Rigid E-Unification Revisited*

A. Tiwari[1], L. Bachmair[1], H. Ruess[2]

[1] Department of Computer Science, SUNY at Stony Brook,
Stony Brook, NY 11794, U.S.A
{leo,astiwari}@cs.sunysb.edu
[2] SRI International, Menlo Park, CA 94025, U.S.A
ruess@csl.sri.com

Abstract. This paper presents a sound and complete set of abstract transformation rules for rigid E-unification. Abstract congruence closure, syntactic unification and paramodulation are the three main components of the proposed method. The method obviates the need for using any complicated term orderings and easily incorporates suitable optimization rules. Characterization of substitutions as congruences allows for a comparatively simple proof of completeness using proof transformations. When specialized to syntactic unification, we obtain a set of abstract transition rules that describe a class of efficient syntactic unification algorithms.

1 Introduction

Rigid E-unification arises when tableaux-based theorem proving methods are extended to logic with equality. The general, simultaneous rigid E-unification problem is undecidable [7] and it is not known if a complete set of rigid E-unifiers in the sense of [10] gives a complete proof procedure for first-order logic with equality. Nevertheless complete tableau methods for first-order logic with equality can be designed based on incomplete, but terminating, procedures for rigid E-unification [8]. A simpler version of the problem is known to be decidable and also NP-complete, and several corresponding algorithms have been proposed in the literature (not all of them correct) [9, 10, 5, 8, 11, 6]. In the current paper, we consider this standard, non-simultaneous version of the problem.

Most of the known algorithms for finding a complete set of (standard) rigid unifiers employ techniques familiar from syntactic unification, completion and paramodulation. Practical algorithms also usually rely on congruence closure procedures in one form or another, though the connection between the various techniques has never been clarified. The different methods that figure prominently in known rigid unification procedures—unification, narrowing, superposition, and congruence closure—have all been described in a framework based on transformation rules. We use the recent work on congruence closure as a starting point [12, 4] and formulate a rigid E-unification method in terms of fairly abstract transformation rules.

* The research described in this paper was supported in part by the National Science Foundation under grant CCR-9902031.

This approach has several advantages. For one thing, we provide a concise and clear explication of the different components of rigid E-unification and the connections between them. A key technical problem has been the integration of congruence closure with unification techniques, the main difficulty being that congruence closure algorithms manipulate term structures over an *extended* signature, whereas unifiers need to be computed over the original signature. We solved this problem by rephrasing unification problems in terms of congruences and then applying proof theoretic methods, that had originally been developed in the context of completion and paramodulation. Some of the new and improved features of the resulting rigid E-unification method in fact depend on the appropriate use of extended signatures.

Almost all the known rigid E-unification algorithms require relatively complicated term orderings. In particular, most approaches go to great length to determine a suitable orientation of equations (between terms to be unified), such as $x \approx fy$, a decision that depends of course on the terms that are substituted (in a "rigid" way) for the variables x and y. But since the identification of a substitution is part of the whole unification problem, decisions about the ordering have to made during the unification process, either by orienting equations non-deterministically, as in [10], or by treating equations as bi-directional constrained rewrite rules (and using unsatisfiable constraints to eliminate wrong orientations) [5]. In contrast, the only orderings we need are simple ones in which the newly introduced constants are smaller than all other non-constant terms. The advantage of such simple orderings is twofold, in that not only the description of the rigid E-unification method itself, but also the corresponding completeness proofs, become simpler.[1] Certain optimizations can be easily incorporated in our method that reduce some of the non-determinism still inherent in the unification procedure. The treatment of substitutions as congruences defined by special kinds of rewrite systems (rather than as functions or morphisms) is a novel feature that allows us to characterize various kinds of unifiers in proof-theoretic terms via congruences.

As an interesting fallout of this work we obtain an abstract description of a class of efficient syntactic unification algorithms based on recursive descent. Other descriptions of these algorithms are typically based on data structures and manipulation of term dags. Since our approach is suitable for abstractly describing sharing, we obtain a pure rule based description.

One motivation for the work presented here has been the generalization of rigid E-unification modulo theories like associativity and commutativity, which we believe are of importance for theorem proving applications. Our approach,

[1] A key idea of congruence closure is to employ a concise and simplified term representation via variable abstraction, so that complicated term orderings are no longer necessary or even applicable. There usually is a trade-off between the simplicity of terms thus obtained and the loss of term structure [4]. In the case of rigid unification, we feel that simplicity outweighs the loss of some structure, as the non-determinism inherent in the procedure limits the effective exploitation of a more complicated term structure in any case.

especially because of the use of extensions of signatures and substantially weaker assumptions about term orderings, should more easily facilitate the development of such generalized unification procedures.

We also believe that our way of describing rigid E-unification will facilitate a simpler proof of the fact that the problem is in NP. Previous proofs of membership of this problem in NP "require quite a bit of machinery" [10]. The weaker ordering constraints, a better integration of congruence closure and a rule-based description of the rigid E-unification procedure should result in a simpler proof.

2 Preliminaries

Given a set $\Sigma = \cup_n \Sigma_n$ and a disjoint set V, we define $\mathcal{T}(\Sigma, V)$ as the smallest set containing V and such that $f(t_1, \ldots, t_n) \in \mathcal{T}(\Sigma, V)$ whenever $f \in \Sigma_n$ and $t_1, \ldots, t_n \in \mathcal{T}(\Sigma, V)$. The elements of the sets Σ, V and $\mathcal{T}(\Sigma, V)$ are respectively called *function symbols*, *variables* and *terms* (over Σ and V). The set Σ is called a *signature* and the index n of the set Σ_n to which a function symbol f belongs is called the *arity* of the symbol f. Elements of arity 0 are called *constants*. By $\mathcal{T}(\Sigma)$ we denote the set $\mathcal{T}(\Sigma, \emptyset)$ of all variable-free, or *ground* terms. The symbols $s, t, u, \ldots$ are used to denote terms; $f, g, \ldots$, function symbols; and $x, y, z, \ldots$, variables.

A *substitution* is a mapping from variables to terms such that $x\sigma = x$ for all but finitely many variables x. We use post-fix notation for application of substitutions and use the letters $\sigma, \theta, \ldots$ to denote substitutions. A substitution σ can be extended to the set $\mathcal{T}(\Sigma, V)$ by defining $f(t_1, \ldots, t_n)\sigma = f(t_1\sigma, \ldots, t_n\sigma)$. The *domain* $\mathcal{D}om(\sigma)$ of a substitution σ is defined as the set $\{x \in V : x\sigma \neq x\}$; and the *range* $\mathcal{R}an(\sigma)$ as the set of terms $\{x\sigma : x \in \mathcal{D}om(\sigma)\}$. A substitution σ is *idempotent* if $\sigma\sigma = \sigma$.[2]

We usually represent a substitution σ with domain $\{x_1, \ldots, x_n\}$ as a set of variable "bindings" $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$, where $t_i = x_i\sigma$. By a *triangular form* representation of a substitution σ we mean a *sequence* of bindings,

$$[x_1 \mapsto t_1 \ ; \ x_2 \mapsto t_2 \ ; \ \ldots \ ; \ x_n \mapsto t_n],$$

such that σ is the composition $\sigma_1\sigma_2 \ldots \sigma_n$ of substitutions $\sigma_i = \{x_i \mapsto t_i\}$.

Congruences

An *equation* is a pair of terms, written as $s \approx t$. The *replacement relation* $\rightarrow_{E^g}$ induced by a set of equations E is defined by: $u[l] \rightarrow_{E^g} u[r]$ if, and only if, $l \approx r$ is in E. The *rewrite relation* $\rightarrow_E$ induced by a set of equations E is defined by: $u[l\sigma] \rightarrow_E u[r\sigma]$ if, and only if, $l \approx r$ is in E and σ is some substitution. In other words, the rewrite relation induced by E is the replacement relation induced by $\cup_\sigma E\sigma$, where $E\sigma$ is the set $\{s\sigma \approx t\sigma : s \approx t \in E\}$.

[2] We use juxtaposition $\sigma\tau$ to denote function composition, i.e., $x(\sigma\tau) = (x\sigma)\tau$.

If $\to$ is a binary relation, then $\leftarrow$ denotes its inverse, $\leftrightarrow$ its symmetric closure, $\to^+$ its transitive closure and $\to^*$ its reflexive-transitive closure. Thus, $\leftrightarrow^*_{E^g}$ denotes the *congruence relation*[3] induced by E. The *equational theory* of a set E of equations is defined as the relation $\leftrightarrow^*_E$. Equations are often called *rewrite rules*, and a set E a *rewrite system*, if one is interested particularly in the rewrite relation $\to^*_E$ rather than the equational theory $\leftrightarrow^*_E$.

Substitutions as Congruences

It is often useful to reason about a substitution σ by considering the congruence relation $\leftrightarrow^*_{E^g_\sigma}$ induced by the set of equations $E_\sigma = \{x\sigma \approx x : x \in \mathcal{D}om(\sigma)\}$. The following proposition establishes a connection between substitutions and congruences.

Proposition 1. *(a) For all terms $t \in \mathcal{T}(\Sigma, V)$, $t \leftrightarrow^*_{E^g_\sigma} t\sigma$. Therefore, $E\sigma \subseteq \leftrightarrow^*_{(E\cup E_\sigma)^g}$. (b) If the substitution σ is idempotent, then for any two terms $s, t \in \mathcal{T}(\Sigma, V)$, we have $s\sigma = t\sigma$ if, and only if, $s \leftrightarrow^*_{E^g_\sigma} t$.*

Proof. Part (a) is straight-forward and also implies the "only if" direction of part (b). For the "if" direction, note that if $u \leftrightarrow_{E^g_\sigma} v$, then $u = u[l]$ and $v = v[r]$ for some equation $l \approx r$ or $r \approx l$ in E_σ. Thus, $u\sigma = (u[l])\sigma = u\sigma[l\sigma] \leftrightarrow_{(E_\sigma\sigma)^g} u\sigma[r\sigma] = (u[r])\sigma = v\sigma$. Therefore, if $s \leftrightarrow^*_{E^g_\sigma} t$, then $s\sigma \leftrightarrow^*_{(E_\sigma\sigma)^g} t\sigma$. But if σ is idempotent, then $E_\sigma\sigma$ consists only of trivial equations $t \approx t$, and hence $s\sigma$ and $t\sigma$ are identical.

Theorem 1. *Let σ be an idempotent substitution and $[x_1 \mapsto t_1; \ldots; x_n \mapsto t_n]$ be a triangular form representation of σ. Then the congruences $\leftrightarrow^*_{E^g_\sigma}$ and $\leftrightarrow^*_{(\cup_i E_{\sigma_i})^g}$ are identical, where $\sigma_i = \{x_i \mapsto t_i\}$.*

Proof. It is sufficient to prove that $E_\sigma \subseteq \leftrightarrow^*_{(\cup_i E_{\sigma_i})^g}$ and $E_{\sigma_i} \subseteq \leftrightarrow^*_{E^g_\sigma}$ for all $1 \leq i \leq n$. If $x_i\sigma \approx x_i$ is an equation in $E\sigma$, then

$$x_i\sigma = x_i\sigma_1 \ldots \sigma_n = x_i\sigma_i \ldots \sigma_n = t_i\sigma_{i+1} \ldots \sigma_n,$$

and therefore, using Proposition 1 part (a),

$$x_i \leftrightarrow^*_{E^g_{\sigma_i}} t_i \leftrightarrow^*_{E^g_{\sigma_{i+1}}} t_i\sigma_{i+1} \leftrightarrow^*_{E^g_{\sigma_{i+2}}} \cdots \leftrightarrow^*_{E^g_{\sigma_n}} t_i\sigma_{i+1} \ldots \sigma_n = x_i\sigma.$$

For the converse, using part of the above proof, we get $t_i \leftrightarrow^*_{(\cup_{k>i} E_{\sigma_i})^g} x_i\sigma \leftrightarrow^*_{E^g_\sigma} x_i$. By induction hypothesis we can assume, $E_{\sigma_k} \subseteq \leftrightarrow^*_{E^g_\sigma}$ for $k > i$, and then the above proof would establish $E_{\sigma_i} \subseteq \leftrightarrow^*_{E^g_\sigma}$.

The theorem indicates that if an idempotent substitution σ can be expressed as a composition $\sigma_1\sigma_2 \ldots \sigma_n$ of finitely many idempotent substitutions σ_i with

[3] A congruence relation is a reflexive, symmetric and transitive relation on terms that is also a replacement relation.

disjoint domains, then the congruence induced by E_σ is identical to the congruence induced by $\cup_i E_{\sigma_i}$. We denote by $E_{\boldsymbol{\sigma}}$ the set $\cup_i E_{\sigma_i}$.

The *variable dependency ordering* $\succ_V^S$ induced by a set S of equations on the set V of variables is defined by: $x \succ_V^S y$ if there exists an equation $t[x] \approx y$ in S. A finite set S of equations is said to be *substitution-feasible* if (i) the right-hand sides of equations in S are all distinct variables and (ii) the variable dependency ordering $\succ_V^S$ induced by S is well-founded. If S is a substitution-feasible set $\{t_i \approx x_i : 1 \leq i \leq n\}$ such that $x_j \not\succ_V^S x_i$ whenever $i > j$, then the idempotent substitution σ represented by the triangular form $[x_1 \mapsto t_1; \ldots; x_n \mapsto t_n]$ is called the substitution *corresponding to* S. Given an idempotent substitution σ and any triangular form representation $\sigma_1\sigma_2\ldots\sigma_n$, the sets E_σ and $\cup_i E_{\sigma_i}$ are substitution-feasible.

Rigid *E*-Unification

Definition 1. *Let E be a set of equations (over $\Sigma \cup V$) and s and t be terms in $\mathcal{T}(\Sigma, V)$. A substitution σ is called a rigid E-unifier of s and t if $s\sigma \leftrightarrow^*_{(E\sigma)^g} t\sigma$.*

When $E = \emptyset$, rigid E-unification reduces to syntactic unification.

Theorem 2. *An idempotent substitution σ is a rigid E-unifier of s and t if and only if $s \leftrightarrow^*_{(E \cup E_\sigma)^g} t$.*

Proof. Let σ be an idempotent substitution that is a rigid E-unifier of s and t. By definition we have $s\sigma \leftrightarrow^*_{(E\sigma)^g} t\sigma$. Using Proposition 1 part (a), we get

$$s \leftrightarrow^*_{E_\sigma^g} s\sigma \leftrightarrow^*_{(E \cup E_\sigma)^g} t\sigma \leftrightarrow^*_{E_\sigma^g} t.$$

Conversely, suppose σ is an idempotent substitution such that $s \leftrightarrow^*_{(E \cup E_\sigma)^g} t$. Then, $s\sigma \leftrightarrow^*_{(E\sigma \cup E_\sigma\sigma)^g} t\sigma$. But $(E_\sigma)\sigma$ consists of trivial equations of the form $t \approx t$ and hence we have $s\sigma \leftrightarrow^*_{(E\sigma)^g} t\sigma$.

If the substitution σ is not idempotent, the above proof does not go through as the set $(E_\sigma)\sigma$ may contain non-trivial equations. However, we may use Theorem 1 to establish that the congruences induced by $E \cup E_\sigma$ and $E \cup E_{\boldsymbol{\theta}}$, where $\theta_1 \ldots \theta_n$ is a triangular representation for σ, are identical.

We obtain a characterization of standard E-unification if we replace the congruence induced by $E \cup E_\sigma$ by the congruence induced by $\cup_\sigma E\sigma \ \cup \ E_\sigma$ in the above theorem, and a characterization of syntactic unifiers if $E = \emptyset$.

Orderings on Substitutions

Unification procedures are designed to find *most general* unifiers of given terms. A substitution σ is said to be *more general* than another substitution θ with respect to a set of variables V, denoted by $\sigma \preceq^V \theta$, if there exists a substitution σ' such that $x\sigma\sigma' = x\theta$ for all $x \in V$.

A substitution σ is called *more general modulo E^g on V* than θ, denoted by $\sigma \preceq^V_{E^g} \theta$, if there exists a substitution σ' such that $x\sigma\sigma' \leftrightarrow^*_{(E\theta)^g} x\theta$ for all $x \in V$. We also define an auxiliary relation $\sqsubseteq$ between substitutions by $\sigma \sqsubseteq^V_{E^g} \theta$ if $x\sigma \leftrightarrow^*_{(E\theta)^g} x\theta$ for all $x \in V$. Two substitutions σ and θ are said to be *equivalent modulo E^g on V* if $\sigma \preceq^V_{E^g} \theta$ and $\theta \preceq^V_{E^g} \sigma$.

If σ is a rigid E-unifier of s and t, then there exists an idempotent rigid E-unifier of s and t that is more general modulo E^g than σ. Hence, in this paper, we will be concerned only with idempotent unifiers. Comparisons between *idempotent* substitutions can be characterized via congruences.

Theorem 3. *Let σ and θ be idempotent substitutions and V the set of variables in the domain or range of σ. Then, $\sigma \preceq^V_{E^g} \theta$ if and only if $E_\sigma \subseteq \leftrightarrow^*_{E^g \cup E^g_\theta}$.*

Proof. If σ is idempotent then we can prove that $\sigma \preceq^V_{E^g} \theta$ if and only if $\sigma\theta \sqsubseteq^V_{E^g} \theta$. Now assuming $\sigma \preceq^V_{E^g} \theta$, we have $x\sigma\theta \leftrightarrow^*_{(E\theta)^g} x\theta$ for all $x \in V$. But $x\theta \leftrightarrow_{(E_\theta)^g} x$ and $E\theta \subseteq \leftrightarrow^*_{E^g \cup E^g_\theta}$ by Proposition 1. Therefore, it follows that $E_{\sigma\theta} \subseteq \leftrightarrow^*_{E^g \cup E^g_\theta}$. But again by Proposition 1, $x\sigma\theta \leftrightarrow^*_{E^g_\theta} x\sigma$ and therefore, $E_\sigma \subseteq \leftrightarrow^*_{E^g \cup E^g_\theta}$.

Conversely, if $E_\sigma \subseteq \leftrightarrow^*_{E^g \cup E^g_\theta}$ then, $E_\sigma\theta \subseteq \leftrightarrow^*_{(E\theta)^g \cup (E_\theta\theta)^g}$, and since the equations in $E_\theta\theta$ are all trivial equations of the form $u \approx u$, it follows that $E_\sigma\theta \subseteq \leftrightarrow^*_{(E\theta)^g}$, which implies $\sigma\theta \sqsubseteq^V_{E^g} \theta$ and hence $\sigma \prec^V_{E^g} \theta$.

3 Rigid *E*-unification

We next present a set of abstract transformation (or transition) rules that can be used to describe a variety of rigid E-unification procedures. By Theorem 2, the problem of finding a rigid E-unifier of two terms s and t amounts to finding a substitution-feasible set S such that $s \leftrightarrow^*_{(E\cup S)^g} t$, and involves (1) constructing a substitution-feasible set S, and (2) verifying that s and t are congruent modulo $E \cup S$. Part (1), as we shall see, can be achieved by using syntactic unification, narrowing and superposition. Efficient techniques for congruence testing via abstract congruence closure can be applied to part (2).

Abstract Congruence Closure

A term t is in *normal form* with respect to a rewrite system R if there is no term t' such that $t \rightarrow_R t'$. A rewrite system R is said to be (ground) *confluent* if for all (ground) terms t, u and v with $u \leftarrow^*_R t \rightarrow^*_R v$ there exists a term w such that $u \rightarrow^*_R w \leftarrow^*_R v$. It is *terminating* if there exists no infinite reduction sequence $t_0 \rightarrow_R t_1 \rightarrow_R t_2 \cdots$ of terms. Rewrite systems that are (ground) confluent and terminating are called (ground) *convergent*.

Let Γ be a set of function symbols and variables and K be a disjoint set of constants. An *(abstract) congruence closure* (with respect to Γ and K) is a ground convergent rewrite system R over the signature $\Gamma \cup K$[4] such that (i) each

[4] We treat variables as constants and in this sense speak of a *ground* convergent system R.

rule in R is either a *D-rule* of the form $f(c_1, \ldots, c_k) \approx c_0$ where f is a k-ary symbol in Γ and $c_0, c_1, \ldots, c_k$ are constants in K, or a *C-rule* of the form $c_0 \approx c_1$ with $c_0, c_1 \in K$, and (ii) for each constant $c \in K$ that is in normal form with respect to R, there exists a term $t \in \mathcal{T}(\Gamma)$ such that $t \rightarrow^*_{R^g} c$. Furthermore, if E is a set of equations (over $\Gamma \cup K$) and R is such that (iii) for all terms s and t in $\mathcal{T}(\Gamma)$, $s \leftrightarrow^*_{E^g} t$ if, and only if, $s \rightarrow^*_{R^g} \circ \leftarrow^*_{R^g} t$, then R is called an (abstract) congruence closure *for* E.

For example, let $E_0 = \{gfx \approx z, fgy \approx z\}$ and $\Gamma = \{g, f, x, y, z\}$. The set E_1 consisting of the rules $x \approx c_1$, $y \approx c_2$, $z \approx c_3$, $fc_1 \approx c_4$, $gc_4 \approx c_3$, $gc_2 \approx c_5$, $fc_5 \approx c_3$ is an abstract congruence closure (with respect to Γ and $\{c_1, \ldots, c_5\}$) for E_0.

The key idea underlying (abstract) congruence closure is that the constants in K serve as *names* for congruence classes, and equations $f(c_1, \ldots, c_k) \approx c_0$ define relations between congruence classes: a term $f(t_1, \ldots, t_k)$ is in the congruence class c_0 if each t_i is in the congruence class c_i.

The construction of a congruence closure will be an integral part of our rigid E-unification method. We will not list specific transformation rules, but refer the reader to the description in [2] which can be easily adapted to the presentation in the current paper.

For our purposes, transition rules are defined on quintuples $(K, E; V, E?; S)$, where Σ is a given fixed signature, V is a set of variables, K is a set of constants disjoint from $\Sigma \cup V$, and $E, E?andS$ are sets of equations. The first two components of the quintuple represent a partially constructed congruence closure, whereas the third and fourth components are needed to formalize syntactic unification, narrowing and superposition. The substitution-feasible set in the fifth component stores an answer substitution in the form of a set of equations. For a given state $(K, E; V, E?; S)$, we try to find a substitution σ with $\mathcal{D}om(\sigma) \subseteq V$, that is a rigid E-unifier of each equation in the set $E?$. By an *initial* state we mean a tuple $(\emptyset, E_0; V_0, \{s \approx t\}; \emptyset)$ where V_0 is the set of all variables that occur in E_0, s or t. Transition rules specify ways in which one quintuple state can be transformed into another such state. The goal is to successively transform a given initial state to a state in which the fourth component is empty.

C-Closure: $$\frac{(K, E; V, E?; S)}{(K', E'; V, E?; S)}$$

if $K \subset K'$ and E' is an abstract congruence closure (with respect to $\Sigma \cup V$ and K') for E.

Note that we need not choose any term ordering, which is one of the main differences of our approach with most other rigid unification methods.

Syntactic Unification

C-closure can potentially extend the signature by a set of constants. Thus we obtain substitutions (or substitution-feasible sets) and terms over an extended signature that need to be *translated* back to substitutions and terms in the original signature, essentially by replacing these constants by terms from the original signature. For example, consider the abstract congruence closure E_1 for $E_0 = \{gfx \approx z, fgy \approx z\}$ described above, and the substitution-feasible set $S = \{c_3 \approx x, x \approx y\}$. This set can be transformed by replacing the constant c_3 by z to give a substitution-feasible set $\{z \approx x, x \approx y\}$. Unfortunately, this may not be possible always. For example, in the substitution-feasible set $S = \{c_1 \approx x\}$, we can't eliminate the constant c_1 since x is the only term congruent to c_1 modulo E_1, but, the resulting set $\{x \mapsto x\}$ is not substitution-feasible.

We say that a (substitution-feasible) set $S = \{t_i \approx x_i : t_i \in \mathcal{T}(\Sigma \cup K, V), x_i \in V, 1 \leq i \leq n\}$ of rules is *E-feasible on V* if, there exist a terms $s_i \in \mathcal{T}(\Sigma \cup V)$ with $s_i \leftrightarrow^*_{E^g} t_i$, such that the set $S{\uparrow_E} = \{s_i \approx x_i : 1 \leq i \leq n\}$ is substitution-feasible.

Recall that if σ is a rigid E-unifier of s and t, then there exists a proof $s \leftrightarrow^*_{E^g \cup E^g_\sigma} t$. The transition rules are obtained by analyzing the above hypothetical proof. The rules attempt to deduce equations in E_σ by simplifying the above proof. We first consider the special case when $s \leftrightarrow^*_{E^g_\sigma} t$, and hence s and t are syntactically unifiable. Trivial proofs can be deleted.

Deletion: $$\frac{(K, E; V, E? \cup \{t \approx t\}; S)}{(K, E; V, E?; S)}$$

If E_σ is a substitution-feasible set and the top function symbols in s and t are identical, then all replacement steps in the proof $s \leftrightarrow^*_{E^g_\sigma} t$ occur inside a non-trivial context, and hence this proof can be broken up into simpler proofs.

Decomposition: $$\frac{(K, E; V, E? \cup \{f(t_1, \ldots, t_n) \approx f(s_1, \ldots, s_n)\}; S)}{(K, E; V, E? \cup \{t_1 \approx s_1, \ldots, t_n \approx s_n\}; S)}$$

if $f \in \Sigma$ is a function symbol of arity n.

Finally, if the proof $s \leftrightarrow^*_{E^g_\sigma} t$ is a single replacement step (at the root position, and within no contexts), we can eliminate it.

Elimination: $$\frac{(K, E; V, E? \cup \{x \approx t\}; S)}{(K \cup \{x\}, E \cup E_\theta; V - \{x\}, E?; S \cup E_\theta)}$$

if (i) $\theta = \{x \mapsto t\}$, (ii) the set $E_\theta = \{t \approx x\}$ is E-feasible on V, and (iii) $x \in V$.

Deletion and decomposition are identical to the transformation rules for syntactic unification, c.f. [1]. However, elimination (and narrowing and superposition described below), do *not* apply the substitution represented by E_θ (or $E_\theta{\uparrow_E}$) to the sets $E?$ and S as is done in the corresponding standard rules for syntactic unification. Instead we add the equations E_θ to the second component of the state.

Decomposition, deletion and elimination can be replaced by a single rule that performs full syntactic unification in one step. We chose to spell out the rules above as they provide a method to abstractly describe an efficient quadratic-time syntactic unification algorithm by recursive descent, c.f. [1].

Narrowing and Superposition

The following rule reflects attempts to identify and eliminate steps in a proof $s \leftrightarrow^*_{E^g \cup E^g_\sigma} t$ that use equations in E^g.

$$\textbf{Narrowing:} \qquad \frac{(K, E; V, E? \cup \{s[l'] \approx t\}; S)}{(K \cup V', E \cup E_\theta; V - V', E? \cup \{s[c] \approx t\}; S \cup E_\theta)}$$

where (i) $l \approx c \in E$, (ii) θ is the most general unifier of l' and l, (iii) the set E_θ is E-feasible on V, (iv) $V' = \mathcal{D}om(\theta) \subset V$, (v) E is an abstract congruence closure with respect to Σ and $K \cup V$, and (vi) either $l' \notin V$ or $l \in V$.

We may also eliminate certain "proof patterns" involving rules in E^g (and E^g_σ) from the proof $s \leftrightarrow^*_{E^g \cup E^g_\sigma} t$ via superposition of rules in E.

$$\textbf{Superposition:} \qquad \frac{(K, E = E' \cup \{t \approx c, C[t'] \approx d\}; V, E?; S)}{(K \cup V', E' \cup \{t \approx c\} \cup T; V - V', E?; S \cup E_\theta)}$$

if (i) θ is the most general unifier of t and t', (ii) E_θ is E-feasible on V, (iii) $T = E_\theta \cup \{C[c] \approx d\}$, (iv) $V' = \mathcal{D}om(\theta) \subset V$, (v) E is an abstract congruence closure with respect to Σ and $K \cup V$, and (vi) either $t' \notin V$ or $t \in V$.

Narrowing, elimination and superposition add new equations to the second component of the state, which are subsequently processed by C-closure.

We illustrate the transition process by considering the problem of rigidly unifying the two terms fx and gy modulo the set $E_0 = \{gfx \approx z, fgy \approx z\}$. Let E_1 denote an abstract congruence closure $\{x \approx c_1, y \approx c_2, z \approx c_3, fc_1 \approx c_4, gc_4 \approx c_3, gc_2 \approx c_5, fc_5 \approx c_3\}$ for E_0 and K_1 be the set $\{c_1, \ldots, c_5\}$ of constants.

i	K_i	E_i	V_i	$E?_i$	S_i	Rule
0	$\emptyset$	E_0	$\{x, y, z\}$	$\{fx \approx gy\}$	$\emptyset$	C-Closure
1	K_1	E_1	$\{x, y, z\}$	$\{fx \approx gy\}$	$\emptyset$	Narrow
2	$K_1 \cup \{x\}$	$E_1 \cup \{x \approx c_5\}$	$\{y, z\}$	$\{c_3 \approx gy\}$	$\{c_5 \approx x\}$	C-Closure
3	K_2	E_3	$\{y, z\}$	$\{c_3 \approx gy\}$	$\{c_5 \approx x\}$	Narrow
4	$K_2 \cup \{y\}$	$E_3 \cup \{y \approx c_3\}$	$\{z\}$	$\{c_3 \approx c_3\}$	$S_3 \cup \{c_3 \approx y\}$	Delete
5	K_4	E_4	$\{z\}$	$\emptyset$	S_4	

where $E_3 = \{x \approx c_1, y \approx c_2, z \approx c_3, fc_1 \approx c_3, gc_3 \approx c_3, gc_2 \approx c_1, c_5 \approx c_1, c_4 \approx c_3\}$ is an abstract congruence closure for E_2. Since the set $E?_5$ is empty, the rigid unification process is completed. Any set $S_4{\uparrow_{E_4}}$ is a rigid unifier of fx and gy. For instance, we may choose gy for the constant c_5 and z for c_3 to get the set $S_4{\uparrow_{E_4}} = \{gy \approx x, z \approx y\}$ and the corresponding unifier $[x \mapsto gy; y \mapsto z]$.

Optimizations

A cautious reader might note that the transition rules contain a high degree of non-determinism in the present form. In particular after an initial congruence closure step, every variable x in the third component of a state occurs as a left-hand side of some rule $x \approx c$ in the second component. Consequently, this rule can be used for superposition or narrowing with any rule in the second or fourth component. A partial solution to this problem is to replace all occurrences of c by x in the second component and then delete the rule $x \approx c$. This is correct under certain conditions.

$$\textbf{Compression1:} \quad \frac{(K \cup \{c\}, E \cup \{x \approx c\}; V \cup \{x\}, E?; S)}{(K, E\theta; V \cup \{x\}, E?\theta; S')}$$

if (i) θ is the substitution $\{c \mapsto x\}$, (ii) $E \cup \{x \approx c\}$ is a fully-reduced abstract congruence closure (with respect to Σ and $K \cup V$), (iii) c does not occur on the right-hand side of any rule in E, and (iv) S' is obtained from S by applying substitution θ *only* to the left-hand sides of equations in S.

A related optimization rule is:

$$\textbf{Compression2:} \quad \frac{(K \cup \{c, d\}, E \cup \{c \approx d\}; V, E?; S)}{(K \cup \{d\}, E; V, E?\theta; S')}$$

if (i) θ is the substitution $\{c \mapsto d\}$, (ii) $E \cup \{c \approx d\}$ is a fully-reduced abstract congruence closure (with respect to Σ and $K \cup V$), and (iii) S' is obtained from S by applying substitution θ *only* to the left-hand sides of equations in S.

These two optimization rules can be integrated into the congruence closure phase. More specifically, we assume that application of C-closure rule is always followed by an exhaustive application of the above compression. We refer to this combination as an "Opt-Closure" rule.

To illustrate these new rules, now note that both $x \approx c_1$ and $y \approx c_2$ can be eliminated from the abstract congruence closure E_1 for E_0. We obtain an optimized congruence closure $\{z \approx c_3, fx \approx c_4, gc_4 \approx c_3, gy \approx c_5, fc_5 \approx c_3\}$. Note that we cannot remove $z \approx c_3$ from the above set.

4 Correctness

Let U be an infinite set of constants from which new constants are chosen in opt-closure. If a state $\xi_i = (K_i, E_i; V_i, E?_i; S_i)$ is transformed to a state $\xi_j = (K_j, E_j; V_j, E?_j; S_j)$ by opt-closure, then (i) E_j is an abstract congruence closure (with respect to Σ and $K \cup V$) for E_i and (ii) E_j is contained in a well-founded simplification ordering[5].

We use the symbol $\vdash_{REU}$ to denote the one-step transformation relation induced by opt-closure, deletion, decomposition, elimination, narrowing and superposition. A *derivation* is a sequence of states $\xi_0 \vdash_{REU} \xi_1 \vdash_{REU} \cdots$ with no two consecutive applications of opt-closure.

[5] For instance, a simple lexicographic path ordering $\succ$ based on a *partial* precedence on symbols in $\Sigma \cup U \cup V$ for which $f \succ c$, whenever $f \in \Sigma$ and $c \in U \cup V$, and $x \succ c$, whenever $x \in V$ and $c \in U - V$, will suffice.

Theorem 4. *(Termination) All derivations starting from an initial state* $(\emptyset, E_0; V_0, \{s \approx t\}; \emptyset)$ *are finite.*

Proof. Define a measure associated with a state $(K, E; V, E?; S)$ to be the pair $(|V|, m_{E?})$, where $|V|$ denotes the cardinality of the set V and $m_{E?} = \{\{s, t\} : s \approx t \in E?\}$. These pairs are compared lexicographically using the greater-than relation on the integers in the first component and the two-fold multiset extension of the ordering $\succ$ in the second component. This induces a well-founded ordering on states with respect to which each transition rule is reducing.

Lemma 1. *Let* $(K_n, E_n; V_n, E?_n; S_n)$ *be the final state of a derivation from* $(\emptyset, E_0; V_0, E?_0; \emptyset)$*, where* $E_0 \cup E?_0$ *are equations over* $\mathcal{T}(\Sigma, V_0)$*. Then*
(a) the set S_n *is* E_n*-feasible on* V_0 *and*
(b) if $E?_n \subseteq \leftrightarrow^*_{(E_n \cup S_n \Uparrow E_n)^g}$*, then* $E?_0 \subseteq \leftrightarrow^*_{(E_0 \cup S_n \Uparrow E_n)^g}$*.*

Theorem 5 (Soundness). *If* $(K_n, E_n; V_n, E?_n; S_n)$ *is the final state of a derivation from* $(\emptyset, E_0; V_0, E?_0; \emptyset)$*, then the set* S_n *is* E_n*-feasible and the substitution corresponding to (any) set* $S_n \uparrow_{E_n}$ *is a rigid* E_0*-unifier of* s *and* t*.*

Proof. The E_n-feasibility of S_n on V_0 follows from Lemma 1. Since $E?_n = \emptyset$, the antecedent of the implication in part (b) of Lemma 1 is vacuously satisfied and hence $E?_0 \subseteq \leftrightarrow^*_{(E_0 \cup S_n \Uparrow E_n)^g}$.

Note that by Theorem 1, the ground congruence induced by $S_n \uparrow_{E_n}$ is identical to the congruence induced by E_σ, where σ is the idempotent substitution corresponding to the set $S_n \uparrow_{E_n}$. Hence, $s \leftrightarrow^*_{E_0^g \cup E_\sigma^g} t$. Using Theorem 2, we establish that σ is a rigid E_0-unifier of s and t.

Theorem 6 (Completeness). *Let* θ *be an idempotent rigid* E_0*-unifier of* s *and* t *and* V_0 *the set of variables in* E_0*,* s *and* t*. Then, there exists a (finite) derivation with initial state* $(\emptyset, E_0; V_0, \{s \approx t\}; \emptyset)$ *and final state* $(K_n, E_n; V_n, \emptyset; S_n)$ *where* $E_{S_n \Uparrow E_n} \subseteq \leftrightarrow^*_{E_0^g \cup E_\theta^g}$*.*

Proof. (Sketch) Let $\xi_i = (K_i, E_i; V_i, E?_i; S_i)$ be a state. We say a substitution-feasible set S^i is a *solution* for state ξ_i if S^i is a E_i-feasible on V_i and

$$E?_i \subseteq \leftrightarrow^*_{(E \cup S_i \cup S^i)^g}.$$

Now, given a state ξ_i and a solution S^i for ξ_i, we show how to obtain a new state ξ_j and a solution S^j for ξ_j such that the pair $\langle \xi_j, S^j \rangle$ is *smaller* in a certain well-founded ordering than the pair $\langle \xi_i, S^i \rangle$ and the congruences induced by $E_j \cup S_j \cup S^j$ and $E_i \cup S_i \cup S^i$ are identical. The well-founded ordering will be a lexicographic combination of the ordering on states ξ_i's used in the proof of Theorem 4 and a well-founded ordering on substitution-feasible sets S_i's. If a pair (ξ_i, S^i) can not be reduced then we show that $E?_i = \emptyset$. This yields the desired conclusion.

The above reduction of a pair (ξ_i, S^i) can be achieved in two ways: (i) by an REU transformation on ξ_i, suitably guided by the given solution S^i, or, (ii) by some simple transformation of the set S^i. The latter transformation rules are defined in the context of the state ξ_i. The initial state is $(S^i, \emptyset)$.

$$\mathbf{R1}: \frac{(D' \cup \{c \approx x\}, C')}{(D', C' \cup \{c \approx x\})} \qquad \text{if } c \in K_i \cup V_i,\ x \not\to^*_{E_i^g \setminus C'^g} c^6.$$

$$\mathbf{R2}: \frac{(D' \cup \{c \approx x\}, C')}{(D', C')} \qquad \text{if } c \in K_i \cup V_i,\ x \to^*_{E_i^g \setminus C'^g} c$$

$$\mathbf{R3}: \frac{(D' \cup \{t[l'] \approx x\}, C')}{(D' \cup \{t[c] \approx x\}, C')} \qquad \text{if } l \approx c \in E_i,\ l \leftrightarrow^*_{C'^g} l'$$

$$\mathbf{R4}: \frac{(D' \cup \{t[l'] \approx x\}, C')}{(D' \cup \{t[y] \approx x\}, C')} \qquad \text{if } l \approx y \in D',\ l \leftrightarrow^*_{C'^g} l',\ l \notin K_i \cup V_i$$

These rule construct a generalized congruence closure for the initial set $D' \cup C'$ (modulo the congruence induced by E_i). If (D', C') can be obtained from $(S^i, \emptyset)$ by repeated application of these rules, then the set $D' \cup C'$ is (i) substitution-feasible, (ii) E_i-feasible with respect to V_i, and (iii) equivalent modulo E_i^g on V_i to S^i. The set of rules $R1 - -R4$ is terminating.

5 Specialization to syntactic unification

Since rigid unification reduces to syntactic unification when the set E_0 is empty, one pertinent question is what procedure the REU transformation rules yield in this special case? Note that elimination does *not* apply a substitution to the fourth and fifth components of the state, but does perform an *occur check* in condition (ii). This is in the spirit of syntactic unification by recursive descent algorithm which works on term directed acyclic graphs and is a quadratic time-complexity algorithm.

In fact, in the case of syntactic unification, every equation in the second component is of a special form where one side is always a variable. Hence, we can argue that for each $c \in K$, there is *at most* one rule in E of the form $f(\ldots) \to c$ where $f \in \Sigma$. We may therefore replace superposition by the following rule:

$$\textbf{Det-Decompose:} \quad \frac{(K, E; V, E? \cup \{c \approx t\}; S)}{(K, E; V, E? \cup \{f(c_1, \ldots, c_k) \approx t\}; S)}$$

if there exist exactly one rule $f(c_1, \ldots, c_k) \approx c$ with right-hand side c in E.

In addition, we may restrict narrowing so that the unifier θ is always the identity substitution, that is, narrowing is used to only for simplification of terms in the fourth component $E?$ by equations in the second component E.

We can get various efficient syntactic unification algorithms by using specific strategies over our abstract description. Other descriptions of the quadratic time syntactic unification algorithms are usually based on descriptions of dags and abstract rules that manipulate the dags directly. However, since we can abstractly capture the notion of sharing, we obtain rules for this class of efficient algorithms that work on terms and are very similar to the rules for describing the naive syntactic unification procedures (with a worst case exponential behavior).

6 Summary

We have presented a formulation of rigid E-unification in terms of fairly abstract transformation rules. The main feature is the integration of (abstract) congruence closure with transformation rules for syntactic unification, paramodulation and superposition. The use of an extended signature (inherent in abstract congruence closure) helps to dispense with term orderings over the original signature. An abstract rule-based description facilitates various optimizations. The specialization of the transformation rules to syntactic unification yields a set of abstract transition rules that describe a class of efficient syntactic unification algorithms. Our transformation rules can be derived from proof simplification arguments.

In [10], a congruence closure algorithm is used in a rigid E-unification procedure, but not as a submodule. Congruence closure is used "indirectly" to do ground completion. The work on abstract congruence closure shows that congruence closure actually is ground completion with extension. But for the purpose of rigid E-unification, we don't need to translate the abstract closure to a ground system over the original signature, though we do need to translate the substitutions back to the original signature. Extended signatures also help as we do not need to guess an ordering to orient equations such as $x \approx fa$ when the substitution for x is not yet known. This is a major concern in [10] where the dependence on orderings complicates the unification process.

In [5], the problem of correctly orienting equations is solved by delaying the choice of orientation and maintaining constraints. Constraint satisfiability is required to ensure that orientations are chosen in a consistent manner, and to guarantee the termination of such a procedure.

We would like to point out that the transformation process involves "don't-care" non-determinism (where it does not matter which rule one applies) and "don't-know" non-determinism (where an application of a wrong rule may lead to a failure even if a unifier exists). Whereas opt-closure, deletion and narrowing with identity substitution can be applied "don't-care" non-deterministically, the other rules have to be applied in a "don't-know" non-deterministic manner. The rules for syntactic unification described in Section 5 are "don't-care" non-deterministic.

All algorithms for computing the set of rigid unifiers for a pair of terms can be seen as a combination of top-down and bottom-up method. In a pure bottom-up approach a substitution is guessed non-deterministically: for every variable one tries every subterm that occurs in the given unification problem, see [13] for details. Superposition and narrowing using a rule that contains a variable as its left-hand side captures the bottom-up aspect in our description. A top-down approach is characterized by the use of narrowing to simplify the terms in the goal equations $E?$.

We note that for variables that cannot be eliminated from the left-hand sides of rules using compression1, we need to try a lot of possible substitutions because they can unify with almost all subterms in the second and fourth components. This is the cause of a bottom-up computation for these variables. For other variables, however, we need to try only those substitutions that are produced by some unifier during an application of narrowing or superposition, and hence a top-down approach works for these variables.

We illustrate some of the above observations via an example. Let $E_0 = \{gx \approx x,\ x \approx a\}$, and suppose we wish to find a rigid E_0-unifier of the terms $gfffgffx$ and $fffx$. The substitution $\{x \mapsto fa\}$ is a rigid E-unifier, but it cannot be obtained unless one unifies the variable x with an appropriate subterm.

We believe that our approach of describing rigid E-unification can be used to obtain an easier proof of the fact that this problem is in NP. We need to show that (i) the length of a maximal derivation from any initial state is bounded by some polynomial in the input size, (ii) each rule can be efficiently applied, and (iii) there are not too many choices between the rules to get the next step in a derivation. It is easy to see that (i) holds. For the second part, a crucial argument involves showing that the test for E-feasibility can be efficiently done. This is indeed the case, but due to space limitations, we don't give a way to do this here.

The notion of an abstract congruence closure is easily extended to handle associative and commutative functions [3]. The use of extended signatures is particularly useful when one incorporates such theories. This leads us to believe that our proposed description of rigid E-unification can be suitably generalized to such applications.

Acknowledgements. We would like to thank David Cyrluk for initial discussions on the problem and the anonymous reviewers for their helpful comments.

References

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

[2] L. Bachmair, C. Ramakrishnan, I.V. Ramakrishnan, and A. Tiwari. Normalization via rewrite closures. In P. Narendran and M. Rusinowitch, editors, *10th Int. Conf. on Rewriting Techniques and Applications*, pages 190–204, 1999. LNCS 1631.

[3] L. Bachmair, I.V. Ramakrishnan, A. Tiwari, and L. Vigneron. Congruence closure modulo associativity and commutativity. In H. Kirchner and C. Ringeissen, editors, *Frontiers of Combining Systems, 3rd Intl Workshop FroCoS 2000*, pages 245–259, 2000. LNAI 1794.

[4] L. Bachmair and A. Tiwari. Abstract congruence closure and specializations. In D. McAllester, editor, *17th Intl Conf on Automated Deduction*, 2000.

[5] G. Becher and U. Petermann. Rigid unification by completion and rigid paramodulation. In B. Nebel and L.D. Fischer, editors, *KI-94: Advances in Artificial Intelligence, 18th German Annual Conf on AI*, pages 319–330, 1994. LNAI 861.

[6] B. Beckert. A completion-based method for mixed universal and rigid E-unification. In A. Bundy, editor, *12th Intl Conf on Automated Deduction, CADE-12*, pages 678–692, 1994. LNAI 814.

[7] A. Degtyarev and A. Voronkov. The undecidability of simultaneous rigid E-unification. *Theoretical Computer Science*, 166(1–2):291–300, 1996.

[8] A. Degtyarev and A. Voronkov. What you always wanted to know about rigid E-unification. *Journal of Automated Reasoning*, 20(1):47–80, 1998.

[9] J. Gallier, P. Narendran, D. Plaisted, and W. Snyder. Rigid E-unification: Np-completeness and applications to equational matings. *Information and Computation*, 87:129–195, 1990.

[10] J. Gallier, P. Narendran, S. Raatz, and W. Snyder. Theorem proving using equational matings and rigid E-unification. *Journal of the Association for Computing Machinery*, 39(2):377–429, April 1992.

[11] J. Goubault. A rule-based algorithm for rigid E-unification. In G. Gottlob, A. Leitsch, and D. Mundici, editors, *Computational logic and proof theory. Proc. of the third Kurt Godel Colloquium, KGC 93*, pages 202–210, 1993. LNCS 713.

[12] D. Kapur. Shostak's congruence closure as completion. In H. Comon, editor, *8th Intl Conf on Rewriting Techniques and Applications*, pages 23–37, 1997. LNCS 1232.

[13] Eric de Kogel. Rigid E-unification simplified. In P. Baumgartner, R. Hahnle, and J. Posegga, editors, *Theorem Proving with Analytic Tableaux and Related Methods, 4th International Workshop, TABLEAUX '95*, pages 17–30, 1995. LNAI 918.