Automatic Recognition of Tractability in Inference Relations

DAVID A. MCALLESTER

Massachusetts Institute of Technology, Cambridge Massachusetts

Abstract: A procedure is given for recognizing sets of inference rules that generate polynomial time decidable inference relations. The procedure can automatically recognize the tractability of the inference rules underlying congruence closure. The recognition of tractability for that particular rule set constitutes mechanical verification of a theorem originally proved independently by Kozen and Shostak. The procedure is algorithmic, rather than heuristic, and the class of automatically recognizable tractable rule sets can be precisely characterized. A series of examples of rule sets whose tractability is non-trivial, yet machine recognizable, is also given. The technical framework developed here is viewed as a first step toward a general theory of tractable inference relations.

Categories and Subject Descriptors: F.4.1 [**Mathematical Logic and Formal Languages**]: Mathematical logic — *computational logic*, *mechanical theorem proving*; I.2.3 [**Artificial Intelligence**]: Deduction and Theorem Proving — *deduction*

General Terms: Deduction, Algorithms

Additional Keywords and Phrases: Proof Theory, Machine Inference, Theorem Proving, Automated Reasoning, Polynomial Time Algorithms, Inference Rules, Proof Systems, Mechanical Verification.

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.

Author's Address: MIT Artificial Intelligence Laboratory, 545 Technology Square, Cambridge Mass, 02139, DAM@ai.mit.edu

This paper appeared in JACM, vol. 40, no. 2, April 1993. A postscript electronic source for this paper can be found in ftp.ai.mit.edu:/pub/dam/jacm2.ps. A bibtex reference can be found in internet file ftp.ai.mit.edu:/pub/dam/dam.bib.

1 Introduction

Certain well known algorithms can be viewed as polynomial time decision procedures for inference relations. For example, transitive closure determines whether a formula of the form $x < y$ can be proven from given inequalities and a transitivity axiom. The union-find procedure determines whether an equality $x = y$ can be proven from given equations and the reflexivity, transitivity, and symmetry axioms of equality. The congruence closure procedure determines whether an equality $s = t$ can be proven from a given set of equations and the symmetry, reflexivity, transitivity, and congruence axioms of equality [1]. Each of these algorithms can be viewed as a decision procedure for the inference relation generated by a certain set of inference rules. This paper identifies a general class of "local" rule sets. These rule sets have the desirable property that the generated inference relation is polynomial time decidable.

Consider a set R of inference rules and let $\vdash_R$ be the inference relation generated by R, i.e., the relation such that $\Sigma \vdash_R \Phi$ if there exists a derivation of Φ from Σ using the rules in R. The rule set R is called *local* if whenever $\Sigma \vdash_R \Phi$ there exists a local derivation of Φ from Σ — a derivation is local if every proper subexpression of a formula in the derivation is either a proper subexpression of Φ, a proper subexpression of a member of Σ, or appears as a subexpression of an inference rule in R. One can show that, for any rule set R, one can determine, in polynomial time in the size of Σ and Φ, whether or not there exists a local derivation of Φ from Σ. If R is local, i.e., the existence of a derivation ensures the existence of a local derivation, then the inference relation $\vdash_R$ is polynomial time decideable.

One can easily show that a rule set consisting of a single transitivity rule is local. The same is true of the rule set underlying the union-find procedure, i.e., the reflexivity, symmetry, and transitivity axioms of equality. The rule set underlying congruence closure, i.e., the reflexivity, symmetry, transitivity, and substitution axioms of equality, is also local, although the locality of this rule set is more difficult to prove.

In addition to providing a theoretical framework for the study of a certain class of polynomial time algorithms, local inference relations provide a tool for the construction of general purpose automated reasoning systems. No sound and complete set of inference rules for first order logic can be local (every local relation is polynomial time decidable, but the entailment relation for first order logic is undecidable). However, local inference rules can be used to reduce proof length in first order proof systems. This can be done through the construction of a *Socratic sequent system* [2]. A *sequent* is an expression of the form $\Sigma \vdash \Phi$ where Σ is a set of formulas and Φ is a formula. If $\Sigma \vdash_R \Phi$ then we say that a rule set R *generates* the sequent $\Sigma \vdash \Phi$. A Socratic sequent system consists of two rule sets — a local rule set and a set of sequent rules. A sequent generated by the local set is called "obvious". A proof in a Socratic sequent system is a series of sequents where each sequent is either obvious or is derivable from earlier sequents by one of the sequent rules. Because obvious sequents can abbreviate long derivations, proofs in a Socratic sequent system can be shorter than proofs in more traditional systems. However, because the rule set generating obvious sequents is local, the correctness of a proof can still be verified in polynomial time. Particular Socratic sequent systems for set theory and first order logic are described in [5] and [4]. It appears that the power of the local rule set in a Socratic sequent system can be enhanced through the use of a nonstandard syntax for first order logic. Aspects of the syntax of English under Mantague semantics have been used to construct particularly powerful general purpose local rule sets [2].

Determining the locality of an arbitrary rule set can be difficult — it is not known whether locality, as a property of rule sets, is decidable. The main result of this paper is that a certain subclass of local rule sets — the bounded local rule sets — can be mechanically identified. More precisely, there exists a procedure that, given an arbitrary rule set, will find a counter-example to locality whenever the rule set is not local, construct a proof that the rule set is bounded local whenever it is bounded local, and will fail to terminate in cases where the rule set is local but not bounded local. The rule set consisting of the reflexivity, symmetry, transitivity, and congruence axioms of equality is bounded local so the locality of this rule set can be mechanically recognized by the procedure given here. This amounts to mechanical verification of a theorem given in [3], [8], and [7]. Several novel examples of non-trivial bounded local rule sets are given in this paper, including a rule set based on the syntax of English. An example is also given of a rule set that is local but not bounded local.

Hopefully, the notion of locality described in this paper is a first step toward a more general understanding of tractable rule sets. Several open technical problems and several directions for further research, are discussed at the end of the paper. A better understanding of tractable inference relations will hopefully result in an improved technology for the construction of semi-automated verification systems and a deeper understanding of inference in general.

1 $$\frac{?\Psi}{\neg\neg?\Psi}$$

2 $$\frac{\neg\neg?\Psi}{?\Psi}$$

3 $$\frac{?\Phi}{?\Psi \vee ?\Phi}$$

4 $$\frac{?\Phi}{?\Phi \vee ?\Psi}$$

5 $$\frac{\begin{array}{l} ?\Phi \vee ?\Psi \\ \neg?\Phi \end{array}}{?\Psi}$$

6 $$\frac{\begin{array}{l} ?\Phi \vee ?\Psi \\ \neg?\Psi \end{array}}{?\Phi}$$

7 $$\frac{\begin{array}{l} \neg?\Phi \\ \neg?\Psi \end{array}}{\neg(?\Phi \vee ?\Psi)}$$

8 $$\frac{\neg(?\Phi \vee ?\Psi)}{\neg?\Phi}$$

9 $$\frac{\neg(?\Phi \vee ?\Psi)}{\neg?\Psi}$$

10 $$\frac{\begin{array}{l} ?\Psi \\ \neg?\Psi \end{array}}{F}$$

11 $$\frac{F}{?\Phi}$$

Figure 1: A tractable set of Boolean inference rules

2 Preliminary Definitions

This paper presents a general procedure for recognizing certain cases in which a set of inference rules generates a computationally tractable inference relation. The first step in constructing such a procedure is to precisely define the notion of an "inference rule". Figure 1 gives basic inference rules for the Boolean connectives $\neg$ and $\vee$. In these rules a question mark in front of a symbol indicates a variable that can be replaced by different expressions in different applications of the rule. Variables in inference rules will be called *metavariables* to distinguish them from variables of the underlying language.

Throughout the remainder of this paper we let B (for Boolean) denote the set of inference rules given in figure 1. All Boolean expressions can be written in terms of the two universal connectives $\neg$ and $\vee$. The rule set B expresses some, but not all, of the inferential properties of these connectives. The inference relation generated by these rules is linear time decidable. Yet, if the above inference rules are augmented by a simple case analysis sequent rule, then the rules become complete for Boolean inference.

As another example of a set of inference rules, consider the following rules for equality.

12 $$\frac{?s = ?t}{?t = ?s}$$

13 $$?t = ?t$$

14 $$\frac{\begin{array}{l} ?r = ?s \\ ?s = ?t \end{array}}{?r = ?t}$$

15 $$\frac{\begin{array}{l} ?s_1 = ?t_1 \\ \cdots \\ ?s_n = ?t_n \end{array}}{?f(?s_1, \ldots ?s_n) = ?f(?t_1, \ldots ?t_n)}$$

The rules 12, 13, and 14 express the symmetry, reflexivity, and transitivity properties of equality respectively, while rule 15 expresses the substitutivity of equals for equals. It is well known that congruence closure provides a polynomial time decision procedure for the inference relation generated by these equality rules. The precise notion of inference rule developed here is not general enough to allow for the notation "..." used in rule 15. Fortunately, however, any inference problem involving function symbols of more than two arguments can be converted to an equivalent problem involving function symbols of at most two arguments. For example, a function f of three arguments can be replaced by two functions `pair` and f' such that $f(x,\ y,\ z)$ equals $f'(x,\ \texttt{pair}(y,\ z))$. Without loss of generality, we can replace rule 15 by the following two rules.

15a $\dfrac{?s = ?t}{?f(?s) = ?f(?t)}$ 15b $\dfrac{\begin{array}{l}?s_1 = ?t_1 \\ ?s_2 = ?t_2\end{array}}{?f(?s_1,\ ?s_2) = ?f(?t_1,\ ?t_2)}$

In the remainder of this paper we let E denote the rule set consisting of rules 12, 13, 14, 15a and 15b.

Different metavariables have different syntactic kinds. For example, the metavariables that appear in the Boolean rule set B range over formulas, while the rule set E has metavariables that range over terms and metavariables that range over function symbols. The phrases "formula", "term", and "monadic function" each refer to a particular syntactic kind.

> **Definition:** A *syntactic kind* is either a kind symbol or an expression of the form $\sigma_1 \times \sigma_2 \times \ldots \sigma_n \rightarrow \tau$ where τ and each σ_i are syntactic kinds.
>
> **Definition:** An *expression* is either a constant symbol or metavariable of a given syntactic kind, or an application of the form $f(s_1 \ldots s_n)$ where f is an expression of kind $\sigma_1 \times \ldots \sigma_n \rightarrow \tau$ and each s_i is an expression of kind σ_i. In the latter case the expression $f(s_1 \ldots s_n)$ is an expression of kind τ.

In first order predicate calculus, an ordinary constant symbol is just a constant of kind `term`; a proposition symbol is a constant of kind `formula`; a function symbol of is a constant of kind `term` $\times \ldots$ `term` $\rightarrow$ `term`; and a predicate symbol is a constant of kind `term` $\times \ldots$ `term` $\rightarrow$ `formula`. The Boolean connectives $\neg$ and $\vee$ are constants of kind `formula` $\rightarrow$ `formula` and `formula` $\times$ `formula` $\rightarrow$ `formula` respectively. Quantifier-free predicate calculus is the language generated by a set of constants of type `term`, a set of constants of type `formula`, a set of function symbols, a set of predicate symbols (including equality) and the Boolean connectives. An expression $\circ(e_1, \ldots e_n)$ will sometimes be written as $(\circ\ e_1 \ldots e_n)$ (Lisp notation), and occasionally as $e_1 \circ e_2$ (infix notation).

The above definitions do not allow for quantified expressions. This paper only discusses inference rules that do not involve quantification. Even without quantifiers, a set of rules can still generate an undecidable or intractable inference relation. On the other hand, the presence of quantifiers does not necessarily prevent tractability. Tractable inference relations involving quantification are discussed in [5] and [4]. A more general notion of locality will be needed to construct a procedure for automatically recognizing tractability in rule sets that involve quantification.

Definition: An expression of kind `formula` will be called a *formula.*

Definition: An *inference rule* is an object of the form

$$\frac{\begin{array}{l}\Psi_1\\ \vdots\\ \Psi_n\end{array}}{\Theta}$$

where $\Psi_1 \ldots \Psi_n$ and Θ are all formulas.

Definition: A metavariable substitution is a mapping ρ from metavariables to expressions such that, for any metavariable $?x$, we have that $\rho(?x)$ is an expression of the same kind as $?x$.

Definition: For any metavariable substitution ρ, and any expression s, we define $\rho(s)$ to be the result of replacing each metavariable in s by its image under ρ. For any set of expressions Υ, we define $\rho(\Upsilon)$ to be the set $\{\rho(s) : s \in \Upsilon\}$.

Observation: For any metavariable substitution ρ, and any expression s, $\rho(s)$ is an expression with the same syntactic kind as s.

Definition: A formula Φ is *one-step derivable* from a set of formulas Σ under inference rules R if there exists an inference rule

$$\begin{array}{l}\Psi_1\\ \vdots\end{array}$$

$$\frac{\Psi_n}{\Theta}$$

in R, and a metavariable substitution ρ, such that $\rho(\Psi_1), \ldots \rho(\Psi_n)$ are all members of Σ and $\rho(\Theta)$ equals Φ.

Definition: A *derivation* of Φ from Σ is a sequence of formulas $\Psi_1,\ \Psi_2, \ldots \Psi_n$ such that each Ψ_i is either a member of Σ, or is one-step derivable under R from previous elements of the sequence, and Ψ_n is the formula Φ. If there exists a derivation of Φ from Σ under rule set R then we write $\Sigma\ \vdash_R\ \Phi$.

Note that $\vdash_R$ is the relation generated by R in the standard way.

3 Local Rule Sets

We are interested in finding general properties of a rule set R that guarantee that the corresponding inference relation $\vdash_R$ is polynomial time decidable. One way of doing this is to consider a "restricted" relation $\Vdash_R$ that is explicitly constructed to be polynomial time decidable. This can be done using the following terminology.

Definition: A formula Ψ will be called a *label formula* of a set of expressions Ω if every proper subexpression of Ψ is a member of Ω.

Definition: For any set of formulas Γ and rule set R we define $\Omega(R,\ \Gamma)$ to be the set of all proper subexpressions of formulas in Γ plus all closed (variable-free) proper subexpressions of formulas in R.

Note that, for any finite rule set R and finite formula set Γ, the set $\Omega(R,\ \Gamma)$ is finite. However, any formula constant or formula metavariable is a label formula of any expression set. This implies that any expression set has an infinite set of label formulas. In spite of the infinity of label formulas, however, restricting the inference process to label formulas of a small finite set yields a tractable inference relation.

Definition: We write $\Sigma \vdash\!\dashv_R \Phi$ if there exists some derivation $\Psi_1, \Psi_2, \ldots \Psi_n$ of Φ from Σ under rule set R such that each Ψ_i is a label formula of $\Omega(R, \Sigma \cup \{\Phi\})$.

Tractability Lemma: For any finite rule set R, the relation $\vdash\!\dashv_R$ is polynomial time decidable.

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

The tractability lemma implies that the inference relation generated by a local rule set is polynomial time decidable. The proof of a refined version of the tractability lemma is given in appendix I. As an example, consider the problem of determining whether or not $\Sigma \vdash\!\dashv_E \Phi$ where Φ and each formula in Σ is an equation between first order terms and E is the equality rule set (rules 12, 13, 14, 15a and 15b). The expression set $\Omega(E, \Sigma \cup \{\Phi\})$ consists of the equality symbol plus all first order terms that appear in Σ and Φ. If s and t are terms in $\Omega(E, \Sigma \cup \{\Phi\})$ then the equation $s = t$ is a label formula of $\Omega(E, \Sigma \cup \{\Phi\})$. Let n be the total size of $\Sigma \cup \{\Phi\}$. There are order n^2 equations that are label formulas of $\Omega(E, \Sigma \cup \{\Phi\})$. This implies that one can enumerate, in polynomial time, all label formulas of $\Omega(E, \Sigma \cup \{\Phi\})$ that can be derived from Σ using derivations restricted to label formulas.

The definition of locality does not provide any obvious way of determining if a given rule set is local. Locality of the equality inference rules was originally proved (using different terminology) independently by Kozen [3] and Shostak [8]. Kozen uses a syntactic argument to show that if $\Sigma \vdash_E \Phi$, then $\Sigma \vdash\!\dashv_E \Phi$. The proof is essentially an induction on the length of the derivation used to establish $\Sigma \vdash_E \Phi$. Shostak's proof of the locality of E is semantic. Shostak observes that the relation $\vdash\!\dashv_E$ is clearly sound under the standard semantics for equality. Furthermore, if $\Sigma \not\vdash\!\dashv_E \Phi$, then one can construct a model of Σ in which Φ is false. In other words, the relation $\vdash\!\dashv_E$ is semantically complete. Since $\vdash_E$ is sound, and $\vdash_E$ is at least as strong as

$\vdash\!\!\dashv_E$, the semantic completeness of $\vdash\!\!\dashv_E$ implies that $\vdash\!\!\dashv_E$ is the same as $\vdash_E$. A semantic proof using a simpler model construction was later given by Nelson and Oppen [7].

Semantic proofs of locality are more compact in many cases than syntactic proofs of the same results. However, it seems difficult to generalize semantic proof techniques to the point where they can be used to mechanically recognize a wide class of local rule sets. The following section shows that syntactic techniques can be used to construct a general locality recognition procedure.

4 Syntactic Proofs of Locality

For any finite rule set R, the relation $\vdash\!\!\dashv_R$ is polynomial time decidable. The rule set R is local if the relation $\vdash\!\!\dashv_R$ is the same as the relation $\vdash_R$. A general syntactic approach to proving locality for particular rule sets can be constructed using the following definitions.

Definition: A set of expressions Υ will be called *subexpression closed* if every subexpression of every member of Υ is also a member of Υ.

Definition: Let R be a rule set, Σ a formula set, and let Υ be an expression set that is subexpression closed and that contains $\Omega(R, \Sigma)$ as a subset. The set $C_R(\Sigma, \Upsilon)$ is defined to be the set of formulas Ψ such that there exists a derivation of Ψ from Σ such that every formula appearing in that derivation is a label formula of Υ.

Observation: $\Sigma \vdash\!\!\dashv_R \Phi$ if, and only if, $\Phi \in C_R(\Sigma, \Omega(R, \Sigma \cup \{\Phi\}))$.

Definition: We say that the set $C_R(\Sigma, \Upsilon)$ is *universal* if $C_R(\Sigma, \Upsilon)$ contains all label formulas of Υ.

Lemma: Let R be a fixed rule set. Let Σ be a formula set, let Υ be a subexpression closed set containing $\Omega(R, \Sigma)$. One can determine whether $C_R(\Sigma, \Upsilon)$ is universal in time polynomial in the size of Υ. If $C_R(\Sigma, \Upsilon)$ is not universal, it is finite and can be enumerated in time polynomial in the size of Υ.

The proof of the above lemma is not given here but is similar to the proof of the refined tractability lemma given in appendix I. It is possible to characterize locality in terms of the closure operator C_R rather than the inference relation $\vdash_R$. To do this we need some additional terminology.

Definition: A *one step extension* of a subexpression closed set Υ is an expression α that is not a member of Υ but such that every proper subexpression of α is a member of Υ.

Definition: An *extension event* for a rule set R is a four-tuple $<\alpha, \Psi, \Sigma, \Upsilon>$ such that Υ is subexpression closed and contains $\Omega(R, \Sigma)$, α is a one step extension of Υ, and Ψ is a member of $C_R(\Sigma, \Upsilon \cup \{\alpha\})$.

The letters $\mathcal{E}$, $\mathcal{E}_1$, $\mathcal{E}_2$, etc. are used below to denote extension events. Consider an extension event $<\alpha, \Psi, \Sigma, \Upsilon>$. Note that the formula Ψ may be "old" in the sense that Ψ may be a member of $C_R(\Sigma, \Upsilon)$. Alternatively, Ψ may be "new" in the sense that Ψ is a member of $C_R(\Sigma, \Upsilon \cup \{\alpha\})$ but not a member of $C_R(\Sigma, \Upsilon)$. The lemma given below states that a rule set R is local if, and only if, it is impossible for a new formula to be a label formula of the old set Υ.

Definition: A *feedback event* for a rule set R is an extension event $<\alpha, \Psi, \Sigma, \Upsilon>$ for R where Ψ is a label formula of Υ but not a member of $C_R(\Sigma, \Upsilon)$.

Lemma: A rule set R is local if, and only if, there are no feedback events for R.

Proof: First, suppose there exists a feedback event $\mathcal{E}$ for R with components $<\alpha, \Psi, \Sigma, \Upsilon>$. The fact that Ψ is a member of $C_R(\Sigma, \Upsilon \cup \{\alpha\})$ implies that $\Sigma \vdash_R \Psi$. The fact that $\mathcal{E}$ is a feedback event implies that Ψ is a label formula of Υ but not a member of $C_R(\Sigma, \Upsilon)$. The fact that Ψ is a label formula of Υ implies that Υ contains $\Omega(R, \Sigma \cup \Psi)$. So Ψ must not be a member of $C_R(\Sigma, \Omega(R, \Sigma \cup \{\Psi\}))$ and so $\Sigma \not\vdash\!\!\dashv_R \Psi$. Thus $\vdash_R$ and $\vdash\!\!\dashv_R$ are different and R is not local.

The above argument shows that if R is local then there can be no feedback events for R. We will now show the converse — if there are no feedback events for R then R is local. Suppose there are no feedback events for R. Now consider any Σ and Φ such that $\Sigma \not\vdash\!\!\dashv_R \Phi$. To show that R is local it suffices to show that $\Sigma \not\vdash_R \Phi$. To show $\Sigma \not\vdash_R \Phi$ it suffices to show that for any finite subexpression closed set Υ containing $\Omega(R, \Sigma \cup \Phi)$ we have $\Phi \notin C_R(\Sigma, \Upsilon)$. By assumption we have that $\Phi \notin C_R(\Sigma, \Omega(R, \Sigma \cup \{\Phi\}))$. Now let Υ be any subexpression closed set containing $\Omega(R, \Sigma \cup \{\Phi\})$ such that $\Phi \notin C_R(\Sigma, \Upsilon)$. For any one-step extension α of Υ we have that Φ is not a member of $C_R(\Sigma, \Upsilon \cup \{\alpha\})$ — otherwise the tuple $<\alpha, \Phi, \Sigma, \Upsilon>$ would be a feedback event. By induction, this implies that Φ is not a member of $C_R(\Sigma, \Upsilon)$ for any finite subexpression closed set Υ containing $\Omega(R, \Sigma \cup \{\Phi\})$ and thus $\Sigma \not\vdash_R \Phi$.

The above lemma reduces the problem of determining locality to the problem of determining the existence of feedback events. The locality recognition procedure is based on a general method of proving the non-existence of feedback events. This general method is best introduced using a simple example. Consider the following rules expressing the monotonicity of an operator f.

16	$?t \subseteq ?t$	18	$\dfrac{?s \subseteq ?u}{f(?s) \subseteq f(?u)}$
17	$\dfrac{?r \subseteq ?s \quad ?s \subseteq ?t}{?r \subseteq ?t}$		

Let M (for monotonicity) be this set of three inference rules.[1] We wish to prove the non-existence of feedback events for M. Consider an extension event $<\alpha, \Psi, \Sigma, \Upsilon>$ for rules M. Either Ψ is an "old" formula, i.e., a member of $C_M(\Sigma, \Upsilon)$, or Ψ is provable from old formulas using the above inference rules. It is possible to characterize all the ways of proving a new formula from old formulas using rules M. More specifically, for any extension event $<\alpha, \Psi, \Sigma, \Upsilon>$ for M, one of the following four conditions must hold.

- Ψ is an "old" formula, i.e., a member of $C_M(\Sigma, \Upsilon)$.
- Ψ is the formula $\alpha \subseteq \alpha$.
- α is of the form $f(s)$ and Ψ is a formula of the form $\alpha \subseteq t$ where $C_M(\Sigma, \Upsilon)$ contains the formulas $s \subseteq u$ and $f(u) \subseteq t$.
- α is of the form $f(s)$ and Ψ is a formula of the form $t \subseteq \alpha$ where $C_M(\Sigma, \Upsilon)$ contains the formulas $t \subseteq f(u)$ and $u \subseteq s$.

If an extension event satisfies one of the above conditions then either Ψ is an old formula (the first condition) or Ψ contains α as a proper subexpression (the last three conditions). Thus Ψ is either an old formula, or Ψ is not a label formula of Υ. So no extension event satisfying one of the above conditions can be a feedback event. The problem of proving the non-existence of feedback events for M has now been reduced to the problem of proving that every extension event for M satisfies one of the above four conditions. This can be be done using the following definitions.

> Let R be a rule set, Σ a formula set, Υ a subexpression closed set containing $\Omega(R, \Sigma)$, and let α be a one step extension of Υ.
>
> **Definition:** The set $C_R^{\alpha,0}(\Sigma, \Upsilon)$ is defined to be $C_R(\Sigma, \Upsilon)$. The set $C_R^{\alpha,j+1}(\Sigma, \Upsilon)$ is defined to be $C_R^{\alpha,j}(\Sigma, \Upsilon)$ plus all label formulas of $\Upsilon \cup \{\alpha\}$ that can be derived from $C_R^{\alpha,j}(\Sigma, \Upsilon)$ via a single application of an inference rule in R.

[1] An in-depth analysis of the computational complexity of the relation $\vdash_M$ is given in [6].

Note that

$$C_R(\Sigma, \Upsilon \cup \{\alpha\}) = \bigcup_{j \geq 0} C_R^{\alpha,j}(\Sigma, \Upsilon).$$

Consider a fixed but arbitrary Σ, Υ and α. To show the non-existence of feedback events for M, it suffices to show that every formula Ψ in $C_M(\Sigma, \Upsilon \cup \{\alpha\})$ satisfies one of the above four conditions with respect to Σ, Υ, and α. The four conditions can be viewed as defining four different types of formulas in the set $C_M(\Sigma, \Upsilon \cup \{\alpha\})$. To prove that every formula in $C_M(\Sigma, \Upsilon \cup \{\alpha\})$ is of one of these four types, it suffices to prove, by induction on j, that every formula in $C_M^{\alpha,j}(\Sigma, \Upsilon)$ is of one of these four types. Every formula in $C_M^{\alpha,0}(\Sigma, \Upsilon)$ is an old formula and so is a formula of the first type. Now assume that every formula in $C_M^{\alpha,j}(\Sigma, \Upsilon)$ is of one of the four given types. Under this assumption one can prove that every formula Ψ in $C_M^{\alpha,j+1}(\Sigma, \Upsilon)$ is of one of the given types. The induction step involves a case analysis on the proof rule used to derive an element of $C_M^{\alpha,j+1}(\Sigma, \Upsilon)$ and the types of formulas used as antecedents in the application of that rule.

The method just described for proving locality for the rule set M can be generalized to a mechanical procedure for recognizing locality.

5 The Locality Recognition Procedure

The mechanical locality recognition procedure is not guaranteed to recognize all local rule sets. However, it is possible to precisely characterize the class of rule sets whose locality is mechanically recognizable by our procedure. This precise characterization involves some additional terminology.

> **Definition:** The *rank* of an extension event $<\alpha, \Psi, \Sigma, \Upsilon>$ for a rule set R is the least natural number j such that Ψ is an element of $C_R^{\alpha,j}(\Sigma, \Upsilon)$.
>
> **Definition:** For any natural number k and rule set R we say that R is *k-bounded-local* if R is local and all extension events for R have rank k or less. The rule set R is bounded-local whenever there exists some k such that R is k-bounded-local.

Note that if R is k-bounded-local then $C_R(\Sigma, \Upsilon \cup \{\alpha\})$ is always equal to $C_R^{\alpha,k}(\Sigma, \Upsilon)$. It would seem that bounded-locality is an extremely strong condition on inference rules and that few rule sets would satisfy this condition. However, all of the examples of local inference rules discussed above are bounded-local — the rule sets E and M are 2-bounded-local while B is 1-bounded local. Unfortunately, there are rule sets which are local but not bounded-local. Let I consist of the reflexivity rule (16), transitivity rule (17), plus rules 19, 20, and 21 given below. The rule set I is local but not bounded-local (the proof is left as a non-trivial exercise for the reader).

19 $\quad \cap(?s, ?t) \subseteq ?s$

20 $\quad \cap(?s, ?t) \subseteq ?t$

21
$$\frac{?w \subseteq ?s \qquad ?w \subseteq ?t}{?w \subseteq \cap(?s, ?t)}$$

Given that I is local (although not bounded-local), the refined tractability lemma implies that the generated inference relation is decidable in order n^3 time (the transitivity rule has order 3).

The following two theorems are the main results of this paper.

First Locality Recognition Theorem: For any rule set R and bound k it is possible to determine whether or not R is k-bounded local.

Second Locality Recognition Theorem: There exists a procedure which, given any rule set R, does the following.

- If R is not local then the procedure terminates and outputs a feedback event for R.
- If R is bounded-local, then the procedure terminates and outputs the least k such that R is k-bounded-local plus an enumeration of the possible "types" of extension events.
- If R is local, but not bounded-local, then the procedure fails to terminate.

Consider the proof of locality for the monotonicity rules described in the preceding section. The proof shows that every monotonicity extension event falls into one of four types and that no extension event of these types can be a feedback event. To mechanize this proof technique we need some way to formally represent extension event types. Consider the third monotonicity extension event type given in the preceding section:

- α is of the form $f(s)$ and Ψ is a formula of the form $\alpha \subseteq t$ where $C_M(\Sigma, \Upsilon)$ contains the formulas $s \subseteq u$ and $f(u) \subseteq t$.

The extension events of this type can be characterized by specifying the form of α, the form of Ψ, and certain formulas that must be in $C_R(\Sigma,\ \Upsilon)$. In general, we allow a formal specification of an extension event type to also include a specification of expressions that must be in Υ. A formal specification of an extension event type is a four-tuple $<\alpha', \Psi', \Sigma', \Upsilon'>$ where α' and Ψ' are patterns giving the form of α and Ψ respectively; Σ' is a set of formulas that must be included in $C_R(\Sigma, \Upsilon)$; and Υ' is a set of expressions that must be included in Υ. The patterns α' and Ψ' are just expressions containing metavariables. The above type of monotonicity extension event can be characterized by the following formal four-tuple.

- $<f(?s),\ \ f(?s) \subseteq ?t,\ \ \{?s \subseteq ?u, f(?u) \subseteq ?t\},\ \ \{\subseteq, f, ?s, ?t, f(?u), ?u\}>$

The above four-tuple specifies the class of extension events in which α has the form $f(?s)$, Ψ has the form $\alpha \subseteq ?t$, and $C_R(\Sigma, \Upsilon)$ contains the formulas $?s \subseteq ?u$ and $f(?u) \subseteq ?t$. Let $<\alpha', \Psi', \Sigma', \Upsilon'>$ be the above four-tuple. Note that Υ' has been constructed so that Υ' is a subexpression closed set containing $\Omega(R, \Sigma')$, and α' is a one-step extension of Υ'. In fact, the tuple $<\alpha', \Psi', \Sigma', \Upsilon'>$ satisfies all of the conditions given in the definition of an extension event — this tuple is itself an extension event. In general, an extension event containing metavariables defines an entire class of "instantiations" of that extension event.

Definition: Let $\mathcal{E}$ be an extension event $<\alpha, \Psi, \Sigma, \Upsilon>$ and let $\mathcal{E}'$ be an extension event $<\alpha', \Psi', \Sigma', \Upsilon'>$. We say that $\mathcal{E}$ is an

R-instance of the template $\mathcal{E}'$, or that the template $\mathcal{E}'$ *R-covers* the extension event $\mathcal{E}$, if there exists a metavariable substitution ρ satisfying the following conditions.

- $\rho(\alpha') = \alpha$
- $\rho(\Psi') = \Psi$
- $\rho(\Sigma') \subseteq C_R(\Sigma, \Upsilon)$
- $\rho(\Upsilon') \subseteq \Upsilon$

We say that a template set T_1 R-covers an event set T_2 if every member of T_2 is R-covered by some member of T_1.

I will often say "covers" or "instance" rather than "R-covers" or "R-instance" respectively when the rule set is clear from context. I will use the term "event template", or just "template", rather than the term "extension event" to describe extension events that are being used as templates or schemas for a whole class of extension events. The following lemmas state useful properties of extension event templates.

Let $\mathcal{E}$ be $<\alpha, \Psi, \Sigma, \Upsilon>$ and let $\mathcal{E}'$ be $<\alpha', \Psi', \Sigma', \Upsilon'>$ such that $\mathcal{E}$ is an instance of $\mathcal{E}'$ by virtue of the metavariable substitution ρ.

Lemma: The set $\rho(C_R(\Sigma', \Upsilon'))$ is a subset of $C_R(\Sigma, \Upsilon)$.

Proof: Consider any formula Θ in $C_R(\Sigma', \Upsilon')$. We must show that $\rho(\Theta)$ is a member of $C_R(\Sigma, \Upsilon)$. Consider a derivation D of Θ from Σ' such that all formulas in the derivation are label formulas of Υ'. Let $\rho(D)$ be the derivation derived from D by replacing each expression in D by its image under the substitution ρ. $\rho(D)$ is a derivation of $\rho(\Theta)$ from $\rho(\Sigma')$. Furthermore, since $\rho(\Upsilon')$ is a subset of Υ, every formula in $\rho(D)$ is a label formula of Υ. Since every element of $\rho(\Sigma')$ is in $C_R(\Sigma, \Upsilon)$, we must have that $\rho(\Theta)$ is also in $C_R(\Sigma, \Upsilon)$.

Lemma: For each natural number j, the set $\rho(C_R^{\alpha',j}(\Sigma', \Upsilon'))$ is a subset of $C_R^{\alpha,j}(\Sigma, \Upsilon)$.

Proof: The proof is by induction on j. The previous lemma establishes the result for $j = 0$. Now assume that the result holds for j and consider $j+1$. Let Θ be any formula in $C_R^{\alpha',j+1}(\Sigma', \Upsilon')$. We must show that $\rho(\Theta)$ is in $C_R^{\alpha,j+1}(\Sigma, \Upsilon)$. Θ is derivable, via a single inference rule, from some formulas $\Phi_1 \ldots \Phi_n$ in $C_R^{\alpha',j}(\Sigma', \Upsilon')$. By the induction hypothesis $\rho(\Phi_1) \ldots \rho(\Phi_n)$ are in $C_R^{\alpha,j}(\Sigma, \Upsilon)$. But $\rho(\Theta)$ is derivable from $\rho(\Phi_1) \ldots \rho(\Phi_n)$ and $\rho(\Theta)$ is a label formula of $\Upsilon \cup \{\alpha\}$. Thus $\rho(\Theta)$ is in $C_R^{\alpha,j+1}(\Sigma, \Upsilon)$.

Lemma: The rank of $\mathcal{E}$ is less than or equal to the rank of $\mathcal{E}'$.

Proof: Let j be the rank of $\mathcal{E}'$. The formula Ψ' is in $C_R^{\alpha',j}(\Sigma', \Upsilon')$. By the above lemma, $\rho(\Psi')$ must be in $C_R^{\alpha,j}(\Sigma, \Upsilon)$. Since $\rho(\Psi')$ equals Ψ, the extension event $\mathcal{E}$ must have rank j or less.

Lemma: If $\mathcal{E}'$ is not a feedback event then $\mathcal{E}$ is not a feedback event.

Proof: Since $\mathcal{E}'$ is not a feedback event the formula Ψ' is either a member of $C_R(\Sigma', \Upsilon')$ or is not a label formula of Υ'. In the first case, the above lemma implies that $\rho(\Psi')$, and hence Ψ, is a member of $C_R(\Sigma, \Upsilon)$. Now suppose that Ψ is not a label formula of Υ. Since Ψ' is a label formula of $\Upsilon' \cup \{\alpha'\}$ but not a label formula of Υ', the expression α' must be a proper subexpression of Ψ'. But this implies that $\rho(\alpha')$ is a proper subexpression of $\rho(\Psi')$ and thus α is a proper subexpression of Ψ. This implies that Ψ is not a label formula of Υ and thus $\mathcal{E}$ is not a feedback event.

The locality recognition procedure takes a bounded-local rule set R and automatically constructs a proof of the locality of R using the same technique as that used above in proving the locality of the rule set M. The proof of locality of M involved showing that every extension event for M is an instance of one of four specific templates. In order to construct an analogous proof for an arbitrary bounded-local rule set R, the procedure must generate a finite set of extension event templates, specific to the rule set R, and must show that this finite set of extension event templates covers all extension events for R. The recognition procedure uses a single process to both generate the extension event templates and to prove that the generated templates cover all extension events. This process starts with a set of "null" templates and generates new templates by iteratively passing existing templates through the inference rules.

Definition: The *null template of kind* τ is $<?\alpha, ?\Psi, \{?\Psi\}, \{\}>$ where $?\alpha$ is a metavariable of kind τ.

Observation: An extension event has rank 0 if, and only if, it is an instance of some null template.

Without loss of generality we can consider only the syntactic kinds used in the inference rules, so we need only consider a finite set of null templates. The following lifting lemma states the existence of a procedure for passing templates through inference rules.

Lifting Lemma: Let R be a finite rule set and let T be a finite template set such that T covers all extension events for R of rank j or less. It is possible to compute a finite template set $R(T)$ that covers all extension events of rank $j+1$ or less.

The proof of the lifting lemma, and a procedure for computing $R(T)$, is given in appendix II.

Definition: For any rule set R, define $T_0(R)$ to be the set of null templates and define $T_{j+1}(R)$ to be $T_j(R) \cup R(T_j(R))$.[2]

Observation: $T_j(R)$ covers every extension event for R with rank j or less.

Lemma: R is local if, and only if, there is no j such that $T_j(R)$ contains a feedback event.

Proof: Suppose there exists some feedback event for R. This extension event must have some finite rank j and must be covered by some element of $T_j(R)$. Templates that are not feedback events

[2]A "more efficient" definition states that $T_{j+1}(R)$ equals $T_j(R)$ plus those elements of $R(T_j(R))$ not already covered by some element of $T_j(R)$.

can not cover feedback events, so $T_j(R)$ must contain a feedback event.

Lemma: R is j-bounded-local if, and only if, $T_j(R)$ does not contain any feedback events, $T_j(R)$ covers $R(T_j(R))$, and every member of $T_j(R)$ has rank j or less.[3]

Proof: First suppose $T_j(R)$ covers $R(T_j(R))$. Since covering is transitive, this implies that $T_j(R)$ covers all extension events of rank $j+1$ or less. But, by the same argument, this implies that $T_j(R)$ covers all extension events of rank $j+2$ or less. In fact, $T_j(R)$ covers all extension events. If, in addition, $T_j(R)$ does not contain any feedback events, then there can be no feedback events for R and R must be local. If all templates in $T_j(R)$ have rank j or less then, since no template can cover an extension event of greater rank, all extension events for R must have rank j or less.

Now suppose that R is j-bounded local. Since there are no feedback events for R, $T_j(R)$ must not contain a feedback event. Since every extension event has rank j or less, $T_j(R)$ must cover all extension events. This implies that $T_j(R)$ covers $R(T_j(R))$. Finally, since all extension events for R have rank j or less, every template in $T_j(R)$ must have rank j or less.

The recognition theorems follow directly from the above lemmas. A procedure based on the above lemmas has been implemented and all claims made in this paper for the bounded-locality of particular rule sets have been mechanically verified.

6 Additional Examples

This section presents additional examples of bounded-local rule sets. These examples are intended to support the hypothesis that bounded-local rule sets

[3]The most natural procedure for constructing $R(T)$ ensures that every extension event in $T_j(R)$ has rank j or less.

are quite common and easily constructed. The examples are also intended to support the hypothesis that recognizing locality is usually difficult.

Three examples of local rule sets are discussed above — a Boolean rule set B, an equality rule set E, and a monotonicity rule set M. Additional examples of bounded-local rule sets can be derived by considering various unions of these rule sets, e.g., $M \cup B$ or $M \cup B \cup E$. It turns out that all such unions are bounded-local. In general, however, a union of local rule sets need not be local. Similarly, a subset of a local rule set need not be local. The locality of the various combinations of B, E, and M has been determined through mechanical verification. Except for the rule set B, which is 1-bounded-local, all combinations of rule sets B, E, and M are 2-bounded-local.

The next example is a rule set based on the syntactic structure of English under Montague semantics. The rules involve expressions of three different syntactic kinds: class expressions, specified noun phrases, and formulas. The expressions can be given a simple semantics in which each class expression denotes a set, each formula denotes a truth value, and each specified noun phrase denotes an operator that maps sets to truth values (a second order predicate). For example if x denotes a set, then (`every` x) is a specified noun phrase and denotes a second order predicate that is true of a set y just in case the set x is a subset of the set y — a formula of the form ((`every` x) y) is true just in case $x \subseteq y$. Similarly, a formula of the form ((`some` x) y) is true just in case some element of the set x is a member of the set y, i.e., just in case $x \cap y$ is non-empty. For any binary relation R, and class expression C, we let (R (`some` C)) and (R (`every` C)) be class expressions. For example, let `kissed` be a binary relation and let `man` and `woman` be class expression constants. We have the class expressions (`kissed (some woman)`) and (`kissed (every woman)`) and we have formulas such as ((`every man`) (`kissed (some woman)`)), or alternatively, ((`some man`) (`kissed (every woman)`)). The meaning of expressions of the form (R (`some` C)) and (R (`every` C)) can be defined so that the above formulas have a natural meaning. The inference rules shown in figure 2 are sound under this natural semantics.

Let N (for Natural) be the set of inference rules given in figure 2. It runs

22 $\quad ((\texttt{every}\ ?x)\ ?x)$

23 $$\frac{((\texttt{every}\ ?x)\ ?y) \qquad ((\texttt{every}\ ?y)\ ?z)}{((\texttt{every}\ ?x)\ ?z)}$$

24 $$\frac{((\texttt{some}\ ?x)\ ?y)}{((\texttt{some}\ ?y)\ ?x)}$$

25 $$\frac{((\texttt{some}\ ?x)\ ?y)}{((\texttt{some}\ ?x)\ ?x)}$$

26 $$\frac{((\texttt{some}\ ?x)\ ?y) \qquad ((\texttt{every}\ ?y)\ ?z)}{((\texttt{some}\ ?x)\ ?z)}$$

27 $$\frac{((\texttt{every}\ ?x)\ ?y)}{((\texttt{every}\ (?R\ (\texttt{some}\ ?x)))\ (?R\ (\texttt{some}\ ?y)))}$$

28 $$\frac{((\texttt{every}\ ?x)\ ?y)}{((\texttt{every}\ (?R\ (\texttt{every}\ ?y)))\ (?R\ (\texttt{every}\ ?x)))}$$

29 $$\frac{((\texttt{some}\ ?x)\ ?y)}{((\texttt{every}\ (?R\ (\texttt{every}\ ?x)))\ (?R\ (\texttt{some}\ ?y)))}$$

Figure 2: A Natural Rule Set

out that the rule set N is not local. However, the notion of locality can be used to construct a polynomial time decision procedure for the relation $\vdash_N$. First, to see that N is not local, note that by combining inference rules 25 and 30 we get

$$((\texttt{some}\ C)\ S)\ \vdash_N\ ((\texttt{every}\ (R\ (\texttt{every}\ S)))\ (R\ (\texttt{some}\ C))).$$

However, the derivation the expression $(\texttt{some}\ S)$ occurs as a proper subexpression in a formula in the derivation and $(\texttt{some}\ S)$ does not appear in the statement of the inference problem so we have

$$((\texttt{some}\ C)S)\ \not\vdash_N\ ((\texttt{every}\ (R\ (\texttt{every}\ S)))\ (R\ (\texttt{some}\ C))).$$

In spite of the fact that N is not local, the locality recognition procedure can be used to show that the relation $\vdash_N$ is polynomial time decidable. Let N' be the rule set constructed from N by replacing formulas of the form $((\texttt{every}\ C)\ S)$ and $((\texttt{some}\ C)\ S)$ by formulas of the form $(\texttt{is-every}\ C\ S)$ and $(\texttt{is-some}\ C\ S)$ respectively. For any formula Φ and set of formulas Σ we similarly define Φ' and Σ'. We now have that $\Sigma\ \vdash_N\ \Phi$ if, and only if, $\Sigma' \vdash_{N'} \Phi'$. It now suffices to show that $\vdash_{N'}$ is polynomial time decidable. But one can machine-verify the fact that N' is 4-bounded-local. The refined tractability lemma then implies that there exists an order n^3 decision procedure for the relation $\vdash_{N'}$.

7 Discussion

Several technical questions remain unanswered. First, although the above procedure shows that k-bounded locality is decidable for arbitrary rule sets, it is not known whether (unbounded) locality is decidable. Another open question regards inference relations rather than rule sets. An inference relation will be called local if it is generated by some local rule set. It is possible for a rule set R to be non-local and yet the relation $\vdash_R$ is generated by some other rule set R' where R' is local — so the relation $\vdash_R$ can be local even though R is not. Given a rule set R can one determine if the relation $\vdash_R$ is local? We will say that a relation is k-bounded-local if it is generated by some k-bounded-local rule set. Can one determine if $\vdash_R$ is k-bounded-local?

It seems likely that the definition of locality can be improved. Consider the "natural" rule set N given above. This rule set is not local but a trivial syntactic transformation yields an essentially equivalent, but bounded-local, rule set N'. In general, replacing formulas of the form $(P\ s\ t)$ by formulas of the form $((P\ s)\ t)$, i.e., Currying the predicate P, can transform a local rule set into one that is not local. The fact that locality is sensitive to such trivial syntactic changes suggests that a more robust notion of locality is possible. Ideally, a definition of locality should have the property that locality of an arbitrary rule set is decidable, locality of a rule set guarantees that the generated inference relation is polynomial time decidable, and the class of local relations is closed under certain simple syntactic transformations such as Currying.

An improved notion of locality might also lead to improvements in the refined tractability lemma. Ideally, one should be able to mechanically recognize that the Boolean inference relation is linear time decidable rather than quadratic as the tractability lemma would indicate. Similarly, the single rule of transitivity generates a relation that is decidable in linear time, rather than cubic. In both of these examples the more efficient algorithm can be viewed as a tighter restriction on forward chaining inference. Automatic construction of a fast congruence closure algorithm is perhaps too much to expect — fast congruence closure is not simply a matter of tightening the restriction on forward chaining inference. However, it may be reasonable to invoke special case mechanisms for rule sets that include the equality rules as a subset. Hopefully, the framework presented in this paper is only a first step toward a more powerful, and more general, theory of tractable inference relations.

APPENDIX I: The Tractability Lemma

The tractability lemma states that for any finite rule set R, the relation $\vdash_R$ is polynomial time decidable. The statement of the tractability lemma can be refined to give a useful upper bound on the order of the polynomial involved. This refinement requires some additional terminology.

> **Definition:** An inference rule r will be said to *have order* k if there exist expressions $e_1 \ldots e_k$, such that each e_i is either a metavariable or a proper subexpression of some formula in the rule r, and such that every metavariable that appears in r also appears in some e_i.

For example, the rule

15b
$$\frac{\begin{array}{l} ?s_1 = ?t_1 \\ ?s_2 = ?t_2 \end{array}}{?f(?s_1,\ ?s_2) = ?f(?t_1,\ ?t_2),}$$

has order two because the two expressions $?f(?s_1,\ ?s_2)$ and $?f(?t_1,\ ?t_2)$ satisfy the requirements of the above conditions. Note that the rule does not have order one because the equation $?f(?s_1,\ s_2) = ?f(?t_1,\ ?t_2)$ is not a *proper* subexpression of a formula in the rule. Similarly, the rule

7
$$\frac{\begin{array}{l} \neg ?\Phi \\ \neg ?\Psi \end{array}}{\neg(?\Phi \vee ?\Psi)}$$

has order one, while the rule

3
$$\frac{?\Phi}{?\Psi \vee ?\Phi}$$

has order two.

> **Refined Tractability Lemma:** For a fixed finite rule set R, it is possible to determine whether $\Sigma \vdash_R \Phi$ in order n^k time where n is the total size of Σ and Φ and all rules in R have order k or less.

Proof: For the purposes of this proof, a rule set R will be called *normal* if, for every rule r in R, every metavariable in r appears as a proper subexpression of some formula in r. We first reduce the problem of determining whether $\Sigma \vdash_R \Phi$ to the the problem of determining whether $\Sigma \vdash_R \Phi$ in the case where R is normal. If Σ is empty, and no inference rule in R has an empty set of antecedents, then $\Sigma \not\vdash_R \Phi$. Thus we can assume without loss of generality that either Σ is non-empty or some rule in R has no antecedents. Consider a rule r and a metavariable $?\Psi$ that appears in r but does not appear as a proper subexpression of any formula in r. The only place $?\Psi$ can appear in r is as an antecedent or conclusion. If $?\Psi$ is both an antecedent and a conclusion, then r can be removed from the rule set without affecting the relation $\vdash_R$. If $?\Psi$ is an antecedent but not a conclusion, then the above comments about Σ and R imply that the rule r can be replaced by the rule r' in which the antecedent $?\Psi$ has been removed. If $?\Psi$ is the conclusion of r, but is not an antecedent of r, then we replace r by the rule r' derived from r by replacing the conclusion $?\Psi$ with a new formula constant F. Let R' be the rule set derived from R by making all such removals and replacements. We now have that $\Sigma \vdash_R \Phi$ just in case $\Sigma \vdash_{R'} \Phi$ or $\Sigma \vdash_{R'} F$. Furthermore, R' is a normal rule set and all rules in R' have order k or less.

Now, without loss of generality, we can assume that R is a normal rule set. Let Υ be the set $\Omega(R, \Sigma \cup \{\Phi\})$. For a fixed rule set R, the set Υ has order n elements. We have that $\Sigma \vdash_R \Phi$ just in case there exists a derivation $\Psi_1, \Psi_2 \ldots \Psi_n$ of Φ from Σ under R such that each Ψ_i is a label formula of Υ. Let r be an inference rule in R. For any metavariable substitution ρ we let $\rho(r)$ be the rule derived from ρ by replacing each metavariable in r by its image under ρ. Since R is normal, we need only consider those instances $\rho(r)$ where ρ maps every metavariable in r to a member of Υ. Let $e_1 \ldots e_j$ be a set of expressions that satisfy the conditions of the definition of r being order j.

Each e_i is either a metavariable or a proper subexpression of some formula in r. This implies that we need only consider those instances $\rho(r)$ where ρ is a substitution such that $\rho(e_1) \ldots \rho(e_j)$ are all members of Υ. Since every metavariable in r appears in some e_i, the set of all such instances $\rho(r)$ can be computed by matching the expressions $e_1 \ldots e_j$ against elements of Υ. For a fixed rule r (independent of the size n), the set of all possible matches of $e_1 \ldots e_j$ to elements of Υ can be computed in order n^j time. The restriction that each $\rho(e_i)$ be an element of Υ does not guarantee that the conclusion and antecedents of $\rho(r)$ are label formulas of Υ. Let $I(r)$ be the set of all such instances $\rho(r)$ such that the conclusion and all the antecedents of $\rho(r)$ are label formulas of Υ. The set $I(r)$ can be computed in order n^j time. Let $I(R)$ be the union of the sets $I(r)$ for rules r in R. The set $I(R)$ can be computed in order n^k time. We now have that $\Sigma \vDash_R \Phi$ just in case Φ can be derived from Σ under the rules $I(R)$ by purely propositional reasoning (we need not consider further substitution into the rules in $I(R)$). This is equivalent to determining if a given proposition symbol can be derived from a set of proposition symbols using a set of propositional Horn clauses. The existence of such a derivation can be determined in time proportional to the total size of the set of propositional Horn clauses. Since $I(R)$ can be computed in order n^k time, its total size is order n^k.

APPENDIX II: The Lifting Lemma

The lifting lemma can be stated as follows.

Lifting Lemma: Let R be a finite rule set and let T be a finite template set such that T covers all extension events for R of rank j or less. It is possible to compute a finite template set $R(T)$ that covers all extension events of rank $j+1$ or less.

The template set $R(T)$ can be constructed from R and T as follows.

Definition: Let R be a set of inference rules and let T be a set of extension event templates such that any individual metavariable appears in at most one rule or template (the rules and templates have all been resolved apart). We define $R(T)$ to be the set of extension event templates that can be generated non-deterministically by the following procedure.

1. Let

   $$\begin{array}{l} \Theta_1 \\ \vdots \\ \Theta_n \\ \hline \Phi \end{array}$$

   be a rule in R and let $<\alpha_1, \Psi_1, \Sigma_1, \Upsilon_1> \ldots <\alpha_n, \Psi_n, \Sigma_n, \Upsilon_n>$ be templates in T such that there exists a metavariable substitution ρ such that $\rho(\Theta_i) = \rho(\Psi_i)$ for $1 \leq i \leq n$ and $\rho(\alpha_i) = \rho(\alpha_j)$ for $1 \leq i \leq j \leq n$.

2. Let ρ be the most general substitution satisfying the above conditions.

3. Let α be the expression $\rho(\alpha_i)$ for any α_i.

4. Let $\{s_1 \ldots s_k\}$ be the set of all *top level* proper subexpressions of $\rho(\Phi)$, i.e., proper subexpressions of $\rho(\Phi)$ that are not proper subexpressions of any (larger) proper subexpression of $\rho(\Phi)$.

5. Let $\{u_1 \ldots u_m\}$ and $\{w_1 \ldots w_p\}$ be disjoint sets whose union is $\{s_1 \ldots s_k\}$ and such that there exists a substitution ρ' such that $\rho'(u_i) = \rho'(\alpha)$ for $1 \leq i \leq m$.

6. Let ρ' be the most general substitution satisfying the above conditions for the selected expressions $u_1 \ldots u_m$.

7. Let α' be $\rho'(\alpha)$.

8. Let Φ' be $\rho'(\rho(\Phi))$.

9. Let Σ' be $\rho'(\rho(\bigcup_{1 \leq i \leq n}(\Sigma_i)))$.

10. Let Υ' be the least subexpression closed set containing all of the following:

    (a) All closed (variable-free) proper subexpressions of formulas that appear in the rule set R.

    (b) All proper subexpressions of Σ'

    (c) All sets of the form $\rho'(\rho(\Upsilon_i))$ for $1 \leq i \leq n$

    (d) All proper subexpressions of α'.

    (e) The expressions $\rho'(w_1) \ldots \rho'(w_p)$.

11. If α' is not a member of Υ' then output $<\alpha', \Phi', \Sigma', \Upsilon'>$.

Lemma: If T is a set of extension event templates for R then $R(T)$ is also a set of extension event templates for R and if all templates in T have rank j or less then all templates in $R(T)$ have rank $j + 1$ or less.

Proof: Let $<\alpha', \Phi', \Sigma', \Upsilon'>$ be some tuple in $R(T)$. An extension event template is just an extension event (which may contain metavariables) so we have to show that this tuple satisfies all of the conditions for being an extension event for R. Step 10 ensures that Υ' is subexpression closed and steps 10a and 10b ensure that Υ' contains $\Omega(R, \Sigma')$. Step 10d, and the condition in step 11 that α' not be in Υ', ensure that α' is a one step extension of Υ'. Steps 3, 4, 5, 6, and 10e ensure that every immediate subexpression of Φ' is either a member of Υ' or is equal to α'. This guarantees that Φ' is a label formula of $\Upsilon' \cup \alpha'$.

We must also show that the formula Φ' is a member of $C_R^{\alpha', j+1}(\Sigma', \Upsilon')$. Let $<\alpha_1, \Psi_1, \Sigma_1, \Upsilon_1> \ldots <\alpha_n, \Psi_n, \Sigma_n, \Upsilon_n>$ be the templates in T selected at step

1 of the procedure. Let ρ'' be the substitution that maps every expression e to $\rho'(\rho(e))$ where ρ and ρ' are the substitutions constructed in steps 2 and 6 respectively. The construction of the substitution ρ' ensures that Φ' is derivable from $\rho''(\Psi_1)\ldots\rho''(\Psi_n)$ via a single inference rule. For each Ψ_i we have that Ψ_i is a member of $C_R^{\alpha,j}(\Sigma_i,\Upsilon_i)$. Now we show that $\rho''(C_R(\Sigma_i,\Upsilon_i))$ is a subset of $C_R(\Sigma',\Upsilon')$. Let Θ be any formula in $C_R(\Sigma_i,\Upsilon_i)$ we must show that $\rho''(\Theta)$ is a member of $C_R(\Sigma',\Upsilon')$. Let D be a derivation of Θ from Σ_i such that every formula in D is a label formula of Υ_i. $\rho''(D)$ is a derivation of $\rho''(\Theta)$ from $\rho''(\Sigma)$. Furthermore, since every proper subexpression of every formula in D is a member of Υ_i, every proper subexpression of every formula in $\rho''(D)$ is a member of Υ'. Thus $\rho''(\Theta)$ is a member of $C_R(\Sigma',\Upsilon')$, and $rho''(C_R(\Sigma_i,\Upsilon_i))$ is a subset of $C_R(\Sigma',\Upsilon')$. Since Ψ_i is a member of $C_R^{\alpha_i,j}(\Sigma_i,\Upsilon_i)$, there exists a depth j derivation of $\rho''(\Psi_i)$ from $\rho''(C_R(\Sigma_i,\Upsilon_i))$. Since $\rho''(C_R(\Sigma_i,\Upsilon_i)$ is a subset of $C_R(\Sigma',\Upsilon')$, there exists a depth j derivation of $\rho''(\Psi_i)$ from $C_R(\Sigma',\Upsilon')$. An argument similar to the one above shows that every formula in this derivation is a label formula of $\Upsilon'\cup\{\alpha'\}$ and thus $\rho''(\Psi_i)$ is a member of $C_R^{\alpha',j}(\Sigma',\Upsilon')$. But Φ' is derivable in one step from $\rho''(\Psi_1)\ldots\rho''(\Psi_n)$ and thus Φ' must be a member of $C_R^{\alpha',j+1}(\Sigma',\Upsilon')$. $\Box$

Lemma: If T is a set of templates that covers all extension events with rank j or less, then $R(T)$ covers all extension events of rank $j+1$.

Proof: Let $\mathcal{E}''$ be an extension event $<\alpha'',\Phi'',\Sigma'',\Upsilon''>$ of rank $j+1$ (the use of double primes allows the names used in this proof to agree with the names used in the above procedure). By definition, Φ'' is a member of $C_R^{\alpha,j+1}(\Sigma,\Upsilon)$ but not a member of $C_R^{\alpha,j}(\Sigma,\Upsilon)$. This implies that there exist formulas $\Psi''_1\ldots\Psi''_n$ in $C_R^{\alpha'',j}(\Sigma'',\Upsilon'')$ and an inference rule r of the form

$$\frac{\begin{array}{l}\Theta_1\\ \vdots\\ \Theta_n\end{array}}{\Phi}$$

in R that allows Φ'' to be derived from $\Psi''_1\ldots\Psi''_n$ by applying a substitution σ to the inference rule. We have that $\sigma(\Theta_i)=\Psi''_i$ and $\sigma(\Phi)=\Phi''$. Let $\mathcal{E}''_1\ldots\mathcal{E}''_n$

be the extension events $<\alpha'', \Psi''_1, \Sigma'', \Upsilon''> \ldots <\alpha'', \Psi''_n, \Sigma'', \Upsilon''>$ respectively. Each extension event $\mathcal{E}''_i$ has rank j or less and thus each $\mathcal{E}''_i$ is covered by some template in T. Let $\mathcal{E}_1 \ldots \mathcal{E}_n$ be templates $<\alpha_1, \Psi_1, \Sigma_1, \Upsilon_1> \ldots <\alpha_n, \Psi_n, \Sigma_n, \Upsilon_n>$ that cover extension events $\mathcal{E}''_1 \ldots \mathcal{E}''_n$ via substitutions $\rho_1 \ldots \rho_n$ respectively. We have assumed that no metavariable appears in more than one of r, $\mathcal{E}_1 \ldots \mathcal{E}_n$. Therefore we can define a substitution τ such that for any metavariable x, if x appears in r then $\tau(x)$ equals $\sigma(x)$; if x appears in $\mathcal{E}_i$ then $\tau(x)$ equals $\rho_i(x)$; otherwise $\tau(x)$ equals x. We now have

$$\tau(\Theta_i) = \sigma(\Theta_i) = \Psi''_i$$

$$\tau(\Psi_i) = \rho_i(\Psi_i) = \Psi''_i$$

$$\tau(\alpha_i) = \rho_i(\alpha_i) = \alpha''.$$

Thus we have that $\tau(\Theta_i) = \tau(\Psi_i)$ for $1 \leq i \leq n$ and $\tau(\alpha_i) = \tau(\alpha_j)$ for $1 \leq i \leq j \leq n$. So the substitution τ satisfies all of the conditions given in step 1 of the procedure. Let ρ be the most general substitution satisfying these conditions, as constructed at step 2 of the procedure.

The substitution ρ is at least as general as τ. This implies that the substitution τ can be written as ρ followed by another substitution τ', i.e., for any expression e we have that $\tau(e)$ equals $\tau'(\rho(e))$. Let α be $\rho(\alpha_i)$ as defined in step 3 of the procedure. Since $\tau'(\rho(\alpha_i))$ equals $\tau(\alpha_i)$ which equals α'', we have that $\tau'(\alpha)$ equals α''. The expression $\tau'(\rho(\Phi))$ equals $\tau(\Phi)$ which equals Φ''. Thus $\tau'(\rho(\Phi))$ is a label formula of $\Upsilon'' \cup \{\alpha''\}$. This implies that, for each immediate subexpression s of $\rho(\Phi)$, we have that $\tau'(s)$ either equals α'' or is a member of Υ''. Let $u_1 \ldots u_k$ be the set of all immediate subexpression u of $\rho(\Phi)$ such that $\tau'(u)$ equals α''. Let $w_1 \ldots w_p$ be the set of immediate subexpressions w of $\rho(\Phi)$ such that $\tau'(w)$ is a member of Υ''. Note that for each u_i we have that $\tau'(u_i)$ equals α'' which equals $\tau'(\alpha)$. Thus τ' is a substitution that satisfies the requirement of step 5. Let ρ' be the substitution defined in step 6 of the procedure, i.e., the most general substitution such that $\rho'(u_i) = \rho'(\alpha)$ for $1 \leq i \leq m$.

The substitution ρ' is at least as general as τ'. As before, this implies that τ' can be written as ρ' followed by another substitution τ'', i.e., for any expression e, $\tau'(e)$ equals $\tau''(\rho'(e))$. We now have that for any expression e, $\tau(e)$ equals $\tau''(\rho'(\rho(u))$. Let α', Φ', Σ', and Υ' be defined as in steps 7, 8, 9, and 10 of the procedure, and let $\mathcal{E}'$ be the tuple $<\alpha', \Phi', \Sigma', \Upsilon'>$. We will now show that $\mathcal{E}'$ is an extension event template that covers the original extension event $<\alpha'', \Phi'', \Sigma'', \Upsilon''>$ via the substitution τ''. We have that $\tau''(\alpha')$ equals $\tau''(\rho'(\alpha))$ which equals α''. Similarly, $\tau''(\Phi')$ equals Φ''. Furthermore, a case analysis on steps 10a through 10d can be used to show that $\tau''(\Upsilon')$ is a subset of Υ''. This implies that α' is not a member of Υ', otherwise we would have that $\tau''(\alpha')$ is a member of $\tau''(\Upsilon')$ and so α'' would be a member of Υ'' which violates the original condition that α'' be a one-step extension of Υ''. Since α' is not a member of Υ' the tuple $\mathcal{E}'$ is output by the procedure and thus is a member of $R(T)$. By the above lemma, $\mathcal{E}'$ is an extension event template. Finally, we must show that $\tau''(\Sigma')$ is a subset of $C_R(\Sigma'', \Upsilon'')$. The set $\tau''(\Sigma')$ equals $\bigcup_{1 \leq i \leq n} \tau''(\rho'(\rho(\Sigma_i)))$ which equals $\bigcup_{1 \leq i \leq n} \tau(\Sigma_i)$. But by assumption, $\tau(\sigma_i)$, which equals $\rho_i(\Sigma_i)$, is a subset of $C_R(\Sigma'', \Upsilon'')$. $\Box$

References

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

[2] 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.

[3] 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.

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

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

[6] Radford Neal. The computational complexity of taxononic inference. University of Toronto, December 1989.

[7] Greg Nelson and Derek Oppen. Fast decision procedures based on congruence closure. *JACM*, 27(2):356–364, April 1980.

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