J. Symbolic Computation (1989) **8**, 449–477

Unification in Boolean Rings and Abelian Groups

ALEXANDRE BOUDET JEAN-PIERRE JOUANNAUD
MANFRED SCHMIDT-SCHAUSS†

LRI, Université Paris-Sud, Bât 490
91405 ORSAY Cedex, France
†Fachbereich Informatik, Postfach 3049, Universität Kaiserslautern
6750 Kaiserslautern, West Germany

A complete unification algorithm is presented for the combination of two theories E in T(F,X) and E' in T(F',X) where F and F' denote two disjoint sets of function symbols. E and E' are arbitrary equational theories for which are given, for E: a complete unification algorithm for terms in $T(F \cup C, X)$, where C is a set of free constants and a complete constant elimination algorithm for eliminating a constant c from a term s; for E': a complete unification algorithm. E' is supposed to be cycle free, i.e., equations $x = t$ where x is a variable occurring in t have no E'-solution. The method adapts to unification of infinite trees. It is applied to two well-known open problems, when E is the theory of Boolean Rings or the theory of Abelian Groups, and E' is the free theory. Our interest to Boolean Rings originates in VLSI verification.

1. Introduction

This section introduces our main notations and definitions.

Given a denumerable set X of variables, and a graded set F of function symbols, $T(F,X)$ denotes the *free algebra over* X, also called *term algebra.* Elements of $T(F,X)$ are called *terms.* The set of variables occurring in t is denoted by $V(t)$. We use sequences of integers to denote positions inside a term. $Dom(t)$ is the set of positions of a term t. We write $p < q$ if the position p is a proper prefix of the position q. $t.head$ denotes the top function symbol of t.
Substitutions are endomorphisms of $T(F,X)$ with a finite domain. We denote by $t\sigma$ the application of a substitution σ to a term t. $Dom(\sigma) = \{x \in X \mid x\sigma \neq x\}$ is the domain of σ, and $V_R(\sigma) = \{y \mid \exists x \in Dom(\sigma),\ y \in V(x\sigma)\}$ is the set of variables in the range of σ. The restriction of σ to $Y \subset X$ is denoted by $\sigma_{|Y}$.

Unification is a fundamental mechanism in computer science, especially in logic programming, automated theorem proving and artificial intelligence in gen-

0747-7171/89/110449+29 $03.00/0

eral: Given two terms s and t, a substitution σ is a *unifier* of s and t, or a *solution* of the equation $s = t$, if $s\sigma = t\sigma$. A substitution θ is an *instance* of a substitution σ if $\theta = \sigma\rho$ (using postfix notation) for some substitution ρ, and σ is said to be *more general* than θ. It is well known (Robinson 65) that there exists a most general solution to any solvable equation $s = t$, called *most general unifier* of s *and* t, and denoted $MGU(s,t)$. Herbrand (Herbrand 30) and Martelli and Montanari (Martelli & Montanari 82) see unification as a simplification process, that transforms a set of equations into another simpler set by applying inference rules, until a solved form is obtained from which the most general unifier can be easily computed.

DEFINITION 1 *A* unification problem P *is a set of equations,* $\{s_i = t_i \mid i \in I\}$. *A unification problem* $\{x_i = t_i \mid i \in I\}$ *such that* $\forall i \neq j$ $x_i \neq x_j$ *and* $\forall i \forall j$ $x_i \notin V(t_j)$ *is said to be in* solved form.

An equation $s = t$ is a particular case of unification problem. Another case is the MGU itself, which is a unification problem in solved form.

The use of unification in equational theories goes back to Plotkin (Plotkin 72) who introduced minimal complete sets of unifiers. Given an equational theory E, a substitution σ is an *E-unifier* of s and t, or an *E-solution* of the equation $s = t$ if $s\sigma =_E t\sigma$. Two substitutions σ and θ are said to be *equal modulo* E *over* $V \subseteq X$, denoted $\sigma =_E^V \theta$, if $\forall x \in V$, $x\sigma =_E x\theta$. A substitution θ is an *E-instance* of a substitution σ *over* V if $\theta =_E^V \sigma\rho$ for some substitution ρ, σ is said to be *more general than* θ *modulo* E, and we write $\sigma \leq_E^V \theta$. Unfortunately, most general solutions do not exist in general (since leq_E may not be well-founded) (Fages, Huet 1983; Baader 1989), and unification in an arbitrary theory E may even be undecidable.

DEFINITION 2 Σ *is a* complete set of unifiers *of* s *and* t modulo E *away from* $W \supseteq V(s) \cup V(t)$, *denoted* $CSU_E(s,t)$, *or simply* $CSU(s,t)$ *when* E *is known from the context, if* Σ *satisfies the following three properties:*

1. $\forall\sigma \in \Sigma$, $Dom(\sigma) \subseteq V(s) \cup V(t)$ *and* $I(\sigma) \cap W = \emptyset$ *(idempotency and protection of variables),*
2. $\forall\sigma \in \Sigma$, $s\sigma =_E t\sigma$ *(correctness),*
3. $\forall\sigma'$, $\sigma' s =_E \sigma' t \Rightarrow \exists\sigma \in \Sigma$ *such that* $\sigma \leq_E^{V(s)\cup V(t)} \sigma'$ *(completeness).*

In addition, $CSU(s,t)$ *is called a* complete set of most general unifiers *or* minimal $CSU(s,t)$ *if* $\sigma \leq_E^W \theta$ *for no* $\sigma, \theta \in CSU(s,t)$.

When it exists, $CSU(s,t)$ is defined up to isomorphism (Fages and Huet 83), and may be type zero, unitary, finite or infinite (Raulefs *et al.* 79). Many theories of interest, e.g., associativity and commutativity (denoted AC), have finite CSUs, and many algorithms computing such CSUs are known. For AC, see (Stickel 75) (Siekmann 78) (Fages 84) and (Kirchner 87).

Constructing these algorithms is a hard task, hence automating (even partially) their construction is a relevant question. (Fay 79), (Hullot 80) and (Jouannaud *et al.* 83) show how to obtain such algorithms for theories defined by a convergent set of rules. (Kirchner 86) shows how to compute unification algorithms for syntactic theories for which the decomposition schemes (called mutation) are computed from the axioms of the theory. (Yelick 85) as well as (Kirchner 85) and (Tidén 86) show how to combine unification algorithms for two disjoint theories E and E' in order to get a unification algorithm for the theory $E \cup E'$. These works all assume that E and E' are *collapse free.* Yelick and Kirchner assume in addition that all axioms in E and E' are *regular.*

DEFINITION 3 *An equational theory E is :*
•collapse free *if it does not contain axioms of the form $x = s$, with $x \in V(s)$,*
•regular *if $V(s) = V(t)$ $\forall s = t \in E$.*

It is remarkable that the proof of a unification algorithm for one AC operator is relatively easy, but becomes very difficult as soon as there are two such operators or some additional free operators. In particular, the termination of Stickel's algorithm has remained an open problem for close to a decade (Stickel 75), (Fages 84). This shows that the combination task is usually the hard part of a unification problem. Allowing collapse axioms should therefore permit using the technique in many cases where an algorithm is known for a simple case, but not for the general one. This includes the boolean ring case, as well as the case of abelian groups.

This paper shows how to solve the problem of combining a theory E with a theory E', under the assumptions that a complete unification algorithm allowing arbitrarily many free constants is known for E, as well as a constant elimination algorithm, and E' is *cycle free.*

DEFINITION 4 *An equational theory E is* cycle free *(or* simple*) if equations of the form $x = t$ where $x \in V(t)$ and $X \neq t$ have no E-solution.*

Note that cycle free theories cannot have non-regular or collapse axioms: if $s = t$ is a non regular axiom with $x \in V(s) \setminus V(t)$, then $\{x \mapsto s\}$ is a solution of $x = s$ and a collapse axiom $x = t$ where $x \in V(T)$ gives a straightforward solution to the equation $x = t$.

Our general result is applied to the combination of B, the theory of boolean rings (Martin & Nipkow 86, Büttner & Simonis 86), or AG, the theory of abelian groups (Lankford *et al.* 83), with the free theory.

Of course E and E' are supposed to be *consistent*, i.e., $T(F,X)/_{=_E}$ and $T(F',X)/_{=_{E'}}$ are not reduced to singletons.

Another solution to the same problem is given by Schmidt-Schauß(Schmidt-Schauß 88). This solution does not assume any hypothesis on E and E'. On the other hand, it is highly nondeterministic.

2. Combined theories

We first recall some notations and definitions. See also (Huet & Oppen 80) and (Dershowitz & Jouannaud 88).

Notations:

F and F' denote two disjoint sets of function symbols, and X a denumerable set of variables. E and E' denote two equational theories on respectively $T(F,X)$ and $T(F',X)$.

Usual unification algorithms in a theory E compute *idempotent unifiers*, i.e., substitutions σ such that $Dom(\sigma) \cap V_R(\sigma) = \emptyset$. Idempotent unifiers are indeed unification problems in solved form.

Our method for handling non-regular equational theories with collapse axioms is based on two main concepts. The first one is introduced now, the second one appears in section 4.

DEFINITION 5 *Given a constant symbol ω and an arbitrary term $s \in T(F,X)$, finding a complete set of E-solutions for the equation $s = \omega$ is called the* constant matching problem in E.

The constant matching problem is solvable in B, since a complete unification algorithm is known for B, which allows uninterpreted constant symbols (Martin & Nipkow 86; Büttner & Simonis 86). It is solvable in AG for the same reason (Lankford *et al.* 84). Note also that constant matching is impossible in the free theory.

EXAMPLE 1 *In B, $\{x \mapsto z, y \mapsto z\}$ is a most general match from $x + a + y$ to a.*

We can now make our basic assumptions precise.

Basic Assumptions:

- F and F' are disjoint sets,
- E and E' are consistent,
- E has a finite complete unification algorithm, for terms in $T(F \cup C,X)$ where C is an arbitrary set of free constants,
- E' is cycle free and has a finite complete unification algorithm.

- In case E is not regular, a complete E-elimination algorithm is assumed known for E, as defined in section 4, in order to be able to break compound cycles (involving E and E') down.

Terms in $T(F \cup F', X)$ may involve function symbols in both F and F', which forbids using E orE'-unification. Such terms will be decomposed into several terms of $T(F, X) \cup T(F', X)$. The following definitions are inspired from (Yelick 85).

DEFINITION **6**
Terms in $T(F, X)$ and $T(F', X)$ are said to be pure,
terms in $T(F \cup F', X) \setminus (T(F, X) \cup T(F', X))$ are said to be heterogeneous.

DEFINITION **7** *An equation $s = t$ is* pure *in E (respectively in E') if s and t are both pure in $T(F, X)$ (respectively in $T(F', X)$). An equation $s = t$ is* impure *if s and t are pure, $s \in T(F, X) \setminus X$ and $t \in T(F', X) \setminus X$. An equation $s = t$ is* heterogeneous *if s or t is heterogeneous.*

Notations: P_E and $P_{E'}$ denote pure unification problems for respectively E and E'. P_I denotes an impure unification problem. P_H denotes an heterogeneous unification problem. $x = y$ where x and y are variables will always be considered as a pure equation in E'.

EXAMPLE **2** *Let $F = \{+, *, 0, 1\}$ and $F' = \{f, g, a\}$, then the equations $x + 1 = 0$, and $x = y * z$ are pure in E, $f(x) = y + z$ is impure, and $f(0) = f(a)$ is heterogeneous.*

DEFINITION **8** *A unification problem P is said to be* separated *if it is of the form $P_E \uplus P_{E'}$, such that P_E and $P_{E'}$ are pure in E and E' respectively, P_E and $P_{E'}$ are both in solved form, there does not exist $x = s \in P_E$ and $x = t \in P_{E'}$ with $t \notin X$, and if $x = y \in P_{E'}$ for $x, y \in X$, one of x or y does not occur anywhere else in P.*

Pure equations will be solved in their own theory. Impure equations will be transformed into pure ones by constant matching in E. Heterogeneous equations will be transformed into pure and/or impure equations by decomposing heterogeneous terms into pure ones, as shown now:

DEFINITION **9** [Yelick] *Let $t \in T(F \cup F', X)$, and $\mathcal{F}$ be either F or F'. The* homogeneous part $\bar{t}^{\mathcal{F}}$ *of t with respect to $\mathcal{F}$ and the* preserving substitution $\rho_{\bar{t} \to t}$ *from $\bar{t}^{\mathcal{F}}$ to t are defined by:*
case
- $t \in X$

then $\bar{t}^{\mathcal{F}} = t$ *and* $\rho_{\bar{t} \to t} = \{\}$

• $t = f(t_1, \ldots, t_n)$ *and* $f \in \mathcal{F}$
then $\bar{t}^{\mathcal{F}} = f(\overline{t_1}^{\mathcal{F}}, \ldots, \overline{t_n}^{\mathcal{F}})$ *and* $\rho_{\bar{t} \to t} = \rho_{\overline{t_1} \to t_1} \cup \cdots \cup \rho_{\overline{t_n} \to t_n}$
else $\bar{t}^{\mathcal{F}} = x$ *and* $\rho_{\bar{t} \to t} = \{x \mapsto t\}$*, for a fresh variable* x
endcase
By convention, if $\mathcal{F}$ *is not specified,* $\bar{t}$ *will denote the homogeneous part of* t *with respect to the set of function symbols its root belongs to.*

EXAMPLE **3** *Let* $F = \{+, *, 0, 1\}$*, and* $t = f(a) * f(0) * x$*. Then* $\bar{t}^{\mathcal{F}} = x_1 * x_2 * x$ *and* $\rho_{\bar{t} \to t} = \{x_1 \mapsto f(a), x_2 \mapsto f(0)\}$*.*

To prove the completeness of our algorithm, we need a more subtle notion of homogeneity:

DEFINITION **10** *(Hsiang & Rusinowitch 87) Let* $<$ *be a simplification ordering total on* $T(F \cup F')$ *and* E *a set of equations. We say that* s rewrites *to* t *and we write* $s \longrightarrow_E t$ *if there exist an equation* $l = r$ *in* E*, a substitution* σ *and a position* p *in* $Dom(s)$ *such that the subterm of* s *at position* p *is equal to* $l\sigma$*, and* t *is the term obtained by replacing this occurrence of* $l\sigma$ *by* $r\sigma$ *in* s*, and* $s > t$*.* $l = r$ *is said to be a* rule *if* $l > r$*. We denote by* $s{\downarrow}$ *the normal-form of* s *for* $\longrightarrow_E$*. The existence of* $s{\downarrow}$ *is guaranteed by the well-foundedness of* $<$*.* E *is said to be* convergent *on ground terms if for arbitrary ground terms* s *and* t*,* $s{\downarrow} = t{\downarrow}$ *iff* s *and* t *are equal under the equational theory* $=_E$*. A proof* $s \to \cdots \to \cdots \leftarrow \cdots \leftarrow t$ *where* s *and* t *both rewrite in a finite number of steps on a same term is called a* rewrite proof*.*

Applying unfailing completion (Bachmair *et al.* 86; Hsiang & Rusinowitch 87), to $E \cup E'$ yields a (possibly infinite) convergent set R of equations for semi-deciding $=_{E \cup E'}$ on ground terms. Since $F \cap F' = \emptyset$, no overlap is possible between equations of E and E', hence $R = R_E \cup R_{E'}$ where the equations in R_E (resp. in $R_{E'}$) are pure in E (resp. in E'). By extending the ordering $>$ to $T(F \cup F', X)$, variables being now treated as constants, we get a semi-decision procedure for $=_{E \cup E'}$ on terms in $T(F \cup F', X)$. The R-normal form of a term t will be denoted by $t\downarrow$. The R-normal form of a substitution σ is $\sigma \downarrow = \{x \mapsto (x\sigma)\downarrow\}_{x \in Dom(\sigma)}$.

DEFINITION **11** *Let* $U \subset X$ *be a set of variables such that there is a one-to-one mapping* h *from* $T(F \cup F', X)/_{=_{E \cup E'}}$ *to* U*. By convention,* $h(t)$ *denotes the image by* h *of the class of* t *in* $T(F \cup F', X)/_{=_{E \cup E'}}$*. Let* $t \in T(F \cup F', X)$ *and* $\mathcal{F} = F$ *or* F'*. The* U-homogeneous part $\bar{\bar{t}}^{\mathcal{F}}$ *of* t *with respect to* $\mathcal{F}$ *is defined by:*
case
• $t \downarrow \in X$
then $\bar{\bar{t}}^{\mathcal{F}} = t \downarrow$
• $t \downarrow = f(t_1, \ldots, t_n)$ *and* $f \in \mathcal{F}$
then $\bar{\bar{t}}^{\mathcal{F}} = f(\overline{\overline{t_1}}^{\mathcal{F}}, \ldots, \overline{\overline{t_n}}^{\mathcal{F}})$

else $\overline{\overline{t}}^{\mathcal{F}} = h(t)$
endcase

The U-homogeneous part $\overline{\overline{\sigma}}^{\mathcal{F}}$ *of a substitution* σ *with respect to a set* $\mathcal{F}$ *of function symbols is* $\overline{\overline{\sigma}}^{\mathcal{F}} = \{x \mapsto \overline{\overline{x\sigma}}^{\mathcal{F}}\}_{x \in Dom(\sigma)}$.

The universal preserving substitution *is* ρ_u *defined by* $\rho_u(h(t)) = t\downarrow$.

For all terms $t \in T(F \cup F', X \setminus U)$, for all substitution σ, for $\mathcal{F} = F$ or F', the following properties are satisfied:

- $t\downarrow = \overline{\overline{t}}^{\mathcal{F}} \rho_u$
- $\sigma\downarrow = \overline{\overline{\sigma}}^{\mathcal{F}} \rho_u$
- $t \in T(\mathcal{F}, X) \Rightarrow (t\sigma)\downarrow = (t\overline{\overline{\sigma}}^{\mathcal{F}})\downarrow \rho_u$
- $t \in T(F, X) \Rightarrow (t\sigma)\downarrow =_E t(\sigma\downarrow)$
- $t \in T(F', X) \Rightarrow (t\sigma)\downarrow =_{E'} t(\sigma\downarrow)$

Our unification algorithm is divided into two main steps: the first one called simplification transforms the original set of equations into a new equivalent separated and solved set P. This first step is mainly based upon slicing heterogeneous terms into pure ones and E-matching from a pure term to a variable considered as a constant. Once step 1 has been applied, either P is solved, or P admits cycles. The role of the second step is to break these cycles down by using a process called constant elimination.

For better readability, we will assume in the sequel that E and E' are unitary (i.e., for any s and t, $CSU(s,t)$ is either empty or reduced to a singleton). When this is not the case, all solutions in $CSU(s,t)$ must be collected. The algorithm can actually enumerate a complete set of solutions, even if the theory E is infinitary. This enumeration terminates iff E is finitary.

3. Simplification inference rules

The algorithm is given as a set of inference rules which transform a unification problem $P = P_E \cup P_{E'} \cup P_I \cup P_H$ into a new one $P' = P'_E \cup P'_{E'} \cup P'_I \cup P'_H$.

The following definition distinguishes several kinds of variables in a unification problem:

DEFINITION **12** *A variable* x *is* E-instanciated *(resp.* E'-instanciated*) if there exists an equation* $x = t \in P_E$ *(resp.* $x = t \in P_{E'}$ *with* $t \notin X$*). A variable is* shared *if it it occurs in* P_E *and in an equation* $s = t$ *of* $P_{E'}$ *where* s *and* t *are not both variables.*

3.1. THE INFERENCE RULES

The following set $\mathcal{S}$ of inference rules transforms an arbitrary unification problem into an equivalent separated one. If E is regular, this section subsumes (Yelick 85) by giving a much simpler algorithm and proof for combining disjoint regular collapse-free theories. In particular, it yields a new simple proof of completeness of AC-unification (Fages 84).

VA
$s = t \in P_H \quad \vdash \quad \{\bar{s} = \bar{t}, x_1 = u_1, \ldots, x_n = u_n\}$
if s or t is heterogeneous and
$\rho_{\bar{s} \to s} \cup \rho_{\bar{t} \to t} = \{x_1 \mapsto u_1, \ldots, x_n \mapsto u_n\}$.

E-Res
$P_E \quad \vdash \quad \{x_1 = s_1, \ldots, x_n = s_n\}$
if $\{x_1 \mapsto s_1, \ldots, x_n \mapsto s_n\}$ is a most general solution for the system P_E, not in solved form.

E'-Res
$P_{E'} \quad \vdash \quad \{y_1 = t_1, \ldots, y_n = t_n\}$
if $\{y_1 \mapsto t_1, \ldots, y_n \mapsto t_n\}$ is a most general solution for the system $P_{E'}$, not in solved form.

E-Match
$s = t \in P_I \quad \vdash \quad \{x_i = s_i\}_{i \in I} \cup \{z = t\}$
if $\{x_i \mapsto s_i\}_{i \in I}$ is a most general E-match from s to a new variable z.

Merge-E-Match
$(P_E \cup \{x = s\}) \cup P_{E'} \quad \vdash \quad P_E\sigma \cup \{x_i = s_i\}_{i \in I} \cup P_{E'}$
if P_E and $P_{E'}$ are both solved,
$x = t \in P_{E'}$ and $t \notin X$,
where $\sigma = \{x_i = s_i\}_{i \in I}$ is a most general E-match from s to x.

Var-Rep
$\{x = y\} \cup P \quad \vdash \quad \{x = y\} \cup P\{x \mapsto y\}$
if x and y occur in P and x is not shared or y is shared.

The rule **Merge-E-Match** preserves the sets of solutions but does not terminate. A slightly different version is presented in section 3.2. where **Merge-E-Match** is transformed into two different rules. These two rules implement a control that yields termination.

3.2. PARTIAL CORRECTNESS

We show that all rules preserve the set of solutions of a unification problem.

Lemma 1 **VA** *preserves the sets of solutions.*

proof : Correctness: Let s and t be two terms, let γ be the preserving substitution $\gamma = \rho_{\overline{s}\to s} \cup \rho_{\overline{t}\to t}$. If θ is a solution to $\overline{s} =_T \overline{t}$, and σ unifies θ and γ, then σ is a solution to $s =_T t$.

$\overline{s}\,\theta =_T \overline{t}\,\theta$	by hypothesis
$\overline{s}\,\theta\sigma =_T \overline{t}\,\theta\sigma$	by application of σ
$\overline{s}\,\gamma\sigma =_T \overline{t}\,\gamma\sigma$	since σ unifies θ and γ
$s\sigma =_T t\sigma$	since $\overline{s}\,\gamma = s$ and $\overline{t}\,\gamma = t$.

Completeness: let ϕ be an $E \cup E'$-unifier of s and t. We show that there exists a substitution $\tilde{\phi}$ which is an $E \cup E'$-solution of

$$\{\overline{s} = \overline{t}, x_1 = u_1, \ldots, x_n = u_n\}$$

such that $\forall x \in Dom(\phi)$, $\tilde{\phi}(x) = \phi(x)$: let $\{p_1, \ldots, p_n\}$ be the positions of $\{x_1, \ldots, x_n\}$ in $\overline{s}$ or $\overline{t}$. We define $\tilde{\phi}$ by:
$\tilde{\phi}(x) = \phi(x)$ if $x \notin \{x_1, \ldots, x_n\}$
$\tilde{\phi}(x_i) = s\phi/p_i$ if $x_i \in Dom(\rho_{\overline{s}\to s})$
$\tilde{\phi}(x_i) = t\phi/p_i$ if $x_i \in Dom(\rho_{\overline{t}\to t})$
$\tilde{\phi} = \phi$ on variables of $V(s) \cup V(t)$ since $(V(s) \cup V(t)) \cap Dom(\rho_{\overline{s}\to s} \cup \rho_{\overline{t}\to t}) = \emptyset$.
$\tilde{\phi}$ is a solution of $\{x_i = u_i\}$ for $x_i \in Dom(\rho_{\overline{s}\to s})$:
$x_i\phi = (\overline{s}/p_i)\tilde{\phi} = (\overline{s}/p_i)\phi$ since $v(t_i) \cap Dom(\rho_{\overline{s}\to s} \cup \rho_{\overline{t}\to t}) = \emptyset$.
But $(s/p_i)\phi = (s\phi)/p_i = x_i\phi$.
Symmetrically, $\tilde{\phi}$ is a solution of $\{x_i = u_i\}$ for $x_i \in Dom(\rho_{\overline{t}\to t})$.
Finally, $\tilde{\phi}$ is a solution of $\overline{s} = \overline{t}$:

$$\begin{aligned}
\overline{s}\,\tilde{\phi} &= s[x_i]_{p_i}\tilde{\phi}\ (x_i \in Dom(\rho_{\overline{s}\to s}))\\
&= s\tilde{\phi}[x_i\tilde{\phi}]_{p_i}\tilde{\phi}\ (x_i \in Dom(\rho_{\overline{s}\to s}))\\
&= s\phi[x_i\tilde{\phi}]_{p_i}\tilde{\phi}\ (x_i \in Dom(\rho_{\overline{s}\to s}))\\
&= s\phi[s\phi/p_i]_{p_i}\tilde{\phi}\ (x_i \in Dom(\rho_{\overline{s}\to s}))\\
&= s\phi =_{E\cup E'} t\phi = \cdots = \overline{t}\,\tilde{\phi}
\end{aligned}$$

□

E-Res and **E'-Res** preserve solutions, as a corollary of the following lemma:

Lemma 2 *Assume that E and E' are consistent. Then $=_{E\cup E'}$ is a conservative extension of both $=_E$ and $=_{E'}$.*

proof : In (Tidén 87). We give a drastically simpler proof. Let $s, t \in T(F, X)$, assume $s =_{E\cup E'} t$, and consider the rewrite proof provided by $R = R_E \cup R_{E'}$. Since E and E' are consistent, R cannot contain any equation of the form $x \mapsto s$, with $x \notin V(s)$. Since $>$ is a simplification ordering, the equation $x = s$ with $x \in V(s)$ is actually the rule $s \to x$. Hence, if a ground term s rewrites to t with the equation $l = r$, then l is not a variable. Moreover, we can chose the ordering $>$ in such a way that pureterms in $T(F, X)$ are strictly smaller then heterogeneous terms. Hence a term in $T(F, X)$ must rewrite to another term in $T(F, X)$ using a rule in R_E. Since s and t are both pure in E, and $R_{E'}$ is pure in E', no equation of $R_{E'}$ applies and the proof is of the form :

$$s \underset{R_E}{\longrightarrow} s_1 \underset{R_E}{\longrightarrow} \cdots \underset{R_E}{\longrightarrow} s\downarrow = t\downarrow \underset{R_E}{\longleftarrow} \cdots \underset{R_E}{\longleftarrow} t$$

hence $s =_E t$. □

The following lemma shows that **E-match** preserves the sets of solutions:

Lemma 3 *Let $s \in T(F,X) \setminus X$ and $t \in T(F',X) \setminus X$. If $s\sigma =_{E \cup E'} t\sigma$, then there exist a variable x and two substitutions σ' and σ'' such that $\sigma =_{E \cup E'} \sigma'\sigma''_{|V(s) \cup V(t)}$ and $s\sigma' =_E x$, $x\sigma'' =_{E'} t\sigma$, $x\sigma''.head \in F'$.*

proof : Let σ be an arbitrary solution of $s = t$. We can assume without loss of generality that σ is in normal form for R. Consider a rewrite proof from $s\sigma$ to $t\sigma$:

$$s\sigma = u_1 \underset{R}{\rightarrow} u_2 \underset{R}{\rightarrow} \dots u_i \underset{R}{\rightarrow} u_{i+1} \underset{R}{\leftarrow} \dots \underset{R}{\leftarrow} u_n = t\sigma$$

This proof is of the form:

$$u_1 \underset{R_E}{\rightarrow} u_2 \underset{R_E}{\rightarrow} \dots u_i \underset{R_E}{\rightarrow} u_{i+1} \underset{R_{E'}}{\leftarrow} \dots \underset{R_{E'}}{\leftarrow} u_n$$

because the rules of R must apply at positions in the homogeneous parts of u_j, for $1 \leq j \leq n$ since σ is in normal form for R and the rules are pure. Since E' is collapse-free, $R_{E'}$ is also collapse-free, hence $u_{i+1}.head \in F'$. As a consequence, u_{i+1}must be a maximal subterm of $u_1 = s\sigma$ such that $u_{i+1}.head \in F'$.
Let $u_j = \overline{\overline{u_j}}^F \rho_u$ for $j \in [1..n]$. The proof between $s\sigma$ and u_{i+1} lifts to a proof on the $\overline{\overline{u_j}}^{\mathcal{F}}$s as follows:

$$s\overline{\overline{\sigma}}^F = \overline{\overline{u_1}}^F \underset{R_E}{\rightarrow} \overline{\overline{u_2}}^F \underset{R_E}{\rightarrow} \dots \overline{\overline{u_i}}^F \underset{R_E}{\rightarrow} \overline{\overline{u_{i+1}}}^F = x$$

Let $\sigma' = \overline{\overline{\sigma}}^F$, $\sigma'' = \rho_u$. Then $s\sigma' =_E x$ and $x\sigma''.head \in F'$, and $x\sigma'' =_{E'} t\sigma$ where $x = h(u_{i+1}) \in X$. □

COROLLARY **1** *The following transformation preserves the sets of solutions :*
Merge-E-Match

$\{x = s, x = t\} \quad \vdash \quad \{x_i = s_i\}_{i \in I} \cup \{x = t\}$
where $s \in T(F,X) \setminus X$ and $t \in T(F',X) \setminus X$
and $\{x_i \mapsto s_i\}_{i \in I}$ is a most general match from s to $x \notin V(s)$

proof :
Correctness: straightforward.
Completeness:
Since $=_{E \cup E'}$ is an equivalence, $\{x =_{E \cup E'} s, x =_{E \cup E'} t\} \Leftrightarrow \{s =_{E \cup E'} t, x =_{E \cup E'} t\}$.
By previous lemma, the $E \cup E'$-solutions of $s = t$ are of the form $\sigma'\sigma''$ where $s\sigma' =_E z$ and $z\sigma'' =_{E'} t\sigma'\sigma''$ and $z \in X$. But $x = t \in P_{E'}$ and since E' is cycle free, we can assume $x\sigma' \in X$. Hence the result. □

Unfortunately this rule causes non termination:
Let $E = B$ and let E' be the free theory. Consider the system

$$\{x = f(a), y = f(b), x = y + z\}$$

a most general match from $y + z$ to x is

$$\{y \mapsto x + z\}$$

(we do not include renamings in the example), and the equivalent obtained system P' is

$$\{x = f(a), y = f(b), y = x + z\}$$

which is symmetric to the input one.

A solution to the non-termination problem is provided by the following lemma :

Lemma 4
Assume $x = s_x$, $x = t_x$ and $y = t_y$ belong to P with $t_x, t_y \in T(F', X) \setminus X$ and $s_x \in T(F, X) \setminus X$. Assume **Merge-E-Match** *is applied to $\{x = s_x, x = t_x\}$ and yields a new equation $y = s_y$ with $s_y \in T(F, X) \setminus X$. Then the application of* **Merge-E-Match** *to $\{y = s_y, y = t_y\}$ can be restricted to the case where x is instanciated by a variable.*

proof : Let σ be a solution of P. By lemma 3 and corollary 1, every solution σ of P is of the form $\sigma = \sigma'_x \sigma''_x$ with $s_x \sigma'_x =_E x$ and $x\sigma''_x =_{E'} t\sigma$.
Assume that a most general E-match σ'_x from s_x to x is such that $y\sigma'_x = s_y \in T(F, X) \setminus X$, hence $y = s_y$ is one of the equations inferred by **Merge-E-Match**. Applying lemma 3 to $\{y = s_y, y = t_y\}$, we get $\sigma''_x = \sigma'_y \sigma''_y$ where σ'_y is an E-match from s_y to y. Since $x\sigma''_x =_{E'} t_x\sigma$, and $t_x\sigma.head \in F'$, $x\sigma'_y$ cannot be a term in $T(F, X) \setminus X$ (otherwise, E-equality steps would be necessary from $x\sigma'_x$ to $t_x\sigma$). □

The following two rules implement **Merge-E-Match** with the following control :

- When a merge on x is performed which E-instanciates previously E'-instanciated variables, the induced potential **Merge-E-Match** are treated immediately after.
- In order not to E-instanciate a variable x to which **Merge-E-Match** has just been applied, x will be marked and treated as a constant while performing these E-matches. By previous lemma, it is not necessary to consider the solutions that E-instanciate x, but treating marked variables as constants in the computation of the induced E-matches induces a loss of completeness: possible solutions that identify two marked variables are lost. Completeness is preserved by allowing to identify x to any E'-instanciated variable after marking it.

At this point we need modify the form of our formulae by stacking induced merges and marked variables in order to formally define the control we want. Items in the stack are pairs of an equation and a set of marked variables.

When P_E contains the equation $x = s$ and $P_{E'}$ the equation $x = t$ (for $s \in T(F,X) \setminus X$ and $t \in T(F',X) \setminus X$) the rule **Mem-Init** initializes the stack by pushing the equation $x = s$ and the set of marked variables $\{x\}$ onto it. **Mem-Init** also removes $x = s$ from P_E.

When the stack is not empty, the rule **Mem-Rec** has priority over all the other rules. **Mem-Rec** pops an item $< x = s, SMV >$ from the stack and computes σ, a $MGU_E(s,x)$ while treating the variables in SMV as constants. The substitution σ is then applied to P_E, and the equivalent equations are added to P. The possible merges that are induced by the resolution of the merge on x are then pushed onto the stack and x can be identified nondeterministically to an E'-instanciated variable z. Then $z = x$ is added to P and $\{z \mapsto x\}$ is applied to P_E and to the stack, and for all y_i such that y_i is E'-instanciated and $y_i\sigma \notin X$, $< y_i = y_i\sigma\{z \mapsto x\}, SMV \cup \{y_i\{z \mapsto x\}\} >$ is pushed onto the stack.

Finally:

Mem-Init

$x = s \in P_E \;\; x = t \in P_{E'}, Empty \;\; \vdash \;\; x = t \in P_{E'} \;\; , push < s = x, \{x\} >$ onto $Empty$

if $s \in T(F,X) \setminus X$ and $t \in T(F',X) \setminus X$

Mem-Rec

$P_E, push < s = x, SMV >$ onto S

$\vdash \quad P_E\sigma\theta \cup \{x_i = s_i\}_{i \in I} \cup \{z = x\}, push < y_1\sigma\theta = y_1\theta, SMV \cup \{y_1\theta\} >$ onto...onto

$push < y_n\sigma\theta = y_n\theta, SMV \cup \{y_n\theta\} >$ onto $S\sigma\theta$

if $\sigma \in CSU_E(s,x)$ where variables in SMV are treated as constants,

$\theta = \{z \mapsto x\}$ for some E'-instanciated variable z that is chosen nondeterministically,

$\{x_i = s_i\}_{i \in I}$ is the restriction of σ to non-E'-instanciated variables.

Of course, all other inference rules apply only if the stack is empty. We do not make this change for sake of readability.

Lemma 5 *The application of* **Mem-Init** *followed by the application of* **Mem-Rec** *as long as possible (denoted* **Mem-Init Mem-Rec****)* *preserves the set of solutions.*

proof : Completeness of **Mem-Init** is straightforward.
Completeness of **Mem-Rec** follows from lemma 4 and from the previous discussion about considering variables as constants. □

If E or E' is not unitary, remember that we must collect all possible solutions every time E-unification or E'-unification is performed.

Finally **Var-Rep** is trivially correct and complete since E-equality is a congruence. Hence:

PROPOSITION **1** *If S terminates, it returns a separated system equivalent to the input system.*

proof : All rules preserve solutions and if a problem is not separated, some rule must apply. □

Note that **Var-Rep** is necessary to make potential merges appear: If $P_E = \{z = s\}$ and $P_{E'} = \{x = t, x = z\}$ with $s \in T(F,X) \setminus X$ and $t \in T(F',X) \setminus X$, then **Var-Rep** yields $P_E = \{z = s\}$ and $P_{E'} = \{z = t, x = z\}$ to which **Mem-Init** applies. We will assume that **Var-Rep** has priority over the other rules. As a consequence, all cycles in a separated system, involve only equations of the form $x = t$ where $t \notin X$.

3.3. TERMINATION

The termination proof needs a few more definitions and notations:

DEFINITION **13**
The theory height $TH(s)$ *of a term s is defined as follows: if s is pure, then $TH(s) = 1$, else let $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\} = \rho_{\overline{s}\to s}$ and $TH(s) = 1 + max(TH(t_i))_{i\in[1..n]}$.*

DEFINITION **14** *$TH_{mul}(P)$ denotes the multiset of the theory heights of heterogeneous terms in P.*
$IE(P)$ denotes the number of impure equations in P,
$SV(P)$ denotes the number of shared variables in P,
$PM(P)$ denotes the number of potential applications of **Merge-E-Match**, *i.e., the number of variables x such that $x = s \in P_E$ and $x = t \in P_{E'}$ for some $s, t \notin X$.*
The weight *of P is*

$$W(P) =< TH_{mul}(P), IE(P), SV(P), PM(P) >$$

The ordering $<_W$ between unification problems is obtained by comparing lexicographically the components of $W(P)$, using $<_{mul}$, the multiset extension of the ordering $<$ on natural numbers, for the first component, and $<$ for the others.

Since $<$ is well founded, and multiset and lexicographic extentions preserve well founded orderings, then $<_W$ is well founded.

Lemma 6
If P_E is solved, then after applying Mem-init Mem-Rec*, *P_E is still solved.*
If applying **Var-Rep**, *then* **E'-Res** *after* Mem-init Mem-Rec* *increases $PM(P)$, then* **Var-Rep** *decreases strictly $SV(P)$.*

proof : **Mem-init** removes the equation $x = s$ from P_E, hence P_E is still in solved form. The domain of the substitution σ that is applied to P_E by **Mem-Rec** contains no E-instanciated variable, hence applying it to P_E, and adding the equivalent equations to P_E leaves P_E in solved form.
The application of **Mem-init Mem-Rec*** followed by **Var-Rep** and **E'-Res** may create a new potential application of **Mem-Init** only if two E'-instanciated variables x and y are identified, othewise, $P_{E'}$ would remain solved and no new E'-instanciated shared variable is created. In this case, x and y were both shared before applying **Mem-init Mem-Rec*** and when **Var-Rep** is applied to x and y, it decreases strictly $SV(P)$. □

PROPOSITION 2 *S terminates.*

proof : **VA** decreases strictly the multiset of theory heights of heterogeneous terms in P. Since no rule can increase $TH_{mul}(P)$, **VA** cannot be applied infinitely many times.
E-Match decreases the number of impure equations in P. Since only **VA** can add impure equations to P, **E-Match** cannot be applied infinitely many times.
We are left to show that the other rules terminate when neither **VA** nor **E-Match** are applied. Note that **VA** and **E-Match** are the only rules that may increase $SV(P)$. **Var-Rep** cannot be applied infinitely many times on a row, and it does not increase $W(P)$ and **E-Res** and **E'-Res** can apply only once in a row.
As long as **Mem-init Mem-Rec*** leaves $P_{E'}$ in solved form, $PM(P)$ decreases at every application of **Mem-init Mem-Rec***, when an application of **E'-Res** becomes possible, then it may increase $PM(P)$, but by lemma 6, $SV(P)$ has decreased in the meantime. □

As a corollary:

THEOREM 1 *Given a unification problem P, either S terminates with failure if there is no separated system equivalent to P or it returns a separated problem $P' = P_E \cup P_{E'}$ equivalent to P otherwise.*

If P' contains no cycle for the *occur-check* relation ($x < y$ if $x = s \in P$ with $s \notin X$ and $y \in V(s)$), then it is almost solved, else possible cycles must be broken up before to apply any replacement.

4. Breaking cycles

4.1. VARIABLE ELIMINATION

If there are cycles in the separated problem $P_E \uplus P_{E'}$ yielded by S, they must all be of the form:
$C = \{x_1 =_{E'} t_1, x_2 =_E t_2, \ldots, x_{2n} =_E t_{2n}\}$ where
$\forall i \in [1..2n]$, $x_{i+1(mod\, 2n)} \in V(t_i)$ and $\forall i \in [1..n]$,

$t_{2i} \in T(F, X) \setminus X$ and $t_{2i-1} \in T(F', X) \setminus X$,
since P is separated.

In order to handle non-regular axioms, as allowed in E, we introduce some more definitions.

DEFINITION **15** *(variable elimination)*
A substitution σ E-eliminates *a variable x from a term $t \in T(F, X)$ if $x\sigma \in X$ and there exists a term $u =_E t\sigma$ such that $x\sigma \notin V(u)$. We will also say that σ is an* E-eliminator *of x in t.*

EXAMPLE **4** *Let E be the theory of boolean rings (B). The substitution $\sigma = \{x \mapsto x', y \mapsto (x'+1) * y'\}$ is a B-eliminator of x in $t = x * y$, because $x\sigma = x'$ is a variable and $t\sigma =_B 0$ in which x' does not occur. The substitution $\{x \mapsto 0\}$ is not a B-eliminator of x in t because it maps x onto a non-variable term.*
A problem containg a cycle can be solved using variable elimination:
Let $P = P_E \cup P_{E'}$ where $P_E = \{z = x * y\}$ and $P_{E'} = \{x = f(z)\}$ be a unification problem in the combination of B and the free theory. There is a cycle in the graph of the occur-check relation since $z < x$ and $x < z$. If $\sigma = \{x \mapsto x', y \mapsto (x'+1)*y'\}$ is applied to P_E and the corresponding equations $x = x'$ and $y = (x'+1) * y'$ are added to P_E, the problem $P' = \{x = x', y = (x'+1) * y', z = 0, x = f(z)\}$ is obtained. Now after applying **Var-Rep** *we get $P'' = \{x = x', y = (x+1) * y', z = 0, x = f(z)\}$ which is a separated problem with no cycle.*

Variable elimination is the basic mechanism for solving cycles:

Lemma 7 *Let $t \in T(F, X)$, and σ be a substitution in R-normal form. If x is not eliminated from t by $\overline{\overline{\sigma}}^F$ and $x\sigma.head \in F'$, then $x\sigma$ is a subterm of $(t\sigma)\downarrow$.*

proof :
$(t\sigma)\downarrow = (t\overline{\overline{\sigma}}^F \rho_u)\downarrow = (t\overline{\overline{\sigma}}^F)\downarrow \rho_u =_E t\overline{\overline{\sigma}}^F \rho_u$. Since ρ_u is normalized and $t\overline{\overline{\sigma}}^F$ is pure in E, all R_E rewrite steps from $t\overline{\overline{\sigma}}^F \rho_u$ to $(t\overline{\overline{\sigma}}^F)\downarrow \rho_u$ are performed in the homogeneous part $t\overline{\overline{\sigma}}^F$.
Since $x\sigma.head \in F'$ and $\overline{\overline{\sigma}}^F$ does not eliminate x from t, $h(x\sigma) \in V((t\overline{\overline{\sigma}}^F)\downarrow)$, hence $x\sigma = h(x\sigma)\rho_u$ is a subterm of $(t\sigma)\downarrow$. □

Lemma 8 *Let $t \in T(F', X)$, and let σ be a substitution in R-normal form. If $x \in V(t)$ and $x\sigma.head \in F$, then $x\sigma$ is a subterm of $(t\sigma)\downarrow$.*

proof : Similar to previous proof. □

Lemma 9 *Let C be the cycle*
$C = \{x_1 =_{E'} t_1, x_2 =_E t_2, \ldots, x_{2n} =_E t_{2n}\}$ *where*
$\forall i \in [1..2n]$, $x_{i+1 (mod\, 2n)} \in V(t_i)$ *and* $\forall i \in [1..n]$,
$t_{2i} \in T(F, X) \setminus X$ *and* $t_{2i-1} \in T(F', X) \setminus X$.
If σ is a solution of C, then there exists $i \in [1..n]$ such that $\overline{\overline{\sigma}}^F$ eliminates $x_{2i+1 (mod\, 2n)}$ from t_{2i}.

proof : We can assume without loss of generality that $x_{2i-1(mod\,2n)}\sigma.head \in F'$ for $i \in [1..n]$ because E' is collapse free. Then, for $i \in [1..n]$, $x_{2i-1}\overline{\overline{\sigma}}^F = h(x_{2i-1}\sigma) \in X$.
If σ is a solution of C, then σ must also be a solution of $C' = \{x_1 =_{E'} t_1, x_2 =_E t_2\overline{\overline{\sigma}}^F, x_3\overline{\overline{\sigma}}^F =_{E'} t_3, \ldots, x_{2n} =_E t_{2n}\overline{\overline{\sigma}}^F\}$ since $\sigma = \overline{\overline{\sigma}}^F\rho_u$ and $\overline{\overline{\sigma}}^F$ is idempotent. Assume $\overline{\overline{\sigma}}^F$ eliminates no $x_{2i+1(mod\,2n)}$ from t_{2i}. We show that this yields a contradiction:
Assume first that $t_{2i}\overline{\overline{\sigma}}^F =_E h(x_{2i+1}\sigma)\ \forall i \in [1..n]$, then C' is equivalent to $\{x_1 = t_1, x_2 = h(t_3\sigma), x_3\overline{\overline{\sigma}}^F = t_3, \ldots, x_{2n-1}\overline{\overline{\sigma}}^F = t_{2n-1}, x_{2n} = h(x_1\sigma)\}$ Therefore σ is a solution of

$$t_1 = t_1\{x_2 \mapsto t_3\{x_4 \mapsto t_5 \cdots \{x_{2n} \mapsto t_1\} \cdots\}\}$$

which is a pure equation in E' between a term and one of its proper subterms, because $x_{i+1} \in V(t_i)$, a contradiction.
Assume now without loss of generality that $t_{2n}\overline{\overline{\sigma}}^F$ is not E-equal to a variable. We show that $(x_1\sigma)\downarrow$ is a proper subterm of $(t_{2n-m}\sigma)\downarrow$ for $m \in [1..2n-1]$, by induction on m.
Basis: If $m = 0$, by lemma 7, $(x_1\sigma)\downarrow$ is a subterm of $(t_{2n}\sigma)\downarrow$. It is a proper subterm since $(x_1\sigma)\downarrow.head \in F'$ and $(t_{2n}\sigma)\downarrow.head \in F$.
Inductive step: Assume $(x_1\sigma)\downarrow$ is a proper subterm of $(t_{2n-m}\sigma)\downarrow$ for $m \in [0..2n-2]$. By lemma 7 (if m is odd) or 8 (if m is even), $(t_{2n-m}\sigma)\downarrow$ is a subterm of $(t_{2n-(m+1)}\sigma)\downarrow$, hence $(x_1\sigma)\downarrow$ is a proper subterm of $(t_{2n-m}\sigma)\downarrow$.
We have shown that $(x_1\sigma)\downarrow$ is a proper subterm of $(t_1\sigma)\downarrow$, i.e., of itself which is a contradiction. □

COROLLARY **2** *The following transformation preserves the set of solutions :*
Elim1

$P_E \cup P_{E'} \quad \vdash \quad P_E\sigma \cup P_{E'} \cup \{x_j = s_j\}_{j\in J}$
If $P = P_E \cup P_{E'}$ *contains the cycle*
$C = \{x_1 =_{E'} t_1, x_2 =_E t_2, \ldots, x_{2n} =_E t_{2n}\}$ *where*
$\forall i \in [1..2n]$, $x_{i+1(mod\,2n)} \in V(t_i)$ *and* $\forall i \in [1..n]$,
$t_{2i} \in T(F,X) \setminus X$ *and* $t_{2i-1} \in T(F',X) \setminus X$,
and $\sigma = \{x_j = s_j\}_{j\in J}$ *is a most general E-eliminator of* $x_{2i+1(mod 2n)}$ *from* t_{2i} *for some* $i \in [1..n]$.

Unfortunately, $S \cup \{$**Elim1**$\}$ does not terminate:

Let $E = B$ and let E' be the free theory. Consider the unification problem : $P_1 = \{x = f(x_1, y), x_1 = x + y\}$. A most general eliminator of x in $x + y$ is $\{y \mapsto x + y'\}$, and applying **Elim1**, we get the equivalent system $P_2 = \{x_1 = y', x = f(x_1, y), y = x + y'\}$ to which **Var-Rep** applies, yielding $P_3 = \{x_1 = y', x = f(x_1, y), y = x + x_1\}$. Applying now **Elim1** to the cycle involving y yields a problem containing P_1.

Solving this problem needs to memorize which edges have been removed from the occur-check graph G associated with the occur-check relation $<$. For this purpose, we adapt the previous notion of a unification problem: in the sequel, a unification problem will be a pair $< P, SC >$ where P is a unification problem as defined until now and SC is a *set of constraints*, i.e., a set $\{(x_k, y_k)\}_{k\in K}$ of oriented pairs of variables representing the edges of G that have been removed when applying **Elim1**.

Lemma 10 *The application of* **Elim1** *can be restricted to the case where it does not introduce a previously removed edge in G.*

proof : Let P be a unification problem containing the cycle
$C = \{x_1 =_{E'} t_1, x_2 =_E t_2, \ldots, x_{2n} =_E t_{2n}\}$ where
$\forall i \in [1..2n],\ x_{i+1(mod\,2n)} \in V(t_i)$ and $\forall i \in [1..n]$,
$t_{2i} \in T(F,X) \setminus X$ and $t_{2i-1} \in T(F',X) \setminus X$.
By lemma 9, all the solutions of P are of the form $\sigma'\sigma''$ where σ' E-eliminates some $x_{2i+1(mod\,2n)}$ from t_{2i}. Assume **Elim1** removes $(x_{2i}, x_{2i+1(mod\,2n)})$ from G and that further applications of **Elim1** yield a cycle free problem for which $(x_{2i}, x_{2i+1(mod\,2n)}) \in G$. Then by lemma 9, there exists some $j \neq i$, $j \in [1..n]$ such that $(x_{2i}, x_{2j+1(mod\,2n)}) \notin G$. Such a solution is subsumed by the solution obtained by chosing nondeterministically the elimination of $x_{2j+1(mod\,2n)}$ from t_{2j}. □

In other words, we compute the solutions that remove an edge $(x_{2i}, x_{2i+1(mod\,2n)})$ forever.

4.2. CONSTANT ELIMINATION

Again, we can omit substitutions that E-instanciate E'-instanciated variables because E' is cycle free. By treating them as constants, we may loose solutions σ such that $x\sigma = z$, $y\sigma = z$ where $z \in X$ and x, y are two E'-instanciated variables. Before to eliminate x_{2i+1} from t_{2i}, and break a cycle down, a substitution will be applied nondeterministically to the variables to be eliminated in order to take into account all such possible substitutions. E'-instanciated are then treated as constants.

DEFINITION **16** *Given a set Y of variables, a* variable identification *over Y is a substitution $\theta = \{x_i \mapsto y_i\}_{i \in I}$ where $x_i, y_i \in Y \forall i \in I$.*

Let us adapt E-elimination to constants :

DEFINITION **17** *A substitution σ is an E-eliminator of a constant $c \in C$ from a term $t \in T(F \cup C, X)$ if there exists a term $u =_E t\sigma$ such that c does not occur in u.*

DEFINITION **18** *Given a set $\{(c_1,t_1), \ldots, (c_n,t_n)\}$ where $c_i \in C$ and $t_i \in T(F \cup C, X)$ for $i \in [1..n]$, finding a set Σ such that*

1. *$\forall \sigma \in \Sigma$, $\forall i \in [1..n]$, σ is an E-eliminator of c_i in t_i*
2. *$\forall \sigma$ such that $\forall i \in [1..n]$ σ is an E-eliminator of c_i in t_i, $\exists \sigma' \in \Sigma$ $\sigma' \leq_E \sigma$*

is called the constant elimination problem *in* E.

Many theories of interest have such a complete constant elimination algorithm. This is the case of Boolean Rings, as well as Abelian Groups, as seen in section 5. This problem, however, is not straightforward in general.

When an edge (x_{2i}, x_{2i+1}) of the occur-check graph G is broken, the pair (x_{2i}, x_{2i+1}) is added to SC, and yields a new elimination problem:

DEFINITION 19 *Given a set of constraints* $SC = \{(x_k, y_k)\}_{k \in K}$, *the* constant elimination problem associated *to* SC *in* P *is* $\{(x_k, s) \mid y_k = s \in P_E, k \in K\}$.

EXAMPLE 5 *Let* $SC = \{(x_1, y_1), (x_2, y_2)\}$ *and* $P_E = \{y_1 = t_1, y_2 = t_2\} \subseteq P$. *the constant elimination problem associated with* SC *in* P *is* $SC = \{(x_1, t_1), (x_2, t_2)\}$.

A cycle of the form
$C = \{x_1 =_{E'} t_1, x_2 =_E t_2, \ldots, x_{2n} =_E t_{2n}\}$ where
$\forall i \in [1..2n]$, $x_{i+1(mod\,2n)} \in V(t_i)$ and $\forall i \in [1..n]$,
$t_{2i} \in T(F, X) \setminus X$ and $t_{2i-1} \in T(F', X) \setminus X$,
in a unification problem $< P, SC >$ is broken down by the following nondeterministic rule:

Elim

$P_E \cup P_{E'} \quad \vdash \quad P_E\theta\sigma \cup \{x_j = s_j\}_{j \in J} \cup P_{E'} \cup \{x_k = y_k\}_{k \in K}$
$SC \quad \vdash \quad SC \cup \{(x_{2i+1(mod2n)}, x_{2i})\}$
if P is separated,
$\theta = \{x_k = y_k\}_{k \in K}$ is a variable identification over the set of E'-instanciated variables in the constant elimination problem CEP associated with $SC \cup \{(x_{2i+1(mod2n)}, x_{2i})\}$ in P,
$\sigma = \{x_j = s_j\}_{j \in J}$ is a solution of $CEP\theta$, where E'-instanciated variables are treated as constants.

Lemma 11 Elim *preserves the sets of solutions.*

proof : By lemma 9, some $x_{2i+1(mod2n)}$ must be eliminated from t_{2i}, and by lemma 10, we need not consider E-eliminators that reintroduce in the occur-check graph some previously removed edge. By treating E'-instanciated variables as constants, we loose the solutions that make equal two different E'-instanciated variables. But such a solution is an instance of the corresponding variable identification. Finally, applying $\theta\sigma$ to P_E while adding $\{x_j = s_j\}_{j \in J}$ and $\{x_k = y_k\}_{k \in K}$ to P is necessary to have P'_E in solved form. □

4.3. TERMINATION

It is possible that applying **Elim** yields a system $< P, SC >$ in which P is not separated anymore. Then the simplification rules of S must be applied again to P. The following lemma shows that this cannot happen infinitely many times:

Lemma 12 *After* **Elim** *is applied,* P_E *is still solved.*
If applying **Var-Rep**, *then* **E'-Res** *after* **Elim** *increases* $PM(P)$, *then* **Var-Rep** *decreases strictly* $SV(P)$.

proof : Similar to proof of lemma 6. □

PROPOSITION **3** $S \cup \{$**Elim**$\}$ *terminates.*

proof : **Elim** cannot apply infinitely many times in a row because it decreases the number of possible edges between shared variables in the graph G. By previous lemma, if an application of **Elim** is followed by applications of the rules of S, then $SV(P)$ decreases. □

A last step can finally be performed to construct the solution of P by using replacement:

Rep

$\{x = s\} \cup P \quad \vdash \quad \{x = s\} \cup P\{x \mapsto s\}$
if x occurs in P and (s is not a variable or s occurs in P), and P is in normal form for $S \cup \{$ **Elim**$\}$.

We can now state our main result:

THEOREM **2** $S \cup \{$ **Elim**$\} \cup \{$**Rep**$\}$ *computes a* CSU_E *for terms in* $T(F \cup F', X)$.

5. Application to Boolean Rings and Abelian Groups

Results in this section are also in (Schmidt-Schauß 88). Based on a theorem by Löwenheim, Martin and Nipkow (Martin & Nipkow 86) give a complete unification algorithm for boolean terms with free constants.

Another solution for boolean unification is given in (Büttner & Simonis 86) which directly computes a most general solution, and has the relevant advantage that it does not E-instanciate all the variables in $V(s) \cup V(t)$. It is possible to chose which variables are to be E-instanciated.This allows to decrease the number of applications of the rules **Mem-Init** and **Mem-Rec** by avoiding many merges. The algorithm for solving the equation $t = 0$ is the following:

```
proc solve ($t$: term) returns(subst)

    if $t = 0$
    then return {} else
        if $t = (x * t_1) + t_2$
        then let $\sigma = solve((t_1 + 1) * t_2)$
```

```
                if $\sigma$ = fail
                then return (fail)
                else return $(\sigma \circ \{x \mapsto ((t_1 + 1) * u) + t_2\})$
                                    (*where $u$ is a fresh variable*)

                endif
            else return (fail)
            endif
        endif
    endproc
```

The constant elimination problem is solvable in B: Let $C = \{a_1, \ldots, a_k\}$ be a set of free constants. Let $EP = \{(c_1, t_1), \ldots, (c_n, t_n)\}$ be a constant elimination problem such that $c_i \in C\ \forall i \in [1..n]$. Let $\{x_1, \ldots, x_m\}$ be the set of variables occurring in $t_1, \ldots, t_n$, and let $\{p_1, \ldots, p_{2^k}\}$ be the set of pairwise distinct products of constants in C modulo associativity, commutativity and idempotence of $*$ (the empty product is denoted by 1). Since $T(\{+, *, 0, 1\}, X)/_{=_B}$ is a vector space over $\{0, 1\}$ with basis $\{p_1, \ldots, p_{2^k}\}$,a most general substitution σ in $T(F \cup C, X)$ can be written:

$$\{x_i \mapsto x_{i,1}p_1 + \cdots + x_{i,2^k}p_{2^k}\}_{i \in [1..m]}$$

where the $x_{i,j}$s are fresh variables. Let us apply σ to $t_1, \ldots, t_n$, and express $t_i\sigma$ with respect to $\{p_1, \ldots, p_{2^k}\}$:

$$t_i\sigma =_B \Sigma_{j=1}^{2^k} s_{i,j}p_j = t_i'$$

where $s_{i,j}$ is a term in which no constants of C occurs. Now $a_h \in C$ does not occur in t_i' iff $s_{i,j} = 0$ for all j such that a_h occurs in p_j. Hence, the solutions of EP are the solutions of the unification problem in B :

$$\{s_{i,j} = 0 \mid (a_k, t_i) \in EP, a_k \in p_j\}$$

The case of abelian groups is quite similar. (Lankford *et al.* 84) describes a complete unification algorithm. This algorithm accepts uninterpreted constants, solving the constant matching problem. Finally, variable elimination is performed in a similar way by applying a most general substitution and making equal to 0 the coefficients of the constants that must be eliminated.

6. Conclusion

Using a very general technique, we have solved two well-known open problems: unification in boolean rings and in abelian groups with free function symbols. The former case has been implemented in LeLisp on a SUN under UNIX4.2.

This work leaves open the problem of combining two arbitrary theories E and E' with E and E' accepting non-regular and collapse axioms. This problem is solved in (Schmidt-Schauß 88), and the solution involves more nondeterminism. Note finally that our termination proof is much simpler than the corresponding proof in (Fages 84), (Yelick 85) and (Kirchner 85). This is mainly due to the absence of an explicit replacement rule, except at the very end. The measure is not drastically different : shared variables are essentially Yelick's *multivars*, but delaying replacement until the end makes much easier to prove that the weight decreases. Variable replacement is sufficient to handle the interactions between the different subproblems.

Of course, replacement may be needed inside **E-Res** or **E'-Res**. This question will be discussed in a forthcoming paper in AC-unification.

Another advantage of using merge instead of replacement is that our method extends automatically to unification of infinite trees by using appropriate unification algorithms in **E-Res** and **E'-Res** for infinite trees and removing the rules **Elim** and **Rep**.

Appendix: Implementation

The algorithm has been implemented in LeLisp 15.2 on a SUN workstation. E and E'-resolution, E-elimination, as well as the function that normalizes terms in $T(F,X)$ are parameters in the program.

The boolean unification algorithm described in (Büttner & Simonis 86) allows a very relevant optimization because it does not compute a MGU that instanciates all the variables of the unified terms. Hence it is possible to avoid many merges that would appear using (Martin & Nipkow 86). Of course this improvement requires a slight modification of the boolean unification algorithm in order to use E'-instanciated variables.

The cycles are all broken at the same time after computing the minimal sets of edges to be removed in the occur-check graph and the variable identifications that are compatible with E'.

The implementation allows to collect all solutions in $MGU_{E \cup E'}(s,t)$, but we present some examples where the user is asked to chose a branch at nondeterministic steps.

An additional rule **Simplif** is added that removes "useless" equations of the form $v = t$ where v is a variable that was not in the original problem and v occurs nowhere else in P and t is not a variable of the original problem, or t occurs somewhere else in P. The rule is applied after every application of **Var-Rep**, in order to remove equations $x = x'$ where x' is a new variable that occurs nowhere else.

The original rule E-match is replaced by the following

$\{s = t\} \quad \vdash \quad \{x = s, x = t\}$
if $s \in T(F,X) \setminus X$ and $t \in T(F',X) \setminus X$, for a fresh variable x.

hence all theory clashes are taken care of by the rules **Mem-Init** and **Mem-Rec**.

The following examples show how the rules transforms a unification problem. When a separated system is obtained which admits no cycle, then the program prints it and stops. The system is not really solved yet: to obtain a solution, it is still necessary to perform replacement.

The program first reads some equations, then it applies variable abstraction as long as possible :

```
TYPE YOUR EQUATIONS (. WHEN FINISHED)
? f(x)+f(y)=f(a)+f(b)
? .



------------------------------------------------------

*****VARIABLE ABSTRACTION

------------------------------------------------------

            *** PE ***
   v1 + v2 = v3 + v4
            *** PE'***
   v1 = f(x)
   v2 = f(y)
   v3 = f(a)
   v4 = f(b)
------------------------------------------------------
```

Now, after variable abstraction, the subproblem $P_{E'}$ in the free theory is in a solved form, while P_E is not. The rule **E-Res** is then applied. Note that the equations between variables are added to $P_{E'}$ rather than to P_E.

```
------------------------------------------------------
*****E-RESOLUTION

------------------------------------------------------
            *** PE ***
   v1 = v7 + v8 + v9
            *** PE'***
   v2 = v7
   v3 = v8
   v4 = v9
   v1 = f(x)
   v2 = f(y)
   v3 = f(a)
   v4 = f(b)
------------------------------------------------------
```

Var-Rep applies, and at the same time, the additional rule **Simplif**, and the obtained problem is:

```
------------------------------------------------------
*****VAR-REP

------------------------------------------------------
            *** PE ***
   v1 = v2 + v3 + v4
```

```
            *** PE'***
   v1 = f(x)
   v2 = f(y)
   v3 = f(a)
   v4 = f(b)
------------------------------------------------------
```

Now the variable V_1 has a value in both the free theory and in the boolean theory. To solve the merge, **Mem-Init** puts the equation $v_1 = v_2 + v_3 + v_4$ onto the stack. The variable v_1 is marked, and the equation is removed from P_E.

```
------------------------------------------------------
*****MEM-INIT
------------------------------------------------------
            *** PE ***
            *** PE'***
   v1 = f(x)
   v2 = f(y)
   v3 = f(a)
   v4 = f(b)
            *** STACK ***
   v1 = v2 + v3 + v4
Marked Variables : (v1)
------------------------------------------------------
```

Mem-Rec applies now and computes a most general match in B from $v_2 + v_3 + v_4$ to v_1. The corresponding equations are added as well as the identification of v_1 and v_4 chosen by the user. The new equation $v_2 = v_{12} + v_{13} + v_1$ on whic a merge applies is pushed onto the stack, but it is simplified because we have $v_4 = v_{13}$ and $v_1 = v_4$, and it becomes $v_2 = v_{12}$.

```
------------------------------------------------------
*****MEM-REC

MATCHING v2 + v3 + v4 ONTO v1
A MGM IS
v2 ---> v12 + v13 + v1
v3 ---> v12
v4 ---> v13
v1 may be identified to a variable in (v1 v2 v3 v4)
chose v4
------------------------------------------------------
            *** PE ***
            *** PE'***
   v13 = v1
   v3 = v12
   v4 = v13
   v1 = f(x)
   v2 = f(y)
   v3 = f(a)
   v4 = f(b)
            *** STACK ***
   v2 = v12
Marked Variables : (v2 v1)
-----------------------------------------------------
```

The next application of **Mem-Rec** is trivial, and we obtain a pure problem in the free theory:

```
-----------------------------------------------------
*****MEM-REC

MATCHING v12 ONTO v2
A MGM IS
v12 ---> v2
-----------------------------------------------------
            *** PE ***
            *** PE'***
   v12 = v2
   v3 = v12
   v1 = f(x)
   v2 = f(y)
   v3 = f(a)
   v1 = f(b)
-----------------------------------------------------
```

Variable replacement and simplification apply:

```
-----------------------------------------------------
*****VAR-REP
-----------------------------------------------------
            *** PE ***
            *** PE'***
   v1 = f(x)
   v2 = f(y)
   v2 = f(a)
   v1 = f(b)
-----------------------------------------------------
```

The problem can now be solved in the free theory, then simplified, and the program stops.

```
-----------------------------------------------------
*****E'-RESOLUTION
-----------------------------------------------------
            *** PE ***
            *** PE'***
   v2 = f(a)
   y = a
   v1 = f(b)
   x = b
------------------------------------------------------
*****SIMPLIFICATION
------------------------------------------------------
            *** PE ***
            *** PE'***
   y = a
   x = b
------------------------------------------------------
******THE SYSTEM IS NOW SOLVED
------------------------------------------------------
            *** PE ***
            *** PE'***
   y = a
   x = b
------------------------------------------------------
```

The following example shows how **Elim** removes the cyclesand how termination is provided by memorizing the previously removed edges. The second elimination problem takes into account that x must not be reintroduced in the value of y and the program terminates with failure.
We start with a problem containing the cycle $x < y < x$ The rule **Elim** removes x from the value of y:

```
------------------------------------------------------
            *** PE ***
   y = x + z
            *** PE'***
   x = f(y,z)
------------------------------------------------------
ELIMINATING
x FROM x + z
IN
------------------------------------------------------
            *** PE ***
   y = x + z
            *** PE'***
   x = f(y,z)
------------------------------------------------------
AN ELIMINATOR IS
z ---> z5 + x
------------------------------------------------------
            *** PE ***
   z = z5 + x
            *** PE'***
   y = z5
   x = f(y,z)
```

Elim has introduced a new cycle $x < z < x$. It applies again trying to remove x from the value of z, but without reintroducing it in the value of y. This is impossible and the program stops with failure.

```
--------------------------------------------------------------
ELIMINATING
x FROM z5
x FROM z5 + x
IN
--------------------------------------------------------------
            *** PE ***
   z = z5 + x
            *** PE'***
   y = z5
   x = f(y,z)
--------------------------------------------------------------
FAILURE IN BOOLEAN UNIFICATION
```

For more detailed experimentation see **LRI internal report 429.** **Acknowledgement** This research was partially supported by GRECO Programmation and FIRTECH Orsay.

References

Baader, F. "Characterizations of Unification Type Zero "*In Proc. 3rd Int. Conf. on rewriting Techniques and Applications, pp. 2-14, Chapel Hill, April 1989*

Bachmair, L., Dershowitz, N. and Hsiang, J. "Orderings for Equational Proofs "*In Proc. First LICS. Boston (1986).*

Boudet, A. "Unification dans les Anneaux Booléens en présence de Symboles Libres "*Rapport de stage de D.E.A., Orsay, (sep 87).*

Bürckert, H.-J. "Matching, a special case of Unification? " *Technical Report, SR-87-08, (1987)*

Büttner, W., Simonis, H. "Embedding Boolean Expressions into Logic Programming "*Preprint, Siemens AG, München (1986)*

Dershowitz, N and Jouannaud J.-P. "Term Rewriting Systems "*Handbook of Theoretical Computer Science, North-Holland, to appear (1989).*

Fages, F. "Associative Commutative Unification "*Proc. 7th Int. Conf. on Automated Deduction. Springer-Verlag (1984). Also in JSC, vol. 3 pp. 257-275, (1987).*

Fages, F. and Huet, G. "Unification and Matching in Equational Theories, "*Proceedings 5th Conference on Automata, Algebra and Programming. L'Aquila Springer-Verlag (1983).*

Fay, M. "First Order Unification in Equational Theories "*Springer-Verlag, Lecture Notes in Computer Science, Volume 87 (1979)*

Gallier, J.H. and Snyder, W "A General Complete E-unification Procedure "*Proc. 2nd Int. Conf. on Rewriting Techniques and Applications. Springer-Verlag (1987).*

Herbrand J. "Recherches sur la theorie de la demonstration " *Thèse, Université de Paris, in "Ecrits logiques de Jacques Herbrand", PUF, Paris, 1968.*

Hsiang, J and Dershowitz, N "Rewrite Methods for Clausal and Nonclausal Theorem Proving "*Proc. First Int. Conf. on Rewriting Techniques and Applications. Springer-Verlag (1985).*

Hsiang, J. and Rusinowitch, M. "On Word Problems in Equational Theories "*13th ICALP, Karlsruhe (1987)*

Huet, G. and Oppen, D. "Equations and Rewrite Rules: A Survey "*Academic Press (1980)*

Hullot, J.M. "Canonical Forms and Unification "*Proceedings, 5th Conference on Automated Deduction, Springer-Verlag (1980).*

Jouannaud, J.-P., Kirchner, C., Kirchner, H. "Automata, Languages and Programming, Barcelona, 1983 "*Lecture Notes in Computer Science, number 154, Springer-Verlag (1983).*

Kirchner, C. "Méthodes et outils de conception systématique d'algorithmes d'unification dans les théories équationnelles " *Thèse de Doctorat d'Etat en Informatique. Université de Nancy I (1985).*

Kirchner, C. "Computing Unification Algorithms "*Proc. 1st Symposium on Logic in Computer Science, Cambridge, (1986).*

Kirchner, C. "From Unification in a Combination of Equational Theories to a new AC-unification algorithm "*Proc. CREAS, Austin, to appear.*

D. Lankford, G. Butler, B. Brady "Abelian Group Unification Algorithms for Elementary Terms "*General Electric Workshop on the Rewrite Rule Laboratory (april 1984).*

Martelli, A. and Montanari, U. "An Efficient Unification Algorithm "*ACM Transactions On Programming Languages And Systems, vol.4, n 2 (1982)*

Martin, U. and Nipkow, T. "Unification in Boolean Rings "*Proc. 8th Int. Conf. on Automated deduction. Springer-Verlag (1986).*

Plotkin, G. "Building-in Equational Theories "*Machine Intelligence, vol.7, pp73-90 (1972).*

Raulefs, P., Siekman, J., Szabo, P. and Unvericht, E. "A short survey on the state of the art in Matching and Unification Problems "*Sigsam Bulletin, vol.13 (1979).*

Robinson, J. A. "A Machine-oriented Logic Based on the Resolution Principle "*Journal of the Association for Computing Machinery, vol.12, pp 23-41 (1965)*

Schmidt-Schauß, M. "Unification in a Combination of Arbitrary Disjoint Equational Theories "*In Proc CADE, Argonne (1988).*

Shostak, R.E. "Deciding combinations of theories "*JACM 31 (1984).*

Siekmann G. "Unification and Matching Problems "*Memo CSM-4-78. University of Essex (1978).*

Stickel, M.E. "A complete unification Algorithm for Associative-Commutative functions "*4th Int. Joint Conference on Artificial Intelligence, Tbilisi, (1975). Also in Journal of the ACM, vol.28, pp423-434 (1981).*

Tidén, E. "Unification in combinations of collapse-free theories with disjoint sets of function symbols "*Proc. 8th Int. Conf. on Automated deduction. Springer-Verlag (1986).*

Tidén, E. "First Order Unification in Combinations of Equational Theories. "*Ph.D. Thesis, Stockholm (1987)*

Toyama, Y. "On the Church-Rosser Property for the Direct Sum of Term Rewriting Systems "*JACM vol.34 (1987)*

Yelick, K. "Combining unification algorithms for confined equational theories "*Proc. First Int. Conf. on Rewriting Techniques and Applications. Springer-Verlag (1985).*