Parsing with Restricted Quantification

Alan M.Frisch

February 1985

Cognitive Science Research Paper

Serial No. CSRP 46

The University of Sussex,
Cognitive Studies Programme
School of Social Sciences
Brighton BN1 9QN
England

Parsing with Restricted **Quantification**

Alan M. Frisch

February 1985

Abstract

The primary goal of this paper is to illustrate how smaller deductive search spaces can be obtained by extending a logical language with restricted quantification and tailoring an inference system to this extension. The illustration examines the search spaces for a bottom-up parse of a sentence with a series of four strongly-equivalent grammars. The grammars are stated in logical languages of increasing expressiveness, each restatement resulting in a more concise grammar and a smaller search space.

A secondary goal is to point out an area where further research could yield results useful to the design of efficient parsers, particularly for grammatical formalisms that rely heavily on feature systems.

This paper appears in Proceedijigs of AIj>B 85, Meeting of the Society for the Study of Artificial Intelligence and Simultion of Behaviour, University of Warwick, April, J985. It is also to appear in a book edited by A.G. Cohn and R. Thomas.

Alan M. Frisch

Cognitive Studies Programme
University of Sussex
Brighton, BN1 9QN

ABSTRACT

The primary goal of this paper is to illustrate how smaller deductive search spaces can be obtained by extending a logical language with restricted quantification and tailoring an inference system to this extension. The illustration examines the search spaces for a bottom-up parse of a sentence with a series of four strongly-equivalent grammars. The grammars are stated in logical languages of increasing expressiveness, each restatement resulting in a more concise grammar and a smaller search space.

A secondary goal is to point out an area where further research could yield results useful to the design of efficient parsers, particularly for grammatical formalisms that rely heavily on feature systems.

1. INTRODUCTION

I am beginning to undertake a study of inference techniques for logica languages with restricted quantification. Though extending a first-order languag with restricted quantification in no way increases what it can express, smalle deductive search spaces can sometimes be obtained by tailoring an inference mechan ism to exploit the extension. A thorough analysis is needed of when, how much, an why such inference techniques pay off. This paper, however, has the much less ambi tious goal of illustrating and briefly examining the advantages of the approach.

The advantages are illustrated by examining the search space for a bottom-u parse of "The horse raced past the barn fell" with each of a series of fou strongly-equivalent grammars.[1] An identification is made between a parsing proble and a deduction problem by expressing a grammar in a logical language. The fou grammars are stated in logical formalisms of increasing expressiveness, the last tw of which incorporate restricted quantification. As the expressiveness of the logi cal formalism increases, the rules of the grammar are collapsed and the deductiv machinery is enhanced in order to obtain smaller search spaces for parsing.

There is a growing body of literature attempting to tie logic and gramma together in order to achieve a unification of inference methods and parsing methods The results achieved with this approach can be classified broadly into tw categories. On one hand, viewing a grammatical formalism as a specialized logica formalism may suggest ways of generalizing the grammatical formalism and the parsir techniques associated with it. As an example, Pereira and Warren (1983) have gen eralized Early's context-free-grammar parsing algorithm to deal with definite claus grammars. On the other hand, advances in logical representation and inference tech niques can lead to advances in grammatical representation and parsing. The idea presented in this paper exemplify this.

2. GRAMMAR WITHOUT QUANTIFICATION

This section considers the problem of parsing a sentence with a standard CF (context-free grammar), Grammar 1. In this, and all succeeding grammars, nor terminals appear in upper case and terminals in lower case. Furthermore, the gram mars use two common notational conventions not found in strict CFG notation. First a non-terminal written in parentheses is optional and hence the rule containing i is merely a shorthand for two rules--one with the non-terminal occurring and or

[1]My use of the notorious sentence, "The horse raced past the barn fell," i perhaps unfortunate as it may suggest erroneously that this paper is concerned wit parsing garden-path sentences. It is, in fact, a meta-garden-path senterce, sinc initially it may lead one to think that the paper is about garden-path sentence when it is not.

```
R1 a)  S              -->  NP VP[fin,intr]
   b)  S              -->  NP VP[fin,tr]
R2 a)  VP[fin,tr]     -->  V[fin,tr] NP (PP)
   b)  VP[pp,tr]      -->  V[pp,tr]  NP (PP)
   c)  VP[pas,tr]     -->  V[pas,tr] (PP)
   d)  VP[fin,intr]   -->  V[fin,intr] (PP)
   e)  VP[pp,intr]    -->  V[pp,intr] (PP)
R3  )  NP             -->  DET N (VP[pas,tr])
R4  )  PP             -->  P NP
R5  )  DET            -->  the
R6  )  N              -->  horse
R7  )  N              -->  barn
R8  )  P              -->  past
R9  )  V[fin,intr]    -->  fell
R10a)  V[fin,intr]    -->  raced
   b)  V[pp,intr]     -->  raced
   c)  V[fin,tr]      -->  raced
   d)  V[pp,tr]       -->  raced
   e)  V[pas,tr]      -->  raced
```

Grammar 1

---

without. Second, a non-terminal may be followed by a list of feature values written in lower case and enclosed in square brackets. To see that this use of features is within the realm of a context-free grammar (i.e., generates only context-free languages) simply consider an entire construction, such as VP[pas,tr], to be a non-terminal. I do not encourage the adoption of such a view as it presents VP[pas,tr] and VP[fin,tr] as two unrelated non-terminals, hiding their important linguistic commonalities; I merely point this out to demonstrate that the notational variation does not increase the generative power of the grammatical formalism.

In this paper, the only non-terminals with feature values are V and VP, which have the same two features. The first feature is voice and its value is either fin (finite), pp (past participle), or pas (passive); the second feature is transitivity and its value is either tr (transitive) or intr (intransitive). I call a nonterminal "finite" to indicate that it has the fin feature value and use a similar convention for nonterminals with other feature values. Furthermore, I call à nonterminal active to indicate that it is either finite or past-participial. A verb is considered transitive if it normally subcategorizes for at least a direct object noun phrase and therefore passive verbs are considered transitive.

A grammar rule can be considered as a syntactic shorthand for a sentence of FOPC (first-order predicate calculus) and accordingly a grammar can be considered as a set of FOPC sentences. The FOPC sentence identified with a grammar rule captures the rewrite conditions expressed by the grammar rule. More precisely, the grammar allows a given non-terminal to be rewritten to a given string if, and only if, the logical formulation of the grammar logically implies a certain logical sentence, which is a function of the given non-terminal and string.

To construct a logical encoding of a grammar we first need a way of encoding strings. One common way to do this (Kowalski, 1979; Pereira and Warren, 1980) is to regard that string as a graph whose arcs are labelled by the words occurring in the string. For example, "The horse raced past the barn fell" is represented by the graph

```
     the      horse      raced      past       the       barn       fell
(1) -----> (2) -----> (3) -----> (4) -----> (5) -----> (6) -----> (7) -----> (8)
```

This graph, in turn, can be described by a set--hereafter called "the Input String"--which contains seven atomic logical sentences, one for each arc.

{the(1,2), horse(2,3), raced(3,4), past(4,5), the(5,6), barn(6,7), fell(7,8)}

The intended interpretation of "the(1,2)," for example, is that the word "the" labels the arc connecting node (1) to node (2). Hence, every terminal in the grammar corresponds to a predicate in the logic. Similarly, every non-terminal corresponds to a predicate; thus the intended interpretation of PP(1,3) is that the labels on the arcs connecting node(1) to node(3) form a PP.

Using these predicates, it is a straightforward matter to encode a grammar rule as a logical formula. A CFG rule can always be encoded as a Horn clause, which for current purposes is a universally-quantified implication whose antecedent is a conjunction of zero or more atomic formulas, and whose consequent is a single atomic formula. For example, rule R4 can be encoded as

```
(1)  PP(?x,?z) <- P(?x,?y) & NP(?y,?z)
```

When written in this notation, frequently employed by the logic-programming community to encode Horn clauses, a sentence appears similar to a grammar rule. The implication sign is written backwards with the antecedent on its right and the consequent on its left. The universal quantifier is eliminated and all variables--those symbols whose names begin with "?"--are taken implicitly to be universally quantified and scoped over the entire sentence. Hence, the intended interpretation of (1) is "For all x, y and z, if there is a preposition from node x to node y and there is a noun phrase from node y to node z, then there is a prepositional phrase from node x to node z."

With two trivial exceptions--optional elements and features--the remainder of the grammar rules can be encoded in FOPC in a similar manner. Since a rule with an optional element is merely a shorthand for two rules, it can be encoded simply as two logical sentences. Likewise, since non-terminals with features can be regarded as distinct featureless non-terminals, they too could be encoded in the obvious manner. However, I will <u>not</u> pursue this strategy; as mentioned before, doing so would obscure some useful generalizations. Rather, I encode each feature as an additional term in the predicate corresponding to the non-terminal. The advantage of this approach is demonstrated in the next section, which examines a grammar that quantifies over the features. That grammar explicates what I have been referring to vaguely as "useful generalizations."

Using the devices presented above, Grammar 1 is encoded as the logical sentences shown in Grammar 1'.

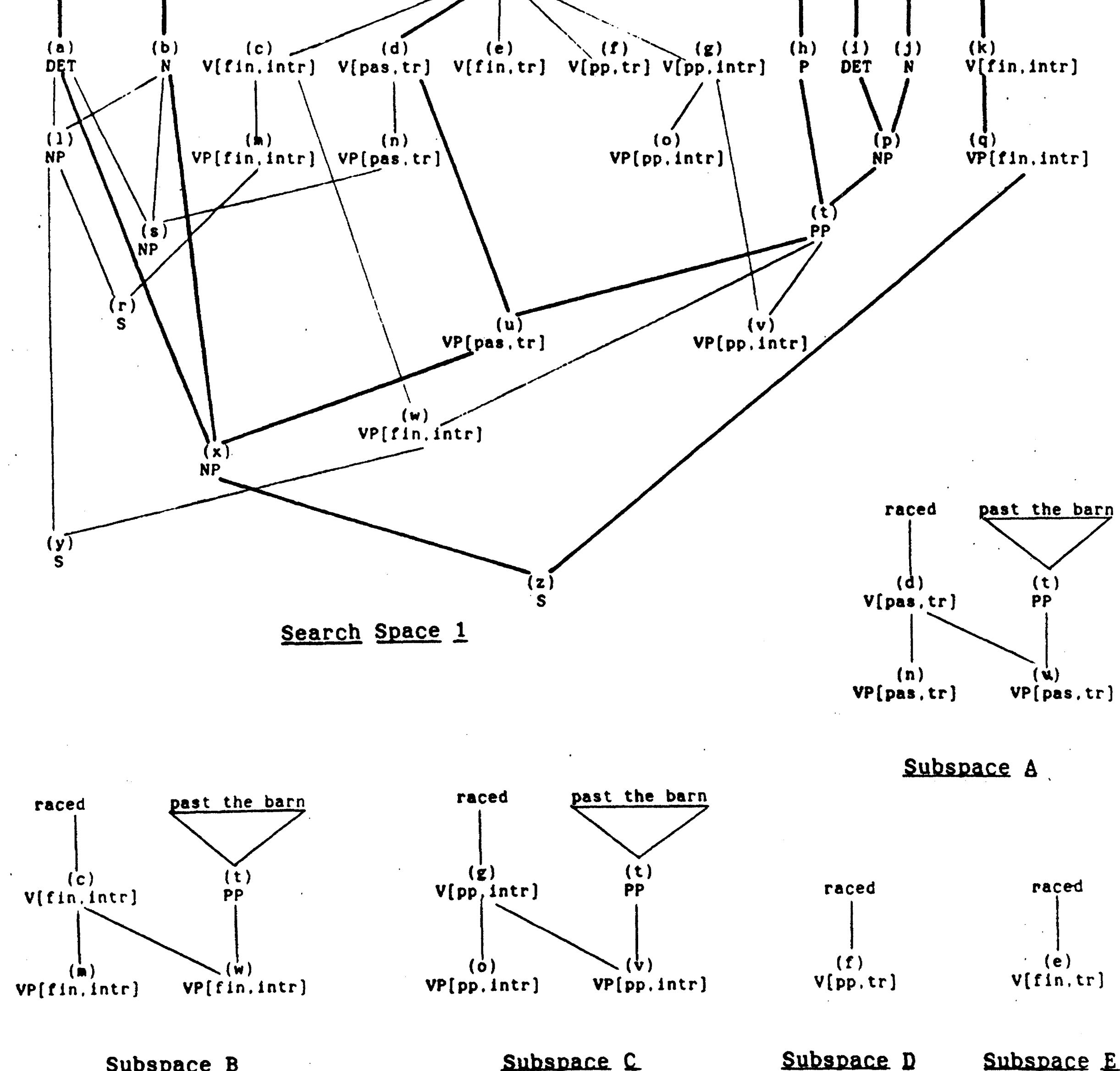
There is a simple mapping of every parsing problem to an equivalent theorem-proving problem. The example parsing problem used in this paper maps to the problem of proving that the logical sentence "S(1,8)" logically follows from the sentences of Grammar 1' and the Input String.

Having laid the foundation for connecting grammar to logic--and hence parsing to theorem proving--we now turn our attention to the search spaces confronted by a bottom-up parser using various grammatical formalisms. In Search Space 1, a search space for Grammar 1, the string to be parsed appears in the top row of nodes, below which is a labelled node for each constituent structure that can be found in the string. For instance, node (b) shows that "horse" is an N while node (l) shows that "the horse" is an NP. Each node corresponds to a completed arc that could be built

---

```
R1 a') S(?x,?z)            <-  NP(?x,?y) & VP(fin,intr,?y,?z)
   b')                     <-  NP(?x,?y) & VP(fin,tr,?y,?z)
R2 a') VP(fin,tr,?x,?z)    <-  V(fin,tr,?x,?y) & NP(?y,?z)
       VP(fin,tr,?x,?z)    <-  V(fin,tr,?x,?y) & NP(?y,?w) & PP(?w,?z)
              :                       :
              :                       :
R3'  ) NP(?x,?z)           <-  DET(?x,?y) & N(?y,?z)
       NP(?x,?z)           <-  DET(?x,?y) & N(?y,?w) & VP(pas,tr,?w,?z)
R4'  ) PP(?x,?z)           <-  P(?x,?y) &  NP(?y,?z)
R5'  ) DET(?x,?y)          <-  the(?x,?y)
R6'  ) N(?x,?y)            <-  horse(?x,?y)
R7'  ) N(?x,?y)            <-  barn(?x,?y)
R8'  ) P(?x,?y)            <-  past(?x,?y)
R9'  ) V(fin,intr,?x,?y)   <-  fell(?x,?y)
R10a') V(fin,intr,?x,?y)   <-  raced(?x,?y)
              :                       :
              :                       :
```

<u>Grammar</u> <u>1'</u>



Search Space 1

Subspace A

Subspace B

Subspace C

Subspace D

Subspace E

by a bottom-up chart parser. Not shown in this search space, or any search space appearing in this paper, are failed attempts at finding a constituent structure. These would correspond to the active arcs in a chart that are never extended into complete arcs.

We can also take a logical view of this search space. The nodes in the top row correspond to the sentences of the Input String. Each node below the top row corresponds to a sentence that can be derived with the UR-resolution inference rule (Wos, Winker, Smith, Veroff and Henschen, 1984; McCharen, Overbeek and Wos, 1976). From Horn clause A <- C1 & C2 & ...& Cn of the grammar and atomic sentences B1, B2, ..., Bn of the search space this rule derives theta(A) provided that

- no two of the sentences in {A <- C1 & ...& Cn, B1, B2, ..., Bn} contain occurrences of the same variable, and
- theta is a most-general substitution that unifies each pair of Bi and Ci.

This is a large inference rule, accomplishing in one step what would require n applications of binary resolution. Though a single application of UR-resolution may require search, it is not shown in Search Space 2, or any of the following search spaces. A chart parse records such search in its active arcs. (Notice that this is in accord with the previous claim that the search spaces do not display anything that corresponds to the active arcs.) The UR-resolution specification of parsing is highly abstract as it hides a great deal of the work involved. It can also be seen as an abstraction of Early deduction, which Pereira and Warren (1983) have developed as a generalization of both chart parsing and the Early algorithm.

How good is this search space? First of all, it contains 26 derived nodes, 13 of which are in the solution tree. So, offhand, it is not extremely bad. However, the space does contain some redundancy. Notice that it contains Subspaces A through E. In all five of them, "raced" is rewritten to a V (nodes d, c, g, e and f) each differing only in its features. In Subspaces A, B and C, the V is subsequently rewritten to a VP (nodes n, m and o) with features identical to those of the V. These same three subspaces contain parses of "raced past the barn" as a VP, each parse identically structured but differing in the features used.

In this example the redundancy does not get out of hand if a chart is used. Even though nodes u, w and v represent distinct parsings of "raced past the barn" they share the common parsing of "past the barn" represented by node t. Hence, a chart parser would only parse "past the barn" once, but a bottom-up parser that doesn't use a chart, such as BUP (Matsumoto, Tanaka and Kiyono, 1984), would contain 3 parsings of "past the barn" in its search space.

A little thought reveals why this redundancy arises in the search space. Both the grammar and the search space show that "raced" can only be rewritten to a V node. The problem is that there is a choice of five pairs of feature values for the V but no way to choose among them solely on the basis of the occurrence of "raced." In order to do the rewriting, though, there is no need to choose the feature values—other than to restrict them to being one of five possible pairs. To choose among these feature values is to make an overcommitment—one that introduces the five-fold redundancy in the search space.

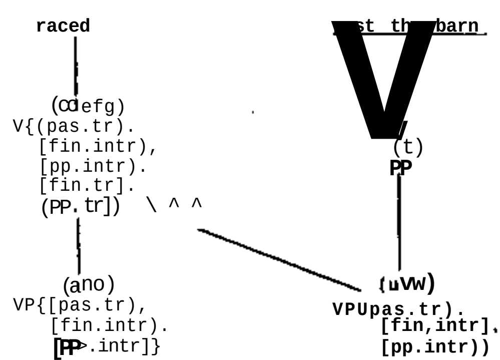
This paper advocates a minimum-commitment search strategy with which a parser could rewrite "raced" to a V coupled with the constraint that its feature values must be one of the five permissible pairs. This could be achieved by replacing grammar rules R10a through R10e with a single, generalized rule. Later sections introduce increasingly powerful formal languages for expressing such generalizations, but for present purposes the rule can be written as

(R10*) V{[pas,tr],[fin,intr],[pp,intr],[fin,tr],[pp,tr]} —> raced

With this modification there is only a single way of rewriting "raced" and it yields a node that functions in the search space as the combination of nodes c, d, e, f and g do in Search Space h This generalized node represents the minimum commitment to the feature values of the V.

When this node is subsequently rewritten to a VP node further commitment is necessary. If that VP has an immediate NP constituent (i.e., an object) then, as seen by grammar rules R2a and R2b, it must be associated with the feature values [fin.tr] or [pp.tr]. Otherwise, as seen by grammar rules R2c, R2d and R2e, the VP must be associated with the feature values [pas.tr], [fin,intr] or [pp.intr]. This is captured by rules R2', which collapses R2a and R2b, and R2", which collapses R2c, R2d and R2e.

(R2') VP{[fin,tr],[pp.tr]} —> V{[fin,tr],[pp.tr]}NP (PP)
(R2") VP{[pas,tr],[fin,intr],[pp.intr]) --> V{[pas,tr],[fin,intr],[pp,intr]}



By appropriate use of these collapsed rules, R10*, R2' and R2", Subspaces A through E collapse into Subspace F. This and all following search spaces and subspaces follow the convention of labelling a node with the concatenation of the labels on the nodes it collapses. Similarly each rule in the remainder of the paper is labelled by concatenation of the labels of the rules it collapses. Notice that Subspace F contains only the single parse for "raced past the barn" located at node uvw. Not only does this node encode what is encoded as three nodes in Search Space 1, its constituent structure encodes the constituent structures of the three corresponding nodes in Search Space 1.

We can think of a parser operating in the collapsed search space as carrying along multiple parses in parallel and dropping some only when necessary. So, five possible parses of "raced" are under consideration at node cdefg but only three are carried along to node mno. As the search progresses the feature values under consideration become progressively fewer.

The remainder of this paper is devoted to developing a logical language in which all structurally similar rules in the grammar can be expressed as a single rule and an inference method in which all structurally similar nodes occur as one. More precisely, the logical language can express any CFG in a form that does not contain two rules that are identical except for feature values. Similarly, the resulting search spaces do not contain any two nodes with constituent structures that are identical except for feature values.

The logic must overcome a difficulty with the notation used above to represent collapsed rules and search-space nodes. An example of this difficulty arises in R2". The rule states that a VP with any one of three feature-value pairs can be written to a V with any one of three feature-value pairs but does not state that the choice of feature values on the V is related to those on the VP. A proper grammar must ensure that VP[pas,tr] cannot be rewritten to V[fin,intr]. The problem is also reflected in Subspace F where there is no connection between the feature values at node cdefg and those at uvw. Roughly speaking, the logic surmounts these problems by using rule and node schemata that are powerful enough to ensure that precisely the proper rule and node instances are generated.

3. GRAMMAR WITH QUANTIFICATION

The development of a search space with collapsed nodes requires a more expressive formalism for expressing these generalized nodes and the generalized rules for rewriting such nodes. The usual way of generalizing a sentence of FOPC is to use universal quantification. So, for example, sentences R1a' and R1b' could be collapsed to the single sentence

(2) S(?x,?z) <- NP(?x,?y) & VP(fin,?t,?y,?z)

Strictly speaking (2) is not logically equivalent to the conjunction of R1a' and R1b'. In (2) the variable ?t ranges over the entire domain, not just the transitivity and intransitivity features. In this paper I ignore the difficulty and ask the reader to consider variables in an argument position normally occupied by a feature as implicitly ranging over appropriate values for that feature. ?t and ?v are used to range over the transitivity and voice feature values respectively. The next section presents a logic in which this can be expressed explicitly.

Since a grammar is viewed as a notational shorthand for certain sentences of FOPC, we extend the grammatical notation to allow variables in place of features. Hence, in grammar notation, (2) can be written as

S --> NP VP[fin,?t]

Note that this extension to the grammatical formalism is a product of identifying grammar rules with logical sentences. Also notice that association with the logical interpretation clarifies the meaning of the above rule and suggests that a parser can use unification to deal with variables in the grammar rules. Continuing along this line, Grammar 1 can be changed to Grammar 2 by changing rules R1 and R10.

```
R10ac) V[fin,?t]   --> raced
   bd) V[pp,?t]    --> raced
    e) V[pas,tr]   --> raced
```

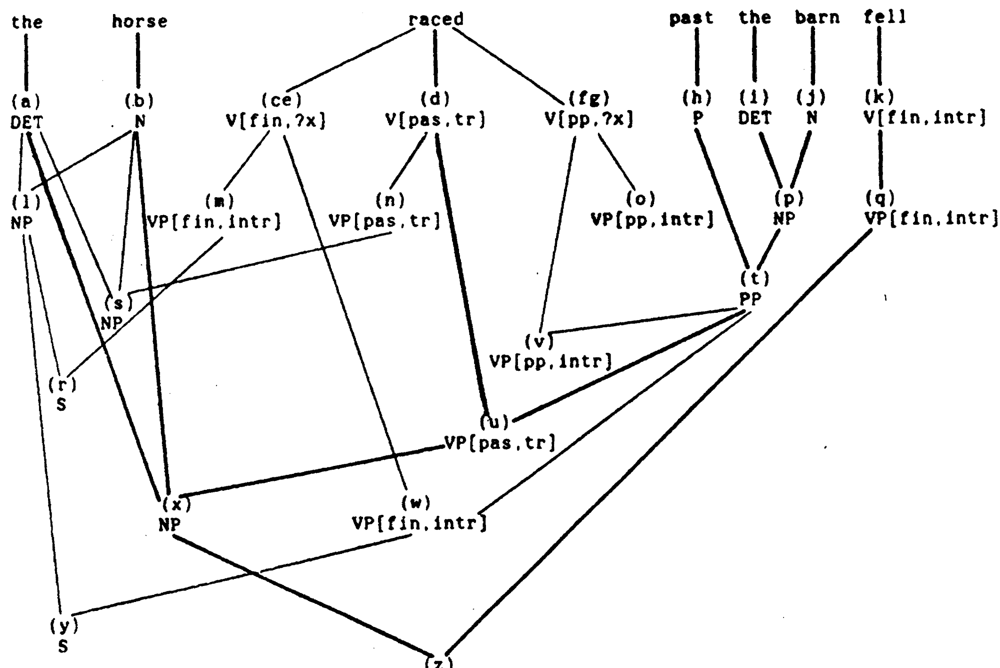
<u>Grammar 2</u>

The search space associated with this grammar, Search Space 2, differs from Search Space 1 only in that Subspaces B and D have been collapsed to Subspace G and Subspaces C and E have been collapsed to Subspace H. (Subspace A remains unchanged.) As a result Search Space 2 contains two nodes fewer than Search Space 1.

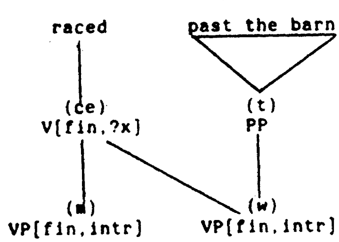
Observe that no more collapsing is possible within the present grammatical and logical formalism. Consider the five rules entered under R2. The first two deal with transitive VPs and are structurally similar. Yet these two cannot be replaced by rule (3) since it would entail rule (4), which could generate sentences not in the language.

```
(3)  VP[?v,tr]  -->  V[?v,tr] NP (PP)
(4)  VP[pas,tr] -->  V[pas,tr] NP (PP)
```

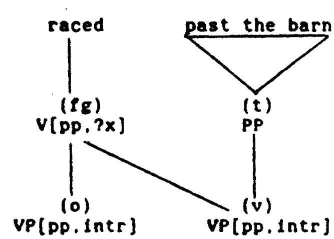
Similarly, R2c, R2d, and R2e cannot be collapsed. Consequently, nodes m, n and o



<u>Search Space 2</u>



<u>Subspace G</u>



<u>Subspace H</u>

cannot be collapsed, leaving a search space with three parsings of "past the barn."

4. GRAMMAR WITH RESTRICTED QUANTIFICATION: UNARY CONSTRAINTS

In the previous section extending the grammatical formalism with quantification over feature values enabled the collapse of several rules and a concomitant reduction in the search space. This section enhances the effect by further extension of the grammatical formalism.

Consider, once again, the difficulty encountered in the last section of replacing rules R2a and R2b with (3). The problem is one of overgeneralization; the variable ?v in (3) quantifies over all values of the voice feature--pp, pas and fin--while the rule is only correct when ?v takes on the values pp and pas. A way to achieve the desired effect is by using a device called <u>restricted quantification</u>. Whereas standard quantification can be used to say that some formula F is true when some variable x is assigned <u>any</u> individual in the domain, restricted quantification can be used to say that F is true when x is assigned to any individual drawn from a <u>subset</u> of the domain denoted by Tau. Syntactically this is written as "F / ?x:Tau". Tau is called a <u>sort symbol</u> and the subset of the domain it denotes is called a <u>sort</u>. On occasion I refer to unrestricted variables as if they were restricted; in such cases I am simply considering the variable to be restricted implicitly by the universal sort, the sort containing the entire domain.

With this enriched notation rules R2a' and R2b' (without the optional PP) can be collapsed into rule R2ab', where ACTIVE is a sort symbol denoting the set containing the pp and fin feature values.

```
R2ab')  VP(?v,tr,?x,?z) <- V(?v,tr,?x,?y) & NP(?x,?z) / ?v:ACTIVE
```

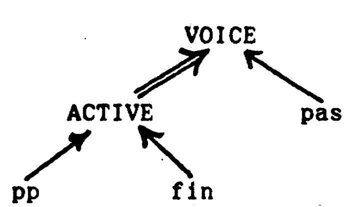
By integrating restricted quantification into the grammatical notation, rules R2 and R10 of Grammar 2 can be re-expressed as:

```
R2ab    )  VP[?v,tr]    --> V[?v,tr] NP (PP) / ?v:ACTIVE
R2c     )  VP[pas,tr]   --> V[pas,tr] (PP)
R2de    )  VP[?v,intr]  --> V[?v,intr] (PP) / ?v:ACTIVE
R10abcd)   V[?v,?t]     --> raced / ?v:ACTIVE
R10e    )  V[pas,tr]    --> raced
```

<u>Grammar 3</u>

A representation language with restricted quantification needs a way of representing certain relationships that hold among the sorts, to express, for example, that the active feature values are a subset of the voice feature values and that the active feature values and the transitivity feature values are disjoint. Furthermore, there is a need to express that certain individuals are members of certain sorts--for example, that fin is an active feature value. I refer to that part of the representation that expresses such information as the taxonomic representation and to a representation language that has restricted quantification and a taxonomic representation as an RQT language.

The taxonomy-based semantic-network systems common in AI can be thought of as weakly-expressive taxonomic representation languages. In such a notation the taxonomy of voice features can be written as



Nodes denoting individuals are labelled in lower case while those denoting sets of individuals are labelled in upper case. Double arrows connect a set to its superset and single arrows connect an individual to a set containing it. Notice that this representation contains complete information about the taxonomy of voice features.

My more-general research (Frisch, 1984) addresses some of the issues that arise when using taxonomic representations that do not contain complete information. This paper Ignores such issues; let us simply assume that there is some representation of the taxonomic information shown in the above semantic network—call it TR 3—and that we have a decision procedure for its logical consequences.

The resulting grammatical representation—that is the combination of Grammar 3 and TR 3—is logically equivalent to Grammar 2 but collapses some of its rules. What is needed now is an inference method that can use these collapsed rules to build a search space that collapses nodes found in Search Space 2. UR-resolution as it now stands does not suffice because it is only defined for clauses with standard quantification, not those with restricted quantification.

Before presenting an inference method that achieves these objectives, I first consider a straightforward approach, which turns out to be inadequate. Notice that every sentence containing a restricted quantifier is equivalent to one without. In particular, if T is a predicate symbol that is true on precisely the elements of Tau, then Horn clauses (5) and (6) are equivalent.

(5) A <- Bl * B2 & ... & Bn / ?x:Tau

(6) A <- Bl & B2 & ... & Bn & T(?x)

Hence, each rule of Grammar 3 could be re-expressed as a rule without restricted quantification and then parsed with UR-resolution. The resulting search space, however, would be isomorphic to Search Space 2. I leave confirmation of this claim as ah exercise for the reader.

The inference methods for handling restricted quantification that concern me derive from Reiter's (1977) work on logic data-bases. The key feature of these methods is that all reasoning with the taxonomic representation is done solely during unification. Hence, by merely extending the definition of unification to handle variables bound by restricted quantifiers, the UR-resolution inference rule becomes capable of handling an RQT language and can be used to parse Grammar 3.

Extending the standard notion of substitution to account for restricted variables leads to the notion of a tau-substitution relative to a taxonomic representation TAX. Only those substitutions that map variables to variables or constants need be considered for the purposes of this paper.

<u>Definition</u>: Let L be a language with taxonomic representation TAX. Then a <u>tau-substitution</u> (for L) is a total function s from the expressions of L to the expressions of L satisfying the conditions:

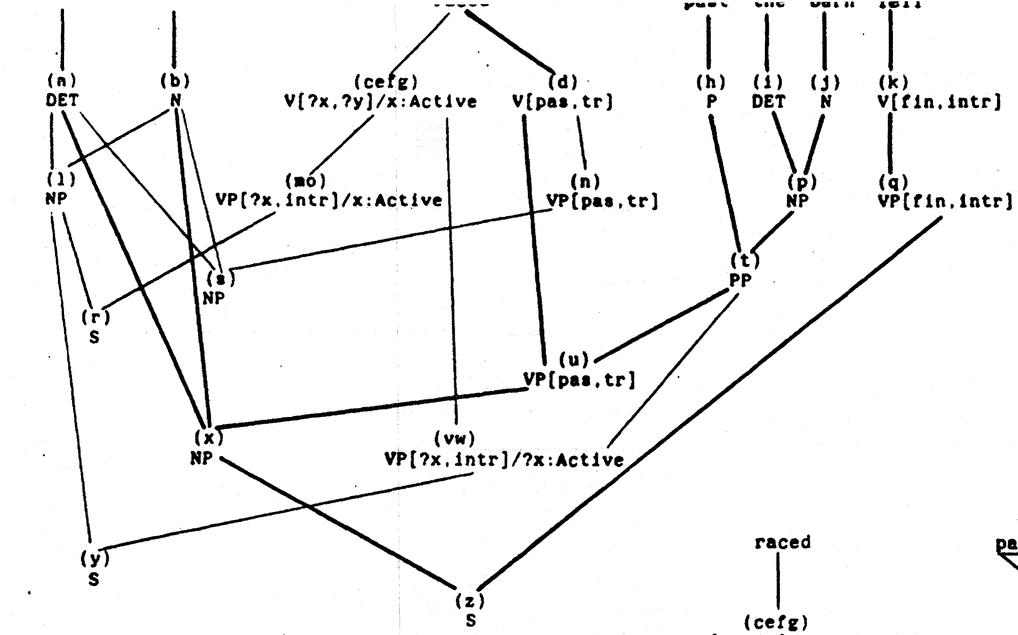
- for any constants c in L, s(c)=c
- for any expression e in L that is composed of expressions el, e2 . . . .en, 8(e) is composed of s(el), s(e2), ..., s(en) in the same manner
- for any variable x in L restricted by sort symbol R, s(x) is either
  a constant such that TAX logically implies that s(x) is an element of R, or
  a variable restricted by sort symbol R* such that TAX logically implies
  that R' is a subset of R

It is now a straightforward matter to extend the standard definitions of unification and UR-resolution to RQT languages. A <u>tau-unifler</u> of a set of expressions is a tau-substitution that maps every expression in the set to a single expression. Tau-UR-resolution is identical to UR-resolution except that the substitution Involved, theta, must be a tau-substitution.

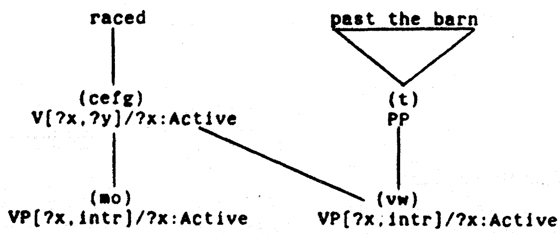
The above definition of tau-unification says nothing about its properties or how it can be computed. Walther (1984b) discusses some of the issues, but I'll remain silent other than to excite your curiosity by stating that there are circumstances in which there are multiple most-general tau-unifiers.

In the style of the first two search spaces, Search Space 3 displays all tau~UR-resolvents that could be produced in parsing the sample sentence with Grammar 3. This search space differs from the previous one in that Subspaces G and H have been collapsed into Subspace I resulting in a reduction of three nodes. Observe that the latest search space has only two parsings of "raced past the barn"—nodes u and vw. The net result of moving from Grammar 1 to Grammar 3 has been the replacement of Subspaces B through E by Subspace I. Subspace A has remained the same.

Now consider the task of collapsing Subspaces A and I and thereby achieving the goal, set in Section 2, of collapsing the five original subspaces, Subspaces A through E. In particular, consider collapsing rules RIOabcd and RlOe which would



Search Space 3



Subspace I

result in the collapse of nodes cefg and d.

The first rule covers the active form of the verb; "raced" is either a finite verb or a participial verb, and in either case it could be transitive or intransitive. If the variable ?v, which is restricted by Active, was broadened to include the passive case then the rule would allow "raced" to be a passive-intransitive verb, which should be excluded by the grammar. The problem arises because the grammar needs not only to impose constraints on the values of individual features but also to impose pairwise constraints on feature values. In this case, if "raced" is a passive verb then it must be transitive. The Generalized Phrase Structure Grammar formalism (Gazdar, Klein, Pullum and Sag, 1985) uses a device called "feature co-occurrence restrictions" to express such constraints. In this case, one could simply write "[pas] -> [tr]." The RQT grammar formalism presented so far contains no device for stating this constraint on feature values. The next section extends the grammar formalism with an analogous device, which is then used to collapse this grammar to its final form.

5. GRAMMAR WITH RESTRICTED QUANTIFICATION: BINARY CONSTRAINTS

This section differs from the others in that it merely sketches a grammatical formalism. Its primary objective is to demonstrate that a grammatical formalism that provides for the expression of binary--and perhaps even higher order--constraints between feature values can yield smaller search spaces. Once again the demonstration uses the same simple parsing problem and once again the smaller search space can be seen to be the result of using a minimum-commitment strategy.

As introduced in the last section, restricted quantification provides a variable that ranges over a subset of the domain. This notion can be generalized to include tuples of variables that range over a subset of tuples of domain elements. For current purposes, only 2-tuples are needed. A statement of the form "F / <?x,?y>:C" is true if the formula F is true when ?x and ?y are assigned to every pair of individuals drawn from the set of pairs denoted by C. C is called a <u>constraint symbol</u> and the set it denotes is called a <u>constraint</u>.

Once again, every sentence with this new notation is equivalent to one witho particular, if C' is a binary predicate symbol that is true only on the pai from C then Horn clauses (7) and (8) are equivalent.

```
(7)  A <- B1 & B2 & ... & Bn / <?x,?y>:C
(8)  A <- B1 & B2 & ... & Bn & C'(?x,?y)
```

Just as a taxonomic representation is needed to express knowledge about sor representation is needed to express knowledge about constraints. Th involved in designing such a language are skirted in this paper. I will write the set of tuples denoted by a constraint symbol. In this case let C be defined as:

```
C1={<fin,tr>,<pp,tr>,<pas,tr>,<fin,intr>,<pp,intr>}
C2={<pas,tr>,<fin,intr>,<pp,intr>}
```

By incorporating quantifiers with constraints into the grammatical notation and R10 of Grammar 3 can be re-expressed as

```
R2ab     ) VP[?v,tr]    --> V[?v,tr] NP (PP) / ?v:ACTIVE
R2cde    ) VP[?v,?t]    --> V[?v,?t] (PP) / <?v,?t>:C2
R10abcde) V[?v,?t]      --> raced / <?v,?t>:C1
```

<u>Grammar 4</u>

Rather than delve into a discussion of the interesting problems associat deduction in this logic, I present a simplistic, ad-hoc strategy that can be this case to get the completely-collapsed search space. Because the con variables always occur in pairs and no variable is involved in more than straint, we can treat a pair of variables as a single variable, and a bina straint as a sort. This results in an RQT logic with only unary constra therefore tau-UR-resolution can be used to parse the grammar. This pars place in Search Space 4, where Subspaces A and I are collapsed to Subspac the current formal system, Subspace J corresponds to Subspace F, which was p at the end of Section 2 as an informal representation of the most desirabl space for this parsing problem.

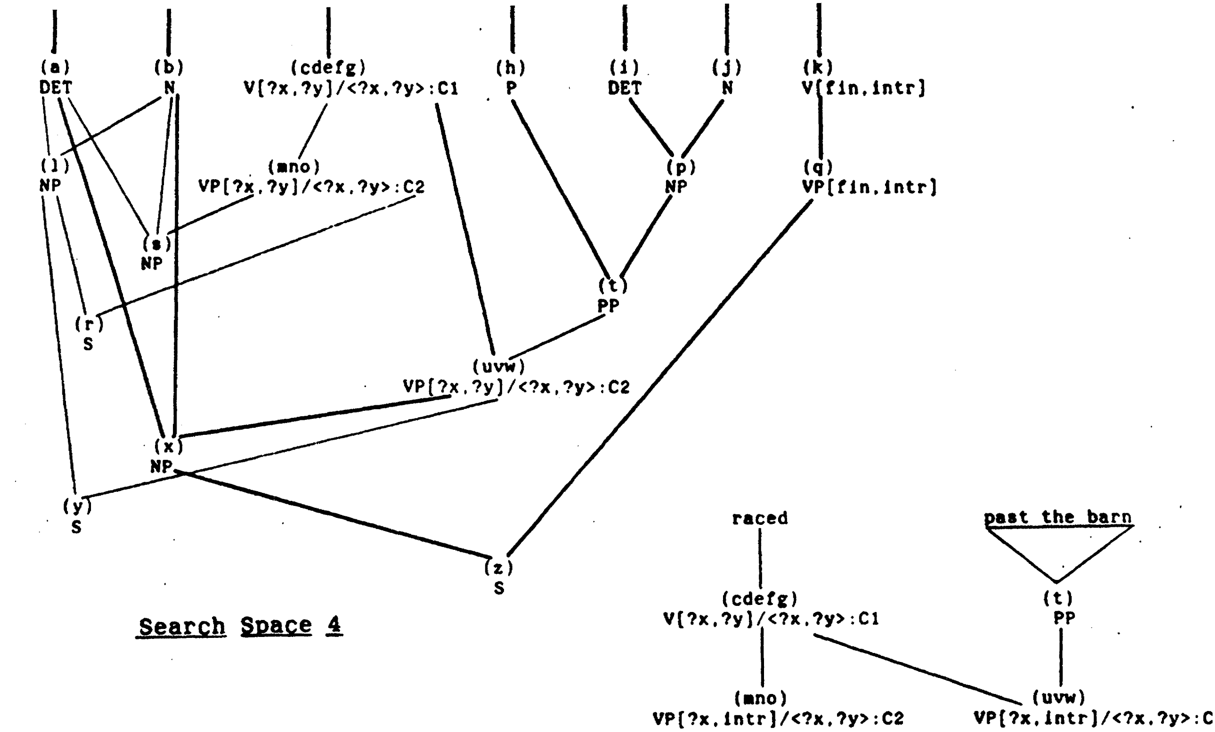
Recall that the last paragraph of Section 2 pointed out the problem o lated feature values in rules R2', R2" and R10*, and in Subspace F. The cor ing objects in the current system--rules R2ab, R2cde and R10abcde and Subsp overcome this difficulty by using variables whose multiple occurrences m stand for the same feature value. Beyond this there is little to be said ab space J as the entire discussion of Subspace F carries over intact.

Despite the elimination of redundancy Search Space 4 still conta garden-path parse represented at node (y). However, this parse and the parse share much more structure in Search Space 4 than in Search Space 1. fore, in recovering from the garden path a parser need not recreate as muc ture.

6. RELATED WORK

Theoretical linguists interested in capturing grammatical generalizatio done a significant amount of work on rule-collapsing techniques. It interesting to see if techniques derived with this motivation diverge fro derived with the motivation of efficient parsing. A conceivable cause divergence is that rule-collapsing may lead to smaller search spaces but not sarily increased parsing efficiency unless the correct instance of a gen rule can be computed easily. Unification, which finds rule instances in a with quantification, has well-studied efficient algorithms. Tau-unificatio finds rule instances in a grammar with unary-restricted quantification, i studied by several researchers and appears well-behaved in a wide range o tions. I know of no work directed at the problem for higher-order restricti

The grammatical formalism used in Section 2 is the standard starting po work on deductive parsing. I suspect that anyone who has played with logic mars has used variables to quantify over feature values as in Section 3, tho been unable to find any publications that specifically discuss the use



Search Space 4

Subspace J

variables in the lexicon.

Logical systems with unary-restricted quantifiers, and their cousins, sorted logics, have drawn moderate attention recently from those interested in efficient deduction. Cohn (1983a; 1983b) has investigated an inference system for a sorted logic featuring a highly-expressive taxonomic representation and restrictions on arguments to both function and predicate symbols. Walther (1982; 1983) has worked on a similar system for a language with a less expressive taxonomic representation, though incorporating restricted quantification and equality. Walther's system automatically has found a proof to Schubert's Steamroller problem (Walther, 1984a), a problem whose solution has eluded automated deduction systems based on standard FOPC. Though Cohn's system is unimplemented, he has argued (Cohn, 1984) that it should solve the Steamroller even more effectively than Walther's system. The HORNE logic-programming system (Allen, Giuliano and Frisch, 1983; Frisch, Allen and Giuliano, 1983) is based on an RQT Horn-clause logic. Its implementation incorporates a number of effective methods for dealing with a taxonomic representation. HORNE has been used to implement an inference-based knowledge retriever that operates on a knowledge base of sentences in an RQT language (Frisch and Allen, 1982). All of the above systems could be used to solve the parsing problem examined in this paper and would exhibit the minimum-commitment inference strategy displayed in Search Space 3.

To my knowledge, the idea presented in Section 5 of using variables with binary restrictions has not been addressed elsewhere. Tony Cohn has suggested privately that binary constraints could be encoded in the sort mechanism of his logic but it is yet to be seen whether this would yield the desired gain in efficiency.

7. CONCLUSIONS

By extending a logical system with restricted quantification, search spaces exhibiting a minimum-commitment strategy can be built. This has been demonstrated by considering search spaces for a simple parsing problem. Several researchers are investigating systems where quantifiers are restricted by unary constraints, but I know of no work directly concerned with higher-order constraints. Because restricted-quantification inference systems eliminate search-space redundancy of the kind examined here (and possibly other kinds), results on these systems may turn out to be extremely useful in the construction of efficient parsers.

ACKNOWLEDGEMENTS

I am most grateful to Roger Evans for generously assisting me in identifying an appropriate parsing problem and in debugging some initial search spaces. I also thank Tony Cohn, Gerald Gazdar, Steve Isard and Chris Hellish for enlightening discussions and useful suggestions that benefited this paper and Fran Evelyn, whose efforts in proofreading and final layout are evident. This research is partially supported by grant GR/D/16062 from the Science and Engineering Research Council.

REFERENCES

Allen, J.F., M.E. Giuliano and A.M. Frisch, "The HORNE reasoning system," Technical Report 126, Computer Science Dept., U. Rochester, December, 1983.

Cohn, A.G., "A Note concerning the axiomisation of Schubert's steamroller in many-sorted logic,** Alvey Workshop on Inference, Imperial College, Sept., 1984.

Cohn, A.G., "Mechanising a particularly expressive many sorted logic," Ph.D. Thesis, Dept. of Computer Science, Univ. of Essex, 1983a.

Cohn, A.G., "Improving the expressiveness of many sorted logic," in Proceedings o£ the Third National Conference on Artificial Intelligence, August, 1983b.

Frisch, A.M., "An investigation into inference with restricted quantification and a taxonomic representation," Cognitive Science Research Report 041, Cognitive Studies Programme, U. Sussex, September, 1984. Also in Alvey IKBS Inference Research Theme Workshop Ji, September, 1984.

Frisch, A.M. and J.F. Allen, "Knowledge retrieval as limited inference," in D.W. Loveland (Ed.), Lecture Notes in Computer Science: 6th Conference on Automated Deduction. New York: Springer-Verlag, 1982.

Frisch, A.M., J.F. Allen and M Giuliano, "An overview of the HORNE logic programming system," SIGART Newsletter, No. 84, 1983. Also in, Logic Programming Newsletter, No. 5, Winter, 1983/4.

Gazdar, G., E. Klein, G.K. Pullum and I.A. Sag, Generalized Phrase Structure Grammar, Oxford: Blackwell, 1985.

Kowalski, R., Logic for Problem Solving. New York: North Holland, 1979.

Matsumoto, Y., H. Tanaka, and M. Kiyono, "BUP: A bottom-up parsing system for natural languages," in M. van Caneghem and D.H.D. Warren (Eds.), Logic Programming and its Applications. Norwood, NJ: Ablex, 1985.

McCharen, J., R. Overbeek and L. Wos, "Complexity and related enhancements for automated theorem-proving programs," Computers and Mathematics with Applications, 2, 1-16, 1976.

Pereira, F.C.N. and D.H.D. Warren. "Parsing as deduction," Proc. o£ the 21st Annual Meeting of the Assoc. for Computational Linguistics, 1983.

Pereira, F.C.N. and D.H.D. Warren. "Definite clause grammars for language analysis— Survey of the formalism and a comparison with augmented transition networks," Artificial Intelligence, 13, 231-278, 1980.

Reiter, R., "An approach to deductive question-answering," BBN Technical Report 3649, Bolt Beranek and Newman, Inc., Cambridge, MA., 1977.

Walther, C, "A mechanical solution of Schubert's steamroller by many-sorted resolution," Proc. of the Fourth National Conference on Artificial Intelligence, 1984a.

Walther, C, "Unification in many-sorted theories," Proc. of the Sixth European Conference on Artificial Intelligence, 1984b.

Walther, C, "A many-sorted calculus based on resolution and paramodulation," Proc. £f the Eighth International Joint Conference on Artificial Intelligence, August, 1983.

Walther, C. "A many-sorted calculus based on resolution and paramodulation/' Interner Bericht Nr. 34/82, Universitat Karlsruhe, Fakultat fur Informatik, 1982.

Wos, L., S. Winker, B. Smith, R. Veroff and L. Henschen, "A new use of an automated reasoning assistant: open questions in equivalential calculus and the study of infinite domains," Artificial Intelligence, 22, 303-356, 1984.