Complete Axiomatizations of the Algebras of Finite, Rational and Infinite Trees

Michael J. Maher

IBM Thomas J. Watson Research Center,
P.O. Box 704, Yorktown Heights, NY 10598, U.S.A.

Abstract

In this paper we present complete axiomatizations for the algebras of finite trees and infinite trees. The axiomatizations are parameterized by the alphabet of function symbols. For both the finite trees and the infinite trees the work divides into two main cases, depending upon whether the number of function symbols is finite or infinite. In the former case an extra axiom is necessary to obtain completeness. The method of proof is an elimination of quantifiers. Although a full elimination of quantifiers is not possible, the method forms the basis of decision procedures for the theories of the corresponding algebras. As a corollary to the results on infinite trees we obtain the elementary equivalence of the algebra of rational trees and the algebra of infinite trees.

Introduction

The algebras of finite, rational and infinite trees are fundamental to symbolic computation. Syntactic terms can be regarded as finite trees, but the relationship is deeper than this. The operations of unification and matching which are the basis of logic and functional programming languages can be regarded as solving certain first-order formulas in the algebra of finite trees. Unification which omits the "occur check" solves the appropriate formula in the algebra of infinite trees. These algebras are also central to the declarative semantics of equational (e.g. [28]), functional (e.g. [32]), and logic programming languages (e.g. [19]), the algebraic approach to the semantics of programming languages (e.g. [27] [1]), the verification of programs using recursive data structures such as lists or circular lists (e.g. [29]), current efforts to determine sufficient-completeness of term rewriting systems (e.g. [14]) and to perform "induction-less induction" over the algebra defined by a term rewriting system (e.g. [13]), and issues in machine learning (e.g. [17]).

In this paper we present axiomatizations for the algebras of finite trees and infinite trees, and show that these axiomatizations are complete, that is, for every first-order sentence σ, σ holds in the algebra if and only if σ is a logical consequence of the axioms. The method used is an elimination of quantifiers. The axiomatizations are parameterized by the alphabet of function symbols (i.e. the signature of the algebra). For both the finite trees and the infinite trees the work divides into two main cases, depending upon whether the number of function symbols is finite or infinite. In the former case an extra axiom is necessary to obtain completeness. (In the case of the algebra of finite trees with an infinite number of function symbols the completeness of the axiomatization is known [15].) A full elimination of quantifiers is not possible. Nevertheless the method of elimination of quantifiers forms the basis of decision procedures for the theories of the corresponding algebras. As a corollary to the results on infinite trees we obtain the elementary equivalence of the algebra of rational trees and the algebra of infinite trees.

We explain how our results are applicable to one of the topics mentioned above, logic programming. Herbrand's Theorem allows the satisfiability of a collection of clauses to be reduced to the question of satisfiability in the algebra of finite trees. As a result, a standard semantics of logic programs is a model over this algebra. Such a semantics treats only finite successful computations. An axiomatization of the finite trees due to Clark [2] has been used, as part of the completion (in the sense of [2]) of a logic program, to give a logic semantics to finitely failed, as well as successful, computations. When the alphabet of function symbols is finite a domain closure axiom can be added to these axioms, and somewhat stronger results hold [24]. It is of interest to determine whether even stronger results can be obtained by adding further axioms. We show that this is not possible since these axioms form a complete theory. Similarly we cannot obtain stronger results by adding to Clark's axioms when the alphabet is infinite since also in this case we have a complete theory [15].

Prolog II and other Prolog implementations which omit the "occur check" during unification can be viewed as computing over the algebra of infinite trees [3]. Logic semantics for such implementations have been given in [7] and, more extensively, in [11]. We show that the axiomatization of equality in [11] forms a complete theory when the alphabet is infinite. The theory of [7] is not complete. When the alphabet is finite the addition of the domain closure axiom to the axioms of [11] produces a complete theory. Thus, although the results of [11] can be strengthened in a manner analogous to [24] when the alphabet is finite, there is no further strengthening possible. The elementary equivalence of the algebras of rational and infinite trees ensures that apparently differing views of Prolog II as computing over the algebra of infinite trees [4] or as computing over the algebra of rational trees [3] [11], are equivalent.

The inclusion in a logic programming language of a universal quantifier (restricted to act only on formulas involving only the equality predicate symbol) can allow a more incremental implementation of negation than the usual negation-as-failure [31] [21] [18]. [15] and [9] suggest more systematic implementations. In such languages it becomes necessary to determine the satisfiability of arbitrary formulas over the algebras of finite or infinite trees. The algorithms presented here can be used to determine the satisfiability of these formulas.

In the context of the constraint logic programming scheme [10] the property of solution-compactness of a domain and its associated constraints is of major importance. It is known that when the domain is the algebra of infinite trees and the constraints are equations and inequations this property holds, since a proof is embedded in the quite complex proof of the central lemma of [12]. Our results enable a simple proof of solution-compactness.

In the next section we introduce some necessary definitions and notations. The third section presents definitions and results specific to the algebra of finite trees. The following two sections present the axiomatizations of algebras of finite trees and the proofs of completeness when the alphabets are, respectively, infinite and finite. The final section presents the corresponding results for the algebras of rational and infinite trees. Due to space limitations some proofs are omitted and others are only sketched. Fuller proofs and discussion of the above-mentioned applications can be obtained from [22].

Preliminaries

In this paper we consider only unsorted languages. However the results extend straightforwardly to many-sorted languages [22]. We use Σ to denote the alphabet of function symbols and consider only one predicate symbol, $=$. We say that Σ is *singular* if it contains exactly one function symbol. In this paper we assume that the alphabet is non-empty and we concentrate on non-singular alphabets. Function symbols of arity 0 are called *constants*. A *term* over a set Z is either an element of Z or has the form $f(t_1, \ldots, t_n)$ where $f \in \Sigma$, f has arity n, and $t_1, \ldots, t_n$ are terms over Z. In the second case f is called the *principal function symbol* of the term. Terms over X will simply be called terms. We use $\equiv$ to denote syntactic identity of terms. We use the notation $x = t(u)$ to denote a conjunction $x_1 = t_1(u) \wedge \ldots \wedge x_n = t_n(u)$. We deliberately confuse such a conjunction of equations, the corresponding set of equations and the corresponding substitution $\{x_1 \leftarrow t_1(u), \ldots, x_n \leftarrow t_n(u)\}$.

A *theory* is a set of sentences. A theory T is *complete* if, for every sentence σ, either $T \models \sigma$ or $T \models \neg\sigma$. An *axiomatization* of an algebra $\mathcal{A}$ is a recursive set of sentences which are true of $\mathcal{A}$. We choose this terminology to emphasize the completeness of the axioms we will give, in contrast with the use of "axiomatization" to refer to a set of sentences with initial model $\mathcal{A}$. The *theory of an algebra* $\mathcal{A}$ is the set of all sentences true of $\mathcal{A}$. Two algebras $\mathcal{A}$ and $\mathcal{B}$ are *elementarily equivalent* if, for every sentence σ, $\mathcal{A} \models \sigma$ iff $\mathcal{B} \models \sigma$.

For technical convenience we assume that there is a propositional constant **True**, which is always True. We use $(\exists)$ to denote the existential closure. *var*(o) denotes the set of variables occurring in the syntactic object o. We define the following functions on terms:

$$size(x) = 1 \text{ when } x \in X$$
$$size(\,f(t_1, \ldots, t_n)\,) = 1 + \sum_{i=1}^{n} size(t_i) \text{ when } f \in \Sigma$$

$$height(x) = 1 \text{ when } x \in X$$
$$height(\,f(t_1, \ldots, t_n)\,) = 1 + \max_{1 \le i \le n} height(t_i) \text{ when } f \in \Sigma$$

The height of a collection or tuple of terms is the maximum of the heights of the terms.

Courcelle [6] formally defines the sets of infinite trees, rational (there called regular) trees and finite trees. Informally, each node of a tree is labelled with a function symbol of arity n, say, and has n children. Thus the finite trees are the variable-free terms, and the rational trees are (finite or) infinite trees with a finite number of subtrees. The algebras formed from these sets with the "tree-forming" functions corresponding to the function symbols of Σ are denoted by $\mathcal{IT}$, $\mathcal{RT}$, and $\mathcal{FT}$. Clearly $\mathcal{FT}$ is a subalgebra of $\mathcal{RT}$, which is in turn a subalgebra of $\mathcal{IT}$. [6] gives an extensive presentation of some fundamental properties of infinite and rational trees.

Finite Trees

The following equality theory E^* (dependent on Σ) is adapted from Clark's [2] axiomatization of the Herbrand domain, the algebra of finite trees. Since we will not be interested in proof theory, and $=$ will be interpreted as identity on each domain of interpretation, we omit the usual axioms of equality (i.e. reflexivity, symmetry etc).

For every $f \in \Sigma$

$$\forall x \, \forall y \, f(x) = f(y) \leftrightarrow x = y \qquad (1)$$

For every $f, g \in \Sigma, f \not\equiv g$

$$\forall x \, \forall y \, f(x) \neq g(y) \qquad (2)$$

For every term $t(x)$ containing x except 'x'

$$\forall x \, x \neq t(x) \qquad (3)$$

In any model $\mathcal{M}$ of E^*, for any element u of $\mathcal{M}$, either u is not in the range of any $f \in \Sigma$ (we call such an element an *isolated element*) or there is a unique $f \in \Sigma$ of arity n and unique elements $u_1, \dots, u_n$ $(n \geq 0)$ of $\mathcal{M}$ such that $u = f(u_1, \dots, u_n)$. Consequently every model of E^* satisfies the following property which we will use frequently

> If a valuation V for variables x in a model of E^* satisfies $\exists v \, x = t(v)$ then there is a unique extension of V which satisfies $x = t(v)$.

Let u, w be elements of a model $\mathcal{M}$ of (1) and (2). We say that w *appears by depth* $d{+}1$ $(d \geq 0)$ in u if $u = w$ or $u = g(u_1, \dots, u_m)$ for some $g \in \Sigma$ and w appears by depth d in some u_i $1 \leq i \leq m$. No element can appear by depth 0. Similarly $f \in \Sigma$ appears by depth d in u if some $w = f(w_1, \dots, w_n)$ appears by depth d in u. A non-f appears by depth d in u if some w appears by depth d in u where $w \neq f(w_1, \dots, w_n)$ for every list $w_1, \dots, w_n$ of elements of $\mathcal{M}$.

The following algorithm is close to Herbrand's original algorithm [8] for the solution of equations in the algebra of finite trees. [25] develops an efficient unification algorithm from this algorithm.

Solved Form Algorithm for Finite Trees

Non-deterministically choose an equation from the equation set to which a numbered step applies. The action taken by the algorithm is determined by the form of the equation:

(1) $f(t_1, \dots, t_n) = f(s_1, \dots, s_n)$
replace by the equations $t_1 = s_1, \dots, t_n = s_n$

(2) $f(t_1, \dots, t_n) = g(s_1, \dots, s_m)$ *where* $f \not\equiv g$
halt with failure

(3) $x = x$
delete the equation

(4) $t = x$ *where t is not a variable*
replace by the equation $x = t$

(5) $x = t$ *where $t \not\equiv x$ and x has another occurrence in the set of equations*
if x appears in t then halt with failure
otherwise replace x by t in every other equation

The test on the second line of (5) is called the *occur check*. The algorithm terminates when no step can be applied or when failure has been returned. When the algorithm does not return failure it returns a solved form. A *solved form* set of equations has the form

$$x_1 = t_1, x_2 = t_2, \dots, x_n = t_n,$$

where no variable x_i appears in any term t_j and the x_i's are distinct. The x_i's are called the *eliminable* variables of the solved form. The remaining variables are called *parameters*. If y is a variable in t_i then we say that x_i *depends* on y in this solved form.

A conjunction of equations e is *solvable* iff $E^* \models (\exists) \, e$. The following proposition summarizes some basic results linking the algebra of finite terms $\mathcal{FT}$, E^* and the solved form algorithm. The proofs are straightforward, except for (d).

Proposition.1 Let e and e' be conjunctions of equations. Then

(a) If Σ contains a constant then $\mathcal{FT}$ is a model of E^*

(b) If Σ is finite and contains a constant then $\mathcal{FT}$ is a model of $E^* \cup DCA$

(c) Let C be a non-empty set of constants. Then $\mathcal{FT}(\Sigma \cup C)$ is a model of E^*

(d) If Σ is finite then $E^* \cup DCA$ has a model

(e) The solved form algorithm always halts.

(f) e is solvable iff e has a solved form which is computed by the solved form algorithm.

(g) Let e' be a solved form of e. Then $E^* \models e' \leftrightarrow e$

(h) e has no solved form iff $E^* \models \neg e$

We close this section by characterizing the models of E^*. A set $Z \subseteq A$ is a *set of generators* of A if for every $a \in A$ there is a term t over Z such that $\mathcal{A} \models a = t$. A set $Z \subseteq A$ is *mutually independent* if for all terms t and t' over elements of Z, $\mathcal{A} \models t = t'$ iff $t \equiv t'$. An algebra $\mathcal{A}$ is *free* if there is a set of generators $Z \subseteq A$ of A which is mutually independent. Algebra $\mathcal{A}$ is *locally free* if every subalgebra generated by a finite set $Z \subseteq A$ is free.

Proposition.2 An algebra $\mathcal{A}$ is locally free iff $\mathcal{A}$ is a model of E^*

Proof If $\mathcal{A}$ is not a model of E^* there is an axiom and elements $a_1, \dots, a_n$ of A for which the negation of the axiom holds. The subalgebra generated by these elements is not free, since the equation corresponding to the axiom gives a counter-example. So $\mathcal{A}$ is not locally free.

Conversely, if $\mathcal{A}$ is a model of E^* then any subalgebra is also a model. Consider the subalgebra $\mathcal{B}$ generated by a finite set $Z \subseteq A$ and suppose there are terms t and t' over elements of Z such that $\mathcal{B} \models t = t'$ but $t \not\equiv t'$. Apply the solved form algorithm to $t = t'$, treating the elements of Z as variables, and let Z' be the set of parameters of the solved form. Clearly Z' generates Z and so generates B, but has a strictly lower cardinality. By repeatedly applying this procedure we eventually obtain $Y \subseteq Z$ which is a mutually independent generating set for B. □

Mal'cev [23] states a similar axiomatization of locally free algebras, but omits the axioms (3) from E^*. The rational trees satisfy axioms (1) and (2), and are not locally free (except in degenerate cases) so the axioms (3) are necessary for the above result. Examples show that there is no complete axiomatization of the free algebras.

Completeness of E^*

The completeness of E^* when Σ is infinite is proved using a method of elimination of quantifiers suggested by proofs in [16]. Since basic formulas contain existential quantifiers, not all quantifiers are eliminated. This completeness result was proved by Kunen in [15]. However we give a (different) proof here since it provides some results, and an introduction to techniques, which we use later.

A *basic formula* is a solved form conjunction of equations where the parameters have been existentially quantified. Thus a basic formula b takes the form

$$\exists u \begin{pmatrix} x_1 = t_1(u) \\ \wedge\, x_2 = t_2(u) \\ \dots \\ \wedge\, x_n = t_n(u) \end{pmatrix}$$

where u and x ($= x_1, \dots, x_n$) are disjoint sets of variables and $t_i(u)$ is a term whose variables come from u. Thus the only free variables in such a basic formula are the variables x, and the collection of equations is in solved form. Note that any solvable conjunction of basic formulas can be simplified to (at most) a single basic formula using the solved form algorithm.

A set of variables v is *unconstrained* by a conjunction of equations $e(v, w, z)$ wrt the set of variables w if $e(v, w, z)$ has a solved form in which all variables of v are parameters and only the variables z depend on v. In this case if a valuation V (in a model of E^*) for w is given where $\exists v\, \exists z\, e(v, w, z)$ is True under V then $\forall v\, \exists z\, e(v, w, z)$ is also True under V. Otherwise we say $e(v, w, z)$ *constrains* v wrt w. When w is empty we simply say that v is constrained/unconstrained by $e(v, z)$. When Σ is non-singular, v is unconstrained by $e(v, z)$ iff $\forall v\, \exists z\, e(v, z)$ is True.

The elimination of quantifiers requires two main lemmas. The first allows the distribution of existential quantifiers over conjunctions in some cases. This generalizes results of [16] [18] on the algebra of finite trees to all locally free algebras.

Lemma.3 If Σ is infinite then

$$E^* \models \exists y\, (b \wedge \bigwedge_{i=1}^{n} \neg b_i) \leftrightarrow \bigwedge_{i=1}^{n} \exists y\, (b \wedge \neg b_i)$$

where b and the b_i are basic formulas

Proof One implication is trivial. For the other let $\exists y\, (b \wedge \neg b_j)$ be the formula

$$\exists y \left(\exists u, v \begin{pmatrix} x = t(u) \\ \wedge\, y = s(u, v) \end{pmatrix} \wedge \exists w, z \begin{pmatrix} x = t'(w) \\ \wedge\, y = s'(w, z) \end{pmatrix} \right)$$

where $x \cup y$ is the collection of free variables in $b, b_1, \dots, b_n$, u is the collection of variables appearing in t and v is the collection of variables appearing in s but not in t, and similarly w is the collection of variables appearing in t' and z is the collection of variables appearing in s' but not in t'. Now

$$\exists y \left(\exists u, v \begin{matrix} x = t(u) \\ \wedge\, y = s(u, v) \end{matrix} \wedge \neg \exists w, z \begin{matrix} x = t'(w) \\ \wedge\, y = s'(w, z) \end{matrix} \right)$$

is equivalent to

$$\exists u, v \left(x = t(u) \wedge \neg \exists w, z \begin{matrix} x = t'(w) \\ \wedge\, s(u, v) = s'(w, z) \end{matrix} \right)$$

Let V be a valuation on a model $\mathcal{M}$ of E^* for x such that $\bigwedge_{i=1}^{n} \exists y\, b \wedge \neg b_i$ is True under V. (The lemma holds trivially unless such a V exists.) Then V has a unique extension V' to u such that

$$\exists v \left(x = t(u) \wedge \neg \exists w, z \begin{array}{c} x = t'(w) \\ \wedge\, s(u, v) = s'(w, z) \end{array} \right)$$

is True under V'. Now either

$$\forall v \neg \exists w, z \begin{pmatrix} x = t'(w) \\ \wedge\, s(u, v) = s'(w, z) \end{pmatrix}$$

is True under V' or

$$\exists v, w, z \begin{pmatrix} x = t'(w) \\ \wedge\, s(u, v) = s'(w, z) \end{pmatrix}$$

is True under V'. In the first case we can extend V to U which assigns $V_*(s(u, v))$ to y for any extension V_* of V' to v and $b \wedge \neg b_j$ is True under U.

In the second case V' has a unique extension V'' to w under which $x = t'(w)$ is True. Now $s(u, v) = s'(w, z)$ must constrain v wrt the variables of u and w since otherwise $\forall v \exists z\, s(u, v) = s'(w, z)$ is True under V''. If this were so then $\forall v \exists w, z\, x = t'(w) \wedge s(u, v) = s'(w, z)$ is True under V' which contradicts the definition of V'.

Let $m = height(s(u, v))$, let θ be a substitution which replaces the variables of v with terms having new distinct principal function symbols that do not appear by depth m in $V''(w)$, and let V_* be an extension of V' to the variables introduced by θ. Consider the extension U of V to the variables y such that y is assigned $V_*(s(u, v\theta))$. U has a unique extension such that $x = t(u) \wedge y = s(u, v)$ is True and in this extension v is assigned $U(v\theta)$. By the choice of θ and since v is constrained by $s(u, v) = s'(w, z)$, this extension evaluates $\neg \exists w, z\, x = t'(w) \wedge s(u, v) = s'(w, z)$ to True. Thus $b \wedge b_j$ is True under U.

Let us now denote the valuation V'' built from b_i by V_i and denote the variables w in b_i by w_i. If we choose θ to have new principal function symbols which do not appear by depth m in $V_i(w_i)$ for $1 \leq i \leq n$, and define U as above, then $b \wedge b_i$ is True under U for $1 \leq i \leq n$. Thus $\exists y\, b \wedge \bigwedge_{i=1}^{n} \neg b_i$ is True under V. Since the choice of V and $\mathcal{M}$ was arbitrary

$$E^* \models \exists y\, (b \wedge \bigwedge_{i=1}^{n} \neg b_i) \leftarrow \bigwedge_{i=1}^{n} (\exists y\, b \wedge \neg b_i)$$

□

With minor alterations to the proof we can show that when Σ is finite and non-singular we have

$$\mathcal{M} \models \exists y\, (b \wedge \bigwedge_{i=1}^{n} \neg b_i) \leftrightarrow \bigwedge_{i=1}^{n} (\exists y\, b \wedge \neg b_i)$$

for every model $\mathcal{M}$ of E^* which contains more isolated elements than $|\, var(b)\,|$. The second lemma allows the elimination of a quantifier. The resulting Boolean expression was incorrectly stated in [16].

Lemma.4 For every formula of the form $\exists y\, b \wedge \neg b'$ where b and b' are basic formulas there is a formula g consisting of a Boolean combination of basic formulas such that

$$E^* \models \exists y\, (b \wedge \neg b') \leftrightarrow g$$

Proof Let $b \wedge \neg b'$ be

$$\exists u \begin{pmatrix} x = t(u) \\ \wedge\, y = s(u) \end{pmatrix} \wedge \neg \exists w \begin{pmatrix} x = t'(w) \\ \wedge\, y = s'(w) \end{pmatrix}$$

where u is the bound variables appearing in b and w is the bound variables appearing in b'. Then by applying the substitution rule and simplifying, $\exists y\, b \wedge \neg b'$ is equivalent to

$$\exists y, u \left(\begin{array}{c} x = t(u) \\ \wedge\, y = s(u) \end{array} \wedge \neg \exists w \begin{pmatrix} x = t(u) \\ \wedge\ \ \phi \end{pmatrix} \right)$$

where ϕ is a solved form of $t(u) = t'(w) \wedge s(u) = s'(w, z)$ if this is solvable, and otherwise ϕ is $\neg$**True**. This is equivalent to

$$\exists y, u \begin{pmatrix} x = t(u) \\ \wedge\, y = s(u) \end{pmatrix} \wedge \neg \exists w \begin{pmatrix} x = t(u) \\ \wedge\ \ \phi \end{pmatrix}$$

which is equivalent to

$$\exists u\, x = t(u) \wedge \neg \exists w \begin{pmatrix} x = t(u) \\ \wedge\ \ \phi \end{pmatrix}$$

Hence we finally have

$$E^* \models \exists y\, (b \wedge \neg b') \leftrightarrow (\exists u\, x = t(u))$$

if $t(u) = t'(w) \wedge s(u) = s'(w)$ is not solvable, and otherwise

$$E^* \models \exists y\, (b \wedge \neg b') \leftrightarrow (\exists u\, x = t(u) \wedge \neg \exists v\, x = t(u\phi))$$

where $v = var(t(u\phi))$. □

We can now give a procedure for transforming a given logical formula into a Boolean combination of basic formulas. First write the formula in prenex normal form, that is, move all quantifiers to the front of the formula (possibly changing variable names in the process) and place the remainder of the formula in disjunctive normal form. We will call these quantifiers *prenex quantifiers*. Let Y denote the set of variables in the prenex normal form. Next transform each conjunction of equations into a basic formula, where the eliminable variables are the variables of Y. For example the equation $f(x, g(y), z) = f(w, w, z)$ has solved form $x = g(y) \wedge w = g(y)$ which becomes the basic formula $\exists u, v\, x = g(v) \wedge w = g(v) \wedge y = v \wedge z = u$. If a conjunction of equations does not have a solved form the conjunction is False in every model of E^* and the whole formula is simplified accordingly. Clearly both these steps preserve logical equivalence.

Consider the innermost prenex quantifier in the formula and suppose it is an existential quantifier. That is, we have $Q\, \exists y\, (D_1 \vee \dots \vee D_n)$ where Q denotes the remaining prenex quantifiers. The existential quantifier can be distributed over the disjunction preserving equivalence. Now each D_i takes the form $b \wedge \bigwedge_{j=1}^{m} \neg b_j$. By lemma 3, $\exists y\, D_i$ is equivalent to $\bigwedge_{j=1}^{m} \exists y\, b \wedge \neg b_j$ and by lemma 4 each formula $\exists y\, b \wedge \neg b_j$ can be transformed into an equivalent Boolean combination of basic formulas. Thus the prenex quantifier has been eliminated. The matrix of the formula can then be placed in disjunctive normal form, treating the basic formulas as atoms. Any conjunction of basic formulas is then replaced by a single basic formula.

If the innermost prenex quantifier is a universal quantifier, that is we have $Q\, \forall y\, (D_1 \vee \dots \vee D_n)$, then we can replace it by $Q\, \neg \exists y\, \neg (D_1 \vee \dots \vee D_n)$, obtain the disjunctive normal form of $\neg (D_1 \vee \dots \vee D_n)$, treating the basic formulas as atoms, and proceed as before. The negation of the resulting formula can then be placed in disjunctive normal form, treating the basic formulas as atoms.

By repeating this procedure for each prenex quantifier the formula is transformed into an equivalent Boolean combination of basic formulas. In particular, when f is closed g evaluates either to True or to False.

Theorem.5 When Σ is infinite E^* is a complete theory.

It is worth clarifying the relationship between this proof and the work of Kunen [15] [16]. Kunen showed in [16] that lemmas 3 and 4 hold for the algebra of finite trees. (These were not expressed logically, but rather in terms of basic sets of tuples of ground terms and projections, which correspond to our basic formulas and existential quantifications.) Since E^* is complete [15], lemmas 3 and 4 follow. Thus Kunen showed that the method used here for proving the completeness of E^* was feasible, but did not explicitly give such a proof.

Complete Axiomatization of Finite Trees under a Finite Alphabet

E^* is not a complete theory when Σ is finite. This arises from the ability to use all the function symbols of the language in a single sentence, such as the following Domain Closure Axiom (DCA) which is independent of E^*.

$$\forall x \bigvee_{f \in \Sigma} \exists z\, x = f(z) \qquad (DCA)$$

The Domain Closure Axiom was introduced by Reiter [30] in the case where Σ consists of a finite number of constants. In this case $E^* \cup DCA$ determines a unique algebra up to isomorphism. The DCA was extended by Lloyd and Topor [20] to a many-sorted alphabet with non-constant function symbols, using one axiom of the form shown above for each sort. When a hierarchically sorted language [20] is used $E^* \cup DCA$ is categorical.

The DCA can be regarded as expressing the non-existence of isolated elements. Thus the theory $E^* \cup DCA$ characterizes the locally free algebras with no isolated elements when Σ is finite.

We now proceed to show the completeness of $E^* \cup DCA$ when Σ is non-singular. In preparation for the next lemma we define a sequence of terms $G_i(h)$, parameterized by h. Let $f \in \Sigma$ have arity $q > 0$ and let $g \in \Sigma$, $g \neq f$. $F_h(l_1, l_2, \dots, l_{q^h})$ denotes the term corresponding to a uniform tree of height h with internal nodes marked by f and leaf nodes $l_1, l_2, \dots, l_{q^h}$. $G_i(h)$ denotes $F_{h \times i}(g(z), g(z), \dots, g(z))$ for some fixed list of variables z. So, for example, if f is unary then $G_i(h)$ denotes $f^{h \times i}(g(z))$ and if f is binary then $G_1(3)$ and $G_3(1)$ denote $f(f(f(g(z),g(z)), f(g(z),g(z))), f(f(g(z),g(z)), f(g(z),g(z))))$.

Lemma 3 does not hold when Σ is finite. However the following restricted form of that lemma does hold.

Lemma.6 Suppose Σ contains a non-constant function symbol f and another function symbol g. If the depth of every occurrence of a parameter of b is greater than the depth of any symbol in any b_i then

$$E^* \models \exists y\, (b \wedge \bigwedge_{i=1}^{n} \neg b_i) \longleftrightarrow \bigwedge_{i=1}^{n} \exists y\, (b \wedge \neg b_i)$$

where b and the b_i are basic formulas
Proof The proof is very close to the proof of lemma 3 up to the point where it is determined that $s(u, v) = s'(w, z)$ constrains v wrt u and w. We can apply the solved form algorithm to $s'(w, z) = s(u, v)$ by first repeatedly stripping off

the common principal function symbols (step 1). By the hypothesis of the lemma we obtain a set of equations where every variable of $w \cup z$ appears at least once alone on the left-hand side (lhs) of some equation. It is easily seen that once a variable appears alone on the lhs of some equation during the execution of the solved form algorithm, it will continue to appear alone on the lhs of some equation. Consequently $w \cup z$ are eliminable variables in the solved form computed by the algorithm and since v is constrained wrt $u \cup w$, the solved form contains an equation in one of the following forms:

(a) $v_1 = \tau$ for variable v_1 from v and some ground term τ, or $v_1 = \tau(v_2)$ for some term τ containing v_2, where v_1, v_2 are in v
(b) $v_1 = \tau(u_1)$ for variable u_1 from u, variable v_1 from v and some term τ containing u_1
(c) $p = \tau(v_1)$ for variable p from u or w, variable v_1 from v, and some term τ containing v_1

Suppose v consists of the variables $v_1, \dots, v_n$. Define the conjunction θ to be $\bigwedge_{i=1}^{n} v_i = G_i(D)$ where we choose D, depending on which case above holds, so that:

(a) $D > height(\ \tau\ )$.
(b) D greater than the depth at which a non-f appears in $V''(\tau(u_1))$ if this is finite. Otherwise $D > 1$.
(c) Let $V^{(3)}$ be the unique extension of V'' to v_1 which makes

$$\exists v', w, z \begin{pmatrix} x = t'(w) \\ \wedge\, s(u, v) = s'(w, z) \end{pmatrix}$$

True under $V^{(3)}$, where v' is $v - v_1$. Choose D greater than the depth at which a non-f appears in $V^{(3)}(v_1)$ if this is finite. Otherwise choose $D > 1$.

Fix an extension V_* of V' to the variables introduced by θ. Now consider the extension U of V to y such that y is assigned $V_*(s(u, v\theta))$. By the choice of θ, U evaluates $\neg \exists w, z\, x = t'(w) \wedge s(u, v) = s'(w, z)$ to True and so $b \wedge \neg b_j$ is True under U.

Finally, we can choose D to be the maximum of the values chosen for each b_j, and define θ and U accordingly, so that $b \wedge \neg b_i$ is True under U for $1 \le i \le n$. Thus $\exists y\, b \wedge \bigwedge_{i=1}^{n} \neg b_i$ is True under V. Since the choice of V and $\mathcal{M}$ was arbitrary

$$E^* \models \exists y\, (b \wedge \bigwedge_{i=1}^{n} \neg b_i) \leftarrow \bigwedge_{i=1}^{n} (\exists y\, b \wedge \neg b_i)$$

□

Remark. This lemma can be sharpened by relaxing the hypothesis. If t and t' are terms we say that t *overlaps the variables* of t' if t and t' are not unifiable, or if for every variable v in t' there is an occurrence o such that o is also an occurrence of some subterm of t. Overlapping for tuples of terms is defined by treating the tuple constructor as just another function symbol. For basic formulas, $\exists u\, x = t(u) \wedge y = s(u)$ *overlaps the y variables* of $\exists v\, x = t'(v) \wedge y = s'(v)$ if $s(u)$ overlaps the variables of $s'(v)$ as tuples. The hypothesis of the above lemma can be relaxed to require only that b overlaps the y variables of each b_i. This can support a more efficient version of the procedure described below.

The procedure for transforming a given formula into a Boolean combination of basic formulas when Σ is finite is similar to the procedure when Σ is infinite. The formula is put in prenex normal form and conjunctions of equations are converted to basic formulas as before. If the innermost quantifier is existential then it distributes over the disjunctions, and we have formulas of the form $\exists y\, b \wedge \bigwedge_{i=1}^{n} \neg b_i$. Suppose the basic formula b, which takes the form $\exists u, v\, x = t(u, v) \wedge y = s(u, v)$, has an occurrence of the variable v at a depth which is not greater than the depth of every symbol of every b_i. It is easily shown that

$$DCA \models b \leftrightarrow \bigvee_{f \in \Sigma} b_f$$

where b is

$$\exists u, v \begin{pmatrix} x = t(u, v) \\ \wedge\, y = s(u, v) \end{pmatrix}$$

and b_f is

$$\exists u, w \begin{pmatrix} x = t(u, f(w)) \\ \wedge\, y = s(u, f(w)) \end{pmatrix}$$

Hence we can replace b in the formula by $\bigvee_{f \in \Sigma} b_f$ and obtain $\bigvee_{f \in \Sigma} \exists y\, b_f \wedge \bigwedge_{i=1}^{n} \neg b_i$. Furthermore, occurrences of the variables w in b_f are at a depth one greater than the depth of the corresponding v in b. Consequently, by repeatedly applying such replacements we obtain $\bigvee_j \exists y\, b_{(j)} \wedge \bigwedge_{i=1}^{n} \neg b_i$ where the depths of every occurrence of parameters in the $b_{(j)}$'s is greater than the depth of any symbol in any b_i. Thus we can apply lemma 6 and then lemma 4 to eliminate the quantifier $\exists y$.

As before, a universal innermost quantifier, $Q\, \forall y\, B$ can be dealt with as $Q\, \neg \exists y \neg B$. Thus we can reduce any formula to a Boolean combination of basic formulas.

Theorem.7 When Σ is finite and non-singular $E^* \cup DCA$ is a complete theory

Mal'cev [23] gives an algorithm which, in fact, shows the decidability of E^* when Σ is finite. This algorithm can be adapted to show the above result. The method of [15] or recent results of [26] and [5] could also be used as a basis for a proof.

Rational and Infinite Trees

In this section we prove results for algebras of rational and infinite trees corresponding to the results in the previous two sections for algebras of finite trees. The algebra of finite trees is very closely related to the algebras of rational and infinite trees (for example it is a subalgebra of these algebras). Thus it is not surprising that the axiomatizations and the patterns of proof have similarities. The definitions and proofs of the previous section and the redefinitions and proofs of this section have been organized so that the similarities of proofs are evident. Since it will turn out (corollary 14) that the algebras of rational trees and infinite trees are elementarily equivalent, only one collection of proofs is necessary.

We use a *rational solved form algorithm*, given in [3], which simplifies a collection of equations over the algebra of rational trees into a rational solved form. A set of equations is *circular* if it has the form

$$x_1 = x_2, x_2 = x_3, \ \dots, x_{n-1} = x_n, x_n = x_1$$

A set of equations θ is in *rational solved form* if it has the form

$$x_1 = t_1(x,y), x_2 = t_2(x,y), \ \dots, x_n = t_n(x,y)$$

where the sets of variables x (i.e. $x_1, \dots, x_n$) and y are disjoint and the equation set contains no circular subset. The variables x are the eliminable variables and y is the parameters. We say that x_i *directly depends on* x_j in this solved form if x_j occurs in t_i. We say that x_i *depends on* x_j to refer to the transitive closure of direct dependence. We also say that x_j *supports* x_i. We define the height of a rational solved form θ by $height(\theta) = \sum_{i=1}^{n} height(t_i(x,y))$.

In [11] the axiomatization E_r^* was given for the algebra of rational trees. E_r^* consists of the axioms (1), (2) and the following axioms (4)

For every rational solved form $x = t(x,y)$

$$\forall y \, \exists! x \; x = t(x,y) \qquad (4)$$

We can express the quantifier $\exists!$ ("there exists a unique") in terms of the usual first-order quantifiers and obtain the following equivalent version of the axioms (4)

$$\forall y \, \forall z \, \exists x \; x = t(x,y) \wedge (z = t(z,y) \rightarrow x = z) \qquad (4')$$

A conjunction of equations e is *solvable* iff $E_r^* \models (\exists)\, e$. We now state fundamental results linking E_r^* , the rational solved form algorithm and the algebras of rational trees and infinite trees.

Proposition.8 Let e and e' be conjunctions of equations. Then

(a) $\mathcal{RT}$ is a model of E_r^*

(b) If Σ is finite then $\mathcal{RT}$ is a model of $E_r^* \cup DCA$

(c) $\mathcal{IT}$ is a model of E_r^*

(d) If Σ is finite then $\mathcal{IT}$ is a model of $E_r^* \cup DCA$

(e) The rational solved form algorithm always terminates.

(f) e is solvable iff e has a solved form which is computed by the solved form algorithm.

(g) Let e' be a solved form of e. Then $E_r^* \models e' \leftrightarrow e$

(h) e has no solved form iff $E_r^* \models \neg e$

For rational/infinite trees basic formulas take a slightly different form. A *rational basic formula* takes the form

$$\exists u \begin{pmatrix} x = t(x,u) \\ \wedge\, u' = r(x,u) \end{pmatrix}$$

where u and x ($= x_1, \dots, x_n$) are disjoint sets of variables, u' is some subset of u, and the set of equations is in rational solved form. Thus the only free variables in such a basic formula are the variables x. The variables $u - u'$ are the parameters of the basic formula. We can assume without loss of generality that the variables x do not appear on the right hand side of equations since

$$\models \exists u \begin{pmatrix} x = t(x,u) \\ \wedge\, u' = r(x,u) \end{pmatrix} \leftrightarrow \exists u, x' \begin{pmatrix} x = t(x',u) \\ \wedge\, x' = t(x',u) \\ \wedge\, u' = r(x',u) \end{pmatrix}$$

The definition of constrained/unconstrained is unchanged, although it now refers to rational solved forms, valuations in models of E_r^* and a slightly more general notion of variable dependence.

The proof of lemma 4 is adapted to prove the following lemma. This is a stronger version of the Independence of Inequations for infinite trees [4].

Lemma.9 If Σ is infinite then

$$E_r^* \models \exists y \, (b \wedge \bigwedge_{i=1}^{n} \neg b_i) \leftrightarrow \bigwedge_{i=1}^{n} (\exists y \, b \wedge \neg b_i)$$

where b and the b_i are rational basic formulas
Proof Let $b \wedge \neg b_j$ be

$$\exists u, v, q \begin{array}{l} x = t(u) \\ \wedge\, u = k(u) \\ \wedge\, y = s(u, v, q) \\ \wedge\, q = r(u, v, q) \end{array} \wedge \neg \exists w, z, p \begin{array}{l} x = t'(w) \\ \wedge\, w = k'(w) \\ \wedge\, y = s'(w, z, p) \\ \wedge\, p = r'(w, z, p) \end{array}$$

where u (respectively w) is the bound variables supporting x in b (b_j), v (z) is the remaining parameters appearing in b (b_j), q (p) is the remaining bound eliminable variables appearing in b (b_j), and $u = k(u)$ ($w = k'(w)$) denotes a rational solved form set of equations where some subset of the variables of u (w) are eliminable.

As in lemma 3 the proof divides into two cases, one which is straightforward and a second case in which v is constrained wrt $u \cup w$ by the equations

$$\left(\begin{array}{l} x = t'(w) \\ \wedge\, w = k'(w) \\ \wedge\, y = s'(w, z, p) \\ \wedge\, p = r'(w, z, p) \end{array} \right) \wedge \left(\begin{array}{l} u = k(u) \\ \wedge\, y = s(u, v, q) \\ \wedge\, q = r(u, v, q) \end{array} \right)$$

We take $m = \mathit{height}(\phi)$ where ϕ is a solved form of the above set of equations, and take θ as in lemma 3. For a given valuation V for x we let V' be the appropriate extension to u, and define the extension U of V to y where y is assigned $V_*(s(u, v\theta, q))$ where V_* is an extension of V' to q and the variables introduced by θ such that $q = r(u, v\theta, q)$ is True under V_*. The remainder of the proof follows the proof of lemma 3. □

Lemma.10 For every formula of the form $\exists y\, b \wedge \neg b'$ where b and b' are rational basic formulas there is a formula g consisting of a Boolean combination of rational basic formulas such that

$$E_r^* \models \exists y\,(b \wedge \neg b') \leftrightarrow g$$

Proof Let $b \wedge \neg b'$ be

$$\exists u \left(\begin{array}{l} x = t(u) \\ \wedge\, y = s(u) \\ \wedge\, u = r(u) \end{array} \right) \wedge \neg \exists w \left(\begin{array}{l} x = t'(w) \\ \wedge\, y = s'(w) \\ \wedge\, w = r'(w) \end{array} \right)$$

where u (w) is the bound variables in b (b'), and $u = r(u)$ ($w = r'(w)$) denotes a rational solved form set of equations where some subset of the variables of u (w) are eliminable. The details of the proof are essentially the same as those of lemma 4, and so are omitted.

If $t(u) = t'(w) \wedge u = r(u) \wedge s(u) = s'(w) \wedge w = r'(w)$ is solvable with solved form ϕ then the required Boolean combination of rational basic formulas is

$$\exists u \left(\begin{array}{l} x = t(u) \\ \wedge\, u = r(u) \end{array} \right) \wedge \neg \exists u, w \left(\begin{array}{l} x = t(u) \\ \wedge\quad \phi \end{array} \right)$$

Otherwise we have

$$E_r^* \models \exists y\,(b \wedge \neg b') \leftrightarrow \exists u \left(\begin{array}{l} x = t(u) \\ \wedge\, u = r(u) \end{array} \right)$$

It is straightforward to delete the variables and parts of ϕ which do not support x. □

Using essentially the same procedure as for finite trees when Σ is infinite every formula can be reduced to a Boolean combination of rational basic formulas. In particular every sentence evaluates to True or False:

Theorem.11 When Σ is infinite, E_r^* is a complete theory.

As is the case for the finite trees, when the alphabet is finite we need a different version of lemma 9. We first need some definitions.

Let θ be the rational solved form

$$\begin{array}{l} u = t(u, v, w, z) \\ v = s(u, v, w, z) \\ \quad \ldots \\ w = r(u, v, w, z) \end{array}$$

where u and v are single variables. The operation which produces the rational solved form θ'

$$\begin{array}{l} u = t(u, s(u, v, w, z), w, z) \\ v = s(u, v, w, z) \\ \quad \ldots \\ w = r(u, v, w, z) \end{array}$$

will be called *unfolding v for u*. Clearly θ and θ' are logically equivalent. Suppose u depends on v. v may not occur in t, but by repeatedly unfolding variables for u, always choosing an eliminable variable of minimal depth, we eventually obtain $u = t'(u, v, w, z)$ where v occurs at depth d in t'. If D is the least such d over all unfoldings we say that *u depends on v at depth D* in the solved form θ. Similarly for basic formulas. For a parameter v which occurs in a basic formula b, the depth of v in the basic formula wrt the variables y is $\mathit{depth}_{b\,y}(v) = \min\limits_{y' \in y} \{D : y' \text{ depends on } v \text{ at depth } D \text{ in } b\}$ where y is a subset of the free variables of b.

Lemma.12 Suppose Σ contains a function symbol f of arity greater than 0 and another function symbol g. Let $M = \max\limits_{v, j} \{\mathit{depth}_{b_j\,y}(v) : v \text{ occurs in } b_j\}$. If every parameter in b has depth greater than M wrt y then

$$E^* \models \exists y\,(b \wedge \bigwedge_{i=1}^{n} \neg b_i) \leftrightarrow \bigwedge_{i=1}^{n} \exists y\,(b \wedge \neg b_i)$$

where b and the b_i are rational basic formulas.

Theorem.13 When Σ is finite and non-singular $E_r^* \cup DCA$ is a complete theory.

When Σ contains a single (non-constant) function symbol f, the algebra consisting of the single element ∞ where $\infty = f(\infty, \infty, \ldots, \infty)$ is a model of $E_r^* \cup DCA$. In fact this algebra is simultaneously the algebra of rational trees and the algebra of infinite trees for this alphabet. Consequently the single axiom $\forall x, y\; x = y$ forms a complete axiomatization for these algebras when the alphabet contains a single function symbol.

There are two significant corollaries of the theorems in this section. The first is the decidability of the theories of the algebras under consideration. This follows directly from the completeness theorems, but the method of elimination of quantifiers provides a more feasible decision procedure. The second also follows immediately from the theorems and consideration of the remaining, degenerate case.

Corollary.14 $\mathcal{RT}$ and $\mathcal{IT}$ are elementarily equivalent.

References

[1] J.A. Goguen, J.W. Thatcher, E.G. Wagner & J.B. Wright, Initial Algebra Semantics and Continuous Algebras, *JACM*, Vol. 24, No. 1, 1977, 68-95.

[2] K.L. Clark, Negation as Failure, in *Logic and Databases*, H. Gallaire and J. Minker (Eds.), Plenum Press, New York, 293-322, 1978.

[3] A. Colmerauer, Prolog and Infinite Trees, in *Logic Programming*, K.L. Clark and S-A. Tarnlund (Eds), Academic Press, New York, 231-251, 1982.

[4] A. Colmerauer, Equations and Inequations on Finite and Infinite Trees, *Proc. 2nd. Int. Conf. on Fifth Generation Computer Systems*, Tokyo, 85-99, November 1984.

[5] H. Comon and P. Lescanne, Equational Problems and Disunification, 1988.

[6] B. Courcelle, Fundamental Properties of Infinite Trees, *Theoretical Computer Science*, 25(2), 95 - 169, March 1983.

[7] M.H. van Emden and J.W. Lloyd, A Logical Reconstruction of Prolog II, *Proc. 2nd. Int. Conf. on Logic Programming*, Uppsala, 35-40, 1984.

[8] J. Herbrand, Recherches sur la theorie de la demonstration, These, Universite de Paris. (In: *Ecrits logiques de Jacques Herbrand*, Paris, PUF, 1968).

[9] T. Hickey, Negation by Constrained Failure, draft manuscript, 1987.

[10] J. Jaffar & J-L. Lassez, Constraint Logic Programming, *POPL-87*, 111-119, 1987.

[11] J. Jaffar, J-L. Lassez & M.J. Maher, Prolog-II as an Instance of the Logic Programming Language Scheme, in: *Formal Descriptions of Programming Concepts III*, M. Wirsing (Ed.), North-Holland, 1987, 275-299.

[12] J. Jaffar & P.J. Stuckey, Semantics of Infinite Tree Logic Programming, *Theoretical Computer Science*, 46, 1986.

[13] J-P. Jouannaud & E. Kounalis, Proof by Induction in Equational Theories without Constructors, *Proc. Symp. on Logic in Computer Science*, Boston, 1986.

[14] D. Kapur, P. Narendran & H. Zhang, On Sufficient-completeness and Related Properties of Term Rewriting Systems, 1985.

[15] K. Kunen, Negation in Logic Programming, *Journal of Logic Programming*, 4, 289-308, 1987.

[16] K. Kunen, Answer Sets and Negation as Failure, *Proc. 4th. Int. Conf. on Logic Programming*, Melbourne, 1987.

[17] J-L. Lassez & K.G. Marriott, Explicit Representation of Terms Defined by Counter Examples, *Journal of Automated Reasoning*, 3, 301-317, 1987.

[18] J-L. Lassez, M.J. Maher & K.G. Marriott, Unification Revisited, in: *Foundations of Deductive Databases and Logic Programming*, J. Minker (Ed.), Kauffman, 1987.

[19] J.W. Lloyd, *Foundations of Logic Programming*, Springer-Verlag, 1984.

[20] J.W. Lloyd & R.W. Topor, A Basis for Deductive Databases, *Journal of Logic Programming*, 2, 93-109, 1985.

[21] M.J. Maher, Logic Semantics for a Class of Committed-choice Programs, *Proc. 4th. Int. Conf. on Logic Programming*, Melbourne, 858-876, 1987.

[22] M.J. Maher, Complete Axiomatizations of the Algebras of Finite, Rational and Infinite Trees, IBM Research Report, T.J. Watson Research Center, 1988.

[23] A. Mal'cev, On the Elementary Theories of Locally Free Universal Algebras, *Soviet Math. Doklady*, 1961, 768-771.

[24] P. Mancarella, S. Martini & D. Pedreschi, Complete Logic Programs with Domain Closure Axiom, *Journal of Logic Programming*, to appear.

[25] A. Martelli & U. Montanari, An Efficient Unification Algorithm, *TOPLAS*, Vol. 4, No. 2, April 1982, 258-282.

[26] P. Nickolas, The Representation of Answers to Logical Queries, *Proc. 11th. Australian Computer Science Conf.*, Brisbane, 246-255, 1988.

[27] M. Nivat, Languages Algebraic sur le Magma Libre et Semantique des Schemas de Programme, in: *Automata, Languages and Programming*, M. Nivat (Ed.), North-Holland, Amsterdam, 293-308, 1972.

[28] M.J. O'Donnell, *Equational Logic as a Programming Language*, MIT Press, 1985.

[29] D.C. Oppen, Reasoning about Recursively Defined Data Structures, *JACM*, Vol. 27, No. 3, 1980, 403-411.

[30] R. Reiter, On Closed World Databases in *Logic and Databases*, H. Gallaire and J. Minker (Eds.), Plenum Press, New York, 55-76, 1978.

[31] T. Sato & H. Tamaki, Transformational Logic Programming Synthesis, Proc. FGCS'84, Tokyo, 195-201, 1984.

[32] D. Turner, An Overview of Miranda, *SIGPLAN Notices* 21, 12, 1986, 158-166.