Inheritance Reasoning and Head-Driven Phrase Structure Grammar

by

Carl M. Vogel

B.S. (Honors)
Loyola University
New Orleans, Louisiana, 1988

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
in the School
of
Computing Science

SIMON FRASER UNIVERSITY
December 1990

Approval

Name: Carl M. Vogel

Degree: Master of Science

Title of Thesis: Inheritance Reasoning and Head-Driven Phrase Structure Grammar

Examining Commitee:

Dr. Veronica Dahl, Chairman

Dr. Nicholas J. Cercone
Co-Senior Supervisor

Dr. Frederick P. Popowich
Co-Senior Supervisor

Dr. Robert F. Hadley
Supervisor

Dr. Randolph Goebel
Department of Computing Science
University of Alberta
External Examiner

December 10, 1990
Date Approved

PARTIAL COPYRIGHT LICENSE

I hereby grant to Simon Fraser University the right to lend my thesis, project or extended essay (the title of which is shown below) to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its users. I further agree that permission for multiple copying of this work for scholarly purposes may be granted by me or the Dean of Graduate Studies. It is understood that copying or publication of this work for financial gain shall not be allowed without my written permission.

Title of Thesis/Project/Extended Essay

Inheritance Reasoning and Head-Driven Phrase Structure Grammars.

Author: _____
(signature)

Carl M. Vogel
(name)

December 14, 1990
(date)

Abstract

Inheritance networks are a type of semantic network which represent both strict (classical implication) and defeasible (non-classical) relationships among entities. We present an established approach to defeasible reasoning which defines inference in terms of the construction of paths through a network. Much of literature on inheritance is concerned with specifying the most "intuitive" system of path construction. However, when considering a fundamental feature of these approaches—the status accorded to redundant links—we find that topological considerations espoused in the literature are insufficient for determining the valid inferences of a network. This implies that the "intuitiveness" of a particular method depends upon the domain being represented. Though Touretzky has demonstrated that it is unsound in some cases, the path-preference algorithm known as *shortest path reasoning*, is actually the most intuitive algorithm to use when reasoning about the inheritance network which represents most of the conceptual structure of Head-Driven Phrase Structure Grammar (HPSG). In this thesis we describe the HPSG formalism and detail the inheritance hierarchy which we abstract from it. The network itself is interesting because it is cyclic and because it contains *supernodes*. We specify the content of nodes (information structures encoded as attribute-value matrices) and the interpretation of links (the relative pseudocomplement relation) in the resulting inheritance hierarchy. The process of reasoning over the hierarchy is demonstrated, and the implications of this work for researchers in both unification grammars and inheritance reasoners are discussed. In particular, when it is applied to the inheritance network for HPSG, an inheritance reasoner functions as a parser for the grammar formalism. To the inheritance reasoning researcher this provides a semantically nontrivial application for representation using inheritance networks against which arguments about the intuitiveness of more complex path construction algorithms may be tested.

"One fine morning in May, a slim young horsewoman might have been seen riding a glossy sorrel mare along the avenues of the Bois, amongst the flowers."

Albert Camus
The Plague

Acknowledgements

I am grateful to my family for spoiling me from the beginning. Nick, Fred, and Bob have each in different ways been kind enough to keep me spoiled. They have helped me out of numerous predicaments and given me fantastic opportunities. They and a lot of other folks, especially Sue, Jane, Tammy, Glenn, Pierre, and Dan (who have also helped me out of some pretty tough spots) have all made this a superlatively fun time. Thank you, all. I will never see my or any other saxophone without smiling and thinking of you.

Thanks also to the Jeep for keeping me on my feet.

Table of Contents

List of Figures

Chapter 1
Introduction

1.1. Background

Inheritance based reasoning is an approach to reasoning about default and non-default knowledge. *Default knowledge* is a computationalist category of knowledge about the world, but underlying this categorization is a very old classification scheme. Aristotle distinguished between the primary and accidental aspects of being, a distinction which corresponds to that between non-default and default knowledge.

> "'Primary Being' may mean: (1) a simple body, such as earth, fire, water, and everything of this sort; and in general bodies and the bodies composed of them, both animals and superior beings, as well as their parts. But all these are called 'primary being,' because they are not attributed to something else; whereas other things are said of them.... (3) It may mean whatever is intrinsic to primary being in the first sense, limiting them and marking them as a this-something, or whatever when destroyed destroys such a primary being" (Aristotle, 1952, p.99).

Aristotle calls the other aspects of being, those which are predicated, and which possibly change over time, the "accidental" properties of being. Accidental properties, which correspond to defaults, include "facts" of the form, "Birds fly," or "Elephants are gray," which are facts inasmuch as people will assert them to be true, even though many counterexamples are available. Touretzky (1986, p.6) calls such sentences *normative*: "Normative statements are statements that are usually true or that can be assumed to be true in the absence of contrary information." Because counterexamples are available, defaults cannot be represented completely by universally quantified formulae in first order logic (FOL; or any logic, for that matter). Furthermore, Aristotle claims, "In view of these many ways of being, we must first consider the accidental and point out that there can be no theory of it. Witness, no practical science, no art, no theoretical science troubles itself about it" (Aristotle, 1952, p.125). However, on that point he is no longer correct, for that is exactly what research in nonmonotonic reasoning is attempting to devise: well motivated approaches to representing and reasoning about the accidental properties of being.

One approach to default reasoning is in the application of nonmonotonic systems based on

classical logics (McDermott and Doyle, 1980). In this approach, an operator M, meaning "is consistent," is added to first order logic. So, "Elephants are gray," is translated to, "All elephants that are not known to be not gray are gray." This statement is represented in the system as: (x)(elephant(x) $\wedge$ M(gray(x)) -> gray(x). McDermott (1982) continues work in this vein by including the axioms of the modal logic S4 and S5 as axioms in the nonmonotonic system. Moore (1985) defines an autoepistemic logic with an operator L, essentially the dual of McDermott and Doyle's M, which translates to, "is believed." This logic, he claims, captures the intuitions of nonmonotonic reasoning and corresponds to a weak version of S5[1]. Delgrande's (1990) approach works similarly, though he adds a conditional operator, =>, whose semantics is based on possible worlds rather than a fixed point construction. In Delgrande's system, "elephants are gray" means that under the least exceptional circumstances, if an individual is an elephant then that individual is gray. Reiter (1983) offers yet another approach by adding default rules of inference to achieve nonmonotonicity rather than by adding a nonmonotonic operator. However, Konolige (1987) demonstrates that this formalism is equivalent to autoepistemic logic, hence by Moore's (1985) argument, Reiter's system is equivalent to the nonmonotonic logic that McDermott and Doyle had set out to create. All of these systems are classified as non-classical logics in the sense that they utilize operations for forming sentences which are not truth functionally compositional.

1.2. Inheritance Reasoning

Inheritance reasoning is an alternative approach to using default logics which experiments with non-classical systems whose syntax is suggested by the pictorial representation of complex hierarchies as directed acyclic graphs. Inheritance networks descend from work on the notation and formal semantics of semantic networks (shown by Schubert (1975) and Schubert, et al. (1979) to have the full semantics of FOL). In the network approach to knowledge representation, concepts are represented as nodes in a network. Networks are compositional: a node in a network can be some other network, and the same subnetwork can be a subnetwork of several larger supernetworks, simultaneously. The supernetwork/subnetwork relationship can exist without the supernetwork possessing a copy of the subnetwork. Instead, the supernetwork can contain a pointer to the subnetwork (or, a *virtual copy* (Fahlman, 1979)). The

[1]It lacks the axiom schema, LP -> P.

supernetwork/subnetwork relationship is also termed *structure sharing*. When a network is defined through structure sharing, that network *inherits* the information contained in the subnetwork, but not a copy of the subnetwork. This relationship is somewhat transitive. The information from the subnetwork is inherited, by default, to all other larger networks which inherit some of their structure from the composite network that made reference to the subnetwork.[2] But, the relationship is not fully transitive, since the inheritance can be explicitly cancelled or overridden by less direct means. A network used to denote the structure sharing in a semantic network is known as an inheritance hierarchy or as an inheritance network. Thus, an inheritance network is a schematic representation of a semantic network.

The nodes of an inheritance network denote concepts defined by possibly complex connections of nodes and links in some semantic network, and the links in an inheritance network indicate the structure of information sharing. In an inheritance network consisting of two nodes connected by a link, the node pointed to is the supernetwork, and the node from which the link emanates represents the subnetwork. The link indicates that some of the information from the subnetwork is inherited to the supernetwork. Some researchers distinguish between *strict* and *non-strict* inheritance links, essentially defining strict links as those for which all of the information in a subnet is inherited and transitivity over chains always holds. Strict links model non-default knowledge and non-strict links model defaults. Because non-strict inheritance links are the more interesting case, we focus on those links and refer to them in the second chapter of this thesis simply as "links" unless the reference would cause ambiguity. Later in the thesis we will apply both strict and non-strict links. Inheritance reasoning is the process of determining among potentially many chains of inheritance links which paths should be cancelled or overridden. All paths of inheritance links represent "true" relationships, but because paths can be cancelled or overridden by exceptions expressed in more specific paths they are termed, "defeasible". A primary concern of the literature on inheritance reasoning is the determination of the "best" method for defining the preferred chains of links through an inheritance network. As in formal logics, we call the construction of preferred paths "inference" and we refer to the

---

[2]Admittedly, it is anomalous to use the phrase "inherit to" rather than something usually used to express the act of inheriting, like "bequeath". Horty et al. (1990) consider a related issue:

> Once one adopts the bottom-up approach, the terminology of "inheritance" is no longer so appropriate; but the terminology has become fixed, and it would introduce more confusion than it would eliminate if we tried to characterize this kind of reasoning process in a phrase more neutral between the top-down and the bottom-up views. (p. 6)

Likewise, we also keep the original language of inheritance.

definitions of path preference in an inheritance system as that system's "proof theory". Unfortunately, however, a semantic theory which unifies the various approaches to the proof theory of inheritance reasoning has yet to be developed (Brachman, 1983, Brachman, 1985, Touretzky, et al., 1987, Boutilier, 1989, Dorosh and Loui, 1989, Delgrande, 1990).

Links in an inheritance network notation can be used to represent the natural language qualification, "typically" as in the sentence, "Typically, humans have two legs" (cf. "Network AB typically contains information contained in network A"). A main reason for adopting the network approach over the traditional work in formal logic stems from the conceptual freedom created by its distinctive graphical syntax. In particular, within the network notation the presence of representations for two contradictory facts, "Birds fly," and, "Birds do not fly," do not ground the proof of any arbitrary fact at all, as the propositions would entail if encoded in the deductive closure of usual logical calculi. Instead, the network notation localizes the disruption caused by logical inconsistency to an incongruity on the fact under consideration. But, because the network approach is so young it has had to spend a great deal of time outlining "proof procedures," the exact methods of path construction where paths correspond to valid inferences on the networks. Arguments for particular methods of specifying valid paths are based upon intuitions and computational complexity (Touretzky, et al., 1987, Sandewall, 1986, Touretzky, 1986, Ballim, et al., 1989).

1.3. The Structure of this Thesis

1.3.1. Extant Systems

In the second chapter of this thesis we give a more formal articulation of inheritance reasoning, including a characterization of some of the more controversial issues which emerge in the literature. The particular family of inheritance network approaches (affectionately) called THAT family for Touretzky, Horty, And Thomason is paradigmatic of the field, and in the second chapter we give a formal specification of THAT approach. By giving formal definitions and providing examples of their application, we demonstrate the syntactic, topology-based reasoning inherent in THAT paradigm. We also describe a closely related system that is defined by Geffner and Verma (1989), as well as a more distantly related approach presented by Ballim et al. (1989). These three specifications of inheritance reasoning do not exhaust the varieties of

reasoners that have been defined in the literature. Horty (1989) identifies 72 different systems for path-based inheritance reasoning, alone. However, the three systems that we discuss do characterize the field by defining the problem and indicating some of the variety that is possible. All of these systems are topologically based. Arguments against the appropriateness of some system often focus upon anomalies which arise when that method is applied to particular examples. The structure of such arguments is to show that for networks of a particular topology, the method will reach a certain conclusion, then an interpretation of the network is provided in which the conclusion achieved seems anomalous. Some thought on these arguments, particularly about their topological nature, leads to doubt about a fundamental tenet of all three approaches detailed within this thesis and of the literature as a whole. The assumption which we reject is that reasoning over a network should not be confounded by the presence of redundant links. We reject this assumption because it is not clear that the "redundancy" of a link is a topological feature of an inheritance network.

1.3.2. Constructive Criticism

Our argument about the status of topologically redundant links is presented in the second chapter of the thesis. Basically, we claim that since a non-strict inheritance link emanating from a node in an inheritance network does not entail the inheritance of all the information contained in that node, there is potentially other information left to be inherited through an additional link. Topologically, any such additional link which converges with a path containing the other link, is considered redundant. But, topologically redundant links are not necessarily redundant with respect to the information inherited. The status of topologically redundant links is significant because they form the basis of Touretzky's (1986) counterexample to the soundness of shortest path reasoning (Fahlman, 1979). If topologically redundant links are not necessarily semantically redundant, then we should give serious reconsideration to shortest path reasoning, because the computational complexity of shortest path reasoning is a linear function of the number of nodes in the directed acyclic graph, while Touretzky's system, which makes provisions for topologically redundant links, has been shown to be NP-Hard (Selman and Levesque, 1989).

1.3.3. An Application

In this light, we turn to the application of inheritance reasoning to linguistic analysis. Thomason (1989) has emphasized the need in this stage of research in inheritance reasoning to address its application. Extant arguments in the literature in favor of one or another approach to path based inheritance primarily focus on particular example networks. But, because the examples are founded upon trivial[3] concept hierarchies it is difficult to comment effectively on the appropriateness of any particular method. We devote the second half of this thesis to providing a significant application, a non-trivial concept hierarchy in which shortest path reasoning is quite useful. The application is in the analysis of language within the framework of Head-Driven Phrase Structure Grammars (HPSG) (Pollard and Sag, 1987), using shortest path inheritance. This is a well motivated application for two main reasons. Many papers have been written about parsing as deduction ((Menzel, 1987), (van der Linden, 1989), and (König, 1989) are just three of them); deduction and inheritance reasoning are both forms of reasoning, so it is promising to consider parsing as inheritance reasoning, and moreover, there may be computational or other methodological advantages to using inheritance reasoners as parsing mechanisms. Another consideration is that inheritance reasoners are built for doing default inference. That is, they intend to represent the generality of statements like, "birds fly," without resolving to an inconsistency given a specific non-flying bird. Similarly for parsing, we have an intuition that *rules* of grammar exist and also the observation that grammar rules have exceptions. *Ergo*, inheritance reasoners are ideal parsers.

Although defaults provide an initial motivation for using inheritance reasoning to parse sentences because of defaults' implicit representation of exceptions, in our application of inheritance reasoning we use default links whose interpretation is weaker than typicality. The interpretation of non-strict links is possibility. Thus, instead of encoding, "Sentences *typically* have a subject followed by a verb and object," we encode, "A noun phrase *can* have a determiner followed by a noun." An unfortunate result of the weaker interpretation is that exceptions must be encoded explicitly. Essentially, this means that non-strict links are disjunctive. But, both sorts of links still represent the classification of concepts, and information is inherited across links. Links in our system are further characterized and differentiated from traditional work on inheritance in Chapter Four.

---

[3]The examples are not especially trivial in topology but in subject matter.

We provide a brief introduction to HPSG in the third chapter of this thesis. HPSG is a frame-based language for describing linguistic phenomena. Like similar unification grammar formalisms, HPSG is distinguished from traditional phrase structure grammar formalisms by its extremely lexical orientation and its de-emphasis of grammar rules. Lexical entries encode constraints (i.e., major category, form, subcategorization, etc.) associated with words and their combination into more complex phrase structures. Objects that take the place of grammar rules are also encoded within the same formal language, as are "universal" principles of grammar. Grammar rules in HPSG are more schematic and, thus, fewer in number than traditional phrase structure rules. Rather than providing a phrase structure rule for each part of speech (i.e., S -> NP VP, NP -> Det N, VP -> V Adv, etc.), HPSG provides rules for different sorts of *headed structures*. For example, one kind of headed structure has a head daughter which is preceded by its complement (i.e., a noun, the head daughter of a noun phrase, is preceded by a determiner which is the head daughter's complement; a verb phrase, the head daughter of a sentence, is preceded by its subject which the head daughter's complement). In the third chapter of the thesis, we show the structure of the formalism provided by HPSG in more detail, and we show how it is used to describe linguistic phenomena.

We then abstract from HPSG a network of concepts which we call the *operative hierarchy* of HPSG. Discussion already exists about the lexical hierarchy built into HPSG (Pollard and Sag, 1987, Flickinger, 1987). But we take this network as a definitional hierarchy, one which is used to define the concepts present as nodes in the operative hierarchy. Inheritance reasoning over the operative hierarchy constructs analyses of linguistic objects in the frame based language of HPSG. This observation is related to research presented by Steel and De Smedt (1983) and by Brachman and Schmolze (1985). In Chapter Four we explain our analysis and demonstrate how it is more complete than the previous, related analyses. Our explanation details the contents of nodes and the interpretation of the links (a specialized sort of structure sharing relationship). We present a picture of the overall network. This network is topologically interesting because it contains a cycle, and because we allow inheritance links to point to the interiors of nodes (a node pointed into in this way is referred to as a *supernode*). We give an illustrative example of the process of reasoning over this network. Shortest path reasoning seems to be the most appropriate form of reasoning to use. In Chapter Five also provide a Prolog implementation of the operative hierarchy and a shortest path reasoner for reasoning over the inheritance network. Since the reasoner constructs HPSG analyses of linguistic objects, this reasoner constitutes an HPSG parser.

1.3.4. Extensions

One advantage of our analysis of HPSG from the point of view of inheritance reasoning is that it leads to a principled treatment of a particular class of ill-formed input to a natural language processor (cf. Fass, et al. (1990), Fass and Hall (1990)). This class is one in which a user has true beliefs whose representation is missing from the system. When a user correctly uses a word which is unknown to a system lexicon, we can, through inheritance, determine some of the information about the word that is missing. To achieve this we add an additional node and link to the operative hierarchy. The fifth chapter of this thesis describes that extension, and details the corresponding extension to the implementation. We discuss some of the limitations of this approach as well. Theoretically, it is quite a restrictive assumption to hold that all system unknown words are used correctly, though we feel that this assumption is a psychologically valid one to make (the assumption is consistent with the *maxim of quality* (Grice, 1975)). Practically, the extension adds complexity to the reasoning process. Observation of our experiments lead us to conclude that by avoiding chronological backtracking, constraint based reasoning could provide a more practical framework than inheritance reasoning from which to explore this problem.

1.3.5. Reflections

Finally, we conclude the thesis by summarizing its contributions, and we describe directions for further research in this area. Our analysis of the status of topologically redundant links argues for resurrecting the shortest path reasoner. Our application of this reasoner to Head-Driven Phrase Structure Grammars provides useful insights into problems of robust parsing. This leads us to try similar applications for other unification grammar formalisms and hints at a potentially better articulation of the problem in the language of constraint based reasoning. In a different direction, our analysis of topologically redundant links points us towards considering the semantics of inheritance systems, with some concrete ideas about semantic foundations in which the redundancy of links is clearly a non-topological issue.

Chapter 2
Path Based Inheritance

In this chapter we present a detailed description of inheritance reasoning. We demonstrate the motivations behind the inheritance-based approach to knowledge representation and present samples from the literature. Our purpose is to give the reader a feel for the discussion that dominates the literature: essentially, the discussion is of proof-theoretic issues about the structure of valid inferences. To this end we present the proof theory of a dominant approach in the field, that defined by Horty, et al. (1987), and in light of this presentation we also describe two related systems (Geffner and Verma, 1989, Ballim, et al., 1989). Throughout, we discuss advantages and limitations of path based inheritance reasoning.

2.1. THAT Family Inheritance

In THAT family of inheritance networks, *nodes* represent individuals, concepts, and properties, and *links* represent the classification of connected nodes. A link between two nodes in the form A--->B denotes the fact that As are typically classified as Bs ("As are Bs"). Another sort of link, one with a slash through it (-/->), is a negative link. The negative link A-/->B indicates that As are not typically classified classified as Bs. The example hierarchy depicted in Figure 2-1 contains only one negative link, the link which connects the node labeled, "Royal Elephant," to the node labeled, "Gray Thing." The other links are positive links and assert positive typicalities. An explanatory paraphrase of the interpretation intended by Touretzky for Figure 2-1 is as follows: elephants are typically gray, royal elephants are not typically gray, royal elephants are typically elephants,.... The goal of an inheritance reasoner presented with a network of such facts, is to determine what additional facts are implicit in the network, in answer to questions such as, "Is Clyde gray?" We determine whether Clyde is gray by finding a valid path between the nodes for Clyde and Gray Thing. A path is made up of a chain of links, but not all chains of links in an inheritance network constitute valid paths. An inheritance reasoner is the set of definitions which specify the method of construction of valid paths through a network.

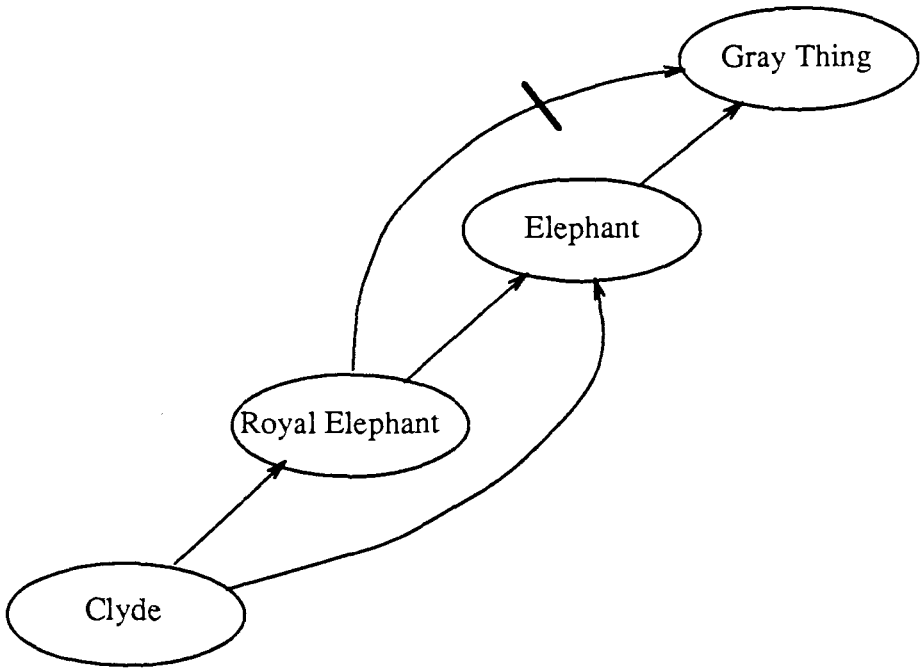


Figure 2-1: The Quintessential Inheritance Hierarchy.

2.1.1. Paths

An expansion of a network, which corresponds to the deductive closure of a theory stated in a logic or an extension of a default system, is a set of permitted (valid) *path* through the network. This set of paths is defined straightforwardly by induction, with the assumption that all paths and networks will be acyclic. Individuals (as opposed to concepts or classes) may appear only in the first node of a path, but they do not necessarily have to occur in the path at all. Any link in a network is a path; if the link is of the form, p--->r, then it is a *positive* path, and if it is of the form p-/->r, then it is a *negative* path. Since we assume the network to be acyclic, every path will have both a first node and a last node. Let π vary over positive paths; *LastNode*(π) denotes the last node in path π, and *FirstNode*(π) denotes the first node in path π. The *length* of a path is the number of links that it contains. For a given network, if π is a positive path, *LastNode*(π)--->r is a link, and r does not occur as a node in π, then π--->r is a positive path as well. Symmetrically, if *LastNode*(π)-/->r is a link contained in the network and r does not occur in π,

then π-/->r is a negative path. Since we assume π to vary over only positive paths, this means that negative links can occur only at the end of a path. The *polarity* of a path is determined by its last link: a path whose last link is negative is called a negative path, and all other paths are positive. An alternative convention used when paths grow cumbrously long is to write paths as strings of nodes; "πr" and "π/r" are abbreviations for the aforementioned paths. Sometimes we use the metanotation -#-> to indicate a link whose polarity is unspecified.

The above definition of paths is not contested in the literature, though Geffner and Verma (1989) do release the restriction against cyclic paths (the "occurs" check) and manage to prove that in the case of negative cycles no complications are introduced which confound their definitions of path construction. A negative cycle is something like p--->q--->r-/->p; the cycle occurs using the single negative link allowed in the path. Since a negative link can occur only at the end of a path, a path containing a negative cycle will still have a last node. Though we can can construct an endless chain of links because of the cycle, we cannot construct a *path* that has a link after the negative link. Negative cycles are less problematic than arbitrary cycles because they do not lead to paths of infinite length.

Figure 2-2 depicts a network containing the chain of links, p--->q-/->r-/->s. This full

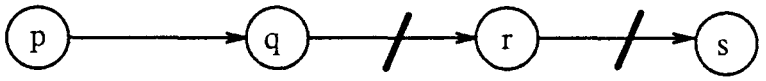


Figure 2-2: A Network with a Generalized Path of Length Three.

chain of links is not a path since it contains two negative links. The paths contained in the network are in the set {pq, q/r, r/s, pq/r}. No *path* connects p or q to s; hence from this graph we can draw no conclusions about p or q in relation to s. A *generalized path* is defined as any succession of links—this is a generalization of the definition of paths in which π may vary over positive or negative paths. Thus, pq/r/s is a generalized path contained in Figure 2-2. Any chain of links through a network qualifies as a generalized path. The *degree* of a path in a specific network is the number of links in the longest generalized path whose endpoints coincide with the path in question (Horty, et al., 1990, p.13). Note that the degree of a path can be greater than its length. For example, a path consisting of a single link (length equal to one) can have a degree of

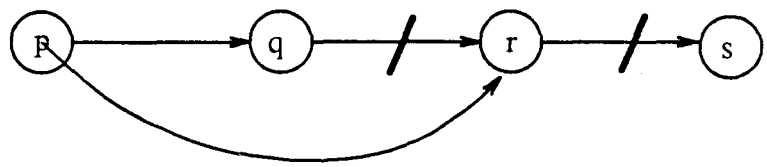


Figure 2-3: The Link p--->r Has a Degree of Two.

more than one. This situation is illustrated in Figure 2-3; the path *pr* has a length of one, but a degree of two, because the path pq/r is the longest generalized path connecting the two endpoints p and r. Another useful fact about the degree of a path of length m is that the path will be strictly greater than the degree of the subpath that contains the path's first m-1 links, as shown in Theorem 1.

Theorem 1: If π is a path of the form α-#->z having degree n, then α is a positive path with degree less than n.

Proof:

1. Regardless of the polarity of the link from *LastNode*(α) to z, no other link in π can be a negative link, or π would not be a path. Thus, α is a positive path.
2. To show that the degree of α is less than n:
   a. Let a = *FirstNode*(α). By the definition of degree, n is the length of the longest generalized path π', between the endpoints a and z of π.
   b. Let τ be the longest generalized path between the endpoints of α. The degree of α is the length of τ. Then τ-#->z constitutes a generalized path between a and z.
   c. If the length of τ (the degree of α) is greater than or equal to n, then τ-#->z has length of at least n + 1. But, this means that the degree of τ-#->z, hence the degree of α-#->z is at least n + 1. This contradicts our assumption that the degree of α-#->z is n. Therefore, the length of τ (the degree of α) must be less than n.

Introducing the concept of a path's degree also introduces the possibility that there can be more than one path between two nodes. Part of the task of an inheritance system is to define procedures for adjudicating among several possibly conflicting paths between two nodes. Further definitions impose the restrictions which allow choices to be made. The chosen paths are *permitted.*

2.1.2. Permitted Paths

Permitted paths are determined by the definitions of a particular reasoning system; different reasoners sanction different paths as permitted. Reasoners can be skeptical or credulous in regard to conflicting paths, upwards or downwards in direction of processing, or any member of a host of different dimensions to defining path permission (Touretzky, et al., 1987). The systems discussed in this thesis have in common the stipulation that all direct links contained in a network are sanctioned as permitted paths. Differences emerge with respect to paths of more than one link, also called *compound paths*, that conflict with other paths. According to the upwards, decoupled, restricted skeptical reasoner of Horty, et al. (1990), a path is *permitted* unless it is preempted, cancelled, or redundant with respect to a path that is not permitted. If a path is permitted in a network, then we say that the expansion of the network contains the *implicit link* between the endpoints of the path. By defining permitted paths, preemption, and cancellation in the language of FOL, instead of adopting the network notation of Horty, et al., we can provide a more transparent translation into an implementation in Prolog.

Expressing the definitions in logic does not give us the semantics of FOL for inheritance reasoning itself, but for our reasoning about inheritance. This is essentially an application of the *syntactic method* proposed by Morgan (1976) for theorem proving in nonclassical logics. In the syntactic method the proof theory of a nonclassical propositional logic is restated in terms of a first order provability predicate. Theorem proving in first order logic is sound and complete with respect to the theorems of the object logic, and "no semantic theory is required, so very exotic systems can be studied even when no semantic theory is available" (Morgan, 1976, p.856). The clauses which result in the first order representation of the logic studied using the syntactic method turn out to be Horn clauses; though we have not used the provability predicate exactly as did Morgan (1976), we are still able to apply his results. Integrated in the following discussion is a restatement of the following definitions in Prolog relations, and these Prolog relations allow us to test the implications of inheritance reasoning definitions on various networks.

Path preemption allows more specific information that is contained in a direct link to override conflicting information in a more general (longer) path. This topological ordering of paths is called the *inferential distance ordering* (Touretzky, 1986). Only direct conflicting links can preempt other paths, though a preempting link may be part of a longer path.

Definition 2: Let π, π', and ρ be variables over positive paths.

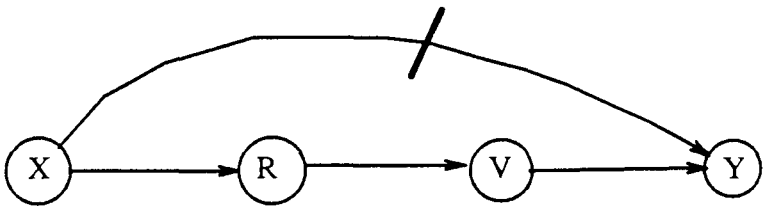


Figure 2-4: Link X-/->Y preempts the path XRVY.

1. A positive path πy is *preempted* by a link p-/->y if there exists a permitted path π' such that *FirstNode*(π)=*FirstNode*(π') and *LastNode*(π)=*LastNode*(π'), and either *FirstNode*(π')-/->y is a link in the network and p=*FirstNode*(π'), or for some subpath ρ of π' *FirstNode*(ρ)-/->y is a link in the network and p=*FirstNode*(ρ).
2. A negative path π/y is *preempted* by a link p--->y if there exists a permitted path π' such that *FirstNode*(π)=*FirstNode*(π') and *LastNode*(π)=*LastNode*(π'), and either *FirstNode*(π')--->y is a link in the network and p=*FirstNode*(π'), or for some subpath ρ of π' *FirstNode*(ρ)--->y is a link in the network and p=*FirstNode*(ρ).

Examples of preemption are presented graphically in Figures 2-4 and 2-5. In both figures a

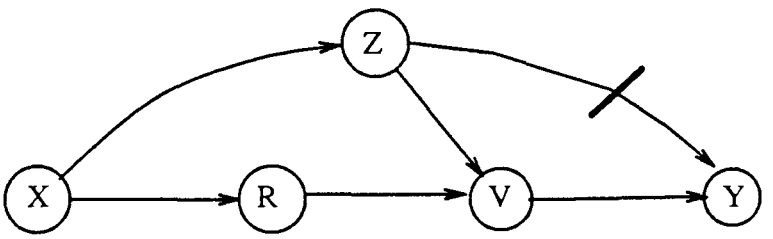


Figure 2-5: Path XRVY is off-path preempted by the link Z-/->Y.

positive path is preempted by a negative path. Matching the definition to the network in Figure 2-4, both π and π' correspond to the path XRV. The endpoints of π and π' coincide, and *FirstNode*(π')-/->Y is a link in the network, so XRVY is preempted by the link X-/->Y. In Figure 2-5, π still corresponds to XRV, but π' corresponds to XZV, and ρ to ZV. *FirstNode*(ρ)-/->y is the link that preempts the path XRVY. This second example is an instance of off-path preemption (Touretzky, et al., 1987). Essentially, XZ/Y is said to be more specific than both XZVY and XRVY.

Compound paths which conflict are subject to cancellation. The difference between cancellation and preemption is that neither conflicting path is permitted after cancellation, but preemption does permit one of its conflicting paths. Consider the network depicted in Figure 2-6 which has a topology somewhat in between that of Figures 2-4 and 2-5. Although paths π and π' exist whose endpoints coincide, no π' exists that has a subpath ρ whose first node participates in a

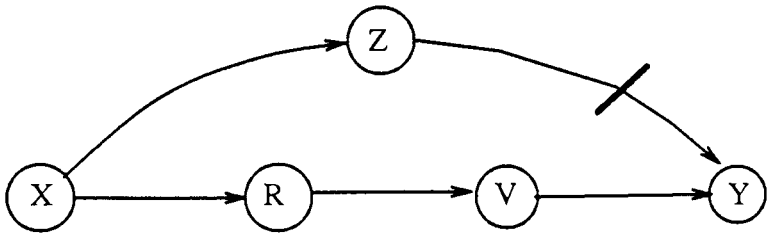


Figure 2-6: Paths XZY and XRVY cancel each other.

preempting link. Neither XZ/Y nor XRVY is favored over the other. Since we are defining a skeptical reasoner, we say that the paths cancel each other—neither is permitted. A credulous reasoner would resolve to two extensions from the network, one in which Xs are Ys and another in which Xs are not Ys. On the other hand, if the network consisted of two conflicting links, then we would say that it contains an inconsistency.

```
preempted(From,Through,To,Preemptor) :-
        chain(From,Through,To),
        complement(To,NotTo),
        link(Preemptor,NotTo),
        lastnode(Front,Last,Through),
        permitted(From,OtherThrough,Last),
        member(Preemptor,[From|OtherThrough]).
```

Figure 2-7: The Definition of *Preemption* in Prolog.

In Figure 2-7 we give a restatement of the definition of preemption as a Prolog relation between a path (specified in the arguments *From*, *Through*, *To*; items beginning with an uppercase letters are understood as variables) and a node on more specific conflicting paths (the argument, *Preemptor*). While *From*, *Through*, and *To* are all variables, we assume that *From* and *To* vary over nodes, and that *Through* varies over lists of nodes with positive, not negative, links implicitly connecting those nodes in the order of occurrence in a given list. The *preempted* relation can hold only if the chain of links described by the path *From--->Through-#->To* is actually a chain of links in the network. The relations *complement* and *link* determine whether there is a conflicting path terminating at the node *To*. The path, *From--->OtherThrough--->Last*, corresponds to π' in Definition 2. The relation *lastnode* verifies that *LastNode*(π) = *LastNode*(π'), and the invocation of *permitted* (defined later) verifies that π' is actually permitted. Finally, the call to *member* is used to determine if the first node of π' or the first node of some subpath of π' participates in the conflicting link.

Figure 2-8 depicts a Prolog translation of the network given in Figure 2-5. Links are

```
link(x,r).
link(r,v).
link(v,y).
link(x,z).
link(z,v).
link(z,not(y)).
```

Figure 2-8: A Prolog Translation of Figure 2-5.

encoded as two place relations. The polarity of a link is indicated in the second argument. A positive link A--->B is encoded as *link(a,b)*, and the negative link A-/->B is encoded as *link(a,not(b))*. In the Prolog session reproduced in Figure 2-9, we show the application of the

```
| ?- preempted(x,Y,y,Z.)

Y = [r,v],
Z = z ? ;

Y = [z,v],
Z = z ? ;

no
```

Figure 2-9: "Which Paths between *x* and *y* Are Preempted?"

relation *preempted* to determine what paths between the nodes *x* and *y* of Figure 2-8 are preempted and what the preempting node is. In a Prolog terminal session, user input is entered after a question mark. In Figure 2-9 user input is shown with added emphasis. The entry of a semicolon indicates the user's request to the interpreter to find another way to satisfy the query. Two preempted paths are returned in this fashion, *xrvy* and *xzvy*, and both paths are preempted by the node *z*. Since *preempted* is a reversible Prolog relation, we can also ask, for instance, for all the paths which terminate at *y* but are preempted. This invocation, also for the network shown in Figure 2-8 is represented in Figure 2-10. We see that there are three preempted paths that

```
| ?- preempted(X,Y,y,W).

W = z,
X = x,
Y = [r,v] ? ;

W = z,
X = x,
Y = [z,v] ? ;

W = z,
X = z,
Y = [v] ? ;

no
```

Figure 2-10: "Which Preempted Paths Terminate at *y*?"

terminate at *y*: *xrvy*, *xzvy*, and *zvy*.

Definition 2 only partially specified what it takes to preempt a path. The definition made reference to the existence of *permitted* paths, which has yet to be defined. The paths sanctioned by an arbitrary network are determined by upwards construction of paths of increasing degree; the formal definition follows:

Definition 3: Path Permission

1. Let π be a path.
   a. If π is a direct link, then π is permitted.
   b. If the degree of π is one then π is a direct link, by the definition of degree, hence π is permitted.
2. Let π be a compound path of degree n. Assume that all permitted paths with degree less than n are known.
   a. If π is a positive path then it has the form αz (i.e., *LastNode*(α)--->z is a link in the network). The path α is positive, and by Theorem 1 the degree of α is less than n. The path π is permitted iff
      i. α is permitted,
      ii. *FirstNode*(α)/z is not a direct link in the net,
      iii. Let β be a variable over positive paths. For all paths β/z with *FirstNode*(β)=*FirstNode*(α) that conflict with the path αz there exists some path which preempts β/z.
   b. If π is a negative path (it has the form, α/z), then π is permitted only under the conditions symmetric to those stated in a. That is, iff:
      i. α is permitted,
      ii. *FirstNode*(α)z is not a direct link in the net,
      iii. For all paths βz with *FirstNode*(β)=*FirstNode*(α) that conflict with the path α/z there exists some path which preempts βz.

Definition 3 follows the inductive structure of the path based definition provided by Horty, et al. (1990), even though Definition 3 is not stated in their network notation. The definition still proceeds with upwards construction of paths of increasing degree. The reason for basing the definition on increasing degree rather than increasing path length is that in some instances, information about a longer path is necessary to determine the permission of a shorter path (Horty, et al., 1990). An example of a situation in which information about a longer path is required occurs when two paths conflict, and one is shorter than the other as in Figure 2-6. According to these definitions, if those paths intersect only at their endpoints, they will cancel each other. Cancellation is stipulated by the third condition on the permission of π in Definition 3. This condition states that π is permitted only if all conflicting paths are preempted. Since π has a degree of n, we know of all paths which could conflict with π. By the definition of degree, none of the conflicting paths is longer than n. The degree of a path becomes significant only

during the examination of compound paths for the existence of conflicting paths. Conflicting paths are handled trivially in the case of direct links. All direct links are sanctioned as paths through a network, even conflicting links. If a direct link conflicts with a compound path then the definition of preemption is satisfied and the direct link preempts the compound path. In the case of conflicting compound paths, it is known that none of the conflicting paths has a degree of greater than n, and all shorter paths between the same endpoints are known, because Definition 3 proceeds on increasing degree.

A restatement of Definition 3 in Prolog is shown in Figure 2-11. This new definition has a different structure from Definition 3 in that the Prolog definition is not stated explicitly in terms of increasing degree, although it relies on the relationship between the degree of a path and the degree of a subpath as stated in Theorem 1. In the component of the logic program that must examine potentially longer paths for conflict, the definition refers to those paths directly using the term, *chain*. This term simply represents a chain of links between the endpoints *From* and *To*, if

```
permitted(From,[],To) :-
        link(From,To).

% From=FirstNode(Pi), To=z.
permitted(From,Through,To) :-
        % Last-#->z is a link.
        link(Last,To),
        % Last=LastNode(Alpha).
        lastnode(Front, Last, Through),
        % From-#->z is not a direct conflicting link.
        complement(To,NotTo),
        not(link(From,NotTo)),
        % Alpha is permitted.
        permitted(From,Front,Last),
        % For all OtherPaths that do conflict,
        % some path preempts each.
        not(unpreempted(From,OtherPath,NotTo)).

%Direct links are not preempted.
unpreempted(From,[],To) :-
        link(From,To).
%True when unpreempted paths exist.
unpreempted(From,Through,To) :-
        %A path exists between From and To.
        chain(From,Through,To),
        lastnode(Front,Last,Through),
        %From--->Through is permitted.
        permitted(From,Front,Last),
        %No preempting path exists.
        not(preempted(From,Through,To,By)).
```

Figure 2-11: The Definition of *Permission* in Prolog.

such a chain exists in the network. The length of the chain, which corresponds to degree, is

insignificant. Reference to the term *chain* stems from the invocation, *not(unpreempted(From,OtherPath,NotTo))*. The definition of *unpreempted* determines whether there is an *OtherPath* which is not preempted. Invoking *not(unpreempted)* with *NotTo* instead of *To* verifies the condition that for all OtherPaths which do exist in conflict, none are permitted. If an *OtherPath* did exist that was not preempted then *unpreempted* would hold true, and *not(unpermitted)* would be false. If *not(unpermitted)* fails to hold, this indicates the existence of a conflicting path, and neither path is permitted.

The basis case in Figure 2-11 is the first *permitted* clause which states that all direct links in a network are permitted. An additional base clause to permit paths of degree one is unnecessary since the set of paths whose degree is one is a subset of the set of paths that are direct links. The three argument positions of the *permitted* predicate represent *From*, *Through*, and *To*, as described above. *From--->Through* maps to α in the formal definition of permission, and *To* maps to *z*. The empty list in the *Through* position of the term for the basis indicates that no intermediate nodes lie on the path. The second rule defines permission in the general case. Since we use the relation *complement*, in defining this rule, the same rule stipulates the permission of both positive and negative paths in a single rule. The relations, *link* and *lastnode*, verify that *LastNode*(α)-#->*z* is a link contained in the network, and the relation *not*(*link*) stipulates that a directly conflicting link cannot be present in the network (for then, that link would be permitted, and the path under consideration is preempted). The recursive reference to *permitted* specifies that the subpath α must itself be permitted (*From--->Through*). Finally, the relation *unpreempted* holds when the path specified as input through its arguments is actually a chain of links through the network which is not itself preempted. Thus, the specification, *not*(*unpreempted*), stipulates that no conflicting, unpreempted paths exist. This is equivalent to the specification in the formal definition that is expressed: for all paths that conflict with α-#->*z*, there exists some path which preempts each conflicting path. The clauses which make up the definitions of *unpreempted* and *permitted* implement Definition 3 even though a different ordering is stated on those constraints. The order of the restrictions stated in the Prolog definition is guided by efficiency considerations in limiting the search space.

In Figure 2-12 we include a Prolog session that applies the Prolog definitions given above to the network shown in Figure 2-5 and translated to Prolog in Figure 2-8. Only the negative path from *x* to *y* is permitted. Other paths which have *x* as a first node are permitted, but only the one negative path ends at *y*.

```
| ?- permitted(x,Through,To).

Through = [],
To = r ? ;

Through = [],
To = z ? ;

Through = [r],
To = v ? ;

Through = [z],
To = v ? ;

Through = [z],
To = not(y) ? ;

no
```

Figure 2-12: "Which Paths Beginning at Node *x* Are Permitted?"

Consider an example taken from Touretzky, et al. (1987) (Figure 2-13). Assuming a

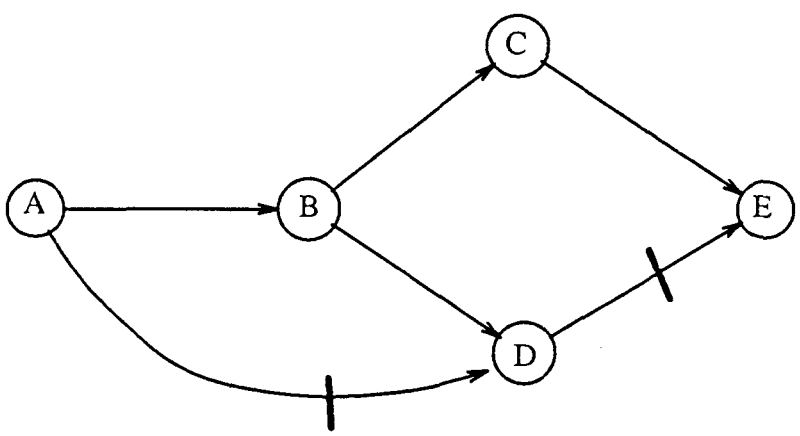


Figure 2-13: Preemption by A-/->D Permits the Conclusion A--->E.

skeptical reasoner, it initially appears that no conclusions about whether As are Es should be forthcoming, since there are conflicting paths from A to E. However, the definitions do not sanction the inference of E from A. ABD/E is not a path permitted by the network, because its subpath, ABD, is preempted by the link A/D. The intuition is that A/D contains more specific information about A's D-ness than is in the chain of links, ABD (and we know from the definitions that no path can contain two negative links, so A/D/E does not conflict). Hence ABCE is a permitted path. Figure 2-14 depicts a translation of the network from Figure 2-13 into Prolog, and Figure 2-15 demonstrates a Prolog session in which the permission and preemption of paths is verified.

```
link(a,b).
link(b,c).
link(b,d).
link(c,e).
link(d,not(e)).
link(a,not(d)).
```

Figure 2-14: A Prolog Translation of the Network Depicted in Figure 2-13.

```
| ?- permitted(a,Y,e).

Y = [b,c] ? ;

no
| ?- preempted(a,Through,To,By).

By = a,
Through = [b],
To = d ? ;

no
```

Figure 2-15: Upwards Reasoning: Preemption of ABD Permits the Conclusion A--->E.

Of course, the difference between implications of a network obtained by inspecting it for direct or conflicting paths, and the implications made according to a given reasoner suggests that the definitions are *process* oriented. The reasoner defined by Horty, et al., (1990) which we describe herein is an upwards reasoner (cf. Touretzky, et al., 1987). In contrast, a downwards reasoner would be able to conclude nothing about the relationship between A and E in Figure 2-13 because of the ambiguity of the relationship between B and E. The difference hinges on the order of links considered in path construction, and this suggests that no declarative semantics can be devised as a foundation to both upwards and downwards restricted skeptical reasoners. Though some of the issues we discuss here are common to upward and downward reasoners, we assume an upward mode of processing, in accordance with the definitions from Horty, et al. (1990), given above.

2.2. Redundancy and Stability

One proof theoretic consideration present in both upward and downward (credulous or restricted skeptical) reasoners is that the reasoning process should not be confounded by the presence of redundant links. The network shown in Figure 2-16 contains a link, A--->C, which is considered redundant with respect to the path ABC. Networks of this topology were used by Touretzky (1986) as counterexamples to illustrate the inappropriateness of shortest path reasoning (Fahlman, 1979). The counterexamples demonstrate that if a shortest path reasoning mechanism

is used, then in the network in Figure 2-16, for example, it is wholly arbitrary whether the reasoner will reach a conclusion based on the positive or negative paths of the same length from A to D. Reasoners which exhibit such nondeterministic behavior are said to be *unstable*. The disproof provided by the example relies on the assumption is that the link B/D provides more specific information about A's D-ness than does the path ABCD, but that the link A--->C is merely redundant with respect to ABCD. We dispute that assumption.

While we do agree that the network {A--->B, A--->B}, consists of a redundant link, we do not agree that this is necessarily the case for the network {A--->B, B--->C, A--->C}. The last

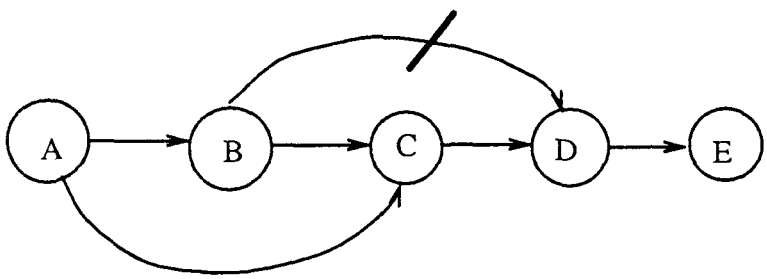


Figure 2-16: A Network Containing a Redundant Link.

link, A--->C, is redundant, according to Horty, et al. (1990), since we already have ABC, and A--->C contains only the information present in the longer path. But, if A--->C is redundant in Figure 2-16, then B-/->D is incoherent, because if the former edge is giving only information already contained in ABC, then the latter is giving information totally at odds with BCD. If there is no more information that is left unexpressed by the path ABC, then the same is true of the path BCD, and there is nothing else to express about the path, let alone something contrary to the path. Rather, if one of those links can contain more specific information than the longer corresponding path, then both paths must have that potential.

Admittedly, it is disconcerting to think of the link A--->C as non-redundant given that the links are intended to represent, "is a." It would seem that the is a-ness of A--->C should be contained already in ABC. However, this is merely because an adequate explication of the meaning of "is a" has yet to be offered (c.f. Aristotle, 1952; Rosch and Lloyds, 1978; Brachman 1983). It is absolutely certain, especially for upwards reasoners, that set containment is not the correct interpretation of "is a". If set containment were the correct interpretation, then *coupling* would be present. Coupling occurs when properties of a subclass are in agreement with

properties of a superclass. In a network that admits defeasible links, coupling is not a necessary condition for transitivity. "Downward reasoners necessarily produce coupled theories because the only properties a node can inherit are those of its superiors. Upward reasoners are not so constrained" (Touretzky, et al., 1987, p.479). This quotation admits that an individual or class has properties independent of the nodes to which it is linked, and these individuals and classes are categorized on the basis of these properties. That is, being connected to a node as in A--->B does not completely specify the node A. Other information about A is contained in A--->C just as information contained about B in B-/->D is not contained in BCD. Researchers in THAT paradigm seem to recognize this possibility when they discuss the intuitions behind their reasoners, particularly Horty, et al. (1990) who, recall, have implemented an upwards reasoner, but their definitions ignore this possibility.

For an example of a situation in which coupling does not occur, consider an analogy from international relations. The United States defends Kuwait because of Kuwait's global importance as an oil producing nation. Kuwait defends Palestine because of its Arab sympathy. The U. S. may or may not defend Palestine. Neither situation follows as a matter of course from the other two. We could represent this in the network shown in Figure 2-17, in which nodes represent nations and links are interpreted as "defends". As the network stands we would

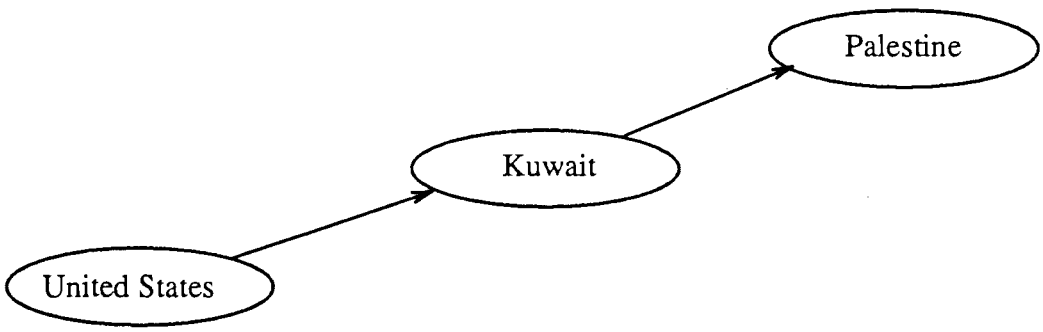


Figure 2-17: An Uncoupled Network.

rightfully conclude using inheritance that the U. S. defends Palestine. But, additional information is required if this is to be an explicit fact encoded in the network. Additional information about the U. S.—that the U. S. defends Palestine because defending nations is the right thing to do—would be sufficient to ground the explicit coding of the link. This information is associated with the node for the U. S. but is not included in the link which indicates that the U. S. defends Kuwait. Equally, we require additional information about the U. S.—that the U. S. does not defend Palestine because Palestine does not control resources of interest to the U. S.—to add a

negative link between the nodes for the U. S. and Palestine. The truth of either situation about the U. S. can ground an explicit link, but neither the positive nor the negative link is contained in the original two links. Either link encoded explicitly is qualitatively different from a link derived from inference. In inheritance reasoning, links do not represent superclass/subclass relations, but they do indicate classification. A link is a classification of some information at a node, but not necessarily of all of that information. If a single link does not classify all of the information at a node, then other links emanating from that node represent different information. They are not redundant. If it is possible to find different information to be classified with the link B-/->D even though B--->C and C--->D, then it is also possible to find different information from that classified by A--->B and B--->C to be classified with the link A--->C.

Touretzky (1986) acknowledges that something is important about "redundant" links. He claims, "we cannot easily ban redundant statements because there *are* no truly redundant statements in a system that allows exceptions" (p.10)[4]. But, it is evident from the example he gives that he considers the link A--->C, above, to be redundant, since he contrasts the status of that link with the link C--->E added to the the network shown in Figure 2-16. The modified network is illustrated in Figure 2-18. He claims that C--->E is not redundant. Touretzky (1986)

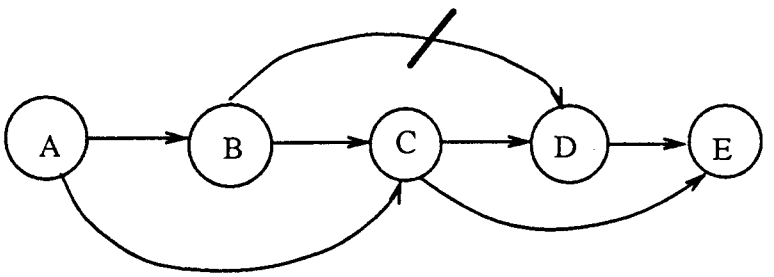


Figure 2-18: A--->C Is Considered Redundant, but C--->E Is Not.

argues that we would want to be able to infer that Bs are Es via BCE, even though Bs are not Ds. Note that this inference cannot occur in the upwards reasoner without the link C--->E. *Stability* is the property of an inheritance reasoner's being able to construct the same inferences from a

[4]Touretzky's statement makes clear that the networks he deals with are quite distinct from the inheritance networks considered by David Israel, (1983). Israel assumes that the descendants of a node in an inheritance taxonomy are all mutually exclusive. But, the presence of redundant links in a network implies the presence of chains of descendancy from a node which are not mutually exclusive. Indeed, a classic example in the inheritance literature, the Nixon Diamond which demonstrates the ambiguity caused by two conflicting paths, is a problem because Quakerism and Republicanism are not mutually exclusive modes of socio-political philosophy. If they were mutually exclusive, the diamond would be held forward as an example of inconsistency, not merely ambiguity.

network augmented by explicit links derived from inferences. Thus, the system described by Touretzky exhibits instability. Horty, et al. (1990) take the same position as Touretzky (1986); they provide the rationale that stability is not always a desirable property for inheritance reasoners, but the importance of stability is relative to the topology of the networks reasoned over. However, given that there are conflicting paths through the networks of both Figures 2-16 and 2-18, using a skeptical style of reasoning we should not be able to conclude anything in either case. Inconclusiveness with regard to the networks corresponds exactly with Boutilier's (1989) minimal-model semantic account of inheritance systems. Boutilier points out that this inconsistency in labeling links redundant makes it hard to justify the use of the inferential distance ordering at all. He provides an alternative set of definitions for preemption that has the effect of labelling both A--->C and C--->E as redundant links. Topologically, this labelling is more consistent than the definitions of Touretzky (1986) and Horty, et al. (1990), and secures stability for the reasoner. Boutilier's reasoner will conclude of both networks (shown in Figures 2-16 and 2-18) that As are not Es. However, this stability is maintained by keeping redundancy as a topological property of a network rather than as a semantic property.

THAT approach to inheritance reasoning is based upon the inferential distance ordering of paths in a network. When conflicting paths arise the most specific path is preferred. Boutilier's work points out an inconsistency in THAT approach and patches it to make the topological definition of "most specific paths" consistent. New links can be added to the networks of both Horty, et al. (1990) and Boutilier (1989). In both systems, some links add new, more specific information and other links are considered redundant. The polarity of an added link relative to the path it spans determines whether the added link is redundant or more specific. Some new links can provide more specific information than was known before the addition, but only if the new link represents information that is completely contrary to what was known before the addition. If the additional information provided by a new link is not completely contrary to what was known, the new link is to be considered redundant. This seems incoherent, particularly in the case of Horty, et al. (1990), who explain the virtues of upwards processing, that a node has properties independent of the more general nodes in the network. While Boutilier (1989) indicates that some links may be "independently justified" he does not indicate how this might be encoded in a network for links which are otherwise topologically redundant.

We are not arguing that redundant paths do not exist, but we do claim that redundancy

cannot be identified from the topology of the semantic network (if it could, the graph could hardly be claimed to represent a *semantic* network). Possible cases of redundancy may be identified. This is shown by the fact that the reasoners discussed herein do reason around "redundant" paths correctly when the network has a friendly interpretation. Consider an unfriendly interpretation of Figure 2-16: let A = Honda; B = dirt bike; C = motor vehicle; D = something used to get to work; E = something used for transportation. None of these nodes stand for individuals. If we take a particular individual, Θ, and add the link Θ--->A, then we are forced to conclude, using THAT style of reasoning, that although Θ is a Honda, it is not used to get to work, and we can reach no conclusion about whether it is a mode of transportation. We do not insist on the conclusion that Θ is used to get to work, but rather that no conclusion is possible since the topology of the network is ambiguous. In this interpretation, the link A--->C contains information which is not in the path ABC. This information is lost when reasoning is based on topological considerations alone. Without examining the interpretation of the net, it is impossible to know whether or not a given path actually is redundant. Topological considerations do not provide enough material to make the classification. This consideration will emerge again later in this thesis, but presently we consider alternative systems and other critical approaches to THAT style of inheritance.

2.3. Two Alternative Systems

In this section we consider two alternative systems of path based inheritance in comparison with THAT approach. We avoid characterizing any system as "more intuitive" than another, since "intuition" is a conclusion based on semantics, and we will not characterize an acceptable semantics for inheritance systems apart from the motivating notion of "typicality." Where possible, we point out formal distinctions which cause one system to behave differently than another. The systems that we describe here do not fall into the space of 72 distinct possibilities that Horty (1989) claims for path based reasoners because the two systems that we describe, those of Geffner and Verma (1989) and of Ballim, et al. (1989) are not purely path based. The system of Geffner and Verma introduces a derivability relation on top of the notion of paths. Work more closely related to THAT family of research classifies the inference corresponding to the endpoints of a derived path as an implicit link with the same status as an explicit link (thus causing the problems with redundancy and instability that we noted in the last section). Instead, Geffner and Verma distinguish inferences by referring to them with the derivability relation, as we shall explain in more detail shortly. The system of Ballim, et al.

(1989) is removed from THAT sort of path precedence in a different direction. Ballim, et al. calculate path precedence from *inclination quantifiers* associated with each path. While their system still bases its preferences upon purely topological considerations, it provides a framework into which other considerations can be incorporated.

2.3.1. Geffner and Verma

Geffner and Verma (1989) present an alternative system for defeasible inheritance which is based in THAT tradition of path based inheritance, but which differs in defining inheritance relative to a derivability relation "|-". We give definitions which are equivalent to those stated by Geffner and Verma, and we accompany the definitions with their Prolog translations.

Definition 4: Derivability for nodes.

1. If p is a node in a network, then p |- p.

Though it is not stated by Geffner and Verma, a careful reading reveals that to obtain behavior expected of the derivability relation, the relation must be reflexive with respect to nodes in a hierarchy even though no links exist in the network which connect node to itself. The usual method of defining path based inheritance is to consider a link connecting two nodes to be the minimum wherewithal required to complete inheritance.

Definition 5: Derivability for links.

1. If p--->r is a link in the network, then p |- r.
2. If p-/->r is a link in the network, then p |- ~r.

On the other hand, a usual treatment is to consider each direct link contained in a network as supporting the inference corresponding to its endpoints. Recall that it is considered a methodological advantage of inheritance reasoners over FOL that a network can contain directly conflicting links supporting contradictory conclusions without supporting any arbitrary conclusion from the basis of contradictory links, unlike a contradiction contained in the deductive closure of a theory expressed in FOL.

Definition 6: Derivability for chains.

1. If
   a. p |- q, and
   b. q--->r is a link in the network, and
   c. every path π-/->r with *FirstNode*(π) = p in the network is *defeated* by the node p,

   then p |- r (*chaining* occurs).
2. If
   a. p |- q, and

b. q-/->r is a link in the network, and

c. every path π--->r with *FirstNode*(π) = p in the network is *defeated* by the node p,

then p |- ~r (*chaining* occurs).

The sense of *path* used here is exactly the same used earlier; *permitted paths* in THAT material occupy the same place as the derivability relation in Geffner and Verma (1989). *Defeat* will be defined shortly in Definition 7.

Note that Definition 5 is a special case of Definition 6 in which p is identical with q. Thus, the two Prolog relations given in Figure 2-19 explicitly cover the conditions of Definitions

```
derived(From,From).
derived(From,To) :-
        %A chain of links exists between the endpoints
        chain(From,Through,To),
        link(Last,To),              % q-#->r is a link.
        derived(From,Last),         % p |- q.
        complement(To,NotTo),
        % No undefeated, conflicting path exists.
        not(undefeated(From,Thru,NotTo)).


undefeated(From,Thru,To) :-
        % A path From--->Thru--->To exists.
        chain(From,Thru,To),
        %The path is not defeated.
        not(defeated(Defeator,From,Thru,To)).
```

Figure 2-19: A Prolog Translation of Definitions 4, 5 and 6.

4 and 6. The first Prolog relation specifies the base case stated in Definition 4. The second relation specifies the general case. The derivability relation holds between the endpoints of a chain of links only if a chain with those endpoints exists in the network. The relation *link* verifies that the link q-#->r is in the network, and *derived* verifies that p |- q. The ordering of these relations is different in the Prolog definitions than in Definition 6 for efficiency considerations. The variables *From*, *Last*, and *To* used in the relations shown in Figure 2-19 correspond to p, q, and r, respectively, in Definition 6. The last two relations in the Prolog rule, *complement* and *not(undefeated)*, verify that every complementary chain of links between *From* and *To* is defeated. This relation is verified using a negated existential statement (it is not the case that a path exists which is not defeated).

Definition 7 defines the notion of *defeat* in terms of the dominance relation. A link or compound path π-#->r is defeated when a conflicting link exists whose first node derives some node in π. Definition 7 is restated in the two Prolog relations shown

Definition 7: Defeat

Let π be a path whose length is greater than zero. Then π is either a direct link of the form β-#->r where β and r are nodes, or π has the form αβ-#->r where α is a path whose length is greater than or equal to zero and β is a path whose length is strictly greater than zero.

A path β-#->r is defeated by a node n if w *dominates* β and n |- w.

```
defeated(N,From,Chain,To) :-
        dominates(SomeNode,From,Chain,To),
        derived(N,SomeNode).
defeated(N,From,[P|Ath],To) :-
        defeated(N,P,Ath,To)
```

Figure 2-20: The Definition of *Defeat* in Prolog.

in Figure 2-20. The first relation is a straightforward translation of the definition for the cases in which β is a single node and in which the length of α is zero. The second rule stipulates that a chain of links is defeated (*[P/Ath]*) if a suffixed constituent chain (*Ath*, which corresponds to β) is defeated; this is the case in which α is greater than zero.

Definition 8: Dominance

1. A path π-/->r is dominated by a link p--->r if p |- *FirstNode*(π).

2. Symmetrically, a path π--->r is dominated by a link p-/->r if p |- *FirstNode*(π).

In either case, it is also said that the path is dominated by the node p.

```
dominates(Node,From,Chain,To) :-
        complement(To,NotTo),
        link(Node,NotTo),
        derived(Node,From).
```

Figure 2-21: The Definition of *Dominance* Stated in Prolog.

A translation of the definition of dominance as a Prolog relation is shown in Figure 2-21.

A difference in these definitions and those of Horty, et al. (1990) is that the inferences sanctioned by the network are those derived from the network using the derivability relation, rather than those which correspond to the endpoints of valid paths. In THAT family networks, we can identify a continuum of pathhood: the links in the net, which are explicitly encoded; paths, which are a subset of the generalized paths (chains of links) through the network; and permitted paths, a subset of the paths. Within the system just defined there are two additional graduations: nodes, as mentioned in the definitions; links as in THAT family; paths, also as in THAT family; the binary derivability relation, corresponding to permitted paths; and *certified paths*. A certified path through a network is just a path which is not defeated. The derivability relation always holds between the endpoints of a certified path. The derivability relation

introduces a finer graduation because in some cases the definitions allow us to conclude that the derivability relation between its endpoints of some defeated paths also holds. An example taken from Geffner and Verma (1989) may clarify how such a derivation can hold.

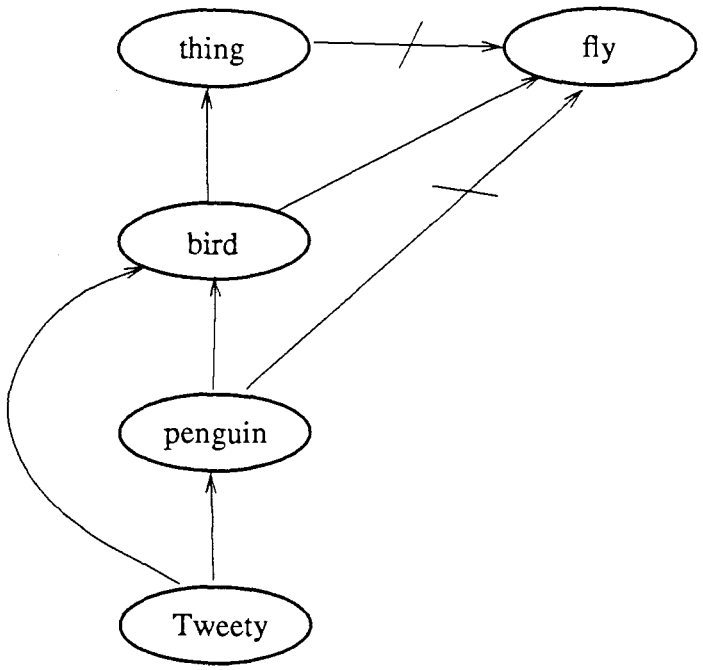


Figure 2-22: Certified Paths vs. Derivability.

In Figure 2-22 from Geffner and Verma (1989), consider the path Tweety--->penguin--->bird--->thing-/->fly (or even the path Tweety--->bird--->thing-/->fly, which is considered redundant with respect to that path). Figure 2-23 shows a Prolog session in which queries are made using the above relations and a translation of the network given in Figure 2-22. We want to know whether Tweety |- ~fly and whether the path sanctioning this derivation is certified. Definition 6 divides this question into constituent questions. Clearly, Tweety |- thing, since that derivation involves no canceled links, and thing -/-> fly is a link in the network. In Figure 2-23, we see that the relation, *derived(tweety,thing)* holds. The second part of the question addresses whether there is any path from Tweety to fly which is not defeated. We invoked the relation *chain* in the Prolog session shown in Figure 2-23 with a variable as an argument so that we could locate all of the paths between Tweety and fly. Two paths exist, the one through

penguin and bird, and the topologically redundant path through bird. No undefeated path exists since paths containing the link bird--->fly (the only paths which could conflict) are defeated by the node penguin (Tweety--->bird--->fly is defeated because a consequence of Tweety, penguin, defeats bird--->fly). So, the derivation is permitted using the path Tweety--->penguin--->bird--->thing-/->fly: Tweety |- ~fly. However, in the same way that the node penguin defeats bird--->fly, the node bird defeats the link thing-/->fly. The last three Prolog rule invocations shown in Figure 2-23 show that the derivation relation holds even though the relevant path is defeated. This means that the derivation between the endpoints of the path is sanctioned, but the path itself is not a certified path in the network.

```
| ?- derived(tweety,thing).

yes
| ?- chain(tweety,X,fly).

X = [penguin,bird] ? ;

X = [bird] ? ;

no
| ?- defeated(By,tweety,[penguin,bird],fly).

By = penguin ? ;

no
| ?- defeated(By,tweety,[bird],fly).

By = penguin ? ;

no
| ?- derived(tweety,not(fly)).

yes
| ?- defeated(By,tweety,[penguin,bird,thing],not(fly)).

By = bird ? ;

no
| ?- defeated(By,tweety,[bird,thing],not(fly)).

By = bird ? ;

no
```

Figure 2-23: A Prolog Session.

Though the derivability relation can exist between the endpoints of a defeated path, when this actually occurs there is some other undefeated path for which the relation holds as well. To understand why, suppose that some complex path is defeated and the derivability relation holds

between its endpoints, but no other path exists between the same endpoints. For the network we used in Figure 2-22 and the discussion in the last paragraph, we suppose that we can remove the link between *penguin* and *fly* and still obtain the inference that tweety does not fly. The resulting network is given in Figure 2-24. Because no other paths exist, there is no other path to defeat the

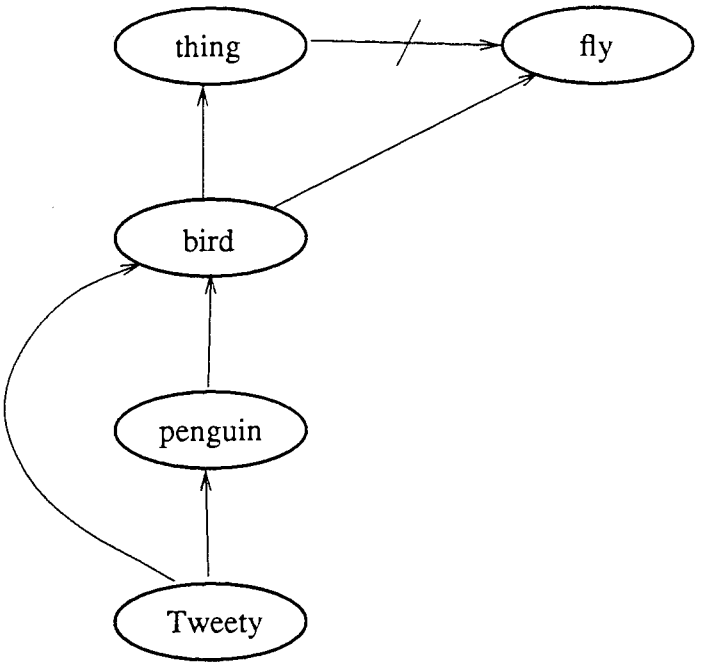


Figure 2-24: Certified Paths vs. Derivability.

defeating path. In the example, the link between *bird* and *fly* forms the basis of the defeating path. But, that means that the defeating path constitutes an undefeated conflicting path between the endpoints of the path originally under consideration (the defeated path). In the network shown in Figure 2-24, the defeated path is one ending with the link *bird-/->thing*. However, if this is the case then by the definition of chaining the derivability relation fails to hold between the endpoints of the defeated path, and this failure contradicts the assumption that we made at the outset. This proves[5] that the derivability relation is closely tied to the existence of some corresponding certified path. For the example given in Figure 2-24 this means that we cannot conclude that tweety does not fly. But, it is interesting that the derivability relation can be proven with respect to some defeated path relying on the certified path only indirectly.

[5]We make a different argument, but the result corresponds to Geffner and Verma's (1989) correspondence theorem.

Indirection is incorporated into the definitions and allows defeat to cascade across paths. In discussing Figure 2-22 we noted that the link *bird--->fly* is defeated, but this link is itself the cause of the defeat of the link *thing-/->fly*. Another interesting consequence of the definitions arises in networks that contain explicit inconsistencies. Given an inconsistent network {A--->B, A-/->B}, for instance, both links are derivable, even though both links are defeated by the node A, as can be seen in the Prolog session shown in Figure 2-25. In THAT work which we described

```
| ?- derived(a,X).

X = a ? ;

X = b ? ;

X = not(b) ? ;

no
| ?- defeated(By,a,[],b).

By = a ? ;

no
| ?- defeated(By,a,[],not(b)).

By = a ? ;

no
| ?- listing(link).

link(a, b).
link(a, not(b)).

yes
```

Figure 2-25: An Inconsistent Network.

earlier, the definitions label both links as permitted paths. Since permitted paths in THAT family of inheritance sanctions inferences corresponding to the endpoints of the paths, permitted paths correspond to the derivability relation. So, both THAT family and Geffner and Verma sanction the inconsistent inferences simultaneously. But, in the system outlined by Geffner and Verma we can label the inconsistent links, "defeated" because of the extra graduations in classifying chains of links. The links are not mutually preemptive under the definitions provided by Horty, et al. (1990).

The specification of defeat and the rippling property associated with it imbue the reasoner defined by Geffner and Verma with acute instability.

> We have chosen to reject cumulativity [stability]. Indeed, we are inclined to believe that cumulativity is not necessarily a property of 'correct' defeasible inference but a property about *belief dynamics*, namely a 'rational' policy of belief revision in the light of new information. (Geffner and Verma, 1989, p.10)

Geffner and Verma continue, pointing out that in some cases it seems appropriate to preserve derivability with the addition of "redundant" and even new information, and in other cases this is inappropriate. What Geffner and Verma see as distinguishing these cases is not clear; however, it is something other than a topological consideration. This facet of the system presented by Geffner and Verma, in distinction to the work more closely aligned with THAT paradigm, motivate its inclusion in the present discussion.

2.3.2. Link Arithmetic

Ballim, et al. (1989) present yet another approach to path based inheritance, and it is representative of the approach to path construction based on link arithmetic. Link arithmetic is the association of numeric values with links and the computation of values for the chaining of adjoining links. Additional rules impose a preference ordering based on the certainty values thus associated with the composite links. In the abstract, a link arithmetic approach is not based on the topology of paths, since the preference of one path over another is determined by an ordering of the certainty factors associated with each inference. The arithmetic for determining those certainty factors does not have to be guided by topological considerations. However, the system of Ballim, et al. (1989) is defined so that the arithmetic respects topological considerations.

Ballim, et al. (1989) call the relationship that holds between the endpoints of links and all other valid paths an *effective relationship*. Associated with each effective relationship (ER) is an *inclination qualifier* or *leaning* which indicates the relative leaning we have towards accepting a given ER. Links in a net are all assigned inclination qualifiers of zero, and the inclination qualifiers for ERs on complex paths are derived from the link basis. Preference is given to paths whose ERs have smaller inclination qualifiers, and since no value smaller than zero can arise, all links in a network have equal preference, just as in THAT family and Geffner and Verma (1989). Ballim, et al. define an arithmetic for combining paths and for giving them a preference ordering. This arithmetic encodes topological considerations like preferring shorter paths over longer paths, and paths of strict links over paths mixed with defeasible links.

Rules are given for constructing complex ERs. There is a set of rules rather than a single rule because Ballim, et al. include both strict and defeasible links in their nets, and the two sorts of links are treated as fundamentally different. The combination of different sorts requires different rules. The heterogeneous system addresses Brachman's criticisms that defeasible nets

cannot represent definitions of composite concepts (cf. Brachman, (1983, 1985)). Horty and Thomason (1988) have also addressed this criticism. A link or ER can be positive or negative, strict or defeasible. Rules govern the computation of inclination qualifiers associated with various combinations. Some combinations, like the chaining of any link after a negative (strict or defeasible) link, are not permitted. For those combinations that are allowed, the leaning associated with the ER for the combined path is taken as the maximum of the leanings of the path's constituents, or in some cases, that same maximum value plus some constant. When the combination involves any sort of link followed by a strict link, the constant is zero; if it involves a strict link followed by a defeasible link, the constant is one; and if two defeasible links are combined the constant is two. Since preference is given to ERs with smaller inclination qualifiers, the net effect is that an ER for a complex path has a greater inclination qualifier (less inclination) than either of its constituents.

An additional set of rules over the different combinations of paths is designed to state relationships among multiple ERs (multiple paths between endpoints). The two general possibilities are *specialization*, in which two ERs have the same polarity, and *conflict*, in which two ERs have opposite polarity. A final set of rules imposes a preference ordering on ERs. Invalidation rules eliminate some conflicting ERs, and precedence rules order paths by increasing inclination qualifiers. As the preference order on ERs is a partial ordering and two ERs may have equal leanings, the theory stated by Ballim, et al. (1989) is a sort of skeptical reasoner. Faced with two ERs with equal inclination qualifiers the reasoner aborts under the recognition of ambiguity.

Without enumerating the rules in detail, we can still see interesting features in this system. First of all, the approach of Ballim, et al. reasons over nets that mix both strict and defeasible links. Horty and Thomason (1988) present an alternative approach to mixed inheritance, but their presentation is squarely path based. The system of Ballim, et al., is different because of its link arithmetic. Ballim, et al. (1989) do not attempt to argue that the particular set of arithmetic functions provided in their tables of rules are the most appropriate functions. However, the behavior induced by the link arithmetic rules in the present system agrees with the heterogeneous system of Horty and Thomason on at least one point. Horty and Thomason (1988) argue:

> In the context of defeasible networks, it makes good sense to say that direct information can be carried only by direct links: any compound path represents an argument that can itself be undermined. In the context of mixed nets, however, certain kinds of compound paths *can* legitimately be thought to carry direct information—namely, compound paths consisting of a single defeasible link followed by a strict segment of any length." (p.430)

In prefering ERs with smaller leanings, and given the particular set of functions for path combination, Ballim, et al. (1989) also indicate a preference for the same form of heterogeneous path described by Horty and Thomason. It is interesting that the path described is prefered in both systems to a path which appears topologically equivalent, in which a strict segment of any length is followed by a single defeasible link.

We also note that an ER is very much like the derivability relation of Geffner and Verma (1989), in that it is a relation between the endpoints of a path. Nonetheless, because of the specification of path precedence in terms of simple link arithmetic we obtain a different sort of instability from that advocated by Geffner and Verma. We refer to Figure 2-26 for our explanation. In Figure 2-26 the dark arrows represent strict links, and the light arrows represent

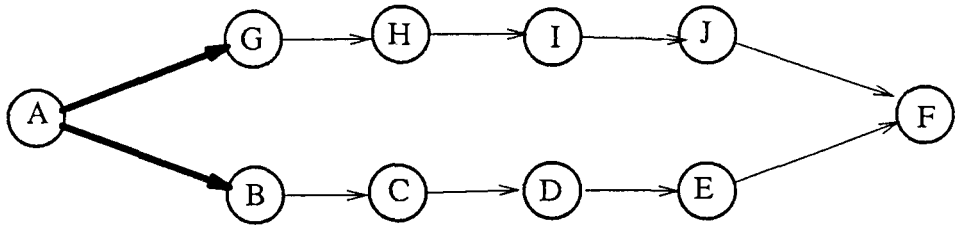


Figure 2-26: An Ambiguous Net with Both Strict and Defeasible Links.

defeasible links. The rules for combining ERs stipulate that the leaning associated with every link is zero. Applying those rules, we determine that there are two conflicting ERs between A and F, and both have the inclination qualifier 7. Hence the net is ambiguous. However, if "redundant" links are added for the ERs between B and D and D and F, respectively, we obtain the network shown in Figure 2-27. Even if the leaning on each of the new links is not specified as

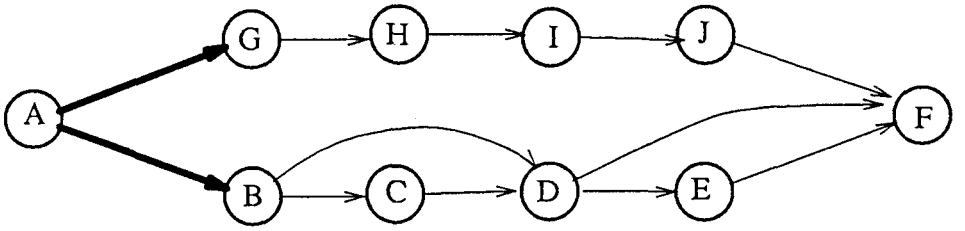


Figure 2-27: Instability: An Ambiguous Net with Redundant Links Added.

zero, but as 2 (which corresponds to the leaning associated with the ER derived from both BCD and DEF according to procedural application of the rules), we are able to construct a path ABDF whose ER is 5. The network depicted in Figure 2-27 is no longer ambiguous because the ER whose leaning is 5 is prefered to the ER for AGHJIF. The addition of topologically redundant

links to a network causes the reasoner to make different inferences from the network; this is unstable behavior.

The instability lies in the link arithmetic which makes the theory interesting. Note that the theory is not a semantic account of inheritance; the leanings assigned to ERs, even to links, have nothing to do with the probabilities that given links represent factual information. Instead, leanings encode information about the number of links in a given chain. The more links the larger the value of the inclination qualifier. Examining modified nets with redundant links added, we can construct shorter chains between the same endpoints and, hence, derive smaller leanings. Using link arithmetic is a promising approach, because if semantic information is encoded in leanings rather than merely topological information, then there is a clear way to identify semantically redundant links from links that are merely topologically redundant.

2.4. Summary

This chapter has presented three systems of path based inheritance, pointing out the syntactic flavor of the systems. They are all topologically based and lacking an associated semantics. Two of the systems have been reformulated as logic programs (the third system has already been implemented in Prolog by its authors) to verify assertions about the derivability of conclusions in both systems. THAT paradigm defines the field, but we react against its position on the primacy of topological considerations in inheritance reasoning and the problem of redundant links. Geffner and Verma take a different position from THAT group on the property of stability, a property which is related to the status of redundant links within an inheritance reasoning system. They reject the view that the property is an essential one for inheritance reasoners to have in all cases, though Geffner and Verma define their system so that it has the property in some cases. Ballim, et al. go even further in allowing instability into their system. Strong arguments for stability have been put forward (Boutilier, 1989), but these have been based on the desire to preserve the inferential distance ordering. The inferential distance ordering is a topological ordering which overlooks the potential for topologically redundant links to contain non-redundant information. We feel that since topologically redundant links can be non-redundant in terms of the information they represent, it is sensible for a reasoner to obtain different results when those links are added. Topological instability in a system designed with the understanding that a topologically redundant link is not necessarily semantically redundant, is

therefore correct: instability means that the system reaches a different set of conclusions in the light of new information.

Touretzky (1986) proposed the inferential distance ordering to correct the problem he saw with shortest path reasoning. Recall the inheritance network in Figure 2-1, with which we began this chapter. Without the link *Clyde--->Elephant*, a shortest path reasoner will reach the correct conclusion that Clyde is not gray, but with that topologically redundant link there is another path of the same length which sanctions the inference that Clyde is in fact gray. It is indeterminate which path will be found first, and therefore there is no way to know what the conclusion will be about Clyde's grayness. Semantically, for the network whose interpretation is about Clyde, we want to conclude that Clyde is not gray. But for the interpretation that we provided in Section 2.2 the correct conclusion was ambiguous because the topologically redundant link carried information not contained in the other path. If it is known, based on the interpretation of the net that topologically redundant links can contain new information, then the counterexample to shortest path reasoning loses its force. This is significant because the computational complexity of shortest path reasoning is a linear function of the number of nodes in the directed acyclic graph that constitutes the network, but Touretzky's (1986) system is NP-Hard, as is the system of Geffner and Verma (1989) (Selman and Levesque, 1989). The de-coupled, upward processing, restricted skeptical reasoner of Horty, et al. (1990) is tractable (Selman and Levesque, 1989), and still maintains the inferential distance ordering, but maintaining the ordering is somewhat inconsistent in this case, since, as we have argued, its being de-coupled allows the possibility for topologically redundant links to contain new information. These considerations reinforce our intuition that shortest path reasoning is still useful. In the following chapters we provide an application for inheritance reasoning in which a shortest path mechanism is extremely useful.

Chapter 3
Head Driven Phrase Structure Grammars

In the next chapter we present a cyclic inheritance network which captures most of the conceptual structure of Head-Driven Phrase Structure Grammar (HPSG) (Pollard and Sag, 1987). We describe the content of nodes and the semantics of links which comprise this inheritance hierarchy. Further, we specify the nature of the path-based reasoning required by HPSG over the network, reasoning which is not confounded by the presence of a cycle.

Before describing our characterization of HPSG as an inheritance hierarchy, we first highlight some of the distinctive features of this unification-based grammar formalism (Shieber, 1986) for representing the structure of natural languages. That is the task of the present chapter. HPSG is a grammar formalism that incorporates aspects of traditional lexical grammar formalisms like *categorial grammar* (Oehrle, Bach and Wheeler, 1988) along with some aspects of contemporary linguistic theories like *generalized phrase structure grammar* (Gazdar et al, 1985). As a lexical formalism HPSG places reduced emphasis on grammar rules and increased emphasis on the lexicon. The "rules" state in very schematic terms how units of discourse may combine to form more complex units (or in a top down analysis, how complex units of discourse may be decomposed into their constituents). Rather than writing a rule of the form, S --> NP VP, we talk of more abstract entities like "heads" and "complements". For example, the head of a sentence is a VP, and the complement is the NP subject; the head of a VP is a verb; the head of a NP is a noun, and the complement is a determiner. Rules for string rewriting are replaced by information structures called signs which are encoded as attribute-value matrices. Signs can have complex internal structure, including a daughter feature whose value is also a sign. Phrasal signs have head and complement daughters. Using the abstraction of heads and complements, a single schematic rule which states a structural relationship between a head daughter and the related complement daughters of a phrasal structure can be used to characterize the grammaticality of a number of constructions. For example, the string rewriting rules S --> NP VP and NP --> Det

Nom can both be replaced by a single sign in which S or NP corresponds to the sign (the mother), NP or Det corresponds to the complement daughter, and VP or Nom corresponds to the head daughter. The mother sign and its daughters are not marked for specific category information, but do encode constraints on other features of lexical signs and phrasal signs marked with those categories to allow particular combinations.

While HPSG requires a number of grammar rules to cover a substantial portion of English, this number is still less than ten (cf. the context-free covering grammar for the transformational grammar system developed by the MITRE corporation in the 60s had 550 grammar rules (Grishman, 1986)). Other unification based formalisms, like Tree Unification Grammar (Popowich, 1989), require only one rule. The information abstracted away from the rules (the labeling of an object as "N" or "Det") is relocated to the lexicon. Universal principles[6] are also used to state additional constraints on the sharing of information between a mother sign and its daughters. For instance, the *Head Feature Principle* states that the head features of a mother sign (its part of speech, among other things) are identical to the head features of its head daughters. The major category of a sentence is "verb" since that is the major category of its head daughter. Likewise, a VP obtains its major category from its head daughter, a lexical entry marked as a verb. Thus, the schematic rules and principles direct the inheritance of information from the lexicon to more complex expressions. Another interesting aspect of unification grammar formalisms like HPSG is that they are *monostratal* in their representational ontology: lexical entries, rules, and general principles are all stated within a single level of formal representation.

3.1. Signs

The fundamental unit of discourse in HPSG is the *sign*. Signs are represented with attribute value matrices (AVMs) in a simple frame representation language. The attributes and values of a sign contain phonological, syntactic and semantic information of linguistic constituents. For our discussion, we need consider only a portion of these features.

The top level attributes of signs are PHON, SYN and SEM which indicate phonology, syntax, and semantics. Signs are partitioned into phrasal signs and lexical signs. Phrasal signs

---

[6]Pollard and Sag (1987) claim that these hold for all natural languages which can be represented using HPSG.

are distinguished from lexical signs by a top level feature called DTRS which contains constituency information; lexical signs do not have this feature. The value of the DTRS feature is called a *Headed-Structure*, though sometimes phrasal signs are themselves referred to as headed-structures as well. HPSG uses the DTRS feature of phrasal signs to represent the constituent structure of a mother sign in terms of its head daughter (HEAD-DTR) and its complements (COMP-DTRS). There are other types of daughters like adjuncts and filler daughters which need not concern us here.

SYN values are classified into LOC and NONLOCAL features. Important LOC features are HEAD, SUBCAT and LEX. HEAD features record syntactic information usually associated with words, information like major category in linguistic classification, form (relative to category), case, aspects of agreement, and constraints on adjuncts. SUBCAT is a list of other signs, in decreasing order of obliqueness, with which the mother sign needs to combine in order to be saturated with respect to its classification in a subsumption hierarchy. A phrasal sign is saturated if its SUBCAT list is empty. "Obliqueness" refers to an ordering imposed on the constituents of a sentence; "indirect object, direct object, subject," is a listing of constituents in decreasing order of obliqueness (Pollard and Sag, 1987;p.71). Later we will discuss a grammatical principle which constrains the sharing of information between the SUBCAT and COMP-DTRS list of a mother sign and its HEAD-DTR. For now, we indicate that lexical entries within a particular classification (like major category) are marked to subcategorize for signs that contain particular information. A noun subcategorizes for a determiner. The phrasal sign for a NP is saturated if it contains the requisite information about its noun head daughter and its determiner complement. In general terms, a phrase structure categorized by a set of head features is a complete structure when its constituent structure represents each of the signs mentioned on the SUBCAT list of the HEAD-DTR of the mother sign, subject to other constraints imposed by the grammar principles to be described shortly. LEX is a binary valued feature which correlates but does not coincide with the distinction between lexical and phrasal signs (Pollard and Sag, 1987, p.73).

Using these features, we may construct a sign to describe the lexical entry for "did" as in Figure 3-1. Note that we have abbreviated this lexical sign by removing the NONLOCAL and the SEM attributes. The appearance of *VP[BSE]* and *NP[NOM]* in the SUBCAT list indicates that "did" must combine with a base-form verb phrase and a nominative noun phrase to produce a

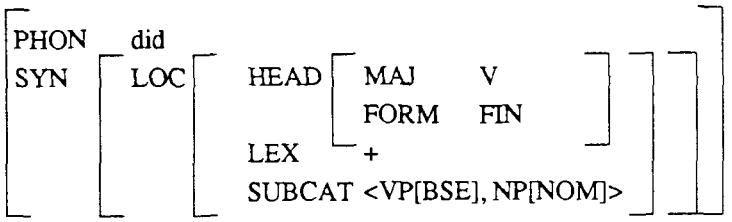
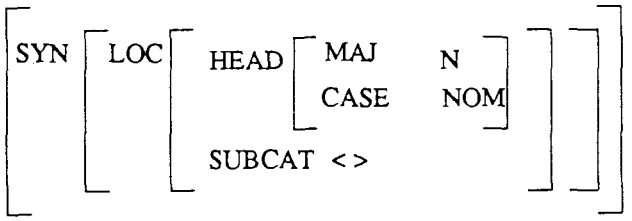
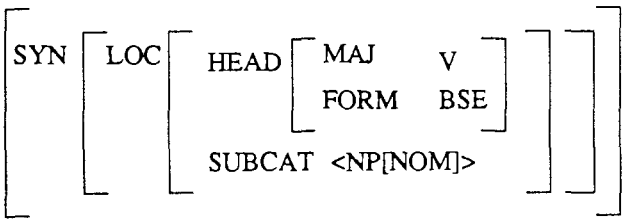


Figure 3-1: Lexical Entry for "did".



NP[NOM]



VP[BSE]

Figure 3-2: Sign Abbreviations.

complete (saturated) constituent. *VP[BSE]* and *NP[NOM]* are actually abbreviations for the signs introduced in Figure 3-2 (Pollard and Sag, 1987, p.69). A similar sign can be used to define a lexical entry for the transitive verb, "kissed". The only differences for the features that we have mentioned here is that *VP[BSE]* would be replaced by *NP*, and, of course, the new lexical sign would have a different phonology. In our discussion, PHON represents just orthography. Signs can be ordered by relative informedness (subsumption) into a lattice structure such that a given abbreviation stands for the class of signs represented by a node in this hierarchy and the class of all subsumed signs. This ordering enables a formal specification of lexical types.

3.2. Lexical Types

A hierarchy of lexical types allows the specification of complex types in terms of less complex types higher in the hierarchy. We say that a sign *A* is more complex than a sign *B* if *A* is subsumed by *B* (i.e., *A* is more informed than *B*). The use of complex types eliminates considerable redundancy from the lexicon since individual lexical entries can be characterized by inheritance over elements from the lexical hierarchy rather than by their complete (explicit) specifications as signs.

Following Pollard and Sag (1987, §8), the lexical type *sign* is the root of lexical hierarchy. This type has two subtypes, *lexical-sign* and *phrasal-sign*. Lexical-signs are either *major-lexical-signs* or *minor-lexical-signs*, with the former partitioned on either the value of their HEAD features (i.e., *noun*, *adjective*, *verb* or *preposition*) or on the value of their SUBCAT features (i.e., *saturated* or *unsaturated*). One can define a subtype like *v-trans* in terms of the types *verb* and *nonempty* which would result in *v-trans* inheriting all of the features specified in those two types and all of the types used to define *verb* and *nonempty*. A particular lexical entry like *walk* could then be defined in terms of *v-trans*, also inheriting supertype information. Inheritance is facilitated by two types of path based reasoning (cf., Touretzky, Horty and Thomason, 1987) over this multiple inheritance hierarchy, *normal* (shortest path reasoning) and *complete* (fully skeptical reasoning) (Shieber, 1986, Flickinger, 1987).

Lexical types are not the only means of structuring information in the lexicon. Dependencies between two lexical entries (or lexical types) can also be stated with the use of *lexical (redundancy) rules*. We will not be concerned with these rules here, as there is some question of their necessity (Flickinger, personal communication).

3.3. Grammar Rules and Principles

HPSG, as a linguistic theory, is a discussion about constraints on the combination of signs. A basic tenet of HPSG is that the HEAD-DTR of a phrasal sign is itself a sign. Syntactic and semantic information is shared between those two signs, as well as among the other complements of the HEAD-DTR, according to the constraints imposed by the few grammar rules and the universal and language specific principles. The rules and principles are expressed as information structures—they are stated in terms of signs just as are lexical entries.

3.3.1. Principles

The grammatical principles independently constrain *information* contained in signs. The Head Feature Principle restricts the sharing of HEAD information between a mother sign and its HEAD-DTR. The Subcategorization Principle mediates SUBCAT information between a mother sign and its HEAD-DTR in terms of the mother sign's COMP-DTRS. The Semantics Principle defines the sharing of SEM information among a mother sign and all of its daughters, as does the English Constituent Ordering Principle for PHON information. The Adjuncts Principle, which plays a key role in the treatment of modifiers, actually violates the formal language used within HPSG since it imposes a relational rather than a functional constraint (Pollard and Sag, 1987, p.163), and for this reason we will not consider it any further at this time.

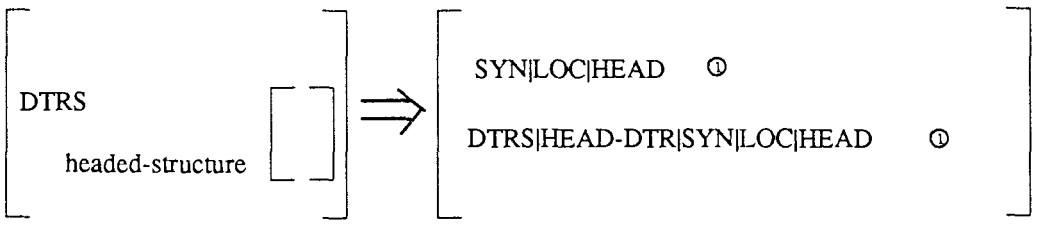


Figure 3-3: Head Feature Principle.

The Head Feature Principle is illustrated in Figure 3-3. Structure sharing of values among features is denoted by a common index. The signs shown in Figure 3-3 follow the abbreviatory convention adopted by Pollard and Sag (1987) which allows reference to values inside nested AVMs by writing the paths of attribute names (separated by vertical bars) which lead to these values. The Subcategorization and Constituent Ordering Principles are abbreviated in Figure 3-4 as signs without the relative pseudocomplement operator (=>). The Subcategorization Principle

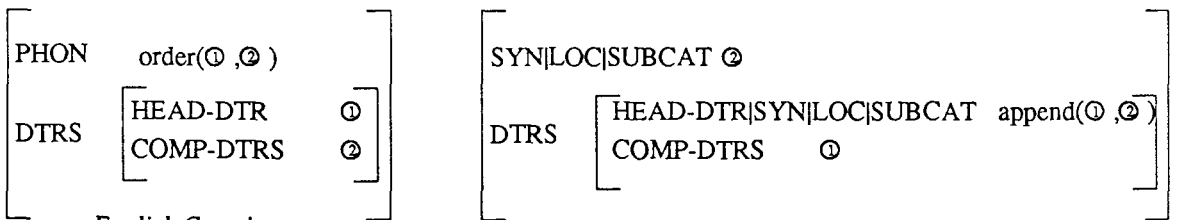


Figure 3-4: Subcategorization and Constituent Ordering Principles.

enforces the following constraint on mother phrasal signs: the first elements (the first elements

are the most oblique elements) of the SUBCAT list of the HEAD-DTR of the mother sign are also the COMP-DTRS of the mother sign; the SUBCAT list of the mother sign takes its value from everything else on the SUBCAT list of the HEAD-DTR (the least oblique elements). Let A, B, C, and D be signs. A mother sign whose SUBCAT list has the value <C, D> and whose COMP-DTRS list has the value <A, B> will also have a HEAD-DTR whose SUBCAT list has the value <A, B, C, D>.

Suppose we have a phrasal sign that is consistent with respect to the Subcategorization Principle. Let that phrasal sign be the mother sign, and let the lexical entry for "kissed", mentioned earlier, be the sign that fills in the value of the HEAD-DTR of the mother sign. Also, suppose that the phrasal sign for "the cat" is the only member of the mother sign's COMP-DTRS list. The mother sign constitutes a partial analysis of the sentence, "Mary kissed the cat." The specification of the value of the SUBCAT list of the HEAD-DTR and the COMP-DTRS list of the mother sign taken in conjunction with the Subcategorization Principle provides enough information to determine that the mother sign still subcategorizes for a constituent structure. The *NP[Nom]* that is the least oblique element (the subject) of the SUBCAT list on the lexical entry for "kissed" is the value of the SUBCAT list of the mother sign. The complete HPSG analysis of the sentence, "Mary kissed the cat," will have the mother sign that we just described as its HEAD-DTR. The mother sign for "Mary kissed the cat," (the grandmother sign) will have the mother sign for "kissed" as its HEAD-DTR, and the COMP-DTRS of the grandmother sign will contain the lexical sign for "Mary" as its single element. The SUBCAT list of the grandmother sign will be empty, as constrained by the Subcategorization Principle.

Figure 3-5 diagrams the structure of the analysis of the sentence, "Mary kissed the cat." Each node in the diagram contains some of the features of the sign represented by that node. The entire diagram corresponds to the grandmother sign for "Mary kissed the cat." Dark arrows point to the sign in the value of a HEAD-DTR feature, and light arrows point to an element on a COMP-DTRS list. The values of some head features for each sign are shown (MAJ, FORM, INV, AUX, NUM, PER, GEN). Elements of SUBCAT lists are represented by the same selection of values for head features that mark nodes. The empty subcat list is indicated with "<>". Figure 3-5 illustrates the assignment of values discussed in the last paragraph. The constituent structure of "the cat" is not depicted.

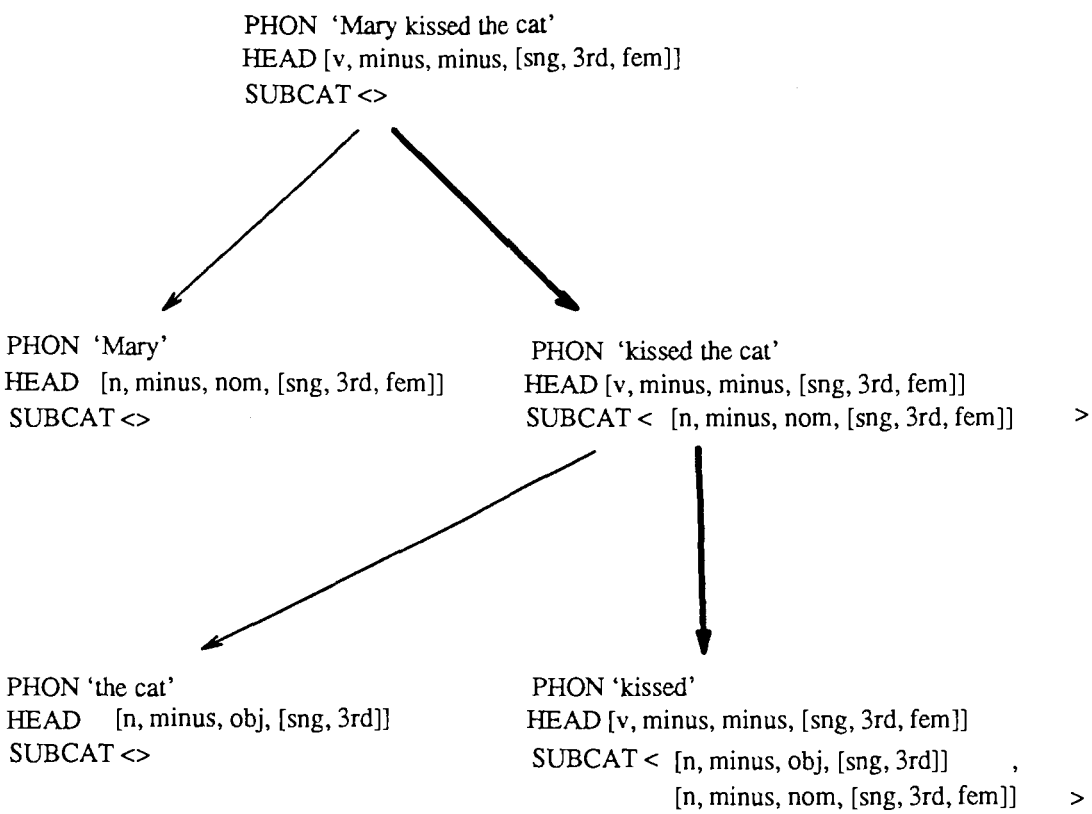


Figure 3-5: Constituent Structure of "Mary kissed the cat."

The Constituent Ordering Principle articulates the constraints between constituent structures of a phrasal sign and the phonology of the phrasal structure. In Figure 3-4 we see that this constraint is stated in terms of an as yet undefined function called "order". Actually, we did indicate some of the constraints imposed by *order* in the introduction to this thesis. Informally, the phonology of a mother sign can be defined as the combination of the phonologies of the constituent daughters (again, we consider only the HEAD-DTR and COMP-DTRS). If the HEAD-DTR is lexical, the phonology of the HEAD-DTR will precede the phonologies of the COMP-DTRS in the combined phonology of the mother sign, just as "kissed" precedes "the cat" in the predicate of the sentence "Mary kissed the cat." If the HEAD-DTR is not lexical, then the phonologies of the COMP-DTRS will precede the phonology of the HEAD-DTR, as "Mary" precedes the phrase, "kissed the cat." These two rules are the basis of the definition of *order*. Some controversy surrounds the exact statement of the principle within the formal structure of HPSG, but this debate is outside the scope of this thesis. A more complete formal discussion can be found in Pollard and Sag (1987).

All of the universal principles are assumed to apply in conjunction with any grammar rule. In the theory, this is achieved through the use of the relative pseudocomplement operator. The details of this operator need not concern us yet, but two example applications of the operator (=>) are shown in Figure 3-6. The net effect of the operator is that the principles all unify with each other into a single relation between a relatively vacuous headed structure (its only specification is that it is a phrasal sign) antecedent and a more fully specified consequent (Pollard and Sag, 1987). The resulting sign contains all of the constraints imposed by each of the principles. The Head Feature Principle as illustrated in Figure 3-3 includes the use of the relative pseudocomplement operator. In Figure 3-4 we omitted the operator and gave only signs produced by the mapping. Any sign which satisfies the constraints stated in the principles will be unifiable with the signs depicted in Figure 3-4. Furthermore, any sign unifiable with the information structure for a rule must also be unifiable with these structures derived from the mapping to be considered valid. We take advantage of the mapping provided by the operator in our implementation of HPSG.

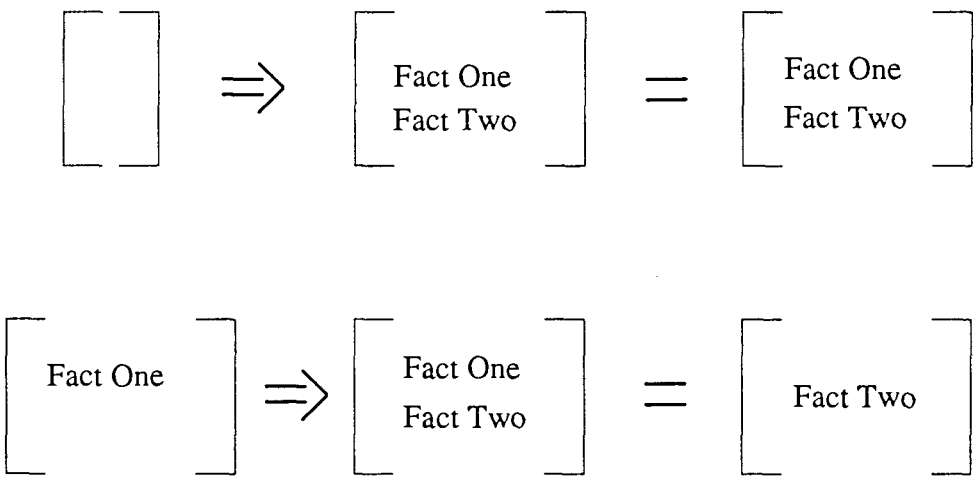


Figure 3-6: Two Example Applications of the Relative Pseudocomplement Operator.

3.3.2. Rules

The "rules" of HPSG constrain the *structure* of constituency. The first rule illustrated in Figure 3-7 is responsible for combining heads with their final complement (i.e., their "subject"). Rule One describes the structure of a saturated phrasal sign (it has an empty SUBCAT list) that has a non lexical HEAD-DTR. The phrasal sign for "kissed John" is an example of a non lexical sign that can fill the value of a HEAD-DTR for such a saturated phrasal sign. The saturated

phrasal sign also has a single complement daughter which, we know from the Subcategorization Principle, is the least oblique complement, the subject of the sentence. The notation <[]> indicates a single element list whose element can be any sign. The second rule is for combining lexical heads with all but their final complements. Rule Two describes the structure of a phrasal

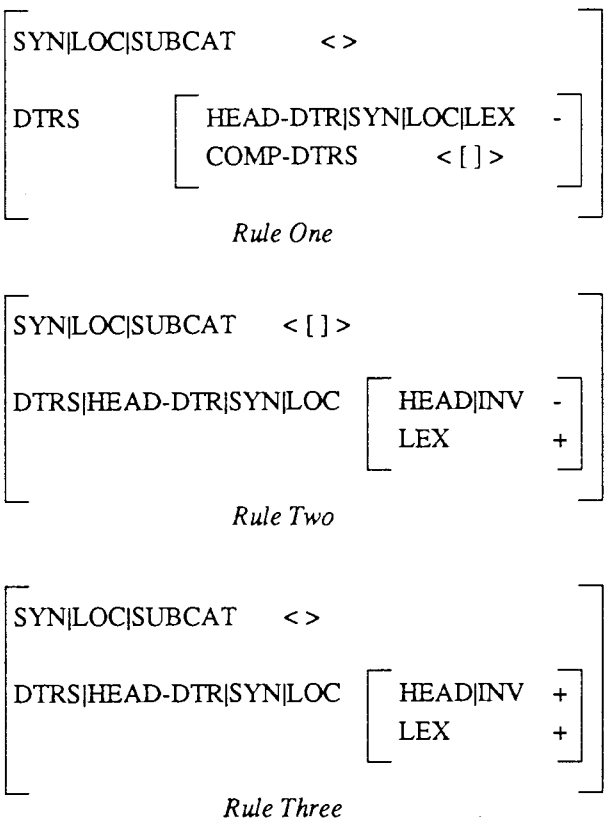


Figure 3-7: Grammar Rules.

sign for an unsaturated constituent of a sentence. The sign for the rule has a single element on its SUBCAT list. When the rule is affixed as the HEAD-DTR of some mother sign the Subcategorization Principle dictates that this element of the SUBCAT list will be the single element on the COMP-DTRS list of the mother. That complement daughter is the final complement. This rule thus describes structures like "kissed John," which have yet to combine with a final complement. The HEAD-DTR is a lexical sign (LEX +), the lexical entry for "kissed". The lexical entry for "John" unifies with the more oblique element of the SUBCAT list on the lexical entry for "kissed", and the remaining element of the SUBCAT list of the lexical sign is the only element of the SUBCAT list of the mother sign. Rule Three coordinates lexical heads in inverted constructions in which lexical heads marked INV + (like "did") combine with all of their complements including the subject. The third rule represents saturated signs for

phrase structures like, "Did Mary Walk?" A fourth rule for treating adjuncts is tentatively put forward in (Pollard and Sag, 1987), but as is indicated (p.165), its necessity is uncertain. Due to the potential problems with the HPSG treatment of adjuncts, in this discussion we have restricted ourselves to the first three grammar rules. However, omitting Rule Four also eliminates constructions like, "Mary walks quickly," or "John kissed Mary on the cheek," from our discussion as well.

In Figure 3-8 we include a phrase structure tree for the sentence, "Mary walks." The tree is drawn with TreeTool (Baker, et al., 1990), but includes some information that is not related to our discussion. Each node represents information contained in some of the values of the corresponding mother sign. The first line at each node represents the value of the PHON attribute, the second includes some HEAD features, and the third line shows HEAD features of the elements on the SUBCAT list of the node. "Mary walks," is an interesting sentence to examine, because even though it is just a two word sentence it has virtually the same constituent structure as the sentence with a transitive verb, "Mary kissed John." The lowest node is the lexical entry for "walks" which is combined with a null COMP-DTRS list according to Rule Two, as discussed above. The mother sign thus created is non-lexical, and by the Subcategorization Principle, it has a most oblique complement remaining on its SUBCAT list, just as the lexical sign for "walks" had a single element on its SUBCAT list. That mother sign can stand as the value of the HEAD-DTR of a grandmother sign that is consistent with respect to Rule One. In this case, the COMP-DTRS list is not empty; it takes as its the single element the lexical entry for "Mary". The resulting grandmother sign is saturated. Its phonology is appropriate for the combination of a non-lexical HEAD-DTR with its complement. Note that a different phonology would obtain if we tried to saturate the grandmother sign with the lexical HEAD-DTR for "walks", instead. In that case, the phonology of the grandmother sign as dictated by the ordering principle would be "Walks Mary." However, that combination is not sanctioned, since Rule One, which such a combination invokes, requires that the HEAD-DTR be non-lexical, as was shown in Figure 3-7.

Each of the grammar rules is a sign that is also subject to the informational constraints imposed by the grammar principles, as we have mentioned. Since all objects of discourse expressed using the formalism (phrase structures, lexical entries, rules, principles) are expressed as signs, a natural way to specify the relationships among these objects and to describe the

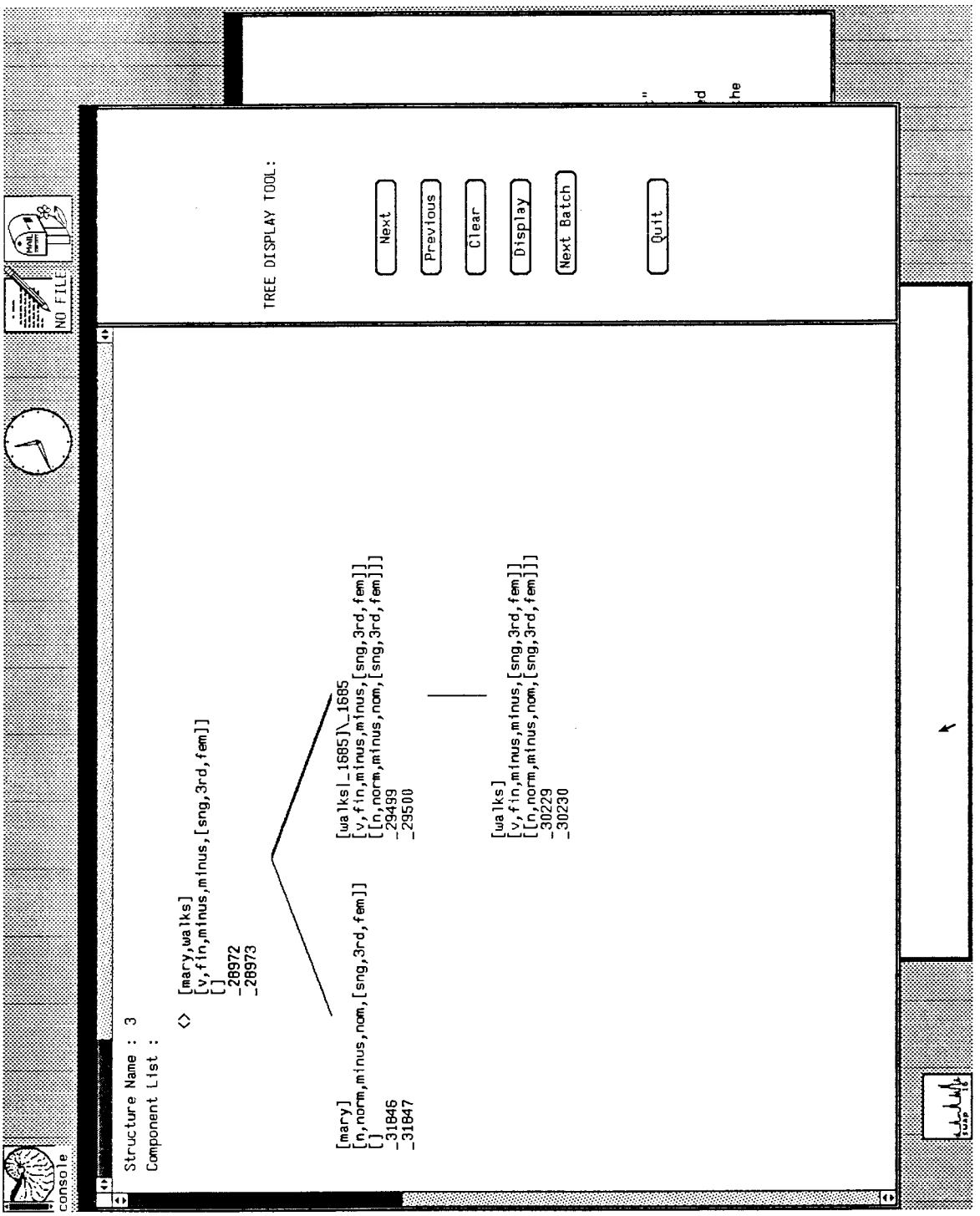


Figure 3-8: A TreeTool Phrase Structure Tree.

accumulation and satisfaction of constraints imposed by each object is to arrange them as nodes in an inheritance hierarchy beyond the lexical hierarchy. Inheritance reasoning over this hierarchy constructs valid HPSG analyses of phrasal structures. The structure of the network so constructed and the significance of reasoning over this network are detailed in the next chapter. Essentially, an inheritance reasoner which operates over the network is a parser for HPSG.

Chapter 4
A non-Trivial Inheritance Network for HPSG

Inheritance reasoning provides a metaphor for describing work done in AI just as systems dynamics, search, and deductive logic have provided ways of articulating problems in the past. This chapter is an analysis, couched in the language of inheritance systems, of the theoretical foundations and implementation of Head-Driven Phrase Structure Grammars. It includes an analysis of the theoretical underpinnings of HPSG as an instance of a two dimensional multiple inheritance hierarchy. The chapter also describes the process of reasoning over this hierarchy. An implementation of the network and reasoner is deferred to Chapter 5. The analysis of HPSG as an inheritance network is based on an interpretation for "is a" links that is quite different from most other approaches to inheritance reasoning. It is different in that the polarity of a link is determined dynamically based upon the exact information contained in the nodes at each end. The interpretation is based on the relative pseudocomplement operator. The relative pseudocomplement of two concepts, as in A=>B, is another concept, C, which consists of inherited material, the information in B that is not in A. An advantage of analyzing HPSG as an inheritance network in this way is that an inheritance reasoner operating over this network constitutes a parser for the formalism.

HPSG parsers have been written before: Proudian and Pollard (1985) developed a chart parser for an early version of the formalism, and a Lisp based parser for the developing formalism has been maintained by researchers at Hewlett-Packard (Gawron, et al., 1982), Prolog and Lisp based chart parsers for later versions of HPSG have been developed by Popowich and Vogel (to appear), McFetridge and Cercone (1990), and Franz (forthcoming). The advantage of pursuing a parsing strategy for HPSG other than chart parsing is the clarity with which the alternative parser might be stated. Once the inheritance network is defined for HPSG, the parser is just a shortest path reasoner that operates over the network. Though chart parsing is quite straightforward and easy to understand, shortest path reasoning is even easier.

In Chapter Two we demonstrated that path based inheritance can be stated concisely in Horn clause logic; it remains to be shown whether an inheritance reasoner can be adapted as a parser in a way that makes it any better than other approaches to parsing. This chapter attempts to address these issues. It is not obvious that inheritance reasoning can be applied, since we have already indicated that the network which we will be examining is cyclic, and that traditional inheritance reasoners are not equipped to handle all cyclic networks. THAT approach does not handle them at all, and Geffner and Verma (1989) handle cyclic networks only if the cycles are negative cycles.

An overriding motivation for this chapter comes from the urge to make analogies. A literature on reasoning and parsing exists already (Porter, 1987, Menzel, 1987, van der Linden, 1989, König, 1989, Evans and Gazdar, 1989, Frisch, 1989). Inheritance reasoning provides another form of inference, so it is hopeful that an analogy will hold between parsing and inheritance reasoning as well. Indeed, Steels and De Smedt (1983) and Brachman and Schmolze (1985) have also noticed parts of this analogy, and we discuss their work as well. But, we feel that ours is a more complete analysis. We outline the network which underlies HPSG, the process of reasoning over the network, and extensions which this approach suggests. For example, we are able to use an inheritance based system to control the inheritance of information to HPSG representations of unknown words that appear in sentences. In the next chapter, we indicate how both the network and reasoner have been implemented in Prolog. We also outline the extension to the network to accommodate unknown words, and provide a corresponding implementation.

4.1. Related Analyses

It is easy to see the relationship between HPSG and frame-based representation languages: HPSG is essentially one of them. Steels and De Smedt (1983) present ideas for syntactic processing based on frame representations of linguistic objects, in which frames for linguistic objects are arranged into a hierarchy. Reasoning over hierarchies created within the framework amounts to the classification of objects, either through *generalization* to find the major category of a word, for instance, or through *refinement* to locate terminal information like the spelling of a word. These two processes are discussed in relation to several example frame descriptions, though the formal details of the representation (to describe with generality the

coverage intended by the representation) and the specification of the reasoning mechanism are omitted. That is, Steels and De Smedt note that a reasoning process over the hierarchy created by a frame based description of linguistic objects can be used to perform functions like constraint verification that are useful to parsing, but they do not give the formal details of such a reasoning process, nor do they commit to a general theory of representation. HPSG provides a general theory of representation for linguistic knowledge, as we saw in the last chapter. Below, we indicate how a network may be abstracted from this theory and detail the reasoning process which serves as a parser for the formalism.

In another frame-based system, Brachman and Schmolze (1985) also describe the application of KL-ONE to parsing. Their system works in unison with RUS, a parser developed by Bobrow (1979) to construct parses which represent syntactic and semantic information. A subsystem called PSI-KLONE is used to translate the syntactic information provided by RUS to concepts represented in KL-ONE.

> The ability to build such interpretations is a test of semantic coherence (if the semantic interpreter fails, the fragment is incoherent); the result of the process is passed back to the RUS parser. RUS does not need to parse an entire sentence before calling upon PSI-KLONE, however, it does impose a certain order upon the fragments it sends:
>
> 1. It first parses enough of a sentence, which is a clause, to find a plausible head verb. PSI-KLONE is informed that a clause has been found with the given head verb and with the remaining constituents unspecified.
> 2. Next, RUS passes the logical subject of the clause to PSI-KLONE. If it must parse further in order to obtain the logical subject, it does so. Otherwise, it does so without further parsing. This strategy of parsing as needed is followed throughout.
> 3. The logical object is passed next.
> 4. Pre-modifiers of the clause are passed, from right-most to left-most.
> 5. Post-modifiers are passed, from left-most to right-most.
> 6. Finally, PSI-KLONE is informed that the clause is complete.
>
> (Brachman and Schmolze, 1985;p.206)

The process described indicates a head-driven sort of algorithm, though it is not as schematically head-driven as is HPSG. "Head-driven" indicates that the parsing algorithm is not concerned with left-to-right processing of the sentence, but inside-out parsing in which the most important parts (i.e., the heads) of phrase structures are located first, and complements are associated with those important parts. Thus, a noun is the head of a noun phrase, and a verb is the head of a clause. A head-driven algorithm is one that finds the head first. The system described by Brachman and Schmolze is much more explicit in referring to objects that can act as heads, rather than just referring to heads schematically. However, that is just a comparison between their

representation of grammar and HPSG, not a statement about how we each use the representations with inheritance. Brachman and Schmolze express interest only in further semantic classification of syntactically identifiable objects. Though they feed this information back to the syntactic processor, they do not recognize the potential for a truly integrated system which represents semantic information in the same way that it represents syntactic information such that inheritance mechanisms can be used to implement both syntactic and semantic classification at the same time during parsing. Brachman and Schmolze do describe the syntactic and semantic recognition processes as executing in parallel, but as distinct processes, without the realization that a single process, the process that they use for semantic classification, can do both. All that is required is a unified representation formalism and the realization that subsequently only a single classification process is required. The representation language is provided by HPSG. Neither Steels and De Smedt (1983) nor Brachman and Schmolze (1985) make explicit the fact that inheritance reasoning over the hierarchy of descriptors (of both syntactic and semantic information) constitutes parsing, and a contribution of this chapter is to make clear that this is the case by elucidating the inheritance network which is implicit in the application of HPSG and specifying the reasoning algorithm required. Further, we point out how easily the analysis may be extended to achieve more robust behavior—the parser is able to construct analyses of many words which are not included in the lexicon.

4.2. HPSG as an Inheritance Network

Two aspects of Head-Driven Phrase Structure Grammar are of particular interest to us: one is its nature as a formal language for expressing a theory of linguistics (i.e., for specifying grammar rules, lexical entries, and universal principles), and the other involves a use of that formal language to describe a specific grammar which actually makes predictions about linguistic phenomena. Both of these views can be captured within the framework of inheritance reasoning. We elucidate from HPSG an inheritance network that represents its conceptualization of linguistic representation, and we demonstrate how both views of HPSG can be obtained through specific sorts of processing over the hierarchy. We describe the peculiar demands that the network imposes on inheritance reasoning systems to yield appropriate results. Finally, we indicate the relationship between inheritance reasoning and parsing.

HPSG is a multiple inheritance hierarchy in two dimensions. The first dimension is

definitional. This *primitive* dimension is used to articulate the structure of signs. The exact specifications are outlined by Pollard and Sag (1987). Pollard and Sag also include discussion of the inheritance which is at work here. The mode of reasoning executed over these networks, at least in the hierarchy of lexical types, is a combination of shortest path and skeptical reasoning. This dimension was discussed in the last chapter, and is the focus of a great deal of research in the application of inheritance reasoning to natural language understanding (Bouma, 1990, Carpenter, 1990, Russell, et al., 1990, Fraser and Hudson, 1990). For a description of the reasoning processes involved specifically in HPSG we point to the work of Shieber (1986) who speculates on the skeptical behavior in the face of inconsistencies, as well as of Flickinger (1985) who describes two corresponding modes called *normal* and *complete* inheritance. We are concerned in the present work with a second dimension of HPSG, its *operative* hierarchy. While the primitive dimension is used to define the structure of signs, the operative dimension relates signs to each other.

4.2.1. Nodes

Signs are the formal constructs that we use to encode the information relevant to linguistic phenomena. HPSG is an interesting formal language because it uses signs both to encode information abstracted from lexical items, phrases, and sentences, and to express constraints on the combination (or decomposition) of signs. Roughly, to be the sign for a specific sentence, the putative sign must *unify* with the sign that expresses the constraints sanctioned by the theory. It is not as simple as this, though, since some of the constraints are expressed through relations between signs stated in terms of the relative pseudocomplement operator (=>). This operator will be described in detail in the next section.

We take signs to correspond to nodes in an inheritance hierarchy. This is reasonable, after all, since signs are attribute-value matrices—they are information structures—and, inheritance networks represent formal encodings[7] of conceptual structures. More precisely, the nodes in the operative hierarchy of HPSG are the signs which encode the grammar's rules and principles. For example, three of the nodes contain the information given in the signs of Figure 3-7, one sign at each node. Nodes are complex structures which can contain signs internally, as well. Recall that

[7]Kasper and Rounds (1986) go so far as to ground these encodings in formal logic.

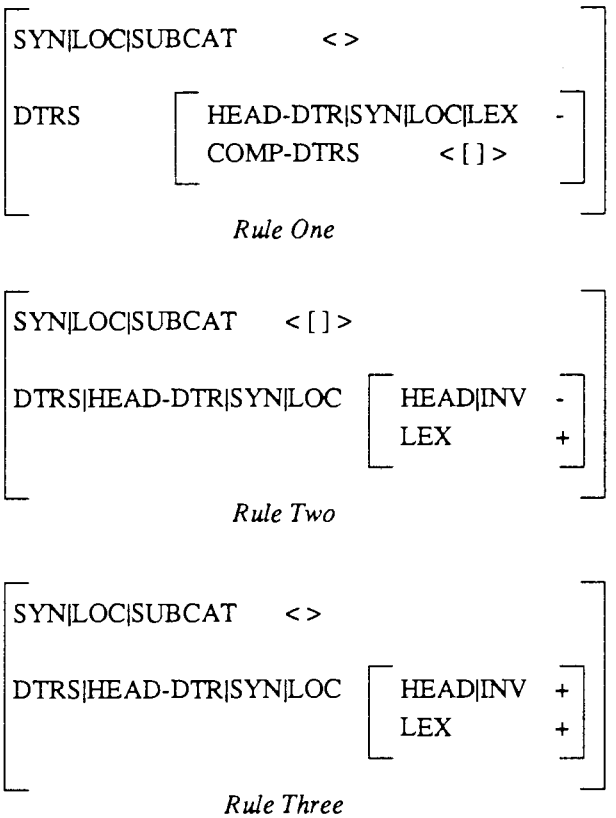
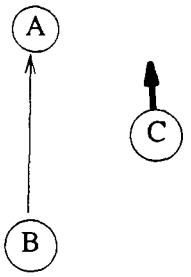


Figure 4-1: Grammar Rules.

the HEAD-DTR of a phrasal sign is itself a sign and that the value COMP-DTRS feature is a list of signs. Nodes can also be lists of signs.

4.2.2. Links

HPSG includes several universal grammar principles which are stated using =>. It is the semantics of this operator that suggests the application of inheritance reasoning to representing HPSG. When this operator is applied to information structures A and B as in, "A => B," it yields an information structure, C, the minimal structure whose unification with A is subsumed by B (Pollard and Sag, 1987, p.43). If A is at least as informed as B, then C is vacuous (it will unify with anything). If B is more informed than A, then C is the information in B that is not in A. In the corresponding inheritance network, A and B are associated with nodes in a network and C is the information that is inherited. The arrowheads on links in Figure 4-2 indicate the direction of information flow. In Figure 4-2, when it is the case that B contains more information than A, C is the information that is inherited. Since the relative pseudocomplement is an operation over a certain type of lattice structure in which $A \vee \sim A = Top$ is not a theorem, for arbitrary feature



A => B

Figure 4-2: Relative Pseudocomplement and the Operative Hierarchy.

structures it is possible that C will not be unique, or equivalently, that it will be an infinite disjunction even though neither A nor B in the expression A => B contains an infinite disjunction. However, Pollard and Sag point out that this does not arise in their application of the operator. Moreover, we are not particularly interested in the inherited object C, in itself. Rather, we are concerned with the inheritance of the information in C to nodes above B in the hierarchy. The node immediately above B is A, so we are interested in the way the information in the relationship between C and A. Particularly, we want to know about the unification of C with A (not with B; see Figure 4-2).

Theorem 1: (Curry, 1963)

$A \wedge (A \Rightarrow B) = A \wedge B.$

Theorem 1 is a useful fact about relatively pseudocomplemented lattices (also called absolute implicative lattices), because it means that the unification of A with C can be calculated simply by taking the unification of A with B without isolating C.

If A and B are inconsistent with respect to each other (A and B fail to unify; $A \wedge B$ yields *bottom*), then the information inherited through C to A causes an inconsistency at that node. Further chaining of links is fruitless, because *bottom* is inherited to A, and *bottom* will be inconsistent with any other node linked to B through A. This situation corresponds to the presence of a negative link in the hierarchy. Given a procedural interpretation, this means that the polarity of a link in the hierarchy with the semantics of => is determined dynamically, depending on the information contained in nodes at either end. Declaratively, we can say that a link is used in a network to represent a relationship between two nodes; the content of the link, its polarity and the information inherited across it, is determined in terms of the stated relation by the information contained at the respective nodes.

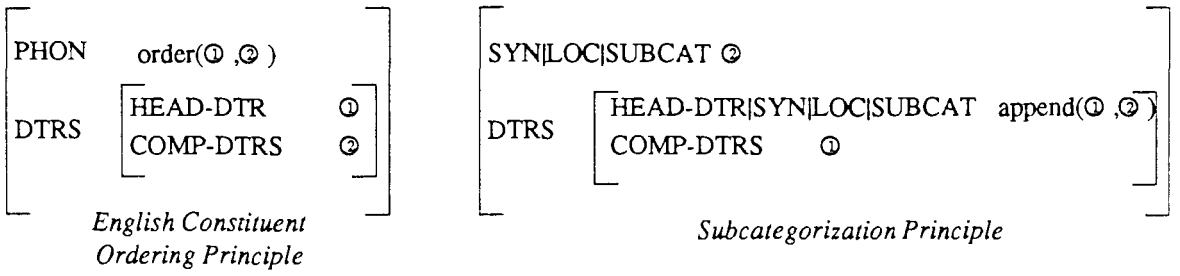


Figure 4-3: Universal Grammar Principles.

In presenting the universal principles of grammar, Pollard and Sag use the relative pseudocomplement operator, and in their usage A is vacuous with respect to a class of signs (headed structures, for instance)—A conveys no information except for the class of signs used as a reference set, but B is also assumed to be a member of this set. This forces the inheritance of all the information contained in B just as in classical implication. This allows us to present the grammar principles as individual signs rather than as relations between signs: the signs in Figure 3-4 show the information that is actually inherited, C, in a couple of the universal grammar principles. In the network, strict links are used to encode the relationship between a vacuous sign and a specified sign.

Our interpretation of links is somewhat different from the interpretation that we encountered in Chapter Two. The difference is in the way we understand polarity as dynamically determined. Aside from that difference, our links have the same function of classification as the links in the upwards de-coupled restricted skeptical reasoner of Horty, et al. (1990): links classify some of the complex structure beneath them in terms of the structures above them. The properties that hold beneath the link do not necessarily all agree with those above the link. Because of our difference on the polarity of links, a more informative English translation of our links than, "is a" might be "can be a". One implication of this difference to which we will return later is that the absence of a link between two nodes makes a strong statement that those two nodes do not relate. When we perform inheritance reasoning over a chain of links we are attempting to determine the polarity of the relationship between its endpoints; we are not attempting to construct an implicit link that can be added to the network. As indicated in the introduction, non-strict links are used to enumerate the ways in which one sign can be classified as another sign. The interpretation of non-strict links is weaker than the typicality interpretation of default links traditionally.

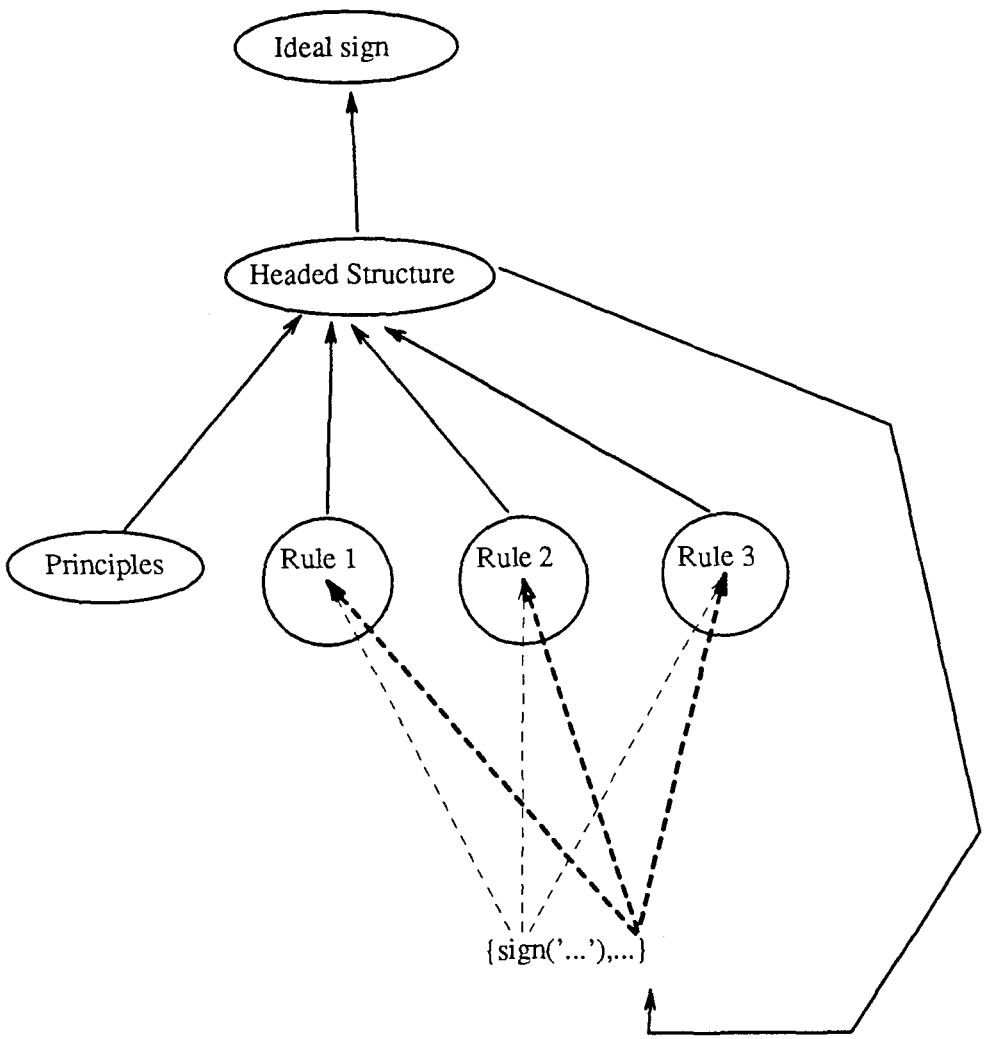


Figure 4-4: The Operative Dimension of HPSG as an Inheritance Hierarchy.

Having described the content of nodes and the semantics of links, we now are in a position to give the inheritance network which corresponds to the operative dimension of HPSG. We actually present the inheritance hierarchy for only that subset of HPSG whose implementation was described by Popowich and Vogel (1990); however, this subset constitutes most of HPSG as presented by Pollard and Sag (1987), and we suspect that when analyzed, the remainder of that incarnation of HPSG will be consistent with the treatment described herein. Observe that the network shown in Figure 4-4 contains several departures from the usual notation for inheritance networks. First we note that there are three sorts of arrows. The dashed arrows are non-strict. This indicates that any given rule may or may not apply to the set of signs below it. A dark

dashed arrow has the semantics of an operator called ->, also a relative pseudocomplement operator, but one which is a generalization of => over lists of signs (i.e., it relates a list of signs to another list of signs). For A->B where A and B are lists of signs, C contains information about the signs in B which is not present in the list A. As we shall see shortly, a path through the hierarchy has breadth as well as length, since a path between the node at the bottom of the hierarchy and the node for one of the rules consists of two parallel links. These links are not redundant as they contain different information. One corresponds to -> and the other to =>. Note that these links point into the nodes for the rules rather than simply to them. This is a useful notion which exploits the fact that inheritance sometimes involves complex concepts at nodes. Inheritance is allowed to a node through component concepts of the node. We refer to such complex nodes as *supernodes*. In reference to HPSG, we can think of a mother sign as a supernode and the HEAD-DTR of a mother sign as a component concept. With respect to the signs presented in Figure 3-7, the list-valued links are for inheritance into the COMP-DTRS attribute, and the other defeasible links point to the HEAD-DTRS attribute. Inheritance through supernodes appears to be equivalent to the inheritance of role values in the extended theory of inheritance reasoning presently being formulated within THAT paradigm (Thomason, personal communication).

A summary of the network follows. Grammar principles are headed structures. Each of the grammar rules is a headed structure. A headed structure can be the sign which describes the complete HPSG analysis of an utterance, and it can be one member of a set of signs that are HPSG analyses for constituent structures of an utterance. A sign can be the HEAD-DTR of any of the grammar rules. A list of signs can be the COMP-DTRS of any of the grammar rules. Other aspects of the hierarchy will become apparent as we explain the process of reasoning over the network. This includes an explanation of the nodes at the top and bottom of the hierarchy, as well as the link which renders the network cyclic. Just as in the inheritance literature detailed in Chapter Two, reasoning is a syntactic operation based upon the construction of valid paths through the network, where the construction of a path is guided by the information contained in the nodes.

4.3. Inheritance Reasoning

We assume the existence of the operative hierarchy which emerges from our language (HPSG) for describing linguistic phenomena. The network can be used to verify the structure of a particular HPSG grammar. We begin with a sentence and a set of features that are relevant and inherent in the structure of that sentence: we know how the sentence is to be classified using the language of signs as if the sentence were a lexical entry. This classification is the "ideal"[8] sign which is at the top of the hierarchy. HPSG gives proposals about the way signs should interact. These are the principles and nodes which constitute the lower nodes in the hierarchy. The bottom node is a partially specified set of signs which can be constituents of the ideal. We use the operative hierarchy to verify that it is possible to trace an inheritance path through the network, from the set of constituent structures at the bottom, beginning with the lexical signs corresponding to words appearing in the sentence, through all of the composite constituent structures, to the structure of the ideal. Recall from Chapter 2 that no path can have a negative link as any but its last link. If a positive path exists, then a sign which is unifiable with ideal sign is also a member of the set at the bottom because of the cyclic link. If more than one path is found, we want to keep all of them, because they all represent valid, if distinct, analyses. That we keep all valid paths, even if they do not agree with each other, indicates that we sanction a form of credulous reasoning — we do not refrain from reaching conclusions when several mutually inconsistent conclusions are available. However, we are most interested in the analysis corresponding to the shortest path. If no path is found, an inconsistency has been inherited through the relative pseudocomplement as was described in Section 5.

Any particular *use* of the operative hierarchy is an exercise in determining the analysis (or, for that matter, the generation) of an utterance expressed in signs, as sanctioned by an HPSG grammar. To demonstrate the form of inheritance reasoning required, we provide an HPSG analysis of a simple sentence, "Mary walks," using the operative hierarchy. The literature on inheritance reasoning systems provides example hierarchies like the one shown again in Figure 4-5 and shows the process of reasoning about its nodes, as in, for instance, determining whether Clyde is a gray thing. So too, for HPSG: we begin with the operative hierarchy and a sentence,

[8]We realize that "ideal" already has a formal definition in reference to partial orders in discrete mathematics, however we use it in its more colloquial sense of "perfect".

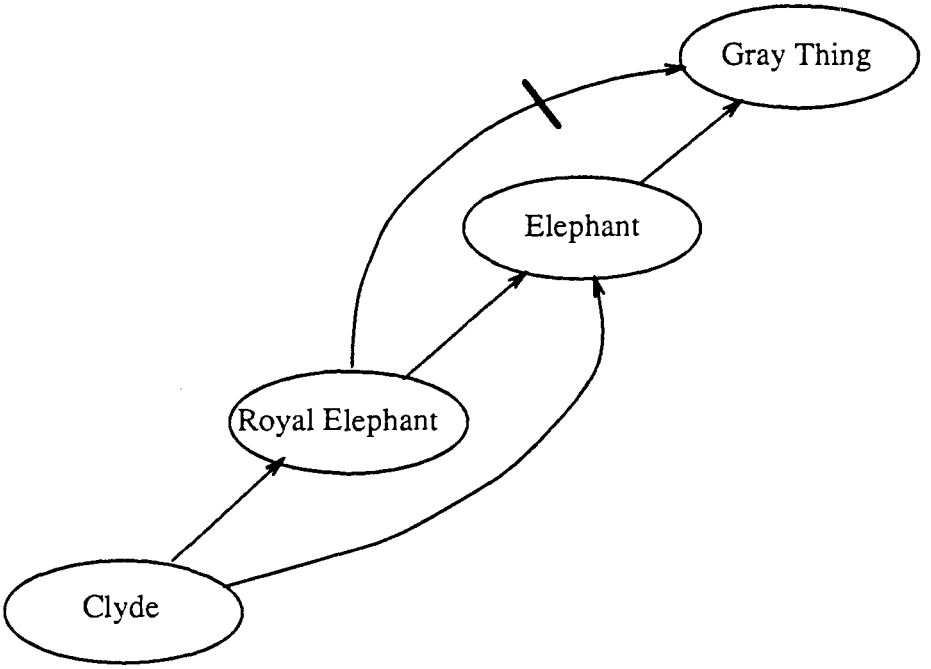


Figure 4-5: Touretzky's Example of a Concept Hierarchy.

"Mary walks", and ascertain whether sign("Mary walks")[9], the node at the top of the hierarchy, is a member of the initially undetermined part of the set of signs at the bottom of the hierarchy, {sign("Mary"),sign("walks"), ...}. If it is not a member then inheritance in the hierarchy yields an inconsistency, and this means that the sentence with which we began is not sanctioned by HPSG as a grammatical utterance.

4.3.1. An Example

Using this example we begin with a partially specified set, {sign("Mary"), sign("walks"),...}. Consider a member of this set, sign("walks"), for instance in Figure 4-6, and recall the sign for Rule One given in Figure 3-7. The HEAD-DTR of Rule One fails to unify with sign("walks"), because the SYN|LOC|LEX feature of sign("walks") is specified as "+".

[9]We allow ourselves the notational ease of selecting the sign relevant to discussion, whether it be from the set of ideals, for complex sentences, or from the lexicon for specific words, by writing it as a function of the appropriate string.

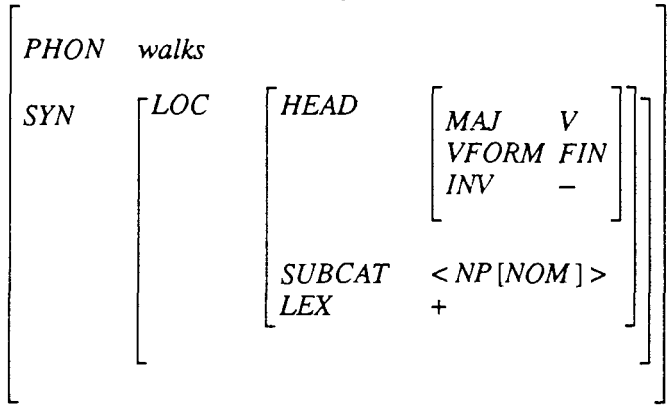


Figure 4-6: Lexical Entry for "walks".

Therefore, we have an inconsistency, a negative link in the hierarchy, and we proceed no further *on that* path (we follow the convention that there can be at most one negative link in a path, and if a negative link exists in a path, it is the last link in the path — recall the earlier mention of futility of chaining past a negative link). But we are still able to consider other paths. The path through the node for Rule Three is also a negative link because of the specification of the SYN|LOC|HEAD|INV feature for sign("walks"). The unification of sign("walks") with HEAD-DTR of Rule Two succeeds. It is important to note, though, that the unification is *not* destructive with respect to the node for Rule Two. The value of the COMP-DTRS feature of Rule Two unifies with the empty list. We consider the inheritance of information from the *result* of that unification with the node for the grammar principles' sign (only one sign, since we constructed from the sign for each principle, the single sign which represents their unification), for that information inherits to the "Headed Structure" node, as well. This unification also succeeds, and though unification with sign("Mary walks") does not, inheritance is admitted along the path to the set of signs at the bottom of the hierarchy, completing a cycle, and further specifying that set — the sign in Figure 4-7 is known to be a member.

This sign will unify with the HEAD-DTR of Rule One (note that it will also unify with the HEAD-DTR of Rule Two since, as mentioned above, the earlier unification was not destructive) since it is not specified for the SYN|LOC|LEX feature at all, and furthermore, <sign("Mary")> will unify with the value of COMP-DTRS on Rule One. The result of these unifications is itself consistent with the node for the principles, hence inheritance is successful to that node. It also happens that sign("Mary walks") is consistent with the result from the last point, which, as before, is also inherited back down to the set of signs that is not fully determined,

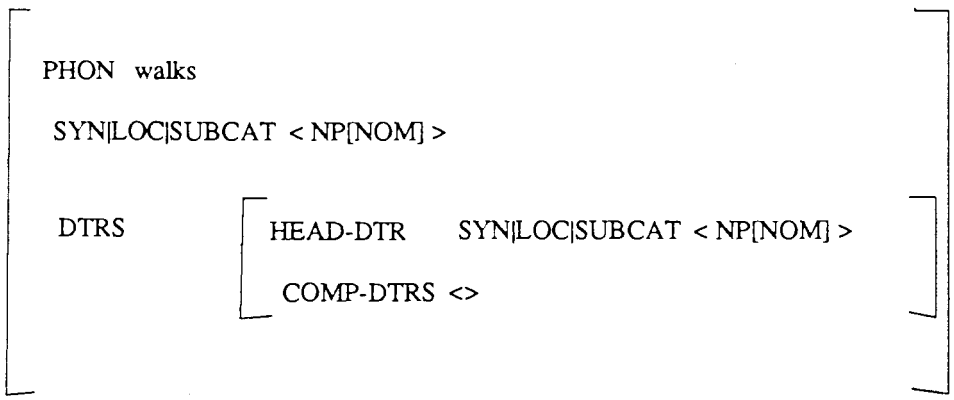


Figure 4-7: Inherited Information.

and this implies that sign("Mary walks") is indeed a member of that set. Hence, what would be considered the "ideal" feature specification of the sentence, "Mary walks," using the primitive hierarchy and the intuitions that the linguists had hoped to capture is actually realized in the HPSG's operative dimension. The sentence, "Mary walks," is considered grammatical. Formally speaking, it is grammatical because at least one path exists from a partially specified set of signs which includes the lexical entries for each word in the sentence, through the concept nodes outlined by the theory, to the ideal sign specification. The presence of more than one such path indicates an ambiguity.

An utterance is ruled ungrammatical only if no path is permitted through the hierarchy. Consider the example utterance, "Walks Mary." Inheritance goes through from the lexical sign for "walks" and the null sublist of analyses to the head and complement daughters, respectively, of Rule Two. The resulting sign is passed through the cyclic link back down, giving further specification to the list of analyses. The phrasal sign for "walks" and the sublist of the analyses containing lexical sign for "Mary" as its single element successfully unify with the head and complement daughters of Rule One, but not with those of any of the other rules. However, there is a strict link between the node for the Principles and the Headed Structure. That inheritance will succeed, and the information inherited through Rule One is consistent with the information inherited from the Constituent Ordering Principle which states that a phrasal head follows its complements. However, this means that the phonology of the sign thus constructed is "Mary walks." This is inconsistent with the specification of the phonology on the ideal sign, causing a negative link. But that inconsistency blocks the only possible path of inheritance, since chaining is abandoned on a particular path after the appearance of an inconsistency and chaining did not succeed through Rule Two or Rule Three. Thus, the utterance is ruled ungrammatical.

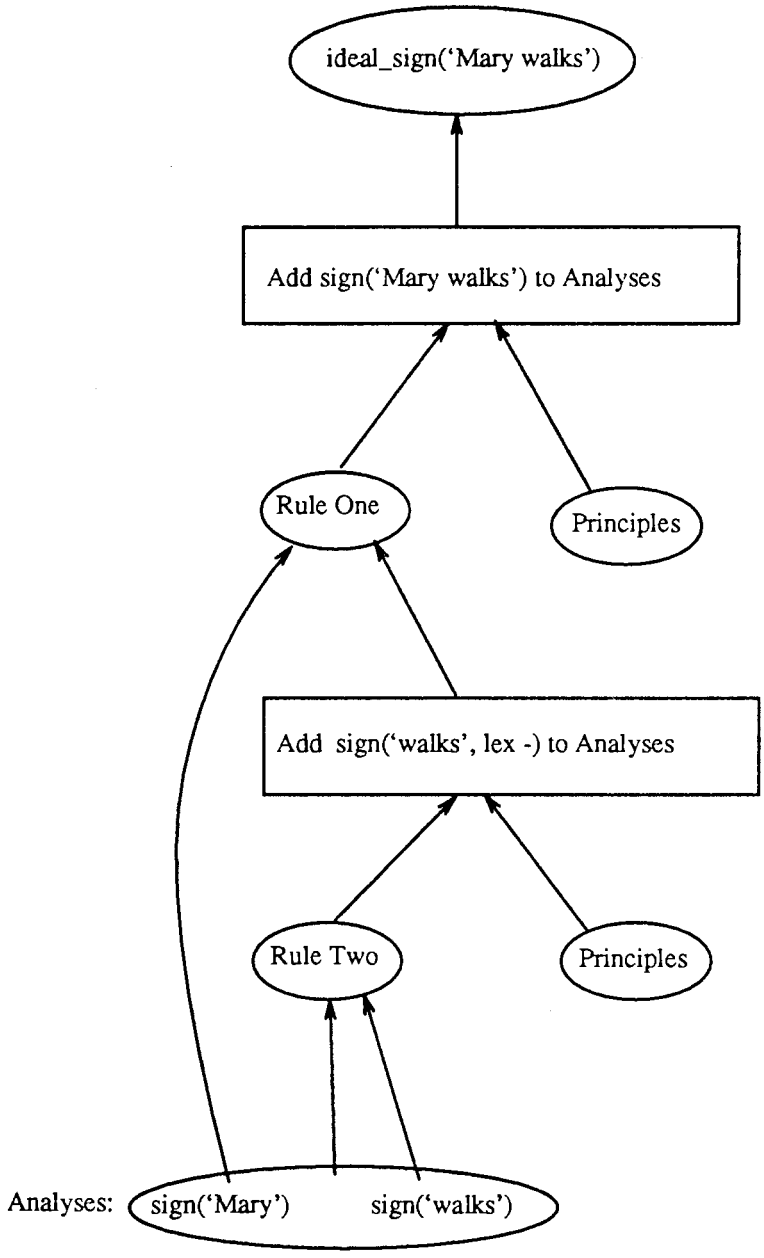


Figure 4-8: A Path Trace through the Operative Hierarchy for "Mary walks".

Informal traces for the paths of reasoning through the operative hierarchy for "Mary walks" and "Walks Mary" are shown in Figures 4-8 and 4-9. Beginning at the bottom, we have the partially specified list of analyses containing entries for each of the words in the sentence. From that node, one link carries the information in the lexical sign for walks to the HEAD-DTR of Rule Two, and the other link carries the empty list to the COMP-DTRS of Rule Two. The destination of both links is inside the node for the rule. The information in the supernode Rule

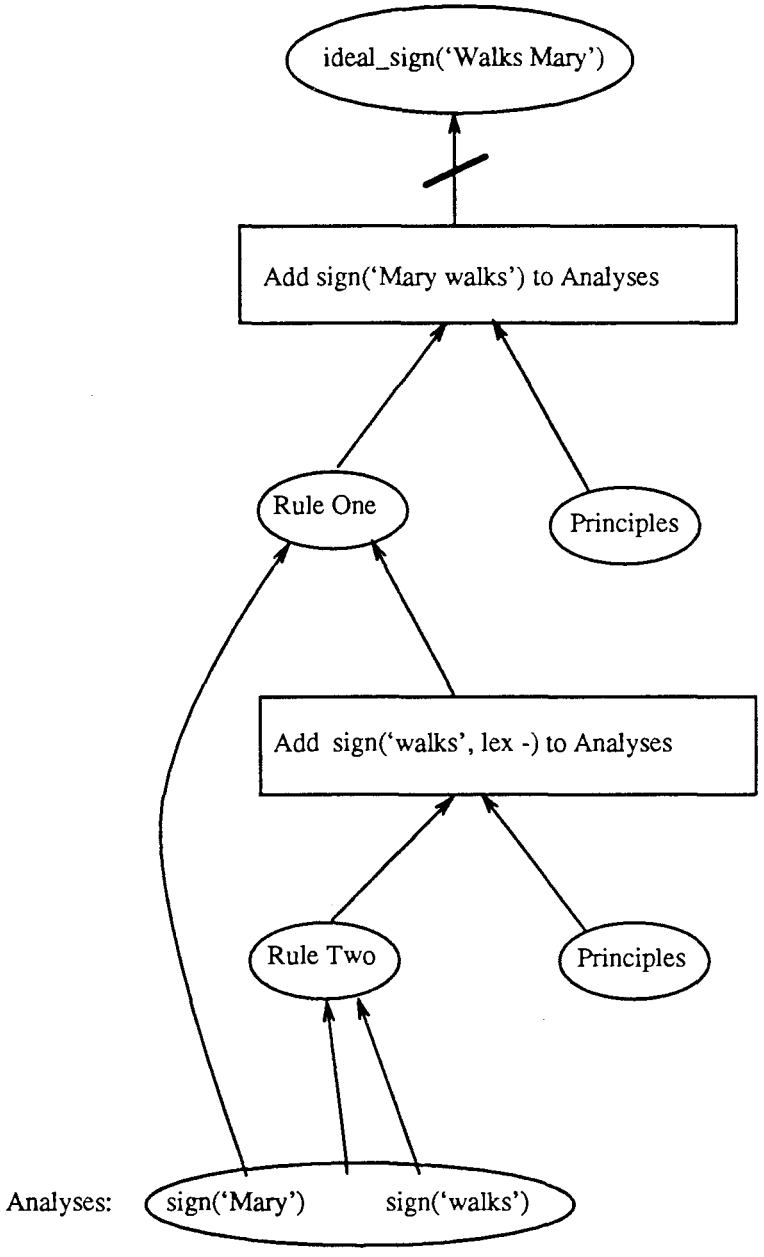


Figure 4-9: A Path Trace through the Operative Hierarchy for "Walks Mary".

Two (including the information inherited from below) is combined with the information in the grammar principles and results in the addition of information to the list of analyses at the bottom of the hierarchy. That cycle is not depicted in the topology of the graph, because it would be difficult to illustrate the sequence of links in the path using graphic cycles. Instead, some nodes are boxed as an indication that a sign is to be added to the partially specified list of analyses. Those directives to augment the specification of the analyses list indicate the traversal of a cycle

in the operative hierarchy. The path of inheritance can thus be traced from the list of analyses at the bottom of the hierarchy to the ideal specification at the top. However, the two figures are not identical. The trace shown in Figure 4-9 shows the discovery of a negative path during the reasoning process. Note that we can construct the same analysis of "Walks Mary" as for "Mary walks". But, this construction is inconsistent with the specification of the ideal sign whose phonology is "Walks Mary." The inheritance does not go through. These traces suggest how reasoning over the operative hierarchy with information specified in the list of analyses can be viewed in a way as involving acyclic paths. A particular sign that appears in the list of constituent analyses will be successfully inherited only once. When a sign is inherited from the list of analyses it is inherited into a supernode, and the information added to the supernode as the result of unification is inherited back down to the list of analyses. If links that classify the list of analyses are understood as emanating instead from individual signs that appear in the list, each sign (node) will occur only once in any particular path. However, this would mean that nodes and links are constructed "on the fly".

4.3.2. Stability

The cyclic structure of the network does not pose the problems that it would for inheritance systems in general, because infinite paths cannot be constructed from the stated hierarchy and because application of this network did not require the resolution of conflicting paths. For this reason, neither is the presence of redundant paths a problem, because multiple paths represent multiple analyses. We do not want to disregard any of them, particularly if those analyses are mutually inconsistent. A collection of paths that are not mutually inconsistent indicates a problem with the grammar; it allows spurious ambiguities. In the network that we have discussed, the addition of a redundancy to the network would be achieved by augmenting the operative hierarchy with the explicit information that the sign for a particular utterance has a particular definition. Really, this means that we add a redundant node to the hierarchy, a new "ideal" node for each utterance added. In such a revised network we are asserting that the ideal sign for an utterance is given either by reasoning over the operative hierarchy, or by simply looking at the redundant entry for that utterance. The only way instability could be introduced is if we add a redundant entry for an utterance which is exceptional in some way (e.g. the sign for "Walks Mary"). If the redundant entry could not have been derived from reasoning over the operative hierarchy (which is what it means for the entry to be exceptional), then different

conclusions will be reached by the reasoner. However, if the redundant entry could not have been derived from reasoning over the operative hierarchy then the entry is not a valid representation of the utterance using HPSG. It is inappropriate to add redundant links to the operative hierarchy, because, as we have discussed, the polarity of each link is determined dynamically depending upon the information stated at both ends. In topological terms, an added redundant link would bypass at least one node in a longer chain of links. But, bypassing nodes means that the information contained in those nodes is excluded from the chain of reasoning. If information is omitted from a chain of reasoning we could indeed obtain unstable behavior. However, the result would again be invalid within the framework of HPSG. The resulting network would no longer be an encoding of HPSG's operative hierarchy.

It is because we know that we will not have to factor out topologically redundant links that we can safely apply the shortest path reasoner to the network. A shortest path still produces a specificity ordering for paths through the network, though it is less complex than the inferential distance ordering of Touretzky (1986). The more complex reasoners outlined in Chapter Two are less applicable than the shortest path reasoner, because those reasoners are intended for networks in which topological redundancy is identical with actual redundancy. Also recall the computational complexity issues raised in the introduction. The computational complexity of shortest path reasoning is a linear function of the number of nodes in the directed acyclic graph, while Touretzky's (1986) system based on the inferential distance ordering has been shown to be NP-Hard (Selman and Levesque, 1989). The system of Geffner and Verma (1989) is NP-hard as well. The reasoner of Horty et al. (1990) is tractable (Selman and Levesque, 1989), but does not appear to be nearly as efficient as the shortest path reasoner. It is therefore fortunate that within the network we require, topologically redundant links do not present a problem.

Chapter 5
Applications of Inheritance Reasoning for HPSG

Recall that there are two aspects of HPSG, grammars written within the formalism, and the formalism itself. Both aspects involve the operative hierarchy, but the distinction is apparent with respect to the sentences we pose against the hierarchy. To use it as a grammar formalism is to use HPSG as a formal language to specify a grammar, which of course may require debugging, so we pose simple structures whose ideal signs are obvious, and we verify that inheritance is permitted from constituent structures. On the other hand, when we use a particular HPSG grammar, we assume that it does not need any adjustment, and we use it to construct signs to discover what they look like and examine its predictions for novel sentences.

The example described for "Mary walks," in Chapter Four demonstrates the use of HPSG's operative hierarchy as a grammar formalism. We began with simple sentences, whose complete signs are apparent, and we verified that the structure of the grammar actually allows the construction of ideal signs from the signs which should be their constituent structures, according to the various principles. If inheritance is not allowed then we know that the grammar has to be adjusted, either in the way information is abstracted into signs in the lexicon, or in the way the combination of information is constrained the principles and rules. It is also possible that further consideration will deem that some paths through the network reflect spurious analyses which need to be ruled out by changing the nodes in the hierarchy. This occurred in our research (Popowich and Vogel, 1990) leading us to a refinement in the statement of Rule Two. Similar tuning of the grammar formalism occurs throughout Pollard and Sag (1987).

In this chapter we focus upon the second use of the operative hierarchy, the construction of signs for novel sentences. This chapter extends the work presented in the last chapter and in Vogel and Popowich (1990). Here we discuss how the network and reasoner outlined in Chapter Four can be implemented in Prolog. We also discuss how the network presented in Chapter Four

can be extended to cope with an extreme form of novel input by representing the inheritance of information to signs for words that are unknown to the lexicon. We provide a corresponding extension to our implementation of the network. Throughout, we indicate the limitations of the theoretical approach and of our implementation.

5.1. Parsing

Clearly, the second use of the operative hierarchy is closely related to parsing. Parsing a sentence is the determination of its constituent analysis just as in the second use of the operative hierarchy is to construct a sign formulation for a sentence (if one exists). So, a machine implementation of an HPSG grammar is a machine implementation of a reasoning mechanism which operates over the HPSG operative hierarchy. Of course, it is valid (and a claim that we in fact make when we say that we have constructed a *tool* which is useful to linguists) to point out that a parser is useful for the grammar verification usage of the hierarchy as well, since if a sign cannot be built for a sentence that should be covered (or if the parser constructs a sign for a sentence that it should not), then we have used the parser to discover errors in the grammar's predictions, and a problem exists with the grammar that needs to be remedied. But, in the abstract, the problem is patched by determining the structure that should be present, the nature of the ideal sign, and how the nodes in the hierarchy need be modified in order for inheritance to go through.

5.1.1. An Implementation

We present a straightforward implementation in Prolog of both the operative hierarchy and a reasoner which operates over that network. The primitive hierarchy and reasoning over it are taken from Popowich and Vogel (to appear). The reasoning is a process of unrestricted skepticism in reasoning over a hierarchy of lexical types. The operative hierarchy is encoded as a set of links annotated with the restrictions entailed by the rules and principles. Shortest path reasoning over this hierarchy is then just simple path construction through the net. The inheritance of information is accomplished through Prolog's built in unification and with shared variables.

The reasoner is given in Figure 5-1. This program uses the backtracking control structure of Prolog to build a path between two nodes in a graph. Since the control structure of Prolog is

```
%link(From,To).

inherit(Composite,[],Classification) :-
        link(Composite,Classification).
inherit(Composite,[OutsideTo|Through],Classification) :-
        link(Composite,OutsideTo),
        inherit(OutsideTo,Through,Classification).
```

Figure 5-1: An Inheritance Reasoner in Prolog.

depth first, the reasoner is not guaranteed to produce the shortest path for an arbitrary directed acyclic graph, nor is it guaranteed to terminate for an arbitrary directed graph with cycles. However, for the specific network that we need to process, the reasoner produces the desired results. Links in a graph are represented using the term notation, *link(From,To)*. The first *inherit* predicate states that a path of inheritance between two nodes exists if there is a direct link between those two nodes. The recursive rule states that there is a path of inheritance between two nodes A and B if there is a direct link between one of the nodes (A) and some other node and if a path of inheritance exists between that other node and B. Consider the toy network shown in Figure 5-2 which represents just the topology of the operative hierarchy. Reasoning over this

```
link(analysis,rules_and_principles).
link(rules_and_principles,ideal).
link(rules_and_principles,analysis).
```

Figure 5-2: A Prolog Network for the Topology of the Operative Hierarchy.

network produces paths in increasing order of length. The shortest path is found first, as depicted in Figure 5-3

```
| ?- inherit(analysis,Through,ideal).

Through =
[rules_and_principles] ? ;

Through =
[rules_and_principles,analysis,rules_and_principles] ?
```

Figure 5-3: Reasoning through the Toy Network.

A short Prolog program which invokes the reasoner on the actual Prolog encoding of the network is given in Figure 5-4. The final line of code shown in Figure 5-4 specifies the actual invocation of inheritance from the partially specified list of analyses (asserted to the database inside the term, *analysis*) to the sketch of the ideal sign (asserted inside the term, *ideal*). The variable *Through* is instantiated as the list of nodes that make up the shortest path between the analyses and the ideal sign.

The structure of the full network is given in Figure 5-5. The complexity of the link

```
parse :-
  abolish(ideal,1),abolish(analysis,1),abolish(real,1),
  read_sent(WordList),
  init_list(WordList,Analysis),
  asserta(analysis(Analysis)),
  sketch_ideal(WordList,Ideal),
  asserta(ideal(Ideal)),
  inherit(analysis(Analysis),Through,ideal(Ideal)).
```

Figure 5-4: Invoking Inheritance.

structure in Figure 5-5 (rather than simply "link(From,To)"), allows the encoding of the more interesting features of nodes in the hierarchy that we described earlier (e.g., a link can connect a node to another node that is inside a larger node). The additional complexity in the actual nodes, as we shall see, increases the complexity of the reasoning process. Notice that only three links

```
link(analysis(Analysis),rules(Rule)) :-
        analysis(Analysis),
        rules(Rule),
        path(Rule,dtrs:head_dtr,Head),
        element(Element,Analysis),
        sign_unify(Head,Element),
        path(Rule,dtrs:comp_dtrs,Comps),
        select_from(Elements,Analysis),
        sign_unify(Elements,Comps).
link(rules(Y),ideal(Ideal)) :-
        nonvar(Ideal),
        sign_unify(Y,Ideal),
        graph(c,[Ideal]),
        asserta(real(Ideal)).
link(rules(Rule),analysis([Rule|Analyses])) :-
        analysis(Analyses),
        not(member(Rule,Analyses)),
        retract(analysis(Analyses)),
        asserta(analysis([Rule|Analyses])).

rules(Rule) :-
        rule(N,Rule).

select_from(Y\Y,Analysis).
select_from(Elements,Analysis) :-
        subset(Subset,Analysis),
        convert2dl([Element],Elements).
```

Figure 5-5: The Operative Hierarchy in Prolog.

are indicated. Some of the individual links have been coalesced. For instance, there is not a separate link for each of the rules since they can be represented by a single clause that represents all of them. An individual rule is selected as a possible path to take through the term *rules(Rule)*. This is done to allow a simpler Prolog representation of the network. An implementation which codes each link separately is shown in Figure 5-6. An examination of the amount of code that is identical among the definitions makes clear why we opted to combine them into a single definition.

```
link(analysis(Analysis),rules(Rule)) :-
        analysis(Analysis),
        rule(1,Rule),
        path(Rule,dtrs:head_dtr,Head),
        element(Element,Analysis),
        sign_unify(Head,Element),
        path(Rule,dtrs:comp_dtrs,Comps),
        select_from(Elements,Analysis),
        sign_unify(Elements,Comps).

link(analysis(Analysis),rules(Rule)) :-
        analysis(Analysis),
        rule(2,Rule),
        path(Rule,dtrs:head_dtr,Head),
        element(Element,Analysis),
        sign_unify(Head,Element),
        path(Rule,dtrs:comp_dtrs,Comps),
        select_from(Elements,Analysis),
        sign_unify(Elements,Comps).

link(analysis(Analysis),rules(Rule)) :-
        analysis(Analysis),
        rule(3,Rule),
        path(Rule,dtrs:head_dtr,Head),
        element(Element,Analysis),
        sign_unify(Head,Element),
        path(Rule,dtrs:comp_dtrs,Comps),
        select_from(Elements,Analysis),
        sign_unify(Elements,Comps).
```

Figure 5-6: A Different Encoding of Three Links.

Recall from the discussion of links that appeared in Chapter Four, if a given link represents A => B and if A is vacuous, then all of the information contained in B is inherited. We take advantage of this fact by simplifying the operative hierarchy so that its vacuous node is replaced with one containing the information that the vacuous node is guaranteed to inherit. In Figure 4-4 the node for the *Headed Structure* (also depicted as a sign in the Head Feature Principle as shown in Figure 3-3) is vacuous with respect to the class of signs denoted by the unification of the signs to the right of the relative pseudocomplement operator in the original statement of each of the grammar principles. This means that all of the information in the node for the *Principles*, as well as for each rule, is inherited to the *Headed Structure* node. We take advantage of this by actually implementing a simplified hierarchy in which inheritance into the vacuous node has been "preprocessed". The simplified hierarchy is depicted in Figure 5-7. It is easier then to see the relationship between the code given in Figure 5-5 and the operative hierarchy. There is a complex link between the node for the list of analyses and the node for the rules which has the node for the principles inherited into it. There is a link between the node for

the rules and the ideal sign. Finally, there is a link from that node back down to the list of analyses. The Prolog code maps directly to the hierarchy in Figure 5-7 which condenses the topological structure of the operative hierarchy.

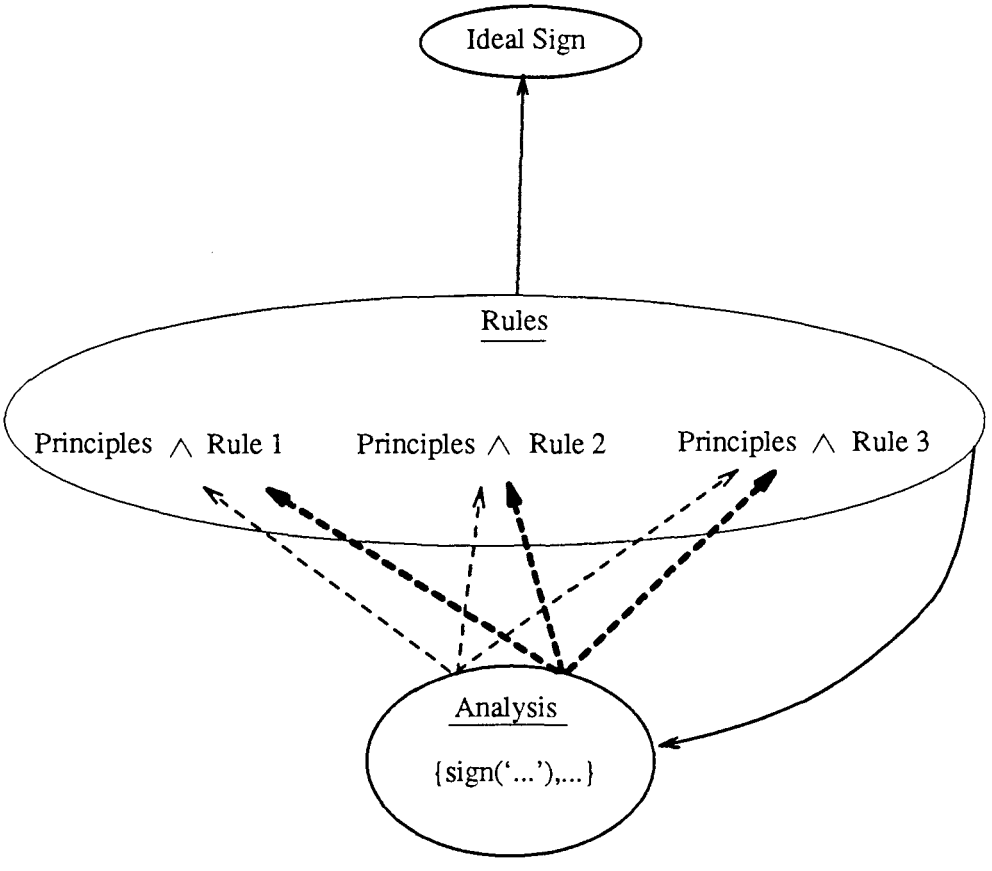


Figure 5-7: Simplified HPSG Inheritance Hierarchy.

Consider the first of the *links*. We have a link from an *element* of the list of analyses to the *head_dtr* of the sign for some rule. The value of the *head_dtr* is referenced using a *path* equation applied to that rule. A *path* equation is a relation which verifies that a particular value lies at the end of a list of nested attributes in a sign. The same equation can also be used to retrieve the value associated with some attribute deep within a sign. The expression *sign_unify(Head,Element)* verifies the unification of the *head_dtr* and the *Element* in accordance with the way we described the calculation of inheritance for A => B in the last chapter[10].

[10]Given A => B, we are interested in A ∧ C where C is is the inherited information; this can be computed with A ∧ B.

Similarly, a sublist of the list of analyses is joined to the *comp_dtrs* feature of the same rule. This *link* predicate coalesces the set of dashed links connecting the list of analyses to the next node in the hierarchy shown in Figure 5-7. The second *link* connects the node for the rules to the ideal sign. For that link, if inheritance goes through we construct a graphic representation (an example of this output as processed by TreeTool (Baker, et al., 1990) is included later in Figure 5-16) and assert the fact that an analysis has been constructed. The final *link* predicate is the link from the rules back down to the list of analyses. We add a sign inherited to this list simply by retracting the old list and asserting the more fully specified list.

Figure 5-8 depicts the application of the inheritance reasoner to the parsing of three

```
| ?- parse.
|: mary walks.
rules (with [mary,walks])  --->
      ideal (with [mary,walks]).
analysis ---> rules (with [mary,walks]).
rules ---> analysis (with [walks]).
analysis ---> rules (with [walks]).

Parse Found at Time: 1.77 secs.

yes

| ?- parse.
|: mary loves john.
rules (with [mary,loves,john])  --->
      ideal (with [mary,loves,john]).
analysis ---> rules (with [mary,loves,john]).
rules ---> analysis (with [loves,john]).
analysis ---> rules (with [loves,john]).

Parse Found at Time: 2.53 secs.


| ?- parse.
|: mary loves several cookies.
rules (with [mary,loves,several,cookies])  --->
      ideal (with [mary,loves,several,cookies]).
analysis ---> rules (with [mary,loves,several,cookies]).
rules ---> analysis (with [loves,several,cookies]).
analysis ---> rules (with [loves,several,cookies]).
rules ---> analysis (with [several,cookies]).
analysis ---> rules (with [several,cookies]).
rules ---> analysis (with [cookies]).
analysis ---> rules (with [cookies]).

Parse Found at Time: 9.09 secs.

yes
```

Figure 5-8: Three Sentences Parsed by an Inheritance Reasoner

sentences. Parsing was invoked with the command *parse*, and a sentence was entered. Beneath each sentence is a listing of system output. The output includes a trace of the links followed through the simplified version of the operative hierarchy shown in Figure 5-7. The links are listed in the reverse order of their traversal, so the actual path must be read from the bottom up. Times required for parse completion are shown beneath the trace of links.

Inheritance reasoning over a lexical, unification-based grammar formalism is clearly a straightforward process, but the current implementation is not particularly efficient. The timings shown in Figure 5-8 indicate exponential growth. Recall the large underspecified node at the bottom of the hierarchy (represented by the term *analysis*); the process of choosing elements and subsets of that node to try to inherit is computationally expensive. That process, which is implemented in the *select_from* predicate illustrated in Figure 5-5, refers to the *subset* relation. Choosing arbitrary subsets of an ever growing list is an expensive operation. One heuristic can be devised by noticing that a great number of the grammatical structures represented using HPSG involve the combination of a HEAD-DTR with a single complement. This implies that subsets taken from the list of analyses need only have one element. Thus, the definition of *select_from* that we actually use chooses potential complements accordingly. The definition is given in Figure 5-9. Note that this redefinition is merely heuristic, though, since for analyses involving

```
select_from(Y\Y,Analysis).
select_from(Elements,Analysis) :-
        element(Element,Analysis),
        convert2dl([Element],Elements).
```

Figure 5-9: Heuristic Selection of Analyses.

Rule Three, a HEAD-DTR combines with two complement daughters. The existing system no longer covers Rule Three since it can accommodate only a HEAD-DTR and a single COMP-DTR. However, it would be straightforward and still more efficient than the original definition to allow subsets of restricted size (up to two elements, for instance) of the analyses to be examined. The subset selection function does not need to be generalized at all to accommodate other unification grammar formalisms. In Tree Unification Grammar (Popowich, 1988) and Unification Categorial Grammar (Calder, Klein, and Zeevat, 1988) all branching is binary, so single element subsets would satisfy all possible constituent structures. While our restriction is heuristic for HPSG it is sufficient for some other UGs. An additional, global restriction can be added to the network to guarantee that the only partial analyses ever constructed are analyses that represent surface structure strings which are actually substrings of the original sentence. The

```
almost_ideal(Rule) :-
        path(Rule,phon,order(A,B)),
        order(A,B,BuiltPhon),
        undiff(BuiltPhon,String),
        ideal(Sign),
        path(Sign,phon,IdealPhon),
        substring(IdealPhon,String).
almost_ideal(Rule) :-
        path(Rule,phon,BuiltPhon),
        undiff(BuiltPhon,String),
        ideal(Sign),
        path(Sign,phon,IdealPhon),
        substring(IdealPhon,String).
```

Figure 5-10: Additional Heuristic Selection of Analyses.

code which implements this restriction is given in Figure 5-10 and can be invoked either from the link which connects analyses to rules or from the link leading from the rules back down to the analyses.

Another source of gross inefficiency is our representation of signs themselves. Note that the definitions of *select_from* in Figures 5-5 and 5-9 reference difference lists. Some values in signs are represented as difference lists, following Popowich and Vogel (1990). The values of PHON, SUBCAT, and COMP-DTRS are all represented in this fashion. While this allows efficient invocation of the *append* function which is used in the Subcategorization Principle (c.f., Figure 3-4), the behavior of difference lists requires that the occurs check be built in to the unification of two signs. This is one reason why the links as implemented in Figure 5-5 invoke *sign_unify* rather than using just Prolog's built in unification and shared variables to accomplish this unification. Another reason is that under the present encoding of signs, other functions besides *append* must also be evaluated. The *order* function which relates the constituents of a sign to its phonology must also be evaluated when unifying phrasal signs that have been inherited through the node containing the information in the grammar principles. The *order* function is invoked during the unification of two signs, and for more complex signs, this unification takes much more time. A better representation of functions used as values in sign would greatly enhance the efficiency of the system.

5.2. Inheritance Reasoning versus Chart Parsing

We need to compare the parser that is constituted by an inheritance reasoner over the HPSG operative hierarchy with other parsing methods. In theory, since we have constructed a concept hierarchy specifically for HPSG, the inheritance reasoner should be extremely efficient in constructing paths, because all we need is a shortest path through the hierarchy. However, complexity is introduced because inheritance is determined across a link by comparing the information contained in nodes at either end. So, the complexity of the reasoner is determined by the complexity of the process for determining the unification of A and B. In our system we have chosen a representation for signs which is not particularly space or time efficient.

In comparison with the algorithm of a chart parser for HPSG (Popowich and Vogel, 1990), the inheritance parser is less efficient. In a chart parser, a sentence is represented by a graph, where nodes in the graph represent positions between words in the sentence being parsed. Edges correspond to analyses of the words and complex structures derived from the words according to the grammar being processed. An edge is marked with the sign (also called the edge's category) for the analyses the edge represents. The endpoints of an edge indicate its position in the chart and the span of its analysis. New edges representing analyses of larger constituent structures of the sentence are introduced to the chart through a waiting list as the product of one of two processes. Rule Invocation determines that the category associated with some edge satisfies the HEAD-DTR of some grammar rule and creates new edges for each rule satisfied in this fashion. These new edges are placed on a waiting list to be processed. The new edges each have an associated list of expectations, signs that the new edge needs to combine with, which is taken from the COMP-DTRS feature of the *sign that marks the new edge.* The Completer compares the next edge (also called the current edge) on the waiting list with the edges in the chart, looking for edges that "meet" the current edge satisfying some of the expectations of one of them. A new composite edge with some or all of its expectations satisfied is created and added to the waiting list. When an edge is created that spans the chart and has all of its expectations satisfied a successful parse has been created.

The chart parsing process is time efficient, though the parser presented in Popowich and Vogel (1990) has unification expenses similar to the inheritance reasoner's built into its subprocesses. The unification employed is built on top of regular Prolog unification to include

the occurs check. This is not a linear time process. During Rule Invocation, the test of satisfaction between the sign marking an edge and the HEAD-DTR is unification. The signs for the grammar principles are assumed to be unified into the signs for each of the rules prior to considering any sentences, so the mother sign which results from the unification is assumed to be consistent with the grammar principles as well as the successful rule. If edges satisfy the meets condition then the Completer step also involves unification to verify that the sign marking one of the edges satisfies a sign on the expectations list on the other.

In the inheritance reasoner the functions of both the Rule Invocation and the Completer steps of the chart parser are taken up by the link from the list of analyses to the node for the rules. One sign is selected from the list of analyses as a potential HEAD-DTR and a sublist of signs is selected from the list of analyses as potential COMP-DTRS. Both selections require verification through unification, just as in the chart parser. However, the chart parser makes good use of the "meets" condition as a preliminary test for whether the unification with the COMP-DTRS is likely to go through. If edges do not meet further unification is not attempted. The inheritance reasoner lacks such a condition. Moreover, selecting sublists of the list of analyses is not an efficient process in itself, as was mentioned earlier. This is exacerbated by the fact that the list of analyses grows during the reasoning process—there are increasingly more sublists to consider, including sublists that has been considered for previous constituent analyses. On the other hand, the chart parser takes advantage of a distinction between edges in the chart and edges waiting in a list to be processed. The chart parser selects one edge from the waiting list at a time, and though the edges archived in the chart can cause the creation of new edges by combining with the current edge during the Completer step, an edge that is archived in the chart never have to be re-examined for the Rule Invocation step. Mike Reape (personal communication) has designed an algorithm which functions very much like the inheritance process described herein, but instead of a list of analyses, his system maintains a bag of unused constituents. Once a sign is used as a daughter in a new sign, the new sign is added to the bag and the old sign is thrown away.

Compare the results obtained by the inheritance parser on a sentence, "Mary loves several cookies," with the results of the same sentence processed by the Prolog chart parser of Popowich and Vogel (1990). Output from the chart parser is shown in Figure 5-11, and the results of the inheritance reasoner are given in Figure 5-12. These two systems are comparable because the are both written in Prolog and based upon the same lexical hierarchy, and they both offer the same

```
| ?- parse.
|: mary loves several cookies.
initializing: place inactive edge 1 ([mary])
              in waitinglist.
initializing: place inactive edge 2 ([loves])
              in waitinglist.
initializing: place inactive edge 3 ([several])
              in waitinglist.
initializing: place inactive edge 4 ([cookies])
              in waitinglist.
predictor: place inactive edge 5 ([cookies])
           in waitinglist (used rule 2).
predictor: place active edge 6, built from [loves]
           in waitinglist (used rule 2).
completer: place inactive edge 7 ([loves,mary])
           in waitinglist.
predictor: place active edge 8, built from [cookies]
           in waitinglist (used rule 1).
completer: place inactive edge 9 ([several,cookies])
           in waitinglist.
predictor: place active edge 10, built from [loves,mary]
           in waitinglist (used rule 1).
completer: place inactive edge 11
           ([loves,several,cookies]) in waitinglist.
predictor: place active edge 12, built from
           [loves,several,cookies] in waitinglist
           (used rule 1).
completer: place inactive edge 13
           ([mary,loves,several,cookies]) in waitinglist.

Parse Found at Time: 4.931 secs.

yes
```

Figure 5-11: A Chart Parse of "Mary loves several cookies."

```
==================================================
| ?- parse.
|: mary loves several cookies.
rules (with [mary,loves,several,cookies])  --->
      ideal (with [mary,loves,several,cookies]).
analysis ---> rules (with [mary,loves,several,cookies]).
rules ---> analysis (with [loves,several,cookies]).
analysis ---> rules (with [loves,several,cookies]).
rules ---> analysis (with [several,cookies]).
analysis ---> rules (with [several,cookies]).
rules ---> analysis (with [cookies]).
analysis ---> rules (with [cookies]).

Parse Found at Time: 9.14 secs.

yes
```

Figure 5-12: An Inheritance Parse of "Mary loves several cookies."

coverage of HPSG. All of the words used in the sentence are represented in the lexicon. The chart parser is clearly more efficient, but some of the extra computation involved in the inheritance reasoner follows from the slightly different representation used for signs, and the

corresponding modification to the unification procedure. Recall from Chapter 2 that functions can appear as the values of features of signs (particularly in the grammar principles). The chart parser implements an efficient way to evaluate these functions when constructing mother signs from constituents. However, the inheritance reasoner leaves the functions in place as values. As a consequence, *order* must be evaluated each time a phrasal sign is unified with another sign. Thus, not only does the inheritance parser have more unifications to consider because it lacks a "meets" condition and attempts Rule Invocation more than it needs to, its unification algorithm is less efficient as well. A better representation of functions within signs could greatly improve the efficiency of the inheritance reasoner.

```
yes
| ?- parse.
|: mary loves several cookies.
initializing:  place inactive edge 1 ([mary]),
               built from entry mary, in stack.
initializing:  place inactive edge 2 ([loves]),
               built from entry loves, in stack.
initializing:  place inactive edge 3 ([several]),
               built from entry several, in stack.
initializing:  place inactive edge 4 ([cookies]),
               built from entry cookies, in stack.
predictor:  place inactive edge 5 ([cookies]),
            built from edge(4) and rule(2), in stack.
predictor:  place active edge 6 ([cookies]),
            built from edge(5) and rule(1), in stack.
completer:  place inactive edge 7 ([several,cookies]),
            built from edge(3) and edge(6), in stack.
predictor:  place active edge 8 ([loves]),
            built from edge(2) and rule(2), in stack.
completer:  place inactive edge 9
            ([loves,several,cookies]),
            built from edge(8) and edge(7), in stack.
predictor:  place active edge 10
            ([loves,several,cookies]),
            built from edge(9) and rule(1), in stack.
completer:  place inactive edge 11
            ([mary,loves,several,cookies]),
            built from edge(1) and edge(10), in stack.

Parse Found at 0.967 secs.

Done at 1.034 secs.

yes
```

Figure 5-13: A Better Chart Parse of "Mary loves several cookies."

However, the modified chart parser constructed by Popowich (Popowich and Vogel, pear) exhibits a dramatic increase in efficiency over the first chart parser and at the same time offers greater coverage of HPSG. Figure 5-13 shows the performance of the modified parser on

the sentences benchmarked earlier. Some of the speedup can be accounted for the fact that the parser was compiled and run under Quintus Prolog rather than interpreted under Sicstus Prolog as was true for the tests shown in Figures 5-11 and 5-12. The modified chart parser also uses a different specification of the lexical hierarchy. But, the impact of these differences is not significant. The increase in efficiency is achieved by representing the constituent structure of phrasal signs implicitly in the edges of the chart. It remains to be seen if an analogous representation can be devised for the inheritance reasoner.

In short, the chart parsing approach provides finer distinctions in the classification of objects being processed (e.g., edges in the chart vs. edges waiting to be processed). These distinctions allow the chart parser to operate more efficiently than the inheritance reasoner, and it is not clear how to incorporate such distinctions into the reasoner. On the other hand, the inheritance reasoner is conceptually simpler than the chart parser. For that reason it is useful as a pedagogical tool to explain the working of the formalism. Popowich (1990) uses an approach related to the inheritance described herein to explain the formal properties of Tree Unification Grammar. Additionally, the inheritance reasoning approach provides a principled way to cope with some types of ill-formed input.

5.3. Robust Parsing

A long standing problem for natural language processors is handling ill-formed input. One aspect of this problem is handling errors that are caused by the user's knowing and using something which is unknown to a parsing system. For instance, the use of a sentence which includes words unknown to the system lexicon (even through morphological analysis) presents a difficult problem to sentence recognition. However, the vocabulary of inheritance offers an elegant way to state a partial solution: information about unknown words may be inferred through inheritance. This section describes how this is accomplished in theory and then details how these extensions are incorporated into the implementation. Though we have seen that the implementation as it stands is not very efficient, the extension for the new link does not cause a significant further decrease in efficiency.

5.3.1. Extending the Operative Hierarchy

First, it is essential to extend the specifications of the operative hierarchy to include a statement about the relationship between the lexicon and the node at the bottom of the operative hierarchy that was shown in Figure 4-4. As with the unmodified operative hierarchy, we can simplify this network to take advantage of its having a vacuous node. The revised operative hierarchy is given in Figure 5-14. In this network the node which formerly was at the bottom and

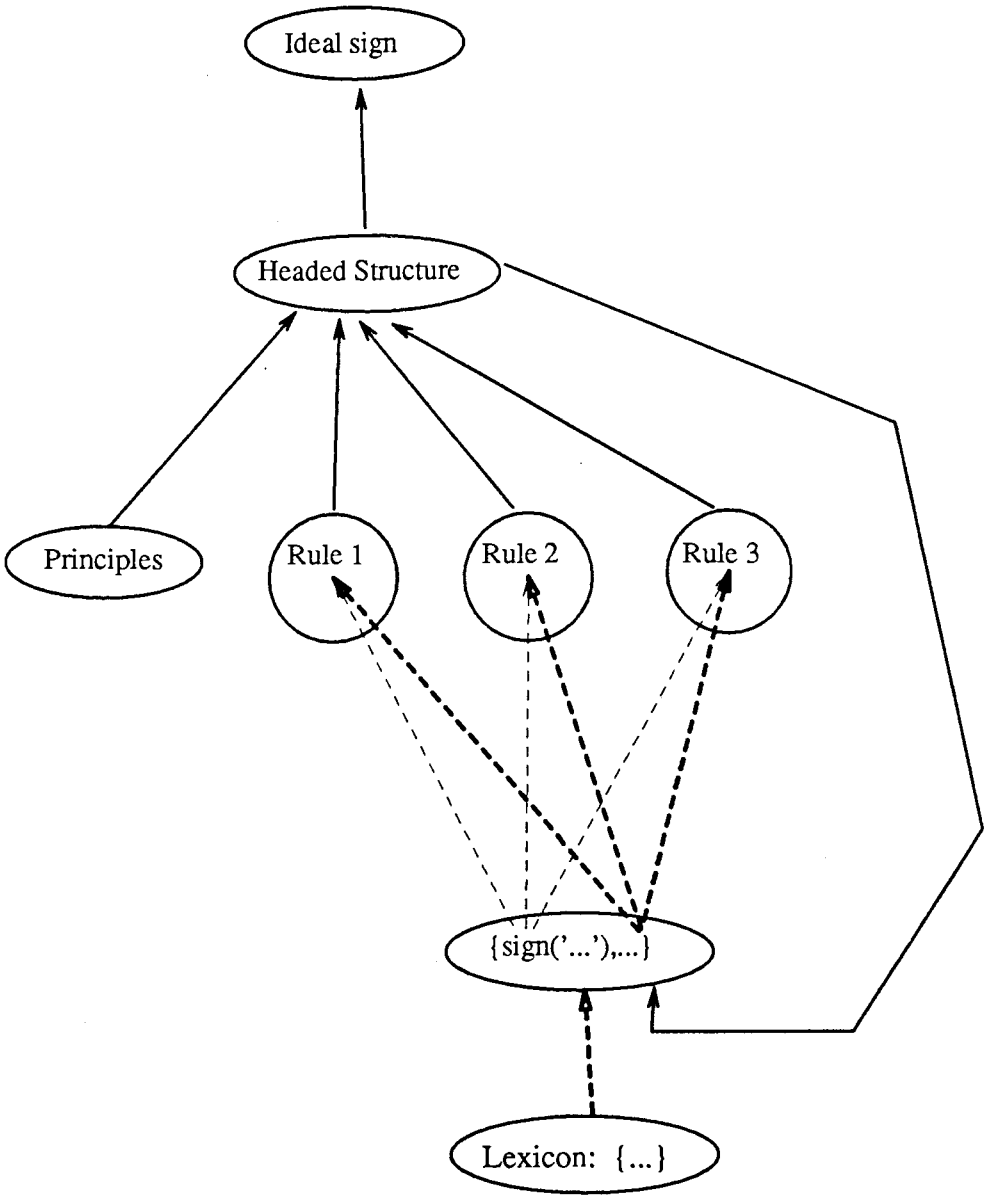


Figure 5-14: Revised HPSG Inheritance Hierarchy.

represented the partially specified list of analyses corresponding to the lexical entries for each of the words used in the sentence under consideration, is now slightly different, initially specified with a simpler set of entries. The only information known about these entries is that their PHON features are specified with strings corresponding to words in the sentence. Other information (major category, etc.) will be inherited from the lexicon. The entire lexicon is represented in the node at the bottom of the revised hierarchy.

The problem presented by a word which is unknown to the lexicon is that there is no origin from which to inherit the information required for parsing. However, recall that the lexicon is founded upon what we called the primitive hierarchy of types. Essentially, this hierarchy will be specified as the origin. A particular word, "kisses", for instance, is defined in terms of inheritance of information from lexical classes like *word*, *verbt* (for transitive verb), etc. So, we assume that a word used in a sentence but not present in the lexicon must really be a word. A fundamental assumption made in using HPSG is that the lexical structure of any word can be represented. Hence, an unknown word can be classified in terms of some general level in the lexical hierarchy. Individual words are just very specific entries in the hierarchy. So, we can generalize the notion of a lexicon from including simply words to also including entries for more general elements like *verbt*. The only difference between the entry for a specific word, "kisses", (in the absence of semantic information) and a general entry for *verbt* is that the latter will lack a specification of the value of PHON.

The problem of recognizing an unknown word is then reduced to finding the most appropriate general entry to use as its classification. This, too, is satisfied through inheritance. For instance, given a sentence like, "Olga walks," in which "Olga" is unknown to the lexicon, from the fact that "walks" subcategorizes for a noun phrase and the assumption that "Olga" is a word which is used correctly, we can determine that "Olga" is a noun phrase. Clearly, when semantic information is present in the known words, this can also be inherited to the unknown component. Note that this is not a foolproof strategy, since if "hates" is a verb that is unknown to the lexicon, the same strategy will determine that "hates" is a noun in the assumed grammatical sentence, "Hates walks." This inheritance is facilitated by the link from the lexicon to the underspecified list of partial analyses: in the usual case an entry in the list marked with the phonology of a particular word inherits the rest of the syntactic and semantic information known about the word in the lexicon, and, in the exceptional case of an unknown word the entry inherits

a hypothetical classification of the word. Given the way we have defined inheritance over the network as the construction of the shortest path to the node for the ideal (now, from the lexicon, instead of from the underspecified list of partial analyses), the only classifications of the word which will contribute to the shortest path are classifications which are consistent with respect to the other nodes in the hierarchy. That is, under the assumption of input correctness, the only consistent classification of "Olga" and "hates" is that they are noun phrases.

5.3.2. Implementing the New Link

The implementation of this extension is straightforward. It involves adding a link to the network corresponding to the link introduced in Figure 5-14. This encoding of the new link is given in Figure 5-15. A parse is denoted in the revised system by a successful instantiation of the

```
link(lexicon(Wordlist),analysis(Analysis)) :-
      link*(lexicon(Wordlist),analysis(Analysis)),
      asserta(analysis(Analysis)).

link*(lexicon([]),analysis([])).
link*(
lexicon([Word|Sentence]),
analysis([[_dtrs,[phon, [Word|E]\E]|Sign]|Others])) :-
      entry([_dtrs,[phon, [Word]]|Sign]),
      link*(lexicon(Sentence),analysis(Others)).
```

Figure 5-15: The Additional Link, in Prolog.

Prolog query:

```
inherit(lexicon(Wordlist),Through,ideal(Ideal)).
```

Finally, we allow a more general lexicon which allows as entries a larger portion of the lexical hierarchy. Within the revised lexicon, we allow as lexical entries such items as *verbt*, *verbi*, *np*, and even *word*, which lack the specification of phonological or semantic information and, in the case of *word*, even lacks a specification of major category. Recall that in theory, the lexicon is the lexical hierarchy, but for pragmatic reasons it becomes useful to draw a cut through the hierarchy and label all things below the cut as indexable by the system. Thus, the old lexicon just allowed terminal symbols, and our new lexicon includes nodes at a more general level in the hierarchy.

It is most striking that in practice this system can be used on a sentence like, "Mary kisses John," for which the lexicon lacks an entry for the word "kisses", and yet, the system is still able to determine the complex structure required of a lexical entry for the word, simply based on the information known through "Mary" and "John". Our system can recognize the sentences, "Mary

smiles," and "Mary kisses John," even though "smiles" and "kisses" are unknown, because the system assumes that the sentences are grammatical, that the words are used correctly, and because our lexicon knows about such classifications as *verbt* and *verbi* for transitive and intransitive verbs, respectively. Figure 5-16 displays the TreeTool output of the phrase structure tree derived from the HPSG sign for the sentence, "Mary ingests several cookies." The depicted HPSG analysis was constructed using the shortest path reasoner over the network that we have given and implemented in Prolog. The word "ingests" was unknown to the system lexicon, yet through inheritance the system is still able to determine that it is a transitive verb. Note that the node with the phonology, [ingests], has two elements on its SUBCAT list. Some of the information did not fit on the screen, but this omission is insignificant since this node obtained its information from other nodes in the phrase structure tree. All of the other terminal nodes in the tree represent lexical entries. But the node for "ingests" obtained only the major category classification from the lexicon, and it obtained that information through the confirmation of the hypothesis that if "ingests" is a verb, then a valid path can be constructed through the operative hierarchy. Other specific information, like the agreement features on the head of "ingests" (recorded in the second line of information on the node) and the head features of the items subcategorized for (partially depicted in the third line for that node), is inherited from the node for the universal principles (the Head Feature Principle and the Subcategorization Principle) with exact values instantiated through inheritance from the known lexical entries in the sentence. It took the inheritance reasoner 9.42 seconds to construct a parse for the sentence (c.f., Figure 5-8 shows the timing for a similar sentence in which all the words were present in the lexicon).

Our ongoing experimentation with this system involves using inheritance to determine the structure of entries that the system does not know about as lexical entries nor in the less specific classification according to major category. For instance, suppose that our lexicon does not include an entry from the lexical hierarchy for determiners. If the word "each" also lacks explicit representation in the lexicon then, of the classifications that we mentioned earlier, *word* is the only one which provides a structure which will sanction the phrase, "each cookie." Since *word* is extremely underspecified, the system should still be able to construct a complete path, and moreover, inheritance through the path should also specify that "each" is a determiner, since that is consistent with the subcategorization of "cookie". However, this is not handled by the present system because of remaining inefficiencies in its memory management.

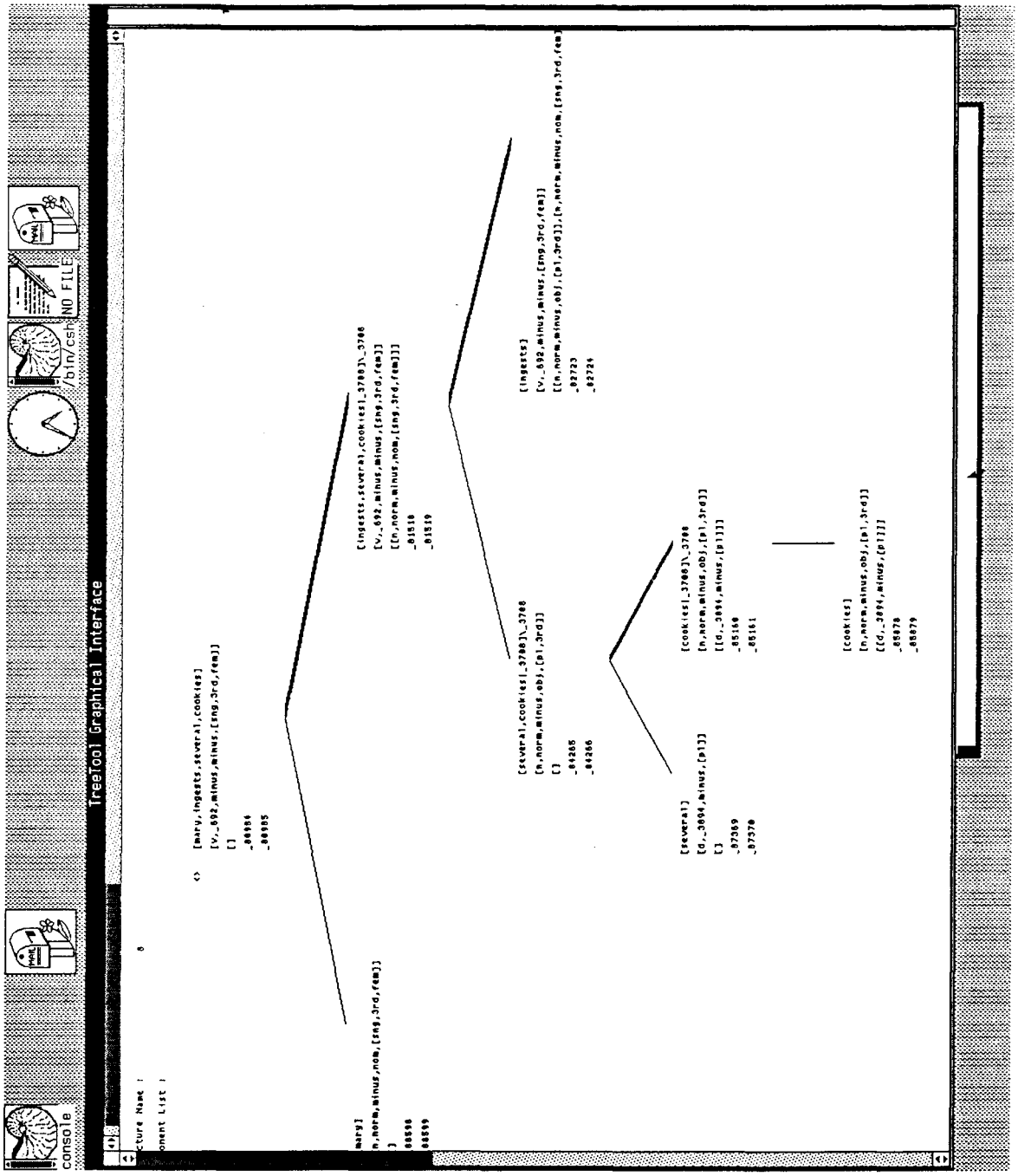


Figure 5-16: "Ingests" Was Not in the Lexicon.

Neither can we argue that the system is actually efficient at recognizing the unknown words that it does recognize. The code for the reasoner given in Figure 5-5 is stated so simply because it relies on the control structure of Prolog to implement its search. This means that it

does a lot of backtracking, particularly in the case of unknown words. Note, however, that prior to including the general classification *word* as a lexical entry, the system would discover in a finite amount of time and space (i.e., without crashing by exhausting the resources) whether a consistent set of classifications is possible. The system will backtrack and move forward until it can find a way to categorize an unknown word in a way that is consistent with respect to the rest of the sentence. While this is not particularly efficient, it does have the utility of suggesting a better solution, since it casts the problem into the more refined vocabulary of constraint satisfaction. In constraint satisfaction problems we try to find consistent labelings of words using the notation of the lexical hierarchy where consistency is defined relative to the operative hierarchy as outlined in this chapter. Our future work in this area includes a restatement of the above system using the tools provided by constraint logic programming, as codified in the language Echidna (Havens, et al., 1990).

Chapter 6
Discussion

We have focussed upon the formal definition and application of inheritance reasoning. We provided a clarification of a system for path based inheritance proposed by Horty, et al. (1990). Our clarification included a reimplementation of their system in simple Prolog relations. The accompanying analysis identified in their system an inconsistent treatment of certain topologically redundant links. We also outlined two other inheritance reasoning systems, those of Geffner and Verma (1989) and Ballim, et al. (1989), and indicated the treatment of topologically redundant links within those systems as well. Traditional inheritance reasoning systems discount the complex structures represented by nodes in an inheritance network as well as the fact that different links emanating from a single node can represent different information about the node. However, reasoning that accommodates this fact is more complex than path based inheritance which uses topology alone. The link arithmetic approach is promising because it provides a framework for inheritance reasoning based on something more than topology.

To the researcher in inheritance reasoning, this thesis provides another dimension to the space of possible path based inheritance systems: the seventy-third possibility will be a reasoner that does not discount topologically redundant links as semantically redundant (cf. Chapter One, p. 5). Further proof theoretic research is needed in path based inheritance reasoning to determine the implications entailed by the assumption that all links in a network convey new information. It would also be interesting to study the impact of the assumption on non-topological approaches to inheritance. The link arithmetic approach to inheritance provides a framework that is amenable to the incorporation of non-topological information into the reasoning process. In such a system, the leaning associated with a path would carry semantic information like the statistic which represents the frequency at which a statement such as "Elephants are gray," is true in some population. A new statistical semantics may be unecessary since such approaches already exist (Zadeh, 1987, Bacchus, 1989), but study is needed to determine exactly how the information

contained in a node and classified by a set of links can be represented using statistical information.

Thomason (1989) has indicated that research in inheritance reasoning has reached a stage in which feedback is required from research in applying these formal systems to the representation of knowledge. This thesis provides one such application by representing the fundamental concepts of HPSG in an inheritance network. The system that we present embodies some of the ideas discussed in Chapter Two, where we argued that the information contained in a node and inherited through a link should not be discounted in favor of purely topological processing. Individual links in network for HPSG do not exhaustively classify the nodes that they connect; multiple links emanating from a single node represent different information. However, in this system the computation of inheritance across a link is complex. The computation of this inheritance requires the comparison of the information contained at each end using unification. While this test on the traversal of a link also makes the system analogous to ATNs (Woods, 1970), the system is different from ATNs in that the test involves the nodes at both ends of the link. Nonetheless, reasoning over this network provides useful inferences in the form of signs that correspond to HPSG analyses of individual sentences; reasoning over the network thus implements a parser. Popowich (1990) is another example of another recent attempt to use inheritance reasoning as a parsing mechanism.

This application does not give rise to the testing of competing inheritance reasoners because it does not require the resolution of conflicting paths, though this could be an issue for a working natural language understanding system that includes a larger subset of HPSG (e.g. adding lexical rules), nor do redundant links present a problem. However, because of these features that simplify the problem, shortest path reasoning is applicable. Nonetheless, in light of the complexity of computing inheritance in the manner that we have described, the feedback that we provide to researchers in inheritance reasoning is a reiteration of the fact that further research would be useful into the strictly topological processing of networks whose links all contain unique information.

The application of inheritance reasoning to parsing does prove fruitful for the researcher in natural language understanding, because it suggests a principled treatment of a form of ill-formed input to a natural language processor. Information about words contained in an input

sentence that are unknown to the system lexicon can be inherited from the lexical hierarchy and from other words used in the sentence. It would be extremely useful to conduct further research in this area. For instance, it would be interesting to develop an inheritance based system which can discover the grammatical category of an unknown word solely from the information contained in signs for other words even if the grammatical category of the unknown word is itself omitted from the lexical hierarchy. The relationship between this treatment of unknown words and the semantic treatment provided by other approaches should also be examined (DeJong and Waltz, 1983, McFetridge and Groeneboer, 1989).

In Chapter Five we pointed out that work is required to make the implementations of the inheritance network and reasoner elegant and efficient. Presently, the implementations are neither elegant nor efficient. The research presented in Popowich and Vogel (to appear) based on the chart parsing methodology suggests some directions to follow in improving the representation of the network. A better representation of signs could eliminate the need for anything more than the built-in Prolog unification algorithm. The distinctions between data structures used by chart parsers to limit the number of times a particular sign needs to be processed may also be useful to incorporate into the network. Further investigation is required to determine whether these ideas can be incorporated into the inheritance reasoner.

References

Aristotle. (1952). *Metaphysics: Book Delta.* Ann Arbor: University of Michigan Press. Richard Hope, trans. Reprinted, 1987.

Bacchus, Fahiem. (1989). A Modest, but Semantically Well Founded, Inheritance Reasoner. *Proceedings of the 11th International Joint Conference on Artificial Intelligence.* Detroit, Michigan.

Baker, Sue, Rob Hamm, and Fred Popowich. (1990). The TreeTool User's Manual. *CMPT TR 90-09.* Simon Fraser University, Burnaby, BC.

Ballim, Afzal, Sylvia Candelaria de Ram, Dan Fass. (1989). Reasoning Using Inheritance from a Mixture of Knowledge and Beliefs. *Knowledge Based Computer Systems, Proceedings of KBCS89 (Bombay, India).* New Delhi, Narosa Publishing House.

Bobrow, R. J. (1979). The RUS Natural Language Parsing Framework. *Research in Natural Language Understanding, Annual Report (Report Number 4274).* Cambridge, MA: Bolt, Beranek and Newman.

Bouma, Gosse. (1990). Non-Monotonic Inheritance and Unification. In W. Daelemans and G. Gazdar (Eds.), *Inheritance in Natural Language Processing Workshop Proceedings.* Institute for Language Technology and Artificial Intelligence, Tilburg University, Holland.

Boutilier, Craig. (1989). A Semantical Approach to Stable Inheritance Reasoning. *Proceedings of the 11th International Joint Conference on Artificial Intelligence.* Detroit, Michigan.

Brachman, Ronald J. (1983). What ISA Is and Isn't: An Analysis of Taxonomic Links in Semantic Networks. *IEEE Computer, 16*(10), 30-6.

Brachman, Ronald J. (1985). I Lied about the Trees, or Defaults and Definitions in Knowledge Representation. *The AI Magazine, 6,* 80-93.

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

Calder, Jo, Ewan Klein, and Henk Zeevat. (1988). Unification Categorial Grammar: A Concise, Extendable Grammar for Natural Language Processing. *12th International Conference on Computational Linguistics.* Budapest, Hungary.

Carpenter, Bob. (1990). Typed Feature Structures: Inheritance, (In)equality and Extensionality. In W. Daelemans and G. Gazdar (Eds.), *Inheritance in Natural Language Processing Workshop Proceedings.* Institute for Language Technology and Artificial Intelligence, Tilburg University, Holland.

Curry, Haskall B. (1963). *The Foundations of Mathematical Logic.* New York: McGraw Hill.

DeJong, Gerald F., and David L. Waltz. (1983). Understanding Novel Language. *Computers and Mathematics with Applications, 9*(1), 131-47.

Delgrande, Jim. (1990). A Semantics for a Class of Inheritance Networks. *Proceedings of the*

8th Biennial Conference of the Canadian Society for Computational Studies of Intelligence. Ottawa, Ontario.

Dorosh, Jennie and Ronald P. Loui, eds. (1989). *Edited Transcription of The Workshop on Defeasible Reasoning with Specificity and Multiple Inheritance, St. Louis, April 1989.* Washington University, St. Louis, MO.

Evans, Roger and Gerald Gazdar. (1989). Inference in DATR. *Proceedings of the 4th Conference of the European Chapter of the Association for Computational Linguistics.* Manchester, England.

Fahlman, S. E. (1979). *NETL: A System for Representing and Using Real-World Knowlege.* Cambridge, MA: The MIT Press.

Fass, Dan and Gary Hall. (1990). A Belief-Based View of Ill-Formed Input. *Computational Intelligence '90.* Milan, Italy.

Fass, Dan, Nick Cercone, Gary Hall, Chris Groenboer, Paul McFetridge, and Fred Popowich. (1990). A Classification of User-System Interactions in Natural Language, with Special Reference to 'Ill-Formed Input'. *Proceedings of the 5th Rocky Mountain Conference on Artificial Intelligence.* RMCAI-90, Las Cruces, NM.

Flickinger, Dan. (1987). *Lexical Rules in the Hierarchical Lexicon.* Doctoral dissertation, Stanford University, CA.

Flickinger, Dan, Carl Pollard and Tom Wasow. (1985). Structure-Sharing in Lexical Representation. *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics.* University of Chicago, Chicago, IL.

Franz, A. (forthcoming). A Parser for HPSG. Carnegie Mellon University Laboratory for Computational Linguistics Technical Report.

Fraser, Norman and Richard Hudson. (1990). Word Grammar: an Inheritance-Based Theory of Language. In W. Daelemans and G. Gazdar (Eds.), *Inheritance in Natural Language Processing Workshop Proceedings.* Institute for Language Technology and Artificial Intelligence, Tilburg University, Holland.

Frisch, Alan M. (1989). A General Framework for Sorted Deduction: Fundamental Results on Hybrid Reasoning. *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning.* Toronto, Canada.

Gawron, J. M., J. King, J. Lamping, E. Loebner, E. A. Paulson, G. K. Pullum, I. A. Sag, and T. Wasow. (1982). Processing English with a Generalized Phrase Structure Grammar. *Proceedings of the 20th Annual Meeting of the Association for Computational Linguistics.* Toronto, Ontario.

Gazdar, Gerald, Ewan Klein, Geoffrey Pullum, and Ivan Sag. (1985). *Generalized Phrase Structure Grammar.* Basil Blackwell, London, England.

Geffner, Hector and Tom Verma. (1989). Inheritance = Chaining + Defeat. *CSD-890039.* University of California, Los Angeles.

Grice, H. Paul. (1975). Logic and Conversation. In Peter Cole and Jerry L. Morgan (Eds.), *Syntax and Semantics, Volume 3.* New York: Academic Press.

Grishman, Ralph. (1986). *Computational Linguistics, An Introduction.* Cambridge University Press, Cambridge, England.

Havens, William S., Susan Sidebottom, Miron Cuperman, Rod Davison, Severin Gaudet, and Greg Sidebottom. (1990). Echidna Constraint Reasoning System: Programming Langage

Manual Version 0. *CSS-IS TR 90-07*. Expert System Laboratory, Centre for Systems Science, Simon Fraser University, Burnaby, BC.

Horty, John. (1989). Discussion Session: Research Strategies: Individual Perspectives. In Dorosh, Jennie and Ronald P. Loui, eds. (Eds.), *Edited Transcription of The Workshop on Defeasible Reasoning with Specificity and Multiple Inheritance, St. Louis, April 1989*. Washington University, St. Louis, MO. Panel Members: David Poole, Fahiem Bacchus, James Delgrande, John Horty; Ben Grosof, Moderator.

Horty, John, Richmond Thomason, and David Touretzky. (1990). A Skeptical Theory of Inheritance in Nonmonotonic Semantic Networks. *Artificial Intelligence, 42*(2-3), 311-48.

Israel, David J. (1983). Interpreting Network Formalisms. *Computers and Mathematics with Applications, 9*(1), 1-13.

König, Ester. (1989). Parsing as Natural Deduction. *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*. Vancouver, BC.

Kasper, Robert, and William Rounds. (1986). A Logical Semantics for Feature Structures. *Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics*. Columbia University, New York, NY.

Konolige, Kurt. (1987). On the Relation Between Default Theories and Autoepistemic Logic. *Proceedings of the 10th International Joint Conference on Artificial Intelligence*. Milan, Italy.

McDermott, Drew and Jon Doyle. (1980). Non-Monotonic Logic I. *Artificial Intelligence, 13*(1-2), 41-72.

McFetridge, Paul, and Nick Cercone. (1990). The Evolution of a Natural Language Interface: Replacing a Parser. *Proceedings of Computational Intelligence 90*. Università di Milano, Milan, Italy.

McFetridge, Paul and Chris Groeneboer. (1989). Novel Terms and Cooperation in a Natural Language Interface. *Knowledge Based Computer Systems, Proceedings of KBCS89 (Bombay, India)*. New Delhi, Narosa Publishing House.

Menzel, Wolfgang. (1987). Automated Reasoning about Natural Language Correctness. *Proceedings of the 3rd Conference of the European Chapter of the Association for Computational Linguistics*. Copenhagen, Denmark.

Moore, Robert C. (1985). Semantical Considerations on Nonmonotonic Logic. *Artificial Intelligence*, Vol. *25*(1).

Morgan, Charles G. (1976). Methods for Automated Theorem Proving in Nonclassical Logics. *IEEE Transactions on Computers*, Vol. *C25*(8).

Oehrle, Richard, Emmon Bach, and Deirdre Wheeler. (1988). *Categorial Grammars and Natural Language Structures*. D. Reidel, Dordrecht, Holland.

Pollard, Carl, and Ivan Sag. (1987). *Information-Based Syntax and Semantics, Volume 1: Fundamentals*. Centre for the Study of Language and Information, Stanford University, CA.

Popowich, Fred. (1988). *Reflexives and Tree Unification Grammar*. Doctoral dissertation, Centre for Cognitive Science, University of Edinburgh, Edinburgh, Scotland.

Popowich, Fred. (1989). Tree Unification Grammar. *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*. Vancouver, BC.

Popowich, Fred. (1990). Tree Unification Grammar, Parsing and Inheritance Networks. *CSS/LCCR TR 90-15, CMPT TR 90-07.* Simon Fraser University, Burnaby, BC.

Popowich, Fred and Carl Vogel. (1990). Chart Parsing Head-Driven Phrase Structure Grammar. *CSS-IS TR 90-01, CMPT TR 90-01.* Simon Fraser University, Burnaby, BC.

Popowich, Fred and Carl Vogel. (to appear). A Logic-Based Implementation of Head-Driven Phrase Structure Grammar. *Proceedings of the Third International Workshop on Natural Language Understanding and Logic Programming.* NLULP3, Lidinigo, Stockholm. January 23-25, 1991, Also to appear in *Natural Language Understanding and Logic Programming III.* Charles Brown and Gregers Koch, eds. Amsterdam: North Holland.

Porter, Harry H., III. (1987). Incorporating Inheritance and Feature Structures into a Logic Grammar Formalism. *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics.* University of Chicago, Chicago, IL.

Reape, Mike. (August 1990). (Personal communication). Centre for Cognitive Science, University of Edinburgh, 2 Buccleuch Place, Edinburgh EH8 9LW.

Reiter, Raymond and Giovanni Criscuolo. (1983). Some Representational Issues in Default Reasoning. *Computers and Mathematics with Applications, 9*(1), 15-27.

Russell, Graham, John Carroll and Susan Warwick. (1990). Multiple Inheritance in a Unification-Based Lexicon. In W. Daelemans and G. Gazdar (Eds.), *Inheritance in Natural Language Processing Workshop Proceedings.* Institute for Language Technology and Artificial Intelligence, Tilburg University, Holland.

Sandewall, E. (1986). Nonmonotonic Inference Rules for Multiple Inheritance with Exceptions. *Proceedings of IEEE, 74*(10), 1345-53.

Schubert, Len. (1975). Extending the Expressive Power of Semantic Networks. *Proceedings of the 4th International Joint Conference on Artificial Intelligence.* Tbilisi, USSR.

Schubert, Len, Randy Goebel, and Nick Cercone. (1979). The Structure and Organization of a Semantic Net for Comprehension and Inference. In N. V. Findler (Eds.), *Associative Networks: Representation and Use of Knowledge by Computers.* New York: Academic Press.

Selman, Bart, and Hector Levesque. (1989). The Tractability of Path-Based Inheritance. *Proceedings of the 11th International Joint Conference on Artificial Intelligence.* Detroit, Michigan.

Shieber, Stuart. (1986). *An Introduction to Unification-Based Approaches to Grammar.* The University of Chicago Press, Chicago, IL.

Steels, L. and Koenraad De Smedt. (1983). Some Examples of Frame-Based Syntactic Processing. In Fr. Daems and L. Goossens (Eds.), *Een Spyeghel voor G. Jo Steenbergen.* Leuven, Amersfoort: Acco.

Thomason, Richmond. (1989). Discussion Session: Questions of Substance or Mere Clashes of Intuition? In Dorosh, Jennie and Ronald P. Loui, eds. (Eds.), *Edited Transcription of The Workshop on Defeasible Reasoning with Specificity and Multiple Inheritance, St. Louis, April 1989.* Washington University, St. Louis, MO. Panel Members: Jon Doyle, David Israel, Kurt Konolige, Richmond Thomason; David Etherington, Moderator.

Touretzky, David. (1986). *The Mathematics of Inheritance Systems.* Los Altos, CA: Morgan Kaufman.

Touretzky, David, John Horty, Richard Thomason. (1987). A Clash of Intuitions: The Current State of Nonmonotonic Inheritance Systems. *Proceedings of the 10th International Joint Conference on Artificial Intelligence.* Milan, Italy.

van der Linden, Erik-Jan. (1989). Lambek Theorem Proving and Feature Unification. *Proceedings of the 4th Conference of the European Chapter of the Association for Computational Linguistics*. Manchester, England.

Vogel, Carl and Fred Popowich. (1990). Head-Driven Phrase Structure Grammar as an Inheritance Hierarchy. In W. Daelemans and G. Gazdar (Eds.), *Inheritance in Natural Language Processing Workshop Proceedings*. Institute for Language Technology and Artificial Intelligence, Tilburg University, Holland. Also appeared as CSS/LCCR TR 90-03, Centre for Systems Science, Simon Fraser University, Burnaby, B.C.

Woods, William A. (1970). Transition Network Grammars for Natural Language Analysis. *Communications of the ACM, 13*, 591-606.

Zadeh, Lofti A. (1987). Commonsense and Fuzzy Logic. In N. Cercone and G. McCalla (Eds.), *The Knowledge Frontier*. New York: Springer Verlag.