tr>
<tr>
<td>before_.</td>
<td>Time like</td>
<td>2 / 4</td>
<td>Many</td>
<td>4 includes finding duration</td>
</tr>
<tr>
<td>after_.</td>
<td>Time like</td>
<td>2 / 4</td>
<td>Many</td>
<td>4 includes finding duration</td>
</tr>
<tr>
<td>follows_.</td>
<td>Time like</td>
<td>2 / 4</td>
<td>Many</td>
<td>4 includes finding duration</td>
</tr>
<tr>
<td>is_in</td>
<td>Time like</td>
<td>2 / 4</td>
<td>Many</td>
<td>4 includes finding duration</td>
</tr>
<tr>
<td>has_duration</td>
<td>Time like</td>
<td>2</td>
<td>One</td>
<td></td>
</tr>
<tr>
<td>in_Frame</td>
<td>Existence</td>
<td>2</td>
<td>Many</td>
<td></td>
</tr>
<tr>
<td>in_language</td>
<td>Language</td>
<td>2</td>
<td>Many</td>
<td></td>
</tr>
<tr>
<td>words_used</td>
<td>Textrefs</td>
<td>n+m</td>
<td>Many</td>
<td></td>
</tr>
<tr>
<td>phrase.means</td>
<td>Textrefs</td>
<td>n+m</td>
<td>Many</td>
<td></td>
</tr>
<tr>
<td>Template Events</td>
<td></td>
<td>2</td>
<td>One</td>
<td>Finding any event’s template</td>
</tr>
</tbody>
</table>

N.p.c: Number of events of this type per concept (per believer).
8.6.2 Determinism of Search

At the level of arcs, and nodes determinism of search is high, since no search through the network is involved. At the level of events qualifying a concept, determinism is again high, assuming delocalization of the action (B.3.1 (p. B-19)). At an even higher level, typing and uniqueness improve determinism of search. Uniqueness' contribution is that a limited number of ways of expressing each statement implies a limited number of ways of trying to find it. Heavy typing further improves determinism of search by ensuring that most events can only be connected to a reduced number of concepts. Despite this, where weak, determinism of search can be further improved by adding specialisations of popular events (resulting in different action.s to profit from action delocalization). The prime example is the introduction of spec. to improve inst.'s determinism of search, but this was also done for dates (D.4.3.4 (p. D-64)). Furthermore controls can be used, such as the family control, to avoid any search at all to optimise frequent checks. The only taint to this bright picture is the introduction of inheritance and synonym event transfer (8.5.2 (p. 378)) which increase the number of places where information about a concept must be looked for, hence reducing determinism of search.

Determinism of search in SemNet is therefore very good overall, with slight reductions for inheritance and synonym event transfer, which are useful in their own right. Again it beats its competitors which suffer from being separative knowledge-bases lacking the same powerful inheritance hierarchy (SNePS, ANALOG) or any analogue to the action delocalization (CGT has only the LINK arc).

8.6.3 Distributedness

SemNet's default information combination rule states that if a concept is qualified by two statements, the total information about the concept is the addition of the information given by each. This means that where it applies, the statements it combines are independent, and thus distributed. For instance, it applies to the first interpretation level, which deals with nodes and arcs, rendering SemNet potentially
distributed to the level of single arcs and nodes. However, nothing prevents a later interpretation level to assume that the interpretation of the previous level must be full with respect to some criterion. This is the case for the second interpretation level which assumes that all subject.s and object.s of the event it is interpreting have been read, so that it knows whether or not it is dealing with partial arcs. Since the fourth and fifth interpretation levels do not introduce new interpretation rules, the question is whether the third one introduces further restrictions.

To answer this question is to answer the question “is there any way in which the absence of some feature will lead to conclude something that would not be true were the feature present?” Indeed, the degree to which a representation is distributed corresponds to the smallest size of the segments of the K.B. whose interpretation is sound with respect to the interpretation of the full K.B. Thus the question becomes “where there is lack of some information, is any default assumed?”. Overall, this should not be the case, as LOLITA assumes that in the lack of explicit positive or negative information, that she does not know. However a default is clearly assumed in the case of partial arcs. Is is also clearly not assumed for those representations that rely exclusively on others for their meaning (8.5.2 (p. 379)): Negation, Time, Ambiguity and potentially union. (etc).

Without an explicit interpretation function, which defines formally what the representation “means” in terms of some other representation it is impossible to prove a lack of any default. Furthermore the question is not really whether such and such an interpretation function is distributed, but rather whether the representation precludes it from being distributed. Short of that, the best is to list possible defaults that could have been made and show they did not occur: see table 8.6 (p. 386). A possible source of confusion with distributedness is that simply because the information read is sound with respect to the knowledge base, it does not mean that the concept is equivalent to one with only some of its properties. Hopefully this list, and the fact SemNet has been tested hard for distributedness for over four

\(^{28}\) By the interpretation function. Inheritance or synonym event transfer, for instance, are by no means assumed defaults.

\(^{29}\) i.e. not simply absence.
years bear sufficient witness to its representation’s distributedness.

Thus it does not appear to be the case that a default is assumed for any of the representations above the second level of interpretation. In particular, it appears that the (group of partial) arc(s) of a given type is the unit of distributedness in SemNet, below which soundness cannot be guaranteed. Clearly, this argument only applies to the representations discussed, since any new representation may introduce assumptions that break distributedness. The argument is not a watertight proof, since a single counter-example would break the claim of distributedness. However, short of formalising the interpretation function, this informal but detailed argument shows that SemNet’s distributedness is very high. This is particularly important since SemNet is a non-separable representation. Indeed, not only is SemNet more cost-effective in arcs and nodes than its closest competitor ANALOG, but it also has complete distributedness above the arc level, rendering it an overall winner in this category (see [Short et al. 96]).

### 8.6.4 Non-Linearity

Linearity is the property of the interpretation function which forces it to traverse its representation in some order. This lack of freedom is detrimental to efficiency. Since this thesis is concerned with the design of the representation, and not the interpretation function, the key question is whether the design of the representation
forces the interpretation process to traverse it in some particular way. To answer this, the cases where the representation can do this must be determined.

The representation could force the interpretation function to traverse it in a linear manner by making it impossible to traverse backwards: for instance by expressing statements as singly linked lists. SemNet does not do this, since its arcs can be traversed in both directions.

The representation could force an order on the interpretation function if some of its terms were designed to fulfil this purpose, like bracketing does in mathematics. SemNet includes no such operators.

The representation could force an order on the interpretation function if the function of a particular term of a statement were implicit in its position in the statement: determining the term's position involves reading other representation – i.e. forces an order on the traversal of the representation. Since the granularity of distributedness has been established to be complete arcs and nodes (8.6.3 (p. 385)), the discussion focuses on representations from this level up.

In a graph, the position of a symbol clearly depends on its neighbourhood: arcs and nodes within some specified distance. Since all representations build upon arcs and nodes, the first question is whether these building blocks are non-linear. Each arc expresses its function by its arc-type, quantification and sorts which it encapsulates. Each node has no function, other than to provide a handle to the concept it represents. Thus, in both cases, the function of the terms is independent of any others. It is not implicit on position.

Although arcs and nodes do not depend on position for their interpretation, the representations that build on them might: the recognition of events. Each event relates concepts in a particular way: reading its subject_ or object_ states that the read concept is qualified by the read event; reading its action_ specifies the type of event; reading both subject_ and object_ states the relevant concepts are related by some event; reading both subject_ and action_ or object_ and action_ states that the relevant concept is in an event of the relevant type; if the event is transitive, reading the whole event states that the subject_ is related to
Chapter 8: Evaluation

<table>
<thead>
<tr>
<th>Link Events</th>
<th>Bridge Events</th>
<th>Internal Operators</th>
</tr>
</thead>
<tbody>
<tr>
<td>inst.</td>
<td>size.</td>
<td>and.</td>
</tr>
<tr>
<td>spec.</td>
<td>has_value</td>
<td>or.</td>
</tr>
<tr>
<td>powerset.</td>
<td>at.time</td>
<td>xor.</td>
</tr>
<tr>
<td>infinite_powerset</td>
<td>has_duration</td>
<td></td>
</tr>
<tr>
<td>union.</td>
<td>in_language</td>
<td>⊕</td>
</tr>
<tr>
<td>belief.</td>
<td>belief.</td>
<td>⊗</td>
</tr>
<tr>
<td>source.</td>
<td>phrase_means</td>
<td>starts.</td>
</tr>
<tr>
<td>cause.</td>
<td>words_used</td>
<td>ends.</td>
</tr>
<tr>
<td>synonym.</td>
<td></td>
<td>before.</td>
</tr>
<tr>
<td>antonym.</td>
<td></td>
<td>after.</td>
</tr>
<tr>
<td>has_part</td>
<td></td>
<td>follows.</td>
</tr>
<tr>
<td>in_Frame</td>
<td></td>
<td>is.in</td>
</tr>
</tbody>
</table>

Table 8.7: Types of SemNet events

the object by a relation of type given by the action. In all the event level cases, there is no obligation to read the event in any particular order. Thus events are non-linear. Furthermore, beyond this level, the function of each event is known. Indeed, there is no way in which the function of an event is influenced by other events not directly connected to it, since every event must be connected to what concerns it. Thus, the problem that CGT probably has, that a whole set of concepts may be qualified if one of them is connected to a particular operator does not occur. Indeed, this relies on “statement-level operators” being applied to concepts of a statement – which only works if one knows what the statement is. But in SemNet one does not, since SemNet is not a separative knowledge base.

The further representations do not depend on each other for their existence, as the dependency graph they form is acyclic (see 8.5.2 (p. 377)). This means that any function implicit on position can only occur within one of the extended representations. Table 8.7 (p. 388) classifies the events by type. Bridge events link two conceptual systems together. For instance, has_value links non-value concepts with values. Internal operator events define one of these conceptual systems by specifying the relations that build it. Link events link two concepts in a particular way, without introducing a specific conceptual system. The function of bridge events is clear, creating a link between concepts of different systems, so there

---

30 Except, arguably, belief events and the belief control
is no dependence on position. Similarly, the function of link events is position independent: each does not involve other concepts of its representation, having no interpretation rules of the third level to combine it with other events. This is because each expresses a different kind of link, and involves no special conceptual system. This leaves internal operators. The belief internal operators (and\(\), or\(\), xor\(\)) do not impose any order as each simply composes by its type of operation its subject\(s\). Neither bracketing nor operator precedence is needed as each of their arguments is an event which can also be an internal belief operator. The belief value of a belief internal operator depends on those of its arguments, which may require reasoning to ascertain. The value internal operators behave similarly, since their value depends on their arguments', which may or may not be further specified by other value internal operators: there is no bracketing or precedence. The time-like internal operators are all based on starts\(\), ends\(\), and has duration. As the last is a bridge, only the first two need be considered: there is no position-dependence of either of these on the other.

Unless some cause of linearity has been missed, it would appear that in no way does SemNet force interpretation functions to be linear. Since this evaluation is more rigorous than that used to evaluate the other representations and which found fault with them, but nonetheless has failed to find causes of linearity, one can conclude that SemNet has a greater degree of non-linearity than the other representations.

### 8.7 Features for Development

#### 8.7.1 Ease of Use

Ease of use appears high from the fact many students have worked on LOLITA, some for a year or less (final year projects and M.Scs), and have complained about not understanding LOLITA's undocumented source-code but not about the also, until now, mainly undocumented representation. However the initial ease of use may be counter-productive, since much of the power of the representation seemed to be ignored. People invented new representations when existing ones were sufficient,
breaking uniqueness and coherence. Had the representation been initially harder to use, people might not have been so confident in their understanding, and thus might have tried to find out more before starting new designs. Hopefully this thesis will reduce this problem.

8.7.2 Compositionality

As 3.4.2.4 (p. 41) points out, useful non-linearity and distributedness assume a certain degree of compositionality in a cohesive representation. A formal discussion would require an interpretation function from SemNet to some other model to show that it could be built as a combination of a small number of independent small sub-functions which access independent parts of SemNet's representation. To some extent this has been done by [Shiu 96], but being a large undertaking, it is beyond the scope of this thesis. The discussion will therefore remain informal.

8.7.2.1 High-level Compositionality

The claims for high-level compositionality are based on the properties of uniqueness and cohesion already discussed. Cohesion means that each representation relies on others when it wishes to express things the others can express. Uniqueness means that no expressions can be formed to express the same statements using two representations (or different combinations of representations). If both are combined, different concepts must be expressed by different parts of the representation. In other words the whole meaning of a statement is given by combining the results of independent interpretation functions specific to each representation. But this still does not preclude the function that combines these results from being byzantine.

The analysis of cohesion showed that the interpretation process involved five levels, each of which may encompass further levels. In all cases, the interpretation rules of each type of representation are independent as they assume only that they know about their own representation (and the results of the previous interpretation level). The assumed default information combination rule – given by the first level
and implicit in the design of the representation – if a concept is qualified by two statements, is that the total interpretation of the two statements is the addition of the information given by the interpretation of each statement. This means that the problem is reduced to whether or not each representation is internally compositional: each interpretation function only knows about its representation and the results of the interpretation function previous to it. So it does not know about other representations. So the meaning of the part of the statement it processes can only depend on other parts of the statement if those other parts are of the same representation type as it. Thus, the remaining question is whether the interpretation rules of each representation type are internally compositional.

8.7.2.2 Low level Compositionality

Although combining the interpretations of different representations may be compositional, the interpretation functions internal to the individual representations need not be.

Although internal compositionality of quantification, sorts and controls would simplify programming, it would have no effect on distributedness or non-linearity since complete arcs are the atomic symbols. Thus, they will not be considered further.

The second level of interpretation moves beyond this by recognising events: it expresses a concept in terms of its relations as stated by its events, rather than by its arcs. It takes the results of the first type, but extends the search so that the events connected to a concept are fully read and interpreted. It is itself multi-layered, so a primitive interpretation can be given to a partially read event: reading both subject_ and object_ states the relevant concepts are related by some event; reading both subject_ and action_ or object_ and action_ states that the relevant concept is in an event of the relevant type. The interpretation rule processes all aspects of events’ arcs identically (quantification, sorts...), except for the arc-type which determines each target’s role in the event. Thus, here too, the behaviour is compositional: the processing applied to each part of the event does not vary depending on its other parts. Much of the meaning of the event is encapsulated
by the atomic symbols on which it is based. For instance, the quantification of
the relation expressed by the event between its subject concept and its object
concept could be determined directly from the quantifications of these arcs. Even
the notion of event could be argued implicit in the types of the arcs.

The third type of representation interprets the results of the second interpretation.
It deals with the composition of many events of a given representation. It introduces
new knowledge, not encapsulated in the symbols of the layer below. For instance,
it introduces the rule that two spec events with a common argument, playing the
role of subject in one and object in the other correspond to one spec event
between the two other concepts involved. Such knowledge is additional, "external"
so to speak, as it comes from external understanding of the terms rather than
being the consequences of some primitive internal model. Thus, the scope for
compositionality is lesser. Similarly, because each level of interpretation processes
the results of the preceding layer, the granularity of each is determined by the
previous level. Thus, any partial interpretation functions of the third level would
only apply to fully read events. Furthermore, unlike the previous level where the
number of legal groups of symbols to be interpreted was severely limited (every arc
belongs to one and only one event), any combination of events to be interpreted
by the third level is legal. This means that no interpretation rules are expected
beyond the level of combining pairs of events or combining a new event and a
previous interpretation. To this extent, the third level is compositional.

8.7.2.3 Conclusion

The representation is overall compositional. The use of many levels of interpre-
tation where each combines the results of the previous level should reduce the
chances that byzantine interpretation rules get introduced. But the addition of
specific behaviour at the third level for new relations is not compositional. Thus
each operator of the new representations added in this thesis constitutes a break

31 It might cause a contradiction, but that's another matter.
of compositionality. But in each case the behaviour is additional\textsuperscript{32}, rendering representations using the new behaviour compositional too.

## 8.8 Weaknesses

Although SemNet satisfies the requirements honourably, it still suffers from a few weaknesses. The author hopes to continue maintaining the representation, and thus to eliminate them in future revisions.

### 8.8.1 Disjunctive Concept definitions

The introduction of logical connectives allows concepts to be defined by a disjunction of events, either directly, or using definitional synonyms. For instance, \texttt{union} is defined by such a disjunction. Thus, any concept defined to be the union of some sets, is defined by the disjunction of the definitional properties of those sets.

Throughout this thesis, the reader has been warned to avoid such constructions, as they may render semantic integration intractable. But this constraint is easy to forget, as the definitions of \texttt{R} and \texttt{N} illustrate. Although a solution to these particular cases may be easy to find, an overall decision needs to be taken: if the simultaneous use of disjunctive connectives and definitional events is to be forbidden, SemNet's interface should refuse to build such statements. But this decision has far reaching consequences. For instance, it is not always clear, during interpretation, whether or not a statement defines another. Here disjunctions of definitional events prove useful.

It is expected that the conundrum will be resolved by some hybrid solution, which distinguishes pre- and post-integrated events: Disjunctions of definitional events are tolerated only until the concept is to be integrated. By then it must have been resolved to a conjunction of restrictions. This would allow interpretation to

\textsuperscript{32}It does not break existing behaviour. It is implemented using existing behaviour. It is independent of all other behaviours.
proceed as before, but would reduce the problems associated with disjunctive definitions. For instance, union, antonym and infinite.powerset would have fewer alternative forms, requiring less or no normalisation. However, since the hybrid solution relies on determining the exact balance between allowing and forbidding disjunctive definitions, it can only be solved by experience, and thus is subject for future study.

8.8.2 Trading Efficiency for Uniqueness

Arbitrary quantification was introduced to pay for other improvements of uniqueness. While there is no doubt that the basic idea is good – sharing nodes where distinct nodes are unnecessary – the means by which it has been achieved should be revised. The problem is that arbitrary quantification must be recognised by all the algorithms that process quantification. That cost could be avoided by moving the problem to a lower level: implementing copy-on-write behaviour for nodes, and adding to SemNet’s interface a compression process to unify structurally identical nodes.

union, i.antonym and antonym were introduced to avoid the myriad of set, cardinality and synonym relations to which they correspond (D.6.3.2 (p. D-78)). This preservation of space is however paid for by a significant complication of the code, for instance in semantic integration and type-checking. In practice, union, i.antonym and antonym do not appear as common place as first expected, probably because they close the knowledge base: they assume that all specialisations of a concept are known. Thus, it is likely that they will be eliminated future versions of SemNet. The same applies to infinite.powerset which was only introduced in case the need for it arose, something which has not yet happened.

8.9 Conclusion

33No, but the extended notion of inheritance takes on some of the burden.
34See main text.
### Table 8.8: Summary of All Representations’ features

<table>
<thead>
<tr>
<th>Feature</th>
<th>FOPL</th>
<th>QLF</th>
<th>CD</th>
<th>CGT</th>
<th>SNePS</th>
<th>ANALOG</th>
<th>KLONE</th>
<th>SemNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Richness</td>
<td>1/3</td>
<td>2/3</td>
<td>1/3</td>
<td>2/3</td>
<td>2/3</td>
<td>1/3</td>
<td>1/3</td>
<td>3/3</td>
</tr>
<tr>
<td>Cohesion</td>
<td>Weak</td>
<td>Weak</td>
<td>Weak</td>
<td>Weak</td>
<td>Weak</td>
<td>Weak</td>
<td>Weak</td>
<td>Strong</td>
</tr>
<tr>
<td>No M-Prim. Lim.</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Language Indep.</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>NL Naturalness</td>
<td>1/4</td>
<td>3/4</td>
<td>2/4</td>
<td>1/4</td>
<td>2/4</td>
<td>1/4</td>
<td>4/4</td>
<td></td>
</tr>
<tr>
<td>Define Explicit New Concepts</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Scripts</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Packed</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Uniqueness</td>
<td>0/3</td>
<td>1/3</td>
<td>2/3</td>
<td>1/3</td>
<td>1/3</td>
<td>1/3</td>
<td>1/3</td>
<td>2.5/3</td>
</tr>
<tr>
<td>Non-linear.</td>
<td>0/2</td>
<td>0/2</td>
<td>0/2</td>
<td>1/2</td>
<td>1/2</td>
<td>0/2</td>
<td>2/2</td>
<td></td>
</tr>
<tr>
<td>Distributed.</td>
<td>0/3</td>
<td>0/3</td>
<td>1/3</td>
<td>1/3</td>
<td>2/3</td>
<td>2/3</td>
<td>3/3</td>
<td></td>
</tr>
<tr>
<td>Topo. Distance</td>
<td>0/2</td>
<td>0/2</td>
<td>1/2</td>
<td>1/2</td>
<td>1/2</td>
<td>1/2</td>
<td>2/2</td>
<td></td>
</tr>
<tr>
<td>Det. of Search</td>
<td>0/2</td>
<td>0/2</td>
<td>1/2</td>
<td>1/2</td>
<td>1/2</td>
<td>1/2</td>
<td>2/2</td>
<td></td>
</tr>
<tr>
<td>Monotonic</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Non-Separative</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Single Rep.</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Used in a K.B.</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td>Sem. Net.</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Ease of Use</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Correctness</td>
<td>2/2</td>
<td>?</td>
<td>?</td>
<td>1/2</td>
<td>1/2</td>
<td>0/2</td>
<td>2/2</td>
<td>1/2</td>
</tr>
</tbody>
</table>

The results of SemNet’s evaluation are summarised in table 8.8 (p. 395). As before each row scores systems qualitatively by functionality. Overall the results are pleasing, as apart from uniqueness, the results obtained are those set out in the requirements. Some of the uniqueness breaks could be avoided in future revisions of SemNet, but, for a representation of this richness, the score is quite good.

Monotonicity has not yet been analysed. It is not guaranteed, since information could be deleted from SemNet. However, deleting information is a way of using SemNet. Nothing prevents it from being used in a monotonic fashion: it is possible only to add information to SemNet and to gradually specify more and more precisely the belief value that information has by adding value events. Depending on its resulting value, the information will be used or discounted, without ever having deleted an event. The same process can be used, as in QLF to flesh out the analysis with more information at each step through the analysis – as was hinted at in the design of the representation of ambiguity. Thus SemNet does support monotonicity.
Overall SemNet scored higher or as well on most points than the competition. Points of weakness are a full script system and correctness. Correctness is not a topic for this thesis. Some of the power of a script system is provided by the notion of inheritance assumed by this thesis, dealing with such things as the template event of to buy involving a seller, a buyer, and two transactions. Further power is expected to come from the use of homogeneity theory, but this is very much a subject for future study.

Finally, SemNet has the great advantage of encapsulating the knowledge gained from real problems with a real implementation: LOLITA, one of the largest NLP systems, has benefited from its academic environment and its industrial evaluations and usage. Thus, despite the few weaknesses outlined in this section, SemNet appears to a strong base on which to build the next generation of NLE tools.
Chapter 9

Conclusion

9.1 Review

This thesis set itself the task of developing a rich efficient representation for large scale natural language processing. It started this task by clarifying the concepts the representation assumes and expresses. This clarification was later used in the development of the basic representation, in particular the notions of sorts, intension, and the omni-present inheritance hierarchy. Existing metrics were used to evaluate representations' performance, to provide a sound basis on which representations could be compared. Where these were insufficient, such as for the problem of search, entirely original metrics were developed.

This thesis fully achieved its goals. It describes completely a representation covering a very wide range of phenomena, and introduces cohesion as an architectural principle for large scale representations. This new notion gives a new scope to the principles of uniqueness and compositionality which are often advocated to be "a good thing". This distinguishes SemNet from other representations which have been extended at different times, by different people, for different purposes, resulting in very different solutions.

Not only is the representation rich and cohesive, but it is also designed for usage in a real system. This underlies the concern for efficiency. SemNet satisfies this,
as it satisfies all the standard metrics and some new ones introduced in this thesis: determinism of search, distributedness, non-linearity, topological distance, and uniqueness. Usage in a real system also means architectural language-independence for commercial reasons, naturalness to simplify development, and usability by reasoning engines. SemNet also addresses all of these problems.

SemNet does not just pay lip service to the real problems encountered in large-scale NLE. Indeed, this thesis has presented a tour of LOLITA's implemented semantic analysis component. This component, with generation, deals with every form of representation in SemNet. Moreover, unlike generation, it is at this component that insufficiencies of representational richness first become apparent. With its critical position between natural language and SemNet, it acquits itself well of its duties, as demonstrated by LOLITA's successes in industrial tasks for Siemens and Rolls Royce, as well as its participation in academic competitions such as MUC'95. The current incarnation of SemNet, and the associated algorithms, owe much to lessons learnt with LOLITA's implementation.

Overall, the thesis has been successful. A rich general purpose representation has been devised, which has satisfied all the requirements drawn up for it. New metrics have been introduced to improve the evaluation of representations. A new architectural principle has been introduced to structure the design of representations. And this theoretical work has been backed up by a real implementation.

### 9.2 Future Work

This thesis is only a small contribution to the field. Practical large scale NL systems are still out of reach, requiring further research. This thesis presented through a representation for NL systems a paradigm of NL systems:

Thus future work not only includes the obvious – maintaining the representation by removing its weaknesses (8.8 (p. 393)), extending it to new phenomena, in particular location, while preserving or strengthening SemNet's strict requirements – but also applying the paradigm to a new non-pipelined architecture for NL analysis.
The main features of this architecture are its use of a single data-structure for all processing, keeping all information explicit and available to any rule: SemNet; its exclusive reliance on graph transformation rules which can be applied at any time: supporting reasoning, NL analysis and generation\(^1\); and a central task allocator determining which rules or reasoning would be most beneficial to the task at hand at any point.

These features improve the system’s flexibility making it easy to implement new rules and improve work-sharing to avoid doing work many times. They also should allow explicit parallelism since each rule may be applied separately. No conflicts should exist if rules only add information, rather than deleting it, because SemNet is distributed. Thus two separate processes reasoning about time could add two different \texttt{at.time} events to an event, which would later be unified into a single event and time, by adding \texttt{synonym}. Instead of deletion, the values representation can be used to assign a greater belief to the unified event, again by adding an event: \(\oplus\).

Research issues clearly include efficiency. For maximal flexibility, a declarative formalism to express the rules is needed. This needs to be compiled to an efficient procedural form for a computer to use. Different rule application strategies need to be devised, and compared to determine their characteristics. Finally, because all rules add events to the network, a variety of garbage collection strategies are needed to control the network’s size.

Further research will also include a deepening of the notion of distributedness: intuitively it seems that distributedness should allow extremely efficient parallelism over networks of computers, requiring a low communication bandwidth between processors and asynchronous message passing. Indeed, each processor can maintain a local cached copy of SemNet, on which it works\(^2\). The conclusions it reaches can be sent through the network to all the other processors. These can update their local copies of SemNet when it is convenient to them, i.e. asynchronously. Distributedness ensures no problems such as deadlock will arise, as long as no

\(^1\)Only at the level of input and output of textrefs would other data structures be used

\(^2\)This assumes that no closed world assumptions will be made.
information is deleted. Thus, no part of the semantic network need be locked from other processes at any time.
Bibliography


[Baring-Gould] Baring-Gould: Please see under Short. [Garigliano et al. 95], [Morgan et al. 94], [Short et al. 94a], [Short et al. 94b], [Short et al. 96]


[Fauconnier 75] Fauconnier G. Do Quantifiers Branch? Linguistic Inquiry, 6(4):555-578


1Now Baring-Gould S.


time of inheritance in nonmonotonic semantic nets. Proceedings of AAAI-
87. Also available as technical report CMU-CS-87-175 from Dept of Computer
Science, Carnegie Mellon University.

Functional Programming Language Haskell, Version 1.2. in ACM SIGPLAN
Notices 27.

[Hwang et al. 93] Hwang C.H. & Schubert L.K. 1993. in Minds and Machines 3:

[Imbs 60] Imbs P. 1960. L’emploi des temps verbaux en Français Moderne. Librarie
C. Klincksieck, Paris.

ural Language Processing: The PLNLP Approach. Kluwer Academic Publishers,
Boston, Ma.


Language. Prentice Hall.

temporal models In Artificial Intelligence Review 7:401-419. Kluwer Academic
Publishers, Holland.

Issue on Propositional Knowledge Representation. In Journal of Experimental
and Theoretical Artificial Intelligence, 5, No 2.

[Kumar 93] Kumar D. 1993. From Beliefs and Goals to Intentions and Actions:
An Amalgated Model of Inference and Acting. Phd Thesis. State University of
New York and Buffalo.

0-14-013487.
BIBLIOGRAPHY


²Now Baring-Gould S.


³Now Baring-Gould S.


