Durham University

# Durham E-Theses

# *Type theoretic semantics for semantic networks: an application to natural language engineering*

Simon K.Y. Shiu

**How to cite:**

Shiu, Simon K.Y. (1996) Type theoretic semantics for semantic networks: an application to natural language engineering. Doctoral thesis, Durham University.

**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:

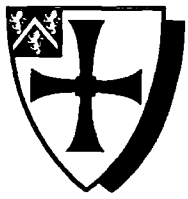
- a full bibliographic reference is made to the original source
- a https://etheses.durham.ac.uk/id/eprint/5397/ is made to the metadata record in Durham E-Theses
- the full-text is not changed in any way

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.

Academic Support Office, The Palatine Centre, Durham University, Stockton Road, Durham, DH1 3LE
e-mail: e-theses.admin@durham.ac.uk Tel: +44 0191 334 6107
https://etheses.durham.ac.uk

# University of Durham



# Type Theoretic Semantics for Semantic Networks: An application to Natural Language Engineering.

Simon K. Y. Shiu

*Laboratory for Natural Language Engineering*
*Department of Computer Science*

September 1996

Submitted in partial fulfilment of the
requirements for the degree of

Doctor of Philosophy

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.

4 JUL 1997

# Abstract

Semantic Networks have long been recognised as an important tool for natural language processing. This research has been a formal analysis of a semantic network using constructive type theory.

The particular net studied is SemNet, the internal knowledge representation for LOLITA[1]: a large scale natural language engineering system. SemNet has been designed with large scale, efficiency, integration and expressiveness in mind. It supports many different forms of plausible and valid reasoning, including: epistemic reasoning, causal reasoning and inheritance.

The unified theory of types (UTT) integrates two well known type theories, Coquand-Huet's (impredicative) calculus of constructions and Martin-Löf's (predicative) type theory. The result is a strong and expressive language which has been used for formalization of mathematics, program specification and natural language.

Motivated by the computational and richly expressive nature of UTT, this research has used it for formalization and semantic analysis of SemNet. Moreover, because of applications to software engineering, type checkers/proof assistants have been built. These tools are ideal for organising and managing the analysis of SemNet.

The contribution of the work is twofold. First the semantic model built has led to improved and deeper understanding of SemNet. This is important as many researchers that work on different aspects of LOLITA, now have a clear and unambigious interpertation of the meaning of SemNet constructs. The model has also been used to show soundess of the valid reasoning and to give a reasonable semantic account of epistemic reasoning. Secondly the research contributes to NLE generally, both because it demonstrates that UTT is a useful formalization tool and that the good aspects of SemNet have been formally presented.

[1]Large-scale, Object based, Linguistic Interactor, Translator and Analyser

# Acknowledgements

I would like to thank my supervisor Roberto Garigliano for his advice, support and intellectual stimulation throughout my time at Durham.

Secondly I am extremely grateful to Zhaohui Luo who has given significant time and help over the last two years.

Thank you to all members (past and present) of the Laboratory for Natural Language Engineering who have helped in numerous ways, most recently by commenting on various drafts of this thesis. Most significant to me has been the extremely pleasant social environment which I will sorely miss.

Finally I would like to thank Donna, for her continuing support "whatever I choose to do".

# Declaration

The material contained within this thesis has not previously been submitted for a degree at the University of Durham or any other university. The research reported within this thesis has been conducted by the author unless indicated otherwise.

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

# List of Figures

# Chapter 1

# Introduction

The subject of this research is knowledge representation (KR) for natural language engineering (NLE). More specifically how to reason about such representations in a formal manner. This introductory chapter is in two parts. The first part discusses methodological issues and assumptions. The general class of problems within which this research fits, and how, in general, success should be measured is described. The second part describes the specific problem addressed by this work, how it fits into the general class of problems and more specific success criteria are given.

## 1.1 Methodology

### 1.1.1 Natural Language Engineering

The building of large scale natural language systems involves the integration of a wide range of techniques and knowledge. In this respect it is a major task of engineering [Grishman, 1986], [Garigliano, 1995], [Smith, 1995], and hence the term NLE. It is clearly related to natural language processing, but differs, in the emphasis given to, or recognition of the need for, an engineering approach.

The objective is to engineer products which deal with natural language and

which satisfy the constraints in which they have to operate. This distinguishes the work from many works in computational linguistics which often emphasise investigating (i.e. verifying or falsifying) a particular linguistic theory.

[Nettleton, 1997c] outlines the major principles of NLE, for example, use of cost-benefit analysis, account taken of scale and resources, robustness, flexibility, openness and efficiency. In most cases these do not differ much from 'classical' engineering principles (as applied to, say, mechanical, civil or software engineering). The point in stating them is that it is felt they are often ignored in the field of NLP.

The starting point for an NLE project is the objective. This work is done under the umbrella of a larger NLE project, the objective of which is to build a core system with natural language competence, capable of supporting many different domains and applications. It is an assumption of this work that to do this requires the design of a KR language, used to represent 'knowledge'[1] of the 'real'[2] world devoid of linguistic aspects.

The solution to this NLE problem involves providing core mechanisms for retrieving and reasoning about the knowledge, and providing core mechanisms for converting to and from natural language statements.

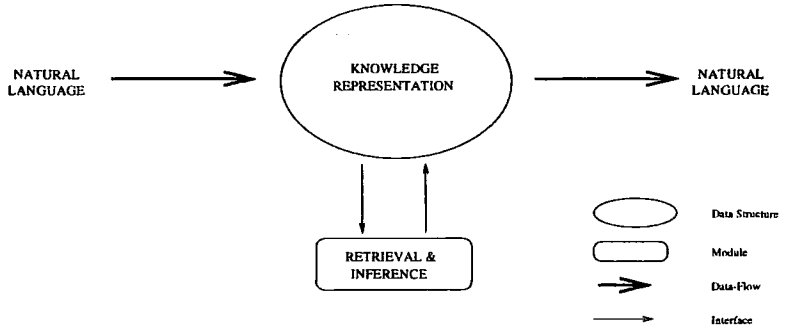



Figure 1.1: Core NLE system for various applications.

Natural language products are built from the core either by using the core

[1]The KR is considered as being used by an agent. Here, knowledge represents a set of statements, which this agent believes.

[2]Real in the sense that the agent using the representation believes it to be real

mechanisms or by interfacing with the KR directly. These requirements demand that the KR should be designed to:

1. be expressive enough for the applications. This varies between applications. The basic requirement is for first order quantification, but representations for time, location, causality and belief may also be required.

2. allow efficient conversion to and from natural language. The main work here is in applying techniques from the appropriate fields of parsing, semantic analysis, and generation. This is helped though if there is a clear relationship between linguistic structures and structures of the KR.

3. allow efficient and robust retrieval and inference of 'knowledge'. There are likely to be many types of inference, some used more often than others. The design should exploit this and be efficient in such cases.

4. be flexible enough to cope with different applications. This is tied to reuse and efficiency. If similar or related 'knowledge' is required for different tasks, it would be useful if they could both access the same structure in the way they need to. A less flexible representation may require duplicating the knowledge for each task.

5. be easily understood by developers working with it. This is key for developers of the core system but is also important when applications are built.

### 1.1.2 Semantic Formalisation

Formal semantics are provided by a mapping $\mathcal{M}$ between two structures.

$$\mathcal{M} : \mathcal{X} \longmapsto \mathcal{Y}$$

$\mathcal{X}$, often referred to as the syntactic or object language, is the language being

analysed. $\mathcal{Y}$, often referred to as the semantic language, usually has properties or structures which allow aspects of syntax (as interpreted by $\mathcal{M}$) to be analysed.

The syntactic language will usually have identifiable well formed formulae (wff's) and rules of inference or operations which define how new wff's can be derived from given ones, thus defining **proof** within the language. The semantic language will have some notion of **truth** so that the well formed formulae of the syntax map to statements that are either *true* or *false* in the semantics.

**Definition 1.1 (Soundness)** *A syntactic theory is sound with respect to its semantics if all provable formulae map to true statements in the semantics.*

**Definition 1.2 (Completeness)** *A syntactic theory is complete with respect to its semantics if all true statements in the semantics are provable in the syntax.*

If a theory is sound then proofs in the syntax are well founded (at least as far as the semantic language is well founded). If the theory is also complete then *proof* can be equated with *truth.*

A useful and well known example is $\mathcal{FOPC}$ which is usually interpreted into set theory [Davis, 1989]. This interpretation is not often challenged and so probably fits with most users' intuition of the meaning of $\mathcal{FOPC}$. Set theory is a well established mathematical theory and so showing the rules of inference to be sound and complete with respect to its interpretation is a meaningful result.

For this work, the syntactic language is the KR used in the parent project. This in turn means that the analysis and results will mainly be applicable to the parent project. However, the KR used does have similarities with a large proportion of other KRs. Moreover, the KR used is felt (informally) to meet many of the requirements for NLE listed above. Therefore the work will contribute to the field more generally by interpreting and analysing aspects and structures that are used by many representations and by showing, formally, how the KR achieves the above requirements.

The choice of semantic language will be determined by the following requirements:

1. 'Correctness' of reasoning results are required. This demands that the semantic language should be a well founded mathematical theory.

2. The language must be expressive enough to represent what the KR can represent. Ideally this should be done using standard features of the semantic language.

3. Because the KR is used as part of a large-scale engineering project (rather than, say a cognitive modelling project), it will be useful if the theory can assist in analysis and design of the KR.

Finally it should be noted that one of the principles of NLE is the appropriate use of the wide range of techniques that have been established in this field. This principle means that in many cases the 'semantics' or behaviour cannot be captured by a single simple theory. This is thought to be similar to natural language itself which is extremely difficult to pin down as a single theory (as many linguists and philosophers have found). This work aims for a framework theory which captures the way the basic mechanisms work and can account for many of the rich aspects of the KR in a coherent manner.

### 1.1.3 Methodological Success Criteria

The criteria given here are general and apply to any instance of the class of problems described so far. They are used as a starting point for the specific success criteria for this research, described in section 1.2.2.

#### A Formal Semantic model must be built

This will consist of:

- A formal *syntactic* definition for the KR. This should encompass what forms a legal *wff*, which subset of legal *wff*'s are in the KR, and how further *wff*'s can be infered from the KR.
- A *semantic* domain, where each *wff* will have a semantic denotation.

Robustness of reasoning requires that the rules (for valid reasoning) should be proven *sound.* To fully characterise the "inference" will also require *completeness.* Without this there will be statements which are entailed (according to the semantics), but which cannot be deduced using the syntactic rules.

KR's often provide plausible reasoning rules and heuristics. Soundness and completeness will not be relevant to these.

### Improved Understanding of the KR

The model built should improve the understanding of the KR. It is expected that the model should fit broadly with the intuitions of the KR developers and users. If it does not, then, improved understanding will come from analysing the discrepancy.

Assuming the model fits with the broad intuition of the KR developers and users, then the model will provide an unambiguous, formal way of understanding the KR. It will be used as a tool to investigate any properties claimed of the KR, in this case: 'expressiveness', 'closeness to natural language', and 'flexibility'.

### Generalising the results

Performing a formalisation of this size is a major task which is relevant to other NLE projects, therefore the methods and tools used will be of interest.

Secondly, assuming the KR has similarities with other representations, modelling it will be of interest to these similar representations.

## 1.2 LOLITA and SemNet

This research is performed as part of the LOLITA[3] research programme. LOLITA is a computer program built using natural language engineering principles. The aim of the project is to develop a core system capable of supporting a variety of natural language products [Garigliano *et al.*, 1993b], [Morgan *et al.*, 1995], [Smith, 1995].

The KR of LOLITA is a graphical representation, SemNet. SemNet shares many similarities with semantic networks [Lehmann, 1992]. Nodes and arcs of SemNet correspond to concepts and relationships. There are nodes for 'entities' and 'events' and they are organised into a hierarchy.

There are constructs for representing quantification, time and location, epistemic knowledge, events and causality. There are no primitive nodes and 'intuitively' the meaning of a node is defined by its relationship with other nodes. The full meaning of a node is only defined by the whole network.

### 1.2.1 The LOLITA architecture

In the operation of LOLITA, SemNet is used in many ways and each module makes assumptions about the meaning of sub-structures of SemNet (and to interpret the full meaning requires traversal of the full graph).

Figure 1.2 shows the architecture of the core of the LOLITA system and how some of its applications are built.

1. **The Inference Engine.** The inference engine retrieves and infers 'knowledge' contained in SemNet [Nettleton, 1997a]. The basic engine performs valid[4] inference, based mainly on inheritance. There are also modules for

[3]Large scale, Object based, Linguistic Interpreter, Translator and Analyser

[4]At least it is intended to be valid, of course an aim for this work is to show more formally that it is valid.

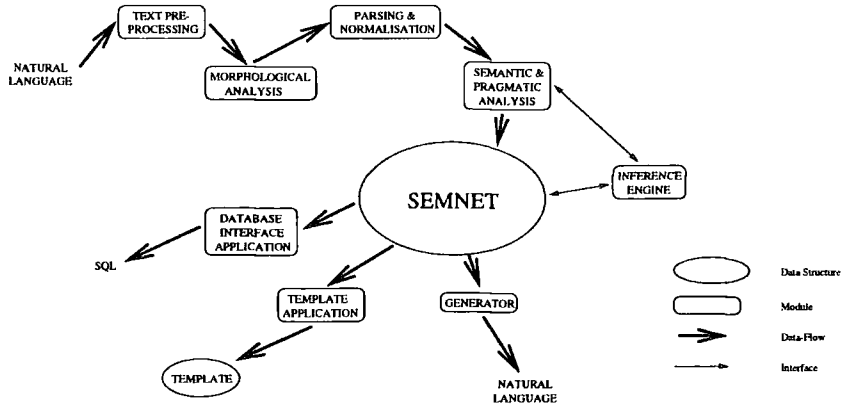



Figure 1.2: LOLITA architecture.

plausible reasoning about epistemic knowledge and by analogy.

2. **Syntactic Analysis.** Syntactic analysis corresponds to modules *text pre-processing*, *morphological analysis*, *parsing* and *normalisation* in figure 1.2. The combined modules transform free text into parse trees [Nettleton, 1997a]. The basic parser produces a large number of possible parse trees and there is a system of features and penalties which discard syntactically unlikely results. Normalisation removes redundant parse trees by converting them to normal forms (e.g. converting passive to active, dative to non-dative, and filling in implicit missing phrases).

3. **Semantic Analysis** Semantic analysis converts parse trees into SemNet structure [Short, 1996]. This means mapping the grammatical structures of the trees onto SemNet nodes. This involves determining if a node already exists in the network, and if it doesn't then also building it. This in turn requires search and inference (i.e. the inference engine) and an implicit interpretation of the meaning of constructs of SemNet.

   Once built, pragmatic analysis [Nettleton, 1997a] and a source control system [Bokma and Garigliano, 1992] are applied to see if the new knowledge should be accepted and with what level of belief.

4. **Generation** The role of generation is to convert SemNet structure into natural language. Because there is no surface linguistic information the generator

has to make many decisions about 'how' to realise statements.

Since each node is defined by the whole network, to 'realise' a node fully requires the whole of SemNet to be used. Indeed the whole of SemNet is passed as an argument to the generator [Smith, 1995]. However, clearly decisions have to be made about how much of a concept should be stated, and this in turn makes assumptions about the meaning of subsections of SemNet. These decisions are handled by the generator.

5. **Applications** The core LOLITA system has been applied in many situations [Nettleton, 1997b] including Chinese tutoring [Wang, 1995], dialogue [Jones and Garigliano, 1994], template filling [Costantino *et al.*, 1996], database interfaces [Garigliano *et al.*, 1995] and content scanning [Garigliano *et al.*, 1993a].

   The applications shown in figure 1.2 (database interfacing and template filling) both take 'knowledge' from SemNet and convert it accordingly. The same assumptions about interpretations of subsections of SemNet as used in generation are applied by these modules.

### 1.2.2 SemNet principles and features

SemNet is a graphical representation language. Nodes represent concepts and arcs represent relationships between them. There are nodes which represent entities and events. The events correspond to statements or propositions, and can be referred to just as any other nodes can. From now on this will be referred to as the *Propositions as nodes* principle. It is a principle that there are no pre-defined nodes in SemNet. The meaning of any node is determined by its relationship with other nodes/concepts, i.e. by its location, and consequently is only fully defined when the whole network has been interpreted. From now on this will be known as the *meaning as location* principle. All the nodes are organised into a hierarchy which allows for reasoning by inheritance.

The representation is free from linguistic styles and features. It has been designed using engineering principles to support the objectives of LOLITA. There is no claim, or attempt to model cognitive behaviour.

The above architecture description has shown that SemNet and its intuitive semantics are fundamental to LOLITA. This means that there are many subjective views of the 'meaning' of SemNet structures, which can lead to incorrect assumptions and code.

Formal semantics will remove this subjectivity and improve the general understanding of SemNet. Furthermore a formal model will allow formal analysis of some of the features which, in section 1.1, were said to be required of a KR for NLE.

For example, flexibility was listed as a key requirement. Clearly SemNet is being used in many different ways throughout the LOLITA system, and so intuitively must be flexible. By providing formal semantics it should be possible to add some formality to this concept.

It is outside the remit of this thesis to try to implement changes to LOLITA based on the results of this work. The aim is quite specifically to understand what is there and to make suggestions based on theoretical analysis. Decisions to change LOLITA can then be made based on both the theory and pragmatic (engineering) considerations.

### 1.2.3 Project Aims and success criteria

This section takes the general criteria given in section 1.1.3 and makes them specific to this research.

#### Build a formal semantic model of SemNet

This must include:

- **A syntactic definition of SemNet**

To analyse the reasoning procedures there must be a clear definition of what constitutes a legal SemNet, which syntactic structures form legal *wff*'s, how it is determined which *wff*'s are in a SemNet and what rules can be applied to allow new *wff*'s to be 'infered' from a SemNet.

To allow analysis of concepts such as: 'closeness to natural language', 'flexibility', and '*meaning as location*' it will be necessary to clearly identify 'syntactic substructures' which combine to form the *wff*'s.

- **Semantic Domain and Denotations**

  An appropriate semantic domain must be given. This domain must have a mathematical foundation which can be 'trusted'. It must also be expressive enough analyse structures purporting to represent a significant subset of natural language.

  Each syntactic structure should have a semantic denotation, and each *wff* should be interpretable as either *true* or *false*.

### Meta theoretic results

The valid reasoning is mainly based on inheritance. The inheritance algorithms are used in various guises throughout the core. Therefore it is extremely important to show that inferences drawn are indeed valid. A soundness proof will provide this (relative to the semantic domain).

To show that the semantics fully characterise the valid reasoning will require a completeness proof. Whilst this will help understanding it is not as useful as a soundness proof, and also might not be possible.

### Investigate the 'good features' of SemNet

- **SemNet structure is close to natural language structure**

  To test this a comparison between the interpretation for different aspects of

SemNet and formal semantics for the 'equivalent' natural language statement will be performed. As discussed in 1.3 this is not entirely satisfactory because there is no agreed formal semantics for natural language and the usage of 'equivalent' is quite vague. However, a crude analysis should be possible.

- **Meaning as Location**

  Is an interpreted node only fully defined when the whole network has been interpreted, and is the manner in which 'meaning' are built in SemNet reflected in the model.

- **Show how SemNet represents complex expressions.**

  It is claimed that SemNet can express and reason about epistemic knowledge, i.e., statements about LOLITA's own beliefs and statements about other agents beliefs. Also that sentences with complex anaphora and quantification can be represented.

  To show 'how' SemNet represents complex expressions it must first be established that it can represent such expressions. To do this they must of course be representable in the semantic language. To test that the meanings really have been captured the inferences and consequences of such statements should be shown to follow.

  To go further and show 'how' SemNet is representing this knowledge there must be a similarity of structure between SemNet and the semantic language. This will be judged subjectively again based on a comparison with natural language semantics and the subjective opinions of LOLITA's designers.

- **Show flexibility of SemNet**

  From the description given, it is clear that SemNet is being used in many ways, and so is presumably flexible. A successful result will be a formal property of the representation that shows if[5]/how it can be used in these

[5]In the case where there is a theoretical problem with the different ways modules use SemNet, the property should be able to highlight this problem.

different ways.

#### Improve developer comprehend-ability

A final measure of usefulness to the LOLITA project is to see how useful the semantics are for designing and using SemNet, i.e. will the designers use the model or will they continue to use their intuitive interpretations.

#### Extract aspects of the formalisation relevant to NLE

As will be discussed in chapter 2, there has been a lack of formal descriptions for semantic networks. Therefore this work will already contribute to the wider community by showing that such formalisations can be done, and that doing so develops intuition. It is thought that there are aspects of SemNet which are useful for NLE. It is thought that the formalisation will show why this is the case, e.g. how it represents complex expressions and how it achieves flexibility.

If the formalisation is successful then there will be some value in examining the methods and tools used to perform the evaluation.

## 1.3 Logical Progression of the Thesis

**Chapter 2** reviews related work. A brief review of research issues in semantic networks is given. After this a critical review of the formal approaches taken for modelling and reasoning about these networks is given. Aspects of networks which are similar to, or held by SemNet are highlighted.

**Chapter 3** gives an overview of constructive type theory. The chapter works chronologically ending with Luo's UTT [Luo, 1994], which is the version of type theory used for the formalisation. Type theory has many applications in computer science and because of this many tools and techniques (such as machine assisted

proof development, and modular reasoning) have been developed which can be used to help manage a formalisation task such as this one. A review of the work of Ranta [Ranta, 1994] is also given, which shows how constructive type theory has aspects for analysing and reasoning about natural language directly.

**Chapter 4** is a detailed description of the principles and design of SemNet. Although placed before the 'semantic model' chapters it is considered a part of the contribution of the thesis. The issues raised and the clearness of the description only came as a result of the formal analysis described in later chapters. An informal discussion of the good aspects (for KR and NLE) is given. The later chapters formalise these aspects further.

**Chapter 5** follows the methods of other researchers in building a semantic model using classical techniques such as set theory and possible world semantics. Many of the 'correctness' problems can be addressed by this model. However, it is argued that there are problems in manageability and differences in structure which limit the usefulness of this model.

**Chapter 6** describes the framework of the type theoretic formalisation. A description of how the tools and techniques developed for type theory are used is given. Soundness results for valid reasoning and a discussion of how 'similar' the model can be made. A further aspect is that since type theory is a programming language an analysis of the implementation can also be given. Finally a discussion of the usage of Lego (a proof assistant based on type theory) is given.

**Chapter 7** concentrates on showing the similarity in structure between SemNet and the type theoretic semantic model. Specific issues addressed are a better coverage of intensionality and complex quantification sentences.

**Chapter 8** starts with an evaluation based on the criteria of section 1.3. Final conclusions are drawn and a discussion of possible further work is given.

# Chapter 2

# Formal Semantics for Semantic Networks

This chapter begins with a short history of semantic networks and concludes with networks that are similar to SemNet. An overview of general research themes and a more detailed coverage of three well known network representations is given. The review concentrates on work related to formal understanding of network representations. Each system covered is compared with the main principles of SemNet.

## 2.1 Issues in Semantic Networks for NLE

Semantic networks with the *meaning as location* principle, date back to Quillan's work in the 60's [Luger and Stubblefield, 1993] pp 360. English words were defined (like a dictionary) in terms of other words (words being nodes in a network) and the meaning rather than involving primitives is defined by its location. A user determines the meaning of a word by traversing the (perhaps circular) graph until they are satisfied that they have understood the meaning of the original word.

This early work established ideas such as 'labelled arcs', hierarchical inheritance and inference by graph traversal. Since then many systems have been defined and

implemented with a variety of definitions and principles [Lehmann, 1992]. They have been popular for NLP research since they are more readable and intuitive than classical logic, and (seemingly) richly expressive and efficient. A continuing theme is the need for formal semantics for these different schemas [Woods, 1975] [Woods, 1991].

[Schubert, 1991] makes a call for a recognition of the fact that most networks (if not all) are merely notational variants of first order logic. However, it is accepted that networks organise 'knowledge' to allow certain types of inference procedure (i.e. inheritance) to be performed efficiently. It is a claim of this work that networks can also be organised to allow certain types expressions to be easily stated, and/or more flexibly available than first order logic. (both of which are specifically required by NLE). SemNet fits this category and formalisation will help to show the essence of the 'organisation' needed to achieve the required behaviour.

### 2.1.1 Defeasible Inheritance

Many inheritance based network representations have developed schemes and rules for handling defeasible inheritance. Defeasible inheritance allows plausible inferences to be drawn. Consider the following example[1]: from the inheritance network in figure 2.1 it can be inferred that:

1. Tweety is a bird and so can fly.

2. Tweety is a penguin and so cannot fly.

The immediate reaction to this is to consider that the network is inconsistent, and so useless. The problem is that the first conclusion, although invalid, is in many cases plausible and useful to draw. Many inheritance systems resolve this by allowing both conclusions to stand, but that if they both appear, the inference

[1]The bird/penguin problem appears many times in the literature and is a standard way to present the defeasible inheritance problem.

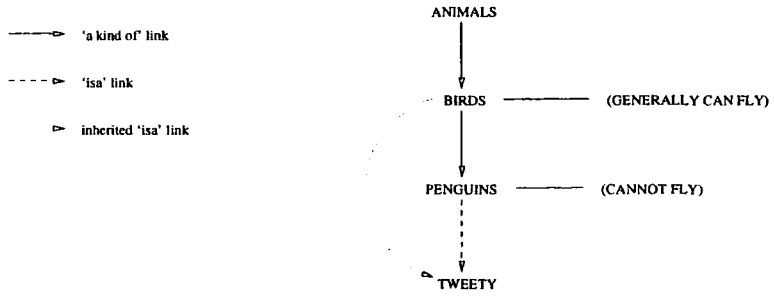



Figure 2.1: The Bird/Penguin problem

based on the most specific information would stand. In this case, since being a penguin is more specific than being a bird, the second conclusion would stand.

There has been much research both formalising this type of reasoning [Touretzky, 1986], [Fahlman, 1979], and relating it to non-monotonic logics [Etherington, 1988] [Froidevaux and Kayser, 1988] [Shastri, 1988]. SemNet does not use such formulations for plausible reasoning [Long and Garigliano, 1994] and so such works are not yet relevant to this research.

## 2.2 KL-ONE

As has already been mentioned, Woods [Woods, 1975] made a call for a more formal description of network constructs. For example, it should be made clear whether the graph shown in figure 2.2 is making the assertion 'telephones are black, or if is defining the concept of black telephones.

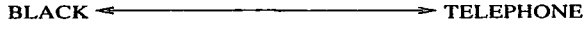



Figure 2.2: 'Black telephones' or 'All telephones are black'

Brachman [Brachman and Schmolze, 1985], in part response to this, developed his theory of structured inheritance (SI) nets. This theory was adopted by the KL-ONE project for natural language understanding.

To explain the theory of KL-ONE it is first necessary to explain frame based systems [Luger and Stubblefield, 1993]. A frame based system consists of data

structures organised into a hierarchy via ako (a kind of) and isa links. Data structures are added to the hierarchy manually. The semantics of the hierarchy is defined by the inheritance algorithms that operate on it. There are no clearly specified criteria for when and where a structure could be added to the hierarchy. This meant that data had to be entered manually by a human expert, and worse still as the network becomes more complicated it becomes harder to understand all the ramifications of extending the network in a particular way.

KL-ONE changed this by insisting that there should be a 'criterial semantics' for the structures of the hierarchy. The inheritance operations would then have to be justified with respect to these criterial semantics. This meant that concepts could be automatically classified into the hierarchy (KL-ONE was the first inheritance network to achieve this [Brachman *et al.*, 1991]). Later work went further than demanding criterial semantics and insisted on a model theoretic understanding for the language [Woods and Schmolze, 1992]. For the basic hierarchy of concepts and roles this was done by postulating a domain of individuals $\mathcal{D}$, and specifying an function[2] $\xi$ which maps concepts to subsets of $\mathcal{D}$ and roles to subsets of $\mathcal{D} \times \mathcal{D}$. The top concept mapped to $\mathcal{D}$, and the top role to $\mathcal{D} \times \mathcal{D}$. Basic concepts lower down the hierarchy are defined by a role, an existing concept and a quantification symbol. Such nodes are interpreted in terms of the interpretations of these concepts. For example, a new concept "$(\forall, r, c)$" is interpreted as:

$$\{x \in \mathcal{D} \mid \forall y (x, y) \in \xi(r) \rightarrow y \in \xi(c)\}.$$

Informally, the concept is defined as the set of objects that are in the role 'r' with all objects in 'c'. There is a problem with this formulation when there is a cyclic/recursive dependency between definitions, for example if the interpretation of 'c' were defined in terms of the concept $(\forall, r, c)$. Nebel [Nebel, 1991] provides an analysis of different methods of providing semantics for this situation, including

[2]The function given is overloaded, in the sense that it takes objects of different types.

fixed point methods, but ends concluding none of the methods analysed is obviously superior.

Later work maintained the importance of the distinction between definition and assertion. Indeed most KL-ONE systems are distinguished by having two separate knowledge bases: a terminology box (T-box) containing the hierarchy of concepts which make up the definitions of concepts, and an assertional box (A-box) usually made up of first order statements based on concepts defined in the T-box. Various papers have been published on the issue of how best to integrate the two types of knowledge: Frisch [Frisch, 1991] describes a substitutional framework in which the T-box is used to provide sortal information for the more standard first order theorem proving algorithms. Work based on the LILOG project [Herzog and Rollinger, 1991], argues that for natural language understanding applications data is often updated in both 'boxes' and so a closer coupling of information between the 'boxes' is needed [Beierle *et al.*, 1992].

There have been attempts to extend the basic KL-ONE language with modal statements. [Graber *et al.*, 1995], uses them to integrate knowledge and belief operators. For example, if $wff$ is a statement (from either the T-box of the A-box) then:

$$\Box^{K}_{joe}\Box^{B}_{chris}wff$$

would represent the statement "(agent) joe knows that (agent) chris believes $wff$". This extension is given a Kripke style possible worlds semantics [Meyer and der Hoek, 1995]. Soundness and decidability of a reasoning algorithm are shown.

### 2.2.1 KL-ONE and SemNet.

There are many similarities between KL-ONE and SemNet. In particular the T-box defines concepts in a manner similar to SemNet (in terms of its position in the

hierarchy and the events it is involved in). Indeed there are very close similarities between the model described for KL-ONE and the set theoretic semantics given for SemNet 'entity' nodes in chapter 5. A problem with the SemNet model (and it is presumably also a problem for the KL-ONE model) is that the model is extensional and so does not distinguish 'intensionally' distinct concepts.

A major difference is that in SemNet statements (both taxonomic and assertional) are represented by nodes. These are used to represent epistemic assertions directly, which is quite different from adding modal operators. Nevertheless, a similar style possible worlds semantics have been developed as an attempt to model these assertions.

## 2.3 Conceptual Graph Theory

Sowa [Sowa, 1984] introduced Conceptual Graph Theory (CGT) as a "natural" formalism for representing knowledge. Since then many researchers have used conceptual graphs as a starting point for knowledge based and natural language processing systems. Different problems and issues have led to a wide range of research [Nagle *et al.*, 1992] including:

- Expressiveness.
- Hierarchical reasoning.
- Representing temporal knowledge.
- Using CGT for NLP, i.e. parsing and semantics analysis into CGT, and generating free text from CGT.

### 2.3.1 CGT basics

The basic system can be viewed as a sorted version of the graphical logic of Charles Sanders Peirce.

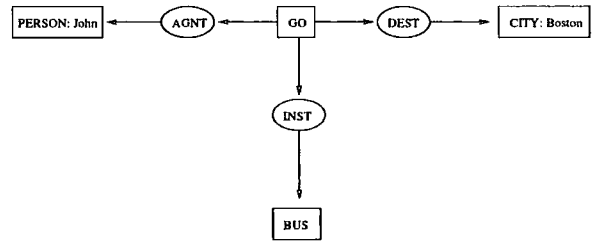



Figure 2.3: CGT for 'John is going to Boston by bus'

Concepts represented by rectangles (see figure 2.3) map to monadic (1-argument) predicates and concepts represented by ellipses map to relations with as many arguments as there are arcs emanating from the corresponding node. The graph shown in figure 2.3 maps to the $\mathcal{FOPC}$ formula:

$$\exists x.\exists y \quad person(John) \wedge go(x) \wedge city(Boston) \wedge bus(y) \\ \wedge agnt(x, John) \wedge inst(x, y) \wedge dest(x, Boston)$$

There is a labelling system (labels on the concept) which allows concepts to be referred to in different ways. The concepts are organised into a type hierarchy, formally understood as a lattice. Types lower down the hierarchy are defined by $\lambda$-abstractions over types higher up. There has been work addressing reasoning on the type hierarchy and how to couple the definitional and assertional information which is connected to the KL-ONE type research.

CGT has a system of contexts, with a context representing a situation as in the situation calculus [Barwise and Perry, 1985]. The contexts can represent negation, modality and epistemic relations.

### 2.3.2 CGT formal aspects

The basic logic of CGT is isomorphic to $\mathcal{FOPC}$. It therefore shares the same (set theoretic) semantics and is sound and complete. The concept hierarchy is seen as adding sorts to the language, which, as shown by Walther [Walther, 1987], can

significantly improve the efficiency of inference. In CGT the hierarchy is defined intensionally (through $\lambda$ abstractions) but there seems to be no formal semantic modelling of this aspect.

The system of contexts is based on the situation calculus, but again there seems to be no formal interpretation of constructs in to this language

### 2.3.3 CGT and SemNet.

There are many aspects of CGT which are similar to SemNet. For example, the type hierarchy being defined by definitions is similar to SemNet. Moreover, since they are defined by $\lambda$ abstractions, an obvious continuation is to model concepts as types (see next chapter).

The system of contexts is quite different from SemNet which if adopted, as discussed in section 4.5, would break distributedness. Since it is based on the situation calculus, this is clearly the semantic language which should be used to understand it. However, this does not mean that situation calculus will be the best tool to model the epistemic aspects of SemNet.

A final difference between the languages is that CGT reflects linguistic structure more closely by using separate items to refer to the same concept (i.e. to model anaphora). This is done using a system of co-reference links (drawn as broken lines) which show when two apparently different nodes, actually refer to the same concept.

## 2.4 SNePS

SNePS (for Semantic Network Processing System) [Shapiro and Rapaport, 1987], [Shapiro, 1979], is a semantic network language with facilities for building semantic networks. There are further facilities for retrieving and inferring information from the networks. Users can interact with SNePS in a variety of ways including an

extendible fragment of English. It has been used in many applications including cognitive modelling and computational linguistics.

### 2.4.1 SNePS basics and principles.

The first principle of SNePS is that it is a *propositional* network [Kumar and Chalupsky, 1993]. This means that all information, including propositions are represented by nodes. In KL-ONE propositions were represented by formulae, and in CGT by 'proposition contexts'.

There is a principle that unique concepts are represented by unique nodes. This means that nodes represent intensional objects.

#### SNePS syntax.

Nodes are structured by the arcs which emanate from them. They are divided into atomic nodes (no arcs emanating from them and so they have no structure):

- *sensory* nodes, which represent interfaces with the real world. Typically words used in some language.
- *base* nodes represent constant individuals.
- *variable* nodes represent arbitrary individuals and propositions.

and molecular (structured) nodes:

- structured individuals
- structured propositions.

Figure 2.4 shows how SNePS builds up complex concepts. Intuitively m9 represents the concept 'yellow' as indicated by its connection to the *sensory* node

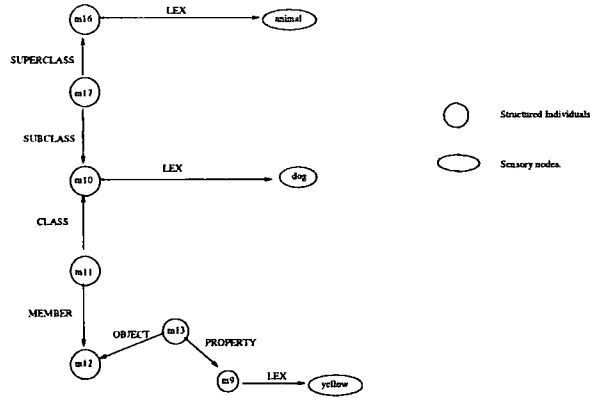



Figure 2.4: Example SNePS graph for concept 'a yellow dog'.

'yellow'. m12 represents the concept 'an individual yellow dog' and can be referred to by other (propositional) nodes.

There are various reasoning mechanisms based on this representation, including reduction and path based inference [Shapiro, 1991]. Both of these are described as modelling subconscious reasoning, since they allow virtual arcs to be inferred, through the presence of others. For example, a MEMBER arc can be inferred (by reduction) to 'virtually' occur between m12 and m16 in figure 2.4. This is basically inheritance, although it is claimed that it is more natural to consider them in this way. The reasoning mechanisms are given a set theoretic interpretation.

## ANALOG

ANALOG (for A NAtural LOGic) [Ali and Shapiro, 1993] extends the SNePS representation with structured (molecular) variables.

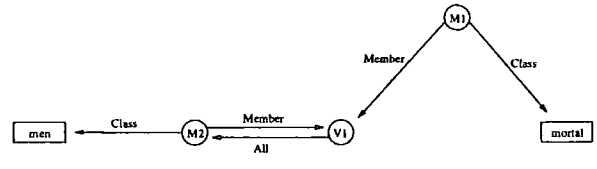



Figure 2.5: ANALOG graph for 'All men are mortal'.

The node 'V1' is a variable node, the all arc provides the quantification information. The representation is described as being close to natural language, providing

a direct mapping for anaphora (see the representation of the 'donkey sentence' in chapter 4) and being able to represent 'branching quantifiers' such as those involved in the sentence: "Some relative of each villager and some relative of each townsman hate each other.

### 2.4.2 Formal semantics for SNePS

The semantics of SNePS are mainly based on Meinongian's theory of objects [Rapaport, 1981]. A Meinongian object is an object of thought. To give an example, taken from [Rapaport, 1981],

> Suppose, e.g. I am thinking that the person in the next room is happy. If there is not such a person, then I am thinking at most of a Meinongian object; if there is such a person, then there is - in addition - an actual object. ... Let us say that actual objects "exemplify" properties while Meinongian objects are "constituted" by properties.

It is held that any 'thing' which is an object of language (e.g. noun phrases or sentences) must be an object of thought. [Shapiro and Rapaport, 1987] describes a mapping of the nodes of SNePS into Meinongian objects, thus building up a formal intensional description of the network.

The interpretation does not cover base nodes, which are presumably considered atomic. Hill [Hill, 1994] argues that base nodes should not be treated in this way. Indeed, an argument is presented which discusses how the 'meaning as location' principle of SNePS should be captured in the semantics, and that base nodes both influence nodes that point to them and have intrinsic meaning themselves. An elegant of interpretation these nodes as non well founded sets[3] [Aczel, 1988] is given.

[3] A modern version of set theory, which rejects the 'well founded' axiom and accepts sets that can be members of themselves.

### 2.4.3 SNePS and SemNet

There are many similarities between SNePS and SemNet. Most pertinent is the *proposition as nodes* principle, which correspond directly with SemNet events. Also similar are the principles of uniqueness and 'meaning through location.

SNePS has been designed to be cognitively realistic, rather than as an engineering system. The semantics defined is used to understand the network from this point of view. SemNet differs in being designed with pragmatic and application relevant aspects (e.g. flexibility and efficiency). It is not clear how Meinong semantics could help in this endeavour.

## 2.5 Review

Semantic networks have a relatively long history in artificial intelligence, although their actual distinction from other knowledge representation languages (specifically first order logic) is disputed. Different versions with different motivations have been built, each sharing many underlying themes.

Formal aspects and semantics have been applied in differing amounts to each of the main systems, and it is through these that differences in structure and assumptions become apparent.

SemNet shares the underlying themes, and also has aspects in common with each of the main systems described. It does have its own motivations and distinctions, which are discussed in detail in chapter 4. Nevertheless it is clear that formal semantic analysis of SemNet is of wide interest to the semantic network community.

# Chapter 3

# Constructive Type Theory

SemNet is formalised in the constructive type theory UTT (Unified Theory of Types). As discussed in chapter 1, UTT is used to define the syntax and semantics of SemNet. This chapter is a self contained introduction to constructive type theory and UTT which explains all the features relevant needed in this work.

## 3.1 Introduction

Type theory is first understood in terms of two universes, one of 'objects' and one of 'types'. A 'judgement' of type theory is a statement of the form:

$$a : A$$

which should be read as 'object a is of type A'. The simplest type constructor is the function constructor $\rightarrow$. Given two types A and B, a new type A $\rightarrow$ B can be formed and its objects will be functions that map objects of type A to objects of type B.

Some principles of type theory (as it is used here) are that

1. Every object has a unique type.

   For example, if the object '4' is considered to be of type $\mathcal{N}$ (the type of natural numbers), then 4 cannot also be considered to be of type $\mathcal{R}$ (the type of rational numbers).

2. Types are understood extensionally, i.e. if two types are inhabited by the same objects then they are the same type. However, functions are intensional, in that they reflect the computational behaviour of the function.

   This is different to set theory, where functions are represented by relations (i.e. extensional sets).

The language is clearly more restrictive than set theory, which allows objects to be members of any number of sets. By being more restrictive it becomes more manageable and so lends itself to many applications.

### 3.1.1 Type Theory as a Programming Language

The notion of computation is primitive in type theory. Function types are inhabited by lambda abstractions. For example an object of type $A \rightarrow B$ (the type of functions from objects of type A to object of type B) is an abstraction of the form $\lambda x : A.B$. Computation is defined by $\beta$ reduction [Hindley and Seldin, 1986].

**Definition 3.1.1 ($\beta$ reduction)** *Any term of the form*

$$(\lambda x : A.M)N$$

*(with A a type and M and N terms) is called a redex.*

$$[N/x]M$$

*is called its contractum.*

$$(\lambda x : A.M)N \rhd_\beta [N/x]M$$

*defines 1-step $\beta$ reduction.*

**Definition 3.1.2 (Computational Equality)** *Computational equality is defined as the transitive, reflexive and symmetric closure of 1-step $\beta$ reduction.*

$$A \cong B$$

*should be read as A is computationally equal to B.*

**Definition 3.1.3 (Church-Rosser)** *Any two computationally equal terms can be reduced to a common term:*

$$\forall M_1, M_2.(M_1 \cong M_2) \Rightarrow \exists M.(M_1 \rhd M) \wedge (M_2 \rhd M)$$

**Definition 3.1.4 (Subject Reduction)** *Computation is type preserving.*

**Definition 3.1.5 (Normal form)** *A term is in normal form if and only if it contains no redexes, i.e. it can only compute to itself.*

**Definition 3.1.6 (Strong Normalisation)** *Every computation starting from a well typed term terminates, i.e. reaches a normal form.*

Different formulations for type theories can be given (as described later in this chapter). For each it is extremely useful if the above properties can be established as they will lead to many desirable properties such as decidability, manageability and implementability.

### 3.1.2 Type Theory as a Logical Language

A key turning point for type theory is the **proposition as types** principle, as discovered by Curry [Curry and Feys, 1958] and Howard [Howard, 1980].

The idea is that any proposition P corresponds to a type Prf(P), and a proof of P corresponds to an object of type Prf(P).

For example, there is a correspondence between function type symbol and the implication symbol of (intuitionistic) logic. If $A$ and $B$ are types, then given objects of $a : A$ and $f : A \to B$, an object of type f(a) of type B can be constructed. This coincides with viewing $A$ and $B$ as propositions, a as a proof of $A$, f as a proof of $A \to B$, and being able to derive (or construct) a proof of $B$.

Again different formulations of type theory will lead to different logics, with further consequences for decidability and manageability.

### 3.1.3 The dependent type constructors

Constructive type theory allows richer type constructors. Two important ones are the 'dependent product type' and the 'dependent strong sum type'.

#### The dependent product type

The dependent product type has functions as objects. For a type A and any family of types B[x] indexed by arbitrary objects x of type A, Πx:A.B(x) is the type of functions 'f' such that for any object a of type A, applying f to a yields an object of type B[a]. The term dependent is used since the type of the resulting object 'depends' on the object the function is applied to. Intuitively it represents the set of (dependent) functions from A to B[x]:

$$\{f \mid \forall a : A.f(a) : B[a]\}$$

If the resulting type is always the same, say B, for all objects a, then the function type simplifies to the type A $\rightarrow$ B. As an example of this concept, consider polymorphic functions for programming languages. The polymorphic equality function Eq has type:

$$Eq : \Pi A : Type.(A \rightarrow A \rightarrow bool)$$

i.e. Eq takes a type as parameter and returns an equality function for that type, but the type of this function 'depends' on the type (object) passed. Here the parameter used to index the 'range' types is a type, but this need not be the case for dependent types.

### The Strong Sum Type

Strong sum types are types of pairs of objects. For any type A and any family of types B[x] indexed by arbitrary object x of type A, $\Sigma$x:A.B(x) is the type of pairs (a,b) where a is an object of type A and b is of type B[a]. Intuitively it represents the set of (dependent) pairs of elements of A and B[x]:

$$\{(a, b) \mid a : A, b : B[a]\}$$

The projection functions

$$\pi 1 : (\Sigma x : A.B(x)) \rightarrow A$$
$$\pi 2 : \Pi z : (\Sigma x : A.B(x)).B(\pi 1(z))$$

extract the first and the second entry of a pair, respectively. For example, if g:($\Sigma$x:A.B(x)), then ($\pi$1 g):A. Because of the inherent dependency they are useful for describing complex types.

### 3.1.4 Inductively Defined Types

Types can be defined inductively. This is done by giving constructors to say how objects of this new type can be formed. By making the definition inductive we are saying that this is the only way in which canonical objects of this type can possibly be formed, for example, the three judgements:

$$\begin{bmatrix} N & : & Type \\ 0 & : & N \\ succ & : & N \rightarrow N \end{bmatrix}$$

defines the type of natural numbers. By definition, the rules define, exhaustively, how objects of type N (natural numbers) may be constructed. Because the rules are exhaustive, an associated elimination rule can be inferred.

$$\begin{aligned} Nelim: \quad & \Pi C : N \rightarrow Type. \\ & C\ (0) \rightarrow \\ & (\Pi x : N. C\ (x) \rightarrow C\ (succ\ (x))) \rightarrow \\ & \Pi n : N\ (C\ (n)) \end{aligned}$$

This gives a method for defining functions that operate on all objects of the inductive type. For example, in this case functions for addition, subtraction and multiplication can be defined. This method can be used to prove theorems about this type N. It turns out that Peano axioms can be proved for this type.

## 3.2 Some versions of Constructive Type Theory

There are different versions of constructive type theory. The main differences are reflected in the different structures of their conceptual universe of types.

### 3.2.1 Martin-Löf's (predicative) type theory

Perhaps the best known is Martin-Löf's type theory [Martin-Lof, 1984] [Nordstrom *et al.*, 1990]. This can be understood in a hierarchical way: one starts by introducing various basic types (e.g. finite types, and natural numbers), and using type constructors (in this case the dependent types $\Pi$ and $\Sigma$), builds up more complex types, until finally, one may introduce a type universe, which is the type consisting of (the names of) each of the types so far introduced. Continuing in this way a sequence of type universes can be built up Type(0) : Type(1) : Type(2) : ... There is not a type of all types, instead such a type is approximated by the (infinite) sequence of universes.

In this theory types are not distinguished from propositions. Since there is not a type of all types it is not possible to quantify over all propositions (although quantification over any of the type universes is allowed).

The theory has been used as a foundational language for constructive mathematics [Bishop, 1967]. Universal quantification is represented by the $\Pi$ constructor and existential quantification by the $\Sigma$ constructor. This 'strong' notion of existential quantification ensures that an object can alway be extracted by the projection function $\pi 1$. For example, in mathematics the statement

$$\exists n : N.Prime(n) \wedge Even(n)$$

is informally read as 'there exists a natural number which is both even and prime'. Whereas when the strong existential is used, i.e.

$$\Sigma n : N.Prime(n) \wedge Even(n)$$

the statement can only be realised (proved) by a pair of objects, one of which is a natural number and the other being a proof that this number is both prime

and even. For example,

(2,Proof that 2 is even and prime)

### 3.2.2 Impredicative type theory

The above theory was predicative, in that there is no type which allows quantification over itself leading to the new object of the same type, i.e. no type T such that:

$$(\Pi t : T.A) : T$$

Such an idea is incorporated in the polymorphic types of some lambda calculus.

#### Polymorphic $\lambda$-calculus

The $\lambda$-calculus in its original form is untyped. Any term can be applied to any term to derive a new object. This section discusses Church style typed systems[1].

Type systems are distinguished by which constructors are allowed into the theory. The simplest version of *lambda*-calculus is $\lambda \rightarrow$. The only type constructor for this theory is $\rightarrow$. The type formation rule is:

$$\frac{\Gamma \vdash A : Type,\ \Gamma \vdash B : Type}{\Gamma \vdash (A \rightarrow B) : Type} \quad \textit{Type formation}$$

and it is defined by the following introduction and elimination rules:

$$\frac{\Gamma \vdash M : (S \rightarrow T), \Gamma \vdash N : S}{\Gamma \vdash (MN) : T} \quad \rightarrow\textit{-elimination}$$

[1] As opposed to Curry style type systems [Barendregt, 1992].

$$\frac{\Gamma, x : S \vdash M : T}{\Gamma \vdash (\lambda x : S.M) : S \to T} \quad \to\textit{-introduction}$$

More generally functions are defined as the dependent product types. A new sort 'Kind' is introduced where Type resides, i.e. Type:Kind. General Type/Kind formation rules are of the form :

$$\frac{\Gamma \vdash A : s1,\ \Gamma, x : A \vdash B : s2}{\Gamma \vdash (\Pi x : A.B) : s3} \quad \textit{Type/Kind formation}$$

where s1, s2 and s3 range over Type and Kind. Allowing the various combinations as rules leads to different theories (and their corresponding logics). Barendregt [Barendregt, 1992] gives an elegant discussion of some of these systems showing how they form a cube of type systems, with $\lambda \to$ at the base and the calculus of constructions at the peak, see figure 3.1. Each orthogonal direction represents the inclusion of one of the above formation rules.

For example $\lambda 2$ has the rule:

$$\frac{\Gamma \vdash A : \mathit{Kind},\ \Gamma, x : A \vdash B : \mathit{Type}}{\Gamma \vdash (\Pi x : A.B) : \mathit{Type}}$$

This allows the judgement:

$$(\Pi A : \mathit{Type}.A \to A) : \mathit{Type}$$

and therefore introduces impredicativity. It corresponds to the 2nd order typed $\lambda$ calculus [Girard, 1986].

$\lambda$P adds the rule:

$$\frac{\Gamma \vdash A : Type,\ \Gamma, x : A \vdash B : Kind}{\Gamma \vdash (\Pi x : A.B) : Kind}$$

corresponds with predicate logic and is used as the basis for the AUTOMATH project [de Bruijn, 1980].

$\lambda\omega$ adds the rule:

$$\frac{\Gamma \vdash A : Kind,\ \Gamma, x : A \vdash B : Kind}{\Gamma \vdash (\Pi x : A.B) : Kind}$$

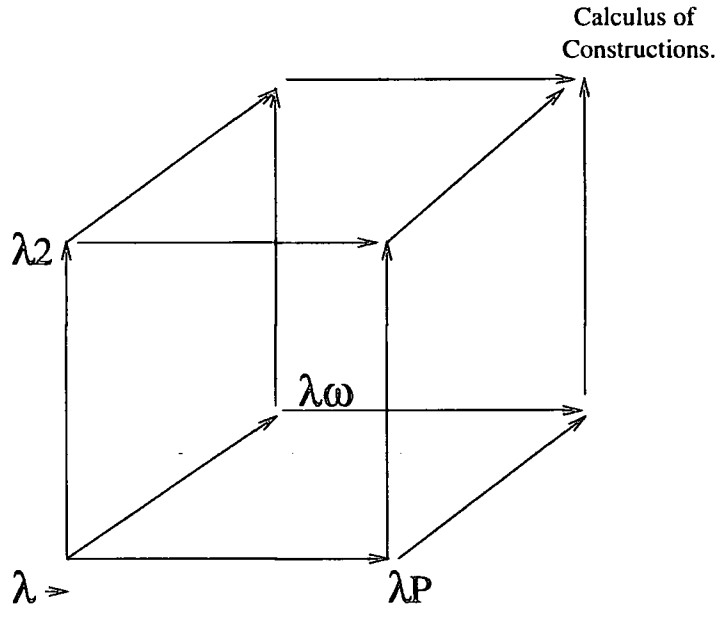



Figure 3.1: The $\lambda$-cube.

### Coquand-Huet's calculus of constructions

In the calculus of constructions (CC) there is a type *Prop* corresponding to the type of propositions (i.e. not all types are propositions). Furthermore this type is impredicative, e.g. $\Pi.P : Prop.P$ is also a proposition (object of type *Prop*). This allows for the notion of predicates and relations over types, e.g. $Pred : A \to Prop$

and $Relation : A \rightarrow A \rightarrow Prop$. The CC corresponds to the intuitionistic higher order predicate logic. Dependent function types are present but the dependent sum type is not.

### 3.2.3 Luo's ECC and UTT

The Extended Calculus of Constructions (ECC) was developed in Luo's PhD thesis [Luo, 1990]. It can be viewed as a unification of Martin-Löf's type theory and CC. ECC extends CC with the notion of type universes and the $\Sigma$ type, and it extends Martin-L"of's type theory by adding the impredicative type *Prop*, of logical propositions, inside the smallest type universe Type(0).

The strong sum ($\Sigma$) type constructor does not reside in *Prop*. Allowing it to do so would cause the logic to become inconsistent. Instead the existential quantifier is defined in terms of the other primitives.

The usual logical operators are defined as (following the usual formulation for second order logic connectives as functions [Leviant, 1994]):

$$
\begin{array}{lll}
\forall x : A.P(x) & =_{def} & \Pi x : A.P(x) \\
P \Rightarrow Q & =_{def} & \forall x : P.Q \\
true & =_{def} & \forall X : Prop.(X \rightarrow X) \\
false & =_{def} & \forall X : Prop.X \\
P \wedge Q & =_{def} & \forall X : Prop.(P \Rightarrow Q \Rightarrow X) \Rightarrow X \\
P \vee Q & =_{def} & \forall X : Prop.(P \Rightarrow X) \Rightarrow (Q \Rightarrow X) \Rightarrow X \\
\neg P & =_{def} & P \Rightarrow false \\
\exists x : A.P(x) & =_{def} & \forall X : Prop.(\forall x : A.(P(x) \Rightarrow X)) \Rightarrow X \\
a =_A b & =_{def} & \forall Pred : A \rightarrow Prop.Pred(a) \Rightarrow Pred(b)
\end{array}
$$

Luo proved that ECC obeys the Church Rosser property, subject reduction and strong normalisation, and uses these to prove the consistency of the internal logic [Luo, 1994]. Computational equality and type checking are both decidable in ECC.

The unified theory of types (UTT) [Luo, 1994] is, essentially, ECC extended with inductive types. Goguen established, through the use of an operational semantics, that UTT is strongly normalising [Goguen, 1994].

## 3.3 Theorem Proving and Lego

The meta-theoretic properties for ECC and UTT described above mean that type checking is decidable. This means that an algorithm can be designed which given a judgement a:A, will check whether the object 'a' really is of type 'A'.

Since propositions are types (of their proofs) a proof checker can be written, which will take a proposition P, and an object p, and determine whether p is a proof of P. Lego [Pollack, 1989] [Luo and Pollack, 1992] is a proof assistant, using this idea, based on ECC and UTT. It assists a user (by providing tactics) in building a proof of a proposition.

Other theorem provers have also been developed based on the different type theories. Examples include NuPRL [Constable *et al.*, 1986], ALF [Augustsson *et al.*, 1990], Coq [Dowek, 1990]. A nice bonus for the method is that the term produced is independent of the program which produced it. A user sceptical that the object really does prove a theorem is free to build their own type checker (a relatively straightforward task) and to type check the object.

**Example ($\mathbf{P} \wedge \mathbf{Q}$) entails $\mathbf{P}$**

Set the proposition up as a goal. The objective is to find an object of this type.

$Goal : \Pi P, Q : Prop.(P \wedge Q) \rightarrow P$

introduce arbitrary propositions A and B, Lego derives the new goal:

$A, B : Prop$

$Goal : (A \wedge B) \rightarrow A$

expand the definition of $\wedge$ based on the definitions given above:

$Goal : (\Pi C : Prop.(A \rightarrow B \rightarrow C) \rightarrow C) \rightarrow A$

introduce a proof of the expanded term, Lego infers the new goal:

$H : (\Pi C : Prop.(A \rightarrow B \rightarrow C) \rightarrow C)$

$Goal : A$

Refine by H(A), i.e. use the object H(A) to infer the goal and make the new goal the antecedent:

$Goal : A \rightarrow B \rightarrow A$

introduce arbitrary objects a:A and b:B, Lego infers the new goal:

$a : A$

$b : B$

$Goal : A$

Refine by a, i.e. use the object a to infer the Goal. This completes the proof. Lego then works back through the steps and builds the proof object:

$\lambda P, Q : Prop.\lambda H : (P \wedge Q).H(\lambda a : P.\lambda b : Q.a)$

## 3.4 Abstract Theories

In doing proof development on a large scale it is useful if a problem can be broken down, and attacked in a modular way. This is particularly so in computer science where the problem often involves reasoning about a class of data types, or code that gets re-used significantly.

With this in mind Luo developed the abstract theory mechanism for modular reasoning in UTT [Luo, 1994] [Luo, 1993].

The following notational convention shall be used for the rest of the thesis:

$$\Sigma[x_1 : A_1, x_2 : A_2, ..., x_n : A_n]$$

will denote the sigma type:

$$\Sigma x_1 : A_1 \Sigma x_2 : A_2, ..., A_n$$

and $\pi i$ will represent the obvious projection function that retrieves the i'th entry of the (dependently typed) n-tuple.

**Definition 3.4.1 (Abstract Theory)** *An abstract theory T, is a 4-tuple*

$$T = (Str[T], Ax[T], Thm[T], Prfs[T])$$

*where*

- *Str[T] contains the structure of T (usually a $\Sigma$ type).*
- *Ax[T] is a predicate over Str[T]. It defines some properties which T must obey.*
- *Thm[T] is also a predicate over Str[T]. These are the theorems which are provable about T.*
- *Prfs[T] is the abstract proof of the theorems of T. Its type is* $\forall t : Str[T].Ax[T](t) \rightarrow Thm[T](t)$

For example, to define an abstract theory for semi-groups the structure type would contain entries for the carrier set and the operation:

$$Str[G] = \Sigma[X : Type, * : X \to X \to X]$$

the axioms would be a predicate ensuring associativity, i.e. $\forall x, y, z : X.(x * (y * z)) = ((x * y) * z)$. This then provides an abstract framework for proving theorems about semi-groups.

Moreover, if a homomorphic mapping can be defined between two abstract theories then it is possible to inherit the theorems and proofs from one theory to another.

For example, a theory for Groups could be defined following the same lines as the theory for semi-groups, only with an extra entry in the structure for the identity element, and an extra axiom corresponding to 'every element has an inverse'. There is an obvious (forgetful) homomorphic map between the abstract theories which allows proofs developed for semi-groups to be inherited by groups.

A framework similar to this will be used to allow for a modular approach to reasoning about SemNet. Note that this framework explicitly uses both $\Sigma$ types and impredicative *Prop* (through predicates) so that it could not be defined in either CC or Martin-Löf's type theory.

## 3.5 Subtyping

As discussed in the introduction type theory is more restrictive than set theory. This results in a more manageable language with useful features (such as decidability of type checking).

A consequence of the restrictions is that type theory has no obvious equivalent of the subset. This is serious for this work since the hierarchy is intuitively understood as a subset hierarchy.

The problem also has implications for the role of type theory as a tool for computer science, since a notion of subtype is extremely useful to model aspects such as re-use. Therefore there is a lot of research being done to establish coherent theories of subtyping that do not impinge on the 'useful properties' that type theories have [Jones, 1996].

This work will make use of subtyping based on coercive functions [Luo, 1996]. Here the basic subtyping relation is generalised from a basic set of coercions. To say A is a subtype of B:

$$A \preceq B$$

means that there is a (coercive) function $\kappa : A \rightarrow B$ and whenever an object a:A is used where an object of type B is expected then $\kappa(a)$ is used. Uniqueness of typing is lost, but it is replaced by a notion of all objects having a unique *principal type*, where the principal type loosely corresponds to the 'smallest' type.

## 3.6 Constructive Type Theory for Natural language

As has already been outlined the main application of type theory has been for formal methods for software development and for formalisation of constructive mathematics. Indeed these applications have driven the design of the modern versions of the theories.

An entirely separate application of type theory has been to use it as a tool for studying natural language. Natural language is extremely complicated with many aspects. Cann [Cann, 1993] describes the task of formal semantics for natural language as

"the study of meaning as expressed by the words, phrases and sentences

> of Human languages. It is, however, more usual within linguistics, to interpret the term more narrowly, as concerning, the study of those aspects of meaning encoded in linguistic expressions that are independent of their use on particular occasions by particular individuals within a particular speech community"

Simple type theory (i.e. without dependent types) has long been a tool in this endeavour [Cann, 1993] [Dowty *et al.*, 1981]. Part of the reason for this is usage of functions and for quantification over types.

### Dependent types for quantifiers

More recently Ranta [Ranta, 1994] has shown that aspects of Martin Löf's type theory are useful for categorising natural language. It is argued that quantifiers for noun phrases are better captured through the dependent types:

$$\left\{ \begin{array}{l} some \\ an \\ a\ certain \end{array} \right\} \longmapsto \Sigma - types$$

$$\left\{ \begin{array}{l} every \\ any \\ each \end{array} \right\} \longmapsto \Pi - types$$

Giving the following (loose) interpretations:

| | | |
|---|---|---|
| a man owns a donkey | *as* | $\Sigma x : man.\Sigma y : donkey.owns(x,y)$ |
| every man owns a donkey | *as* | $\Pi x : man.\Sigma y : donkey.owns(x,y)$ |

The $\Sigma$ type is also used to build up complex noun phrases. For example,

| | | |
|---|---|---|
| old man | *as* | $\Sigma x : man.old\ x$ |
| man that owns a donkey | *as* | $\Sigma x : man.\Sigma y : donkey.owns(x, y)$ |

In this way, such noun phrases can be treated directly as constituents (rather than being dissolved as an antecedent in predicate calculus) of sentences. For example, "*every old man walks*" can be interpreted as

$$\Pi z : (\Sigma x : man.old\ x) walks\ \pi 1\ z$$

rather than as $\forall x : man.old\ x \Rightarrow walks\ x$.

### $\Sigma$ types model progression

A main argument for using the dependent types is that they can model progression of a text or discourse. For example, to capture the conjunction and implication involved in:

"a man walks and he whistles"
"if a man walks he whistles"

The initial statement, in both cases, "a man walks" is interpreted as:

$$\Sigma x : man.walks\ x$$

interpreting conjunction as a $\Sigma - type$ and implication as a $\Pi - type$ (rather than as $\wedge$ and $\Rightarrow$) means that the interpretations of the second statement, "he whistles" can model the progression of the statements by extracting the relevant parts of the initial statement (as occurs in natural language).

$$\Sigma z : (\Sigma x : man.walks\ x).whistles\ \pi 1\ z$$
$$\Pi z : (\Sigma x : man.walks\ x).whistles\ \pi 1\ z$$

Similarly the 'donkey sentence' can be interpreted as:

$$\Pi z : (\Sigma x : man.\Sigma y : donkey.owns(x,y)).beats(\pi 1\ z, \pi 1\ (\pi 2\ z)) \qquad (3.1)$$

This aspect of dependent types is used in this work to model how SemNet builds and re-uses concepts, see sections 7.2 and 7.5.

### Contexts and possible world semantics

Ranta also considers modelling statements of belief. Essentially each agent is assigned a context consisting of judgements they have made. The contexts then act like possible worlds [Meyer and der Hoek, 1995], except that since the context can be progressive, later beliefs can depend on earlier ones. What the agent believes are all the judgements that are provable in this context.

This is of course extremely relevant to this work. In particular SemNet has constructs for representing the above features of language and so it will be interesting to see if the constructive aspects of UTT are useful for capturing these constructs.

## 3.7 Motivations for using UTT

In summary there seem to be many good reasons for attempting to formalise SemNet using UTT and Lego. These are summarised as follows:

- The intuitive meaning of SemNet nodes is defined by their properties, rather than their extension. It seems that type theory with intensionally defined

functions may provide a better tool for a better interpretation of nodes than sets.

- Event nodes in SemNet correspond to statements. This causes difficulties in set theory, as statements cannot naturally be interpreted as sets. However in type theory propositions are first class objects (i.e. types) and so events can be interpreted as objects of type *Prop*.

- The work of Ranta has shown that constructive type theory has features that describe complex aspects of natural language. Since SemNet claims to represent various complex aspects of natural language, constructive type theory seems an ideal tool for testing this out.

- UTT is a well established mathematical theory with many useful properties, including an internal logic which is consistent. As well as this research developed for applications to computer science have left behind a useful legacy of tools and techniques including:

  - Abstract reasoning mechanism for modular approach to problem solving.
  - The proof assistant, Lego, which helps in the development of proofs and provides machine based proof (type) checking.
  - SemNet has been written in Haskell [Hudak *et al.*, 1992], a strongly typed functional programming language [Bird and Wadler, 1988], [Holyer, 1991]. Therefore it may be straightforward to convert this code into Lego code, and so apply techniques from formal methods.

# Chapter 4

# SemNet

The role of this chapter is to informally introduce SemNet and its associated reasoning mechanisms. Unless stated the current implementation will be described. Some representation issues are under current development and are described since it is felt that the work of this thesis contributes to the discussion.

As discussed in chapter 1, a KR for NLE must be expressive, natural, readable, efficient, robust, and flexible. In this chapter the aim will be to convince the reader, intuitively, that SemNet meets these aims. A metric **distributedness** which is directly related to these attributes is described. To show the relevance of this metric a short analysis of its application to SemNet and other representations is described.

No formal attempt at interpretation is made in this chapter, however, in some cases where ambiguity could arise, classical set theory is used to express informal meanings.

## 4.1 SemNet basics

In common with semantic networks SemNet is a graph based representation, where concepts and relationships are represented by nodes and arcs. "Knowledge" is

elicited by graph traversal. SemNet has been designed specifically for NLE [Shiu *et al.*, 1996], in particular it needs to be expressive, efficient, robust, flexible and easily integratable with other modules. SemNet supports many forms of reasoning as well as fully exploiting inheritance. There are, for example, models of epistemic reasoning, time and location [Short, 1996], reasoning by analogy [Long and Garigliano, 1994] and standard logical connective reasoning.

### 4.1.1 Nodes and Arcs

There are 3 types of nodes: entities, events (assertions) and actions. There are 3 types of directed arcs: subject, object and action which can be read/traversed in either direction. Only event nodes can have a subject, object or action arc attached. Only action nodes can be an action for an event node.

Figure 4.1 shows a section of SemNet graph. Event nodes correspond to statements. The event node E1 states that "Every FARMER1 OWNS a DONKEY1". The two 'spec' links are a shorthand for events with 'specialisation' as action. From now on such 'hierarchy' events will regularly be considered and drawn as links, and will be termed 'event links' or 'hierarchy events'. The subject/object arcs determine the direction of the statement. Intuitively events state that the referenced concepts are involved in a relation (labelled by the action). On each node there is a quantification tag which makes explicit the way in which the referenced concept is used.

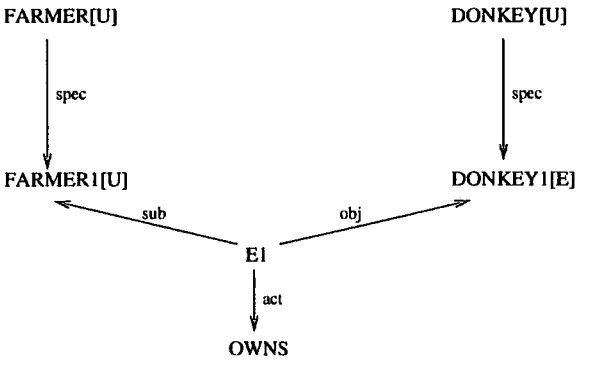



Figure 4.1: Basic SemNet graph

The meaning of the tags are as follows:

- **Universal [U]** refers to "instances" of the concept and says that all the "instances" of the concept are involved in relationship specified by the event.
- **Existential [E]** refers to the "instances" of the concept, but the "instance" involved depends on the particular "instance" of some other universally quantified concept involved in the event.
- **Framed Universal [FU]** (not used in the diagram) is used when a dependency works in both directions, i.e. as a shorthand for having two Universal-Existential events. For example, if FARMER1 and DONKEY1 were both tagged with FU quantifications, then the event would state that:

  "Every FARMER1 OWNS a DONKEY1' and 'every DONKEY1 is owned by a FARMER1"

- **Individual [I]** (not used in the diagram) refers to the concept as a "whole" and says that it is involved in the relationship specified by the event.
- **Named Individual [NI]** (not used in the diagram) is the same as the individual tag except that the concept has a fixed name.

As well as making assertions, events define the concepts which they reference. For example, FARMER1 is defined as the concept "farmers that own donkeys" and DONKEY1 as "donkeys owned by a farmer".

### 4.1.2 Negation of events

An action can be negated, so that an event states that the referenced concepts are explicitly not in the labelled relationship.

For example, E1 in figure 4.2 asserts that "Every FARMER2 does not own one of the DONKEY2's", thus defining FARMER2 as the "farmers that do not own all

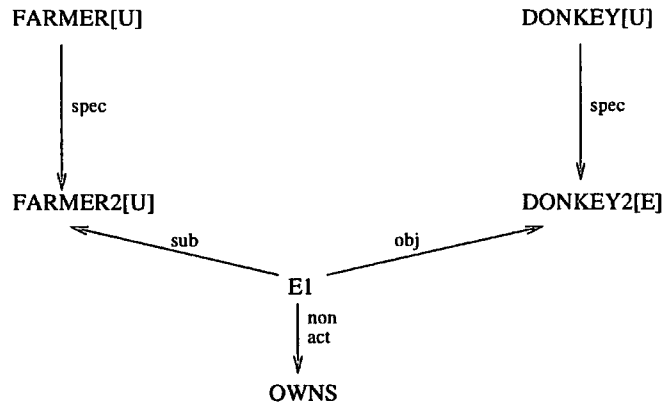



Figure 4.2: Negated Event

donkeys" and DONKEY2 as the "donkeys that are not owned by some farmer". In first order logic:

$$\forall x \exists y FARMER2(x) \rightarrow (DONKEY2(y) \wedge \neg OWNS(x, y))$$

This is equivalent to insisting that negations can only be applied to 'atomic'[1] propositions in $\mathcal{FOPC}$. Therefore to negate an event, as well as negating the action the quantification tags also need to be changed[2]. For example, the negation of $E_1$ in figure 4.1 is $E_3$ in figure 4.3. The 'inst' link is a shorthand for an event with 'instance' as action.

[1]That is, an un-quantified proposition without any logical connectives.
[2]Except when the referred concept acts as a constant, e.g. if it is a Named Individual.

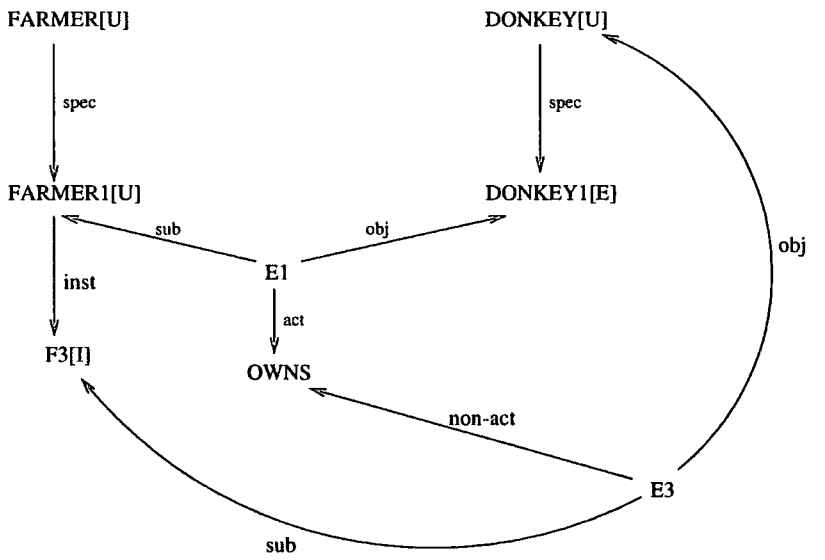



Figure 4.3: Logical Negation of an Event

In first order logic:

$$\exists x.\forall y(FARMER1(x) \wedge DONKEY(y) \wedge \neg OWNS(x,y))$$

There is a negation function $not : Event \rightarrow Event$ which implements this.

## 4.2 KR and the world

The information which is recorded within SemNet is intended to reflect the world as it is understood by the agent that uses the network (LOLITA). No claim is made that the representation reflects the world as it really is (if there is such a thing), nor even that the representation reflects some consensus view of the way the world is.

### 4.2.1 SemNet and Language

It should be stressed that SemNet is intended to represent knowledge declaratively and independently of natural language. As discussed in chapter 1, many of the surface linguistic features of an utterance have been removed. For example, active and passive versions of statements are normalised out during analysis and replaced (according to requirements) during generation [Smith, 1995].

The concepts are finer grained than words, however, with the principle that words only occur in a language when they correspond with a useful concept, many of the nodes of SemNet correspond directly with words. WordNet [Miller, 1990] has been used to help in building these concepts into SemNet.

### 4.2.2 Meaning as Location

No concepts have a pre-defined meaning in SemNet. The meaning of a node/concept is defined by its location in the network[3]. For example, from figure 4.1:

> FARMER1 is the concept of 'FARMER's that own a DONKEY'

but of course, FARMER and DONKEY have no pre-defined meaning, their meaning is established by reading their local connections. Doing this defines FARMER1 further, i.e.

> FARMER1 is the concept of "HUMANs that own and cultivate a FARM'
> that own a DONKEY"

and FARMER1 is only defined when FARMER and DONKEY are fully defined, i.e. when the whole network has been read.

[3]This is an informal notion that will be investigated further in the later chapters.

## 4.3 Control Variables

Each node has an associated set of controls. Controls contain standard information shared by a large number of nodes. For example, the type of a node (i.e. event, action or entity), and the quantification tags of the entity nodes are stored as controls.

In theory the information stored as controls could be expressed as part of the graph. However, in each case, a design decision based on the fact that this information is looked up regularly, has been made to store the information locally/internally.

## 4.4 Real and Hypothetical Events

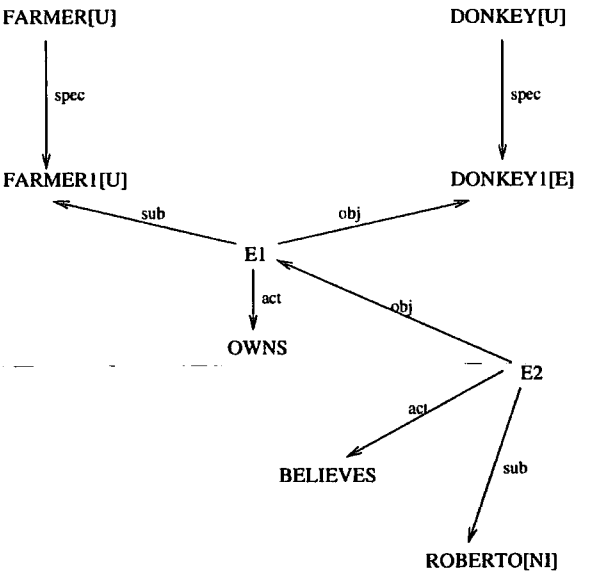



Figure 4.4: Epistemic Event

It is possible for LOLITA to believe that another agent believes some event to hold. For example, LOLITA may believe that "Roberto believes that every farmer owns a donkey.", see figure 4.4. This syntax is similar to the 'proposition' nodes of SNePS (see section 2.4.1) and allows recursive nesting of beliefs.

According to the description given so far, there is no difference between the

way $E_1$ is represented when LOLITA believes it, and when it is there merely as a part of some other event which LOLITA believes (of course it could be both). A 'status control' makes this distinction, status 'real' means that the event is part of LOLITA's belief set (i.e. she believes it) and status 'hypothetical' means the event is there merely as a substructure to some other 'real' event.

## 4.5 Representation Issues

The above description is how SemNet is implemented currently. Some design issues can already be discussed.

### 4.5.1 Observed Events

It is often useful to refer to complex concepts without affecting their meaning. For example to represent the well known donkey sentence,

"Every farmer that owns a donkey beats it"

requires the concept of "farmers that own donkeys" as the subject for the beating event, but with the current structure if a beating event is added, then the concept is changed to "farmers that own and beat a donkey". The proposed solution is to have a different event type, which makes assertions but does not define its reference concepts, as in $E_2$ in figure 4.5. This mimics the progression that occurs so often in natural languages as discussed in section 3.6. By re-referring to the same node, unique concepts for unique nodes is preserved, see section 2.3.3.

A further rule required here is that when a node is re-visited, then the reference is to the same instance previously scoped. In this example this means that if $E_1$ has been 'traversed' then a 'donkey owning farmer' and 'the donkey he owns' will have been specified, after this if $E_2$ is 'traversed' the interpretation is that it is this same 'donkey owning farmer' that 'beats' the same 'donkey'. Without this rule, $E_2$

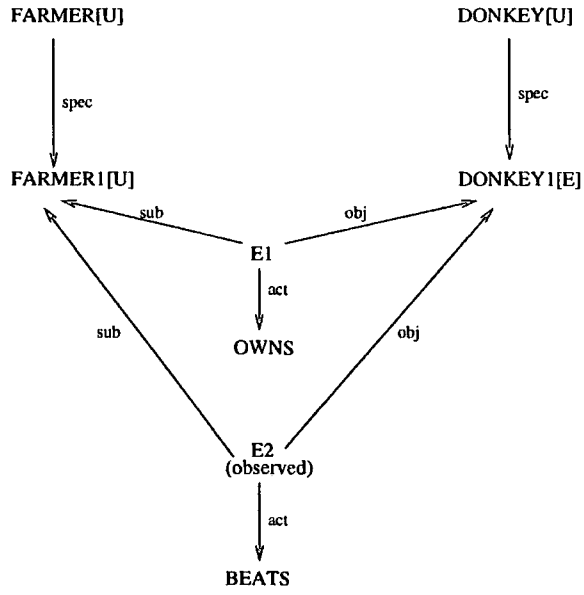



Figure 4.5: Proposed SemNet graph for the 'donkey sentence'

would simply say that every 'donkey owning farmer ' beats some 'donkey owned by a farmer'.

### Connections with known networks

The separation of defining and observing events is the same distinction made between T-box statements and A-box statements in KL-ONE (see section 2.2). However in SemNet both event types are subject to the same *semantic analysis* and *inference engine* modules so that there is a very close coupling between the two sorts of information, as demanded by [Beierle *et al.*, 1992].

### Defined Events and Necessity

Observed events have a different effect on the meaning of a concept. In figure 4.6 node X can be interpreted as: "Computer Science staff that play football", and it is an observed fact that "all X's study AI". It is reasonable to state that:

"all X's necessarily play football"

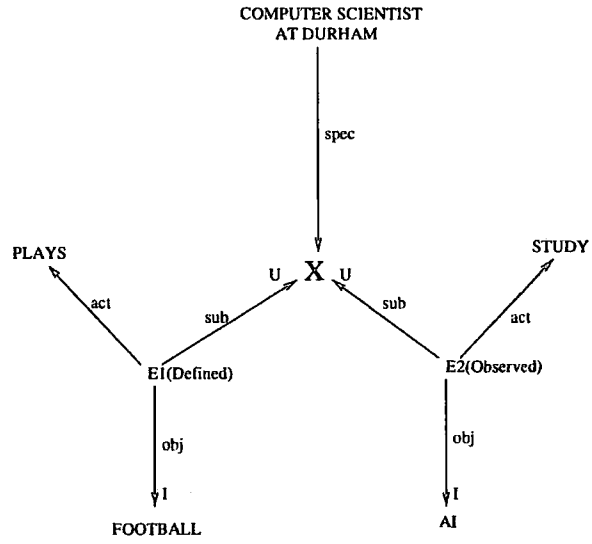



Figure 4.6: SemNet graph, showing necessity

but it is not reasonable to state that:

"all X's necessarily study AI"

Algorithms which need this distinction (e.g. causality [Poria and Garigliano, 1996] and generation) should use the observed event accordingly.

### 4.5.2 Quantification on the arcs

A concept can be referred to by many events. It is possible that a node be referenced with different quantifications. For example to represent the sentence:

"Every mother has a brother each of whom owns a parrot."

The concept for brother is used twice, once as an existential concept and once as a universal. Since quantification is tied to the concept the only way this can be done at the moment is to make two copies of the concept see figure 4.7. Clearly this is not efficient in terms of net size. Alternative solutions are to move the quantification tags on to the arcs, see figure 4.8, or to have them as controls on the event.

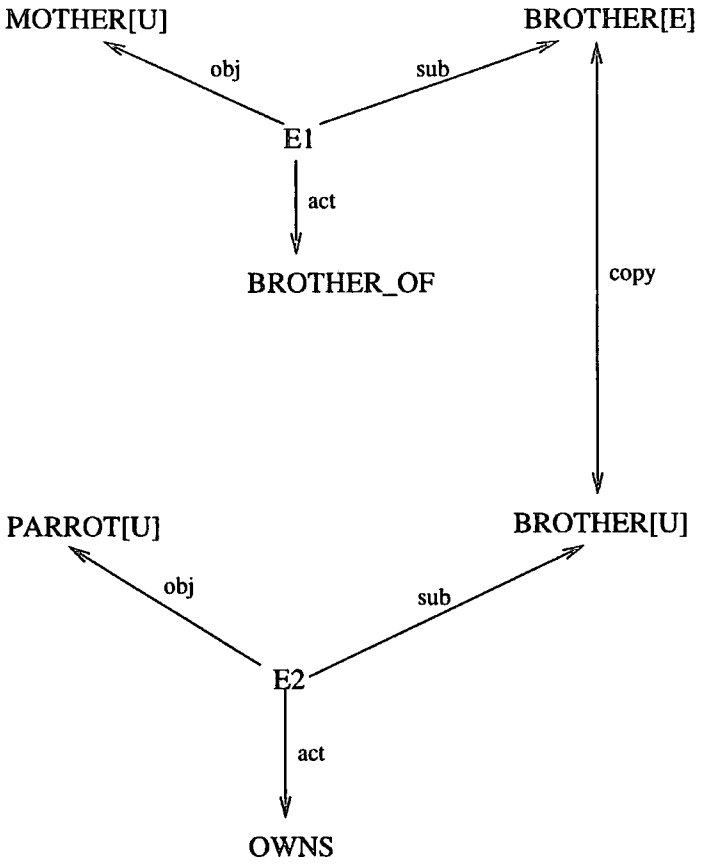



Figure 4.7: Current quantification scheme, example problem.

The hierarchy events actually treat 'Universal' concepts as 'Individual' since they refer to the 'set' as a whole. Changing the scheme will allow a more general approach to concept referencing.

From now on it is assumed that quantification is attached to the event node.

### 4.5.3 Named Individuals and constants

An initial, intuitive interpretation for concepts might be to treat individuals as variables and named individuals as constants.

For example, interpretations for $E_1$ and $E_2$ in figure 4.9 could be:

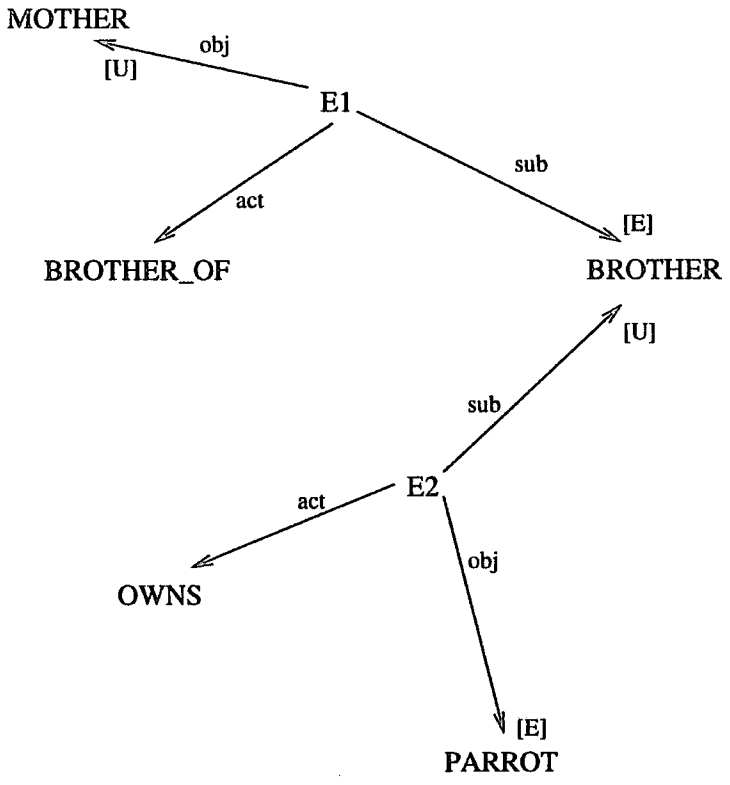



Figure 4.8: New quantification scheme, solution.

$$E_1 = \forall x.x \in dogs \rightarrow Likes(Sanjay, x)$$
$$E_2 = \exists y.\forall x.x \in dogs \rightarrow Likes(y, x)$$

Further analysis shows the situation to be more complex. If an individual is referred to by an observed event, then it too behaves like a constant, e.g. $E_3$ refers to the constant 'man that likes all dogs' (i.e. MAN1) and states that he 'hates all cats'. Also the concept 'Sanjay' is not defined at all by event $E_1$, it is defined as the concept with name 'Sanjay'. Therefore, quantification and 'naming' of concepts are separate. For the remainder of this thesis it will be assumed that if a concept is named, this will be stored as a control on the node (effectively a shorthand to say this concept is defined as the concept with this name). The 'named individual' quantification will not be used.

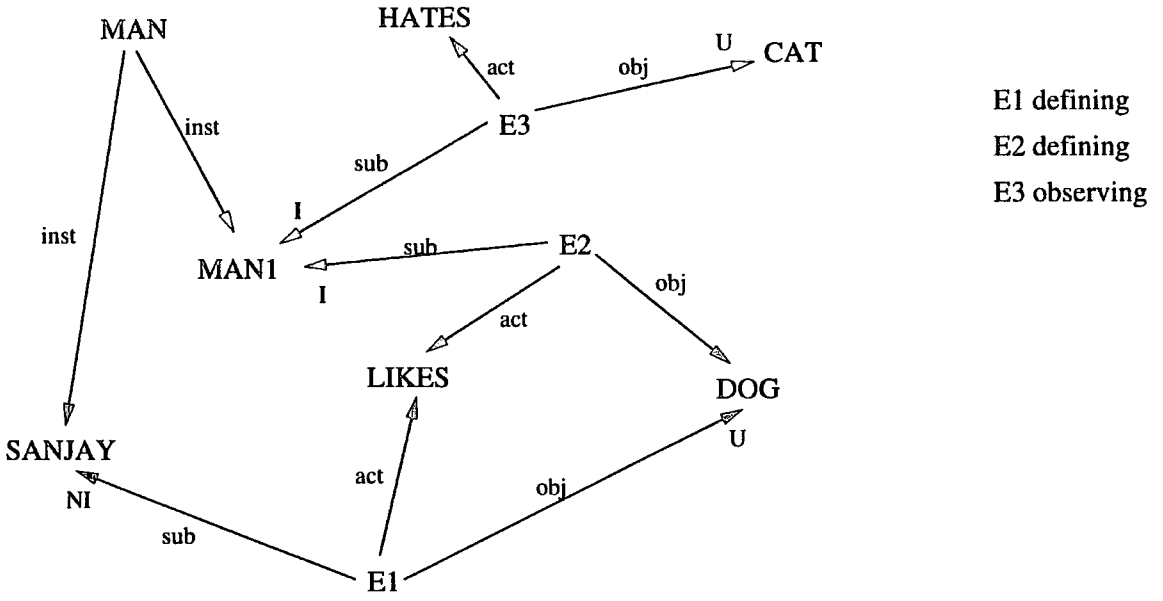



Figure 4.9: Example showing names as properties

## 4.6 Reasoning Mechanisms on SemNet

As stated there are many inference algorithms that have been designed and implemented for SemNet. All reasoning proceeds by passing an event (intuitively a proposition) to SemNet and an algorithm determines whether the event (proposition) or its negation is entailed (validly or plausibly) by SemNet. Before allowing an event to be added to SemNet the reasoning mechanisms first check that neither it or its negation can be inferred.

### 4.6.1 SemNet linear Notation

To list the inference rules, the following notation will be used:

| | |
|---|---|
| $R_1, R_2, R_3, ...$ | will represent arbitrary actions |
| $A, B, C, D$ (occasionally indexed) | will represent arbitrary entities |
| $a, b, c$ | will represent individual nodes |
| | (when it is clear that a node is an individual) |
| $E_1, E_2, E_3, ...$ | will represent arbitrary events |
| $bb$ | will represent arbitrary boolean values |
| $Q_1, Q_2, Q_3, ...$ | will represent arbitrary quantification tags |

The structure of events will be written as:

$$(R_i, bb, A, Q_j, B, Q_k)$$

For example, (OWNS,true,FARMER1,U,DONKEY1,E), represents $E_1$ in figure 4.1. Occasionally, when no confusion can arise the boolean and quantification values will be omitted. For example, (BELIEVES,ROBERTO, E1) represents $E_2$ in figure 4.4. Hierarchy events will be written as:

| | |
|---|---|
| A $\succeq_s$ B | informally, B is a specialisation of A |
| $\neg$ (A $\succeq_s$ B) | informally B is not a specialisation of A |
| a $\in$ A | informally, a is an instance of A |
| a $\notin$ A | informally, a is not an instance of A |

### 4.6.2 The Entity Hierarchy

All the entity concepts lie in a hierarchy. There is a top concept (called "Entity") and all the other entities are either specialisations or instances of "Entity", see figure 4.10. Specialisation and instances may usefully be interpreted as the subset and membership relations of set theory, however, since entities will be formally interpreted as types, this will not be strictly correct. Again the specialisation and instance events have been drawn as links. Negated occurences here actually

correspond to the logical negation, as the entities are referred to as constants.

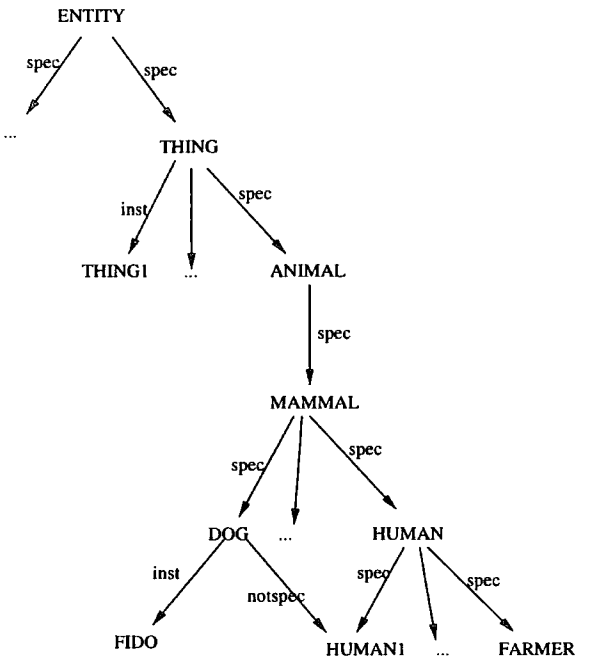



Figure 4.10: Section of the Entity hierarchy of SemNet

Each event which is explicitly present in the network is treated as an axiom, and so can immediately be inferred. There are two rules for deriving spec events:

$$4.1 \quad \frac{A \succeq_s B, B \succeq_s C}{A \succeq_s C} \qquad 4.2 \quad \frac{A \succeq_s B, \neg(A \succeq_s C)}{\neg(B \succeq_s C)}$$

At this stage there is no closed world assumption, so that the second rule is the only way in which a nonspec relation can be inferred. Inheritance rules for the basic events are:

$$4.3 \quad \frac{b \in B, A \succeq_s B}{b \in A} \qquad 4.4 \quad \frac{a \notin A, A \succeq_s B}{a \notin B}$$

$$4.5 \quad \frac{A \succeq_s B, (R, bb, A, U, C, Q)}{(R, bb, B, U, C, Q)}$$

4.6
$$\frac{a \in A, (R, bb, A, U, C, Q)}{(R, bb, a, I, C, Q)}$$

4.7
$$\frac{D \succeq_s C, (R, bb, C, E, A, U)}{(R, bb, D, E, A, U)}$$

4.8
$$\frac{a \in A, (R, bb, A, U, B, E)}{f(a) \in B, (R, bb, a, I, f(a), I)} \quad f(a) \text{ new}$$

These rules allow for very efficient inference by searching up and down the hierarchy. The above operate only on the subject of each event and there are an equivalent set of rules for the objects.

The intuition behind these rules is as follows:

4.5 captures the inheritance involved in inferring that "all HUMANs eat food" implies "all FARMER's eat food".

4.6 captures the inheritance involved in inferring that "all DOG's like food" implies "Fido likes food".

4.7 captures the inheritance involved in inferring that "There is a DOG that likes all HUMANs" implies "There is an ANIMAL that likes all HUMANs".

4.8 captures the inheritance involved in inferring that "All DOGS like a HUMAN" implies "There is a HUMAN that FIDO likes"

### 4.6.3 The action hierarchy

As well as the entity hierarchy there is an action hierarchy.

Writing an action spec event as

$$R_i \succeq_{sa} R_j$$

The associated inference rules are:

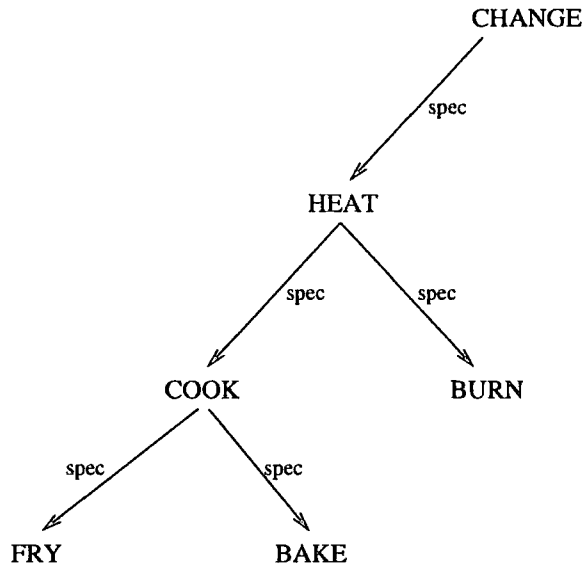



Figure 4.11: A section of the Action Hierarchy

4.9
$$\frac{R_2 \succeq_{sa} R_1, (R_1, true, A, Q_1, B, Q_2)}{(R_2, true, A, Q_1, B, Q_2)}$$

4.10
$$\frac{R_2 \succeq_{sa} R_1, (R_2, false, A, Q_1, B, Q_2)}{(R_1, false, A, Q_1, B, Q_2)}$$

4.9 captures the intuition behind inferring
"Simon fries an egg" implies "Simon cooks an egg"

4.10 captures the intuition behind inferring
"Simon did not cook an egg" implies "Simon did not fry an egg"

### 4.6.4 Connective Reasoning

A further type of events are the logical connective events. $E_3$ in figure 4.12 is a logical connective event with action 'Implies'.

The logic actions are: Implies, Or, and And. Their intended meanings are the standard logical connectives of **relevant** propositional logic [Anderson and Belnap,

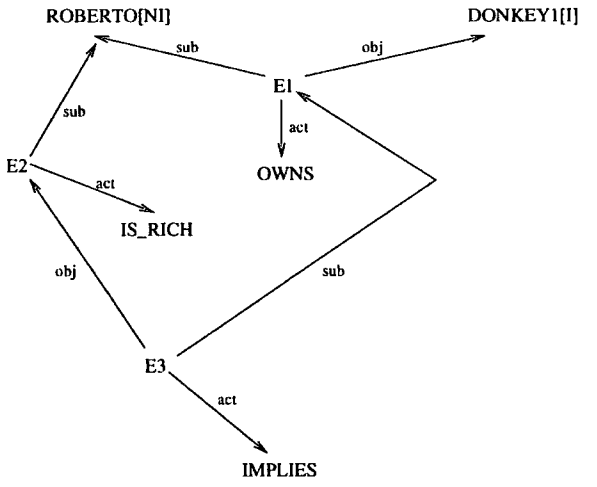



Figure 4.12: An example logical event

1975]. The associated inference rules ('not' is the negation function discussed in section 4.1.2) are:

$$ModusPonens \qquad \frac{(implies, E_1, E_2), E_1}{E_2}$$

$$ModusTolens \qquad \frac{(implies, E_1, E_2), not(E_2)}{not(E_1)}$$

$$\frac{(or, E_1, E_2), not(E_1)}{E_2}$$

$$\frac{E_1, E_2}{(and, E_1, E_2)}$$

$$\frac{E_1}{(or, E_1, E_2)}$$

There is a 'Cause' action which behaves similarly to 'Implies' except that it is intended to capture the situation where there is a temporal dependency between the referenced events, [Short, 1996].

Since 'Implies' is used relevantly, rules such as:

$$\frac{E_2}{(implies, E_1, E_2)}$$

are not valid. Capturing the meaning of relevant implication formally is difficult since its derivation is not truth functional.

Many of the other representations which could be used in inference, such as:

$$\frac{(and, E_1, E_2)}{E_1}$$

are not listed as they are normalised out during the analysis phase.

### 4.6.5 Epistemic reasoning

Epistemic events have an agent as subject, an event as object and an epistemic relation (for example, know, believe or think) as action, see figure 4.4. Currently all epistemic relations (i.e. know, believe and think are treated in the same way).

It plausibly follows that a man that believes that "all farmers own a donkey" also believes that "all small farmers own a donkey". The intuition behind the epistemic rules are that LOLITA assumes that all agents are capable of making the same inferences as she can. Therefore there is an epistemic rule for each of the rules previously described, essentially allowing LOLITA to assume that the agent can apply that rule. For example, the epistemic version for rule 4.1 (transitivity of spec relations) is:

4.11
$$\frac{(R_{epi}, Agent, (A \succeq_s B)), (R_{epi}, Agent, (B \succeq_s C))}{(R_{epi}, Agent, (A \succeq_s C))}$$

which captures the intuition in the above inference.

As well as these there are also rules which assume that if 'an agent believes something' then 'the agent believes that they believe that something', i.e.

$$\frac{(R_{epi}, Agent, E)}{(R_{epi}, Agent, (R_{epi}, Agent, E))} \quad 4.12$$

and that 'if an agent believes 'E is not the case" then 'the agent does not believe that 'E is the case", i.e.

$$\frac{(R_{epi}, true, Agent, not(E))}{(R_{epi}, false, Agent, E)} \quad 4.13$$

## 4.7 NLE principles

Chapter 1 outlined the aim of adding formality to the intuitive good points of SemNet. These were based around the NLE principles. Having described SemNet in more detail, we are now in a position to describe (although still informally) how it meets some of these principles. These will motivate the areas which this project will attempt to formalise.

### 4.7.1 'Correctness' of Reasoning

In order to be more formal, a mapping to a formal language needs to be given. The rules should then be shown to be sound with respect to this formal interpretation. If the rules are not sound then it might be expected that the rules correspond to some intuitively plausible sequence/operation in the semantics.

The other aspect of correctness that could be looked at is how well the implementation corresponds to the definitions given. Ideally there would be a translation to some programming language structures and a proof that the code implements

the declarative rules.

### 4.7.2 Expressiveness for NLE

This chapter has shown that SemNet can represent basic relationships, and statements with anaphora and complex dependent quantifications (including the well known problematic 'donkey sentence' structure).

Mechanisms are provided to represent and reason about epistemic statements and the standard logical connectives.

SemNet defines concepts in terms of their properties (which result from their location in the network). Some properties are necessary facts and some are not.

To show this more formally a semantic model is required. To analyse the above aspects the semantic language must be capable of 'expressing' each of them. Analysis, apart from establishing that the different structures have a clear and distinguished interpretation, will involve giving semantic counterparts to any algorithms which operate on these structures.

It would be useful if the manner in which these aspects are built is similar in both SemNet and its semantic counterpart. To give a straightforward example, if the semantic language were $\mathcal{FOPC}$ the logical action 'AND' will clearly map to the logical connective '$\wedge$'. This correspondence means that syntactic operations involving this 'action' are easily understood in terms of the formal semantics.

As discussed in chapter 2, there has been much research investigating the idea that semantic networks give up expressiveness in order to gain tractable sound and complete reasoning (something not possible for full $\mathcal{FOPC}$). Instead various heuristics are used for various efficiency reasons and no claim is made for completeness. With this in mind it is still interesting to attempt some form of metric for how expressive SemNet is. The formal model will consist of a mapping of all SemNet structures into a language. This will serve as a starting point for considering an inverse mapping from the semantic language into SemNet structures. This in turn

will be a starting point for measuring the expressiveness of SemNet.

### 4.7.3 Developer comprehend-ability

Subjectively, SemNet is readable and easily comprehendible. It is important from a team development (and hence engineering/pragmatic) viewpoint that this is so. However, again, as the scale and complexity increases there is a danger that different developers have different interpretations which could lead to incorrect assumptions and code.

A formal semantic model will address this problem as it provides an unambiguous reference point for the meaning of constructs.

### 4.7.4 Flexibility and Robustness

Information is retrieved and inferred from SemNet by graph traversal. Intuitively there is a lot of flexibility in the structure. It seems that any node can be picked and from there any arc can be traversed (in any direction) and 'reasonable' information can be 'read'.

Such flexibility is extremely useful since it allows the inference engine designer the freedom to choose the most appropriate path for any inference algorithm without worrying about the interpretation. Similarly the generator can take any amount of the net and 'say' it, and rely on it being a reasonable part of the belief set of LOLITA.

This flexibility is useful from a robustness point of view as if it is necessary to stop traversing a graph (for whatever reason) the information gained is still reliable. The next section elaborates on these informal notions, starting to build a proper theory for it.

## 4.8 Distributedness

This section is a cut down description of the work described in [Short *et al.*, 1996].

A network is said to be **distributed** if any section of network gives meaningful information which is sound with respect to the full reading of the network.

If a formal semantic model were in place, 'full reading' could be defined as the full model of the network, and the interpretation of any section should be entailed by the full model.

Distributedness is related to compositionality, which demands that the full meaning of the network should be a function of the meaning of its syntactic sections. Distributedness adds the requirement that not only must they be an argument of the function, but must also be sound with respect to the result of the function.

### 4.8.1 Distributedness of SemNet

In SemNet a single node (say E1 in figure 4.1) tells us nothing, except that some concept exists. Its controls will specify its type (event, real in this case). Every arc attached to the node specifies $E_1$ further: the action arc specifies the relation, the subject arc specifies that it is all the instances of $FARMER_1$ that participate in the owning relation in the subject role, and the object arc specifies that there is a (scoped) instance of $DONKEY_1$ which participates in the relation in the object role. This information is a sound sub-part of the interpretation that all instances of $FARMER_1$ own a (scoped) instance of $DONKEY_1$ Thus each arc conveys an independent and sound piece of information about the node. As a further example, the spec link tells us that $FARMER_1$ is a 'subset'[4] of FARMER which only adds to the interpretation. $E_1$ is still not entirely defined: each node is only fully defined by the whole semantic network. Information which must be picked up to preserve soundness of interpretation is the control determining whether an event is

[4]The terms subset and superset are used loosely here; formal interpretations are considered in chapters 5, 6 and 7.

hypothetical or real, as it can be discarded if it is hypothetical.

This aspect is exploited by the inheritance algorithm. Although the rules are listed in terms of fully defined events, the implementation (of say rule 4.1) relies on the 'subject' of an event being read independently from the rest of it. For example, if the subject is 'universal' then the inheritance algorithm will search 'up' the entity hierarchy, whatever the object and actions are.

### 4.8.2 Distributedness of other Semantic Networks

The remainder of this section describes some initial investigations into the distributedness of other representations. This is done not as a criticism of other networks, but to test out the relevance of these new properties and also to try and show where SemNet differs from other well known networks.

It is perhaps easiest to consider $\mathcal{FOPC}$. A knowledge base of $\mathcal{FOPC}$ would be a list of statements.

$$\forall x \forall y (Farmer(x) \wedge Donkey(y) \wedge Owns(x,y)) \rightarrow Beats(x,y)$$

Distributedness is the extent to which subsections of this can be taken and interpreted. Each individual statement can be taken and used independently. However, in most cases, it will not be possible to take a subsection of a $\mathcal{FOPC}$ statement and consider it as a sound part of the knowledge base. For example, if the initial segment of the above statement is taken, i.e.

$$\forall x \forall y (Farmer(x) \wedge Donkey(y) \wedge Owns(x,y))$$

this is clearly not something that follows from the full knowledge base.

The T-Box of KL-ONE based systems [Woods and Schmolze, 1992], [Beierle *et*

*al.*, 1992] is Semantic Net based, the A-Box usually consisting of FOL statements. Thus the A-Box suffers from the same problems as above. However the T-Box which defines concepts can be traversed, seemingly in an unrestricted way. This shows that this aspect of KL-ONE could be distributed, although to show this formally the semantic model described in section 2.2 would need to be used.

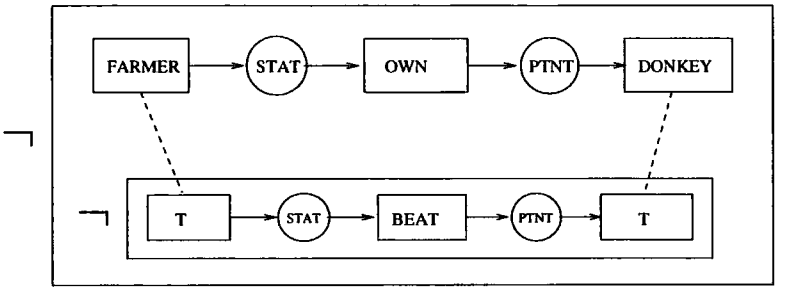



Figure 4.13: CGT graph for the donkey sentence

CGT [Sowa, 1984] builds complex logical assertions using contexts. Figure 4.13 shows how the donkey sentence is represented by CGT. This use of contexts requires the whole context to be read/traversed for any sense to be made. For example, the innermost sub-context is interpreted as "Farmers do not beat Donkeys". If this is read independently from the rest, the interpretation derived is not sound with respect to that provided by the full context. For CGT the independent pieces of network must be at the level of a context rather than its components. This is less distributed than SemNet, where arcs form the smallest independent pieces of the network.

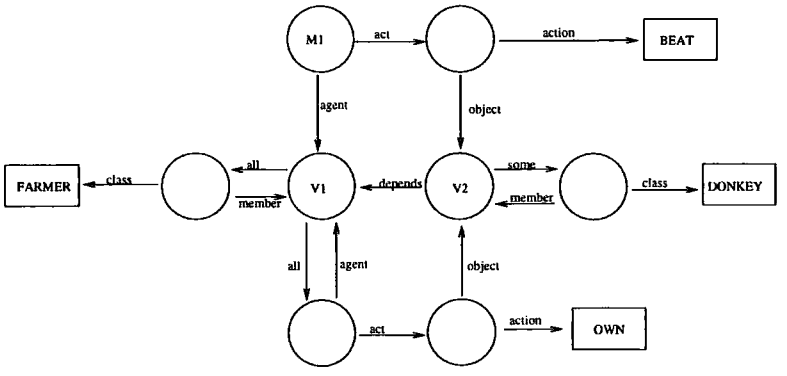



Figure 4.14: ANALOG/SNePS graph for the donkey sentence

Figure 4.14 shows how ANALOG represents the donkey sentence. This representation is similar to SemNet. The structured variable nodes V1 and V2 are defined by their outgoing arcs. These defined V1 as: "the intersection of all farmers and all things that own a donkey" and V2 as "some donkeys, scoped by the variable V1". These variables are re-used by M1 which states the beating event. As in SemNet, a rule is needed to state that the beaten donkey is the same donkey that is owned by the farmer.

Intuitively each arc can be read independently and soundly. However it is not clear how negative statements are made in ANALOG. In SemNet, negation is always attached to the action (i.e. the relation), so that other information can be read independently. SNePs would also have to mimic this to be distributed.

## 4.9 Review

This chapter has given a description of the structure of SemNet. In particular more detail has been added to the *proposition as types* and *meaning as location* principles. An explanation of the hierarchy, the inference engine and some of the current research issues of SemNet have been described. It should be stressed again that performing the semantic analysis has changed the authors view of it leading to better understanding, hopefully more abstract and hopefully a better exposition of it.

For the rest of this thesis, this aspect of the contribution will not be further expanded. Instead, two models will be built, one set theoretic and one type theoretic and they will be judged by how well they formalise the intuitive concepts described here.

# Chapter 5

# Formalisation Issues

The role of this chapter is to highlight some of the problems involved in formalising SemNet. A set theoretic semantic model of SemNet is developed. This is then used to analyse some of the properties of SemNet, including distributedness.

## 5.1 Set Theoretic Semantics

In chapter 4 the primitive syntactic objects of SemNet were the three node types (entity, event and action) and the three arc types (subject, object and action). The inference rules were each given in terms of fully defined events. Their substructures (defined by the arcs) were not used. Since the first objective of the formalisation is to understand the inference engine, arcs will not be interpreted.

It turns out that concepts which are quantified with 'Universal' and 'Existential' tags behave differently from those with 'Individual' tags. These are split into two further primitives universals and individual concepts (nodes).

SemNet is re-defined as:

- a set of universal concepts, CU
- a set of individual concepts, CI

- a set of action concepts, CA
- a set of event concepts, CE (defined in terms of the other concepts)

As indicated in 4.4.1 the natural starting point is to interpret universal concepts as sets, and the entity hierarchy as subset and membership statements.

More formally, postulate a set of objects, $\mathcal{D}$ which form the domain of discourse. A model of SemNet is formed by a set of mapping functions $\mathcal{MS}_{index}$ which map the concepts into the following types[1]:

$$\begin{aligned}
&\mathcal{MS}_u : CU \rightarrow P(\mathcal{D}) \\
&\mathcal{MS}_i : CI \rightarrow \mathcal{D} \\
&\mathcal{MS}_a : CA \rightarrow P(\mathcal{D} \times \mathcal{D}) \\
&\mathcal{MS}_e : CE \rightarrow \{true, false\}
\end{aligned}$$

The hierarchy events map to subset and membership statements:

$$\begin{aligned}
\mathcal{MS}_e(A \succeq_s B) &\mapsto (\mathcal{MS}_u(A) \supseteq \mathcal{MS}_u(B)) \\
\mathcal{MS}_e(a \in A) &\mapsto (\mathcal{MS}_i(a) \in \mathcal{MS}_u(A))
\end{aligned}$$

Basic events take the dual role of making assertions and defining concepts. For example, the nodes in figure 4.1: F, $F_1$, D, and $D_1$ (for FARMER, FARMER1, DONKEY, and DONKEY1) map to subsets of $\mathcal{D}$, and O (for OWNS) into $\mathcal{D} \times \mathcal{D}$. $E_1$ defines $D_1$ as:

$$\mathcal{MS}_u(D_1) = \{x \mid x \in D \wedge \exists y(y \in F \wedge O(y, x))\}$$

and makes the statement:

[1]Where $\mathcal{P}$ returns the power set (i.e. set of subsets) of a set.

$$E_1 = \forall x \exists y (x \in F_1 \rightarrow (y \in D \wedge O(x,y)))$$

In general the assertions and definitions made by an event will depend on the quantifications involved, see figure 5.1.

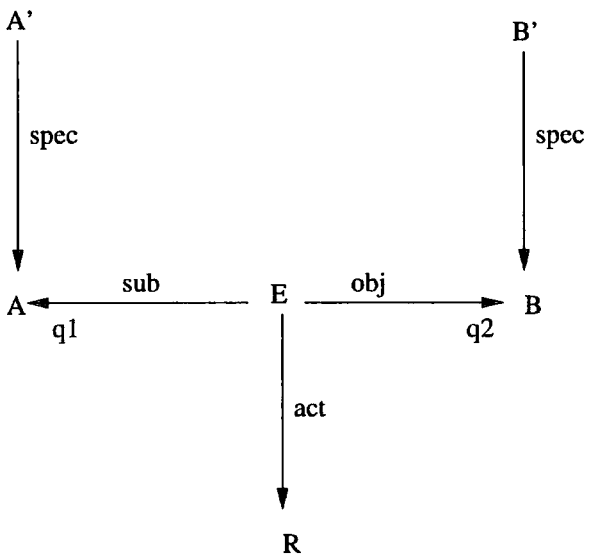



Figure 5.1: General Event Structure.

### Events with a bounded existential

If q1 is universal and q2 is existential, then E is interpreted as:

$$(R, true, A, U, B, E) \mapsto \begin{cases} \text{The statement:} & (\forall x \exists y (x \in A \rightarrow (y \in B \\ & \wedge R(x,y)))) \\ \text{defining A as :} & \{x \mid x \in A' \wedge \exists y (y \in B' \\ & \wedge R(x,y))\} \\ \text{defining B as:} & \{x \mid x \in B' \wedge \exists y (y \in A \\ & \wedge R(y,x))\} \end{cases}$$

Notice that A is defined in terms of $A'$, $B'$ and $R$, whereas the definition of

B uses A. Essentially, A must be built and defined before it can be validly used to build and define B. Without this distinction their definitions would depend, recursively on each other. The event would also be symmetrical.

### Events with an individual

If q1 is universal and q2 is individual, then E is defined as

$$(R, true, A, U, B, I) \mapsto \begin{cases} \text{The statement:} & \exists y \forall x (x \in A) \rightarrow (y \in B' \wedge R(x, y)) \\ \text{defining B as:} & B \in B' \\ \text{defining A as:} & \{x \mid x \in A' \wedge R(x, B)\} \end{cases}$$

Individuals are interpreted as arbitrary, but specified, members of the set meeting the definition provided by the defining event. Named individual concepts map to constants, as they are already defined as being the concept with the specified name.

### Universal-Universal Events

Intuitively events with just universals would map as:

$$(R, true, A, U, B, U) \mapsto \begin{cases} \text{The statement:} & \forall x \forall y ((x \in A) \wedge (y \in B)) \\ & \rightarrow Act(x, y) \\ \text{defining A as:} & \{x \mid x \in X \wedge \forall y \in B \\ & (Act(x, y))\} \\ \text{and B as:} & \{x \mid x \in X \wedge \forall y \in A \\ & (Act(y, x))\} \end{cases}$$

except that the definitions of A and B depend recursively on each other. This is exactly the problem mentioned in section 2.2 and discussed in [Nebel, 1991]. The problem is best highlighted by an example. Consider the graph in figure 5.2.

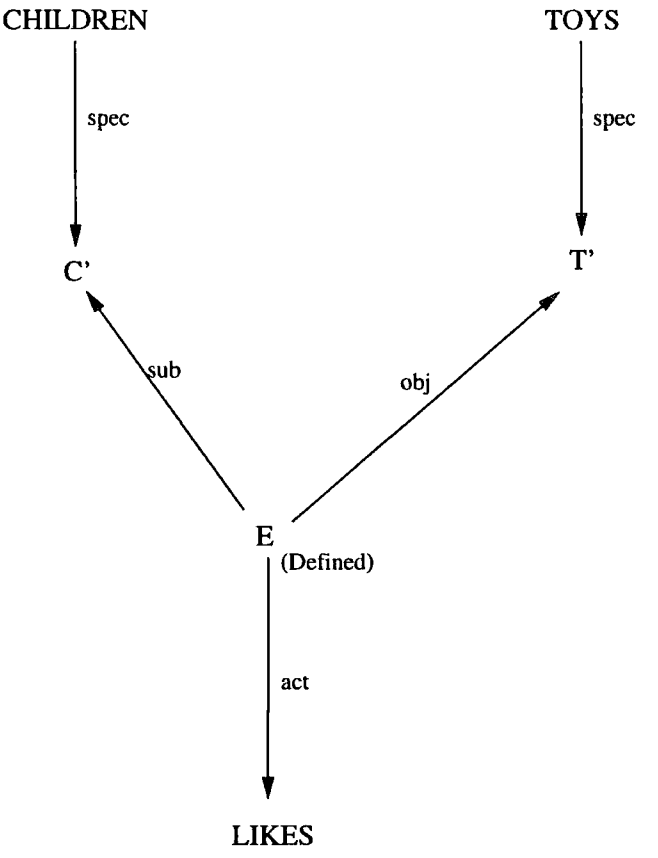



Figure 5.2: Example of ill defined concepts

If the concepts are interpreted as:

$$\begin{aligned} Children &= \{Simon, Daniel, Amanda\} \\ Toys &= \{Car, Soldier, Doll\} \end{aligned}$$

and suppose that:
Simon likes the car and the soldier
Daniel likes the soldier and the doll
Amanda likes the car and the doll

In this case the concepts C' and T' are undefined, and there is no way of inferring whether or not Simon is in C'. Therefore an event should never define two universal nodes.

Events with non-actions map to equivalent formulas, except that the 2-place relation is negated. For example, E1 in figure 4.2 is mapped to:

$$\forall x \exists y (x \in F2 \rightarrow (y \in D2 \wedge \neg O(x,y)))$$

Action spec events map to the subset relation between 2-place relations over $\mathcal{D}$. The logical actions OR and AND map to the usual logical connectives:

$$\begin{aligned} \mathcal{MS}_e(R_1 \succeq_{sa} R_2) &\mapsto \mathcal{MS}_a(R_1) \supseteq \mathcal{MS}_a(R_2) \\ \mathcal{MS}_e(OR, E_1, E_2) &\mapsto \mathcal{M}_e(E_1) \vee \mathcal{M}_e(E_2) \\ \mathcal{M}_e(AND, E_1, E_2) &\mapsto \mathcal{M}_e(E_1) \wedge \mathcal{M}_e(E_2) \end{aligned}$$

As mentioned in chapter 4, the implication action of SemNet is intended to reflect 'relevant' implication. Relevant implication is not truth theoretic and so cannot be captured by classical logic. There is a wide community of researchers working on how the meaning of relevant implication can be captured [Anderson and Belnap, 1975], [Dunn, 1986]. For this project it is observed that the operations performed by SemNet should certainly be sound with respect to material implication. Further analysis will not be considered.

Hypothetical events are mapped as above except that they are not necessarily 'true' in the set theoretic model.

### 5.1.1 Soundness of the Inference Rules

Given this interpretation it is straightforward to show the soundness of the inheritance rules. Each rule has to be 'true' for each event type. For example, the (set theoretic) semantic counterpart rule 4.5 applied to the bounded existential case is:

$$\frac{B \subseteq A, (\forall x \exists y (x \in A \rightarrow (y \in C \wedge R(x,y))))}{\forall x \exists y (x \in B \rightarrow (y \in C \wedge R(x,y)))}$$

We prove this as follows:

**Theorem 5.1.1 (Soundness of Rule 4.5)** *Given any sets A, B and C, if* $B \subseteq A$ *and* $\forall x \exists y (x \in A \rightarrow (y \in C \wedge R(x,y)))$ *then*

$$\forall x \exists y (x \in B \rightarrow (y \in C \wedge R(x,y)))$$

**Proof**

For any x:

$$\begin{array}{rcll} x \in B & \Rightarrow & x \in A & \text{since } B \subseteq A \\ & \Rightarrow & \exists y (y \in C \wedge R(x,y)) & \text{from 2nd assumption} \end{array}$$

and so

$$\forall x \exists y (x \in B \rightarrow (y \in C \wedge R(x,y)))$$

□

## 5.2 Defined and Observed Events

It seems straightforward to extend these semantics to cover defined and observed events. The only difference is to ensure that 'observing' events do not 'define' the nodes they are connected to. In this sense observed events do not contribute to the meaning of nodes, which seems converse to the *meaning as location* principle.

Assuming SemNet is extended to include defined and observed events, the effect on this principle needs to be understood.

For example, returning to the sentence:

"Every mother has a brother each of whom owns a parrot."

The SemNet representation given in figure 5.3 is repeated[2].

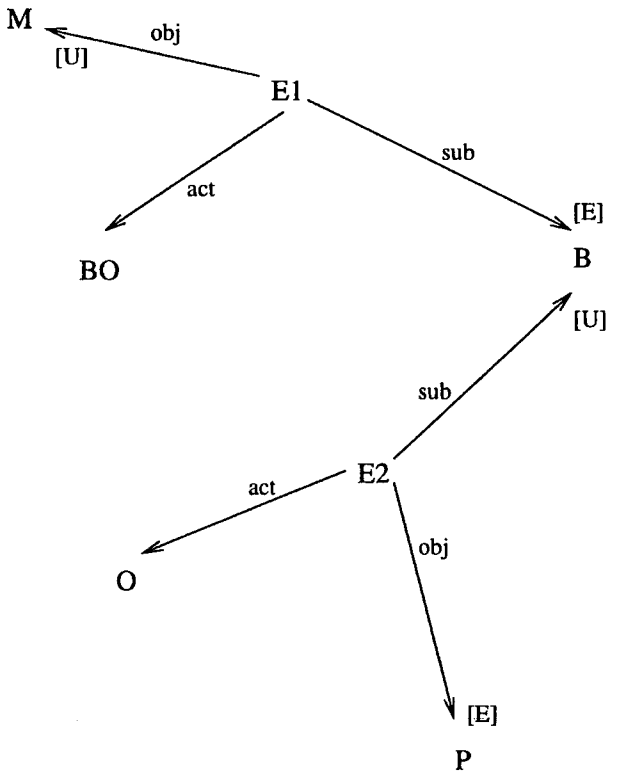



Figure 5.3: SemNet representation of the Brother, Mother Parrot sentence.

The events can now be interpreted as follows:

$$E_1 \mapsto \begin{cases} \text{The statement:} & \forall x \exists y (x \in M \rightarrow (y \in B \wedge BO(x,y))) \\ \text{defining } B_1 \text{ as:} & \{x \mid x \in B \wedge (\exists y (y \in M \wedge BO(x,y)))\} \\ & \text{i.e. brothers of a mother} \end{cases}$$

[2]With M, B, $B_1$, P, $P_1$, O, and BO for MOTHER, BROTHER, BROTHER1, PARROT, PARROT1 OWNS, and BROTHER-OF respectively

$$E_2 \mapsto \begin{cases} \text{The statement:} & \forall x \exists y (x \in B_1 \rightarrow (y \in P \wedge O(x,y))) \\ \text{defining } \mathrm{P}_1 \text{ as:} & \{x \mid x \in P \wedge (\exists y (y \in B_1 \wedge O(y,x)))\} \\ & \text{i.e. Parrots owned by a } B_1 \end{cases}$$

$E_1$ only observes for M and $E_2$ only observes for $B_1$. Otherwise the statements would be about mothers that have brothers, and brothers of mothers that own a parrot.

This example seems to work well, but there are more problematic cases. Consider again the donkey sentence, see figure 5.4.

"Every farmer that owns a donkey beats it"

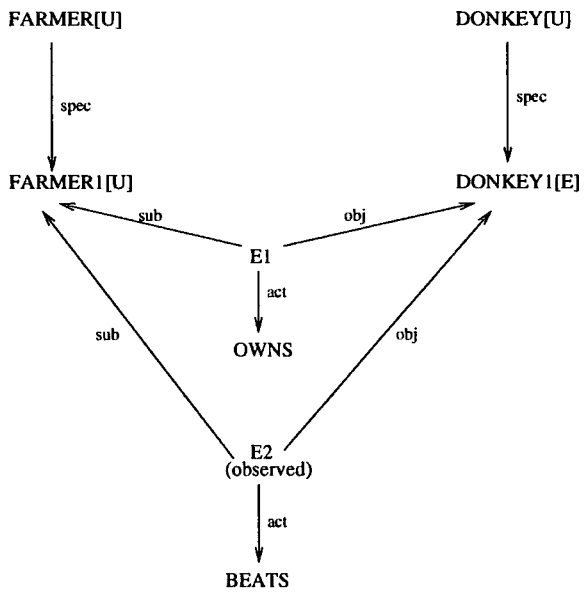



Figure 5.4: SemNet graph for the 'donkey sentence'

which in the model defined would have interpretation:

$$\forall x \forall y (x \in F \wedge y \in D \wedge O(x,y)) \rightarrow B(x,y)$$

where F is the subset of $\mathcal{D}$ corresponding to farmers, D to donkeys, O to Owning

relations and B to Beating relations. As discussed earlier there has been much research into sentences with this construction. The general problem is to find a satisfactory account for why it is necessary to have a universal scoping for the owned (and therefore beaten) donkey, when intuition suggests it should be an existential.

SemNet seems more intuitive by representing the donkeys as existentials. Figure 4.5 is interpreted as:

$$E_1 \mapsto \begin{cases} \text{The statement:} & \forall x \exists y(x \in F_1 \rightarrow (y \in D \wedge O(x,y))) \\ \text{Defines } F_1 \text{ as:} & \{x \mid x \in F \wedge (\exists y(y \in D \wedge O(x,y)))\} \\ & \text{i.e. 'farmers that own a donkey':} \\ \text{Defines } D_1 \text{ as:} & \{x \mid x \in D \wedge (\exists y(y \in F \wedge O(y,x)))\} \\ & \text{i.e. 'donkeys owned by a farmer'} \end{cases}$$

$E_2$ being an observed event makes no definitions. Naively it would be interpreted as:

$$E_2 = \forall x \exists y(x \in F_1 \rightarrow (y \in D_1 \wedge B(x,y)))\}$$

but this corresponds to saying that "every 'farmer that owns a donkey' beats a 'donkey owned by a farmer"'. It does not capture the additional aspect that 'farmers beat the same donkey that they own'. In SemNet there is a rule which allows the owned donkey to be extracted and referred to again. In the model this corresponds to assuming a function f ($:F_1 \rightarrow D_1$) which given a 'donkey owning farmer' will return 'the donkey the farmer owns', so that $E_2$ can be defined as:

$$E_2 = \forall x(x \in F_1 \rightarrow B(x, f(x)))$$

Therefore to replicate the way SemNet builds the meaning of the donkey sen-

tence, a structure with a (witness extracting) function such as f is needed[3].

The formulation used so far does not distinguish statements of necessity. For example, the events of figure 4.6

$$\left\{ \begin{array}{l} E_1 = \forall x(x \in X \rightarrow PLAYS(x, FOOTBALL)) \\ E_2 = \forall x(x \in X \rightarrow STUDIES(x, AI)) \end{array} \right\}$$

are both 'true' statements in the semantics, and thus indistinguishable. To represent this behaviour a richer semantics is required.

## 5.3 Possible World Semantics

The set theoretic semantics defined so far are reasonable for capturing and analysing the extensional aspects of SemNet. However, in chapter 4, it was claimed that SemNet defines concepts by their properties. This is close in spirit to the definition of intensionality as discussed in [Woods, 1991]. To add formality to this claim, the semantic model should also be able to make this distinction. As discussed there are not many operations on SemNet that rely on these distinctions, however, this work now moves from the realm of checking that the past work is well founded, to paving the way for future versions and algorithms which will work on SemNet.

The set theoretic model described in the previous section assumed a single 'real' world, where each proposition is either 'true' or 'false', and each concept can be identified with a single object or set of objects.

The classical way [Meyer and der Hoek, 1995], [Moore, 1995], to define the intension of a concept is to postulate a set of situations (possible worlds), where that concept may have an extension. Two concepts are said to be intensionally equivalent if and only if they have the same extension in all the possible worlds.

[3]This could be done, but in section 7.2.2 it is shown that UTT provides just such a function, naturally.

Often there is a certain world (intuitively the 'real' one, or the one which some agent believes in) distinguished, and two concepts are said to be extensionally equivalent if and only if they have the same extension in this 'real' world.

More formally, postulate a set of world $\mathcal{W}$ and a domain $\mathcal{D}$. The new semantic mapping functions $\mathcal{MPWS}_{index}$ have types:

- $\mathcal{MPWS}_u : CU \rightarrow \mathcal{W} \rightarrow P(\mathcal{D})$
- $\mathcal{MPWS}_i : CI \rightarrow \mathcal{W} \rightarrow \mathcal{D}$
- $\mathcal{MPWS}_a : CA \rightarrow \mathcal{W} \rightarrow P(\mathcal{D} \times \mathcal{D})$
- $\mathcal{MPWS}_e : CE \rightarrow \mathcal{W} \rightarrow \{true, false\}$

A particular world $w^r \in \mathcal{W}$ is distinguished as 'real'. The previous interpretations are now taken to be 'true' in $w^r$.

### 5.3.1 Intensionality of Universals

Returning to the example of figure 4.6, possible world semantics can be used to distinguish the 'intension' of the concept X from its 'extension'.

$$\mathcal{MPWS}_u(X) : \mathcal{W} \rightarrow P(\mathcal{D})$$

The defined event $E_1$ is interpreted as:

$$\forall w \in \mathcal{W}.\forall x(x \in \mathcal{MPWS}_u\ X\ w \rightarrow PLAYS(x, FOOTBALL))$$

i.e. all instances play football in all possible worlds.

The observed event $E_2$ is interpreted as:

$$\forall x(x \in \mathcal{MPWS}_u \; X \; w^r \rightarrow STUDIES(x, AI))$$

i.e. all instances study AI in the 'real' world. A statement is said to be 'necessarily true' if it is 'true' in all possible worlds.

The possible worlds interpretation succeeds in distinguishing the concepts, however, as before there seems to be a difference in the way the semantic language (now possible world structures) distinguish objects, and the way in which SemNet does (by the defining properties of nodes). It would be more ideal if the semantic language could mimic this notion.

A second problem is that possible worlds do have their limitations, (albeit seemingly pathological). For example it is easy to conceive mathematical cases of concepts which have the same extension in all possible worlds, and yet have different intensions, which manifest themselves occasionally.

For example, consider the concepts:

1. The first two natural numbers.
2. The first two powers of 2.

These will both have the same extension in all possible worlds, i.e. $\{1, 2\}$ but they clearly have different intensions.

### 5.3.2 Intensionality of Propositions

As discussed earlier the extension of a proposition is just whether it is true or false. In many cases inference only involves the extension of a proposition (event). As the 'truth' of an event will usually depend only on the truth of other related events. However, there are cases where the intension (the idea formed by a mind using SemNet) of an event is important.

In this case there are algorithms in SemNet which rely on these differences. For example, the epistemic events and associated inference rules. Naively epistemic actions (such as believes) are mapped to relations over the mappings of the events subject and object. If the extensional interpretations of events is used this results in:

$$Believes \mapsto \mathcal{D} \times \{true, false\}$$

so that any events that have the same truth value in the 'real' world are indistinguishable. Clearly this cannot account for the epistemic inference rules.

To give a truth theoretic account for epistemic events, the 'agent' (the concept with the belief) is postulated to 'believe' that a subset of $\mathcal{W}$ are possible (and the rest are not), and the 'agent' believes the event if its interpretation is true in all the worlds the agent deems as possible.

More formally for each agent a (an individual concept, CI) there is a subset of $\mathcal{W}$[4] defined by the semantic mapping function $\mathcal{MPWS}_i : CI \rightarrow P(\mathcal{W})$.

The meaning of an epistemic event can then be defined by the function:

$$\mathcal{M}(Bel, A, I, E_1) = \forall v \in (\mathcal{MPWS}_i(A)).\mathcal{MPWS}_e(E_1, v)$$

Intuitively this states that such an epistemic event is true if and only if $E_1$ is true in all the worlds that the agent A deems possible. All defining events are mapped to statements which are 'true' in all worlds, and so are necessarily believed by all agents. Hypothetical events must be 'true' in some world. Observed events are 'true' in the real world.

The nodes in figure 4.4 can now be interpreted as:

[4]Usually an agent is identified with a relation $R_a$, and there is a notion of accessibility between worlds based on this relation

$$\begin{aligned} Roberto &\mapsto R_{Roberto}(\text{ a subset of } \mathcal{W}) \\ E_1 &\mapsto F(: \mathcal{W} \rightarrow \{true, false\} \\ E_2 &\mapsto \forall v \in R_{Roberto}.F\ v \end{aligned}$$

Intuitively, $E_2$ is 'true' if and only if $E_1$ is 'true' in all the worlds in $R_{Roberto}$.

Inference rules 4.11 and 4.13 can be shown sound with respect to these semantics.

**Theorem 5.3.1 (Soundness of rule 4.11)** *For all A,B,C* $\in CU$

$$\begin{aligned} &(\mathcal{M}(Bel(Agent, A \succeq_s B)) \wedge \mathcal{M}(Bel(Agent, B \succeq_s C))) \\ &\Rightarrow \mathcal{M}(Bel(Agent, A \succeq_s C)) \end{aligned}$$

**Proof**

$$\begin{aligned} & (\mathcal{M}(Bel(Agent, A \succeq_s B)) \wedge \mathcal{M}(Bel(Agent, B \succeq_s C))) \\ \Rightarrow\ & Bel(Agent, (A \subseteq B)) \wedge Bel(Agent, (B \subseteq C)) \\ \Rightarrow\ & \forall w \in R_{Agent}.(A \subseteq B) \wedge \forall w \in R_{Agent}.(B \subseteq C) \\ \Rightarrow\ & \forall w \in R_{Agent}.(A \subseteq C) \\ \Rightarrow\ & Bel(Agent, (A \subseteq C)) \\ \Rightarrow\ & \mathcal{M}(Bel(Agent, A \succeq_s C)) \ \square \end{aligned}$$

**Theorem 5.3.2 (Soundness of rule 4.13.)** *For all events E* $\in CE$*:*

$$\mathcal{M}(Bel(Agent, \neg E)) \Rightarrow \neg(\mathcal{M}(Bel(Agent, E)))$$

**Proof**

$$
\begin{aligned}
& \mathcal{M}Bel(Agent, \neg E) \\
\Rightarrow\ & \forall w \in R_{Agent}.\neg E \\
\Rightarrow\ & \exists w \in R_{Agent}.\neg E \text{ (assuming the agent believes in a possible world)} \\
\Rightarrow\ & \neg(\forall w \in R_{Agent}.E) \\
\Rightarrow\ & \neg\mathcal{M}Bel(Agent, E)\ \square
\end{aligned}
$$

But with rule 4.12 the situation is not so clear. There is nothing in the structure described so far that relates the 'truth' of an assertion in a world, and the 'truth' of an agent 'believing the assertion' in this same world. But there is nothing naturally recursive in the possible worlds framework to capture the 'circularity' of this rule.

The system does not recognise all intensional differences. For example, all tautologies (facts) will be true in all possible worlds and so will be indistinguishable, and yet intuitively:

I understand "2 + 2 = 4"

I understand "Fermats' last theorem"

should have different interpretations. In SemNet when an event is definitional it is a tautology, so possible worlds cannot distinguish between any of these, even when they are hypothetical.

Finally as before, although in general the semantics does succeed in distinguishing, what for SemNet are distinct concepts, it does not do so in a similar way to SemNet. The intuition behind the epistemic events of SemNet is that their meaning comes from their structure and relationship with other concepts rather than requiring an external set of worlds. A better model would map events to propositions which take their meaning more directly from related interpreted concepts.

## 5.4 Distributedness

For ease of notation, the semantic counterpart to any node $X$ will be written as $X^M$.

Although the semantic model of SemNet has a very different structure from SemNet, it does still provide a formal model of the meaning of SemNet. Therefore, in theory, it can be used to analyse the distributedness of SemNet. However, because the model considers SemNet as a list of events (rather than as a set of links), it does not allow for analysis of meaning of subparts of events. Therefore distributedness can only be established at this level of granularity.

The 'full model of SemNet' is defined as 'the conjunction of meanings of the 'real' events of the network'. From now on this will be known as the full model $\mathcal{M}_{full}$.

The 'meaning of a section of SemNet' is defined as the conjunction of meanings of the real events within the section. From now on an interpretation for an arbitrary section 'S', will be written as $\mathcal{M}_{section}(\mathrm{S})$.

Distributedness can be formally defined as:

$$\forall S : Section \mathcal{M}_{full} \Rightarrow \mathcal{M}_{section}(S)$$

The result (at this level) is trivial. Essentially the interpretation of the section postulates the existence of various concepts which $\mathcal{M}_{full}$ provides.

For example, if the immediately adjacent nodes of an existential event are interpreted, $(R, true, A, U, B, E)$, the interpretation is (in the 'real' world):

$$\begin{aligned}
&(A^M, B^M \subseteq \mathcal{D}) \wedge \\
&(R^M \subseteq \mathcal{D}^2) \wedge \\
&\forall x \exists y (x \in A^M \rightarrow (y \in B^M \wedge R^M(x, y))
\end{aligned}$$

and clearly $\mathcal{M}_{full}$ will contain a statement the same as this with $A^M$, $B^M$ and $R^M$ fully specified, so that:

$$\mathcal{M}_{full} \rightarrow \mathcal{M}_{section}(R, true, A, U, B, E)$$

as required.

Similarly for an epistemic event $(Bel, A, E_1)$, the interpretation will be:

$$E_1^M \in Statements \wedge$$
$$A^M \subseteq (\mathcal{W} \times \mathcal{W}) \wedge$$
$$\forall w \in (\mathcal{MPWS}_{agent}(A)) \rightarrow E_1^M(w)$$

for which the full reading will contain the above with $E_1^M$ and $A^M$ fully specified.

### 5.4.1 Arc level analysis

The above analysis was largely trivial, as the section interpretations were conjuncts of the full reading. The intuition behind distributedness went much further, claiming that sub parts of events could be read independently and reliably. To show this, a formal interpretation to the sub-parts of events (i.e. arcs) must be given. It is then shown that if an arc $\alpha$ is part of an event $E$ then $\mathcal{M}(E) \Rightarrow \mathcal{M}(\alpha)$, so that $\mathcal{M}_{full} \Rightarrow \mathcal{M}(\alpha)$ and that SemNet is distributed at the level of the arcs.

An arc is interpreted as a an open proposition which when fully read would form one of a set of alternative propositions. For example interpreting a universal subject arc between and event E and an entity C, gives:

$$\begin{aligned}
\exists R, X: \quad & C, X \subseteq \mathcal{D}, R \subseteq \mathcal{D} \times \mathcal{D} \\
& \forall c(c \in C \rightarrow (\exists x(x \in X \wedge R(x, c))) \\
& \vee(\forall x(x \in X \wedge R(x, c)))) \\
\vee & \\
\exists R, X: \quad & X \in \mathcal{D}, C \subseteq \mathcal{D}, R \subseteq \mathcal{D} \times \mathcal{D} \\
& \forall c(c \in C \rightarrow R(c, X))
\end{aligned}$$

and whatever the full reading is, it will certainly imply the above situation (as it will be an expansion of one of the disjunction of situations).

The above analysis is straightforward, but extremely messy. It is improved if quantification is moved onto the event. In such a situation the controls of the event would state whether it is 'real' or 'hypothetical', 'defined' or 'observed', and what the quantifications of the 'subject' and 'object' are. For example, if E were 'real', 'observed', universal subject and individual object, the interpretation would be:

$$\begin{aligned}
\exists R, X: \quad & X \in \mathcal{D}, C \subseteq \mathcal{D}, R \subseteq \mathcal{D} \times \mathcal{D} \\
& \forall c(c \in C \rightarrow R(c, X))
\end{aligned}$$

The situation becomes complex again if epistemic events are considered, as R and X will have different 'types' if E is epistemic. However, the results are still straightforward. Thus it can be claimed that SemNet is distributed with respect to these semantics.

## 5.5 Review

The classical approach to formalising SemNet has delivered many benefits. It provides a rigourous and unambiguous way of understanding the meaning of SemNet structures. As outlined at the beginning this has resulted, in general, improved

understanding and presentation of SemNet. In addition the problem of ill-defined concepts has arisen.

Soundness of reasoning and a semantic foundation for intensionality have been given. The interpretation given to defined and observed events has shown that their introduction will affect the *meaning as location* principle. Overall problems with the semantic model have been pathological and, perhaps, are not objections that will lead to serious repercussions for NLE.

The main problem with the semantics is that it captures the meaning in a different way from SemNet. The main motivation for building a semantic model in type theory is that there seem to be constructions which could allow for more direct interpretation of SemNet.

# Chapter 6

# Formalisation of SemNet in UTT

This chapter describes how UTT is used in the formalisation of SemNet. It is a major extension of the work described in [Shiu *et al.*, 1996]. Essentially the mechanism for defining theories as described in 3.4 is used to define syntactic (object) and semantic (meta) versions of SemNet, thus allowing modular reasoning. The internal logic of UTT is used to reason directly about these models.

Two further aspects of UTT which are exploited by this work are that it is a programming language and so can be used to reason directly about the implementation of SemNet, and that the logic is supported by the proof assistant Lego [Pollack, 1989].

The aim is very much to show the applicability of the type theory framework, including use of the abstract theory mechanism and Lego, rather than to show that type theory is an appropriate semantic language. This latter task is postponed until the next chapter.

## 6.1 Framework of the Formalisation

Figure 6.1 shows the four versions of SemNet, they are described as:

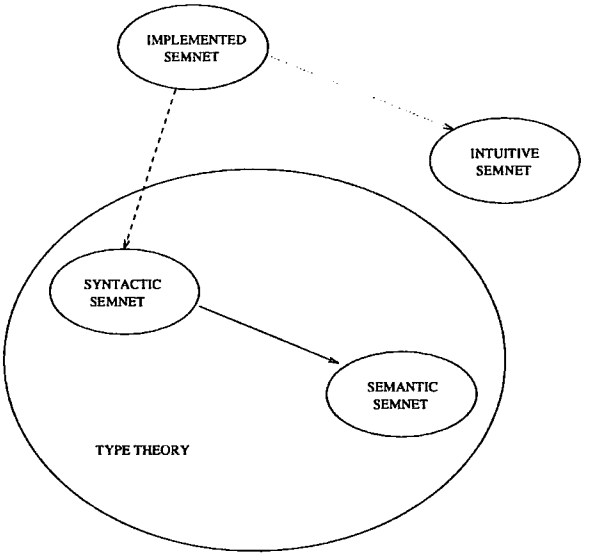



Figure 6.1: Framework for the formalisation

- **Implemented** SemNet refers to the actual Haskell data structures and associated functions that form SemNet in the LOLITA system.

- **Intuitive** SemNet refers to the informal concept of SemNet that exists amongst the LOLITA group members. This is what was described in chapter 4, i.e. the graphical structures and the declarative rules of inference. There is an informal link between this and the implemented SemNet, as this is what the designers had in mind as it was built.

- **Syntactic** SemNet refers to the abstract theory of UTT which defines SemNet in UTT. It consists of declarative rules of inference which correspond to the intuitive SemNet, and algorithms (based on UTT as a programming language) that correspond closely to the implemented SemNet. This latter correspondence is informal but can be considered as closer than the other links as both algorithms are written in functional languages and so can be viewed in terms of lambda calculus.

- **Semantic** SemNet refers to the abstract theory of UTT which formally captures the intended meaning of SemNet in UTT. There are functions of UTT which map between the syntactic and semantic SemNet and so this correspondence is formal.

The UTT models are packaged as 'knowledge base theories' similar to the abstract theory mechanism described in section 3.4. This allows a theory for a simple version of SemNet to be developed, and the results inherited by more complex versions. See figure 6.2.

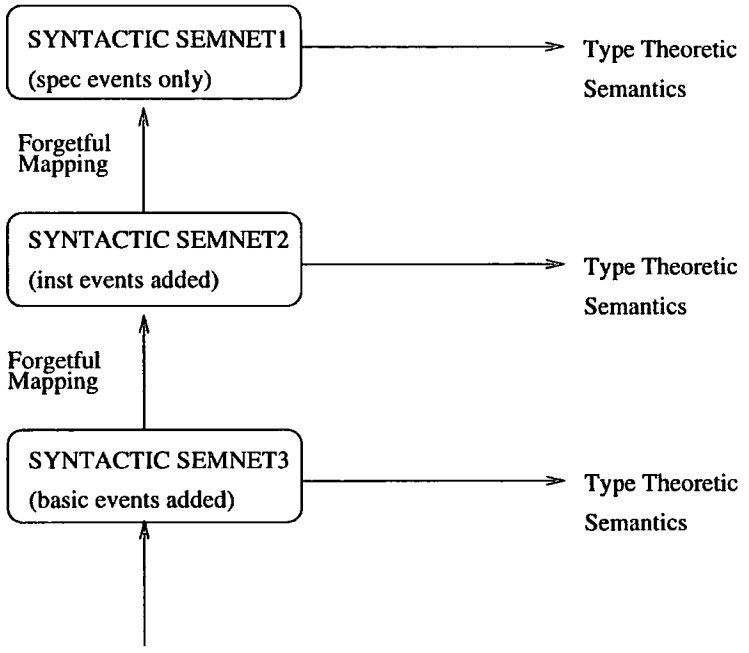



Figure 6.2: Abstract theories of SemNet

The abstract 'knowledge base theory' type is:

$$KBT = \Sigma \left[ \begin{array}{ll} wff: & Type \\ kb: & wff \rightarrow Bool \\ IR: & wff \rightarrow Prop \end{array} \right]$$

The wff type represents objects which make legal statements. The kb type represents a distinguished set of these 'statements' which are explicitly included in the knowledge base. The IR type represents the declarative inference rules for the theory, thus establishing which 'statements' are entailed by the 'knowledge base'. IR will be defined by a set of functions $ir_{index}$ one for each inference rule.

A separate predicate over the KBT type is used to ensure that the knowledge base meets any specified requirements.

$$cond : KBT \rightarrow Prop$$

In this work this is used to ensure that the specialisation events form a legal hierarchy, but it could be used for other properties, e.g. to ensure consistency.

To prove soundness and completeness theorems, it needs to be assumed that the IR functions are the only way in which such propositions can arise. Therefore the rules are defined as inductive relations, and the corresponding elimination rule can be inferred as discussed in 3.1.4.

Functions that (are intended to) implement the inference rules can be converted to functions in UTT (Lego) with type:

$$F : \Pi k : KBT.wff[k] \rightarrow Bool$$

Soundness and completeness of the implementation of the rules are defined by the propositions[1]:

$$\begin{aligned} SoundImp &= \Pi k : KBT.\Pi w : wff[k].iscase(F(k,w)) \rightarrow IR[k](w) \\ CompImp &= \Pi k : KBT.\Pi w : wff[k].IR[k](w) \rightarrow iscase(F(w,k)) \end{aligned}$$

Proving soundness would show that $F$ (the functional algorithm) will only find proofs that follow from $IR$ (the inference rules) and completeness would show that $F$ will find all possible proofs based on $IR$. They should not be confused soundness and completeness with respect to type theoretic semantics.

Type theoretic semantics are defined by meaning functions with types:

[1] $iscase : Bool \rightarrow Prop$

$$\begin{aligned}\mathcal{M}_{kb} &: KBT \rightarrow Prop \\ \mathcal{M}_{wff} &: \Pi k : KBT.wff[k] \rightarrow Prop\end{aligned}$$

Soundness and completeness of the rules with respect to the type theoretic semantics are defined by:

$$\begin{aligned}SoundSem &= \Pi k : KBT.\Pi w : wff[k].IR[k](w) \rightarrow (\mathcal{M}_{kb} \rightarrow \mathcal{M}_{wff}[k](w)) \\ CompSem &= \Pi k : KBT.\Pi w : wff[k].(\mathcal{M}_{kb} \rightarrow \mathcal{M}_{wff}[k](w)) \rightarrow IR[k](w)\end{aligned}$$

**Propositional Calculus Example**

To define a simple version of propositional calculus, (with only negation and implication symbols). The wff type would be defined inductively with two constructors:

$$\begin{aligned} wff &: Type \\ P_i &: wff \text{ (i} \in \text{Nat)} \\ \Rightarrow &: wff \rightarrow wff \rightarrow wff \\ \neg &: wff \rightarrow wff \end{aligned}$$

If the only inference rules are that a wff is explicitly in the knowledge base, or it is inferred by modus ponens, then inference is defined by the inductive relation:

$$\begin{aligned} IR &: wff \rightarrow Prop \\ ir_{base} &: \Pi P : wff.iscase(kb(P)) \rightarrow IR(P) \\ ir_{mp} &: \Pi P, Q : wff.IR(P) \rightarrow IR(\Rightarrow (P, Q) \rightarrow IR(Q) \end{aligned}$$

To prove an implementation of the inference rules, F, is complete there are two cases to prove:

$$\begin{aligned} &\Pi P : wff.iscase(kb(P)) \rightarrow iscase(F(P)) \\ &\Pi P, Q : wff.iscase(F(P)) \rightarrow iscase(F(\Rightarrow (P, Q)) \rightarrow iscase(F(Q)) \end{aligned}$$

and similarly to prove the semantics sound:

$$\begin{aligned} &\Pi P : wff.iscase(kb(P)) \rightarrow (\mathcal{M}_{kb}(kb) \rightarrow \mathcal{M}_{wff}(P)) \\ &\Pi P, Q : wff.IR(P) \rightarrow IR(\Rightarrow (P, Q) \rightarrow (\mathcal{M}_{kb}(kb) \rightarrow \mathcal{M}_{wff}(Q)) \end{aligned}$$

Having defined a (basic) KBT, a richer theory can be defined by extending the wff type.

**Example continued**

Let $pc_1$:KBT be the theory defined earlier and suppose that a new 'wff constructor' is to be added for conjunction:
A new wff type is defined based on $wff[pc_1]$

$$\begin{aligned} wff_2 &: Type \\ \varepsilon &: wff[pc_1] \rightarrow wff_2 \\ \cap &: wff_2 \rightarrow wff_2 \rightarrow wff_2 \end{aligned}$$

Essentially $wff[pc_1]$ is a subtype of $wff_2$ with $\varepsilon$ as coercion. A knowledge base can be defined which extends $kb[pc_1]$ in the natural way. Further inference rules can be added for the 'new' wff's

$$\begin{aligned} &IR[pc_2](P) \\ &ir_{2,andl} : \Pi P, Q : wff[pc_2].IR[pc_2](\cap(P,Q)) \rightarrow IR[pc_2](P) \\ &ir_{2,andr} : \Pi P, Q : wff[pc_2].IR[pc_2](\cap(P,Q)) \rightarrow IR[pc_2](Q) \end{aligned}$$

In this way a new object $pc_2$ : KBT, can be defined which is a natural extension of $pc_1$.

In the formulation of SemNet, wff's defined in later/complex versions do not effect those defined in earlier/simpler versions. Because of this when a version of SemNet is extended to handle richer structures (wff's and inference rules) a new and separate KBT object is defined, and sits alongside the original.

$$KBT_2 = \begin{bmatrix} k_1 & : KBT \\ k_2 & : KBT \end{bmatrix}$$

$k_2$ is defined in terms of objects and structures of $k_1$, but the properties of $k_1$ itself are not affected. Properties proved about $k_1$ are then trivially inherited to objects of type $KBT_2$.

As well as being a formal and declarative description of SemNet, the syntactic model is a formal bridge between the implementation and the type theoretic semantics. It allows a formal mapping to the semantic model to be defined, and for reasoning about (abstracted, but very close) versions of the algorithms used on SemNet.

The semantic SemNet models are a formal representation of the meaning of SemNet structures. They correspond to the set theoretic model developed in chapter 5.

The remainder of this chapter is devoted to describing the structure of these models, although of course extensive appeal to the intuitive SemNet will be made to keep the reader aware of which aspects are being formalised.

## 6.2 A simple Hierarchy - SemNet$_1$

In this model only universal (CU) nodes and specialisation events between them are considered. The $wff$'s will be the links, the structure conditions ensure the links form a hierarchy, and the reasoning rules will correspond to rules 4.1 and 4.2.

### 6.2.1 Syntax for SemNet$_1$

The only primitive needed for this structure will be a node type, CU. Specialisation links are defined as 3-tuples $CU \times CU \times Bool$. The entries representing subject, object and polarity of the statement respectively.

For example, (Man, Mammal, true) represents the statement "men are mammals".

The knowledge base (a function of type $wff \rightarrow Bool$) is implemented via a list of links and a membership function. To form a hierarchy there must be a designated 'top' node and a proof that all the other nodes are 'below' this node.

More formally with $s_1 = SemNet_1$ ( : KBT),

$$
\begin{array}{lcl}
CU & : & Type \\
top_{s_1} & : & CU \\
wff_{s_1} & = & CU \times CU \times Bool \\
K_{s_1} & = & list(wff_{s_1}) \text{ (i.e. a list of 3-tuples)} \\
kb_{s_1}\ (w) & = & member\ w\ K_{s_1} \\
cond_{s_1}\ K & = & P(K_{s_1}, top_{s_1})
\end{array}
$$

where, $w$ is of type $wff_{s_1}$, member is the usual list membership function, $P$ is a predicate that ensures $K$ forms a hierarchy and that $top_{s_1}$ is the top node.

The inference rules (corresponding to the declarative rules 4.1 and 4.2) are defined inductively:

$$
\begin{array}{lcl}
IR_{s_1} & : & wff_{s_1} \rightarrow Prop \\
ir & : & \Pi w : wff_{s_1}.(iscase(kb_{s_1}(w))) \rightarrow IR_{s_1}(w) \\
ir_1 & : & \Pi A, B, C : CU.(IR_{s_1}(A, B, true)) \rightarrow (IR_{s_1}(B, C, true)) \\
 & & \rightarrow (IR_{s_1}(A, C, true)) \\
ir_2 & : & \Pi A, B, C : CU.(IR_{s_1}(A, B, true)) \rightarrow (IR_{s_1}(A, C, false)) \\
 & & \rightarrow (IR_{s_1}(B, C, false))
\end{array}
$$

$ir$ states that any event present in the knowledge base can be inferred. $ir_1$ states that 'positive' spec links are transitive. $ir_2$ states that 'negative' spec links can be inferred in a similar way[2]

### 6.2.2 Implementation Analysis for SemNet$_1$

The algorithms for adding Links to SemNet and for inferring Links from SemNet can be given counterparts in Lego. A problem in converting algorithms is that

[2]The Lego module for handling inductive types does not pattern match against the link type, and so a function for building the link type is used.

UTT is a decidable language and all algorithms must terminate. Therefore general recursive functions need to be converted (if possible) to well founded versions [Wand, 1992], [Hoffman, 1992].

In this work the method used (as described in [Wand, 1992]) is to provide an explicit complexity measure which reduces on each recursive call to the function.

With this in mind the structural condition type for $\text{SemNet}_1$ included a complexity measure function:

$$w_{s_1} : CU \rightarrow Nat$$

which has constraints:

$$w_{s_1}\ top = 1$$
$$\Pi A.w_{s_1}\ A < w_{s_1}\ (Gen\ A)$$

where Gen :CU $\rightarrow$ CU, is a function defined on $\text{SemNet}_1$ which returns the node immediately above in the hierarchy.

$w$ is used by all functions that search up and down the hierarchy recursively.

For example, in pseudo Haskell notation:

$$\begin{aligned} AllUp: &\ CU \rightarrow [CU] \\ AllUp &\ top = [top] \\ AllUp &\ A = cons\ A\ (AllUp\ (Gen\ A) \end{aligned}$$

is actually implemented using $w$ on each CU to ensure that the recursive function will reach the top node and terminate.

The inference function is defined as:

$$
\begin{array}{lll}
Inf_{s_1}: & wff_{s_1} & \rightarrow Bool \\
Inf_{s_1} & (A, B, true) & = A \in AllUp\ B \\
 & (A, B, false) & = A \in AllDown\ (NotGen\ B)
\end{array}
$$

where $AllDown : CU \rightarrow [CU]$ searches down rather than up the hierarchy, and $NotGen : CU \rightarrow CU$ returns a node (if there is one) which is explicitly not its generalisation.

### 6.2.3 Semantics for SemNet$_1$

Previously CU nodes were interpreted as sets and spec links as subset relations. Analogously in type theory nodes will be interpreted as types and the links as subtype judgements.

As discussed in chapter 3 subtyping is a current research topic for type theory. In this section when one type A, is said to be a subtype of another B, B $\succeq$ A, then it is assumed that there is an implicit coercion $\kappa$ : A $\rightarrow$ B between them and a:A, can operate as an object of type B, without the need to specify that the object used is really $\kappa$(a), [Luo, 1996].

Intuitively each node is interpreted in terms of its parent node and the events its involved in (not present in this case). The top node is interpreted as an arbitrary type T. Since the events are not present, the predicates can only be implicit. They are defined by an implicit function $ImpPred : CU \rightarrow (T \rightarrow Prop)$.

All nodes are then defined in terms of their implicit predicate and the predicates of all their parents. More formally:

$$
\begin{array}{lll}
\mathcal{M}_u & : & CU \rightarrow Type \\
\mathcal{M}_u\ top & = & T \\
\mathcal{M}_u\ n & = & \Sigma x : T.(\mathcal{M}_{p1}\ \mathrm{n})\ \mathrm{x}
\end{array}
$$

where

$$\begin{aligned} \mathcal{M}_{p1} &: \quad CU \rightarrow (T \rightarrow Prop) \\ \mathcal{M}_{p1}\ top &= \lambda x : T.true \\ \mathcal{M}_{p1}\ A &= \lambda x : T.(\mathcal{M}_{p1}\ ((Gen\ x)\ x)) \wedge (ImpPred\ A\ x) \end{aligned}$$

From now on, where the meaning is clear, subscripting will be used to represent the semantic interpretation of syntactic structures. For example, $A_m$ will represent $\mathcal{M}_u$ A, and $A_{pred}$ will represent $ImpPredA$.

**Intuitive Example**

The node for MAMMAL which appears below animal in the hierarchy (see figure 4.10) will be interpreted as:

$$MAMMAL_m = \Sigma x : T.MAMMAL_{pred}(x)$$

i.e. an instance of a MAMMAL is an object of type T, together with a proof that this object meets all the requirements for being a MAMMAL. The requirements are encoded in $MAMMAL_{pred}$ which is defined as:

$$MAMMAL_{pred} = \lambda x : T.ANIMAL_{pred}(x) \wedge Pred_{MAMMAL}(x)$$

where $ANIMAL_{pred}$ specifies conditions for being an animal (i.e. an organised being endowed with life, sensation and voluntary motion), and $MAMMAL_{pred}$ specifies further conditions for being a mammal (i.e. has mammae for nourishment of young).

Intuitively $\mathcal{M}_{p1}$ should map to a predicate, i.e. T $\rightarrow$ Prop, but in some cases a witness extraction function is required, see section 7.2. Where possible a shorthand notation for expressing semantic counterparts of concepts, i.e. the subscripted notation given above.

The coercive function between the meaning of a node and its 'parent' is a forgetful map that forgets the node specific predicate.

The links of the syntax are the statements and so map to objects of type Prop. As constructed they map to the judgement that the two types are in a subtype relation, or not as the case may be. This is a 'higher truth' in the sense that such judgements are axioms in type theory, however it is helpful for the analysis if a proposition can be extracted from the judgement, and so links are interpreted as follows:

$$\begin{aligned} \mathcal{M}_{spec} &: wff_{s_1} \rightarrow Prop \\ \mathcal{M}_{spec}\ (A, B, true) &= \Pi x : B_m . B_{pred}\ \mathrm{x} \rightarrow A_{pred}\ \mathrm{x} \\ \mathcal{M}_{spec}\ (A, B, false) &= \exists x : B_m . (B_{pred}\ x) \wedge \neg (A_{pred}\ x) \end{aligned}$$

The intuition here is that when a spec occurs the predicate of a node will entail that of its parent (clearly true for mammals and animals, see above example).

The meaning of the whole network is the conjunction of the meanings of the links of the knowledge base.

$$\begin{aligned} \mathcal{M}_{kb} &: K_{s_1} \rightarrow Prop \\ \mathcal{M}_{kb}\ k &= \bigwedge_{\forall w \in k} w_m \end{aligned}$$

### Soundness of SemNet$_1$

Soundness of the inference rules with respect to the semantics is characterised in the following:

**Theorem 6.2.1 (Soundness of Spec Inheritance)**

$$\Pi w : wff_{s_1} . IR_{s_1}(w) \rightarrow K_{s_1,m} \rightarrow w_m$$

**Proof**

This is an informal version of the Lego proof.

Assume (introduce) $k : kb_{s_1}$, $w : wff_{s_1}$. Proceed by induction on the structure of $IR_{s_1}$. There are 3 cases:

**Case 1 :** $ir$

$ir \Rightarrow$ iscase $(kb_{s_1}\ w)$

$\Rightarrow \mathrm{w}_m \in \mathrm{k}_m$

$\Rightarrow \mathrm{k}_m \rightarrow \mathrm{w}_m$

**Case 2 :** $ir_1$ w = (A,C,true)

$ir_1 \Rightarrow \exists$B:CU.(IR (A,B,true) and IR (B,C,true))

$\Rightarrow \mathrm{k}_m \rightarrow$

$(\mathrm{A,B,true})_m \wedge (\mathrm{B,C,true})_m$

$\Rightarrow \mathrm{k}_m \rightarrow$

$(\Pi\ \mathrm{x{:}B}_m.(\mathrm{B}_{pred}\ \mathrm{x}) \rightarrow (\mathrm{A}_{pred}\ \mathrm{x})$

$\wedge$

$(\Pi\ \mathrm{x{:}C}_m.(\mathrm{C}_{pred}\ \mathrm{x}) \rightarrow (\mathrm{B}_{pred})\ \mathrm{x})$

$\Rightarrow \mathrm{k}_m \rightarrow$

$(\Pi\ \mathrm{x{:}C}_m.(\mathrm{C}_{pred}\ \mathrm{x}) \rightarrow (\mathrm{A}_{pred}\ \mathrm{x})$

$\Rightarrow \mathrm{k}_m \rightarrow \mathrm{w}_m$

**Case 3 :** $ir_2$ similar to case 2! □.

**Completeness of SemNet$_1$**

The Completeness theorem is characterised in:

$$\Pi w : wff[s_1].(K[s_1]_m \rightarrow w_m) \rightarrow IR[s_1](w)$$

This has not been proved. Without this result it is not known that all entailments (that follow from the rules of UTT) would be realised by the *spec inference* rules of $\text{SemNet}_1$. Lacking this result does not, of course, invalidate the meaning of the soundness result above, i.e. that all proofs in $\text{SemNet}_1$ are valid with respect to UTT.

### 6.2.4 $\text{SemNet}_1$ discussion

$\text{SemNet}_1$ is a concrete realization of the framework shown in figure 6.1. There has been a thorough attempt to analyse all the aspects discussed in 6.1. There are syntactic and semantic versions of the entity hierarchy of 4.6.1 with functions between them providing a formal interpretation, and algorithms that correspond to the implemented code of LOLITA. The logic of UTT has been used to reason directly about this model.

The semantics defined may seem disproportionately complicated with respect to the intuitively simple entity hierarchy, especially when compared with the straightforward subset interpretation. The motivation for using type theory was to exploit its 'intensionality'. Because it is intensional it is less natural to use it to model the 'extensional' behaviour of inheritance. However, the soundness result shows that the interpretation does fit with the intuition, and that it is reasonable to build further semantic structure to reason about richer aspects of SemNet. This has significance beyond pure inheritance as these rules are used many times over by other inference algorithms and other modules.

In this model it has been possible to build counterparts to LOLITA functions and to reason about their behaviour. An abstracted version of the inference algorithm for *spec* events has been built and is available for analysis. However, the problem of dealing with general recursion adds a significant amount of work to this process. Since more complex structures can only mean more problems, and this for code that is used less often, it was decided, based on pragmatic grounds, no further implementation analysis would be done.

## 6.3 Individuals and Instance events - $SemNet_2$

Intuitively this abstraction is $SemNet_1$ with instance events and individual concepts. The only new primitive is the Individual node type (CI). The new $wff$ type 'Instance' links will also be defined as 3-tuples, $CU \times CI \times Bool$, for example (Man,Roberto,true) represents the statement "Roberto is a man". The new knowledge base will be a list of the new link type, there are no structural conditions. The inference rules will correspond to rules 4.3 and 4.4.

### 6.3.1 $SemNet_2$ Syntax

More formally:

$$\begin{aligned} CI &: Type \\ S_2 &= (s_1, s_2) \end{aligned}$$

where $s_1$ is defined as in 6.2 and:

$$\begin{aligned} s_2 &: KBT \\ wff_{s_2} &= CU \times CI \times Bool \\ K_{s_2} &: [wff_{s_2}] \\ kb_{s_2}\ \mathrm{w} &= Member\ w\ K_{s_2} \end{aligned}$$

The new rules corresponding to rules 4.3 and 4.4 are defined by :

$$
\begin{array}{lll}
IR_{s_2} & : & wff_{s_2} \rightarrow Prop \\
ir_{2,basic} & : & \Pi w : wff_{s_2}.iscase(kb_{s_2}(w)) \rightarrow IR_{s_2}(w) \\
ir_3 & : & \Pi A, B : CU.\Pi a : CI.IR_{s_2}(A, a, true) \rightarrow IR_{s_1}(B, A, true) \\
 & & \rightarrow IR_{s_2}(B, a, true) \\
ir_4 & : & \Pi A, B : CU.\Pi a : CI.IR_{s_2}(A, a, false) \rightarrow IR_{s_1}(A, B, true) \\
 & & \rightarrow IR_{s_2}(B, a, false)
\end{array}
$$

Note that $IR_{s_2}$ is defined in terms of $IR_{s_1}$ but not vice versa. Precisely the same functions are used to implement inference on $s_1$, so that results are clearly inherited.

$$Inf_{1,S_2} = Inf_1\ (\pi 1(S_2)))$$

New functions inferring Inst links are defined:

$$Inf_{2,S_2} : wff_{s_2} \rightarrow Bool$$

### 6.3.2 Semantics for SemNet$_2$

Continuing the analogy with set theory, individuals will be interpreted as objects (of types) and Inst events as typing judgements. Once again there will be some work to extract propositions out of the judgements.

The individuals are interpreted as being (arbitrary, but specified) objects of their 'parent' type, but which are also objects of an implicit subtype defined by the defining event. This assumes that this subtype, is non-empty and that an arbitrary member can be selected. This is not possible for all types (or else all propositions would be provable and the logic inconsistent). However the type of non-empty types can be defined:

$$NonEmpty = \Sigma T : Type.T$$

and a choice function C can be defined for this type:

$$\begin{aligned} C &: \Pi N : NonEmpty.N \\ C\ N &= \pi 2\ \mathrm{N} \end{aligned}$$

Throughout this section it will be assumed that all Nodes with an instance link will be interpreted as *NonEmpty* types so that the function C can be applied.

The semantic functions are defined as:

$$\begin{aligned} \mathcal{M}_{u,S_2} &: CU \rightarrow Type \\ \mathcal{M}_{u,S_2}\ A &= \mathcal{M}_{u,s_1}\ A \end{aligned}$$

$$\begin{aligned} \mathcal{M}_{type,S_2} &: CI \rightarrow Type \\ \mathcal{M}_{type,S_2}\ a &= \Sigma x : T.a_{pred}\ x \end{aligned}$$

$$\begin{aligned} \mathcal{M}_{i,S_2} &: \Pi x : CI.\mathcal{M}_{type,S_2}\ x \\ \mathcal{M}_{i,S_2}\ a &= C\ a_{type} \end{aligned}$$

$$\begin{aligned} \mathcal{M}_{s,S_2} &: wff[s_1] \rightarrow Prop \\ \mathcal{M}_{s,S_2}\ w &= \mathcal{M}_{s,s_1}\ w \end{aligned}$$

$$\begin{aligned} \mathcal{M}_{I,S_2} &: wff_{s_2} \rightarrow Prop \\ \mathcal{M}_{I,S_2}\ (A, a, true) &= a_{pred}\ (a_m) \\ \mathcal{M}_{I,S_2}\ (A, a, false) &= \neg a_{pred}\ (a_m) \end{aligned}$$

where $a_{pred}$ is given by an implicit function for the implicit defining event.

**Intuitive Example**

Taking an instance of the CU node DOG (with name FIDO, as in figure 4.10).

$$FIDO_m = C(\Sigma x : T.FIDO_{pred}(x))$$

where

$$FIDO_{pred} = \lambda x : T.DOG_{pred}(x) \wedge Pred_{FIDO}(x)$$

i.e. $FIDO_m$ is an object of type T, together with a proof that this object 'obeys' the 'being a dog' predicate and the 'being FIDO' predicate.

Theorems proved for $\text{SemNet}_1$ are inherited as discussed. Further theorems for $ir3$ and $ir4$ are that the code implements them soundly and that they are sound with respect to the type theoretic semantics.

### 6.3.3 $\text{SemNet}_2$ discussion

As discussed in 6.2.3 no further analysis of LOLITA algorithms has been done. However, the new rules defined by $IR[s_2]$ are defined in terms of $IR[s_1]$ and the results of analysing the implementation of $IR[s_1]$ are relevant to this model.

There are two comments to make on the semantics. Firstly, to note that this semantic model merely extends the hierarchy with further extensional aspects. The domain $\mathcal{D}$ of set theory has been replaced with an arbitrary type T. However, it is interesting that a 'choice' function has been required. Up until now types have been entirely arbitrary, but by specifying that some types are inhabited, and that arbitrary objects can be chosen from these types two assumptions have been made.

First there is an commitment that these non-empty types correspond to objects that 'exist in the real world' in some sense. Second, the use of a choice function

corresponds to assuming the 'axiom of choice' of ZF set theory [Fraenkel and Bar-Hillel, 1958]. This axiom is provable within type theory [Martin-Lof, 1982] (since it is constructive). The point is noted as an aspect of SemNet that assumes some essence of 'constructivism'. In chapter 7, further observations along these lines are made.

## 6.4 Standard Events - SemNet$_3$

The next abstraction includes the basic events. Basic events are all defining and real. They include all the possible quantification combinations.

There are two new primitive types CA and CE. There is a choice in level of abstraction in deciding how to build the syntactic statements ($wff$'s). Either to ignore the substructure of events and define SemNet$_3$ as a list of events or to model the implemented SemNet directly, as a set of subject, object and action links, with events extracted from out of these links.

Choice 1 has the advantage of comprehend-ability and manageability.

Choice 2 allows for proper modelling, but intuition is lost. Also the rules were specified for events not for links, and so to keep with the knowledge base structure events must be abstracted out.

As an attempt to get the best of both worlds, the model is first detailed as a set of links, as it is done in LOLITA. Events are then abstracted out of this structure and these form the knowledge base.

### 6.4.1 SemNet$_3$ - Syntax

More formally:

$$CA : Type$$
$$CE : Type$$

$$SemNet_3 = S_3 = (s_1, s_2, s_3) : KBT_3$$

where $s_1$ and $s_2$ are defined as in 6.2 and 6.3 and $s_3$ is defined as follows:

Links are defined inductively

$$\begin{aligned} Link &: Type \\ Subject &: CE \rightarrow (CU \mid CI) \rightarrow Q \rightarrow Link \\ Object &: CE \rightarrow (CU \mid CI) \rightarrow Q \rightarrow Link \\ Role &: CE \rightarrow CA \rightarrow Bool \rightarrow Link \end{aligned}$$

with Q a type for quantification tags. A Net is then a sum type:

$$Net = \Sigma n : (list(Link)).P(n)$$

where P is a predicate that ensures that the links form legal events.

An inductive type (for what will be the new $wff_{s_3}$'s) is defined:

$$
\begin{aligned}
Event &: Type(0) \\
Euu &: CU \rightarrow CU \rightarrow CA \rightarrow Bool \rightarrow Event \\
Eue &: CU \rightarrow CU \rightarrow CA \rightarrow Bool \rightarrow Event \\
Eeu &: CU \rightarrow CU \rightarrow CA \rightarrow Bool \rightarrow Event \\
Eiu &: CI \rightarrow CU \rightarrow CA \rightarrow Bool \rightarrow Event \\
Eui &: CU \rightarrow CI \rightarrow CA \rightarrow Bool \rightarrow Event \\
Eii &: CI \rightarrow CI \rightarrow CA \rightarrow Bool \rightarrow Event
\end{aligned}
$$

The knowledge base $\mathrm{kb}_{s_3}$ will be extracted from $\mathrm{Net}_{s_3}$ as the set of events that are explicitly present. The new inference rules, corresponding to rules 4.5 - 4.8, are defined inductively. For example for rule 4.5:

$$
\begin{aligned}
IR_{s_3} &: Event \rightarrow Prop \\
ir_5 &: \Pi A, B, C : CU.\Pi R : CA.\Pi bb : Bool. \\
&\quad (IR_{s_3}\ \text{(Euu A C R bb)}) \rightarrow (IR_{s_1}\ (A, B, true)) \\
&\quad \rightarrow IR_{s_3}\ \text{(Euu B C R bb)}
\end{aligned}
$$

Note that this rule uses, but does not affect, IR[$s_1$]. The previous functions for inferring on the hierarchy can be used. In theory inference functions could be built for this model, but for pragmatic reasons this has not been done, see discussion at the end of this chapter.

### 6.4.2 SemNet$_3$ Semantics

As with the set theoretic model, actions will be left as implicit relations (over the top type T).

$$\mathcal{M}_3 : CA \rightarrow (T \rightarrow T \rightarrow Prop)$$

In this interpretation CU and CI will be interpreted as before, except that the events will be used to build the $\mathcal{M}_p$ predicates. Events will be interpreted as propositions. More formally:

$$\begin{array}{lcl}
\mathcal{M} & : & Event \rightarrow Prop \\
\mathcal{M}\ (Euu\ \text{A B R bb}) & = & \Pi x : A_m.\Pi y : B_m. \\
 & & R_m\ \text{x y} \\
\mathcal{M}\ (Eue\ \text{A B R bb}) & = & \Pi x : A_m.\exists y : B_m. \\
 & & R_m\ \text{x y} \\
\mathcal{M}\ (Eeu\ \text{A B R bb}) & = & \Pi y : B_m.\exists x : A_m. \\
 & & R_m\ \text{x y} \\
\mathcal{M}\ (Eiu\ \text{a A R bb}) & = & \Pi x : A_m.R_m\ a_m\ x \\
\mathcal{M}\ (Eui\ \text{A a R bb}) & = & \Pi x : A_m.R_m\ \text{x}\ a_m \\
\mathcal{M}\ (Eii\ \text{a b R bb}) & = & R_m\ a_m\ b_m
\end{array} \tag{6.1}$$

As well as all the quantification cases there will be separate functions defining predicates, dependent on whether the concept is use as a subject or an object by the defining event. Its type will be:

$$Pred_1 : Event \rightarrow (T \rightarrow Prop)$$

assuming that the concept is the subject of an event which has 'universal subject' and 'existential object' then the predicate is defined as:

$$Pred_1(Eue(A, B, R, bb) = \Pi x : T.\exists y : B_m.R_m(x, y) \tag{6.2}$$

**Intuitive Example**

For example, consider again the nodes in figure 4.1,

$$E1_m \quad = \quad \Pi x : F1_m.\exists y : D1_m.O_m\ x\ y$$

where

$$F1_m = \Sigma x : T.F_{pred}(x) \wedge (\exists y : D_m.O_m(x,y))$$
$$D1_m = \Sigma x : T.D_{pred}(x) \wedge (\exists y : F_m.O_m(y,x))$$

i.e. an object of type $F1_m$ is an object of type T paired with a proof that it 'obeys' the predicate for 'being a farmer' (i.e. $F_{pred}$) and it 'obeys' the predicate for 'owning a donkey' (as defined by the event E1).
$E1_m$ makes the statement that each of these objects 'owns' an object of type $D_1$. It does this without using any logical connectives such as $\rightarrow$ and $\wedge$ which were used in the set theoretic counterpart, see section 5.1.

Soundness of the basic rules 4.1 and 4.2 can be 'inherited' from the results of SemNet1 and SemNet2.

**Theorem 6.4.1 (Soundness of rule universal inheritance)**

$$\Pi A, B, C : Node.\Pi R : CA.\Pi bb : Bool. \quad (\mathcal{M}(Euu\ A\ C\ R\ bb)) \rightarrow (\mathcal{M}(A, B, true)) \rightarrow \mathcal{M}(Euu\ B\ C\ R\ bb)$$

**Proof**

This is an informal version of the Lego proof.

Assume (introduce) A,B,C:CU; bb:Bool;
$H1 : \Pi x : A_m \Pi y : C_m.R_m(\pi 1(x), \pi 1(y))$;

$H2 : \Pi x : TB_{pred}(x) \rightarrow A_{pred}(x)$;

$b : B_m = \Sigma x : T.B_{pred}(x)$;

$c : C_m = \Sigma x : T.C_{pred}(x)$

The new goal is to prove (find a term of type) :

$R_m(\pi 1(b), \pi 1(c))$

From the assumptions we can build an object of type $A_m$:

$a = (\pi 1(b), H2(\pi 1(b), \pi 2(b)))$

and then

$H1(a, c) : R_m(\pi 1(b), \pi 1(c))$

as required □.

### 6.4.3 SemNet$_3$ discussion

In this model the previously implicit predicates have been made explicit (by being defined in terms of the defining events). The only remaining 'implicit' objects are the postulated domain type T, and the relations over these ($T \times T \rightarrow Prop$) that correspond to actions.

Although the hierarchy interpretations may have seemed less intuitive than their set theoretic counterparts, the reverse could be said for basic event interpretations. For example, there are no logical connectives used in an interpretation for a basic event. This is expanded in chapter 7.

Intuitively theorem 6.4.1 is straightforward. It states that if something holds for all objects of a type, then it holds for all objects of subtypes of this type. It should be straightforward, as the intuition behind the inheritance is straightforward. The result gives further evidence that the semantic model given does actually capture the intended meaning of SemNet.

The only change required to handle defined and observed events in the above

model is to label some links[3] as observed. Semantically, the corresponding events would be mapped to propositions in the same way, but would produce vacuous (i.e. always true) predicates so that the interpretation of entities is not effected. Because of the problem of recursive definitions (see section 5.1), Euu events cannot be defining for both the subject and the object.

## 6.5 Epistemic events - SemNet$_4$

To treat epistemic events as a new $wff$ type, there must be a new primitive type, CEE and a corresponding new link type for those links involving CEE nodes. As with SemNet$_3$ a new inductive type is defined and legal epistemic events are extracted from a set of 'new' links.

Correspondingly cond$_{s_4}$ will need rules to ensure that a given network (list of links) corresponds to a set of legal epistemic events. Finally of course IR$_{s_4}$ will define rules corresponding to 4.11.

### 6.5.1 SemNet$_4$ Syntax

More formally:

$$CEE \ : Type$$
$$CEA \ : Type$$

The new link type is defined inductively:

[3]Since an event could be defining for its subject, but not for its object.

$$
\begin{aligned}
&EpLink && : Type \\
&EpSubject && : CEE \rightarrow (CU \mid CI) \rightarrow EpLink \\
&EpObject && : CEE \rightarrow (CU \mid CI) \rightarrow EpLink \\
&EpAction && : CEE \rightarrow CEA \rightarrow EpLink
\end{aligned}
$$

$\text{wff}_{s_4}$ = EpEvent where EpEvent is defined inductively as:

$$
\begin{aligned}
&EpEvent && : Type \\
&EEspec && : CI \rightarrow wff_{s_1} \rightarrow Bool \rightarrow EpEvent \\
&EEinst && : CI \rightarrow wff_{s_2} \rightarrow Bool \rightarrow EpEvent \\
&EEbasic && : CI \rightarrow CE \rightarrow Bool \rightarrow EpEvent \\
&EErec && : CI \rightarrow CEE \rightarrow Bool \rightarrow EpEvent
\end{aligned}
$$

The inference rules for 4.11, 4.12 and 4.13 are defined by:

$$IR_{s_4} : EpEvent \rightarrow Prop$$

$$
\begin{aligned}
ir_{11} \quad : \quad & \Pi a : CI.\Pi A, B, C : CU.IR(EEspec\ a\ (A, B, true)\ true)) \rightarrow \\
& IR(EEspec\ a\ (B, C, true)\ true) \rightarrow \\
& IR(EEspec\ a\ (A, C, true)\ true))
\end{aligned}
$$

$$
\begin{aligned}
ir_{12} \quad : \quad & \Pi a : CI.\Pi e : CE.IR\ (\text{EEbasic a e true}) \rightarrow \\
& IR\ (\text{EErec a (EEbasic a e true) true})
\end{aligned}
$$

$$
\begin{aligned}
ir_{13} \quad : \quad & \Pi a : CI.\Pi e : CE.IR\ (\text{EEbasic a } \neg \text{e true}) \rightarrow \\
& IR\ (\text{EEbasic a e false})
\end{aligned}
$$

The semantics of $\text{SemNet}_4$ involves exploiting the intensionality of Prop. It is postponed to the next chapter.

## 6.6 Review

Performing a formalisation of this size is a large and complicated task. This chapter has shown that the techniques and tools of type theory can help to break such problems and allow modular and machine assisted development of ideas and proofs.

The closeness of UTT code and Haskell code did allow for some formal analysis of LOLITA algorithms, although because of the difficulties of recursion, this is only feasible from an engineering standpoint if there is a serious need.

The basic syntactic and semantic models have now been defined. Chapter 7 takes the semantic model and the motivation that there are many similarities between SemNet and UTT to analyse some of the richer aspects of SemNet.

# Chapter 7

# Formal type theoretic semantics

As outlined in the opening chapters, much of this research was motivated by the idea that the constructive and intensional aspects of type theory would make it a suitable semantic language for analysing SemNet. This chapter describes how, because of these aspects, the semantic models developed in chapter 6 are able to model SemNet more directly than set theory.

## 7.1 Type theoretic intuitions

The starting point for building a set theoretic model in chapter 5, was the intuition that the entity hierarchy formed a set hierarchy. If instead events are considered first, it is observed that the interpretations are mainly of the form:

$$\forall x.x \in A \ \rightarrow \ (...)$$
$$\exists x.x \in A \ \wedge \ (...)$$

This suggests that the underlying form for the logic is sorted, and that the intended statement is about all objects of 'sort A' rather than all objects in some universe. In chapter 6, this more intuitive interpretation has been achieved.

It may seem that the subtyping scheme described in chapter 6 is more complex and less intuitive than viewing the hierarchy as a partial order defined by the subset relation. But this is not really so, as the type theoretic interpretations can be viewed in terms of sets with extra structure attached, see figure 7.1.

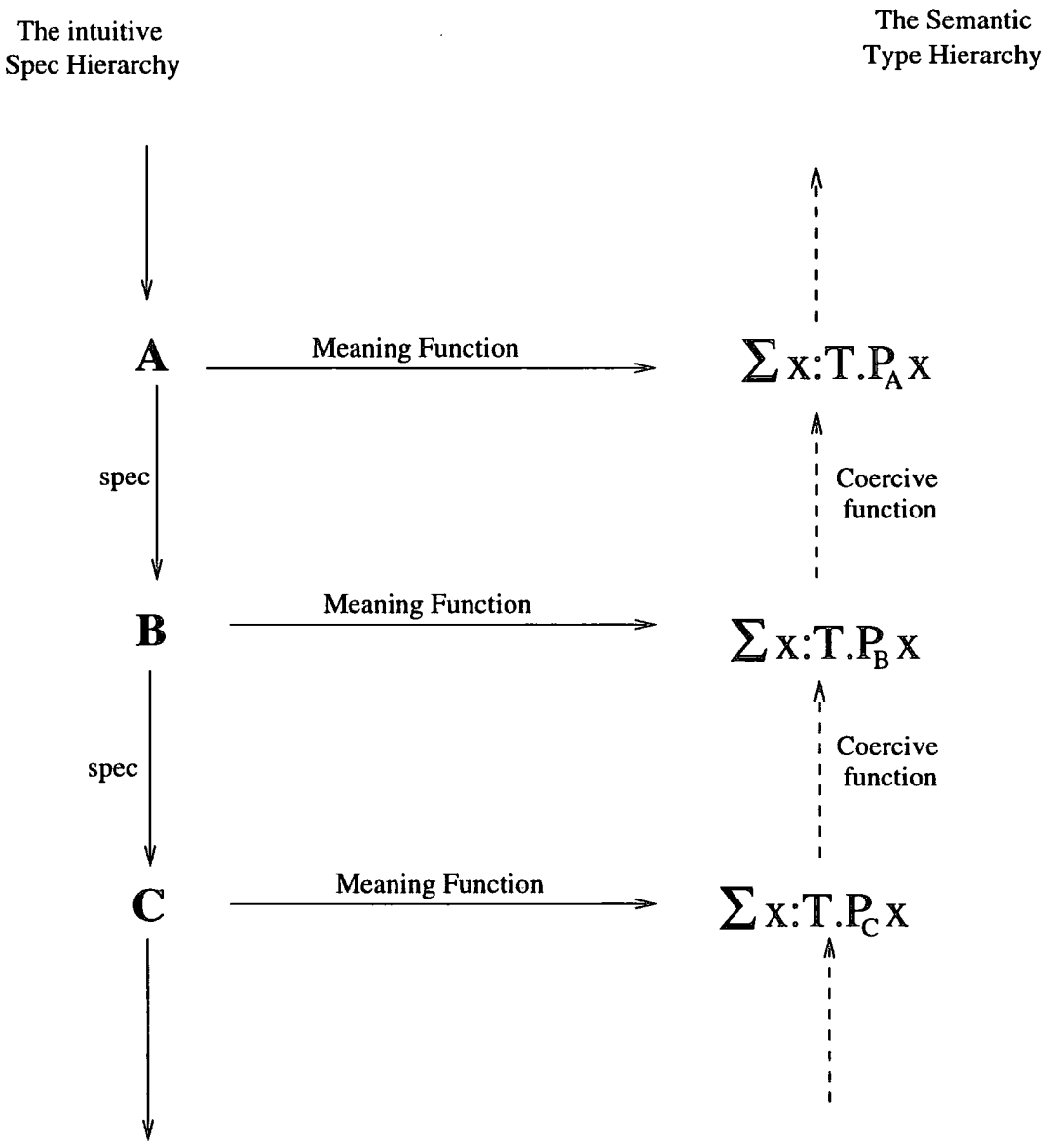



Figure 7.1: The 'semantic' type hierarchy.

Each type corresponds to a 'subset' of the type T, together with a proof of some property about each of the objects in the subset. Thus the interpretation preserves the intuition of the subset relation and carries further structure which is exploited by other aspects of the model.

## 7.2 Defined and Observed Events

As with the set theoretic model (see section 5.2) the observed events do not affect the definition (i.e. meaning) of refered subjects and objects. Thus as was concluded with the set theoretic model if observed events are introduced to SemNet then the *meaning as location* principle only applies as far as defining events. A fuller discussion is given in section 8.1.1.

Intuitively the CU nodes of SemNet represent noun phrases in natural language. The 'defined' events define such nodes and the 'observed' events refer back to them. In section 3.6 it was shown how dependent types are able to model this 'progression' which manifests itself many times in natural language. In section 6.2 CU nodes were interpreted as $\Sigma$ types of the form

$$\Sigma[x : T, p_1 : P_1(x), ..., p_n : P_n(x)]$$

with $P_i : T \rightarrow Prop$[1].

With this interpretation, each of the predicates can be referred to by 'future' UTT statements in the same way as was done in section 3.6.

However, Ranta uses Martin-Löf's type theory, which uses the $\Sigma$ type directly as a 'constructive' existential quantifier. This was used to interpret the 'donkey sentence', see equation 3.1. It turns out that to model 'directly' the way in which SemNet represents this sentence the same quantifier is needed. Therefore a slight change in the interpretation is needed.

For a universal-existential event the defining predicate needs to allow the existential 'witness' to be extracted. To allow this instead of interpreting as in equation 6.2, it is interpreted as:

[1]The predicates were actually packaged together as a conjunction, but this does not affect our present purpose.

$$Pred_2(Eue\ A\ B\ R\ bb) = \lambda x : T.(\Sigma y : B_m.R_m(x, y))$$

This means that the type changes, $(Pred_2 : Event \rightarrow (T \rightarrow Type))$. But the proposition given in equation 6.2 can be extracted so that the soundness results proved in chapter 6 still hold.

From now on CU nodes will be interpreted as:

$$\Sigma[x : T, q_1 : Q_1(x), ..., q_n : Q_n(x)]$$

where $Q_i : T \rightarrow Type$. Moreover the initial segment can be packaged as a single type, so that the node for, say, a mammal can be written as:

$$\Sigma x : Animal_m.Mammal_{pred}(x)$$

### 7.2.1 Basic observing events

In section 5.2 an attempt to model the SemNet representation of the 'donkey sentence' was given. In type theory the analysis follows the same lines, the 'predicate' for an event is only used if the event is defining for the entity concept. For example, the nodes of figure 5.3 are interpreted as:

$$E1 \mapsto \begin{cases} \text{The statement:} & \Pi x : M.\exists y : B.BO(x, y) \\ & \text{Every mother has a brother.} \\ \text{defining B1 as:} & \Sigma x : B.\Sigma y : M.BO(y, x) \\ & \text{Brothers of a mother.} \end{cases}$$

$$E2 \mapsto \left\{ \begin{array}{ll} \text{The statement:} & \Pi x : B1.\exists y : P.O(x,y) \\ & \text{Every 'brother of a mother' owns a parrot.} \\ \text{defining P1 as:} & \Sigma x : P.\Sigma y : B1.O(y,x) \\ & \text{Parrots owned by a 'brother of a mother'.} \end{array} \right.$$

### 7.2.2 Complex observing events

#### Interpreting the donkey sentence graph

In section 5.2 it was shown that to replicate the way in which SemNet builds the meaning of the 'donkey sentence' a witness extracting function is needed. Since UTT (via the Strong sum type, and its associated projection functions) has such a function the SemNet graph can be handled directly:

$$E1 \mapsto \left\{ \begin{array}{ll} \text{The statement:} & \Pi x : F1.\exists y : D.O(x,y) \\ & \text{Every F1 owns a donkey.} \\ \text{defining F1 as:} & \Sigma x : F.\Sigma y : D.O(x,y) \\ & \text{Farmers that own a donkey.} \\ \text{defining D1 as:} & \Sigma x : D.\Sigma x : F.O(x,y) \\ & \text{Donkeys owned by a farmer.} \end{array} \right\} \quad (7.1)$$

As before, intuitively E2 should be interpreted as "F1's beat the donkey that they own". This time, because of the constructive nature of type theory, the witness of the existential can be extracted:

$$E2 \mapsto \Pi x : F1.B(x, \pi 1(\pi 2(x))) \quad (7.2)$$

### Complex Actions

Although this may seem an esoteric example, which does not affect the general working of LOLITA, there is an immediate application. A current design problem is how to capture 'structural' facts about actions. For example, the action 'is ancestor of' is transitive, and the action 'is cousin of' is symmetric. Such rules are easy to specify in $\mathcal{FOPC}$.

A relation R is transitive if and only if:

$$\forall x \forall y \forall z((R(x,y) \wedge R(y,z)) \Rightarrow R(x,z)) \tag{7.3}$$

A relation R is symmetric if and only if:

$$\forall x \forall y(R(x,y) \Rightarrow R(y,x)) \tag{7.4}$$

It is not clear how such statements could be captured in SemNet. One problem is that the statements appear to be untyped, secondly it is not clear what the definition of the subject and objects of any events should be.

Starting from the formulation of the donkey sentence in $\mathcal{FOPC}$ and considering its counterpart in UTT:

$$\forall x \forall y(F(x) \wedge D(y) \wedge O(x,y)) \Rightarrow B(x,y)$$

$$\mapsto$$

$$\Pi x : F.\Pi y : D.O(x,y) \rightarrow B(x,y)$$

it is observed that this is isomorphic to:

$$\Pi z : (\Sigma x : F.\Sigma y : D.O(x,y))B(\pi 1(z), \pi 1(\pi 2(z)))$$

(equivalent to the formulation given by Ranta, see section 3.6) from which types equivalent to F1 and D1 can be extracted. Following a similar pattern for equation 7.4 gives, R is symmetric if and only if:

$$\Pi f : (\Sigma x : T.\Sigma y : T.R(x,y))R(\pi 1(\pi 2(f)), \pi 1(f))$$

From which subject and object concepts can be extracted as:

$$X = \Sigma x : T.\Sigma y : T.R(x,y)$$
$$Y = \Sigma y : T.\Sigma x : T.R(x,y)$$

and the statement becomes[2]:

$$\Pi x : X.R(\pi 1(x), \pi 1(\pi 2(x))) \tag{7.5}$$

which can be converted to SemNet, as in figure 7.2

Similarly the transitive statement gets converted to:

$$\Pi f : (\Sigma x : T.\Sigma y : T.\Sigma z : T.(R(x,y) \wedge R(y,z)))R(\pi 1(f), \pi 2(\pi 1(f))) \tag{7.6}$$

from which the following relevant types can be extracted:

$$X = \Sigma x : T.\Sigma y : T.\Sigma z : T.(R(x,y) \wedge R(y,z))$$
$$Y = \Sigma y : T.\Sigma x : T.\Sigma z : T.(R(x,y) \wedge R(y,z))$$
$$Z = \Sigma z : T.\Sigma y : T.\Sigma x : T.(R(x,y) \wedge R(y,z))$$

A problem here is that each of these types is defined by a conjunction of rela-

[2]The implicit subtyping assumed in chapter 6, avoided the use of the projection functions

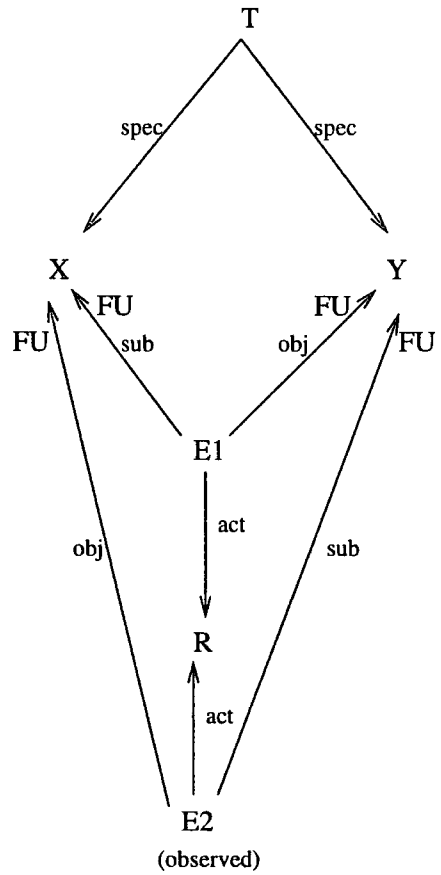



Figure 7.2: Proposed SemNet structure for Symmetric action

tions, leading to the graph in figure 7.3. Here E3 is the defining event for each of the entity concepts X, Y and Z, but E3 is not directly connected to any of them. Therefore if SemNet is to represent transitive relations (actions) in this way some extension to the representation is needed.

## 7.3 Necessary statements

Section 5.3 discussed how possible world semantics could be used to give a semantic account for 'necessarily true' statements. Here it is shown how the semantics defined so far can account for necessity. Consider again the nodes from figure 4.6.

$$X_m = \Sigma x : T.CS(x) \wedge PF(x)$$

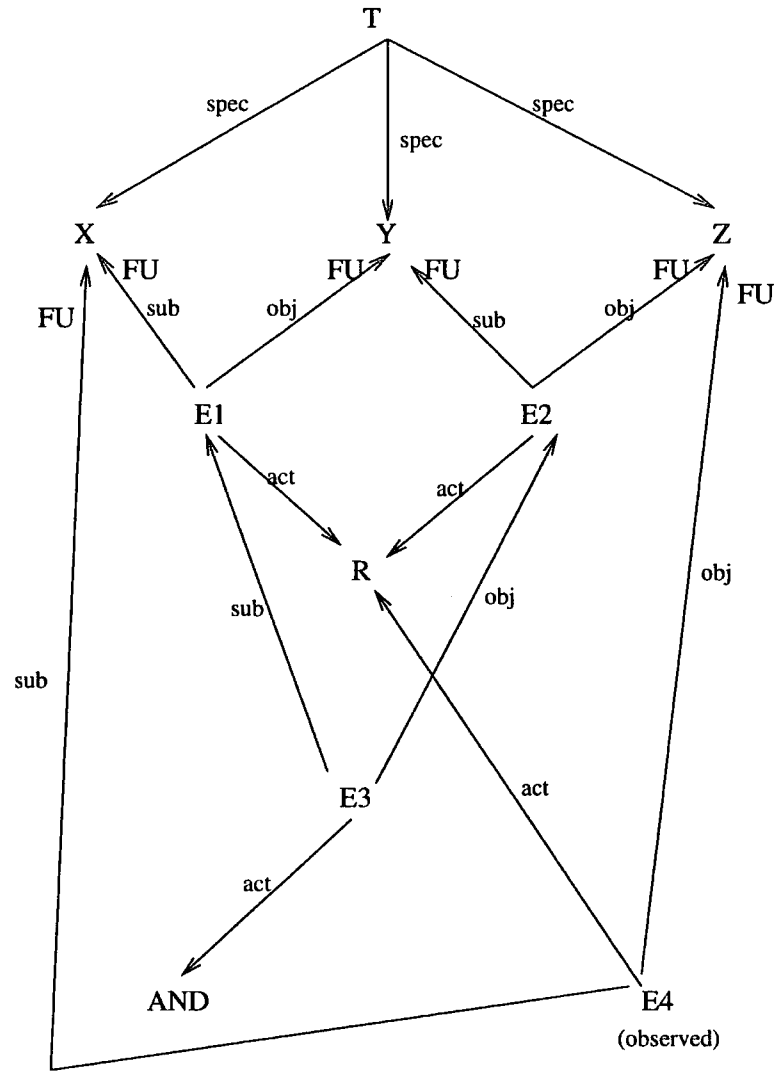



Figure 7.3: Proposed SemNet structure for Transitive action

where CS is a predicate for 'being a computer scientist at Durham', and PF is a predicate for 'plays football'. The events are interpreted as:

$$E_1 = \Pi x : X_m . PF(x)$$
$$E_2 = \Pi x : X_m . AI(x)$$

where AI is a predicate for 'studies AI'. The difference between these statements is that $E_1$ is a tautology and so is a necessary truth.

## 7.4 Intensionality of Propositions

As discussed in chapter 5, the intuitive interpretation for the epistemic actions is as relations over the domains of the subject and object. The problem there was that, propositions are only interpreted extensionally, i.e. as members of the set {true,false}. This time however, the initial semantics have mapped events to objects of type Prop in UTT. Therefore there is an intensional distinction between objects. For example:

$$\begin{aligned}(R_{epi}, Roberto, E1) &\mapsto Believes(Roberto_m, E1_m)\\ (R_{epi}, Roberto, E2) &\mapsto Believes(Roberto_m, E2_m)\end{aligned} \tag{7.7}$$

(with $Believes : (T \times Prop) \rightarrow Prop$). This seems reasonable since $E1_m$ and $E2_m$ are distinct types (propositions).

In this section rather than leaving the relation implicit (as was done with ordinary actions) epistemic relations are defined explicitly as the inductive relation[3]:

Define a relation relation B,

$$\begin{aligned}B &: (T \times Prop) \rightarrow Prop\\ b_{spec} &: \Pi a : CI.\Pi e : wff[s_1].IR(EEspec(a, e, true)) \rightarrow B(a_m, e_m)\\ b_{inst} &: \Pi a : CI.\Pi e : wff[s_2].IR(EEinst(a, e, true)) \rightarrow B(a_m, e_m)\\ b_{basic} &: \Pi a : CI.\Pi e : CE.IR(EEbasic(a, e, true)) \rightarrow B(a_m, e_m)\end{aligned}$$

EEvents are interpreted by a recursive function over this structure.

[3]With $a_m$ and $e_m$ being the semantic counterparts of $a$ and $e$.

$$
\begin{array}{lll}
\mathcal{M}_{eevent} & EEspec(a,e,true) & = B(a_m,e_m) \\
& EEinst(a,e,true) & = B(a_m,e_m) \\
& EEbasic(a,e,true) & = B(a_m,e_m) \\
& EEspec(a,e,false) & = \neg B(a_m,e_m) \\
& EEinst(a,e,false) & = \neg B(a_m,e_m) \\
& EEbasic(a,e,false) & = \neg B(a_m,e_m) \\
& EErec(a,e,true) & = B(a_m,\mathcal{M}_{eevent}(e)) \\
& EErec(a,e,false) & = \neg B(a_m,\mathcal{M}_{eevent}(e))
\end{array}
$$

The 'complexity measure function' $w : EEvent \rightarrow nat$ is easily defined on the structure of EEvent.

$$
\begin{array}{lll}
w & EEspec & = zero \\
w & EEinst & = zero \\
w & EEbasic & = zero \\
w & EErec(a,e,bb) & suc(w(e))
\end{array}
$$

**Intuitive Example**

For example, consider the events:

| | | |
|---|---|---|
| E1 | (Likes,Man,U,Dog,U) | All men like all dogs. |
| E2 | (Roberto $\in$ Man) | Roberto is a man. |
| E3 | (Likes,true,Roberto,I,Dog,U) | Roberto likes all dogs. |
| E4 | ($R_{epi}$,Rick,E1) | Rick believes 'all men like all dogs'. |
| E5 | ($R_{epi}$,Rick,E2) | Rick believes 'Roberto is a man'. |

with interpretations:

$$
\begin{aligned}
E1_m &= \Pi x : Man_m.\Pi y : Dog_m.Likes_m(x,y) \\
E2_m &= Man_{pred}(Roberto_m) \\
E3_m &= \Pi x : Dog_m.Likes(Roberto_m,y) \\
E4_m &= B(Rick_m, E1_m) \\
E5_m &= B(Rick_m, E2_m)
\end{aligned}
$$

Operationally the rules of $IR[s_4]$ entail the event:

E6 ($R_{epi}$,Rick,E3) Rick believes 'Roberto likes all dogs'.

and so by definition semantically the proposition:

$$E6_m = B(Rick_m, E3_m) \tag{7.8}$$

is entailed.

This may seem like the problem of semantics has been avoided, and that 'truth' has been directly equated with proof. To a certain extent this is the case. However some advantages have been accrued. In fact the interpretation reflects the semantics of epistemic events directly, since the true (operational) semantics of epistemic events is defined by the inference rules. By 'shifting' these rules into the world of 'Prop' in UTT, the 'meaning' can be analysed in terms of formal logical propositions of UTT. This is done in two phases, first a description of the current rules is given, and then an analysis of richer epistemic statements is given.

### 7.4.1 Definitional Events are Distinguished

It has already been established that the interpretation distinguishes interpretations for different events, see equation 7.7. In fact the distinction is finer grained than might be expected as even the definitional events (which are all tautologies) are distinguished.

For example, consider the interpretations for E1 in equation 7.1 and for E2 in the following:

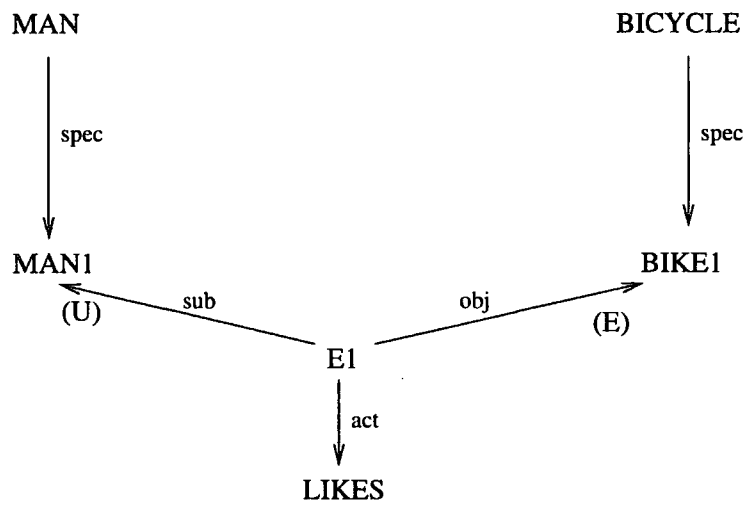



Figure 7.4: Definitional Event, 'Men that like bikes'.

$$E2 \mapsto \left\{ \begin{array}{ll} \text{The statement:} & \Pi x : M1_m.\exists y : B_m.L_m(x,y) \\ & \text{Every M1 likes a bike.} \\ \text{defining } \mathrm{M1}_m \text{ as:} & \Sigma x : M_m.\Sigma y : B_m.L_m(x,y) \\ & \text{Men that like a bike.} \\ \text{defining } \mathrm{B1}_m \text{ as:} & \Sigma x : B.\Sigma x : M_m.L_m(x,y) \\ & \text{Bikes liked by a man.} \end{array} \right\}$$

The statement E2 is proved by an object such as:

$$\lambda x : M1_m.L_m(x, \pi 1(\pi 2(x)))$$

This is a different object from a proof of E1, thus showing that E1 and E2 are different types.

### 7.4.2 Semantic analysis of Epistemic Rules

The intuition behind the basic epistemic rules, is that they assume all agents are able to use all the rules of inference known to LOLITA. Since these rules have been shown to be sound with respect to type theory, the assumption seems reasonable.

The more general rules involving internal relationships of B are open to analysis. For example, rule 4.12 (corresponding to $ir12$) is interpreted as:

$$\Pi P : Prop.\Pi a : T.B(a, P) \rightarrow B(a, B(a, P)) \tag{7.9}$$

This rule is harder to state in possible worlds semantics (see 5.3.2), but, because of the impredicativity of Prop, it can be made directly in UTT.

Finally rule 4.13 (corresponding to $ir13$) is interpreted as:

$$\Pi P : Prop.\Pi a : T.B(a, \neg P) \rightarrow \neg B(a, P) \tag{7.10}$$

This rule translated as something that is true (provable) in PWS. It is interesting that this proposition is not provable in UTT (from the rules given). This means that there is the flexibility in this model of allowing inconsistent agents, i.e. for some agent a, and some event e LOLITA could believe:

$$B(a_m, e_m) \wedge B(a_m, \neg e_m)$$

without her own beliefs being inconsistent. Its not clear if this is directly applicable, but it is certainly true that many agents express contradictory utterances and beliefs, and so in the longer term it may be useful to give a semantic account

of this behaviour.

### 7.4.3 Design/Prescription for Epistemic events

The current implementation of SemNet treats all epistemic actions in the same way. Intuitively there are some relationships which can be made between the actions. For example, if you know a fact, then you presumably believe that fact, although the converse may not hold, i.e.

$$\Pi a : T.\Pi P : Prop.Knows(a, P) \rightarrow Believes(a, P)$$

this corresponds to an epistemic action hierarchy shown in figure 7.5.

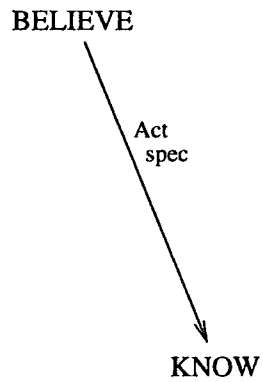



Figure 7.5: Epistemic Action Hierarchy.

Other expressions which are easily stated in the semantics include:

Simon believes everything that Donna believes.

which is stated as:

$$\Pi P : Prop.B(Donna, P) \rightarrow B(Simon, P)$$

This time the counterpart in SemNet is not so obvious as there is no mechanism

for treating events as parameters. However, the semantics are in place for such a structure.

### 7.4.4 Impredicativity and Paradox

The danger/difficulty of treating events as parameters, especially when impredicativity has been allowed into the language, are paradoxes. It is perhaps worth noting that (seemingly) paradoxical statements such as:

"This statement is not true."

which are discussed extensively in [Barwise and Etchemendy, 1987] cannot be stated in SemNet. This is because the recursive epistemic events (i.e. statements of belief about belief) are all grounded by basic epistemic events.

## 7.5 Distributedness.

Distributedness of SemNet (with respect to the set theoretic model) was established in section 5.4. In this section rather than giving a full analysis (which would mostly repeat points already made) an interpretation of the 'syntactic arcs' of sections 6.4 and 6.5 are given. It is clear that the interpretation of each arc is 'sound' with respect to the full event and so to the full net.

It is assumed (to save space) that quantifications are stored on the events rather than on the arcs and for the basic events the values are only given for Eue events. The result for basic events is similar to those for set theoretic semantics, however, because of the similarity in structure between epistemic events and their semantics the results for epistemic events are much clearer.

$$
\begin{aligned}
\mathcal{M}_{link} &: Link \rightarrow Prop \\
\mathcal{M}_{link}\ (Subject, Eue, A) &= \exists R' : (T^2 \rightarrow Prop).\Pi x : A_m. \\
&\quad \exists y : T.R'(x, y) \\
\mathcal{M}_{link}\ (Object, Eue, B) &= \exists R' : (T^2 \rightarrow Prop).\exists X : Type.\Pi x : X. \\
&\quad \exists y : B_m.R'(x, y) \\
\mathcal{M}_{link}\ (Action, Eue, R) &= \exists X, Y : T.\Pi x : X.\exists y : Y. \\
&\quad R(x, y)
\end{aligned}
$$

with X and Y subtypes of T. For epistemic links:

$$
\begin{aligned}
\mathcal{M}_{eplink} &: EpLink \rightarrow Prop \\
\mathcal{M}_{eplink}\ (EpSubject(E, a)) &= \exists P : Prop.B(a_m, P) \\
\mathcal{M}_{eplink}\ (EpObject(E, e)) &= \exists a : T.B(a, e_m) \\
\mathcal{M}_{eplink}\ (EpAbject(E, R_{epi})) &= \exists P : Prop.\exists a : T.B(a, P)
\end{aligned}
$$

This shows that SemNet is distributed with respect to the type theoretic model.

### 7.5.1 What makes SemNet distributed

As discussed in chapter 1 semantic nets have been described as notational variants of set theory or classical logic [Schubert, 1991]. It is usually accepted that they are organised so that commonly used inferences can be performed efficiently. In section 4.8 it was claimed that SemNet went further, in being designed to be distributed it is organised so as to be flexible and robust for NLE in ways that classical logic cannot be. Having defined and formally analysed distributedness, this section looks for aspects of the representation that make it distributed.

UTT (like classical logic) is only distributed as far as separate statements. The method will be to look at UTT interpretations of SemNet structures and to consider whether statements of these forms are somehow more distributed than other UTT

statements.

Consider the events of this chapter and their type theoretic interpretations, i.e. equations 7.1, 7.2, 7.5 and 7.8:

$$\Pi x : F1_m.\exists y : D1_m.O_m(x,y)$$
$$\Pi x : F1_m.Beats(x, f(x)) \qquad (f : F1_m \rightarrow D1_m)$$
$$\Pi x : (\Sigma x : T.\Sigma y : T.R(x,y)).R(\pi 1(x), (pi1(pi2(x))) \quad (f,g : X \rightarrow T)$$
$$B(a_m, E_m) \qquad (E_m : Prop)$$

and considering the structures of interpretations for basic events, from equation 6.1:

$$\Pi x : A_m.\Pi y : B_m.R_m\ x\ y$$
$$\Pi x : A_m.\exists y : B_m.R_m\ x\ y$$
$$\Pi x : B_m.\exists y : A_m.R_m\ y\ x$$
$$\Pi y : B_m.R_m\ a_m\ y$$
$$\Pi x : A_m.R_m\ x\ a_m$$
$$R_m\ a_m\ b_m$$

**Pre-Event forms**

The general pattern seems to mirror the intuition that SemNet builds complex concepts and allows 'new' events to refer to these concepts. events. In each case there is a 'quantification structure' (from now on the prefix) where types are defined, and a 'statement structure' (from now on the matrix) which refers to objects built by the prefix. Each of the above statements are convertible to a type in pre-event form:

$$\Pi x : (Structure).R(f(x), g(x)) \tag{7.11}$$

The question is whether statements in such a form are more distributed than general UTT statements.

The 'matrix' of 7.11 can be read independently from the prefix. It states that two things are in the relationship R.

For example, the matrix of equation 7.1 states that some x 'Owns' some y but the types and quantifications of x and y are unknown without the prefix.

The types and quantifications of the statement are implicit in the prefix. In SemNet the corresponding event specifies the types and quantifications locally. To mimic this, similar tagging mechanisms would be needed, e.g. add type and quantification tags to the referenced variable:

$$\{prefix\}R(f(x)_{quant,type}, g(x)_{quant,type})$$

for the example from equation 7.1

$$\Pi x : F1_m.\exists y : D_m.B_m(x_{U,F1}, y_{E,D})$$

This extra tagging is not needed for distributedness, as the matrix can be read independently anyway. However, there is value in adding the tags as this makes the interpretation more useful.

The 'prefix' consists of type judgements and quantifications. It defines types in terms of other types and event structures. It can be read independently from the matrix, but it makes no statement.

For example, the prefix of equation 7.5, defines the structure required to make the statement. $\pi 1(x)$ defines those objects that are the subject of a relation R, and $\pi 1(\pi 2(x))$ defines those objects which are the object of a relation R

### Coverage of Pre-Event Forms

Pre-event forms may seem restricted to statements involving binary relations, and these relations being restricted to types 'below' T in the subtype hierarchy. However, it does not seem difficult to widen the definition of an event to allow for as many (labelled) arcs as are required, thus being interpreted as n-ary relations (for any n). Also in theory the types need not be drawn from the entity hierarchy, indeed the epistemic relations have already shown that they can be proposition types as well.

The main aspect missing from general UTT statements are the logical connectives. But then these will clearly interpret the logical connective events of SemNet.

It is not claimed here that all statements in pre-event form have a counterpart in SemNet. Indeed the statement for transitivity is of this form and yet because of the problem of defining concepts via a 'conjunction' event, it is not clear how the current SemNet should represent it.

## 7.6 Review

This chapter has focussed on how features of UTT have been exploited to model and understand SemNet semantically. This has been shown in four distinct areas:

1. The sigma type has been used to allow a mimicking of the strong existential quantification which seems to be assumed in the representation of the 'donkey sentence'.

2. The intensionality of types has been exploited to show how intensional aspects of entity concepts can be distinguished.

3. The intensionality of propositions has been used to model the epistemic reasoning of SemNet.

4. Sigma types have been used to model how SemNet builds up and re-uses complex concepts.

# Chapter 8

# Evaluation, Conclusion and Further work

This chapter begins with an evaluation of the project based on the original objectives. A conclusion section is given and finally some suggestions for further work.

## 8.1 Evaluation

This evaluation section is structured to fit the methodological success and project specific criteria described in chapter 1.

### 8.1.1 The semantic model

A type theoretic semantic model has been built. Each of the basic constructs of SemNet have an interpretation in UTT. The success of the project rests on how closely the model fits the subjective 'intuitions' of the meaning of SemNet constructs. As was pointed out in chapter 1, this is almost as difficult as trying to establish an agreed semantics for natural language, nevertheless a crude analysis is

attempted.

### Basic events

The interpretation of the basic events seems entirely reasonable as an interpretation for their natural language counterparts. The examples below show each of the event types, a natural language statement that would be represented by such an event, and the type theoretic semantics for the event. Each type theoretic statement seems a reasonable interpretation.

| | | |
|---|---|---|
| Euu | All children like all toys. | $\Pi x : C.\Pi y : T.L(x,y)$ |
| Eue | Every child likes a toy. | $\Pi x : C.\exists y : T.L(x,y)$ |
| Eui | There is a toy that every child likes. | $\exists y : T.\Pi x : C.L(x,y)$ |
| Eii | There is a child that likes all toys. | $\exists x : C.\Pi y : T.L(x,y)$ |

Where C, T, L are the obvious counterparts/types for children, toys and liking.

As outlined in 7.1 there are arguments for suggesting that these interpretations capture the English statements more intuitively than their set theoretic counterparts since the quantifications are over the concepts involved, rather than over a universe.

### The hierarchy

There can be little argument that the developers of LOLITA consider the entity hierarchy as a subset and membership hierarchy. However, the subtype hierarchy can be viewed as a subset hierarchy (over the type T of chapter 6, as opposed to the set theoretic domain $\mathcal{D}$ of chapter 5) except that the 'sets' lower down the hierarchy are paired with properties which they hold. In this sense it can be seen that the underlying semantics are the same, it is just that more structure is added

to the concepts lower down the hierarchy. With this proviso the subtype hierarchy meets the developers' intuitions.

### Meaning as location.

A major principle of SemNet is that no nodes have a pre-defined meaning and the meaning of a node depends on how it is related to the other nodes, i.e. on its location. Moreover, the full meaning of a node can only be determined by interpreting the whole network.

In type theory an entity node A, is interpreted as a type:

$$A_m = \Sigma x : T.P(x)$$

where P is a predicate over T. This is a partial interpretation, as P is implicit, to interpret more (find out more about P) the defining event must be interpreted (as $E_{pred}$), giving:

$$\Pi x : T.(E_{pred}\ x \wedge P_1\ x)$$

where $P_1$ is an implicit predicate over T, which is defined by the nodes 'above' A in the hierarchy. Again this is only partial and to interpret more involves interpreting the defining events for the generalisations of the original node. A full definition is reached once the top entity node is reached. A similar analysis could be performed for events, which require their subject and object nodes to be interpreted fully before they are fully defined.

The idea of meaning as location is certainly captured by the semantics. A local interpretation is possible, and this can be built upon by reading more nodes. However, not all the nodes are required to reach a full interpretation. Observed events (if introduced) and nodes 'below' an entity in the hierarchy do not seem to

be required. For example, the interpretation of the node for MAMMAL is not used at all to define the type for ANIMAL. This seems reasonable, unless it is insisted that a part of the meaning of ANIMAL is that some of them are MAMMALs.

### Complex Concepts

Intuitively SemNet builds complex statements by building up complex concepts (nodes) through defining events and allowing new events to refer to them. The type theoretic semantics models this with all interpretations being in pre-event form (see equation 7.11).

Furthermore, in representing the 'donkey sentence' SemNet assumes that the witness of a previous existential can be extracted. Constructive type theory models this directly, and as discussed in 7.2, this leads to a statement that is isomorphic to Ranta's interpretation of the 'donkey sentence'.

### Belief as a relation

Intuitively actions are relations over the subject and object of the event. This is also true for epistemic events. This is modelled directly in type theory by the inductively defined relation $B : (T \times Prop) \rightarrow Prop$.

Moreover, the rules of inference are reflected directly in $B$, since $B$ is defined by them. Therefore, in this case even if $B$ does not reflect intuitions, it does reflect directly the meaning of epistemic events.

### Summary

The above analysis shows, intuitively, that the model reflects SemNet. However, simply mapping to a type theoretic model does not ensure that SemNet is well founded in any sense. This can only be established by analysing the model in terms of the logic of type theory.

Of course it is also intuitive that SemNet meets many of the properties described and so showing this in the semantics model adds further verification that the model captures the intuitive SemNet.

### 8.1.2 Correctness of Reasoning

The 'valid' inference rules have been interpreted into type theory and shown to be sound.

There is the added element that the proofs have been machine assisted checked. Furthermore, an abstraction of the algorithm for implementing inheritance has been shown to implement rules 4.1 and 4.2 soundly and completely.

### 8.1.3 Expressiveness

The three aspects of SemNet related to expressiveness that were analysed were rich quantification, epistemic knowledge and intensionality. These have all been analysed as follows.

The analysis of the 'donkey sentence' representation (see section 7.2) shows that SemNet can express the quantification needed here. In particular the whole sentence is represented, but also all the substructures involved represent statements that are entailed by the full sentence.

The analysis of epistemic actions showed that the rules used are reasonable (when considered in the impredicative world of *Prop*).

The analysis of entities showed that although the inheritance works extensionally, the structure does contain the information to make intensional distinctions.

### 8.1.4 Flexibility

In chapter 4 distributedness was put forward as a reasonable measure of how flexible a network is. To show SemNet is distributed required a formal semantic model, and this was completed for both models developed.

Essentially distributedness shows that information can be gleaned at various levels of granularity. The point is that different algorithms have the flexibility to choose the 'depth' of information that they require right down to the level of individual nodes and arcs.

### 8.1.5 Developer Comprehend-ability

The project has contributed to developer comprehend-ability. As discussed from the beginning it should be emphasised that the exposition given of SemNet in chapter 4 was not a starting point for the thesis. The presentation given and issues raised were mainly the result of the semantic analysis presented in later chapters.

For a newcomer to the project chapter 4 serves as the best way to understand, intuitively how SemNet operates. When a developer is writing algorithms that operate on SemNet they should now use the formal semantics developed in chapter 6, as the intended meaning of the constructs.

For example, in semantic analysis when deciding whether a concept exists already, the verification should be considered in relation to the semantics of the nodes concerned.

### 8.1.6 Issues for SemNet.

The main issues raised by the work were outlined in section 4.5. The semantic model may contribute further, by providing a semantic basis for decisions. Of course all decisions will have to be weighed up against engineering principles such as cost-benefit, resource constraints and so on.

- Semantic analysis (see section 7.2) has shown that it is equally meaningful to place the quantification tags on the links, entities or events of SemNet, but that each effects interpretation efficiency.

- The mapping of epistemic actions to a relation over *Prop* and T was so successful that it is easy to consider extensions to the epistemic reasoning module in the semantics (see section 7.4).

### 8.1.7 Contribution to NLE.

The main contribution to NLE has been to give a formal presentation for SemNet, which is an extremely powerful KR language. This meets the criticism discussed in chapter 2, that semantic networks are often presented informally with unclear semantics.

Secondary to this is that type theory has many features that are extremely useful for formalisation work in NLE. In this section an evaluation of this claim is given.

#### Semantic Similarities

Many of the a priori reasons for using UTT as a semantic language for SemNet were because of similarities in structure between these two specific languages. Whether the work is relevant to other networks rests on how similar SemNet is to other networks.

The model of the hierarchy is of the most general interest. In both KL-ONE and CGT concepts lower down the hierarchy are defined by the 'parent' nodes and by the roles they play. In CGT these roles are defined by lambda abstractions. By defining the nodes as types this captures this definition precisely.

The notion of defined and observed events is actually very similar to the taxonomy and assertion distinctions of KL-ONE type systems. This suggests that

the section modelling building and re-use of concepts may be applicable to these networks.

As discussed in section 4.8.2, SNePS represents the 'donkey sentence' in a manner very similar to SemNet. By re-referring to the variable 'donkeys owned by a farmer' and claiming the graph represents the donkey sentence, then a witness extracting function is being assumed. Therefore the constructive interpretation provided by UTT is ideal for capturing this.

Also relevant to the SNePS representation is the interpretation of event nodes as objects of type *Prop* in UTT. This allowed for example, an interpretation for the recursive nesting of propositions.

The work of Ranta [Ranta, 1994] shows that many aspects of natural language can be treated naturally in constructive type theory. This work found similarities between results there and how SemNet operated. Since other networks have been designed with natural language understanding in mind these similarities should manifest themselves here as well.

### Manageability aspects

Performing a large analysis such as this is a major task. Type theory through its 'manageability' features provided many tools and methods which helped to break the task down.

The abstract theory mechanism helped to bring a modular approach to the problem. In this work it was fortunate that the hierarchy was entirely independent of the basic events, and that they in turn independent of the epistemic events. Otherwise the results achieved for $\text{SemNet}_1$ in section 6.2 would have to be re-analysed when basic events were considered. But in the end this just helped in the presentation (i.e. chapter 6, could present the model of SemNet in distinct phases). The modular approach would still have helped to establish some results for the hierarchy before seeing how 'basic events' effected their behaviour.

Machine assistance was occasionally useful. Lego did stop some analysis from going further and demanded closer inspections, of course this generally occured when some aspect had been overlooked.

## 8.2 Conclusions

As has been stated the major result of this work is the provision of a formal semantics for SemNet. This is significant as many networks are put forward without proper regard for a formal account of meaning. The work has shown, formally, that SemNet has many features which help overcome some of the problems of KR for NLE. In particular:

- SemNet is richly expressive, being able to represent intension, epistemic knowledge and complex quantification.
- SemNet's basic reasoning mechanisms are sound with respect to the internal logic of UTT.
- SemNet is distributed. This means that 'knowledge' can be retrieved in a flexible manner, as is required by the different modules of a large-scale natural language system.

The work has also shown that constructive type theory has many features which make it a useful tool for studying aspects of NLE. For example:

- It can express many aspects of natural language in a natural manner. Moreover, these are the same aspects that are often expressed in semantic networks (or more generally KR's for NLE).
- It has useful meta-theoretic properties which have lead to tools and techniques developed for formal methods. These tools are useful for NLE.

The work has also characterised aspects of SemNet which are of more general interest to the semantic network community, in particular formally interpreting nodes in hierarchies as 'intensional' types, a constructive witness extracting function, and a direct interpretation for representing nodes as propositions.

## 8.3 Further Work

### 8.3.1 Implementing a Maths Vernacular

A research proposal has been written [Luo *et al.*, 1996] and accepted to implement a mathematical vernacular. The idea is to integrate theory and technology in Computer-Assisted Formal Reasoning (CAFR) and Natural Language Processing.

More specifically the project will attempt to provide type theoretic semantics for a 'subset' of natural language (i.e. the mathematical vernacular). This semantics will then be the focal point for linking the work of Lego (which is based on type theory) and LOLITA (which now has a type theoretic semantics). The longer term aim is to provide natural language support and capability to CAFR technology.

Although this work will not feed this project directly, (as it provides a formalisation of SemNet itself, rather than the knowledge it represents), it was the original motivation, and it is expected that a great deal of the modelling work will be re-usable for the 'linking' work.

### 8.3.2 Further aspects of SemNet

The original aim for this work was to better understand how SemNet represents and reasons with knowledge. The foundation for this is in place, and the basic mechanisms have been interpreted and understood. Two modules which build on the basic model are 'the reasoning by analogy module' and 'reasoning about time and location module'. A possible further project could consider these modules and

what their operations 'mean' in the type theoretic model

### 8.3.3 The semantic model as a tool

Finally, it has been mentioned that the semantic model has raised some issues for SemNet, and also that it has a role to play in the future development of SemNet and LOLITA. To ensure this happens there is further work to be done, working out how the model could best be communicated, stored and updated so that it performs this role in the future.

# References


[Aczel, 1988] P. Aczel, *Non-Well-Founded Sets*, Number 14, CSLI lecture notes, 1988.

[Ali and Shapiro, 1993] S. S. Ali and S. C. Shapiro, "Natural Language Processing Using a Propositional Semantic Network with Structured Variables", *Minds and Machines*, 3, No 4, 1993.

[Anderson and Belnap, 1975] A. R. Anderson and N. D. Belnap, *Entailment: The logic of relevance and necessity.*, volume 1, Princeton University Press, 1975.

[Augustsson *et al.*, 1990] L. Augustsson, T. Coquand, and B. Nordstrom, "A short description of another logical framework.", in G. Huet and G. Plotkin, editors, *Preliminary Proceedings of Logical Frameworks*, 1990.

[Barendregt, 1992] H. P. Barendregt, "Lambda Calculi with Types", in *Handbook of Logic in Computer Science, Volume 2. Background: Computational Structures*, Clarendon Press, 1992.

[Barwise and Etchemendy, 1987] J. Barwise and J. Etchemendy, *The Liar: An Essay on Truth and Circularity*, Oxford University Press, 1987.

[Barwise and Perry, 1985] J. Barwise and J. Perry, *Situations and attitudes*, MIT press, Cambridge, 1985.

[Beierle *et al.*, 1992] C. Beierle, U. Hedstuck, U. Pletat, P. H. Schmitt, and J. Siekmann, "An order-sorted logic for knowledge representation systems", *Artificial Intelligence*, 55, 1992.

[Bird and Wadler, 1988] R. Bird and P. Wadler, *Introduction to Functional Programming*, International Series in Computer Science, Prentice Hall, 1988.

[Bishop, 1967] E. Bishop, *Foundations of Constructive Analysis*, McGraw-Hill, 1967.

[Bokma and Garigliano, 1992] A. Bokma and R. Garigliano, "Uncertainty Management through Source control: A heuristic approach", *Proceedings, International Conference on Information Processing and Management of Uncertainty in knowledge based systems, Mallorca, Spain.*, 1992.

[Brachman and Schmolze, 1985] R. J. Brachman and J. Schmolze, "An overview of the KL-ONE knowledge representation system", *Cognitive Science*, 9, 1985.

[Brachman *et al.*, 1991] R. J. Brachman, H. Levesque, and R. Reiter, "Introduction to the special volume on Knowledge Representation", *Artificial Intelligence*, 49, 1991.

[Cann, 1993] R. Cann, *Formal Semantics, An Introduction*, Cambridge University Press, 1993.

[Constable *et al.*, 1986] R. L. Constable, S. F. Allen, H. M. Bromley, et al., *Implementing Mathematics with the NuPRL Proof Development System*, Prentice Hall, 1986.

[Costantino *et al.*, 1996] M. Costantino, R. Collingham, and R. G. Morgan, "Financial Information Extraction at the University of Durham", *Proceedings, II International Meeting of Artificial Intelligence in Accounting, Huelva, Spain*, 1996.

[Curry and Feys, 1958] H. B. Curry and R. Feys, *Cominatory Logic, volume 1*, North Holland Publishing Company, 1958.

[Davis, 1989] R. E. Davis, *Truth, Deduction and Computation: Logic and Semantics for Computer Science*, Computer Science Press, 1989.

[de Bruijn, 1980] N. G. de Bruijn, "A survey of the project AUTOMATH.", in J. Hindley and J. Seldin, editors, *To H. B. Curry: Essays on Combinatory logic, Lambda Calculus and Formalism*, Academic Press, 1980.

[Dowek, 1990] G. Dowek, "The Coq Proof Assistant: User's Guide (version 5.6)", Technical report, INRIA-Rocquencourt and CNRS-ENS Lyon, 1990.

[Dowty *et al.*, 1981] D. R. Dowty, R. E. Wall, and S. Peters, *Introduction to Monatgue Semantics*, Texts and studies in Linguistics and Philosophy, D. Reidel Publishing Company, 1981.

[Dunn, 1986] J. M. Dunn, "Relevance logic and Entailment.", in D. M. Gabbay, editor, *Handbook of Philosophical Logic*, D. Reidel Publishing Company, 1986.

[Etherington, 1988] D. W. Etherington, *Reasoning With Incomplete Information*, Reasearch Notes In Artificial Intelligence, Pitman, 1988.

[Fahlman, 1979] S. E. Fahlman, *NETL: A system for representing and Using Real World Knowledge*, The MIT press, 1979.

[Fraenkel and Bar-Hillel, 1958] A. R. Fraenkel and Y. Bar-Hillel, *Foundations of Set Theory.*, North Holland Publishing Company, 1958.

[Frisch, 1991] A. Frisch, "The substitutional framework for sorted deduction: fundamental results in hybrid reasoning", *Artificial Intelligence*, 49, 1991.

[Froidevaux and Kayser, 1988] C. Froidevaux and D. Kayser, "Inheritance in Semantic Networks and Default Logic", in P.Smets, E. H. Mamdani, D. Dubois, and H. Prade, editors, *Non-Standard Logics for Automated Reasoning*, Academic Press, Harcourt Brace Jovanovich Publishers, 1988.

[Garigliano *et al.*, 1993a] R. Garigliano, R. Morgan, and M. Smith, "LOLITA as a content scanning tool", *Proceedings, 13th International conference on Artificial Intelligence, Expert Systems and Natural Language Processing, Avignon, France.*, 1993.

[Garigliano *et al.*, 1993b] R. Garigliano, R. Morgan, and M. Smith, "LOLITA progress report", Technical report, University of Durham, 1993.

[Garigliano *et al.*, 1995] R. Garigliano, R. Morgan, M. Smith, and S. P. Jones, "DEAR project summary", *Proceedings, 1st AIKMS conference, Oxford.*, 1995.

[Garigliano, 1995] R. Garigliano, "Editorial", *Natural Language Engineering*, 1, part 4, 1995.

[Girard, 1986] J. Y. Girard, "The system F of variable types, fifteen years later.", *Theoretical Computer Science*, (45), 1986.

[Goguen, 1994] H. Goguen, *A Typed Operational Semantics for Type Theory*, PhD thesis, University of Edinburgh, 1994.

[Graber *et al.*, 1995] A. Graber, H. J. Burckert, and A. Laux, "Terminological Reasoning with Knowledge and Belief", in *Knowledge and Belief in Philosophy and Artificial Intelligence*, Akadamie Verlag, 1995.

[Grishman, 1986] R. Grishman, *Computational Linguistics*, Cambridge University Press, 1986.

[Herzog and Rollinger, 1991] O. Herzog and C. R. Rollinger, editors, *Text understanding in LILOG (Integrating Computational Linguistics and Artificial Intelligence. Final report on the IBM LILOG-Project*, Springer-verlag, 1991.

[Hill, 1994] R. K. Hill, *Issues of Semantics in a Semantic-Network Representation of Belief*, PhD thesis, University of Buffalo, New York, 1994.

[Hindley and Seldin, 1986] J. R. Hindley and J. P. Seldin, *Introduction to Combinators and lambda calculus*, London Mathematical Society Student texts, 1, Cambridge University Press, 1986.

[Hoffman, 1992] M. Hoffman, "Formal development of functional programs in type theory - a case study.", Technical Report ECS-LFCS-92-228, LFCS: Edinburgh University, 1992.

[Holyer, 1991] I. Holyer, *Functional Programming with Miranda*, Pitman Publishing, 1991.

[Howard, 1980] W. A. Howard, "The formulae-as-types notion of construction", in J. Hindley and J. Seldin, editors, *To H. B. Curry: Essays on Combinatory Logic*, Academic Press, 1980.

[Hudak *et al.*, 1992] P. Hudak, S. P. Jones, P. Wadler, et al., "Report on the Programming language. A Non-strict, purely functional language, version 1.2", Technical report, Glasgow University, 1992.

[Jones and Garigliano, 1994] C. Jones and R. Garigliano, *Dialogue Structure Models: An engineering approach to the analysis and generation of natural english dialogues*, PhD thesis, University of Durham, 1994.

[Jones, 1996] A. Jones, "The addition of subtyping to the Unifying Theory of dependent Types: A literature survey", Technical report, University of Durham, 1996.

[Kumar and Chalupsky, 1993] D. Kumar and H. Chalupsky, "Guest Editorial for Special Issue on Propositional Knowledge Representation", *Journal of Experimental and Theoretical Artificial Intelligence*, 5, No 2 and 3, 1993.

[Lehmann, 1992] F. Lehmann, "Semantic Networks", in F. Lehmann, editor, *Semantic Networks in Artificial Intelligence (part 1)*, volume 2-5 of *Computers and mathematics with applications*, Pergamon Press, 1992.

[Leviant, 1994] D. Leviant, "Higher Order Logic", in D. M. Gabbay, C. J. Hogger, and J. A. Robinson, editors, *Handbook of logic in AI and Logic programming*, volume 2, Clarendon Press, 1994.

[Long and Garigliano, 1994] D. Long and R. Garigliano, *Reasoning By Analogy And Causality: A model and application*, Artificial Intelligence, Ellis Horwood, 1994.

[Luger and Stubblefield, 1993] G. F. Luger and W. A. Stubblefield, *Artificial Intelligence: structures and stragies for complex problem solving*, Benjamin/Cummings, 1993.

[Luo and Pollack, 1992] Z. Luo and R. Pollack, "Lego proof development system: Users Manual", Technical Report ECS-LFCS-92-211, LFCS, Edinburgh University, 1992.

[Luo *et al.*, 1996] Z. Luo, R. Garigliano, and S. Shiu, "Research Proposal: Computer Aided Reasoning with Natural Language: Implementing a Maths Vernacular", Technical report, University of Durham, 1996.

[Luo, 1990] Z. Luo, *An Extended Calculus of Constructions*, PhD thesis, University of Edinburgh, 1990.

[Luo, 1993] Z. Luo, "Program specification and data refinement in type theory.", *Mathematical Structures in Computer Science, 3*, 1993.

[Luo, 1994] Z. Luo, *Computation and Reasoning: A Type Theory for Computer Science*, Oxford Science Publications, 1994.

[Luo, 1996] Z. Luo, "Coercive subtyping in Type Theory.", *submitted Proceedings, Computer Science Logic '96, Utrecht.*, 1996.

[Martin-Lof, 1982] P. Martin-Lof, "Constructive Mathematics and Computer Programming", in L. J. Cohen, editor, *Logic, Methodology and Philosophy of Science*, North-Holland, 1982.

[Martin-Lof, 1984] P. Martin-Lof, *Intuitionistic Type Theory*, Bibliopolis, 1984.

[Meyer and der Hoek, 1995] J. J. C. Meyer and W. V. der Hoek, *Epistemic Logic in AI and Computer Science*, Cambridge tracts in Computer Science, Cambridge University Press, 1995.

[Miller, 1990] G. Miller, "Wordnet: An online lexical database.", *International Journal of Lexicography.*, 1990.

[Moore, 1995] R. Moore, *Logic and Representation*, CSLI Lecture Notes, 39, 1995.

[Morgan *et al.*, 1995] R. Morgan, R. Garigliano, et al., "Description of the LOLITA system as used in MUC-6", in *Proceedings Sixth Message Understanding Conference (MUC-6)*, 1995.

[Nagle *et al.*, 1992] T. E. Nagle, J. A. Nagle, L. L. Gerholz, and P. W. Eklund, editors, *Concpetual Structures: current research and practice*, Ellis Horwood, 1992.

[Nebel, 1991] B. Nebel, "Terminological Cycles: Semantics and Computational Properties", in J. F. Sowa, editor, *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, chapter 11, Morgan Kauffman, 1991.

[Nettleton, 1997a] D. J. Nettleton, editor, *The LOLITA book volume 1: System Core.*, To be published, Springer-Verlag, 1997.

[Nettleton, 1997b] D. J. Nettleton, editor, *The LOLITA book volume 2: Applications.*, To be published, Springer-Verlag, 1997.

[Nettleton, 1997c] D. J. Nettleton, editor, *The LOLITA book volume 3: Philosophy and Methodology.*, To be published, Springer-Verlag, 1997.

[Nordstrom *et al.*, 1990] B. Nordstrom, K. Petersson, and J. Smith, *Programming in Martin-Lofs Type Theory: An Introduction*, Oxford University Press, 1990.

[Pollack, 1989] R. Pollack, "The theory of LEGO", Technical report, University of Edinburgh, 1989.

[Poria and Garigliano, 1996] S. Poria and R. Garigliano, "An Exploration of Explanation.", *submitted to 9th European Conference on Machine Learning*, 1996.

[Ranta, 1994] A. Ranta, *Type-Theoretical Grammar*, Oxford Science Publications, 1994.

[Rapaport, 1981] W. Rapaport, "How to make the world fit our language: An essay in Meinongian semantics.", *Grazer Philosophiche Studien*, (14), 1981.

[Schubert, 1991] L. K. Schubert, "Semantic Nets Are in the Eye of the Beholder", in J. F. Sowa, editor, *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, chapter 2, Morgan Kauffman, 1991.

[Shapiro and Rapaport, 1987] S. C. Shapiro and W. J. Rapaport, "SNePS considered as a fully intensional propositional semantic network", in N. Cercone and G. McCalla, editors, *The Knowledge Frontier*, Springer-Verlag, 1987.

[Shapiro, 1979] S. C. Shapiro, "The SNePS Semantic Network Processing System", in N. Findler, editor, *Associative Networks: Representation and use of Knowledge by Computers*, Academic Press, 1979.

[Shapiro, 1991] S. C. Shapiro, "Cables, Paths and "Subconscious" Reasoning in Propositional Semantic Networks", in J. F. Sowa, editor, *Principles of Semantic Networks*, Morgan Kaufmann, 1991.

[Shastri, 1988] L. Shastri, *Semantic Networks: An Evidential formalization and its connectionist realization*, Research Notes In Artificial Intelligence, Pitman, 1988.

[Shiu *et al.*, 1996] S. Shiu, Z. Luo, and R. Garigliano, "Type theoretic semantics for SemNet", in D. M. Gabbay and H. J. Ohlbach, editors, *Practical Reasoning, International Conference on Formal and Applied Practical Reasoning, FAPR'96, Bonn, Germany*, Lecture Notes in Artificial Intelligence 1085, subseries of Lecture notes in Computer Science, Springer-Verlag, 1996.

[Short *et al.*, 1996] S. Short, S. Shiu, and R. Garigliano, "Distributedness and Non-Linearity of LOLITA's Semantic Network", in *COLING96*, 1996.

[Short, 1996] S. Short, *Knowledge Representation of Lolita*, PhD thesis, (to be submitted) Department of Computer Science, University of Durham, 1996.

[Smith, 1995] M. H. Smith, *Natural Language Generation in the LOLITA System: An Engineering Approach*, PhD thesis, Department of Computer Science, University of Durham, April 1995.

[Sowa, 1984] J. F. Sowa, *Conceptual Structures: Information Processing in Mind and Machine*, The Systems Programming Series, Addison Wesley, 1984.

[Touretzky, 1986] D. Touretzky, *Mathematics of Inheritance systems*, Research Notes In Artificial Intelligence, Pitman, 1986.

[Walther, 1987] C. Walther, *A Many-Sorted Calculus Based on Resolution and Paramodulation*, Reasearch Notes In Artificial Intelligence, Pitman, 1987.

[Wand, 1992] P. Wand, "Functional Programming and Verification with Lego: MSc. Project Report", Technical report, LFCS, University of Edinburgh, 1992.

[Wang, 1995] Y. Wang, *An intelligent computer based tutoring approach for the management of negative transfer*, PhD thesis, Department of Computer Science, University of Durham, 1995.

[Woods and Schmolze, 1992] W. A. Woods and J. G. Schmolze, "The KL-ONE Family", *Computers Mathematics and Applications*, 23, No 2 and 3:133–177, 1992.

[Woods, 1975] W. A. Woods, "Whats in a link: Foundations for semantic networks.", in D. G. Bobrow and A. M. Collins, editors, *Representation and Understanding*, Academic Press, 1975.

[Woods, 1991] W. A. Woods, "Understanding Subsumption and Taxonomy: A Framework for Progress", in J. F. Sowa, editor, *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, chapter 1, Morgan Kauffman, 1991.