Natural Language Based Inference Procedures applied to Schubert's Steamroller

Robert Givan
rlg@ai.mit.edu

David McAllester
dam@ai.mit.edu

Sameer Shalaby

Abstract

We have previously argued that the syntactic structure of natural language can be exploited to construct powerful polynomial time inference procedures. This paper supports the earlier arguments by demonstrating that a natural language based polynomial time procedure can solve Schubert's steamroller in a single step.

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the work described in this paper was provided in part by Misubishi Electric Research Laboratories, Inc. Support for the laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-85-K-0124.

This paper appeared in AAAI-91. A postscript electronic source for this paper can be found in ftp.ai.mit.edu:/pub/dam/aaaib.ps. A bibtex reference can be found in ftp.ai.mit.edu:/pub/dam/dam.bib.

1 Introduction

Schubert's steamroller is a well known challenge problem for automated reasoning systems [Cohn, 1989], [Cohn, 1984], [Stickel, 1985], [Bibel *et al.*, 1987], [Davies, 1988], [Oppacher and Suen, 1986], and [Walther, 1984]. All previous automated solutions of Schubert's steamroller have been based on search procedures — procedures that search for proofs and which are not guaranteed to halt. In this paper we describe a different kind of solution to Schubert's steamroller — a solution without search. Although there is no established technical meaning to the term "search", it seems reasonable to assert that a procedure guaranteed to terminate in polynomial time does not search. We present a natural general purpose polynomial time inference procedure capable of solving Schubert's steamroller given only the axioms of the problem plus three user specified "focus terms".

The polynomial time inference procedure presented in this paper exploits, in an apparently essential way, aspects of natural language syntax. The inference procedure is defined by a set of inference rules. If R is a set of inference rules then we let $\vdash_R$ be the inference relation generated by R, i.e., for any set of formulas Σ and formula Φ we write $\Sigma \vdash_R \Phi$ if there exists a derivation of Φ from the formulas in Σ using the inference rules in R. A rule set R will be called *tractable* if $\vdash_R$ is polynomial time decidable, i.e., there exists a procedure that is guaranteed to terminate in polynomial time in the written length of Σ and Φ and that determines whether or not $\Sigma \vdash_R \Phi$. There exist useful, though incomplete, tractable sets of inference rules for first order logic. However, much more powerful tractable rule sets can be given if first order formulas are written in a non-standard syntax. A "taxonomic syntax" for first order logic is presented in [McAllester *et al.*, 1989]. A more elaborate "Montagovian syntax" for first order logic, incorporating quantificational aspects of English noun phrases, is presented in [McAllester and Givan, 1989]. The polynomial time inference procedure used here to solve Schubert's steamroller is defined by a set of inference rules stated in a Montagovian syntax. The inference relation defined by these inference rules appears not to be definable in the classical syntax of first order logic.

2 Montagovian Syntax

In this section we present a Montagovian syntax for first order logic similar to that described in [McAllester and Givan, 1989]. The classical syntax for first order logic involves two grammatical categories — formulas and terms. The Montagovian syntax presented below also involves two syntactic categories — formulas and class expressions. Formulas denote truth values and class expressions denote sets. Constant symbols and variables are treated as class expressions that denote singleton sets. In the following, Φ is a formula; C, C_1, C_2 are class expressions; and R is a binary relation symbol.

- A *class expression* is one of the following:
  - A class symbol (monadic predicate symbol).
  - A constant symbol or variable.
  - An intersection expression (intersection C_1 C_2).
  - A union expression (union C_1 C_2).
  - A λ-expression $(\lambda\ x\ \Phi(x))$, where x is a variable.
  - An expression (R (some C)) or (R (every C)).
- A *formula* is one of the following:
  - A subset formula of the form (every C_1 C_2).
  - An intersection formula of the form (some C_1 C_2).
  - An existence formula of the form (there-exists-a C).
  - An at-most-one formula of the form (at-most-one C).
  - Any Boolean combination of the above formulas.

Before giving a formal semantics, it is useful to consider some examples of formulas and their associated meanings. If P and Q are class symbols then (every P Q) is a formula which is true if the set denoted by P is a subset of the set denoted by Q. If man is a class symbol that denotes the set of all men, and runs is a class symbol that denotes the set of all things that run, then the formula (every man runs) is true if every man runs. The formula (some man runs) is true if some man runs.

If John is a constant symbol (or variable) then the formulas (every John runs) and (some John runs) are semantically equivalent and we can use (John runs) as an abbreviation for either formula. Similarly, we write (likes John) as an abbreviation for either of the class expressions (likes (every John)) or (likes (some John)).

If owns is a relation symbol, and denotes the predicate which is true of two objects if the first owns the second, then the class expression (owns (some car)) denotes the set of individuals that own some car. If policeman is a class symbol that denotes the set of all policemen, then the formula (every policeman (owns (some car))) is true if every policeman owns a car.

Our formal semantics for the Montagovian syntax is a (drastic) simplification of Montague's original semantics for English [Montague, 1973].[1] Just as in classical syntax, a model of our Montagovian language is a first order model, i.e., a domain D together with an interpretation of constant, class, and relation symbols. Each first order model interprets each constant symbol as an element of its domain. A model also interprets each class symbol as a subset of its domain and each relation symbol as a binary relation on its domain, i.e., a set of pairs of domain elements.

If $\mathcal{M}$ is a first order model, and ρ is a variable interpretation over $\mathcal{M}$, i.e., a mapping from variables to elements of the domain of $\mathcal{M}$, then we write $\mathcal{V}(e, \mathcal{M}, \rho)$ for the semantic value of the expression e in the model $\mathcal{M}$ under variable interpretation ρ. If C is a class expression then $\mathcal{V}(C, \mathcal{M}, \rho)$ is a subset of the domain of $\mathcal{M}$. If Φ is a formula, then $\mathcal{V}(\Phi, \mathcal{M}, \rho)$ is a truth value, either **T** or **F**.

- For class symbol P, $\mathcal{V}(P,\ \mathcal{M},\ \rho)$ is the set $\mathcal{M}(P)$.
- For constant c, $\mathcal{V}(c,\ \mathcal{M},\ \rho)$ is the singleton set $\{\mathcal{M}(c)\}$.
- For variable x, $\mathcal{V}(x,\ \mathcal{M},\ \rho)$ is the singleton set $\{\rho(x)\}$.
- $\mathcal{V}($`(intersection` C_1 C_2`)`$,\ \mathcal{M},\ \rho)$ is $\mathcal{V}(C_1,\ \mathcal{M},\ \rho) \cap \mathcal{V}(C_2,\ \mathcal{M},\ \rho)$.
- $\mathcal{V}($`(union` C_1 C_2`)`$,\ \mathcal{M},\ \rho)$ is the set $\mathcal{V}(C_1,\ \mathcal{M},\ \rho) \cup \mathcal{V}(C_2,\ \mathcal{M},\ \rho)$.
- $\mathcal{V}((\lambda\ x\ \Phi(x)),\ \mathcal{M},\ \rho)$ is the set of all d such that $\mathcal{V}(\Phi(x),\ \mathcal{M},\ \rho[x := d])$ is **T** where $\rho[x := d]$ is the same as ρ except that it interprets x as d.
- $\mathcal{V}($`(`R `(every` C`))`$, \mathcal{M}, \rho)$ is the set of all d such that for every d' in $\mathcal{V}(C, \mathcal{M}, \rho)$ the pair $<d,\ d'>$ is an element of the relation denoted by R. (Consider the class expression `(loves (every child))`.)
- $\mathcal{V}($`(`R `(some` C`))`$, \mathcal{M}, \rho)$ is the set of all d such that there exists an element d' in $\mathcal{V}(C, \mathcal{M}, \rho)$ such that that the pair $<d,\ d'>$ is an element of the relation denoted by R. (Consider the class expression `(loves (some child))`.)
- $\mathcal{V}($`(every` C W`)`$,\ \mathcal{M},\ \rho)$ is **T** if the set $\mathcal{V}(C,\ \mathcal{M},\ \rho)$ is a subset of $\mathcal{V}(W,\ \mathcal{M},\ \rho)$.
- $\mathcal{V}($`(some` C W`)`$,\ \mathcal{M},\ \rho)$ is **T** if the set$\mathcal{V}(C,\ \mathcal{M},\ \rho) \cap \mathcal{V}(W,\ \mathcal{M},\ \rho)$ is non-empty.
- $\mathcal{V}($`(there-exists-a` C`)`$,\ \mathcal{M},\ \rho)$ is **T** if $\mathcal{V}(C,\ \mathcal{M}\ \rho)$ is non-empty.
- $\mathcal{V}($`(at-most-one` C`)`$,\ \mathcal{M},\ \rho)$ is **T** if $\mathcal{V}(C,\ \mathcal{M}\ \rho)$ has at most one member.
- Boolean combinations of atomic formulas have their standard meaning.

Binary relation symbols, in the presence of equality, are in some sense sufficient to express arbitrary first order facts. We leave it to the reader to verify that, if we restrict our attention to languages with only constants and unary and binary

[1] Our class expressions play the role of both verb phrases, as in `(owns (some car))`, and of incomplete noun phrases, as in `(brother-of (some policeman))`. Montague, of course, treated these as separate syntactic categories. Montague also treated complete noun phrases, such as `(every policeman)`, as another syntactic category with its own denotational semantics. The treatment of propositional attitudes makes Montague's formal language yet more complex.

relation symbols, then every classical first order formula can be translated to a logically equivalent formula of Montagovian syntax and vice versa. Montagovian syntax is really just a syntactic variant of first order logic.

3 Schubert's Steamroller

Schubert's steamroller is a logical puzzle originally stated in English. Each sentence of the English statement of the problem is given below along with a translation of that sentence into a set of formulas in our Montagovian syntax for first order logic.

Wolves, foxes, birds, caterpillars, and snails are animals, and there are some of each of them.

```
(every wolf animal)  (there-exists-a wolf)
(every fox animal)   (there-exists-a fox)
...
```

There are some grains, and grains are plants.

```
(there-exists-a grain) (every grain plant)
```

Caterpillars and snails are much smaller than birds, which are much smaller than foxes, which are much smaller than wolves.

```
(every caterpillar (is-smaller-than (every bird)))
(every snail (is-smaller-than (every bird)))
(every bird (is-smaller-than (every fox)))
(every fox (is-smaller-than (every wolf)))
```

Wolves do not like to eat foxes or grains, while birds like to eat caterpillars but not snails.

```
¬(some wolf (eats (some fox)))            ¬(some wolf (eats (some grain)))
(every bird (eats (every caterpillar)))  ¬(some bird (eats (some snail)))
```

Caterpillars and snails like to eat some plants.

```
(every caterpillar (eats (some plant)))
(every snail (eats (some plant)))
```

Every animal either likes to eat all plants or all animals much smaller than itself that like to eat some plants.

```
(every animal
  (union
    (eats (every plant))
    (λ x (x (eats (every
                    (intersection
                      animal
                      (is-smaller-than x))))))))
```

Prove there is an animal that likes to eat a grain-eating animal.

```
(some animal
      (eats (some (intersection
                    animal
                    (eats (some grain))))))
```

A formula of Montagovian Syntax is called *quantifier-free* if it does not contain any λ-expressions. The "quantifiers" `some` and `every` that appear in noun phrases are considered to be quantifier-free combinators. In [McAllester and Givan, 1989] we show that satisfiability is decidable (NP-complete) for the quantifier-free fragment of the Montagovian syntax presented in that paper. Although the Montagovian syntax presented here is somewhat more elaborate, we conjecture that the quantifier-free fragment remains decidable. When translated into our Montagovian syntax, all of the sentences of Schubert's steamroller are quantifier-free except for the second to last sentence above, which involves a single λ-expression.

4 Polynomial Time Inference

Figure 1 gives a set of 33 inference rules stated in our Montagovian syntax. We are actually interested in the rules in figure 1 *plus* all contrapositives of those rules. Each inference rule is analogous to an implication of the form $\Psi_1 \wedge \cdots \Psi_n \rightarrow \Phi$ where each Ψ_i is an antecedent and Φ is the conclusion. A *contrapositive* of a rule $\Psi_1 \wedge \cdots \Psi_n \rightarrow \Phi$ is a rule of the form

$$\Psi_1 \wedge \cdots \wedge \Psi_{i-1} \wedge \neg\Phi \wedge \Psi_{i+1} \wedge \cdots \wedge \Psi_n \rightarrow \neg\Psi_i.$$

In the contrapositive, the conclusion has been interchanged with one of the antecedents and both of the interchanged formulas have been negated. If a given rule is semantically sound, then so is each of its contrapositives. We conjecture that the rule set consisting of the rules in figure 1 plus all contrapositives of those

(1) $\dfrac{\Psi,\ \neg\Psi}{\Phi}$ (2) $\dfrac{\Psi}{\neg\neg\Psi}$ (3) $\dfrac{\neg\neg\Psi}{\Psi}$

(4) $\dfrac{\Phi}{\Psi \vee \Phi}$ (5) $\dfrac{\Phi}{\Phi \vee \Psi}$ (6) $\dfrac{\Phi \vee \Psi,\ \neg\Phi}{\Psi}$

(7) $\texttt{(every } C\ C\texttt{)}$ (8) $\texttt{(there-exists-a } c\texttt{)}$ (9) $\texttt{(at-most-one } c\texttt{)}$

(10) $\dfrac{\texttt{(there-exists-a } C\texttt{)}}{\texttt{(some } C\ C\texttt{)}}$ (11) $\dfrac{\texttt{(some } C\ W\texttt{)}}{\texttt{(there-exists-a } C\texttt{)}}$ (12) $\dfrac{\texttt{(some } C\ W\texttt{)}}{\texttt{(some } W\ C\texttt{)}}$

(13) $\dfrac{\begin{array}{l}\texttt{(every } C\ W\texttt{)}\\ \texttt{(every } W\ Z\texttt{)}\end{array}}{\texttt{(every } C\ Z\texttt{)}}$ (14) $\dfrac{\begin{array}{l}\texttt{(some } C\ W\texttt{)}\\ \texttt{(every } C\ Z\texttt{)}\end{array}}{\texttt{(some } Z\ W\texttt{)}}$ (15) $\dfrac{\begin{array}{l}\texttt{(some } C\ W\texttt{)}\\ \texttt{(at-most-one } C\texttt{)}\end{array}}{\texttt{(every } C\ W\texttt{)}}$

(16) $\dfrac{\begin{array}{l}\texttt{(at-most-one } W\texttt{)}\\ \texttt{(every } C\ W\texttt{)}\end{array}}{\texttt{(at-most-one } C\texttt{)}}$ (17) $\dfrac{\neg\texttt{(at-most-one } C\texttt{)}}{\texttt{(there-exists-a } C\texttt{)}}$ (18) $\dfrac{\neg\texttt{(every } C\ W\texttt{)}}{\texttt{(there-exists-a } C\texttt{)}}$

(19) $\texttt{(every } C\ \texttt{(union } C\ W\texttt{))}$ (20) $\texttt{(every } W\ \texttt{(union } C\ W\texttt{))}$

(21) $\texttt{(every (intersection } C\ W\texttt{) } C\texttt{)}$ (22) $\texttt{(every (intersection } C\ W\texttt{) } W\texttt{)}$

(23) $\dfrac{\texttt{(every } C\ Z\texttt{), (every } W\ Z\texttt{)}}{\texttt{(every (union } C\ W\texttt{) } Z\texttt{)}}$ (24) $\dfrac{\texttt{(every } Z\ C\texttt{), (every } Z\ W\texttt{)}}{\texttt{(every } Z\ \texttt{(intersection } C\ W\texttt{))}}$

(25) $\dfrac{\texttt{(some } C\ W\texttt{)}}{\texttt{(there-exists-a (intersection } C\ W\texttt{))}}$ (26) $\dfrac{\neg\texttt{(there-exists-a } C\texttt{)}}{\texttt{(every } W\ \texttt{(}R\ \texttt{(every } C\texttt{)))}}$

(27) $\dfrac{\begin{array}{l}\texttt{(every } C\ \texttt{(union } W\ Z\texttt{))}\\ \neg\texttt{(some } C\ W\texttt{)}\end{array}}{\texttt{(every } C\ Z\texttt{)}}$ (28) $\dfrac{\begin{array}{l}\texttt{(every } C\ \texttt{(union } W\ Z\texttt{))}\\ \neg\texttt{(some } C\ Z\texttt{)}\end{array}}{\texttt{(every } C\ W\texttt{)}}$

(29) $\dfrac{\texttt{(every } C\ W\texttt{)}}{\texttt{(every (}R\ \texttt{(some } C\texttt{)) (}R\ \texttt{(some } W\texttt{)))}}$ (30) $\dfrac{\texttt{(every } C\ W\texttt{)}}{\texttt{(every (}R\ \texttt{(every } W\texttt{)) (}R\ \texttt{(every } C\texttt{)))}}$

(31) $\dfrac{\texttt{(some } C\ W\texttt{)}}{\texttt{(every (}R\ \texttt{(every } C\texttt{)) (}R\ \texttt{(some } W\texttt{)))}}$ (32) $\dfrac{\texttt{(there-exists-a (R (some } C\texttt{)))}}{\texttt{(there-exists-a } C\texttt{)}}$

Figure 1: Some inference rules for Montagovian Syntax. The letters C, W, and Z, range over class expressions, c ranges over constants and variables, Φ and Ψ range over formulas, and R ranges over relation symbols.

rules is *local* (see below), and thus generates a polynomial time decidable inference relation.

The inference rules in figure 1, together with their contrapositives, determine a sound inference relation for formulas expressed in our Montagovian syntax for first order logic. This (incomplete) first order inference relation appears not have any definition in the classical syntax for first order logic. [2] We have constructed a polynomial time inference procedure based on this set of inference rules. A general theoretical framework for constructing polynomial time inference procedures is presented in [McAllester, 1990]. Let R be *any* set of inference rules. The following definition is from [McAllester, 1990].

> **Definition:** We write $\Sigma \vdash\!\!\dashv_R \Phi$ if there exists a proof of Φ from the premise set Σ such that every *proper subexpression* of a formula used in the proof appears as a proper subexpression of Φ, a proper subexpression of some formula in Σ, or as a closed (variable free) expression in the rule set R.

The following lemma is proved in [McAllester, 1990] .

> **Lemma:** For any given rule set R, there exists a procedure for determining whether or not $\Sigma \vdash\!\!\dashv_R \Phi$ which runs in time polynomial in the written length of Σ and Φ.

The inference relation $\vdash\!\!\dashv_R$ is a restricted version of $\vdash_R$. For any rule set R, the relation $\vdash\!\!\dashv_R$ is polynomial time decidable. If the relation $\vdash_R$ is intractable, as is the case for any sound and complete set of rules for first order logic, then the polynomial time relation $\vdash\!\!\dashv_R$ will be weaker than the relation $\vdash_R$. However, there is a large class of rule sets for which these two relations are the same. The following definition is also from [McAllester, 1990] .

> **Definition:** A set R of inference rules is called *local* if the relation $\vdash\!\!\dashv_R$ is the same as the relation $\vdash_R$.

An immediate consequence of the above definitions and lemma is that local rule sets are tractable, i.e., they generate polynomial time decidable inference relations. A variety of nontrivial local rule sets is presented in [McAllester, 1990]. Let M be the set of inference rules in figure 1 together with the contrapositives of those rules. We conjecture, although we have not yet proved, that M is local. Even if M is not local, $\vdash\!\!\dashv_M$ is still polynomial time decidable, and it appears to be a very powerful inference relation.

---

[2] This is because the variables in the rules of figure 1 range over class expressions, but there are no class expressions in classical syntax. Consider for example the classical equivalent of the Montagovian class expression `(brother-of (every man))`.

5 Socratic Proof Systems

Local rule sets define polynomial time inference procedures. Of course, no polynomial time inference procedure can be complete for first order logic — the 32 inference rules given in the previous section are not complete for our Montagovian syntax for first order logic. However, it is possible to exploit fast and powerful inference procedures based on Montagovian syntax in constructing semi-automated verification systems. In this section we describe a particular kind of semi-automated verification system called a *Socratic Sequent system.*[3] A proof in a Socratic sequent system is a series of lines where each line is a sequent of the form $\Sigma \vdash \Phi$ where Σ is a set of formulas and Φ is a formula.

> **Definition:** A *Socratic sequent system* is a pair $<R,\ S>$ where R is a set of inference rules (deriving formulas from formulas) and S is a set of sequent rules (deriving sequents from sequents).

> **Definition:** An *acceptable derivation* in a Socratic sequent system $<R, S>$ is a series of sequents where, for each sequent $\Sigma \vdash \Phi$, either $\Sigma \ \vdash_R \Phi$ (in which case the sequent is called *obvious*), or the sequent follows from earlier sequents using a rule in S.

If the rule set R that defines the obvious sequents is local, then the inference relation $\vdash_R$ is polynomial time decidable, and one can therefore determine, in polynomial time, whether a series of sequents is an acceptable derivation in the sequent system $<R,\ S>$. (Note that *finding* an acceptable derivation of Φ from Σ is still an undecidable operation—the critical point is that once we have such a derivation, we can *verify* that it is acceptable in polynomial time).

In this section we give a Socratic sequent system that is complete for our Montagovian syntax for first order logic and show how this Socratic system yields a one-step solution to Schubert's steamroller. The sequent rules for our proof system are given in figure 2. The rules of obviousness of our Socratic system include all of the inference rules in figure 1 plus the following two rules concerning λ-expressions:

(33) $$\frac{\Phi(y),\ \texttt{(focus-on}\ y\texttt{)}}{\texttt{(every}\ y\ (\lambda\, x\, \Phi(x)))}$$

[3] The term "Socratic proof" was introduced in [Crawford and Kuipers, 1989] to describe any system in which steps of a proof are verified using an automated reasoning procedure. Our notion of a Socratic sequent system is a special case of this general concept.

(S1) $\dfrac{\Sigma \cup \{\Psi\} \vdash \Phi \quad \Sigma \cup \{\neg\Psi\} \vdash \Phi}{\Sigma \vdash \Phi}$

(S2) $\dfrac{\Sigma \vdash \Psi \quad \Sigma \cup \{\Psi\} \vdash \Phi}{\Sigma \vdash \Phi}$

(S3) $\dfrac{\Sigma \cup \texttt{(focus-on } x\texttt{)} \vdash \Phi}{\Sigma \vdash \Phi}$

(S4) $\dfrac{\Sigma \vdash \Phi}{\Sigma \cup \{\Psi\} \vdash \Phi}$

(S5) $\dfrac{\Sigma \vdash \texttt{(there-exists-a } C\texttt{)} \quad \Sigma \cup \{\texttt{(every } x \ C\texttt{)}\} \vdash \Phi}{\Sigma \vdash \Phi}$

(S6) $\dfrac{\Sigma \vdash \texttt{(some } C \ W\texttt{)} \quad \Sigma \cup \{\texttt{(every } x \ C\texttt{)}, \texttt{(every } x \ W\texttt{)}\} \vdash \Phi}{\Sigma \vdash \Phi}$

(S7) $\dfrac{\Sigma \cup \{\texttt{(every } x \ C\texttt{)}\} \vdash \texttt{(every } W \ \texttt{(}R \ x\texttt{))}}{\Sigma \vdash \texttt{(every } W \ \texttt{(}R \ \texttt{(every } C\texttt{)))}}$

(S8) $\dfrac{\Sigma \cup \{\texttt{(every } x \ C\texttt{)}\} \vdash \texttt{(every } x \ W\texttt{)}}{\Sigma \vdash \texttt{(every } C \ W\texttt{)}}$

(S9) $\dfrac{\Sigma \vdash \texttt{(at-most-one } Z\texttt{)} \quad \Sigma \vdash \texttt{(every } Z \ \texttt{(}R \ \texttt{(some } C\texttt{)))} \quad \Sigma \cup \{\texttt{(every } x \ C\texttt{)}, \texttt{(every } Z \ \texttt{(}R \ x\texttt{))}\} \vdash \Phi}{\Sigma \vdash \Phi}$

(S10) $\dfrac{\Sigma \cup \{\texttt{(every } x_1 \ C\texttt{)}, \texttt{(every } x_2 \ C\texttt{)}\} \vdash \texttt{(every } x_1 \ x_2\texttt{)}}{\Sigma \vdash \{\texttt{(at-most-one } C\texttt{)}\}}$

Figure 2: The Socratic Proof Rules. In these rules C, W, and Z, are class expressions, Φ is a formula, and x, x_1 and x_2 are variables that do not appear free in Σ, Φ, C, Z, or W.

(34) $\dfrac{\texttt{(every } y \ (\lambda\, x\, \Phi(x))\texttt{)}, \texttt{(focus-on } y\texttt{)}}{\Phi(y)}$

Each of these rules has an antecedent of the form (focus-on$y$), where y must be variable. Formulas of this form are used to control the inference process and have no semantic content. The focus-on antecedents of the above rules restrict the application of these rules to "focus variables", i.e., variables y such that the formula (focus-on$y$) is given as a premise (there are no inference rules for deriving formulas of the form (focus-on$y$)). Note that the sequent rule S3 in figure 2 can be used to eliminate focus-on premises from sequents. If the λ-expression rules were not restricted with focus-on antecedents, then the inference relation defined by those rules, together with the rules of figure 1, would be undecidable. Let M' be the set of inference rules including all rules in figure 1 and their contrapositives, plus the above two rules for quantifiers. We have a polynomial time implementation of the inference relation $\vDash_{M'}$, provided there is a bounded level of λ-nesting. This implementation is constructed along the lines described in [McAllester, 1989]. We conjecture that M' is local, and thus that $\vDash_{M'}$ is the same as $\vdash_{M'}$.

Now let Σ be the set of formulas of Montagovian syntax used to represent the premises of Schubert's steamroller as given in section 3 and let Φ be the formula to be proven. Our implementation of an inference procedure for the rule set M'

has been used to verify that:

$$\Sigma \cup \left\{ \begin{array}{l} (x_w \texttt{ wolf}), \texttt{ (focus-on } x_w\texttt{)}, \\ (x_f \texttt{ fox}), \texttt{ (focus-on } x_f\texttt{)}, \\ (x_b \texttt{ bird}), \texttt{ (focus-on } x_b\texttt{)} \end{array} \right\} \vdash_{M'} \Phi$$

This sequent expresses the English statement "to see that Φ follows from Σ, consider a wolf x_w, a fox x_f, and a bird x_b — the result is then obvious". Repeated use of the Socratic inference rules S3 and S5 can be used to eliminate all premises other than Σ, and hence derive the sequent $\Sigma \vdash \Phi$. A simple user interface to the Socratic proof system can be used to automatically apply sequent rules, such as S3 and S5, that remove extraneous premises. Given this user interface, the above sequent is a one line solution to Schubert's steamroller.

6 Discussion

We have constructed a complete proof system for a non-standard syntax for first order logic. This proof system has the simultaneous features that proofs are short and yet, if our conjectures are correct, the acceptability of a proof is quickly verifiable. The proofs in our system are so short that Schubert's steamroller can be proved in a single line, by far the shortest known proof in a proof system with polynomial time checkable proofs.

The conciseness of the proofs in our proof system appears to be due to the power (and conjectured locality) of the inference rules given in figure 1. This power appears to depend fundamentally on the use of a non-standard syntax to express the inference rules—just what aspect of the new syntax makes this added expression possible is unclear, but one relevant observation is that the quantifier free fragment of the new syntax can express many facts which require quantifiers in classical syntax (e.g. `(every man mammal)`). Our experience indicates that the decision procedure for the inference relation $\vDash_M$ immediately solves the vast majority of inference problems that can be stated in the fragment of Montagovian syntax that does not contain λ-quantifiers.[4] The statement of Schubert's steamroller in Montagovian syntax contains only a single λ-quantifier — a quantifier needed to represent the English anaphora "itself". Three instantiations of this quantifier are needed in the solution of Schubert's steamroller. Our one-line solution specifies the objects on which the quantifier is to be instantiated — the focus-on premises in the one-line solution control the use of the instantiation rules 34 and 35.

[4] We conjecture that validity in the λ-free fragment of our Montagovian syntax is decidable, although it is known that the inference rules in figure 1 are not complete for λ-free Montagovian formulas.

The inference relation defined by the inference rules in figure 1 appears not to have any definition in the classical syntax of first order logic. Thus, Montagovian syntax appears to play an essential role in the specification of the inference relation and therefore in the construction of the a Socratic sequent system with extremely concise proofs. Although this suggests that natural language syntax plays an important role in human reasoning, it seems sufficient to merely claim that aspects of natural language syntax can be used to build powerful inference algorithms.

References

[Bibel *et al.*, 1987] W. Bibel, R. Letz, and J. Schumann. Bottom-up enhancements of deductive systems. In *Proceedings of the Fourth International Conference on Artificial Intelligence and Information Control Systems of Robots*, pages 1–9. North-Holland, Amsterdam, Netherlands, October 1987.

[Cohn, 1984] A. G. Cohn. A note concerning the axiomatization of schubert's steamroller in many sorted logic. In *Alvey IKBS Inference Research Theme Workshop*, pages 14–21. Alvey Directorate, London, England, September 1984.

[Cohn, 1989] A. G. Cohn. Taxonomic reasoning with many-sorted logics. *Artificial Intelligence Review*, 3(2-3):89–128, 1989.

[Crawford and Kuipers, 1989] J. M. Crawford and Benjamin Kuipers. Towards a theory of access-limited logic for knowledge representation. In *First International Conference on Principles of Knowledge PUBLISHER = Morgan Kaufmann Publishers, Representation and Reasoning*, pages 67–78, 1989.

[Davies, 1988] N. Davies. Schubert's steamroller in a natural deduction theorem prover. In *Proceedings of Computer Society Specialist Group on Expert Systems*, pages 89–102. Cambridge University Press, Cambridge, UK, December 1988.

[McAllester and Givan, 1989] D. McAllester and R. Givan. Natural language syntax and first order inference. Memo 1176, MIT Artificial Intelligence Laboratory, October 1989. To Appear in AIJ.

[McAllester *et al.*, 1989] D. McAllester, R. Givan, and T. Fatima. Taxonomic syntax for first order inference. In *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, pages 289–300, 1989. To Appear in JACM.

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

[McAllester, 1990] D. McAllester. Automatic recognition of tractability in inference relations. Memo 1215, MIT Artificial Intelligence Laboratory, February 1990. To appear in JACM.

[Montague, 1973] Richard Montague. The proper treatment of quantification in ordinary english. In *Approaches to Natrual Language: Proceedings of the 1970 Stanford Workshop on Grammar and Semantics*. Reidel, 1973. Reprinted in: *Formal Philosophy: Selected Papers of Richard Montague*, ed. by R. H. Thomason, Yale University Press, 1974.

[Oppacher and Suen, 1986] F. Oppacher and E. Suen. Controlling deduction with proof condensation and heuristics. In *International Conference on Automated Deduction*, pages 384–93. Springer-Verlag, Berlin, Germany, July 1986.

[Stickel, 1985] Mark E. Stickel. Automated deduction by theory resolution. *Journal of Automated Reasoning*, 1:333–355, 1985.

[Walther, 1984] Christoph Walther. A mechanical solution of schubert's steamroller by many-sorted resolution. In *Proceedings of AAAI-84*, pages 330–334. Morgan Kaufmann Publishers, 1984.