In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*
Cambridge, MA: AAAI/MIT Press, 1991.

Negation and Proof by Contradiction in Access-Limited Logic [*]

J. M. Crawford **B. J. Kuipers**
Department of Computer Sciences
The University of Texas At Austin
Austin, Texas 78712
jc@cs.utexas.edu
kuipers@cs.utexas.edu

Abstract

Access-Limited Logic (ALL) is a language for knowledge representation which formalizes the access limitations inherent in a network structured knowledge-base. Where a deductive method such as resolution would retrieve all assertions that satisfy a given pattern, an access-limited logic retrieves all assertions reachable by following an available access path.

In this paper, we extend previous work to include negation, disjunction, and the ability to make assumptions and reason by contradiction. We show that the extended ALL$_{neg}$ remains Socratically Complete (thus guaranteeing that for any fact which is a logical consequence of the knowledge-base, there exists a series of preliminary queries and assumptions after which a query of the fact will succeed) and computationally tractable. We show further that the key factor determining the computational difficulty of finding such a series of preliminary queries and assumptions is the depth of assumption nesting. We thus demonstrate the existence of a family of increasingly powerful inference methods, parameterized by the depth of assumption nesting, ranging from incomplete and tractable to complete and intractable.

Introduction

Past work in ALL [Crawford & Kuipers, 89, 90] has shown Socratic Completeness and computational tractability for a language with expressive power equivalent to that of a deductive database. In this paper, we extend the expressive power of ALL to that of first order logic without existential quantification, and thus for the first time demonstrate the existence of a tractable and Socratically Completeness inference method for a language in which complete inference is known to be NP complete [Cook, 1971]. The heart of the extension is the addition of 'classical' negation (*i.e.*, not negation by failure) to ALL. The addition of negation is important because it allows one to express negative and disjunctive information. Further, the addition of negation allows us to augment inference by *modus ponens* (the only inference method required in a deductive database) with inference by *reductio ad absurdum* — the ability to make assumptions and reason by contradiction.

Socratic Completeness guarantees that for any fact which is a logical consequence of the knowledge-base *there exists* a series of preliminary queries and assumptions after which a query of the fact will succeed. Since complete inference in first order logic without existential quantification is known to be NP complete, the problem of *generating* such series must be NP complete. However, we will show that the key factor determining the difficulty of finding such a series is the depth of assumption nesting in the series. Given any knowledge-base and any fact, if there exists a series of preliminary queries and assumptions after which a query of the fact will succeed, and the series only nests assumptions to depth n, then the series can be found in time proportional to the size of the knowledge-base raised to the nth power. This result is particularly significant since we expect common-sense reasoning to require large knowledge-bases, but relatively shallow assumption nesting.

It has been apparent at least since the work of [Woods, 75] that a knowledge representation language sufficient to support the building of large knowledge bases must have a clear semantics. Without a clear semantics one can never be sure exactly *what* a given expression represents, which deductions should follow from it, or how it compares to an expression in a different knowledge representation language. Experience with formally specified knowledge representation systems has revealed a trade-off between the expressive power of knowledge representation systems and their computational complexity [Levesque, 86]. If, for exam-

[*]This work has taken place in the Qualitative Reasoning Group at the Artificial Intelligence Laboratory, The University of Texas at Austin. Research of the Qualitative Reasoning Group is supported in part by the Texas Advanced Research Program under grant no. 003658-175, NSF grants IRI-8905494 and IRI-8904454, and by NASA grant NAG 2-507.

ple, a knowledge representation language is as expressive as first-order predicate calculus, then the problem of deciding what an agent could logically deduce from its knowledge is unsolvable [Boolos & Jeffrey, 80]. There are several possible solutions to this problem: (1) restrict the expressive power of the representation language, (2) describe the reasoning ability of the system with an operational semantics rather than a model theoretic semantics, or (3) give up completeness but guarantee Socratic Completeness.

Unfortunately, there are problems with each of these solutions. If one is interested in expressing the full range of human common-sense knowledge then restricting the expressive power of the representation language is not an option. An operational semantics may suffice to describe the reasoning ability of the system, but it does not define the semantics of statements in the representation language.

One can attempt to give a model theoretic semantics for the representation language, and an operational semantics to reasoning. However, reasoning will then be incomplete with respect to the semantics of the representation language.[1] That is, there will be cases in which according to the model theoretic semantics $Th \models f$, but the system cannot derive f from Th. Thus the meaning ascribed to statements by the model theoretic semantics will not always match their meaning to the system.

The third solution, Socratic Completeness, does guarantee that the model theoretic semantics describes the potential reasoning ability of the system, and thus (together with soundness) guarantees that the meaning ascribed to statements by the model theoretic semantics is the same as that ascribed to statements by the system. However, Socratic Completeness is a weak property and can be satisfied by any system of inference with a complete set of proof rules.

In ALL we use a model theoretic semantics to describe the potential reasoning ability of the system (by guaranteeing Socratic Completeness), but uses an operational semantics to describe the behavior of the system on individual operations. In a sense it is not surprising that complete inference (with respect to a model theoretic semantics) is intractable; the model theoretic semantics was not intended to describe which inferences are easy to make, or are useful in a given context. The model theoretic semantics describes which inferences are permissible, and is thus an appropriate description of the potential long term reasoning ability of the system. We expect a single operation to permit only deductions which are "obvious". The behavior of the system on single operations can thus best be described operationally.[2] In ALL, deductions which follow simply by rule chaining are "obvious" and can be made by a single operation. The deductive power of ALL on a single operation is thus roughly equivalent to that of a deductive database.[3] As will be seen below, the more complex deductions are those which require reasoning by contradiction (or equivalently, reasoning by cases).

Our approach in ALL begins with the well known mapping between atomic propositions in predicate calculus and slots in frames; the atomic proposition that the object a stands in relation r to the object b can be written logically as $r(a, b)$ or expressed, in frames, by including object b in the r slot of object a [Hayes, 79]:

$$r(a,b) \quad \equiv \quad \boxed{\begin{array}{l}\texttt{a:}\\ \quad\texttt{r:}\\ \qquad\texttt{values: }\{\texttt{ ...b... }\}\end{array}}$$

ALL allows forward-chaining and backward-chaining inference rules. Antecedents of ALL rules must always define *access paths* through the network of frames. Access paths are sequences of predicates which can be evaluated by retrieving values from known slots of known frames. For example, in ALL one could write the backward-chaining rule:

$$aunt(John, y) \leftarrow parent(John, x), sister(x, y)$$

But one *cannot* write the (logically equivalent) rule:

$$aunt(John, y) \leftarrow sister(x, y), parent(John, x),$$

since evaluation of the antecedent would require a global search of the knowledge-base for pairs of frames in a *sister* relation.[4] The use of access paths allows ALL operations to be performed in time proportional to the size of the accessible portion of the knowledge-base [Crawford & Kuipers, 89].

In order to add negation to ALL, we begin by adding for each relation r an additional relation $\neg r$. If we made no other changes this would give us a language with the expressive power to represent negation, but we would have lost Socratic Completeness. In order to maintain Socratic Completeness, we have to connect r with $\neg r$. We do this by adding the notion of an inconsistent knowledge-base, and the ability to make

[1] One could achieve completeness by giving up tractability, however, if a system is intractable then in practice this means that it does not return on some inputs (or returns only after some unacceptable period of time), and thus is effectively incomplete.

[2] A model theoretic semantics may well not be appropriate, since obviousness often depends as much on the form of knowledge as its meaning — *e.g.*, p is obvious from q and $q \rightarrow p$, but is not so obvious from q and $\neg p \rightarrow \neg q$.

[3] ALL allows both backward and forward chaining rules. If one uses only backward chaining rules (or only forward chaining rules) then the deductive power of ALL on a single operation is equivalent to that of a deductive database.

[4] The restriction to access paths limits the syntax of ALL, but is not a fundamental limit on its expressive power since one could always add a new constant and make it the first argument to every predicate. This would amount to making the entire knowledge-base a single frame.

Example 1:

1. If Mary waters the flowers, then the flowers bloom.
2. If the flowers bloom, then Mary is happy.
3. Mary waters the flowers.

Q: Is Mary happy?

Example 2:

1. If Mary waters the flowers and the flowers bloom, then Mary is happy.
2. If Mary does not water the flowers, then they do not bloom.
3. Mary is not happy.

Q: Do the flowers bloom?

To a human, the solution to first example is obvious, while the second requires some thought. To a resolution theorem-prover, both are two-step proofs. In ALL$_{neg}$, the first question is answered immediately, while the second requires a nested proof by contradiction. It is our goal to build systems that reason efficiently on problems people find easy, even at the cost of requiring help on problems people find difficult.

Figure 1: Two Examples.

assumptions and reason by *reductio ad absurdum.* Figure 1 shows an example of a conclusion which can be drawn using rules alone, and a conclusion which requires reasoning by *reductio ad absurdum.*

The first section below presents the formal development of ALL$_{neg}$, including sketches of the proofs of Socratic Completeness and polynomial time complexity. We next turn to the problem of generating the preliminary queries and assumptions, and shows that the key factor determining the complexity of generating such series is the depth of assumption nesting. Finally we overview related work, and discuss current and future work.

Formal Development

This section formally develops ALL with negation. A more detailed development can be found in [Crawford, 90].

Syntax

ALL$_{neg}$ relies on the definitions of queries and assertions in ALL. The syntax and semantics of ALL (without negation) are over-viewed in [Crawford & Kuipers, 89, 90b], and discussed in detail in [Crawford, 90]. For purposes of the discussion below, it suffices to know that ALL defines the operations *query* and *assert*, both of which operate on a knowledge-base by applying all accessible inference rules, and both of which return pairs:

$$\langle \text{ 'set of substitutions', 'new knowledge-base' } \rangle.$$

We use *sub* and *kb* as accessors on the first and second components respectively.

In ALL$_{neg}$, we use the function *negate* to return the negation of a relation or proposition in the obvious way: $negate(r(t_1,\ldots,t_n)) = \neg r(t_1,\ldots,t_n)$, and $negate(\neg r(t_1,\ldots,t_n)) = r(t_1,\ldots,t_n)$.[5] A knowledge-base is *f-inconsistent* iff there exists a fact f such that $f \in K$ and $negate(f) \in K$.[6]

When an assumption is made in ALL$_{neg}$, the state of the knowledge-base is saved so that, if the assumption leads to a contradiction, it can be restored. Thus, in general ALL$_{neg}$ operations operate on stacks of knowledge-bases which we refer to as *knowledge-base structures.* Formally, a knowledge-base structure, or *kbs* for short, $\mathcal{K}$, is a stack of pairs:

$$\langle a_n, K_n \rangle, \ldots, \langle a_1, K_1 \rangle$$

such that, for $1 \leq i \leq n$, K_i is a knowledge-base and a_i is *nil* or a fact allowed in K_i.[7] We access the top element of such a stack using the function *head*, and rest of the stack using *rest*. $height(\mathcal{K})$ denotes the number of pairs in the structure. For any pair $\langle a, K \rangle$ we access the first and second components using the functions *assump* and *kb* respectively. We use *top_kb* and *top_assump* to access the 'top' knowledge-base and assumption in a structure: $top\_kb(\mathcal{K}) = kb(head(\mathcal{K}))$, and $top\_assump(\mathcal{K}) = assump(head(\mathcal{K}))$. We denote the structure formed by adding the pair $\langle a, K \rangle$ to the top of the the structure $\mathcal{K}$ by: $push(\langle a, K \rangle, \mathcal{K})$.

To differentiate them from ALL operations, we subscript ALL$_{neg}$ operations by *neg*. Thus $query(q)$ is an ALL operation and operates on a knowledge-base, while $query_{neg}(q)$ is an ALL$_{neg}$ operation which operates on a kbs. The same holds for variables ranging over operations: if $\mathcal{O} = query(q)$ then $\mathcal{O}_{neg}$ is the corresponds ALL$_{neg}$ operation $query_{neg}(q)$.

Three types of operations are supported in ALL$_{neg}$: queries, assertions, and assumptions. If α is a nonempty path then $query_{neg}(\alpha)$ is a *query*. If f is a fact then $assert_{neg}(f)$ is an *assertion*. Finally, if a is a fact then $assume_{neg}(a)$ is an *assumption*. If an operation $\mathcal{O}_{neg}$ is allowed in a kbs $\mathcal{K}$ then $\mathcal{O}_{neg}(\mathcal{K})$ is an ALL$_{neg}$ *formula*.

[5] The distinction between $\neg$ and *negate* is important: $\neg$ is simply a character which can occur in names of relations, while *negate* is a function; $\neg$ is a part of ALL, while *negate* is a part of the mathematical language used to define ALL.

[6] Since knowledge-bases may not be deductively closed, a knowledge-base may be inconsistent (*i.e.*, it may not have a model), but not be f-inconsistent.

[7] A fact is allowed in a knowledge-base iff it contains only constants and relations which are allowed in the knowledge-base.

Knowledge Theory

The knowledge theory defines the values of ALL$_{neg}$ formulas by defining the effect of operations. ALL$_{neg}$ operations map a kbs to a set of substitutions and a new kbs. We denote these returned values with pairs: $\langle$ 'set of substitutions', 'knowledge-base structure' $\rangle$. We use *sub* and *kbs* as accessors on the first and second components respectively.

Performing an ALL$_{neg}$ operation involves performing it as an ALL operation on the top knowledge-base in the structure, and then checking for, and dealing with, contradictions. Consider a kbs $\mathcal{K}$ and a query or assertion $\mathcal{O}_{neg}$ (allowed in $\mathcal{K}$). If $kb(\mathcal{O}(top\_kb(\mathcal{K})))$ (*i.e.*, the knowledge-base produced when $\mathcal{O}$ is performed as an ALL operation on the top knowledge-base in the structure) is f-consistent then it just replaces the old top knowledge-base in $\mathcal{K}$. If, however, $kb(\mathcal{O}(top\_kb(\mathcal{K})))$ is f-inconsistent, then the most recent assumption is retracted (*i.e.*, the stack is 'popped'), and the negation of the most recent assumption is asserted as a fact (unless the stack has only one element in which case there are no assumptions to drop and the knowledge-base is inconsistent).[8] Figure 2 shows the formal definitions of ALL operations.

Socratic Completeness

As discussed in the introduction, reasoning in ALL$_{neg}$ is not complete but it is Socratically Complete: for any fact which is a logical consequence of a knowledge-base, there exists a series of preliminary queries and assumptions, after which a query of the fact will succeed. To prove this, we first define the semantics of ALL by a mapping to predicate calculus, and then define provability in ALL$_{neg}$ and show it satisfies the proof rules *Modus Ponens* and *Reductio Ad Absurdum.* From these proof rules we show Socratic Completeness.

Mapping ALL$_{neg}$ to predicate calculus is straightforward; each knowledge-base in the structure is believed under its assumption and all assumptions 'under' it. For any kbs $\mathcal{K}$ such that $\mathcal{K} = \langle a_n, K_n\rangle, \ldots, \langle a_1, K_1\rangle$:

$$\mathcal{PC}(\mathcal{K}) = (\bigwedge i \ : 1 \leq i \leq n \ : (\bigwedge j : 1 \leq j \leq i : \mathcal{PC}(a_j)) \Rightarrow \mathcal{PC}(K_i))$$

[8]The assumption management techniques used in this formalism have been chosen to make the formal development as transparent as possible. More efficient techniques are used in the implementation of ALL. The major difference between the formalism and the implementation is that the formalism uses *chronological* backtracking while the implementation uses *dependency directed* backtracking. In chronological backtracking, when a contradiction is found the most recent assumption is retracted. In dependency directed backtracking, the dependencies of facts in the knowledge-base on assumptions are explicitly maintained (often as labels on the facts [Stallman & Sussman, 77]). When a contradiction is found, an assumption which the fact depends on is retracted.

For a knowledge-base, K, $\mathcal{PC}(K)$ simply returns the conjunction of the facts and rules in K (with all variables in the rules universally quantified). Note that we overload $\mathcal{PC}$ in that it maps knowledge-base structures, knowledge-bases, and facts to predicate calculus. A kbs K is *consistent* iff there exists a model $\mathcal{M}$ such that $\mathcal{M} \models \mathcal{PC}(\mathcal{K})$.

A fact is provable in ALL$_{neg}$ iff there exists a series of queries and assumptions deriving it. For a series of operations ?, we use the notation $?(\mathcal{K})$ to denote the result produced by performing the operations in ? on $\mathcal{K}$ in turn (*e.g.*, unless ? of length zero: $?(\mathcal{K}) = kbs(head(?)(rest(?)(\mathcal{K})))$).

Def 1 *Given a kbs $\mathcal{K} = \langle a, K\rangle$, and a ground path $f_1, \ldots, f_n$ allowed in $\mathcal{K}$:*

$$\mathcal{K} \vdash_{ALL} f_1, \ldots, f_n$$

iff there exists a series ? of queries and assumptions allowed in $\mathcal{K}$ such that:

$$?(\mathcal{K}) = \langle a, K'\rangle \wedge (\forall i : 1 \leq i \leq n : f_i \in K').$$

The key lemmas for $\vdash_{ALL}$ are the proof rules *Modus Ponens* and *Reductio Ad Absurdum.* In stating these lemmas we use the shorthand $\mathcal{K} + f$ to denote the addition of f to the top knowledge-base in $\mathcal{K}$.

Lemma 1 (*Reductio Ad Absurdum)*
For any well-formed kbs $\mathcal{K}$ and any fact a allowed in $\mathcal{K}$, if there exists a fact f allowed in $\mathcal{K}$ such that: $(\mathcal{K} + a \vdash_{ALL} f) \wedge (\mathcal{K} + a \vdash_{ALL} negate(f))$ then: $\mathcal{K} \vdash_{ALL} negate(a)$.

Proof: (Sketch) Intuitively, the series of operations deriving $negate(a)$ is one which first assumes a, and then derives f and $\neg f$ (one can show such a series exists since $\mathcal{K}+a \vdash_{ALL} f$ and $\mathcal{K}+a \vdash_{ALL} negate(f)$).[9] This leads to a contradiction, causing the assumption a to be retracted and $negate(a)$ asserted. ∎

Lemma 2 (*Modus Ponens) For any well-formed kbs $\mathcal{K}$, any (if-added or if-needed) rule $\rho \in top\_kb(\mathcal{K})$ such that $Ant(\rho) = b_1, \ldots, b_n$, and any ground substitution θ such that $vars(\rho) \subset domain(\theta)$, if $(\forall i : 1 \leq i \leq n : \mathcal{K} \vdash_{ALL} b_i\theta)$ then: $\mathcal{K} \vdash_{ALL} Conseq(\rho)\theta$.*

Proof: (Sketch) $(\forall i : 1 \leq i \leq n : \mathcal{K} \vdash_{ALL} b_i\theta)$ implies that for all i there is some $?_i$ such that $b_i\theta \in top\_kb(?_i(\mathcal{K}))$. Intuitively, one can append these $?_i$ together and produce some ? such that $Ant(\rho)\theta \in top\_kb(?(\mathcal{K}))$ (formally a bit more work is required, but one can show that such a ? exists). Finally, if a knowledge-base contains the antecedent of a rule then we know from the study of ALL without negation [Crawford, 90] that there must always exist a series of queries deriving the consequent of the rule. ∎

[9]Actually such a series may not exist as some other 'accidental' contradiction may occur before f and $negate(f)$ are derived. Such contradictions are not a problem, however, as they only cause $negate(a)$ to be derived more quickly.

For any query or assertion, $\mathcal{O}_{neg}$, allowed in a kbs $\mathcal{K}$:

If $kb(\mathcal{O}(top\_kb(\mathcal{K}))$ is f-consistent or $height(\mathcal{K}) = 1$ then:

$$\mathcal{O}_{neg}(\mathcal{K}) = \langle sub(\mathcal{O}(top\_kb(\mathcal{K})), push(\langle top\_assump(\mathcal{K}), kb(\mathcal{O}(top\_kb(\mathcal{K})))\rangle, rest(\mathcal{K}))\rangle$$

else (an inconsistency has been found and there is an assumption to drop):

$$\mathcal{O}_{neg}(\mathcal{K}) = \langle \emptyset, kbs(assert_{neg}(negate(top\_assump(\mathcal{K})))(rest(\mathcal{K})))\rangle$$

For any fact a allowed in a kbs $\mathcal{K}$, assuming a requires adding a new pair to the stack and then asserting a:

$$assume(a)(\mathcal{K}) = assert_{neg}(a)(push(\langle a, top\_kb(\mathcal{K})\rangle, \mathcal{K}))$$

Figure 2: Formal definitions of ALL operations.

Proving Socratic Completeness is now only a matter of using lemmas 1 and 2 to show that $\mathcal{PC}(\mathcal{K}) \models \mathcal{PC}(f)$ implies $\mathcal{K} \vdash_{ALL} f$. This can be done using a standard Henkin style proof [Hunter, 71].

Theorem 1 (Socratic Completeness for ALL$_{neg}$) *If $\mathcal{K}$ is a well-formed kbs of height one and f is a fact allowed in $\mathcal{K}$ then: $\mathcal{PC}(\mathcal{K}) \models \mathcal{PC}(f) \Rightarrow \mathcal{K} \vdash_{ALL} f$.*

Time Complexity

We show in [Crawford & Kuipers, 89, Crawford, 90] that ALL operations can be performed in time polynomial in the size of the accessible portion of the knowledge-base. ALL$_{neg}$ operations require only a linear number of ALL operations and thus must also be computable in time polynomial in the size of the accessible portion of the knowledge-base.

Generating the Preliminary Operations

Consider a kbs $\mathcal{K}$ and a fact f logically entailed by $\mathcal{K}$. Socratic Completeness guarantees that there exists a series of operations deriving f, but says nothing about the difficulty of finding such a series. Intuitively, such a series makes assumptions and then performs queries to uncover contradictions. In this section, we show that the *depth of assumptions nesting* in the series (*e.g.*, the maximum height reached by the kbs as the series of operations is performed) determines the complexity of generating the series.

In order to state this result formally, we need some additional notation.

Def 2 *For a kbs $\mathcal{K}$, let $?_{\mathcal{K}}$ be the series of queries of ground instances of consequents of the backward-chaining rules in $\mathcal{K}$.*

$?_{\mathcal{K}}$ is important because one can show (from lemmas used in proving the Socratic Completeness of ALL without negation [Crawford, 90]) that iterating $?_{\mathcal{K}}$ a sufficient number of times is sufficient to derive any fact which can be derived without making assumptions.

We also formalize the idea of limiting the depth of assumption nesting. Consider a kbs $\mathcal{K}$ and series of operations allowed in $\mathcal{K}$, $? = \mathcal{O}_1, \ldots, \mathcal{O}_m$. Let $\mathcal{K}_0 = \mathcal{K}$ and $\mathcal{K}_i = \mathcal{O}_i(\mathcal{K}_{i-1})$. We use the shorthand:

$$max\_height(\mathcal{K}, ?) = (Max\ i : 1 \leq i \leq m : height(\mathcal{K}_i))$$

Def 3 *For any kbs $\mathcal{K}$, any fact f allowed in $\mathcal{K}$, and any $n > 0$:*

$$\mathcal{K} \vdash^{n}_{ALL} f$$

iff $\mathcal{K} \vdash_{ALL} f$ by some series of queries and assumptions $?$ such that $max\_height(\mathcal{K}, ?) \leq n$.

Finally, we define parameters which measure the size of a kbs. Consider a kbs $\mathcal{K}$. Let len be the maximum length of any rule in $\mathcal{K}$, $mvars$ be the maximum number of variables in any rule in $\mathcal{K}$, and ma be the maximum arity of any relation allowed in $\mathcal{K}$. Further, let r be the number of rules in $\mathcal{K}$, f be the number of frames (*e.g.*, constants) in $\mathcal{K}$, and s be the number of slots (*e.g.*, relations) in $\mathcal{K}$. One important measure of the size of $\mathcal{K}$ is the bound on the number of facts which could be added to $\mathcal{K}$ by any operation or series of operations:[10]

$$m = s \times c^{ma}$$

The other important measure is the bound on the time complexity of any single ALL operation. Past work has shown that the time complexity of any ALL operation is bounded by a polynomial function of the size of the accessible portion of the knowledge-base. The size of the accessible portion of the knowledge-base varies from one operation to the next, but it clearly cannot exceed the bound on the possible size of the knowledge-base. From the previously derived bound on the time complexity of ALL operations [Crawford & Kuipers, 89, Crawford, 90], one can show that for a kbs of height n or less, the time complexity of any ALL operation (performed on $\mathcal{K}$ or on any

[10] Such a bound exists since ALL operations never create new frames or new slots.

```
Function $Prove(\mathcal{K}, f, n)$
  Until $\mathcal{K}$ unchanged do
    $\mathcal{K} := ?_{\mathcal{K}}(\mathcal{K})$
    Unless $n = 0$
      For each fact $a$ allowed in $\mathcal{K}$ do
        $\mathcal{K}' := Prove(\ assume(a)(\mathcal{K}),\ \neg a,\ n-1)$
        if $height(\mathcal{K}') \leq height(\mathcal{K})$ then $\mathcal{K} := \mathcal{K}'$
        od
    od
```

Figure 3: Algorithm for determining whether $\mathcal{K} \vdash^n_{ALL} f$.

kbs produced by a series of ALL operations on $\mathcal{K}$) is bounded by:

$$o = n \times r \times len^5 \times m^{2+6mvars+ma}$$

Thus we have:

Theorem 2 *For any kbs $\mathcal{K}$, any fact f allowed in $\mathcal{K}$, and any $n > 0$: there exists an algorithm with worst case time complexity of order $o \times m^{2n}$ for determining whether $\mathcal{K} \vdash^n_{ALL} f$.*

Proof: (Sketch) The algorithm is shown in figure 3. It performs a search for a series of operations deriving f. A more efficient algorithm could certainly be found, but our focus here is on illustrating the dependence of the time complexity of inference on the depth of assumption nesting.

The analysis of the time complexity of *Prove* is straightforward — one need simply notice that both loops can be executed at most m times. The key correctness property for *Prove* is that:

$$\mathcal{K} \vdash^n_{ALL} f \Leftrightarrow f \in Prove(\mathcal{K}, f, n).$$

The reverse implication ($\Leftarrow$) can be seen by observing that *Prove* only performs ALL operations (and the depth of assumption nesting never exceeds n). The forward implication ($\Rightarrow$) is more complex. Assume $\mathcal{K} \vdash^n_{ALL} f$. This implies that there exists some ? such that $f \in ?(\mathcal{K})$. One can show that every query and proof by contradiction performed by ? is also performed by *Prove* (or is unnecessary), and so $f \in Prove(\mathcal{K}, f, n)$. ∎

Related Work

Work on *vivid* reasoning has similar goals to ALL. A vivid knowledge-base "... trades accuracy for speed ..." [Etherington et al., 89] by constructing a database of ground facts, from statements in a more expressive language. ALL represents all the knowledge that has been asserted (though some of it may not be accessible at a given time) while a vivid knowledge-base is an *approximation* of the asserted knowledge (thus Socratic Completeness does not hold for vivid reasoning).

A comparison between ALL and Prolog is illuminating. ALL implements 'classical' negation (*i.e.*, negation as in predicate calculus), while Prolog implements negation as failure (*i.e.*, the negation of a goal succeeds iff the goal fails). In ALL, the negation of a fact is implied by a knowledge-base iff the fact is false in all models of the knowledge-base. ALL's implementation of negation allows one to express disjunctions (*e.g.*, "John is the banker or the lawyer") which cannot be expressed directly in Prolog. Negation as failure is provided in Algernon, the lisp implementation of ALL, by a syntactically distinct form.

A complete natural deduction system (or resolution based theorem prover) with a bound on proof length is trivially Socratically Complete. Such systems differ from ALL in several important respects. While natural deduction like inference rules do appear in the meta-theory of ALL, single operations in ALL do not correspond to a bounded number of applications of these rules. In a single ALL operation, rules may forward and backward chain to a depth bounded only by the size of the accessible portion of the knowledge-base. Thus, any deduction which follows simply by rule chaining will succeed in a single ALL operation. This gives a considerably simpler picture of the power of a single operation than is possible in a system in which tractability is guaranteed by a bound on proof length. Further, when preliminary operations are required, the difficulty of finding a series of preliminary operations is related to the depth of assumption nesting, which is a considerably more intuitive metric that proof length.

Recently there has been growing interest in different aspects of tractable inference. One interesting result is that the tractability of inference rules is dependent on the syntax of the language used [McAllester et. al., 89, ,McAllester, 90, p. 1115]. [Shastri & Ajjanagadde, 90] show that, using a highly parallel implementation, certain queries can be answered in time proportional to the length of the shortest derivation of the query. Term subsumption languages (over-viewed in [Patel-Schneider et al., 90]) also support a particular kind of tractable inference (the computation of subsumption relations between descriptions). ALL provides a tractable inference method which applies to a network structured knowledge-base, and which is powerful enough to guarantee Socratic Completeness.

Current and Future Work

ALL cannot currently represent mixed existential and universal quantification. When mixed quantification is allowed, inference can result in the creation of a potentially unbounded number new frames. The key requirement for tractable inference with full quantification is thus careful control over the creation of new frames. We expect that common-sense reasoning requires the creation of a relatively small numbers of new frames, while the cleverness required for more complex reasoning often involves knowing which sets of new frames to construct.

Algernon, our Lisp implementation of ALL, is an effective knowledge-representation language which has been used to implement several substantial knowledge-based
systems, including QPC, a qualitative model builder for QSIM [Crawford, Farquhar, & Kuipers, 90]. Algernon has been an integral part of our work on ALL since our research methodology involves an interplay between theory and experimentation (for example, the importance of the depth of assumption nesting was first observed in solving logic puzzles using Algernon). The natural language group at MCC has recently begun using Algernon, and has developed functions which translate from an abstract knowledge-base interface based on Ontolingua [Gruber, 90] down to Algernon queries and assertions [Barnett et al., 91]. Algernon is somewhat more powerful than the currently formalized version of ALL and supports full quantification and certain types of default reasoning.

References

Barnett, J., Rich, E., and Wroblewski, D. (1991). A functional interface to a knowledge base for use by a natural language processing system. Manuscript. Knowledge-Based Natural Language Project, MCC, 3500 West Balcones Center Dr., Austin, Texas.

Boolos, George S., and Jeffrey, Richard C. (1980). *Computability and Logic*, Cambridge University Press, New York.

Brachman, R.J. and Levesque, H.J. (1985). *Readings in Knowledge Representation*, Morgan Kaufmann, Los Altos, Cal.

Cook, S.A. (1971). The complexity of theorem-proving procedures. *Proc. 3rd Ann. ACM Symp. on Theory of Computing* Association for Computing Machinery, New York, pp. 151-158.

Crawford, J., Farquhar, A., and Kuipers, B. (1990). QPC: A compiler from physical models into Qualitative Differential Equations. *AAAI-90.*

Crawford, J. M., and Kuipers, B. (1989). Towards a theory of access-limited logic for knowledge representation. In *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann, Los Altos, California.

Crawford, J. M., and Kuipers, B. (1991). Algernon — A tractable system for knowledge representation. AAAI Spring Symposium on *Implemented Knowledge Representation and Reasoning Systems.* Palo Alto, CA. Also to appear in special issue of *SIGART* on implemented knowledge representation systems.

Crawford, J. M., and Kuipers, B. (1990). Access-Limited Logic — A language for knowledge-representation. University of Texas at Austin dissertation. Available as Technical Report number AI90-141, Artificial Intelligence Laboratory, The University of Texas at Austin.

Etherington, David W., Borgida, Alex, Brachman, Ronald J., and Kautz, Henry (1989). Vivid knowledge and tractable reasoning: preliminary report. *IJCAI-89* pp. 1146-1152.

Fine, Kit (1985). *Reasoning With Arbitrary Objects*, Aristotelian Society Series, Volume 3, Basil Blackwell, Oxford.

Gruber, T., Pang, D., and Rice, J. *Ontolingua: A Language to Support Shared Ontologies.* Technical Report, Stanford Knowledge Systems Lab, Palo Alto, California.

Hayes, Patrick J. (1979). The logic of frames. In *Frame Conceptions and Text Understanding*, ed. D. Metzing, Walter de Gruyter and Co., Berlin, pp. 46-61. (Reprinted in [Brachman & Levesque, 85], pp. 288-295.)

Hunter, G. (1971). *Metalogic: An Introduction to the Metatheory of Standard First Order Logic.* University of California Press, Berkeley, CA.

Levesque, H. J. (1986). Knowledge representation and reasoning. In *Ann. Rev. Comput. Sci.* 1:255-87. Annual Reviews Inc, Palo Alto, California.

McAllester, D., Givan, R., and Fatima, T. (1989). Taxonomic syntax for first order inference. In *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann, Los Altos, California, pp. 289-300.

Patel-Schneider, P.F., et at. (1990). Term subsumption languages in knowledge representation. In *AI Magazine* 11(2):16-22.

An Optimally Efficient Limited Inference System. (1990). In *AAAI-90*, pp. 563-570.

Stallman, R.M. and Sussman, G.J. (1977). Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis, *Artificial Intelligence* 9:135-196.

Woods, W. (1975). What's in a link: foundations for semantic networks. In *Representation and Understanding: Studies in Cognitive Science*, ed. Bobrow, D. and Collins, A., Academic, New York, pp. 35-82.