A "NATURAL LOGIC" FOR NATURAL LANGUAGE PROCESSING AND KNOWLEDGE REPRESENTATION

by

Syed S. Ali

A dissertation
submitted to the Faculty of the Graduate School
of State University of New York at Buffalo
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

November 1993

Abstract

We define a knowledge representation and inference formalism that is well suited to natural language processing. In this formalism every subformula of a formula is closed. We motivate this by observing that any formal language with (potentially) open sentences is an inappropriate medium for the representation of natural language sentences. Open sentences in such languages are a consequence of the separation of variables from their quantifier and type constraints, typically in the antecedents of rules. This is inconsistent with the use of descriptions and noun phrases corresponding to variables in language. Variables in natural language are constructions that are typed and quantified as they are used. A consequence of this is that variables in natural language may be freely reused in dialog. This leads to the use of pronouns and discourse phenomena such as ellipsis involving reuse of entire subformulas. We present an augmentation to the representation of variables so that variables are not atomic terms. These "structured" variables are typed and quantified as they are defined and used. This leads to an extended, more "natural" logical language whose use and representations are consistent with the use of variables in natural language. Structured variables simplify the tasks associated with natural language processing and generation, by localizing noun phrase processing.

The formalism is defined in terms of a propositional semantic network, starting from nodes and arcs connecting nodes, subsumption, matching, to inference. It allows the resolution of some representational difficulties with certain classes of natural language sentences (e.g. the so-called "donkey" sentences and sentences involving branching quantifiers). Reuse phenomena, such as pronominalization and ellipsis, are captured in the representation by structure-sharing. A major advantage of this structured representation of variables is that it allows a form of terminological and derived subsumption similar to surface reasoning in natural language.

Acknowledgments

The work described in this dissertation owes a significant debt to the ideas and previous work of my advisor, Dr. Stuart C. Shapiro. Without his help and guidance this work would not have been possible.

Dr. William J. Rapaport, a member of my dissertation committee, who brought philosophical and cognitive perspectives to my research. His guidance made this a much better dissertation.

Dr. Bharat Jayaraman, a member of my dissertation committee, made many helpful suggestions regarding the formalization of my work.

Dr. Matthew Dryer, a member of my dissertation committee, provided me with many useful linguistic insights.

Dr. Robert M. MacGregor of the USC/Information Sciences Institute, who was my outside reader, for his comments on my dissertation.

My colleagues in the SNePS Research Group (SNeRG) who put up with numerous presentations of my work and made helpful suggestions.

Eloise Benzel and the other Computer Science Department Secretaries: Sally Elder, Leslie Russo, Jaynee Straw, and Gloria Koontz, for their friendliness, helpfulness and kindness. Their sunny disposition always cheered me up when my work was not going well.

Most importantly, my family: my wife Susan Haller for her constant support and guidance; my parents for instilling in me the the desire to learn.

Contents

List of Figures

Chapter 1

Introduction

1.1 Overview

The general task of work in knowledge representation and reasoning (KRR) is, trivially, the representation of knowledge. The kind of knowledge for which the representation task has been undertaken is, often, secondary to the description of the overall task as an exercise in knowledge representation. Thus, the task of representing and using mathematical, commonsense, visual, language-based, and logical knowledge is "lumped" together under the rubric of KRR (for examples of the diversity of goals of KRR see [Rich, 1991]). We will argue that, in the task of natural language processing and understanding, language syntax, semantics, and pragmatics imposes constraints on any computational formalism.

The task of knowledge representation for natural language processing and understanding is a knowledge-intensive one. To understand a natural language sentence, a natural language processing (NLP) system must, minimally, be able to represent the content of the sentence in the language of representation. For the most part, representation languages have been largely unmotivated with respect to the natural language they may be representing. In [Rich, 1991], only six of the twenty-two KRR systems presented are driven by natural language processing concerns. This lack of concern with the natural language that is being represented is typified by Conceptual Dependency (CD) [Schank, 1972;

Schank and Colby, 1973]. In CD, the goal is the representation of a language-independent canonical form corresponding to the "meaning" of a sentence. The attempt to represent a natural language sentence in this form does not take the form of the language into account, making the task of translation from surface sentences into knowledge representation more difficult. In this respect, CD is no different as a choice of KR language for NLP than first-order predicate logic. Logic is a more popular choice of representation language because of its prefabricated syntax and model-theoretic semantics. However, the syntax and semantics of first-order predicate logic is arguably not that of natural language. I will argue that if the domain of representation is natural language text, then the (minimally, syntactic) form of the natural language should strongly influence the representation. In particular, the language of first-order predicate logic is not the best choice of representation languages (it is probably better suited to a mathematical domain; see [McAllester, 1989] for a good example).

If we contrast the various goals of researchers in the domains for which these kinds of knowledge are to be used, we may come to the conclusion that the goals differ significantly and, indeed, may conflict. Consider the traditional use of representations based on first-order logic. Here, we have a system that has a powerful, well-understood inferential machinery but weak expressive power. Collections of logical formulas do not seem to capture the intuitive use of concepts by people. This representation for knowledge is unstructured and disorganized. What is missing is that first-order predicate logic does not provide any special assistance in the problem of what Brachman called "knowledge structuring" [Brachman, 1979]. That is, the specification of the internal structure of concepts in terms of roles and relations between them as well as the subsumption and inheritance relationships between concepts. Attempts to incorporate knowledge-structuring into the representation language are typified by frames, or frame description languages. These are slot-filler structures representing concepts that stand in various relationships (e.g., taxonomic) to each other. However, this shift to slot-filler structures, where the expressive power is greater but the inferential machinery is underspecified, perhaps goes too far. The cost of knowledge-structuring is a weakening of the inferential machinery available.

Logic is a traditional representational medium for attempts to understand natural language. As a formal language, logic has the desirable property that expressions of a formal language can be paired with interpretations. Thus, a "correct" mapping from natural language sentences into logic allows the original sentences to be interpreted. However, the mapping from natural language structures into the standard syntax of logic (and from logic to natural language) is unnatural in that the structure of the natural language (which can be significant) is usually lost. This is typified by the way in which complex noun phrases are separated into simple terms that impose type and other restrictions, in the translation to logic. This is a consequence of the atomic nature of variables in logic and results in the separation of constraints on variables from the variables themselves. This most often occurs when these constraints are moved to the antecedents of rules that type the variables. This results in the loss of the simple predicate-argument structure of the original sentence being represented. A simple example and its first-order predicate logic representation serves to illustrate this point:

$$\textit{All red sports cars are expensive} \overset{logic}{\Longleftrightarrow} \forall x\,((\text{red}(x) \wedge \text{sports-car}(x)) \Rightarrow \text{expensive}(x)).$$

Notice how the noun phrase *all red sports cars* is represented as two terms and a quantifier prefix. A simple-minded re-rendering of this in English would be *for all things x, if x is red and if x is a sports car, then x is expensive.* The unnatural form of the straightforward natural language re-rendering of this predicate logic representation reflects the unnatural form of the representation.

Logic satisfies criteria in a purely abstract domain, that of an ideal model. In contrast, language reflects an imperfect and unknowable world. Because of this, any *direct* mapping from language to first-order predicate logic (FOPL) is probably doomed to failure. This, however, does not preclude the possibility of more *natural* logics that resemble more closely the natural language they purport to represent.

1.2 Motivation

This work is based on the assumption that the domain of the knowledge to be represented and its associated goals have a profound effect on the design of a KRR system. In particular, we will present a KRR system for the representation of knowledge associated with natural language dialog. We will argue that this may done with minimal loss of inferential power and will result in an enriched representation language capable of supporting complex natural language descriptions, some discourse phenomena, standard first-order inference, inheritance, and terminological subsumption. Characterizing these features explicitly, some of the goals of a more natural logic and its computational implementation in a knowledge representation and reasoning system are:

- It should possess as much of the machinery of traditional FOPL as possible. It is not an accident that logical form representations are popular; it is a consequence of their power and generality. We feel, however, that, for the purposes of natural logics, completeness (and possibly traditional forms of soundness) are secondary issues because language, as it is used by people, does not appear to be either complete or sound, always.

- The mapping from natural language sentences into logical form sentences should be as direct as possible, and the representation should reflect the structure of the natural language sentence it purports to represent. This is particularly well illustrated by rule-type sentences, such as "small dogs bite harder than big dogs," whose representation takes the form of an implication whose antecedent constraints specify what types of dog bite harder than another type. This representation, as a logical rule, contrasts with the predicate-argument structure of the original sentence, as below:

  $$\forall x, y((\text{small}(x) \wedge \text{dog}(x) \wedge \text{large}(y) \wedge \text{dog}(y)) \Rightarrow \text{bites-harder}(x, y))$$

  By comparison, the representation of *Fido bites harder than Rover* is more consistent with the original sentence,

  $$\text{bites-harder(Fido, Rover)}$$

  This is so, despite the intuitive observation that the two sentences have nearly identical syntactic structure and similar meaning.

- The subunits of the representation should be conceptually complete in the sense that any component of the representation of a sentence should have a meaningful interpretation independent of the interpretation of the entire sentence representation. For example, the sentence “dogs bite” is typically translated as:

  $$\forall x[\text{dog}(x) \Rightarrow \text{bites}(x)].$$

  With this translation, we might ask what the meaning of x is? Presumably, some thing in the world or a set denoting a non-empty universe. Note that the original sentence mentions only dogs. We suggest that a better translation might be:

  $$\text{bites}(\forall\, x \text{ such that } \text{dog}(x))$$

  where the variable, x, has its own internal structure that reflects its conceptualization. Note that we are suggesting something stronger than just restricted quantification (the above could also be represented as $(\forall x : \text{dog}(x))\,[\text{bites}(x)]$ using restricted quantifiers). Complex internalized constraints (that is, other than simple type) and internalized quantifier structures characterize this proposal for the representation of variables. Thus the representation of the sentence: *Every big dog that is owned by a bad-tempered person bites* should have the same structure as the representation of *dogs bite.*

- A high degree of structure sharing should be possible, since multi-sentence connected discourse often uses reduced forms of previously used terms in subsequent reference to those terms. This corresponds to the use of pronouns and some forms of ellipsis in discourse. An example of this phenomenon is the representation of intersentential pronominal reference to scoped terms, e. g.,

  > Every apartment had *a dishwasher.* In some of them, *it* had just been installed.
  >
  > Every chess set comes with *a spare pawn. It* is taped to the top of the box.

  (examples from [Heim, 1990]). The structures that are being shared in these sentences are the variables corresponding to the italicized noun phrases. Logical representations can only model this "sharing" by combining multiple sentences of natural language into one sentence of logic. This method is unnatural for at least two reasons. First, when several sentences must be so combined into one sentence, the resulting logical sentence is overly complex as a conjunction of several potentially disparate sentences. Second, this approach is counter-intuitive in that language users can re-articulate the original sentences that they represent, which argues for some form of separate representations of the original sentences. The problem with logic in this task is that logic requires the complete specification of a variable, corresponding to a noun phrase, and its constraints in the scope of some quantifier. This difficulty is not restricted to noun phrases; indeed, it is frequently the case that entire subclauses of sentences are referred to using reduced forms such as "too," e. g.,

John *went to the party.* Mary did, *too.*

A language-motivated knowledge representation formalism should model this sort of reference, minimally by some form of structure sharing or co-referring notation.

- Any computational theory must incorporate knowledge-structuring mechanisms, in particular, subsumption and inheritance of the sort supported in frame-based and semantic network based systems. A taxonomy provides "links" that relate more general concepts to more specific concepts. This allows information about more specific concepts to be associated with their most general concept, and information filters down to more specific concepts in the taxonomy via inheritance. More general concepts in such a taxonomy *subsume* more specific concepts, the subsumee inheriting information from its subsumers. For atomic concepts, subsumption relations between concepts is specified by the links of the taxonomy. A clear example of subsumption in natural language is the use of descriptions such as *person that has children* subsuming *person that has a son.* If one were told: *People that have children are happy,* then it follows that *People that have a son are happy.* The intuitive idea is that more general descriptions should subsume more specific descriptions of the same sort, which in turn inherit attributes from their more general subsumers.

1.3 ANALOG

We have presented some general arguments for considering the use of a more "natural" (with respect to language) logic for the representation of natural language sentences. We have also presented some characteristics of natural language that a knowledge representation and reasoning system should support. In the rest of this dissertation, we will clarify the motivations for this work with specific examples; present an alternative representation for simple unstructured variables, which involves according variables potentially complex internal structure; and specify the logic of these variables. This involves providing a syntax and semantics of the logic of structured variables. The syntax of the logic is specified by a complete definition of a propositional semantic network representation formalism. The implemented system is called ANALOG (A NAtural LOGic).

The representations that result from using these structured variables have the following advantages:

- The representation of numerous types of quantifying expressions, using structured variables, is more "natural" than typical logics in the sense that the mapping of natural language sentences is more direct. Thus, parsing and generation of natural language sentences (particularly those involving restrictive relative clauses) is simplified.

- With structured variables that contain their own binding structures (quantifiers), no open sentences are possible. This is consistent with natural language, where few, if any, open sentences occur (minimally, noun phrases are, by definition, typed).

- Sentences involving non-linear quantifier scopings, such as branching quantifiers and the donkey sentences can be represented because quantifier scope specification is internal to the structured variable and inherently nonlinear.

- Subsumption can be defined syntactically in terms of the structure of variables. For example, from a rule representing *Every boy loves a girl*, all rules involving more restricted boys follow directly by variable subsumption (e.g., *Every boy that owns a* *red car loves a girl*). This sort of subsumption can be defined without the traditional distinction between terminological and assertional components, and has much utility.

- Since the underlying representation is semantic network-based, the representation of sentences that share structures (e.g., *Every farmer that owns a donkey beats it. His wife does, too.*) is both possible and reasonable. This allows the representation of ellipsis and pronouns directly.

We will show that the cost of gaining these advantages with respect to natural language processing is a loss in representational completeness. Not all possible quantified sentences can be represented. We will show how complementary work on the representation and use of collections can rectify this difficulty.

1.4 Dissertation Outline

We have just presented some general arguments for considering the use of a more "natural" (with respect to language) logic for the representation of natural language sentences. We also presented some characteristics of natural language that a knowledge representation and reasoning system should support. Chapter 2 addresses the general goals of natural form, conceptual completeness, the representation of non-linear quantifier scoping, and the specific problems of the branching quantifier and donkey sentences with respect to logic-based knowledge representation and reasoning formalisms. Chapter 3 reviews the related literature and places this work in context. Chapter 4 motivates and presents a structured representation for variables, as an initial attempt to address the goals of this work. Chapter 5 presents a formalization of a propositional semantic network-based knowledge representation and reasoning system, called ANALOG, that addresses our general goals for natural language processing. Chapter 6 illustrates the utility of the formalism for natural language processing with specific examples of natural language. Chapter 7 summarizes our work, indicates the major contributions, and suggests several areas for further work.

Chapter 2

Knowledge Representation and Reasoning for Natural Language Processing

In designing a knowledge representation and reasoning formalism for natural language processing, we feel that there are several language-related issues that must be considered. These issues include (but are not limited to):

- Expressiveness and generality of the representation language with respect to natural language, for example, coverage of complex object descriptions and treatment of quantification.
- Inference methods that parallel reasoning in natural language. Natural deduction systems are so-called because of the "naturalness" of the proof procedure.
- Ability of the formalism or system to capture important semantic or pragmatic aspects of natural language, for example, the computational relationship between the representation language and the parser or generator.

First-order predicate logic (FOPL) is a common choice of representation language for natural language. Indeed, if more expressive languages are used, they are, typically, mapped into first-order representations for actually performing inference. As such, it is instructive to consider those natural language constructions which are problematic for FOPL to see where an alternative representation language (possibly based on the first-order language) might be better. Additionally, there are a variety of expressions that can be readily used in ordinary language that are not expressible (in a straightforward manner) in standard knowledge representation languages, especially those based on first-order predicate logic.

2.1 Natural Form

Whatever logic (and associated language) we design, it would be desirable if our representations for English sentences bore a closer structural resemblance to their natural language origins than those of first-order predicate logics. This is desirable for two reasons. First, it would make the mapping process from language to a meaning representation (i.e., parsing or understanding) and back out to language (i.e., generation) computationally simpler. Second, the structure of the representations of syntactically similar sentences would also be similar, forming "classes" of representations. Logic does not preserve much of the structure of the original language. FOPL imposes a significant notational burden on representations that is hidden, or implicit, in the natural language. Figure 2.1 illustrates these assumptions by showing direct mappings from English to FOPL and back to English. Note the separation of type and other constraints, as well as the imposition of an implication and a conjunction to restrict the variables to the right sorts. The example of Figure 2.1 is a simple one, and the literal re-translation is, perhaps, a strawman since with minimal additional processing a more reasonable re-translation is possible. However, more complex examples where the re-translation process would be more difficult are trivial to construct. For brevity, we use this example and provide more interesting examples later.

Clearly, the structure of the representation is unnatural, relative to the original language (but see [Russell, 1920] Chapter XVI: Descriptions, for an opposing view). In part, this is caused by the separation of the intrinsic type (and other) constraints from the variables and the introduction of explicit quantifier prefixes. We would prefer to represent

Every boy loves a girl

$\Downarrow$ FOPL

$$\forall x\ (\mathrm{boy}(x) \Rightarrow \exists y\ (\mathrm{girl}(y)\ \&\ \mathrm{loves}(x,y)))$$

$\Downarrow$ Literal English

For everything, if it is a boy, then there exists a thing that is a girl and is such that the first thing loves it.

Figure 2.1: FOPL for: *Every boy loves a girl*

the above as just: loves(every boy, a girl), where the variables are typed and quantified as part of their internal structure. This desired representation reflects the goal of knowledge structuring and efficient representations of complex descriptions. There are two current solutions: using restricted quantifiers where variables are typed and using frame description languages (FDLs) to represent complex descriptions. Restricted quantifiers do not provide a general mechanism for complex descriptions (such as sentences with restrictive relative clauses, ie., *every boy that owns a dog*). FDLs can represent the descriptions: every boy and a girl in a terminological component and the `loves` predicate a assertional component that is typically first-order-logic based. This is what is done in the KRYPTON system [Brachman *et al.*, 1983; Brachman *et al.*, 1985]. Certainly, these represent productive attempts to address the motivation of natural form (among other motivations). However, we would prefer a more uniform representation in one logical language that allows complex restrictions and maintains natural form without two different components and the attendant potential for semantic difficulties (Section 3.3 has an example of these difficulties).

2.2 Completeness of Subformulas

In typical logics, terms in one formula are not referenced in other formulas. In general, re-using a term involves re-writing the term in the new formula. For example, a representation of a plan for building a pile of three blocks might involve performing three subacts (putting a block on the table, putting a second block on the first block, and finally putting a third block on the second block). A NLU system that processed this type of sentence might take a sentence like:

```
A plan to pile a block on another block on a third block
        is to put the third block on the table and then
        put the second block on the third block and then
        put the first block on the second block.
```

as input. This sentence is from an actual NLP system that discusses, uses and recognizes plans [Kumar *et al.*, 1988; Shapiro *et al.*, 1989]. Given this input, it would be useful (for plan recognition, for example) to represent the knowledge that each "putting" subact is a component of a plan to build a pile, by following with the sentence:

```
Each of the acts of piling one block on another block is a
        component of the plan for piling blocks.
```

where the acts referred to are the acts of the first sentence. However, in a logic-based representation we must re-write the subacts as separate terms in new formulas that represent this sentence. Moreover, even if we use some sort of co-referencing notation (for example, as in a unification notation), the subacts are open sentences, since the quantifiers are binders associated with the top-level plan representation of the first sentence. An alternative is to combine the representation of both sentences into one representation and use the *same* quantified blocks. The problem then becomes how to decide when to combine successive sentences when processing natural language text. Additionally, the resulting representations are overly complex relative to the natural language sentences they represent.

This difficulty is even more apparent in propositional semantic network representations. Figure 2.2 gives a (slightly abridged) representation for these sentences in the Semantic Network Processing System (SNePS) [Shapiro and Rapaport, 1987]. The node labelled *M1!* represents the proposition that a plan for piling three blocks is to put the third block on the table, then put the second block on the first block, followed by putting last block on the second block. The nodes corresponding to the constituent putting acts (the shaded nodes, *P5, P6,* and *P7*) are open terms (their quantifiers and types are specified separately at the level of the whole formula, *M1!*). Thus, the attempt to represent the proposition that *P5, P6,* and *P7* are components of the plan *P10* would result in the assertion of an open sentence *M2!*, which is meaningless (and not possible in SNePS). The problem is that the type constraints (that the variables are blocks) is associated with the proposition rather than the variables. This seems strange since it is perfectly plausible to say the proposition expressed by *M2!*.

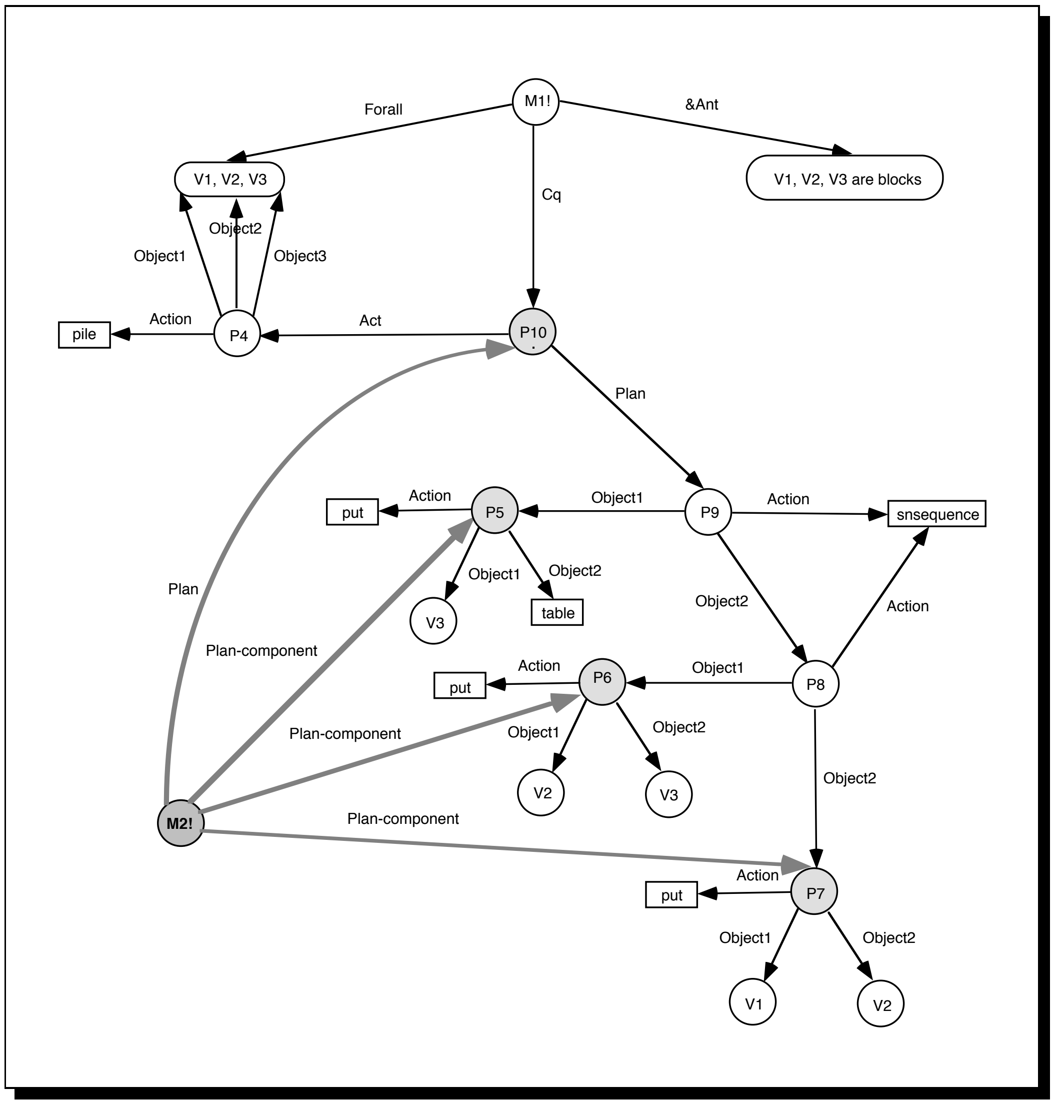


Figure 2.2: Difficulty in Representation of Plan-Component

Ideally, for examples such as this, we would like to re-use exactly the same terms in different formulas, and we would like these terms to be closed formulas, so that they may be meaningfully re-used and shared by multiple formulas. Terms that can be re-used in this manner we call *conceptually complete* and are formally defined in Chapter 5. This is also an issue in the representation of multisentential dialog, where intersentential reference to sententially scoped objects frequently occurs. For example,

> Every apartment had *a dishwasher.* In some of them, *it* was new.
> *A train* leaves for NYC in the morning. *It* is slow.
> All cars come with *a spare tire. It* is not a full-sized tire.

are sentences that can only be represented in FOPL by re-writing the terms corresponding to the shared variables in all representations of sentences that use them or combining several sentences into one logic representation. To see the difficultly, consider a representation of the last example:

$$\forall x\ (\text{car}(x) \Rightarrow \exists y(\text{spare-tire}(y) \wedge \text{comes-with}(y, x)))$$

$$\forall z\ \exists w[(\text{comes-with}(w, z) \wedge \text{car}(z) \wedge \text{spare-tire}(w)) \Rightarrow \neg\text{full-sized}(w)]$$

There is no connection between terms in one sentence representation and the other, even though the original language explicitly makes the connection (with the use of the pronoun *it*). The alternative solution is to combine the two natural language sentences into one logical sentence, as in:

$$\forall x(\text{car}(x) \Rightarrow \exists y(\text{spare-tire}(y) \wedge \text{comes-with}(y, x) \wedge \neg\text{full-sized}(y)))$$

However, for any connected series of sentences much longer than this example, this is an unwieldy solution. Logical sentences would get lengthy, and there would be no reasonable way to reconstruct the original natural language sentences. In these cases, there is a clear advantage to a conceptually complete (closed) variable representation (since there are no scope traps) as well as the structure sharing associated with a semantic network representation. We present the representation for this example in our formalism in Figure 4.4.

2.3 Quantifier Scoping

Any representation must account for the expression of quantifier scoping at least as simply as the FOPL linear notation. The linear notation implicitly expresses a partial ordering of quantifiers and terms, by the order and bracketing of the quantifiers. For example, given an unbracketed formula such as $\forall x\ \exists y\ P(x,y) \Rightarrow Q(x)$, we may express three distinct formulas by re-ordering quantifiers and bracketing terms, as below.

$$\begin{array}{l}\forall x\ \exists y\ (P(x,y) \Rightarrow Q(x))\\ \forall x\ ((\exists y\ P(x,y)) \Rightarrow Q(x))\\ \exists y\ \forall x\ (P(x,y) \Rightarrow Q(x))\end{array}$$

Thus, a desirable property of any complete representation language is the ability to express, unambiguously, any or all the above readings of the unbracketed and unscoped formula. As we show, this is not trivial for a formalism with the properties we desire. Additionally, we would like to account for some of the traditionally difficult (for FOPL) types of scoping associated with natural language sentences, described below.

2.3.1 Branching Quantifiers

There is a class of natural language sentences that are not expressible in any linear notation [Quine, 1969; Quine, 1970; Fauconnier, 1975; Barwise, 1979]. These sentences require a nonlinear quantifier prefix. An example is:

(1) `Some relative of each villager and some relative of each townsman hate each other.`

Linear notation requires that one existentially quantified relative (of the townsman or villager) scope inside both universally quantified variables. A FOPL representation for (1) would look something like:

$$\forall v \exists r_1 \forall t \exists r_2 \text{Hates}(r_1, r_2)$$

where (in this representation) r_2 existentially depends on both v and t (villager and townsman, respectively). Because of this, there is a dependency that should not be there, since the relative of the villager depends only on the particular villager and the relative of the townsman depends only on the particular townsman. Examples of these types of quantifiers are called *branching quantifiers*, because expression of their scoping requires a tree-like notation (the Henkin prefix [Henkin, 1961]). For example, (1) could be expressed as in Figure 2.3. Using this augmentation to the logical language, the expression of all partial orderings of quantifiers is possible. This is not expressible in the standard syntax of first-order logic, but can be expressed in second-order logic as in Figure 2.4, where the existentially quantified functions P and Q are Skolem functions that express the non-linear scope dependencies.

$$
\begin{array}{ll}
\forall x \text{ — } \exists y & \\
\qquad\quad \backslash & \\
 & [(\text{villager}(x) \wedge \text{townsman}(z)) \Rightarrow \\
\qquad\quad / & \qquad (\text{relative}(x,\, y) \wedge \text{relative}(z,\, w) \wedge \text{hates}(y,\, w))] \\
\forall z \text{ — } \exists w &
\end{array}
$$

Figure 2.3: **Branching quantifier representation for:** *Some relative of each villager and some relative of each townsman hate each other.*

$$
\begin{array}{l}
\exists P\ \exists Q\ \forall x\ \forall z\ [(\text{villager}(x) \wedge \text{townsman}(z)) \Rightarrow \\
\qquad (\text{relative}(x, P(x)) \wedge \text{relative}(z, Q(z)) \wedge \text{hates}(P(x), Q(z)))]
\end{array}
$$

Figure 2.4: **Second-order representation for:** *Some relative of each villager and some relative of each townsman hate each other.*

2.3.2 Donkey Sentences

Another class of sentences that are difficult for first-order logics are the so-called *donkey sentences* [Geach, 1962]. These correspond to sentences that pronominally refer to quantified variables in closed subclauses, for example:

```
Every farmer who owns a donkey beats it.
```

where the noun phrase *a donkey* is a variable inside the scope of a universally quantified variable (*every farmer*) and is referred to pronominally outside the scope of the existentially quantified donkey. Consider some attempts to represent the above sentence in FOPL:

(a) $\forall x$ (farmer(x) $\Rightarrow \exists y$ (donkey(y) & owns(x, y) & beats(x, y)))

(b) $\forall x$ (farmer(x) $\Rightarrow$ ($\exists y$ (donkey(y) & owns(x, y)) $\Rightarrow$ beats(x, y)))

(c) $\forall x$ $\forall y$ ((farmer(x) & donkey(y) & owns(x, y)) $\Rightarrow$ beats(x, y))

Representation (a) says that *every* farmer owns a donkey that he beats, which is clearly more than the original sentence intends. Representation (b) is a better attempt, since it captures the notion that we are considering only farmers who own donkeys; however, it contains a free variable. Representation (c) fails to capture the sense of the original sentence in that it quantifies over *all* farmers and donkeys, rather than just farmers that own donkeys. To see this, consider the case of a model with only one farmer who owns two donkeys and beats only one of them. Clearly, the $\forall\exists$ reading of the donkey sentence can apply to this case, but interpretation (c) does not.

2.4 Summary

We have presented some goals and problems for any knowledge representation and reasoning formalism for natural language processing. These are the general goals of natural form, conceptual completeness, and the representation of non-linear quantifier scoping and the specific problems of the branching quantifier and donkey sentences. In the remaining chapters, we survey the related work on these (and other related) problems and present a partial solution to these difficulties that has additional desirable properties for natural language processing. These properties include a representation for complex descriptions, a specification of subsumption between these descriptions, and a "natural" inference procedure.

Chapter 3

Related Work

> Much of the work in KR has involved inventing new KR formalisms and embedding these within KR systems. In retrospect, a good proportion of this research can be seen as the search for useful compromises between expressive power, on the one hand, and tractability of reasoning, on the other. [Levesque, 1986]

A useful test of any augmentation of a KR system is the extent to which that augmentation compromises the ability of the system to do inference. Before we discuss other systems, we note that the augmentation suggested in this dissertation, structured variables, is intended to extend the expressibility of the representation to account for a variety of natural language sentences without losing the inferential machinery and model-theoretic semantics of first-order logic.

A fruitful path towards a more natural logic (and one that is compatible with our work) is that of the nonmonotonic logics [McCarthy, 1980; McDermott and Doyle, 1980; Reiter, 1980]. These logics attempt to make the reasoning process more "natural" and generally more tractable. They are augmentations of first-order logics that allow processes such as default reasoning and circumscription in inference. In general, since these logics are based on first-order logic and address different issues than this proposal, these systems are not of immediate interest (except to the extent that they might be augmented with structured variables).

This dissertation will suggest a hybrid approach to KR, by combining semantic network representations and structured object representations (such as frames) in a limited way (only for variables). This produces a KR formalism that provides the advantages of both without too many of their respective disadvantages. For comparative purposes, we group the related work by the amount of structure accorded the representation of variables. This results in three natural categories: structured variable representations, atomic (unstructured) variable representations, and hybrid variable representations.

3.1 Structured Variable Representations

There is a large body of work in structured object representation that characterizes very complex structured variables as frames, scripts, and so on. Frame-based systems such as KL-ONE and KRL use highly structured concept representations to express the "meaning" of concepts [Bobrow and Winograd, 1977; Brachman and Schmolze, 1985; Woods and Schmolze, 1992]. These concept representations are constructed using structural primitives. These highly structured objects, which typically consist of slots and fillers (with other mechanisms, such as defaults), can be viewed as complex structured variables that bind objects with the appropriate internal structure (although they are not, typically, so viewed). Their use of structural primitives allows the specification of a subsumption mechanism between concepts. A difficulty with these representations is that structured representations correspond directly to predicates in the underlying logic. Thus, constituents of a structured concept are not available as terms in the logic. In the donkey sentence, the donkey in *farmer that owns a donkey* cannot be used as a term.

An alternative to the frame-based, highly structured object representations is that of a logical form representation. The general motivation for a logical form is the need for a mediating representation between the syntactic and meaning representations, often in the context of determining quantifier scoping. Representative examples of logical-form based approaches include [Woods, 1978; Schubert and Pelletier, 1982; Hobbs and Shieber, 1987]. In particular, Hobbs and Shieber present a logical form that allows quantified variables to be expressed without any information about the relative scope of each quantifier. Logical form representations resemble our work in that, typically, quantifiers are bundled with typed variables that are "complete" in our preferred sense. Additionally, the utility of such representations lies in the relative ease of mapping natural language into a representation, which is also, clearly, a goal of this proposal. However, logical form is not a meaning representation, unlike the other representational work considered here. The representations of this work are fully scoped and unambiguous. In general, logical form representations provide a "weakest" ambiguous interpretation that is subject to further computation before its meaning is apparent. For example, Hobbs and Shieber provide an algorithm for determining all the possible scoping of an ambiguous logical form. Thus, logical form representations are, at best, an intermediate step in the process toward a meaning representation of the sort suggested in this research. It is possible to view this our research as an "improved" logical form that has the advantage of having a "natural" mapping from language to representation; however, the improvements are significant and include clear specification of quantifier scoping, incorporated into a semantic network system, with structure sharing (of nodes and terms) and cyclic structures of the sort seen in English (as in the donkey sentence example), which are not easily represented in a linear notation.

3.2 Atomic Variable Representations

Previous work in AI using atomic (that is, unstructured) variable representation has been primarily based on first-order logic. Thus, much of it suffers from the previously discussed shortcomings for the representation of natural language. However, some of the atomic variable approaches can address some (but not all) of the difficulties we address in this research. A representative first-order logic-based approach is that of Webber [Webber, 1983], which attempts to specify logical forms for various types of complex descriptions. The work suggests that restrictions on variables can be complex (in much the same manner as our research); however, Webber's syntax and semantics are informally presented, and the utility of the logic to the problems addressed in our research is unclear.

In the work of Schubert et al. [Schubert *et al.*, 1979; Cercone *et al.*, 1992], which uses a semantic-network-based formalism (lately named ECO), variables are atomic nodes in the network. Type (and other) restrictions are specified by links to the variable nodes. There are no explicit universal or existential quantifiers. Free variables are implicitly universally quantified; Skolem arcs specify existentially quantified variable nodes. In Figure 3.1, the dotted arrow denotes a scope dependency between the universally quantified "man" and the existentially quantified "woman." The implication (as in the FOPL representation) is represented by the "$\Rightarrow$" connected to the variables and the predicate "likes." Because this is a nonlinear notation, sentences with branching quantifiers can be represented. However, the separation of variables from their constraints causes the representation to be not "natural" relative to the original natural language. Moreover, since restrictions on possible fillers for variables appear to be simple type restrictions, there is no representation for noun phrases with restrictive relative clause complements, and consequently no representation for donkey sentences. The work of [Sowa, 1984; Sowa, 1992] with *conceptual graphs*, a form of semantic networks, is similar and has the same difficulties. Sowa does discuss the donkey sentence and his treatment of it resembles the DRS approach, described below.

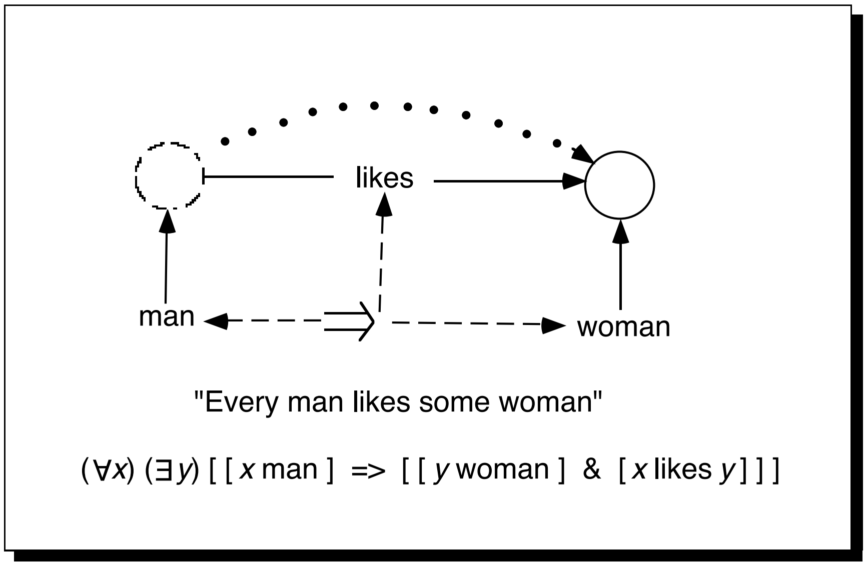

likes
man
woman
"Every man likes some woman"
(∀x) (∃y) [[x man] => [[y woman] & [x likes y]]]

Figure 3.1: Schubert's representation for: *Every man likes a woman*

Another semantic-network based formalism is the SNePS system of [Shapiro and Rapaport, 1987; Shapiro and Rapaport, 1992]. SNePS has many of the same difficulties in addressing the goals of this dissertation as Schubert's work. However, many of the motivations for the SNePS formalism come directly from concerns associated with natural language processing. Thus, it formed a natural starting point for new work. The work presented here builds on SNePS, by augmenting it to address our concerns, and so we defer discussion of SNePS's salient features until Chapters 4 and 5, where they are discussed at length.

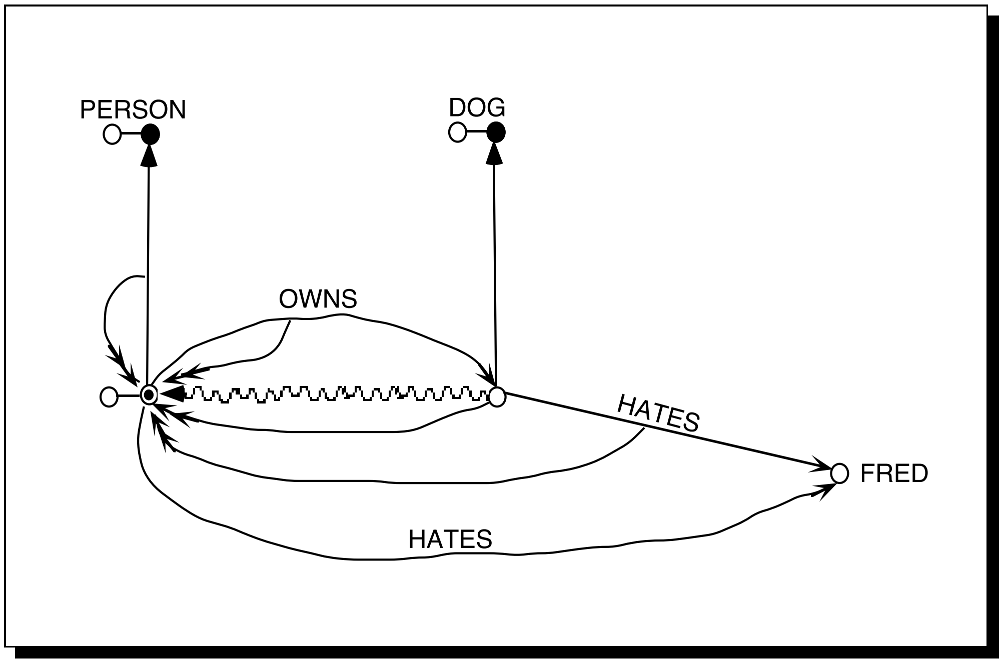


Figure 3.2: NETL representation for: *Any person who owns a dog that hates Fred also hates Fred*

Fahlman's [Fahlman, 1979] work specified two representations for variables. The first representation is as a *TYPE-node that represents the "typical-member" of a set. This approach suffers from the typical inheritance problems (what Fahlman calls the "copy-confusion" problem) associated with such representations [Shapiro, 1980]. The second representation for variables corresponded to universally quantified nodes called *EVERY-NODEs. Restrictions on these nodes can be general type or relative clause restrictions and are directly linked to the *EVERY-NODE. Existentially quantified variables are of type *INDV with skolem arcs to *EVERY-NODE variables to indicate scope dependencies.

An example is shown in Figure 3.2 of his representation for *Any person that owns a dog that hates Fred also hates Fred.* In Figure 3.2 the node with the black center dot is the universally quantified *person that owns a dog that hates Fred.* The labelled unfilled-filled node pairs correspond to classes. The arcs with double arrowheads correspond to what Fahlman calls *clauses* which correspond directly to clausal restrictions on variables. The wavy line expresses the scope dependency between it and the *dog that hates Fred.* Finally, the *hates* relation of the top-level sentence is expressed by the arrow from the *person* node to the *Fred* node. This is more general than the variable representation of Schubert et al., because of the representation of complex clause restrictions. However, the form of the representation and its interpretation (as defining a set, rather than stating a proposition) is, again, unnatural relative to the sentence it represents. In principle, Fahlman's representation of variables appears capable of representing donkey sentences and branched quantifiers sentences; however, he does not discuss this, and semantics for such representations are unclear. Fahlman's representation is similar to this proposal to the extent that his variables have (potentially complex) internal structure (are structured) and are implicitly quantified (conceptually complete). However, his semantics for these variables are specified in terms of node matching (a necessary task for any computational theory), and incompletely at that. We will provide a semantics embedded in predicate logic based on Fine's generic semantics for structured variables.

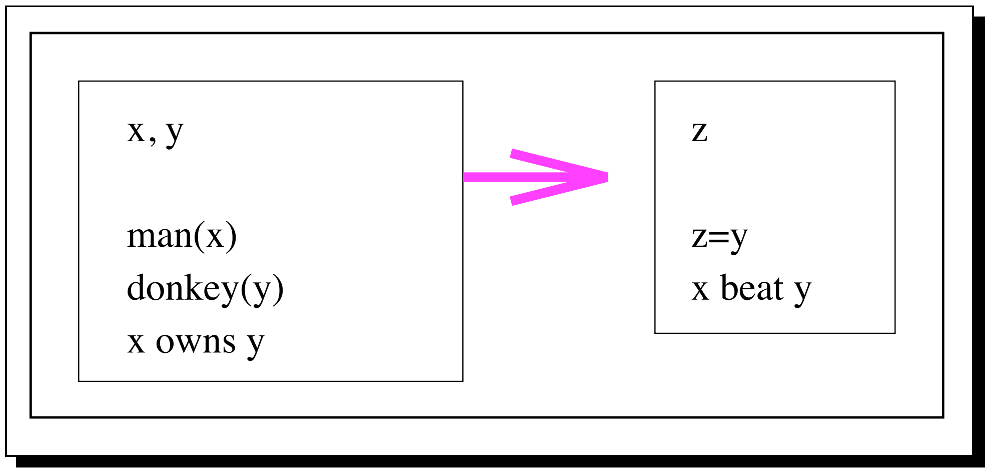


Figure 3.3: DRT representation of: *Every man who owns a donkey beats it*

An alternative atomic variable representation theory is that of Discourse Representation Theory, due to Kamp [Kamp, 1984]. DRT is a semantic theory that (among other things) accords indefinite noun phrases the status of referential terms rather than the standard quantified variables, and accords definite noun phrases the status of anaphoric terms. These terms are scoped by discourse representation structures (DRSs), and the theory provides rules to expand these DRSs based on the discourse being represented, as well as rules for interpreting the DRSs. DRT was directly motivated by the difficulties in the representation of donkey (other sorts of) sentences and deals with them by making the scope of terms (variables) be the DRS rather than the sentence (proposition). DRSs themselves may scope inside other DRSs, creating a hierarchy of DRSs and scoped terms. An example is shown in Figure 3.3, where the DRS for *every man who owns a donkey beats it* is given. Note that their interpretation of this DRS correspond to the reading where both the *farmer* and *donkey* are universally quantified, they *cannot* construct the existential interpretation without re-formulating their rules for DRS construction. The approach is similar to that of Hendrix's partitioned semantic networks but provides computational rules for processing discourse [Hendrix, 1977; Hendrix, 1979]. The underlying formal system is FOPL-based (though it, perhaps, need not be) and as with all the atomic variable representations, there is a separation of constraints from variables, and the form of DRSs is not "natural" in the same sense that a proposition that represented a sentence would be. Terms (variables or open sentences) are still scoped (although perhaps more widely) and not conceptually complete. Further, the rules of construction of DRS prohibit the representation of intersentential pronominal reference to scoped terms, e.g.,

Every apartment had *a dishwasher.* In some of them *it* had just been installed.

Every chess set comes with *a spare pawn. It* is taped to the top of the box.

(examples from Heim [Heim, 1990]). Thus these examples cannot actually be mapped into their representations (which may exist) in a way that is consistent with the theory. As DRT is primarily a theory of discourse these objections, springing from knowledge representation goals, are not serious. The research described here could be the underlying formal system for DRT with appropriate re-formulation of their construction rules.

An important philosophically motivated attempt to represent the semantics of natural language is Montague grammar [Dowty *et al.*, 1981; Montague, 1973; Montague, 1974]. Montague grammar is a theory of natural language in that it is a complete formal specification of the syntax, semantics, and knowledge representation for natural language understanding. Montague grammar actually *uses* the syntactic structure of the surface sentence in specifying the mapping to logic and interpretation. In that, it resembles this work, but its coverage is far more ambitious than, and exceeds the scope of, the work in this paper. For instance, we do not provide a theory for processing natural language into representations. However, as a compositional semantic theory based on a higher-order intensional logic, it provides no inherent facility for the description of discourse relations and anaphoric connections [Halvorsen, 1986]. Further, it suffers from the same, previously discussed, problems that all logic-based variable representations do. Montague grammar is significant because it provides strong evidence that the syntax of natural language sentences can itself aid the processing and representational task. A related body of work is that of Barwise and Cooper [Barwise and Cooper, 1981] on *generalized quantifiers* in natural language. They attempt to represent and provide semantics for more general types of quantified natural language sentences (e.g., *many, most*) and specify a translation of a fragment of English using phrase structure rules. Their discussion of semantic issues related to these generalized quantifiers (which are, typically, manifested as noun phrases) forms a productive basis for any attempt to specify the semantics of quantified noun phrases. They do not address the same issues of this work, but their work with other natural language quantifiers suggests useful ways to extend our theory.

Sommers [Sommers, 1982] defines a logical system called TFL which attempts to satisfy the goal of natural form (although he characterizes it as a logic based on natural language). Sommers' logical system is term algebra where the "+" and "−" are used to denote positive or negative quality (the latter corresponding to negated and universally quantified terms). So, for example, in the rules of transformation, there is a law of double negation expressed as $- - \alpha = \alpha$, where α is a term. In standard FOPL this corresponds to the equivalence: $\neg\neg t \equiv t$. Sommers' law *Dictum de Omni* below, provides a law of *modus ponens*,

$$\begin{array}{c} -\alpha \pm \beta \\ \underline{\ldots \alpha \ldots} \\ \ldots \pm \beta \ldots \end{array}$$

This corresponds to the FOPL derivation $\neg\alpha \vee \beta, \alpha \vdash \beta$. Using Sommers' term algebra the categorical propositions "every S is P" is represented as "— S + P". The nice thing about this system is that it allows the natural and direct representation of most natural language sentences. A difficulty, relative to our goals, is that noun phrases with restrictive relative clauses must be represented as conjunctions of simple terms and are not unit terms. The proofs associated with simple natural language inference are not very natural.

There is one category of logics for natural language that is missing from our enumeration, namely a logic without variables. While it may seem counter-intuitive that such a system could be appropriate for the task we are addressing, the work of [Purdy, 1991a; Purdy, 1991b] attempts to do so. Purdy describes a logical language (which makes use of Barwise and Cooper's work with generalized quantifiers) whose structure mirrors natural language. The language has no variables or individual constants; instead, singular predicates serve in this role. The language is many-sorted and expressively between the predicate calculus without identity and the predicate calculus with identity. Deduction in this language attempts to parallel reasoning in natural language (which Purdy calls "surface reasoning") in much the same way as this work. Purdy uses Barwise and Cooper's monotonicity principles for generalized quantifiers in specifying derived rules and suggests that this captures an essential and important element of natural language reasoning. Purdy's system incorporates a small attribute grammar for a fragment of English that he uses for examples. Purdy's motivations and goals are similar to ours, and it is interesting to see that a system that dispenses with variables can define a plausible logic for natural language. Interestingly, Purdy's work most resembles that of McAllester (described below), who provides a computational framework that is largely variable and quantifier free. However, Purdy's system is expressively equivalent to the predicate calculus, he does not address natural language examples such as the branching quantifiers and donkey sentences which are of interest here. His use of monotonicity for generalized quantifiers in his language parallels our use of subsumption, described later, although we allow for derived subsumption and he does not.

Another philosophically motivated attempt to represent variables is the work of Fine [Fine, 1983; Fine, 1985a; Fine, 1985b]. He suggests augmenting the ontology of classical models to include arbitrary objects. These arbitrary objects resemble structured variables in that they may have complex constraints on the individual objects in their range. The work of Fine in motivating and presenting a theory of arbitrary objects is particularly important for this work. It provides a formal interpretation for the syntax and semantics of arbitrary objects that is equally applicable to the structured variables of this dissertation. Thus, we examine Fine's theory in some detail and we summarize his generic semantics for arbitrary objects. Fine argues that there exist, in ordinary reasoning, procedures for reasoning to universal conclusions and from existential premiss from arguments about arbitrary objects. For example, we may establish that all triangles have interior angles summing to 180 degrees by showing that an arbitrary triangle has this property. From these informal arguments, Fine develops a formal augmentation to the standard FOPL that includes arbitrary objects.

L is an arbitrary first-order language, and **M** a classical model for **L** of the form (I, ...), where I is a non-empty set of domain individuals and ... indicates the interpretation of the non-logical constants of **L**. The latter may consist of a function that assigns to each predicate of degree n a set of n-tuples from I. Any classical model **M** may be augmented to one that contains arbitrary objects. Any such model $\mathbf{M}^+$ is of the form (I, ..., A, $\prec$, V), where:

(i) (I, ...) is the model **M**;

(ii) A is a finite set of objects disjoint from I;

(iii) $\prec$ is a relation on A;

(iv) V is a non-empty set of partial functions from A into I, i.e., functions v whose domain Dm(v) is a subset of A and whose range Rg(v) is a subset of I.

A is the set of arbitrary objects, and $\prec$ is the relation of dependence between arbitrary objects (thus, "$a \prec b$" indicates that the value of the arbitrary object a depends on the value of the arbitrary object b). V is the family of value assignments of individuals to arbitrary objects that are *admissible* in the model. So, if v $\in$ V, with domain $\{a_1, a_2, \ldots, a_n\}$ and $v(a_1) = i_1$, $v(a_2) = i_2$, ..., $v(a_n) = i_n$, then v is admissible.

The augmented language $\mathbf{L}^*$ includes the symbols $a, b, c, \ldots$ (called the A-letters) that refer to arbitrary objects and syntactically behave like individual names. Formulas of $\mathbf{L}^*$ may contain free variables, called *pseudo-formulas.* So Fa is a formula, Fx is a pseudo-formula. The augmented language contains three kinds of symbols that can appear in subject position: the variables $x, y, z, \ldots$; the constant symbols $m, n, p, \ldots$; and the letters $a, b, c, \ldots$. The latter two correspond to individuals and arbitrary individuals, respectively, the former to variables of quantification. $\mathbf{M}$ is extended to $\mathbf{L}^*$ by adding a designation function d which is a function taking A-letters of $\mathbf{L}^*$ into A. The resulting model $\mathbf{M}^*$ is of the form (I, ..., A, $\prec$, V, d).

For a $\in$ A, the *value-range* VR(a) of a is {v(a): v $\in$ V} and is the set of values it can assume. If VR(a) $\neq$ I, then a is *value-restricted*; otherwise it is *universal.* An a $\in$ A is *dependent* if a $\prec$ b for some b $\in$ A; otherwise it is *independent.* An a $\in$ A is *restricted* if it is value-restricted or dependent; otherwise it is unrestricted. We note that this will correspond directly to our use of these terms for structured variables.

Truth for generic statements is defined in two ways: *relative* and *absolute.* If $\varphi(a_1, \ldots, a_n)$ is a formula with A-letters as displayed, then φ is *true in* the model $\mathbf{M}^*$ *relative to* v $\in$ V (written $\mathbf{M}^* \models_{\mathrm{V}} \varphi$) if $a_1, \ldots, a_n \in$ Dm(v) and $\mathbf{M} \models [\mathrm{v}(a_1), \ldots, \mathrm{v}(a_n)]$. φ is *absolutely true in* $\mathbf{M}^*$ (written $\mathbf{M}^* \models \varphi$), if $\mathbf{M}^* \models_v \varphi$ for any v $\in$ V for which $a_1, \ldots, a_n \in$ Dm(v). In other words, a statement concerning arbitrary objects is true just in case it is true for all their values. Fine then proceeds to define validity for these models in terms of these two notions of truth. He then applies his generic semantics to a variety of formal systems, and shows how the augmented systems function, including soundness results.

What is interesting about Fine's formulation is the manner in which he characterizes the A-letters (corresponding to arbitrary objects) as an ordered pair $(a, \triangle)$ where a is the *defined term* and $\triangle$ (called *defining conditions*) is a set of pseudo-formulas in a single variable x. As we shall see, this corresponds directly to our definition of a structured variable. Fine's generic semantics serves as a useful basis on which to build a semantics for arbitrary objects, and in particular the formulation of structured variables in this dissertation. However, the theory suggested by Fine is not readily amenable to computational methods, as he provides no computational procedure for unifying terms with arbitrary objects and performing inference. Further, it does not address the specific issues (representation and use of natural language sentences involving branching quantifiers, donkey sentences, and anaphora) of this work. This is appropriate as his goal is a generic semantics for any first-order system. We will make use of Fine's generic semantics, as necessary formal underpinning for our theory, and focus on our main concerns; a computational knowledge representation theory that accounts for the natural language phenomena discussed here.

3.3 Hybrid Variable Representations

Hybrid variable representations accord concepts potentially complex internal structure. These concepts are then used to restrict suitably variables in the theorem prover. One representative system is the work of Brachman with KRYPTON [Brachman *et al.*, 1983; Brachman *et al.*, 1985]. KRYPTON is a KR system that supports (and separates) two kinds of knowledge: terminological (in the TBox) and assertional (in the ABox). A pictorial view of KRYPTON's architecture can be seen in Figure 3.4, where an example representation for *Some person's child is a doctor* is shown. KRYPTON can represent very complex descriptions in the Tbox by providing a diverse group of concept constructors (for an example of them look at the KRYPTON example to follow). Logic-based assertional representations in the Abox make use of the structured concepts, syntactically as unary predicates or as binary relations. In principle, general structured concepts with almost arbitrary restrictions are possible in the Tbox. Thus it is straightforward to represent the concept *farmer that owns a donkey* and use it in a quantified sentence as the restriction on a variable. However, constituents of complex terms (concepts) in the TBox are not available in the ABox, so the donkey sentence cannot actually be represented. What is appealing about KRYPTON is that the form of the representation is more natural than FOPL, since restrictions on objects, which can take the form of complex terminological constraints in the TBox, are simple predicates (in the ABox) on those objects. Further,

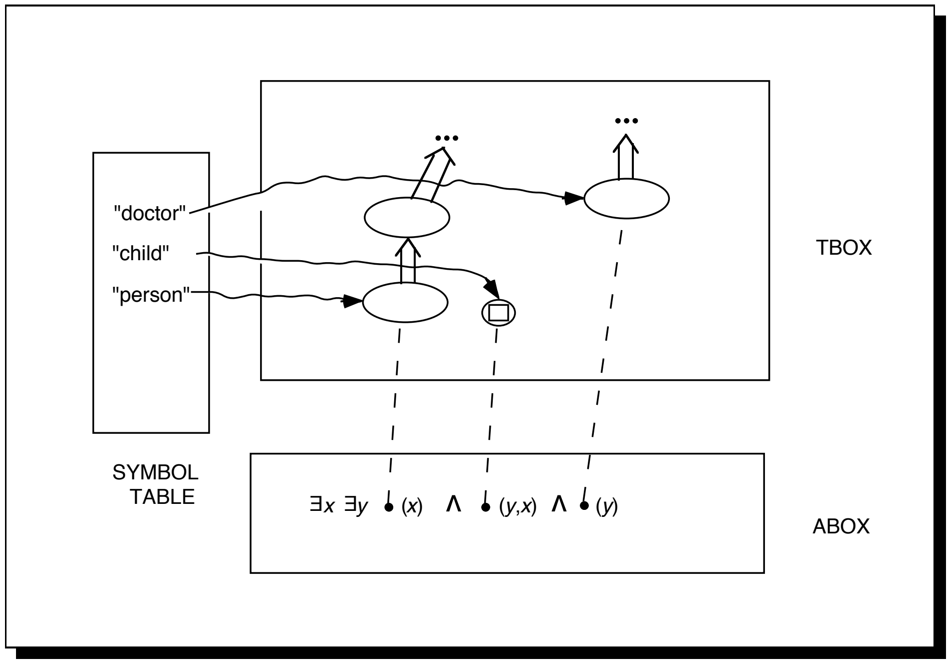


Figure 3.4: Architecture of the KRYPTON system

concept subsumption is defined on terms in the TBox allowing the obvious kinds of natural language based reasoning (e. g., if *Every woman is happy* then *Every rich woman is happy.* We shall have more to say about this when we present subsumption in our research. Finally, variables of the ABox are still atomic and since sentences of the ABox are FOPL-based, KRYPTON cannot represent branched quantifiers or donkey sentences.

A general problem with hybrid systems such as KRYPTON is the potential for inconsistency between reasoning in the TBox and reasoning in the Abox. [Beierle *et al.*, 1992] makes the point that hybrid KRR systems distinguish between taxonomical information (T-box) and assertional information (A-box) too rigidly. Thus, while information in the ABox can change, during inference or when new propositions are entered, information in the T-box tends to be static.

In some applications of NLP, this approach is insufficient since the taxonomical hierarchy may be changed while parsing new sentences. For instance, starting from an existing concept hierarchy, after processing the sentence *canaries are birds* a new entry in the hierarchy may have to be made (to add *canaries* under *bird*). Beierle and his colleagues argue for two types of "coupling" between information in the T-box and information in the A-box: close and loose. Loose couplings correspond to static sort hierarchies not influenced by the assertional KB. That is, inferential activity in the assertional component cannot change information in the terminological component. This is the approach of KRYPTON, where there is no interaction between the TBox and Abox. Beierle argues that in NLU this loose coupling is insufficient because the system should be capable of explicitly reasoning about the taxonomic information. This may in turn change the sort hierarchy. This latter interaction corresponds to what that they call a close coupling of the terminological and assertional components.

Beierle et al. specify an order-sorted predicate logic with a close coupling between the taxonomic information and the assertional part, using a model-theoretic semantics. Their system is called L_{LILOG}. In this system, the sort hierarchy taxonomic information is also expressible in the axiomatic part of a knowledge base, what they call a "close" coupling of taxonomic and axiomatic information. They point out the inconsistent semantics of systems with loose couplings such as the KL-ONE based systems, in particular, KRYPTON, with the following example:

$$\mathrm{KB}_0 := \mathrm{NEWKB}[]$$

defines a new knowledge base (KB).

$$\begin{aligned}\mathrm{KB}_1 := \mathrm{DEFINE}[&\textit{amphibious-vehicle},\\ &(\textbf{ConGeneric}\ \textit{car ship}),\\ &\mathrm{KB}_0]\end{aligned}$$

augments the KB_0 by defining the concept *amphibious-vehicle* as the conjunction of the concepts *car* and *ship* in the T-box. KB_1 is then augmented as follows:

$$\begin{aligned}\mathrm{KB}_2 := \mathrm{DEFINE}[&\textit{toy-amphibious-vehicle},\\ &(\textbf{VRGeneric}\\ &\quad \textit{amphibious-vehicle owner child}),\\ &\mathrm{KB}_1]\end{aligned}$$

VRGeneric builds the new concept *toy-amphibious-vehicle* from the generic concept *amphibious-vehicle* by restricting the value of the role (owner) to a specific concept (child). Now, under the KRYPTON semantics:

(1) ASK[$\forall x$ *toy-amphibious-vehicle*(x) $\rightarrow$ *car*(x), KB$_2$] = yes.

(2) SUBSUMES[*car*, *toy-amphibious-vehicle*, KB$_2$] = yes

(1) asks the theorem prover of the Abox to prove that all toy amphibious vehicles are cars. It can do so since the concept *car* subsumes *toy amphibious vehicle* (since the latter is a more specialized type of car), as shown in (2). However, replace the definition of KB$_1$ by KB$_1'$:

$$\mathrm{KB}_1' := \mathrm{TELL}[\forall x \; \textit{amphibious-vehicle}(x) \iff \textit{car}(x) \wedge \textit{ship}(x), \mathrm{KB}_0]$$

TELL puts its argument proposition in the Abox. This places similar information as in KB$_1$, namely that amphibious vehicles are cars, but in the Abox. Now, asking the same questions (1) and (2) we get different answers:

(1) ASK[$\forall x$ *toy-amphibious-vehicle*(x) $\rightarrow$ *car*(x), KB$_2$] = yes.

(2) SUBSUMES[*car*, *toy-amphibious-vehicle*, KB$_2$] = no

(2) fails because the terminological component does not have access to reasoning from the Abox. This argues that the coupling of the A-box and T-box is insufficiently close and the overall semantics (with both the T-box and A-Box) is far from being clear. L$_{LILOG}$ is primarily of interest, here, because of the strength of the argument its authors make for a close coupling between the assertional and terminological components of a knowledge representation formalism for NLP. As we shall show, our system make no distinction between the two components, everything is terminological, and a propositions belief status is the only assertional operator.

A class of systems that attempts to rectify these difficulties are the systems based on "description logic" [MacGregor, 1988; Peltason, 1991]. All these systems, like KRYPTON, are descendents of KL-ONE and feature separate assertional and terminological components. The LOOM KR system is a representative example [MacGregor, 1991]. LOOM specifies features for descriptions (which correspond to restrictions on variables) and a description reasoner, called a *classifier*. The classifier tries to determine the location of descriptions in a description hierarchy in an ongoing and distributed manner. Classification can use subsumption involving deduction to determine if one concept subsumes another. This corresponds to what we are calling derived subsumption. LOOM's classifier is highly tuned to produce derivations in support of subsumption reasoning that have relatively high utility in certain applications (natural language processing being one of them). While this accords well with our goal of natural form, systems like LOOM lack expressive power. The range of assertional knowledge that can be used is limited to assertions about subset-superset and disjointness relationships. Structure sharing is not direct, two sets (corresponding to the extensions of two descriptions) are the same by virtue of their elements. Additionally, branching quantifiers cannot be represented at all. However, because the subsumption mechanism is geared to natural language it may perform better than a general-purpose reasoner, such as the work here proposes.

The Ontic system [McAllester, 1989] also provides the ability to define structured variables using a combination of type expressions and functions that reify these types into sets. Ontic is a system for verifying mathematical arguments, and, as such, the selection of type expressions and functions is limited to the mathematical domain. Additionally, Ontic is first-order and set-theoretic with quantification over terms (which may be variables of complex type). In principle, one could represent natural language in Ontic; however, the type system would have to be enriched, and it would still suffer from the disadvantages outlined for KRYPTON. In later work, McAllester has addressed natural language issues explicitly [Givan *et al.*, 1991; McAllester and Givan, 1992]. They argue that the syntactic form of language may be a source of inferentially powerful syntax, corresponding to our notion of the natural form constraint.

In this later work, McAllester and Givan argue for the utility of nonstandard logical syntax as an aid to automated inference. They make two claims that are wholly compatible with this dissertation:

- The efficiency of inference is sensitive to the syntax used to express statements.
- Natural language is a source of inferentially powerful syntax.

They use a Montagovian syntax, based on aspects of natural language syntax under a compositional semantics. Using class expressions (that denote sets) such as "brother-of" leading to more complex class expressions (by composition) such as "(brother-of a-person)," which denotes the set of all individuals that are brothers of some person. Any monadic predicate symbol of classical syntax can function as a class symbol. For any binary relation R and class expression s, one can construct the class expression (Rs), which denotes the set of individuals related under R to an element of s. They incorporate a notion of existential and universal quantifiers as in (every s) and (some s). E.g,

(loves (some person))
(loves (every person))

which are actually what they term "quantifier-free combinators" or sets. They then provide inference rules corresponding to commonsense rules. For example one inference rule is: (every C (union C W)) where C and W are class expressions, which means that every member of a set is a member of a superset of that set. What is interesting is how much mileage they manage to get out of such rules. Real quantification is needed and is done by lambda abstraction. This leads to much the same problems as in the KRYPTON system where the *donkey* in the donkey sentence (which would be in a lambda abstract) cannot be used in the sentence representation. When they restrict their attention to a quantifier-free fragment of their language they prove satisfiability is polynomial time decidable. While an interesting result, the fragment is not particularly useful for real NLP. However, this work is of interest because they are arguing for the natural form constraint and present good evidence for the power of natural language syntax in aiding inference, if only by providing lots of commonsense rules of inference. Finally, their system can represent complex descriptions, but only extensionally as sets.

3.4 Summary

A considerable body of work has been devoted to the representation of complex descriptions in the representation of knowledge. However, none of this work addresses the same collection of issues that this dissertation addresses or does so in the same way. I propose to use a conceptually complete variable representation incorporating quantifier structures as part of the variable meaning.

Chapter 4

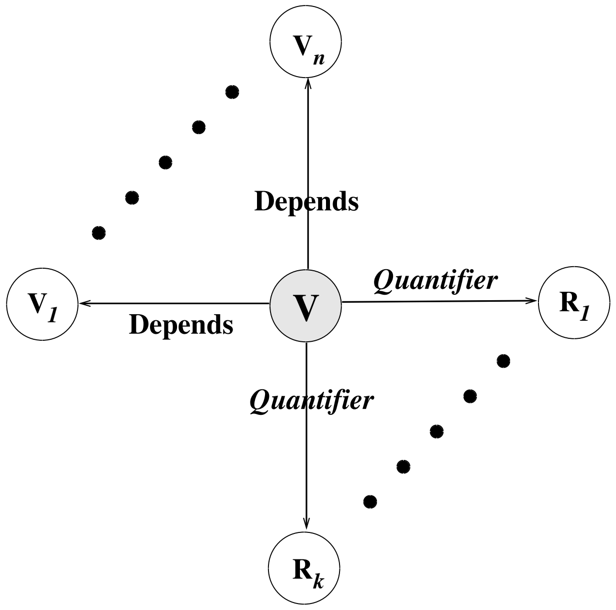
Structured Variables

We are attempting to represent a variable as a "bundle" of constraints and a binding structure (quantifier). We term these bundles "structured variables" because variables, in this scheme, are non-atomic terms. The implemented language of representation is a semantic network representation system called ANALOG (A NAtural LOGic), which is a descendant of SNePS [Shapiro, 1979; Shapiro and Rapaport, 1987; Shapiro and Rapaport, 1992].

SNePS is a knowledge representation and reasoning system that is based on a semantic network paradigm. It is *semantic* in that it represents concepts in the mind of a cognitive agent who is capable of using language. It is a *network* because the representation is a graph composed of nodes and labeled directed arcs. These graphs are constrained in the form they may take in the following ways: 1) each node represents a unique concept; 2) each concept represented in the network is represented by a node; 3) each concept represented in the network is represented by a unique node; 4) arcs represent non-conceptual binary relations between nodes; 5) the knowledge represented about each concept is represented by the structure of the entire network connected to the node representing the concept [Maida and Shapiro, 1982]. Propositional information is represented by nodes and not arcs, so SNePS is a propositional semantic network. Since the formalism associated with ANALOG builds on SNePS, we make use of some of these same ideas. The major divergence is in the representation of variables; in SNePS, variables are atomic; in

ANALOG, variables are structured. In Chapter 5 we formally specify the similarities and differences. In this chapter, we present our representation of variables and illustrate its utility more informally.

SYNTAX:
If $V, V_1, \ldots, V_n$ $(n \geq 0)$ are distinct variable nodes, and if $R_1, \ldots, R_k$ $(k \geq 0)$ are proposition nodes, and all the R_i dominate[a] V, then



is a network (actually a class of networks) and V is a structured variable node.

SEMANTICS:
$[\![V]\!]$[b] is an arbitrary *quantifier*-constrained (any- or some-constrained) individual, dependent on $[\![V_1]\!], \ldots, [\![V_n]\!]$, such that all the restriction propositions $[\![R_1]\!], \ldots, [\![R_k]\!]$ hold for that arbitrary individual.

[a] One node dominates another if there is a path of directed arcs from the first node to the second node.
[b] $[\![V]\!]$ is the intensional individual denoted by V.

Figure 4.1: Case Frame for Structured Variables

Figure 4.1 gives a case frame for the representation of variables in ANALOG. The shaded node labelled `V` is the structured variable. The restrictions on the variable are expressed by nodes $R_1, \ldots, R_k$. Scoping of existential structured variables (with respect to universal structured variables) is expressed by the **depends** arcs to universal structured variable nodes $\mathtt{V}_1, \ldots, \mathtt{V}_n$. When $n = k = 0$ the structured variable corresponds directly to a variable in standard FOPL. An example of a structured variable (the node labelled `V1`) is given in Figure 4.2 (described below). The semantics of structured variables is an augmented (by the addition of arbitrary individuals) semantic theory based on [Fine, 1983; Fine, 1985a; Fine, 1985b; Shapiro and Rapaport, 1987]. The representations of sentences, shown as examples in this dissertation, are based on the case frames specified by [Shapiro and Rapaport, 1987] augmented with structured variables.

It is possible to express these representations in a linear notation (and we do so in Chapter 5). However, the resulting representations are not particularly clear. The pictorial networks used in this chapter are clearer and easier to understand. Where necessary, explanation of the networks will be provided, but we defer a full, formal, specification of nodes and case frames to Chapter 5. The intent of this chapter is to motivate this representation before a complete specification.

4.1 Expressibility of the Representation

Previously, we outlined some of the criteria that a logic based on natural language should satisfy. Additionally, we pointed out some natural language constructs that were not directly expressible in FOPL-based representation language. We formulated the following goals for a natural logic:

1. The logic should resemble natural language as much as possible.
2. The logic should use conceptually complete variables (no open sentences are possible) and all terms (nodes) should be meaningful independently of the sentences they may appear in.
3. There should be representations for quantifier scoping, including nonlinear scopings such as those of branched quantifier or donkey sentences.

We deal with all of these goals by introducing a logic with structured variables.

4.2 Natural Form

We suggest that the representation of numerous types of quantifying expressions, using structured variables, is more "natural" than typical logics, because the mapping of natural language sentences is direct. We give an example in Figure 4.2. The shaded node `M1!` corresponds to the asserted proposition: *all men are mortal.* We shall shade nodes corresponding to the propositions that represent the example sentences. The "!" associated with some nodes (propositions) indicates that they are believed to be true. Node `V1` is the structured variable corresponding to the arbitrary man. Finally, node `M2` is the restriction proposition that the arbitrary man is a member of the class of men. The `member-class` case frame is the representation for the proposition that an object is a member of a class. This representation is more natural than the FOPL representation in that the top-level proposition is one of class membership, rather than a rule-like conditional proposition.

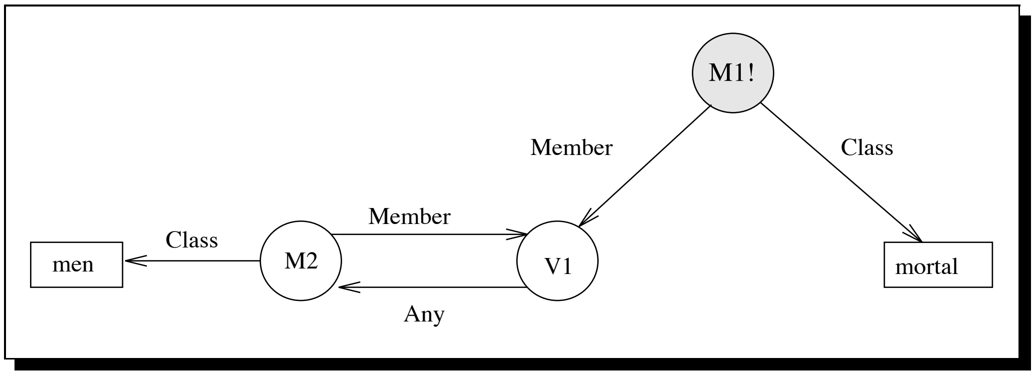


Figure 4.2: Structured Variable Representation of: *All men are mortal.*

A major advantage of adhering to the natural form constraint is the uniformity of the resulting representations. In the example involving class membership, any member-class proposition of the form "X is Y" would be represented as a similar network structure, no matter how complex the X and Y descriptions. For example, the representation of *All rich young men that own a car are mortal* is given in Figure 6.9 in Chapter 6 is similar to that of Figure 4.2. This is particularly useful for defining subsumption between propositions of the same sort (e.g., class membership), defined in Chapter 5.

The representation of structured variables suggested here can represent first-order quantifying expressions directly. Also, we can represent other quantifying expressions (for example, those involving "many" and "most") directly (although their semantics are more difficult and we do not address this issue, here). In general, there is a direct mapping from natural language quantifying expressions into structured variable representations, since structured variables correspond directly to noun phrases with restrictive relative clause complements. The natural language processing subsystem of ANALOG is based on an interpreter for generalized augmented transition network (GATN) grammars [Woods, 1970; Shapiro, 1982a]. These are networks of nodes and arcs, similar to finite-state machines, in which arcs are labeled by parts of speech or other GATN network names. Transitions from state to state occur when the correct parts of speech are matched to words in the text to be parsed. Each transition consumes the matched portion of the input. Because these networks are non-deterministic all correct parses are found.

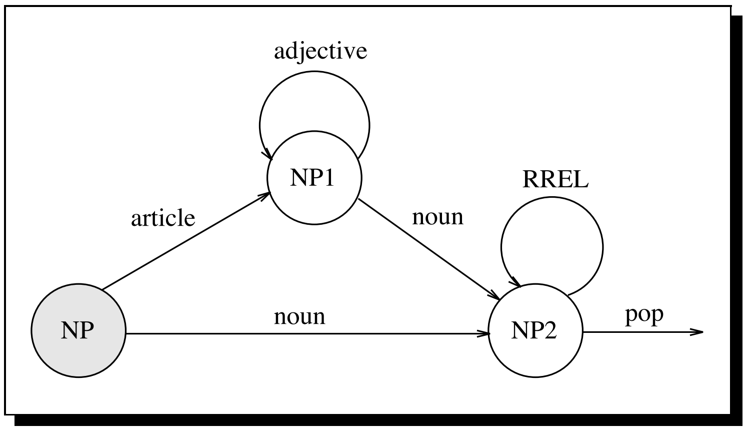


Figure 4.3: GATN NP subnetwork for noun phrases corresponding to structured variables.

Figure 4.3 shows a fragment of the GATN grammar that processes noun phrases corresponding to structured variables. So in Figure 4.3, if the text to be processed were: `the happy man` the sequence of transitions would be NP, NP1, NP1, NP2. The arc labeled "pop" out of state NP2 corresponds to a successful parse. The NP subnetwork processes surface noun phrases and builds the appropriate representation by processing articles, adjectives, and restrictive relative clauses. Because all restrictions on a surface noun phrase correspond directly to a structured variable, the restrictions can be locally collected (in the portion of the grammar that parses noun phrases) and the structured variable built.

This includes restrictive relative clauses, which are parsed in the RREL subnetwork (not shown here). By contrast, to build a FOPL-based representation, restrictions on noun phrases can be collected but must be incorporated into the representation of the entire sentence, typically an antecedent-consequent rule. This is typically not at the same stage of processing as the noun phrase processing. The localization of constraints on variables is also useful when generating natural language descriptions (done by the same GATN grammar that parses the natural language text) from structured variables, for similar reasons.

4.3 Conceptual Completeness

In typical logics, terms in one sentence are not referenced in other sentences. In general, re-using a term involves re-writing the term in the new formula. Ideally, we would like to re-use exactly the same terms in different sentences (in much the same way that language re-uses noun phrases), and we would want this re-use to result in closed sentences. This requires that variables (typically corresponding to noun phrases, or anaphora) in such terms be meaningful, independent of the sentence(s) that use or define them. We call such variables *conceptually complete*, and they may be shared by multiple sentences. This is an issue in the representation of multisentential dialog, where intersentential reference to sententially scoped objects frequently occurs. For our previous example, *All cars come with a spare tire. It is not a full-sized tire*, can only be represented, in standard logics, by re-writing the terms corresponding to the shared variables in all representations of sentences that use them or combining several sentences into one representation. To see the difficulty, consider a representation of the example:

$$\forall x[\text{car}(x) \Rightarrow \exists y(\text{spare-tire}(y) \wedge \text{comes-with}(y,x))]$$

$$\forall z\ \exists w[(\text{comes-with}(w,z) \wedge \text{car}(z) \wedge \text{spare-tire}(w)) \Rightarrow \neg\text{full-sized}(w)]$$

In these cases, there is a clear advantage to a conceptually complete (closed) variable representation as well as the structure sharing associated with a semantic network representation. Minimally, the improvement is a notational saving. We would like the representation of the example to be:

Any car, x, comes with a spare tire y

y is not full-sized

using two distinct sentences; however, in the second sentence, y is a free variable. We want two distinct sentence representations because of the natural form constraint and because FOPL-based representations that combine the representation of several natural language sentences into a single representation are unwieldy and do not permit re-articulation of the original sentences. For example, we could continue the spare tire example with a lengthy diatribe about the spare tires that all cars come with.

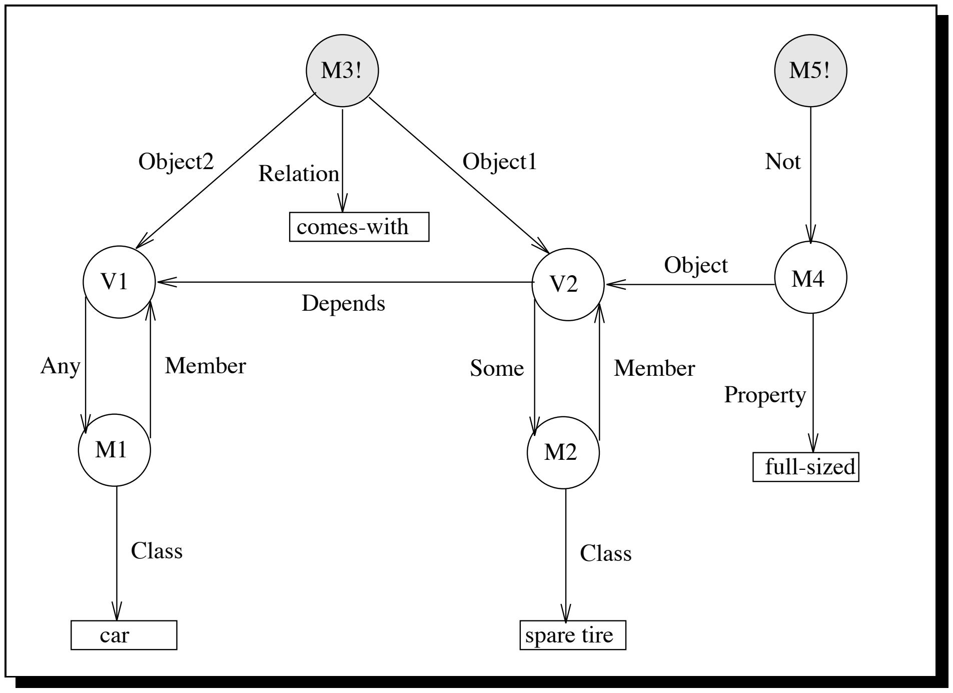


Figure 4.4: Representation of: *All cars come with a spare tire. It is not a full-sized tire.*

Figure 4.4 shows the representation of the spare tire sentences. Note that there are two proposition nodes (the shaded nodes) that correspond to the two sentences. The `Relation-Object1-Object2` case frame represents a proposition about a relation between two objects (here, the "comes-with" relation), the `Object-Property` case frame represents the proposition that an object has a property (being full-sized), and the `Not` case frame represents the negation of a proposition (in this case, `M4` that the spare tire is full-sized). Node `M3` represents the proposition that *All cars have a spare tire*, and node `M5` the proposition that *It is not full-sized.* Nodes `V1` and `V2` are structured variables corresponding to the arbitrary car and an arbitrary spare tire that depends on the car, respectively. The scope dependency between them is expressed by the *depends* arc between them. The restriction propositions are `M1` and `M2` which specify the variable type (car and spare tire, respectively). With structured variables that contain their own binding structures (quantifiers), no open sentences are possible. Thus, in the example, the variable y (node `V2`) is not free. Further, with structure sharing, distinct sentences may share constituents which would have been open sentences in a logical representation. Note that the representation takes advantage of this.

4.4 Quantifier Scoping

Any representation must account for the expression of quantifier scoping, at least as simply as the first-order predicate logic linear notation. The linear notation implicitly expresses a partial ordering of quantifiers and terms, by the order and bracketing of the quantifiers. With structured variables, quantifier scoping is expressed explicitly by the presence or absence of dependency arcs.

Figure 4.5 shows the representation of the most natural scope reading of the example sentence, *Every boy loves a girl that owns a dog.* The particular representation corresponds to the scope reading: $\forall b \exists g \exists d$ (where b, g, and d are the variables corresponding to the boy, girl, and dog, respectively). Note that while the dog actually depends on the girl, by transitivity and the syntactic constraints on structured variables, the depends arc is to the boy. For brevity, rather than redisplay the same representation with different `depends` arcs, we chose to highlight the dependency arcs (dotted lines) and indicate which arc's presence or absence corresponds to which scope reading. If the depends arc between nodes `V1` and `V2` is not present, then the scoping is $\exists g \forall b \exists d$. If the depends arc between nodes `V1` and `V3` is not present, then the scoping is $\exists d \forall b \exists g$. Finally, if neither of the depends arcs

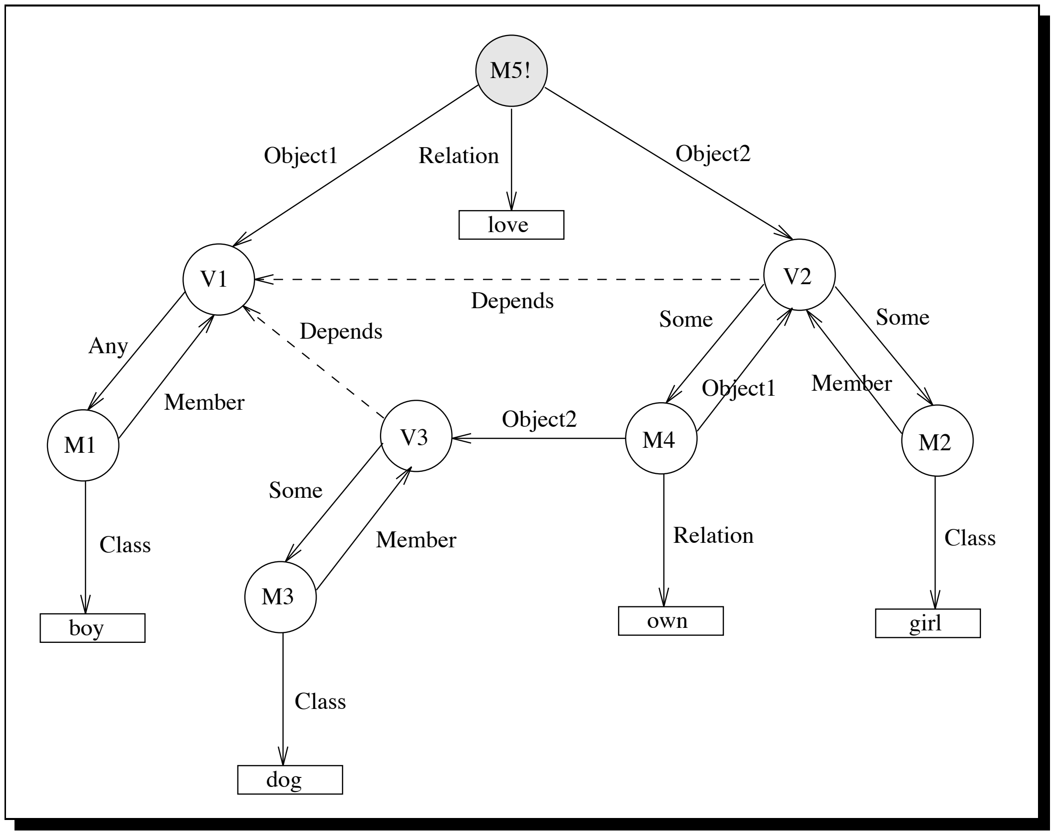


Figure 4.5: One Scoping for: *Every boy loves a girl that owns a dog.*

is present, then the scoping is $\exists g \exists d \forall b$. Note that the top-level proposition is one about "loving" for all representations, as desired by the natural form constraint.

The nonlinearity of the representation introduces a new problem in specifying limited quantifier scopings normally expressed by bracketing. For example, the differences in the sentences $\forall x\ \forall y\ (P(x,y) \Rightarrow Q(x))$ and $\forall x\ ((\forall y\ P(x,y)) \Rightarrow Q(x))$ are associated with bracketing and cannot be expressed (directly) in the nonlinear representation suggested here. The first sentence can be represented using structured variables; the second requires augmenting the representation, as its antecedent states a property of the collection of all ys. This should not be seen as a major shortcoming of this representation as several solutions to the problem are possible. Partitioned semantic networks [Hendrix, 1977; Hendrix, 1979] or a representation for collections [Cho, 1992; Franconi, 1993] would allow the representation of all possible quantifier scopings. This issue is discussed in more detail in Chapter 5.

4.4.1 Branching Quantifiers

The dependency arcs associated with structured variables form a partial ordering of variables in a formula. That is, any tree-like ordering of quantifier prefixes may be represented. Thus, we may express any sort of dependency, including those of the type associated with branching quantifiers.

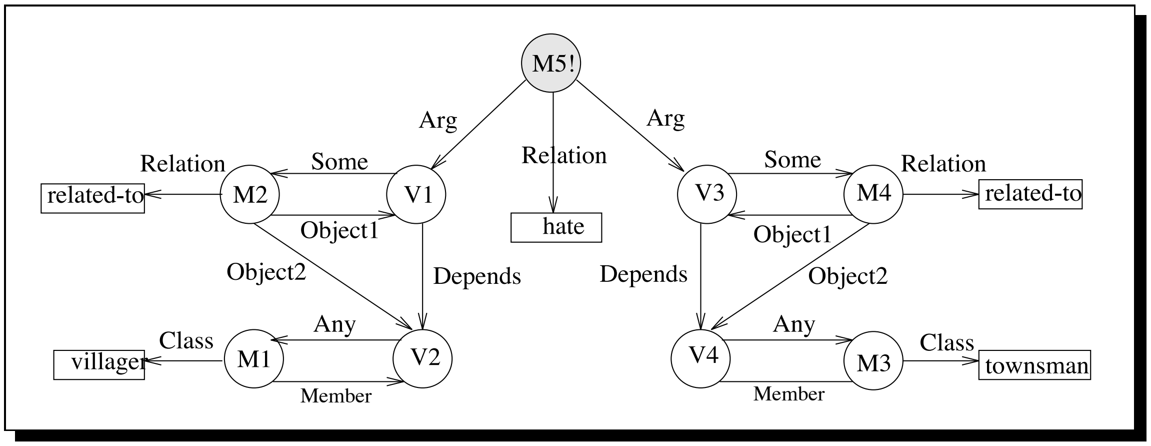


Figure 4.6: **Representation for the Branched Quantifier Sentence:** *Some relative of each villager and some relative of each townsman hate each other.*

For the villager and townsman example, Figure 4.6 illustrates how this is done. Nodes `V1` and `V3` represent the existential relatives that are scope dependent on nodes `V2` (each villager) and `V4` (each townsman), respectively. The shaded node `M5` corresponds to the proposition the nodes connected by the arcs labeled `Arg` have the relation of hating between them. As a technical point, we note that the two `related-to` nodes are actually the same single node, but for ease of drawing are drawn separately.

4.4.2 Donkey Sentences

Since all variables are conceptually complete, there are no open subformulas, and it is possible for formulas to reference constituent terms anywhere. This includes references that would result in an unscoped variable in a FOPL representation, such as the donkey sentence. Figure 4.7 shows the representation for *Every farmer that owns a donkey beats it.* This representation corresponds to the farmer beating at least one of his donkeys (the ∀∃ reading), but not necessarily all of them (the ∀∀ reading).

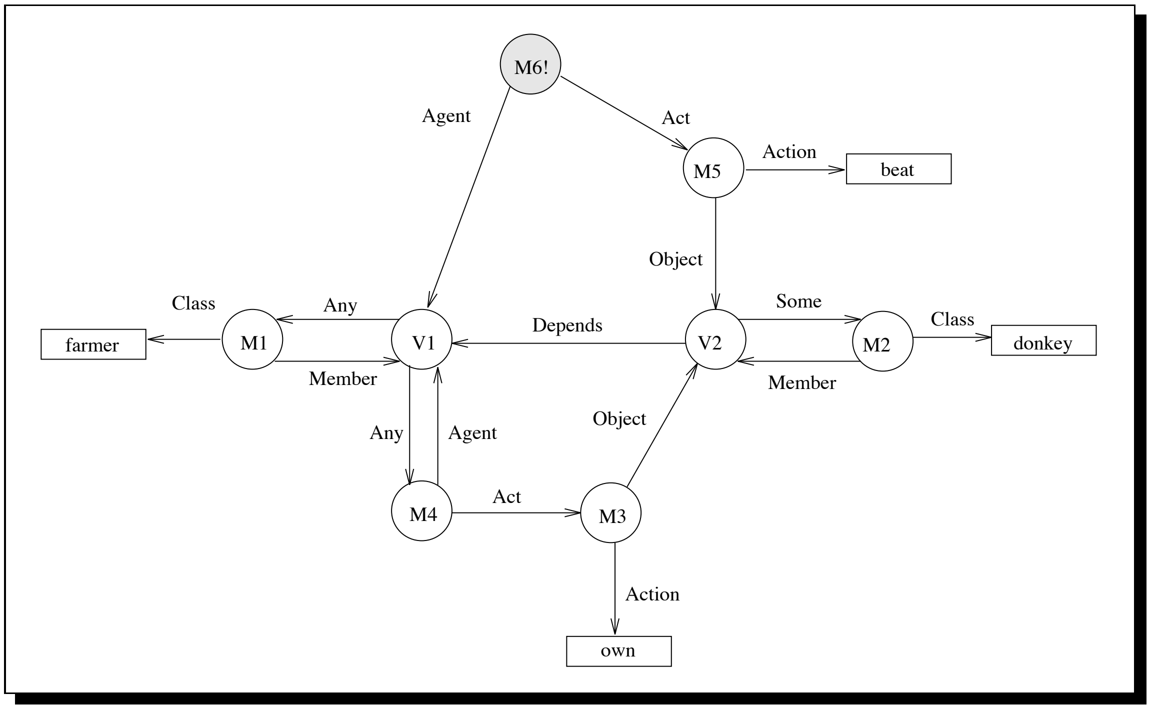


Figure 4.7: Representation for the Donkey Sentence: *Every farmer that owns a donkey beats it.*

In Figure 4.7 node `M6` represents the donkey sentence proposition. The `Agent-Act` case frame represents the proposition that an agent performs some act. Nodes `V1` and `V2` are the structured variables corresponding to *any farmer that owns a donkey* and *some donkey that is owned by a farmer.* Because of conceptual completeness of terms and the nonlinearity of the representation, it is possible for the representation of the main clause (the "beating" clause) to refer to the donkey that is owned in the restrictive relative clause of the farmer noun phrase.

4.5 Summary

The intent of this chapter was to motivate and present a conceptually complete representation for variables. To that end, we reiterated the goals for our logic, namely the natural form constraint, conceptual completeness of terms, and representation of quantifier scope including nonlinear scopings. We provided an initial representation that accords variables structure. We informally motivated its utility in addressing these goals and presented sample representations for the difficult sentences discussed in Chapter 2. It is, clearly, insufficient to just show representations without indication as to how they are to be used. The next chapter formalizes this informal presentation of the representation and specifies how these representations can be used in reasoning.

Chapter 5

The Knowledge Representation Formalism

5.1 Syntax and Semantics of the Formalism

In this chapter, we provide a syntax and semantics of a logic whose variables are not atomic and have structure. We call these variables *structured variables* and we have, informally, presented them in Chapter 4. The syntax and semantics of the logic are specified by a complete definition of a propositional semantic network representation formalism. This formalism is an augmentation of [Shapiro, 1991] which defines propositional SNePS, a semantic network KRR formalism without variables. Our formalism, defined here, extends the work of Shapiro to deal with quantified sentences and variables corresponding to an extended first-order logic (it is not strictly first-order as quantification over propositions is possible). By a propositional semantic network, we mean that all information, including propositions, "facts", etc., is represented by nodes in a network. The implemented system is called ANALOG. The name ANALOG is also used to refer to the logical system independent of its implementation.

We note that portions of this chapter have previously been published in [Ali, 1993c; Ali and Shapiro, 1993] in slightly different forms.

5.1.1 Semantics

Any theory of semantics that ascribes propositional meaning to nodes can be the semantics (augmented with the semantics for structured variables) of ANALOG. In this dissertation, examples and representations are used that follow the case frame semantics of [Shapiro and Rapaport, 1987] which provides a collection of propositional case frames and their associated semantics based on an extended first-order predicate logic. We augment that logic further with arbitrary individuals (for the semantics of structured variables) as defined in the semantic theory of [Fine, 1983; Fine, 1985a; Fine, 1985b]. We provide the semantics for nodes and case frames as needed.

5.1.2 The Domain of Interpretation

ANALOG nodes are terms of a formal language. The interpretation of a node, called an entity, is an object in the domain of interpretation. Every ANALOG node denotes an entity, and if n is an ANALOG node, then $[\![n]\!]$ denotes the entity represented by n. It is useful for discussing the semantics of ANALOG networks to present them in terms of an "agent" who has beliefs and performs actions. This agent is a computational model of a cognitive agent [Shapiro and Rapaport, 1991]. ANALOG relations are labelled directed arcs that connect nodes in the network. Particular collections of ANALOG nodes and relations correspond to case frames which express propositions. Nodes that express propositions are called *propositional nodes* and their case frame semantics is given in section 5.2.3. In the rest of this explication, the term "node" will be used for ANALOG node, and "relation" for ANALOG relation.

5.1.3 Metapredicates

To help formalize this description, we follow Shapiro who introduced the metapredicates *Conceive*, *Believe*, and $=$ [Shapiro, 1991]. If n, n_1, n_2 are metavariables ranging over nodes, and p is a metavariable ranging over proposition nodes, the semantics of these metapredicates are:

Conceive(n)	Means that the node is actually constructed in the network. *Conceive*(n) may be true without $[\![n]\!]$ being known to exist or be true or false. Also note that n need not be a proposition.
Believe(p)	Means that the agent believes the proposition $[\![p]\!]$.
$n_1 = n_2$	Means that n_1 and n_2 are the same, identical, node.

Belief implies conception, as specified in the axiom:

Axiom 1

Believe$(p) \Rightarrow$ *Conceive*(p)

In practical terms, this means that for a proposition to have a belief status it must be a node in the semantic network. This is a reasonable assumption, since, for an agent to believe something, the agent must have some conceptualization of it. Additionally, for us to do anything with a proposition it must be a node.

5.1.4 Definition of Nodes

Informally, a node consists of a set of labeled (by relations) directed arcs to one or more nodes. Additionally, a node may be labeled by a "name" (e.g., `BILL, M1, V1`) as a useful (but extra-theoretic) way to refer to the node. The naming of a proposition node is of the form `M`n, where n is some integer. A "!" is appended to the name to show that the proposition represented by the node is believed to be true (*Believe*(`M1`)). However, the "!" does not affect the identity of the node or the proposition it represents. Similarly, variable nodes are labeled `V`n, where n is some integer. Nodes that are atomic' but that do not correspond to named concepts, for example nodes associated with sensory input, are called base nodes and are named `B`n, where n is some integer (additionally, base nodes may be named for the concept they represent, e.g., `man, bill`). More formally we define a node as follows:

Axiom 2

There is a non-empty collection of labelled atomic nodes called **base nodes.** Typically, base nodes are labelled by the entity they denote. *Example:* `bill, B3` are base nodes.

Definition 1: A **wire** is an ordered pair <r, n>, where r is a relation, and n is a node. Metavariables $w, w_1, w_2, \ldots$ range over wires. *Example:* `<member, john>` is a wire.

Definition 2: A **nodeset** is a set of nodes, $\{n_1, \ldots, n_k\}$. Meta-variables $ns, ns_1, ns_2, \ldots$ range over nodesets. *Example:* {`john, bill`} is a nodeset if `john` and `bill` are nodes.

Definition 3: A **cable** is an ordered pair <r, ns>, where r is a relation, and ns is a non-empty nodeset. Meta-variables $c, c_1, c_2, \ldots$ range over cables. *Example:* `<member`, {`john, bill`}`>` is a cable.

Definition 4: A **cableset** is a non-empty set of cables, {<r_1, ns_1>, ..., <r_k, ns_k>}, such that $r_i = r_j \iff i = j$. Meta-variables $cs, cs_1, cs_2, \ldots$ range over cablesets. *Example:* {`<member`, {`john, bill`}`>`, `<class`, {`man`}`>`} is a cableset.

Definition 5: Every **node** is either a base node or a cableset. *Example:* `bill` is a base node, {`<member`, {`john, bill`}`>`, `<class`, {`man`}`>`} is a cableset. Both are nodes.

Note that at this point we have recursively defined a node as either a base node or a cableset. Pictorically, a node can be visualized as a head node connected by cable relations to the nodes in the nodesets of each cable in the cableset that defines the node. Variable nodes can now be defined in the same way. Because of their special status for inference, some additional definitions are required.

Definition 6: We overload the **membership** relation "$\in$" so that $x \in s$ holds just under the following conditions:

1. If x is a node and s is a nodeset, $x \in s \iff \exists y\ [y \in s \land \mathit{Subsume}(y, x)]$ (where y is a node).
   Example: `M1` $\in$ {`M1, M2, M3`}
   Because we need more definitions before *Subsume* can be defined, we defer its definition to Figure 5.3. Intuitively, we want to allow membership between a node and any nodeset containing a variable that subsumes the node.

2. If x is a wire such that $x =$ `<`r_1, n`>`, and s is a cable such that $s =$ `<`r_2, ns`>`, then $x \in s \iff r_1 = r_2 \land n \in ns$.
   Example: `<member`, `john>` $\in$ `<member`, {`john, bill`}`>`

3. If x is a wire and s is a cableset, then $x \in s \iff \exists c[c \in s \land x \in c]$.
   Example: `<member`, `john>` $\in$ {`<member`, {`john, bill`}`>`, `<class`, {`man`}`>`}

Definition 7: An **nrn-path** from the node n_1 to the node n_{k+1} is a sequence,

$$n_1, r_1, \ldots, n_k, r_k, n_{k+1}$$

for $k \geq 1$ where the n_i are nodes, the r_i are relations, and for each i, `<`r_i, n_{i+1}`>` is a wire in n_i. *Example:* If `M1` = {`<member`, {`john, bill`}`>`, `<class`, {`man`}`>`}, then `M1,member,john` and `M1,class,man` are some nrn-paths.

Definition 8: A node n_1 **dominates** a node n_2 just in case there is an nrn-path from n_1 to n_2. We define the predicate $\mathit{dominate}(n_1,\ n_2) \iff n_1$ dominates n_2. *Example:* If `M1` = {`<member`, {`john, bill`}`>`, `<class`, {`man`}`>`}, then `M1` dominates `john`, `bill`, and `man`.

Definition 9: A **variable** node is a cableset of the form {`<any`, ns`>`} (**universal variable**

node) or {`<some`, ns_1`>`, `<depends`, ns_2`>`} (**existential variable node**). Meta-variables $v, v_1, \ldots$ range over variables. The choice of naming the relation `any` rather than `all` was made, because this choice reflects the intended interpretation of these variables as arbitrary individuals. A variable node is further restricted in the form it may take in the following ways:

1. If it has the form {`<any`, ns`>`}, then every $n \in ns$ *must* dominate it.
2. If it has the form {`<some`, ns_1`>`, `<depends`, ns_2`>`}, then every $n \in ns_1$ *must* dominate it *and* every $n \in ns_2$ *must* be a universal variable node.
3. Nothing else is a variable node.

Example: `V1` = {`<any`, {{`<member, `{`V1`}`>, <class, `{`man`}`>`}}`>`} is the variable node corresponding to *every man*. The variable label `V1` is just a convenient extra-theoretic method of referring to the variable.

We define two selectors for variable nodes:

$$rest(v) = \begin{cases} ns & \text{if } v = \{<any, ns>\} \\ ns_1 & \text{if } v = \{<some, ns_1>, <depends, ns_2>\} \end{cases}$$

$$depends(v) = \; ns_2 \quad \text{if } v = \{<some, ns_1>, <depends, ns_2>\}$$

Informally, *rest*(v) is the set of restriction propositions on the types of things that may be bound to the variable node v. *Depend*(v) is the set of universal variable nodes on which an existential variable node, v, is scope-dependent.

Definition 10: A **molecular** node is a cableset that is *not* a variable node. *Example:* {`<member`, {`john, bill`}`>`, `<class`, {`man`}`>`} is a molecular node, since it is a cableset but not a variable node.

Definition 11: A **rule** node is a molecular node that dominates a variable node that does not, in turn, dominate it.

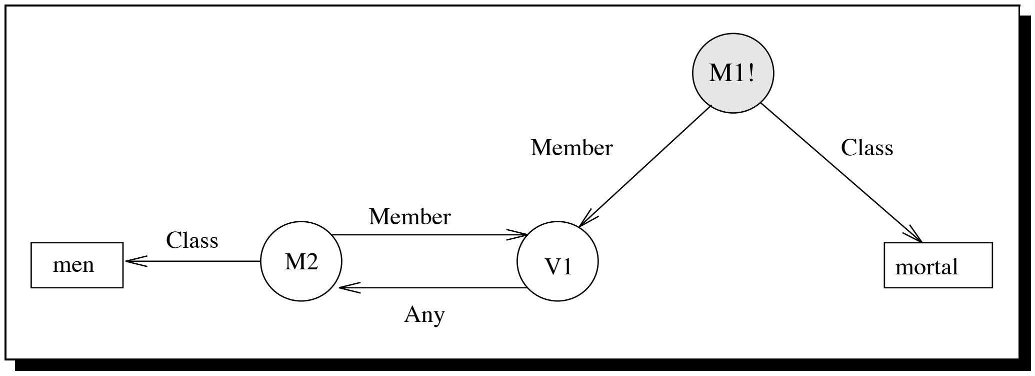


Figure 5.1: Rule for: *All men are mortal.*

Example: Given the labelled nodes below:

`V1` = {`<any`, {`M1`}`>`}

`M1` = {`<member`, {`V1`}`>`,`<class`, {`man`}`>`}

`M2` = {`<member`, {`V1`}`>`, `<class`, {`mortal`}`>`}

corresponding to the diagram in Figure 5.1, `M2` is a rule node since `M2` dominates `V1`, which does not, in turn, dominate `M2`. `M1` is *not* a rule node because, while it dominates `V1`, it is also dominated by `V1`. The non-rule nodes that dominate variable nodes correspond to restrictions on possible bindings of those same variable nodes.

At this point we have defined the structure of all the required kinds of nodes. To use them meaningfully, we define the means of expressing propositional knowledge, the case frame.

Definition 12: A **case frame** is a set of pairs `<`r, i`>`, where r is a relation, and i is an integer ≥ 0.

Example: {`<member`, `1>`, `<class`, `1>`} is a caseframe (in particular, the one used for expressing class membership).

Case frames are used to represent propositional knowledge. Informally, the first component of each pair specifies relation arcs that are associated with any instance of this case frame and the second component is the minimum number of such arcs in any instance of this caseframe. Other semantic baggage associated with case frames is described in Section 5.2.3.

Definition 13: We define the **caseframeset** (and function *csfset*) of a cableset *cs* as follows:

$$\mathit{csfset}(\{\texttt{<}r_1, ns_1\texttt{>}, \ldots, \texttt{<}r_k, ns_k\texttt{>}\}) = \{\texttt{<}r_1, |ns_1|\texttt{>}, \ldots, \texttt{<}r_k, |ns_k|\texttt{>}\}$$

The next two definitions are required primarily for node matching and inference, however it is easiest to define them now.

Definition 14: A **wireset** of a node n is defined as $\{w | w \in n\}$.

Example: If `M2 =` $\{$`<member,` $\{$`john,bill`$\}$`>`, `<class,` $\{$`man`$\}$`>`$\}$ then:

wireset(`M2`) $=$ $\{$`<member, john>,<member, bill>,<class, man>`$\}$.

Definition 15: The **variableset** of a node n is defined as $\{v \mid \mathit{dominate}(n, v)\}$, where v ranges over variable nodes.

Example: If `M2` $=$ $\{$`<member,` $\{$`V1`$\}$`>`, `<class,` $\{$`mortal`$\}$`>`$\}$ then:

variableset(`M2`) $=$ $\{$`V1`$\}$.

5.1.5 The ANALOG model

Definition 16: An **ANALOG model** is a tuple $(A, B, M, R, U, E, \Gamma)$, where A is a set of relations, B is a set of base nodes, M is a set of non-rule molecular nodes, R is a set of rule nodes, U is a set of universal variable nodes, E is a set of existential variable nodes, and $\Gamma \subseteq M \cup R$. B, M, R, U, and, E are disjoint. Γ consists of believed propositions. Note that the metapredicates *Believe* and *Conceive* are, with respect to a particular model:

$$\begin{aligned}&\mathit{Believe}(n) \equiv n \in \Gamma\\&\mathit{Conceive}(n) \equiv n \in M \cup R.\end{aligned}$$

However, we will continue to use the metapredicates, as they are more perspicuous.

5.1.6 Reduction

We follow [Shapiro, 1986; Shapiro, 1991] in arguing for a form of reduction inference as being useful and natural, relative to language. This is a form of structural subsumption [Woods, 1991], peculiar to semantic network formalisms, which allows a proposition (say n_1) to "reduce" to (logically imply) a proposition (say n_2) whose wires are a subset of the wires of the original proposition. This is written $\mathit{Reduce}(n_2, n_1)$. Figure 5.2 gives an example of a proposition expressing a brotherhood relation among a group of men. Node `M1` represents the proposition that `bill, john, ted,` and `joe` are brothers. By reduction subsumption, all proposition nodes (such as `M2` and `M3`) involving fewer brothers follow.

However, we must restrict the use of reduction inference to precisely those propositions and rules which are reducible through the use of the *IsReducible* metapredicate.

Axiom 3

$$\mathit{Reduce}(cs_1, cs_2) \iff \begin{cases} (\forall w[w \in cs_2 \Rightarrow w \in cs_1] \land \mathit{IsReducible}(\mathit{csfset}(cs_1), \mathit{csfset}(cs_2))) \\ \lor \\ (\forall w[w \in cs_1 \Rightarrow w \in cs_2] \land \mathit{IsExpandible}(\mathit{csfset}(cs_1), \mathit{csfset}(cs_2))) \end{cases}$$

Note that the semantics of the metapredicates *IsReducible* and *IsExpandible* will be specified in terms of the particular case frames used in a representation language. Some sample case frames are given in Section 5.2.3. Propositions like `M1` are clearly reducible, but not all propositional case frames are reducible. For example,

$$\forall x((\text{man}(x) \land \text{rich}(x)) \Rightarrow \text{happy}(x)) \tag{5.1}$$

In the following model (A, B, M, R, U, E, Γ):

A ={`relation, arg`}
B ={`bill, john, ted, joe`}
M ={`M1, M2, M3`}
Γ ={`M1, M2, M3`}

where:

```
M1 = {<relation, {brothers}>, <arg, {bill, john, ted, joe}>}
M2 = {<relation, {brothers}>, <arg, {john, ted, joe}>}
M3 = {<relation, {brothers}>, <arg, {bill, john}>}
```

Some reductions:

```
Reduce(M2, M1) = T
Reduce(M3, M1) = T
```

Figure 5.2: Example of Subsumption by Reduction for a Particular Model

should not allow the reduced proposition:

$$\forall x(\text{man}(x) \Rightarrow \text{happy}(x)) \tag{5.2}$$

which involves fewer constraints on x than proposition 5.1, as the latter does not follow from the former. New propositions derived by reduction should be implied by, and more or less restrictive (depending on the particular case frame) than, the propositions from which they are derived. For example, note that reduction to Proposition 5.2 is appropriate when the constraints in the antecedent of the rule are disjunctive as in:

$$\forall x((\text{man}(x) \lor \text{rich}(x)) \Rightarrow \text{happy}(x)) \tag{5.3}$$

Also note that *IsExpandible* is valid for Proposition 5.1 when expanding the antecedent set. For example:

$$\forall x((\text{man}(x) \land \text{rich}(x) \land \text{young}(x)) \Rightarrow \text{happy}(x)) \tag{5.4}$$

follows from Proposition 5.1. *IsReducible* and *IsExpandible* should be defined appropriately for the particular representation language to allow (or disallow) these reductions. In Section 5.2.3, we specify the reducible case frames we use in this dissertation.

A proposition that is a reducible reduction of a believed proposition is also a believed proposition. Since nodes are, by definition, cablesets, we state this as an axiom.

Axiom 4
$(\mathit{Reduce}(n_1, n_2) \wedge \mathit{Believe}(n_1)) \Rightarrow \mathit{Believe}(n_2)$

So, for example, in Figure 5.2, if node `M1` is believed, then since nodes `M2` and `M3` are reductions of `M1`, both `M2` and `M3`, once conceived, would be believed.

5.1.7 Types of Nodes

We have defined four types of nodes: base, molecular, rule, and variable nodes. Informally, base nodes correspond to individual constants in a standard predicate logic, molecular nodes to sentences and functional terms, rule nodes to closed sentences with variables, and variable nodes to variables. Note that syntactically all are terms in ANALOG, however.

5.2 Semantic Issues

ANALOG is a propositional semantic network. By this is meant that all information, including propositions, "facts", etc., is represented by nodes. Labelled arcs are purely structural and carry no assertional import. The benefit of representing propositions by nodes is that propositions about other propositions may be represented. This means that ANALOG is not strictly first-order, as variables may quantify over nodes that correspond to propositions rather than individuals. This has useful consequences when processing natural language, particularly in dealing with questions whose answers are propositions rather than individuals. We note that ANALOG is an extension of [Shapiro, 1991] which defines propositional SNePS. The ANALOG formalism extends the work to deal with quantified sentences and variables corresponding to an extended first-order logic. The intensional representation, uniqueness principle, and some of the case frame semantics are described more completely in [Shapiro and Rapaport, 1987; Rapaport, 1991].

5.2.1 Intensional Representation

For the purposes of modeling a cognitive agent's beliefs and concepts it is necessary to be able to represent intensional rather than extensional entities.

An extensional entity's identity does not depend on the representation used, because it represents an object in the world. So the names *morning star* and *evening star* are distinct names, but extensionally they are equivalent representations of the planet Venus, and may be used interchangeably.

ANALOG nodes correspond to objects of thought which may be non-existent, theoretical, or even impossible (e. g., a square circle). This requires that intensional entities be represented. What is being represented, in ANALOG, is an agent's mind. Objects in that mind need not represent any extensional object in the world. To connect objects of mind to extensional objects in the world, ANALOG uses a case frame with a `lex` arc from the intensional object to its extension. Additionally, sensory nodes can also make this connection to the external world [Shapiro and Rapaport, 1987; Rapaport, 1988].

5.2.2 The Uniqueness Principle

No two nodes in the network represent the same individual, proposition, or rule.

Axiom 5
$n_1 = n_2 \iff [\![n_1]\!] = [\![n_2]\!]$

This is a consequence of the intensional semantics, since nodes correspond to objects of thought in an agent's mind. The objects of thought are intensional: a mind can have two or more objects of thought that correspond to only one extensional object, or no extensional object. The classic example of this is the Morning Star and the Evening Star, which might be distinct objects of thought but have a single extensional referent. Thus, the nodes representing the Morning Star and the Evening Star are distinct, and any node can denote only *one* intensional object, which no other node can denote.

Variables pose an interesting problem for the uniqueness principle in that, on the surface, it appears that there can only, ever, be one variable of a particular kind. So a sentence like:

Every elephant hates every other elephant

which, necessarily, involves two distinct elephants could, apparently, not be represented. This difficulty is more apparent than real, as the natural language itself makes clear the distinction between the two variables. The first variable corresponds to *any elephant*, the second to *any other elephant*, where the latter variable has the additional restriction that it is distinct from the first. As a consequence, the uniqueness principle is not violated.

A benefit of the uniqueness principle is a high degree of structure-sharing in large networks. Additionally, the network representation of some types of sentences (such as the donkey sentence) will reflect the re-use of natural language terms expressed by pronouns and other reduced forms. This has additional advantages for inference as described in Section 5.7.2, allowing a form of bi-directional inference [Shapiro *et al.*, 1982].

5.2.3 Case Frame Semantics

ANALOG can support any propositional representations that have a consistent syntax and semantics. In this dissertation, examples of representations used will follow the syntax and semantics of [Shapiro and Rapaport, 1987], although we augment their semantics to allow for use of structured variables and subsumption. For the following case frames, we specify their syntax, semantics, and their status for subsumption.

1. $\{$`<object,`ns_1`>`, `<property,`ns_2`>`$\}$: For any $n_1 \in ns_1$ and any $n_2 \in ns_2$, $[\![n_1]\!]$ has the property $[\![n_2]\!]$. Valid reductions are specified by:

   $IsReducible(\{$`<object,` i_1`>`, `<property,` j_1`>`$\}$, $\{$`<object,` i_2`>`, `<property,` j_2`>`$\})$

where $1 \leq i_2 \leq i_1$ and $1 \leq j_2 \leq j_1$.

2. $\{$`<member,`ns_1`>`, `<class,`ns_2`>`$\}$: For any $n_1 \in ns_1$ and any $n_2 \in ns_2$, $[\![n_1]\!]$ is a member of class $[\![n_2]\!]$. Valid reductions are specified by:

   $$\mathit{IsReducible}(\{\texttt{<member}, i_1\texttt{>}, \texttt{<class}, j_1\texttt{>}\}, \{\texttt{<member}, i_2\texttt{>}, \texttt{<class}, j_2\texttt{>}\})$$

   where $1 \leq i_2 \leq i_1$ and $1 \leq j_2 \leq j_1$.

3. $\{$`<relation,`ns_1`>`, `<arg,`ns_2`>`$\}$: For every $n_1 \in ns_1$ and every $n_2, n_3 \in ns_2$, such that $n_2 \neq n_3$, $[\![n_2]\!]$ is in relation $[\![n_1]\!]$ to $[\![n_3]\!]$, and $[\![n_3]\!]$ is in relation $[\![n_1]\!]$ to $[\![n_2]\!]$. Valid reductions are specified by:

   $$\mathit{IsReducible}(\{\texttt{<relation}, i_1\texttt{>}, \texttt{<arg}, j_1\texttt{>}\}, \{\texttt{<relation}, i_2\texttt{>}, \texttt{<arg}, j_2\texttt{>}\})$$

   where $1 \leq i_2 \leq i_1$ and $2 \leq j_2 \leq j_1$.

4. $\{$`<relation,`ns_1`>`, `<object1,`ns_2`>`, `<object2,`ns_3`>`$\}$: For every $n_1 \in ns_1$ every $n_2 \in ns_2$, and every $n_3 \in ns_3$, $[\![n_2]\!]$ is in relation $[\![n_1]\!]$ to $[\![n_3]\!]$. Valid reductions are specified by:

   $$\begin{aligned}\mathit{IsReducible}(&\{\texttt{<relation}, i_1\texttt{>}, \texttt{<object1}, j_1\texttt{>}, \texttt{<object2}, k_1\texttt{>}\},\\ &\{\texttt{<relation}, i_2\texttt{>}, \texttt{<object1}, j_2\texttt{>}, \texttt{<object2}, k_2\texttt{>}\})\end{aligned}$$

   where $1 \leq i_2 \leq i_1$ and $1 \leq j_2 \leq j_1$ and $1 \leq k_2 \leq k_1$.

5. $\{$`<not,`ns_1`>`$\}$: For any $n \in ns_1$, it is not the case that $[\![n]\!]$. Note that each $n \in ns_1$ must be a propositional node. Valid reductions are specified by:

   $$\mathit{IsReducible}(\{\texttt{<not}, i_1\texttt{>}\}, \{\texttt{<not}, i_2\texttt{>}\})$$

   where $i_2 \leq i_1$.

6. $\{$`<ant,`ns_1`>`, `<cq,`ns_2`>`$\}$: For any $n_1 \in ns_1$ and any $n_2 \in ns_2$, $[\![n_1]\!]$ implies $[\![n_2]\!]$. Valid reductions are specified by:

   $$\begin{aligned}\mathit{IsReducible}(&\{\texttt{<ant}, i_1\texttt{>}, \texttt{<cq}, j_1\texttt{>}\},\\ &\{\texttt{<ant}, i_2\texttt{>}, \texttt{<cq}, j_2\texttt{>}\})\end{aligned}$$

where $1 \leq i_2 \leq i_1$ and $1 \leq j_2 \leq j_1$. Additional reductions are given by:

$$\mathit{IsExpandible}(\{\texttt{<ant}, i_1\texttt{>}, \texttt{<cq}, j_1\texttt{>}\}, \{\texttt{<ant}, i_2\texttt{>}, \texttt{<cq}, j_2\texttt{>}\})$$

where $\leq i_1 \leq i_2$ and $1 \leq j_2 \leq j_1$.

7. $\{\texttt{<\&ant}, ns_1\texttt{>}, \texttt{<cq}, ns_2\texttt{>}\}$: The conjunction $(\bigwedge_{n_i \in ns_1} [\![n_i]\!])$ implies $[\![n_2]\!]$, for any $n_2 \in ns_2$. Valid reductions are specified by:

$$\mathit{IsReducible}(\{\texttt{<\&ant}, i_1\texttt{>}, \texttt{<cq}, j_1\texttt{>}\}, \{\texttt{<\&ant}, i_1\texttt{>}, \texttt{<cq}, j_2\texttt{>}\})$$

where $1 \leq j_2 \leq j_1$. Additional reductions are given by:

$$\mathit{IsExpandible}(\{\texttt{<\&ant}, i_1\texttt{>}, \texttt{<cq}, j_1\texttt{>}\}, \{\texttt{<\&ant}, i_2\texttt{>}, \texttt{<cq}, j_2\texttt{>}\})$$

where $1 \leq i_1 \leq i_2$ and $1 \leq j_2 \leq j_1$.

Note that most of the valid reductions of these propositional case frames follow directly from their semantics. More complex networks may be built by recursive application of these case frames. There are numerous other case frames (for a complete dictionary of them, see [Shapiro *et al.*, 1993]), but we will restrict our examples, in this dissertation, to these for simplicity.

5.3 Subsumption

Semantic network formalisms provide "links" that relate more general concepts to more specific concepts; this is called a *taxonomy*. It allows information about concepts to be associated with their most general concept, and it allows information to filter down to more specific concepts in the taxonomy via inheritance. More general concepts in such a taxonomy *subsume* more specific concepts, the subsumee inheriting information from its subsumers. For atomic concepts, subsumption relations between concepts are specified by the links of the taxonomy. For non-atomic concepts, to specify subsumption, some additional definitions are required.

Definition 17: A **binding** is a pair t/v, where either t and v are both structured variables of the same type (universal or existential), or v is a universal variable node and t is any node.

Examples: `V1/V2`

`JOHN/V1`

Definition 18: A **substitution** is a (possibly empty) set of bindings, $\{t_1/v_1, \ldots, t_n/v_n\}$.

Examples: {`V1/V2, JOHN/V3`}

{`B1/V1, M1/V2`}

Definition 19: We overload the definition of **variableset** to a substitution s and define it as: $\{v \mid \exists t(t/v \in s)\}$.
Example: variableset({`V1/V2, JOHN/V3`}) = {`V2, V3`}.

Definition 20: The result of **applying** a substitution $\theta = \{t_1/v_1, \ldots, t_m/v_m\}$ to a node n is the new node (called an instance) $n\theta$ of n obtained by simultaneously replacing each of the v_i dominated by n with t_i. If $\theta = \{\}$, then $n\theta = n$.

Example: If `M1 =` {`<member`, {`V1`}`>`, `<class`, {`MAN`}`>`} then:

`M1`{`JOHN/V1`} = {`<member`, {`JOHN`}`>`, `<class`, {`MAN`}`>`}

Definition 21: Let $\theta = \{s_1/u_1, \ldots, s_n/u_n\}$ and $\rho = \{t_1/v_1, \ldots, t_m/v_m\}$ be substitutions. Then the **composition** $\theta \cdot \rho$ of θ and ρ is the substitution obtained from the set

$$\{s_1\rho/u_1, \ldots, s_n\rho/u_n, t_1/v_1, \ldots, t_m/v_m\}$$

by deleting any binding $s_i\rho/u_i$ for which $u_i = s_i\rho$.

Example: $\theta =$ {`V1/V2, V4/V3`}

$\rho =$ {`V2/V1, JOHN/V4`}

$\theta \cdot \rho =$ {`JOHN/V3, JOHN/V4`}

Definition 22: A substitution θ is **consistent** iff neither of the following hold:

$$\exists u, t, s[t/u \in \theta \wedge s/u \in \theta \wedge s \neq t]$$

$$\exists u, v, t[t/u \in \theta \wedge t/v \in \theta \wedge u \neq v]$$

We note that this is different from the standard definition of consistent for substitution. A substitution that is not consistent is termed **inconsistent**. The motivation for the second constraint (called the unique variable binding rule, UVBR) is that in natural language, users seldom want different variables in the same sentence to bind identical objects [Shapiro, 1986]. For example, *Every elephant hates every elephant* has a different interpretation from *Every elephant hates himself.* Typically, the most acceptable interpretation of the former sentence requires that it not be interpreted as the latter. UVBR requires that within an individual sentence that is a rule (has bound variables), any rule use (binding of variables) must involve different terms for each variable in the rule to be acceptable.

Examples: {`JOHN/V2, BILL/V2`} is inconsistent.
{`JOHN/V1, JOHN/V2`} is inconsistent.
{`JOHN/V1, BILL/V2`} is consistent.

Definition 23: The predicate *occurs-in*(x, y), where x is a variable, is defined as follows:

$$\textit{occurs-in}(x, y) \iff \textit{dominate}(y, x).$$

occurs-in is used to perform the standard occurs check of the unification algorithm (and is just a more perspicuous naming of *dominate*) [Lloyd, 1987]. It is needed to rule out x subsuming $f(x)$.

In ANALOG, we specify subsumption as a binary relation between arbitrary nodes in the network. We define subsumption between two nodes x and y in Figure 5.3. This definition of subsumption includes subsumption mechanisms that Woods classifies as *structural, recorded, axiomatic,* and *deduced* subsumption [Woods, 1991]. In Figure 5.3, case (1) corresponds to identical nodes (a node, obviously, subsumes itself). Case (2) is the reduction inference case discussed in section 5.1.6. Case (3) applies when a universal structured vari-

Subsume(x, y) in a model $(A,\ B,\ M,\ R,\ U,\ E,\ \Gamma)$ if any one of:

1. $x = y$.
2. *Reduce*(x, y)
3. For $x \in U$ and $y \in B \cup M \cup R$, if not *occurs-in*$(x,\ y)$ and there exists a substitution S such that

$$\forall r[r \in \mathit{rest}(x),\ \Gamma \vdash r\{y/x\} \cdot S].$$

Derivation is here denoted by "$\vdash$" and is defined in section 5.7. It corresponds, more or less directly, to standard logical derivation.

4. For $x \in U$ and $y \in U \cup E$, if

$$\forall r[r \in \mathit{rest}(x) \Rightarrow \exists s[s \in \mathit{rest}(y) \wedge \mathit{Subsume}(r, s)]]$$

5. For $x, y \in E$, if all of the following hold:

$$\forall s[s \in \mathit{rest}(y) \Rightarrow \exists r[r \in \mathit{rest}(x) \wedge \mathit{Subsume}(r, s)]]$$
$$\forall r[r \in \mathit{rest}(x) \Rightarrow \exists s[s \in \mathit{rest}(y) \wedge \mathit{Subsume}(r, s)]]$$
$$\forall d[d \in \mathit{depends}(y) \Rightarrow \exists c[c \in \mathit{depends}(x) \wedge \mathit{Subsume}(c, d)]]$$

Otherwise, fail.

Figure 5.3: Subsumption Procedure

able node subsumes another node. This corresponds to a description like *any rich woman* subsuming *Mary* if *Mary* is known to be a woman and rich. Such a variable will subsume another node if and only if every restriction on the variable can be derived (in the current model) for the node being subsumed. Subsumption, consequently, requires derivation, which is defined in Section 5.7. For the examples shown here, standard first-order logical derivation will be assumed. Case (4) allows a more general universal variable node to subsume a less general existential variable node. For this to happen, for every restriction in the universal variable node there must be a restriction in the existential variable node, and the former restriction must subsume the latter restriction. For example, the variable node corresponding to *every rich girl* would subsume *some rich happy girl* (but not *some girl*). Case (5) allows one existential variable node to subsume another. The requirement for this case is, essentially, that the variables be notational variants of each other. This is because it is not, in general, possible for any existential variable to subsume another except when they are structurally identical. The reason this case is needed (rather than just excluding it entirely) is that for a rule node corresponding to *every girl loves some boy* to subsume *every rich girl loves some boy*, the existential variable node corresponding to the *some boy* in the first rule node must subsume the existential variable node corresponding to the *some boy* in the second rule node (see Section 6.2 for numerous examples of this sort of subsumption in natural language).

Case 1: *Subsume*(`john`, `john`)

Case 2: *Subsume*({`<member`, {`john, bill`}`>`, `<class`, {`man`}`>`},
{`<member`, {`john`}`>`, `<class`, {`man`}`>`})
Subsume(All rich women are happy,
All young rich women that own a car are happy)

Case 3: *Subsume*(All dogs, Fido), provided $\vdash$ dog(Fido).

Subsume(All things have a mass, Fido has a mass)

Case 4: *Subsume*(Every girl, Every happy girl)
Subsume(Every happy girl, Every happy girl that owns a dog)
Subsume(Every girl, Some happy girl)
Subsume(Every happy girl, Some happy girl that owns a dog)

Case 5: *Subsume*(Every girl that loves a boy,
Every girl that loves a boy and that owns a dog)

Figure 5.4: Examples of Subsumption (cases refer to cases of Figure 5.3)

The commonsense nature of the subsumption cases can most easily be illustrated by examples, some of which, for conciseness, are given in natural language in Figure 5.4. The examples illustrate the idea that more general quantified descriptions should subsume less general quantified descriptions of the same sort. In Figure 5.5, a more detailed example for a particular model is given. Node `M2` represents the proposition that `all men are mortal`, `M3` the proposition that `Socrates is a man`, and `M4` the proposition that `Socrates is mortal`. `V1` is the structured variable representing any man. It then follows that `M4` is a special case of `M2` directly by subsumption, since `V1` subsumes `Socrates`. Note that the restrictions on subsumption involving variables is stricter than *Reduce*, which only requires that the wires of one node be a subset of the other.

In the following model (A, B, M, R, U, E, Γ):

A ={`member, class, any`}
B ={`man, mortal, Socrates`}
M ={`M1, M3, M4`}
R ={`M2`}
U ={`V1`}

where:

```
M1 = {<member, {V1}>, <class, {man}>}
M2 = {<member, {V1}>, <class, {mortal}>}
M3 = {<member, {Socrates}>, <class, {man}>}
M4 = {<member, {Socrates}>, <class, {mortal}>}
V1 = {<any, {M1}>}
```

The resulting subsumption:

```
subsume(M2, M4) = T
```

Figure 5.5: Example of Subsumption for a Particular Model

As with reduction (Axiom 4), a proposition that is subsumed by a believed proposition is also a believed proposition. This can be stated as a more general form of Axiom 4.

Axiom 6
$(Subsume(n_1, n_2) \wedge Believe(n_1)) \Rightarrow Believe(n_2)$

So, for example, in Figure 5.5 if nodes `M2` and `M3` are believed and since nodes `M2` subsumes `M4`, then `M4`, once conceived would be believed. This axiom allows the sorts of commonsense description subsumption evident in natural language.

5.4 Quantifier Scoping in Rules

With structured variables, quantifier scoping is expressed explicitly by the presence or absence of dependency arcs. However, the nonlinearity of the representation introduces a new problem in specifying limited quantifier scopings normally expressed by bracketing. For example, the differences in the sentences $\forall x\ \forall y\ (P(x,y) \Rightarrow Q(x))$ and $\forall x\ ((\forall y\ P(x,y)) \Rightarrow Q(x))$ are associated with bracketing and cannot be expressed (directly) in the nonlinear representation suggested here. The first sentence can be represented using structured variables; the second requires augmenting the representation, as its antecedent states a property of the collection of all ys. This should not be seen as a major shortcoming of this representation as several solutions to the problem are possible. Partitioned semantic networks [Hendrix, 1977; Hendrix, 1979] or a representation for collections [Cho, 1992; Cho, 1994] would allow the representation of all possible quantifier scopings.

The representations of [Cho, 1994] illustrate how the collective scope readings can be expressed in ANALOG. Cho introduces a case frame for atomic collections:

$$\{\texttt{<NecAndSuf},\{v\}\texttt{>}\}$$

where v is a structured variable whose restrictions specify necessary and sufficient conditions on members of this collection. Figure 5.6 shows the representation of the collective reading of *All the papers in a folder are a dissertation* (probably the only sensible interpretation). Node `M5!` represents the proposition that the *collection* of papers defined by being in a folder, node `M4`, are a dissertation. Thus, with this augmentation ANALOG can represent narrow scope readings. The semantics of such collections, however, are not clear (i. e., what subcollection of those papers is still a dissertation?) and are dealt with in [Cho, 1994].

In general, an implication whose antecedent and consequent share structured variables is equivalent to a "flat" proposition without the antecedent-consequent rule structure (see Section 5.4.1). In Figure 5.7, a generic antecedent-consequent rule and its propositional form is shown. The flat propositional representation of a rule is derived by moving all constraints on variables in the antecedent into the variable restriction set, and discarding

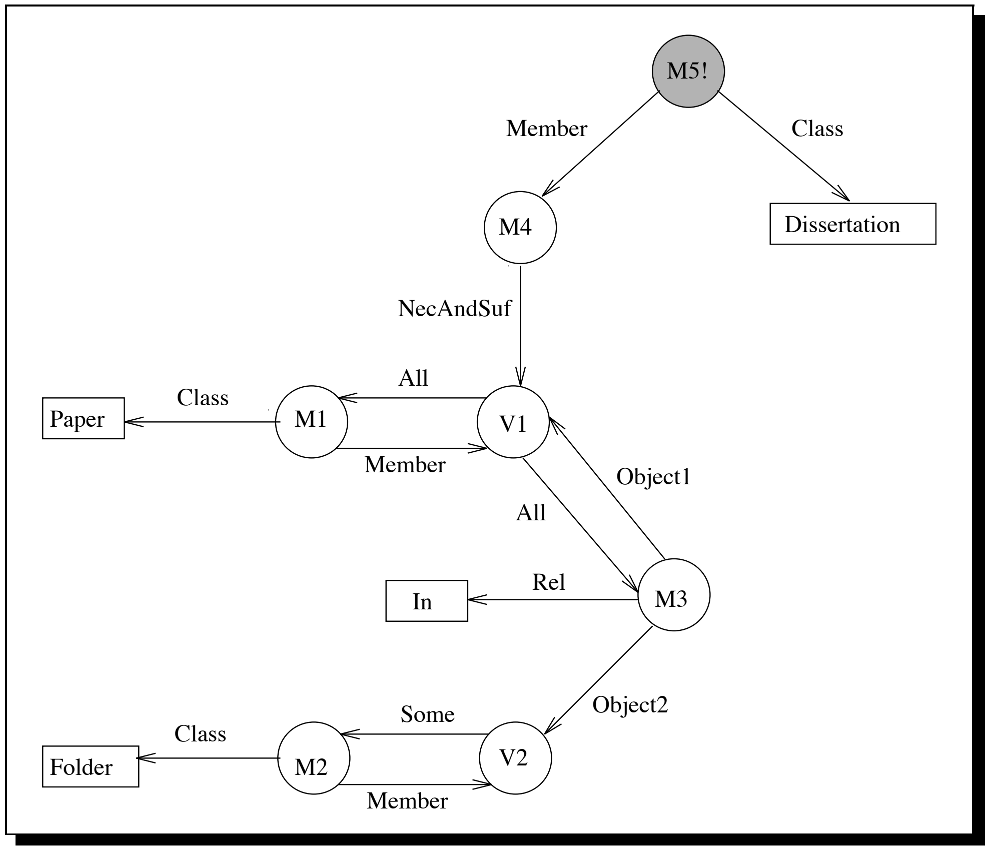


Figure 5.6: All the papers in a folder are a dissertation

the conditional rule structure. So, for example the proposition: *if a block is on another block, the latter is not clear* could be represented as an antecedent-consequent rule as in Figure 5.8(a), whereas its preferred representation should be Figure 5.8(b). This is because the interpretation of 5.8(a) is ambiguous between the intended interpretation, that corresponding to the FOPL sentence:

$$\forall b_1, b_2\ [(\text{block}(b_1) \wedge \text{block}(b_2) \wedge \text{on}(b_1, b_2)) \ \Rightarrow\ \neg\,\text{clear}(b_2)] \qquad (5.5)$$

and

$$\forall b_2\ [(\text{block}(b_2) \wedge (\forall b_1\, \text{block}(b_1) \Rightarrow \text{on}(b_1, b_2))) \ \Rightarrow\ \neg\,\text{clear}(b_2)] \qquad (5.6)$$

and, finally, the interpretation in which if all blocks are on all blocks then all blocks are

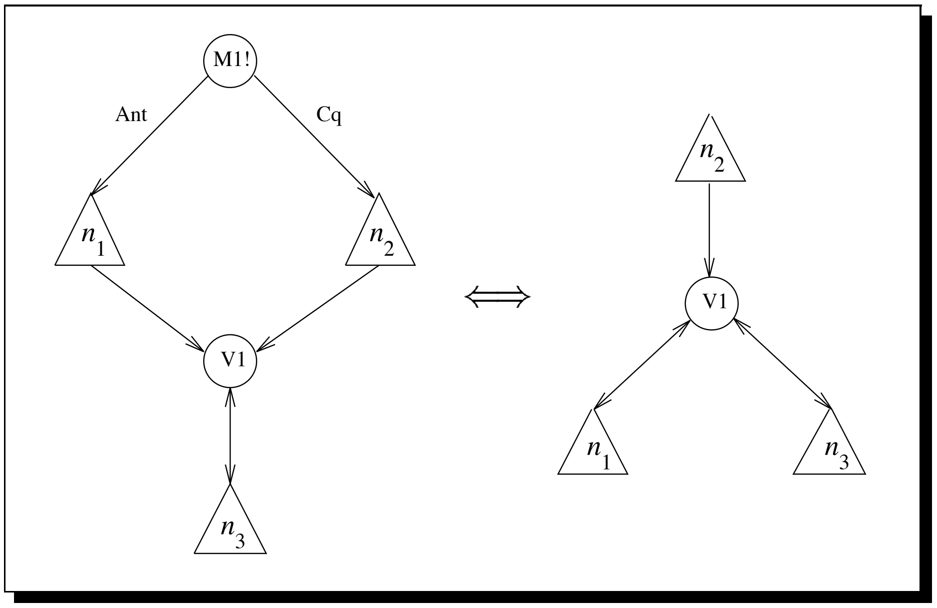


Figure 5.7: Generic rule whose antecedent and consequent share a SV and its equivalent propositional form

not clear,

$$\forall b_1, b_2\ [(\text{block}(b_1) \wedge \text{block}(b_2)) \Rightarrow \text{on}(b_1, b_2)] \Rightarrow \forall b(\text{block}(b) \Rightarrow \neg\,\text{clear}(b)). \quad (5.7)$$

Any of these interpretations of the representation provided in Figure 5.8(a) are possible because quantifier scope is, as yet, unspecified. To resolve this, we stipulate that the intended scope reading is that corresponding to the equivalent rule where the antecedent and consequent do not share any structured variables. Thus, the scope of variables in rules is the entire rule.

One consequence of this is that certain types of quantifier scoping are inexpressible in this specification of the logic. In the block on block example, sentence 5.5 is the only expressible sentence in ANALOG. The sentences expressed by 5.6 and 5.7 cannot be expressed. This is because there is no mechanism for expressing collections in this presentation of the ANALOG formalism. To see this, note that sentence 5.6 is a statement about a *collection* of blocks being on a single block. Finally, sentence 5.7 is a statement about two collections of blocks being on each other. The addition of a collection representation

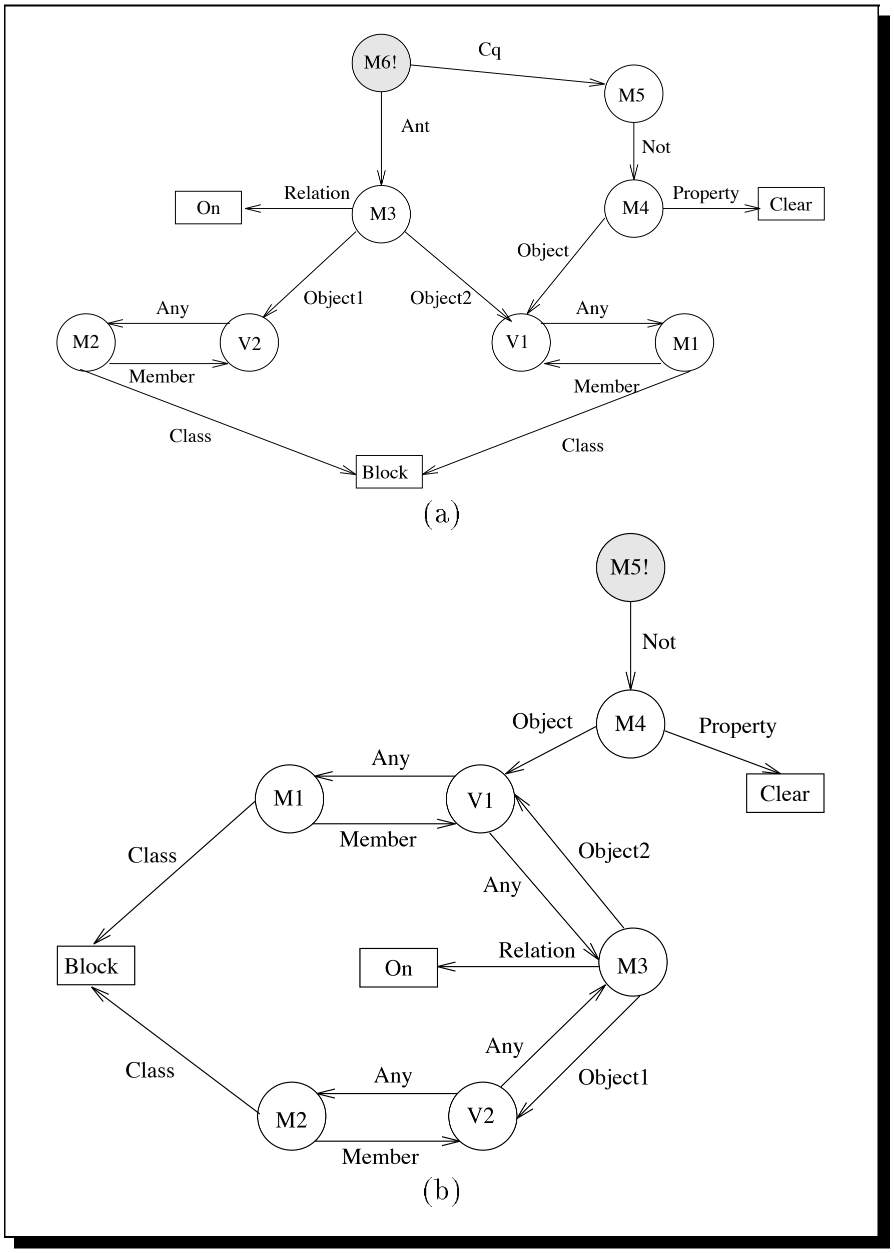


Figure 5.8: Example whose antecedent and consequent share a SV and its equivalent propositional form

is compatible with this formalism (as shown) and would allow the representation and use of sentences such as 5.6 and 5.7.

5.4.1 Rule Node Equivalence

We may formalize the previous presentation of the equivalence of implication rule nodes and their "flat" propositional form. Before doing this we note that rules whose antecedent and consequent share no variables cannot be represented in a flat propositional form. In natural language, such rules can occur (e.g., *If it is raining, then I'm unhappy*). Rules that we are concerned with, however, always involve variables that are shared between antecedent and consequent, because they make general statements. Also, note that rules with multiple consequents will, when converted into a "flat" representation, be represented by multiple propositions. For example, the rule below:

$$\forall x(man(x) \Rightarrow (mortal(x) \wedge happy(x)))$$

when converted would result in the two "flat" propositions:

$$mortal(x)$$

$$happy(x)$$

where x is the arbitrary man.

For rules whose antecedent and consequent share variables, we can specify rule node equivalence below (and write it as "$\equiv$").

1. If $n = \{\texttt{<}ant,\ \{n_1\}\texttt{>},\ \texttt{<}cq,\ \{n_2\}\texttt{>}\}$ or $n = \{\texttt{<}\&ant,\ \{n_1\}\texttt{>},\ \texttt{<}cq,\ \{n_2\}\texttt{>}\}$ and $\forall v_i(\textit{occurs-in}(v_i, n_1) \Rightarrow \textit{occurs-in}(v_i, n_2))$, then $n \equiv n_2\{u_1/v_1, \ldots, u_k/v_k\}$ where

$$u_i = \begin{cases} \{\texttt{<}any, rest(v_i) \cup \{n_1\}\texttt{>}\} & \text{if } v_i \text{ is a universal variable} \\ \{\texttt{<}some, rest(v_i) \cup \{n_1\}\texttt{>},\texttt{<}depends, depends(v_i)\texttt{>}\} & \text{if } v_i \text{ is a existential variable} \end{cases}$$

2. If $n = \{\texttt{<}ant,\ \{n_1\}\texttt{>},\ \texttt{<}cq, ns\texttt{>}\}$ or $n = \{\texttt{<}\&ant,\ \{n_1\}\texttt{>},\ \texttt{<}cq, ns\texttt{>}\}$ then since $n \equiv \{\{\texttt{<}ant,\ \{n_1\}\texttt{>},\ \texttt{<}cq,\ \{u\}\texttt{>}\} \mid u \in ns\}$ or $n \equiv \{\{\texttt{<}\&ant,\ \{n_1\}\texttt{>},\ \texttt{<}cq,\ \{u\}\texttt{>}\} \mid u \in ns\}$, respectively, the previous result applies.

3. If $n = \{\texttt{<}ant, ns_1\texttt{>},\ \texttt{<}cq, ns_2\texttt{>}\}$, then since $n \equiv \{\{\texttt{<}ant,\ \{u\}\texttt{>},\ \texttt{<}cq, ns_2\texttt{>}\} \mid u \in ns_1\}$,

we can use the previous result.

4. If $n = \{\texttt{<}\&ant, \{n_1\} \cup ns_1\texttt{>}, \texttt{<}cq, \{n_2\}\texttt{>}\}$ and $\forall v_i(\mathit{occurs\text{-}in}(v_i, n_1) \Rightarrow \mathit{occurs\text{-}in}(v_i, n_2))$ then, $n \equiv \{\texttt{<}\&ant, \{ns_1\}\texttt{>}, \texttt{<}cq, n_2\{u_1/v_1, \ldots, u_k/v_k\}\texttt{>}$ where

$$u_i = \begin{cases} \{\texttt{<}any, \mathit{rest}(v_i) \cup \{n_1\}\texttt{>}\} & \text{if } v_i \text{ is a universal variable} \\ \{\texttt{<}some, \mathit{rest}(v_i) \cup \{n_1\}\texttt{>}, \texttt{<}depends, \mathit{depends}(v_i)\texttt{>}\} & \text{if } v_i \text{ is a existential variable} \end{cases}$$

5. If $n = \{\texttt{<}\&ant, ns_1\texttt{>}, \texttt{<}cq, ns_2\texttt{>}\}$ then since $n \equiv \{\{\texttt{<}\&ant, ns_1\texttt{>}, \texttt{<}cq, \{u\}\texttt{>}\} \mid u \in ns_2\}$ the previous result applies.

With this specification of rule node equivalence, it is straightforward to represent any standard (widely scoped) first-order-logic rule as a "flat" rule using structured variables. This is desirable because of the natural form constraint and the natural language inference illustrated in Chapter 6.

5.5 Match

Matching is the process of comparing two (or more) nodes to determine if they have a most general common instance (MGI). In the simple cases, one node is an MGI of the other. More complex matchings may return a pair of substitutions which, when applied to the matched nodes, produce an MGI and a node subsumed by the MGI. In Figure 5.9, a pictorial view of the result of matching two nodes is shown. Nodes n_1, called the *source* node, and n_2, called the *target* node, are the nodes being matched. The result of the matching is two substitutions s_1 and s_2 which, when applied to n_1 and n_2, respectively, result in nodes n_3 and n_4. Node n_3 will subsume n_4. Note that there may be more than one pair of substitutions s_1 and s_2 which produces two new nodes in this subsumption relationship.

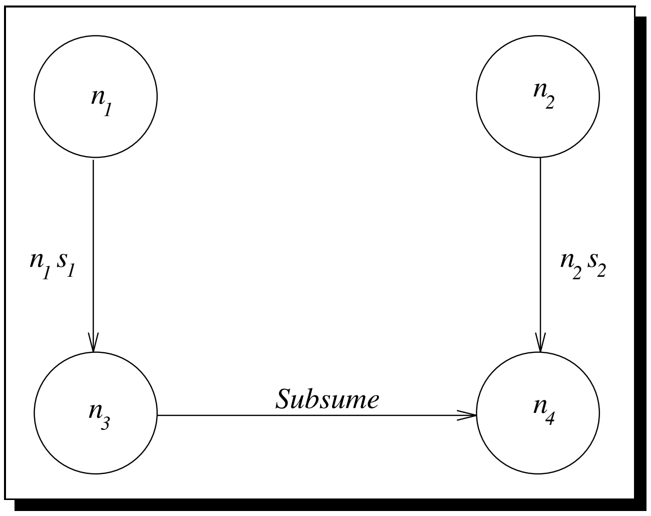


Figure 5.9: Pictorial View of Matching Process

5.5.1 Match Definitions

To formally specify the matching process, we define the following. Note that the examples provided assume a model $(A, B, M, R, U, E, \Gamma)$ where:

$$\{\texttt{member}, \texttt{class}\} \subseteq A$$
$$\{\texttt{JOHN}, \texttt{BILL}, \texttt{MAN}, \texttt{B1}\} \subseteq B$$
$$\{\texttt{M1}, \texttt{M2}\} \subseteq R$$
$$\{\texttt{V1}, \texttt{V2}, \texttt{V3}, \texttt{V4}\} \subseteq U.$$

Definition 24: A **source-target substitution** is an ordered pair (b_1, b_2), where b_1, b_2 are substitutions.

Examples: ({JOHN/V1, BILL/V2}, {JOHN/V3})
({BILL/V2}, {})

Definition 25: A **substitutionset** is a set of source-target substitutions.

Example: {({JOHN/V1,BILL/V2},{JOHN/V3}),({BILL/V2},{})}

5.5.2 Match Specification

The two-way matching procedure takes a pair of nodes i, j and returns a substitution set consisting of source-target substitutions (s_i, t_i) such that *Subsume*$(is_i,\ jt_i)$. To account for the possibility of multiple source-target substitutions, match also takes a substitution set which constrains the possible matches (based on previous recursive matchings). Thus *match* is defined as a function:

$$\textit{match}\text{: } \textit{node} \times \textit{node} \times \textit{substitutionset} \rightarrow \textit{substitutionset} \cup \{\texttt{fail}\}$$

The match procedure recursively generates all possible consistent source-target substitutions on a case by case basis. Each case of the match procedure potentially adds a new binding to the source or target substitutions in the substitution set that is the result of the matching. The match algorithm is given in Figure 5.10.

```
Algorithm 1 Match(i, j, S) in a model $(A,\ B \cup M \cup R \cup U \cup E,\ \Gamma)$

  IF $i = j$ THEN                                                        (1)
      RETURN $S$
  ELSEIF $i \in U$ and $Subsume(ia_k, jb_k)$ THEN                        (2)
      RETURN $\{(a_k \cdot \{j/i\}, b_k) \mid (a_k, b_k) \in S \wedge a_k \cdot \{j/i\}$ is consistent.$\}$
  ELSEIF $j \in U$ and $Subsume(jb_k, ia_k)$ THEN                        (3)
      RETURN $\{(a_k, b_k \cdot \{i/j\}) \mid (a_k, b_k) \in S \wedge b_k \cdot \{j/i\}$ is consistent.$\}$
  ELSEIF $i, j \in E$ THEN
      IF $Subsume(ia_k, jb_k)$ THEN                                      (4)
          RETURN $\{(a_k \cdot \{j/i\}, b_k) \mid (a_k, b_k) \in S \wedge a_k \cdot \{j/i\}$ is consistent.$\}$
      ELSEIF $Subsume(jb_k, ia_k)$ THEN                                  (5)
          RETURN $\{(a_k, b_k \cdot \{i/j\}) \mid (a_k, b_k) \in S \wedge b_k \cdot \{j/i\}$ is consistent.$\}$
      END
  ELSEIF $i, j \in M \cup R$ THEN                                        (6)
      RETURN matchwires(wireset(i), wireset(j), S)
  ELSE
      RETURN fail
  END
```

Figure 5.10: Match Algorithm

```
**Algorithm 2** *matchwires($ws_1$, $ws_2$, $S$)*

IF $S = \textsf{fail}$ or $ws_1 = \{\}$ THEN
    RETURN fail
ELSEIF $ws_2 = \{\}$ THEN
    RETURN $S$
ELSE
    $S \leftarrow \bigcup_{\substack{<r,n>\in ws_1 \\ <r,m>\in ws_2}} \left[ \begin{array}{ll} \textit{matchwires}( & ws_1 - \{<r,n>\}, \\ & ws_2 - \{<r,m>\}, \\ & \textit{match}(n,m,s)) \end{array} \right] - \{\texttt{fail}\}$
    IF $S = \{\}$ THEN
        RETURN fail
    ELSE
        RETURN $S$
    END
END
```

Figure 5.11: Matchwires Algorithm

Case (1) will succeed only if the two nodes are identically the same node. This follows from the uniqueness principle. Since this match involves no variables, no new bindings are added in the event of a successful match. This case occurs when matching two terms that share a subterm. This is particularly likely to occur for base nodes, since there will only be one base node for any intensional individual in the network.

Cases (2) and (3) deal with attempted matches where one node is a universal variable. If the universal variable node subsumes the other node (subject to the current source and target substitutions), a new binding is added to all the source-target substitution sets and the consistent source-target substitutions become the resulting substitution set.

Cases (4) and (5) deal with attempted matchings where both nodes are existential variable nodes. Depending on whether the source subsumes the target or the target subsumes the source (subject to the current source and target substitutions), a new binding is added to all the source or target substitutions and the consistent source-target substitutions become the resulting substitution set.

Case (6) is the general case and deals with attempts to match two molecular or rule nodes. Both the source and target nodes are converted into their equivalent wiresets, and all possible matchings of wires are attempted to determine if the source node can match the target node. This is done using the *matchwires* function:

$$\textit{matchwires}\text{: } \textit{wireset} \times \textit{wireset} \times \textit{substitutionset} \rightarrow \textit{substitutionset} \cup \{\texttt{fail}\}$$

Matchwires is defined in Figure 5.11 and works by attempting to find consistent source-target substitutions such that all wires of a source instance's wireset are in the target instance's wireset.

5.5.3 Example of Match

In Figure 5.12, an example model and result of matching is given. In this example, the node `M1` represents the brotherhood amongst three men. We then ask match to determine all possible pairwise brother relations by matching it against rule node `M7`. This is possible because the `relation-arg` case frame is a reducible case frame. The result of this successful matching is the substitutionset shown in Figure 5.12. There are nine possible source-target substitutions (`V1` and `V2` can match any of `bill`, `john`, and `ted`) in this example. Notice that match returns only six of these. The reason for this is that a substitution must be consistent to be in a source-target substitution. Thus, the substitutions {`bill/V1,bill/V2`}, {`john/V1,john/V2`}, and {`ted/V1,ted/V2`} are eliminated because they violate UVBR (they bind two distinct variables to identical terms).

Note that match generates all permutations of possible bindings in the substitutions returned. Some of these substitutions may be (and in this example are) redundant. To see this in the brothers example, notice that while there are six substitutions, the result of applying all six substitutions to the target will be the new nodes:

```
{<arg, {john, bill}>, <relation, {brother}>},
{<arg, {ted, bill}>, <relation, {brother}>},
{<arg, {john, ted}>, <relation, {brother}>}
```

The model $(A,\ B \cup M \cup R \cup U \cup E,\ \Gamma)$:

A ={member, class, arg, relation, any}
B ={man, brother, bill, john, ted}
M ={M1, M2, M2, M3, M4, M5, M6}
R ={M7}
U ={V1,V2}

where:

```
M1 = {<arg, {bill,john,ted}>, <relation, {brother}>}
M2 = {<member, {john}>, <class, {man}>}
M3 = {<member, {bill}>, <class, {man}>}
M4 = {<member, {ted}>, <class, {man}>}
M5 = {<member, {V1}>, <class, {man}>}
M6 = {<member, {V2}>, <class, {man}>}
V1 = {<any, {M5}>}
V2 = {<any, {M6}>}
M7 = {<arg, {V1,V2}>, <relation, {brother}>}
```

The resulting match:

```
match(M1, M7, {}) = {({}, {john/V1,bill/V2})
                     ({}, {bill/V1,john/V2})
                     ({}, {ted/V1,bill/V2})
                     ({}, {ted/V1,john/V2})
                     ({}, {bill/V1,ted/V2})
                     ({}, {john/V1,ted/V2})
                     }
```

Figure 5.12: **Example of Match for a Particular Model**

rather than six new nodes. This happens because the nodesets associated with cables are sets and thus order of nodes in the nodeset is not important. The uniqueness principle enforces that only three new nodes are built. It is not, in general, desirable or possible to avoid this, because the relation associated with the proposition being matched may or may not be symmetric. The brothers relation is clearly symmetric (i.e., $brothers(x, y) \iff brothers(y, x)$), but relations such as *on* are not.

5.6 Subsumption, Matching, and Unification

Subsumption is a subcase of matching in which one of the terms being matched is an instance of the other (for example *any man* subsumes *some man*). Matching, however, is possible between terms that are not instances of each other (the brothers example in Figure 5.12, for instance). Additionally, subsumption does not produce a most general instance and associated substitutionsets. Matching, in ANALOG, is not standard unification. It is more like two-way matching because of UVBR, and, because reduced forms of terms may match (reduce and subsumption), it differs from the standard formulation of unification.

5.7 Inference

Inference, the procedure for generating new knowledge from old, may now be specified in terms of the metapredicates and functions previously defined. In [Shapiro and Rapaport, 1987], a variety of non-standard rule case frames are defined. In this dissertation we specify how the rules for instantiation, conjunctive implication, and disjunctive implication may be used. Since we have provide representations for conjunction, disjunction, and negation, this suffices for derivation not involving numerical quantifiers. To this end, we define the following node-based inference metarules.

5.7.1 Instantiation

$$\frac{\begin{array}{l} \mathit{Believe}(v) \\ \mathit{Conceive}(u) \\ (\sigma, \tau) \in \mathit{match}(v, u, \{\}) \end{array}}{\mathit{Believe}(u\tau)}$$

This metarule for instantiation of nodes requires that there be a believed node and a conceived node that match. In this event, the instance of the conceived node that is an instance of the believed node may be believed. Typically the conceived node is a node whose deduction has been requested as in a question.

The model (A, $B \cup M \cup R \cup U \cup E$, Γ):

$A = \{\texttt{member, class, any}\}$
$B = \{\texttt{man, mortal, Socrates}\}$
$M = \{\texttt{M1, M3, M4}\}$
$R = \{\texttt{M2}\}$
$U = \{\texttt{V1}\}$
$\Gamma \supseteq \{\texttt{M2, M3}\}$

where:

```
M1 = {<member, {V1}>, <class, {man}>}
M2 = {<member, {V1}>, <class, {mortal}>}
M3 = {<member, {Socrates}>, <class, {man}>}
M4 = {<member, {Socrates}>, <class, {mortal}>}
V1 = {<any, {M1}>}
```

The resulting inference:

$$\texttt{M4} \in \Gamma$$

Figure 5.13: Example of Instantiation for a Particular Model

Figure 5.13 gives an example of this type of inference. In this example $v = \texttt{M2}$ and $u = \texttt{M4}$, so $\mathit{match}(v, u, \{\}) = \{(\{\}, \{\texttt{Socrates/V1}\})\}$. Note that `V1` (*any man*) can be bound to `Socrates` only because `M3` asserts that he is a man, and thus $\mathit{Subsume}(\texttt{V1}, \texttt{Socrates})$. This inference corresponds to the standard FOPL derivation:

$$\{\forall x\ \texttt{man}(x) \Rightarrow \texttt{mortal}(x),\ \texttt{man(Socrates)}\} \vdash \texttt{mortal(Socrates)}.$$

Note that in ANALOG, this derivation is done by instantiation rather than derivation (and forward or backward inference) and is a more abbreviated process (in this example, one step).

5.7.2 Instantiation of Shared Terms

$$\frac{\begin{array}{l} \mathit{Believe}(v) \\ \mathit{Conceive}(u) \\ (\sigma,\tau) \in \mathit{match}(v,u,\{\}) \\ \mathit{Believe}(w) \\ \mathit{variableset}(\tau) \subseteq \mathit{variableset}(w) \end{array}}{\begin{array}{l} \mathit{Believe}(u\tau) \\ \mathit{Believe}(w\tau) \end{array}}$$

This metarule for instantiation of nodes requires that there be a believed node and a conceived node that match. In this event, the instance of the conceived node that is an instance of the believed node may be believed. Additionally, any other believed node whose variableset is a superset of the variableset of the target substitution may have its instance with the target substitution applied believed. Typically the conceived node is a node whose deduction has been requested as in a question.

Figure 5.14 gives an example of this type of inference. In this example v = `M2`, u = `M4`, and w = `M5` so $\mathit{match}(v,\ u,\ \{\}) = \{(\{\},\ \{$`Socrates/V1`$\})\}$, and $\mathit{varset}(\tau) = \mathit{varset}(w) =$ {`V1`}. This corresponds to the system answering the question *Is Socrates mortal?* and also returning, by the way, *Socrates is brave.* Note that, in general, this additional derivation of w is felicitous only if there was some prior derivation requested that failed (possibly because the required propositions were not asserted) to deduce w. Thus the derivation of w should be "pending" new facts. We will not formalize this idea here, but merely indicate how it fits into the formalism.

5.7.3 Or-Entailment

Or-entailment rules are rule nodes of the form: $\{<\mathit{ant}, \mathit{ns}_1>,\ <\mathit{cq}, \mathit{ns}_2>\}$, corresponding to standard antecedent-consequent rules. The antecedent propositions are disjunctive. It suffices for any $n \in \mathit{ns}_1$ to match a believed node for all the consequents $u \in \mathit{ns}_2$ to become believed, thus the term "or-entailment". We specify the metarule below.

The model (A, $B \cup M \cup R \cup U \cup E$, Γ):

A =\{`member, class, property, any`\}
B =\{`man, mortal, Socrates, brave`\}
M =\{`M1, M3, M4`\}
R =\{`M2`\}
U =\{`V1`\}
$\Gamma \supseteq$\{`M2, M3`\}

where:

```
M1 = {<member, {V1}>, <class, {man}>}
M2 = {<member, {V1}>, <class, {mortal}>}
M3 = {<member, {Socrates}>, <class, {man}>}
M4 = {<member, {Socrates}>, <class, {mortal}>}
M5 = {<object, {V1}>, <property, {brave}>}
M6 = {<member, {Socrates}>, <class, {brave}>}
V1 = {<any, {M1}>}
```

The resulting inference:

$$\texttt{M4, M6} \in \Gamma$$

Figure 5.14: Example of Instantiation of Shared Terms for a Particular Model

$$\frac{\begin{array}{l} \mathit{Believe}(\{\texttt{<}\mathit{ant}, \{u_1, \ldots, u_n\}\texttt{>}, \texttt{<}\mathit{cq}, \mathit{ns}\texttt{>}\}) \\ \mathit{Believe}(w) \\ (\sigma, \tau) \in \mathit{match}(w, u_i, \{\}) \end{array}}{\bigwedge_{v \in \mathit{ns}} \mathit{Believe}(v\tau))}$$

This metarule for or-entailment of nodes requires that there be a believed or-entailment rule node and a believed node that matches *one* of the antecedents of the rule. In this event the appropriate instances of the consequents of the rule node are then believed. Figure 5.15 gives an example of this type of inference, corresponding to the standard FOPL derivation:

$$\{(\texttt{rich(John)} \vee \texttt{lucky(John)}) \Rightarrow \texttt{happy(John)},\ \texttt{rich(John)}\} \vdash \texttt{happy(John)}$$

The model $(A,\ B \cup M \cup R \cup U \cup E,\ \Gamma)$:

$A =$\{`object, property`\}
$B =$\{`rich, lucky, happy, John`\}
$M =$\{`M2, M3, M4`\}
$R =$\{`M1`\}
$\Gamma \supseteq$\{`M1, M2`\}

where:

```
M1 = {<ant, {M2,M3}>, <cq, {M4}>}
M2 = {<object, {John}>, <property, {rich}>}
M3 = {<object, {John}>, <property, {lucky}>}
M4 = {<object, {John}>, <property, {happy}>}
```

The resulting inference:

$$\texttt{M4} \in \Gamma$$

Figure 5.15: Example of Or-entailment for a Particular Model

5.7.4 And-Entailment

And-entailment rules are rule nodes of the form: $\{<\&ant, ns_1>, <cq, ns_2>\}$, corresponding to standard antecedent-consequent rules. The antecedent propositions are conjunctive. It is necessary for all $n \in ns_1$ to match believed nodes consistently for all the consequents $u \in ns_2$ to become believed, thus the term "and-entailment". We specify the metarule below.

$$\begin{array}{l}
Believe(\{<\&ant, \{u_1, \ldots, u_n\}>, <cq, ns>\}) \\
(\bigwedge_{i=1}^{n} Believe(w_i)) \\
(\sigma_i,\ \tau_i) \in match(w_i, u_i, \{\}) \\
(\sigma_1 \cdot \sigma_2 \cdot \ldots \cdot \sigma_n,\ \tau_1 \cdot \tau_2 \cdot \ldots \cdot \tau_n) \text{ is consistent.} \\
\hline
\bigwedge_{v \in ns} Believe(v\tau_1 \cdot \tau_2 \cdot \ldots \cdot \tau_n))
\end{array}$$

This metarule for and-entailment of nodes requires that there be a believed and-entailment rule node and believed nodes that consistently match all the antecedents of the rule. In this event, the appropriate instances of the consequents of the rule node are then believed. Figure 5.16 gives an example of this type of inference, corresponding to the standard FOPL derivation:

$$\{(\texttt{rich}(\texttt{John}) \wedge \texttt{lucky}(\texttt{John})) \Rightarrow \texttt{happy}(\texttt{John}),\ \texttt{rich}(\texttt{John}),\ \texttt{lucky}(\texttt{John})\} \vdash \texttt{happy}(\texttt{John})$$

The model (A, $B \cup M \cup R \cup U \cup E$, Γ):

$A = \{\texttt{object, property}\}$
$B = \{\texttt{rich, lucky, happy, John}\}$
$M = \{\texttt{M2, M3, M4}\}$
$R = \{\texttt{M1}\}$
$\Gamma \supseteq \{\texttt{M1, M2, M3}\}$

where:

```
M1 = {<&ant, {M2,M3}>, <cq, {M4}>}
M2 = {<object, {John}>, <property, {rich}>}
M3 = {<object, {John}>, <property, {lucky}>}
M4 = {<object, {John}>, <property, {happy}>}
```

The resulting inference:

$$\texttt{M4} \in \Gamma$$

Figure 5.16: Example of And-entailment for a Particular Model

5.8 Summary

We have formally specified the syntax, semantics, subsumption, matching, and inference mechanisms in the ANALOG system. The subsumption mechanism takes advantage of the conceptual completeness of the structured variable representation to allow the kinds of common sense description subsumption relationships that are pervasive in natural language. This leads to a "flat" representation for rules which allows the kinds of inference procedures that are pervasive in natural language.

Chapter 6

Natural Language Processing

Since a motivation for developing this system was to facilitate natural language processing, in this chapter we present examples of the kinds of NLP that ANALOG is capable of.

We note that portions of this chapter have previously been published in [Ali, 1993a; Ali, 1993b; Ali and Shapiro, 1993].

6.1 Processing Natural Language

A parser/generator is one component of the ANALOG system. A generalized augmented transition network grammar [Shapiro, 1982a] is used in the natural language demonstrations that follow. The grammar parses and generates a limited subset of English, corresponding to the constructions used in the demonstrations.

6.2 ANALOG for Natural Language Processing

So far, we have motivated some aspects of the logic underlying the ANALOG KRR system and formalized some important concepts, such as subsumption and derivation, associated with the logical system. At this point, we will attempt to illustrate the utility of the system for natural language processing with specific examples of natural language.

ANALOG includes a generalized augmented transition network (GATN) natural language parser and generation component linked up to the knowledge base (based on [Shapiro, 1982b]). A GATN grammar specifies the translation/generation of sentences involving complex noun phrases into/from ANALOG structured variable representations.

We present four demonstrations of the NLP component of ANALOG. The first illustrates the representation and use of complex noun phrases, the second illustrates the use of non-linear quantifier scoping and structure sharing, the third is a detailed presentation (with most of the underlying ANALOG representations) of a demonstration that illustrates the use of rules as valid and useful answers to questions, and the fourth demonstrates nested subsumption derivation. The last three demonstrations also have examples of subsumption and derivation, with the fourth one being the most complex example. We note that all these demonstrations are packaged with the standard ANALOG distribution.

6.2.1 Representation of Complex Noun Phrases

An advantage of the use of structured variables lies in the representation and generation of complex noun phrases that involve restrictive relative clause complements. The restriction set of a structured variable typically consists of a type constraint along with property constraints (adjectives) and other more complex constraints (restrictive relative clause complements). So, when parsing a noun phrase, all processing is localized and associated with building its structured variable representation. When generating a surface noun phrase corresponding to the structured variable, all constraints associated with the variable are part of its structure and can be collected and processed easily. This is in contrast to non-structured variable representations (such as FOPL) where the restrictions on variables are disassociated from the variables themselves, in the antecedents of rules.

In Figure 6.1, user input is italicized; the text at the beginning is a standard message and will be omitted from the remaining figures. These are actual interactions with ANALOG's NLP component. The ":" at the beginning of each line of user input, is the parser prompt. Figure 6.1 shows example sentences with progressively more complex noun phrases being used. These noun phrases are uniformly represented using structured

```
(parse -1)
ATN parser initialization...
Input sentences in normal English orthographic convention.
Sentences may go beyond a line by having a space followed by a <CR>
To exit the parser, write ^end.
: Every man owns a car
I understand that every man owns some car.
: Every young man owns a car
I understand that every young man owns some car.
: Every young man that loves a girl owns a car that is sporty
I understand that every young man that loves any girl owns some sporty
car.
: Every young man that loves a girl that owns a dog owns a red car that is sporty
I understand that every young man that loves any girl that owns any
dog owns some red sporty car.
: Every young man that loves a girl and that is happy owns a red sporty car that
wastes gas
I understand that every young happy man that loves any girl owns some
sporty red car that wastes gas.
: ^end
ATN Parser exits...
```

Figure 6.1: Examples of complex noun phrase use that correspond to structured variables

variables. Parsing and generation of these noun phrases is simplified because structured variables collect all relevant restrictions on a variable into one unit, a structured variable. The parser parses the user's sentence and builds an ANALOG representation for the user input. The resulting representation is then passed to the generation component, which generates the output response (sometimes prefixed by the canned phrase `I understand that`). If constraints on variables corresponding to the complex noun phrases were represented using FOPL, then it would be difficult to generate natural language noun phrases corresponding to these variables. This is because the constraints on variables would, likely, be well separated from the variables in the antecedents of rules involving these variables. This is not the case in a structured variable representation.

6.2.2 Non-Linear Quantifier Scopings and Structure Sharing

Since this representational formalism is grounded in an inherently nonlinear notation (semantic networks), the representation of tree-like quantifier scopings is straightforward. Thus, sentences involving branching quantifiers can be represented. In addition, conceptual completeness constraint and the semantics of ANALOG's semantic network representation allows structure-sharing (and, indeed, may require it) to a high degree. This addresses one of the initial goals; the representation of co-reference and structure sharing in multisentential dialog.

A discourse model requires maintaining a structure containing focus of attention, linguistic structure, and text purpose. The natural form and conceptual completeness constraints of this work address these tasks and facilitates discourse processing Since discourse modelling was not a primary goal of this work the parser and generator used in these examples maintains a very simple discourse model (a simple focus list containing structured variables currently in focus). Consequently, we do not illustrate complex examples (such as the branching quantifier sentence) of structure sharing using natural language (although such representations may readily be built). A simpler example, which can be handled by our simple discourse model, is the donkey sentence where a scoped constituent of a noun phrase (the *donkey* in *Every farmer who owns a donkey beats it*) is used in the main clause of the sentence. Figure 6.2 illustrates a dialog involving questions about the donkey sentence.

In Figure 6.2, note that the system is initially unable to determine whether Fred beats Doc or Dumbo. This is because the initial rule (`every man that owns some donkey beats it`) is satisfied in a model where only one of the donkeys is being beaten. After the system is told that *all* such donkeys are beaten, it does determine that Fred beats Doc. Note that this determination also requires that the description `every man` subsume `Fred`, who is a boy (and consequently a man). This is an example of derived subsumption. Also note that the answers to many questions are often rules themselves (e.g., `Who beats a donkey` has as one answer `Every man that owns some donkey`). This is discussed in the next section.

```
: Every man that owns some donkey beats it
I understand that every man that owns some donkey beats some donkey.
: Fred is a boy
I understand that Fred is a boy.
: Every boy is a man
I understand that every boy is a man.
: Doc is a donkey
I understand that Doc is a donkey.
: Dumbo is a donkey
I understand that Dumbo is a donkey.
: Fred owns Doc
I understand that Fred owns Doc.
: Fred owns Dumbo
I understand that Fred owns Dumbo.
: Does Fred beat Doc
I don't know.
: Does any man beat some donkey
Yes, every man that owns some donkey beats some donkey.
: Every man that owns any donkey beats it
I understand that every man that owns any donkey beats every donkey.
: Does Fred beat Doc
Yes, Fred beats Doc.
: Does any man beat some donkey
Yes, every man that owns some donkey beats some donkey.
: Does any man beat any donkey
Yes, Fred beats Doc and every man that owns any donkey beats every
donkey and every man that owns some donkey beats some donkey.
: Who beats a donkey
Fred beats Doc and every man that owns any donkey beats every donkey
and every man that owns some donkey beats some donkey.
```

Figure 6.2: Example of structure sharing in donkey sentence.

6.2.3 Rules as Answers to Questions

Because the structure of the representation of rules is "flat", that is, there is not the artificial antecedent-consequent structure associated with first-order logic-based representations, it is possible to frame questions whose answers are rules and not just ground formulas. Since the structure of the question will mirror the structure of the rule, any rule that is subsumed by a question is an answer to that question. Figure 6.3 gives a sample

```
: Every man is mortal                                                    (1)
I understand that every man is mortal.                                   (2)
: Who is mortal                                                          (3)
Every man is mortal.                                                     (4)
: Is any rich man mortal                                                 (5)
Yes, every rich man is mortal.                                           (6)
: John is a man                                                          (7)
I understand that John is a man.                                         (8)
: Is John mortal                                                         (9)
Yes, John is mortal.                                                    (10)
: Who is mortal                                                         (11)
John is mortal and every rich man is mortal and every man is mortal.    (12)
: Are all rich young men that own some car mortal                       (13)
Yes, every young rich man that owns some car is mortal.                 (14)
: Any rich young man that owns any car is happy                         (15)
I understand that every young rich man that owns any car is happy.      (16)
: Is John happy                                                         (17)
I don't know.                                                           (18)
: Young rich John owns a car                                            (19)
I understand that mortal rich young John owns some car.                 (20)
: Who owns a car                                                        (21)
Mortal rich young John owns some car.                                   (22)
: Is John happy                                                         (23)
Yes, mortal rich young John is happy.                                   (24)
```

Figure 6.3: Examples of questions that have rules as answers.

dialog involving questions whose answers are ground propositions (e.g., *Is John mortal*) as well as questions whose answers are rules (e.g., *Who is mortal*).

This dialog also illustrates the uses of subsumption. Since we told the system *Every man is mortal*, it follows that any more specifically constrained man (e.g., *Every rich young man that owns some car*) must also be mortal. Note that this answer (a rule) follows directly by subsumption from a rule previously told to the system. This is another way in which rules may be answers to questions.

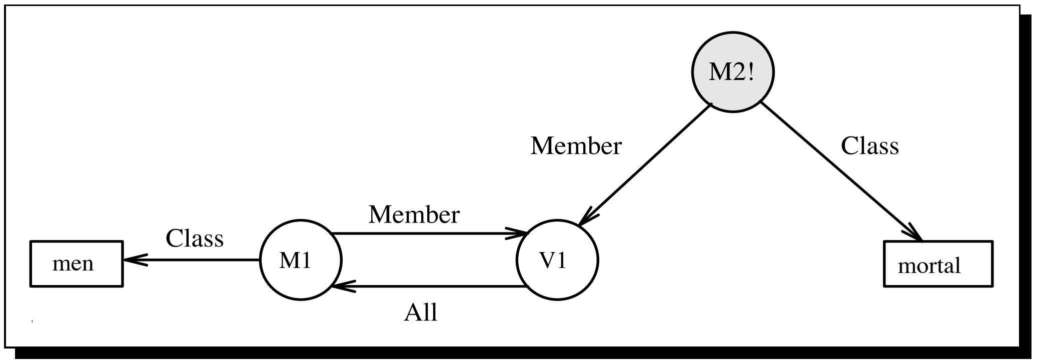


Figure 6.4: Representation of sentence (1): *Every man is mortal.*

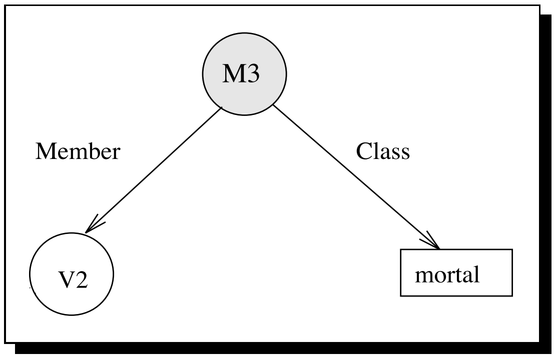


Figure 6.5: Representation of sentence (3): *Who is mortal?*

6.2.4 A Detailed Demonstration Examination

In this section, we present the representations and processing associated with the last demonstration in detail. All references to sentences will be to the numbered sentences in Figure 6.3. The representation for sentence (1) is that of Figure 6.4. Sentence (3) then asks who is mortal. In a standard FOPL-based system, no answer could be given because there are, as yet, no instances of men in the knowledge base. This is contrary to the commonsense answer of sentence (4), which reiterates the rule of sentence (1). This is possible in ANALOG because the structure of the representation of the question (*Who is mortal*) is similar to that of any of its answers. Thus, any asserted proposition that is subsumed by the question is a valid answer (including rules).

Sentence (5) is an example of a question about a rule. Since *every man is mortal* is believed (the system was told this in sentence (1)) it follows that any more restricted sort of man is also mortal. The subsumption procedure specifies this explicitly. The represen-

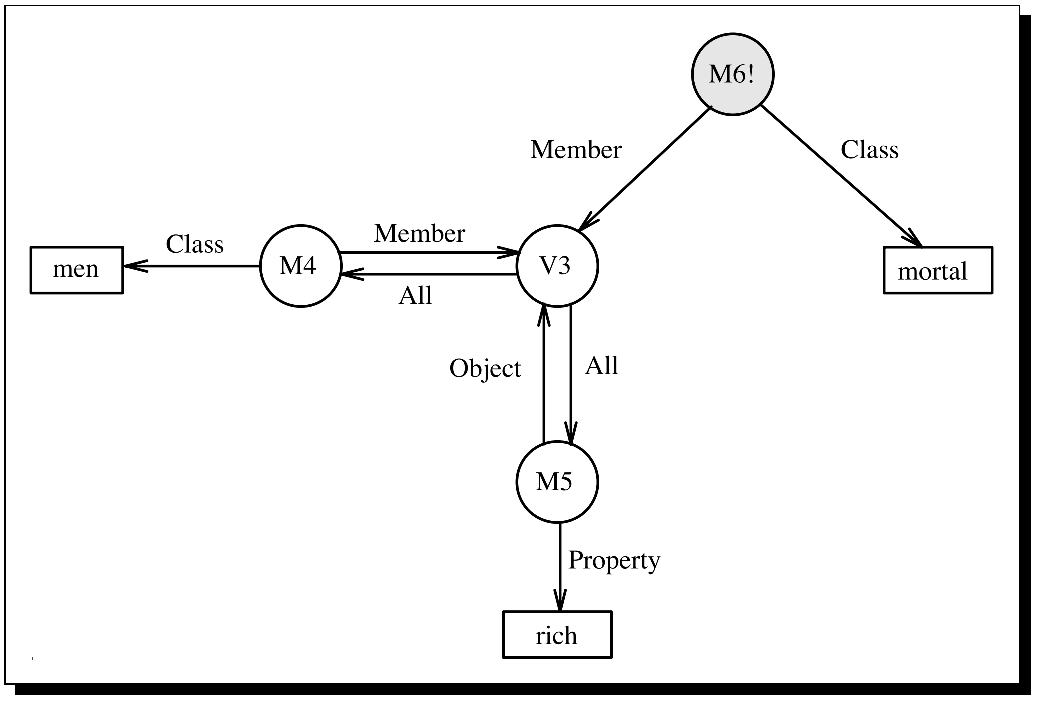


Figure 6.6: Representation of sentence (5): *Is any rich man mortal?*

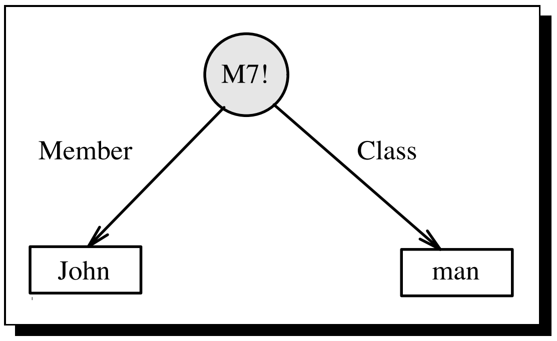


Figure 6.7: Representation of sentence (7): *John is a man.*

tation of sentence (5) in Figure 6.6 is a less general form of sentence (1) in Figure 6.4, since `V1` (any man) subsumes `V3` (any rich man). Since the rule in Figure 6.6 is subsumed by a believed node (that of sentence (1)), it follows by Axiom 5 that sentence (5) is believed (thus, the representation of the question itself is a believed proposition), and the system answers yes to the question. Sentence (7) asserts that *John is a man*, and the result is the representation of Figure 6.7. At this point, the system knows *all men are mortal* and *John is a man.* When the question of sentence (9) (whose representation is in Figure 6.8) is asked, the system finds the rule of sentence (1) and determines that it subsumes sentence

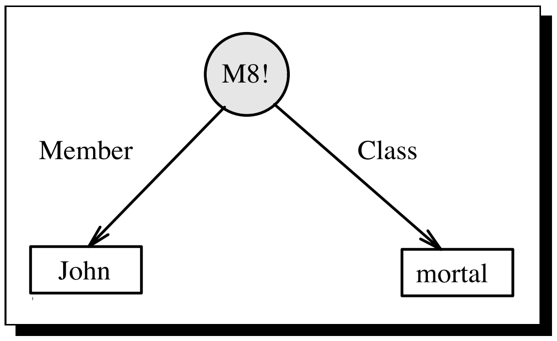


Figure 6.8: **Representation of sentence (9):** *Is John mortal?*

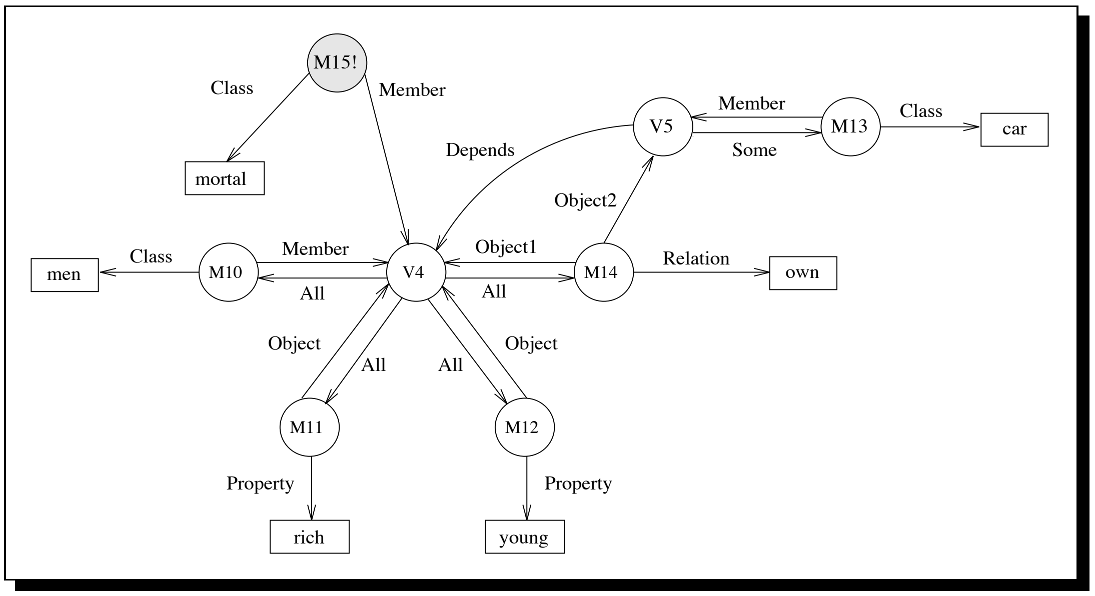


Figure 6.9: **Representation of sentence (13):** *Are all rich young men that own some car mortal?*

(9) because John is a man, and again by Axiom 5 the result follows. However, note that in this derivation the result is a ground formula rather than a rule. Sentence (11) illustrates the retrieval of the system's information about who is mortal; note the additional believed propositions. Sentence (13) is an example of a more complex noun phrase in a rule. The representation of (13) is in Figure 6.9 and is subsumed by that of sentence (1) or (5) leading to the yes answer. In Figure 6.9, `V4` represents the arbitrary rich young man that owns some car, and `V5` represents some owned car of `V4`. In sentence (15) (Figure 6.10), a new rule about rich young car-owning men (`V6`) being happy (`M21`) is introduced. The question of sentence (17) (*is John happy*) cannot be answered, because the system cannot

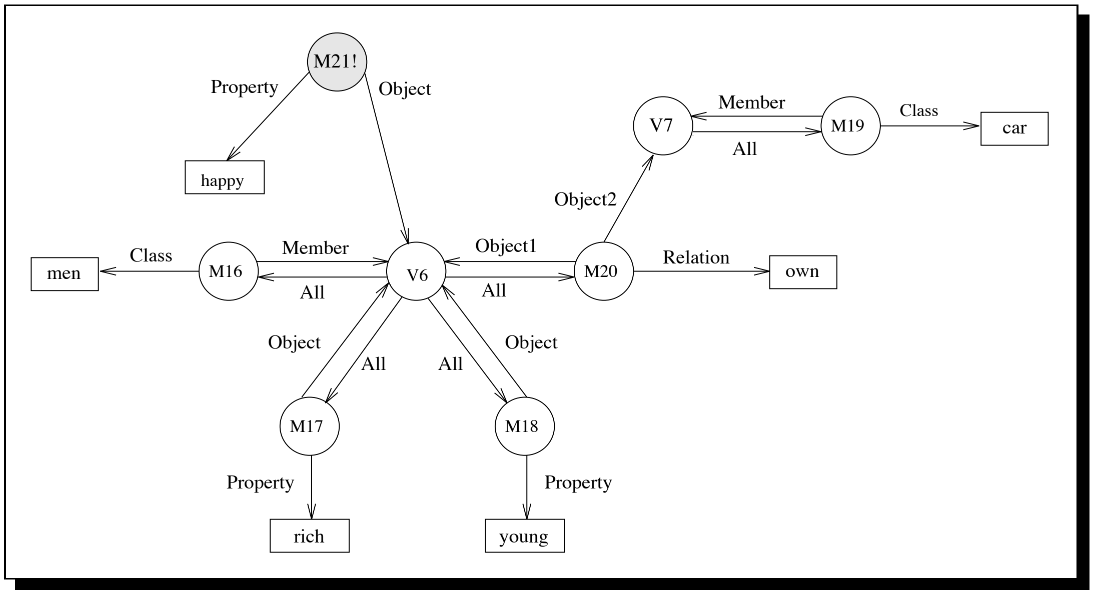


Figure 6.10: Representation of sentence (15): *Any rich young man that owns any car is happy.*

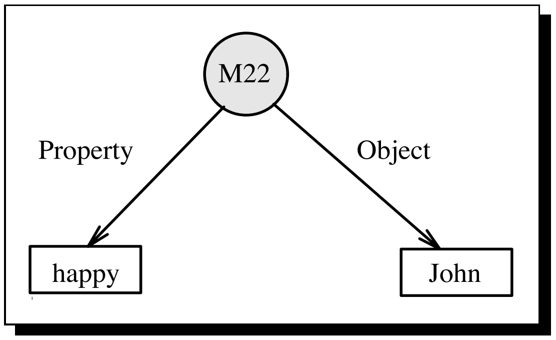


Figure 6.11: Representation of sentence (17): *Is John happy?*

determine that node `V6` subsumes `John`. This is because, while `John` is a man, he is not known to be young, rich, and owning a car (requirements for this subsumption). Sentence (19) informs the system of these requirements the systems understanding is verified by question (21), whose representation is shown in Figure 6.12. Note that the structure of the question involving two variables (*Who* and *a car*) is identical to that of the structure of its answer, which would not be the case if constraints were separated from variables in the antecedents of rules (as is done in typical logics). The question is asked again and, because the subsumption relationship can be determined, is answered in the affirmative.

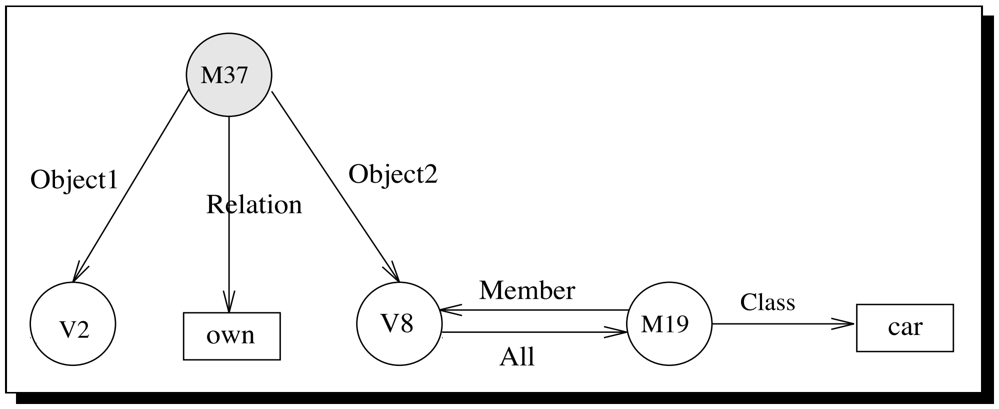


Figure 6.12: Representation of sentence (21): *Who owns a car?*

6.2.5 Nested Subsumption Derivation

To show how powerful the subsumption inference procedure is, our final demonstration gives an example involving several nested derivations. In the interest of brevity, we omit showing any more representations.

In Figure 6.13 sentences (1) through (19) provide all required information for the affirmative answer to question (21). To answer (21) the system must use the rule of sentence (1), namely *every rich man that loves a sweet girl is happy*, since that is the only applicable rule. In attempting to apply this rule the description *every rich man* must subsume *John*, thus the system attempts to determine there are any assertions or rules about *John* being a man and rich. To show he is a man the system uses the only rule about things being men, the rule of sentence (6), *every farmer is a man.* Since *John* is a *farmer* (sentence (11)) it follows by subsumption that he is a man. To show that *John* is rich, the rules of sentences (4), (7), and (14) must be used. The attempt to use the rule of sentence (4), namely *every old farmer is rich*, requires that *John* be a farmer (given in sentence (11)) and that he be old. To establish the latter, the rule of sentence (7), namely *every man that drives a car that wastes gas is old*, is used with the fact that *John* drives such a car. At this point he system has established that *John* is a rich man, so it then moves onto trying to establish whether he loves any sweet girl. This is established in a similar manner, using sentences (9) and (17). We note that most of the preceding derivation was the result of the subsumption query: *Subsume(Every rich man, John)* and

: *Every rich man that loves a sweet girl is happy* *(1)*
`I understand that every rich man that loves any sweet girl is happy.` *(2)*
: *Every old farmer is rich* *(3)*
`I understand that every old farmer is rich.` *(4)*
: *Every farmer is a man* *(5)*
`I understand that every farmer is a man.` *(6)*
: *Every man that drives a car that wastes gas is old* *(7)*
`I understand that every man that drives any car that wastes gas is old.` *(8)*
: *Every girl that owns a dog is sweet* *(9)*
`I understand that every girl that owns any dog is sweet.` *(10)*
: *John is a farmer* *(11)*
`I understand that John is a farmer.` *(12)*
: *John drives a car that wastes gas* *(13)*
`I understand that John drives some car that wastes gas.` *(14)*
: *Lucy is a girl* *(15)*
`I understand that Lucy is a girl.` *(16)*
: *Lucy owns a dog* *(17)*
`I understand that Lucy owns some dog.` *(18)*
: *John loves Lucy* *(19)*
`I understand that John loves Lucy.` *(20)*
: *Is John happy* *(21)*
`Yes, rich old John is happy.` *(22)*

Figure 6.13: Nested Subsumption Derivation Example.

its successful conclusion is due to the fact that the structure of representations (rules as well as facts) are identical and thus no complex back-chaining through rules is required.

6.3 Summary

In the preceding chapter, we described, in some detail, a propositional semantic network based knowledge representation and reasoning system that facilitate many aspects of language processing. In particular, the representation and generation of complex noun phrases, the representation of various types of quantified variable scoping, a high degree of structure sharing, and subsumption of the sort typically associated with ordinary natural language use. In this chapter we presented examples of natural language dialog and some of their associated representations in the ANALOG system that illustrate the utility of this formalism for natural language processing.

Chapter 7

Conclusions

7.1 Summary

We have specified a knowledge representation and inference formalism that is particularly well suited to natural language processing tasks. In this formalism variables are structured terms containing quantificational, type and other information. Consequently, every term of a sentence is closed. The need for this property was motivated by the observation that any language with (potentially) open sentences is an inappropriate medium for the representation of natural language sentences. Open sentences in such languages are a consequence of the separation of quantifier and type constraints on variables from these variables, typically in the antecedents of rules. In contrast, variables in natural language are constructions, such as noun phrases, that are typed and quantified as they are used. A consequence of this is that variables in natural language may be freely re-used in dialog. In language, this leads to the use of pronouns and discourse phenomena such as ellipsis involving reuse of entire subformulas. We presented an augmentation to the representation of variables so that variables are not atomic terms. These "structured" variables are typed and quantified as they are defined and used. This has three important consequences with respect to natural language. First, this leads to an extended, more "natural" formalism whose use and representations are consistent with the use of variables in natural language (e. g., allowing re-use phenomena such as pronouns and ellipsis). Secondly, the formalism allows the specification of terminological subsumption as a partial ordering on related concepts (variable nodes in a semantic network) that relates more general concepts to more specific instances of that concept, as is done in language. Finally, this structured variable representation simplifies the resolution of some representational difficulties with certain classes of natural language sentences; donkey sentences and sentences involving branching quantifiers. We justified all of the above with a formal specification of the ANALOG system, and an implementation that demonstrates the use of the formalism with natural language demonstrations.

7.2 Contributions

In this work we have specified a KRR formalism that is well suited to NLP applications. The primary contributions of this work lie in showing that a logic motivated by natural language concerns is both plausible and computationally well-specified. We took some general issues that are problematic for representation languages based on first-order logic and showed how their resolution led to an elegant computational formalism. We consider each of these issues in turn.

The natural form constraint is a long-standing problem that this work addresses by stating that one representation is more "natural" than another by virtue of two properties. First, by minimizing the complexity of the computational procedure for mapping natural language into the representation language (NLP) and back out into natural language again (NLG), the representation language is more natural. In localizing the processing of potentially complex noun phrases into a structured variable representation, this is accomplished. Second, by requiring that sentences with similar grammatical structure map into similar representational structures processing is simplified. We have shown how the structure of the representation of natural language sentences reflects the structure of the language. This is particularly evident in the representation of sentences that, in FOPL, would be represented as rules. Such representations in ANALOG are "flat" and reflect the structure of all sentences of that type. Obviously, the more expressive the underlying representation language the easier the mapping. However, the problem then becomes how to use the more expressive representation to perform inference. What we have done, in this dissertation, is to motivate a change in one aspect of a KRR formalism based on FOPL, namely the status of variables as non-atomic terms. We have shown how this leads to a representation that satisfies the natural form constraint, and also formulated proof procedures for this representation. Moreover, we have illustrated the utility of these enhanced variables for NLP and NLG with specific examples of "natural" natural language processing.

The conceptual completeness constraint reflected an aspect of natural language, as it is used by people in discourse, namely the pervasiveness of re-use phenomena as illustrated by pronouns, reduced forms, and ellipsis. Standard formulations of FOPL could not felicitously reflect this sort of behavior, since terms of sentences are sentence-bound and cannot be re-used. In specifying structured variables as a "bundle" of information about a variable (containing quantificational and restriction information) we find terms are no longer sentence-bound. Moreover, in line with the natural form constraint, all constituents of the representation of a natural language sentence can be meaningfully associated with the original sentence. These properties allow the representations of sentences in discourse to reflect natural language re-use by structure-sharing of terms.

Quantifier scoping issues are more problematic. We have shown how, in this formalism, quantifier scopings associated with the branching quantifier sentence and the donkey sentence can be resolved. This was a useful result of the non-linearity of the representation language and the conceptual completeness of terms. However, in doing so, we noted that certain quantifier scopings (those corresponding to what we call "collective" scopings) cannot be represented in this formulation of our logic. We have argued that this is not a major problem in that the formalism can be readily extended to account for these quantifier scopings by the addition of a collection representation, but this needs to be done in future work.

The specification of subsumption and inference procedures in this dissertation represents a contribution to the formalization of these ideas in the context of a logic that does not distinguish between terminological and assertional components. In ANALOG, everything is terminological and the only assertional component is the assertion operator denoted by "!" or by the *Believe* metapredicate. This uniform representation leads to a formalism in which there is no conflict between reasoning in the terminological component and in the assertional component. Moreover, there is only one inference engine whose primary form of inference is instantiation.

Finally, we have shown how in addressing these concerns motivated by language, we can specify a computational formalism that accounts for these goals. We have illustrated this with a simple GATN grammar that processes potentially complex descriptions into a structured variable representation. The grammar illustrated the use of the implemented system in performing "natural" inference corresponding to description subsumption and inference.

7.3 Future Work

The work described here is not comprehensive. Several enhancements to both the formal system and its implementation are possible.

7.3.1 Operator Scoping

Because of the semantics of structured variables, namely that they are always widely scoped, there is a problem representing widely scoped operators. For example, a belief statement whose object was a proposition containing structured variables could not be represented. A specific example is:

$$\text{Believes}(\text{John}, \forall x\ (\text{raven}(x) \rightarrow \text{black}(x)))$$

where the intended reading is that John believes the proposition that ravens are black. This reading cannot be represented because the semantics of ANALOG specifies that the arbitrary raven scopes outside the belief operator. This is the non-opaque reading where the arbitrary raven is believed by John to be black. This make it difficult to state properties of generic propositions.

A solution to this is to allow some kind of closure (binding) arc from the proposition where the quantifiers are to the variables. This would augment the syntax and semantics of structured variables to allow depends arcs from universally quantified variables to proposition nodes. This would allows structured variables to scope inside operators such as belief. However, this makes the subsumption procedure more complex. Previously, when determining if a subsumption relationship exists between two universal structured variables, it was sufficient to check if the restrictions on one implied the other. If the dependencies are to propositions, then this determination is not straightforward. We are working on extending the syntax and semantics of ANALOG to deal with this case. ANALOG, without these special cases is a very general and useful formalism.

7.3.2 Collections

As previously discussed, the use of structured variables leads to scoping difficulties for sentences involving collections. The current system contains no facility for dealing with these sorts of quantifier scoping. However, the work here is entirely compatible with possible extensions to support reasoning about and using collections [Cho, 1992; Franconi, 1993]. The specification of the subsumption and matching procedures would have to be augmented to account for the use of collective terms. As an example, a straightforward extension of the subsumption procedure is to allow subsumption between a collection and any of its subcollections. Thus, if the system is told *John, Bill, and Mary went to the zoo*, it can answer the question *Did John go to the zoo?* by subsumption.

7.3.3 Language to Logic

The current generalized augmented network grammar (GATN) is rather specific to the particular natural language demonstrations shown in Chapter 6. The grammar is an augmented subset of the grammar of [Shapiro, 1982b; Shapiro, 1989]. While the portion of the grammar that parses complex noun phrases with restrictive relative clauses is rather robust, the capability of the system to process general natural language sentences is weak. As this general capability was not a primary goal of this dissertation, general, robust, natural language processing remains as a goal.

Most of the weakness in the GATN grammar lies in the small size of the dictionary used. These weaknesses are easily fixed by expanding the dictionary to include new terms. However, the GATN grammar is small and needs to be expanded to cover more English constructions. Based on the success of the limited grammar used in this dissertation, we feel the use of structured variables facilitates the NLP task by localizing the processing associated with complex natural language descriptions. The utility of this localization is particularly apparent with generation of natural language descriptions from structured variables. This portion of the grammar is small, but capable of generating complex descriptions.

7.3.4 Discourse Processing

One of the goals of this work was to model discourse phenomena involving use of reduced forms (such as pronoun use and ellipsis) in the representation language by structure-sharing. We do not, however, provide any well-motivated means to process connected text and determine when terms are being re-used; i. e., we have no discourse model. However, our prior work in generating felicitous descriptions [Haller and Ali, 1990] suggests that structured variables are a practical representation for discourse models that use focus lists.

Much work in discourse-level processing of language is compatible with this work [Kamp, 1984; Grosz and Sidner, 1986; Haller, 1994; Haller, 1993], since much of the processing associated with discourse modeling is at a higher level than the representation language. An interesting question associated with augmenting this work with a discourse model is whether it facilitates the task of maintaining a discourse model. In terms of the tripartite structure of [Grosz and Sidner, 1986], we provide two of their three components: the linguistic structure and the attentional structure. It is not clear, however, how to extend this work to account for their third component, the intentional structure of discourse. Haller [Haller, 1994] is extending the SNePS system (which this work augments) to account for this, arguing that this work will facilitate discourse modelling.

7.3.5 Distributed Inference

The current implementation of inference, while quite powerful, is largely subsumption inference-based. A node-based distributed inference engine such as that associated with the SNePS 2.1 system is lacking [Shapiro *et al.*, 1982; Hull, 1986; Choi and Shapiro, 1991; Choi and Shapiro, 1992]. Node-based distributed inference has the advantage that, in principle, it is realizable on parallel hardware. This was not a focus of this work, but it is entirely compatible with the formal system specified in this dissertation.

Appendix A

The ANALOG System

A.1 The ANALOG System

The logic, as described in chapter 5, has been fully implemented in the ANALOG propositional semantic network processing system. The natural language processing component consists of various GATN grammars and associated support files that are part of the standard distribution (see below).

A.1.1 Implementation Status

The knowledge representation and reasoning formalism has been implemented in the ANALOG propositional semantic network processing system. It is implemented as approximately 10,000 lines of Common Lisp and CLOS code. It runs under various Lisps, including various IBM PC Common Lisps.

A.1.2 The GATN Grammar

The grammar that implements the preceding demonstrations is a GATN grammar for a subset of English. This subset includes parsing for noun phrase-verb phrase sentences where the noun phrase subnetwork allows complex adjectival and restrictive relative clause complements. In addition "is" and "who" type questions can be processed. The generation portion of the grammar can generate English sentences corresponding to the results of the parsing process. Note that in the case of answers to questions, this may consist of multiple sentences.

The current grammar maintains no model of the discourse, and consequently cannot generate reduced forms of terms that have previously been generated. Thus, it can be somewhat wordy. The use of a discourse model was not a primary focus of this dissertation. However, since nodes are unique, keeping track of discourse item corresponds to placing them on the discourse stack/list associated with the model. Generating reduced forms of previously generated discourse items involves selecting those components of the discourse item that uniquely identify it with respect to the preceding discourse. For structured variables, this would mean examining the restriction set and picking some subset to generate. For sentence terms, a similar process can be used (since such terms are also nodes) to generate reduced forms such as "too". Note that the current grammar generates *all* identifying characteristics of a term.

A.1.3 Availability

The ANALOG system is available by anonymous ftp from `ftp.cs.buffalo.edu` and is copyright under the GNU CopyLeft.

Instructions for Obtaining ANALOG by ftp

The sequence of commands below illustrate how to obtain and install ANALOG. The text in italics is what you would enter. Note that ANALOG comes with minimal documentation.

```
[51][7:48pm]hydra/syali:  ftp ftp.cs.buffalo.edu
Connected to talos.cs.buffalo.edu.
220 talos.cs.Buffalo.EDU FTP server (SunOS 4.1) ready.
Name (ftp.cs.buffalo.edu:syali):  anonymous
331 Guest login ok, send ident as password.
```

```
Password:  user@where.ever.edu
230 Guest login ok, access restrictions apply.
ftp>  cd pub/analog
250 CWD command successful.
ftp>  binary
200 Type set to I.
ftp>  get analog-1.0.tar.gz
200 PORT command successful.
 ...
ftp>  quit
```

If you choose to use this software, please email the author a note to this effect. He can be reached at the email addresses: `syali@cs.buffalo.edu` or `ssa231f@csm560.smsu.edu`. Note that ANALOG comes with no support, although the author will try to help wherever possible.

Bibliography

[Ali and Shapiro, 1993] Syed S. Ali and Stuart C. Shapiro. Natural Language Processing Using a Propositional Semantic Network with Structured Variables. *Minds and Machines*, 3(4), November 1993. Special Issue on Knowledge Representation for Natural Language Processing.

[Ali, 1993a] Syed S. Ali. A Propositional Semantic Network with Structured Variables for Natural Language Processing. In *Proceedings of the Sixth Australian Joint Conference on Artificial Intelligence.*, November 17-19 1993.

[Ali, 1993b] Syed S. Ali. A Structured Representation for Noun Phrases and Anaphora. In *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society*, pages 197–202, Hillsdale, NJ, June 1993. Lawrence Erlbaum.

[Ali, 1993c] Syed S. Ali. Node Subsumption in a Propositional Semantic Network with Structured Variables. In *Proceedings of the Sixth Australian Joint Conference on Artificial Intelligence.*, November 17-19 1993.

[Barwise and Cooper, 1981] Jon Barwise and Robin Cooper. Generalized Quantifiers and Natural Language. *Linguistics and Philosophy*, 4:159–219, 1981.

[Barwise, 1979] Jon Barwise. On Branching Quantifiers in English. *J. Phil. Logic*, 8:47–80, 1979.

[Beierle *et al.*, 1992] C. Beierle, U. Hedtstuck, U. Pletat, P. H. Schmitt, and J. Siekmann. An Order-sorted Logic for Knowledge Representation Systems. *Artificial Intelligence*, 55(2-3):149–191, June 1992.

[Bobrow and Winograd, 1977] Daniel G. Bobrow and Terry Winograd. An Overview of KRL, a Knowledge Representation Language. *Cognitive Science*, 1(1):3–46, 1977.

[Brachman and Schmolze, 1985] Ronald J. Brachman and J. Schmolze. An Overview of the KL-ONE Knowledge Representation System. *Cognitive Science*, 9(2):171–216, 1985.

[Brachman *et al.*, 1983] Ronald J. Brachman, Richard E. Fikes, and Hector J. Levesque. KRYPTON: a Functional Approach to Knowledge Representation. *IEEE Computer*, 16(10):67–73, 1983.

[Brachman *et al.*, 1985] Ronald J. Brachman, Victoria Pigman Gilbert, and Hector J. Levesque. An Essential Hybrid Reasoning System: Knowledge and Symbol Level Accounts of KRYPTON. *Proceedings IJCAI-85*, 1:532–539, 1985.

[Brachman, 1979] Ronald J. Brachman. On the Epistemological Status of Semantic Networks. In N. V. Findler, editor, *Associative Networks: Representation and Use of Knowledge in Computers*. Academic Press, New York, 1979.

[Cercone *et al.*, 1992] Nick Cercone, Randy Goebel, John De Haan, and Stephanie Schaeffer. The ECO Family. *Computers and Mathematics with Applications*, 23(5):95–131, 1992. Special issue on Semantic Networks in Artificial Intelligence (Part 1).

[Cho, 1992] Sung-Hye Cho. Collections as Intensional Entities and Their Representations in a Semantic Network. In *Proceedings of the Second Pacific Rim International Conference on Artificial Intelligence*, pages 388–394, 1992.

[Cho, 1994] Sung-Hye Cho. *Representations of and Reasoning about Collections*. PhD thesis, State University of New York at Buffalo/Computer Science, Amherst, NY 14260, 1994. Forthcoming.

[Choi and Shapiro, 1991] Joongmin Choi and Stuart C. Shapiro. Experience-based deductive learning. In *Third International Conference on Tools for Artificial Intelligence TAI '91*, pages 502–503. IEEE Computer Society Press, Los Alamitos, CA, 1991.

[Choi and Shapiro, 1992] Joongmin Choi and Stuart C. Shapiro. Efficient implementation of non-standard connectives and quantifiers in deductive reasoning systems. In

Proceedings of the Twenty-Fifth Hawaii International Conference on System Sciences, pages 381–390. IEEE Computer Society Press, Los Alamitos, CA, 1992.

[Dowty *et al.*, 1981] David R. Dowty, Robert E. Wall, and Stanley Peters. *Introduction to Montague Semantics*. D. Reidel Publishing Co., Boston, 1981.

[Fahlman, 1979] Scott E. Fahlman. *NETL: A System for Representing and Using Real-World Knowledge*. MIT Press, Cambridge, MA, 1979.

[Fauconnier, 1975] G. Fauconnier. Do Quantifiers Branch. *Linguistic Inquiry*, 6(4):555–578, 1975.

[Fine, 1983] Kit Fine. A Defense of Arbitrary Objects. In *Proceedings of the Aristotelian Society*, volume supp. LVII, pages 55–77, 1983.

[Fine, 1985a] Kit Fine. Natural Deduction and Arbitrary Objects. *Journal of Philosophical Logic*, 14:57–107, 1985.

[Fine, 1985b] Kit Fine. *Reasoning with Arbitrary Objects*. Basil Blackwell, Oxford, 1985.

[Franconi, 1993] Enrico Franconi. A Treatment of Plurals and Plural Quantifications Based on a Theory of Collections. *Minds and Machines*, 3(4), November 1993. Special Issue on Knowledge Representation for Natural Language Processing. To appear.

[Geach, 1962] Peter Thomas Geach. *Reference and Generality*. Cornell University Press, Ithaca, New York, 1962.

[Givan *et al.*, 1991] Robert Givan, David A. McAllester, and Sameer Shalaby. Natural Language Based Inference Procedures Applied to Schubert's Steamroller. In *Proceedings of AAAI-91*, pages 915–920, 1991.

[Grosz and Sidner, 1986] B. J. Grosz and C. L. Sidner. Attention, intentions, and the structure of discourse. *Computational Linquistics*, 12:175–204, 1986.

[Haller and Ali, 1990] Susan M. Haller and Syed S. Ali. Using Focus for Generating Felicitous Locative Expressions. In *Proceedings of the Third International Conference on Industrial and Engineering Applications of Artificial Intelligence*, pages 472–477. ACM, 1990.

[Haller, 1993] S. M. Haller. Interactive generation of plan justifications. In *Proceedings of the Fourth European Workshop on Natural Language Generation*, pages 79–90, 1993.

[Haller, 1994] S. M. Haller. *Interactive Generation of Plan Descriptions and Justifications.* PhD thesis, Department of Computer Science, State University of New York at Buffalo, 1994. In progress.

[Halvorsen, 1986] Per-Kristian Halvorsen. Natural Language Understanding and Montague Grammar. *Computational Intelligence*, 2:54–62, 1986.

[Heim, 1990] Irene Heim. Discourse Representation Theory, 1990. Tutorial material from ACL-90.

[Hendrix, 1977] Gary G. Hendrix. Expanding the Utility of Semantic Networks through Partitioning. *Proc. 4th IJCAI*, 1977.

[Hendrix, 1979] Gary G. Hendrix. Encoding Knowledge in Partitioned Networks. In N. V. Findler, editor, *Associative Networks: The Representation and Use of Knowledge in Computers.*, pages 51–92. Academic Press, New York, 1979.

[Henkin, 1961] L. Henkin. Some Remarks on Infinitely Long Formulas. In *Infinitistic Methods*, pages 167–183. Pergamon Press, Oxford, 1961.

[Hobbs and Shieber, 1987] J. R. Hobbs and S. M. Shieber. An Algorithm for Generating Quantifier Scopings. *Computational Linguistics*, 13(1-2):47–63, 1987.

[Hull, 1986] R. G. Hull. A New Design for SNIP the SNePS Inference Package. SNeRG Technical Note 14, Department of Computer Science, SUNY at Buffalo, 1986.

[Kamp, 1984] Hans Kamp. A Theory of Truth and Semantic Representation. In Jeroen Groenendijk, Theo M. V. Janssen, and Martin Stokhof, editors, *Truth, Interpretation and Information*, pages 1–41. Forbis, Cinnaminson, 1984.

[Kumar *et al.*, 1988] Deepak Kumar, Syed Ali, and Stuart C. Shapiro. Discussing, using and recognizing plans in SNePS preliminary report—SNACTor: An acting system. In P V S Rao and P Sadanandan, editors, *Modern Trends in Information Technology:*

Proceedings of the Seventh Biennial Convention of South East Asia Regional Computer Confederation, pages 177–182. Tata McGraw-Hill, New Delhi, India, 1988.

[Levesque, 1986] Hector J. Levesque. Knowledge Representation and Reasoning. In Joseph F. Traub, editor, *Annual Review of Computer Science*, volume 1, pages 255–287. Annual Reviews Inc., Palo Alto, CA, 1986.

[Lloyd, 1987] J. W. Lloyd. *Foundations of Logic Programming.* Springer-Verlag, New York, 1987. 2nd Ed.

[MacGregor, 1988] Robert MacGregor. A deductive pattern matcher. In *Proceedings of AAAI-88, The National Conference on Artificial Intelligence*, pages 403–408, St. Paul, MINN, August 1988. AAAI.

[MacGregor, 1991] Robert MacGregor. Inside the LOOM description classifier. *SIGART Bulletin*, 2(3):88–92, June 1991.

[Maida and Shapiro, 1982] A. S. Maida and S. C. Shapiro. Intensional concepts in propositional semantic networks. *Cognitive Science*, 6(4):291–330, 1982. Reprinted in R. J. Brachman and H. J. Levesque, eds. *Readings in Knowledge Representation*, Morgan Kaufmann, Los Altos, CA, 1985, 170–189.

[McAllester and Givan, 1992] David A. McAllester and Robert Givan. Natural Language Syntax and First-order Inference. *Artificial Intelligence*, 56(1):1–20, 1992.

[McAllester, 1989] David A. McAllester. *Ontic: A Knowledge Representation System for Mathematics.* MIT Press, Cambridge, MA, 1989.

[McCarthy, 1980] John McCarthy. Circumscription—A Form of Non-monotonic Reasoning. *Artificial Intelligence*, 13(1,2):27–39, 1980.

[McDermott and Doyle, 1980] D. McDermott and D. Doyle. Nonmonotic Logic. *Artificial Intelligence*, 13(1,2):41–72, 1980.

[Montague, 1973] Richard Montague. The Proper Treatment of Quantification in Ordinary English. In J. Hintikka, J. Moravcsik, and P. Suppes, editors, *Approaches to Natural Language*, pages 221–242. Reidel, Dordrecht, 1973. Also in R. Montague, 1974, *Formal Philosophy: Selected Papers of Richard Montague*, ed. by Richard Thomason, New Haven: Yale University Press.

[Montague, 1974] Richard Montague. English as a Formal Language. In Richmond H. Thomason, editor, *Formal Philosophy*, pages 188–221. Yale University Press, 1974.

[Peltason, 1991] Christof Peltason. The back system – an overview. *SIGART Bulletin*, 2(3):114–119, June 1991.

[Purdy, 1991a] William C. Purdy. A Logic for Natural Language. *Notre Dame Journal of Formal Logic*, 32(3):409–425, 1991.

[Purdy, 1991b] William C. Purdy. Surface Reasoning. *Notre Dame Journal of Formal Logic*, 33(1):13–36, 1991.

[Quine, 1969] W. V. Quine. *Ontological Relativity and Other Essays.* Columbia University Press, London and New York, 1969.

[Quine, 1970] W. V. Quine. *Philosophy of Logic.* Prentice-Hall, Englewood Cliffs, NJ, 1970.

[Rapaport, 1988] W. J. Rapaport. Syntactic semantics: Foundations of computational natural-language understanding. In J. Fetzer, editor, *Aspects of Artificial Intelligence*, pages 81–131. Kluwer Academic Publishers, Dordrecht, Holland, 1988.

[Rapaport, 1991] William J. Rapaport. Predication, fiction, and artificial intelligence. *Topoi*, 10:79–111, 1991.

[Reiter, 1980] R. Reiter. A Logic for Default Reasoning. *Artificial Intelligence*, 13(1,2):81–132, 1980.

[Rich, 1991] Charles Rich, editor. *Special Issue on Inplemented Knowledge Representation and Reasoning Systems*, volume 2. ACM Press, June 1991. SIGART Bulletin.

[Russell, 1920] Bertrand Russell. *Introduction to Mathematical Philosophy.* Macmillan, New York, 1920.

[Schank and Colby, 1973] Roger C. Schank and K. Colby. *Computer Models of Thought and Language.* Freeman, San Francisco, 1973.

[Schank, 1972] Roger C. Schank. Conceptual Dependency: A Theory of Natural Language Understanding. *Cognitive Psychology*, 3(4), 1972.

[Schubert and Pelletier, 1982] L. K. Schubert and F. J. Pelletier. From English to Logic: Context-free Computation of Conventional Logical Translation. *American Journal of Computational Linguistics*, 8:165–176, 1982. Reprinted (with corrections) in B. J. Grosz, K. Sparck-Jones and B. L. Webber (eds.), *Readings in Natural Language Processing*, 293–311, Morgan Kaufman, 1986.

[Schubert *et al.*, 1979] Lenhart K. Schubert, Randolph G. Goebel, and Nicholas J. Cercone. The Structure and Organization of a Semantic Net for Comprehension and Inference. In N. V. Findler, editor, *Associative Networks: Representation and Use of Knowledge in Computers*, pages 121–175. Academic Press, New York, 1979.

[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*, pages 263–315. Springer–Verlag, New York, 1987.

[Shapiro and Rapaport, 1991] Stuart C. Shapiro and William J. Rapaport. Models and minds: Knowledge representation for natural-language competence. In Robert Cummins and John Pollock, editors, *Philosophy and AI: Essays at the Interface*, pages 215–259. MIT Press, Cambridge, MA, 1991.

[Shapiro and Rapaport, 1992] S. C. Shapiro and William J. Rapaport. The SNePS Family. *Computers and Mathematics with Applications*, 23(5):243–275, 1992. Special issue on Semantic Networks in Artificial Intelligence (Part 1).

[Shapiro *et al.*, 1982] S. C. Shapiro, J. Martins, and D. McKay. Bi-directional inference. In *Proceedings of the Fourth Annual Conference of the Cognitive Science Society*, pages 90–93, Ann Arbor, MI, 1982. the Program in Cognitive Science of The University of Chicago and The University of Michigan.

[Shapiro *et al.*, 1989] S. C. Shapiro, D. Kumar, and S. Ali. A propositional network approach to plans and plan recognition. In *Proceedings of the 1988 Workshop on Plan Recognition*, page 21, Los Altos, CA, 1989. Morgan Kaufmann.

[Shapiro *et al.*, 1993] S. C. Shapiro, W. J. Rapaport, and the SNePS Research Group. A Dictionary of Case Frames, 1993. In preparation.

[Shapiro, 1979] S. C. Shapiro. The SNePS semantic network processing system. In N. V. Findler, editor, *Associative Networks: The Representation and Use of Knowledge by Computers*, pages 179–203. Academic Press, New York, 1979.

[Shapiro, 1980] S. C. Shapiro. Review of Fahlman, Scott NETL: A system for representing and using real-world knowledge. *American Journal of Computational Linguistics*, 6(3):183–186, 1980.

[Shapiro, 1982a] S. C. Shapiro. Generalized Augmented Transition Network Grammars for Generation from Semantic Networks. *The American Journal of Computational Linguistics*, 8(1):12–25, 1982.

[Shapiro, 1982b] S. C. Shapiro. Generalized augmented transition network grammars for generation from semantic networks. *The American Journal of Computational Linguistics*, 8(1):12–25, 1982.

[Shapiro, 1986] S. C. Shapiro. Symmetric relations, intensional individuals, and variable binding. *Proceedings of the IEEE*, 74(10):1354–1363, 1986.

[Shapiro, 1989] S. C. Shapiro. The CASSIE projects: An approach to natural language competence. In *Proceedings of the 4th Portugese Conference on Artificial Intelligence*, pages 362–380, Lisbon, Portugal, 1989. Springer-Verlag.

[Shapiro, 1991] S. C. Shapiro. Cables, Paths, and "Subconscious" Reasoning in Propositional Semantic Networks. In John F. Sowa, editor, *Principles of Semantic Networks*, pages 137–156. Morgan Kaufmann, 1991.

[Sommers, 1982] Frederic Tamler Sommers. *The Logic of Natural Language*. Oxford University Press, New York, 1982.

[Sowa, 1984] John F. Sowa. *Conceptual Structures.* Addison-Wesley, 1984.

[Sowa, 1992] John F. Sowa. Conceptual Graphs as a Universal Knowledge Representation. *Computers and Mathematics with Applications*, 23(5):75–93, 1992. Special issue on Semantic Networks in Artificial Intelligence (Part 1).

[Webber, 1983] Bonnie Lynn Webber. So What Can We Talk About Now? In Michael Brady and Robert C. Berwick, editors, *Computational Models of Discourse*, pages 331–371. MIT Press, 1983.

[Woods and Schmolze, 1992] William A. Woods and James G. Schmolze. The KL-ONE Family. *Computers and Mathematics with Applications*, 23(5):133–177, 1992. Special issue on Semantic Networks in Artificial Intelligence (Part 1).

[Woods, 1970] W. A. Woods. Transition network grammars for natural language analysis. *Communications of the ACM*, 13(10):591–606, 1970.

[Woods, 1978] W. A. Woods. *Semantics and Quantification in Natural Language Question Answering*, volume 17. Academic Press, New York, 1978.

[Woods, 1991] William A. Woods. Understanding Subsumption and Taxonomy: A Framework for Progress. In John F. Sowa, editor, *Principles of Semantic Networks*, pages 45–94. Morgan Kaufmann, 1991.