New Results on Local Inference Relations

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

Abstract

We consider the concept of a *local* set of inference rules. A local rule set can be automatically transformed into a rule set for which bottom up evaluation terminates in polynomial time. The local rule set transformation gives polynomial time evaluation strategies for a large variety of rule sets that can not be given terminating evaluation strategies by any other known automatic technique. This paper discusses three new results. First, it is shown that every polynomial time predicate can be defined by an (unstratified) local rule set. Second, a new machine recognizable subclass of the local rule sets is identified. Finally we show that locality, as a property of rule sets, is undecidable in general.

This paper appeared in KR-92. A postscript electronic source for this paper can be found in ftp.ai.mit.edu:/pub/dam/kr92.ps. A bibtex reference can be found in internet file ftp.ai.mit.edu:/pub/dam/dam.bib.

1 INTRODUCTION

Under what conditions does a given set of inference rules define a computationally tractable inference relation? This is a syntactic question about syntactic inference rules. There are a variety of motivations for identifying tractable inference relations. First, tractable inference relations sometimes provide decision procedures for semantic theories. For example, the equational inference rules of reflexivity, symmetry, transitivity, and substitutivity define a tractable inference relation that yields a decision procedure for the entailment relation between sets of ground equations [Kozen, 1977], [Shostak, 1978]. Another example is the set of equational Horn clauses valid in lattice theory. As a special case of the results in this paper one can show (automatically) that validity of a lattice theoretic Horn clause is decidable in cubic time.

Deductive data bases provide a second motivation for studying tractable inference relations. A deductive data base is designed to answer queries using simple inference rules as well as a set of declared data base facts. The inference rules in a deductive data base usually define a tractable inference relation. The inference rules are usually of a special form known as a datalog program. A datalog program is a set of first order Horn clauses that do not contain function symbols. Any datalog program defines a tractable inference relation [Ullman, 1988], [Ullman, 1989]. Recently there has been interest in generalizing the inference rules used in deductive databases beyond the special case of datalog programs. In the general case, where function symbols are allowed in Horn clause inference rules, a set of inference rules can be viewed as a Prolog program. Considerable work has been done on "bottom up" evaluation strategies for these programs and source to source transformations that make such bottom up evaluation strategies more efficient [Naughton and Ramakrishnan, 1991] [Bry, 1990]. The work presented here on local inference relations can be viewed as an extension of these optimization techniques. For example, locality testing provides an automatic source to source transformation on the inference rules for equality (symmetry, reflexivity, transitive, and substitution) that allows them to be completely evaluated in a bottom-up fashion in cubic time. We do not know of any other automatic transformation on inference rules that provides a terminating evaluation strategy for this rule set.

A third motivation for the study of tractable inference relations is the role that such relations can play in improving the efficiency of search. Many practical search algorithms use some form of incomplete inference to prune nodes in the search tree [Knuth, 1975], [Mackworth, 1977], [Pearl and Korf, 1987]. Incomplete inference also plays an important role in pruning search in constraint logic programming [Jaffar and Lassez, 1987], [van Hentenryck, 1989], [McAllester and Siskind, 1991]. Tractable inference relations can also be used to define a notion of "obvious inference" which can then be used in "Socratic" proof verification systems which require proofs to be reduced to obvious steps [McAllester, 1989], [Givan *et al.*, 1991].

As mentioned above, inference rules are syntactically similar to first order Horn clauses. In fact, most inference rules can be syntactically represented by a Horn clause in sorted first order logic. If R is a set of Horn clauses, Σ is a set of ground atomic formulas, and Φ is a ground atomic formula, then we write $\Sigma \vdash_R \Phi$ if $\Sigma \cup R \vdash \Phi$ in first order logic. We write $\vdash_R$ rather than $\models_R$ because we think of R as a set of syntactic inference rules and $\vdash_R$ as the inference relation generated by those rules. Throughout this paper we use the term "rule set" as a synonym for "set of Horn clauses". Technically this phrase refers to a finite set of Horn clauses. We give nontrivial conditions on R which ensure that the inference relation $\vdash_R$ is polynomial time decidable.

As noted above, a rule set R that does not contain any function symbols is called a datalog program. It is well known that the inference relation defined by a datalog program is polynomial time decidable. Vardi and Immerman independently proved, in essence, that datalog programs provide a characterization of the complexity class P — any polynomial time predicate on finite data bases can be written as a datalog program provided that one is given a successor relation that defines a total order on the domain elements [Vardi, 1982], [Immerman, 1986], [Papadimitriou, 1985].

Although datalog programs provide an interesting class of polynomial time inference relations, the class of tractable rule sets is much larger than the class of datalog programs. First of all, one can generalize the concept of a datalog program to the concept of a *superficial* rule set. We call a set of Horn clauses superficial if any term that appears in the conclusion of a clause also appears in some antecedent of that clause. A superficial rule set has the property that forward chaining inference does not introduce new terms. Superficial rule sets provide a different characterization of the complexity class P. While datalog programs can encode any polynomial time predicate on finite data bases, superficial rule sets can encode any polynomial time predicate on first order terms. Let $\mathcal{P}$ be a predicate on first order terms constructed from a finite signature. We define the DAG size of a first order term t to be the number of distinct terms that appear as subexpressions of t.[1] It is possible to show that if $\mathcal{P}$ can be computed in polynomial time in sum of the the DAG size of its arguments then $\mathcal{P}$ can be represented by a superficial rule set. More specifically, we prove here that for any such predicate $\mathcal{P}$ on k first order terms there exists a superficial rule set R such that $\mathcal{P}(t_1,\ t_2,\ \ldots\ t_k)$ if and only if $\texttt{Input}(t_1,\ t_2,\ \ldots\ t_k) \vdash_R \texttt{Accept}$ where `Input` is a predicate symbol and `Accept` is a distinguished proposition symbol. The characterization of P in terms of superficial rule sets differs from Immerman's characterization of P in terms of datalog programs in two ways. First, the result is stated in terms of predicates on terms rather than predicates

[1]The DAG size of a term is the size of the Directed Acyclic Graph representation of the term.

on databases. Second, unlike the datalog characterization, no separate total order on domain elements is required.

Superficial rule sets are a special case of the more general class of *local* rule sets [McAllester, 1993]. A set R of Horn clauses is local if whenever $\Sigma \vdash_R \Phi$ there exists a proof of Φ from Σ such that every term in the proof is mentioned in Σ or Φ. If R is local then $\vdash_R$ is polynomial time decidable. All superficial rule sets are local but many local rule sets are not superficial. The set of the four inference rules for equality is local but not superficial. The local inference relations provide a third characterization of the complexity class P. Let $\mathcal{P}$ be a predicate on first order terms constructed from a finite signature. If $\mathcal{P}$ can be computed in polynomial time in the sum of the DAG size of its arguments then there exists a local rule set R such that for any terms $t_1, t_2, \ldots t_k$ we have that $\mathcal{P}(t_1,\ t_2,\ \ldots\ t_k)$ if and only if $\vdash_R P(t_1,\ t_2,\ \ldots\ t_k)$ where P is a predicate symbol representing $\mathcal{P}$. Note that no superficial rule set can have this property because forward chaining inference from a superficial rule set can not introduce new terms. We find the characterization of polynomial time predicates in terms of local rule sets to be particularly pleasing because it yields a direct mapping from semantic predicates to predicates used in the inference rules.

Unlike superficiality, locality can be difficult to recognize. The set of four inference rules for equality is local but the proof of this fact is nontrivial. Fortunately, there are large classes of mechanically recognizable local rule sets. A notion of a bounded local rule set is defined in [McAllester, 1993] and a procedure is given which will automatically recognize the locality of any bounded local rule set. The set of the four basic rules for equality is bounded local. As another example of a bounded local rule set we give the following rules for reasoning about a monotone operator from sets to sets.

$$x \leq x$$

$$x \leq y,\ y \leq z \Rightarrow x \leq z$$

$$x \leq y \Rightarrow f(x) \leq f(y)$$

Let R_f be this set of inference rules for a monotone operator.

There is a simple source to source transformation on any local rule set that converts the rule set to a superficial rule set without losing completeness. For example consider the above rules for a monotone operator. We can transform these rules so that they can only derive information about terms explicitly mentioned in the query. To do this we introduce another predicate symbol $\mathtt{M}$ (with the intuitive meaning “mentioned”). The rules can be rewritten as follows.

$$\mathtt{M}(f(x)) \Rightarrow \mathtt{M}(x)$$

$$x \leq y \Rightarrow \mathtt{M}(x)$$

$$x \leq y \Rightarrow \mathtt{M}(y)$$

$$\mathtt{M}(x) \Rightarrow x \leq x$$

$$\mathtt{M}(x),\ \mathtt{M}(y),\ \mathtt{M}(z),\ x \leq y,\ y \leq z \Rightarrow x \leq z$$

$$\mathtt{M}(f(x)),\ \mathtt{M}((f(y)),\ x \leq y \Rightarrow f(x) \leq f(y)$$

Let this transformed rule set be R'_f. Note that R'_f is superficial and hence bottom up (forward chaining) evaluation must terminate in polynomial time.[2] Then to determine if $\Sigma\ \vdash_{R_f}\ t \leq u$ we determine, by bottom up evaluation, whether $\{\mathtt{M}(t),\ \mathtt{M}(u)\} \cup \Sigma\ \vdash_{R'_f}\ t \leq u$. An analogous transformation applies to any local rule set.

A variety of other bounded local rule sets are given [McAllester, 1993]. As an example of a rule set that is local but not bounded local we give the following rules for reasoning about a lattice.

$$x \leq x$$

$$x \leq y,\ y \leq z \Rightarrow x \leq z$$

$$x \leq x \vee y$$

$$y \leq x \vee y$$

$$x \leq z,\ y \leq z \Rightarrow x \vee y \leq z$$

$$x \wedge y \leq x$$

$$x \wedge y \leq y$$

$$z \leq x,\ z \leq y \Rightarrow z \leq x \wedge y$$

These rules remain local when the above monotonicity rule is added. With or without the monotonicity rule, the rule set is not bounded local.

In this paper we construct another machine-recognizable subclass of the local rule sets which we call *inductively local* rule sets. All of the bounded local rule sets given in [McAllester, 1993] are also inductively local. The procedure for recognizing inductively local rule sets has been implemented and has been used to determine that the above rule set is inductively local. Hence the inference relation defined by the rules is polynomial time decidable. Since these rules are complete for lattices this result implies that validity for lattice theoretic Horn clauses is polynomial time decidable.

We been able to show that there are bounded local rule sets which are not inductively local, although our examples are somewhat artificial. We have not found any natural examples of local rule sets that fail to be inductively

[2]For this rule set bottom up evaluation can be run to completion in cubic time.

local. Inductively local rule sets provide a variety of mechanically recognizable polynomial time inference relations.

In this paper we also settle an open question from our previous analysis [McAllester, 1993] and show that locality as a general property of rule sets is undecidable. Hence the optimization of logic programs based on the recognition of locality is necessarily a somewhat heuristic process.

2 BASIC TERMINOLOGY

In this section we give more precise definitions of the concepts discussed in the introduction.

> **Definition:** A Horn clause is a first order formula of the form $\Psi_1 \wedge \Psi_2 \wedge \cdots \wedge \Psi_n \Rightarrow \Phi$ where Φ and the Ψ_i are atomic formulas. For any set of Horn clauses R, any finite set Σ of ground terms, and any ground atomic formula Φ, we write $\Sigma \vdash_R \Phi$ whenever $\Sigma \cup U(R) \vdash \Phi$ in first order logic where $U(R)$ is the set of universal closures of Horn clauses in R.

There are a variety of inference relations defined in this paper. For any inference relation $\vdash$ and sets of ground formulas Σ and Γ we write $\Sigma \vdash \Gamma$ if $\Sigma \vdash \Psi$ for each Ψ in Γ.

The inference relation $\vdash_R$ can be given a more direct syntactic characterization. This syntactic characterization is more useful in determining locality.

> **Definition:** A *derivation* of Φ from Σ using rule set R is a sequence of ground atomic formulas Ψ_1, Ψ_2, ... Ψ_n such that Ψ_n is Φ and for each Ψ_i there exists a Horn clause $\Theta_1 \wedge \Theta_2 \wedge \ldots \wedge \Theta_k \Rightarrow \Psi'$ in R and a ground substitution σ such that $\sigma[\Psi']$ is Ψ_i and each formula of the form $\sigma[\Theta_i]$ is either a member of Σ or a formula appearing in earlier in the derivation.

> **Lemma:** $\Sigma \vdash_R \Phi$ if and only if there exists a derivation of Φ from Σ using the rule set R.

The following restricted inference relation plays an important role in the analysis of locality.

> **Definition:** We write $\Sigma \vDash_R \Phi$ if there exists a derivation of Φ from Σ such that every term appearing in the derivation appears as a subexpression of Φ or as a subexpression of some formula in Σ.

Lemma: For any finite rule set R the inference relation $\Vdash_R$ is polynomial time decidable.

Proof: Let n be the number of terms that appear as subexpressions of Φ or a formula in Σ. If P is a predicate of k arguments that appears in the inference rules R then there are at most n^k formulas of the form $P(s_1, \ldots s_k)$ such that $\Sigma \Vdash_R P(s_1, \ldots s_k)$. Since R is finite there is some maximum arity over all the predicate symbols that appear in R. The total number of formulas that can be derived under the restrictions in the definition of $\Vdash_R$ is order n^k where k is the maximum arity of the predicates in R. ■

Clearly, if $\Sigma \Vdash_R \Phi$ then $\Sigma \vdash_R \Phi$. But the converse does not hold in general. By definition, if the converse holds then R is local.

Definition([McAllester, 1993]): The rule set R is *local* if the restricted inference relation $\Vdash_R$ is the same as the unrestricted relation $\vdash_R$.

Clearly, if R is local then $\vdash_R$ is polynomial time decidable.

3 CHARACTERIZING P WITH SUPERFICIAL RULES

In this section we consider predicates on first order terms that are computable in polynomial time. The results stated require a somewhat careful definition of a polynomial time predicate on first order terms.

Definition: A *polynomial time predicate on terms* is a predicate $\mathcal{P}$ on one or more first order terms which can be computed in polynomial time in the sum of the DAG sizes of its arguments.

Superficial Rule Set Representation Theorem: If $\mathcal{P}$ is a polynomial time predicate on first order terms then there exists a superficial rule set R such that for any first order terms $t_1, \ldots, t_n$, we have that $\mathcal{P}$ is true on arguments $t_1, \ldots, t_n$ if and only if `INPUT`$(t_1, \ldots, t_n) \vdash_R$ `ACCEPT`.

As an example consider the *Acyclic* predicate on directed graphs — the predicate which is true of a directed graph if that graph has no cycles. It is well known that acyclicity is a polynomial time property of directed graphs. This property has a simple definition using superficial rules with one level of stratification — if a graph is not cyclic then it is acyclic. The above theorem implies that the acyclicity predicate can be defined by superficial rules without any stratification. The unstratified rule set for acyclicity is somewhat complex and rather than give it here we sketch a proof of the above general theorem. The proof is rather technical, and casual readers are advised to skip to the next section.

We only consider predicates of one argument. The proof for predicates with of higher arity is similar. Let $\mathcal{P}$ be a one argument polynomial time predicate on terms, i.e., a predicate on terms such that one can determine in polynomial time in the DAG size of a term t whether or not $\mathcal{P}(t)$ holds. We construct a data base that represents the term t. For each subterm s of t we introduce a data base individual c_s, i.e., a new constant symbol unique to the term s. We have assumed that the predicate $\mathcal{P}$ is only defined on terms constructed from a fixed finite signature, i.e., a fixed finite set of constant and function symbols. We will consider constants to be functions of no arguments. For each function symbol f of n arguments in this finite signature we introduce a database relation Q_f of $n+1$ arguments, i.e., Q_f is a $n+1$-ary predicate symbol. Now for any term t we define Σ_t to be the set of ground formulas of the form $Q_f(c_{f(s_1 \ldots s_n)},\ c_{s_1},\ \ldots,\ c_{s_n})$ where $f(s_1,\ \ldots,\ s_n)$ is a subterm of t (possibly equal to t). The set Σ_t should be viewed as a data base with individuals c_s and relations Q_f. Let Γ be a set of formulas of the form $S(c_s,\ c_u)$ where s and u are subterms of t such that S represents a successor relation on the individuals of Σ_t, i.e., there exists a bijection ρ from the individuals of Σ_t to consecutive integers such that $S(s,\ u)$ is in Γ if and only if $\rho(u) = \rho(s) + 1$. The result of Immerman and Vardi [Immerman, 1986], [Vardi, 1982] implies that for any polynomial time property of the set $\Sigma_t \cup \Gamma$ there exists a datalog program R such that $\Sigma_t \cup \Gamma$ has the given property if and only if $\Sigma_t \cup \Gamma \ \vdash_R \texttt{ACCEPT}$. Since the term t can be easily recovered from the set Σ_t, there must exist a datalog program R such that $\Sigma_t \cup \Gamma \ \vdash_R \texttt{ACCEPT}$ if and only if $\mathcal{P}(t)$. We can assume without loss of generality that no rule in R can derive new formulas involving the data base predicates Q_f.

We now add to the rule set R superficial rules that construct analogues of the formulas in Σ_t and Γ. First we define a mentioned predicate $\texttt{M}$ such that $\texttt{M}(s)$ is provable if and only if s is a subterm of t.

$$\texttt{INPUT}(t) \Rightarrow \texttt{M}(t)$$

$$\texttt{M}(f(x_1,\ \ldots,\ x_n)) \Rightarrow \texttt{M}x_i$$

The second rule is a schema for all rules of this form where f is one of the finite number of function symbols in the signature and x_i is one of the variables $x_1,\ \ldots,\ x_n$. Now we give rules that simulate the formula set Σ_t.

$$\texttt{M}(f(x_1,\ \ldots,\ x_n)) \Rightarrow Q_f(f(x_1,\ \ldots,\ x_n),\ x_1,\ \ldots,\ x_n)$$

Now we write rules that simulate the formula set Γ, i.e., that define a successor relation on the terms in t. We start by defining a simple subterm predicate Su such that $Su(s,\ u)$ is provable if s and u are subterms of t such that s is a subterm of u.

$$\mathtt{M}(x) \Rightarrow Su(x,\ x)$$

$$\mathtt{M}(f(x_1,\ \ldots,\ x_n)), Su(y,\ x_i) \Rightarrow Su(y,\ f(x_1,\ \ldots,\ x_n))$$

.

Next we define a "not equal" predicate NE such that $NE(s,\ u)$ is provable if and only if s and u are distinct subterms of the input t.

$$\begin{array}{rl} \mathtt{M}(f(x_1,\ \ldots,\ x_n)), \mathtt{M}(g(y_1,\ \ldots,\ y_m)) & \Rightarrow \\ NE(f(x_1,\ \ldots,\ x_n),\ g(y_1,\ \ldots,\ y_m)) & \end{array}$$

$$\begin{array}{rl} \mathtt{M}(f(x_1,\ \ldots,\ x_i,\ \ldots\ x_n)), & \\ f(x_1,\ \ldots,\ y_i,\ \ldots\ x_n), & \\ NE(x_i,\ y_i) & \Rightarrow \\ NE(f(x_1,\ \ldots,\ x_i,\ \ldots\ x_n), & \\ f(x_1,\ \ldots,\ y_i,\ \ldots\ x_n)) & \end{array}$$

In the first rule f and g must be distinct function symbols and in the second rule x_i and y_i occur at the same argument position and all other arguments to f are the same in both terms. Next we define a "not in" predicate NI such that $NI(s,\ u)$ if s is not a subterm of u. We only give the rules for constants and functions of two arguments. The rules for functions of other numbers of arguments are similar. In the following rule c must be a constant.

$$NE(s,\ c) \ \Rightarrow NI(s,\ c)$$

$$\begin{array}{rl} NE(z,\ f(x, y)), & \\ NI(z,\ x), & \\ NI(z,\ y) & \Rightarrow NI(z,\ f(x,\ y)) \end{array}$$

Now for any subterm s of the input we simultaneously define a three place "walk" relation $W(s,\ u,\ w)$ and a binary "last" relation $L(s,\ u)$. $W(s,\ u\ w)$ will be provable if s and u are subterms of w and u is the successor of s in a left to right preorder traversal of the subterms of w with elimination of later duplicates. $L(s,\ u)$ will be provable if s is the last term of the left to right preorder traversal of the subterms of u, again with elimination of later duplicates. In these definitions, we also use the auxiliary three place relation W', which can be viewed as "try to conclude W, checking for duplicates". Once again, c must be a constant.

$$
\begin{aligned}
\mathtt{M}(f(x,\ y)) &\Rightarrow W(f(x,\ y),\ x,\ f(x,\ y)) \\
\mathtt{M}(f(x,\ y)),\ W(u,\ v,\ x) &\Rightarrow W(u,\ v,\ f(x,\ y)) \\
\mathtt{M}(f(x,\ y)),\ NI(y,\ x), & \\
L(s,\ x) &\Rightarrow W(s,\ y,\ f(x,\ y)) \\
\mathtt{M}(f(x,\ y)),\ W(u,\ v,\ y) &\Rightarrow W'(u,\ v,\ f(x,\ y)) \\
W'(u,\ v,\ f(x,\ y)), & \\
NI(u,\ x),\ NI(v,\ x) &\Rightarrow W(u,\ v,\ f(x,\ y)) \\
W'(u,\ v,\ f(x,\ y)), & \\
W'(v,\ w,\ f(x,\ y)), & \\
Su(v,\ x) &\Rightarrow W'(u,\ w,\ f(x,\ y)) \\
&\Rightarrow L(c,\ c) \\
\mathtt{M}(f(x,\ y)),\ L(ylast,\ y), & \\
NI(ylast,\ x) &\Rightarrow L(ylast,\ f(x,\ y)) \\
\mathtt{M}(f(x,\ y)),\ Su(y,\ x), & \\
L(xlast,\ x) &\Rightarrow L(xlast,\ f(x,\ y)) \\
L(ylast,\ y),\ Su(ylast,\ x), & \\
NI(y,\ x), & \\
W'(flast,\ ylast,\ f(x,\ y)), & \\
NI(flast,\ f(x,\ y)) &\Rightarrow L(flast,\ f(x,\ y))
\end{aligned}
$$

Finally we define the successor predicate S.

$$\mathtt{INPUT}(z),\ W(x,\ y,\ z) \Rightarrow S(x,\ y)$$

Let R' be the the datalog program R plus all of the above superficial rules. We now have that $\Sigma_t \cup \Gamma \ \vdash_R$ `ACCEPT` if and only if $\mathtt{INPUT}(t) \ \vdash_{R'}$ `ACCEPT` and the proof is complete.

4 CHARACTERIZING P WITH LOCAL RULES

Using the theorem of the previous section one can provide a somewhat different characterization of the complexity class P in terms of local rule sets.

> **Local Rule Set Representation Theorem:** If $\mathcal{P}$ is a polynomial time predicate on first order terms then there exists a local rule set

R such that for any first order terms $t_1, \ldots, t_n$, we have that $\mathcal{P}$ is true on arguments $t_1, \ldots, t_n$ if and only if $\vdash_R P(t_1 \ldots t_n)$ where P is a predicate symbol representing $\mathcal{P}$.

Before giving a proof of this theorem we give a simple example of a local rule set for a polynomial time problem. Any context free language can be recognized in cubic time. This fact is easily proven by giving a translation of grammars into local rule sets. We represent a string of words using a constant symbol for each word and the binary function `CONS` to construct terms that represent lists of words. For each nonterminal symbol A of the grammar we introduce a predicate symbol P_A of two arguments where $P_A(x\ y)$ will indicate that x and y are strings of words and that y is the result of removing a prefix of x that parses as category A. For each production $A \rightarrow c$ where c is a terminal symbol we construct a rule with no antecedents and the conclusion $P_A(\texttt{CONS}(c, x)\ x)$. For each grammar production $A \rightarrow BC$ we have the following inference rule.

$$P_B(x\ y) \wedge P_C(y\ z) \rightarrow P_A(x\ z)$$

Finally, we let P be a monadic predicate which is true of strings generated by the distinguished start nonterminal S of the grammar and add the following rule.

$$P_S(x\ \texttt{NIL}) \Rightarrow P(x)$$

Let R be this set of inference rules. R is a local rule set, although the proof of locality is not entirely trivial. The rule set R also has the property that $\vdash_R P(x)$ if and only if x is a string in the language generated by the given grammar. General methods for analyzing the order of running time of local rule sets can be used to immediately give that these clauses can be run to completion in order n^3 time where n is the length of the input string.[3] We have implemented a compiler for converting local rule sets to efficient inference procedures. This compiler can be used to automatically generate a polynomial time parser from the above inference rules.

We now prove the above theorem for local inference relations from the preceding theorem for superficial rule sets. By the preceding theorem there must exist a superficial rule set R such that for any first order terms $t_1, t_2, \ldots t_k$ we have that $\mathcal{P}(t_1, t_2, \ldots t_n)$ if and only if $\texttt{INPUT}(t_1, t_2, \ldots t_k) \vdash_R \texttt{ACCEPT}$ where `INPUT` is a predicate symbol and `ACCEPT` is a distinguished proposition symbol. For each predicate symbol Q of m arguments appearing in R let Q' be a new predicate symbol of $k + m$ arguments. We define the rule set R' to be the rule set containing the following clauses.

- $\texttt{Input}'(x_1, \ldots x_k, x_1, \ldots x_k)$

[3] An analysis of the order of running time for decision procedures for local inference relations is given in [McAllester, 1993].

- All clauses of the form

$$\begin{array}{r} Q_1'(x_1,\ \ldots\ x_k,\ t_{1,1},\ \ldots\ t_{1,m_1}) \wedge \cdots \wedge \\ Q_n'(x_1,\ \ldots\ x_k,\ t_{n,1},\ \ldots\ t_{n,m_n}) \quad \Rightarrow \\ W'(x_1,\ \ldots\ x_k,\ s_1,\ \ldots\ s_j) \end{array}$$

  where the clause $Q_1(t_{1,1},\ \ldots\ t_{1,m_1}) \wedge \cdots \wedge Q_n(t_{n,1},\ \ldots\ t_{n,m_n}) \Rightarrow W(s_1,\ \ldots\ s_j)$ is in R.

- The clause $\texttt{Accept}'(x_1,\ \ldots\ x_k) \Rightarrow P(x_1,\ \ldots\ x_k)$.

Given the above definition we can easily show that $\vdash_{R'} Q'(t_1,\ \ldots\ t_k,\ s_1,\ \ldots\ s_m)$ if and only if $\texttt{Input}(t_1,\ \ldots\ t_k) \vdash_R Q(s_1,\ \ldots\ s_m)$. Therefore, it follows that $\texttt{Input}(t_1,\ \ldots\ t_k) \vdash_R \texttt{Accept}$ if and only if $\vdash_{R'} P(t_1,\ \ldots\ t_k)$. It remains only to show that R' is local. Suppose that $\Sigma \vdash_{R'} \Phi$. We must show that $\Sigma \vDash_{R'} \Phi$. Let $t_1, \ldots t_k$ be the first k arguments in Φ. Every inference based on R' involves formulas which all have the same first k arguments. Given that $\Sigma \vdash_{R'} \Phi$ we must have that $\Sigma' \vdash_{R'} \Phi$ where Σ' is the set of formulas in Σ that have $t_1, \ldots$ t_k as their first k arguments. Let Σ'' be and Φ' be the result of replacing each formula $Q'(t_1,\ \ldots\ t_k,\ s_1,\ \ldots, s_m)$ by $Q(s_1,\ \ldots, s_m)$. Since $\Sigma' \vdash_{R'} \Phi$ we must have $\{\texttt{Input}(t_1,\ \ldots\ t_k)\} \cup \Sigma'' \vdash_R \Phi'$. But since R is superficial this implies that every term in the derivation underlying $\{\texttt{Input}(t_1,\ \ldots\ t_k)\} \cup \Sigma'' \vdash_R \Phi'$ either appears in some t_i or appears in Σ''. This implies that every term in the derivation appears in either Σ' or Φ. This implies $\Sigma \vDash_{R'} \Phi$.

5 ANOTHER CHARACTERIZATION OF LOCALITY

In this section we give an alternate characterization of locality. This characterization of locality plays an important role in both the definition of bounded local rule sets given in [McAllester, 1993] and in the notion of inductively local rule sets given here.

> **Definition:** A *bounding set* is a set Υ of ground terms such that every subterm of a member of Υ is also a member of Υ.

> **Definition:** A ground atomic formula Ψ is called a *label formula* of a bounding set Υ if every term in Ψ is a member of Υ.

> **Definition:** For any bounding set Υ, we define the inference relation $\vDash_{R,\Upsilon}$ to be such that $\Sigma \vDash_{R,\Upsilon} \Phi$ if and only if there exists a derivation of Φ from Σ such that every formula in the derivation is a label formula of the term set Υ.

We have that $\Sigma \Vdash_R \Phi$ if and only if $\Sigma \Vdash_{R,\Upsilon} \Phi$ where Υ is the set of all terms appearing as subexpressions of Φ or formulas in Σ. The inference relation $\Vdash_{R,\Upsilon}$ can be used to give another characterization of locality. Suppose that R is not local. In this case there must exist some Σ and Φ such that $\Sigma \not\Vdash_R \Phi$ but $\Sigma \vdash_R \Phi$. Let Υ be the set of terms that appear in Σ and Φ. We must have $\Sigma \not\Vdash_{R,\Upsilon} \Phi$. However, since $\Sigma \vdash_R \Phi$ we must have $\Sigma \Vdash_{R,\Upsilon'} \Phi$ for some finite superset Υ' of Υ. Consider "growing" the bounding set one term at a time, starting with the terms that appear in Σ and Φ.

Definition: A *one step extension* of a bounding set Υ is a ground term α that is not in Υ but such that every proper subterm of α is a member of Υ.

Definition: A feedback event for R consists of a finite set Σ of ground formulas, a ground formula Φ, a bounding set Υ containing all terms that appear in Σ and Φ, and a one step extension α of Υ such that $\Sigma \Vdash_{R,\Upsilon\cup\{\alpha\}} \Phi$ but $\Sigma \not\Vdash_{R,\Upsilon} \Phi$.

By abuse of notation, a feedback event will be written as $\Sigma \Vdash_{R,\Upsilon\cup\{\alpha\}} \Phi$.

Lemma([McAllester, 1993]): R is local if and only if there are no feedback events for R.

Proof: First note that if R has a feedback event then R is not local — if $\Sigma \Vdash_{R,\Upsilon\cup\{\alpha\}} \Phi$ then $\Sigma \vdash_R \Phi$ but if $\Sigma \not\Vdash_{R,\Upsilon} \Phi$ then $\Sigma \not\Vdash_R \Phi$. Conversely suppose that R is not local. In there case there is some Σ and Φ such that $\Sigma \not\Vdash_R \Phi$ but $\Sigma \Vdash_{R,\Upsilon} \Phi$ for some finite Υ. By considering a least such Υ one can show that a feedback event exists for R. ■

The concepts of bounded locality and inductive locality both involve the concept of a feedback event. We can define bounded locality by first defining $C_R(\Sigma, \Upsilon)$ to be the set of of formulas Ψ such that $\Sigma \Vdash_{R,\Upsilon} \Psi$. R is bounded local if it is local and there exists a natural number k such that whenever $\Sigma \Vdash_{R,\Upsilon\cup\{\alpha\}} \Psi$ there exists a derivation of Ψ from $C_R(\Sigma,\Upsilon)$ such that every term in the derivation is a member of $\Upsilon \cup \{\alpha\}$ and such that the derivation is no longer than k. As mentioned above, the set of the four basic inference rules for equality is bounded local and there exists a procedure which can recognize the locality of any bounded local rule set. The definition of inductive locality is somewhat more involved and is given in the next section.

6 INDUCTIVE LOCALITY

To define inductive locality we first define the notion of a feedback template. A feedback template represents a set of potential feedback events. We also define a backward chaining process which generates feedback templates from a rule set R. We show that if there exists a feedback event for R then such an event will be found by this backchaining process. Furthermore, we define an "inductive" termination condition on the backchaining process and show that if the backchaining process achieves inductive termination then R is local.

Throughout this section we let R be a fixed but arbitrary set of Horn clauses. The inference relation $\vdash_{R,\Upsilon}$ will be written as $\vdash_{\Upsilon}$ with the understanding that R is an implicit parameter of the relation.

We define feedback templates as ground objects — they contain only ground first order terms and formulas. The process for generating feedback templates is defined as a ground process — it only deals with ground instances of clauses in R. The ground process can be "lifted" using a lifting transformation. Since lifting is largely mechanical for arbitrary ground procedures [McAllester and Siskind, 1991], the lifting operation is only discussed briefly here.

> **Definition:** A *feedback template* consists of a set of ground atomic formulas Σ, a multiset of ground atomic formulas Γ, a ground atomic formula Φ, a bounding set Υ, and a one step extension α of Υ such that Φ and every formula in Σ is a label formula of Υ, every formula in Γ is a label formula of $\Upsilon \cup \{\alpha\}$ that contains α, and such that $\Sigma \cup \Gamma \vdash_{\Upsilon \cup \{\alpha\}} \Phi$.

By abuse of notation a feedback template will be written as $\Sigma, \Gamma \vdash_{\Upsilon \cup \{\alpha\}} \Phi$. Γ is a multiset of ground atomic formulas, each of which is a label formula of $\Upsilon \cup \{\alpha\}$ containing α, and such that the union of Σ and Γ allow the derivation of Φ relative to the bounding set $\Upsilon \cup \{\alpha\}$. A feedback template is a potential feedback event in the sense that an extension of Σ that allows a derivation of the formulas in Γ may result in a feedback event. The requirement that Γ be a multiset is needed for the induction lemma given below. Feedback templates for R can be constructed by backward chaining.

Procedure for Generating a Template for R:

1. Let $\Psi_1 \wedge \Psi_2 \wedge \cdots \wedge \Psi_n \Rightarrow \Phi$ be a ground instance of a clause in R.
2. Let α be a term that appears in the clause but does not appear in the conclusion Φ and does not appear as a proper subterm of any other term in the clause.
3. Let Υ be a bounding set that does not contain α but does contain every term in the clause other than α.
4. Let Σ be the set of antecedents Ψ_i which do not contain α.
5. Let Γ be the set of antecedents Ψ_i which do contain α.

6. Return the feedback template $\Sigma, \Gamma \Vdash_{\Upsilon \cup \{\alpha\}} \Phi$.

We let $\mathcal{T}_0[R]$ to be the set of all feedback templates that can be derived from R by an application of the above procedure. We leave it to the reader to verify that $\mathcal{T}_0[R]$ is a set of feedback templates. Now consider a feedback template $\Sigma, \Gamma \Vdash_{\Upsilon \cup \{\alpha\}} \Phi$. We can construct a new template by backward chaining from $\Sigma, \Gamma \Vdash_{\Upsilon \cup \{\alpha\}} \Phi$ using the following procedure.

Procedure for Backchaining from $\Sigma, \Gamma \Vdash_{\Upsilon \cup \{\alpha\}} \Phi$

1. Let Θ be a member of Γ
2. Let $\Psi_1 \wedge \Psi_2 \wedge \cdots \wedge \Psi_n \Rightarrow \Theta$ be a ground instance of a clause in R that has Θ as its conclusion and such that each Ψ_i is a label formula of $\Upsilon \cup \{\alpha\}$.
3. Let Σ' be Σ plus all antecedents Ψ_i which do not contain α.
4. Let Γ' be Γ minus Θ plus all antecedents Ψ_i which do contain α.
5. Return the template $\Sigma', \Gamma' \Vdash_{\Upsilon \cup \{\alpha\}} \Phi$.

In step 4 of the above procedure, Γ' is constructed using multiset operations. For example, if the multiset Γ contains two occurrences of Θ, then "Γ minus Θ" contains one occurrence of Θ. We need Γ to be a multiset in order to guarantee that backchaining operations commute in the proof of the induction lemma below—in particular, we will use the fact that if a sequence of backchaining operations remove an element Θ of Γ at some point, then there exists a permutation of that sequence of backchaining operations producing the same resulting template, but that removes Θ first.

For any set $\mathcal{T}$ of feedback templates we define $\mathcal{B}[\mathcal{T}]$ to be $\mathcal{T}$ plus all templates that can be derived from an element of $\mathcal{T}$ by an application of the above backchaining procedure. It is important to keep in mind that by definition $\mathcal{B}[\mathcal{T}]$ contains $\mathcal{T}$. We let $\mathcal{B}^n[\mathcal{T}]$ be $\mathcal{B}[\mathcal{B}[\cdots\mathcal{B}[\mathcal{T}]]]$ with n applications of $\mathcal{B}$.

Definition: A feedback template is called *critical* if Γ is empty.

If $\Sigma, \emptyset \Vdash_{\Upsilon \cup \{\alpha\}} \Phi$ is a critical template then $\Sigma \Vdash_{\Upsilon \cup \{\alpha\}} \Phi$. If $\Sigma \not\Vdash_{\Upsilon} \Phi$ then $\Sigma \Vdash_{\Upsilon \cup \{\alpha\}} \Phi$ is a feedback event. By abuse of notation, a critical template $\Sigma, \emptyset \Vdash_{\Upsilon \cup \{\alpha\}} \Phi$ such that $\Sigma \not\Vdash_{\Upsilon} \Phi$ will be called a feedback event. The following lemma provides the motivation for the definition of a feedback template and the backchaining process.

Lemma: There exists a feedback event for R if and only if there exists a j such that $\mathcal{B}^j[\mathcal{T}_0[R]]$ contains a feedback event.

Proof: To prove the above lemma suppose that there exists a feedback event for R. Let $\Sigma \vdash_{\Upsilon \cup \{\alpha\}} \Phi$ be a minimal feedback event for R, i.e., a feedback event for R which minimizes the length of the derivation of Φ from Σ under the bounding set $\Upsilon \cup \{\alpha\}$. The fact that this feedback event is minimal implies that every formula in the derivation other than Φ contains α. To see this suppose that Θ is a formula in the derivation other than Φ that does not involve α. We then have $\Sigma \vdash_{\Upsilon \cup \{\alpha\}} \Theta$ and $\Sigma \cup \{\Theta\} \vdash_{\Upsilon \cup \{\alpha\}} \Phi$. One of these two must be a feedback event — otherwise we would have $\Sigma \vdash_{\Upsilon} \Phi$. But if one of these is a feedback event then it involves a smaller derivation than $\Sigma \vdash_{\Upsilon \cup \{\alpha\}} \Phi$ and this contradicts the assumption that $\Sigma \vdash_{\Upsilon \cup \{\alpha\}} \Phi$ is minimal. Since every formula other than Φ in the derivation underlying $\Sigma \vdash_{\Upsilon \cup \{\alpha\}} \Phi$ contains α, the template $\Sigma, \emptyset \vdash_{\Upsilon \cup \{\alpha\}} \Phi$ can be derived by backchaining. ∎

The above lemma implies that if the rule set is not local then backchaining will uncover a feedback event. However, we are primarily interested in those cases where the rule set is local. If the backchaining process is to establish locality then we must find a termination condition which guarantees locality. Let $\mathcal{T}$ be a set of feedback templates. In practice $\mathcal{T}$ can be taken to be $\mathcal{B}^j[\mathcal{T}_0[R]]$ for some finite j. We define a "self-justification" property for sets of feedback templates and prove that if $\mathcal{T}$ is self-justifying then there is no n such that $\mathcal{B}^n[T]$ contain a feedback event. In defining the self-justification property we treat each template in $\mathcal{T}$ as an independent induction hypothesis. If each template can be "justified" using the set of templates as induction hypotheses, then the set $\mathcal{T}$ is self-justifying.

Definition: We write $\Sigma, \Gamma \vdash_{\mathcal{T},\Upsilon} \Phi$ if $\mathcal{T}$ contains templates

$$\begin{gathered}\Sigma_1, \Gamma_1 \vdash_{\Upsilon \cup \{\alpha\}} \Psi_1 \\ \Sigma_2, \Gamma_2 \vdash_{\Upsilon \cup \{\alpha\}} \Psi_2 \\ \vdots \\ \Sigma_k, \Gamma_k \vdash_{\Upsilon \cup \{\alpha\}} \Psi_k\end{gathered}$$

where each Σ_i is a subset of Σ, each Γ_i is a subset of Γ and $\Sigma \cup \{\Psi_1,\ \Psi_2,\ \ldots\ \Psi_k\} \vdash_{\Upsilon} \Phi$.

Definition: $\mathcal{T}$ is said to *justify* a template $\Sigma, \Gamma \vdash_{\Upsilon \cup \{\alpha\}} \Phi$ if there exists a $\Theta \in \Gamma$ such that for each template $\Sigma', \Gamma' \vdash_{\Upsilon \cup \{\alpha\}} \Phi$ generated by backchaining from $\Sigma, \Gamma \vdash_{\Upsilon \cup \{\alpha\}} \Phi$ by selecting Θ at step 1 of the backchaining procedure we have $\Sigma', \Gamma' \vdash_{\mathcal{T},\Upsilon} \Phi$.

Definition: The set $\mathcal{T}$ is called *self-justifying* if every member of $\mathcal{T}$ is either critical or justified by $\mathcal{T}$, and $\mathcal{T}$ does not contain any feedback events.

Induction Theorem: If $\mathcal{T}$ is self-justifying then no set of the form $\mathcal{B}^n[\mathcal{T}]$ contains a feedback event.

Proof: We must show that for every critical template $\Sigma, \emptyset \vdash_{\Upsilon \cup \{\alpha\}} \Phi$ in $\mathcal{B}^n[\mathcal{T}]$ we have that $\Sigma \vdash_{\Upsilon} \Phi$. The proof is by induction on n. Consider a critical template $\Sigma, \emptyset \vdash_{\Upsilon \cup \{\alpha\}} \Phi$ in $\mathcal{B}^n[\mathcal{T}]$ and assume the theorem for all critical templates in $\mathcal{B}^j[\mathcal{T}]$ for j less than n. The critical template $\Sigma, \emptyset \vdash_{\Upsilon \cup \{\alpha\}} \Phi$ must be derived by backchaining from some template $\Sigma', \Gamma' \vdash_{\Upsilon \cup \{\alpha\}} \Phi$ in $\mathcal{T}$. Note that Σ' must be a subset of Σ. If Γ' is empty then Σ' equals Σ and $\Sigma \vdash_{\Upsilon} \Phi$ because $\mathcal{T}$ is assumed not to contain any feedback events. If Γ' is not empty then, since $\mathcal{T}$ is self justifying, there must exist some Θ in Γ' such that for each template $\Sigma'', \Gamma'' \vdash_{\Upsilon \cup \{\alpha\}} \Phi$ derived from $\Sigma', \Gamma' \vdash_{\Upsilon \cup \{\alpha\}} \Phi$ by backchaining on Θ we have $\Sigma'', \Gamma'' \vdash_{\mathcal{T}, \Upsilon} \Phi$. We noted above that backchaining operations commute. By the commutativity of backchaining steps there exists a backchaining sequence from $\Sigma', \Gamma' \vdash_{\Upsilon \cup \{\alpha\}} \Phi$ to $\Sigma, \emptyset \vdash_{\Upsilon \cup \{\alpha\}} \Phi$ such that the first step in that sequence is a backchaining step on Θ. Let $\Sigma'', \Gamma'' \vdash_{\Upsilon \cup \{\alpha\}} \Phi$ be the template that results from this backchaining step from $\Sigma', \Gamma' \vdash_{\Upsilon \cup \{\alpha\}} \Phi$. Note that Σ'' is a subset of Σ. We must now have $\Sigma'', \Gamma'' \vdash_{\mathcal{T}, \Upsilon} \Phi$. By definition, $\mathcal{T}$ must contain templates

$$\Sigma_1, \Gamma_1 \vdash_{\Upsilon \cup \{\alpha\}} \Psi_1$$

$$\Sigma_2, \Gamma_2 \vdash_{\Upsilon \cup \{\alpha\}} \Psi_2$$

$$\vdots$$

$$\Sigma_k, \Gamma_k \vdash_{\Upsilon \cup \{\alpha\}} \Psi_k$$

such that each Σ_i is a subset of Σ'', each Γ_i is a subset of Γ'', and $\Sigma'' \cup \{\Psi_1, \ \Psi_2, \ \ldots \ \Psi_k\} \vdash_{\Upsilon} \Phi$. Note that each Σ_i is a subset of Σ. Since Γ_i is a subset of Γ'' there must be a sequence of *fewer than* n backchaining steps that leads from $\Sigma_i, \Gamma_i \vdash_{\Upsilon \cup \{\alpha\}} \Psi_i$ to a critical template $\Sigma'_i, \emptyset \vdash_{\Upsilon \cup \{\alpha\}} \Psi_i$. Note that Σ'_i is a subset of Σ. This critical template is a member of $\mathcal{B}^j[\mathcal{T}]$ for j less than n so we have $\Sigma'_i \vdash_{\Upsilon} \Psi_i$ and thus $\Sigma \vdash_{\Upsilon} \Psi_i$. But if $\Sigma \vdash_{\Upsilon} \Psi_i$ for each Ψ_i, and $\Sigma \cup \{\Psi_1, \ \Psi_2, \ \ldots \ \Psi_k\} \vdash_{\Upsilon} \Phi$, then $\Sigma \vdash_{\Upsilon} \Phi$. ■

We now come the main definition and theorem of this section.

Definition: A rule set R is called *inductively local* if there exists some n such that $\mathcal{B}^n[\mathcal{T}_0[R]]$ is self-justifying.

Theorem: There exists a procedure which, given any finite set R of Horn clauses, will terminate with a feedback event whenever R is not local, terminate with "success" whenever R is inductively local, and fail to terminate in cases where R is local but not inductively local.

The procedure is derived by lifting the above ground procedure for computing $\mathcal{B}^n[\mathcal{T}[R]]$. Lifting can be formalized as a mechanical operation on arbitrary nondeterministic ground procedures [McAllester and Siskind, 1991]. In the lifted version the infinite set $\mathcal{B}^j[\mathcal{T}_0[R]]$ is represented by a finite set of "template schemas" each of which consists of a template expression $\Sigma, \Gamma \Vdash_{\Upsilon \cup \{\alpha\}} \Phi$ involving variables plus a set of constraints on those variables.

7 LOCALITY IS UNDECIDABLE

We prove that locality is undecidable by reducing the Halting problem. Let T be a specification of a Turing machine. We first show one can mechanically construct a local rule set R with the property that the machine T halts if and only if there exists a term t such that $\vdash_R H(t)$ where H is a monadic predicate symbol. Turing machine computations can be represented by first order terms and the formula $H(t)$ intuitively states that t is a term representing a halting computation of T.

To prove this preliminary result we first construct a superficial rule set S such that T halts if and only if there exists a term t such that $\texttt{INPUT}(t) \vdash_S H(t)$. The mechanical construction of the superficial rule set S from the Turing machine T is fairly straightforward and is not given here. We convert this superficial rule set S to a local rule set R as follows. For each predicate symbol Q of m arguments appearing in S let Q' be a new predicate symbol of $m+1$ arguments. The rule set R will be constructed so that $\vdash_R Q'(t,\ s_1,\ \ldots\ s_m)$ just in case $\texttt{Input}(t) \vdash_S Q(s_1,\ \ldots\ s_m)$. We define the rule set R to be the rule set containing the following clauses.

- $\texttt{Input}'(x)$
- All clauses of the form

$$\begin{array}{r} Q_1'(x,\ t_{1,1},\ \ldots\ t_{1,m_1}) \wedge \cdots \wedge \\ Q_n'(x,\ t_{n,1},\ \ldots\ t_{n,m_n}) \quad \Rightarrow \\ W'(x,\ s_1,\ \ldots\ s_j) \end{array}$$

  where the clause $Q_1(t_{1,1},\ \ldots\ t_{1,m_1}) \wedge \cdots \wedge Q_n(t_{n,1},\ \ldots\ t_{n,m_n}) \Rightarrow W(s_1,\ \ldots\ s_j)$ is in R.
- The clause $H'(x,\ x) \Rightarrow H(x)$.

Given that $\vdash_R Q'(t,\ s_1,\ \ldots\ s_m)$ if and only if $\texttt{Input}(t) \vdash_S Q(s_1,\ \ldots\ s_m)$ it directly follows that $\texttt{Input}(t) \vdash_S H(t)$ if and only if $\vdash_R H(t)$. So the Turing machine T halts if and only if $\vdash_R H(t)$ for some term t. The proof that the rule set R is local closely follows the proof of the Local Rule Set Representation Theorem proven above.

We have now constructed a local rule set R with the property that T halts if and only if there exists some term t such that $\vdash_R H(t)$. Now let R' be R plus the single clause $H(x) \Rightarrow \texttt{Halts}$ where `Halts` is a new proposition symbol. We claim that R' is local if and only if T does not halt. First note that if T halts then we have $\vdash_{R'}$ `Halts` but $\not\Vdash_{R'}$ `Halts` so R is not local. Conversely, suppose that T does not halt. In this case we must show that R' is local. Suppose that $\Sigma \ \vdash_{R'} \Phi$. We must show that $\Sigma \Vdash_{R'} \Phi$. Suppose Φ is some formula other than `Halts`. In this case $\Sigma \ \vdash_{R'} \Phi$ is equivalent to $\Sigma \ \vdash_R \Phi$. Since R is local we must have $\Sigma \Vdash_R \Phi$ and thus $\Sigma \Vdash_{R'} \Phi$. Now suppose Φ is the formula `Halts`. If `Halts` is a member of Σ then the result is trivial so we assume that `Halts` is not in Σ. Since $\Sigma \ \vdash_{R'}$ `Halts` we must have $\Sigma \ \vdash_{R'} H(c)$ for some term c. To show $\Sigma \Vdash_{R'}$ `Halts` it now suffices to show that c is mentioned in Σ. By the preceding argument we have $\Sigma \Vdash_R H(c)$. Since the rule set R was generated by the construction given above, we have that every inference based on a clause in R is such that that every formula in the inference has the same first argument. This implies that $\Sigma' \Vdash_R H(c)$ where Σ' is the subset of formulas in Σ that have c as a first argument. We have assumed that T does not halt, and thus $\not\vdash_R H(c)$. Hence Σ' must not be empty. So Σ must mention c and since $\Sigma \Vdash_R H(c)$ we have $\Sigma \Vdash_{R'}$ `Halts`.

8 OPEN PROBLEMS

In closing we note some open problems. There are many known examples of rule sets which are not local and yet the corresponding inference relation is polynomial time decidable. In all such cases we have studied there exists a conservative extension of the rule set which is local. We conjecture that for every rule set R such that $\vdash_R$ is polynomial time decidable there exists a local conservative extension of R. Our other problems are less precise. Can one find a "natural" rule set that is local but not inductively local? A related question is whether there are useful machine recognizable subclasses of the local rule sets other than the classes of bounded local and inductively local rule sets?

Acknowledgement

We would like to thank Franz Baader for his invaluable input and discussions. Support for the work described in this paper was provided in part by Mitsubishi 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. Robert Givan is supported by a Fannie and John Hertz graduate fellowship.

References

[Bry, 1990] Francois Bry. Query evaluation in recursive databases: bottom-up and top-down reconciled. *Data and Knowledge Engineering*, 5:289–312, 1990.

[Givan *et al.*, 1991] 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.

[Immerman, 1986] Neal Immerman. Relational queries computable in polynomial time. *Information and Control*, 68:86–104, 1986.

[Jaffar and Lassez, 1987] J. Jaffar and J. L. Lassez. Constraint logic programming. In *Proceedings of POPL-87*, pages 111–119, 1987.

[Knuth, 1975] Donald E. Knuth. Estimating the efficiency of backtrack programs. *Mathematics of Computation*, 29(129):121–136, January 1975.

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

[Mackworth, 1977] A. K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8(1):99–181, 1977.

[McAllester and Siskind, 1991] David Allen McAllester and Jeffrey Mark Siskind. Lifting transformations. MIT Artificial Intelligence Laboratory Memo 1343, 1991.

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

[McAllester, 1993] D. McAllester. Automatic recognition of tractability in inference relations. *JACM*, 40(2):284–303, April 1993.

[Naughton and Ramakrishnan, 1991] Jeff Naughton and Raghu Ramakrishnan. Bottom-up evaluation of logic programs. In Jean-Louis Lassez and Gordon Plotkin, editors, *Computational Logic*. MIT Press, 1991.

[Papadimitriou, 1985] Christos H. Papadimitriou. A note on the expressive power of prolog. *EATCS Bulletin*, 26:21–23, 1985.

[Pearl and Korf, 1987] Judea Pearl and Richard Korf. Search techniques. *Ann. Rev. Comput. Sci.*, 2:451–467, 1987.

[Shostak, 1978] R. Shostak. An algorithm for reasoning about equality. *Comm. ACM.*, 21(2):583–585, July 1978.

[Ullman, 1988] J. Ullman. *Principles of Database and Knowledge-Base Systems.* Computer Science Press, 1988.

[Ullman, 1989] J. Ullman. Bottom-up beats top-down for datalog. In *Proceedings of the Eigth ACM SIGACT-SIGMOD-SIGART Symposium on the Principles of Database Systems*, pages 140–149, March 1989.

[van Hentenryck, 1989] Pascal van Hentenryck. *Constraint Satisfaction in Logic Programming.* MIT Press, 1989.

[Vardi, 1982] M. Vardi. Complexity of relational query languages. In *14th Symposium on Theory of Computation*, pages 137–146, 1982.