Decision Procedures in Automated Deduction

a dissertation presented

by

Ashish Tiwari

to

The Graduate School

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

in

computer science

State University of New York

at Stony Brook

August 2000

Abstract of the Dissertation

Decision Procedures in Automated Deduction

by
Ashish Tiwari

Doctor of Philosphy
in
Computer Science
State University of New York at Stony Brook
2008

We present a practical variant of the Nelson-Oppen abstract combination result which can be used for combining decision procedures based on completion. We suitably instantiate this general theorem to obtain combinations of a class of congruence closure algorithms, procedures for deciding the word problem in ground AC-theories (generalization of the word problem for commutative semigroups), and algorithms for polynomial ideals over general rings. The combination result can also be used to integrate several other theories.

Our description of abstract congruence closure suitably captures the logical essence of most of the conventional algorithms for congruence closure. Additionally, it can be used to obtain new efficient implementations. Experimental results are presented to illustrate the relative efficiency and explain differences in performance of these various algorithms. The transition rules for computation of abstract congruence closure are obtained from rules for *standard completion* enhanced with an *extension* rule that enlarges a given signature by new constants.

We use our general combination result to define the notion of an *associative-commutative congruence closure* and give a complete set of transition rules for construction of such closures. This solves the word problem for ground AC-theories without the need for AC-simplification orderings total on ground terms. Associative-commutative congruence closure provides a novel way to construct a convergent rewrite system for a ground AC-theory. The concept of an abstract congruence closure also helps to clarify and generalize procedures that are based on congruence closure, for example construction of convergent rewrite systems, non-oblivious normalization, and the problem of rigid E-unification.

Finally, we describe the Gröbner bases based decision procedure for polynomial ideals using completion-like transition rules, which is a generalization of the theory of Gröbner bases for polynomial ideals over fields to polynomials over commutative Noetherian rings. In the same spirit as what Gröbner bases do for polynomials over fields, we can use the new generalization to solve several problems in the theory of multivariate polynomials over rings.

Dedicated to the cause of all complexity in this world and man's quest for simplicity underlying it all.

Contents

List of Tables

List of Figures

Acknowledgements

I am grateful to my advisor Prof. Leo Bachmair for providing the necessary support and technical inputs that made this thesis possible. The work presented here is highly influenced by some of his research. I am also highly indebted to Prof. I. V. Ramakrishnan for suitably guiding me through my doctoral program at Stony Brook. Significant contributions from my other co-authors—Prof. C. R. Ramakrishnan, Prof. Laurent Vigneron and Dr. Harald Ruess—is also sincerely acknowledged.

I have also been lucky to have had the oppurtunity to discuss some of my research work with various researchers whose inputs have helped a lot. I thank Dr. D. Cyrluk (SRI, Menlo Park), Prof. N. Dershowitz (Tel Aviv University, Tel Aviv), Prof. H. Ganzinger (MPI, Saarbrücken), Prof. D. Kapur (Univ. of New Mexico, Albuquerque), Dr. N. Kumar (SPIC Mathematical Institute, Chennai), Dr. P. Lincoln (SRI, Menlo Park), Prof. C. Lynch (Clarkson University, Potsdam), Prof. K. Madlener (Universitat Kaiserslautern, Kaiserslautern), Dr. J. Rushby (SRI, Menlo Park), Dr. N. Shankar (SRI, Menlo Park), Dr. J. Sifakis (Verimag, Gieres), and Prof. D. Warren (SUNY, Stony Brook) for the same. Dr. Ta Chen and Dr. Prasad Rao provided me an oppurtunity to spend a summer at Bellcore (Morristown), and Dr. N. Shankar and SRI gave me a chance to work with them at Menlo Park during another summer.

Finally, several people have inspired and motivated me through the years, most notable amongst them being Prof. G.K. Dubey, Dr. Gaurav Khanna, my grandfather Sh. M.P. Tiwari, my parents, and the rest of my family. Diksha, Sarah, and music were a source of renewed vitality. Many other friends from IIT-Kanpur and Stony Brook have helped in variously different ways.

The research described in this thesis was supported in part by the National Science Foundation under grants CCR-9510072, CCR-9711386, CCR-9712383, EIA-9705998, CCR-9902031, and by DARPA AO D855 under US Air Force Rome Laboratory contract F30602-96-C-0204.

Chapter 1

Introduction

Formal logic is a means of expressing declarative knowledge. Deduction is the mechanism used to infer more facts from such knowledge. Theorem provers are systems that automate this process to various degrees. Such systems have been used—both independently and under human assistance—to verify complex programs and system designs, derive or prove many mathematical theorems, deduce properties of computational systems, synthesize provably correct programs, etc. Deduction is also part of several AI systems and database systems, amongst others.

Decision procedures, on the other hand, are specialized theorem proving methods designed for subclass of formulas from a particular domain. These are faster, predictable and more efficient than general purpose deduction systems. Examples include model checking algorithms, the large class of mathematical procedures that are usually part of a computer algebra system, etc.

Integration of specialized decision procedures into larger general purpose deduction systems is crucial for various applications. There are several distinct lines of investigation in this broad area.

One area of research focus has been the combination of computer algebra with theorem proving. A number of different architectures have been proposed in the literature for achieving this combination. For instance, there are attempts and proposals to embed a computer algebra system within a theorem prover [31, 32, 27]. In contrast, there are also proposals to embed a reasoning system within a computer algebra system, for example, by using the type system of the computer algebra system to represent a logic [69]. The third kind of architecture for integration involves an independent theorem prover communicating with an independent computer algebra system.

There have also been attempts at integrating certain specific decision procedures—for linear arithmetic, theory of arrays, theory of lists, theory of bit vectors, etc—with general purpose theorem provers. A pioneering work in this direction is that of Boyer and Moore who considered a "black-box" integration for decision procedures in their prover [21, 23]. However, they realized that this open architecture based approach did not work due to several complications involved in defining the interface [22]. Some other attempts have tried for a tighter integration, as in the PVS system [73] and the Stanford Pascal Verifier [63].

Nelson and Oppen integrate decision procedures with the Stanford Pascal Verifier by first combining all the decision procedures into a single decision procedure. This combination has been later studied and abstractly presented [71, 78]. Integration of decision procedures in PVS is based on Shostak's method [75], where any theory with a *canonizer* and a *solver* can be embedded into the prover. The formal semantics and correctness of Shostak's method form a continuing strand of research [35, 51].

1.1 Outline

In this thesis, we present a formal grounding of some of the aspects of integration. Our starting point is the Nelson and Oppen combination result for certain first-order theories. The Nelson and Oppen combination procedure is one of the central results in the theory of combination of decision procedures. We present a suitable "constructive" version of this theorem in the form of an abstract combination result for decision procedures in Chapter 3. A large part of the remaining thesis concerns itself with illustrations of the abstract result via suitable instantiations that give us decision procedures for combinations of several different theories.

Broadly speaking, consider a signature Σ which is a disjoint union of signatures Σ_i's, where certain of these signatures Σ_i's could contain "interpreted" symbols (i.e. we are interested in certain specific theories over that signature). To fix notation, let $\mathcal{E}_i$ denote the set of (first-order) axioms over the signature Σ_i for the theory of interest. Let us assume that we are given algorithms that can construct a convergent system (modulo $\mathcal{E}_i$) for a set E_i of ground equations over Σ_i (allowing for additional constants). Now, if we are given a set of equations E over the signature Σ, we show how (and under what conditions) one can construct a convergent system (modulo $\cup\mathcal{E}_i$) for a conservative extension of the equational theory E.

A powerful technique we use to achieve the above result is that of variable abstraction. Variable abstraction is also at the core of the Nelson and Oppen combination result. It refers to the simple idea of introducing new names to "purify" mixed terms. For example, an equation $f(a) \approx g(a)$ containing three different function symbols a, f and g, can be split into the equations $a \approx c_1, f(c_1) \approx c_2$ and $g(c_1) \approx c_2$ each containing exactly one of the symbols a, f or g. Using extensions of signatures (rather than just the original signature) gives more flexibility (in terms of ordering on terms), and also helps in avoiding repeated work (due to structure sharing).

The simplest application of our abstract combination result is to the ground theory of equality where each set $\mathcal{E}_i$ is empty. In this case, we are led to the concept of an abstract congruence closure. In Chapter 4 we define this concept and study the problem of construction of congruence closure in detail. Intuitively speaking, a finite set of a ground equations over a signature Σ can be thought of as a set of ground equations over a *combined* signature $\Sigma = \cup_i \Sigma_i$ where each Σ_i is singleton. Thus, we introduce new constants (a lá variable abstraction) to "purify" equations in E and obtain a presentation of a new ground equational theory that is a conservative extension of the original theory. Informally, an abstract congruence closure is a convergent rewrite system for this conservative extension.

Construction of an abstract congruence closure can be presented by suitably modifying the abstract transition rules for completion as presented by Bachmair and Dershowitz [7]. The main difference is that we additionally have rules for extending the signature. Extensions allow us to dispense with term orderings in this case. The correctness is fairly simple and is not based on proof transformation techniques.

Semantically, the new constants can be interpreted in a number of different ways. They can be thought of as names of vertices in a term directed acyclic graph. This simple but crucial observation leads us to abstractly capture the notion of "sharing". In the example above, the terms $f(a)$ and $g(a)$ both share the subterm a. By introducing a new name c_1 for a, we now share the constant c_1. (If a is replaced by a really large term, then the new presentation using c_1 will be more compact.) This interpretation allows us to cast the various well-known graph based congruence closure algorithms in the language of an abstract congruence closure. Chapter 5 is devoted to this aspect of abstract congruence closure.

Not only do the transition rules for constructing an abstract congruence closure provide a logical description of the well-known algorithms, they can themselves be

used to obtain fairly efficient implementations—essentially using the fairly well understood notions of indexing and redundancy that have been used to speed up rewrite based engines. Experimental results comparing the performance of the traditional graph-based algorithms with the new completion-based algorithm turned out to be interesting, and demonstrated various shortcomings and inefficiencies of certain algorithms. Details about this comparison can be found in Section 5.5 of Chapter 5.

After a comprehensive treatment of congruence closure, we turn to a second application of the abstract combination result. We again consider ground equational theories over some signature Σ, but now we allow for certain function symbols in Σ to be associative and commutative. Thus, now the individual theories over singleton signatures Σ_i are either (i) the pure theory of equality as before, or (ii) the theory of commutative semigroups, i.e. the set $\mathcal{E}_i$ contains the associativity and commutativity axiom of some binary function symbol. The choice of going into the details of this combination and not many others (like commutative monoids, commutative theories, commutative semigroups with idempotence etc.) is pragmatic. Associativity and commutativity are important and useful properties satisfied by a variety of symbols. Moreover, details for most of the other theories can be worked out in a manner similar to what is done for AC-theories.

The completion procedure obtained for ground AC-theories can be seen as an extension of the one for abstract congruence closure, and hence we call it "congruence closure modulo associativity and commutativity", see Chapter 6. The result allows us to naturally arrive at decidability of ground AC-theories using the decidability result for commutative semigroups. As before, one of the distinct advantages of our approach is that it allows us to dispense with complicated term orderings. A naive completion of a set of ground AC equations requires an AC-compatible simplification ordering total on ground terms. Proving the existence of one such ordering was a challenge [62, 72].

Is it possible to get rid of the constants introduced as a result of variable abstraction after an (AC) congruence closure has been computed? We discuss this interesting question in Chapter 6 and obtain an intriguing answer for the AC case.

Chapter 7 is a slight digress and discusses some applications of abstract congruence closure: efficient normalization, rigid E-unification and Shostak's combination method. Congruence closure is a crucial component in procedures for all of these problems. The problem of non-oblivious normalization is natural: given a set of rewrite rules $\mathcal{E}$, we are interested in finding an $\mathcal{E}$-normal form for a given term t

without repeating any reduction steps. For instance, if a term s can be reduced to t in n steps, then we would like to rewrite $f(s,s)$ to $f(t,t)$ in $n+1$ steps (and not in $2n$ steps). This is achieved by maintaining a *rewrite closure*, a suitable extension of the concept of an abstract congruence closure, of all the rules used in the reduction sequence. Our results generalize previous results on congruence closure-based normalization methods. The description of known methods within our formalism also allows for a better understanding of these procedures.

Rigid E-unification is a restricted version of the problem of unification modulo an equational theory. It arises naturally when tableaux based procedures for first-order theorem proving are extended to handle equality. We present 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. A novel feature in the correctness argument is the use of a characterization of substitutions as congruences, which 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. This again illustrates how extended signatures can be used to abstractly capture the notion of structure sharing.

With a clear understanding of congruence closure and its interaction with syntactic unification, we present in Section 7.3 a detailed presentation of Shostak's procedure for combining congruence closure with E-unification procedures.

The final part of this thesis instantiates the abstract combination result by a non-trivial equational theory, the theory of polynomial ideals over Noetherian rings. The theory of polynomial ideals is clearly a generalization of the theory of commutative semigroups discussed in Chapter 6. In fact, in the case of commutative semigroups, every ground equation in the theory of commutative semigroups is of the form $f(c_{i_1}, c_{i_2}, \ldots, c_{i_k}) \approx f(c_{j_1}, c_{j_2}, \ldots, c_{j_k})$. This equation can be thought of as a polynomial equation $X_{i_1} X_{i_2} \cdots X_{i_k} \approx X_{j_1} X_{j_2} \cdots X_{j_l}$.

The Gröbner basis algorithm provides a decision procedure for the word problem in this equational theory. The theory of polynomials can be presented by set of rewrite rules that are ground convergent modulo AC. Our combination result extends the Gröbner basis algorithm to additionally handle function symbols that are *not* part of the signature of polynomial rings. Since this extension follows in a

straight-forward manner, we focus our attention to the theory of polynomial rings (allowing for arbitrary number of constants in the signature, as is required by our combination result). Chapter 8 defines two distinct kinds of Gröbner bases: *weak* and *strong* Gröbner bases. We first focus on weak bases, and present a completion-like procedure for constructing weak basis for polynomial ideals over commutative Noetherian rings with unit. This is a generalization of all the known algorithms for computing Gröbner bases for polynomial ideals over various different coefficient domains. The coefficient domain is incorporated using constraints. Constraints allow us to describe an optimized procedure for computing Gröbner bases. The optimization restricts the number of superpositions that need to be considered.

Subsequently, we define strong Gröbner bases and show that weak bases can be extended to strong bases under an additional ordering assumption on the coefficient domain. The conditions on the coefficient domain are the weakest possible and are shown to carry over from ring $\mathcal{B}$ to $\mathcal{B}[X]$, thus giving a hierarchic algorithm for construction of Gröbner bases. Correctness of the procedure is established through proof simplification techniques.

Figure 1 gives a dependency between the various chapters. A dashed line indicates that the main results of the following chapter can be proved independently of the results in the former chapter, but the results from the former chapter help to put the latter results in a larger context. We begin by fixing the notation for the rest of this thesis and recall the basic concepts from theorem proving and term rewriting in Chapter 2.

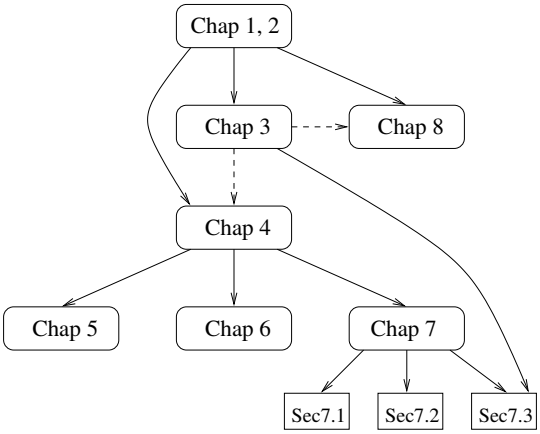


Figure 1: Dependency graph for chapters in this thesis.

Chapter 2

Preliminaries

In this chapter we discuss the basic concepts that will be used in the rest of the thesis. The intent here is not to provide a comprehensive treatment of the concepts, but just to set the notation and terminology for future use. The notation is consistent with that in [38]. For a complete treatment of the theory of term rewriting the reader should consult [38, 54, 2]. The notion of proof orderings and proof transformations appears in [5]. There are several good references on unification, see [4, 49].

2.1 Terms and Substitutions

Given a set $\Sigma = \cup_n \Sigma_n$ and a disjoint (denumerable) set $\mathcal{V}$, we define $\mathcal{T}(\Sigma, \mathcal{V})$ as the smallest set containing $\mathcal{V}$ and such that $f(t_1, \ldots, t_n) \in \mathcal{T}(\Sigma, \mathcal{V})$ whenever $f \in \Sigma_n$ and $t_1, \ldots, t_n \in \mathcal{T}(\Sigma, \mathcal{V})$. The elements of the sets Σ, $\mathcal{V}$ and $\mathcal{T}(\Sigma, \mathcal{V})$ are respectively called *function symbols*, *variables* and (first-order) *terms* (over Σ and $\mathcal{V}$). 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 term $t \in \mathcal{T}(\Sigma, \mathcal{V})$ can be viewed as a finite ordered tree in which nodes are labeled by a symbol in $\Sigma \cup \mathcal{V}$ and have an outdegree equal to the arity of this symbol. Nodes labeled by a variable have outdegree zero. Terms represented by the various nodes in this tree are called *subterms* of t. The *position* of a subterm within a term t may be represented—in Dewey decimal notation—as a sequence of positive integers, describing the path from the root of the tree to the node of the subterm. A subterm

of a term t is called *proper* if it is distinct from t.

We write $t[s]$ to indicate that a term t contains s as a subterm and (ambiguously) denote by $t[u]$ the result of replacing a particular occurrence of s by u.

A *substitution* is a mapping from variables $\mathcal{V}$ to terms $\mathcal{T}(\Sigma, \mathcal{V})$ such that $x\sigma = x$ for all but finitely many variables $x \in \mathcal{V}$. 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 $\mathcal{T}(\Sigma, \mathcal{V})$ 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 $\{x \in \mathcal{V} : x\sigma \neq x\}$; 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 $\sigma\sigma = \sigma$[1].

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 individual substitutions $\sigma_i = \{x_i \mapsto t_i\}$.

2.2 Equations, Rewrite Systems and Theories

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

If $\rightarrow$ is a binary relation, then $\leftarrow$ denotes its inverse, $\leftrightarrow$ its symmetric closure, $\rightarrow^+$ its transitive closure and $\rightarrow^*$ its reflexive-transitive closure. Thus, $\leftrightarrow^*_{\mathcal{E}^g}$ denotes the *congruence relation*[2] induced by $\mathcal{E}$. The *equational theory* of a set $\mathcal{E}$ of equations is defined as the relation $\leftrightarrow^*_{\mathcal{E}}$. Equations are often called *rewrite rules*, and a set $\mathcal{E}$ a *rewrite system*, if one is interested particularly in the rewrite relation $\rightarrow^*_{\mathcal{E}}$ rather than the equational theory $\leftrightarrow^*_{\mathcal{E}}$.

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

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

A *(equational) proof* of $s \approx t$ (in $\mathcal{E}$) is a finite sequence $s = s_0 \leftrightarrow_{\mathcal{E}} s_1,\ s_1 \leftrightarrow_{\mathcal{E}} s_2,\ \cdots, s_{k-1} \leftrightarrow_{\mathcal{E}} s_k = t (k \geq 0)$, which is usually written in abbreviated form as $s = s_0 \leftrightarrow_{\mathcal{E}} s_1 \leftrightarrow_{\mathcal{E}} \cdots \leftrightarrow_{\mathcal{E}} s_k = t (k \geq 0)$.

The pure theory of equality is obtained when the set $\mathcal{E}$ is empty. On the other hand, if $\Sigma_{AC} \subset \Sigma$ is a finite set of binary function symbols, and the set $\mathcal{E}$ contains the identities

$$f(x,y) \approx f(y,x) \qquad f(f(x,y),z) \approx f(x,f(y,z))$$

for each symbol $f \in \Sigma_{AC}$, then, what we obtain is an associative-commutative (AC) theory.

A *multiset* over a set S is a mapping M from S to the natural numbers. Any ordering $\succ$ on a set S can be extended to an ordering $\succ^{mult}$ on multisets over S as follows: $M \succ^{mult} N$ if, and only if, $M \neq N$ and whenever $N(x) > M(x)$ then $M(y) > N(y)$, for some $y \succ x$. The multiset ordering $\succ^{mult}$ (on finite multisets) is well founded if the ordering $\succ$ is well founded [39]. Multiset inclusion is defined as $M \subseteq N$ if $M(x) \leq N(x)$ for all $x \in S$.

2.3 Term Rewriting

Rewriting methods have been successful for equational theorem proving. The basic idea is that of *completion* which utilizes an ordering on terms to perform forward reasoning on the axioms and simplification of equations. In terms of proof-orderings, completion attempts to deduce enough consequences so that an arbitrary proof can be transformed to a normal-form proof. In other words, the intent of completion is to obtain a set of equations which define the same equational theory as before, but with respect to which every term has a unique normal form, as defined next.

A subterm of u at which the equational replacement takes place is called a *redex*. A term t is *irreducible*, or in *normal form*, if it has no redex, i.e. a term t is in *normal form* with respect to a rewrite system $\mathcal{R}$, or in $\mathcal{R}$-*normal form*, if there is no term u, such that $t \rightarrow_{\mathcal{R}} u$. We write $s \rightarrow^{!}_{\mathcal{R}} t$ to indicate that t is a $\mathcal{R}$-normal form of s.

A rewrite system $\mathcal{R}$ is said to be (ground) *confluent* if for any pair s, s' of (ground) terms, if there exists a (ground) term t such that $s \leftarrow^{*}_{\mathcal{R}} t \rightarrow^{*}_{\mathcal{R}} s'$, then, there also exists a (ground) term t' such that $s \rightarrow^{*}_{\mathcal{R}} t' \leftarrow^{*}_{\mathcal{R}} s'$. If a rewrite system $\mathcal{R}$ is (ground) confluent, then every (ground) term t has at most one normal form. A rewrite system $\mathcal{R}$ is *terminating* if there exists no infinite reduction sequence $s_0 \rightarrow_{\mathcal{R}} s_1 \rightarrow_{\mathcal{R}} s_2 \cdots$ of

terms. If $\mathcal{R}$ is terminating then every term has at least one $\mathcal{R}$-normal form. Rewrite systems that are (ground) confluent and terminating are called (ground) *convergent*.

Equational axioms, such as commutativity, that can not be oriented into a rewrite rule without compromising the termination characteristics of the relation are handled using *extensions* of rules and the rewriting relation modulo the equational axioms. For our purposes, it suffices to consider the associativity and commutativity axioms, denoted by AC. By $AC\backslash\mathcal{R}$ we denote the rewrite system consisting of all rules $u \to v$ such that $u \leftrightarrow^*_{AC} u'\sigma$ and $v = v'\sigma$, for some rule $u' \to v'$ in $\mathcal{R}$ and some substitution σ. We say that $AC\backslash\mathcal{R}$ is confluent modulo AC if for all terms s, t such that $s \leftrightarrow^*_{\mathcal{R}\cup AC} t$, there exist terms w and w' such that $s \to^*_{AC\backslash\mathcal{R}} w \leftrightarrow^*_{AC} w' \leftarrow^*_{AC\backslash\mathcal{R}} t$. We speak of *ground confluence* if this condition is true for all ground terms s and t. The other definitions are analogous.

Part of the condition for confluence modulo AC can be satisfied by the inclusion of so-called *extensions* of rules [67]. Given an AC-operator f and a rewrite rule $\rho : f(c_1, c_2) \to c$, we consider its extension $\rho^e : f(f(c_1, c_2), x) \to f(c, x)$. Given a set of rewrite rules $\mathcal{R}$, by $\mathcal{R}^e$ we denote the set $\mathcal{R}$ plus extensions of rules in $\mathcal{R}$. Extensions have to be used for rewriting terms and computing critical pairs when working with AC-symbols. The key property of extended rules is that whenever an expression e is AC-reducible by $\mathcal{R}^e$ and $e \leftrightarrow^*_{AC} e'$, then e' is also AC-reducible by $\mathcal{R}^e$.

The rewrite relation induced by $S\backslash R$ is different from the rewriting relation defined by R on congruence classes induced by S. More specifically, we define $s \to_{R/S} t$ if $s \leftrightarrow^*_S u[l\sigma]$ and $t \leftrightarrow^*_S u[r\sigma]$ for some rule $l \to r$ in R and some substitution σ.

2.4 Algebra and Unification

Let $\Sigma = \cup_n \Sigma_n$ be a signature. An Σ-algebra $\mathbf{A} = (A, \Sigma_{\mathbf{A}})$ consists of a set A and a family of functions $\Sigma_{\mathbf{A}}$ such that for each $f \in \Sigma_n$ there exists a n-ary function $f_{\mathbf{A}} \in \Sigma_{\mathbf{A}}$ that maps A^n to A. An algebra $\mathbf{A}$ is a *model* of a set of equations $\mathcal{E}$ if for every equation $s \approx t \in \mathcal{E}$, and every assignment of values in A to variables in $\mathcal{V}$, the interpretation of s and t are the same in A. An $\mathcal{E}$-algebra is any algebra that is a model of the set $\mathcal{E}$ of equations.

The quotient algebra $\mathcal{T}(\Sigma, \mathcal{V})/\leftrightarrow^*_{\mathcal{E}}$ is the *$\mathcal{E}$-free algebra* with generators $\mathcal{V}$. *$\mathcal{E}$-unification* is the solving of equations in the $\mathcal{E}$-free algebra $\mathcal{T}(\Sigma, \mathcal{V})/\leftrightarrow^*_{\mathcal{E}}$. More precisely, a *unification problem* is a pair of terms s, t, and an *$\mathcal{E}$-unifier* of the problem is a *substitution* σ such that $s\sigma \leftrightarrow^*_{\mathcal{E}} t\sigma$. The set of all $\mathcal{E}$-unifiers of s, t will be denoted

by $U_{\mathcal{E}}(s,t)$. Let $\leq$ be a quasi-order on substitutions (usually the $\mathcal{E}$-instantiation ordering). A *complete set $cU_{\mathcal{E}}(s,t)$ of $\mathcal{E}$-unifiers of s and t with respect to* $\leq$ is characterized by, (i) $cU_{\mathcal{E}}(s,t) \subseteq U_{\mathcal{E}}(s,t)$, and (ii) for all $\sigma \in U_{\mathcal{E}}(s,t)$ there exists $\theta \in cU_{\mathcal{E}}(s,t)$ such that $\theta \leq \sigma$.

An *$\mathcal{E}$-unification algorithm* decides if a pair s,t of terms is unifiable, and if the answer is yes, it computes a (finite) complete set of $\mathcal{E}$-unifiers. Any such algorithm can be easily extended to give an algorithm to compute a complete set of $\mathcal{E}$-unifiers for any finitely many pairs s_i, t_i of terms.

Given a Σ-structure $\mathcal{M}$, by $\mathcal{T}h(\mathcal{M})$ we denote the (first-order) complete theory defined by all first-order sentences that are true in $\mathcal{M}$.

2.5 Word Problem

Let $\mathcal{E}$ be a set of equations. By the word problem for the theory presented by $\mathcal{E}$ we mean: given a set of ground equations E and a ground equation $s \approx t$, decide if $s \leftrightarrow^*_{\mathcal{E}\cup E} t$. The class of all $\mathcal{E}$-algebras (over the signature Σ) is usually called a *variety* of algebras. Examples include the varieties of groups, semigroups, abelian groups, rings, commutative rings and lattices. With a set of ground equations E (over the signature $\Sigma \cup K$) we can associate an $\mathcal{E}$-algebra $\mathcal{T}(\Sigma \cup K)/\leftrightarrow^*_{\mathcal{E}\cup E}$ in the variety. The set K is usually also called the set of *generators* and the above $\mathcal{E}$-algebra is said to be *presented* by the set E. In this terminology, the word problem for $\mathcal{E}$-algebras asks if given an $\mathcal{E}$-algebra finitely presented by E and two terms in this algebra, are the two terms equal.

Suppose that the word problem for $\mathcal{E}_1$-algebras and $\mathcal{E}_2$-algebras is decidable. Under what conditions can we obtain decidability of the word problem for $\mathcal{E}_1 \cup \mathcal{E}_2$-algebras, using the decision procedures for the word problem for $\mathcal{E}_1$- and $\mathcal{E}_2$-algebras? We consider this general problem in the next chapter. Note that in case the set $\mathcal{E}_1$ is empty, congruence closure provides an efficient decision procedure for the word problem. For the case when $\mathcal{E}_1$ consists of the AC axioms, we introduce a corresponding notion of a AC-congruence closure.

Chapter 3

Abstract Combination Methods

We consider the problem of combining decision procedures to obtain a single decision procedure for the combined theory. In the first part we shall study the Nelson and Oppen's combination procedure which combines satisfiability procedures for a class of first-order theories. Thereafter, we shall present a constructive version of this method for combining procedures for the word problem for different equational theories.

We give only an abstract description of the Nelson and Oppen's combination result here, and refer the reader to the original papers [63, 65], and the recently published more abstract descriptions [78, 71] for more details and proof of correctness.

3.1 The Nelson and Oppen Combination Method

The combination problem can be stated in a general form as follows: Given two first-order (equational) theories specified by set of first-order axioms (equations) $\mathcal{E}_1$ and $\mathcal{E}_2$, respectively over the signature Σ_1 and Σ_2, how is it possible to build a decision algorithm for deciding satisfiability of arbitrary first-order formulas ϕ over $\Sigma_1 \cup \Sigma_2$, in the theory specified by $\mathcal{E}_1 \cup \mathcal{E}_2$, using the decision procedures for individual theories? We can consider several cases of this problem depending on the choice of the signatures Σ_i, axioms of theories $\mathcal{E}_i$ and formulas ϕ. For our purposes, we are mainly interested in the equational case, i.e. when Σ_i contains only function symbols and constants (and no predicates), and $\mathcal{E}_i$ is a set of equational axioms.

Nelson and Oppen give a method to solve this problem for the case when the signatures are allowed to be arbitrary disjoint first-order signatures Σ_1 and Σ_2, and the individual theories are *stably infinite*.

We formulate Nelson and Oppen's combination procedure in terms of transition rules. Since we are interested only in the special case of equational theories, the state is given by a formula of the form,

$$s_1 \approx t_1 \wedge s_2 \approx t_2 \wedge \ldots \wedge s_k \approx t_k \wedge s'_1 \not\approx t'_1 \wedge \ldots \wedge s'_l \not\approx t'_l$$

which is just a conjunction of equations and disequations. The first step is that of variable abstraction, which can be formulated as follows:

$$\textbf{Var Abstraction:} \qquad \frac{\phi[t(s)]}{\phi[t(x)] \ \wedge \ x \approx s}$$

where $x \notin \mathcal{V}(\phi[t(s)])$. In practice, it is useful to apply this rule only when s is an alien subterm of the term t, or when s is the term at the top of an equation or disequation.

Using the above transition rules, we can transform any formula ϕ over a combined signature $\Sigma_1 \cup \Sigma_2$, into a conjunction of *pure* formulas ϕ_1 (over $\Sigma_1 \cup \mathcal{V}$) and ϕ_2 (over $\Sigma_2 \cup \mathcal{V}$).

The next phase involves guessing a suitable equivalence relation on the set of variables that occur in both ϕ_1 and ϕ_2.

$$\textbf{Guess:} \qquad \frac{\phi_1 \ \wedge \ \phi_2}{\phi_1 \ \wedge \ \phi_2 \ \wedge \ \phi_e}$$

where e is an equivalence relation on the set of all variables in ϕ_1 and ϕ_2; and, $\phi_e = \bigwedge_{(x,y)\in e}(x \approx y) \ \wedge \ \bigwedge_{(x,y)\notin e}(x \not\approx y)$ is the conjunction of all equations and disequations that are true of this equivalence e.

The final phase involves a test for satisfiability in individual theories.

$$\textbf{Check:} \qquad \frac{\phi_1 \ \wedge \ \phi_2 \ \wedge \ \phi_e}{\textbf{true}}$$

if $\phi_1 \wedge \phi_e$ is satisfiable in the theory presented by $\mathcal{E}_1$, and $\phi_2 \wedge \phi_e$ is satisfiable in theory presented by $\mathcal{E}_2$.

We shall skip the proof of correctness and refer the reader to [78, 71, 65].

Proposition 1 (Soundness) *Let ϕ_1 and ϕ_2 be first-order formulas over $\Sigma_1 \cup \Sigma_2$ (and variables $\mathcal{V}$) such that ϕ_2 can be derived from ϕ_1 using one of the above mentioned transition rules. If ϕ_2 is satisfiable, then ϕ_1 is satisfiable too assuming that the theories $\mathcal{E}_1$ and $\mathcal{E}_2$ are stably infinite.*

Proposition 2 (Completeness) *Let ϕ be satisfiable in the theory defines by $\mathcal{E}_1 \cup \mathcal{E}_2$. Then, there exists a derivation starting from ϕ with final state* **true**.

3.2 Shostak's Combination Method

In sharp contrast to the Nelson and Oppen's combination result, obtaining a formal description and proof of correctness of Shostak's combination method [75] has been challenging [35, 51]. In this chapter, we shall fill this gap partially, and present Shostak's combination method abstractly as a special case of the more general result of Nelson and Oppen. The abstract description here will be refined further in Section 7.3. However, none of the remaining parts of the thesis depend on the results contained in this section.

Assume that we are given (i) a finitary $\mathcal{E}_i$-unification algorithm, and (ii) a procedure that decides validity of equations in $\mathcal{E}_i$. If this is the case, then given a conjunction $\phi = \phi_{\approx} \wedge \phi_{\not\approx}$ of equation and disequations over $\mathcal{T}(\Sigma_i, \mathcal{V})$, we can
(i) first compute a complete set $cU_{\mathcal{E}_i}(\phi_{\approx})$ of $\mathcal{E}_i$-unifiers for the equations $\phi_{\approx}$ in ϕ, and
(ii) if for some unifier $\sigma \in cU_{\mathcal{E}_i}(\phi_{\approx})$, it is the case that for each conjunct $s_i\sigma \not\approx t_i\sigma$ in the conjunction $\phi_{\not\approx}\sigma$, $s_i\sigma \approx t_i\sigma$ is *not* valid in $\mathcal{E}_i$, then return *true*. If not, then we return *false*.

Clearly, this procedure decides satisfiability of the (existential closure of) formula ϕ in the theory $\mathcal{T}h(\mathcal{T}(\Sigma, \mathcal{V})/\leftrightarrow^*_{\mathcal{E}})$. If there exists a function *canonizer*(t) that returns a unique representative of the term t in its congruence class (modulo $\mathcal{E}_i$), then decidability of the validity of equations in $\mathcal{E}_i$ follows.

Shostak's combination procedure also deals with the pure theory of equality, i.e., a theory defined by the empty set of axioms. Satisfiability in the pure theory of equality can be decided using a *congruence closure* algorithm. We construct a congruence closure of all equation in the formula, and then check to see if the terms in the disequations are distinct in the closure. If this is true for every disequation, then the formula is satisfiable, otherwise it is not. We remark here that this satisfiability algorithm checks for satisfiability in the pure theory of equality, and not in the specific model $\mathcal{T}(\Sigma_0, \mathcal{V})$. We would get the latter *if* we used a unification algorithm to get the most-general unifier for the equations and tested the disequations after applying the unifier to them. Consequently, Shostak's combination method can be thought of as combining complete theories (defined by models) with the pure theory of equality. In other words, Shostak's method combines $\mathcal{E}_i$-unification algorithms with congruence closure.

Thus we have the pure theory $\mathcal{E}_0 = \phi$ of equality over signature Σ_0, and (complete) equational theories $\mathcal{T}h_i = \mathcal{T}h(\mathcal{T}(\Sigma_i, \mathcal{V})/\leftrightarrow^*_{\mathcal{E}_i})$ over signatures Σ_i. The signatures Σ_i

are assumed to be disjoint. For the sake of uniformity, we define $\mathcal{T}h_0$ as $\mathcal{T}h(\mathcal{T}(\Sigma_0, \mathcal{V}))$. Thus, Shostak's procedure decides satisfiability of conjunction of equations and disequations over the combined signature $\Sigma_0 \cup \Sigma_1 \cup \cdots \cup \Sigma_k$ in the combined theory presented by $\mathcal{T}h_1 \cup \mathcal{T}h_2 \cup \cdots \cup \mathcal{T}h_k$.

3.2.1 Abstract Presentation of Shostaks Method

Having thus formulated the problem of combination, we find that we can use Nelson and Oppen's general procedure to obtain a decision procedure for the above problem. Each equational theory $\mathcal{T}h_i$, for $i = 0, 1, \ldots, k$, is stably infinite. Furthermore, the signatures Σ_i are all mutually disjoint. However, we can avoid the non-deterministic guess in the Nelson and Oppen procedure and deduce all equations that need to be exchanged between the different individual theories using the unification algorithm and the validity checking procedure.

The rule for variable abstraction remains the same. From now on, we assume that we are working with three theories: $\mathcal{T}h_0$, the pure theory of equality, and, two equational theories $\mathcal{T}h_1$ and $\mathcal{T}h_2$ as described before, respectively over the mutually disjoint signatures Σ_0, Σ_1 and Σ_2. Thus, using variable abstraction, we obtain a conjunction $\phi_0 \wedge \phi_1 \wedge \phi_2 \wedge \phi_3$, where ϕ_3 contains equations and disequations over variables; and ϕ_0, ϕ_1 and ϕ_2 respectively contain the remaining pure formulae over $\Sigma_0 \cup \mathcal{V}, \Sigma_1 \cup \mathcal{V}$ and $\Sigma_2 \cup \mathcal{V}$ respectively.

We get rid of the non-deterministic *guess*, and replace it by *solve*.

$$\textbf{Solve:} \quad \frac{\phi_0 \ \wedge \ \phi_1 \ \wedge \ \phi_2 \ \wedge \ \phi_3}{\phi_0 \ \wedge \ \phi_1 \ \wedge \ \phi_2 \ \wedge \ \phi_3 \ \wedge \ x \approx y}$$

where $x \approx y$ is an equation between variables deduced by, either, (i) the congruence closure procedure, using the equations in $\phi_0 \wedge \phi_3$; or, (ii) the unification and equality checker for theory $\mathcal{T}h_i$, using the equations in $\phi_i \wedge \phi_3$, for $i = 1, 2$.

The exact way in which these equations are deduced is abstracted away in this description and will be discussed in Sections 5.1 and 7.3. Here we simply assume (to prove correctness) that if an equation between variables can be deduced from the individual theories, then we can always apply the *solve* rule that introduces that equation.

A derivation can be terminated if we deduce an inconsistency.

$$\textbf{Terminate:} \quad \frac{\phi_0 \ \wedge \ \phi_1 \ \wedge \ \phi_2 \ \wedge \ \phi_3 \ \wedge \ x \approx y \ \wedge \ x \not\approx y}{\text{false}}$$

Using the correctness of Nelson and Oppen's combination method, we obtain the following.

Proposition 3 *Let $\mathcal{T}h_0, \mathcal{T}h_1$ and $\mathcal{T}h_2$ be theories defined as above over signatures Σ_0, Σ_1 and Σ_2 respectively. A formula ϕ (over signature $\Sigma_0 \cup \Sigma_1 \cup \Sigma_2$) is unsatisfiable in the theory presented by $\mathcal{T}h_1 \cup \mathcal{T}h_2$ if, and only if, there is a derivation $\phi \vdash \cdots \vdash$ false.*

Example 1 *Consider the formula*

$$\phi \equiv z \approx f(x - y) \ \wedge \ x \approx z + y \ \wedge \ -y \not\approx -(x - f f z)$$

The individual theories under consideration here are the pure theory of equality $\mathcal{E}_0$ over signature $\Sigma_0 = \{f\}$, and the theory of real linear arithmetic over the signature $\Sigma_1 = \{+, -, \mathcal{R}\}$, where $\mathcal{R}$ denotes all the reals.

Variable abstraction step introduces four new variables say z_0, z_1, z_2 and z_3 to give the separated *form $\phi \equiv phi_0 \wedge \phi_1 \wedge \phi_2$, where:*

$$\begin{aligned}
\phi_0 &\equiv z \approx f z_0 \ \wedge \ f f z \approx z_1 \\
\phi_1 &\equiv z_0 \approx x - y \ \wedge \ x \approx z + y \ \wedge \ -y \approx z_2 \ \wedge \ -(x - z_1) \approx z_3 \\
\phi_2 &\equiv z_2 \not\approx z_3
\end{aligned}$$

Using a solver (unifier) and canonizer for real linear theory of arithmetic, we infer the equation $z \approx z_0$ from the formula ϕ_1. Adding this equation, we get

$$\begin{aligned}
\phi_0 &\equiv z \approx f z_0 \ \wedge \ f f z \approx z_1 \\
\phi_1 &\equiv z_0 \approx x - y \ \wedge \ x = z + y \ \wedge \ -y \approx z_2 \ \wedge \ -(x - z_1) \approx z_3 \\
\phi_2 &\equiv z_2 \not\approx z_3 \ \wedge \ z_0 \approx z
\end{aligned}$$

Now, $\phi_0 \wedge \phi_2$ can be used to deduce (via congruence closure algorithm) the equation $z_1 \approx z$. Using this new equation and $\phi_1 \wedge \phi_2$, the real arithmetic equality decision procedure deduces the equation $z_2 \approx z_3$, which leads to a contradiction. Hence the original formula is unsatisfiable.

3.3 A Constructive Combination Result

We note that a decision procedure for testing satisfiability of conjunction of equations and disequations in a theory presented by $\mathcal{E}_i$ yields a decision procedure for the word problem for $\mathcal{E}_i$-algebras. One aspect about the Nelson and Oppen description is the non-deterministic guess of an equivalence relation on the shared variables. In practice (and as described by Nelson and Oppen originally), equations between variables are deduced by individual theories which are then shared across the different deduction engines.

In an equational framework, completion provides a convenient way to deduce equations—and if we are interested in equations between certain constants (or variables), we just need to make sure that these constants (or variables) are minimal in the ordering with respect to which we perform completion.

Therefore, we address the following combination scenario: Let $\mathcal{E}_1$ and $\mathcal{E}_2$ be equational specifications for two theories of interest. We assume that are given a completion-based procedure that can transform any finitely presented $\mathcal{E}_i$-algebra into a convergent presentation (under an ordering that satisfies some conditions). Can we obtain a completion-like procedure to construct a convergent presentation of any given $\mathcal{E}_1 \cup \mathcal{E}_2$-algebra?

One crucial difference with most of the other studies in the area of modularity in term rewriting systems is that we allow for extensions of signatures. In other words, when performing completion of a set of equations over the union $\Sigma_1 \cup \Sigma_2$ of signatures, rather than working over the signature $\Sigma_1 \cup \Sigma_2$, we work over an extended signature $\Sigma_1 \cup \Sigma_2 \cup K$ where K is a new set of constant symbols.

3.3.1 Conditions on the theories

Let $\mathcal{E}_1 \cup \mathcal{R}_1$ be a finite set of equations and rules over a signature $\Sigma_1 \cup \mathcal{V}$, where $\mathcal{V}$ is a set of variables. We assume that

Condition 1 The set of rules $\mathcal{R}_1$ is ground convergent modulo $\mathcal{E}_1$.

Therefore, we have,

$$s \;\leftrightarrow^*_{\mathcal{E}_1 \cup \mathcal{R}_1}\; t \qquad \text{iff} \qquad s \;\rightarrow^*_{\mathcal{E}_1 \backslash \mathcal{R}_1^{e_1}}\; \circ \;\leftrightarrow^*_{\mathcal{E}_1}\; \circ \;\leftarrow^*_{\mathcal{E}_1 \backslash \mathcal{R}_1^{e_1}}\; t,$$

for all ground terms $s, t \in \mathcal{T}(\Sigma)$. The notation $\mathcal{R}_1^{e_1}$ denotes the extension of the rewrite system $\mathcal{R}_1$, which is needed when the set $\mathcal{E}_1$ encodes certain axioms as associativity and commutativity.

We can specify many first-order equational theories in the formulation specified above. We consider some examples below.

Example 2 *In the special case when both the sets $\mathcal{E}_1$ and $\mathcal{R}_1$ are empty, we get the pure theory of equality. In this case, we can assume Σ_1 to be any arbitrary set of constants and function symbols.*

Example 3 *Let $\Sigma_1 = \{f^{(2)}, c_1, \ldots, c_k\}$ consist of a single binary function symbol f and finitely many constants. The theory of free commutative semigroup (over the generators $c_1, \ldots, c_k$) is obtained by choosing the set $\mathcal{R}_1$ to be empty, and $\mathcal{E}_1$ to be the set of following AC axioms:*

$$f(x,y) \approx f(y,x) \qquad f(f(x,y),z) \approx f(x,f(y,z)).$$

Example 4 *To the theory of commutative semigroups specified in example 3, if we additionally add a distinguished constant 1 in the signature Σ_1, and the identity axiom*

$$f(x,1) \approx x$$

is taken as the set $\mathcal{R}_1$, then we get the theory of commutative monoids.

Example 5 *If the signature Σ_1 contains a finite set of binary function symbols, and $\mathcal{E}_1$ is the set of AC axioms (for each binary symbol in Σ_1), we obtain associative-commutative theories.*

Example 6 *Commutative theories are given by choosing $\mathcal{E}_1 = \{f(x,y) = f(y,x)\}$, $\mathcal{R} = \emptyset$ and $f \in \Sigma_1$.*

Example 7 *Let $\Sigma_1 = \{\Omega, I, X_1, X_2, \ldots, X_n, \cdot, \oplus, \ominus, \otimes, B, M\}$. Let $\mathcal{E}_1$ be the set of AC axioms for the binary operations $\oplus, \otimes$ and $\cdot$. Then, we can specify the equational theory of polynomial rings (with coefficients from the domain B, and indeterminates $X_1, X_2, \ldots, X_n$) using a set of rewrite rules $\mathcal{R}$, which is ground convergent modulo $\mathcal{E}_1$ (see Chapter 8).*

We further assume that

Condition 2 Corresponding to any finite set E_1 of ground equations over an extended signature $\Sigma_1 \cup K$ where K is a set of constants, and any given total ordering $\succ_K$ on the set K, there exists a finite set R_1 of rewrite rules, such

that,
(i) for all $s, t \in \mathcal{T}(\Sigma_1 \cup K)$,

$$s \leftrightarrow^*_{E_1 \cup \mathcal{R}_1 \cup \mathcal{E}_1} t \quad \text{iff} \quad s \rightarrow^*_{\mathcal{E}_1 \backslash R_1^{e_1} \cup \mathcal{R}_1^{e_1}} \circ \leftrightarrow^*_{\mathcal{E}_1} \circ \leftarrow^*_{\mathcal{E}_1 \backslash R_1^{e_1} \cup \mathcal{R}_1^{e_1}} t;$$

(ii) the system $R_1^{e_1} \cup \mathcal{R}_1^{e_1}$ is reducing with respect to an ordering which is an extension of the ordering $\succ_K$ on K, and in which the only terms smaller than any constant in K are (possibly) other constant in K.

Let $\mathcal{C}om$ be a functor that maps a given set E_1 and ordering $\succ_K$ to the set R_1 specified in the above condition, i.e., $\mathcal{C}om(E_1, \succ_K) = R_1$.

3.3.2 The combination result

We formally state the abstract result now.

Theorem 1 *Let $\mathcal{E}_1 \cup \mathcal{R}_1$ and $\mathcal{E}_2 \cup \mathcal{R}_2$ be two theories over disjoint signatures Σ_1 and Σ_2 respectively, satisfying the conditions 1 and 2 mentioned in Section 3.3.1. If E is a finite set of ground equations over $\Sigma_1 \cup \Sigma_2$, then, there exists a finite set $R = R_1 \cup R_2$ of rewrite rules over the signature $\Sigma_1 \cup \Sigma_2 \cup K$, such that for any $s, t \in \mathcal{T}(\Sigma_1 \cup \Sigma_2)$,*

$$s \leftrightarrow^*_{E, \mathcal{R}_1, \mathcal{E}_1, \mathcal{R}_2, \mathcal{E}_2} t \quad \textit{iff} \quad s \rightarrow^*_{\mathcal{E}_1, \mathcal{E}_2 \backslash R^e, \mathcal{R}_1^{e_1}, \mathcal{R}_2^{e_2}} \circ \leftrightarrow^*_{\mathcal{E}_1, \mathcal{E}_2} \circ \leftarrow^*_{\mathcal{E}_1, \mathcal{E}_2 \backslash R^e, \mathcal{R}_1^{e_1}, \mathcal{R}_2^{e_2}} t;$$

where R_1 is over the signature $\Sigma_1 \cup K$, R_2 is over $\Sigma_2 \cup K$, and $R^e = R^{e_1} \cup R^{e_2}$. Additionally, the rewrite system $\mathcal{E}_1, \mathcal{E}_2 \backslash R^e, \mathcal{R}_1^{e_1}, \mathcal{R}_2^{e_2}$ is terminating.

The statement of theorem 1 can be generalized from considering two disjoint theories to considering a finite number of them. The general result can be proved by suitably generalizing the proof of the above case.

3.3.3 Correctness

We shall prove theorem 1 now. Our proof is constructive, and a crucial component is the process of variable abstraction. Variable abstraction is useful to get "pure" terms from mixed terms. More specifically, if $s[t]$ is a term over $\Sigma_1 \cup \Sigma_2$, and t is a term over, say Σ_1, then we can replace t by a new constant c. Eventually we would obtain a term over either $\Sigma_1 \cup K$, or, $\Sigma_2 \cup K$ where K is a set of constants. If E is a set

of equations over $\Sigma_1 \cup \Sigma_2$ as specified in the theorem, we start with state $(\emptyset, E)$ and apply the following transition rule,

$$\textbf{Var Abstraction:} \qquad \frac{(K, E \cup \{s[t] \approx t'\}}{(K \cup \{c\}, E \cup \{s[c] \approx t', c \approx t\}}$$

where $c \notin \Sigma \cup K$. In practice, it is useful to apply this rule when t is an alien subterm of the term $s[t]$.

Repeated application of the above rule will lead to a state $(K, E_1 \cup E_2)$ where E_1 is a set of equations over $\Sigma_1 \cup K$ and E_2 is a set of equations over $\Sigma_2 \cup K$.

Lemma 1 *Let E be a set of ground equations over $\Sigma_1 \cup \Sigma_2$. Then there exists a finite derivation $(\emptyset, E) \vdash \cdots \vdash (K, E_1 \cup E_2)$ using only the* Variable-Abstraction *rule such that the sets E_1 and E_2 are respectively over the signatures $\Sigma_1 \cup K$ and $\Sigma_2 \cup K$; and furthermore, for all terms $s, t \in \mathcal{T}(\Sigma_1 \cup \Sigma_2)$,*

$$s \ \leftrightarrow_E^* \ t \qquad \textit{iff} \qquad s \ \leftrightarrow_{E_1 \cup E_2}^* \ t$$

Given set E as specified in the statement of the theorem, we use variable abstraction steps to decompose the set into two sets E_1 and E_2 as stated in lemma 1. The set K of new constants is finite, since the number of variable abstraction steps needed is finite. We choose any total ordering $\succ_K$ on the finite set K.

By condition 2, let $R_1 = \mathcal{C}om(E_1, \succ_K)$ and $R_2 = \mathcal{C}om(E_2, \succ_K)$.

Define the C-component of the set R_i as $C_{R_i} = \{c \to d \in R_i : c, d \in K\}$. Two sets C_1 and C_2 of C-rules are said to be equivalent if they define the same equivalence on the set K.

We define two sequences of sets $R_1^{(j)}$ and $R_2^{(j)}$ as follows:

$$R_i^{(j+1)} \ = \ \begin{cases} R_i & \texttt{if} \quad j = 0 \\ \mathcal{C}om(E_i \cup C_{R_1^{(j)}} \cup C_{R_2^{(j)}}, \succ_K) & \texttt{if} \quad j > 0 \end{cases}$$

where $i = 1, 2$. We assume that if $\leftrightarrow_C^* = \leftrightarrow_{C'}^*$, then $\mathcal{C}om(E \cup C, \succ) = \mathcal{C}om(E \cup C', \succ)$.

Lemma 2 *The sequences $\langle R_1^{(j)} \rangle$ and $\langle R_2^{(j)} \rangle$ are both eventually constant sequences, i.e., there exists a k such that $R_i^{(j)} = R_i^{(k)}$ for all $j \geq k$.*

Proof. Define the measure associated with a set $R_i^{(j)}$ to be the number of different equivalence classes on K modulo the set $R_i^{(j)}$. Our first claim is that both sequences

are non-increasing in this measure. Since the $>$ ordering on natural numbers is well-founded, there exists a k such that the measure of $R_i^{(j)}$ remains unchanged for $j \geq k$. Because of the assumption on $\mathcal{C}om$, each set $R_i^{(k+1)}$ will be equal to the set $R_i^{(k)}$.

In order to prove the claim we note that if $c \leftrightarrow^*_{R_i^{(j)}} d$ for any i, j and $c, d \in K$, then, $c \leftrightarrow^*_{C_{R_i^{(j)}}} d$ since $R_i^{(j)}$ is convergent, and the only rules that can reduce a constant in K can be a C-rule (see condition 2). Since $R_i^{(j+1)}$ is obtained from $E_i \cup C_{R_1^{(j)}} \cup C_{R_2^{(j)}}$, therefore,

$$\begin{array}{rcl} \leftrightarrow^*_{C_{R_1^{(j)}}} & \subset & \leftrightarrow^*_{C_{R_i^{(j+1)}}} \\ \leftrightarrow^*_{C_{R_2^{(j)}}} & \subset & \leftrightarrow^*_{C_{R_i^{(j+1)}}} . \end{array}$$

This implies that the sequences are non-increasing in the measure, and hence the claim is established. □

Lemma 3 (Soundness and Completeness) *Let R_1 and R_2 be the sets to which the above two sequences converge. For $s, t \in \mathcal{T}(\Sigma_1 \cup \Sigma_2)$,*

$$s \quad \leftrightarrow^*_{R_1 \cup R_2 \cup \mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{E}_1 \cup \mathcal{E}_2} \quad t$$

if and only if,

$$s \quad \leftrightarrow^*_{E \cup \mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{E}_1 \cup \mathcal{E}_2} \quad t.$$

Proof. This is a simple consequence of lemma 1 and condition 2. □

Lemma 4 (Convergence) *The rewriting system $\mathcal{E}_1 \cup \mathcal{E}_2 \backslash R_1^{e_1} \cup R_2^{e_2} \cup \mathcal{R}_1 \cup \mathcal{R}_2$ is convergent.*

Proof. Let $c \in K$ be the largest constant in K in the ordering $\succ_K$. To any ground term $t \in \mathcal{T}(\Sigma_1 \cup \Sigma_2 \cup K)$, we assign a measure, $m(t)$, defined recursively as follows:

$$m(t) \quad = \quad \begin{cases} (\{t\}, \emptyset) & \text{if} \quad t \in \mathcal{T}(\Sigma_1 \cup K) \\ (\emptyset, \{t\}) & \text{if} \quad t \in \mathcal{T}(\Sigma_1 \cup K) \\ m(t_1) \cup m(C[f(\ldots, c, \ldots)]) & \text{if} \quad t = C[f(\ldots, t_1, \ldots)], f \in \Sigma_2, \\ & \qquad t_1 \in \tau(\Sigma_1 \cup K) \\ m(t_1) \cup m(C[f(\ldots, c, \ldots)]) & \text{if} \quad t = C[f(\ldots, t_1, \ldots)], f \in \Sigma_1, \\ & \qquad t_1 \in \tau(\Sigma_2 \cup K) \end{cases}$$

where $\cup$ denotes component-wise multiset union. The above measure is clearly well-defined. Two terms are compared by comparing their measures using the lexicographic combination of the multiset extensions of the orderings $\succ_1$ and $\succ_2$ respectively, where $\succ_i$ is the reduction ordering with respect to which the rewrite system $\mathcal{E}_i \backslash R_i^e \cup \mathcal{R}_i^e$ is reducing. The set of rules in $R_1^e \cup R_2^e$ are each reducing with respect to this new ordering on terms. This ordering is also well-founded as it is the multiset combination of well-founded orderings.

To establish confluence, we need to show that peaks and cliffs have rewrite proofs. Since $\Sigma_1 \cap \Sigma_2 = \emptyset$, there are no cliffs arising from proper overlaps. The only non-trivial peaks arise from overlap between C-rules. But such peaks are easily observed to have rewrite proofs. □

This concludes the proof of Theorem 1.

3.4 Specific Instantiations of the Abstract Result

Research activity related to the combination and integration of systems in the areas of logic and automated deduction has been diverse, see [53] for recent trends. There is also a large literature on modularity of term rewriting systems [48]. The crucial difference with most of this work is that we allow for extension of signature. Techniques such as variable abstraction have also been used in obtaining unification algorithms in the union of theories [3, 4]. Bjorner proposes a framework for the integration of decision procedures in his thesis [20] which is a constraint-based version of Shostak's integration method. The framework outlines an interface that various decision procedures should provide which is obtained via suitable generalization of Shostak's *canonizer* and *solver* requirements. However, this framework "is intended to approach concrete problems in verification" and "does not provide deep new theoretical results". Bjorner mentions that Shostak's method can be seen as a specific application of Nelson and Oppen's general result. The fact that Nelson and Oppen's result can be used to obtain satisfiability testing algorithms for certain combination of model generated equational theories was pointed out by Ringeissen [71].

We have presented a result that combines completion algorithms for finitely presented $\mathcal{E}_i \cup \mathcal{R}_i$-algebras satisfying certain conditions. Informally, we require that (i) the class of $\mathcal{E}_i \cup \mathcal{R}_i$-algebras should itself be presented by a convergent rewrite system (i.e. the rewrite system $\mathcal{R}_i$ modulo $\mathcal{E}_i$ should be convergent), and (ii) there should exist algorithms that can obtain a convergent representation for any finitely presented

$\mathcal{E}_i \cup \mathcal{R}_i$-algebra over a signature extended by a finite set of constants under some ordering restrictions. The rest of this thesis discusses in detail several instantiations of this general result.

For any particular class of algebras under consideration, we need to satisfy the two conditions above. In the subsequent chapters, we first start off with the most simple $\mathcal{E}_1 \cup \mathcal{R}_1$-algebras—viz where $\mathcal{E}_1 \cup \mathcal{R}_1 = \emptyset$—and gradually move to more complex algebras.

Chapter 4

Abstract Congruence Closure

We first consider the simplest possible application of the combination result presented in Chapter 3. Consider the case when the set of equational axioms is empty, and hence the theory under consideration is that of the pure theory of equality, which has also been referred to as the theory of equality with uninterpreted function symbols in the literature.

We are interested in the word problem for this theory. Congruence closure algorithms provide a decision procedure for the word problem in this case. In other words, congruence closure algorithms can be used to decide if an equality $s \approx t$ logically follows from a set of equalities $E = \{s_1 \approx t_1, s_2 \approx t_2, \cdots, s_k \approx t_k\}$, where all terms are constructed from uninterpreted or free function symbols and constants. They also provide a decision procedure for validity problem in the quantifier-free theory of equality (with uninterpreted function symbols) [64].

Algorithms to compute "congruence closure" have typically been described in terms of directed acyclic graphs (dags) representing a set of terms, and a union-find data structure storing an equivalence relation on the vertices of this graph. In this chapter, we use the abstract theorem discussed in Chapter 3 to obtain an abstract procedure to decide the word problem.

On closer inspection, we shall find that the set of inference rules obtained by us essentially constitute an abstract description of congruence closure. In particular, the description, called *abstract congruence closure* c.f. [10, 11], can be used to capture abstractly some of the well-known graph-based congruence closure algorithms while still maintaining the "sharing" and "efficiency" offered by the data structures used by these algorithms [13]. In Chapter 5 we shall provide a formulation of logical aspects of the graph-based congruence closure algorithms using the concept of an abstract

congruence closure. Additionally, the abstract congruence closure rules can be used to obtain specific implementations, and in Section 5.5 we describe experimental results comparing these implementations with those of the graph-based congruence closure algorithms.

4.1 Abstract Congruence Closure

Let Σ be a signature, and E a set of ground equations over Σ. The set Σ can be written as a disjoint union of singleton sets Σ_i's where each Σ_i contains exactly one function symbol from Σ. The equations in the set E can be considered to be over the combined signature $\cup_i \Sigma_i$, and hence our first step would be to "purify" them. We introduce constants to name impure subterms and specify the correspondence between the new constants and the original terms by rewrite rules. One can insist that every "pure" equation be either a *simple D-rule* or a *C-rule*.

Definition 1 *Let Σ be a signature and K be a set of constants disjoint from Σ. By a* simple D-rule *(with respect to Σ and K) we mean a rewrite rule of the form*

$$f(c_1, \ldots, c_k) \;\rightarrow\; c_0$$

where $f \in \Sigma$ and $c_1, \ldots, c_k$ are constants in K.

An equation $c \rightarrow d$, where c and d are constants in K, is called a C-rule *(with respect to K).*

Clearly each simple D-rule is over exactly one signature $\Sigma_i \cup K$, whereas each C-rule is over every signature $\Sigma_i \cup K$. The $\mathcal{C}om$ functor corresponds to a completion procedure that works only on equations over the signature $\Sigma_i \cup K$.

For example, let Σ consist of three function symbols, a, b and f, and let E_0 be a set of two equations $a \approx b$ and $fffa \approx fb$. We can choose to represent each different subterm in E_0 by a new constant and get the following set of simple D-rules,

$$D_0 = \{a \rightarrow c_0,\ b \rightarrow c_1,\ fc_0 \rightarrow c_2,\ fc_2 \rightarrow c_3,\ fc_1 \rightarrow c_4\}.$$

We can simplify the original equations in E_0 using these D-rules to obtain a set C_0 of two C-equations, $c_0 \approx c_1$ and $c_3 \approx c_4$.

For certain applications of congruence closure that we describe later, a more general definition of D-rules is useful.

Definition 2 *By a* D-rule *(with respect to* Σ *and* K*) we mean a rewrite rule of the form*

$$t \rightarrow c_0$$

where $t \in \mathcal{T}(\Sigma, K) - K$ *and* c_0 *is a constant in* K*.*

Simple D-rules are also D-rules, and hence in the rest of the chapter we shall consider D-rules in this generality.

The constants in K can be interpreted in various ways. Each constant can be thought of as the name of a vertex in the directed acyclic graph representing all terms in E_0. For instance, in the above example, we have introduced exactly *one* new constant for each subterm. A different choice would be to introduce different constants for different occurrences of the same term, which would obviously result in more D-rule and would correspond to representation of the set of terms as trees.

The constants in the set K can also be thought of as names for equivalence classes of terms. In this interpretation, a simple D-rule $f(c_1, \ldots, c_k) \rightarrow c_0$ can be semantically understood as saying that a term with top symbol f and with arguments in the equivalence classes $c_1, \ldots, c_k$ belongs to the equivalence class c_0. Thus, a set of simple D-rules defines a transition relation of a bottom-up tree automaton [34].

Definition 3 *Let* R *be a set of* D*-rules and* C*-rules (with respect to* Σ *and* K*). We say that a constant* c *in* K represents *a term* t *in* $\mathcal{T}(\Sigma \cup K)$ *(via the rewrite system* R*) if* $t \leftrightarrow_R^* c$*. A term* t *is also said to be* represented *by* R *if it is represented by some constant via* R*.*

For example, the constant c_2 represents the term fa via D_0.

Definition 4 *Let* Σ *be a signature and* K *be a set of constants disjoint from* Σ*. A ground rewrite system* $R = D \cup C$ *is said to be an* (abstract) congruence closure *(with respect to* Σ *and* K*) if*

(i) D *is a set of* D*-rules,* C *is a set of* C*-rules, and every constant* c *in* K *represents at least one term* $t \in \mathcal{T}(\Sigma)$ *via* R*, and*

(ii) R *is ground convergent.*

If E *is a set of ground equations over* $\mathcal{T}(\Sigma \cup K)$ *and in addition* R *is such that,*

(iii) for all terms s *and* t *in* $\mathcal{T}(\Sigma)$*, then* $s \leftrightarrow_E^* t$ *if, and only if,* $s \rightarrow_R^* \circ \leftarrow_R^* t$*,*

then R *will be called an (abstract) congruence closure* for E*.*

Condition (i) essentially states that the rewrite system R is an abstract representation of a data structure called a "signature table" (see Chapter 5) and that no superfluous constants are introduced; condition (ii) ensures that equivalent terms have the same representative; and condition (iii) implies that R is a conservative extension of the equational theory induced by E over $\mathcal{T}(\Sigma)$.

For instance, the rewrite system $R_0 = D_0 \cup \{c_0 \to c_1,\ c_3 \to c_4\}$ above is not a congruence closure for E_0, as it is not a ground convergent rewrite system. But we can transform R_0 into a suitable rewrite system, using a completion-like process described in more detail in the next section, to obtain a congruence closure,

$$R_1 = \{a \to c_1,\ b \to c_1,\ fc_1 \to c_3,\ fc_3 \to c_3,\ c_0 \to c_1,\ c_2 \to c_3,\ c_4 \to c_2\},$$

that provides a more compact representation of E_0. Note that the constant c_3 represents infinitely many terms via R_1, as we have $f^n a \to^*_{R_1} c_3$, for all $n \geq 1$.

4.2 Construction of Congruence Closures

We next present a general method for construction of congruence closures. Our description is fairly abstract, in terms of transition rules that manipulate triples (K, E, R), where K is the set of constants that extend the original fixed signature Σ, E is the set of ground equations (over $\Sigma \cup K$) yet to be processed; and R is the set of C-rules and D-rules that have been derived so far. Triples represent possible *states* in the process of constructing a congruence closure. Construction starts from *initial state* $(\emptyset, E, \emptyset)$, where E is a given set of ground equations.

As outlined above, an abstract congruence closure finds and implicitly stores the equivalence relation between terms, defined by the equations E, through a rewrite relation R over an *extended* signature $\Sigma \cup K$. The transition rules can be derived from those for standard completion as described in [7], with some differences. In particular, in our case (i) application of the transition rules is guaranteed to terminate, and (ii) a convergent system is constructed over an extended signature.

The transition rules do *not* require any reduction ordering[1] on terms in $\mathcal{T}(\Sigma)$, but only only a simple ordering $\succ$ on terms in $\mathcal{T}(\Sigma \cup U)$ (terms in $\mathcal{T}(\Sigma)$ may not be comparable by $\succ$), where U is an infinite set of constants disjoint from Σ and from which new constants $K \subset U$ are chosen. In particular, we assume $\succ_U$ is any ordering

[1] An *ordering* is any irreflexive and transitive relation on terms. A *reduction ordering* is an ordering that is also a well-founded replacement relation.

on the set U, and define $\succ$ by: $c \succ d$ if $c \succ_U d$ and $t \succ c$ if $t \to c$ is a D-rule. For simplicity we take the set U to be $\{c_0, c_1, c_2, \ldots\}$, and assume that $c_i \succ_U c_j$ whenever $i < j$.

A key transformation rule introduces new constants as names for subterms.

$$\textbf{Ext}\text{ension:} \qquad \frac{(K, E[t], R)}{(K \cup \{c\}, E[c], R \cup \{t \to c\})}$$

where $t \to c$ is a D-rule, t is a term occurring in (some equation in) E, and $c \in U - K$.

The following three rules are suitable specializations to the ground case of the corresponding rules for standard completion.

$$\textbf{Sim}\text{plification:} \qquad \frac{(K, E[t], R \cup \{t \to c\})}{(K, E[c], R \cup \{t \to c\})}$$

where t occurs in some equation in E.

It is fairly easy to see that by repeated application of *extension* and *simplification*, any equation in E can be reduced to an equation that can be oriented by the ordering $\succ$.

$$\textbf{Ori}\text{entation:} \qquad \frac{(K \cup \{c\}, E \cup \{t \approx c\}, R)}{(K \cup \{c\}, E, R \cup \{t \to c\})}$$

if $t \succ c$.

Trivial equations may be deleted.

$$\textbf{Del}\text{etion:} \qquad \frac{(K, E \cup \{t \approx t\}, R)}{(K, E, R)}$$

In the case of completion of ground equations, deduction steps can all be replaced by suitable simplification steps, in particular, by *collapse* steps. However, in order to guarantee termination, we formulate *collapse* using two separate transition rules. The usual side condition in the *collapse* rule, which refers to the *encompassment ordering*, can also be suitably simplified.

$$\textbf{Ded}\text{uction:} \qquad \frac{(K, E, R \cup \{t \to c,\ t \to d\})}{(K, E \cup \{c \approx d\}, R \cup \{t \to d\})}$$

$$\textbf{Col}\text{lapse:} \qquad \frac{(K, E, R \cup \{s[t] \to d,\ t \to c\})}{(K, E, R \cup \{s[c] \to d,\ t \to c\})}$$

if t is a proper subterm of s.

As in standard completion the simplification of right-hand sides of rules in R by other rules is optional and not necessary for correctness. The right-hand side term in any rule in R is always a constant.

$$\textbf{Com}\text{position:} \qquad \frac{(K, E, R \cup \{t \to c,\ c \to d\})}{(K, E, R \cup \{t \to d,\ c \to d\})}$$

The core logical aspects of various known congruence closure algorithms can be conveniently described in terms of specific strategies over the above transition rules. We shall elaborate more on this subsequently in Chapter 5, All the above transition rules with the exception of the *composition* rule, constitute the *mandatory* set of transition rules.

We illustrate by an example the process of constructing a congruence closure for the set of equations E_0 using these transition rules.

Example 8 *Consider the set of equations $E_0 = \{a \approx b,\ fffa \approx fb\}$. An abstract congruence closure for E_0 can be derived from the initial state $(K_0, E_0, R_0) = (\emptyset, E_0, \emptyset)$ as depicted in Table 1. The rewrite system R_6 in this table is the required congruence closure.*

i	*Constants* K_i	*Equations* E_i	*Rules* R_i	*Transition Rule*
0	$\emptyset$	E_0	$\emptyset$	
1	$\{c_0\}$	$\{c_0 \approx b, fffa \approx fb\}$	$\{a \to c_0\}$	**Ext**
2	$\{c_0\}$	$\{fffa \approx fb\}$	$\{a \to c_0, b \to c_0\}$	**Ori**
3	$\{c_0\}$	$\{ffc_0 \approx fc_0\}$	$\{a \to c_0, b \to c_0\}$	**Sim** *(twice)*
4	$\{c_0, c_1\}$	$\{fc_1 \approx fc_0\}$	$R_3 \cup \{fc_0 \to c_1\}$	**Ext**
5	$\{c_0, c_1\}$	$\{fc_1 \approx c_1\}$	$R_3 \cup \{fc_0 \to c_1\}$	**Sim**
6	K_5	$\{\}$	$R_5 \cup \{fc_1 \to c_1\}$	**Ori**

Table 1: Example: Intermediate states in a derivation illustrating the abstract congruence closure construction transition rules.

4.3 Correctness

We show that the transition rules presented above compute an abstract congruence closure. We use the symbol $\vdash_{CC}$ (or, $\vdash$ in short) to denote the one-step transition

relation on states induced by the above transition rules. A *derivation* is a sequence of states $(K_0, E_0, R_0) \vdash (K_1, E_1, R_1) \vdash \cdots$ related by the transition relation.

Theorem 2 (Soundness) *If $(K_0, E_0, R_0) \vdash (K_1, E_1, R_1)$, then, for all terms s and s' in $\mathcal{T}(\Sigma \cup K_0)$, $s \leftrightarrow^*_{E_1 \cup R_1} s'$ iff $s \leftrightarrow^*_{E_0 \cup R_0} s'$.*

Proof. For *simplification*, *orientation*, *deletion* and *composition*, the claim follows from correctness result for the standard completion transition rules [7]. We can also easily verify the claim for the specialized *collapse* and *deduction* rules.

Finally, suppose (K_1, E_1, R_1) is obtained from (K_0, E_0, R_0) by *extension* that introduces the rule $t \to c$. For $s, s' \in \mathcal{T}(\Sigma \cup K_0)$, if $s \leftrightarrow_{E_0 \cup R_0} s'$, then clearly $s \leftrightarrow^*_{E_1 \cup R_1} s'$. Conversely, if $s \leftrightarrow_{E_1 \cup R_1} s'$, then, $s\sigma \leftrightarrow_{E_1\sigma \cup R_1\sigma} s'\sigma$, where σ is (homomorphic extension of) the mapping $\langle c \mapsto t \rangle$. But $s\sigma = s$ and $s'\sigma = s'$ as c does not occur in s and s' (see side condition of *extension*). Furthermore, $E_1\sigma = E_0$, and $R_1\sigma = R_0 \cup \{t \to t\}$. Therefore, $s = s\sigma \leftrightarrow^*_{E_0 \cup R_0} s'\sigma = s'$. □

Lemma 5 *Let K_0 be a finite set of constants (disjoint from Σ), E_0 a finite set of equations (over $\Sigma \cup K$) and R_0 a finite set of D-rules and C-rules such that for every C-rule $c \to d \in R_0$, we have $c \succ_U d$. Then, each derivation starting from the state (K_0, E_0, R_0) is finite. Furthermore, if $(K_0, E_0, R_0) \vdash^* (K_m, E_m, R_m)$, then R_m is terminating.*

Proof. We first define a measure on a state (K, E, R) to be the number of occurrences of Σ-symbols in E. Two states are compared by comparing their measures using the usual "greater-than" ordering on natural numbers. It can be easily verified that each transformation rule either reduces this measure, or leaves it unchanged. Specifically, *extension* always reduces this measure.

Any such derivation starting from the state (K_0, E_0, R_0) can be written as

$$(K_0, E_0, R_0) \vdash^* (K_n, E_n, R_n) \vdash (K_{n+1}, E_{n+1}, R_{n+1}) \vdash \cdots$$

where the derivation $(K_n, E_n, R_n) \vdash (K_{n+1}, E_{n+1}, R_{n+1}) \vdash \cdots$ contains no applications of *extension*, and hence, the set $K_n = K_{n+1} = \cdots$ is finite. Therefore, $\succ_{K_n}$, defined as the restriction of $\succ_U$ on the set K_n, is well-founded.

Next we prove that the derivation $(K_n, E_n, R_n) \vdash (K_{n+1}, E_{n+1}, R_{n+1}) \vdash \cdots$ is finite. Assign a weight $w(c)$ to each symbol c in K_n so that $w(c) > w(d)$ iff $c \succ_{K_n} d$. Since K_n is finite, assign $w(f) = \max\{w(c) : c \in K_n\} + 1$, for each $f \in \Sigma$. Let $\gg$

be the Knuth-Bendix ordering using these weights. Define a secondary measure on a state (K, E, R) to be the set $\{\{\{s,t\}\} : s \approx t \in E\} \cup \{\{\{s\},\{t\}\} : s \to t \in R\}$. Two states are compared by comparing their secondary measures using a two-fold multiset extension[2] of the ordering $\gg$ on terms. It is straight-forward to see that application of any transition rule reduces to a state reduces its secondary measure. Therefore, the above derivation is finite. Moreover, every rule in R_j is reducing in the reduction ordering $\gg$, and hence each R_j is terminating. □

The following lemma says that *extension* introduces no superfluous constants.

Lemma 6 *Suppose* $(K_0, E_0, R_0) \vdash (K_1, E_1, R_1)$. *If every constant* c *in* K_0 *represents some term* t *in* $\mathcal{T}(\Sigma)$ *(via* $E_0 \cup R_0$*), then every constant* d *in* K_1 *represents some term* t' *in* $\mathcal{T}(\Sigma)$ *via* $E_1 \cup R_1$.

Proof. If $d \in K_1$ also belongs to the set K_0, then the claim follows from Theorem 2. Otherwise let $d \in K_1 - K_0$. The only non-trivial case is when (K_1, E_1, R_1) is obtained using *extension*. Let $s[c_1, \ldots, c_k] \to d$ be the rule introduced by *extension*, where $c_1, \ldots, c_k$ are all the constants that occur in s. Since $c_1, \ldots, c_k \in K_0$, therefore there are terms $s_1, \ldots, s_k \in \mathcal{T}(\Sigma)$ such that $s_i \leftrightarrow^*_{E_0 \cup R_0} c_i$, and hence, using Theorem 2, $s_i \leftrightarrow^*_{E_1 \cup R_1} c_i$. The term $s[s_1, \ldots, s_k]$ is the required term. □

We call a state (K, E, R) *final* if no mandatory transition rule is applicable. We now prove that in a final state, the third component is a congruence closure.

Theorem 3 *Let* Σ *be a signature and* K_1 *a finite set of constants disjoint from* Σ. *Let* E_1 *be a finite set of equations over* $\Sigma \cup K_1$ *and* R_1 *be a finite set of D-rules and C-rules such that every* $c \in K_1$ *represents some term* $t \in \mathcal{T}(\Sigma)$ *via* $E_1 \cup R_1$, *and* $c \succ_U d$ *for every C-rule* $c \to d$ *in* R_1. *If* (K_n, E_n, R_n) *is a final state such that* $(K_1, E_1, R_1) \vdash^* (K_n, E_n, R_n)$, *then* $E_n = \emptyset$ *and* R_n *is an abstract congruence closure for* $E_1 \cup R_1$ *(over* Σ *and* K_n*).*

Proof. By Lemma 5, we know that K_n, E_n and R_n are all finite sets. If $E_n \neq \emptyset$, then either *extension* or *orientation* will be applicable. Since (K_n, E_n, R_n) is a final state, $E_n = \emptyset$.

In order to show that R_n is an abstract congruence closure for $E_1 \cup R_1$, we need to prove the three conditions in Definition 4.

[2] A *multiset* over a set S is a mapping M from S to the natural numbers. Any ordering $\succ$ on a set S can be extended to an ordering $\succ^m$ on multisets over S as follows: $M \succ^m N$ iff $M \neq N$ and whenever $N(x) > M(x)$ then $M(y) > N(y)$, for some $y \succ x$. The multiset ordering $\succ^m$ (on finite multisets) is well founded if the ordering $\succ$ is well founded [39].

(i) Lemma 6 implies that every $c \in K_n$ represents some term $t \in \mathcal{T}(\Sigma)$ via R_n.

(ii) To prove that R_n is convergent, we first note that R_n is terminating (Lemma 5). Furthermore, since (K_n, E_n, R_n) is a final state, R_n is left-reduced. By the critical pair lemma [5], therefore, R_n is confluent.

(iii) Finally, Theorem 2 establishes that if $s \leftrightarrow^*_{E_1 \cup R_1} t$ for some $s, t \in \mathcal{T}(\Sigma)$, then $s \leftrightarrow^*_{E_n \cup R_n} t$. Since $E_n = \emptyset$ and R_n is convergent, $s \rightarrow^*_{R_n} \circ \leftarrow^*_{R_n} t$. □

4.4 Removing Redundant Constants

To summarize, we have presented an abstract notion of congruence closure and given a method to construct such an abstract congruence closure for a given set of ground equations. The only parameters required by the procedure are a denumerable set U of constants (disjoint from Σ) and an ordering (irreflexive and transitive relation) on this set. It might appear that the abstract congruence closure obtained depends on the ordering $\succ_U$ used. In this section, we show that we can construct an abstract congruence closure that is independent of the ordering on constants.

In the process of construction of an abstract congruence closure, we may deduce an equality between two constants in K, and we require an ordering $\succ_U$ to deal with such equations. Since constants act as "names" for equivalence classes, it is redundant to have two different names for the same equivalence class. Hence, one such constant can be eliminated, and thus the ordering dependence is eliminated.

Definition 5 *Any constant $c \in K$ that occurs as a left-hand side of a C-rule in R will be called a* redundant *constant.*

If R is the third component of a state in construction of an abstract congruence closure, then all constants on left-hand sides of the C rules in R are redundant, and they can be removed after all possible composition and collapse steps using the C-rules have been done.

$$\textbf{Comp}\text{ression:} \qquad \frac{(K \cup \{c, d\}, E, R \cup \{c \rightarrow d\})}{(K \cup \{d\}, E[c \mapsto d], R[c \mapsto d])}$$

if no other C-rule in R has c as the left-hand side term.

Correctness of the new enhanced set of transition rules for construction of congruence closure can be established in the same way as done before.

Theorem 4 *Let Σ be a signature and K_1 a finite set of constants disjoint from Σ. Let E_1 be a finite set of equations over $\Sigma \cup K_1$ and $R_1 = D_1 \cup C_1$ be a finite set of D-rules and C-rules such that every $c \in K_1$ represents some term $t \in \mathcal{T}(\Sigma)$ via $E_1 \cup R_1$, and the rewrite system C_1 is terminating. Then, there exists an abstract congruence closure D_n for $E_1 \cup R_1$ (over Σ and K_1) consisting only of D-rules. Additionally, the set $D_n \cup C_1$ is also an abstract congruence closure for $E_1 \cup R_1$.*

Proof. Let $(K_1, E_1, R_1) \vdash^* (K_n, E_n, R_n)$ such that none of the mandatory transition rules nor *compression* is applicable to the state (K_n, E_n, R_n).

We observe that the following version of soundness (Theorem 2) is still true: if $(K_i, E_i, R_i) \vdash (K_j, E_j, R_j)$, then, for all terms s and t in $\mathcal{T}(\Sigma)$, $s \leftrightarrow^*_{E_j \cup R_j} t$ iff $s \leftrightarrow^*_{E_i \cup R_i} t$. Additionally, Lemma 5 and Lemma 6 continue to hold with the new set of transition rules, and the proofs remain essentially unchanged. This establishes that we can use Theorem 3 in this new setting to conclude that R_n is an abstract congruence closure. Moreover, since none of the mandatory rules (including *compression*) is applicable, there can be no C-rules in R_n. □

The above result states that we can compute an abstract congruence closure that imposes no ordering restrictions (whatsoever) on the set of constants K. This will be useful for applications to rigid E-unification.

A second observation is regarding the definition of D-rules and simple D-rules. The graph-based congruence closure algorithms that have been studied in the literature can be described using only simple D-rules. However, this corresponds to completely "flattening" out the terms and leads to a complete loss of term structure, which could harm certain applications.

A key idea of abstract congruence closure is the use of new constants as names for subterms which yields a concise and simplified term representation. Consequently, 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. If we insist of having only simple D-rules, we are on the extreme that favors simplicity of terms, and if we allow for equalities between arbitrary terms (in the $\mathcal{R}$-component), then we keep the complete term-structure but require complicated orderings. The definition of D-rules gets a middle ground where we keep the term structure as much as possible while eliminating all term ordering requirements.

In fact, we can have an additional rule that eliminates constants which name equivalence classes containing only one "signature", i.e. it undoes the effect of *extension* and attempts to put back more term-structure into the presentation of the

equational theory via the D- and C-rules.

$$\textbf{Pro}\text{jection:} \qquad \frac{(K \cup \{d\}, \emptyset, R \cup \{t \to d\})}{(K, \emptyset, R[d \mapsto t])}$$

if (i) $R \cup \{t \to d\}$ is a left-reduced abstract congruence closure, and (ii) d does not occur as the left- or right-hand side term in any rule in R.

If R is a left-reduced abstract congruence closure for E, then R' obtained using *projection* is also a left-reduced abstract congruence closure for E. The proof is similar to the case above involving addition of *compression*.

4.5 Related Work

The fact that names are introduced for all given terms and subterms is a key characteristic of congruence closure algorithms was also pointed out by Kapur [51]. Kapur describes Shostak's congruence closure using completion with extensions. He uses rewrite rules of the form of simple D-rules and C-rules to describe the procedure. Our description is more general and abstract—and thus allows us to capture other algorithms as well. We shall discuss these aspects further in the next chapter.

Some of the earliest mention of using new constants and reducing a set of ground equations to a set consisting of only D-rules and C-rules appears in the work of Evans [42, 43, 44]. In [42], Evans considers the problem of deciding if two terms are equal in the algebra $\mathbf{A}$ defined over the set $\mathcal{T}(\Sigma)/(\leftrightarrow^*_{\mathcal{E}\cup E})$, where $\mathcal{E}$ is a set of axioms for the class of algebras, and E is a set of ground equations (a finite presentation of the particular $\mathcal{E}$-algebra). It is shown that one can always generate a finite presentation consisting of simple D-rules, of an algebra isomorphic to the algebra $\mathbf{A}$, where the finite presentation satisfies certain additional conditions. In case $\mathcal{E} = \emptyset$, the procedure corresponds to application of *deduction*, *collapse* and *compression* in a specific way. In the general case (i.e. when $\mathcal{E} \neq \emptyset$), the new presentation is obtained by the same process enhanced with an additional rule that introduces certain instances of the axioms in $\mathcal{E}$. Evan's paper also contains a characterization for decidability of the word problem for such finitely presented algebras—but decidability of the characterization is equivalent to the decidability of the word problem itself [43].

There are some similarities between these transformation rules and the "calculus of rewriting with sharing" designed by Sherman [74]. This calculus operates on relations that can be described by D-rules and C-rules, and employs several transformations

rules, most of which can be derived from transformations used in completion. The calculus contains a rule similar to *deduction*, though surprisingly, its application is not obligatory. This may reflect the fact that Sherman uses his calculus to provide an equational semantics for an implementation of a symbolic computation system, but does not address issues such as termination or completeness.

Using names or constants for subterms also captures the notion of sharing, which is fairly well studied. This idea is similar to the idea of using caching techniques for logic programming (also referred to as memoing), and the idea of SOUR graphs [57].

4.6 Summary

In this chapter, we presented an abstract definition of congruence closure—as a ground convergent system over an extended signature; and we also presented completion-like transition rules to construct such a closure. The abstract description is obtained using the framework of Nelson and Oppen's combination method, suitably modified to the context of completion.

In the subsequent chapters, we shall argue for the usefulness of this description. As is obvious, this framework is ideal for adding in theories and we shall concentrate on a specific extension to incorporate associative and commutative function symbols in Chapter 6, see also [11]. We do not describe in detail combination with theories other than associativity and commutativity, as the basic idea remains the same.

We shall also provide a rule-based abstract description of the logical aspects of the various published congruence closure algorithms and that would justify the terminology (*abstract* congruence closure). Additionally, using the abstract rules, we can also get efficient implementation of completion based congruence closure procedure—one can effectively utilize the theory of redundancy to figure out and eliminate inferences which are not necessary, and moreover also use knowledge about efficient indexing mechanisms. These aspects will be discussed in Chapter 5.

The concept of an abstract congruence closure is also relevant for describing applications that use congruence closure algorithms. Some of these applications include efficient normalization by rewrite systems [30, 10], computing a complete set of rigid E-unifiers [79] and combination of congruence closure and E-unification algorithms—all of which are discussed in Chapter 7,

The concept of an abstract congruence closure as detailed here and the rules for computation open up new frontiers too. For example, the transition rules presented

in Section 4.1 can be naturally implemented in MAUDE [33]. Moreover, specific strategies, such as the ones presented in Chapter 5 can be encoded easily too. This might provide a basis for automatically verifying the correctness of congruence closure algorithms[3].

[3]Personal communication with Manuel Clavel.

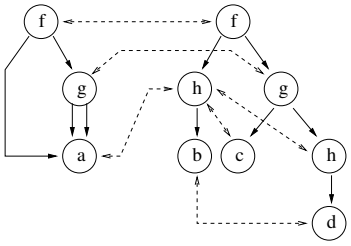
Chapter 5

Congruence Closure Stategies

In this chapter we substantiate our claim that the description of congruence closure given in Chapter 4 is indeed an *abstract* description. This we do by casting the logical aspects of various congruence closure algorithms that appear in the literature in our general framework. In particular, we discuss the algorithms proposed by Downey, Sethi and Tarjan [41], Nelson and Oppen [64] and Shostak [75] as specific variants of our general abstract description. That is, we provide a description of these algorithms (modulo some implementation details) using abstract congruence closure transition rules.

Term directed acyclic graphs (dags) is a common data structure used to implement algorithms that work with terms over some signature—such as the congruence closure algorithm. In fact, many algorithms that have been described for congruence closure assume that the input is an equivalence relation on vertices of a given dag, and the desired output is an equivalence on the same dag that is defined by the congruence relation.

Figure 2 illustrates how a given term dag is (abstractly) represented using D-rules. The solid lines represent *subterm* edges, and the dashed lines represent a binary relation on the vertices. We have a D-rule corresponding to each vertex, and a C-rule for each dashed edge. Note that the D-rules corresponding to a conventional term dag representation are all simple D-rulesas defined in Section 4.1. The definition of D-rules is more general, and allows for arbitrary terms on the left-hand sides. In a sense this corresponds to storing *contexts*, rather than just symbols from Σ, in each node (of the term dag). This is an attempt to keep as much of the term structure information as possible and still get advantages offered by a simplified term representation via extensions.



D-rules representing the term dag:

$$\begin{array}{rclrclrcl} a & \to & c_1 & gc_1c_1 & \to & c_2 & fc_1c_2 & \to & c_3 \\ b & \to & c_4 & hc_4 & \to & c_5 & c & \to & c_6 \\ d & \to & c_7 & hc_7 & \to & c_8 & gc_6c_8 & \to & c_9 \\ fc_5c_9 & \to & c_{10} & & & & & & \end{array}$$

C-rules representing the relation on vertices:

$$\begin{array}{lllllllll} c_1 & \approx & c_5 & c_2 & \approx & c_9 & c_3 & \approx & c_{10} \\ c_4 & \approx & c_7 & c_6 & \approx & c_5 & c_5 & \approx & c_8 \end{array}$$

Figure 2: A term dag and a relation on its vertices

Before we start discussing specific congruence closure algorithms, we need to specify a set U and an ordering $\succ_U$ on this set. Since elements of U serve only as names, we can choose U to be any countable set of symbols. An ordering $\succ_U$ need not be specified a-priori but can be defined on-the-fly as the derivation proceeds. (The ordering has to be extended so that the irreflexivity and transitivity properties are preserved).

Traditional congruence closure algorithms employ data structures such as the following:

(i) Input dag: Starting from the state $(\emptyset, E_0, \emptyset)$, if we apply *extension* and *simplification* to create simple D-rules using the strategy $(\mathbf{Ext} \circ \mathbf{Sim}^*)^*$, we finally get to a state (K_1, E_1, D_1) where all equations in E_1 are of the form $c \approx d$, for $c, d \in K_1$. The set D_1, then, represents the input dag and E_1 represents the (input) equivalence on the vertices of this dag. Note that due to eager *simplification*, we obtain representation of a dag with maximum possible sharing. For example, if $E_0 = \{a \approx b, fffa \approx fb\}$, then $K_1 = \{c_0, c_1, c_2, c_3, c_4\}$, $E_1 = \{c_0 \approx c_1, c_3 \approx c_4\}$ and $R_1 = \{a \to c_0, b \to c_1, fc_0 \to c_2, fc_2 \to c_3, fc_1 \to c_4\}$.

(ii) Signature table: The *signature* table (indexed by vertices of the input dag) stores a *signature*[1] for some or all vertices. Clearly, the signatures are fully left-reduced simple D-rules.

(iii) Use table: The *use* table (also called predecessor list) is a mapping from the constant c to the set of all vertices whose signature contains c. This translates, in our presentation, to a method of indexing the set of D-rules.

[1]The signature of a term $f(t_1, \ldots, t_k)$ is defined as $f(c_1, \ldots, c_k)$ where c_i is the name of the equivalence class containing term t_i.

(iv) Union Find: The union-find data structure that maintains equivalence classes on the set of vertices is represented by the set of C rules. If we apply *orientation* and *simplification* to the state (K_1, E_1, D_1) described above, using the strategy $(\mathbf{Ori} \circ \mathbf{Sim}^*)^*$, we obtain a state $(K_1, \emptyset, D_1 \cup C_1)$. The set C_1 is a representation of the Union-Find structure capturing the input equivalence on vertices. Continuing with the same example, C_1 would be the set $\{c_0 \to c_1, c_3 \to c_4\}$.

We note that, D-rules serve a two-fold purpose: they represent the input term dag, and also a signature table. We shall also note that *composition* is used only implicitly in the various algorithms via path-compression on the union-find structure.

5.1 Shostak's Method

Shostak's congruence closure procedure was first described using *simple* D-rules and C-rules by Kapur [51]. We show here that Shostak's congruence closure procedure is a specific strategy over the general transition rules for abstract congruence closure presented in Section 4.2.

Shostak's congruence closure is *dynamic*: it can accept new equations after it has processed some equations, and can incrementally take care of the new equation. Its input state is $(\emptyset, E_0, \emptyset)$. Shostak's procedure can be described (at a fairly abstract level) as:

$$\mathbf{Shos} = ((\mathbf{Sim}^* \circ \mathbf{Ext}^*)^* \circ (\mathbf{Del} \cup \mathbf{Ori}) \circ (\mathbf{Col} \circ \mathbf{Ded}^*)^*)^*$$

which is implemented as (i) pick an equation $s \approx t$ from the E-component, (ii) use *simplification* to normalize the term s to a term s' (iii) use *extension* to create *simple* D-rules for subterms of s' until s' reduces to a constant, say c, whence *extension* is no longer applicable. Perform steps (ii) and (iii) on the other term t as well to get a constant d. (iv) if c and d are identical then apply *deletion* (and continue with (i)), and if not, create a C-rule using *orientation*. (v) Once we have a new C-rule, perform all possible *collapse* step by this new rule, where each *collapse* step is followed by all the resulting *deduction* steps arising out of that *collapse*. The whole process is now repeated starting from step (i).

Shostak's procedure uses indexing based on the idea of the *use*() list. This *use*() based indexing is used to identify all possible applications of *collapse*.

If the E-component of the state is empty while attempting to apply step (i), Shostak's procedure halts. It is fairly easy to observe that Shostak's procedure halts

in a *final* state. Hence, Theorem 3 establishes that the R-component of Shostak's halting state contains a convergent system and is an abstract congruence closure.

Example 9 *We use the set E_0 used in Example 8 of Section 4.1 to illustrate Shostak's method. We show some of the important intermediate steps of a Shostak derivation in Table 2.*

i	*Constants* K_i	*Equations* E_i	*Rules* R_i	*Transition*
0	$\emptyset$	E_0	$\emptyset$	
1	$\{c_0, c_1\}$	$\{fffa \approx fb\}$	$\{a \to c_0,\ b \to c_1\}$	$\mathbf{Ext}^2 \circ \mathbf{Ori}$
2	$\{c_0, c_1\}$	$\{fffc_1 \approx fb\}$	$\{a \to c_0,\ b \to c_1,\ c_0 \to c_1\}$	$\mathbf{Sim}$
3	$\{c_0, \ldots, c_3\}$	$\{c_3 \approx fb\}$	$R_2 \cup \{fc_1 \to c_2,\ fc_2 \to c_3\}$	$\mathbf{Ext}^2$
4	$\{c_0, \ldots, c_3\}$	$\{c_3 \approx c_2\}$	R_3	$\mathbf{Sim}^2$
5	$\{c_0, \ldots, c_3\}$	$\emptyset$	$R_4 \cup \{c_3 \to c_2\}$	$\mathbf{Ori}$

Table 2: Example: Intermediate states in a derivation illustrating Shostak's dynamic congruence closure algorithm.

5.2 The Downey-Sethi-Tarjan Algorithm

The Downey, Sethi and Tarjan [41] procedures assumes that the input is a dag and an equivalence relation on its vertices, which, in our language, means that the starting state for this procedures is $(K_1, \emptyset, D_1 \cup C_1)$, where D_1 represents the input dag and C_1 represents the initial equivalence. It can be succinctly abstracted as:

$$\mathbf{DST} = ((\mathbf{Col} \circ (\mathbf{Ded} \cup \{\epsilon\}))^* \circ (\mathbf{Sim}^* \circ (\mathbf{Del} \cup \mathbf{Ori}))^*)^*$$

where ϵ is the null transition rule. This strategy is implemented as follows (i) if a *collapse* rule is applicable, it is applied and any resulting new *deduction* steps are also done. This is repeated until no more *collapse* steps are possible. (ii) if no *collapse* steps are possible, then each C-equation in the E-component is picked up sequentially, fully-simplified (*simplification*) and then either deleted (*deletion*) or oriented (*orientation*).

Although the above description captures the essence of the Downey, Sethi and Tarjan procedure, a few implementation details need to be pointed out. Firstly, the Downey, Sethi and Tarjan procedure keeps the original dag (represented by D_1)

intact[2], but changes signatures in a signature table. Hence, in the actual implementation described in [41], the $(\mathbf{Col} \circ (\mathbf{Ded} \cup \{\epsilon\}))^*$ strategy is applied by: (i) deleting all signatures that will be changed, i.e., deleting all D-rules which can be collapsed; (ii) computing new signatures using the *original* copy of the signatures stored in the form of the dag D_1; and, finally, (iii) inserting the newly computed signatures into the signature table and checking for possible *deduction* steps. Our description achieves the same end result, but, by doing fewer inferences.

Secondly, in the Downey, Sethi and Tarjan procedure, for efficiency, an equation $c \approx d$ is oriented to $c \to d$ if the constant c occurs fewer times than d in the signature table. This is done to minimize the number of *collapse* steps. Additionally, indexing based on the *use*() tables is used for efficiently implementing the specific strategy.

Let $(K_1, \emptyset, D_1 \cup C_1) \vdash^! (K_n, E_n, D_n \cup C_n)$ be a derivation using the DST strategy. Then, it is easily seen that the state $(K_n, E_n, D_n \cup C_n)$ is a final state, and hence the set $D_n \cup C_n$ is convergent, and also an abstract congruence closure. We remark here that D_n holds the information that is contained in the signature table, and C_n holds information in the union-find structure. The set C_n is usually considered the output of the Downey, Sethi and Tarjan procedure.

Example 10 *We illustrate the Downey-Sethi-Tarjan algorithm by using the same set of equations E_0, used in Example 8 of Section 4.1. The start state is $(K_1, \emptyset, D_1 \cup C_1)$ where $K = \{c_0, \ldots, c_4\}$, $D_1 = \{a \to c_0,\ b \to c_1,\ fc_0 \to c_2,\ fc_2 \to c_3,\ fc_1 \to c_4\}$, and, $C_1 = \{c_0 \to c_1,\ c_3 \to c_4\}$. We show some of the important intermediate stages in Table 3. Note that $c_4 \approx c_2$ is oriented in a way that no further collapses are needed thereafter.*

i	*Consts* K_i	*Eqns* E_i	*Rules* R_i	*Transition*
1	K_1	$\emptyset$	$D_1 \cup C_1$	
2	K_1	$\emptyset$	$\{a \to c_0,\ b \to c_1,\ fc_1 \to c_2,$ $fc_2 \to c_3,\ fc_1 \to c_4\} \cup C_1$	**Col**
3	K_1	$\{c_2 \approx c_4\}$	R_2	**Ded**
4	K_1	$\emptyset$	$R_3 - \{fc_1 \to c_2\} \cup \{c_4 \to c_2\}$	**Ori**

Table 3: Example: Intermediate states in a derivation illustrating the congruence closure algorithm proposed by Downey, Sethi and Tarjan.

[2]We could make a copy of the original D_1 rules and not change them, while keeping a separate copy as the signatures.

5.3 The Nelson-Oppen Procedure

The Nelson-Oppen procedure is not exactly a completion procedure and it does *not* generate a congruence closure in our sense. The initial state of the Nelson-Oppen procedure is given by the tuple (K_1, E_1, D_1), where D_1 is the input dag, and $\mathcal{E}_1$ represents an equivalence on vertices of this dag. The sets K_1 and D_1 remain unchanged in the Nelson-Oppen procedure. In particular, the inference rule used for deduction is different from the conventional *deduction* rule[3].

$$\textbf{NODed}\text{uction:} \qquad \frac{(K, E, D \cup C)}{(K, E \cup \{c \approx d\}, D \cup C)}$$

if there exist two D-rules $f(c_1, \ldots, c_k) \to c$, and, $f(d_1, \ldots, d_k) \to d$ in the set D; and, $c_i \to^!_C \circ \leftarrow^!_C d_i$, for $i = 1, \ldots, k$.

The Nelson-Oppen procedure can now (at a certain abstract level) be represented as:

$$\mathbf{NO} = (\mathbf{Sim}^* \circ (\mathbf{Ori} \cup \mathbf{Del}) \circ \mathbf{NODed}^*)^*$$

which is applied in the following sense: (i) select a C-equation $c \approx d$ from the E-component, (ii) simplify the terms c and d using *simplification* steps until the terms can't be simplified any more, (iii) either delete, or orient the simplified C-equation, (iv) apply the *NODeduction* rule until there are no more non-redundant applications of this rule, (v) if the E-component is empty, then we stop, otherwise continue with step (i).

Certain details like the fact that newly added equations to the set E are chosen before the old ones in an application of *orientation* and indexing based on the *use*() table, are abstracted away in this description. Additionally, indexing is used to identify which possible D-rules should be considered for *NOCongruence* applications.

Assume that the derivation $(K_1, \mathcal{E}_1, D_1) \vdash^*_{\mathbf{NO}} (K_n, \mathcal{E}_n, D_n \cup C_n)$. is obtained using the Nelson-Oppen strategy. One consequence of using a non-standard deduction rule, *NODeduction*, is that the resulting set $D_n \cup C_n = D_1 \cup C_n$ need not necessarily be convergent, although the the rewrite relation D_n/C_n [38] is convergent.

[3]This rule performs deduction modulo C-equations, i.e., we compute critical pairs between D-rules modulo the congruence induced by C-equations. Hence, the Nelson-Oppen procedure can be described as an *extended completion* [38] (or completion modulo C-equations) method over an extended signature.

Example 11 *Using the same set E_0 as equations, we illustrate the Nelson-Oppen procedure. The initial state is given by (K_1, E_1, D_1) where $K_1 = \{c_0, c_1, c_2, c_3, c_4\}$; $E_1 = \{c_0 \approx c_1,\ c_3 \approx c_4\}$; and, $D_1 = \{a \to c_0,\ b \to c_1,\ fc_0 \to c_2,\ fc_2 \to c_3,\ fc_1 \to c_4\}$. We summarize the details in Table 4.*

i	*Constants* K_i	*Equations* E_i	*Rules* R_i	*Transition*
1	K_1	E_1	D_1	
2	K_1	$\{c_3 \approx c_4\}$	$D_1 \cup \{c_0 \to c_1\}$	**Ori**
3	K_1	$\{c_2 \approx c_4,\ c_3 \approx c_4\}$	R_2	**NODed**
4	K_1	$\{c_3 \approx c_4\}$	$R_2 \cup \{c_2 \to c_4\}$	**Ori**
5	K_1	$\emptyset$	$R_4 \cup \{c_3 \to c_4\}$	**Ori**

Table 4: Example: Intermediate states in a derivation illustrating Nelson and Oppen's congruence closure algorithm.

Consider deciding the equality $fa \approx ffb$. Even though $fa \leftrightarrow^*_{E_0} ffb$, the terms fa and ffb have distinct normal forms with respect to R_5. But equivalent terms in the original term universe have identical normal forms.

5.4 Finite Equivalence Classes

In certain special cases, we can get an almost linear time algorithm for computing the congruence closure for a given set of equations. Let us assume that the signature Σ is finite, and that the set E is finite and is such that every congruence class defined by E (over terms $\mathcal{T}(\Sigma)$) is of finite cardinality. This condition is easily seen to be equivalent to the *acyclicity* condition on the graph formed from the initial term universe dag by contracting vertices that are in the same congruence class. The acyclicity condition is mentioned in [41].

Lemma 7 *Let E be a finite set of ground equations over the signature Σ. Let G be a subterm (directed) graph representing all terms that occur in E. Let G' be the graph obtained from G by collapsing vertices representing terms related by the congruence induced by E into a single vertex. Then, G' is acyclic, if and only if, every congruence class induced by E is finite.*

Proof. If there is a cycle in G', then clearly some equivalence class has infinitely many terms.

To prove the converse we need the correctness of graph-based congruence closure algorithms. Let $c_1, \ldots, c_m$ denote vertices of the graph G'. For each term $f(t_1, \ldots, t_k)$ that occurs in E, introduce a rule $f(c'_1, \ldots, c'_k) \to c'$ where c'_i is the equivalence class of term t_i, and c' is the class of $f(t_1, \ldots, t_k)$. Let R denote the set of all such rules. From correctness of, say Downey, Sethi and Tarjan algorithm, we know that $s \leftrightarrow^*_E t$ iff $s \to^!_R \circ \leftarrow^!_R t$ for all terms $s, t \in \mathcal{T}(\Sigma)$.

Since G' is acyclic, the vertices of G' can be (reverse) topologically ordered, say c_1,-$c_2, \ldots, c_m$ is this order. By an induction on i we can conclude that the set $\{t : t \to^!_R c_i\}$ is finite. Additionally, by an induction on the subterm relation we can conclude that for any term t' in R-normal form, the set $\{t : t \to^!_R t'\}$ is finite. □

Under the above mentioned condition, we can get an almost linear time implementation of the following strategy to compute a congruence closure for a set of equations:

$$\mathbf{Fin} = (\mathbf{Ori}^* \circ \mathbf{NOCon}^*)^*$$

An immediate and useful consequence of this acyclicity condition is that we can define an ordering on the constants K, as follows: if $f(c_1, \ldots, c_k) \to c \in D_1$, then $c \succ c_i$, for each i. It can be easily verified that $\succ$ defines a partial order on K which is also compatible with the congruence relation on the vertices.

The trick is to restrict the application of *orientation* to certain C-equations only. Specifically, we identify a minimal (with respect to $\succ$) equivalence class, and only equations over constants in that equivalence class are oriented. Let us call all such constants, that are equivalent to constants that have been chosen by *orientation*, *fixed*. Now, the application of the *NOCongruence* rules is restricted to those rules which have only the fixed constants on their left-hand sides.

The question of how to efficiently perform *orientation* and *NOCongruence* remains to be discussed. In order to identify minimal constants for *orientation* without ever repeating work, we can do a depth-first search on the dag D_1 (defining the order $\succ$) enhanced with edges representing the equations in $E \cup C$.

In order to efficiently apply *NOCongruence*, rather than keeping complete signatures of vertices, we can just group the D_1 rules so that all potentially overlapping rules would be in the same group of the partition. This is a kind of indexing different from the one based on *use*() list.

If we start with the Downey, Sethi and Tarjan's procedure's start state $(K_1, \emptyset, D_1 \cup C_1)$, we can obtain a linear time implementation for computing congruence closure under the acyclicity condition. This is obtained by suitable choices of data-structures,

see [41] for details.

However, using concepts similar to the ones used above, we can also get an equally efficient implementation of a slightly different strategy, namely,

$$\mathbf{SFin} = (\mathbf{Ori}^* \circ (\mathbf{Col} \circ (\mathbf{Ded} \cup \{\epsilon\}))^*)^*$$

The basic idea is to reduce the number of applications of *collapse* (which correspond to updating of the signatures) to $O(m)$. In an application of *collapse*, we replace some constant c by its C-normal form c^* *only when* we are sure that the equivalence class of c^* has been completely determined (i.e. no other terms need to be added to it) and hence, c^* will not have to be collapsed to any other constant later. The fixed equivalence classes can be identified as they will appear in the set C (because we restrict *orientation* to minimal constants as before). Again, we can restrict *deduction* so that we consider overlaps only on signatures in which every constant that appears has been fixed.

Note that if the assumption that all congruence classes are finite is violated, then $\succ$ will not be a partial order, and *orientation* (subject to the minimality condition) will fail.

Example 12 *If we try to construct a congruence closure for the set of equations E_0 (from example 8), then we will note that we get stuck. This is because the equivalence class containing fa is not finite - it contains $f^n a$, for all $n \geq 1$. In the following, we give some of the important intermediate stages (using* **Fin***):*

i	*Constants* K_i	*Equations* E_i	*Rules* R_i
1	K_1	$\{c_0 \approx c_1,\ c_3 \approx c_4\}$	D_1
2	K_1	$\{c_3 \approx c_4\}$	$D_1 \cup \{c_0 \rightarrow c_1\}$
3	K_1	$\{c_2 \approx c_4,\ c_3 \approx c_4\}$	R_2

We get stuck at this last stage as we are unable to perform orientation *since there is no minimal equivalence class in E (actually there is only one equivalence class, that of, say, c_2, but, $c_4 \succ c_2$.)*

5.5 Experimental Results

We have implemented five congruence closure algorithms, including those proposed by Nelson and Oppen (NO) [64], Downey, Sethi and Tarjan (DST) [41], and Shostak [75],

and two algorithms based on completion—one with an indexing mechanism (IND) and the other without (COM). The implementations of the first three procedures are based on the representation of terms by directed acyclic graphs and the representation of equivalence classes by a union-find data structure. The completion procedure COM uses the following strategy:

$$((\mathbf{Sim}^* \circ \mathbf{Ext}^*)^* \circ (\mathbf{Del} \cup \mathbf{Ori}) \circ (\mathbf{Com} \circ \mathbf{Col})^* \circ \mathbf{Ded}^*)^*.$$

The indexed variant IND uses a slightly different strategy

$$((\mathbf{Sim}^* \circ \mathbf{Ext}^*)^* \circ (\mathbf{Del} \cup \mathbf{Ori}) \circ (\mathbf{Col} \circ \mathbf{Com} \circ \mathbf{Ded})^*)^*.$$

Indexing in the case of completion refers to the use of suitable data structures to efficiently identify which D-rules contain specified constants.

In a first set of experiments, we assume that the input is a set of equations presented as pairs of trees (representing terms). We added a preprocessing step to the NO and DST algorithms to convert the given input terms into a dag and initialize the other required data-structures. The other three algorithms interleave construction of a dag with deduction steps. The published descriptions DST and NO do not address the construction of a dag. Our implementation maintains the list of terms that have been represented in the dag in a hash table and creates a new node for each term not yet represented. We present below a sample of our results to illustrate some of the differences between the various algorithms.

The input set of equations E can be classified based on: (i) the size of the input and the number of equations, (ii) the number of equivalence classes on terms and subterms of E, and, (iii) the size of the *use* lists. The first set of examples are relatively simple and developed by hand to highlight strengths and weaknesses of the various algorithms. Example (a)[4] contains five equations that induce a single equivalence class. Example (b) is the same as (a), except that it contains five copies of all the equations. Example (c)[5] requires slightly larger *use* lists. Finally, example (d)[6] consists of equations that are oriented in the "wrong" way.

In Table 5 we compare the different algorithms by their total running time, *including* the preprocessing time. The times shown are the averages of several runs on

[4]The equation set is $\{f^2(a) \approx a, f^{10}(a) \approx f^{15}(b), b \approx f^5(b), a \approx f^3(a), f^5(b) \approx b\}$.

[5]The equation set is $\{g(a,a,b) \approx f(a,b),\ gabb \approx fba,\ gaab \approx gbaa,\ gbab \approx gabb,\ gbba \approx gbab,\ gaaa \approx faa,\ a \approx c,\ c \approx d,\ d \approx e,\ b \approx c1,\ c1 \approx d1,\ d1 \approx e1\}$.

[6]The set is $\{g(f^i(a), h^{10}(b)) \approx g(a,b), i = \{1, \cdots, 25\}, h^{47}(b) \approx b, b \approx h^{29}(b), h(b) \approx c0, c0 \approx c1, c1 \approx c2, c2 \approx c3, c3 \approx c4, c4 \approx a, a \approx f(a)\}$.

a Sun Ultra workstation under similar load conditions. The time was computed using the *gettimeofday* system call. In Table 5 *Eqns* refers to the number of equations; *Vert* to the number of vertices in the initial dag; and *Class* to the number of equivalence classes induced on the dag.

	Eqns	Vert	Class	DST	NO	SHO	COM	IND
Ex.a	5	27	1	1.286	1.640	0.281	0.606	0.409
Ex.b	20	27	1	2.912	2.772	0.794	1.858	0.901
Ex.c	12	20	6	1.255	0.733	0.515	0.325	0.323
Ex.d	34	105	2	10.556	22.488	7.275	12.077	4.416

Table 5: Comparison between congruence closure algorithms in terms of the total running time (in milliseconds) on small examples.

Table 6 contains similar comparisons for considerably larger examples consisting of randomly generated equations over a specified signature. Again we show total running time, including preprocessing time[7]. The column Σ_i denotes the number of function symbols of arity i in the signature and d denotes the maximum term depth.

	Eqns	Vert	Σ_0	Σ_1	Σ_2	d	Cla ss	DST	NO	SHO	IND
Ex.1	10000	17604	2	0	2	3	7472	11.1	3.2	10.2	13.0
Ex.2	5000	4163	2	1	1	3	3	2.3	306.2	3.1	0.8
Ex.3	5000	7869	3	0	1	3	2745	2.4	1.4	3.5	4.0
Ex.4	6000	8885	3	0	1	3	9	3.6	1152.7	52.4	7.1
Ex.5	7000	9818	3	0	1	3	1	4.6	1682.8	47.8	5.5
Ex.6	5000	645	4	2	0	23	77	1.2	1.6	0.4	0.4
Ex.7	5000	1438	10	2	0	23	290	1.5	3.7	0.4	0.4

Table 6: Comparison between congruence closure algorithms in terms of the total running time (in seconds) on large randomly generated examples.

In Table 7 we show the time for computing a congruence closure assuming terms are already represented by a dag. In other words, we do not include the time it takes to create a dag. Note that we include no comparison with Shostak's method,

[7]Times for COM are not included as indexing is indispensable for larger examples.

as the dynamic construction of a dag from given term equations is inherent in this procedure. However, a comparison with a suitable strategy (in which all extension steps are applied before any deduction steps) of IND is possible. We denote by IND* indexed completion based on a strategy that first constructs a dag. The examples are the same as in Table 6.

	DST	NO	IND*
Ex1	0.919	0.296	0.076
Ex2	0.309	319.112	1.971
Ex3	0.241	0.166	0.030
Ex4	0.776	1117.239	7.301

	DST	NO	IND*
Ex5	0.958	1614.961	9.770
Ex6	0.026	0.781	0.060
Ex7	0.048	2.470	0.176

Table 7: Comparison between congruence closure algorithms in terms of the running time (in seconds) assuming input is in a dag form.

Several observations can be drawn from these results. First, the Nelson-Oppen procedure NO is competitive only when few deduction steps are performed and the number of equivalence classes is large. This is because it uses a non-standard deduction rule, which forces the procedure to unnecessarily repeat the same deductions many times over in a single execution. Not surprisingly, straight-forward completion without indexing is also inefficient when many deduction steps are necessary. Indexing is of course a standard technique employed in all practical implementations of completion.

The running time of the DST procedure critically depends on the size of the hash table that contains the signatures of all vertices. If the hash table size is large, enough potential deductions can be detected in (almost) constant time. If the hash table size is reduced, to say 100, then the running times increased by a factor of up to 50. A hash table with 1000 entries was sufficient for our examples (which contained fewer than 10000 vertices). Larger tables did not improve the running times.

Indexed Completion, DST and Shostak's method are roughly comparable in performance, though Shostak's algorithm has some drawbacks. For instance, equations are always oriented from left to right. In contrast, Indexed Completion always orients equations in a way so as to minimize the number of applications of the collapse rule, an idea that is implicit in Downey, Sethi and Tarjan's algorithm. Example (*b*) illustrates this fact. More crucially, the manipulation of the *use* lists in Shostak's method is done in a convoluted manner due to which redundant inferences may be done when

searching for the correct non-redundant ones[8]. As a consequence, Shostak's algorithm performs poorly on instances where *use* lists are large and deduction steps are many such as in Examples (c), 4 and 5.

Finally, we note that the indexing used in our implementation of completion is simple—with every constant c we associate a list of D-rules that contain c as a subterm. On the other hand DST maintains at least two different ways of indexing the signatures, which makes it more efficient when the examples are large and deduction steps are plenty. On small examples, the overhead to maintain the data structures dominates. This also suggests that the use of more sophisticated indexing schemes for indexed completion might improve its performance.

5.6 Related Work and Other Remarks

Kapur [51] considered the problem of casting Shostak's congruence closure [75] algorithm in the framework of ground completion on rewrite rules. Our work has been motivated by the goal of formalizing not just one, but several congruence closure algorithms, so as to be able to better compare and analyze them.

We have suggested that, abstractly, congruence closure can be *defined* as a ground convergent system; and that this definition does not restrict the applicability of congruence closure. The rule-based abstract description of the logical aspects of the various published congruence closure algorithms leads to a better understanding of these methods. It explains the observed behaviour of implementations and also allows one to identify weaknesses in specific algorithms.

The paper also illustrates the use of an extended signature as a formalism to model and subsequently reason about data structures like the term dags, which are based on the idea of structure sharing. Term dag representations are also crucial for obtaining efficient syntactic unification algorithms. Hence this formalism can be used to describe some of these unification algorithms too [14]. In Section 7.2.4 we shall elaborate more on this.

[8]The description in Section 5.1 accurately reflects the logical aspects of Shostak's algorithm, but does not provide details on data structures like the *use* lists.

Chapter 6

Congruence Closure Modulo Associativity and Commutativity

We next generalize the inference rules presented in section 4.2 for computing a congruence closure to the case when certain function symbols are known to be associative and commutative. Associativity and commutativity properties appear in many applications. Being non-ground, it is not clear how one can incorporate these properties in the classical description of congruence closure algorithms. Our approach of describing congruence closure using completion is easily generalized to handle the presence of AC symbols.

We also consider the problem of constructing a convergent rewrite system (over the same signature) for a ground AC-theory. However, rather than obtaining a system R of rules such that $\rightarrow_{AC\setminus R^e}$ is convergent, we construct a set R such that a different reduction relation, which we denote by $\rightarrow_{P\setminus R^s}$ is convergent. This new notion is much simpler than usual rewriting modulo AC. This set is constructed by transforming the obtained AC-congruence closure, which is a convergent system over an extended signature, to a set over the original signature, see Section 6.6.

An interesting fall out of this second result is the case when there are no associative and commutative symbols in the signature. In this special case, the definition of AC-congruence closure specializes to that of abstract congruence closure defined before. Additionally, the transformation from a (AC) congruence closure to a convergent system over the original signature also specializes to a simple method for construction of ground convergent rewrite systems (with the standard definition of rewriting) for theories presented by a set of ground equations. This provides a technique, different from the traditional critical-pair completion (over original signature) approach, for

obtaining a ground convergent system. The new method is efficient (as it allows for sharing structure via constants) and does not use any total reduction ordering over terms in the original signature.

6.1 Preliminaries

Let Σ be a signature consisting of constants and function symbols, and $\mathcal{V}$ be a set of variables. The arity of a symbol f in Σ, denoted by $\alpha(f)$, is a set of natural numbers. The set of ground terms $\mathcal{T}(\Sigma)$ over Σ is the smallest set containing $\{c \in \Sigma : \alpha(c) = \{0\}\}$ and such that $f(t_1, \ldots, t_n) \in \mathcal{T}(\Sigma)$ whenever $f \in \Sigma, n \in \alpha(f)$ and $t_1, \ldots, t_n \in \mathcal{T}(\Sigma)$.

If $\Sigma_{AC} \subset \Sigma$ is a finite set of function symbols, we denote by P the identities

$$f(x_1, \ldots, x_k, s, y_1, \ldots, y_l, t, z_1, \ldots, z_m) \approx f(x_1, \ldots, x_k, t, y_1, \ldots, y_l, s, z_1, \ldots, z_m)$$

where $f \in \Sigma_{AC}$ and $k + l + m + 2 \in \alpha(f)$; and by F the set of identities

$$f(x_1, \ldots, x_m, f(y_1, \ldots, y_r), z_1, \ldots, z_n) \approx f(x_1, \ldots, x_m, y_1, \ldots, y_r, z_1, \ldots, z_n)$$

where $f \in \Sigma_{AC}$ and $\{m + n + 1, m + n + r\} \subset \alpha(f)$. The congruence induced by P is called a *permutation congruence*. Flattening refers to normalizing a term with respect to the set F (considered as rewrite rules). The set $AC = F \cup P$ defines an AC-theory. The symbols in Σ_{AC} are called associative-commutative operators[1]. We require that $\alpha(f)$ be singleton for all $f \in \Sigma - \Sigma_{AC}$ and $\alpha(f) = \{2, 3, 4, \ldots\}$ for all $f \in \Sigma_{AC}$.

Rewriting in presence of associative and commutative operators requires *extensions* of rules [67]. Given an AC-operator f and a rewrite rule $\rho : f(c_1, \ldots, c_k) \to c$, we define its extension ρ^e as $f(c_1, \ldots, c_k, x) \to f(c, x)$. Given a set of rewrite rules R, by R^e we denote the set R plus extensions of rules in R. In short, when working with AC-symbols (i) extensions have to be used for rewriting terms and computing critical pairs, and (ii) syntactic matching and unification is replaced by matching and unification modulo AC. Reduction modulo AC is defined by $s \to_{R \setminus AC} t$ iff there exists a rule $l \to r$ in R and a substitution σ such that $s = s[l']$, $l' \leftrightarrow^*_{AC} l\sigma$, and $t = s[r\sigma]$; see [38] for details.

[1]The equations $F \cup P$ define a conservative extension of the theory of associativity and commutativity to varyadic terms. For a fixed arity binary function symbol, the equations $f(x, y) \approx f(y, x)$ and $f(f(x, y), z) \approx f(x, f(y, z))$ define an AC-theory.

6.2 Associative-Commutative Rules

Let Σ be a signature, and E a set of ground equations over Σ. Let Σ_{AC} be some subset of Σ, containing all the associative-commutative operators and AC be the set $F \cup P$ as defined above. The set Σ can be written as a disjoint union of singleton sets Σ_i's, where each Σ_i contains exactly one function symbol in Σ.

We note that apart from the simple D-rules and the C-rules obtained as a result of variable abstraction, in the presence of AC-symbols, we shall additionally obtain AC-rules. In other words, the word problem for finitely presented commutative semigroups can *not* be solved by turning all equations to either D- or C-rules; we *require* slightly more general rules.

Definition 6 *Let Σ be a signature and K be a set of constants disjoint from Σ. Rewrite rules, which when fully flattened are of the form $f(c_1, \ldots, c_k) \to f(d_1, \ldots, d_l)$, where $f \in \Sigma_{AC}$, and $c_1, \cdots, c_k, d_1, \cdots, d_l \in K$, will be called AC-rules.*

Note that each simple D- and AC-rule is over exactly one signature $\Sigma_i \cup K$. The C-rules are equations over every signature $\Sigma_i \cup K$.

For example, let Σ consist of function symbols, a, b, c, f and g, (f is AC) and let E_0 be a set of three equations $f(a, c) \approx a,\ f(c, g(f(b, c))) \approx b$ and $g(f(b, c)) \approx f(b, c)$. Viewing Σ as the disjoint union of signatures $\Sigma_1 = \{a\}$, $\Sigma_2 = \{b\}$, $\Sigma_3 = \{c\}$, $\Sigma_4 = \{f\}$ and $\Sigma_5 = \{g\}$, we can introduce new constants $K = \{c_0, c_1, c_2, c_3, c_4\}$ so that each equation (or rule) is over some $\Sigma_i \cup K$. Therefore, if we let

$$D_1 = \{a \to c_1,\ b \to c_2,\ c \to c_3,\ f(c_2, c_3) \to c_4,\ g(c_4) \to c_5\},$$

then, using these rules we can simplify the original equations in E_0 to obtain the new set $E_1 = \{f(c_1, c_3) \approx c_1,\ f(c_3, c_5) \approx c_2,\ c_5 \approx c_4\}$.

We can now generalize all definitions made in the context of an abstract congruence closure ($\mathcal{E} = \emptyset$) to the case when $\mathcal{E}$ is the set of AC axioms.

Definition 7 *Let R be a set of D-rules, C-rules and AC-rules (with respect to Σ and K). We say that a constant c in K* represents *a term t in $\mathcal{T}(\Sigma \cup K)$ (via the rewrite system R) if $t \leftrightarrow^*_{AC \setminus R^e} c$. A term t is also said to be* represented *by R if it is represented by some constant via R.*

For example, the constant c_4 represents the term $f(c, b)$ via D_1.

Definition 8 *Let Σ be a signature and K be a set of constants disjoint from Σ. A ground rewrite system $R = AC^e \cup D^e \cup C$ is said to be an* associative-commutative congruence closure *(with respect to Σ and K) if*

(i) D is a set of D-rules, C is a set of C-rules, AC is a set of AC-rules, and every constant $c \in K$ represents at least one term $t \in \mathcal{T}(\Sigma)$ via R, and

(ii) $AC\backslash R$ is a ground convergent modulo AC over $\mathcal{T}(\Sigma \cup K)$.

In addition, if E is a set of ground equations over $\mathcal{T}(\Sigma)$ such that,

(iii) If s and t are terms over $\mathcal{T}(\Sigma)$, then $s \leftrightarrow^*_{AC\cup E} t$ if, and only if, $s \rightarrow^*_{AC\backslash R} \circ \leftrightarrow^*_{AC} \circ \leftarrow^*_{AC\backslash R} t$,

then R will be called an associative-commutative congruence closure for E.

When Σ_{AC} is empty this definition specializes to that of an abstract congruence closure in Definition 4.

For instance, the rewrite system $D_1 \cup E_1$ above is not a congruence closure for E_0, as it is not a ground convergent rewrite system. But we can transform $D_1 \cup E_1$ into a suitable rewrite system, using a completion-like process described in more detail in the next section, to obtain a congruence closure (Example 13),

$$\begin{aligned} R' \;=\; \{&a \to c_1,\ b \to c_2,\ c \to c_3,\ fc_2c_3 \to c_4,\ fc_3c_4 \to c_2,\ fc_1c_3 \to c_1,\\ &fc_2c_2 \to fc_4c_4,\ fc_1c_2 \to fc_1c_4,\ gc_4 \to c_4\} \end{aligned}$$

that provides a more compact representation of E_0. Attempts to replace every AC-rule by two D-rules (introducing a new constant in the process) leads to non-terminating derivations.

6.3 Construction of Congruence Closures

The completion procedure is obtained by putting together completion procedures over individual signatures $\Sigma_i \cup K$. Clearly, there are two distinct cases that need to be combined: when $f_i \in \Sigma_i$ is AC, and when it is not. The individual completion procedures working on equations in $\Sigma_i \cup K$ need to exchange equations between constants in K that are generated during the completion procedure. For this purpose, we choose an ordering in which the only terms smaller than a constant in K are other constants in K.

Let U be a set of symbols from which new names (constants) are chosen. We need a (partial) AC-compatible reduction ordering which orients the D-rules in the right

way, and orients all the C- and AC-equations. The precedence-based AC-compatible ordering $\succ$ of [72], with any precedence in which $f \succ_{\Sigma\cup U} c$, whenever $f \in \Sigma$ and $c \in U$, serves the purpose. However, much simpler partial orderings would suffice too, but for convenience we use the ordering in [72]. In our case, this simply means that, orientation of D-rules is from left to right. Additionally, the orientation of an AC-rule will be given by: $f(c_1, \ldots, c_i) \succ f(c'_1, \ldots, c'_j)$ iff either $i > j$, or $i = j$ and $\{c_1, \ldots, c_i\} \succ^{mult} \{c'_1, \ldots, c'_j\}$, i.e., if the two terms have the same number of arguments, we compare the multisets of constants using a multiset extension $\succ^{mult}$ of the precedence $\succ_{\Sigma\cup U}$, see [39]. Note that we have defined the ordering on fully flattened terms. Since we work modulo the AC congruence, in the sequel we shall only consider fully flat terms and equations.

We next present a general method for construction of associative-commutative congruence closures. Our description is fairly abstract, in terms of transition rules that operate on triples (K, E, R), where K is a set of new constants that are introduced (the original signature Σ is fixed); E is a set of ground equations (over $\Sigma \cup K$) yet to be processed; and R is a set of C-rules, D-rules and AC-rules. Triples represent possible *states* in the process of constructing a closure. The initial state is $(\emptyset, E, \emptyset)$, where E is the input set of ground equations.

6.3.1 Abstraction

New constants are introduced by the following transition.

Extension:
$$\frac{(K, E[t], R)}{(K \cup \{c\}, E[c], R \cup \{t \to c\})}$$

where $t \to c$ is a D-rule, $c \notin \Sigma \cup K$; and, t occurs in some equation in E that is not an AC-equation.

Once a D-rule $t \to c$ has been introduced by *extension*, it can be used to eliminate any other occurrence of t using the following rule.

Simplification:
$$\frac{(K, E[s], R)}{(K, E[t], R)}$$

where s occurs in some equation in E, and, $s \to_{AC\setminus R^e} t$.

We will also use *flattening* to replace a term in E or R by its corresponding flattened form.

6.3.2 Deduction

It is fairly easy to see that any equation in E can be transformed to a C- or a AC-equation by suitable *extension*, *flattening* and *simplification*. Once we have obtained rules that contain terms only from a specific signature $\Sigma_i \cup K$, we can perform deduction steps on the separate parts.

In the case when $f_i \in \Sigma_i$ is an AC-symbol we use *AC-superposition* wherein we consider overlaps between extensions of AC-rules.

$$\textbf{ACSup}\text{erposition:} \quad \frac{(K, E, R)}{(K, E \cup \{f(s, x\sigma) \approx f(t, y\sigma)\}, R)}$$

if $f \in \Sigma_{AC}$, there exist D- or AC-rules $f(c_1, \ldots, c_k) \to s$ and $f(d_1, \ldots, d_l) \to t$ in R, substitution σ is a most-general unifier modulo AC of $f(c_1, \ldots, c_k, x)$ and $f(d_1, \ldots, d_l, y)$[2] and $\{c_1, \ldots, c_k\} \cap \{d_1, \ldots, d_l\} \neq \emptyset$.

In the special case when one multiset is contained in the other, we obtain the *AC-collapse* rule.

$$\textbf{ACCol}\text{lapse:} \quad \frac{(K, E, R \cup \{t \to s\})}{(K, E \cup \{t' \approx s\}, R)}$$

if for some $u \to v \in R$, $t \to_{AC \setminus \{u \to v\}^e} t'$, and if $t \leftrightarrow^*_{AC} u$ then $s \succ v$.

In case of rules that do not contain AC-symbols at the top, superposition reduces to simplification, and hence *AC-collapse* also captures such superpositions. Note that we do not explicitly add AC extensions of rules to the set R, and so any rule in R is either a C-rule, or a D-rule, or an AC-rule, and *not* its extension. We implicitly work with extensions in *AC-superposition*.

6.3.3 Orientation

Equations are moved from the E-component of the state to the R-component by *orientation*. All rules added to the R-component are either C-rules, D-rules or AC-rules.

$$\textbf{Ori}\text{entation:} \quad \frac{(K, E \cup \{s \approx t\}, R)}{(K, E, R \cup \{s \to t\})}$$

[2]For the special case in hand, there is exactly one most general unifier modulo AC, and it is easy to compute.

if $s \succ t$, and, $s \to t$ is either a D-rule, or a C-rule, or a AC-rule.

Deletion allows us to delete trivial equations.

$$\textbf{Del}\text{etion:} \qquad \frac{(K, E \cup \{s \approx t\}, R)}{(K, E, R)}$$

if $s \leftrightarrow^*_{AC} t$.

6.3.4 Simplification

We need additional transition rules to incorporate interaction between the individual completion processes. In particular, C-rules can be used to perform simplifications on the left- and right-hand sides of other rules. The use of C-rules to simplify left-hand sides of rules is captured by *AC-collapse*. The simplification on the right-hand sides is subsumed by the following generalized *composition* rule.

$$\textbf{Com}\text{position:} \qquad \frac{(K, E, R \cup \{t \to s\})}{(K, E, R \cup \{t \to s'\})}$$

where $s \to_{AC \setminus R^e} s'$.

Example 13 *Let* $E_0 = \{f(a,c) \approx a,\ \ f(c, g(f(b,c))) \approx b,\ \ g(f(b,c)) \approx f(b,c)\}$. *Table 8 shows some intermediate stages of a derivation.*

i	*Constants* K_i	*Equations* E_i	*Rules* R_i	*Transitions*
0	$\emptyset$	E_0	$\emptyset$	
1	$\{c_1, c_3\}$	$\{fcgfbc \approx b,$ $gfbc \approx fbc\}$	$\{a \to c_1, c \to c_3,$ $fc_1c_3 \to c_1\}$	$\mathbf{Ext}^2 \circ \mathbf{Sim} \circ$ $\mathbf{Ori}$
2	$K_1 \cup \{c_2, c_4\}$	$\{fcgfbc \approx b\}$	$R_1 \cup \{b \to c_2,$ $fc_2c_3 \to c_4, gc_4 \to c_4\}$	$\mathbf{Sim}^2 \circ \mathbf{Ext}^2 \circ$ $\mathbf{Sim} \circ \mathbf{Ori}$
3	K_2	$\emptyset$	$R_2 \cup \{fc_3c_4 \to c_2\}$	$\mathbf{Sim}^6 \circ \mathbf{Ori}$
4	K_2	$\emptyset$	$R_3 \cup \{fc_1c_2 \to fc_1c_4\}$	$\mathbf{ACSup} \circ \mathbf{Ori}$
5	K_2	$\emptyset$	$R_4 \cup \{fc_2c_2 \to fc_4c_4\}$	$\mathbf{ACSup} \circ \mathbf{Ori}$

Table 8: Example: Intermediate states in a derivation illustrating congruence closure modulo associativity and commutativity.

The derivation moves equations, one by one, from the E-component of the state to the R-component through simplification, extension *and* orientation. *It can be verified*

that the set R_5 is an AC congruence closure for E_0. Note that the side-condition in extension *disallows breaking of an AC-rule into two D-rules, which is crucial for termination. We assume that f is AC and $c_i \succ c_j$ if $i < j$.*

6.4 Termination and Correctness

Definition 9 *We use the symbol $\vdash$ to denote the one-step transition relation on states induced by the above transition rules. A derivation is a sequence of states $\xi_0 \vdash \xi_1 \vdash \cdots$. A derivation is said to be fair if any transition rule which is continuously enabled is eventually applied. The set R_∞ of persisting rules is defined as $\cup_i \cap_{j>i} R_j$; and similarly, $K_\infty = \cup_i \cap_{j>i} K_j$.*

We shall prove that any fair derivation will only generate finitely many persisting rewrite rules (in the third component) using Dickson's lemma [17]. Multisets over K_∞ can be compared using the multiset inclusion relation. If K_∞ is finite, this relation defines a Dickson partial order.

Lemma 8 *Let E be a finite set of ground equations. The set of persisting rules R_∞ in any fair derivation starting from state $(\emptyset, E, \emptyset)$ is finite.*

Proof. We first claim that K_∞ is finite. To see this, note that new constants are created by *extension*. Using finitely many applications of *extension*, *simplification*, *flattening* and *orientation*, we can move all rules from the initial second component E of the state tuple to the third component R. Fairness ensures that this will eventually happen. Thereafter, any equations ever added to E can be oriented using *flattening* and *orientation*, hence we never apply *extension* subsequently (see the side condition of the *extension* rule). Let $K_\infty = \{c_1, \ldots, c_n\}$.

Next we claim that R_∞ is finite. Suppose, R_∞ contains infinitely many rules. No transition rule (except *extension*) increases the number of D- and C-rules in $E \cup R$. Therefore, R_∞ contains infinitely many AC-rules, and since Σ_{AC} is finite, one AC-operator, say $f \in \Sigma_{AC}$, must occur infinitely often on the left-hand sides of R_∞. By Dickson's lemma, there exists an infinite chain of rules, $f(c_{11}, \ldots, c_{1k_1}) \to s_0, f(c_{21}, \ldots, c_{2k_2}) \to s_1, \ldots$, such that $\{c_{11}, \ldots, c_{1k_1}\} \subseteq \{c_{21}, \ldots, c_{2k_2}\} \subseteq \cdots$, where $\{c_{i1}, \ldots, c_{ik_i}\}$ denotes a multiset and $\subseteq$ denotes multiset inclusion. But, this contradicts fairness (in application of *AC-collapse*). □

6.4.1 Proof ordering

The correctness of the procedure will be established using proof simplification techniques for associative-commutative completion, as described by Bachmair [5] and Bachmair and Dershowitz [6]. In fact, we can directly use the results and the proof measure from [6]. However, since all rules in R have a special form, we can choose a simpler proof ordering. Other differences are (i) we consider AC-symbols to be varyadic, but since the ordering is compatible modulo $AC = P \cup F$, this does not introduce any technical issues (as AC axioms for fixed arity AC-symbols are contained in $P \cup F$); (ii) we do not have explicit transition rules that create extensions of rules in the R-component. Instead we use extensions of rules for simplification and computation of superpositions.

Let $s = s[u\sigma] \leftrightarrow s[v\sigma] = t$ be a proof step using the equation (rule) $u \approx v \in AC \cup E \cup R$. The complexity of this proof step is defined by

$$\begin{array}{rlll} (\{s,t\},\bot,\bot) & \text{if} & u \approx v \in E & \qquad (\{s\},\bot,t) \quad \text{if} \quad u \approx v \in AC \\ (\{s\},u,t) & \text{if} & u \to v \in R & \qquad (\{t\},v,s) \quad \text{if} \quad v \to u \in R \end{array}$$

where $\bot$ is a new symbol. Tuples are compared lexicographically using the multiset extension of the reduction ordering $\succ$ on terms over $\Sigma \cup K_\infty$ in the first component, and the ordering $\succ$ in the second and third component. The constant $\bot$ is assumed to be minimum. The complexity of a proof is the multiset of complexities of its proof steps. The multiset extension of the ordering on tuples yields a proof ordering, denoted by the symbol $\succ_{\mathcal{P}}$. The ordering $\succ_{\mathcal{P}}$ on proofs is well founded as it is a lexicographic combination of well founded orderings.

Lemma 9 *Suppose $(K,E,R) \vdash (K',E',R')$. Then, for any two terms $s,t \in \mathcal{T}(\Sigma)$, it is the case that $s \leftrightarrow^*_{AC\cup E'\cup R'} t$ iff $s \leftrightarrow^*_{AC\cup E\cup R} t$. Further, if π is a ground proof, $s_0 \leftrightarrow s_1 \leftrightarrow \cdots \leftrightarrow s_k$, in $AC \cup E \cup R$, then there is a proof π', $s_0 = s'_0 \leftrightarrow s'_1 \leftrightarrow \cdots \leftrightarrow s'_l = s_k$, in $AC \cup E' \cup R'$ such that $\pi \succeq_{\mathcal{P}} \pi'$.*

Proof. (Sketch) The first part of the lemma, which states that the congruence on $\mathcal{T}(\Sigma)$ remains unchanged, is easily verified by exhaustively checking it for each transition rule. In fact, except for *extension*, all the other transition rules are standard rules for completion modulo a congruence, and hence the result follows. In case $(K,E,R) \vdash (K' = K \cup \{c\}, E', R' = R \cup \{t \to c\})$ by *extension*, then $s \leftrightarrow^*_{AC\cup E\cup R} t$ clearly implies $s \leftrightarrow^*_{AC\cup E'\cup R'} t$. For the other direction, if $s \leftrightarrow^*_{AC\cup E'\cup R'} t$, we replace all occurrences of c in this proof by t to get a proof in $AC \cup E \cup R$.

For the second part, one needs to check that each equation in $(E - E') \cup (R - R')$ has a simpler proof in $E' \cup R' \cup AC$ for each transition rule application, see [6]. In detail, we have the following cases:
(i) *Extension*. The proof $s[t] \leftrightarrow_E u$ is replaced by a proof $s[t] \rightarrow_{R'} s[c] \leftrightarrow_{E'} u$ and the new proof is smaller as $\{s[t], u\} \succ^m \{v\}$ for $v \equiv s[t]$ and $v \equiv s[c]$, and $\{s[t], u\} \succ^m \{s[c], u\}$.
(ii) *Simplification*. The proof $r[s] \leftrightarrow_E u$ is replaced by the new proof $r[s] \leftrightarrow^*_{AC} r' \rightarrow_{R'} r[t] \leftrightarrow_{E'} u$[3]. Now, $\{r[s], u\} \succ^m \{r''\}$ for every r'' in $r[s] \leftrightarrow^*_{AC} r'$, and $\{r[s], u\} \succ^m \{r[t], u\}$.
(iii) *ACCollapse*. The proof $t \rightarrow_R s$ is transformed to the smaller proof $t \leftrightarrow^*_{AC} t' \rightarrow_{\{u \rightarrow v\}} t'' \leftrightarrow_{E'} s$. This new proof is smaller because the rewrite step $t \rightarrow_R s$ is more complex than (a) all proof steps in $t \leftrightarrow^*_{AC} t'$ (in the second component), (b) the proof step $t' \rightarrow_{\{u \rightarrow v\}} t''$ in the second component if $t \not\leftrightarrow^*_{AC} u$, and in the third component if $t \leftrightarrow^*_{AC} u$ (see side condition in *ACCollapse*); and, (c) the proof step $t'' \leftrightarrow_{E'} s$ (in the first component).
(iv) *Orientation*. In this case, $s \leftrightarrow_E t$ is more complex than the new proof $s \rightarrow_{R'} t$ follows from $\{s, t\} \succ^m \{s\}$.
(v) *Deletion*. We have $s \leftrightarrow_E t$ more complex than $s \leftrightarrow^*_{AC} t$ because $\{s, t\} \succ^m \{s'\}$ for every s' in $s \leftrightarrow^*_{AC} t$.
(vi) *Composition*. We have the proof $t \rightarrow_R s$ transformed to the smaller proof $t \rightarrow_R s' \leftarrow_{R'} s'' \leftrightarrow^*_{AC} s$. This new proof is smaller because the rewrite step $t \rightarrow_R s$ is more complex than (a) the rewrite step $t \rightarrow_{R'} s'$ in the third component, (b) all proof steps in $s'' \leftrightarrow^*_{AC} s$ in the first component, and (c) the rewrite step $s'' \rightarrow_{R'} s'$ in the first component.

This completes the proof of the lemma. □

Note that when in state (K, E, R), we have proofs over $AC \cup E \cup R$, extensions of rules are not added explicitly, and hence, they are never deleted either. Once we converge to R_∞, we introduce extensions to take care of cliffs in proofs.

Lemma 10 *If R_∞ is a set of persisting rules of a fair derivation starting from the state $(\emptyset, E, \emptyset)$, then, R^e_∞ is a ground convergent (modulo AC) rewrite system. Furthermore, $E_\infty = \emptyset$.*

Proof. (Sketch) Fairness implies that all critical pairs (modulo AC) between rules

[3]Note that we used extended rule in the specifying *simplification*, but for purposes of proof transformations, we only consider the original (non-extended) rules as being present in the R-component.

in R^e_∞ are contained in the set $\cup_i E_i$. Since a fair derivation is non failing, $E_\infty = \emptyset$. Since the proof ordering is well-founded, for every proof in $E_i \cup R_i \cup AC$, there exists a minimal proof π in $E_\infty \cup R_\infty \cup AC$. We argue by contradiction that certain proof patterns can not occur in the minimal proof π: specifically, there can't be any peaks $s \leftarrow_{R^e_\infty} u \rightarrow_{AC \setminus R^e_\infty}$, non-overlap cliffs or variable overlap cliffs.

(i) *Peaks.* A peak caused by a non-overlap or a variable overlap $s \leftarrow_{R^e_\infty} u \rightarrow_{AC\setminus R^e_\infty} t$ can be transformed to a simpler proof $s \rightarrow^*_{AC\setminus R^e_\infty} v \leftarrow^*_{AC\setminus R^e_\infty} t$. The new proof is simpler because u is bigger than each term in the new proof. Next suppose that the above pattern is caused by a proper overlap. By the Extended Critical Pair Lemma (see [6]), $s \leftrightarrow^*_{AC} s' \leftrightarrow_{CP_{AC}(R^e_\infty)} t' \leftrightarrow^*_{AC} t$, call this proof π. Since by fairness $CP_{AC}(R^e_\infty) \subseteq \cup_k E_k$, there is a proof $s \leftrightarrow^*_{AC} s' \leftrightarrow_{E_k} t' \leftrightarrow^*_{AC} t$ for some $k \geq 0$. Using Lemma 9, we may infer that there is a proof π' in $AC \cup R_\infty$ such that π' is strictly smaller than π, a contradiction.

(ii) *Cliffs.* A non-overlap cliff $w[v,s] \leftrightarrow_{AC} w[u,s] \rightarrow_{AC\setminus R^e_\infty} w[u,t]$ can be transformed to the following less complex proof: $w[v,s] \rightarrow_{AC\setminus R^e_\infty} w[v,t] \leftrightarrow_{AC} w[u,t]$. Clearly, the $w[v,s] \succ w[v,t]$ and hence the proof $w[v,t] \leftrightarrow_{AC} w[u,t]$ is smaller than the proof $w[v,s] \leftrightarrow_{AC} w[u,s]$ (in the first component). The complexity of the proof $w[u,s] \rightarrow_{AC\setminus R^e_\infty} w[u,t]$ is identical to the complexity of the proof $w[v,s] \rightarrow_{AC\setminus R^e_\infty} w[v,t]$.

In case of AC, a variable overlap cliff $s \leftrightarrow_{AC} u \rightarrow_{AC\setminus R^e_\infty} t$ can be eliminated in favour of the proof $s \rightarrow_{AC\setminus R^e_\infty} t' \leftrightarrow_{AC} t$. Note that the proof $u \rightarrow_{AC\setminus R^e_\infty} t$ and the proof $s \rightarrow_{AC\setminus R^e_\infty} t'$ are of the same complexity, and additionally the proof $s \leftrightarrow_{AC} u$ is larger than the proof $t' \leftrightarrow_{AC} t$ as all terms in the latter proof are smaller than u.

In summary, the proof π can not contain peaks $s \leftarrow_{R^e_\infty} u \rightarrow_{AC\setminus R^e_\infty}$, or, non-overlap or variable overlap cliffs $s \leftrightarrow_{AC} u \rightarrow_{AC\setminus R^e_\infty} t$. The cliffs arising from proper overlaps can be replaced by extended rules, as $(R^e_\infty)^e = R^e_\infty$. The minimal proof π in $R_\infty \cup AC$ can, therefore, only be of the form $s \rightarrow^*_{AC\setminus R^e_\infty} s' \leftrightarrow^*_{AC} t' \leftarrow^*_{AC\setminus R^e_\infty} t$; which is a rewrite proof. □

Note that we did not define the proof complexities for the extended rules as the extended rules are introduced only in the end. Hence, the argument given here is not identical to the one in [6], though it is similar. Using Lemmas 9 and 10, we can easily prove that:

Proposition 4 *Let R_∞ be the set of persisting rules of a fair derivation starting from state $(\emptyset, E, \emptyset)$. Let $s, t \in \mathcal{T}(\Sigma)$. Then $s \leftrightarrow^*_{E\cup AC} t$ iff $s \rightarrow^*_{AC\setminus R^e_\infty} \circ \leftrightarrow^*_{AC} \circ \leftarrow^*_{AC\setminus R^e_\infty} t$. Furthermore, the set R^e_∞ is an associative-commutative congruence closure for E.*

Since R_∞ is finite, there exists a k such that $R_\infty \subset R_k$. Though there exist non-terminating fair derivations, we can still identify the finite set of persisting rules if we eagerly apply *AC-collapse* before any application of *AC-superposition* at any point after stage k.

Example 14 *Let us describe one interesting example. Consider the initial set of equations* $E_0 = \{\ f(a,b) \approx b,\ g(b) \approx f(a,a),\ f(b,g(f(a,a,b))) \approx c\}$. *Let* f *be an AC-symbol. We show some of the intermediate states of a fair derivation in Table 9. Note that since we apply* AC-superposition *in step* 3 *before simplifying the last equation in* E*, we avoid introducing a lot of new names.*

i	*Constants* K_i	*Equations* E_i	*Rules* R_i
0	$\emptyset$	E_0	$\emptyset$
1	$\{c_1, c_2\}$	$\{gb \approx faa,\ fbgfaab \approx c\}$	$\{a \to c_1,\ b \to c_2,\ fc_1c_2 \to c_2\}$
2	$\{c_1, c_2, c_3\}$	$\{fbgfaab \approx c\}$	$R_1 \cup \{gc_2 \to c_3,\ fc_1c_1 \to c_3\}$
3	$\{c_1, c_2, c_3\}$	$\{fbgfaab \approx c\}$	$R_2 \cup \{fc_2c_3 \to c_2\}$
4	$\{c_1, c_2, c_3\}$	$\emptyset$	$R_3 \cup \{c \to c_2\}$

Table 9: Example: Intermediate states in a derivation illustrating the congruence closure modulo associativity and commutativity transition rules.

6.5 Optimizations

Certain optimizations can be incorporated into the basic set of transition rules given above for computation of congruence closures. A lot of inferences can be avoided if we note that we do not need to consider extensions of all rules that have an AC symbol on top of their left-hand sides (for the purpose of critical pair computations). For example, we need not consider extensions of those D-rules that are created by *extension* to name a *proper subterm* in E. This fact can be easily incorporated using constraints similar to the constraints used in Chapter 8, but we chose not to show it here for the sake of simplicity and clarity. Rather than formalizing this completely here, we illustrate this with an example below.

Example 15 *Again suppose that the initial set of equations* E_0 *is:* $\{f(a,b) \approx b$,- $g(b) \approx f(a,a),\ f(b,g(f(b,g(f(a,a,b))))) \approx c\}$. *If we choose to first apply* extension

and simplification *until all the initial equations can be oriented using* orientation*, then, we will generate the following set of eleven D-rules:*

$$\begin{array}{rcl@{\qquad}rcl@{\qquad}rcl@{\qquad}rcl} a & \to & c_1 & b & \to & c_2 & fc_1c_2 & \to & c_2 & fc_1c_1 & \to & c_3 \\ gc_2 & \to & c_3 & fc_3c_2 & \to & c_4 & gc_4 & \to & c_5 & fc_2c_5 & \to & c_6 \\ gc_6 & \to & c_7 & fc_2c_7 & \to & c_8 & c & \to & c_8 & & & \end{array}$$

Now, note the huge number of possible AC-superpositions*. However, we need not consider all of them. In particular, we need to consider extensions only of three rules:* $fc_1c_2 \to c_2$*,* $fc_1c_1 \to c_3$*, and* $fc_2c_7 \to c_8$*, and hence,* AC-Superpositions *between only these.*

Secondly, note that, as in the case of congruence closure discussed in Chapter 4 we can choose the ordering between two constants in K on-the-fly. As an optimization we could always choose it in a way so as to minimize the applications of *AC-collapse* and *composition* later. In other words, when we need to choose the orientation for $c \approx d$, we can look at the number of occurrences of c and d in the set of D- and AC-rules (in the R-component of the state), and the constant with fewer occurrences is made larger.

6.6 Construction of Ground Convergent Systems

We next discuss the problem of obtaining a ground convergent AC rewrite system for the given ground AC-theory *in the original signature*. Hence, now we focus our attention to the problem of transforming a convergent system over an extended signature to a convergent system in the original signature. If we can successfully achieve this goal, it would mean that we can do ground AC-completion *without* having an AC-compatible ordering total on ground terms.

The basic idea of transforming back is elimination of constants from the presentation R as follows: (i) if a constant c is not redundant, then we pick a term $t \in \mathcal{T}(\Sigma)$ that is represented by c, and replace all occurrences of c by t in R; (ii) if a constant c is redundant (and say $c \to d$ is a C-rule in which c occurs as the left-hand side term), then c can be replaced by d in R.

In the case when there are no AC-symbols in the signature, the above method constructs a ground convergent system from any given abstract congruence closure. This gives an indirect way to construct ground convergent systems equivalent to a given set of ground equations. However, we run into problems when we use the same

method for translation in presence of AC-symbols. Typically, after translating back, the set of rules one gets is usually non-terminating modulo AC (Example 16). But if we suitably define the notion of AC-rewriting, the rules are seen to be convergent in the new definition. This is useful in two ways: (i) the new notion of AC-rewriting seems to be more practical, in the sense that it involves strictly less work than a usual $R^e \backslash AC$ reduction; and, (ii) it helps to clarify the advantage offered by the use of extended signatures when dealing with a set of ground AC-equations.

6.6.1 Transition Rules

We describe the process of transforming a rewrite system R over an extended signature $\Sigma \cup K$ to a rewrite system R over the original signature Σ by transformation rules on states (K, R), where K is the set of constants to be eliminated, and R is a set of rewrite rules over $\Sigma \cup K$ to be transformed.

Redundant constants can be easily eliminated by the *compression* rule.

$$\textbf{Comp}\text{ression:} \quad \frac{(K \cup \{c\}, R \cup \{c \to t\})}{(K, R[c \mapsto t])}$$

The basic idea for eliminating a constant c that is not redundant in R involves picking a representative term t (over the signature Σ) in the equivalence class of c, and replacing c by t everywhere in R.

$$\textbf{Sel}\text{ection:} \quad \frac{(K \cup \{c\}, R \cup \{t \to c\})}{(K, R[c \mapsto t] \cup R')}$$

if (i) c is not redundant in R, (ii) $t \in \mathcal{T}(\Sigma)$, and (iii) if $t \equiv f(t_1, \ldots, t_k)$ with $f \in \Sigma_{AC}$ then $R' = \{f(t_1, \ldots, t_k, X) \to f(f(t_1, \ldots, t_k), X)\}$, otherwise $R' = \emptyset$.

In case $\Sigma_{AC} = \emptyset$, we note that R' will always be empty. We also require that terms are *not* flattened after application of substitution in $R[c \mapsto t]$. The variable X is a special variable and its role will be be discussed later.

Example 16 *Consider the problem of constructing a ground convergent system for the set E_0 of Example 13. A fully-reduced congruence closure for E_0 is given by the set R_0*

$$\begin{array}{rclrclrclrcl}
a & \to & c_1 & b & \to & c_2 & c & \to & c_3 & fc_2c_3 & \to & c_4 \\
fc_3c_4 & \to & c_2 & fc_1c_3 & \to & c_1 & fc_2c_2 & \to & fc_4c_4 & fc_1c_2 & \to & fc_1c_4 \\
gc_4 & \to & c_4 & & & & & & & & &
\end{array}$$

under the ordering $c_2 \succ c_4$ between constants. For the constants c_1, c_2 and c_3 we have no choice but to choose a, b and c as representatives respectively. Thus after three applications of selection*, we get*

$$\begin{array}{llllllllll} fcc_4 & \to & b & fac & \to & a & fbb & \to & fc_4c_4 & fab & \to & fac_4 \\ fbc & \to & c_4 & gc_4 & \to & c_4 \end{array}$$

Next we are forced to choose fbc as the representative for the class c_4. This gives us the transformed set R_1,

$$\begin{array}{llllllllll} fc(fbc) & \to & b & fac & \to & a & fbb & \to & f(fbc)(fbc) \\ fab & \to & fa(fbc) & fbcX & \to & f(fbc)X & gfbc & \to & fbc \end{array}$$

The relation $\to_{AC\setminus R_1^e}$ is clearly non-terminating (with the variable X considered as a regular term variable).

6.6.2 Rewriting with sequence extensions modulo permutation congruence

Let X denote a variable ranging over sequences of terms. A sequence substitution σ is a substitution that maps variables to the sequences. If σ is a sequence substitution that maps X to the sequence $\langle s'_1, \ldots, s'_m \rangle$, then $f(s_1, \ldots, s_k, X)\sigma$ is the term $f(t_1, \ldots, s_k, s'_1, \ldots, s'_m)$.

Definition 10 *Let ρ be a ground rule of the form $f(t_1, \ldots, t_k) \to g(s_1, \ldots, s_m)$ where $f \in \Sigma_{AC}$. We define the* sequence extension ρ^s *of ρ as, $f(t_1, \ldots, t_k, X) \to f(s_1, \ldots, s_m, X)$ if $f = g$ and, $f(t_1, \ldots, t_k, X) \to f(g(s_1, \ldots, s_m), X)$ if $f \neq g$.*

Now we are ready to define the notion of rewriting we use. Recall that P denotes the equations defining the permutation congruence, and that $AC = F \cup P$.

Definition 11 *Let R be a set of ground rules. For ground terms $s, t \in \mathcal{T}(\Sigma)$, we say that $s \to_{P\setminus R^s} t$ if there exists a rule $l \to r \in R^s$ and a sequence substitution σ such that $s = C[l']$, $l' \leftrightarrow_P^* l\sigma$ and $r' = r\sigma$.*

Note that the difference with standard rewriting modulo AC is that instead of performing matching modulo AC, we do matching modulo P. For example, if ρ is $fac \to a$, then the term $f(f(a, b), c)$ is not reducible by $\to_{P\setminus\rho^s}$, although it is reducible by $\to_{AC\setminus\rho^e}$. The term $f(f(a, b), c, a)$ can be rewritten by $\to_{P\setminus\rho^s}$ to $f(f(a, b), a)$.

Example 17 *Following up on Example 16, we note that the relation $P\backslash R_1^s$ is convergent. For instance, a normalizing rewrite derivation for the term $fabc$ is,*

$$fabc \rightarrow_{P\backslash R_1^s} fa(fbc)c \rightarrow_{P\backslash R_1^s} fab \rightarrow_{P\backslash R_1^s} fa(fbc).$$

On closer inspection, we find that we are essentially doing a derivation in the original rewrite system R_0 (over the extended signature).

$$fabc \rightarrow_{P\backslash R_0^s} fc_1c_2c_3 \rightarrow_{P\backslash R_0^s} fc_1c_4c_3 \rightarrow_{P\backslash R_0^s} fc_1c_2 \rightarrow_{P\backslash R_0^s} fc_1c_4.$$

There is a one-to-one bijection between a step using $P\backslash R_1^s$ and a step using $P\backslash R_0^s$. This essentially is at the core of the proof of correctness.

6.6.3 Correctness

We note that any derivation starting in the state (K, R), where R is an AC-congruence closure over Σ and K, is finite. This is because K is finite, and each application of *compression* and *selection* reduces the cardinality of K by one. Furthermore, in any intermediate state (K, R), R is always a rewrite system over $\Sigma \cup K$. Hence, in the final state (K_∞, R_∞), if $K_\infty = \emptyset$, then, R_∞ is a rewrite system over Σ, the original signature. We will show that K_∞ is actually empty, and that the reduction relation $P\backslash R_\infty^s$ is terminating on $\mathcal{T}(\Sigma)$ and confluent on fully flattened terms in $\mathcal{T}(\Sigma)$.

Lemma 11 *Suppose (K_1, R_1) is obtained from (K_0, R_0) via* selection *or* compression*. If R_0 is left-reduced and terminating (with respect to $P\backslash R_0^s$-reduction) then R_1 is left-reduced and terminating (with respect to $P\backslash R_1^s$-reduction). Additionally, for all flat terms $s, t \in \mathcal{T}(\Sigma)$, $s \leftrightarrow^*_{P\backslash R_0^s} t$, if and only if, $s \leftrightarrow^*_{P\backslash R_1^s} t$.*

Proof. First consider the case of *compression*. Say $(K_0 = K_1 \cup \{c\}, R_0 = R_0' \cup \{c \rightarrow t\}) \vdash (K_1, R_1 = R_0'[c \mapsto t])$. This step has the same effect as a sequence of composition steps followed by a deletion step. Hence, it preserves termination. The left-hand side terms do not change, and hence the system continues to remain left-reduced. Note that R_0 and $R_1 \cup \{c \approx t\}$ define the same equational theory. But since c does not occur in R_1, the equational theories induced by $R_0^s \cup P$ and by $R_1^s \cup P$ over $\mathcal{T}(\Sigma)$ are identical.

The second case is that of *selection*. Say $(K_0 = K_1 \cup \{c\}, R_0 = R_0' \cup \{t \rightarrow c\}) \vdash (K_1, R_1 = R_0'[c \mapsto t] \cup R_0')$. Let σ denote the substitution $\langle t \mapsto c \rangle$ and σ^{-1} denote the substitution $\langle c \mapsto t \rangle$. The crucial observation is that if $l_1 \rightarrow r_1$ is a rule in R_1

obtained from $l_0 \to r_0$ in R_0, then, for *any* context $C[\_]$ and *any* sequence substitution σ^s, we have, $(C[l_1\sigma^s])\sigma = (C\sigma)[l_1\sigma(\sigma^s\sigma)] = (C\sigma)[l_0(\sigma^s\sigma)]$. If R_1 is not left-reduced, then, by application of σ to the left-hand sides of the overlapping rules we obtain two rules in R_0 that overlap.

Additionally, if $C[l_1\sigma^s] \to_{P\backslash R_1^s} C[r_1\sigma^s]$, then,

$$(C[l_1\sigma^s])\sigma = (C\sigma)[l_0(\sigma^s\sigma)] \to_{P\backslash R_0^s} (C\sigma)[r_0(\sigma^s\sigma)] \to^*_{P\backslash R_0^s} (C[r_1\sigma^s])\sigma.$$

Hence, every sequence of n rewrite steps in $P\backslash R_1^s$ can be projected to n or more steps in $P\backslash R_0^s$ (by applying substitution σ to each term in the proof). Hence, if $P\backslash R_0^s$ is terminating, so is $P\backslash R_1^s$.

Finally, since the constant c does not occur in R_1 any more, it is easily established that for all flat terms $s, t \in \mathcal{T}(\Sigma)$, $s \leftrightarrow^*_{P\backslash R_0^s} t$, if and only if, $s \leftrightarrow^*_{P\backslash R_1^s} t$. □

The second step in the correctness argument involves showing that if $K_i \neq \phi$, then we can always apply either *selection* or *compression* to get to a new state.

Lemma 12 *Let (K_i, R_i) be a state in the derivation starting from (K_0, R_0), where R_0 is a left-reduced associative-commutative congruence closure over the signature $\Sigma \cup K_0$. If $K_i \neq \phi$, then either* selection *or* compression *is applicable to the state (K_i, R_i).*

Proof. Since R_0 is an AC-congruence closure, every $c \in K_0$ represents some term $t \in \mathcal{T}(\Sigma)$ via R_0, i.e.,

$$t \ \leftrightarrow^*_{AC\backslash R_0^e} \ c.$$

By convergence (see definition 8 condition (ii) and (iii)),

$$t \ \to^*_{AC\backslash R_0^e} \ c' \ \leftarrow^*_{AC\backslash R_0^e} \ c.$$

Without loss of generality we can assume that t is a fully flattened term, and hence we also have

$$t \ \to^*_{P\backslash R_0^s} \ c' \ \leftarrow^*_{P\backslash R_0^s} \ c.$$

Using lemma 11 we have,

$$t \ \to^*_{P\backslash R_i^s} \ c' \ \leftarrow^*_{P\backslash R_i^s} \ c.$$

If $c \neq c'$ then there exists at least one redundant constant which can be eliminated by *compression*. If there are no redundant constants, then $c = c'$, and hence, if $l \to d \in R_i^s$ is the rule used in the first step of this proof, then we can choose l as the representative for d and hence *selection* is applicable. □

Theorem 5 *If (K_∞, R_∞) is the final state of a maximal derivation starting from state (K, R), where R is a left-reduced AC-congruence closure, then (i) $K_\infty = \emptyset$, (ii) $\rightarrow_{P \setminus R^s_\infty}$ is ground convergent on all fully flattened terms over Σ, and (iii) the equivalence over flattened $\mathcal{T}(\Sigma)$ terms defined by this relation is the same as the equational theory induced by $R \cup AC$ over flattened $\mathcal{T}(\Sigma)$ terms.*

Proof. Statement (i) is a consequence of Lemma 12. The other two statements are a consequence of Lemma 11. □

Note that in the special case when Σ_{AC} is empty, the notion of rewriting corresponds to the standard notion, and hence R_∞ is convergent in the standard sense by this theorem.

6.7 Conclusion

Graph-based congruence closure algorithms have also been used to construct a convergent set of ground rewrite rules in polynomial time by Snyder [76]. Plaisted et. al. [68] gave a *direct* method, not based on using congruence closure, for completing a ground rewrite system in polynomial time. Hence our work completes the missing link, by showing that congruence closure is nothing but ground completion. Our approach is different from that of Snyder, and can be used to obtain a *more efficient* implementation partly because Snyder's algorithm needs *two* passes of the congruence closure algorithm, whereas we would need to compute the abstract congruence closure just once.

Snyder works with a *particular implementation* of congruence closure which forces Snyder to do some processing on the computed congruence closure followed by a run of congruence closure *once again*. We, on the other hand, work with abstract congruence closure and are free to choose any implementation. All the steps of Snyder's algorithm can be described using our *construction of abstract congruence closure* steps, and the final output of Snyder's algorithm corresponds to an abstract congruence closure. The rules for *translating back* in our work, actually correspond to what Snyder calls *printing-out the reduced system* and this is not included in the algorithms time complexity of $O(n \log(n))$ as computed by Snyder. But, as a simple example in Snyder's paper shows, we cannot hope to do better than quadratic time if we explicitly represent the reduced system.

These facts and differences are also reflected in the observation that Snyder's

algorithm for constructing a ground convergent system cannot use other better implementations of congruence closure, for example, Shostak's *dynamic* congruence closure. Clearly, we can use any implementation. In effect, by using a strategy that does early simplification, we can remove redundancies from the equational system early on. This is the source of efficiency in Shostak's approach and the same can be used here. Another difference from Snyder's work is that Snyder solves the problem "by abandoning rewriting techniques altogether and recasting problem in graph theoretic terms." On the other hand, we stick to rewriting over extensions.

Plaisted and Sattler-Klein [68] show that ground term-rewriting systems can be completed in a polynomial number of rewriting steps by using an appropriate data structure for terms and processing the rules in a certain way. Our work describes the construction of ground convergent systems using congruence closure as completion with *extensions*, followed by a translating back phase. Plaisted and Sattler-Klein prove a quadratic time complexity of their completion procedure. However, by mimicking the Downey, Sethi and Tarjan's algorithm for example, we can obtain an $O(n \log(n))$ time complexity for computing the abstract congruence closure. However, the translating back phase takes time proportional to the size of the reduced system, which could be exponential in the worst case.

The fact that one can use certain rules backwards (similar to what we did with $t \rightarrow c$ rules in *selection*) and obtain a convergent system after performing some simplification steps is mentioned in Snyder [76]. But, rather than using this to obtain a ground convergent system in the original signature, Snyder mentions this to show how one can go from one ground convergent system over the original signature to another one (for the same equational theory, but in a different ordering).

The fact that we can construct an AC-congruence closure implies that the word problem for finitely presented ground AC-theories is decidable, see [59], [62] and [40]. Note that we arrive at this result *without* assuming the existence of an AC-simplification ordering that is total on ground terms. The existence of such AC-simplification orderings was established in [62], but required a non-trivial proof.

Since we construct a convergent rewrite system, even the problem of determining whether two finitely presented ground AC-theories are equivalent, is decidable. Since commutative semigroups are special kinds of AC-theories, where the signature consists of a single AC-symbol and a finite set of constants, these results carry over to this special case [60, 56][4].

[4]The isomorphism problem for commutative semigroups is mentioned as an open problem in [44].

We have shown that we can construct an AC-congruence closure for a finite set E of ground equations (over a signature Σ containing AC-symbols) using procedures that construct such closures for finitely presented commutative semigroups. In fact the number of invocations of the latter procedure is bounded above by $n|\Sigma|$, where n denotes the size of set E, and $|\Sigma|$ denotes the cardinality of the set Σ. This establishes that the word problem for ground AC-theories is no more difficult (modulo a polynomial time factor) than the problem of constructing a convergent system for finitely presented commutative semigroups.

The idea of using variable abstraction to transform a set equations over several AC-symbols into a set of equations in which each equation contains exactly one AC-symbol appears in [40]. All equations containing the same AC-symbol are separated out, and completed into a canonical rewriting system (modulo AC) using the method proposed in [15]. However, the combination of ground AC-theories with other ground theories is done differently here. In [40], the ground theory (non-AC part) is handled using ground completion (and uses a recursive path ordering during completion). We, on the other hand, use a congruence closure. The usefulness of our approach can also be seen from the simplicity of the correctness proof and the results we obtain for transforming a convergent system over an extended signature to one over the original signature.

The method for completing a finitely presented commutative semigroup (using what we call AC-rules here) has been described in various forms in the literature, e.g. [15])[5]. It is essentially a specialization of Buchberger's algorithm for polynomial ideals to the case of binomial ideals (i.e. when the ideal is defined by polynomials consisting of exactly two monomials with coefficients $+1$ and -1).

The basic idea behind our construction of associative-commutative congruence closure is that we consider only certain ground instantiations of the non-ground AC axioms. If we are interested in the $\mathcal{E}$-algebra presented by E (where $\mathcal{E}$ consists of only AC axioms for some function symbols in the signature Σ in our case, and E is a set of ground equations), then since $\mathcal{E}$ consists of non-ground axioms, one needs to worry about what instantiations of these axioms to consider. For the case when $\mathcal{E}$ is a set of AC axioms, we show that we need to consider ground instances in which every variable is replaced by some subterm occurring in E. This observation can

[5]Actually there is a subtle difference between the proposed method here and the various other algorithms for deciding the word problem for commutative semigroups too. For example, working with rule extensions is not the same as working with rules on equivalence classes (under AC) of terms. Hence, in our method, we can apply certain optimizations as mentioned in Section 6.5.

be generalized and one can ask for what choices of $\mathcal{E}$ axioms does considering such restricted instantiations suffice to decide the word problem in $\mathcal{E}$-algebras? Evans [42, 44] gives a characterization in terms of *embeddability of partial $\mathcal{E}$-algebras*. Apart from commutative semigroups, this method works for lattices, groupoids, quasigroups, loops, etc.

Chapter 7

Applications of Congruence Closure

In this chapter, we illustrate some applications of congruence closure. We show that the notion of an abstract congruence closure helps in discerning the logical aspects from the other implementation details, which helps us to suitably generalize some of the known results.

The first part of this chapter concerns itself with the problem of normalizing a term with respect to a given rewrite system. The essential idea is that congruence closure allows a compact way to store normal forms of some terms (w.r.t the given rewrite system), so that one can look into the saved information to compute the normal form of a new term. The method uses the concept of a rewrite closure, which is a generalization of the idea of a congruence closure. Our results generalize previous results on congruence closure-based normalization methods. The description of known methods within our formalism also allows a better understanding of these procedures.

The second application addresses the problem of rigid E-unification. We provide a sound and complete set of abstract transformation rules for rigid E-unification. Abstract congruence closure forms one of the transition rules.

As a final use of congruence closure, we discuss Shoatak's combination algorithm which attempts to combine congruence closure with E-unification procedures. In all the three problems discussed, congruence closure plays a crucial part.

7.1 Normalization via Rewrite Closures

Efficient procedures for normalization of expressions are crucial for the practical performance of rewrite-based systems. A straightforward approach to normalization, by a "straight-line" sequence of individual reduction steps, each consisting of matching and subterm replacement, may be expensive if it requires many steps. For instance, suppose a term $f(a)$ can be rewritten to a normal form a in n reduction steps. Then the normalization of the term $f(f(a))$ may require twice as many steps, as $f(f(a))$ is first rewritten to $f(a)$ (in n steps) and then to a (in n more steps). Note that the two subsequences consist essentially of the same reduction steps, though applied to different terms. There have been attempts to store the "history" of reductions in a suitable way so as to avoid repetition of such "equivalent" reduction sequences. The aim is to generate, once the subterm in $f(f(a))$ has been reduced to a, the normal form a from the intermediate term $f(a)$ in a *single* additional step. Such a "non-oblivious" normalization method requires $n+1$ steps in this case, though the individual steps are usually different from standard matching and subterm replacement. The key question is how to store the application of rewrite rules in a way that can be efficiently exploited for performing future reductions.

Chew [29, 30] had the fundamental insight to adapt techniques developed for congruence closure algorithms (cf., Nelson and Oppen [64]) to normalization with history. Congruence closure algorithms yield a compact representation of the equational theory induced by a finite sets of variable-free equations. Normalization usually needs to be done for rewrite systems (i.e., sets of directed equations) with variables, which may represent infinitely many variable-free equations over the given term domain. Chew's method therefore combines a "dynamic" version of congruence closure with a method for selecting rewrite rule instances needed to normalize a given input term. Whenever additional rule instances are selected, the congruence closure algorithm is applied incrementally to update the representation of term equivalences. Once a term has been rewritten, the congruence closure represents it by its current normal form and thus effectively stores the history of previous reductions. If no further useful rule instances can be selected, one either obtains a normal form for the input term or else detects non-termination of the rewriting process. (Non-terminating rewrite systems may also cause the selection process to continue indefinitely.)

Chew's work deals with normalization by orthogonal rewrite systems, but was extended by Verma [81] to normalization by priority rewrite systems. The description

of these methods is technically involved. We develop a different, more abstract view of this approach to normalization by formulating it in terms of standard techniques from term rewriting, such as completion and narrowing. More specifically, the basic method, and various optimizations, are described in terms of transformation rules in the style of Bachmair and Dershowitz [7].

We explain application of abstract congruence closure to normalization in Section 7.1.2. Optimizations of the basic method by using a modified congruence closure, called a *rewrite closure*, are discussed in Section 7.1.3. Finally we outline further optimizations for the special cases of orthogonal and convergent rewrite systems in Section 7.1.4.

7.1.1 Incremental Congruence Closure

For purposes of normalization, we use an incremental congruence closure, i.e., given a $(K_i, \emptyset, R_i)$ congruence closure final state, we add ground equations E_i over $\Sigma \cup K_i$ to obtain the new state (K_i, E_i, R_i). We start a congruence closure derivation again beginning from this state to finally get $(K_{i+1}, \emptyset, R_{i+1})$. The *R-extension* of E is defined to be the set $E{\uparrow}_R^\rho$ of all equations $s\rho \to t\rho$, where $s \approx t$ is an equation in E and ρ is a mapping from K to $\mathcal{T}(\Sigma)$, such that $c \leftrightarrow_R^* c\rho$, for all constants c in K. We denote by $E{\uparrow}_R$ the set $\{l' \approx r' : l \in \mathcal{T}(\Sigma), r \in \mathcal{T}(\Sigma), l \approx r \in E, l' \leftrightarrow_R^* l, r' \leftrightarrow_R^* r\}$. Note that $E{\uparrow}_R^\rho \subseteq E{\uparrow}_R$.

Corollary 1 *Let $(K_i, E_i, R_i) \vdash_{CC}^* (K_{i+1}, \emptyset, R_{i+1})$ for $i = 0, 1, 2, \ldots$, where the states $(K_i, \emptyset, R_i)$ are all congruence closure final states, $K_0 = \emptyset$, $R_0 = \emptyset$, and, E_i are equations over $\Sigma \cup K_i$. Define $F_j = \cup_{i=0}^{j-1} E_i$. Then, for any $s, t \in \mathcal{T}(\Sigma)$,*

$$s \leftrightarrow_{R_j}^* t \quad \textit{iff} \quad s \leftrightarrow_{F_j{\uparrow}_{R_j}^\rho}^* t \quad \textit{iff} \quad s \leftrightarrow_{F_j{\uparrow}_{R_j}}^* t.$$

Proof. We prove by induction on j that

$$\leftrightarrow_{R_j}^* \subseteq \leftrightarrow_{F_j{\uparrow}_{R_j}^\rho}^* \subseteq \leftrightarrow_{F_j{\uparrow}_{R_j}}^* \subseteq \leftrightarrow_{R_j}^* .$$

The base case $j = 1$ is established by theorem 3 and by noting that $E_0{\uparrow}_{R_1} = E_0{\uparrow}_{R_1}^\rho = E_0$.

Let $s, t \in \mathcal{T}(\Sigma)$. Suppose $s \leftrightarrow_{R_j}^* t$. Then, using Theorem 2, $s \leftrightarrow_{R_{j-1} \cup E_{j-1}}^* t$. Let ρ be an arbitrary mapping from K_{j-1} to $\mathcal{T}(\Sigma)$ such that $c \leftrightarrow_{R_{j-1}}^* c\rho$. Since R_{j-1} is convergent, we obtain the proof diagram of Figure 3. By induction hypotheses,

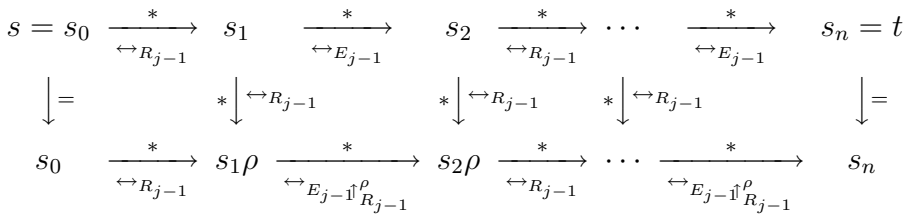


Figure 3: Correctness of Incremental Congruence Closure.

$\leftrightarrow^*_{R_{j-1}}$ is equivalent to $\leftrightarrow^*_{F_{j-1}\Uparrow^{\rho}_{R_{j-1}}}$. This shows that $s \leftrightarrow^*_{F_j\Uparrow^{\rho}_{R_j}} t$.

The second inclusion is trivial. For the final inclusion, without loss in generality, consider the one step derivation $s \leftrightarrow_{F_j\Uparrow_{R_j}} t$. Therefore $s = C[l']$ and $t = C[r']$, and hence,

$$s = C[l'] \leftrightarrow^*_{R_j} C[l] \leftrightarrow_{F_j} C[r] \leftrightarrow^*_{R_j} C[r'] = t.$$

Using Theorem 2, $C[l] \leftrightarrow^*_{R_j} C[r]$, and this completes the proof. □

In the context of this incremental approach, we generalize the definition of an abstract congruence closure by replacing the condition (iii) by the (stronger) one mentioned in this corollary.

7.1.2 Normalization Using Congruence Closure

We next outline how to apply congruence closures to the problem of finding, given a rewrite system $\mathcal{E}$ and a ground term t, a normal form of t with respect to $\mathcal{E}$.[1] Let us first consider the simple case when $\mathcal{E}$ is a ground rewrite system.

For example, suppose we want to normalize the term f^5a with respect to the rewrite system $\mathcal{E}_0 = \{fa \to fb, ffb \to a, fb \to a\}$. The set

$$R_4 = \{a \to c_0,\ b \to c_1,\ fc_0 \to c_0,\ fc_1 \to c_0\}$$

is an abstract congruence closure for $\mathcal{E}_0$. First we compute the normal form of f^5a by R_4, which is c_0. Then we identify an (irreducible) term over the signature Σ that is represented by c_0. The definition of a congruence closure guarantees that such a term exists, in this example, we get a. Thus, we have $f^5a \to^*_{R_4} c_0 \leftarrow^*_{R_0} a$ and conclude that a is a normal form of f^5a.

[1] It is sufficient to consider *ground* terms t, The normal form of a general term u can be easily obtained from a normal form of the ground term $\hat{u}$, where $\hat{u}$ is obtained from u by replacing each variable by a new constant.

This approach is simple, but needs to be generalized to rewrite systems with variables. The basic idea is to select one or more instances $l\sigma \to r\sigma$ of rules in $\mathcal{E}$ that can be used to reduce (a subterm of) t and to apply congruence closure to them, so that a normal form t' of t (with respect to the selected instances) can be identified. If t' is also in normal form with respect to $\mathcal{E}$, we are done; otherwise, further rule instances need to be selected to reduce t'. This yields a method that incrementally applies congruence closure to selected instances of given rewrite rules. Two key issues to be addressed are selection—how to select instances, and termination—how to efficiently identify that a term is in $\mathcal{E}$-normal form. Selection, it turns out, can be done by a simple narrowing process.

7.1.2.1 Narrowing

We say that a term t *narrows* to a term t' (with respect to a rewrite system R) if there exist a non-variable subterm s of t and a rewrite rule $l \to r \in R$, such that (i) s is unifiable with l and (ii) t' is obtained from $t\sigma$ by replacing the subterm $s\sigma$ by $r\sigma$, where σ is a most general unifier of s and l.

In our context, the term t to be normalized, and all its subterms, are represented by constants via a congruence closure R. We will present a simple narrowing procedure to determine whether some left-hand side l of a rule in $\mathcal{E}$ can be narrowed to a constant via R, i.e., whether there exists a substitution σ, such that $l\sigma \to_R^* c$. Any rule instance $l\sigma \to r\sigma$ selected by narrowing in this way will then be used for incremental extension of the congruence closure R.

We use transformation rules to describe the narrowing process for selection of rule instances from a rewrite system $\mathcal{E} = \{l_1 \to r_1, \ldots, l_n \to r_n\}$. The transformation rules operate on states (K, R, S), where K is a set of constants disjoint from Σ, R is a congruence closure, and S is sequence of n sets $\langle L_1, \ldots, L_n \rangle$. Each set L_i consists of pairs (l_i', σ_i), where $l_i\sigma_i \to_R^* l_i'$. The pairs (l_i', σ_i) indicate candidates to be selected among rule instances. Selection of $l_i\sigma \to r_i\sigma$ is possible if a term l_i' is a constant. If a term l_i' is not a constant, but can not be reduced further, then the corresponding candidate pair can be deleted, as selection will be impossible.

The formal transformation rule is as follows.

$$\textbf{Narrowing:} \quad \frac{(K, R, \langle \ldots, L_j \cup \{(s[t], \sigma)\}, \ldots \rangle)}{(K, R, \langle \ldots, L_j \cup L_j', \ldots \rangle)}$$

where t is either a constant or a non-variable non-constant subterm of s and either (i)

t can be narrowed by R, in which case L'_j is the set of all pairs $(s[c]\sigma_1, \sigma\sigma_1)$ such that $t\sigma_1 = u$ for some rule $u \to c$ in R, or (ii) t is not a constant and cannot be narrowed, in which case $L'_j = \emptyset$.

We write $\eta \vdash_S \eta'$ to indicate that a state η can be obtained from η' by application of this narrowing rule.

Lemma 13 *Any derivation using the relation $\vdash_S$ and starting from state (K, R, S) is finite whenever K, R and each component L_i of S is finite.*

Proof. We first extend the ordering on ground terms used in the proof of Lemma 5 to an ordering on terms containing variables by treating all variables as equivalent and being larger than every constant in K, but smaller than every symbol in Σ. Next, we define the complexity of a state $(K, R, \langle L_1, \ldots, L_n\rangle)$ by the multiset $\{l : (l, \sigma) \in L_i \text{ for some } i\}$. Two such multisets are compared using the multiset extension of the ordering in which the set R is reducing. The narrowing transformation rule is reducing with respect to this well-founded ordering. □

The *final state* of such a derivation is a triple $(K, R, \langle L_1, \ldots, L_n\rangle)$ where each set L_i contains only pairs $(c, \sigma), c \in K$.

7.1.2.2 Normalization

We now have the two main components of a non-oblivious normalization method–the congruence closure transformation relation $\vdash_{CC}$ and the narrowing transformation relation $\vdash_S$. We describe normalization by rules operating on tuples $(\mathcal{E}, t, K, E, R, S)$, where $\mathcal{E}$ is a rewrite system, t is the term to be normalized, (K, E, R) represents the current state of a congruence closure computation, and S indicates current candidates for selection. We define:

$$(\mathcal{E}, t, K, E, R, \Lambda) \vdash_N (\mathcal{E}, t, K', E', R', \Lambda)$$

if $(K, E, R) \vdash_{CC} (K', E', R')$, where Λ is a special sequence different from all other; and

$$(\mathcal{E}, t, K, \emptyset, R, S) \vdash_N (\mathcal{E}, t, K, \emptyset, R, S')$$

if $(K, R, S) \vdash_S (K, R, S')$ and $(K, \emptyset, R)$ is a congruence closure final state.

In short, we distinguish between two phases during normalization: congruence closure rules are applied when no candidates for selection are available, whereas narrowing is only performed in the presence of a (completed) congruence closure.

An *initial state* for a normalization derivation is a tuple $(\mathcal{E}, t, \{c\}, \{c \approx t\}, \emptyset, \Lambda)$, where c is a constant not contained in Σ. The first stage will consist of a congruence closure computation, which has the effect of representing all subterms of the term t to be normalized. The following transformation rules are used to connect congruence closure and narrowing stages and to determine when the normalization process is done.

Narrowing is initiated as follows.

$$\textbf{Init}\text{ialization:} \qquad \frac{(\mathcal{E}, t, K, \emptyset, R, \Lambda)}{(\mathcal{E}, t, K, \emptyset, R, \langle\{(l_1, \mathrm{id})\}, \ldots, \{(l_n, \mathrm{id})\}\rangle)}$$

if the state $(K, \emptyset, R)$ is a congruence closure final state. (The symbol id denotes the identity mapping.)

If the narrowing phase is successful, further rule instances can be selected.

$$\textbf{Sel}\text{ection:} \qquad \frac{(\mathcal{E}, t, K, \emptyset, R, \langle L_1, \ldots, L_j \cup \{(c, \sigma)\}, \ldots, L_n\rangle))}{(\mathcal{E}, t, K, \{l_j\sigma \to r_j\sigma\}, R, \Lambda)}$$

if c is a constant in K.[2]

The rule $l_j\sigma\sigma' \to r_j\sigma\sigma'$ which is moved to the set E in this rule, will be called a *selected*, or, *processed* rule. In general, we can move more than one rule instance to the E component without affecting any of the subsequent results.

Computation of a congruence closure may change the representation of equivalences. Instead of initiating another narrowing phase, we may check whether a normal form term is already represented.

$$\textbf{Det}\text{ection:} \qquad \frac{(\mathcal{E}, t, K, \emptyset, R, \Lambda)}{t^*}$$

if (i) the state $(K, \emptyset, R)$ is a congruence closure final state, (ii) there is a $t^* \in \mathcal{T}(\Sigma)$ such that $t \to_R^! c \leftarrow_R^* t^*$, and (iii) t^* is not further reducible by $\mathcal{E}$.[3]

Terminating in a state t^* means that we output t^* as the normal form of t.

[2] We assume that R consists only of simple D-rules and C-rules.

[3] This inference rule can be effectively applied. We simply non-deterministically guess for each constant $d \in K$, a D rule $f(\cdots) \to d$. Then the required t^* is one which reduces to c using these guessed rules, and which satisfies condition (iii).

Example 18 *Consider the problem of normalizing the term fa with respect to $\mathcal{E} = \{a \to b,\ fx \to gfx,\ gfb \to c\}$. The initial state obtained by the initialization rule is $\xi_0 = (\mathcal{E}, fa, \{c_0\}, \{c_0 \approx fa\}, \emptyset, \Lambda)$. In Table 10 we show some intermediate steps of a derivation. Note that in state* 10*, we identify c as a normal form-term of fa and terminate using* detection*. State* 9 *is reached by applying* Initialization*,* Narrowing *and* Selection *to state* 8*.*

i	K_i	*Eqns* $\mathcal{E}_i$	*Rules* $\mathcal{R}_i$	*Sequence* S_i	
0	$\{c_0\}$	$\{c_0 \approx fa\}$	$\emptyset$	Λ	
1	$\{c_0, c_1\}$	$\emptyset$	$\{a \to c_0, fc_0 \to c_1\}$	Λ	**CC**
2	K_1	$\emptyset$	R_1	$\langle\{(a, id)\}, \{(fx, id)\}, \{(gfb, id)\}\rangle$	**Init**
3	K_1	$\emptyset$	R_1	$\langle\{(c_0, id)\}, \{(fx, id)\}, \{(gfb, id)\}\rangle$	**Narr**
4	K_1	$\{a \approx b\}$	R_1	Λ	**Sel**
5	K_1	$\emptyset$	$R_1 \cup \{b \to c_0\}$	Λ	**CC**
6	K_1	$\emptyset$	R_5	$\langle\{(c_0, id)\}, \{(c_1, [x \mapsto c_0])\}, \{(gc_1, id)\}\rangle$	**Init**, **Narr**
7	K_1	$\{fc_0 \approx gfc_0\}$	R_5	Λ	**Sel**
8	K_1	$\emptyset$	$R_5 \cup \{gc_1 \to c_1\}$	Λ	**CC**
9	K_1	$\{gfb \approx c\}$	R_8	Λ	**Narr**$^+$
10	K_1	$\emptyset$	$R_8 \cup \{c \to c_1\}$	Λ	**CC**

Table 10: Example: Intermediate states in a derivation illustrating a naive congruence closure based normalization algorithm.

We emphasize that the above transformation rules are only correct when the rewrite system $\mathcal{E}$ is confluent. For example, if we attempt to normalize the term gfa with respect to the non-confluent rewrite system $\mathcal{E} = \{a \to b, fa \to gfa, gfb \to c\}$, then we could terminate with fb as the normal form of gfa.

7.1.2.3 Soundness and Completeness

Let $i_K(\mathcal{E}) = \{l\sigma \to r\sigma | \sigma : \mathcal{V} \mapsto K, l \to r \in \mathcal{E}\}$. Corresponding to every derivation we have a set $F \subset i_K(\mathcal{E})$ of rules processed during that derivation.

Theorem 6 *(Soundness) If a rewrite system $\mathcal{E}$ is confluent and a derivation from $(\mathcal{E}, t, \{c\}, \{c \approx t\}, \emptyset, \Lambda)$ terminates with t^*, then t^* is an $\mathcal{E}$-normal form of t.*

Proof. The only way we can terminate in state t^* is by applying the *Detection* rule, say to the state $(\mathcal{E}, t, K, \emptyset, R, \Lambda)$. Therefore, $t \rightarrow_R^! c^* \leftarrow_R^* t^*$. By correctness of congruence closure (corollary 1), we get, $t \leftrightarrow_{F\uparrow_R^\rho}^* t^*$ where $F \subseteq i_K(\mathcal{E})$ is the set of processed rule instances of $\mathcal{E}$. Using confluence of $\mathcal{E}$, $t \rightarrow_{\mathcal{E}}^* \circ \leftarrow_{\mathcal{E}}^* t^*$. But t^* is $\mathcal{E}$-irreducible. Therefore, t^* is the $\mathcal{E}$-normal form of t. □

Corollary 2 *Let $(\mathcal{E}, t, \{c\}, \{c \approx t\}, \emptyset, \Lambda) \vdash_N^* (\mathcal{E}, t, K, \emptyset, R, \Lambda) \vdash_N t^*$ be any derivation such that $F\uparrow_R^\rho$ is confluent. Then, t^* is an $\mathcal{E}$-normal form of t.*

In order to establish completeness of the procedure, we need to make sure that some normal form of t is eventually represented. For this, we require fairness conditions.

Definition 12 *A rule instance $l_i\sigma \rightarrow r_i\sigma$ is said to be selectible with respect to K and R if $l_i\sigma \rightarrow_R^* c$ where c is some constant in K.*

Definition 13 *A derivation is said to be* fair *if (i) the* Detection *rule is (eventually) applied if it is ever applicable, or, (ii) every selectible rule instance with respect to any intermediate sets K_i and R_i of the derivation, is eventually selected.*

A fair reduction strategy ensures that enough rule instances are processed so as to guarantee the representation of normal form term.

Theorem 7 *(Completeness) If $t \in \mathcal{T}(\Sigma)$ has a $\mathcal{E}$-normal form then a fair derivation starting from state $(\mathcal{E}, t, \{c\}, \{c \approx t\}, \emptyset, \Lambda)$ terminates in state t^*, where t^* is in $\mathcal{E}$-normal form.*

Proof. If a $\mathcal{E}$-normal form term of t ever gets represented via some R_i, then, the *Detection* rule will be applicable at the completion of every congruence closure phase thereafter. Hence, a fair derivation would terminate in a state defined by some normal form of t (by soundness).

Now say $t = t_0 \rightarrow_{\mathcal{E}} t_1 \rightarrow_{\mathcal{E}} t_2 \cdots t_{n-1} \rightarrow_{\mathcal{E}} t_n = t^*$ is a normalizing sequence. We claim that in a fair derivation, t^* is eventually represented. This we prove by an easy induction on the length n of the reduction sequence.

Initially t_0 is represented. Assume that at a certain intermediate step k of the execution, t_j is represented via R_k. Suppose we use the instance $l_i\sigma_i \rightarrow r_i\sigma_i$ (from $\mathcal{E}$) to reduce t_j to t_{j+1}. Therefore, $t_j = C[l_i\sigma_i]$. Since t_j is represented via R_k,

the term $l_i\sigma_i$ is represented via R_k too, and hence the rule instance $l_i\sigma_i \rightarrow r_i\sigma_i$ is selectible. By fairness it is eventually selected and processed. Once it is processed, $r_i\sigma_i$ is represented and hence $t_{i+1} = C[r_i\sigma_i]$ is represented. □

Note that we can ensure condition (ii) of fairness in any derivation by getting to a narrowing final state before applying the *Selection* rule. This is a consequence of the completeness of narrowing procedure for convergent systems.

One difficulty with the inference rules for normalization is that the termination checks essentially involve a non-deterministic step. To make this more efficient, we can store some information about which rules to use to find normal form terms in the congruence closure itself. This will be discussed next.

7.1.3 Rewrite Closure

In order to make the termination checks more efficient and useful, the basic congruence closure procedure requires additional refinements, so that one can determine whether a given represented term is in normal form or not. We will achieve this by *marking* certain rules, or, in other words, we will partition the set D into two sets: marked rules X, and unmarked rules N. The idea would be that terms represented by the left-hand sides of N rules will be $F{\uparrow}^{\rho}_{D}$-irreducible. Hence while searching for normal forms in the termination rules, instead of guessing, we will use the N-rules directly.

Definition 14 *An abstract congruence closure $R = D \cup C$ for F is called a* rewrite closure *if, the set D can be partitioned into $N \cup X$ such that for all terms t in $\mathcal{T}(\Sigma)$ represented by D, t is in normal form with respect to $F{\uparrow}^{\rho}_{R}$ if, and only if, it is represented by N.*

Equivalently, we can also say that a congruence closure $D = N \cup X$ is a rewrite closure for F if for every $t \in \mathcal{T}(\Sigma)$, t is $F{\uparrow}^{\rho}_{R}$-irreducible, iff, any reduction sequence starting with t contains only N steps, and no X steps. For example, let $F = \{fffa \rightarrow ffffa,\ ffffa \rightarrow fa\}$. If we let N_0 be the set of two rules, $a \rightarrow c_0$ and $fc_0 \rightarrow c_1$; and X_0 be a set of one rule, $fc_1 \rightarrow c_1$, then $N_0 \cup X_0$ is a rewrite closure for F. Rewrite closures need not always exist, though. Consider the set of equations $F' = \{ffffa \rightarrow fffa, ffffa \rightarrow fa\}$. We cannot get a rewrite closure from the abstract congruence closure $D_1 = \{a \rightarrow c_0,\ fc_0 \rightarrow c_1,\ fc_1 \rightarrow c_1\}$ for F'. Since a, fa and ffa are all in normal forms, we are forced to have all the D-rules in R_1 in the set N_0.

Note that there are several ways in which the set of D-rules can be partitioned. If $s \rightarrow t$ is a rewrite rule in F, then its left-hand side s is called an *F-redex* (or simply a

redex). One method is to put all D-rules, whose left-hand sides represent F-redexes, into the set X. Rules in X are therefore also called *redex rules*. We write $s \xrightarrow{n} t$ to indicate that $s \to t$ is a rule in N, and $s \xrightarrow{x} t$ to indicate that $s \to t$ is a rule in X. If $f(c_1, \ldots, c_k) \to c_0$ is a rule in X, then the term $f(c_1, \ldots, c_k)$ is also called a *redex template*. However, as we observed above, using this scheme for marking rules we may not still get a rewrite closure. We need the additional property of *persistence*.

Let $t[l\sigma] \to_R t[r\sigma]$ be a (one step) reduction using the rewrite rule $l \to r \in R$. This reduction will be called a *non-root reduction*, denoted by $\to_R^{nr}$ if $l\sigma$ is a proper subterm of t; otherwise this is called a *root reduction*, denoted by $\to_R^{root}$.

Definition 15 *Let R be a abstract congruence closure for a set of ground rules F over $\mathcal{T}(\Sigma \cup K)$. The set F is said to have the* persistence *property with respect to R if whenever, there exist terms $f(t_1, \cdots, t_n), f(t'_1, \cdots, t'_n) \in \mathcal{T}(\Sigma)$ such that, $f(t_1, \cdots, t_n)$ is $F{\uparrow}_R^\rho$-reducible at the top (root) position, and $f(t_1, \cdots, t_n) \leftrightarrow_{F{\uparrow}_R^\rho}^{*,nr} f(t'_1, \cdots, t'_n)$, it is always the case that, $f(t'_1, \cdots, t'_n)$ is $F{\uparrow}_R^\rho$-reducible.*

The idea behind the persistence property is simple. Since we put every redex-template in the set X, this simply means that we assume that all the terms represented by that template are reducible. The persistence property is true whenever this is actually the case. We note the following general result from [76].

Lemma 14 *Let R be a fully-reduced ground rewrite system. Then any reduction sequence $t \to_R^* s$ is unique up to a rearrangement of the individual steps.*

Using this result, we can easily establish the following.

Lemma 15 *Let F be a finite set of equations over $\mathcal{T}(\Sigma \cup K)$. A fully reduced congruence closure R of F can be extended to a rewrite closure if F has the* persistence *property with respect to R.*

Proof. Suppose R is a fully reduced congruence closure of F such that F has the persistence property. For every F-redex s, if

$$s \to_R^* t \to_D c,$$

then we put the rule $t \to c$ in the set X. All remaining rules are kept in N. We claim that $N \cup X \cup C$ is now a rewrite closure for F. Let $t \in \mathcal{T}(\Sigma)$ be any term represented by D. If t is *not* in $F{\uparrow}_R^\rho$-normal form, then it is *not* represented by N. This follows

from the definition of N and lemma 14. Next, suppose t is not represented by N. Therefore, it has a subterm t' such that

$$t' \rightarrow_R^* t'' \rightarrow_X c.$$

The way we partitioned rules implies that there is an F-redex s such that $s \rightarrow_R^* t'' \rightarrow_R c$. Since R is a congruence closure, there exists at least one term $s' \in \mathcal{T}(\Sigma)$ such that

$$s' \rightarrow_R^* s \rightarrow_R^* c$$

and therefore, $s' \leftrightarrow_R^{*,nr} t'$, which implies

$$s' \leftrightarrow_{F\uparrow_R^\rho}^{*,nr} t'.$$

By persistence, therefore, t' is $F\uparrow_R^\rho$-reducible, and hence, so is t. □

The converse of this theorem is however false, as the set $F = \{a \rightarrow b,\ fa \rightarrow c,\ c \rightarrow fb\}$ is *not* persistent (with respect to its abstract congruence closure), but the congruence closure can be extended to a rewrite closure.

7.1.3.1 Construction of Rewrite Closures

We give a set of transition rules (similar to the ones for congruence closure), that would compute the rewrite closure for a given F,assuming that the persistence property holds.

The *extension* inference rule is responsible for creating either N- or X-rules. For purposes of simplicity, we only introduce *simple* D-rules.

$$\textbf{Extension:} \quad \frac{(K, F[t], R)}{(K \cup \{c\}, F[c], R \cup \{t \xrightarrow{\alpha} c\})}$$

where $t \rightarrow c$ is a simple D-rule, t is a term occurring in (some equation in) F, $c \notin \Sigma \cup K$, α is x if t occurs as a left-hand side term in F and α is n otherwise.

Simplification can change marks on D-rules.

$$\textbf{Simplification:} \quad \frac{(K, F[t], R \cup \{t \xrightarrow{\alpha_1} c\})}{(K, F[c], R \cup \{t \xrightarrow{\alpha_2} c\})}$$

where α_2 is x if $t \rightarrow c$ is a D-rule and t occurs as a left-hand side term in F, and $\alpha_2 = \alpha_1$ otherwise.

Orientation always introduces either X-rules or C-rules.

$$\textbf{Orientation:} \quad \frac{(K, F \cup \{t \to c\}, R)}{(K, F, R \cup \{t \xrightarrow{\alpha} c\})}$$

if $t \succ c$ and α is x if $t \to c$ is a D-rule, and α is empty if $t \to c$ is a C-rule.

Deduction can change markings on rules too.

$$\textbf{Deduction:} \quad \frac{(K, F, R \cup \{t \xrightarrow{\alpha_1} c, t \xrightarrow{\alpha_2} d\})}{(K, F \cup \{c \to d\}, R \cup \{t \xrightarrow{\alpha_3} d\})}$$

where α_3 is x if either α_1 or α_2 is x; α_3 is n if both α_1 and α_2 are n; in case $t \in K$, α_3 is empty.[4]

The other rules like deletion, collapse and composition are identical to the respective transition rules for congruence closure and there are no changes in the markings of any rules.

The following Table 11 shows some states of a derivation from the initial state $(\emptyset, E_0, \emptyset)$, where $E_0 = \{fa \approx fb,\ ffb \approx a,\ fb \approx a\}$. The rewrite system R_4 is a rewrite closure for E_0.

i	Constants K_i	Equations E_i	Rules R_i
0	$\emptyset$	E_0	$\emptyset$
1	c_0, c_1, c_2	$ffb \approx a, fb \approx a$	$a \xrightarrow{n} c_0,\ b \xrightarrow{n} c_1,\ fc_1 \xrightarrow{n} c_2,\ fc_0 \xrightarrow{x} c_2$
2	K_1	$fb \approx a$	$R_1,\ fc_2 \xrightarrow{x} c_0$
3	K_2	$\emptyset$	$R_2,\ fc_1 \xrightarrow{x} c_0$
4	K_2	$\emptyset$	$a \xrightarrow{n} c_2,\ b \xrightarrow{n} c_1,\ fc_1 \xrightarrow{x} c_2\ fc_2 \xrightarrow{x} c_2\ c_0 \to c_2$

Table 11: Example: Intermediate states in a derivation illustrating the abstract rewrite closure construction transition rules.

Using lemma 15 we know that any derivation constructs a rewrite closure whenever F satisfies the persistence property. We use the symbol $\vdash_{RC}$ to denote the one-step transformation relation on states induced by the above transformation rules. *Final states* are states to which no further transition rules can be applied. Final states are always of the form $(K, \emptyset, R)$.

[4]In other words, $t \to d$ is a redex rule in the new state if, and only if, at least one of the two superposed rules is a redex rule.

7.1.3.2 Normalization via Rewrite Closure

The normalization procedure described earlier, can now be optimized by replacing the use of *congruence-closure* by *rewrite-closure*. The additional marking information in the rewrite closure can be used to optimize the *Detection* inference rule: in order to find a normal form in the equivalence class c, we need to check for only those term t' such that $t' \rightarrow_N^* c$.

We just mention those inference rules of the normalization procedure which now look different. The *congruence-closure* phase now is replaced by the *rewrite-closure* phase. The *initialization*, and *selection* rules are the same as in section 7.1.2. The new *Marked-Detection* rule uses the unmarked N rules to find a normal form term.

$$\textbf{Marked-Detection:} \quad \frac{(\mathcal{E}, t, K, \emptyset, R, \Lambda)}{t^*}$$

if (i) $(K, \emptyset, R)$ is a rewrite closure final state, (ii) there is a $t^* \in \mathcal{T}(\Sigma)$ such that $t \rightarrow_R^! c \leftarrow_N^* t^*$, where $R = N \cup X \cup C$ and (iii) t^* is in $\mathcal{E}$-normal form.

The soundness theorem 6 holds under these new rules. As before all we need for soundness is the confluence of $\mathcal{E}$. To establish completeness, now we need (a) persistence of the processed set of rules, and (b) a strategy that ensures that the normal form term is eventually represented. To ensure (b), we still have to use all the D rules for narrowing (and not just N rules). Fairness then would guarantee (b). Condition (a) then would ensure that the termination check is "complete".

Theorem 8 *(Completeness) If $t \in \mathcal{T}(\Sigma)$ has a $\mathcal{E}$-normal form t^* then a fair derivation starting from state $(\mathcal{E}, t, \{c\}, \{c \approx t\}, \emptyset, \Lambda)$ in which the processed rule instances F is always (eventually) persistent, terminates in state t^*.*

Proof. Similar to previous case. The proof of the fact that t^* is eventually represented is the same as before. Now consider the first time we test for termination after t^* is represented. By correctness of congruence closure, we would have, $t \rightarrow_D^* c^* \leftarrow_D^* t^*$. Since we have a rewrite closure (under the assumption of persistence), we can conclude that $t^* \rightarrow_N^* c^*$. This implies that the *marked-detection* rule is now applicable. Hence we identify t^* as being the desired normal-form and terminate. □

The conditions of *fairness* and *persistence* are complementary. In order to satisfy *persistence*, we should process fewer and only particular rules. On the other hand, to satisfy fairness we are required to process as many rules as possible. For example, informally, an innermost strategy in choosing instances to process shall always process

sets of instances that are persistent. But, unfortunately, such a strategy may violate fairness. In the next section, we consider two special cases of rewrite systems $\mathcal{E}$ where we can effectively satisfy both conditions together and introduce additional optimizations to the normalization transition rules.

7.1.4 Optimized Normalization and Special Cases

It appears wasteful to use all of the simple D-rules in the narrowing process. To compute normal forms we only need to select rule instances that reduce current *irreducible* terms. Intuitively, only simple N-rules should be employed by narrowing, as they represent irreducible terms.

We redefine the narrowing phase of normalization as follows:

$$(\mathcal{E}, t, K, \emptyset, N \cup X \cup C, S) \vdash_N (\mathcal{E}, t, K, \emptyset, N \cup X \cup C, S')$$

if $(K, N, S) \vdash_S (K, N, S')$ and $(K, \emptyset, N \cup X \cup C)$ is a rewrite closure final state.

Now the detection rule can be refined also.

$$\textbf{Refined-Detection:} \qquad \frac{(\mathcal{E}, t, K, \emptyset, R, S)}{t^*}$$

if assuming $R = N \cup X \cup C$, we have (i) (K, N, S) is a narrowing final state, (ii) there is a $t^* \in \mathcal{T}(\Sigma)$ such that $t \rightarrow_R^! c \leftarrow_N^* t^*$, and (iii) none of the constants in the first component of elements in the sets in S occur in the derivation $t^* \rightarrow_N c$.

Since we are narrowing with a restricted set N, we can also terminate if a narrowing phase produces no candidates for selection.

$$\textbf{Termination:} \qquad \frac{(\mathcal{E}, t, K, \emptyset, R, \langle \emptyset, \ldots, \emptyset \rangle)}{\Omega}$$

if assuming $R = N \cup X \cup C$ we have (i) the state $(K, \emptyset, R)$ is a rewrite closure final state, (ii) the state $(K, N, \langle \emptyset, \ldots, \emptyset \rangle)$ is a narrowing final state, and (iii) $\Omega = t^*$ if there is a $t^* \in \mathcal{T}(\Sigma)$ such that $t \rightarrow_R^! c \leftarrow_N^* t^*$; and $\Omega = \perp$, otherwise.

Terminating in a state $\perp$ means that we output "no normal form of t exists".

7.1.4.1 Correctness

Even under these restrictions, soundness (Theorem 6) is preserved.

Theorem 9 *(Soundness) If a rewrite system $\mathcal{E}$ is confluent and a derivation from $(\mathcal{E}, t, \{c\}, \{c \approx t\}, \emptyset, \Lambda)$ terminates with t^*, then t^* is an $\mathcal{E}$-normal form of t.*

Proof. We can terminate in state t^* using *Refined-Detection* or *Termination*. Assume that we got to state t^* by applying *Refined-Detection* to the state $(\mathcal{E}, t, K, \emptyset, R = N \cup X \cup C, S)$. Therefore, $t \rightarrow_R^! c^* \leftarrow_N^* t^*$. Using arguments identical to the proof of Theorem 6, we conclude that $t \rightarrow_{\mathcal{E}}^* \circ \leftarrow_{\mathcal{E}}^* t^*$. We claim that t^* is $\mathcal{E}$-irreducible. Suppose not, and let t^* be reducible by $l_i \rightarrow r_i \in \mathcal{E}$. Since $t^* \rightarrow_N^* c^*$, therefore l_i narrows via N to some constant that that occurs in the derivation $t^* \rightarrow_N^* c^*$. By completeness of narrowing for convergent systems, the set L_i contains an element whose first component is a constant that occurs in the derivation $t^* \rightarrow_N^* c^*$, thus violating condition (iii) of *Refined-Detection*.

The argument for *Termination* is identical. □

Completeness, however, is only preserved in under certain strong conditions—which are satisfied in certain special cases discussed later.

We shall assume that a rewrite closure is always fully reduced set of simple D-rules.

Definition 16 *Let $F \subset i_K(\mathcal{E})$ and let $R = D \cup C$ be a congruence closure for F. We say F is overlapping modulo R if there exist rules $l_1 \rightarrow r_1, l_2 \rightarrow r_2 \in F$ such that $l_1 \leftrightarrow_R^* l_2'$ for some non-constant subterm l_2' of l_2.*

For nonoverlapping rewrite systems reducibility of terms is preserved in the following sense.

Lemma 16 *Let F be non-overlapping modulo R. Let t be root reducible by some rule in $F\uparrow_R^\rho$ and also be reducible to t' at a non-root position by some rule in $F\uparrow_R^\rho$. Then t' is reducible by some rule in $F\uparrow_R^\rho$.*

Proof. Let t be reducible at the root position by rule $l_1 \rightarrow r_1 \in F$ with substitution $\rho_1 : K \mapsto \mathcal{T}(\Sigma)$ so that $t = l_1\rho_1$. Furthermore, let t be reducible at a non-root position by the rule $l_2 \rightarrow r_2 \in F$ with substitution $\rho_2 : K \mapsto \mathcal{T}(\Sigma)$. Therefore, $t = l_1\rho_1 = l_1\rho_1[l_2\rho_2]$, and

$$l_2 \leftrightarrow_R^* l_2\rho_2 \leftrightarrow_R^* l_1'\rho_1 \leftrightarrow_R^* l_1'.$$

Since F is nonoverlapping modulo R, the subterm l_1' of l_1 is necessarily a constant, say d. If d occurs more than once in l_1, then clearly t' is reducible by the rule $l_2\rho_2 \rightarrow r_2\rho_2$. If d occurs exactly once in l_1, say at position p, then the term at position p in t' is

still represented by d via R. Hence there exists a $\rho' : K \mapsto \mathcal{T}(\Sigma)$ such that $t' = l_1\rho'$.

□

Now, we can obtain one sufficient condition to establish persistence.

Lemma 17 *Suppose F is non-overlapping modulo R, where R is a congruence closure for F. Then, F is persistent with respect to R whenever $F{\uparrow}^{\rho}_{R}$ is confluent.*

Proof. Suppose $t \in \mathcal{T}(\Sigma)$ is root-reducible by $F{\uparrow}^{\rho}_{R}$. Let $t' \in \mathcal{T}(\Sigma)$ be any term such that

$$t \leftrightarrow^{nr,*}_{F{\uparrow}^{\rho}_{R}} t'.$$

Since $F{\uparrow}^{\rho}_{R}$ is confluent, we have the following

$$t \rightarrow^{nr,*}_{F{\uparrow}^{\rho}_{R}} \circ \leftarrow^{nr,*}_{F{\uparrow}^{\rho}_{R}} t'.$$

If t' is reducible by $F{\uparrow}^{\rho}_{R}$, then we are done. In the other case, we have

$$t \rightarrow^{nr,*}_{F{\uparrow}^{\rho}_{R}} t'.$$

Now performing induction on the length of this derivation and using lemma 16, we establish that t' is reducible. □

We can now state a completeness theorem for the new optimized set of transition rules as follows.

Theorem 10 *(Completeness) Suppose $t \in \mathcal{T}(\Sigma)$ and $\mathcal{E}$ is a set of rules. If there exists a strategy such that starting from state $(\mathcal{E}, t, \{c\}, \{c \approx t\}, \emptyset, \Lambda)$ it is the case that*
(i) the processed rule instances F_i is always non-overlapping modulo R_i,
(ii) the set $F_i{\uparrow}^{\rho}_{R_i}$ is confluent, and
(iii) a normal form t^* of t is eventually represented,
(where $(\mathcal{E}, t, K_j, \emptyset, R_j, S_j)$ are states at the end of the j-th narrowing phase), then every fair derivation obtained using that strategy terminates in state t^*.

Proof. Suppose t^* is represented via R_j. We show that the *Refined-Detection* rule is applicable at state $(\mathcal{E}, t, K_k, \emptyset, R_k, S_k)$ for $k \geq j$. Since t^* is a $\mathcal{E}$-normal form, therefore t^* is irreducible by $F_k{\uparrow}^{\rho}_{R_k}$.

Using conditions (i) and (ii) and Lemma 17, we know that F_k is persistent with respect to R_k. Since F_k is persistent, it follows from Lemma 15 that R_k is a rewrite closure. Therefore, $t^* \rightarrow^{*}_{N_k} c$. Furthermore, since t^* is in $\mathcal{E}$-normal form, condition (iii)

of *Refined-Detection* holds too. Therefore, the derivation would terminate in state t^*.

□

Since we restrict the narrowing phase to use only the unmarked N rules, it is possible that a fair strategy may not eventually represent normal forms of terms. In two special cases, we shall argue that the normal form term is eventually represented if it exists.

7.1.4.2 Normalization in Orthogonal Systems

We first discuss normalization of terms with respect to so-called "orthogonal" rewrite systems. A rewrite system $\mathcal{E}$ is called *orthogonal* if (i) all its rewrite rules are *left-linear* (i.e., contain no multiple occurrences of the same variable in any left-hand side) and (ii) it is nonoverlapping (i.e., without critical pairs except trivial ones of the form $t \approx t$). We need the following well-known result [54].

Lemma 18 *Every orthogonal term rewriting system is confluent.*

Theorem 11 *If $\mathcal{E}$ is an orthogonal rewriting system, then given any term and a fair strategy, the inference system outlined above finds its normal form with respect to $\mathcal{E}$, if one exists.*

Proof. Lemma 18 and Theorem 9 together imply the soundness of the result. In order to complete the proof, we need to satisfy conditions (i)–(iii) of Theorem 10.

Using the fact that $\mathcal{E}$ is orthogonal, we can establish the confluence of $F_j{\uparrow}^{\rho}_{R_j}$ using arguments similar to those used in proof of Lemma 18. Since $\mathcal{E}$ is non-overlapping, therefore F_j is non-overlapping modulo R_j (as rules in F_j are identical to rules in $\mathcal{E}$ with constants substituted for variables).

Finally we show that if t^* is an $\mathcal{E}$-normal form of t, then t^* is eventually represented. Suppose that $t = t_0 \rightarrow t_1 \cdots t_{n-1} \rightarrow t_n = t^*$ is a normalizing reduction sequence. We claim that eventually each term t_i is represented. The claim is true for $i = 0$. Suppose t_i is represented. Therefore, $t_i \rightarrow^*_{D_j} c$ for some $c \in K_j$, where $(K_j, \emptyset, R_j)$ is a rewrite closure final state. Suppose $t_i \rightarrow t_{i+1}$ using some ground instance represented by the rule $l\sigma \rightarrow r\sigma \in i_{K_j}(\mathcal{E})$. If $l\sigma$ is reducible to a constant via N_k for all $k \geq j$, then by fairness, this instance is selected and eventually t_{i+1} is represented too. For the other case assume $l\sigma$ is not reducible to a constant via N_j. Therefore, we have

$$l\sigma \not\rightarrow^*_{N_j} d \text{ and } l\sigma \rightarrow^*_{N_j \cup X_j} d$$

But we know that $N_j \cup X_j \cup C_j$ is a rewrite closure for F_j. Since $\mathcal{E}$ is non-overlapping the only instance whose processing could have caused the marking of any rule used in the reduction $l\sigma \rightarrow^*_{R_j} d$ can be $l\sigma \rightarrow r\sigma$. Therefore, this rule has been processed and hence t_{i+1} is represented. This completes the induction step. Hence eventually t^* is represented. □

7.1.4.3 Normalization in Convergent Systems

In order to perform normalization of terms with respect to convergent systems, we need a strategy (to choose instances to select) so that the processed rule instances are nonoverlapping and their extension confluent. But, if F is non-overlapping modulo R, then $F\uparrow^{\rho}_{R}$ is locally confluent. Since convergent systems are *terminating*, $F\uparrow^{\rho}_{R}$ is also confluent. Using an innermost strategy allows us to make sure that F is non-overlapping.

Definition 17 *Let $l\sigma \rightarrow r\sigma$ and $l'\sigma' \rightarrow r'\sigma'$ be two selectible instances (with respect to only the unmarked rules now). Say $l\sigma \rightarrow^*_N c$ and $l'\sigma' \rightarrow^*_N c'$. An* innermost strategy *is one that makes sure that if there exists a term $t[c]$ containing c such that $t[c] \rightarrow^*_N c'$, then the rule $l\sigma \rightarrow r\sigma$ is chosen first.*

Innermost strategy is important because of the following.

Lemma 19 *Let $\mathcal{E}$ be convergent. Let $(\mathcal{E}, t, K_j, \emptyset, R_j, \Lambda)$ denote the state of any derivation at the end of the j-th rewrite closure phase, i.e.,*

$$(\mathcal{E}, t, \{c\}, \{c \approx t\}, \emptyset, \Lambda) \vdash^{(RC^!)}_N (\mathcal{E}, t, K_1, \emptyset, R_1, \Lambda) \vdash^{(S)}_N (\mathcal{E}, t, K_1, \{\rho_1\}, R_1, \Lambda) \vdash^{(RC^!)}_N (\mathcal{E}, t, K_2, \emptyset, R_2, \Lambda) \vdash^{(S)}_N \cdots$$

where the instances ρ_i, defined as $l_i\sigma_i \rightarrow r_i\sigma_i$, added to the closure in the $(i+1)$-th rewrite closure phase ($j \geq 1$) is selected using an innermost strategy. Let $F_j = \{l_k\sigma_k \rightarrow r_k\sigma_k : 0 < k < j\}$. Then the following statements are true, for all $j \geq 1$:

1. *The set F_j is non-overlapping modulo R_j.*
2. *For every $u \in \mathcal{T}(\Sigma \cup K_j)$ in $D_j \cup C_j$-normal form, there exists a term $t^u \in \mathcal{T}(\Sigma)$ such that $t \rightarrow^*_{N_j} u$. This t is unique.*
3. *The innermost strategy to select the next instance to process after the end of the rewrite closure phase j is well-defined, i.e., is unambiguous.*

4. *If u is a proper subterm of $l_j\sigma_j$ and $u \rightarrow^!_N u^!$, then the term $t^{u^!}$ as specified in property 2 is in $\mathcal{E}$-normal form.*

5. *The set F_{j+1} is non-overlapping modulo R_{j+1}.*

Proof. We shall prove the implications (1) $\Rightarrow$ (2), (2) $\Rightarrow$ (3), (3) $\Rightarrow$ (4), and (4) $\Rightarrow$ (5) in that order. As the base case we note that each of the statements 1—4 are true for $j = 0$. Initially N_1 contains the rules needed to represent the term t and $F_1 = \emptyset$. Since $F_1 = \emptyset$, therefore statements 4 and 1 are vacuously true. Statements 2 and 3 are true because each term $u \in \mathcal{T}(\Sigma \cup K_1)$ represents exactly one term in $\mathcal{T}(\Sigma)$ via R_1.

(1) $\Rightarrow$ (2): Assume that F_j is non-overlapping modulo R_j. Clearly, the set $F_j{\uparrow}^{\rho}_{R_j}$ is locally confluent. Since $\mathcal{E}$ is terminating, the set $F_j{\uparrow}^{\rho}_{R_j}$ is terminating, and hence confluent. Therefore, using Lemma 17 we know that F_j is persistent with respect to R. Let $u \in \mathcal{T}(\Sigma \cup K_j)$ be any term in R_j-normal form. Equivalence class represented by u has a unique $F_{j+1}{\uparrow}^{\rho}_{Rj+1}$-normal form t^u. Since $N_{j+1} \cup X_{j+1}$ is a rewrite closure, therefore, we have $t^u \rightarrow^!_{N_j} u$.

(2) $\Rightarrow$ (3): We now show that the innermost strategy is well-defined. Suppose $l\sigma \rightarrow r\sigma$ and $l'\sigma' \rightarrow r'\sigma'$ are both selectible after the end of rewrite closure phase j. Say, $l\sigma \rightarrow^!_{N_j} u$, and, $l'\sigma' \rightarrow^!_{N_j} u'$. Suppose the innermost strategy is ambiguous. Therefore, there exist terms $s[u]$ and $s'[u']$ containing the terms u and u' respectively, such that $s[u] \rightarrow^*_{N_j} u'$ and $s'[u'] \rightarrow^*_{N_j} u$. This means we have $s'[s[u]] \rightarrow^*_{N_j} s'[u'] \rightarrow^*_{N_j} u$. By assumption there exists a unique term t^u represented by u via N_j, i.e., $t^u \rightarrow^*_{N_j} u$. This means that $s'[s[t^u]]$ is another term which satisfies conditions in property 2, but this contradicts uniqueness of t^*.

(3) $\Rightarrow$ (4): Suppose statement 4 is false. We have $l_j\sigma_j \rightarrow^*_{N_j} u_j$, for some u_j occurring (as a proper subterm) in R_j. Without loss of generality, assume that all constants in $l_j\sigma_j$ are in R_j-normal form. For all such constants, define a mapping $\rho : K \mapsto \mathcal{T}(\Sigma)$ such that $c\rho = t^c$ (i.e., $c\rho \rightarrow^*_{N_j} c$). Now let $u \in \mathcal{T}(\Sigma \cup K_j)$ be the smallest proper subterm of $l_j\sigma_j$ such that $u\rho$ is *not* in $\mathcal{E}$-normal form.[5] Since $u\rho$ is root-reducible by $\mathcal{E}$, and u is a proper subterm of $l_j\sigma_j$, by completeness of narrowing, we violate the assumption that $l_i\sigma_i \rightarrow r_i\sigma_i$ was chosen using an innermost strategy.

(4) $\Rightarrow$ (5): Next, suppose that $F_{j+1} = \{l_i\sigma_i \rightarrow r_i\sigma_i : 1 \leq i \leq j\}$ is overlapping modulo R_{j+1}. Say, $l_m\sigma_m \rightarrow r_m\sigma_m$, $l_n\sigma_n \rightarrow r_n\sigma_n \in F_{j+1}$ are the two overlapping

[5]Note that since u is a subterm of $l_j\sigma_j$, we also have $u \rightarrow^!_{N_j} u^!$, and $u^!$ occurs in R_j. By Lemma 14, we know that $t^{u^!} = t^u = u\rho$.

rules. Therefore, there exist mappings $\rho_m, \rho_n : K_j \mapsto \mathcal{T}(\Sigma \cup K_j)$ and a non-constant proper subterm l'_n of $l_n\sigma$ such that $l_m\sigma_m\rho_m = l'_n\rho_n$ with $c\rho \leftrightarrow^*_{R_{j+1}} c$ for all constants c. This shows that the $\mathcal{E}$-normal form term $t^{l'_n}$ is reducible by the rule $l_m \to r_m \in \mathcal{E}$, a contradiction. □

Thus we have the following result.

Theorem 12 *Let $\mathcal{E}$ be a convergent rewriting system. Then given any term, the inference system outlined above finds its normal form.*

Proof. We choose the next instance to be processed in the *selection* rule using an innermost fair strategy. In every application of this rule, we choose to process *exactly one* instance. Again soundness of the method follows from Theorem 9. In order to complete the proof we need to show that an innermost strategy guarantees satisfiability of conditions (i)–(iii) of Theorem 10. Conditions (i) and (ii) follow from Lemma 19.

Finally, we show that the normal form term t^* of any term t is always eventually represented. It suffices to prove that if $t \in \mathcal{T}(\Sigma)$ is such that t is represented (via R_j), and t is reducible by $\mathcal{E}$, then some term t' such that $t \to_{\mathcal{E}} t'$ is eventually represented. Since R_j is a rewrite closure, if t is *not* represented via N_j, then t is reducible by the processed instances, and hence some t' with $t \to_{\mathcal{E}} t'$ is represented too. Suppose t is represented via N_j. Let $t = t[l\sigma] \to_{\mathcal{E}} t' = t[r\sigma]$ using an innermost reduction step. Since $\mathcal{E}$ is convergent all proper subterms of $l\sigma$ are in $\mathcal{E}$-normal form. Clearly, $l\sigma \to^*_{N_j} c$ for some c. Hence, some instance of $l \to r$ is selectible (in an innermost strategy). □

7.1.5 Conclusion

Normalization of terms by a given set of rewrite rules is critical for the efficient implementation of rewrite-based systems. Simple straight-line reduction methods can be made more efficient by incorporating a history of reduction steps into the normalization process, so as to avoid repeating similar rewrite steps. Chew [30] adapted congruence closure techniques to obtain a practical technique such a non-oblivious normalization procedure. Chew's procedure applies to orthogonal systems, but was refined and generalized by Verma and Ramakrishnan [82] and Verma [81].

We have presented a general formalism, based on transformation rules, within which both Chew's original method and variants thereof, as well as more general normalization procedures, can be described. The most comprehensive previous results

were obtained by Verma [81], who specified several postulates on a (priority) rewrite relation, which suffice to ensure completeness of a rewrite closure-based procedure for normalization. The postulates that are relevant for standard rewrite relations imply that the given rewrite system be confluent and non-overlapping, which means that our results cover a broader class of rewrite systems.

We believe that our approach sheds new light on the basic concepts underlying non-oblivious normalization in that it relates normalization methods to standard term rewriting techniques, such as completion and narrowing. The transformation rules for the two basic components of non-oblivious normalization—congruence closure and narrowing—are essentially specialized versions of standard rules used to describe rewrite-based deduction and computation methods. Rules specific to non-oblivious normalization control the interface between the two components and the termination of the overall process.

7.2 Rigid E-Unification

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 [36] and it is not known if a complete set of rigid E-unifiers in the sense of [46] 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 [37]. 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) [45, 46, 16, 37, 47, 18]. 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 [51, 13] and formulate a rigid E-unification method in terms of fairly abstract transformation rules.

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 [46], or by treating equations as bi-directional constrained rewrite rules (and using unsatisfiable constraints to eliminate wrong orientations) [16]. 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.[6] 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

[6] 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 [13]. 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.

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, 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" [46]. 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.

7.2.1 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 5 *(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 13 *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 5 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 disjoint domains, then the congruence induced by E_σ is identical to the congruence induced by $\cup_i E_{\sigma_i}$. We denote by $E_{\vec{\sigma}}$ the set $\cup_i E_{\sigma_i}$.

The *variable dependency ordering* $\succ^S_V$ induced by a set S of equations on the set V of variables is defined by: $x \succ^S_V y$ 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 $\succ^S_V$ 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 $x_j \not\succ^S_V x_i$ 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 18 *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 14 *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 5 part (a), we get

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

Conversely, suppose σ is an idempotent substitution such that $s \leftrightarrow^*_{(E\cup E_\sigma)^g} t$. Then, we have $s\sigma \leftrightarrow^*_{(E\sigma\cup E_\sigma\sigma)^g} t\sigma$. 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 13 to establish that the congruences induced by $E \cup E_\sigma$ and $E \cup E_{\vec{\theta}}$, where $\theta_1 \dots \theta_n$ 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 15 *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 5. Therefore, it follows that $E_{\sigma\theta} \subseteq \leftrightarrow^*_{E^g\cup E^g_\theta}$. But again by Proposition 5, $x\sigma\theta \leftrightarrow^*_{E^g_\theta} x\sigma$ 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 $\sigma \preceq^V_{E^g} \theta$. □

7.2.2 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 14, 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).

We recapitulate here the notion of an abstract congruence closure suitably modified to treat all variables essentially as constants in the signature. 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 $\Gamma \cup K$[7] such that (i) each rule in R is either a *D-rule* of the form $t \approx c_0$ where $t \in \mathcal{T}(\Gamma \cup K) - K$ and c_0 is a constant 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 $u \in \mathcal{T}(\Gamma \cup K)$ that is in normal form with respect to R, there exists a term $t^u \in \mathcal{T}(\Gamma)$ such that $t^u \rightarrow^*_{R^g} u$. 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 $gfx \approx c_1,\ fgy \approx c_1,\ z \approx c_1$ is an abstract congruence closure (with respect to Γ and $\{c_1\}$) for E_0.

The rules for construction of a congruence closure can easily be modified to deal with the new symbols—variables which are considered as part of the signature Γ. Additionally, we merge the second and third components used in the construction of a congruence closure to just one component.

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 $E, E?$ and S 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

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

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 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 $S = \{c_1 \approx x, x \approx y\}$. This set can be transformed by replacing the constant c_1 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 $S = \{c_1 \approx x, x \approx y, y \approx z\}$, we can't eliminate the constant c_1 since every term congruent to c_1 modulo E_1 contains one of x, y or z.

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. [2]. 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. [2].

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.

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

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 $\Sigma \cup V$ and K, 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.

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

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 $\Sigma \cup V$ and K, 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 $\{gfx \approx c_1, fgy \approx c_1, z \approx c_1, \}$ for E_0. Table 12

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	$\{c_1\}$	E_1	$\{x, y, z\}$	$\{fx \approx gy\}$	$\emptyset$	*Narrow*
2	$K_1 \cup \{x\}$	$E_1 \cup \{gy \approx x\}$	$\{y, z\}$	$\{c_1 \approx gy\}$	$\{gy \approx x\}$	*C-Closure*
3	$K_2 \cup \{c_2\}$	E_3	$\{y, z\}$	$\{c_1 \approx gy\}$	$\{gy \approx x\}$	*Narrow*
4	$K_2 \cup \{y\}$	$E_3 \cup \{y \approx c_1\}$	$\{z\}$	$\{c_1 \approx c_1\}$	$S_3 \cup \{c_1 \approx y\}$	*Delete*
5	K_4	E_4	$\{z\}$	$\emptyset$	S_4	

Table 12: Example: Intermediate states in a derivation illustrating the rigid E-unification transition rules.

describes some intermediate stages of a derivation. The set $E_3 = \{x \approx c_2, gy \approx c_2, z \approx c_1, gc_1 \approx c_1, fc_2 \approx c_1\}$ is an abstract congruence closure for E_2. Since the set $E?_5$ is empty, the rigid unification process is complete. Any set $S_4{\uparrow}_{E_4}$ is a rigid unifier of fx and gy. Since $S_4 = \{gy \approx x, c_1 \approx y\}$ we need to choose a term in the equivalence class of c_1 (not containing x). For instance, we may choose z for c_1 to get the set $S_4{\uparrow}_{E_4} = \{gy \approx x, z \approx y\}$ and the corresponding unifier $[x \mapsto gy; y \mapsto z]$.

A cautious reader might note that the transition rules *Compress1* and *Compress2* are crucial optimizations for this application. In fact, we should use *Extension* minimally so that term structure is preserved as much as possible.

7.2.3 Correctness

Let U be an infinite set of constants from which new constants are chosen in *C-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 *C-closure*, then (i) E_j is an abstract congruence closure (with respect to $\Sigma \cup V$ and K) for E_i and (ii) E_j is contained in a well-founded simplification ordering.

We use the symbol $\vdash_{REU}$ to denote the one-step transformation relation induced by *C-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 *C-closure*.

Theorem 16 *(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 20 *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* $E?_0 \subseteq \leftrightarrow^*_{(E_0 \cup S_n \uparrow E_n)^g}$.

Proof. We prove both claims by induction on the length i of the derivation. For claim (a), we show that in the i-th state, S_i is E_i-feasible on V_0. The base case is trivial as $S_0 = \emptyset$. For the induction step, assuming S_i is E_i-feasible on V_0, we need to prove that S_{i+1} is E_{i+1}-feasible on V_0. We verify this for individual rules now. The E- and S-components are left unchanged by *deletion* and *decomposition* rule. In case of *elimination*, *narrowing* and *superposition*, if S_i is E_i-feasible on V_0, then S_{i+1} is E_i-feasible on V_0 (because of condition (ii) in *elimination* and *superposition* and condition (iii) in *narrowing*). But since $E_i \subset E_{i+1}$ for each of these rules, S_{i+1} is also E_{i+1}-feasible on V_0. Since an abstract congruence closure does not change the congruence defined on the original signature, the claim holds for the *C-Closure* rule.

For claim (b), we prove that if $E?_{i+1} \subseteq \leftrightarrow^*_{(E_{i+1} \cup S_n\uparrow_{E_n})^g}$, then, $E?_i \subseteq \leftrightarrow^*_{(E_i \cup S_n\uparrow_{E_n})^g}$ by considering each transition rule separately. Let $E?_{i+1} \subseteq \leftrightarrow^*_{E^g_{i+1} \cup (S_n\uparrow_{E_n})^g}$. If either *deletion* or *decomposition* was used to obtain state ξ_{i+1}:

$$E?_i \subseteq \leftrightarrow^*_{E?^g_{i+1}} \subseteq \leftrightarrow^*_{E^g_{i+1} \cup (S_n\uparrow_{E_n})^g} \subseteq \leftrightarrow^*_{E^g_i \cup (S_n\uparrow_{E_n})^g}$$

If either *elimination*, *narrowing* or *superposition* was used to obtain state ξ_{i+1}:

$$E?_i \subseteq \leftrightarrow^*_{E^g_{i+1} \cup (S_n\uparrow_{E_n})^g} \subseteq \leftrightarrow^*_{E^g_i \cup (S_n\uparrow_{E_n})^g \cup E^g_\theta} \subseteq \leftrightarrow^*_{E^g_i \cup (S_n\uparrow_{E_n})^g}$$

as either E_θ is always contained in S_n.

In case of the *C-Closure* rule, we note that the congruence induced by E_i is identical to that induced by E_{i+1} on the set $\mathcal{T}(\Sigma \cup V_0 \cup K_i)$ and that any term in $E_i \cup E?_i$ is over the signature $\Sigma \cup K_i \cup V_0$. Therefore, $E?_i \subseteq \leftrightarrow^*_{E^g_{i+1} \cup (S_n\uparrow_{E_n})^g}$ and hence, $E?_i \subseteq \leftrightarrow^*_{E^g_i \cup (S_n\uparrow_{E_n})^g}$. □

Theorem 17 (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 20. Since $E?_n = \emptyset$, the antecedent of the implication in part (b) of Lemma 20 is vacuously satisfied and hence $E?_0 \subseteq \leftrightarrow^*_{(E_0 \cup S_n\uparrow_{E_n})^g}$.

Note that by Theorem 13, 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, $s \leftrightarrow^*_{E^g_0 \cup E^g_\sigma} t$. Using Theorem 14, we establish that σ is a rigid E_0-unifier of s and t. □

Theorem 18 (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 $E_{S_n\uparrow_{E_n}} \subseteq \leftrightarrow^*_{E^g_0 \cup E^g_\theta}$.*

Proof. 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 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 16 and a well-founded ordering on substitution-feasible sets S_i's. Using proof transformation arguments, we argue that if a pair (ξ_i, S^i) can not be reduced then the set $E?_i$ is empty. 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 $(S^i, \emptyset)$.

$$\mathbf{R1}: \quad \frac{(D' \cup \{c \approx x\}, C')}{(D', C' \cup \{c \approx x\})} \quad \text{if } c \in K_i \cup V_i,\ x \not\rightarrow^*_{E_i^g/C'^g} c^8$$

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

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

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

These rule construct a generalized congruence closure for the initial set $D' \cup C'$ (modulo the congruence induced by E_i). One can also think about them as converting the S^i presentation of a substitution into some sort of a minimal triangular form. 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. This fact is proved in Lemma 23. Furthermore, Lemma 22 establishes that the set of rules $R1$–$R4$ is terminating.

Finally using proof transformation arguments, Lemma 24 shows that if S is a solution for state ξ, then there exists an REU derivation such that the $E?$-component in the final state is empty.

□

Lemma 21 *Let S be substitution-feasible. If (D', C') is any intermediate state in a derivation (using rules R1–R4) starting from the state $(S, \emptyset)$ and $x \in V$, then, the set $\{c \in K : c \leftrightarrow^*_{C'^g} x\}$ contains at most one element from the set K, and that too only if a rule with right-hand side x is in C'.*

Proof. Consider the directed graph where the symbols in $K \cup V$ are the vertices, and there is a directed edge from c to x if $c \approx x \in C'$. The set $\{c \in K \cup V : c \leftrightarrow^*_{C'^g} x\}$ represents a connected component. The vertices K have no incoming edges. But, if a directed connected graph contains two vertices with no incoming edges, then there exists a vertex with indegree at least two. But all vertices in our graph have indegree at most one. This argument also shows that $C'^{g^{-1}}$ is confluent, and the rest of the lemma follows. □

Lemma 22 *Let S be substitution-feasible, and let E-component of the current REU state be a fully-reduced abstract congruence closure (with respect to $\Sigma \cup V$ and K) containing no redundant C-equations. Any derivation $(D'_0, C'_0) \vdash_{R1-R4} (D'_1, C'_1) \vdash_{R1-R4} \cdots$ with starting state $(D'_0 = S, C'_0 = \emptyset)$ and using the rules* R1*,* R2*,* R3 *and* R4 *is finite.*

Proof. To every state (D', C'), assign a measure $m(D', C')$ equal to the multiset of left-hand sides of all rules in D'. States are ordered by comparing the associated measures using a multiset extension of the ordering $\succ'$ on terms, where $\succ'$ is a simple lexicographic path ordering based on the following precedence on symbols in $\Sigma \cup V \cup K$: $f \succ c$ if $f \in \Sigma \cup V$ and $c \in K$; $f \succ x$ if $f \in \Sigma$ and $x \in V$; and, $c \succ d$ if $c, d \in K$ and $c \rightarrow^*_{E^g/C'^g_\infty} d$, where $C'_\infty = \cup_i C'_i$. It is easy to see that all rules reduce the defined measure on states. We need to establish that $\succ'$ restricted to constants is well-founded.

Suppose not. Then, at some point we go from state $(D'_i \cup \{c \approx x\}, C'_i)$ to $(D'_{i+1}, C'_{i+1} = C'_i \cup \{c \approx x\})$ using rule *R1* such that E^g/C'^g_i is well-founded whereas $E^g/{C'_{i+1}}^g$ is not (both restricted to constants in K). This means that there exist $c_0, d_0 \in K$ such that $d_0 \rightarrow^*_{E^g/C'^g_i} c_0$ and $c_0 \rightarrow_{E^g/{C'_{i+1}}^g} d_0$. Since this latter proof should use the newly added rule $c \approx x$, we would have $c_0 \leftrightarrow^*_{C'^g_i} c \rightarrow x \leftrightarrow^*_{C'^g_i} y \rightarrow_{E^g} d_0$. The only possibility for the use of $c \approx x$ is in the direction shown since, by Lemma 21, the equivalence class of x modulo C'_i contains only variables. Putting these two proofs together, we get $x \rightarrow^*_{E^g/C'^g_i} c$, which means that the condition in rule *R1* is violated. □

Lemma 23 *Let E be as in lemma 22. Let S be a E-feasible (w.r.t V) and substitution-feasible set. Consider any maximal derivation*

$$(S, \emptyset) = (D_0, C_0) \vdash_{R1-R4} (D_1, C_1) \vdash_{R1-R4} \cdots \vdash_{R1-R4} (D_k, C_k).$$

such that none of the rules is applicable to the final state. Let $S_i = D_i \cup C_i$.

1. *The set S_i is a substitution-feasible set, and consequently S_i^{-1} is convergent.*
2. *The set S_{i+1} is equivalent modulo E^g on V to S_i. In fact, for any $t \approx x \in S_i$, $t \rightarrow^*_{(E \cup D_{i+1})^g / C^g_{i+1}} x$.*
3. *Each substitution-feasible set S_i is also E-feasible.*
4. *Any proof pattern of the form given in the left-hand side of $\Rightarrow$ can be replaced by a proof of the form on the right.*

$$\begin{array}{lcl} \circ \leftarrow_{D^g_k} \circ \leftrightarrow^*_{C^g_k} \circ \rightarrow_{D^g_k} \circ & \Rightarrow & \circ \rightarrow_{D^g_k / C^g_k} \circ \leftarrow_{D^g_k / C^g_k} \circ \\ \circ \leftarrow_{D^g_k} \circ \leftrightarrow^*_{C^g_k} \circ \rightarrow_{E^g} \circ & \Rightarrow & \circ \rightarrow_{E^g / C^g_k} \circ \leftarrow_{D^g_k / C^g_k} \circ \end{array}$$

5. *If $t_1 \leftrightarrow^*_{S_i^g} t_2$, then t_1 and t_2 are syntactically unifiable. Moreover, the most general unifier σ' is E-feasible. The set $S_{i_{new}} = S_i - \{t \approx x \in S_k : \exists(t').t' \approx x \in E_{\sigma'}\} \cup E_{\sigma'}$ is substitution-feasible and is equivalent modulo E^g on V to the set S_i.*

Proof. All right-hand sides of equations in S_i are variables as rules $R1$–$R4$ do not simplify equations on the right. We next prove that the variable dependency ordering $\succ_V^{S_i}$ is well-founded by induction on the length i of derivations. For the case when $i = 0$, we note that S is substitution-feasible. We establish the induction step by contradiction. Given that $\succ_V^{S_{j-1}}$ is well-founded, assume that the partial order $\succ_V^{S_j}$ is *not* well-founded. There are four ways in which we could have obtained state (D_j, C_j) from the state (D_{j-1}, C_{j-1}). In all cases, we can show that if $\succ_V^{S_j}$ is not well-founded, then $\succ_V^{S_{j-1}}$ is not well-founded too. Since right-hand sides of all rules in E are constants in K, the only non-trivial case is when $(D_{j-1}, C_{j-1}) \vdash_{R4} (D_j, C_j)$. Since $\succ_V^{S_j}$ is not well-founded, there exist rules

$$t_0[x_0] \approx x_1,\ t_1[x_1] \approx x_2,\ \ldots,\ t_m[x_m] \approx x_0$$

in S_j. Since $\succ_V^{S_{j-1}}$ is well-founded, not all of the above rules are in S_{j-1}. Wlog we assume that $D_{j-1} = D' \cup \{t_m[l'] \approx x_0\}$ and $D_j = D' \cup \{t_m[x_m] \approx x_0\}$. But the

only rule with right-hand side x_m in $D' \cup C_{j-1}$ is $t_{m-1}[x_{m-1}] \approx x_m$, and therefore, $l' \leftrightarrow^*_{C^g_{j-1}} t_{m-1}[x_{m-1}]$. Let $l' = t'_{m-1}[y]$ where $y \leftrightarrow^*_{C^g_{j-1}} x_{m-1}$. Since ${C^g_{j-1}}^{-1}$ is confluent (Lemma 21), therefore $y \leftarrow^*_{C^g_{j-1}} z \rightarrow^*_{C^g_{j-1}} x_{m-1}$. But since all right-hand sides are distinct variables, $z = x_i$ for some $0 \leq i \leq m-1$. In all cases, we can argue that $x_0 \succ_V^{S_{k-1}} y$. But then, we already have $y \succ_V^{S_{k-1}} x_0$.

We note that for every substitution-feasible set S_i the set $(S^g_i)^{-1}$ is convergent (terminating and non-overlapping).

Property 2: Let $t \approx x \in S_{i-1}$. If $t \approx x \in S_i$, then there is nothing to prove. If not, then it is deleted in an application of either the *R3* or the *R4* rule. In both case however, it is easy to see that $t \rightarrow^*_{S^g \cup D^g_i/C^g_i} x$. The converse direction is trivial as the rules $R1$–$R4$ are all sound.

Property 3: Every *R3* or *R4* step can be thought of as consisting of multiple applications of the following basic steps:

$$\textbf{R5}: \quad \frac{(D' \cup \{t[y] \rightarrow x\}, C')}{(D' \cup \{t[z] \rightarrow x\}, C')} \text{if } y \leftrightarrow_{C'^g} z \qquad \textbf{R3}': \quad \frac{(D' \cup \{t[l] \rightarrow x\}, C')}{(D' \cup \{t[c] \rightarrow x\}, C')} \text{if } l \rightarrow_{E^g} c$$

$$\textbf{R4}': \quad \frac{(D' \cup \{t[l] \rightarrow x\}, C')}{(D' \cup \{t[y] \rightarrow x\}, C')} \text{if } l \rightarrow_{D'^g} y$$

It suffices to prove that for $(D_i, C_i) \vdash_{R3', R4', R5} (D_{i+1}, C_{i+1})$, if $S_i = D_i \cup C_i$ is E-feasible, then so is $S_{i+1} = D_{i+1} \cup C_{i+1}$. This is straight-forward for rule *R3'*. For rule *R5*, first assume that $y \rightarrow z \in C'$ (Argument for *R4'* is similar). Since S_i is E-feasible, let $S_i{\uparrow}_E$ be the witness. Suppose $t'[y'] \rightarrow x \in S_i{\uparrow}_E$. We claim that the set $S_{i+1}{\uparrow}_E = S_i{\uparrow}_E - \{t'[y'] \rightarrow x\} \cup \{t'[z] \rightarrow x\}$ is the required witness for E-feasibility of S_{i+1}. We only need to prove that this set is substitution-feasible. Given that $\succ_V^{S_i{\uparrow}_E}$ is well-founded, we need to prove that $\succ_V^{S_{i+1}{\uparrow}_E}$ is well-founded. Wlog we assume that all instances of y in left-hand sides of S_i are *expanded* to the same term in left-hand sides of $S_i{\uparrow}_E$. The the set $\succ_V^{S_{i+1}{\uparrow}_E} = (\succ_V^{S_i{\uparrow}_E} \cup \{(z, x)\})^+$ is not well-founded only if $x \succ_i z$. But, since $y \approx z \in S_i$, therefore if if $x \succ_V^{Subs_i{\uparrow}_E} z$ then $x \succ_V^{S_i{\uparrow}_E} x$.

For the other case assume $z \approx y \in C'$. Suppose $z' \approx y \in S_i{\uparrow}_E$. We claim that the set $S_i{\uparrow}_E - \{t'[y'] \approx x\} \cup \{t'[z'] \approx x\}$ is the required witness for E-feasibility of S_{i+1}. That this set is substitution-feasible follows from noting that $\succ_V^{S_{i+1}{\uparrow}_E} \subseteq \succ_V^{S_i{\uparrow}_E}$.

Property 4: Note that there are no non-trivial overlaps of the form

$$\circ \leftarrow_{D^g_\sigma} \circ \leftrightarrow^*_{C^g_\sigma} \circ \rightarrow_{D^g_\sigma} \circ$$

because if there is one, then (assuming that the rules $R1$–$R2$ are not applicable) rule $R4$ would be applicable. For proof patterns arising out of non-overlaps, we can get new proof patterns by commuting the proof steps (modulo C-steps). Similarly, since rule $R3$ is not applicable to the state (D_σ, C_σ), therefore there are no non-trivial proof patterns

$$\circ \leftarrow_{D_\sigma^g} \circ \leftrightarrow^*_{C_\sigma^g} \circ \rightarrow_{S_k^g} \circ.$$

Any such patterns arising out of non-overlaps can be eliminated by commuting the proof steps.

Property 5: Since S_i is substitution-feasible, let θ be the idempotent substitution corresponding to S_i. If $t_1 \leftrightarrow^*_{S_i^g} t_2$, then by Theorem 13 $t_1 \leftrightarrow^*_{S^g} t_2$, and therefore using theorem 14, the substitution θ is a syntactic unifier for t_1 and t_2. Let $\sigma' \preceq^V \theta$ be the most-general unifier. The proof of Theorem 15 shows that the substitution-feasible set $S_{i\text{new}} = S_i - \{t \rightarrow x \in S_i : x \in \mathcal{D}om(\sigma')\} \cup E_{\sigma'}$ is equivalent (modulo $\emptyset$ on V) to the substitution-feasible set S_i.

Now, S_i is equivalent to $S_{i\text{new}}$ and hence the idempotent substitution corresponding to these two sets is identical. Hence, since S_i is E-feasible, the set $S_{i\text{new}}$ is also E-feasible. Furthermore, if a substitution-feasible set S_i' is E-feasible, then so is any subset of it (with respect to the same set of variables). □

Lemma 24 *Let $\xi_1 = (K_1, E_1; V_1, E?_1; S_1)$ be a state with solution feasible set S^1. If $E?_1 \neq \emptyset$, then there exists a smaller problem state (ξ_2, S^2) that can be obtained using either an REU transition step or the transition rules $R1$–$R4$.*

Proof. We show how to get a simpler problem instance $\langle \xi', (D', C') \rangle$ from a given one $\langle \xi, (D, C) \rangle$ such that $D' \cup C'$ is a solution for ξ'.

1. If *C-Closure* is *not* the last rule applied or *deletion* is applicable, then by an application of the respective rule we get the required smaller instance $\langle \xi', (D, C) \rangle$.

2. If some equation $s \approx t \in E?$ has a proof $\mathcal{P}$

   $$s \ \leftrightarrow^*_{(E \cup D \cup C)^g} \ t$$

   in which there is no step applied at the top redex, then the proof $\mathcal{P}$ can be rewritten as $C[\mathcal{P}]$ where $C[\_]$ is a non-trivial context. Clearly, we can apply *decomposition* to get a simpler instance ξ' of the rigid E-unification problem. The pair $\langle \xi', (D, C) \rangle$ is the desired tuple.

3. If case 1 is not applicable, then we can assume that the set E is fully-reduced, and the right-hand side of every rule in E is a constant in K and no left-hand side is a constant in K. If any of the rules *R1*, *R2*, *R3* or *R4* are applicable to the state (D, C), then, we know we can use these rules to get a *smaller* state (D', C'). As seen above, the new set $D' \cup C'$ is E-feasible and equivalent (modulo E on V) to $D \cup C$. Thus, the pair $\langle \xi, (D', C') \rangle$ is the desired tuple.

4. If there is a non-trivial proof pattern of the following form:

$$\circ \leftarrow_{E^g} \circ \leftrightarrow^*_{C^g} \circ \rightarrow_{E^g} \circ$$

then there exist two rules $t_1 \approx c_1, t_2 \approx c_2 \in E$ such that $t_1 \leftrightarrow^*_{C^g} t_2$. Now, using property 5 in Lemma 23, we know that we can use *superposition* on the current state to get a new state $\xi' = (K', E'; V', E?'; S')$.

The solution for this new smaller instance is obtained thus: Let θ^* be the most-general unifier for t_1 and t_2. Define $R' = R_{\theta_{\text{new}}} - E_{\theta^*}$. The substitution-feasible set R' is trivially E'-feasible, and is also a solution for ξ'. Hence, the pair $\langle \xi', (R', \emptyset) \rangle$ is the required smaller pair.

5. Next assume that case 3 is not applicable. Suppose there exists a proof $\mathcal{P}$ in $E \cup D \cup C$ for $s \approx t \in E?$, of the form

$$s = C[t'] \ \rightarrow^*_{D^g/C^g} \ C[t] = s_1 \ \rightarrow_{E^g} \ C[c] \ \leftrightarrow^*_{(E \cup R)^g} \ t.$$

It follows from property 5 of Lemma 23 that we can apply *narrowing* (as t' and t are unifiable by a feasible substitution) to get a new smaller state ξ'. As in case 4, we can construct a new solution R' for ξ'.

6. Under the assumption that all the previous cases are not applicable, suppose there exists a proof in $E \cup D \cup C$ for $s \approx t \in E?$, of the form

$$s \ \leftrightarrow^+_{(D \cup C)^g} \ t.$$

By theorem 14, it follows that s and t are syntactically unifiable. Since case 2 is not applicable, either s or t is a variable. Moreover since $D \cup C$ is E-feasible, so is the substitution arising in *elimination*. Hence, we can apply *elimination* to get a smaller problem instance ξ'. Accordingly we can obtain the new solution R' as we did in the last two cases.

Assume that none of the above mentioned cases are applicable to the given $(\xi, \langle D, C\rangle)$ pair. We claim that $E? = \emptyset$, and hence, the state is a final state. Suppose not. Let $s \approx t \in E?$. Since case 1 is not applicable, the set E is fully-reduced ground convergent. Since $D \cup C$ is a solution, therefore,

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

Since case 4 is not applicable, we can eliminate all patterns of the form,

$$\circ \leftarrow_{E^g} \circ \leftrightarrow^*_{C^g} \circ \rightarrow_{E^g} \circ$$

by commuting the proof steps. Similarly, since case 3 is not applicable, therefore by property 4 in Lemma 23, there are no patterns of the form,

$$\circ \leftarrow_{E^g} \circ \leftrightarrow^*_{C^g} \circ \rightarrow_{D^g} \circ \quad \text{or,} \quad \circ \leftarrow_{D^g} \circ \leftrightarrow^*_{C^g} \circ \rightarrow_{D^g} \circ$$

Hence, there is a proof for $s \approx t$ of the form $s \rightarrow^*_{(R\cup D)^g/C^g} \circ \leftarrow^*_{(R\cup D)^g/C^g} t$. Because of case 5, there are no E-steps in the above proof, and hence it can only be of the form $s \leftrightarrow^*_{D^g/C^g} t$. And because of case 6, it is not of the form $s \ \leftrightarrow^+_{(D\cup C)^g} \ t$. This means that the proof is trivial and $s = t$, but, this is impossible since case 1 would be applicable otherwise. □

7.2.4 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 $f(\ldots) \rightarrow c$ 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). Further details can be looked up in [14].

7.2.5 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 [46], 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 [46] where the dependence on orderings complicates the unification process.

In [16], 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 *C-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 7.2.4 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 [55] 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 $gfffgfffx$ 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 [11]. 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.

7.3 Shostak's Combination Method Revisited

In this section, we revisit Shostak's combination method that was first discussed in Section 3.2. The intent here is to provide some more details by giving rules responsible for performing deductions over individual theories.

For simplicity, we consider the combination of (i) the pure theory of equality over the signature Σ_0 and (ii) an $\mathcal{E}_1$-theory over the signature Σ_1 (such that $\Sigma_0 \cap \Sigma_1 = \emptyset$). The rest of the assumptions and notations are as in Section 3.2. We focus our attention on the deductive aspects of Shostak's method, and hence ignore the disequations $s \not\approx t$ as they do not participate in the deductive inferences.

Since we describe the procedure in more detail, the state is now characterized by a quadruple (K, E, S, R), where K is a set of constant (more appropriately variable) symbols, and E, S and R are sets of equations. The initial state for a derivation is $(V, E, \emptyset, \emptyset)$ where E is a set of equations over $\Sigma_0 \cup \Sigma_1 \cup V$.

Deductions in the pure theory of equality are done via congruence closure, as outlined in Section 5.1. Therefore, we have $(K, E, S, R) \vdash_{Sho} (K', E', S, R')$ if $(K, E, R) \vdash_{CC} (K', E', R')$ using any of the congruence closure rules except *orientation* and *composition*. Note that *extension* creates rules that contain symbols from some Σ_i and K. The set R can always be partitioned into disjoint sets as $R_0 \cup R_1 \cup C$, where R_0 contains all D-rules over the signature $\Sigma_0 \cup K$, R_1 contains all D-rules over the signature $\Sigma_1 \cup K$, and C contains all C-rules.

We next focus on rules for deduction in the theory over Σ_1.

$$\textbf{Uni}\text{fication:} \qquad \frac{(K, E \cup \{s \approx t\}, \emptyset, R_0 \cup R_1 \cup C)}{(K, E, E_\sigma, R_0 \cup R_1 \cup C)}$$

if $s, t \in \mathcal{T}(\Sigma_i \cup K)$, and σ is the most-general $\mathcal{E}_1$-unifier of $s{\uparrow}_{R_1 \cup C}$ and $t{\uparrow}_{R_1 \cup C}$.[9]

[9]The requirement that the S-component be empty is not part of Shostak's method. We introduce it here as without it we get an incomplete set of rules (for example, consider $y \approx 1 \in S$ and $y \approx 2 \in E$). Shostak's description contains this subtle bug.

Note that no variable (constant) occurring in the unification problem $s{\uparrow_{R_1 \cup C}} \approx^{?} t{\uparrow_{R_1 \cup C}}$ can occur as right-hand side of any rule in R_1. If these two terms are not $\mathcal{E}_1$-unifiable, then we stop in a failure state $\perp$. Intuitively, if we think of R_1 as a set of equations representing a substitution, then the term $s{\uparrow_{R_1}}$ denotes the term obtained by applying this substitution to the term s.

Equations in S can be simplified and oriented using the standard congruence closure rules. Hence, we also have $(K, E, S, R) \vdash_{Sho} (K', E, S', R')$ if $(K, S, R) \vdash_{CC} (K', S', R')$ using any congruence closure rule except *deduction*.

$$\textbf{E-Col}\text{lapse:} \qquad \frac{(K, E, S, R \cup \{s \to c\})}{(K, E, S \cup \{s{\uparrow_{R_1 \cup C}} \approx c\}, R \cup \{s \to c\})}$$

if $s \in \mathcal{T}(\Sigma_1 \cup K)$.

$$\textbf{E-Sim}\text{plification:} \qquad \frac{(K, E[s], S[s], R)}{(K, E[s'], S[s'], R)}$$

if s occurs in either E or S and $s \to^{!}_{\mathcal{E}_1} s'$.

Example 19 *Consider the formula*

$$z \approx f(x - y) \wedge x \approx z + y \wedge y \not\approx x - f(f(z))$$

in the combination of the theory of real linear arithmetic and the pure theory oof equality. We have $\Sigma_0 = \{f\}$ *and* $\Sigma_1 = \{+, -\}$*. We apply the transition rules to the initial state* $(\{x, y, z\}, E_0 = \{z \approx f(x - y), x \approx z + y\}, \emptyset, \emptyset)$*. In Table 13, we show some of the important intermediate steps of a derivation that closely mimics Shostak's combination procedure.*

7.3.1 Correctness

It is straight forward to verify that each transition rule is sound in the sense that any equation added to the E-, S- or R-component is, in fact, a consequence of the equations present in these components and the theory of interest. Proposition 3 suggests that the abstract set of transition rules presented in Section 3.2 are complete. Hence, in order to prove that the new set of rules outlined here are complete, we only need to ensure that if there is an equality $x \approx y$ that is implied by the set $R_1 \cup C$ (and the theory under consideration), then it is added to the C-component.

i	*Constants* K_i	*Equations* E_i	*Equations* S_i	*Rules* R_i	
0	$\{x, y, z\}$	E_0	$\emptyset$	$\emptyset$	
1	$K_0 \cup \{x_1, x_2\}$	$\{z \approx x_2,$ $x \approx z + y\}$	$\emptyset$	$\{x - y \to x_1,$ $fx_1 \to x_2\}$	**Ext** **Ext**
2	K_1	$\{x \approx z + y\}$	$\{x_2 \approx z\}$	$\{x - y \to x_1,$ $fx_1 \to x_2\}$	**Uni**
3	K_1	$\{x \approx z + y\}$	$\emptyset$	$R_2 \cup \{x_2 \to z\}$	**Ori**
4	K_1	$\emptyset$	$\{x_2 + y \approx x\}$	R_3	**Uni**
5	K_1	$\emptyset$	$\emptyset$	$R_3 \cup \{x_2 + y \to x\}$	**Ori**
6	K_1	$\emptyset$	$\{x_2 + y - y \approx x_1\}$	R_5	**E-Col**
7	K_1	$\emptyset$	$\{x_2 \approx x_1\}$	R_5	**E-Sim**
8	K_1	$\emptyset$	$\emptyset$	$\{x - y \to z, fz \to z,$ $z + y \to x\}$	**Cong** **Clo**

Table 13: Example: Intermediate states in a derivation illustrating the Shostak's combination approach.

First, since we are interested in proving completeness, we remove redundant formulas to simplify our argument below. In particular, in *E-collapse* we remove the equation $s \to c$ from the set R_1 in the R-component. Furthermore, we consider a particular strategy in which (i) the set C is always eventually convergent, (ii) if any variable (constant) that occurs in $E \cup S \cup R_0 \cup R_1$ is not in C-normal form, then we simplify it to a C-normal form term, (iii) if any of the rules *simplification*, *composition*, *E-simplification* or *E-Collapse* is applicable, then it is applied *before* the application of any of the other rules.

The idea behind these assumptions is that the set R_1 of D-rules over $\Sigma_1 \cup K$ is always eventually a substitution-feasible set. In particular, the set R_1 can be written as

$$R_1 = \{t_1 \to x_1, t_2 \to x_2, \ldots, t_k \to x_k\}$$

where $x_1, x_2, \ldots, x_k$ are distinct variables and none of these variables occur in any of the terms $t_1, t_2, \ldots, t_k$. Hence, the set R_1 defines an idempotent substitution σ_{R_1} defined as $[x_1 \mapsto t_1, x_2 \mapsto t_2, \ldots, x_k \mapsto t_k]$. This is a consequence of the condition in *unification* which prohibits any variable that already occurs as a right-hand side term in R_1 to occur in the new equations generated by *unification*. Additionally, if one of the variables x_i ever occurs in the left-hand side of any other rule, then *E-collapse* eliminates it. It is also clear that R_1^{-1} is convergent.

Now, if $x \leftrightarrow^*_{\mathcal{E}_1 \cup R_1} y$, then $x\sigma_{R_1} \leftrightarrow^*_{\mathcal{E}_1} y\sigma_{R_1}$, which means there are two left-hand side terms in the set R_1 that are equal modulo $\mathcal{E}_1$. Therefore, $x\sigma_{R_1} \rightarrow^!_{E_1} \circ \leftarrow^!_{E_1} y\sigma_{R_1}$. In this case, *E-collapse* and *E-simplification* rules will ensure that the equality $x \approx y$ is deduced. This shows that all possible equalities between variables will be deduced in a derivation satisfying the conditions mentioned earlier.

7.3.2 Summary

We presented a set of transition rules that capture the essence of the deductive aspects of Shostak's combination procedure. One important observation is that if suitable redundant rules are eliminated, then the set of D-rules R_1 over the signature $\Sigma_1 \cup K$ is always a substitution-feasible set. This observation considerably simplifies the understanding of Shostak's combination method and also allows us to correctly formulate the transition rules like *unification*.

This concludes our digress to discuss applications of abstract congruence closure.

Chapter 8

Gröbner Basis Methods

We next discuss the theory of polynomial ideals, and present a set of transition rules that describe the construction of Gröbner basis for a polynomial ideal over a commutative Noetherian ring in a proof-theoretic setting. The results contained in this chapter establish that the theory of polynomial rings satisfies the conditions outlined in Chapter 3, and hence it can be integrated with the other theories discussed earlier. Additionally, this chapter illustrates the second aspect of integration—use of constraints to separate non-logical and domain specific aspects of an algorithm from other abstract syntactic operations. In particular, constraints provide a clean way of separating the two modes of computation that are employed in the construction of polynomial ideal bases: rewriting (or reduction) on the polynomial ring and computations with ideals in the coefficient domain (e.g., computation of the intersection of finitely generated ideals).

Buchberger's algorithm for constructing a *Gröbner basis* for polynomial ideals over a field is one of the central methods in computational algebra [24, 26]. It has been extended in two directions: to polynomials over coefficient domains other than fields and to non-commutative polynomial rings, cf. [17]. For instance, Kandri-Rody and Kapur [50] and Pan [66] considered polynomials over Euclidian rings and principal ideal domains, respectively.

A Gröbner basis for an ideal I induces a terminating rewrite relation on polynomials, such that two polynomials have the same normal form if, and only if, they belong to the same remainder class modulo I. The algorithm proposed by Pan [66] constructs a weak Gröbner basis (D-basis), whereby any polynomial that belongs to the ideal I (and only these polynomials) can be reduced to zero. (Given a weak

Gröbner basis, one can determine whether p and q are equivalent modulo I by checking whether their difference $p - q$ can be rewritten to zero.) In this chapter, we present a procedure for constructing weak Gröbner bases for polynomials over more general Noetherian rings (Section 8.3) and show how one can obtain a Gröbner basis from a weak basis in the next chapter. Further, we argue that the conditions on the coefficient domain that allow effective computation of the weak Gröbner basis are the weakest possible (Section 8.7). These conditions can also be lifted from rings B to $B[X]$ (Section 8.8), thus providing a framework to obtain a hierarchic Gröbner basis computation algorithm for an arbitrary polynomial rings $B[X_1][X_2]\dots[X_n]$, which is isomorphic to the ring $B[X_1, \cdots, X_n]$.

We describe our method in proof-theoretic terms, as an abstract completion procedure in the sense of Bachmair and Dershowitz [5, 7]. The presentation is by transition (or inference) rules operating on sets of equations and rewrite rules. Transitions describe basic steps for changing the presentation of the underlying equational theory. If the rules are applied until they can be applied no more, we obtain a weak Gröbner basis. The correctness of the approach is proved by proof simplification arguments: we show that any proof of $p \approx 0$ can be normalized to a rewrite proof. Formally, our method is a constraint-based associative-commutative completion procedure, where all algebraic computations on the coefficient domain are embedded via constraints.

Bachmair and Ganzinger [8] describe Buchberger's original algorithm for constructing a Gröbner basis for a polynomial ideal over a field in a proof-theoretic setting similar to the one used here. But in the case of polynomials over fields, a considerably optimized and simplified procedure is possible that, for instance, does not require the flexibility of the full constraint formalism. The addition of constraints gave rise to a number of technical problems, which required us to redesign some transition rules and to add new ones, while also changing the term signature slightly. On the other hand, constraints helped in incorporating certain optimizations, related to restricting the number of superpositions that need to be considered.

Weak Gröbner bases can be extended to strong Gröbner bases under some additional assumptions on the coefficient domain. We shall state these conditions and also argue that these assumptions are the minimal possible required for the extension. We also show that assumptions on the coefficient domain can be lifted from a ring B to the ring $B[X]$.

In this chapter, we require that the theory of term rewriting and completion [5, 7]

be extended to include constraints. The integration of constraints is fairly straightforward and fits into the framework proposed by Kirchner, Kirchner and Rusinowitch [52] for constraint-based completion.

8.1 Polynomials

We first outline how polynomials may be represented as (variable-free) first-order terms over a many-sorted signature with sorts *Coef* (coefficients), *PProd* (power products) and *Poly* (polynomial expressions), see also Bündgen [28] and Bachmair and Ganzinger [8].

8.1.1 Coefficients

We assume that the coefficient domain is a commutative ring with unit, specified by some algebra B over a given signature such that every element of B is represented by some ground term (i.e., B is term-generated). The specific term representation of coefficients is not significant for our purposes and may vary with the coefficient domain, but we assume that the signature contains at least the declarations

$$\begin{array}{rrcl} 0,1: & & \to & \mathit{Coef} \\ -: & \mathit{Coef} & \to & \mathit{Coef} \\ +,\cdot: & \mathit{Coef} \times \mathit{Coef} & \to & \mathit{Coef} \end{array}$$

Variables of sort *Coef*, also called *coefficient variables*, are denoted by the letters x, y, and z (possibly with subscripts or other annotation). We also use (meta-variables) a, b, c, and d to denote ground terms of sort *Coef*, also called *coefficient terms*.

We will simplify the notation, writing, for instance, ax instead of $a \cdot x$, or using summations, $\sum_{j=1}^{k} a_j x_j$. We also use vector notation: $\vec{a}$ and $\vec{b}$ denote vectors of coefficient terms, e.g., $(a_1, \ldots, a_k)$, whereas $\vec{x}$, $\vec{y}$, and $\vec{z}$ denote *transposed* vectors (of dimension $k \times 1$) of coefficient variables. Thus $\vec{a} \cdot \vec{x}$ also denotes a sum $a_1 \cdot x_1 + a_2 \cdot x_2 + \cdots a_k \cdot x_k$.

We define B (the "base theory") as the set of all ground equations $a \approx b$ such that a and b denote the same element in B. We need to be able to decide whether a given equation $a \approx 0$ belongs to B, and hence the ground theory B must be decidable. The construction of ideal bases depends on additional properties of the coefficient domain, which will be given later.

8.1.2 Polynomial expressions

Polynomial expressions are constructed from coefficients, power products and function symbols representing ring operations. We use the following signature:

$$\begin{array}{rrcl} I, X_1, \ldots, X_n : & & \to & \mathit{PProd} \\ \cdot : & \mathit{PProd} \times \mathit{PProd} & \to & \mathit{PProd} \\ M : & \mathit{Coef} \times \mathit{PProd} & \to & \mathit{Poly} \\ \Omega : & & \to & \mathit{Poly} \\ \oplus, \otimes : & \mathit{Poly} \times \mathit{Poly} & \to & \mathit{Poly} \end{array}$$

Variables of sort *Poly* are denoted by the letters α, β, and γ; variables of sort *PProd* by ν and μ. The constants X_i are called *indeterminates*. Terms of sort *PProd* are called *power products* and denoted by s, t and u; terms of sort *Poly*, which are also called *polynomial expressions*, are denoted by p, q, and r. Terms of the form $M(a, t)$ are called *monomials*.

We will use standard algebraic notation for power products, monomials, and polynomial expressions. For instance, for our purposes it is sufficient to characterize power products up to associativity and commutativity, and hence we write, say, X^3Y. The symbol M will usually also be represented by juxtaposition. For instance, we write $3X^2Y$ instead of $M(3, X \cdot (X \cdot Y))$.

8.1.3 Constraints

The construction of ideal bases requires different modes of computation for coefficients and polynomial expressions. Computation with coefficients requires algebraic operations such as intersection and union of ideals, whereas the manipulation of the non-coefficient parts of polynomials involves syntactical operations such as (associative-commutative) matching and unification and term replacement. A constraint-based formalism allows for a clean separation of the two parts.

The *simple constraints* we use are either *equality constraints* of the form $x = a$, $x = y + z$, $x = y \cdot z$, or $x = \sum_{j=1}^{k} a_j y_j$ ($x = \vec{a} \cdot \vec{y}$ in short); or *negated* equality constraints of the form $\mu \neq s\nu$. Thus, we have only simple constraints of the form $\mu \neq s\nu$ on variables and terms of sort *PProd*. All other constraints are on terms of sort *Coef*. A *constraint* is a conjunction of (zero or more) simple constraints.[1] We

[1] Thus, as special cases we have (i) the empty constraint and (ii) constraints consisting of a single simple constraint.

use the matrix notation $\vec{x} = \mathbf{A} \cdot \vec{y}$ to represent a conjunction of constraints $x_i = \vec{a_i} \cdot \vec{y}$, for $i = 1, \ldots, l$, where $\mathbf{A}$ contains the vectors $\vec{a_i}$ as rows. In the same spirit, the conjunction of $\mu \neq t_i\nu$ will be compactly written as $\mu \notin \vec{t\nu}$. The letters C and D are used to denote constraints.

By a *solution* of a constraint we mean a substitution σ that assigns ground terms to coefficient variables, and ground power products to variables of sort *PProd*, such that: (i) in the case of an equality constraint, both terms denote the same element of the coefficient domain B; and, (ii) in case of the constraint $\mu \neq t\nu$, $\mu\sigma \not\rightarrow^*_{AC} t\nu\sigma'$ for any substitution σ', where AC denotes the associativity and commutativity of $\cdot$ operator. In other words, the inequality constraint is satisfiable if μ is mapped to a power product that is not a multiple of t. Any substitution is a solution of the empty constraint.

A *constrained equation* is an expression $p \approx q$ `if` C, where p and q are polynomial expressions and C is a constraint. If C is empty, we simply write $p \approx q$. By an (*unconstrained*) *instance* of $p \approx q$ `if` C we mean any equation $p\sigma \approx q\sigma$, where the restriction of σ to coefficient variables and power product variables is a solution of the constraint C. By the ground theory represented by a set of constrained equations we mean the equational theory induced by the set of all its unconstrained ground instances.

We shall enforce a sharp distinction between coefficient and non-coefficient terms by transforming expressions via *variable abstraction*. That is, we replace a non-variable coefficient subterm a in a polynomial expression by a new coefficient variable x defined by a simple constraint $x = a$.

For example, the polynomial equation $3X^2Y \oplus (-2)XY^2 \approx \Omega$ is transformed to

$$uX^2Y \oplus vXY^2 \approx \Omega \quad \texttt{if} \;\; u = 3,\; v = -2.^2$$

Evidently, the ground instances of the new equation are equivalent (up to the representation of coefficients) to the ground instances of the original equation.

8.1.4 Polynomials

Let us next discuss the equational theory of polynomials. By H we denote the set of equations

$$\begin{array}{lcll} x\nu \oplus y\nu & \approx & z\nu & \texttt{if} \quad z = x + y \\ x\mu \otimes y\nu & \approx & z\mu \cdot \nu & \texttt{if} \quad z = x \cdot y \end{array}$$

[2] Conjunctions of simple constrained are usually written as sequences.

that relate addition and multiplication of polynomials to the corresponding operations on coefficients and power products; by AC we denote the set of associativity and commutativity axioms,

$$\begin{array}{rcl@{\qquad}rcl} \alpha \oplus (\beta \oplus \gamma) & \approx & (\alpha \oplus \beta) \oplus \gamma & \alpha \oplus \beta & \approx & \beta \oplus \alpha \\ \alpha \otimes (\beta \otimes \gamma) & \approx & (\alpha \otimes \beta) \otimes \gamma & \alpha \otimes \beta & \approx & \beta \otimes \alpha \\ \mu \cdot (\mu' \cdot \mu'') & \approx & (\mu \cdot \mu') \cdot \mu'' & \mu \cdot \mu' & \approx & \mu' \cdot \mu \end{array}$$

and, by R the set of ring axioms,

$$\begin{array}{rcll} \mu \cdot I & \approx & \mu & \\ \alpha \oplus \Omega & \approx & \alpha & \\ \alpha \oplus \alpha \otimes xI & \approx & \Omega & \texttt{if} \quad x = -1 \\ \alpha \otimes xI & \approx & \alpha & \texttt{if} \quad x = 1 \\ \alpha \otimes (\beta \oplus \gamma) & \approx & (\alpha \otimes \beta) \oplus (\alpha \otimes \gamma) & \end{array}$$

The set $R \cup AC \cup H$ is an equational specification of polynomial monoid rings over commutative rings. We next present a rewrite system that defines unique normal forms for ground terms of sort *Poly*.

$$\begin{array}{rcll} \mu \cdot I & \to & \mu & (1) \\ x\nu & \to & \Omega & \texttt{if } x = 0 \quad (2) \\ \Omega \oplus \alpha & \to & \alpha & (3) \\ \Omega \otimes \alpha & \to & \Omega & (4) \\ x\nu \oplus y\nu & \to & z\nu & \texttt{if } z = x + y \quad (5) \\ x\mu \otimes y\nu & \to & z\mu \cdot \nu & \texttt{if } z = x \cdot y \quad (6) \\ \alpha \otimes (\beta \oplus \gamma) & \to & (\alpha \otimes \beta) \oplus (\alpha \otimes \gamma) & (7) \end{array}$$

Table 14: Convergent rewrite system presenting the theory of polynomials.

Let P be the (constrained) rewrite system[3] presented in Table 14. Then P^e consists of P plus the extended versions of all rules in P except (2). For instance, the extensions of rules (5) and (6) are

$$\begin{array}{rcll} x\nu \oplus y\nu \oplus \alpha & \to & z\nu \oplus \alpha & \texttt{if } z = x + y \\ x\mu \otimes y\nu \otimes \alpha & \to & z\mu \cdot \nu \otimes \alpha & \texttt{if } z = x \cdot y \end{array}$$

[3]We should point out that our notation closely resembles the one in [28]. The convergent rewrite system we use is much simpler, though, as we deal with coefficients through constraints, whereas Bündgen handles them by rewriting also.

(The extensions of rules (1), (3), and (4) are actually not needed, as these rules may serve as their own extensions, so to say.)

Proposition 6 *The rewrite system $AC\backslash P^e$ is ground convergent modulo $B \cup AC$ and the ground equational theories induced by $B \cup P \cup AC$ and $B \cup R \cup H \cup AC$ are the same.*

The uniqueness of normal forms up to equivalence under $B \cup AC$ can be shown easily. Termination of P^e can be established by showing that all rules are decreasing in a suitable associative path ordering (Bachmair and Plaisted [9]). Any precedence $\succeq$ will work in which (i) $a \sim b$, for all coefficients a and b, and, (ii) $I \succ \otimes \succ M \succ \oplus \sim \cdot \succ \Omega$ and all constants $X_1, \ldots, X_n$ precede I.[4] We denote this associative path ordering by $\succ_{apo}$ or simply $\succ$.

Proposition 7 *The ordering $\succ$ is a $(AC \cup B)$-compatible reduction ordering that contains all (ground instances of) rewrite rules in P.*

Proposition 6 shows that condition 1 of Chapter 3 is satisfied. We next show that convergent systems corresponding to a set of ground equations can be constructed in the theory of polynomial rings, thus establishing condition 2 as well. Ground terms (of sort *Poly*) in normal form correspond to (and will be called) *polynomials*. They can be written as sums of monomials $\sum_j a_j t_j$.

8.2 Polynomial Rules and Equations

By a *standardized (constrained) equation* we mean an equation of the form

$$\sum_i x_i t_i \mu \approx \Omega \quad \texttt{if} \quad \vec{x} = \mathbf{A} \cdot \vec{y},\ \mu \notin \vec{s}\nu \tag{1}$$

where $\mathbf{A}$ is a matrix of ground coefficient terms and the coefficient variables in $\vec{x}$ and $\vec{y}$ are all distinct; furthermore, as described earlier, $\vec{s}$ is a vector consisting of ground power products s_i. In other words, the coefficient variables x_i in a standardized equation are all constrained by (and only by) linear equations, and the power product variable μ is constrained by linear inequalities.

[4] Any precedence on the indeterminates will do. If all indeterminates are distinguished, we obtain an extension of a "lexicographic ordering," whereas an extension of the "total degree ordering" is obtained by declaring all indeterminates to be equivalent.

Polynomial equations can easily be converted to standardized form. For example, the equation $4X^2Y^3 \approx -7X^4Y$ can be written as (the extended equation) $x_1X^2Y^3\mu \oplus x_2X^4Y\mu \approx \Omega$ `if` $x_1 = 4y, x_2 = 7y$, where the coefficient domain is the ring of integers. We will assume that all equations have been thus standardized before the start of the completion procedure.

By a (constrained) *polynomial rule*, denoted by ρ, we mean an equation of the form $x_0t_0 \rightarrow \sum_i x_it_i$ `if` C, where $x_0t_0 \succ x_it_i$, for all i. Such a rule is said to be in *standardized form* if it can be written as

$$x_0t_0\mu \rightarrow \sum_i x_it_i\mu \quad \texttt{if} \quad x_0 = \vec{a}\cdot\vec{y},\ \vec{x} = \mathbf{A}\cdot\vec{y},\ \mu \notin \vec{s}\nu \tag{2}$$

where $\vec{a}$ is a vector of ground coefficient terms, $\mathbf{A}$ a matrix of ground coefficient terms, and the variables $x_0, x_1, \ldots, x_n$ and $y_1, \ldots, y_k$ are all distinct; furthermore, $\vec{s}$ is a vector of ground power product terms, the leading power product t_0 does not contain I as a subterm, and at most one monomial can be of the form $x_j\mu$ (this corresponds to the case when $t_j = I$). As a special case, rules of the form $x_0t_0\mu \rightarrow \Omega$ `if` $x_0 = \vec{a}\cdot\vec{y}, \mu \notin \vec{s}\nu$ will also be considered to be in standardized form (optionally t_0 may not be present too). In the sequel, when we write a polynomial rule in the above form, we will mean to include these other cases (when some t_j is absent, or, the right-hand side is Ω) as well. We also call x_0 the *leading coefficient* and x_0t_0 the *leading monomial* of the rule.

As mentioned before, while performing associative-commutative completion, we need to consider extensions of rules. For example, technically, orienting an equation $2X \approx 1$ would yield $2X \rightarrow 1$, but, we use the extension $(2x)X\mu \rightarrow x\mu$. In other words, we need to add, for any polynomial rule

$$a_0t_0 \rightarrow \sum a_it_i$$

an *extended polynomial rule*

$$x_0t_0\mu \rightarrow \sum x_it_i\mu \quad \texttt{if} \quad x_0 = a_0y,\ x_i = a_iy$$

which, in general, takes the form 2 defined above. However, for simplicity, we assume we perform the required extensions on the equations itself and hence, we work on standardized equations of the form 1.

Suppose we want to reduce the monomial ct_0 using such a constrained rewrite rule. This is possible if x_0 can be "matched" with c, i.e., if the constraint $c = \vec{a}\cdot\vec{y}$ is solvable,

which is the case precisely if c is an element of the ideal $Id(a_1, \ldots, a_k)$ generated by the elements of $\vec{a} = (a_1, \ldots, a_k)$. Therefore, we call the constraint $x_0 = \vec{a} \cdot \vec{y}$ the *ideal membership constraint* of the polynomial rule, and sometimes write $x_0 \in Id(\vec{a})$. Once a solution for the ideal membership constraint has been obtained, the variables in $\vec{y}$ get instantiated, so that the remaining equality constraints can be trivially satisfied by instantiating the variables in $\vec{x}$ appropriately.

We can naturally associate with every constrained polynomial rule (and constrained equation), a constrained polynomial. Specifically, corresponding to the standardized polynomial rule ρ described above, we have the polynomial p,

$$x_0 t_0 \mu \oplus \sum_j (-x_j) t_j \mu \;\; \texttt{if} \;\; x_0 = \vec{a} \cdot \vec{y},\ \vec{x} = \mathbf{A} \cdot \vec{y},\ \mu \notin \vec{s}\nu.$$

If N is a set of polynomial rules or equations, then by $Id(N)$ we denote the ideal generated by ground instances of all polynomials associated with the rules or equations in N, i.e., $Id(N) = \{\sum_i q_i p_i : q_i \in B[X_1, \ldots, X_n], p_i$ is some ground instance of a polynomial which corresponds to some polynomial rule in $N\}$.

Definition 19 *Let E be a set of ground polynomial equations. A set of polynomial rules R is called a* weak Gröbner basis *(for E) if for every ground polynomial expression p we have $p \leftrightarrow^*_{B \cup R \cup H \cup AC \cup E} \Omega$, if and only if, $p \rightarrow^*_{AC \setminus (P^e \cup R)} \Omega$. Equivalently we can say $p \in \mathrm{Id}(E)$, if and only if, $p \rightarrow^*_{AC \setminus (P^e \cup R)} \Omega$.*

8.2.1 Computations on Coefficients

We assume that the coefficient domain B is a commutative ring with unit that satisfies the following conditions:

B1 (Decidability and solvability of ideal membership) Given coefficients a and $a_1, \cdots, a_k$, it is decidable whether $a \in Id(a_1, \cdots, a_k)$, and if so, elements $b_1, \cdots, b_k$ can be computed such that $a = \sum_{i=1}^k b_i a_i$.

B2 The intersection of any two finitely generated ideals in B is finitely generated.

B3 (Computation of intersection ideals) Given two finitely generated ideals I_1 and I_2, the generators of $I_1 \cap I_2$ can be computed.

B4 The ideal $\{x \in B | ax = 0\}$ is finitely generated, for all $a \in B$.

B5 (Computation of zero divisors) Given $a \in B$, the generators of the ideal $\{x|ax = 0\}$ can be computed.

Conditions (B2)-(B5) can also be replaced by the requirement that syzygies be effectively computable, see the discussion in section 8.7.

We have seen that condition (B1) is needed for the matching part of rewriting with polynomial rules. The computation of intersections of ideals will be needed when left-hand sides of rules have to be unified during critical pair computation. Finally, the condition about zero divisors is required when left-hand side of a polynomial rule is unified with that of P-rule (2).

8.3 Polynomial Completion

We use a completion approach for constructing weak Gröbner bases. We follow the methodology of Bachmair and Dershowitz [5, 7], and describe completion by transition rules operating on sets of (polynomial) equations and rules. Each set E of equations defines a set of (ground) proofs. Transition rules are designed to enable proof transformations that eventually yield certain normal-form proofs. More specifically, we have to transform ground proofs $p \leftrightarrow^*_{E \cup B \cup P \cup AC} \Omega$ to *rewrite proofs* of the form $p \rightarrow^*_{AC \setminus R \cup P} \Omega$. (The transition rules are also sound, in that the underlying equational theory does not change.)

We classify transition rules into four groups: (i) the *orientation rule* turns equations into directed rewrite rules; (ii) *deduction rules* eliminate subproofs that are not allowed in normal-form proofs; (iii) *constraint manipulation rules* are needed to standardize equations and to simplify constraints, and are crucial for efficiency and termination; and (iv) *simplification rules* induce additional proof transformations that are critical for effective application of the orientation rule.

8.3.1 Orientation

The transformation of unoriented equations into directed rewrite rules is one of the key steps in completion. In the context of polynomial equations the idea is simply to shift the leading monomial to the left-hand side of a rule and the remaining monomials to the right-hand side. The presence of zero divisors causes technical complications, which we handle via constraints. More specifically, we "split off" those instances for which the leading coefficient is zero.

Orientation:
$$\frac{N \cup \{x_0 t_0 \mu \oplus \sum x_i t_i \mu \approx \Omega \ \texttt{if} \ x_0 = \vec{a} \cdot \vec{y},\ \vec{x} = \mathbf{A} \cdot \vec{y},\ \mu \notin \vec{s}\nu\}}{N \cup \{x_0 t_0 \mu \to \sum x_i t_i \mu \ \texttt{if} \ C_1\} \cup \{\sum x_i t_i \mu \approx \Omega \ \texttt{if} \ C_2\}}$$

if $x_0 t_0 \mu \to \sum x_i t_i \mu$ if C_1, with C_1 defined as $x_0 = \vec{a} \cdot \vec{y},\ \vec{x} = -\mathbf{A} \cdot \vec{y},\ \mu \notin \vec{s}\nu$, is a standardized polynomial rule; and C_2 is defined as $\vec{a} \cdot \vec{y} = 0,\ \vec{x} = \mathbf{A} \cdot \vec{y},\ \mu \notin \vec{s}\nu$. The notation $-\mathbf{A}$ is used to denote the matrix obtained from $\mathbf{A}$ by replacing each entry a_{ij} by $-a_{ij}$. (One may also use a more conventional orientation rule without splitting, and describe splitting by a separate transition rule. The "split off" equation is a critical pair arising from the overlap between the oriented rule and P-rule (2).)

8.3.2 Deduction

The deduction rules form the core of completion. They ensure that undesirable subproofs can be eliminated. Since we are dealing with a special case of (constrained) associative-commutative completion, subproofs to be eliminated include "peaks" of the form $q \leftarrow_R p \to_{AC \setminus R} r$ and "cliffs" of the form $q \leftrightarrow_{AC} p \to_{AC \setminus R} r$, where R is the set of rewrite rules (including P rules) under consideration. The standard way of getting rid of these proof patterns is by critical pair computation (in the case of peaks) and by extended rules, see section 8.2 (in the case of cliffs), cf. Bachmair and Dershowitz [6]. Roughly speaking, critical pairs are equations $p \approx q$ obtained from "minimal" peaks called "overlaps."

An overlap between a polynomial rule $x_0 t_0 \mu \to \sum x_i t_i \mu \ \ \texttt{if} \ \ C$ and extension of P-rule (5) yields an *extended equation*

$$(x_0 + z_0) t_0 \mu \ \ \approx \ \ z_0 t_0 \mu \oplus \sum x_i t_i \mu \ \ \texttt{if} \ \ C. \tag{3}$$

However, we ignore these rules for this section.[5] Extended equations will not be explicitly deduced via transition rules, but instead are implicitly used in the deduction of critical pairs (see below).

In describing critical pairs, we will make use of some algebraic terminology. Subsequently we will assume that power products $s, t, \ldots$ do not contain I as a proper subterm. We say that s *properly divides* t if $s\nu$ AC-matches t, and that s *divides* t, written $s|t$, if either $s \leftrightarrow^*_{AC} t$, or s properly divides t. The *least common multiple* of

[5] Technically, this is the main reason why we obtain weak Gröbner bases rather than more general Gröbner bases.

s and t is a shortest power product divided by both s and t. Similarly, we can define the *greatest common divisor* of s and t. We denote it by $\gcd(s,t)$. By $\frac{\vec{t}}{t_0}$, we denote the vector whose i-th component is $t_i/\gcd(t_i,t_0)$.

An analysis of peaks involving polynomial rules (and their extensions) shows that two kinds of critical pairs need to be computed.

Definition 20 *Consider the two (standardized) polynomial rules*

$$\begin{array}{rcl} \rho_1 & : & x_0 s_0 \mu \to \sum_i x_i s_i \mu \quad \texttt{if} \quad x_0 = \vec{a}\cdot\vec{z},\ \vec{x} = \mathbf{A}\cdot\vec{z},\ \mu \notin \vec{s'}\mu' \\ \rho_2 & : & y_0 t_0 \nu \to \sum_j y_j t_j \nu \quad \texttt{if} \quad y_0 = \vec{b}\cdot\vec{z'},\ \vec{y} = \mathbf{B}\cdot\vec{z'},\ \nu \notin \vec{t'}\nu' \end{array}$$

and let u be the least common multiple of the two power products s_0 and t_0, and s_0' and t_0' be power products such that $u \leftrightarrow^*_{AC} s_0 s_0' \leftrightarrow^*_{AC} t_0 t_0'$.[6] *If s_0' is not a multiple of any of the power products in $\vec{s'}$, and t_0' is not a multiple of any of the power products in $\vec{t'}$, then the set $\xi_{int}(\rho_1,\rho_2)$*

$$\begin{array}{l} \{\sum_i x_i(s_i s_0')\mu \ \oplus\ \sum_j y_j(t_j t_0')\mu \ \approx\ \Omega \\ \qquad\qquad \texttt{if} \quad \vec{a}\cdot\vec{z} = \vec{b}\cdot\vec{z'},\ \vec{x} = \mathbf{A}\cdot\vec{z},\ \vec{y} = -\mathbf{B}\cdot\vec{z'},\ \mu \notin \frac{\vec{s'}}{s_0'}\nu,\ \mu \notin \frac{\vec{t'}}{t_0'}\nu'\} \end{array}$$

is called an intersection critical pair equation set*; and, additionally if* $\mathrm{Id}(\vec{a}) \not\subset \mathrm{Id}(\vec{b})$ *and* $\mathrm{Id}(\vec{b}) \not\subset \mathrm{Id}(\vec{a})$*, then the set $\rho_{union}(\rho_1,\rho_2)$*

$$\begin{array}{l} \{x_0(s_0 s_0')\mu \ \to\ \sum_i x_i(s_i s_0')\mu \ \oplus\ \sum_j y_j(t_j t_0') \\ \qquad\qquad \texttt{if} \quad x_0 = \vec{a}\cdot\vec{z} + \vec{b}\cdot\vec{z'},\ \vec{x} = -\mathbf{A}\cdot\vec{z},\ \vec{y} = -\mathbf{B}\cdot\vec{z'},\ \mu \notin \frac{\vec{s'}}{s_0'}\nu,\ \mu \notin \frac{\vec{t'}}{t_0'}\nu'\} \end{array}$$

is called a union critical pair rule set*. In other cases these sets are defined to be empty.*

We speak of "intersection" and "union" critical pairs because we consider the intersection of the two ideals, $Id(\vec{a}) \cap Id(\vec{b})$, and their union $Id(\vec{a}) \cup Id(\vec{b})$ in the process of computing the respective rules. The computation of the least common multiple of two power products formally corresponds to (a special case of) AC-unification. (Note that both of the above critical pairs may have to be computed even when the two power products, s_0 and t_0, share no common indeterminates, because, the coefficients may still overlap, so to say.) Union critical pairs are obtained from overlaps between a (extended) polynomial rule and an *extended equation* defined in equation 3.

[6] *If $s_0' = I$, then $s_0 s_0'$ is replaced by s_0; and similarly for $t_0 t_0'$.*

The computation of critical pairs is achieved by a deductive transition rule:

$$\textbf{Superposition:} \quad \frac{N \cup \{\rho_1, \rho_2\}}{N \cup \{\rho_1', \rho_2'\} \cup \rho_{\text{union}}(\rho_1, \rho_2) \cup \xi_{\text{int}}(\rho_1, \rho_2)}$$

where, assuming ρ_1 and ρ_2 are defined as in Definition 20,
(1) whenever $\rho_{\text{union}}(\rho_1, \rho_2) \neq \emptyset$, ρ_1' is obtained from ρ_1 by adding the constraint $\mu \neq s_0'\mu''$, and, similarly ρ_2' is obtained from ρ_2 by adding the constraint $\nu \neq t_0'\nu''$; and,
(2) whenever $\rho_{\text{union}}(\rho_1, \rho_2) = \emptyset$, the rule ρ_1' is the same as ρ_1 *if* either (i) $Id(\vec{b})$ is a proper subset of $Id(\vec{a})$, or, (ii) $Id(\vec{a}) = Id(\vec{b})$ and s_0 is smaller than t_0, or, (iii) $Id(\vec{a}) = Id(\vec{b})$ and $s_0 \leftrightarrow_{AC}^* t_0$ and either the right-hand side of rule ρ_1 is smaller than the right-hand side of rule ρ_2, or, the rule ρ_2 is larger (say, has a larger time-stamp) than ρ_1.

If the union critical pair equation set is empty, then one of the two rules ρ_1 or ρ_2 can be used to reduce all instances of $xlcm(s_0, t_0)$ where $x \in Id(\vec{a}) \cup Id(\vec{b})$. This rule, so to say, acts like the union critical pair and hence, in this case, we don't add additional constraints to it. In the special case when s_0 divides t_0, and $Id(\vec{b}) \subseteq Id(\vec{a})$, then, we can *collapse* the rule ρ_2 (by ρ_1). We note that in this case, we add the constraint $\mu \neq I\nu$, which is unsatisfiable. Hence the above formulation subsumes the *collapse* rule. The other technical side conditions in the superposition rule ensure that we always delete the "larger" rule when collapsing.

A few remarks on this formulation of the superposition rule are in order here. First, adding extra inequality constraints (of the form $\mu \notin \vec{s}\nu$) on the power product variable helps in optimizing the completion procedure by limiting the number of critical pairs that will be computed later. This is because, in essence, adding new constraints simply means that certain rules from the set N are being deleted. Secondly, technically one could delete these instances only *after* the union critical pair was added as a rule, for otherwise, the deleted instances would not have simpler proofs in the new set N.

The optimization suggested above by the added constraints uses information from power products. We could also incorporate certain other optimizations based on the leading coefficients. For example, if the intersection ideal $Id(\vec{a}) \cap Id(\vec{b})$ is generated by the set $\{a_i b_j\}$, and s_0 and t_0 share no common indeterminates, then, we don't need to compute the intersection critical pair (corresponds to non-overlap), and we could define $\xi_{\text{int}}(\rho_1, \rho_2)$ to be empty. We might still need to compute the union critical pair.

8.3.3 Constraint Manipulation

The computation of the intersection critical pair equation, as defined above, involves only syntactic constraint transformations, but results in non-standardized equations, and potentially unsatisfiable constraints. Standardization essentially amounts to solving certain constraints so as to make the orientation rule applicable, and identifying unsatisfiable constraints to delete certain polynomial rules. In other words, our formalism allows for "lazy evaluation" of constraints. (While we have chosen to require constraint solving before orientation, other ways of constraint handling are conceivable.)

Converting the constraints into standardized form involves simple algebraic manipulations. It should be pointed out that the conditions (B2) and (B3) are needed to make sure that we can carry out the computations needed to standardize the constraints in the intersection critical pair equation. Moreover, along with the additional conditions (B4) and (B5) we can standardize the equation produced in the orientation rule.

The union critical pair rule is actually already in standardized form. This is because the constraint $x_0 = \vec{a} \cdot \vec{z} + \vec{b} \cdot \vec{z'},\ \vec{x} = \mathbf{A} \cdot \vec{z},\ \vec{y} = \mathbf{B} \cdot \vec{z'},\ \mu \notin \vec{s'}\nu,\ \mu \notin \vec{t'}\nu'$ can be rewritten as

$$x_0 = [\vec{a}, \vec{b}] \cdot \begin{bmatrix} \vec{z} \\ \vec{z'} \end{bmatrix},\ \begin{bmatrix} \vec{x} \\ \vec{y} \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{B} \end{bmatrix} \cdot \begin{bmatrix} \vec{x} \\ \vec{y} \end{bmatrix},\ \mu \notin [\vec{s'}, \vec{t'}]\nu,$$

where $[\vec{a}, \vec{b}]$ denotes the vector (of dimension $1 \times (m+n)$) obtained by concatenating the vectors $\vec{a}$ and $\vec{b}$ (of dimension $1 \times m$ and $1 \times n$ respectively).

Now suppose we add a constraint $\vec{b} \cdot \vec{y} = 0$ to a standardized constraint $\vec{x} = \mathbf{A} \cdot \vec{y}$, where $\vec{y}$ contains the variables $y_1, \ldots, y_k$. To transform the extended constraint to standardized form we *change variables* as follows: (i) If $k = 1$ (i.e. if the leading coefficient is constrained by membership in a principal ideal), then assuming the $1 \times l$ vector $\vec{c}$ generates the ideal $\{x : b_1 x = 0\}$, the constraint $\vec{x} = \mathbf{A} \cdot \vec{y}$ is replaced by $\vec{x} = (\mathbf{A} \cdot \vec{c}) \cdot \vec{z}$, where $\vec{z}$ is a $l \times 1$-dimension vector of new variables, and $\mathbf{A} \cdot \vec{c}$ denotes usual matrix multiplication. This process can be seen as replacing variable y_1 by $\vec{c} \cdot \vec{z}$. (ii) If $k > 1$, then we first compute the intersection of the two ideals $I_1 = Id(a_1, \ldots, a_{k-1})$ and $I_2 = Id(a_k)$, say $I = I_1 \cap I_2 = Id(c_1, \ldots, c_l)$. Then, for each j with $1 \leq j \leq l$, there exist coefficients $d_{i,j}$, such that $c_j = a_1 d_{1,j} + \ldots + a_{k-1} d_{k-1,j} = a_k(-d_{k,j})$. Replace in C each variable y_i by $d_{1i} z_1 + \ldots + d_{li} z_l$. In other words, if $\mathbf{D} = (d_{ij})$ is the $k \times l$ matrix, then we are replacing $\vec{y}$ by $\mathbf{D} \cdot \vec{z}$.

For any non-standardized constraint C (that can be produced during completion) a standardized constraint $\Theta(C)$ can be obtained [12], such that $p \approx 0$ `if` C and $p \approx 0$ `if` $\Theta(C)$ have the same (unconstrained) ground instances.

Constraint solving: $$\frac{N \cup \{p \approx \Omega \ \texttt{if} \ C\}}{N \cup \{p \approx \Omega \ \texttt{if} \ \Theta(C)\}}$$

In addition to constraint solving, there are various mechanisms for constraint simplification, including

- deletion of equations with unsolvable constraints: For example, we could get unsatisfiable constraints of the form $C \wedge \mu \neq \nu$. Eager application of this rule is crucial for termination of the completion procedure.
- simplification of membership constraints: For example, if the leading ideal of a rule is generated by a different set of generators, then we can change variables accordingly. Rather than formally describe the rule, we illustrate it below. Since application of this rule deletes certain instances, we restrict it and apply this rule only in conjunction with the superposition rule (i.e. we apply this rule to the union critical pair rule *before* adding it to the set N).
- deletion of redundant parts of constraints.

For example, consider the two rules $3X \to 1$ and $2X \to 3$ in the polynomial ring $\mathcal{Z}[X]$ over the integers. In our notation, the rules are written as

$$\begin{aligned} x_0 X\mu &\to x_1\mu \quad \texttt{if} \quad x_0 = 3z,\ x_1 = z \\ y_0 X\mu &\to y_1\mu \quad \texttt{if} \quad y_0 = 2z,\ y_1 = 3z \end{aligned}$$

The union critical pair rule corresponding to these two rules is

$$z_0 X\mu \to z_1\mu \oplus z_2\mu \ \ \texttt{if} \ \ z_0 = 3z_1' + 2z_2',\ z_1 = z_1',\ z_2 = 3z_2'.$$

Rather than adding this rule directly to the set N, we first choose to do some constraint simplification. Since the integers are a principal ideal domain we can express the ideal membership constraint in terms of a principal ideal. We have $Id(2,3) = Id(1)$ or, more specifically, $3z_1' + 2z_2' = 1$ if $z_1' = 1$ and $z_2' = -1$. Thus we substitute z' for z_1' and $-z'$ for z_2' and obtain a simplified constrained equation

$$z_0 X\mu \to z_1\mu \oplus z_2\mu \ \ \texttt{if} \ \ z_0 = z',\ z_1 = z',\ z_2 = -3z',$$

which can be simplified (using P-rules as described in the next section) into $z_0 X\mu \rightarrow z_1\mu$ if $z_0 = z'$, $z_1 = -2z'$. Similarly, the intersection critical pair equation would eventually get oriented into $z_0\mu \rightarrow \Omega$ if $z_0 = 7z'$. The original equations are both deleted because we add the unsatisfiable constraint $\mu \neq \nu$ to both of them.

8.3.4 Simplification

Mechanisms for simplifying equations and rules are crucial for the efficiency of completion. In the present case, *superposition* implicitly takes care of *collapse*.

We note that instances of polynomial rules ρ_i that do not conform to the inequality constraint $\mu \notin \vec{s}\nu$ can also be used to simplify other equations and rules. The additional inequality constraints only help to cut down on the number of superpositions that need to be done. As a general principle, in any completion procedure, we can always use rules that were once present in some N_i (but are now no longer present since they were simplified during the completion) for simplifications.

We say that a set R of polynomial rules is *left-reduced* if R contains no two rules

$$\rho_1 : \; x_0 s_0 \mu \rightarrow \sum_i x_i s_i \mu \quad \texttt{if} \quad x_0 = \vec{a} \cdot \vec{z},\; \vec{x} = \mathbf{A} \cdot \vec{z},\; \mu \notin \vec{s'}\mu'$$

$$\rho_2 : \; y_0 t_0 \nu \rightarrow \sum_j y_j t_j \nu \quad \texttt{if} \quad y_0 = \vec{b} \cdot \vec{z'},\; \vec{y} = \mathbf{B} \cdot \vec{z'},\; \nu \notin \vec{t'}\nu'$$

such that s_0 divides t_0, t_0/s_0 is not a multiple of any power product in $\vec{s'}$ and $Id(\vec{b}) \subseteq Id(\vec{a})$. If a set R is not left-reduced, then some rule can be "collapsed", as we described earlier, using *superposition*.

We can use rules in the set N to simplify terms occurring in polynomial equations.

Simplification:
$$\frac{N \cup \{yt\mu \oplus \sum_i y_i t_i \mu \approx \Omega \texttt{ if } y = \vec{b} \cdot \vec{z'} \wedge \vec{y} = \mathbf{B} \cdot \vec{z'} \wedge C, \rho_1\}}{N \cup \{\sum_j x_j s_j s'\mu \oplus \sum_i y_i t_i \mu \approx \Omega \texttt{ if } C', \rho_1\}}$$

if ρ_1 is as defined above; $s \leftrightarrow^*_{AC} s_0 s'$ for some power product s'; $Id(\vec{b}) \subset Id(\vec{a})$; $\vec{b} = \vec{a} \cdot \mathbf{A}'$; the constraint C' is $\vec{x} = (\mathbf{A} \cdot \mathbf{A}') \cdot \vec{z'} \wedge \vec{y} = \mathbf{B} \cdot \vec{z'} \wedge C$.

Other common simplification techniques include

- deletion of trivial equations,
- simplification of polynomial equations by rules in P,
- simplification of right-hand sides of polynomial rules.

These can be formulated as transition rules in a similar way as the *Simplification* rule.

8.3.5 Correctness

A *derivation* is a (possibly infinite) sequence of transition steps $N_0 \vdash N_1 \vdash N_2 \vdash \cdots$ where N_0 is a set of polynomial equations in standardized form. The *limit* N_∞ of the derivation is the set $\bigcup_i \bigcap_{j \geq i} N_j$ of all *persisting* equations and rules.

Definition 21 *A derivation is said to be* fair *if (i) N_∞ is a left-reduced set of polynomial rules with satisfiable constraints, and (ii) $\bigcup_i N_i$ contains all intersection and union critical pairs between (extended) rules in N_∞.*

The following technical lemma will be used in the proofs of termination and correctness.

Lemma 25 *Let N_∞ be the limit of a fair derivation. Suppose $x_0 t_0 \mu \oplus \sum x_i t_i \mu \approx \Omega$ `if` C is an equation in $\bigcup_i N_i$ such that $x_0 t_0 \succ x_i t_i$, for all i, and x_0 is constrained by an ideal membership constraint $x_0 \in I$. If the constraint set C is satisfiable, then there is a rule in N_∞ with left-hand side $y_0 u_0$ and ideal membership constraint $y_0 \in I'$, such that u_0 divides t_0 and $I \subseteq I'$.*

Intuitively, the lemma states that all leading monomials of polynomial equations can be reduced by persisting rules. This is a consequence of part (i) of the fairness conditions. (Polynomial equations do not persist. Thus the leading monomial is either reduced at some point, or else it will become the left-hand side of a rule when the equation is oriented.)

Theorem 19 (Termination) *Suppose the coefficient domain B is Noetherian and satisfies conditions (B1)-(B5). Then the limit N_∞ of any fair derivation is a finite set of polynomial rules.*

Proof. (By contradiction) Suppose N_∞ contains infinitely many polynomial rules. Let $x_n t_n$ be the corresponding left-hand sides, for $n \in \mathbf{N}$, where x_n is constrained by an ideal membership constraint $x_n \in I_n$. Since the divisibility relation on power products is a Dickson partial order, there exists an infinite subsequence $t_{n_1}, t_{n_2}, \ldots$ such that t_{n_i} properly divides t_{n_j} for all $i < j$ (cf. [17], pp. 162 & 189). Since N_∞ is left-reduced, we have $I_{n_{i+1}} \not\subseteq I_{n_i}$, for all i. Suppose we have $I_{n_j} \not\subseteq I_{n_{j+1}}$, for some j. Then there exists a union critical pair with maximal monomial $x t_{n_{j+1}}$, where x is constrained by $x \in I_{n_j} \cup I_{n_{j+1}}$. By Lemma 25, the monomial $x t_{n_{j+1}}$ is reducible by some rule in N_∞, which contradicts the assumption that N_∞ is left-reduced. We conclude that

$I_{n_j} \subset I_{n_{j+1}}$, for all j, which contradicts the assumption that the coefficient domain is Noetherian. □

We note that in order to ensure that we obtain finite fair derivations, we need to (i) eventually always apply *superposition* and *orientation*, whenever any of these transition rules is continuously applicable; and, (ii) eagerly apply the constraint manipulation rule which deletes rules with unsatisfiable constraints, Since N_∞ is finite, after a finite number of steps we would obtain a set N_k which contains all the finitely many rules in N_∞. Thereafter, under condition (ii), any fair derivation is terminating as after reaching state N_k, a fair derivation would would carry out finitely many applications of the superposition and constraint manipulation rules, before reaching a state $N_{k'}$ where no further rules are applicable.

8.4 Weak Gröbner bases

In this section we show that the limit N_∞ of a fair derivation gives us a weak Gröbner basis. To this effect, we first establish that we need to consider proofs on sums of monomials only. This we show by projecting a proof on arbitrary polynomial expressions to a proof on sums of monomials.

Consider the P-rules (4), (6) and (7), which are

$$\begin{aligned} \Omega \otimes \alpha &\rightarrow \Omega && && (4)\\ x\mu \otimes y\nu &\rightarrow z\mu \cdot \nu && \texttt{if } z = x \cdot y && (6)\\ \alpha \otimes (\beta \oplus \gamma) &\rightarrow (\alpha \otimes \beta) \oplus (\alpha \otimes \gamma) && && (7) \end{aligned}$$

Using the set of these three rules, which we denote by P_{467}, we can rewrite any ground polynomial expression to a sum of monomials, SOMs in short (consider Ω to be a monomial for this discussion),

$$p \rightarrow^!_{AC\setminus P^e_{467}} \sum_i x_i t_i \oplus \sum_j \Omega. \tag{4}$$

The set of polynomial rules in the set N will be referred to by R, and E would denote the set of equations in $N = E \cup R$.

Proposition 8 *If $p \leftrightarrow_{AC\setminus R,P^e} q$, then there is a proof of the following form,*

$$p \rightarrow^*_{AC\setminus P^e_{467}} p' \leftrightarrow^*_{AC\setminus R,P^e} q' \leftarrow^*_{AC\setminus P^e_{467}} q$$

where we have only sums of monomials in the proof of $p' \leftrightarrow^*_{AC\setminus R,P^e} q'$.

Proof. Since application of $R \cup P^e$ rules to SOMs yields SOMs, we need to only consider the case where either p or q is not a SOM. Without loss of generality, assume $p \rightarrow_{AC\setminus R,P^e} q$ and p is not a SOM. In this case we will have $p \rightarrow_{AC\setminus P^e_{467}} p'$. Since non-overlaps and variable overlaps commute, we just need to consider critical overlaps between arbitrary $P^e \cup R$ rules and the P rules (4), (6) and (7). In all such cases, it is easily shown that we can obtain the desired proof pattern. □

We can generalize proposition 8 to handle multiple applications of $R \cup P^e$ rules.

Proposition 9 *If $p \approx_{\mathrm{Id}(R)} q$, then there is a proof of the following form,*

$$p \rightarrow^!_{AC\setminus P^e_{467}} p' \leftrightarrow^*_{AC\setminus R,P^e} q' \leftarrow^!_{AC\setminus P^e_{467}} q$$

where we have only sums of monomials in the proof of $p' \leftrightarrow^*_{AC\setminus R,P^e} q'$.

Proof. Since $p \approx_{Id(R)} q$ implies $p \ominus {}^7q \in Id(R)$, therefore, there exist ground instances p_i of polynomials represented by the polynomial rewrite rules in R such that, $p \ominus q \leftrightarrow^*_{AC\setminus P^e} \sum_i p_i$. Therefore,

$$p \leftrightarrow^*_{AC\setminus P^e} p \ominus q \oplus q \leftrightarrow^*_{AC\setminus P^e} q \oplus \sum_i p_i \leftrightarrow^*_{AC\setminus R,P^e} q.$$

Using the result of proposition 8, we replace each P^e or R step by the proof on sums of monomials. Since P^e is convergent modulo AC, and since application of P^e rules to SOMs gives only SOMs, we have the desired result. □

8.4.1 Reduction ordering on sums of monomials

In order to establish the correctness of the procedure for computing weak Gröbner basis, we use the restriction of the predefined ordering $\succ$ on sums of monomials, denoted by $\succ_{\mathrm{SOM}}$. Later on, we would extend this ordering using a given ordering on coefficients. This will allow us to extend a weak Gröbner basis to a Gröbner basis. Its straight-forward to prove the following results about this ordering.

Lemma 26 *If p and q are sums of monomials and $p \succ_{SOM} q$ then for any p' we have $p \oplus p' \succ_{SOM} q \oplus p'$.*

Lemma 27 *The ordering $\succ_{SOM}$ is $AC \cup B$-compatible reduction ordering.*

Also, the polynomial rules R and the P rules are simplifying with respect to this ordering.

[7] The operator $\ominus$ is defined so that it is the inverse of $\oplus$, and yields a SOM when applied to a SOM.

8.4.2 Proof ordering

The correctness of the procedure will be established using proof simplification techniques for completion modulo associativity and commutativity, as described by Bachmair and Dershowitz [6, 5]. We define a complexity measure for ground proofs on sums of monomials in $AC \cup B \cup P^e \cup R \cup E$.

Let $p = p[\alpha\sigma] \leftrightarrow p[\beta\sigma] = q$ be an equational replacement step with an equation (or a rule ρ) $\alpha \approx \beta$ `if` C applied to the subterm of p ($\alpha\sigma \approx \beta\sigma$ is an unconstrained ground instance of $\alpha \approx \beta$ `if` C). Let t_0 and I denote the leading power product and the ideal defining the ideal membership constraint of $\alpha \approx \beta$ `if` C whenever this equation is actually a rule. Furthermore, let τ denote the time stamp when this rule was first added to N. The complexity of the proof step is the tuple

$$\begin{aligned}
(\{p,q\}, \bot, \bot, \bot, 0) \quad &\text{if} \quad \alpha \approx \beta \in E \\
(\{p\}, I, t_0, \{q\}, \tau) \quad &\text{if} \quad \alpha \to \beta \in R \\
(\{q\}, I, t_0, \{p\}, \tau) \quad &\text{if} \quad \beta \to \alpha \in R \\
(\{p\}, \bot, \bot, \{q\}, 0) \quad &\text{if} \quad \alpha \to \beta \in P \cup P^e \\
(\{p\}, \bot, \bot, \bot, 0) \quad &\text{if} \quad \alpha \approx \beta \in AC \cup B.
\end{aligned}$$

Such tuples are compared using the multiset extension of the reduction ordering $\succ_{\mathrm{SOM}}$ in the first and fourth component; the subset relation on sets in the second component; the less than relation on natural numbers in the fifth component; and the reduction ordering restricted to power products in the third component. (The constants $\top$ and $\bot$ denote additional maximum and minimum elements.) The complexity of a proof is the multiset of complexities of its proof steps. The multiset extension of the ordering on tuples yields a proof ordering, denoted by the symbol $\succ$.

Certain remarks on the proof ordering are required here. First we don't mention extensions R^e of the rules R because R are their own extensions. Secondly, in our completion procedure, the P rules and their extensions remain static, i.e., they are neither simplified or added. Hence the assigned proof complexity tuple for proof steps using P rules.

Lemma 28 *If $E \cup R \vdash E' \cup R'$, then corresponding to any proof in $E \cup R$ there is a proof in $E' \cup R'$ which is either equal to, or strictly simpler than, the original proof in the proof ordering $\succ$.*

Proof. Exhaustively do a test for each of the inference rules. Specifically, we have to check that the deleted instances have a simpler proof. This is easily verified

for simplification rules, orientation rule and all of the constraint manipulation rules except the one which simplifies the membership constraint. To show that the instances deleted in the superposition rule have smaller proofs,, it is crucial that we have I as the second, and t_0 as the third component in the complexity of a rewrite step. The technical conditions in the superposition rule ensure that the lemma is true in the case that this rule acts like a collapse rule. Similarly, the technical condition with the constraint manipulation rule which simplifies the membership constraint ensures that the lemma is true. □

Now we know that application of transition rules gives smaller proofs. Since the proof ordering is well-founded, there is a minimal proof. In the following last step, we show that if we carry out a *fair* derivation, a rewrite proof will be the minimal one. This will complete the proof of correctness for construction of weak Gröbner basis.

Theorem 20 ((Correctness)) *Suppose the coefficient domain B is Noetherian and satisfies conditions (B1)-(B5). If $N_\infty = R$ is the limit of a fair derivation, then N_∞^e is a weak Gröbner basis.*

Proof. Any fair derivation is unfailing as given any constrained equation, we can always orient it using constraint manipulation, simplification and orientation rules. Hence, if $N_\infty = E \cup R$, then $E = \emptyset$.

Note that N_∞ is finite by Theorem 19 and hence, let $N_\infty = \{\rho_1, \ldots, \rho_k\}$. By Lemma 28, the ideal generated by the input polynomials is exactly equal to the ideal generated by the set of all instances of p_i, $i = 1, \ldots, k$. Let $f \in Id(R)$. As none of the P rules can be applied to Ω, by Proposition 9 we get $f \rightarrow^!_{AC\setminus P^e_{467}} f' \leftrightarrow^*_{AC\setminus R,P^e} \Omega$ where we have only sums of monomials in the proof of $f' \leftrightarrow^*_{AC\setminus R,P^e} \Omega$.

Let $N_\infty = R = \{\rho_1, \ldots, \rho_k\}$ be the set of polynomial rewrite rules ρ_i: $x_0 t_{i0}\mu \rightarrow \sum_j x_j t_{ij}\mu$ `if` $x_0 = \vec{g_i} \cdot \vec{y} \wedge \vec{x} = \mathbf{A_i} \cdot \vec{y} \wedge x_0 \neq 0 \wedge \mu \notin \vec{s_i}\nu$ that occur in N_∞, and let $E = \{\xi_1, \ldots, \xi_k\}$ be the corresponding set of extended equations ξ_i, defined as, $(x_0 + x_0')t_{i0}\mu \approx x_0' t_{i0}\mu \oplus \sum_j x_j t_{ij}\mu$ `if` $x_0 = \vec{g_i} \cdot \vec{y} \wedge \vec{x} = \mathbf{A_i} \cdot \vec{y} \wedge \mu \notin \vec{s_i}\nu$. A left to right use of these equations, denoted by $\Rightarrow$ and a right to left use, denoted by $\Leftarrow$ will be used later to represent certain proof patterns that are not eliminated.

Let $f \in Id(R)$. As none of the P rules can be applied to Ω, by Proposition 9 we get $f \rightarrow^!_{AC\setminus P_{467}} f' \leftrightarrow^*_{AC\setminus R,P} \Omega$ where we have only sums of monomials in the proof of $f' \leftrightarrow^*_{AC\setminus R,P} \Omega$. We show that any such proof can be simplified to a simpler proof (in the proof ordering mentioned before), until we get a rewrite proof $f' \rightarrow^*_{AC\setminus P^e \cup R} \Omega$.

First we eliminate the following proof patterns:

1. $p \leftarrow_{AC\setminus P^e} p' \rightarrow_{AC\setminus P^e} q$: Since P^e is convergent modulo AC, we have a simpler proof of the form $p \rightarrow^*_{AC\setminus P^e} q' \leftarrow^*_{AC\setminus P^e} q$.

2. $p \leftarrow_{AC\setminus \rho_i} p' \rightarrow_{AC\setminus P^e} q$: Since we are dealing with sums of monomials, we cannot apply P rules (4), (6) and (7). There are no (non-trivial) overlaps of any of the R rules with the P^e-rules (1) and (3).

   Overlap between ρ_i and rule (2) produces:

$$\sum_j c_j t_{ij} \;\leftarrow_{AC\setminus \rho_i}\; ct_{i0} \;\rightarrow_{AC\setminus P^e}\; \Omega$$

   where $c = \vec{g_i} \cdot \vec{y} = 0$ and $\vec{c} = \mathbf{A_i} \cdot \vec{y}$. We get a smaller one-step proof using the same equation that got added to N when rule ρ_i was added (by either orientation, or superposition).

   A rule ρ_i non-trivially overlaps rule (5). Here we get the following peak:

$$\sum_j c_j t_{ij} \oplus c' t_{i0} \;\leftarrow_{AC\setminus \rho_i}\; ct_{i0} \oplus c' t_{i0} \;\rightarrow_{AC\setminus P^e}\; (c + c')t_{i0}$$

   where $c = \vec{g_i} \cdot \vec{y}$ and $\vec{c} = \mathbf{A_i} \cdot \vec{y}$. We do not eliminate such patterns right now. In the following, we reason about such patterns by referring to them by

$$\sum_j c_j t_{ij} \oplus c' t_{i0} \;\Leftarrow_{AC\setminus \xi_i}\; (c + c')t_{i0}$$

   and the inverse thereof.

3. $p \Leftarrow_{AC\setminus \xi_i} p' \rightarrow_{AC\setminus P^e} q$: Proof patterns with P rules (1) and (3) can be got ridden of by commuting and pushing the $\Leftarrow_{\xi_i}$ pattern below the application of the P rule.

   With rule (2) we need to consider the following pattern:

$$\begin{array}{ccc} (c-c)t_{i0} & \xrightarrow[AC\setminus P_2]{} & \Omega \\ {\scriptstyle AC\setminus \xi_i}\downarrow & & *\uparrow{\scriptstyle AC\setminus P^e} \\ (-c)t_{i0} \oplus \sum_j c_j t_{ij} & \xrightarrow[AC\setminus \rho_i]{} & \sum_j (-c_j)t_{ij} \oplus \sum_j c_j t_{ij} \end{array}$$

   Since $c = \vec{g_i} \cdot \vec{y}$, therefore $-c = \vec{g_i} \cdot (-\vec{y})$. The new proof is simpler because a peak is eliminated.

Patterns with rule (5):

$$\begin{array}{ccc} (c_0 + c')t_{i0} \oplus c''t_{i0} & \xrightarrow[AC\setminus\xi_i]{} & \sum_j c_j t_{ij} \oplus c't_{i0} \oplus c''t_{i0} \\ {\scriptstyle AC\setminus P^e}\downarrow & & \downarrow{\scriptstyle AC\setminus P^e} \\ (c_0 + c' + c'')t_{i0} & \xrightarrow[AC\setminus\xi_i]{} & \sum_j c_j t_{ij} \oplus (c' + c'')t_{i0} \end{array}$$

where $c_0 = \vec{g_i} \cdot \vec{y}$ and $\vec{c} = \mathbf{A_i} \cdot \vec{y}$. The new proof is simpler because the peak implicit in $\Leftarrow$ is smaller in the new proof.

4. $p \leftarrow_{AC\setminus\rho_i} p' \rightarrow_{AC\setminus\rho_j} q$: These proof patterns are explicitly considered in the procedure. Intersection critical pair obtained through overlap between rules ρ_i and ρ_j can be used to replace the peak.

$$\begin{array}{ccc} c_0 t_0 & \xrightarrow[AC\setminus\rho_j]{} & \sum_k b_k t_{jk} \\ {\scriptstyle AC\setminus\rho_i}\downarrow & & \uparrow{\scriptstyle AC\setminus\xi_{\text{int}}{}^{(\rho_i,\rho_j)\cup P^e}} \\ \sum_k c_k t_{ik} & \xleftarrow[AC\setminus P^e]{*} & \sum_k c_k t_{ik} \oplus \sum_k b_k t_{jk} \oplus \sum_k (-b_k) t_{jk} \end{array}$$

where $c_0 = \vec{g_i} \cdot \vec{y} = \vec{g_j} \cdot \vec{z}$ and $\vec{c} = \mathbf{A_i} \cdot \vec{y}$, and $\vec{b} = \mathbf{A_j} \cdot \vec{z}$. If $i = j$, we use the following simpler proof:

$$\sum_k c_k t_{ik} \;\leftrightarrow_{AC\setminus\xi}\; \sum_k c_k t_{ik} \oplus \sum_k (b_k - c_k) t_{ik} \;\rightarrow^*_{AC\setminus P^e}\; \sum_k b_k t_{ik}$$

The equation ξ is the equation produced by either the orientation rule or, the superposition rule, depending on which of these two created the rule ρ_i. The new proof is simpler in the proof ordering.

5. $p \Leftarrow_{AC\setminus\xi_i} p' \rightarrow_{AC\setminus\rho_j} q$: Union critical pair obtained through overlap between rules ρ_i and ρ_j can be used to replace such proof patterns.

$$\begin{array}{ccc} (c_0 + c'_0)t_0 & \xrightarrow[AC\setminus\rho_j]{} & \sum_k b_k t_{jk} \\ {\scriptstyle AC\setminus\xi_i}\downarrow & & *\uparrow{\scriptstyle AC\setminus P_e} \\ c'_0 t_0 \oplus \sum_k c_k t_{ik} & \xrightarrow[AC\setminus\rho_{\text{union}}{}^{(\rho_i,\rho_j)}]{} & \circ \end{array}$$

where $c_0 = \vec{g_i} \cdot \vec{y}$, $c_0 + c'_0 = \vec{g_j} \cdot \vec{z}$, $\vec{c} = \mathbf{A_i} \cdot \vec{y}$, and $\vec{b} = \mathbf{A_j} \cdot \vec{z}$. When $i = j$, then the simplified proof looks like this:

$$c'_0 t_0 \oplus \sum_k c_k t_{ik} \;\rightarrow_{AC\setminus\rho_i^e}\; \sum_k (b_k - c_k) t'_{ik} \oplus \sum_k c_k t_{ik} \;\rightarrow^*_{AC\setminus P^e}\; \sum_k b_k t_{ik}$$

The new proof is simpler as the peak implicit in $\Leftarrow$ pattern in the proof is eliminated. In the case when the set $\rho_{\text{union}}(\rho_i, \rho_j)$ is empty, we obtain a simpler proof using either ρ_i, or ρ_j, whichever "acts" as the union critical pair in this case.

Now, amongst all possible proofs of $f' \leftrightarrow_{AC\setminus N_\infty} \Omega$, let $\mathcal{P}$ be the minimal proof. None of the R or P^e rules can be applied to Ω. If $\mathcal{P}$ contains no steps using the extended equations ξ_i's, then $\mathcal{P}$ should be a rewrite proof $f' \rightarrow^*_{AC\setminus R\cup P^e} \Omega$.

If not, then there are certain ξ_i steps in the proof. Hence, the proof $\mathcal{P}$ can be written as,

$$f' \ \leftrightarrow^*_{AC\setminus R\cup P^e} \ f_1 \ \Rightarrow_{AC\setminus \xi_j} \ f_2 \ \rightarrow^+_{AC\setminus R\cup P^e} \ \Omega$$

Note that all other possible patterns involving $\Rightarrow$ have been eliminated. Clearly, f_1 can only be of the form $\sum_{j=1}^k c_j t'_j \oplus (c_0 + c'_0)t_i z$, such that there are *no* R or P^e steps in $\sum_{j=1}^k c_j t'_j$. Hence, f_2 is $\sum_{j=1}^k c_j t'_j \oplus c'_0 t_i z \oplus \sum_j b_j t_{ij} z$ where $c_0 = \vec{g_i} \cdot \vec{y}$. Since $f_2 \rightarrow^+_{AC\setminus P^e\cup R} \Omega$, hence, there should be $R \cup P^e$ steps to rewrite terms with the largest power product in f_2 to Ω. It is clear that in a minimal proof, the largest power product terms are eliminated before the others. Hence, $c'_0 t_i z$ is the largest monomial in f_2 (there might be others as large as this one). Hence, we need to consider only two (restricted) proof patterns: $\Rightarrow_{AC\setminus \xi_i} \circ \rightarrow_{AC\setminus R}$; and, $\Rightarrow_{AC\setminus \xi_i} \circ \rightarrow_{AC\setminus P^e}$.

The first pattern is eliminated by the use of the union critical pair rule. The second one is eliminated by commuting the two steps, thus pushing the ξ_i step further down (see the second commuting diagram in case 3 above). This completes the proof. □

8.5 Gröbner Bases

A weak Gröbner basis $\mathcal{R}$ for an ideal I defines a unique normal form, Ω, for all polynomial expressions equivalent to Ω modulo I. However, two two polynomial expressions equivalent modulo the ideal I (but not both equivalent to Ω) may not be reducible to the same expression with respect to $\mathcal{R}$. When equivalent polynomial expressions have identical normal forms, we say $\mathcal{R}$ is a *Gröbner basis*. Under an additional assumption on the coefficient domain, a weak Gröbner basis for polynomial ideals over commutative noetherian rings can be suitably extended to a Gröbner basis.

Assuming an ordering $>$ on the coefficient ring B, we can define ordered equations, see 3.

Definition 22 *If*

$$x_0 t_{i0}\mu \ \rightarrow\ \textstyle\sum_j x_j t_{ij}\mu \ \ \texttt{if}\ \ x_{i0} = \vec{g_i}\cdot\vec{y},\ \vec{x_i} = \mathbf{A_i}\cdot\vec{y},\ \mu \notin \vec{s_i}\nu$$

is a polynomial rule ρ_i, then

$$x_0 t_{i0}\mu \rightarrow x_0' t_{i0}\mu \oplus \textstyle\sum_j x_j t_{ij}\mu \ \ \texttt{if}\ \ x_0 > x_0',\ x_0 - x_0' = \vec{g_i}\cdot\vec{y},\ \vec{x_i} = \mathbf{A_i}\cdot\vec{y},\ \mu \notin \vec{s_i}\nu$$

will be called an ordered equation ζ_i corresponding to the polynomial rule ρ_i.

Next we define what we mean by a Gröbner basis.

Definition 23 *A set of ordered equations $\mathcal{G}$ is a Gröbner basis if for any two polynomial expressions p, q, we have $p \ominus q \in \mathrm{Id}(\mathcal{G})$ if and only if*

$$p \ \ \rightarrow_{AC\setminus P^e \cup \mathcal{G}} \ \ p' \ \ \leftarrow_{AC\setminus P^e \cup \mathcal{G}} \ \ q.$$

8.5.1 Additional assumption on coefficient domain

In addition to the conditions (B1)–(B5) on the coefficient domain, now we also assume that there exists a *well-founded ordering* $>$ that satisfies the following condition:

B6 (Uniqueness of minimal element) There exists a unique element of B in any equivalence class modulo any ideal in the ring B. In particular, 0 is the minimal element in the whole ring B, i.e., $x > 0$ for all $x \neq 0$.

Let I be a finitely generated ideal in B. Then, if there is a Gröbner basis for this ideal, then by the definition of Gröbner basis, there should a unique element in each equivalence class modulo I. This argues for the minimality of this additional assumption for establishing the existence of Gröbner basis. We note that in the assumption (B6), we can replace *any ideal* by *any finitely generated ideal* as every ideal is finitely generated in a Noetherian ring.

8.5.2 Reduction ordering on sums of monomials

We first note that the Propositions 8 and 9 were proved in their generality, and hence the correctness proof for construction of Gröbner bases can be restricted to considering proof patterns on sums of monomials. Nevertheless, since we now use

ordered equations as directed rules, we need to define a new ordering[8] on sums of monomials.

Let $\succ_{\text{pp}}$ and $>$ respectively denote some suitable (AC-compatible and well-founded) orderings on power-products and coefficients. Now, let $p = \sum_{i=1}^{k} a_i t_i$ and $q = \sum_{i=1}^{l} b_i s_i$ be any two sums of monomials. Without loss of generality, assume $t_i \succeq_{\text{pp}} t_j$ and $s_i \succeq_{\text{pp}} s_j$ for all $i < j$. Suppose $t_1 \leftrightarrow^*_{AC} t_2 \leftrightarrow^*_{AC} \cdots \leftrightarrow^*_{AC} t_{k'}$ and $s_1 \leftrightarrow^*_{AC} s_2 \leftrightarrow^*_{AC} \cdots \leftrightarrow^*_{AC} s_{l'}$ ($1 \leq k' \leq k$ and $1 \leq l' \leq l$). Now, $\succ_{\text{som}}$ is defined recursively by $p \succ_{\text{som}} q$ if, and only if,

(1) $\{t_1, \cdots, t_{k'}\} \succ^m_{\text{pp}} \{s_1, \cdots, s_{l'}\}$[9], or,

(2) $t_1 \leftrightarrow^*_{AC} s_1$, $k' = l'$ and $\{a_1, \cdots, a_{k'}\} >^m \{b_1, \cdots, b_{l'}\}$, or,

(3) $t_1 \leftrightarrow^*_{AC} s_1$, $k' = l'$, $\{a_1, \cdots, a_{k'}\} =^m \{b_1, \cdots, b_{l'}\}$, and $\sum_{i=k'+1}^{k} a_i t_i \succ_{\text{som}} \sum_{i=l'+1}^{l} b_i s_i$.

We establish some useful properties of this relation on sums of monomials.

Lemma 29 *The relation $\succ_{som}$ is an $AC \cup B$-compatible well-founded ordering.*

Proof. We first establish that $\succ_{\text{som}}$ is irreflexive and transitive. Let $p = \sum_{i=1}^{k} a_i t \oplus q$, $p' = \sum_{i=1}^{k'} a'_i t' \oplus q'$, and $p'' = \sum_{i=1}^{k''} a''_i t'' \oplus q''$ be arbitrary sums of monomials, where t, t' and t'' are respectively the largest power products in p, p' and p'' and we assume that they do not occur in q, q' and q''.

To prove that $\succ_{\text{som}}$ is irreflexive, we induct on the number of monomials in the polynomial expression. From definition it follows that $p \succ_{\text{som}} p$ if, and only if, $q \succ_{\text{som}} q$. However, induction hypothesis impies that $q \not\succ_{\text{som}} q$, and hence, $p \not\succ_{\text{som}} p$.

Next we establish transitivity. Assume that $p \succ_{\text{som}} p'$ and $p' \succ_{\text{som}} p''$. We prove that $p \succ_{\text{som}} p''$ by considering all possible cases based on which clause in the definition of $\succ_{\text{som}}$ witnesses $p \succ_{\text{som}} p'$ and $p' \succ_{\text{som}} p''$.

(i) if $p \succ_{\text{som}} p'$ by clause (1), then clearly $p \succ_{\text{som}} p''$ by clause (1) also.

(ii) if $p \succ_{\text{som}} p'$ by clause (2) and $p' \succ_{\text{som}} p''$ by clause (1), then $p \succ_{\text{som}} p''$ follows from clause (1).

(iii) if $p \succ_{\text{som}} p'$ by clause (2) and $p' \succ_{\text{som}} p''$ by clause (2) or (3), then $p \succ_{\text{som}} p''$ follows using clause (2).

(iv) if $p \succ_{\text{som}} p'$ by clause (3) and $p' \succ_{\text{som}} p''$ by clause (i), then $p \succ_{\text{som}} p''$ follows using clause (i), where i can be $1, 2$ or 3.

[8]Note that in the old ordering, the ordered equations defined in definition 22, are ordered in the *wrong* direction.

[9]We write $\succ^m$ to denote the multiset extension of the relation $\succ$.

We prove that *neworder* is well-founded by showing that any descending chain of elements $p_0 \succ_{\text{som}} p_1 \succ_{\text{som}} p_2 \succ_{\text{som}} \cdots$ is finite. This we do by induction on the largest power product in p_0. Assume $p_0 \succ_{\text{som}} p_1 \succ_{\text{som}} p_2 \succ_{\text{som}} \cdots$ is an infinite decreasing chain. First note that there can not be infinite number of steps witnessed by clause (1), since if so, we can extract an infinite chain of multiset of power products decreasing in the order $\succ^m_{\text{pp}}$. Therefore, without loss of generality, we can assume each step $p_i \succ_{\text{som}} p_{i+1}$ is witnessed by either clause (2) or clause (3). But since $>^m$ is well-founded, we can not have infinite steps witnessed by clause (2). Thus, there should exist an infinite chain where each step is witnessed by clause (3). But this is impossible by induction hypothesis.

Finally, the ordering $\succ_{\text{som}}$ is easily seen to be $AC \cup B$-compatible. □

Lemma 30 *Let $p = \sum_{i=1}^{k} a_i t_i$ and $q = \sum_{i=1}^{l} b_i s_i$ be two sums of monomials such that $p \succ_{som} q$. If at is a monomial with coefficient a and power product t, then (i) $p \oplus at \succ_{som} q \oplus at$.*
(ii) $\sum_{i=1}^{k} a_i t_i t \succ_{som} \sum_{i=1}^{l} b_i s_i t$.

Proof. If t is strictly larger than all power products in p, then $p \oplus at \succ_{\text{som}} q \oplus at$ follows from clause (3). In the other case, if $p \succ_{\text{som}} q$ is witnessed by clasue (i), then $p \oplus at \succ_{\text{som}} q \oplus at$ is witnessed by clause (i) too.

The second part of this lemma follows from the fact that the ordering $\succ_{\text{pp}}$ on power products is closed under contexts. □

It is easy to check that the polynomial rules and the ordered equations are all reducing with respect to this ordering.

We also need to extend the proof ordering as we have to assign a complexity to a proof step using the ordered equation step. Since the ordered equations ζ replace the polynomial rules ρ, we assign the same proof complexity to steps using them. Specifically, if $p = p[\alpha\sigma] \leftrightarrow p[\beta\sigma] = q$ is an equational replacement step with an ordered equation $\alpha \rightarrow \beta$ `if` C applied to the subterm of p, then, the complexity of this proof step is the tuple $(\{p\}, I, t_0, \{q\}, \tau)$, where I, t_0 and τ are as for the corresponding polynomial rule from which this ordered equation is obtained. Now the first and fourth components of the tuple are compared using the multiset extension of this modified ordering on sums of monomials.

8.5.3 Correctness proof for construction of Gröbner bases

In this section, R denotes a weak Gröbner basis, and $\mathcal{G}$ is the set containing all ordered equations corresponding to polynomial rules in R. We also assume that the coefficient domain B additionally satisfies condition (B6). In this case, every instance of any rule in R is also an instance of some rule in $\mathcal{G}$.

We first prove the following lemma that will be used repeatedly in the proof of correctness.

Lemma 31 *If $p_1 \leftarrow_{AC\setminus\mathcal{G},P^e} p \rightarrow_{AC\setminus\mathcal{G},P^e} p_2$ is a peak, and $p \succ_{SOM} p_2 \oplus p_1 \ominus p_2$, then there is a simpler proof for this peak.*

Proof. Since R is a weak basis, and since $p_1 \ominus p_2 \in Id(R)$, hence $p_1 \ominus p_2 \rightarrow^*_{AC\setminus R,P^e} \Omega$. It can be checked that the following is a simpler proof.

$$p_1 \ \leftarrow^*_{AC\setminus P^e} \ p_2 \oplus p_1 \ominus p_2 \ \rightarrow^*_{AC\setminus\mathcal{G},P^e} \ p_2.$$

Note that $\ominus$ is defined such that when applied to two SOMs, it yields a SOM. □

Now we are ready to prove the main result of this section.

Theorem 21 (Correctness) *If $R = \{\rho_1, \rho_2, \ldots, \rho_k\}$ is a weak Gröbner basis, then the set $\mathcal{G} = \{\zeta_1, \zeta_2, \ldots, \zeta_k\}$, containing all ordered equations corresponding to polynomial rules in R (Definition 22), is a (strong) Gröbner basis.*

Proof. Using Lemma 9 and reasoning just as in the proof for weak Gröbner basis, we have that for $p \ominus q \in Id(\mathcal{G})$,

$$p \ \rightarrow^!_{AC\setminus P_{467}} \ p' \ \leftrightarrow^*_{AC\setminus\mathcal{G},P} \ q' \ \leftarrow^!_{AC\setminus P_{467}} \ q$$

where the subproof $p' \leftrightarrow^*_{AC\setminus\mathcal{G},P} q'$ contains only sums of monomials[10].

We show that we can get a rewrite proof using proof simplification arguments on the proof on SOMs. Note that since the P-rules (4), (6) and (7) can not be applied on SOMs, in the sequel, P^e would just mean the rules (1), (2), (3) and (5).

There are three kinds of peaks to consider. In the simple case, peaks between two P^e-rules can be eliminated since P^e is convergent (modulo AC). Secondly, we consider peaks involving a P-rule and an ordered equation. Proof patterns with P rule (3) can be eliminated by commuting. With rule (2), due to the ordering constraint in

[10]Note that $Id(\mathcal{G}) = Id(R)$.

ζ_i rule, no non-trivial overlap is possible. For rule (5) the argument is similar to the weak Gröbner basis case (Section 8.4).

Finally, we consider overlap between rules ζ_i and ζ_j. Suppose $t_{i0}s \leftrightarrow^*_{AC} t_{j0}s'$ and $c = a_0 + a'_0 = b_0 + b'_0$ such that $a_0 = \vec{g_i} \cdot \vec{y}, c > a'_0$ and $b_0 = \vec{g_j} \cdot \vec{z}$, $c > b'_0$. Then we have the following peak:

$$a'_0 t_{i0}s \oplus \sum_k a_k t_{ik}s \;\leftarrow_{AC\setminus\zeta_i}\; ct_{i0}s \;\rightarrow_{AC\setminus\zeta_j}\; b'_0 t_{j0}s' \oplus \sum_k b_k t_{jk}s'$$

Now we have $p = (a'_0 - b'_0)t_{i0}s \oplus \sum_k a_k t_{ik}s \oplus \sum_k (-b_k)t_{jk}s'$ belongs to the ideal $Id(R)$. Since R is a weak Gröbner basis, therefore, $p \rightarrow^*_{AC\setminus R,P^e} \Omega$.

There are two cases now, either $a'_0 = b'_0$ or $a'_0 \neq b'_0$. If $a'_0 = b'_0$, then we can use Lemma 31 and get a simpler proof (on $a'_0 t_{i0}s \oplus \sum_k a_k t_{ik}s \oplus \sum_k b_k t_{jk}s' \oplus \sum_k (-b_k)t_{jk}s'$). If $a'_0 \neq b'_0$, then, since $p \rightarrow^*_{AC\setminus R,P^e} \Omega$, therefore there should be an R-rule ρ_k that rewrites the monomial $(a'_0 - b'_0)t_{i0}s$. By assumption (B6) there is a minimal element c^* in the equivalence class of a'_0 (which is the same as the equivalence class of b'_0) modulo the ideal generated by the single element $a'_0 - b'_0$. Using this observation, the two polynomial expressions involved in the original peak can be rewritten as:

$$a'_0 t_{i0}s \oplus \sum_l a_l t_{il}s \;\rightarrow_{AC\setminus\zeta_k}\; c^* t_{i0}s \oplus \sum_l c_l t_{kl}s'' \oplus \sum_l a_l t_{il}s$$

$$b'_0 t_{i0}s \oplus \sum_l b_l t_{il}s \;\rightarrow_{AC\setminus\zeta_k}\; c^* t_{i0}s \oplus \sum_l d_l t_{kl}s'' \oplus \sum_l b_l t_{il}s$$

where $a'_0 - c^* = \vec{g_k} \cdot \vec{x}$, $\vec{c} = \mathbf{A_k} \cdot \vec{x}$, $b'_0 - c^* = \vec{g_k} \cdot \vec{y}$, and $\vec{d} = \mathbf{A_k} \cdot \vec{y}$. Now we note that the largest power product term $c^* t_{i0}s$ is the same in both the expressions. This is the same as the $a'_0 = b'_0$ case. Hence we can complete the proof just as we did in that case.

Note that the case when $i = j$ (i.e., overlap of ordered rules with themselves) can be similarly handled. Thus we can eliminate all peaks and finally we will be left with a rewrite proof. This proves that we have a Gröbner basis. □

8.6 Other Approaches for Computing Gröbner Bases

An algorithm for constructing (weak) Gröbner bases for polynomial ideals over Noetherian commutative rings has been discussed in [61] and [1]. The basic method

involves adding generalized critical pairs obtained using syzygy computations over the coefficient ring. Additional optimizations can be incorporated to the computation of syzygies [61].

We briefly describe the above mentioned approach for computing weak bases. We will also point out some known optimizations that have been done. A weak Gröbner basis for a set of polynomials R is computed by adding generalized S-polynomials (to be defined below) to the set R. In other words, rather than computing an intersection critical pair $\xi_{\text{int}}(\rho_1, \rho_2)$ between *two* rules (polynomials) ρ_1 and ρ_2, one computes critical pairs over all possible *subsets* of rules. This is required since there is no concept of a union critical pair in these algorithms.

Let $N = \{p_1, p_2, \ldots, p_m\}$ be a set of (unconstrained) *ground* polynomial rules with $c_{i0}t_{i0}$ being the leading monomial in p_i. We compute a generating set $\mathcal{A} \subset B^m$ for the syzygies $\text{Syz}(c_{10}, c_{20}, \ldots, c_{m0})$. Corresponding to each element in $\mathcal{A}$, we have an S-polynomial. Specifically, if $\vec{a} = (a_1, \ldots, a_m) \in \mathcal{A}$, then we have an S-polynomial equation

$$a_1 \cdot p_1 \cdot t/t_{10} \oplus a_2 \cdot p_2 \cdot t/t_{20} \oplus \cdots \oplus a_m \cdot p_m \cdot t/t_{m0} \approx \Omega$$

where $t = lcm(t_{10}, \ldots, t_{m0})$. These S-polynomial equations are added to the set N in case they are not simplifiable to Ω by N. The process is repeated until no more polynomials get added. Reduction is defined as simultaneous reduction where several rules are used at once for reducing a term.

The computation of syzygies can be optimized ([61]). We can do it incrementally - starting from computing syzygies for small subsets, and gradually going to larger subsets. Further details will be given in the example below.

The conditions assumed of the coefficient domain in [61] and [1] are equivalent to the assumptions we make, see section 8.7. However, it is difficult to formally say the connection of the "optimizations" we propose with the ones suggested by Möller. The idea of using only *saturated* subsets of G for computing the S-polynomials is, informally speaking, subsumed by the inequality constraints we introduce. This point will be highlighted in the example. Certain cases of the second optimization, that avoids computing certain other S-polynomials based on conditions on the coefficients, are also handled in our framework by the condition that avoids the union critical pair computation in certain cases (when one ideal is subset of the other). Effective implementation of Möller's algorithm requires computation of remainder ideals. Next, we illustrate some of these remarks through an example.

Example 20 *Consider the ring $\mathcal{Z}[X,Y]$ where $\mathcal{Z}$ denotes the set of integers. Let $N_0 = \{3X^2Y \approx 0, 4XY^2 \approx 0, 15X^2 \approx 0\}$. We will use the lexicographic extension of $Y > X$ as the ordering on power-products. In this example, we use $\mathcal{Z}$ as the coefficient domain. Converting the equations to standardized form, we start with the following set N_0.*

$$\begin{aligned} N_0 \;= \{\;& xX^2Y\mu \approx \Omega \quad \texttt{if} \quad x = 3y_1 \\ & xXY^2\mu \approx \Omega \quad \texttt{if} \quad x = 4y_1 \\ & xX^2\mu \approx \Omega \quad \texttt{if} \quad x = 15y_1\}. \end{aligned}$$

Next we apply the orientation (split) rule to the three equations. Since $\mathcal{Z}$ is an integral domain (no zero divisors), we simply get back N_0 with the equations replaced by directed rules. Next say we apply the superposition rule (taking the first two rules). The union critical pair rule is $xX^2Y^2 \rightarrow \Omega \oplus \Omega$ if $x = 3y_1 + 4y_2$. The intersection critical pair equation similarly is $\Omega \approx \Omega$ if $4y_2 = 3y_1$. The intersection critical pair equation is deleted, whereas the union critical pair simplifies (using constraint manipulation and simplification rules) to $xX^2Y^2 \rightarrow \Omega$ if $x = 1y_1$. Note now that in the superposition rule, additional constraints are added to the original rules. So, now the new set of rules N_1 is:

$$\begin{aligned} xX^2Y\mu &\rightarrow \Omega \quad \texttt{if} \quad x = 3y_1,\ \mu \neq Y\nu \\ xXY^2\mu &\rightarrow \Omega \quad \texttt{if} \quad x = 4y_1,\ \mu \neq X\nu \\ xX^2Y^2\mu &\rightarrow \Omega \quad \texttt{if} \quad x = 1y_1 \\ xX^2\mu &\rightarrow \Omega \quad \texttt{if} \quad x = 15y_1 \end{aligned}$$

Note that the added constraints strictly restrict the application of the superposition rule. But still, in this example, there are peaks left. Now, say we next compute critical pairs between the first and the last rules. Clearly the union critical pair is not defined as $\mathrm{Id}(15) \subset \mathrm{Id}(3)$. *Furthermore, similar to what happened before, the intersection critical pair is deleted soon. However, the superposition step does add an extra constraint on the last rule. So, the new set N_2 we will obtain is N_1 minus the last rule, plus the following rule:*

$$xX^2\mu \rightarrow \Omega \quad \texttt{if} \quad x = 15y_1,\ \mu \neq Y\nu$$

Now, we see that due to this change, no more peaks remain. Thus, N_2 is a weak Gröbner basis.

The next example shows briefly how Möller's optimized method would compute the basis for set N_0.

Example 21 *Let us first number the rules for ease of reference.*

$$1.\ \ 3X^2Y \ \rightarrow\ \Omega \qquad 2.\ \ 4XY^2 \ \rightarrow\ \Omega \qquad 3.\ \ 15X^2 \ \rightarrow\ \Omega$$

We need to generate S-polynomials corresponding to different subsets of these rules. Since integers form an integral domain, we don't need to consider singleton sets. Secondly, we only need to consider saturated *sets: the* lcm *of the leading power products of rules in the set is not divisible by the leading power product of any rule not in the set. The only saturated (non singleton) sets are {1,2}, {1,3} and {1,2,3}.*

Corresponding to the choice $\{1, 2\}$*, the* generalized *intersection critical pair, or the S-polynomial, is computed. We first find generators for remainder ideal*[11] $\mathrm{Id}(3) : \mathrm{Id}(4) = \mathrm{Id}(3)$*. This shows that the only syzygy for* $\langle 3, 4\rangle$ *is* $(-4, 3)$*. The corresponding S-polynomial is seen to simplify to* $\Omega \approx \Omega$*.*

For $\{1, 3\}$*, we compute* $\mathrm{Id}(3) : \mathrm{Id}(15) = \mathrm{Id}(1)$ *giving the syzygy* $(5, 0, -1)$*. The corresponding S-polynomial is also* $\Omega \approx \Omega$*. Finally, we need to compute syzygies and the generalized S-polynomial for the rules* $(1), (2)$ *and* (3) *taken together (corresponding to the choice* $\{1, 2, 3\}$*). We compute* $\mathrm{Id}(3, 4) : \mathrm{Id}(15) = \mathrm{Id}(1)$*, and hence we may use the same syzygy* $(5, 0, -1)$ *as before. Therefore, no work is done in this case. There are no more possibilities to consider for set J. Hence, we stop.*

The equivalent of considering only saturated sets, and not arbitrary subsets, for computing S-polynomials is captured in our optimization as well. In the example above, the set $\{2, 3\}$ wasn't considered, and identically, we did not apply the *Superposition* rule on these rules in Example 20. This can be seen to be true in general as well.

On the other hand, the second optimization, based on using same syzygies for different saturated sets, is also handled in our setting in most of the cases (though it is difficult to argue about it in general). In the above example, Möller's method did no work on the saturated set $\{1, 2, 3\}$. The corresponding *Superposition* rule, applied on the union critical pair of rules $1, 2$ and rule 3, wasn't even considered in our method. We further note that Möller's method required computation of the remainder ideal, whereas we deal with ideal inclusion questions. Moreover, the definition of *reduction* is different from the standard notion.

[11] *Given ideals I and J, the remainder ideal* $I : J$ *is defined as* $\{r : rJ \subset I\}$*.*

8.7 A note on the minimality of assumptions

We have presented a procedure for computing strong Gröbner basis for finitely generated ideals in polynomial rings over a coefficient domain B that is a commutative ring with unit that satisfies the additional assumptions (B1)-(B6) mentioned in section 8.2. We will shortly show that the set of conditions (B1)-(B5) are *equivalent* to the following:

B1 (Decidability and solvability of ideal membership) Same as condition (B1).

B2' (Solvability of syzygies) Given coefficients $a_1, \cdots, a_k$, the generators of the module of syzygies of $(a_1, \cdots, a_k)$ can be computed.

Proposition 10 *The set of assumptions (B1)-(B5), defined in Section 8.2 is equivalent to the two assumptions (B1) and (B2').*

Proof. Assume the ring B satisfies the first set of assumptions. Let us say we need to compute the generators for the module of syzygies of $(a_1, \cdots, a_k)$. If $k = 1$ then assumption (B5) implies that we can compute the required generators. If $k > 1$ then we group the coefficients into two disjoint sets, say $S_1 = \{a_1\}$ and $S_2 = \{a_2, \cdots, a_k\}$. We can compute the generators $b_1, \cdots, b_l$ of the intersection ideal $I = Id(S_1) \cap Id(S_2)$ (by assumption (B3)). Corresponding to each generator g_i, we can compute vectors $\vec{b_i} = \langle b'_{i1}, \cdots, b'_{ik} \rangle$ such that $g_i = \sum_{j=2}^{k} b'_{ij} a_j$ and $g_i = b_{i1} a_1$ (follows from assumption (B1)). It is a trivial exercise to verify that the set $\{\vec{b_1}, \cdots, \vec{b_l}\}$ is the required set of generators.

The other direction of the equivalence can be established similarly. We just need to note that the assumptions $(B2)$ and $(B4)$ are implicitly implied by the assumption (B2'). □

We now prove the following result which helps to argue that the above set of assumptions is the weakest set of assumptions necessary to obtain a weak Gröbner basis algorithm.

Lemma 32 *If there is a procedure for computing a weak Gröbner basis for any finitely generated ideal in $B[Y]$, then there is a procedure for computing generators for the intersection of any two finitely generated ideals in B.*

Proof. Let $I_1 = Id(a_1, \cdots, a_k)$ and $I_2 = Id(b_1, \cdots, b_l)$ be two finitely generated ideals in B. Compute the weak Gröbner basis for $\{a_1 Y, \cdots, a_k Y, b_1(1 \ominus Y), \cdots, b_l(1 \ominus Y)\}$,

using B as the coefficient domain and an ordering on power products such that any product containing Y is larger than any product which does not contain a Y. We claim that the set of polynomials $c_1, \cdots, c_m$ corresponding to those rules in the weak Gröbner basis[12] that do not contain Y (elimination ideal) form a set of generators for the ideal $I_1 \cap I_2$.

The proof of the claim is easy to see. Suppose $c \in Id(c_1, \cdots, c_m)$. Therefore, $c = \sum_1^k a_i' a_i Y \oplus \sum_1^l b_i' b_i (1 \ominus Y)$. Substituting $Y = 0$ and $Y = 1$ respectively gives $c \in I_1$ and $c \in I_2$.

Now suppose $c \in I_1$ and $c \in I_2$. Clearly, $c = cY \oplus c(1 \ominus Y)$. We get $c = (\sum_1^k a_i' a_i)Y \oplus (\sum_1^l b_i' b_i)(1 \ominus Y)$. Hence proved. □

If there is a procedure to compute a weak Gröbner basis for finitely generated ideals in $B[X]$, then there is a procedure to (1) decide ideal membership in B, and (2) compute syzygy generators for any finite set A of elements in B. The first claim can is established by noting that elements in B are also elements in $B[X]$. The second claim can be easily proved by observing that if $|A| > 1$, then using the Lemma 32, and the equivalence of assumptions proof, we can compute syzygy generators. If $|A| = 1$, say $A = \{a\}$, then we generate the weak Gröbner basis for $\{aY = 1\}$. It is easily seen that the rules in the weak Gröbner basis not containing Y give generators for the solution set of $ax = 0$. The Noetherian ring assumption is required essentially for termination. This suggests that the set of assumptions we make for the coefficient domain B are, in this sense, *minimal* required in order to obtain a Gröbner basis computation procedure for the ring $B[X]$.

8.8 Lifting of assumptions from B to $B[X]$

We next show that the conditions (B1)–(B5) are satisfied by a large class of rings. This we do by proving that whenever a ring B satisfies them, then so does $B[X]$. In this section whenever we write $p = q$ for ground polynomial expressions p and q, we mean $p \leftrightarrow^*_{AC \setminus P^e} q$.

Lemma 33 (Lifting lemma) *Suppose that ideal membership in ring B is decidable and that there is an effective procedure to compute syzygies in B. Then, if there is a procedure for computing a weak Gröbner basis for any finitely generated ideal in*

[12]There can be more than one polynomial associated with one rewrite rule in the weak Gröbner basis. For each generator of the ideal in the left-hand side of a rule, we generate one ground polynomial.

$B[X]$, then there is a procedure for computing the generators of syzygies of any finite set of polynomials in $B[X]$.

Proof. First consider a weak Gröbner basis $R = \{\rho_1, \cdots, \rho_n\}$, where the rule ρ_i, ignoring the extension variable μ and the inequality constraints, is,

$$x_0 t_{i0} \ \rightarrow \ \sum_j x_j t_{ij} \ \ \texttt{if} \ \ x_0 = \vec{a_i} \cdot \vec{y}, \ \vec{x} = \mathbf{A_i} \cdot \vec{y}.$$

In the sequel, we will denote the right-hand side expression of the above rule by $q_i(\mathbf{A_i} \cdot \vec{y})$. If $\vec{a_i} = \langle a_{i1}, \cdots, a_{ik_i} \rangle$ then with the rule ρ_i, we can associate polynomials $p_{i1}, \cdots, p_{ik_i}$, where polynomial p_{ik} is defined as,

$$p_{ik} = a_{ik} t_{i0} \ominus q_i(\mathbf{A_i}(k)^T)$$

where $\mathbf{A_i}(k)^T$ denotes the transposed k-th column of the matrix $\mathbf{A_j}$.[13]

We first give a method to compute the generators for the module S of syzygies of $(p_{11}, \ldots, p_{1k_1}, p_{21}, \ldots, p_{2k_2}, \ldots, p_{n1}, \ldots, p_{nk_n})$.

$$S = \{(q_{11}, \cdots, q_{nk_n}) | \sum_{i,j} q_{ij} p_{ij} = \Omega\}$$

The finite set B of generators of S will be obtained as a union of two sets B_1 and B_2.

B_1 **or the zero-divisor/split case :** Let $\vec{b_{i1}}, \cdots, \vec{b_{il_i}}$ be the generators of the syzygies of $(\vec{a_i})$, i.e., generators of the module $\{\vec{x} | \vec{a_i} \cdot \vec{x} = 0\}$. Clearly then,

$$q_i(\mathbf{A_i} \cdot \vec{b_{il}}) \in Id(R)$$

for $l = 1, \ldots, l_i$. Since R is a weak Gröbner basis,

$$q_i(\mathbf{A_i} \cdot \vec{b_{il}}) = \sum_{j,k} q_{iljk} p_{jk} \tag{5}$$

where[14] $q_{iljk} \in B[X]$ and $\mathrm{HT}(q_{iljk} p_{jk}) \leq \mathrm{HT}(q_i(\mathbf{A_i} \cdot \vec{b_{il}}))$. The left-hand side of equation 5 can be rewritten to give the new equation,

$$\sum_{k=1}^{k_i} b_{ilk} p_{ik} = (\sum_{k=1}^{k_i} b_{ilk} a_{ik} t_{i0}) \ominus q_i(\mathbf{A_i} \cdot \vec{b_{il}}) = \sum_{j,k} -q_{iljk} p_{jk} \tag{6}$$

where $\vec{b_{il}} = \langle b_{il1}, \cdots, b_{ilk_i} \rangle$ is a syzygy of $\vec{a_i}$. Note that $q_{ilik} = 0$ for $1 \leq k \leq k_i$.

[13] Note that $p_{i1}, \cdots, p_{ik_i}$ generate any polynomial instance of the rule ρ_i.

[14] We write $\mathrm{HT}(p)$ to denote the largest power product in the sum of monomials p.

Define

$$r_{iljk} = \begin{cases} q_{iljk} & \texttt{if} \quad j \neq i \\ b_{ilk} & \texttt{if} \quad j = i \end{cases}$$

Define

$$\vec{r_{il}} = \langle r_{il11}, \cdots, r_{ilnk_n} \rangle$$

Clearly, $\vec{r_{il}} \in S$ for $1 \leq i \leq n$ and $1 \leq l \leq l_i$. Define the set $B_1 = \{\vec{r_{il}} | 1 \leq i \leq n, 1 \leq l \leq l_i\}$.

B_2 or the intersection of ideals case : Let

$$\{(\vec{b_{ij1}}, \vec{b_{ji1}}), (\vec{b_{ij2}}, \vec{b_{ji2}}), \cdots, (\vec{b_{ijl_{ij}}}, \vec{b_{jil_{ij}}})\}$$

be the generators of the syzygies of $(\vec{a_i}, \vec{a_j})$.

Suppose $t_{i0}s_i \approx_{AC} t_{j0}s_j$. Define for $m = 1$ to l_{ij},

$$q^{cp}_{ijm} = \sum_{k=1}^{k_i} b_{ijmk} s_i p_{ik} \oplus \sum_{k=1}^{k_j} b_{jimk} s_j p_{jk}$$

Clearly, the highest power product term in q^{cp}_{ijm} vanishes. Hence, $\mathrm{HT}(q^{cp}_{ijm}) < t_{i0}s_i$. Also $q^{cp}_{ijm} \in Id(R)$ and therefore,

$$q^{cp}_{ijm} = \sum_{k,l} q_{ijmkl} p_{kl}$$

Eliminating q^{cp}_{ijm} from these two equations we get,

$$\sum_{k=1}^{k_i} b_{ijmk} s_i p_{ik} \oplus \sum_{k=1}^{k_j} b_{jimk} s_j p_{jk} = \sum_{k,l} q_{ijmkl} p_{kl}$$

Define,

$$r_{ijmkl} = \begin{cases} q_{ijmil} \ominus b_{ijml} s_i & \texttt{if} \quad k = i \\ q_{ijmjl} \ominus b_{jiml} s_j & \texttt{if} \quad k = j \\ q_{ijmkl} & \text{otherwise} \end{cases}$$

Note that $\mathrm{HT}(q_{ijmil}) < s_i$ and $\mathrm{HT}(q_{ijmjl}) < s_j$. Define,

$$\vec{r_{ijm}} = \langle r_{ijm11}, \cdots, r_{ijmnk_n} \rangle$$

Clearly $\vec{r_{ijm}} \in S$ for $1 \leq i < j \leq n, 1 \leq m \leq l_{ij}$. We put all these vectors in $B_2 = \{\vec{r_{ijm}}\}$.

We have thus found elements in the set S of syzygies. We prove that this set of elements B obtained through the above methodology is the desired generating system for the module of syzygies in question.

Lemma 34 *The set $B = B_1 \cup B_2$ generates S as an $B[X]$ module.*

Proof. Assume for a contradiction that the set $M = S - (G)$ is non-empty. Let $(h_{11}, \cdots, h_{nk_n}) \in M$ be such that

$$t = \max\{\mathrm{HT}(h_{ij}p_{ij}) | 1 \leq i \leq n, 1 \leq j \leq k_i\}$$

is minimal w.r.t. the term order used for constructing the weak Gröbner basis in the set

$$\max\{\mathrm{HT}(g_{ij}p_{ij}) \,|1 \ \leq i \leq n, 1 \leq j \leq k_i \mid (g_{11}, \cdots, g_{nk_n}) \in M\}$$

Assume further that among all possible choices that satisfy this requirement, $\vec{h} = (h_{11}, \cdots, h_{nk_n})$ is such that the cardinality of the set

$$J \ = \ \{i \mid 1 \leq i \leq n, \ 1 \leq j \leq k_i, \ \mathrm{HT}(h_{ij}p_{ij}) \ = \ t\}$$

is minimal. We have

$$\sum_{i,j} h_{ij}p_{ij} \ = \ \Omega$$

since $\vec{h} = (h_{11}, \cdots, h_{nk_n}) \in S$, and so it follows that the sum of all monomials in this sum that have t as their power product equals 0.

We have three cases.

$|J| = 1$ **:** Without loss of generality, assume $J = \{1\}$. Therefore we have,

$$\sum_{j=1}^{k_1} \mathrm{HC}(h_{1j}p_{1j}) = 0$$

Let $\vec{b_{h_1}}^T$ denote the column vector of coefficients of the highest power product t/t_{10} in $h_{11}, \cdots, h_{1k_1}$. Clearly then, $\vec{a_1} \cdot \vec{b_{h_1}}^T = 0$. Therefore, $\vec{b_{h_1}} = \sum_{j=1}^{l_1} b'_{1j}\vec{b_{1j}}$ where b'_{1j} are arbitrary elements of B and $\vec{b_{1*}}$ are generators of the syzygies of $\vec{a_1}$.

Now, define

$$\vec{b} \ = \ \sum_{j=1}^{l_1} b'_{1j}\vec{r_{1j}}(t/t_{10}) \ \in \ (G)$$

where $\vec{r_{1j}} \in B_1$. Consider $\vec{h'} = \vec{h} - \vec{b}$. Clearly, $\mathrm{HT}(\vec{h'}) < \mathrm{HT}(\vec{h})$, and therefore, $\vec{h'} \in (G)$. But then since $\vec{b} \in (G)$, $\vec{h} \in (G)$.

$|J| = 2$ **:** Without loss of generality, assume $J = \{1, 2\}$. Therefore we have,

$$\sum_{j=1}^{k_1} \mathrm{HC}(h_{1j}p_{1j}) + \sum_{j=1}^{k_2} \mathrm{HC}(h_{2j}p_{2j}) = \Omega$$

where HC denotes the coefficient of t in the argument. Let $\vec{b_{h_i}}$ denote the row vector of coefficients of the highest power product t/t_{i0} in $h_{i1}, \cdots, h_{ik_i}$. Clearly then, $[\vec{a_1}, \vec{a_2}] \cdot [\vec{b_{h_1}}, \vec{b_{h_2}}]^T = 0$. Therefore, $[\vec{b_{h_1}}, \vec{b_{h_2}}] = \sum_{j=1}^{l_{12}} b'_{12j}[\vec{b_{12j}}, \vec{b_{21j}}]$ where b'_{12j} are arbitrary elements of B and $[\vec{b_{12j}}, \vec{b_{21j}}]$'s are generators of the syzygies of $[\vec{a_1}, \vec{a_2}]$.

Now, define

$$\vec{b} \;=\; \sum_{j=1}^{l_{12}} b'_{12j} \vec{r_{12j}} (t/lcm(t_{10}, t_{20})) \;\in\; (G)$$

where $\vec{r_{12j}} \in B_2$. Consider $\vec{h'} = \vec{h} + \vec{b}$. Clearly, $\mathrm{HT}(\vec{h'}) < \mathrm{HT}(\vec{h})$, and therefore, $\vec{h'} \in (G)$. But then since $\vec{b} \in (G)$, $\vec{h} \in (G)$.

$|J| > 2$ **:** Without loss of generality, assume $\{1, 2\} \subset J$. Let $\vec{b_1}$ denote the row vector of coefficients of the highest power product t/t_{10} in $h_{11}, \cdots, h_{1k_1}$ and let $\vec{b_2}$ denote the row vector of coefficients of the highest power product t/t_{20} in $h_{21}, \cdots, h_{2k_2}$. If $[\vec{a_1}, \vec{a_2}] \cdot [\vec{b_{h_1}}, \vec{b_{h_2}}]^T = 0$, then we carry out the reasoning of the previous case.

If not, then if $Id(\vec{a_1}) \subset Id(\vec{a_2})$ then since $Id(\vec{a_1}) \cap Id(\vec{a_2}) = Id(\vec{a_1})$, hence $(\vec{b_{121}}, \cdots, \vec{b_{12l_{12}}})$ (the first $k_1 = |\vec{a_1}|$ components of the generators of syzygies of $(\vec{a_1}, \vec{a_2})$, see definition of set B_2) generate whole of B^{k_1}. Hence the vector $\vec{b_{h_1}}$ can be expressed as a linear combination of $(\vec{b_{121}}, \cdots, \vec{b_{12l_{12}}})$.

$$\vec{b_{h_1}} \;=\; \sum_{j=1}^{l_{12}} b'_{12j} \vec{b_{12j}}$$

where b'_{12j} are arbitrary elements of B and $[\vec{b_{12j}}, \vec{b_{21j}}]$'s are generators of the syzygies of $[\vec{a_1}, \vec{a_2}]$. Now, define

$$\vec{b} \;=\; \sum_{j=1}^{l_{12}} b'_{12j} \vec{r_{12j}} (t/lcm(t_{10}, t_{20})) \;\in\; (G)$$

Consider $\vec{h'} = \vec{h} + \vec{b}$. The number of occurrences of the highest term is reduced by 1, i.e., the J corresponding to $\vec{h'}$ has cardinality one less than the J for $\vec{h}$. Therefore, $\vec{h'} \in (G)$. But then since $\vec{b} \in (G)$, $\vec{h} \in (G)$.

The case when $Id(\vec{a_2}) \subset Id(\vec{a_1})$ is symmetric.

If neither of the ideals is contained in the other, then using lemma 35, it follows that all instances of $(\vec{a_1} \cdot \vec{y_1} + \vec{a_2} \cdot \vec{y_2}) lcm(t_i, t_j)$ should be reducible by a single R-rule ρ_3. Clearly, $Id(\vec{a_1}) \subset Id(\vec{a_3})$ and $Id(\vec{a_2}) \subset Id(\vec{a_3})$. Using the same argument as above (for the case when we had $Id(\vec{a_1}) \subset Id(\vec{a_2})$) *twice* we can get a new vector $\vec{h'}$ such that the corresponding J' set (defined as J was defined for $\vec{h}$) satisfies $J' = J \cup \{k\} - \{i, j\}$. Hence, again we can conclude that $\vec{h} \in (G)$.

This concludes the proof of the fact that G is a generating set for the syzygies of $p_{11}, \cdots, p_{nk_n}$. □

Having obtained a method to compute syzygy generators for sets representing weak Gröbner basis, we now let $P = \{p_1, p_2, \cdots, p_m\}$ be a set containing arbitrary polynomials in $B[X]$. Let $R = \{\rho_1, \cdots, \rho_n\}$ be a weak Gröbner basis of the ideal generated by P. Let us denote by S_P and S_G the module of syzygies of $(p_1, \cdots, p_m)$ and $(p_{11}, \cdots, p_{nk_n})$, respectively. We know how to find a generating system for S_G, and we are going to show how this can be used to obtain a generating system for S_P. Since F and G generate the same ideal I in $B[X]$, we have "forward" and "backward" transformations between these ideals, i.e., there exist $a_{ijk}, b_{kij} \in B[X]$ with

$$p_{ij} = \sum_{k=1}^{m} a_{ijk} p_k \quad \text{for} \quad 1 \leq i \leq n, 1 \leq j \leq k_i \text{ and}$$
$$p_k = \sum_{i,j} b_{kij} p_{ij} \quad \text{for} \quad 1 \leq k \leq m.$$

Composing these transformations, we obtain

$$p_k = \sum_{i,j} b_{kij} p_{ij} = \sum_{i,j} b_{kij} \sum_{l=1}^{m} a_{ijl} p_l = \sum_{l=1}^{m} (\sum_{i,j} b_{kij} a_{ijl}) p_l \tag{7}$$

for $1 \leq k \leq m$. Let δ_{ij} be the Kronecker symbol, i.e., $\delta_{ii} = 1$ and $\delta_{ij} = 0$ for $i \neq j$. We can rewrite equation 7 in the form

$$\sum_{l=1}^{m} (\delta_{kl} - \sum_{i,j} b_{kij} a_{ijl}) p_l = 0 \text{ for } 1 \leq k \leq m.$$

Now if we set

$$c_{kl} = \delta_{kl} - \sum_{i,j} b_{kij} a_{ijl} \text{ for } 1 \leq k, l \leq m.$$

then we see that the m-tuple $\vec{c_k} = (c_{k1}, \cdots, c_{km})$ is an element of S_P for $1 \leq k \leq m$.

To obtain the desired generating system for S_P, we set $A = \{\vec{c_1}, \cdots, \vec{c_m}\}$. Let $B = \{\vec{d_1}, \cdots, \vec{d_s}\}$ be a generating system for the $B[X]$-module S_G of syzygies of $(p_{11}, \cdots, p_{nk_n})$. Hence we have for $1 \leq k \leq s$

$$\begin{aligned} \Omega &= \sum_{i,j} d_{kij} p_{ij} \\ &= \sum_{i,j} d_{kij} \sum_{l=1}^{m} a_{ijl} p_l \\ &= \sum_{l=1}^{m} (\sum_{i,j} d_{kij} a_{ijl}) p_l. \end{aligned}$$

Let $\vec{d_k}^* = (d^*_{k1}, \cdots, d^*_{km})$ where $d^*_{kl} = \sum_{i,j} d_{kij} a_{ijl}$. Let $B^* = \{\vec{d_1}^*, \cdots, \vec{d_s}^*\}$. Clearly B^* is a transformation of B to a set of solutions of S_P and $B^* \subset S_P$. We claim that $A \cup B^*$ is a generating set for the $R[X]$-module S_P of syzygies of $(p_1, \cdots, p_m)$.

We have already seen that $A \cup B^* \subset S_P$. In order to show that $A \cup B^*$ generates S_P as $B[X]$-module, let $\vec{h} = (h_1, \cdots, h_m) \in S_P$ be arbitrary and define $\vec{h}_* = (h_{*11}, \cdots, h_{*nk_n})$ by

$$h_{*ik} = \sum_{j=1}^{m} h_j b_{jik}$$

Then $\vec{h} \in S_P$ implies

$$\begin{aligned} \Omega &= \sum_{j=1}^{m} h_j p_j = \sum_{j=1}^{m} h_j \sum_{i,k} b_{jik} p_{ik} \\ &= \sum_{i,k} (\sum_{j=1}^{m} h_j b_{jik}) p_{ik} = \sum_{i,k} h_{*ik} p_{ik} \end{aligned}$$

and so $\vec{h}_* \in S_G$. Now S_G is generated by B, and so $\vec{h}_*$ is a linear combination of elements of B, i.e., there exist $q_1, \cdots, q_s \in B[X]$ with

$$\vec{h}_* = \sum_{l=1}^{s} q_l \vec{d_l}.$$

Define $\vec{k} = (k_1, \cdots, k_m)$ by setting

$$k_i = \sum_{k,l} h_{*kl} a_{kli} \text{ for } 1 \leq i \leq m.$$

Then

$$k_i = \sum_{k,l} (\sum_{j=1}^{s} q_j d_{jkl}) a_{kli} = \sum_{j=1}^{s} q_j \sum_{k,l} d_{jkl} a_{kli} = \sum_{j=1}^{s} q_j d^*_{ji}$$

and so $\vec{k} \in (B^*)$. We claim that $\vec{h} - \vec{k}$ (where subtraction is performed componentwise) is in (A). Indeed,

$$\begin{aligned} h_i \ominus k_i &= h_i \ominus \sum_{k,l} h_{*kl} a_{kli} \\ &= h_i \ominus \sum_{k,l} (\sum_{j=1}^{m} h_j b_{jkl}) a_{kli} \\ &= h_i \ominus \sum_{j=1}^{m} h_j (\sum_{k,l} b_{jkl} a_{kli}) \\ &= \sum_{j=1}^{m} h_j (\delta_{ij} \ominus \sum_{k,l} b_{jkl} a_{kli}) \;=\; \sum_{j=1}^{m} h_j c_{ji} \end{aligned}$$

for $1 \leq j \leq m$, and so $\vec{h} \ominus \vec{k} \in (A)$. Together, we conclude that $\vec{h} \in (A)$. This completes the proof. □

To complete the above proof, we use the fact that if ρ_i and ρ_j are two rules in the weak Gröbner basis, then there is also a rule ρ_k in the basis which reduces all instances of $(\vec{a_i} \cdot \vec{y_1} + \vec{a_j} \cdot \vec{y_2}) lcm(t_i, t_j)$. This fact is a consequence of the following lemma.

Lemma 35 *Let B be an arbitrary ring, and $k > 1$. If for ideals $I_1, \ldots, I_k \subset B$, it is the case that $I_1 \cup \ldots \cup I_k = \mathrm{Id}(I_1 \cup \ldots \cup I_k)$, then $I_1 \cup \ldots \cup I_k - I_i \subset I_i$ for some i.*

Proof. If not then let $g_i \in I_1 \cup \ldots \cup I_k - I_i$ for all i. The element $g_1 + \ldots + g_k \in B$ does not belong to any I_i, but it belongs to $Id(I_1 \cup \ldots \cup I_k)$. □

Theorem 22 *If B is a commutative Noetherian ring with unit that satisfies the conditions $(B1) - (B5)$, then the same is true for the polynomial ring $B[X]$.*

Proof. Suppose B is a commutative Noetherian ring with 1 satisfying the ideal membership and syzygies assumption mentioned above. Then clearly $B[X]$ too is a commutative ring with 1. Ideal membership in $B[X]$ can be decided by constructing a weak Gröbner basis. The Noetherian property follows from the Hilbert's basis theorem. The above lemma shows that computation of syzygies is possible too, thus completing the proof. □

A similar result was proved by Richman [70], but the proof was not constructive.

Using the results of the previous sections, we can now compute weak Gröbner bases in the polynomial ring $K[X_0, \ldots, X_n]$, where K is a field (or a commutative ring satisfying other conditions, or a PID, or a Euclidean Domain) by considering this polynomial ring as the ring $K[X_0][X_1] \ldots [X_n]$.

8.9 Summary

Algorithms for constructing polynomial ideal bases have been extensively studied, cf. Becker and Weispfenning [17]. Our work generalizes previous results for commutative polynomials in that we consider a larger class of coefficient domains (including domains with zero divisors). The basic conditions we impose on a coefficient domain are (i) decidability of ideal membership (for finitely generated ideals) and (ii) computability of syzygy generators. In addition, termination of the completion process is guaranteed only if the coefficient domain is Noetherian. These assumptions are shown to be the weakest required to obtain a weak Gröbner cases construction procedure.

We believe that most of the rewrite-based methods for constructing ideal bases for commutative polynomial rings can be described within our formalism. In particular, this is the case for Buchberger's original algorithm and for the method proposed by Pan [66] for polynomials over principal ideal domains. Buchberger's original algorithm works for coefficient domains that are fields, in which case we also obtain (strong) Gröbner bases (and not just weak Gröbner bases), because extended equations and union critical pairs become irrelevant. We also have shown that our method produces a Gröbner basis whenever a partial order on the coefficient domain is given, such that for any ideal I, every remainder class of I has a unique minimal element. This additional assumption is also easily seen to be the weakest required to get a strong basis. It remains to be investigated to what extent our techniques can be carried over to non-commutative polynomials.

There have also been various attempts at characterizing the class of rings that admit a Gröbner basis algorithm. These characterizations typically involve as one key parameter a (axiomatically characterized) reduction relation; see Buchberger [25] and Stifter [77] for one notion of "reduction rings" and Madlener and Reinert [58] for another. We have taken a different approach that employs only standard associative-commutative rewriting (with constraints). We believe our results indicate that in the case of commutative polynomial rings standard rewriting is sufficient. Non-standard reduction relations have been used, for instance, by Trinks [80] and Möller [61]. There several rewrite rules may be simultaneously applied to a single monomial. We actually obtain a similar effect, to some extent, via constraints. For example, a weak basis for the two polynomial equations $2X = 0$ and $3X = 0$ over the $\mathcal{Z}$ consists of just one rule $xX \to \Omega$ `if` $x = 2y_1 + 3y_2$ (which is obtained via a union critical pair and renders the other two rules redundant). From a computational point of view (and probably also

conceptually) constrained rewriting appears to be preferable to arbitrary "simultaneous reduction" by several rules. Other features of constraints, such as the possibility of delaying constraint solving, and encoding additional optimizations (as suggested by the inequality constraints that reduce the number of possible superpositions in our presentation) might also be of interest for implementation purposes.

Chapter 9

Conclusions

Methods for mechanical theorem proving have recently attracted a lot of interest from the verification community. Techniques to prove correctness of specifications (of hardware or software systems) that rely on exhaustive state space enumeration are limited to analysis of only finite state systems. On the other hand, methods based on symbolic representation of state space are capable of handling all kinds of systems—finite or not. Theorem proving naturally comes up when reasoning with such symbolic presentations.

More recently, theorem proving technology is also being integrated with program analysis methods. For instance, static analysis techniques can become greatly useful if syntactic analysis is enhanced with certain amount of symbolic reasoning capability [19]. Most of such applications however need assistance with symbols from specific domains or theories like real arithmetic, or deal with structures satisfying certain properties like queues, etc.

General purpose theorem provers have been found lacking in predictability. In fact, they seem to not terminate (or take an arbitrarily long time to finish) even when challenged with simple formulas that fall in well-known decidable fragments of logics or theories. Hence, most of the general purpose theorem provers that have been used for such applications have integrated decision procedures in them. But there is no standard way to achieve such an integration, and there are several independent and diverse efforts.

This thesis presents one of the first attempts towards formalizing the problem of combination and proposes a uniform solution. The essential idea behind integrating useful decision procedures into a deduction system involves identifying and separating

the logical or deductive aspect of a procedure from the non-deductive (like computational) part. The latter part can usually be incorporated using constraint based formalisms. A description of the former often requires the adoption of the general inference rules. Obtaining decision procedures for various different theories using the abstract combination result (presented in Chapter 3) leads essentially to a presentation of the decision procedure in terms of specialized inference rules, thus proposing theoretical results that help in tight integration of decision procedures with general purpose provers.

For instance, Chapter 4 discusses the problem of construction of congruence closure, and presents a set of abstract transition rules, derived from the rules for standard completion, to construct congruence closures. Congruence closure algorithms have been central in the combination and integration of decision procedures. Our description not only brings out a means for integration of congruence closure with general purpose provers, but also shows how other interesting theories can be added to the combination as well. In particular, ground AC theories can be added to the theory of ground equational theories naturally—again using the abstract combination result [11]. This extension is worked out in detail in Chapter 6. Many other theories can be integrated as well—but we chose not to go into the details of others. Instead we only mention conditions that require to be satisfied for proper integration.

As a last non-trivial illustration of our combination result, we describe an abstract completion-like procedure for computing Gröbner bases for polynomial ideals over certain very general rings, see Chapter 8. Thus, the theory of polynomial ideals can be seen to satisfy the conditions required for integration specified in Chapter 3.

The work presented in the thesis has several interesting fall-outs. In most instances, an abstract description via transition rules of algorithms leads to suitable generalizations of the algorithm. As a prime example of this fact, we note that our procedure to construct Gröbner bases can handle polynomial rings over various coefficient domains [12]. Originally, the Gröbner basis algorithm was proposed to deal with polynomial rings over fields. Later, when there were applications dealing with the polynomial ring over integers ($\mathcal{Z}[X_1, \ldots, X_n]$), there were techniques developed to handle polynomial rings over Euclidean domains and principal ideal domains. Our description clearly brings out the minimal set of assumptions that the coefficient domain needs to satisfy in order to admit an algorithm for construction of Gröbner bases. One example of a ring that can be handled now is the ring $K[X_1, X_2][X_3, \ldots, X_n]$, where the ring $K[X_1, X_2]$ forms the coefficient domain.

Another powerful tool that comes out of this thesis is the use of extended signatures. Introduction of new constants into the signature as names for subterms allows for greater flexibility (in choosing term orderings for example) and formally capturing the notion of "sharing" [13]. Term dag based algorithms for computing congruence closure are naturally cast in the framework of abstract congruence closure using this observation, as is shown in Chapter 5. Moreover, applications or algorithms that crucially depend on structure sharing for efficiency can now be abstractly described using an extended signature. Efficient algorithms for syntactic unification is one such example (Section 7.2.4. Chapter 7 describes two other applications that essentially use the abstract concept of congruence closure to obtain a class of algorithms for non-oblivious normalization and rigid E-unification. In particular, a specialization of our general method for normalization yields a new algorithm for efficient normalization of a term with respect to a *convergent* system [10]. Similarly, we also obtain a description of a class of algorithms for rigid E-unification using abstract congruence closure, superposition and paramodulation that are more general in scope than previously known algorithms [79].

Bibliography

[1] W. W. Adams and P. Loustaunau. *An Introduction to Gröbner Bases*, volume 3 of *Graduate Studies in Mathematics*. American Mathematical Society, 1994.

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

[3] F. Baader and K. U. Schulz. Unification in the union of disjoint equational theories: Combining decision procedures. In C. Tinelli and M. Harandi, editors, *Proceedings of the 11th Int. Conference on Automated Deduction*, pages 50–65. Springer-Verlag, 1992. volume 607 of Lecture Notes in AI.

[4] F. Baader and W. Snyder. Unification theory. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*. Elsevier Science Publisher B.V., 2000.

[5] L. Bachmair. *Canonical Equational Proofs*. Birkhäuser, Boston, 1991.

[6] L. Bachmair and N. Dershowitz. Completion for rewriting modulo a congruence. *Theoretical Computer Science*, 67(2 & 3):173–201, October 1989.

[7] L. Bachmair and N. Dershowitz. Equational inference, canonical proofs, and proof orderings. *J. of the Association for Computing Machinery*, 41:236–276, 1994.

[8] L. Bachmair and H. Ganzinger. Buchberger's algorithm: A constraint-based completion procedure. In *Constraints in Computational Logic*, Lect. Notes in Comput. Sci., Berlin, 1994. Springer-Verlag.

[9] L. Bachmair and D. Plaisted. Termination orderings for associative-commutative rewriting systems. *J. Symbolic Computation*, 1:329–349, 1985.

[10] L. Bachmair, C.R. Ramakrishnan, I.V.Ramakrishnan, and A. Tiwari. Normalization via rewrite closures. In P. Narendran and M. Rusinowitch, editors, *Rewriting Techniques and Applications, RTA 1999*, volume 1631 of *Lecture Notes in Computer Science*, pages 190–204, Trento, Italy, July 1999. Springer-Verlag.

[11] L. Bachmair, I.V. Ramakrishnan, A. Tiwari, and L. Vigneron. Congruence closure modulo Associativity-Commutativity. In H. Kirchner and C. Ringeissen, editors, *Frontiers of Combining Systems, Third International Workshop, FroCoS 2000*, volume 1794 of *Lecture Notes in Artificial Intelligence*, pages 245–259, Nancy, France, March 2000. Springer-Verlag.

[12] L. Bachmair and A. Tiwari. D-bases for polynomial ideals over commutative noetherian rings. In H. Comon, editor, *Rewriting Techniques and Applications, RTA 1997*, volume 1103 of *Lecture Notes in Computer Science*, pages 113–127, Sitges, Spain, July 1997. Springer-Verlag.

[13] L. Bachmair and A. Tiwari. Abstract congruence closure and specializations. In D. McAllester, editor, *Conference on Automated Deduction, CADE 2000*, volume 1831 of *Lecture Notes in Artificial Intelligence*, pages 64–78, Pittsburgh, PA, June 2000. Springer-Verlag.

[14] L. Bachmair and A. Tiwari. Congruence closure and syntactic unification. In C. Lynch and D. Narendran, editors, *14th International Workshop on Unification*, 2000.

[15] A. M. Ballantyne and D. S. Lankford. New decision algorithms for finitely presented commutative semigroups. *Comp. and Maths. with Appls.*, 7:159–165, 1981.

[16] 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.

[17] T. Becker and V. Weispfenning. *Gröbner bases: a computational approach to commutative algebra*. Springer-Verlag, Berlin, 1993.

[18] 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.

[19] S. Bensalem, et.al. An overview of SAL. In B.L. De Vito, editor, *Langley Workshop on Formal Methods, LFMW 2000*, 2000.

[20] N. S. Bjorner. *Integrating decision procedures for temporal verification*. PhD thesis, Stanford University, 1998.

[21] R. S. Boyer and J. S. Moore. *A computational logic*. Academic Press, New York, 1979.

[22] R. S. Boyer and J. S. Moore. Integrating decision procedures into heuristic theorem provers: A case study of linear arithmetic. In *Logic and Acquisition of Knowledge, Machine Intelligence*, pages 83–124, 1984.

[23] R. S. Boyer and J. S. Moore. *A computational logic handbook*. Academic Press, Boston, 1988.

[24] B. Buchberger. *An algorithm for finding a basis for the residue class ring of a zero-dimensional ideal*. PhD thesis, University of Innsbruck, Austria, 1965.

[25] B. Buchberger. A critical-pair completion algorithm for finitely generated ideals in rings. In *Proc. Logic and Machines: Decision Problems and Complexity*, volume 171 of *Lect. Notes in Comput. Sci.*, pages 137–161, Berlin, 1983. Springer-Verlag.

[26] B. Buchberger. Gröbner bases: An algorithmic method in polynomial ideal theory. In N. K. Bose, editor, *Recent Trends in Multidimensional Systems Theory*, pages 184–232. Reidel, 1985.

[27] B. Buchberger. Symbolic computation: Computer algebra and logic. In F. Baader and K. U. Schulz, editors, *Frontiers of Combining Systems*, pages 193–219, 1996. Kluwer Academic Publishers.

[28] R. Bündgen. Buchberger's algorithm: The term rewriter's point of view. *Theoretical Computer Science*, 1996. To appear.

[29] L. P. Chew. An improved algorithm for computing with equations. In *21st Annual Symposium on Foundations of Computer Science*, 1980.

[30] L. P. Chew. *Normal forms in term rewriting systems*. PhD thesis, Purdue University, 1981.

[31] E. Clarke and X Zhao. Analytica - a theorem prover in mathematica. In C. Tinelli and M. Harandi, editors, *Proceedings of the 11th Int. Conference on Automated Deduction*, pages 50–65. Springer-Verlag, 1992. volume 607 of Lecture Notes in AI.

[32] E. Clarke and X Zhao. *Analytica - An experiment in combining theorem proving and symbolic computation.* Technical Report CMU-CS-92-17, School of Computer Science, Carnegie Mellon University, 1992.

[33] M. Clavel and et. al. *Maude: Specification and Programming in Rewriting Logic.* http://maude.csl.sri.com/manual/, SRI International, Menlo Park, CA, 1999.

[34] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available on: `http://www.grappa.univ-lille3.fr/tata`, 1997.

[35] D. Cyrluk, P. Lincoln, and N. Shankar. On Shostak's decision procedure for combination of theories. In M. A. McRobbie and J.K Slaney, editors, *Proceedings of the 13th Int. Conference on Automated Deduction*, pages 463–477, 1996. Vol. 1104 of *Lecture Notes in Computer Science*, Springer, Berlin.

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

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

[38] N. Dershowitz and J. P. Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science (Vol. B: Formal Models and Semantics)*, Amsterdam, 1990. North-Holland.

[39] N. Dershowitz and Z. Manna. Proving termination with multiset orderings. *Communications of the ACM*, 22(8):465–476, 1979.

[40] E. Domenjoud and F. Klay. Shallow ac theories. In *Proceedings of the 2nd CCL Workshop*, La Escala, Spain, September 1993.

[41] P. J. Downey, R. Sethi, and R. E. Tarjan. Variations on the common subexpressions problem. *J. of the Association for Computing Machinery*, 27(4):758–771, 1980.

[42] T. Evans. The word problem for abstract algebras. *Journal of London Mathematical Society*, 26:64–71, 1951.

[43] T. Evans. Embeddability and the word problem. *Journal of London Mathematical Society*, 28:76–80, 1953.

[44] T. Evans. Word problems. *Bulletin of the American Mathematical Society*, 84(5):789–802, 1978.

[45] 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.

[46] 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.

[47] 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.

[48] B. Gramlich. *Termination and confluence properties of structured rewrite systems*. PhD thesis, Universitat Kaiserslautern, 1996.

[49] J.-P. Jouannaud and C. Kirchner. Solving equations in abstract algebras: A rule-based survey of unification. In J.-L. Lassez and G. Plotkin, editors, *Computational Logic: Essays in Honor of Alan Robinson*, pages 257–321. MIT Press, Cambridge, MA, 1991.

[50] A. Kandri-Rody and D. Kapur. Computing a Gröbner basis of a polynomial ideal over a Euclidean domain. *J. Symbolic Computation*, 6:37–57, 1988.

[51] D. Kapur. Shostak's congruence closure as completion. In H. Comon, editor, *Rewriting Techniques and Applications, RTA 1997*, volume 1103 of *Lecture Notes in Computer Science*, pages 23–37, Sitges, Spain, July 1997. Springer-Verlag.

[52] C. Kirchner, H. Kirchner, and M. Rusinowitch. Deduction with symbolic constraints. Revue Française d'Intelligence Artificielle, 4:9–52, 1990.

[53] H. Kirchner and C. Ringeissen. *Frontiers of Combining Systems, Third International Workshop, FroCoS 2000*. LNAI 1794, Springer, Berlin, Hiedelberg, 2000.

[54] J. W. Klop. Term rewriting systems. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 1, chapter 6, pages 2–116. Oxford University Press, Oxford, 1992.

[55] E. 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.

[56] U. Koppenhagen and E. W. Mayr. An optimal algorithm for constructing the reduced Gröbner basis of binomial ideals. In Y. D. Lakshman, editor, *Proceedings of the International Symposium on Symbolic and Algebraic Computation*, pages 55–62, 1996.

[57] C. Lynch. Paramodulation without duplication. In D. Kozen, editor, *Proceedings 10th IEEE Symposium on Logic in Computer Science*, pages 167–177, 1995. San Diego.

[58] K. Madlener and B. Reinert. Computing Gröbner bases in monoid and group rings. In *Proc. ISSAC 93*, pages 254–263, 1993.

[59] C. Marche. On ground AC-completion. In R. V. Book, editor, *4th International Conference on Rewriting Techniques and Applications*, pages 411–422, 1991. Vol. 488 of *Lecture Notes in Computer Science*, Springer, Berlin.

[60] E. W. Mayr and A. R. Meyer. The complexity of the word problems for commutative semigroups and polynomial ideals. *Advances in Mathematics*, 46:305–329, 1982.

[61] H. M. Möller. On the construction of Gröbner bases using syzygies. *J. Symbolic Computation*, 6:345–359, 1988.

[62] P. Narendran and M. Rusinowitch. Any ground associative-commutative theory has a finite canonical system. In R. V. Book, editor, *4th International Conference on Rewriting Techniques and Applications*, pages 423–434, 1991. Vol. 488 of *Lecture Notes in Computer Science*, Springer, Berlin.

[63] G. Nelson and D. Oppen. Simplification by cooperating decision procedures. *ACM Transactions on Programming Languages and Systems*, 1(2):245–257, October 1979.

[64] G. Nelson and D. Oppen. Fast decision procedures based on congruence closure. *Journal of the Association for Computing Machinery*, 27(2):356–364, April 1980.

[65] D. Oppen. Complexity, convexity and combinations of theories. *Theoretical Computer Science*, 12:291–302, 1980.

[66] L. Pan. On the D-bases of polynomial ideals over principal ideal domains. *J. Symbolic Computation*, 7:55–69, 1988.

[67] G. E. Peterson and M. E. Stickel. Complete sets of reductions for some equational theories. *J. of the Association for Computing Machinery*, 28(2):233–264, April 1981.

[68] D. Plaisted and A. Sattler-Klein. Proof lengths for equational completion. *Information and Computation*, 125:154–170, 1996.

[69] E. Poll and S. Thompson. Integrating computer algebra and reasoning through the type system of Aldor. In H. Kirchner and C. Ringeissen, editors, *Frontiers of Combining Systems, Third International Workshop, FroCoS 2000*, pages 136–150, 2000. LNAI 1794.

[70] F. Richman. Constructive aspects of Noetherian rings. *Proc. of the American Mathematical Society*, 44(2):436–441, 1974.

[71] C. Ringeissen. Cooperation of decision procedures for the satisfiability problem. In F. Baader and K. U. Schulz, editors, *Frontiers of Combining Systems*, pages 121–139, 1996. Kluwer Academic Publishers.

[72] A. Rubio and R. Nieuwenhuis. A precedence-based total AC-compatible ordering. In C. Kirchner, editor, *Proceedings of the 5 Intl. Conference on Rewriting Techniques and Applications*, pages 374–388, 1993. Vol. 690 of *Lecture Notes in Computer Science*, Springer.

[73] N. Shankar, S. Owre, and J. M. Rushby. *The PVS Proof Checker: A Reference Manual*. Computer Science Lab, SRI International, 1993.

[74] D. J. Sherman and N. Magnier. Factotum: Automatic and systematic sharing support for systems analyzers. In *Proc. TACAS, LNCS 1384*, 1998.

[75] R. E. Shostak. Deciding combinations of theories. *Journal of the ACM*, 31(1):1–12, 1984.

[76] W. Snyder. A fast algorithm for generating reduced ground rewriting systems from a set of ground equations. *Journal of Symbolic Computation*, 15(7), 1993.

[77] S. Stifter. A generalization of reduction rings. *J. Symbolic Computation*, 4:351–364, 1987.

[78] C. Tinelli and M. Harandi. A new correctness proof of the Nelson-Oppen combination procedure. In F. Baader and K. U. Schulz, editors, *Frontiers of Combining Systems*, pages 103–119, 1996. Kluwer Academic Publishers.

[79] A. Tiwari, L. Bachmair, and H. Ruess. Rigid E-unification revisited. In D. McAllester, editor, *Conference on Automated Deduction, CADE 2000*, volume 1831 of *Lecture Notes in Artificial Intelligence*, pages 220–234, Pittsburgh, PA, June 2000. Springer-Verlag.

[80] W. Trinks. On B. Buchberger's method for solving algebraic equations. *J. Number Theory*, 10(4):475–488, 1978. (German).

[81] R. M. Verma. A theory of using history for equational systems with applications. *J. of the Association for Computing Machinery*, 42:984–1020, 1995.

[82] R. M. Verma and I. V. Ramakrishnan. Nonoblivious normalization algorithms for nonlinear systems. In *Proc. of the Int. Colloquium on Automata, Languages and Programming*, New York, 1990. Springer-Verlag.