Approximating E-Unification

Christopher Lynch *and Barbara Morawska

January 26, 2001

Abstract

We give a set of inference rules for solving E-unification. We prove the completeness with Eager Merging for linear theories and goals with no repeated variables. If the theory is further restricted to have no repeated variables, we show that E-unification is decidable and has linear complexity when the theory is considered constant, and cubic when it is part of the input. For any E-unification problem, equations can be transformed into this class and the problem can be quickly approximated.

1 Introduction

Equational logic commonly arises in deductive problems. A particularly important problem is to decide whether there is a substitution that makes a given goal true in an equational theory, or to find a set of substitutions that will generate all possible solutions. This it the problem of *E-unification*[1]. The problem arises in automated deduction, formal verification, type inference and many other areas of computer science. Often deductions are performed modulo a given equational theory, and calls to an E-unification procedure are made constantly through a deductive process.

Unfortunately, E-unification is undecidable in general, even for the problem of only deciding whether a solution exists. However, even when it is decidable, it may have very high complexity. Therefore, it would be useful to have some way of reducing the number of calls to the procedure. One way of doing this would be to approximate E-unification. By this we mean to use an efficient algorithm for deciding E-unification that is complete but not sound. This means that if it says that the E-unification problem is not satisfiable, then we can be sure of the answer. If it says it is satisfiable, we must then run a sound E-unification procedure to determine if that answer is correct. A deductive procedure could then call the approximating algorithm before calling the real algorithm, and avoid many unnecessary calls to the real procedure.

In this paper, we give such an approximating E-unification algorithm. The running time of the algorithm is linear if the equational theory is considered

*Department of Math and Computer Science, Clarkson University, Potsdam, NY 13699-5815. Phone: (315) 268-2384. Fax: (315) 268-2371. Email: clynch@clarkson.edu

constant, and cubic if the equational theory is considered as part of the input. Our algorithm will approximate any E-unification problem for any equational theory.

What we exactly do in this paper is to show that E-unification is decidable for equational theories such that each equation has no repeated variables and the goal has no repeated variables, with the complexity given above. This class of theories contains all ground equations, but it is larger than that. The theory is not very interesting in its own right, but it is interesting for the purpose of approximating E-unification.

The way the approximation works is to take each equation in the goal and the equational theory and rename the variables so that no variable occurs more than once. If this new E-unification problem is unsatisfiable, then the original one is also.

In order to create the decision procedure, we first create a sound and complete goal-directed inference procedure for the case where the equational theory is linear and the goal contains no repeated variables. The inference rules given for this procedure are similar to the inference rules for Syntactic Theories[4, 6, 7], but in our case we do not require that the theory is syntactic. An interesting feature of this inference system is that we can prove the completeness with Eager Merging. The problem of proving completeness with Eager Merging (or Eager Variable Elimination) is an open problem for many goal-directed inference systems[2, 3, 5]. Since so little is known about it, it is interesting to have a completeness result with Eager merging, even for such a restricted inference system. It is difficult to create goal-directed inference systems for equational theories. However, we believe they are useful for directing the search for an E-unifier. We have created a related goal-directed inference system in [8]. That system is complete for all equational theories, however it has more rules and does not have Eager Merging.

In the conclusion of the paper, we show that our approximation technique is not only useful for E-unification. It is also useful for Logic Programming, where we must decide if there is a substitution that can be applied to the goal so that the goal follows from the given Horn Clauses. We get similar nice complexity results for this case. Our techniques have some similarities to what is done for Local Theories[10], but they are only concerned with universal goals, and we can cover existential goals.

The format of the paper is as follows: After some preliminary definitions, we give the inference rules and a soundness and completeness result with Eager Merging. After that, we give the algorithm for the restricted case with no repeated variables and give our complexity results. To conclude, we talk about the relationship with approximation and with Local Theories. Some missing proofs will appear in a longer version of the paper[9].

In the introduction, we will first talk about why E unification is an important problem. We will give an informal definition of deciding unification, solving unification, and deciding the word problem. We will talk about goal directed inference systems and why they are important. We will talk about how we want to have decision procedures for some classes of theories. And we will give an idea of the results we get in this paper.

2 Preliminaries

We assume we are given a set of variables and a set of uninterpreted function symbols of various arities. An arity is a non-negative integer. *Terms* are defined recursively in the following way: each variable is a term, and if $t_1, \cdots, t_n$ are terms, and f is of arity $n \geq 0$, then $f(t_1, \cdots, t_n)$ is a term, and f is the symbol at the *root* of $f(t_1, \cdots, t_n)$. A term (or any object) without variables is called *ground*. We consider equations of the form $s \approx t$, where s and t are terms. Let E be a set of equations, and $u \approx v$ be an equation, then we write $E \models u \approx v$ (or $u =_E v$) if $u \approx v$ is true in any model of E. If G is a set of equations, then $E \models G$ means that $E \models e$ for all e in G.

A *substitution* is a mapping from the set of variables to the set of terms, such that it is almost everywhere the identity. We identify a substitution with its homomorphic extension. If θ is a substitution then $Dom(\theta) = \{x \mid x\theta \neq x\}$. A substitution θ is an *E-unifier* of an equation $u \approx v$ if $E \models u\theta \approx v\theta$. θ is an *E-unifier* of a set of equations G if θ is an E-unifier of all equations in G.

If σ and θ are substitutions, then we write $\sigma \leq_E \theta[Var(G)]$ if there is a substitution ρ such that $E \models x\sigma\rho \approx x\theta$ for all x appearing in G. If G is a set of equations, then a substitution θ is a *most general unifier of* G, written $\theta = mgu(G)$ if θ is an E unifier of G, and for all E unifiers σ of G, $\theta \leq_E \sigma[Var(G)]$. A complete set of E-unifiers of G, is a set of E-unifiers Θ of G such that for all E-unifiers σ of G, there is a θ in Θ such that $\theta \leq_E \sigma[Var(G)]$.

Given a unification problem we can either solve the unification problem or decide the unification problem. Given a goal G and a set of equations E, to *solve* the unification problem means to find a complete set of E-unifiers of G. To *decide* the unification problem simply means to answer true or false as to whether G has an E-unifier. In this paper, we consider both of these problems.

We say that a term t (or an equation or a set of equations) has *varity* n if each variable in t appears at most n times. An equation $s \approx t$ is linear if s and t are both of varity 1. Note that the equation $s \approx t$ is then of varity 2, but it might not be of varity 1. A set of equations is *linear* if each equation in the set is linear. For example, the axioms of group theory $(\{f(x,, f(y,z)) \approx f(f(x,y),z), f(w,e) \approx w, f(u,i(u)) \approx e.$ are of varity 2.

Let G be a set of equations. We define an *equation graph* of G by letting the equations of G be the nodes, and placing an edge from an equation e_1 to an equation e_2 if there is an occurrence of a variable x in e_1 and a different occurrence of x in e_2. For example if $G = \{f(x) \approx g(x), f(y) \approx g(z), f(z) \approx g(y)\}$, then there is an edge from $f(x) \approx g(x)$ to itself, since there are two occurrences of x in $f(x) \approx g(x)$. Also, there are two edges between $f(y) \approx g(z)$ and $f(z) \approx g(y)$, because two different variables occur in both equations. Paths and cycles are defined as usual. In the above example, we have described two different cycles. A set of equations G is defined to be *cyclic* if the equation graph of G has a cycle.

3 Inference Rules

In this section, we will give a set of inference rules for finding a complete set of E-unifiers of a goal G, and in the following sections we prove that for a linear equational theory E, every goal G of varity 1 and substitution θ such that $E \models G\theta$ can be converted into a *normal form* which determines a substitution which is more general than θ. The inference rules decompose an equational proof by choosing a potential step in the proof and leaving what is remaining when that step is removed.

An equation $x \approx t$ appearing in G where x only appears once in G is called *solved*. We define the *unsolved part of* G to be the set of all equations in G that are not solved.

As in Logic Programming, we can have a selection rule for goals. For each goal G, we don't-care nondeterministically select an equation $u \approx v$ from G, such that u and v are not variables. We say that $u \approx v$ is *selected in* G. If there is no such equation $u \approx v$ in the goal, then nothing is selected. We will see in the next section that if nothing is selected, then the goal is in normal form and a most general E-unifier can be easily determined.

The inference rules are given in figure 1. Except for Mutate, these are the usual inference rules for syntactic unification. Notice that Orientation is not allowed in the case of variable-variable equations. We want to avoid infinite loops in such cases.

The Mutate rule is so-called because it is similar to the inference rule Mutate that is used in the inference procedure for syntactic theories[4]. The rule assumes that there is an equational proof of the goal equation at the root of the equation. If one of the equations in this proof is $s \approx t$ then that breaks up the proof at the root into two separate parts.

The Mutate Rule for Syntactic Theories performs a Decomposition on $u \approx s$ and $v \approx t$ immediately after the Mutate rule. However, that is only complete for Syntactic Theories. Our inference procedure is not just for syntactic theories, and such a rule is not complete in our case. Our inference system would be complete if we allowed an immediate Decomposition on $v \approx t$ but not $u \approx s$. This is proved in our full paper[9]. However, for the complexity results we need for this paper, we don't need the immediate Decomposition, so we will use the simpler version for this paper.

We will write $G \longrightarrow G'$ to indicate that G goes to G' by one application of an inference rule. Then $\stackrel{*}{\longrightarrow}$ is the reflexive, transitive closure of $\longrightarrow$.

We want our inference rules to be applied deterministically or don't-care nondeterministically whenever possible. Therefore, we restrict the application of our inference rules in the following way:

Assume that Trivial, Orient and Merge are performed eagerly whenever applicable. It is usual in inference systems for the Trivial and Orient rules to be performed eagerly. However, it is an open question in many inference systems whether the Merge rule can be applied eagerly. Since we restrict our inference rules to the case where E is linear and G is of varity 1, we can prove the completeness when Merge is performed eagerly. Eager inferences are a form of

Decomposition:

$$\frac{\{f(s_1,\cdots,s_n) \approx f(t_1,\cdots,t_n)\} \cup G}{\{s_1 \approx t_1,\cdots,s_n \approx t_n\} \cup G}$$

Mutate:

$$\frac{\{u \approx v\} \cup G}{\{u \approx s, t \approx v\} \cup G} \quad \text{where } s \approx t \in E.\ ^{a}$$

Merge:

$$\frac{\{x \approx s, x \approx t\} \cup G}{\{x \approx s, s \approx t\} \cup G}$$

Orient:

$$\frac{\{t \approx x\} \cup G}{\{x \approx t\} \cup G} \quad \text{where } t \text{ is not a variable.}$$

Trivial:

$$\frac{\{t \approx t\} \cup G}{G}$$

[a] For simplicity, we assume that E is closed under symmetry.

Figure 1: The inference rules

determinism, because when inferences are performed eagerly, that means that there is no need to backtrack and try other rules.

For the Decomposition and Mutate rules, we will assume that they must be performed on a selected equation. That is a source of don't-care nondeterminism in our procedure. However, there are still some sources of don't-know nondeterminism. If Decomposition and Mutate are both applicable to the selected equation, we don't know which one to do, and therefore have to try them both. Similarly, there may be more than one equation we can use in order to perform Mutate on the selected equation. In that case, we must also try all the possibilities.

We will prove that the above inference rules solve a goal G by transforming it into normal forms representing a complete set of E-unifiers of G.

4 Normal Form

In order for the final result of the procedure to determine a unifier, it must not be cyclic. We will consider a goal G of varity 1 and a set of linear equations. Since G is of varity 1, it is not cyclic. We need to show that that property is preserved. Also, since G is of varity 1, it is trivially of varity 2. We prove that that property is preserved.

Lemma 1 *Assume the unsolved part of G is of varity 2 and E is linear. If $G \xrightarrow{*} G'$, then the unsolved part of G' is of varity 2.*

Lemma 2 *Suppose that E is linear and G is not cyclic. If $G \xrightarrow{*} H$, then H is not cyclic.*

Corollary 1 *Suppose that E is linear and G has varity 1. If $G \xrightarrow{*} H$, then H is not cyclic.*

A goal G is in *normal form* if the equations of G are all of the form $x \approx t$, where x is a variable, and the equations of G can be arranged in the form $\{x_1 \approx t_1, \cdots, x_n \approx t_n\}$ such that for all $i \leq j$, x_i is not in t_j. Then define θ_G to be the substitution $[x_1 \mapsto t_1][x_2 \mapsto t_2] \cdots [x_n \mapsto t_n]$. θ_G is a most general E-unifier of G. Notice that if a noncyclic goal has no selected equation, then the goal is in normal form, since Merge is applied eagerly.

5 Soundness

Theorem 1 *The procedure in Figure 1 is sound, i.e. if $G' \xrightarrow{*} G$ and G is in normal form, then $E \models G'\theta_G$.*

6 A Bottom Up Inference System

In order to prove the completeness of this procedure, we first define an equational proof using Congruence and Equation Application rules. We prove that this equational proof is equivalent to the usual definition of equational proof, which involves Reflexivity, Symmetry, Transitivity and Congruence.

We will define completeness with respect to any equational theory obtained by the following rules of inference from a set of equations closed under symmetry:

$$\text{Congruence:} \quad \frac{s_1 \approx t_1 \cdots s_n \approx t_n}{f(s_1, \cdots, s_n) \approx f(t_1, \cdots, t_n)}$$

$$\text{Equation Application:} \quad \frac{u \approx s \quad t \approx v}{u \approx v},$$

if $s \approx t$ is a ground instance of an equation in E.

For Congruence $n \geq 0$. In the special case where $n = 0$, f is a constant.

We define $E \vdash u \approx v$ if there is a proof of $u \approx v$ using the Congruence and Equation Application rules. If π is a proof, then $|\pi|$ is the number of steps in the proof. $|u \approx v|_E$ is the number of steps in the shortest proof of $u \approx v$.

We need to prove that $\{u \approx v \mid E \vdash u \approx v\}$ is closed under Reflexivity, Symmetry and Transitivity. First we prove Reflexivity.

Lemma 3 *Let E be an equational theory. Then $E \vdash u \approx u$ for all u.*

Next we prove closure under symmetry.

Lemma 4 *Let E be an equational theory such that $E \vdash u \approx v$ and $|u \approx v|_E = n$. Then $E \vdash v \approx u$, and $|v \approx u|_E = n$.*

Next we show closure under Transitivity.

Lemma 5 *Let E be an equational theory such that $E \vdash s \approx t$ and $E \vdash t \approx u$. Suppose that $|s \approx t|_E = m$ and $|t \approx u|_E = n$. Then $E \vdash s \approx u$, and $|s \approx u|_E \leq m + n$.*

Closure under Congruence is trivial. Now we put these lemmas together to show that anything true under the semantic definition of Equality is also true under the syntactic definition given here.

Theorem 2 *If $E \models u \approx v$, then $E \vdash u \approx v$.*

7 Completeness

Now we come to the completeness of the inference rules given in Figure 1, where E is linear, and G is of varity 1.

First we define a measure on the equations in the goal, which will be used in the completeness proof.

Definition 1 *Let E be an equational theory and G be a goal. Let θ be a substitution such that $E \models G\theta$. We will define a measure μ, parameterized by θ and G. Define $\mu(G,\theta)$ to be the triple (m,n,p), where m is the sum of all $|u\theta \approx v\theta|_E$, where $u \approx v$ is an unsolved equation of G, n is the number of unsolved equations in G, and p is the number of equations of the form $t \approx x$, where x is a variable and t is not. We will compare these triples lexicographically.*

Now we come to the completeness theorem:

Theorem 3 *Suppose that E is an equational theory, G is a set of goal equations, E is linear, G is of varity 2, the unsolved part of G is not cyclic, and θ is a ground substitution. If $E \models G\theta$ then there exists a goal H such that $G \xrightarrow{*} H$ and $\theta_H \leq_E \theta[Var(G)]$.*

Proof. Let G be a set of goal equations, and θ a ground substitution such that $E \models G\theta$. Let $\mu(< G, \theta >) = M$. We will prove by induction on M that there exists a goal H such that $G \xrightarrow{*} H$ and $\theta_H \leq_E \theta[Var(G)]$. In addition we will prove that if a Merge, Orient or Trivial rule is applicable to G, and if the application of that rule yields G', then there exists a goal H such that $G' \xrightarrow{*} H$ and $\theta_H \leq_E \theta[Var(G)]$.

If nothing is selected in G, then G must be in normal form, and θ_G is the most general unifier of G, so $\theta_G \leq_E \theta[Var(G)]$.

Now suppose that one of the eager rules applies to an equation in E, resulting in G'. We need to show that there is an extension θ' of θ such that $E \models G'\theta'$, and $\mu(G',\theta') < \mu(G,\theta)$. Then by the induction assumption there is an H such that $G' \xrightarrow{*} H$ with $\theta_H \leq_E \theta'[Var(G')]$. This implies that $G \xrightarrow{*} H$ and $\theta_H \leq_E \theta[Var(G)]$.

Case 1: Suppose that Merge is Applicable. So $G = \{u \approx s, u \approx t\} \cup G_1$, where u is a variable. Then Merge is eagerly applied, and the new goal, $G' = \{u \approx s, s \approx t\} \cup G_1$, where $u \approx s$ is solved. By lemmas 4 and 5 $|s\theta \approx t\theta|_E \leq |u\theta \approx s\theta|_E + |u\theta \approx t\theta|_E$. Hence the first component of the measure, $\mu(G',\theta)$ will either stay the same as $\mu(G,\theta)$ or will decrease. In both cases the number of unsolved equations will decrease at least by 1, hence $\mu(G',\theta) < \mu(G,\theta)$. Also, by symmetry and transitivity, $E \models G'\theta$.

Case 2: If Orient is applicable, then G is of the form $\{t \approx x\} \cup G_1$. An application of Orient decreases the measure μ, because although its first component remains the same, the second one may get decreased, and even if it stays the same the third component is decreased by 1. It also yields an equivalent set of equations, so θ is still an E-unifier.

Case 3: If Trivial is applied, the measure gets decreased in the first component, and θ is still an E-unifier.

If some equation is selected in G, we will prove that there is a goal G' and a extension θ' of θ such that $E \models G'\theta'$, $G \longrightarrow G'$, and $\mu(G',\theta') < \mu(G,\theta)$.

Then by the induction assumption there is an H such that $G' \xrightarrow{*} H$ with $\theta_H \leq_E \theta'[Var(G')]$. This implies that $G \xrightarrow{*} H$ and $\theta_H \leq_E \theta[Var(G)]$.

So assume that some equation $u \approx v$ is selected in G. Then G is of the form $\{u \approx v\} \cup G_1$. Neither u nor v is a variable, since $u \approx v$ is selected.

Consider the rule used at the root of the proof tree that $E \vdash u\theta \approx v\theta$. This was either an application of Congruence or Equation Application.

Case 1: Suppose the rule at the root of the proof tree of $E \vdash u\theta \approx v\theta$ is an Equation Application. Then there exists a ground instance $s\theta' \approx t\theta'$ of an equation $s \approx t$ in E, such that $E \vdash u\theta' \approx s\theta'$ and $E \vdash t\theta' \approx v\theta'$, where θ' is an extension of θ such that $s\theta' = t\theta'$. Let $|u\theta' \approx s\theta'|_E = p$. Let $|t\theta' \approx v\theta'|_E = q$. Then $|u\theta \approx v\theta|_E = p + q + 1$.

$$\frac{\begin{array}{cc} \vdots & \vdots \\ u\theta \approx s\theta' & t\theta' \approx v\theta \end{array}}{u\theta \approx v\theta}$$

There is an application of Mutate that can be applied to $u \approx v$, resulting in the new goal $G' = \{u \approx s, t \approx v\} \cup G_1$.

$$\frac{\begin{array}{c} \vdots \\ \{u \approx v\} \cup G_1 \end{array}}{\{u \approx s, t \approx v\} \cup G_1}$$

Then $|u\theta' \approx s\theta'|_E = p$, and $|t\theta' \approx v\theta'|_E = q$, and $p + q < |u\theta \approx v\theta|_E = p + q + 1$. so $\mu(G', \theta') < \mu(G, \theta)$. Note that θ' is an E-unifier of G'.

Case 2 Now suppose that the rule at the root of the proof tree of $E \vdash u\theta \approx v\theta$ is an application of Congruence.

Then $u = f(u_1, \cdots, u_n)$, $v = f(v_1, \cdots, v_n)$ and $E \vdash u_i\theta \approx v_i\theta$ for all i.

$$\frac{\begin{array}{ccc} \vdots & & \vdots \\ u_1\theta \approx u_1\theta & \cdots & v_n\theta \approx v_n\theta \end{array}}{f(u_1, \cdots, u_n)\theta \approx f(v_1, \cdots, v_n)\theta}$$

There is an application of Decomposition that can be applied to $u \approx v$, resulting in the new goal $G' = \{u_1 \approx v_1, \cdots, u_n \approx v_n\} \cup G_1$. Then $\Sigma_{1 \leq i \leq n} |u_i\theta \approx v_i\theta|_E < |u\theta \approx v\theta|_E = \Sigma_{1 \leq i \leq n} |u_i\theta \approx v_i\theta|_E + 1$, so $\mu(G', \theta) < \mu(G, \theta)$. Note that θ' is an E-unifier of G'.

□

Corollary 2 *Suppose that E is an equational theory, G is a set of goal equations, E is linear, G is of varity 1, and θ is a ground substitution. If $E \models G\theta$ then there exists a goal H such that $G \xrightarrow{*} H$ and $\theta_H \leq_E \theta[Var(G)]$.*

8 Decidability and Complexity of Varity 1

In this section, we will assume that E and G both have varity 1. We will give an algorithm to decide the E-unification problem, but we will not find a complete set of unifiers.

Since E and G have varity 1, the inference rules in Figure 1 are complete by Theorem 3. In addition we give an algorithm to decide the E-unification problem in time $O((|G| + |E|) \times |E| \times n)$, where $|G|$ is the size of G, $|E|$ is the size of E, and n is the number of equations in E.

The algorithm for this is presented in figure 2. Given a goal G the function *visit* is called on each equation e in G. If it returns T, then e is E-unifiable. If it returns F, then e is not E-unifiable. If it returns T for all of the equations in G, then G is E-unifiable. This is because no two equations share variables.

The function *visit* first returns T if e is a variable term pair or a trivial equation. Otherwise *visit* creates a list of lists of equations. Each such list consists of the equations that can be derived from e with the applicable rules of the top-down inference system. For example, suppose that we call *visit*($fgfx \approx fggy$), where $E = \{ffw \approx fgz\}$.[1] Then the list that is created for e is the following: $((fgfx \approx ffw, fgz \approx fggy)(fgfx \approx fgz, ffw \approx fgfx)(gfx \approx ggy))$. This is a list containing three lists. Each of the first two lists is the result of a Mutate inference. The third list is the result of a Decomposition. Each list will always contain two equations in the case of Mutate, and for Decomposition the number of equations will be equal to the arity of the function symbol at the root of the two sides of the equation. If there are n equations in E, then the number of such lists is at most $2n + 1$.

Each equation in each list is a subproblem to be solved, and if one whole list becomes true, then e is also true. *visit* then attempts to solve each of those subproblems e'. If e' is not already in the process of being solved, it is visited. If e' has already been visited. Then it may have already been solved.

If e' has not been solved yet, then T will not be returned from the function. In that case, it may be that e' is not true. But it also may be that e' is still in the process of being solved. We set a pointer from the node e' to this list. This pointer is called a *back-edge*, because it is similar to the back-edge from the depth-first search algorithm. If e' is ever proved true, then the *update* function will eventually follow this back-edge and handle the consequences.

If e' has already been solved, or if the call to *visit*(e') returns T then the function *update* is called to deal with the consequence. The first thing *update* does is to remove e' from the list, since it is solved. If the list is now empty, then e has been proved true, and we follow back-edges and call *update* recursively. The *update* procedure is essential for the efficiency of *visit*.

Proposition 1 *If G is of varity 1, then Merge does not apply to G.*

Lemma 6 *Assume G and E are of varity 1. If $G \xrightarrow{*} G'$, then G' is of varity 1.*

[1] We assume E is closed under symmetry, so $fgz \approx ffw$ is also assumed to be in E.

```
function visit(e)
if one side of e is a variable
    return T
if e is of the form t ≈ t
    return T
create list for e
for each L in list
    for each e' in L
        if e'-list exists
            answer := e'-list
        else
            answer := visit(e')
        if answer = T
            solved = update(e', L, e)
            if solved = T
                return T
        else
            set back edge from node e' to L
return F // but maybe we have just not solved it yet


function update(e', L, e)
remove e' from L
if L is empty
    set e-list to T // this removes associated back edges
    for each back edge from e to L' in e''-list
        update(e, L', e'')
    return T
else
    return F
```

Figure 2: Algorithm

Since Merge will never apply in our derivations, we can view a derivation of a goal equations as a tree. Such a tree is a top-down proof that a goal is E-unifiable. It also gives us an E-unifier, but we will not be concerned what the E-unifier is, only its existence. The next lemma tells us that any goal that is E-unifiable has a top-down proof such that no equation is a renaming of an descendant of itself. The reason for this is because we could then replace the ancestor with a renaming of the subtree rooted by the descendant and get a new top-down proof for the goal. The new proof might give a different E-unifier, but it would still be an E-unifier.

Lemma 7 *If $E \models e$, then there is a top-down proof that e is E-unifiable, such that no equation is a descendant of a renaming of itself.*

Theorem 4 *(Soundness/Completeness) Let E be a set of equations and e be an equation, both of varity 1. Then there exists an E-unifier of e if and only if $visit(e) = T$.*

Proof. If $visit(e) = T$, then that gives a top-down proof that e is E-unifiable. We need to prove the other direction. So assume that e is E-unifiable. We need to prove that $visit(e) = T$. The proof is by induction on the size of the top-down proof that e is E-unifiable.

Case 1 Assume that in the top-down proof, the Mutate rule was applied to an equation e, i.e. e is $u \approx v$, and the proof is of the following form:

$$\dfrac{\begin{matrix}\vdots\\ u \approx v\end{matrix}}{\begin{matrix}u \approx s & t \approx v\\ \vdots & \vdots\end{matrix}}$$

where $s \approx t \in E$.

$visit(u \approx v)$ will initialize a list of consequences for $u \approx v$, and $L = (u \approx s, t \approx v)$ will be one of its elements, as a consequence of the Mutate rule. The algorithm will proceed with checking each list in the list of consequences and either it finds the proof earlier (we are done) or it will eventually check the list L. By the induction hypothesis, $visit(u \approx s) = T$ and $visit(t \approx v) = T$, because both equations have shorter proofs. But there are 2 cases here (we will consider them for $u \approx s$ only, for $t \approx v$ the cases are the same):

1. $u \approx s$ has already been visited and the list for it was initiated. Then the list can be equal to T or not yet.
2. $u \approx s$ has not yet been visited.

In the first case, $visit(u \approx v)$ will not call $visit(u \approx s)$. If the list for $u \approx s$ is equal to T, the equation is erased from the list L, and we are done. If the list is not equal to T, it means that *visit*($u \approx s$) was already called in the run of the algorithm, but didn't finish. Then a back-edge is created from $u \approx s$ to the list L. We know by the inductive assumption that eventually *visit*($u \approx s$) will return T, and the equation will be removed from L by the update function.

The second case is similar to the first. By the induction argument we know, that *visit*($u \approx s$) will return T.

The same applies for $t \approx v$.

When the second equation becomes T, the list L will be empty, and the function *visit* will then return T.

Case 2 Assume that in the top-down proof the Decomposition rule was applied to an equation e, i.e. $e = f(u_1, \cdots, u_n) \approx f(v_1, \cdots, v_n)$, and the proof is of the form:

$$\dfrac{\begin{array}{c}\vdots\\ f(u_1, \cdots, u_n) \approx f(v_1, \cdots, v_n)\end{array}}{u_1 \approx v_1, \cdots, u_n \approx v_n}$$

The argument here is similar to the argument in Case 1.

□

The complexity argument relies on the fact that only certain equations will be created by the algorithm.

Lemma 8 *Assume that* $e \xrightarrow{*} G'$*. Then for every* $s \approx t \in G'$*, either*

1. $s \approx t$ *is obtained by a sequence of Decompositions of* e*, or*

2. s *is a renaming of a subterm of a term appearing in* $E \cup \{e\}$ *and* t *is a renaming of a subterm of a term appearing in* E *(or vice versa).*

Theorem 5 *visit*(e) *has complexity* $O((|e| + |E|) \times |E| \times n)$*, where* n *is the number of equations in* E*.*

Proof. First we will proof that there are only $O(|e| + |E|) \times |E|)$ possible equations in the proof.

It follows from Lemma 8 that we can bound the number of possible equations:

1. the equations obtained by the Decomposition of the goal, e, hence $|e|$ equations;

2. the equations of the form $s \approx t$, where s is a subterm of $E \cup \{e\}$ and t is a subterm of $|E|$, hence $(|e| + |E|) \times |E|$ equations.

For each equation the algorithm initializes a list of equations, that in the worst case, can be length $2n + 1$.

Hence, there will be only $O((|e| + |E|) \times |E| \times n)$ equations in the lists initialized during the run of the algorithm in the worst case.

Now, when *visit* is called on the goal, e, it may go on to create all possible equations in the worst case, i.e. $O(|e| + |E|) \times |E|)$ equations. (*visit* is called only once on each equation.) For each of them it can go through the list of $O(n)$ equations, hence overall *visit* can take $O((|G| + |E|) \times |E| \times n)$ time.

We must also consider the time for *update*. *update* in the worst case may remove all the equations from all the lists, i.e. it will take $O((|G|+|E|) \times |E| \times n)$ time. □

If we want to decide whether a goal G is E-unifiable, and we consider G to be the input to the algorithm, then we can consider $|E|$ as a constant, and the running time is linear in the size of the input. However, if the input to the algorithm is considered to be both the goal G and the set of equations E, then the running time for the algorithm is cubic in the size of the input.

9 Conclusion

We have given an inference procedure, similar to the inference procedure for Syntactic Theories[4, 6, 7], which gives a complete set of E-unifiers for a linear equational theory with a goal with no repeated variables. We show it is complete even with Eager Merging. The problem of Eager Merging is an open problem for many inference systems[2, 3, 5].

We also gave an algorithm for deciding E-unification for theories and goals with no repeated variables. The algorithm is linear if the equational theory is considered to be constant, and cubic if it is considered to be part of the input.

The algorithm is useful for approximating any E-unification problem. Suppose we want to decide if a goal G is E-unifiable. We then rename the variables in E and G so that there are no repeated variables. Run the algorithm. If the answer is NO, then the answer to the original problem is NO also. If the answer is YES, we must run an E-unification procedure for the original problem.

The same idea can be used to decide whether a goal is unifiable in a Horn Clause Theory. By the same techniques used in this paper, we can show that if each Horn Clause is such that no variable is repeated in the body, no variable is repeated in the head, and when a variable x appears in both the body and the head, then if x appears in a term t in the body, t also must appear in the head. If the goal has no repeated variables, then our techniques will show that this problem is decidable and efficient. This is similar to what is done in Local Theories[10], except that we can decide existential problems, while Local Theories only talk about universal problems.

References

[1] F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge, 1998.

[2] J. Gallier and W. Snyder. A general complete E-unification procedure. In *RTA 2*, ed. P. Lescanne, LNCS vol. 256, 216-227, 1987.

[3] J. Gallier and W. Snyder. Complete sets of transformations for general E-unification. In *TCS*, vol. 67, 203-260, 1989.

[4] C. Kirchner. Computing unification algorithms. In *Proceedings of the First Symposium on Logic in Computer Science*, Boston, 200-216, 1990.

[5] C. Kirchner and H. Kirchner. *Rewriting, Solving, Proving.* `http://www.loria.fr/~ckirchne/` , 2000.

[6] C. Kirchner and F. Klay. Syntactic Theories and Unification. In *LICS 5*, 270-277, 1990.

[7] F. Klay. Undecidable Properties in Syntactic Theories. In *RTA 4*,ed. R. V. Book, LNCS vol. 488, 136-149, 1991.

[8] C. Lynch and B. Morawska. Goal Directed E-Unification. Submitted.

[9] C. Lynch and B. Morawska. Approximating E-Unification (full version). `http://www.clarkson.edu/~clynch/papers/approx_full.ps/`, 2001.

[10] D. McCallester. Automated Recognition of Tractability in Inference Relations. In *Journal of the ACM*, vol.40(2) , pp. 284-303, 1993.