Unification Modulo Presburger Arithmetic and Other Decidable Theories

Mauricio Ayala-Rincón* and Ivan E. Tavares Araújo**

Departamento de Matemática, Universidade de Brasília, 70910-900 Brasil
ayala@mat.unb.br, ivan@mat.unb.br,ivan.araujo@psy.ox.ac.uk

Abstract. We present a general unification algorithm modulo Presburger Arithmetic for a restricted class of modularly specified theories where function symbols of the target theory have non arithmetic codomain sorts. Additionally, we comment on conditions guaranteeing decidability of matching and unification problems modulo more general theories than the arithmetic ones, which appear when automated deduction is implemented by combining conditional rewriting techniques and decision algorithms for built-in predicates.

Keywords: *Equational unification, automated reasoning, algebraic specification, conditional rewriting systems*

1 Introduction

Unification modulo general theories is very important in the context of automated reasoning and algebraic specification. In particular, unification modulo arithmetic theories, such as Presburger Arithmetic ($\mathcal{PA}$) [Pre29], is relevant since many deductive systems are specified so as to contain an arithmetic theory as parameter. The need for arithmetic unification is illustrated, for example, in [ARG97] where it is shown how Knuth-Bendix completion of conditional equational modular specifications containing an arithmetic parameter and how the cover-set method (applied in order to realize inductive proofs as it is made in [KS96]) can be improved by searching for solutions of the arithmetic constraints resulting when new critical pairs are deduced and induction schemes are generated, respectively.

As was noted by Dershowitz and Jouannaud in [DJ90], $\mathcal{PA}$ is an example of a theory with decidable unification problems. But, a semi-decision procedure like the one described in [Sho79], surprisingly cited in [DJ90], cannot be used for directly solving unification problems in $\mathcal{PA}$. We should rather provide a method allowing us to understand unification problems in arithmetic theories within more general contexts, such as the algebraic setting of monoidal theories. Procedures for solving unification problems with constants in monoidal theories

* Partly supported by FEMAT and CAPES Brazilian Foundations.

** Supported by CAPES Brazilian Foundation. Author currently at the Department of Experimental Psychology, University of Oxford, England

based on algebraic techniques, as given in [Nut92,BN96], could be adapted for solving "unification problems" appearing as purely equational expressions in the class of $\mathcal{PA}$ formulas. We show that *general* unification problems in $\mathcal{PA}$ with non-interpreted function symbols whose sorts are different from the arithmetic one would correspond to homogeneous systems of linear equations, while problems that additionally admit non-interpreted constant symbols belonging to the arithmetic sort would correspond to inhomogeneous systems over the ring $\mathbb{Z}$. This kind of presentation of general unification modulo arithmetic could lead to the discussion of interesting points in algebraic modular specification concerning the characterization of general unification in arithmetic theories. An initial version of the algorithmic method explained here was presented in [dAAR98].

Additionally, we present conditions guaranteeing decidability of matching and unification problems modulo more general theories than the arithmetic ones. These conditions were motivated when making effective simple rewrite properties, such as *decidability of one-step reduction* or *joinability of standard premises*, in the context of conditional rewriting systems with built-in predicates as premises over general decidable theories [AR93,AR00]. We consider modularly specified theories with a parameter theory T_0 whose codomain sorts of functions in the target theory do not belong to sorts of the parameter theory T_0. For these restricted theories we show that matching and general unification modulo T_0 is decidable whenever the universal-existential formulas of T_0 are decidable as well.

2 Unification in Monoidal Theories

An equational theory is monoidal if it contains both a binary operation with identity which is associative and commutative and an arbitrary number of unary operations which are homomorphisms with respect to the binary operation and its identity. The class of monoidal theories generalizes the class of commutative monoids. Nutt [Nut92] gives an algebraic characterization of unification problems in such theories. Solving unification problems in monoidal theories amounts to solve systems of linear equations over semirings. More precisely, for a given monoidal theory *elementary* unification problems can be viewed as homogeneous systems of linear equations and unification problems *with constants* as inhomogeneous ones over the canonical semiring determined by the monoidal theory. From this characterization one can obtain algebraic descriptions of the unification types of monoidal theories with and without constants.

It remains to solve the case of general unification. Since the solution of a unification problem depends on the structure of a semiring, it is not clear that one could give a characterization of the unification type of general problems by algebraic means. It is known ([BS94,BS99]) that a monoidal theory has general unification problems of type *finitary* if and only if problems with constants are of the same type. We can state some formal analogies between $\mathcal{PA}$ and the main features of monoidal theories that make possible adaptation of Nutt's method to solve elementary unification problems and problems with constants modulo $\mathcal{PA}$ (for brevity $\mathcal{PA}$-unification) by solving homogeneous and inhomogeneous systems of linear equations in the (semi)ring $\mathbb{Z}$, respectively. In fact, we could conceive addition as the required associative commutative binary operation with identity and multiplication by constants as unary operations which are homomorphisms for addition. Of course, the inductive part of $\mathcal{PA}$ cannot be presented equationally. That should be done by means of an inductive schema.

3 General $\mathcal{PA}$-unification Algorithm

We present a general unification algorithm for a restricted class of modularly specified theories in which the language of the arithmetic parameter theory is considered as a signature over the arithmetic sort, say *int*, and the remaining function symbols of the target theory range over extended sorts except constant symbols which can range over the arithmetic sort too. Additionally, only $\mathcal{PA}$-consistent axioms (about constants of arithmetic sort) are admitted. In this way one guarantees that the whole specification is a conservative extension of the specification of the arithmetic parameter. This restriction seems very strong, but it allows reasonable manipulation of many important examples because it occurs often when implementing formal specifications, where new sorts are constructed from the concrete ones [AR00].

Let $\mathcal{F}$ be a set of funtion symbols over a many-sorted signature Σ. Let S be the set of sorts of Σ and let $\mathcal{V}$ be a countably infinite many-sorted family of sets of variables.

We assume that the reader is familiar with the term algebra $\mathcal{T}(\mathcal{F},\mathcal{V})$. Let $t \in \mathcal{T}(\mathcal{F},\mathcal{V})$. We denote the sort of t by $sort(t)$. A substitution is a mapping from variables to terms of the corresponding sort which almost everywhere equal to the identity. We use the usual explicit notation for substitutions: $\{x_1/t_1,\ldots,x_n/t_n\}$. Substitutions are represented by Greek letters (σ,θ,λ) as usual. The application of a substitution σ to a term $t \in \mathcal{T}(\mathcal{F},\mathcal{V})$, denoted $t\sigma$, is defined by

$$t\sigma := \begin{cases} x\sigma, \text{ if } t = x \in \mathcal{V} \\ f(t_1\sigma,\ldots,t_n\sigma), \text{ if } t = f(t_1,\ldots,t_n) \end{cases}$$

Definition 31 *Let $t \in \mathcal{T}(\mathcal{F},\mathcal{V})$. The set of positions of the term t, denoted by $O(t)$, is the set of strings over the alphabet of positive integers, inductively defined as follows:*

- *if $t = x \in \mathcal{V}$, then $O(t) = \{\varepsilon\}$, where ε denotes the empty string;*
- *if $t = f(t_1,\ldots,t_n)$, then $O(t) = \{\varepsilon\} \cup_{i=1}^{n} \{i \cdot \pi \mid \pi \in O(t_i)\}$.*

The position ε is called the *root* position of the term t, and the function or variable symbol at this position is called the root symbol of t. Positions are denoted by Greek letters π,γ. Positions $\pi,\gamma \in O(t)$ are compared by the prefix order defined by $\pi \leq \gamma$ iff there exists π' such that $\pi\pi' = \gamma$. Additionally, if $\pi' \neq \varepsilon$ then one can write $\pi < \gamma$. The prefix order over positions is a partial order. Incomparable positions are called *parallel* or *disjunct* positions.

By $t|_\pi$ we denote the subterm at position $\pi \in O(t)$ of t.

By $s[\pi \leftarrow t]$ we denote the term resulting form s by replacing the subterm at position π in s, $s|_\pi$, with t. Of course, we suppose that $sort(t) = sort(s|_\pi)$.

Definition 32 *A term algebra $\mathcal{T}(\mathcal{F},\mathcal{V})$ with arithmetic reduct is one whose set of sorts, S, contains an arithmetic sort $S_0 = \{ int\}$ and whose signature Σ contains arithmetic function symbols for addition, $+ : int \times int \to int$, and successor, $succ : int \to int$, constant symbol for zero, $0 : \to int$, and an ordering predicate symbol, say $\leq : int \times int$.*

Terms $t \in \mathcal{T}(\mathcal{F},\mathcal{V})$ such that $sort(t) = int$ are called *arithmetic terms*. Terms with $sort(t) \neq int$ are called *extended terms*.

Since our proposal is to treat the arithmetic reduct of general term algebras by means of built-in decision algorithms, we adopt standard decimal notation for integer numbers: $0, 1, -1, 2, -2, 3, -3, \ldots$. Observe that negative integers cannot be constructed from zero and successor directly, but they can be deduced from arithmetic consistent equations such as $succ(succ(x)) = 0$. Additionally, repeated addition of an arithmetic term will be abbreviated with the multiplication of the corresponding integer constant and the term; for instance, $x + x + x$ is briefly written as $3x$.

In a term algebra with arithmetic reduct, arithmetic symbols and predicates are interpreted as in standard arithmetic models, such as the structure of integers with addition and an ordering predicate $\langle \mathbb{Z}, +, \leq \rangle$.

Our term algebras with arithmetic reduct are restricted in such a way that no other function symbol than the arithmetic ones can range over the sort int. This means that for all function symbols $f \in \mathcal{F}$, $f \neq +$ and $f \neq succ$ and $f \neq 0$, $sort(f) \neq int$.

We select the standard model of the Presburger arithmetic $\mathcal{PA}$ and denote the equality modulo $\mathcal{PA}$ by $=_{\mathcal{PA}}$. For example, $f(x+a, g(z)) =_{\mathcal{PA}} f(a+x, g(z))$, because of the commutativity of the addition in the structure of integers.

Definition 33 *Let $\mathcal{T}(\mathcal{F},\mathcal{V})$ be a term algebra with arithmetic reduct. A $\mathcal{PA}$-unification problem over $\mathcal{T}(\mathcal{F},\mathcal{V})$ is a finite set of equations*

$$\{s_1 =^?_{\mathcal{PA}} t_1, \ldots, s_n =^?_{\mathcal{PA}} t_n\}$$

where each pair of terms s_i and t_i, for $i = 1, \ldots, n$, are terms in $\mathcal{T}(\mathcal{F},\mathcal{V})$ of the same sort.

Definition 34 *Let $P = \{s_1 =^?_{\mathcal{PA}} t_1, \ldots, s_n =^?_{\mathcal{PA}} t_n\}$ be a $\mathcal{PA}$-unification problem over the term algebra with arithmetic reduct $\mathcal{T}(\mathcal{F},\mathcal{V})$. A substitution σ is a unifier of this problem iff $s_i\sigma =_{\mathcal{PA}} t_i\sigma$ for all $i = 1, \ldots, n$.*

Example 31 Consider the unification problem $\{f(x + x, g(u)) =^?_{\mathcal{PA}} f(z + z + z, v)\}$ over $\mathcal{T}(\mathcal{F},\mathcal{V})$, where x, z are variables of sort int and u, v are variables of other sort than int in $\mathcal{V}$. A unifier for this problem is the substitution $\theta = \{x/6, z/4, v/g(u)\}$. •

Definition 35 *Let $\mathcal{T}(\mathcal{F},\mathcal{V})$ be a term algebra with arithmetic reduct and let σ, θ be substitutions on $\mathcal{V}$. The substitution σ is more general modulo $\mathcal{PA}$ than the substitution θ iff there exists a substitution λ such that $x\theta =_{\mathcal{PA}} x\sigma\lambda$ for all $x \in \mathcal{V}$. In this case we write $\sigma \leq^{\mathcal{V}}_{\mathcal{PA}} \theta$.*

Example 32 (Continuing example 31). Observe that the unifier $\sigma = \{x/y + y + y, z/y + y, v/g(u)\}$ is more general than θ. In fact, $\theta = \sigma\{y/2\}$. $\bullet$

A complete set of unifiers for a $\mathcal{PA}$-unification problem over $\mathcal{T}(\mathcal{F}, \mathcal{V})$ consists of a set of unifiers Θ such that for all unifier, σ, of the problem there exists $\theta \in \Theta$ such that $\theta \leq^{\mathcal{V}}_{\mathcal{PA}} \sigma$.

Unlike the case of equational unification, when comparing unifiers of a $\mathcal{PA}$-unification problem P the notion of (minimal) complete set of $\mathcal{PA}$-unifiers is unnecessary. What should be used in the arithmetic case is the space of solutions.

Example 33 The space of solutions of the $\mathcal{PA}$-unification problems:
$\{f(2x, 2z + x) =^?_{\mathcal{PA}} f(3z, z + y)\}$,
$\{f(g(2x), 2x) =^?_{\mathcal{PA}} f(g(3z), 3z + 2)\}$ and
$\{f(g(2x), h(3x)) =^?_{\mathcal{PA}} f(g(3z), h(2z))\}$
is given by $\{x = 3u, y = 5u, z = 2u \mid u \in \mathbb{Z}\}$, $\emptyset$ and $\{x = 0, z = 0\}$, respectively. This can be easily checked by resolving the corresponding linear systems. $\bullet$

Nutt's method for unification problems with constants can be adapted directly for the (semi)ring $\mathbb{Z}$. Then one can treat $\mathcal{PA}$-unification problems with arithmetic constants and variables.

Example 34 Consider the $\mathcal{PA}$-unification problem:
$\{f(h(x+2c), 2z+x) =^?_{\mathcal{PA}} f(h(3z+c), z+y+c)\}$, where x, y, z and c are arithmetic varibles and constant, respectively. The space of solutions of the corresponding inhomogeneous linear system of equations: $x + 2c = 3z + c$ and $2z + x = z + y + c$ in $\mathbb{Z}$ corresponds to $\{(x = 3au + 3bc - c, y = 4au + 4bc - 2c, z = au + bc) \mid a, b, u \in \mathbb{Z}\}$. This can be easily computed by applying some algorithm for solving linear systems of equations in $\mathbb{Z}$ selecting the homogeneous and inhomogeneous part of the linear system. Observe that, unifiers can be written in terms of the new variable u by selecting integer values for a and b; for instance, if $a = b = 1$, $\sigma = \{x/3u + 2c, y/4u + 2c, z/u + c\}$. $\bullet$

Presenting the space of solutions of the corresponding integer linear system of a $\mathcal{PA}$-unification problem is the most appropriate way to exhibit explicitly a method to effectively computing all its (most general) unifiers.

Since an unifier of a $\mathcal{PA}$-unification problem P over $\mathcal{T}(\mathcal{F}, \mathcal{V})$ is (either an element or) a sub-space of the space of solutions of the associated integer linear system, we can conceive a unification algorithm for our restricted class of term algebras with arithmetic reduct as a combination of standard syntactic unification for the non-integer variables and resolution of integer linear systems of equations for the arithmetic ones.

The algorithm presented in table 1 basically realizes syntactic unification on the extended part of the $\mathcal{PA}$-unification problem and unifies the arithmetic part of the unification problem by using Nutt's unification method for monoidal theories. In our case, by solving homogeneous and inhomogeneous systems of linear equations in the (semi)ring $\mathbb{Z}$. Soundness and completeness of the algorithm constitute a constructive proof of the following

Lemma 31 *The complete set of unifiers of a $\mathcal{PA}$-unification problem over a term algebra with arithmetic reduct $\mathcal{T}(\mathcal{F}, \mathcal{V})$, restricted in such a way that only the arithmetic symbols range over the arithmetic sort int, can be effectively computed.*

Table 1. General $\mathcal{PA}$-unification Algorithm for Restricted Specifications

```
INPUT: s, t well-formed (extended) terms in T(F, V)
OUTPUT: σ unifier of s and t
BEGIN
LET Π be the maximal set
        {π_1, ..., π_n ∈ O(s) | sort(s|_{π_i}) = int and ∀π' < π_i, sort(s|_{π'}) ≠ int}
LET s' := s[π_1 ← x_1]...[π_n ← x_n], where x_i, i = 1, ..., n are
                        new variables and sort(x_i) = int
LET Υ be the maximal set
        {γ_1, ..., γ_m ∈ O(t) | sort(t|_{γ_j}) = int and ∀γ' < γ_j, sort(t|_{γ'}) ≠ int}
LET t' := t[γ_1 ← y_1]...[γ_m ← y_m], where y_j, j = 1, ..., m are
                        new variables and sort(y_j) = int
Apply any algorithm of syntactic unification to s' and t'
IF s' and t' do not unify THEN FAIL
ELSE LET σ be mgu of s' and t'
Decompose σ = σ' ∪ σ'', where
        σ' consist of the bindings for non integer variables and
        σ'' of the bindings for variables of sort int
LET σ'' = {x_{i_1}/y_{j_1}, ..., x_{i_k}/y_{j_k}}, where
        i_1, ..., i_k is a subsequence of 1, ..., n and 1 ≤ j_1, ..., j_k ≤ m
Apply Nutt's algorithm for unification with constants in order
        to resolve the (in)homogeneous linear system:
        {s|_{π_{i_1}} = t|_{γ_{j_1}}, ..., s|_{π_{i_k}} = t|_{γ_{j_k}}}
IF a space of solutions Ψ is found for {s|_{π_{i_1}} = t|_{γ_{j_1}}, ..., s|_{π_{i_k}} = t|_{γ_{j_k}}}
    THEN
        PRINT "solutions are compositions of σ' and ψ ∈ Ψ"
ELSE FAIL
END
```

The introduction of new variables of sort *int* eliminates the possibility of "occur-check" problems between sub-terms of sort *int* in the application of the syntactic unification algorithm. Direct application of syntactic unification for the input terms s, t does not work since it avoids correct arithmetic interpretations (for example, syntactic unification of $f(0+1)$ and $f(1+0)$ gives rise to the $\mathcal{PA}$-inconsistent difference set $\{0 = 1, 1 = 0\}$ which does not unify syntactically).

Observe also that the set of bindings for arithmetic variables could be written effectively in the form $\sigma'' = \{x_{i_1}/y_{j_1}, ..., x_{i_k}/y_{j_k}\}$ because of the use of different variables $x_1, \ldots, x_n, y_1, \ldots, y_m$ and the in-existence of non-variable sub-terms of sort *int* in s' and t'. Of course different arithmetic variables can replace identical arithmetic terms, but this is checked when resolving the (in)homogeneous linear system corresponding to σ''.

Our algorithm does not interpret extended function symbols as the direct application of Shostak's method does. For example, Shostak's semi-decision algorithm solves the equation $f(x) = x$ by replacing $f(x)$ with a new variable, say y, and by assigning x and y the same integer value. In this way, the non-interpreted function symbol f is erroneously interpreted as a function with a fixed point. Techniques developed by Baader and Shulz in [BS96] for combining decision procedures for equational theories could be more adequate in order to adapt Shostak's semi-decision algorithm to effectively calculate unifiers over arithmetic theories rather than just decide solubility of unification problems.

Soundness of the algorithm for the restricted class of specifications is obvious and its completeness results from the completeness of (syntactic unification and) Nutt's algorithm for unification with constants in monoidal theories. Polynomial run time complexity of the algorithm depends on the syntactic unification algorithm applied and on the method used to solve (in)homogeneous systems of linear equations in the ring $\mathbb{Z}$.

Example 35 Consider the specification of "arrays" of arbitrary "objects" indexed by "integer" parameters as in [AR00], where one defines two extended operations

$$\langle \cdot, \cdot, \cdot \rangle \ : \ \mathit{array} \times \ \mathit{int} \times \mathit{object} \to \mathit{array} \quad \text{and} \quad \cdot\,[\cdot] \ : \ \mathit{array} \times \ \mathit{int} \to \mathit{object}$$

with the intended meaning of *insertion* of objects at a position of an array and *selection* of the object at a position of an array, respectively.

Consider the problem of unification of

$$s \equiv \langle X, 2z_1 + z_2 + c_1, \mathcal{X} \rangle [z_1 + c_1]$$

and

$$t \equiv \langle \langle A, z_2 + c_1, \mathcal{L} \rangle, z_2 + z_3 + c_2, \mathcal{L} \rangle [z_2 + c_1 + c_2]$$

where X and $\mathcal{X}$ and z_1, z_2, z_3 are variables of sort *array*, *object* and *int*, respectively and A and $\mathcal{L}$ and c_1, c_2 are constants of sort *array*, *object* and *int*, respectively.

Following steps of the algorithm, initially one should find a syntactic unifier for

$$s' \equiv \langle X, x_1, \mathcal{X} \rangle [x_2] \ \text{ and } \ t' \equiv \langle \langle A, y_1, \mathcal{L} \rangle, y_2, \mathcal{L} \rangle [y_3]$$

which gives $\sigma = \{X/\langle A, y_1, \mathcal{L} \rangle, x_1/y_2, \mathcal{X}/\mathcal{L}, x_2/y_3\}$. Decomposing σ into the arithmetic and the non-arithmetic bindings one has

$$\sigma = (\sigma' = \{X/\langle A, y_1, \mathcal{L} \rangle, \mathcal{X}/\mathcal{L}\}) \cup (\sigma'' = \{x_1/y_2, x_2/y_3\})$$

Subsequently, one should resolve the corresponding unification problem with constants:

$$\{2z_1 + z_2 + c_1 = z_2 + z_3 + c_2, z_1 + c_1 = z_2 + c_1 + c_2\}$$

Following Nutt's method for unification problems with constants in monoidal theories, one should find solutions for the corresponding elementary unification problem:

$$\{2z_1 + z_2 = z_2 + z_3, z_1 = z_2\}$$

which corresponds to solutions of the homogeneous linear system:

$$\left[\begin{pmatrix} 2\,1\,0 \\ 1\,0\,0 \end{pmatrix} - \begin{pmatrix} 0\,1\,1 \\ 0\,1\,0 \end{pmatrix} = \begin{pmatrix} 2 & 0 & -1 \\ 1 & -1 & 0 \end{pmatrix}\right] \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

and combine it with solutions for the inhomogeneous linear systems generated by the constant part:

$$\begin{pmatrix} 2 & 0 & -1 \\ 1 & -1 & 0 \end{pmatrix} \begin{pmatrix} v_1 & w_1 \\ v_2 & w_2 \\ v_3 & w_3 \end{pmatrix} + \left[\begin{pmatrix} 1\,0 \\ 1\,0 \end{pmatrix} - \begin{pmatrix} 0\,1 \\ 1\,1 \end{pmatrix}\right] = \begin{pmatrix} 0\,0 \\ 0\,0 \end{pmatrix}$$

Space solution for the first system corresponds to $\{(u_1 = a, u_2 = a, u_3 = 2a) \mid a \in \mathbb{Z}\}$ and for the second and third to $\{(v_1 = b, v_2 = b, v_3 = 2b + 1) \mid b \in \mathbb{Z}\}$ and $\{(w_1 = c+1, w_2 = c, w_3 = 2c+1) \mid c \in \mathbb{Z}\}$, respectively. Consequently, the space of solutions corresponds to

$$\Psi = \left\{ \begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix} = \begin{pmatrix} a & b & c+1 \\ a & b & c \\ 2a & 2b+1 & 2c+1 \end{pmatrix} \begin{pmatrix} z \\ c_1 \\ c_2 \end{pmatrix} \middle| a, b, c \in \mathbb{Z} \right\}$$

Selecting, for example, $a = 5, b = -4$ and $c = 1$ one obtains the unifier $\psi = \{z_1/5z - 4c_1 + 2c_2, z_2/5z - 4c_1 + c_2, z_3/10z - 7c_1 + 3c_2\}$ that composed with σ' generates the unifier

$$\sigma = \{X/\langle A, 5z - 4c_1 + c_2 + c_1, \mathcal{L}\rangle,$$

$$\mathcal{X}/\mathcal{L}, z_1/5z - 4c_1 + 2c_2, z_2/5z - 4c_1 + c_2, z_3/10z - 7c_1 + 3c_2\}$$

for the original unification problem. •

At this point of our discussion we find very important to remark that frequently Shostak's procedure [Sho77,Sho79] has been erroneously cited as a complete method for deciding arithmetic with extended function and predicate symbols. Shostak's procedure is an incomplete semi-decision procedure. This is consequence of the use of Bledsoe's SUP-INF method for computing real intervals for all the integer variables occurring in arithmetic formulae. Of course, if a complete method, such as Cooper's one [Coo72], is used to compute intervals of solutions, then the resulting procedure will be complete, but impractical because of its inefficiency. Shostak himself presents a lot of informal comments on his work about this fact. See, for example, third paragraph before fourth section in [Sho77] (pp 534): "Fortunately this incompleteness manifests itself only rarely in practice. ..."; first paragraph of the fifth section in [Sho77] (pp 536): "... The success of the method derives at least in part from the fact that the real problem is easier to solve than the integer problem. The price to paid for this ease of solution is, of course, the resulting incompleteness"; last paragraph of the third section in [Sho79] (pp 353): "It should be noted that the completeness of the procedure depends on the completeness of the method used to test for integer feasibility. At present, there are no known complete integer programming methods that are also efficient. In prectice, however, this point is of little concern. ..."; etc. The first author has carefully examined, implemented and improved Shostak's procedure (using Bledsoe's method) showing that very simple arithmetic formulae, that occur in practice often, cannot be decided by Shostak's method [ARG97].

The natural question that arises is why to work with Presburger arithmetic if inequalities are not treated? In fact, what is interesting about Presburger arithmetic is the presence of the ordering predicate. Our $\mathcal{PA}$-unification algorithm can be modified to treat constrained $\mathcal{PA}$-unification problems, where the constraints are pure arithmetic predicates. These kind of unification problems occurs in practice. For example, when conditional rewriting methods are applied for deduction in equational conditional specifications with arithmetic premisses. We will remark on this class of unification problems in the next section.

4 Unification Modulo More General Theories

We present a decision algorithm for matching modulo theories more general than the arithmetic ones. By applying simple and obvious modifications our algorithm can be transformed into one for unification. Subsequently, one can conceive this algorithm for the case of the Presburger arithmetic as one for constrained $\mathcal{PA}$-unification. Our matching algorithm is presented in the particular context of a class of conditional equational theories with built-in predicates that were introduced and made effectively decidable by combining conditional rewriting techniques and decision algorithms in [AR93]. We decided to maintain this presentation of the algorithm because in this way we motivate its application in the particular setting of rewrite automated deduction and because it can be adapted to other automated deduction mechanisms in a straightforward manner. Initially, definitions of the particular class of modularly specified theories and their implementation by conditional rewriting techniques are given. Subsequently, the matching algorithm, as a mechanism for deciding existence of redices, is given.

A *basic theory* T_0 over an S_0-sorted signature, Σ_0, is a many-sorted first-order Henkin theory with equality. *Built-in predicates* are quantifier-free formulae of a basic theory. In a Henkin theory, for all formulae of the form $\exists x P(x)$ there is a ground term t in $T_{\Sigma_0}(\emptyset)$ such that $T_0 \models \exists x P(t) \; \underline{if} \; P(x)$.

Let $\mathcal{F}$ be a set of funtion symbols over a many-sorted signature $\Sigma \supset \Sigma_0$, where Σ_0 conforms the language of a basic theory. Let $S \supset S_0$ be the set of sorts of Σ and let $\mathcal{V}$ be a countably infinite many-sorted family of sets of variables over S. In order to incorporate built-in predicates as conditions into the structure of universal Horn clauses, defined over the signature Σ, built-in objects are described in the built-in language given by the S_0-sorted signature Σ_0.

As in the previous section, where extended terms cannot range over the (basic) arithmetic sort, here we restrict our signatures such that function symbols over $\Sigma \backslash \Sigma_0$ do not have codomain sort in S_0. A term t in the term algebra $\mathcal{T}(\mathcal{F}, \mathcal{V})$ with $sort(t) \in S_0$ is called a *basic term* and with $sort(t) \in S \setminus S_0$ an *extended term.*

Definition 41 *Let T_0 be a basic theory over the language of an S_0-sorted signature Σ_0 and let $\mathcal{F}$ be a set of funtion symbols over an S-sorted signature $\Sigma \supset \Sigma_0$. The corresponding term algebra $\mathcal{T}(\mathcal{F}, \mathcal{V})$, if restricted as above(: function symbols over $\Sigma \setminus \Sigma_0$ have codomain sort in $S \setminus S_0$), is called a* term algebra over T_0.

Let Σ, Σ_0 and T_0 denote signatures and basic theories satisfying the previous restriction. A universal Horn clause of the form:

$$l = r \; \underline{if} \; t_1 = s_1 \wedge \ldots \wedge s_k = t_k \wedge P$$

where, for $i = 1, \ldots, k$, t_i, s_i and l and r are S-sorted extended terms in $\mathcal{T}(\mathcal{F}, \mathcal{V})$ of the same sort and P is a built-in predicate, is called a *universal Horn clause with built-in predicate* P over the theory T_0. P is called *built-in condition*, $l = r$ the *conclusion* and $t_1 = s_1 \wedge \ldots \wedge t_k = s_k$ the *standard condition* of the clause. Attempting to made effective decision in theories specified by a set H of this class of universal Horn clauses one can transform all clauses into conditional rewrite rules of the corresponding form:

$$l \to r \; \underline{if} \; s_1 \downarrow t_1 \wedge \ldots \wedge s_k \downarrow t_k \wedge P$$

obtaining a *conditional rewriting system*, R_H, with built-in predicates and standard conditions as premises. We call this restricted kind of conditional rewriting systems *conditional rewriting systems over* $\mathcal{T}(\mathcal{F}, \mathcal{V})$ and T_0. Usual restriction on variables apply: only variables occurring in l can occur in the standard conditions and in the right-hand side of the conclusion, r. One can admit extra variables in the built-in condition, P. In standard conditional rewriting systems, applicability of rules is decided by recursively checking joinability of standard conditions.

Table 2. T_0-matching Algorithm for General Theories

INPUT: R a conditional rewriting system over $\mathcal{T}(\mathcal{F},\mathcal{V})$ and T_0 and
$u \in \mathcal{T}(\mathcal{F},\mathcal{V})$ an extended term

OUTPUT: π such that a left-side l of a rule in R matches $u|_\pi$
modulo T_0

COMMENTS: rename variables in rules of R using other variables
than the ones occurring in u

BEGIN

FOR all rules $\mathcal{R} \equiv l \rightarrow r \ \underline{if}\ s_1 \downarrow t_1 \wedge \ldots \wedge s_k \downarrow t_k \wedge P$ in R and
$\pi \in O(u)$ with $sort(u|_\pi) = sort(l)$ DO

BEGIN FOR

LET Π be the maximal set
$\{\pi_1, ..., \pi_n \in O(l) \mid sort(l|_{\pi_i}) \in S_0$ and $\forall \pi' < \pi_i,\ sort(l|_{\pi'}) \in S \setminus S_0\}$

LET $l' := l[\pi_1 \leftarrow x_1]...[\pi_n \leftarrow x_n]$, where x_i, $i = 1, \ldots, n$ are
new variables and $sort(x_i) = sort(l|_{\pi_i})$

LET Q be the conjunction of equalities $x_i = l|_{\pi_i}$, $i = 1, \ldots, n$

Compute the match θ from l' to $u|_\pi$
applying any syntactic matching algorithm

IF θ is defined THEN

BEGIN IF

Decompose $\theta = \theta' \cup \theta''$, where
θ' consist of the bindings for variables of sort in $S \setminus S_0$ and
θ'' of the bindings for variables of sort in S_0

LET $\theta'' = \{x_1/v_1, \ldots, x_n/v_n\}$

LET $match(l', u|_\pi)$ be the conjunction of equalities $x_i = v_i$, $i = 1, \ldots, n$

/* Observe that all v_js are maximal basic subterms of $u|_\pi$ */

LET $\boldsymbol{Y}$ be the union of all variables in $l|_{\pi_i}$ and variables x_i, $i = 1, \ldots, n$

LET $\boldsymbol{X}$ be the union of all variables in v_i, $i = 1, \ldots, n$

Decide whether $T_0 \models \exists \boldsymbol{Y} \forall \boldsymbol{X}((P \wedge Q \wedge \text{match}(l', u|_\pi)))$

IF $T_0 \models \exists \boldsymbol{Y} \forall \boldsymbol{X}((P \wedge Q \wedge \text{match}(l', u|_\pi)))$ THEN
PRINT "u could be reduced at position π applying rule $\mathcal{R}$" and STOP

END IF

END FOR

PRINT "u doesn't reduce"

END

Let R_H be a conditional rewriting system of the above class over a basic theory T_0 and u be an extended term. In order to decide one-step reduction of u, one should decide whether or not a position π, a rule $l \rightarrow r \ \underline{if}\ s_1 \downarrow t_1 \wedge \ldots \wedge s_k \downarrow t_k \wedge P \in R$ and a substitution σ exist such that $T_0^{=} \models u|_\pi = l\sigma$, where $T_0^{=}$ denotes the basic theory extended with non-interpreted symbols of the whole specification. Simultaneously, one can verify whether $T_0 \models P\sigma$ in the case of non-extra variable occurrences in P or T_0-consistence of $P\sigma$ (i.e., search for its solutions) in the case of extra variable occurrences in P. In the matching algorithm presented in table 2 we don't consider neither occurrence of extra variables in the built-in condition nor recursive verification of joinability of standard conditions. The matching algorithm answers the sole question of existence of potential redices for an extended term u. This answer partially the difficulty question of applicability of a conditional rewrite rule, that is a more specific problem to be considered in the context of conditional rewriting theory. Observe that this algorithm can be used for the case of the (basic) theory of the Presburger arithmetic.

The algorithm in table 2 doesn't compute effectively a matching substitution. Observe that the question $T_0 \models \exists \boldsymbol{Y} \forall \boldsymbol{X}((P \wedge Q \wedge match(l', u|_\pi)))$? is equivalent to the question: exists there σ such that $T_0^{=} \models P\sigma \wedge l\sigma = u|_\pi$? Consequently, if one could answer effectively the first question, that means to present specific solutions for all variables $\boldsymbol{Y}$, these can be propagated to the bindings of θ'' and θ' obtaining a matching substitution σ.

A universal-existential formula is one of the form

$$\forall x_1 \ldots \forall x_n \exists y_1 \ldots \exists y_m P$$

where P is quantifier-free. To guarantee decidability of the problem of search of redices at least universal-existential formulae of the basic theory should be decidable. In fact, relating to the notation of the algorithm again, $T_0 \models \exists \boldsymbol{Y} \forall \boldsymbol{X}((P \wedge Q \wedge match(l', u|_\pi)))$ holds exactly when its negation

$$\forall \boldsymbol{Y} \exists \boldsymbol{X} \neg((P \wedge Q \wedge match(l', u|_\pi)))$$

is not T_0-valid. The last formula is universal-existential. One can conclude that if the class of universal-existential formulae of the basic theory T_0 is decidable then our algorithm decide whether or not a left-side of a rule matches some subterm of an extended term u, validating simultaneously its built-in condition. Additionally, if decision of the universal-existential formulae of T_0 can be done effectively, that means giving specific solutions, then the matching substitution can be presented explicitly.

Definition 42 *Let $\mathcal{T}(\mathcal{F}, \mathcal{V})$ be a term algebra over a basic theory T_0. A constrained T_0-matching problem in $\mathcal{T}(\mathcal{F}, \mathcal{V})$ is the problem of deciding whether for two (extended) terms $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ and a basic predicate P over T_0 there exists a substitution σ such that $T_0^{=} \models P\sigma \wedge s\sigma = t$.*

In the general setting of T_0-matching a simple modification of the previous argumentation conform a proof of the following

Lemma 41 *Let $\mathcal{T}(\mathcal{F}, \mathcal{V})$ be a term algebra over a basic theory T_0. If the class of universal-existential formulae of T_0 is decidable then constrained T_0-matching problems are decidable too.*

Proof. Let $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ and P be a basic predicate over T_0. To decide if there exists σ such that $T_0^{=} \models P\sigma \wedge s\sigma = t$, apply a unique iteration of the main "FOR" loop of the algorithm for the terms $l := s$ and the root position of t ($\pi := \varepsilon$

and $u := t$). Following the notation of the algorithm, if the syntactic matching succeds the problem reduces to decide if $T_0 \models \exists \boldsymbol{Y} \forall \boldsymbol{X}((P \wedge Q \wedge \text{ match}(l', t|_\varepsilon)))$, that can be decided because of the assumption of decidability of the class of universal-existential formulae of T_0. $\square$

The algorithm can be straightforwardly modified by introducing new variables for the maximal basic subterms of $u|_\pi$, as it is done in the unification algorithm of the previous section, and by considering existential quantification for both sets of variables $\boldsymbol{X}$ and $\boldsymbol{Y}$ in order to obtain a unification algorithm modulo general basic theories T_0. This gives rise to a procedure that decides constrained T_0-unification when universal-existential formulae of T_0 are decidable too. This unification algorithm is essential when deciding simple rewrite properties such as joinability as well as when implementing more sophisticated decision techniques based on rewriting such as narrowing.

Definition 43 *Let $\mathcal{T}(\mathcal{F}, \mathcal{V})$ be a term algebra over a basic theory T_0. A constrained T_0-unification problem in $\mathcal{T}(\mathcal{F}, \mathcal{V})$ is the problem of deciding whether for two (extended) terms $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ and a basic predicate P over T_0 there exists a substitution σ such that $T_0^= \models P\sigma \wedge s\sigma = t\sigma$.*

Lemma 42 *Let $\mathcal{T}(\mathcal{F}, \mathcal{V})$ be a term algebra over a basic theory T_0. If the class of universal-existential formulae of T_0 is decidable then constrained T_0-unification problems are decidable too.*

Both lemmata apply for the Presburger arithmetic since the whole theory of $\mathcal{PA}$ is decidable. The corresponding constrained $\mathcal{PA}$-unification algorithm involves finally what is interesting about Presburger arithmetic: the treatment of inequalities. Of course, for this important theory one can always explicitly compute a substitution σ. In fact, Presburger's and Cooper's (and Shostak's) decision algorithms for $\mathcal{PA}$ (see [Pre29] and [Coo72] (and [Sho77]), respectively) search for explicit solutions of existentially quantified formulae (essentially by the method of elimination of quantifiers). For theories decided by model theoretical methods, such as $\mathcal{DNO}$ (totally and densely orders), no explicit solutions are exposed (see Rabin's chapter on decidable theories in [Bar77]).

In the context of the class of rewriting systems mentioned in this section effective computation of the matching substitution σ can also be guaranteed giving syntactical restrictions on the conditional rewrite rules. One can, for example, restrict left-hand sides of the conditions of the rules in order to contain only basic subterms which are either basic ground terms or basic variables. Under this restriction the search of redices is effectively computable and only decidability of the universal part of the basic theory is required.

5 Conclusion

We showed that direct application of Shostak's semi-decision algorithm for $\mathcal{PA}$ enlarged with non-interpreted function symbols is not appropriate to effectively solve $\mathcal{PA}$-unification problems. Nutt's unification algorithms for both unification with constants and elementary unification in monoidal theories result appropriate for resolving (and characterize) $\mathcal{PA}$-unification problems with and without constants, respectively. In addition, we presented a complete algorithm for solving general unification problems in theories specified modularly over arithmetic parameters which have the syntactic restriction that they admit only extended symbols of a sort different from the arithmetic one and new constant symbols that only can be of the arithmetic sort. Our algorithm reduces the problem of general unification to an initial application of syntactic unification and a subsequent resolution of an (in)homogeneous system of linear equations in the ring $\mathbb{Z}$. To make effective deduction in our restricted class of modularly specified theories, decidability of universal-existential formulas of T_0 is not enough. Of course, decidability of unification does not imply that one can effectively compute complete sets of unifiers. One needs effective decision algorithms which compute or at least characterize all solutions [BS94]. For parameter theories, such as $\mathcal{PA}$, with decision algorithms based on the quantifier elimination method, at least one solution can be calculated. It remains open the question of how to solve general unification modulo arithmetic theories without the syntactic restrictions we suppose here.

An interesting extension of our results to be developed is related with the theory of $\mathcal{PA}$ enlarged with rational numbers. Decidability of that theory was proved in [HI94] and improved decision algorithms have been showed adequate for the interpretation of programming logical languages as BQL and SAMPL [ITH]. These languages cover the linear subset of Igarashi's v acts, that allow for a nice mathematical representation of logical programs without the main restrictions of languages like PROLOG: grammatical limitation to clausal forms and use of non standard notions of negation. Since decision algorithms for that theory are based on the impracticable Cooper's method [Coo72], the first step to be done is to realize a practical (semi)decision method, perhaps based on the Shostak's one, that makes the interpreter of these languages useful in the practice.

References

[AR93] M. Ayala-Rincón. *Expressiveness of Conditional Equational Systems with Built-in Predicates.* PhD thesis, Universität Kaiserslautern, Kaiserslautern (Germany), December 1993.

[AR00] M. Ayala-Rincón. Church-Rosser Property for Conditional Rewriting Systems with Built-in Predicates as Premises. In D. M. Gabbay and M. de Rijke, editors, *Frontiers of Combining Systems 2*, Studies on Logic and Computation, 7, chapter 2, pages 17–38. Research Studies Press/Wiley, 2000.

[ARG97] M. Ayala-Rincón and L. M. R. Gadelha. Some Applications of (Semi-)Decision Algorithms for Presburger Arithmetic in Automated Deduction based on Rewriting Techniques. *La Revista de La Sociedad Chilena de Ciencia de la Computación*, 2(1):14–23, 1997.

[Bar77] J. Barwise, editor. *Handbook of Mathematical Logic*, volume 90 of *Studies in Logic and the foundations of Mathematics.* North-Holland, 1977.

[BN96] F. Baader and W. Nutt. Combination Problems for Commutative/Monoidal Theories or How Algebra Can Help in Equational Unification. *Journal of Applicable Algebra in Engineering, Communication and Computing*, 7(4):309–337, 1996.

[BS94] F. Baader and J. H. Siekmann. Unification Theory. In D. M. Gabbay, C. J. Hogger, and J. A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, pages 41–125. Oxford University Press, 1994.

[BS96] F. Baader and K. U. Schulz. Unification in the Union of Disjoint Equational Theories: Combining Decision Procedures. *Journal of Symbolic Computation*, 21:211–243, 1996.

[BS99] F. Baader and W. Snyder. Unification Theory. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*. Elsevier Science Publishers, 1999.

[Coo72] D. C. Cooper. Theorem Proving in Arithmetic without Multiplication. *Machine Intelligence*, 7:91–99, 1972.

[dAAR98] I. E. T. de Araújo and M. Ayala-Rincón. An Algorithm for General Unification Modulo Presburger Arithmetic. In *I Brazilian Workshop on Formal Methods, Porto Alegre, Brazil*, pages 146–151, October 1998.

[DJ90] N. Dershowitz and J.-P. Jouannaud. Rewrite Systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume 2, chapter 6, pages 244–320. Elsevier Science Publishers B. V. (North-Holland), 1990.

[HI94] C. Hosono and Y. Ikeda. A Formal Derivation of the Theory SA. *Theoretical Computer Science*, 127:1–23, 1994.

[ITH] Y. Ikeda, K. Tomita, and C. Hosono. A Programming Language SAMPL and its Interpreter. Unpublished document.

[KS96] D. Kapur and M. Subramaniam. New Uses of Linear Arithmetic in Automated Theorem Proving by Induction. *Journal of Automated Reasoning*, 16(1/2), 1996.

[Nut92] W. Nutt. Unification in Monoidal Theories is Solving Linear Equations over Semirings. Research Report RR-92-01, Deutsche Forschungszentrum für Künstliche Intelligenz, DFKI GmbH, Stuhlsatzenhausweg 3, D-66123 Saarbrücken, Germany, 1992.

[Pre29] M. Presburger. Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. In *1. Kongres matematyków krajow slowiańskich, Warsaw*, pages 92–101, 1929. In German.

[Sho77] R. E. Shostak. On the SUP-INF Method for Proving Presburger Formulas. *Journal of the Association for Computing Machinery*, 24(4):529–543, October 1977.

[Sho79] R. E. Shostak. A Practical Decision Procedure for Arithmetic with Function Symbols. *Journal of the Association for Computing Machinery*, 26(2):351–360, April 1979.