Natural Language Syntax and First Order Inference

David McAllester and Robert Givan
MIT Artificial Intelligence Laboratory
545 Technology Square
Cambridge Mass. 02139
dam@ai.mit.edu

Abstract: We have argued elsewhere that first order inference can be made more efficient by using non-standard syntax for first order logic. In this paper we define a syntax for first order logic based on the structure of natural language under Montague semantics. We show that, for a certain fairly expressive fragment of this language, satisfiability is polynomial time decidable. The polynomial time decision procedure can be used as a subroutine in general purpose inference systems and seems to be more powerful than analogous procedures based on either classical or taxonomic syntax.

This paper appeared in Artificial Intelligence vol. 56, 1992. A postscript electronic source for this paper can be found in ftp.ai.mit.edu:/pub/dam/aij1.ps. A bibtex reference can be found in internet file ftp.ai.mit.edu:/pub/dam/dam.bib.

1 Introduction

This paper presents a new polynomial time procedure for automated inference. Although no polynomial time procedure can be complete for first order logic, polynomial time inference procedures can often be used as powerful subroutines in general purpose reasoning systems [7], [21], [17]. The procedure presented here is a significant strengthening of the procedure presented in [16], which is in turn a significant strengthening of the well known procedure for congruence closure [14], [9].

Both the procedure presented here and the one presented in [16] are defined by inference rules written in a nonstandard syntax for first order logic. Nonstandard syntax is essential to both the specification and to the implementation of these procedures. The procedure given in [16] uses a syntax based on taxonomic relations between expressions that denote sets. This procedure can be viewed as an extension of earlier work on knowledge representation languages, e.g., [3], [11], [4], [5]. Knowledge representation languages have traditionally been organized around taxonomic relationships between classes. AI researchers often express the intuition that such taxonomic representations facilitate inference. The technical results in [16] support this intuition. The relationship between the work described here and previous work on knowledge representation languages is discussed in more detail in section 5.

The inference procedure presented here is based on a new nonstandard syntax for first order logic which we call a *Montagovian syntax*. This syntax is based on aspects of natural language syntax under compositional semantics [1], [15], [18], [19], [10], [2]. In particular, our Montagovian syntax is centered around class expressions, i.e., expressions that denote sets. In the earlier taxonomic syntax we allowed for class symbols, such as the symbol `a-person`, and class expressions such as `(brother-of a-person)`. The expression `(brother-of a-person)` denotes the set of all individuals that are the brother of some person. In general, any monadic predicate symbol of classical syntax can be used as a class expression, and for any binary relation R, and class expression s, one can construct the class expression $(R\ s)$ which denotes the set of individuals which are related under R to an element of

s. In the new Montagovian syntax we write the class expression (R s) as (R (some s)) and we allow the construction of the (different) class expression (R (every s)). For example, if `loves` is a binary relation symbol, and `person` is a class symbol, then we have the class expressions `(loves (some person))` and `(loves (every person))`. The former class expression denotes the set of all individuals that love some person while the latter class expression denotes the set of all individuals that love all people. This expansion of the vocabulary of class expressions results in an inference procedure that is, in most respects, more powerful than the one based on taxonomic syntax given in [16]. A precise specification of the syntax and semantics of our Montagovian version of first order logic is presented below.

This paper is intended to provide technical support for the following two somewhat informal claims.

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

This paper makes no contribution to the traditional study of the syntax or semantics of natural language. We do not provide any new theory for predicting which strings of words are grammatical natural language sentences (the traditional study of syntax). Nor do we provide any new theory for assigning meaning to natural language utterances (the traditional study of semantics). There is a large literature on these topics with theories far more sophisticated than the ones used here. This paper addresses a different topic, the relationship between natural language syntax and efficient inference techniques. In studying the relation between syntax and inference we have focused on only the most fundamental properties of natural language.

2 A Montagovian Syntax for First Order Logic

Our Montagovian syntax is a syntactic variant of first order predicate calculus — every Montagovian formula can be translated to a classical formula, and every classical formula can be translated to a Montagovian formula. However, the quantifier-free fragment of Montagovian syntax is more expressive than the quantifier-free fragment of either classical syntax or our earlier taxonomic syntax. In spite of increased expressive power, the quantifier-free fragment of Montagovian syntax retains most of the nice computational properties of the quantifier-free fragment of classical syntax.

Classical syntax involves terms and formulas. In both taxonomic and Montagovian syntax terms are replaced by class expressions where each class expression denotes a set. The syntax of our Montagovian language is defined as follows.

- A *class expression* is one of the following.
  - A variable or constant symbol.
  - A monadic predicate symbol.
  - An expression of the form (R `(some` s`))` or (R `(every` s`))` where R is a binary relation symbol and s is a class expression.
  - An expression of the form $(\lambda\ x\ \Phi(x))$ where x is a variable and $\Phi(x)$ is a formula.

  .

- A *formula* is one of the following.
  - An expression of the form `(every` s w`)` or `(some` s w`)` where s and w are class expressions. Expressions of this type are called *atomic formulas.*
  - A Boolean combination of formulas. Atomic formulas and negations of atomic formulas are called *literals.*

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.

Constant symbols and variables are taken to denote singleton sets. 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.

Unlike Montague, we make no distinction between nouns and verbs. As a result, there are formulas of our Montagovian syntax that do not correspond to grammatical sentences. For example, consider the formulas `(every dog mammal)` and `(every (loves John) (loves Mary))`.

The formal semantics for our Montagovian syntax is a (drastic) simplification of Montague's original semantics for English. 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. Any binary relation R can be transformed to a function R' from elements to sets such that y is an element of $R'(x)$ if only if the pair $<y,\ x>$ is in the relation R. We adopt a superficial modification of the definition of a first order structure so that a binary relation symbol denotes a function from elements to sets rather than a relation. Under our definition, a first order model interprets each constant symbol as an element of its domain, each class symbol as a subset of its domain, and each relation symbol as a function from domain elements to subsets of the domain. If the interpretation of a relation symbol R is clear from context, we will often write $R(d)$ to denote the set that is the result of applying (the value of) R to the domain element d.

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 s is a class expression then $\mathcal{V}(s, \mathcal{M}, \rho)$ 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**. The semantic evaluation function $\mathcal{V}$ is defined by structural induction on expressions as follows.

- If P is a class symbol then $\mathcal{V}(P,\ \mathcal{M},\ \rho)$ is the set $\mathcal{M}(P)$.
- If c is a constant then $\mathcal{V}(c,\ \mathcal{M},\ \rho)$ is the singleton set $\{\mathcal{M}(c)\}$.
- If x is a variable then $\mathcal{V}(x,\ \mathcal{M},\ \rho)$ is the singleton set $\{\rho(x)\}$.
- $\mathcal{V}((R\ \texttt{(every}\ s\texttt{))}, \mathcal{M}, \rho)$ is the set of all d such that, for every d' in $\mathcal{V}(s, \mathcal{M}, \rho)$, d is an element of $R(d')$. (Consider the class expression `(loves (every child))`.)
- $\mathcal{V}((R\ \texttt{(some}\ s\texttt{))}, \mathcal{M}, \rho)$ is the set of all d such that there exists an element d' in $\mathcal{V}(s, \mathcal{M}, \rho)$ such that d is in the set $R(d')$. (Consider the class expression `(loves (some child))`.)
- $\mathcal{V}((\lambda\ x\ \ \Phi(x)),\ \mathcal{M},\ \rho)$ 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.
- $\mathcal{V}(\texttt{(every}\ s\ \ t\texttt{)},\ \mathcal{M},\ \rho)$ is **T** if $\mathcal{V}(s,\ \mathcal{M},\ \rho)$ is a subset of $\mathcal{V}(t,\ \mathcal{M},\ \rho)$.
- $\mathcal{V}(\texttt{(some}\ s\ \ t\texttt{)},\ \mathcal{M},\ \rho)$ is **T** if $\mathcal{V}(s,\ \mathcal{M},\ \rho)$ has a non-empty intersection with $\mathcal{V}(t,\ \mathcal{M},\ \rho)$.
- Boolean combinations of atomic formulas have their standard meaning.

Although explicit quantification has not been allowed in formulas, the language is rich enough to express quantified formulas. Let `THING` be an abbreviation for the class expression $(\lambda\ x\ \texttt{(every}\ x\ \ x\texttt{))}$. Note that in any first order model `THING` denotes the universal set, i.e., the entire domain of the model. The formula $\forall x\ \ \Phi(x)$ can be taken to be an abbreviation for (every thing (λ x $\Phi(x)$)). Similarly, the formula $\exists x\ \Phi(x)$ can be treated as an abbreviation for (some thing (λ x $\Phi(x)$)). It is fairly easy to show that any formula in our Montagovian language can be faithfully translated into classical first order logic, and that any formula of classical first order logic can be faithfully translated into our Montagovian language.

Montague gives an independent semantic value to noun phrases such as (some person) and (every person) where these expressions denote functions from sets to truth values. A formula such as (every s w) can then be analyzed as ((every s) w), i.e., the function (every s) applied to the argument w. Montague also gives a compositional meaning to class expressions of the form (R (every s)) in terms of the independent meaning of the expression (every s). Although we have no particular objection to Montague's analysis, we have decided to simplify the exposition of our semantics by avoiding any independent meaning for expressions of the form (every s).

3 Literal Satisfiability

Since Montagovian syntax is expressively equivalent to full first order logic, it is impossible to construct a procedure which can always determine whether a given formula is satisfiable. However, it is possible to define a fragment of the language for which satisfiability is polynomial time decidable. In constructing a decision procedure we consider only "quantifier-free" formulas. A formula of Montagovian syntax is called quantifier-free if it does not contain any λ-classes. For example, the formula (every woman (likes (some man))) is considered to be quantifier free, while the formula (every man (λ x (x (likes x)))) involves a λ-class and is therefore not considered to be quantifier-free. The quantifier-free fragment of the language has no bound variables and a purely compositional semantics. We view bound variables and noncompositional semantics as the essence of quantification. This notion of quantifier-freeness is motivated, at least in part, by an analogy between the quantifier-free fragment of our Montagovian syntax and the quantifier-free fragment of classical first order logic. The decision procedure for the quantifier-free Montagovian syntax is similar to the decision procedure for the quantifier-free fragment of classical syntax.

The quantifier-free fragment of our Montagovian language roughly corresponds to simple subject-verb-object sentences. For example, `(every dog (ate (some bone)))` or

$$\texttt{(every (child-of Sally) (married (some (child-of John))))}.$$

Sentences that involve traces or anaphora can usually not be expressed in the quantifier-free fragment of our Montagovian language. For example, the sentence "every man likes himself" involves the anaphora "himself". Translating this into a Montagovian formula introduces a quantifier — (every man (λ x (x (likes x)))). As another example, consider the sentence "Mary read some book John bought". Most linguists would agree that the word "bought" in this sentence has an invisible argument called a trace. The following translation of this sentence into a Montagovian formula involves a quantifier.[1]

$$\texttt{(Mary (read (some (}\lambda\ x\ (x\ \texttt{book)}\ \wedge\ \texttt{(John (bought}\ x\texttt{))))))}$$

It is difficult to precisely characterize the expressive power of the quantifier-free fragment of Montagovian syntax. The quantifier-free Montagovian formula `(every dog (likes (every person)))` can not be expressed in either the quantifier-free fragments of classical or taxonomic taxonomic syntax.[2] However, because classical and taxonomic syntax allow function symbols and predicates of more than two arguments, the quantifier-free fragments of these languages can express statements that are not expressible in quantifier free Montagovian syntax. If we restrict classical and taxonomic syntax to constant symbols and predicates of no more than two arguments, then quantifier-free Montagovian syntax is strictly more expressive than quantifier-free classical or taxonomic syntax. It seems likely that the basic results of this section can be extended to handle function symbols and predicates of more than two arguments, although the proofs of theorems analogous to those given here are

[1] A more satisfying translation of the second sentence would be an expression of the form (Mary (read (some (book (λ x (John (bought x))))))) where the λ-class is treated as an intersectional adjectival phrase. Unfortunately, our simple Montagovian syntax does not allow for direct intersection of class expressions.

[2] The formula `(every dog (likes (some person)))` can be expressed in quantifier-free taxonomic syntax but not in quantifier-free classical syntax.

likely to be much more complex.[3] The quantifier-free fragment of Montagovian syntax is expressively incomparable with previously studied knowledge representation languages such as those discussed in section 5.

Although satisfiability is undecidable for unrestricted first order formulas, satisfiability is decidable for quantifier-free Montagovian syntax. Since the quantifier free fragment of Montagovian syntax includes arbitrary Boolean formulas, determining satisfiability is NP-hard and we can not expect to find a polynomial time decision procedure. A more tractable problem is the *literal satisfiability problem.* This is the problem of determining if a given set of literals[4] is satisfiable. In classical syntax, and in taxonomic syntax, the literal satisfiability problems are polynomial time decidable. In Montagovian syntax the literal satisfiability problem is NP-complete. A proof of the NP-hardness of the literal satisfiability problem for Montagovian syntax is given in appendix I. The NP hardness of the Montagovian literal satisfiability problem arises from the fact that, for a given class expression appearing in the input, we may not know whether or not that exspression denotes the empty set. If, for each class expression, we know whether or not that expression denotes the empty set then the literal satisfiability problem becomes polynomial time decidable.

To simplify the presentation of the remainder of this paper we use the notation $\exists s$ where s is a class expression as an abbreviation for the formula `(some` s s`)`. Formulas of the form $\exists s$ express the statement that there exist elements of the set denoted by s, i.e., s does not denote the empty set.

> **Definition:** We say that a set of formulas Σ *determines existentials* if, for every class expression s that appears in any formula in Σ, Σ contains either the formula $\exists s$ or the formula $\neg\exists s$.
>
> **Montagovian Literal Satisfiability Theorem:** The satisfiability of a set of quantifier-free Montagovian literals that determines existentials is polynomial time decidable.

[3]The use of function symbols in taxonomic syntax greatly increases the complexity of the completeness theorem for the decision procedure for the quantifier-free fragment.

[4]As defined above for Montagovian syntax, a literal is either an atomic formula or the negation of an atomic formula where an atomic formula is any formula of the form `(every` s w`)` or `(some` s w`)`.

The above theorem implies that one can determine whether an arbitrary set Σ of quantifier-free Montagovian literals is satisfiable by searching for a superset of Σ that determines existentials and is satisfiable. If there are n class expressions in Σ then there are at most 2^n extensions of Σ that need to be searched. This also implies that the satisfiability problem for quantifier-free Montagovian formulas is in the complexity class NP — a quantifier free formula Φ is satisfiable if and only if there exists a truth assignment to the atomic formulas in Φ, and a truth assignment to existential statements about the class expressions in Φ, such that the truth assignment is satisfiable according to the above procedure and satisfies the Boolean part of Φ.

4 The Decision Procedure

We start by transforming the given set of literals Σ into an equi-satisfiable set Σ' which contains no literals of the form (some s t) where s and t are distinct class expressions. We will call such literals *positive intersection literals*. This transformation can be achieved by simply replacing any positive intersection literal (some s t) with the three literals (every w s), (every w t) and $\exists w$ where w is a new class symbol. Any model of Σ' is also a model of Σ, and any model of Σ yields a model of Σ'. For the remainder of this section we assume that Σ contains no positive intersection literals. Negative intersection literals, i.e. literals of the form $\neg$(some s t), may still be present.

The literal satisfiability procedure is based on the inference rules given in figure 1. These rules introduce a new formula, (at-most-one s) where s is a class expression. The formula (at-most-one s) is true just in case the set denoted by s contains at most one member. Inference rule 15 allows for the derivation of positive intersection formulas — although we can assume that Σ does not contain positive intersection formulas it is convenient to allow such formulas to be inferred. By assuming that Σ does not contain positive intersection formulas we can ensure that whenever we can infer (some s t) there exists some expression w such that we can infer $\exists w$, (every w s), and (every w t) We now introduce a restricted inference relation $\vDash$.

Definition: We write $\Sigma \vDash \Phi$ if Φ can be proven from Σ using

(1) $$\frac{\texttt{(every } s\ t\texttt{)}}{\texttt{(every (R (some } s\texttt{)) (R (some } t\texttt{)))}}$$

(2) $$\frac{\texttt{(every } s\ t\texttt{)}}{\texttt{(every (R (every } t\texttt{)) (R (every } s\texttt{)))}}$$

(3) $$\frac{\texttt{(every } r\ s\texttt{)},\ \texttt{(every } s\ t\texttt{)}}{\texttt{(every } r\ t\texttt{)}}$$

(4) $\texttt{(every } t\ t\texttt{)}$

(5) $\exists c$

(6) $\texttt{(at-most-one } c\texttt{)}$

(7) $$\frac{\exists\texttt{(R (some } s\texttt{))}}{\exists s}$$

(8) $$\frac{\exists r,\ \texttt{(every } r\ t\texttt{)}}{\exists t}$$

(9) $$\frac{\texttt{(at-most-one } t\texttt{)},\ \texttt{(every } r\ t\texttt{)}}{\texttt{(at-most-one } r\texttt{)}}$$

(10) $$\frac{\neg\texttt{(every } r\ t\texttt{)}}{\exists r}$$

(11) $$\frac{\begin{array}{l}\exists s\\ \texttt{(at-most-one } t\texttt{)}\\ \texttt{(every } s\ t\texttt{)}\end{array}}{\texttt{(every } t\ s\texttt{)}}$$

(12) $$\frac{\begin{array}{l}\exists r\\ \texttt{(every } r\ s\texttt{)}\\ \texttt{(every } r\ t\texttt{)}\end{array}}{\texttt{(every (R (every } s\texttt{)) (R (some } t\texttt{)))}}$$

(13) $$\frac{\neg\exists s}{\texttt{(every } t\ \texttt{(R (every } s\texttt{)))}}$$

(14) $$\frac{\texttt{(at-most-one } t\texttt{)},\ \texttt{(every } s\ t\texttt{)}}{\texttt{(every (R (some } s\texttt{)) (R (every } t\texttt{)))}}$$

(15) $$\frac{\begin{array}{l}\texttt{(every } r\ s\texttt{)}\\ \texttt{(every } r\ t\texttt{)}\\ \exists r\end{array}}{\texttt{(some } s\ t\texttt{)}}$$

(16) $$\frac{\begin{array}{l}\mathbf{\Psi}\\ \neg\mathbf{\Psi}\end{array}}{\mathbf{F}}$$

Figure 1: The inference rules for quantifier-free literals. In these rules the letters r, s, and t range over class expressions, c ranges over constant symbols, and R ranges over relation symbols.

the rules in figure 1 *such that every class expression appearing in the proof appears in* Σ.

The definition of the relation $\vdash$ ensures that to determine whether $\Sigma \vdash \Phi$ we need only consider formulas all of whose class expressions appear in Σ. For a given finite set Σ there are only finitely many class expressions that appear in Σ — the number of class expressions can grow at most linearly in the written length of Σ. The inference rules have the property that they can only be used to infer formulas of the form `(every` s w`)`, `(some` s w`)`, or `(at-most-one` s`)` (recall that $\exists s$ is actually an abbreviation for `(some` s s`)`). If we only consider formulas whose class expressions appear in Σ, then there are at most order $|\Sigma|^2$ such formulas. This implies that by simply enumerating all derivable formulas one can determine, in polynomial time in the size of Σ, determine whether or not $\Sigma \vdash \Phi$.

Satisfiability Completeness Lemma: If Σ is a set of quantifier-free Montagovian literals that determines existentials, then Σ is satisfiable if and only if $\Sigma \not\vdash \mathbf{F}$.

Given that one can determine in polynomial time whether $\Sigma \vdash \mathbf{F}$, the above satisfiability completeness lemma immediately implies the Montagovian literal satisfiability theorem of the preceding section. The proof of the above completeness lemma is given in appendix II.

5 Other Knowledge Representation Languages

Our Montagovian syntax for first order logic is related to a large family of knowledge representation languages known as *concept languages* or *frame description languages* (FDLs) [6], [20], [22], [8].

Each FDL is similar to our Montagovian syntax in that it provides a simple recursive definition of a particular set of class expressions built from constant, predicate, and relation symbols.[5] The class expressions of a particular FDL can be considerably different from the class expressions of our Montagovian syntax. For example, all FDLs discussed in the knowledge representation literature include intersection operations on class expressions — given any two class expressions s and w the class expression $\mathbf{AND}(s,\ w)$ denotes the intersection of the sets denoted by s and w. A Montagovian syntax that includes a class intersection operation is described in [12].

All languages in the knowledge representation literature also include class expressions of the form $\forall R.C$ where R is a relation symbol and C is a class expression. An object x is a member of the class expression $\forall R.C$ if, for every y such that the relation R holds between x and y, the individual y is in the set denoted by C. For example, the class expression $\forall$`child-of.human` denotes the set of all individuals x such that every child of x is human. The statement that every child of a human is human can be expressed as the formula

`(every human (`$\forall$` child-of . human)).`

Intuitively, this formula states that every human has the property that every child of that human is human. This same statement can be expressed in our Montagovian syntax (or in our earlier taxonomic syntax) with the formula

(every (child-of (some human)) human).

It is important to note that class expressions of the form $\forall R.C$ are quite different from class expressions of the form `(`R `(every` C`))`. For example, $\forall$`loves.human` is the class of individuals that love *only* humans, while `(loves (every human))` is the class of individuals that love all humans (and possibly other things as well).

Class expressions of the form $\forall R.C$ are not expressible in our Montagovian syntax. In particular, there appears to be no way to express the formula `(every (`$\forall R.C$`) W)` in Montagovian syntax. Conversely, there is no way that class expressions of the form $\forall R.C$ can be used to express the class expression

[5] Within the knowledge representation literature an FDL is not viewed as an alternative syntax for full first order logic. Rather, the formulas of an FDL are restricted to include only subset relations between restricted types of class expressions. These languages are less expressive than full first order logic.

$(R$ `(some` C`))`. In particular, there appears to be no way of translating the formula `(every W (`R `(some` C`)))` into a formula involving class expressions of the form $\forall R.C$. There does not appear to be any simple relationship between the expressive power of Montagovian syntax and previously studied FDLs.

6 Montagovian vs. Classical Syntax

We have presented a polynomial time inference procedure defined by a set of inference rules stated in a Montagovian syntax for first order logic. These inference rules cannot be stated in classical syntax without resorting to higher order unification. For example, consider inference rule 1.

$$\frac{\texttt{(every } s\ t\texttt{)}}{\texttt{(every (R (some } s\texttt{)) (R (some } t\texttt{)))}}$$

This inference rule might be written in classical syntax as follows.

$$\frac{\forall x\ P(x) \rightarrow Q(x)}{\forall y\ (\exists x\ P(x) \wedge R(x,\ y)) \rightarrow (\exists x Q(x) \wedge R(x,\ y))}$$

Note, however, that to use the rule in classical syntax the predicates P and Q must be treated as variables that can bind to arbitrary predicates. Theorem provers that instantiate predicate variables have traditionally used higher order unification [13]. Consider applying the Montagovian version of inference rule 1 to the Montagovian formula

`(every (child-of (some bird)) (friend-of (every bird-watcher)))`.

This formula states that any child of a bird is a friend of any bird watcher. An application of inference rule 1 allows us to conclude

```
(every (owner-of (some (child-of (some bird))))
       (owner-of (some (friend-of (every bird-watcher))))).
```

This formula says that anyone who owns the child of a bird also owns a friend of every bird watcher. In Montagovian syntax inference rule 1 can be applied using simple (classical) unification to bind the variables s and t of the inference rule to the expressions `(child-of (some bird))` and `(friend-of (every bird-watcher))` respectively. Now consider the same inference in classical syntax. The premise can be stated as follows.

$$\forall x\, (\exists y\ \texttt{bird}(y) \wedge \texttt{child-of}(y\ x)) \rightarrow (\forall y\ \texttt{bird-watcher}(y) \rightarrow \texttt{friend-of}(y\ x))$$

To apply the classical syntax version of the inference rule one must bind the predicate variable P to the λ-predicate

$$\lambda x\ \ \exists y\ \texttt{bird}(y) \wedge \texttt{child-of}(y\ x)$$

and bind Q to the λ-predicate

$$\lambda x\ \ \forall y\ \texttt{bird-watcher}(y) \rightarrow \texttt{child-of}(y\ x).$$

Given this binding of P and Q in the classical syntax rule, the conclusion of the rule must be translated back into classical syntax by β-reducing applications of these λ-predicates. Applying inference rule 1 in classical syntax requires both higher order unification and β-reduction.

The inference procedure described in the previous section has a simple termination condition. Inference is restricted so that all class expressions mentioned by derived formulas must already appear in the input set of literals. This restriction implies that only a finite (polynomial) number of formulas can be derived and hence the inference process must terminate. If the inference rules were expressed in classical rather than Montagovian syntax the termination condition would be much more difficult to state. A similar comparison can be made between classical syntax and other knowledge representation languages such as the FDLs discussed earlier.

7 Conclusions

We have argued that the effectiveness of inference is coupled to the selection of the syntax in which formulas are expressed. If such a coupling does indeed exist then one can speak informally of "effective syntax" — a syntax is effective to the extent that inference processes defined in that syntax can be made effective. Classical syntax appears to be particularly ineffective.

If one accepts the proposition that the effectiveness of inference is coupled to the syntax in which formulas are expressed then it is perhaps not too surprising that natural language is a source of effective syntax. The Montagovian syntax presented here is, of course, only distantly related to the much richer and more complex syntax of actual natural languages. We hope that natural language syntax will continue to be an inspiration for the construction of yet more effective formal languages.

8 Appendix I: The Montagovian Literal Satisfiability Problem

In this appendix we show that determining the satisfiability of a set of Montagovian literals (that need not determine existentials) is NP-hard. The proof of NP-hardness is by reduction of a special case of monotone 3-SAT. More specifically, we start with a set of propositional clauses where each clause either contains three negative literals or two positive literals. We leave it to the reader to verify that satisfiability of an arbitrary 3SAT problem can be reduced to satisfiability of this special case. For each proposition symbol P in our restricted 3SAT problem we introduce a class symbol P' where the truth of P will correspond to the existence of elements of the set denoted by P'. We reduce the set of clauses to a set of Montagovian literals as follows:

For each clause of the form $P \vee Q$ we add the literal

$$\texttt{(every (R (every } P'\texttt{)) (G (some } Q'\texttt{)))}$$

where R and G are new relation symbols. Any model of this literal must satisfy either $\exists P'$ or $\exists Q'$ — if both P' and Q' are assigned the empty set then `(R (every` P'`))` denotes the universal set, which must be non-empty, while `(G (some` Q'`))` denotes the empty set. Conversely, for any interpretation of the class symbols P' and Q' as sets, if at least one of the two sets is non-empty then one can ensure that the above literal is satisfied by making `R` the empty relation and `G` the universal relation.

Now for any class symbols s, t and w we define $[\exists s \rightarrow$ `(every` t w`)`$]$ to be the two literals `(every` t `(H (every` s`)))` and `(every (H (some` s`))` w`)`, where `H` is a new relation symbol specific to this constraint. Any model of these literals must satisfy the constraint that if s denotes a non-empty set then the set denoted by t must be a subset of the set denoted by w. Conversely, for any assignment of sets to the class symbols s, t, and w satisfying the desired constraint, there exists an interpretation of `H` satisfying the above literals — if s is assigned the empty set then the above literals are satisfied by any interpretation of `H`; if s is non-empty then t must denote a subset of the set denoted by w and the above literals are satisfied by interpreting `H` as the relation that maps every domain element to the set denoted by w.

Finally, for any clause of the form $\neg P \vee \neg Q \vee \neg U$ we add the literals that constitute the constraints

$$[\exists P' \rightarrow \texttt{(every } s \ w_1\texttt{)}]$$

$$[\exists Q' \rightarrow \texttt{(every } w_1 \ w_2\texttt{)}]$$

$$[\exists U' \rightarrow \texttt{(every } w_2 \ t\texttt{)}]$$

$$\neg\texttt{(every } s \ t\texttt{)}$$

where s, t, w_1, and w_2 are new class symbols specific to this clause. Any model of the above formulas must assign one of the class symbols P', Q', or R' the empty set. Conversely, for any interpretation of P', Q', and R' as sets at least one of which is empty, there exist interpretations of s, t, w_1 and w_2 as sets that satisfy the above constraints.

We leave it to the reader to verify that the set of literals generated by this reduction is satisfiable if and only if the original restricted 3SAT problem is satisfiable.

9 Appendix II: Proof of the Completeness Lemma

This appendix contains a proof of the completeness lemma, i.e., that if Σ is a set of Montagovian literals that determines existentials then Σ is satisfiable if and only if $\Sigma \not\vdash \mathbf{F}$. This implies that if Σ determines existentials then the satisfiability of Σ can be determined in polynomial time. This second statement implies that the satisfiability of quantifier-free Montagovian formulas is in the complexity class NP, and hence is NP-complete.

Suppose that Σ is a set of quantifier-free Montagovian literals that determines existentials and contains no positive intersection literals. We must show that Σ is satisfiable if and only if $\Sigma \not\vdash \mathbf{F}$. If $\Sigma \vdash \mathbf{F}$ then the soundness of the individual inference rules guarantees that Σ is unsatisfiable. If $\Sigma \not\vdash \mathbf{F}$ we must show that there exists a model of Σ. To simplify the presentation we introduce the notation $\Sigma \vdash s = w$ to indicate that we have both $\Sigma \vdash$ (every s w) and $\Sigma \vdash$ (every w s).

Assume that $\Sigma \not\vdash \mathbf{F}$. We will construct a formal model of Σ where the elements of the domain are constructed from the class expressions that appear in Σ. Given that Σ is quantifier-free we can replace any variable in Σ by a constant symbol without affecting satisfiability. We can therefore assume without loss of generality that no there are no variables in Σ. The definition of the semantic domain of the model involves two complications. First, we must construct equivalence classes of class expressions. If $\Sigma \vdash s = t$, and $\Sigma \vdash$ (at-most-one s), then s and t must denote the same singleton set. In this case the single object in the set denoted by s is (essentially) the equivalence class of all class expressions that are provably equal to s. The second complication involves the need for both "minimal" and "maximal" elements of the set denoted by a class expresison. If $\Sigma \not\vdash$ (every (R (some s)) t) then we will guarantee that the set denoted by s contains a maximal element d such that R(d) is a "large" set, and in particular, that R(d) includes something not in the set denoted by t. Let $|s|$ be the equivalence class of the class expression s. We use the notation "some-$|s|$" to denoted the pair of the symbol "some" and the class $|s|$. The pair "some-$|s|$" will be the desired maximal element of the class denoted by s. If $\Sigma \not\vdash$ (every t (R (every s)))

then we will guarantee that s contains some minimal element d such that $\mathtt{R}(d)$ denotes a small set, and in particular, that the set denoted by t contains something not in $\mathtt{R}(d)$. By analogy with maximal elements, we use the notation "every-$|s|$" to denote the formal object that will be the minimal element of the set denoted by s.

We say that a class expression s is a *domain expression* if it appears in Σ and $\Sigma \vdash \exists s$. If s is a domain expression then we use the notation $|s|$ to denote the set of all domain expressions t such that $\Sigma \vdash s = t$. Inference rules 3 and 4 guarantee that the sets of the form $|s|$ form a partition of the domain expressions into equivalence classes. The semantic domain D of our model will consist of minimal elements "every-$|s|$" and maximal elements "some-$|s|$" where $\Sigma \vdash \exists s$, i.e., s is a domain expression. If $\Sigma \vdash$ (at-most-one s) then only the minimal element "every-$|s|$" will be included in the domain. Inference rule 9 guarantees that if $\Sigma \vdash$ (at-most-one s) and $\Sigma \vdash s = t$ then $\Sigma \vdash$ (at-most-one t). This implies that the choice of whether or not to include the domain element "some-$|s|$" in the semantic domain is independent of the choice of the representative s of the class $|s|$.

Given this semantic domain D, we must define an interpretation for the class symbols and relation symbols in Σ such that each literal of Σ is satisfied. The model we construct will satisfy a certain *denotation invariant* — the set denoted by a class expression s that appears in Σ will consist of all domain elements "some-$|t|$" and "every-$|t|$" such that $\Sigma \vdash$ (every t s). We define the interpretation of constant symbols, class symbols, and relation symbols using this desired denotation invariant as a guide. We use the notation "Q-$|s|$" to mean either the object "some-$|s|$" or the object "every-$|s|$". The denotation of a class symbol P is defined to be the set of all domain members of the form "Q-$|s|$" such that $\Sigma \vdash$ (every s P). This definition immediately guarantees the denotation invariant for class symbols.

We define the denotation of a constant symbol c that appears in Σ to be the domain member "every-$|c|$". Inference rules 5, 6, 9 and 11 guarantee that the denotation invariant holds for constant symbols. We interpret each constant symbol that does not appear in Σ as an arbitrary element of the semantic domain.

We will now define the interpretation of relation symbols. To define the function denoted by a relation symbol `R` we need to define the set `R("`Q`-`$|s|$`")` for any domain element `"`Q`-`$|s|$`"`. Intuitively, the set `R("`Q`-`$|s|$`")`, where Q is either `some` or `every`, should be the set of domain members `"`Q'`-`$|t|$`"` such that $\Sigma \mathrel{\vdash\!\!\mid}$ `(every t (R (Q s)))`. This intuitive definition fails because the class expression `(R (Q s))` need not appear in Σ. To remedy this situation we define a new relation $\mathrel{\vdash\!\!\circ}$.

- We write $\Sigma \mathrel{\vdash\!\!\circ}$ `(every t (R (some s)))` if any one of the following conditions hold:
  - For some `(R (some w))` appearing in Σ, $\Sigma \mathrel{\vdash\!\!\mid}$ `(every w s)` and $\Sigma \mathrel{\vdash\!\!\mid}$ `(every t (R (some w)))`.
  - For some `(R (every w))` appearing in Σ, $\Sigma \mathrel{\vdash\!\!\mid}$ `(some s w)` and $\Sigma \mathrel{\vdash\!\!\mid}$ `(every t (R (every w)))`.
- We write $\Sigma \mathrel{\vdash\!\!\circ}$ `(every t (R (every s)))` if any one of the following conditions hold:
  - For some `(R (every w))` appearing in Σ, $\Sigma \mathrel{\vdash\!\!\mid}$ `(every s w)` and $\Sigma \mathrel{\vdash\!\!\mid}$ `(every t (R (every w)))`.
  - For some `(R (some w))` appearing in Σ, $\Sigma \mathrel{\vdash\!\!\mid}$ `(at-most-one w)`, $\Sigma \mathrel{\vdash\!\!\mid} w = s$ and $\Sigma \mathrel{\vdash\!\!\mid}$ `(every t (R (some w)))`.

If $\Sigma \mathrel{\vdash\!\!\mid}$ `(every t (R (Q s)))` then $\Sigma \mathrel{\vdash\!\!\circ}$ `(every t (R (Q s)))`. Conversely, if `(R (Q s))` appears in Σ, and $\Sigma \mathrel{\vdash\!\!\circ}$ `(every t (R (Q s)))` then $\Sigma \mathrel{\vdash\!\!\mid}$ `(every t (R (Q s)))`. The difference between the two relations is restricted to expressions of the form `(R (Q s))` that do not appear in Σ. The reader can also check that if $\Sigma \mathrel{\vdash\!\!\mid} \exists s$ and $\Sigma \mathrel{\vdash\!\!\circ}$ `(every t (R (every s)))` then $\Sigma \mathrel{\vdash\!\!\circ}$ `(every t (R (some s)))`.

We now define the set `R("`Q`-`$|s|$`")` to be the set of all domain elements `"some-`$|t|$`"` and `"every-`$|t|$`"` such that $\Sigma \mathrel{\vdash\!\!\circ}$ `(every t (R (Q s)))`. We must check that this definition is well formed, i.e., that the definition is independent of the choice of s and t used as the representatives of the equivalence classes $|s|$ and $|t|$. Fortunately, the transitivity of the subset relation guarantees that if t' is equivalent to t and s' is equivalent to s then $\Sigma \mathrel{\vdash\!\!\circ}$ `(every t' (R (Q s')))` if and only if $\Sigma \mathrel{\vdash\!\!\circ}$ `(every t (R (Q s)))`.

This completes the definition of a first order structure — we have defined a semantic domain and assigned an appropriate meaning to all constant symbols, class symbols, and relation symbols. We will now prove that every class expression that appears in Σ satisfies the desired denotation invariant.

> **Denotation Invariant:** For any class expression s that appears in Σ, the denotation of s equals the set of domain elements "Q-$|t|$" such that $\Sigma \vdash$ (every t s).

We prove this invariant by induction on the structure of class expressions. Every class expression appearing in Σ is either a class symbol, a constant symbol, or an expression of the form (R (Q s)) for some relation symbol R, specifier Q, and class expression s. We have already argued that the denotation invariant holds for class symbols and constant symbols. Now we assume that s satisfies the denotation invariant and consider an expression in Σ of the form (R (Q s)). It now suffices to show that (R (Q s)) satisfies the denotation invariant, i.e., the set denoted by (R (Q s)) is the set of domain elements "Q'-$|t|$" such that $\Sigma \vdash$ (every t (R (Q s))). We consider four cases corresponding to whether Q is "some" or "every" and to which direction of the inclusion we are trying to show.

First we consider expressions of the form (R (some s)). Let "Q-$|t|$" be an element of the set denoted by (R (some s)). We must show that $\Sigma \vdash$ (every t (R (some s))). Since "Q-$|t|$" is in the set denoted by (R (some s)) there must be some element "Q'-$|s'|$" in the set denoted by s such that the set R("Q'-$|s'|$") contains "Q-$|t|$". By the induction hypothesis we must have $\Sigma \vdash$ (every s' s). By the definition of the meaning of R, we must have $\Sigma \vdash\!\circ$ (every t (R (Q' s'))). Since s' is a domain expression we must have $\Sigma \vdash \exists s'$. As noted above, the definition of $\vdash\!\circ$ implies that if $\Sigma \vdash \exists s'$ and $\Sigma \vdash\!\circ$ (every t (R (every s'))) then $\Sigma \vdash\!\circ$ (every t (R (some s'))). So $\Sigma \vdash\!\circ$ (every t (R (some s'))) (it is possible that $\Sigma \vdash\!\circ$ (every t (R (some s'))) even if "some-$|s'|$" is not a domain member.) By the definition of of $\vdash\!\circ$ there must exist some expression (R (Q'' w)) that appears in Σ such that $\Sigma \vdash$ (every t (R (Q'' w))) and such that (R (Q'' w)) satisfies one of the two ways of establishing $\Sigma \vdash\!\circ$ (every t (R (some s'))). Let (R (Q'' w)) be an expression that satisfies one of these two cases. We leave it to the reader to verify that in each case the expression (R $(Q''\ w)$) ensures that $\Sigma \vdash\!\circ$ (every t (R (some s))) and thus that $\Sigma \vDash$ (every t (R (some s))).

Now suppose that $\Sigma \vDash$ (every t (R (some s))). We must show that domain elements of the form "Q-$|t|$" are members of the set denoted by (R (some s)). Since t is a domain expression we have $\Sigma \vDash \exists t$. Inference rules 7 and 8 now guarantee that $\Sigma \vDash \exists s$. Now suppose that $\Sigma \vDash$ (at-most-one s). In that case the definition of $\vdash\!\circ$ ensures that $\Sigma \vdash\!\circ$ (every t (R (every s))). Since s satisfies the denotation invariant, and $\Sigma \vDash \exists s$, the element "every-$|s|$" must be in the set denoted by s. Finally, since $\Sigma \vdash\!\circ$ (every t (R (every s))), we have that the set R("every-$|s|$") contains "Q-$|t|$" and thus "Q-$|t|$" is in the set denoted by (R (some s)). Now suppose that $\Sigma \nvDash$ (at-most-one s). In this case the fact that $\Sigma \vDash \exists s$ and the denotation invariant for s guarantee that the set denoted by s includes the element "some-$|s|$". But the fact that $\Sigma \vDash$ (every t (R (some s))) immediately implies that "Q-$|t|$" is in the set R("some-$|s|$") and thus "Q-$|t|$" is in the set denoted by (R (some s)).

Now we consider expressions of the form (R (every s)). Let "Q-$|t|$" be an element of the the set denoted by (R (every s)). We must show that $\Sigma \vDash$ (every t (R (every s))). Suppose that $\Sigma \nvDash \exists s$. Since Σ determines existentials, we must have $\Sigma \vDash \neg\exists s$. In this case inference rule 13 guarantees that $\Sigma \vDash$ (every t (R (every s))). Now suppose $\Sigma \vDash \exists s$. In this case the denotation invariant, and inference rule 4, guarantees that the set denoted by s contains the element "every-$|s|$". Since "Q-$|t|$" is in the set denoted by (R (every s)), we must have that "Q-$|t|$" is in the set R("every-$|s|$"). But, by the definition of the denotation of R, this implies that $\Sigma \vdash\!\circ$ (every t (R (every s))). Since (R (every s)) appears in Σ, we have $\Sigma \vDash$ (every t (R (every s))).

Finally, suppose $\Sigma \vDash$ (every t (R (every s))). We must show that domain elements of the form "Q-$|t|$" are members of the set denoted by (R (every s)). Let "Q'-$|s'|$" be an arbitrary member of the set denoted by s. We must show that "Q-$|t|$" is a member of the set R("Q'-$|s'|$"). Since "Q'-$|s'|$" is a domain member, we must have $\Sigma \vDash \exists s'$. The denotation invariant for s implies that $\Sigma \vDash$ (every s' s). These two facts, plus inference rules 15 and 4, imply that $\Sigma \vDash$ (some s' s). The definition of $\vdash\!\circ$

now guarantees that $\Sigma \vdash\!\circ$ `(every` t `(R (`Q' s'`)))` and thus "Q-$|t|$" is in the set `R("`Q'-$|s'|$`")` as desired. This completes the proof of the denotation invariant.

We now conclude our proof of the completeness lemma by showing that the model defined above satisfies every literal φ in Σ. φ must be of the form `(every` s t`)`, $\neg$`(every` s t`)`, $\exists s$, or $\neg$`(some` s t`)` (formulas of the form $\neg\exists s$ are a special case of negative intersection formulas and we have assumed that Σ does not contain any positive intersection formulas other than formulas of the form $\exists s$). First, consider a literal in Σ of the form `(every` s t`)`. The denotation invariant (and the transitivity inference rule) implies that the set denoted by s must be a subset of the set denoted by t. Now consider a formula in Σ of the form $\neg$`(every` s t`)`. Inference rule 12 guarantees that $\Sigma \vdash\!\!\!\dashv \exists s$. Thus the semantic domain includes the object `"every-`$|s|$`"`. But since $\Sigma \not\vdash\!\!\!\dashv$ **F**, we must have $\Sigma \not\vdash\!\!\!\dashv$ `(every` s t`)`. Thus by the denotation invariant, `"every-`$|s|$`"` must be a member of the set denoted by s that is not a member of the set denoted by t, and thus the formula `(every` s t`)` must be false in the defined model. Now consider a formula in Σ of the form $\exists s$. The denotation invariant, and definition of the semantic domain immediately imply that the set denoted by s includes the object `"every-`$|s|$`"` and thus the formula $\exists s$ is true in the defined model. Finally, consider a formula in Σ of the form $\neg$`(some` s t`)`. Suppose this formula were false in the defined model, i.e., there exists a domain element that is in both the set denoted by s and the set denoted by t. Let "Q-$|w|$" be a domain element that is in both s and t. The definition of the semantic domain implies that $\Sigma \vdash\!\!\!\dashv \exists w$. The denotation invariant for s and t implies that $\Sigma \vdash\!\!\!\dashv$ `(every` w s`)` and $\Sigma \vdash\!\!\!\dashv$ `(every` w t`)`. But the definition of the relation $\vdash\!\!\!\dashv$ implies that in this case we have $\Sigma \vdash\!\!\!\dashv$ `(some` s t`)` and hence $\Sigma \vdash\!\!\!\dashv$ **F** which we have assumed is not so. This concludes the proof of the completeness lemma.

ACKNOWLEDGEMENT

This research was supported in part by National Science Foundation Grant IRI-8819624 and in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-85-K-0124 and N00014-89-j-3202.

References

[1] Kasimierz Adjuciewicz. Die syntaktische konnexitat. *Studia Philophica*, 1:1–27, 1935. Translated as "Sytactic Connection" in Strolls McCall (ed), *Polish Logic: 1920-1939* (Oxford University Press, 1967).

[2] Emmon Bach. Categorial grammars as theories of language. In Richard Oehrle and Edmond Bach, editors, *Categorial Grammars and Natural Language Structures*, pages 17–34. D. Reidel, 1988.

[3] D. Bobrow and T. Winograd. An overview of krl, a knowledge representation language. *Cognitive Science*, 1(1):3–46, 1977.

[4] R. Brachman, R. Fikes, and H. Levesque. Krypton: A functional approach to knowledge representation. *IEEE Computer*, 16:63–73, 1983.

[5] R. J. Brachman. What is-a is and isn't: An analysis of taxonomic links in semantic networks. *IEEE Computer*, 16(10):30–36, October 1983.

[6] Ronald Brachman and James Schmolze. An overview of the kl-one knowledge representation system. *Computational Intelligence*, 9(2):171–216, 1985.

[7] S. D. Johnson Constable, R. L. and C. D. Eichenlaub. *An Introduction to the PL/CV2 Programming Logic*, volume 135 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1982.

[8] Francesco Donini, Maurizio Lenzerini, Daniele Nardi, and Werner Nutt. The complexity of concept languages. In *Proceedings of KR91*, pages 151–162. Morgan Kaufmann Publishers, 1991.

[9] Peter J. Downey, Ravi Sethi, and Robert E. Tarjan. Variations on the common subexpression problem. *JACM*, 27(4):758–771, October 1980.

[10] D. Dowty, R. E. Wall, and S. Peters. *Introduction to Montague Semantics*. D. Reidel, 1981.

[11] Scott E. Fahlman. *NETL: A System for Representing Real World Knowledge*. MIT Press, 1979.

[12] Robert Givan, David McAllester, and Sameer Shalaby. Natural language based inference procedures applied to schubert's steamroller. In *AAAI-91*, pages 915–920. Morgan Kaufmann Publishers, July 1991.

[13] G. P. Huet. A unification algorithm for typed lambda-calculus. *Theoretical Computer Science*, 1:27–57, 1975.

[14] Dexter C. Kozen. Complexity of finitely presented algebras. In *Proceedings of the Ninth Annual ACM Symposium on the Theory of Computation*, pages 164–177, 1977.

[15] J. Lambek. The mathematics of sentence structure. *American Mathematical Monthly*, 65:154–169, 1958.

[16] D. McAllester and R. Givan. Taxonomic syntax for first order inference. *JACM*, 40(2):246–283, April 1993.

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

[18] Richard Montague. Universal grammar. *Theoria*, 36:373–398, 1970. Reprinted in: *Formal Philosophy: Selected Papers of Richard Montague*, ed. by R. H. Thomason, Yale University Press, 1974.

[19] 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.

[20] Bernhard Nebel. Computational complexity of terminological reasoning in back. *Artificial Intelligence*, 34(3):371–384, 1988.

[21] Greg. Nelson and Derek Oppen. Simplification by cooperating decision procedures. *ACM Trans. Prog. Lang. and Syst.*, 1:245–257, October 1979.

[22] M. Schmidt-Schaub and G. Smalka. Attributive concept descriptions with complements. *Artificial Intelligence*, 47:1–26, 1991.