On the Word, Subsumption, and Complement Problem for Recurrent Term Schematizations*

(Extended Abstract)

Miki Hermann[1] and Gernot Salzer[2]

[1] LORIA (CNRS), BP 239, 54506 Vandœuvre-lès-Nancy, France. hermann@loria.fr
[2] Technische Universität Wien, Karlsplatz 13, 1040 Wien, Austria. salzer@logic.at

Abstract. We investigate the word and the subsumption problem for recurrent term schematizations, which are a special type of constraints based on iteration. By means of unification, we reduce these problems to a fragment of Presburger arithmetic. Our approach is applicable to all recurrent term schematizations having a finitary unification algorithm. Furthermore, we study a particular form of the complement problem. Given a finite set of terms, we ask whether its complement can be finitely represented by schematizations, using only the equality predicate without negation. The answer is negative as there are ground terms too complex to be represented by schematizations with limited resources.

1 Introduction

Infinite sets of first-order terms with structural similarities appear frequently in several branches of automated deduction, like logic programming, model building, term rewriting, equational unification, or clausal theorem proving. They are usually produced by saturation-based procedures, like equational completion or hyper-resolution. A usual requirement for effective use of such sets is the possibility to handle them by finite means. There exist several approaches to cope with this phenomenon, like lazy evaluation, set constraints, or term schematizations. Lazy evaluation usually does not combine well with unification or other operations. Set constraints allow to describe regular sets of first-order terms, using the potential of regular tree grammars and tree automata, and having the good properties of regular tree languages. Schematizations exploit the recurring term structure in infinite sets, as produced by self-resolving clauses or by self-overlapping rewrite rules.

Several formalisms for recurrent term schematizations were introduced within the last years. They rely on the same principle, namely the iteration of first-order contexts, but differ in the expressive power. The main concern in this work is the decidability of unification and the construction of finite complete

* Full version is at http://www.loria.fr/~hermann/publications/redelim.ps.gz. This work was done while the second author was visiting LORIA and was funded by Univeristé Henri Poincaré, Nancy 1.

sets of unifiers. Formalisms satisfying these requirements are ρ-terms [CH95], I-terms [Com95], R-terms [Sal92], and primal grammars [HG97], all of them with a finitary unification algorithm. Set operations were studied in [AHL97].

Applications of recurrent schematizations are quite rare and mostly theoretical, like in model building [Pel97] or cycle unification [Sal94]. One reason is that there are still some open problems to be solved prior to a successful implementation. A *sine qua non* of automated deduction is redundancy elimination. The elementary tools in this respect are testing for equality and subsumption. In other words, we need to solve the word problem and the subsumption problem for recurrent term schematizations. Moreover, only positive set operations were studied in [AHL97] without considering the complement. Complement building is interesting from the algebraic and logic point of view, e.g., during construction of counter-examples or for quantifier elimination.

In the first part of the paper, we investigate the word and the subsumption problem for primal grammars. By means of unification, we reduce them to a problem in Presburger arithmetic. Our approach is applicable to all recurrent term schematizations having a finitary unification algorithm. In the second part, we study a particular form of the complement problem. Given a finite set of terms, we ask whether its complement can be represented finitely by schematizations, using only the equality predicate without negation. The answer is negative as there are ground first-order terms too complex to be represented by primal grammars with limited resources.

2 Term schematizations

2.1 Syntax

The language of primal terms is based on four kinds of symbols: first-order variables $\mathcal{V}$, counter variables $\mathcal{C}$, function symbols $\mathcal{F}_p$ of arities $p \geq 0$, and defined symbols $\mathcal{D}_{q,p}$ of counter arities $q \geq 1$ and first-order arities $p \geq 0$. Nullary function symbols are called constants. The set of all function and defined symbols is denoted by $\mathcal{F}$ and $\mathcal{D}$, respectively.

Let $\mathbb{N}$ be the set of natural numbers. The set of *counter expressions* $\mathcal{L}$ is the set of linear expressions over $\mathcal{C}$ with coefficients in $\mathbb{N}$. Two counter expressions are considered equal if they are equivalent with respect to the usual equalities of addition and multiplication. Furthermore, we drop parentheses where possible and do not distinguish between natural numbers and their symbolic representation. The set of *primal terms* $\mathcal{P}$ is defined inductively as the smallest set satisfying the following conditions: $\mathcal{V} \subseteq \mathcal{P}$; $f(\mathbf{t}) \in \mathcal{P}$ if $f \in \mathcal{F}_p$ and $\mathbf{t} \in \mathcal{P}^p$; $\hat{f}(\mathbf{l};\mathbf{t}) \in \mathcal{P}$ if $\hat{f} \in \mathcal{D}_{q,p}$, $\mathbf{l} \in \mathcal{L}^q$, and $\mathbf{t} \in \mathcal{P}^p$. The sets of counter variables and first-order variables of a primal term t are denoted by $\mathcal{CV}ar(t)$ and $\mathcal{V}ar(t)$, respectively.

2.2 Semantics

In the sequel, we assume that the reader is familiar with the basic notions of term rewriting. With each defined symbol $\hat{f} \in \mathcal{D}_{q,p}$, we associate two rewrite

rules $\hat{f}(0, \mathbf{n}; \mathbf{x}) \to r_1^{\hat{f}}$ and $\hat{f}(m+1, \mathbf{n}; \mathbf{x}) \to r_2^{\hat{f}}[\hat{f}(m, \mathbf{n}+\boldsymbol{\delta}; \mathbf{x})]_A$, where $m, \mathbf{n}$ and $\mathbf{x}$ are counter variables and first-order variables, respectively; $r_1^{\hat{f}}$ and $r_2^{\hat{f}}$ are primal terms, whose variables are among those of the left hand sides of the rules; all defined symbols in $r_1^{\hat{f}}$ and $r_2^{\hat{f}}$ are smaller than $\hat{f}$ with respect to a given precedence relation on the defined symbols; A is a set of independent first-order positions of $r_2^{\hat{f}}$ without the root position; $\boldsymbol{\delta}$ is either the null vector or a k-dimensional unit vector, i.e., all components of $\boldsymbol{\delta}$ are zero except one which may be zero or one. The first-order positions are those not below a defined symbol. Two positions are independent if none is a prefix of the other.

Let $\mathcal{R}$ be the set of all rewrite rules associated with the defined symbols. The rewrite relation $\longrightarrow_{\mathcal{R}}$ generated by $\mathcal{R}$ is the smallest relation that contains $\mathcal{R}$, and is closed under congruence and substitution. By $t{\downarrow_{\mathcal{R}}}$ we denote the normal form of t with respect to $\mathcal{R}$. Note that $t{\downarrow_{\mathcal{R}}}$ is a first-order term if t contains no counter variables. The first-order terms represented by a primal term t are defined as $L(t) = \{t\xi{\downarrow_{\mathcal{R}}} \mid \xi{:}\,\mathcal{C} \longrightarrow \mathbb{N}\}$. Two primal terms s and t are *equivalent*, denoted by $s \doteq t$, if $s\xi{\downarrow_{\mathcal{R}}} = t\xi{\downarrow_{\mathcal{R}}}$ holds for all substitutions $\xi{:}\,\mathcal{C} \longrightarrow \mathbb{N}$.

2.3 Unification

A substitution is a mapping $\sigma{:}\,(\mathcal{V} \cup \mathcal{C}) \longrightarrow (\mathcal{P} \cup \mathcal{L})$, which is well-typed and whose domain is finite, i.e., $\sigma(x) \in \mathcal{P}$ for $x \in \mathcal{V}$, $\sigma(n) \in \mathcal{L}$ for $n \in \mathcal{C}$, and $\mathrm{dom}(\sigma) = \{v \in (\mathcal{V} \cup \mathcal{C}) \mid \sigma(v) \neq v\}$ is finite. The application of σ to a term t is written as $t\sigma$; the composition of two substitutions σ, τ is written as $\sigma\tau$ with the understanding that $t\sigma\tau = (t\sigma)\tau$ for all terms t. We denote σ by the set $\{v \mapsto v\sigma \mid v \in \mathrm{dom}(\sigma)\}$. Normalization is extended to substitutions in the natural way, i.e., $\sigma{\downarrow_{\mathcal{R}}} = \{v \mapsto v\sigma{\downarrow_{\mathcal{R}}} \mid v \in \mathrm{dom}(\sigma)\}$.

A substitution σ is a unifier of two primal terms s and t iff for all $\xi{:}\,\mathcal{C} \longrightarrow \mathbb{N}$ the first-order substitution $\sigma\xi{\downarrow_{\mathcal{R}}}$ unifies the first-order terms $s\xi{\downarrow_{\mathcal{R}}}$ and $t\xi{\downarrow_{\mathcal{R}}}$. A set of unifiers Σ is complete iff for every counter substitution ξ there exists $\sigma \in \Sigma$, such that $\sigma\xi{\downarrow_{\mathcal{R}}}$ is a most general unifier of $s\xi{\downarrow_{\mathcal{R}}}$ and $t\xi{\downarrow_{\mathcal{R}}}$. Note that σ is a unifier of s and t iff $s\sigma \doteq t\sigma$, i.e., our notion of unifiability corresponds to the standard one in unification theory. This is not true for completeness: a unifier need not be an instance of any substitution in a given complete set of unifiers. Unification of primal terms is decidable and finitary, i.e., for any pair of primal terms there exists a finite set of unifiers which is complete. Moreover, complete sets of unifiers can be effectively computed [HG97].

2.4 First-order formulas

In this paper, we use first-order formulas to define the word problem in a concise way and to compare different notions of subsumption. Quantified counter variables are interpreted over the domain of natural numbers, quantified first-order variables over the Herbrand universe with respect to the underlying set of function symbols. Free variables are treated as constants.

Additionally, we use vectors and notations from linear algebra as a compact representation of similar objects. For example, $\mathbf{x} \doteq \mathbf{s}(\mathbf{k})$ stands for a set of equations of the form $x \doteq s(\mathbf{k})$, where x is a variable from $\mathbf{x}$ and $s \in \mathbf{s}$ is a term containing variables $k_1, k_2, \ldots$ from $\mathbf{k}$. Furthermore, $\{\mathbf{n} \mapsto \mathbf{Ck} + \mathbf{c}\}$ represents the substitution replacing each variable in $\mathbf{n}$ by the corresponding row in the vector of linear expressions, which is obtained by multiplying the matrix $\mathbf{C}$ of natural numbers by the vector $\mathbf{k}$ of counter variables and adding the vector $\mathbf{c}$.

Let s and t be primal terms containing the variables $\mathbf{x} = \mathcal{V}ar(s)$, $\mathbf{y} = \mathcal{V}ar(t)$, $\mathbf{m} = \mathcal{CV}ar(s)$ and $\mathbf{n} = \mathcal{CV}ar(t)$. A complete set of unifiers for s and t can be considered as a solved form of the equation $s \doteq t$ in the following way. A unifier $\sigma = \{\mathbf{x} \mapsto s'(\mathbf{k}), \mathbf{y} \mapsto t'(\mathbf{k}), \mathbf{m} \mapsto \mathbf{Ck} + \mathbf{c}, \mathbf{n} \mapsto \mathbf{Dk} + \mathbf{d}\}$, where $\mathbf{k}$ are auxiliary counter variables introduced during unification, corresponds to the formula $\phi_\sigma(\mathbf{x}, \mathbf{y}, \mathbf{m}, \mathbf{n}) = \exists \mathbf{k} \big(\mathbf{x} \doteq s'(\mathbf{k}) \wedge \mathbf{y} \doteq t'(\mathbf{k}) \wedge \mathbf{m} = \mathbf{Ck} + \mathbf{c} \wedge \mathbf{n} = \mathbf{Dk} + \mathbf{d}\big)$. Note that unification does not introduce auxiliary first-order variables. However, s' and t' may contain variables from $\mathbf{x}$ and $\mathbf{y}$; in this case these variables do not occur in the domain of the substitution. The formula associated with a complete set of unifiers Σ is the disjunction of the formulas corresponding to the single unifiers: $\phi_\Sigma(\mathbf{x}, \mathbf{y}, \mathbf{m}, \mathbf{n}) = \bigvee_{\sigma \in \Sigma} \phi_\sigma(\mathbf{x}, \mathbf{y}, \mathbf{m}, \mathbf{n})$. Therefore the formulas $s \doteq t$ and $\phi_\Sigma(\mathbf{x}, \mathbf{y}, \mathbf{m}, \mathbf{n})$ are equivalent.

2.5 Miscellaneous notations

If t is a primal term and $A \subseteq \mathcal{P}os(t)$ is a set of independent first-order positions, then $t[\circ]_A$ is called a *context*. If s is a context and t is a context or primal term, then the concatenation of s and t, denoted by $s \cdot t$, is the context or primal term $s\{\circ \mapsto t\}$. Concatenation is associative, hence we drop parentheses where possible. The empty context $\circ$ serves as unit element with respect to concatenation. Exponentiation is defined by $s^0 = \circ$ and $s^{i+1} = s \cdot s^i$.

The depth of a primal term t, denoted by $\mathit{depth}(t)$, is recursively defined as $\mathit{depth}(t) = 0$ for $t \in (\mathcal{V} \cup \mathcal{F}_0)$, and $\mathit{depth}(f(\mathbf{t})) = \mathit{depth}(\hat{f}(\mathbf{l}; \mathbf{t})) = 1 + \mathit{depth}(\mathbf{t})$ for $f \in \mathcal{F}_p$ $(p > 0)$ and $\hat{f} \in \mathcal{D}$. The depth of a set or vector of terms $\mathbf{t}$ is defined as $\mathit{depth}(\mathbf{t}) = \max\{\mathit{depth}(t) \mid t \in \mathbf{t}\}$. The depth of the set of rewrite rules $\mathcal{R}$ associated with $\mathcal{D}$ is the depth of the set of all right hand sides: $\mathit{depth}(\mathcal{R}) = \mathit{depth}(\{r_1^{\hat{f}}, r_2^{\hat{f}}[\hat{f}(m, \mathbf{n} + \boldsymbol{\delta}; \mathbf{x})]_A \mid \hat{f} \in \mathcal{D}\})$.

3 Redundancy elimination

Recurrent term schematizations are of potential use in all areas concerned with first-order terms, mostly in automated deduction, like term rewriting with equational completion and proofs by consistency, or clausal theorem proving. An ubiquitous problem appearing there is the duplication of objects. Redundancy elimination plays therefore a vital role. In the simplest case, we need to maintain the set property, where no element (term, clause, literal) must occur twice. Another case of redundancy is the presence of two elements, where one is an instance of the other. In the first case we have to solve the *word problem*, i.e., to

determine whether two terms s and t represent the same object in the underlying theory. The latter case is usually referred to as the *subsumption problem.*

3.1 Word problem

Definition 1. The **word problem** for two primal terms s and t is the question whether the formula $\forall \mathbf{n}\ (s \doteq t)$ is valid in the equational theory generated by $\mathcal{R}$, where $\mathbf{n} = \mathcal{CV}ar(s) \cup \mathcal{CV}ar(t)$.

One possibility to solve the word problem is to reduce s and t to unique normal forms, followed by a check whether the latter are syntactically equal. This approach is described for R-strings in [Sal91]. In this paper, we choose a different approach: we transform the word problem to a unification problem and a subsequent problem in Presburger arithmetic. The first method is efficient but works only if we can define a unique normal form. In general, there is no obvious way of defining the normal form of a primal term. Our approach does not depend on a specific syntactic representation for schematizations, but requires only the existence of a finitary and terminating unification algorithm. Therefore, our method is applicable to all known recurrent schematizations, i.e., to ρ-terms, I-terms, R-terms, and primal grammars.

We proceed in three steps.

1. *Elimination of first-order variables:* replace all first-order variables by new constants. Observe that the formula $\forall \mathbf{n}(s \doteq t)$ is valid if and only if the corresponding formula $\forall \mathbf{n}(s^* \doteq t^*)$ is valid, where the terms s^*, t^* are obtained from the terms s, t by replacing each first-order variable x by a new constant c_x.
2. *Unification:* solve the equation $s^* \doteq t^*$. We solve the equation $s^* \doteq t^*$ by means of unification. Note that a finitary and terminating unification algorithm exists for all four known recurrent schematizations. This means that the output of the unification algorithm is a finite disjunction of formulas $\exists \mathbf{k}(\mathbf{n} = \mathbf{N}_i\mathbf{k} + \mathbf{d}_i)$, where $\mathbf{N}_i$ and $\mathbf{d}_i$ is a matrix and a vector of non-negative integers, respectively, and $\mathbf{k}$ are new counter variables introduced during unification. The resulting formula $\phi(\mathbf{n}) = \exists \mathbf{k} \bigvee_i (\mathbf{n} = \mathbf{N}_i\mathbf{k} + \mathbf{d}_i)$ contains only counter variables, since there are no first-order variables in s^* and t^*.
3. *Validity check:* check whether the formula $\forall \mathbf{n}\ \phi(\mathbf{n})$ is valid. The formula $\phi(\mathbf{n})$ represents a complete set of unifiers, one per disjunct, of the problem $s^* \doteq t^*$. To show that the universally quantified formula $\forall \mathbf{n}(s^* \doteq t^*)$ is valid, we need to prove that the unifiers from $\phi(\mathbf{n})$ cover the whole Cartesian product $\mathbb{N}^{|\mathbf{n}|}$. By correctness of the applied unification algorithm, the formulas $\forall \mathbf{n}(s^* \doteq t^*)$ and $\forall \mathbf{n}\ \phi(\mathbf{n})$ are equivalent. The latter expression is a Π_2-formula of Presburger arithmetic and can be solved by usual methods [Coo72].

3.2 Subsumption problem

In the first-order case, a term s subsumes a term t if there exists a substitution σ, such that $s\sigma = t$. In the free algebra, this is equivalent to $\exists \mathbf{x}(s = t)$, where

$\mathbf{x} = \mathcal{V}ar(s)$. An alternative definition is that the formula $\forall \mathbf{y} \exists \mathbf{x}(s = t)$ is valid, where $\mathbf{x} = \mathcal{V}ar(s)$ and $\mathbf{y} = \mathcal{V}ar(t)$. These two definitions are equivalent, except for singular signatures, since in the empty theory (without axioms) validity in the equational theory is equivalent to validity in the inductive theory.

For schematizations, there are several possibilities to define subsumption. Let s and t be two primal terms from a schematization G, where $\mathbf{m} = \mathcal{CV}ar(s)$, $\mathbf{n} = \mathcal{CV}ar(t)$, $\mathbf{x} = \mathcal{V}ar(s)$, and $\mathbf{y} = \mathcal{V}ar(t)$. Recall that we check the validity of formulas in the equational theory of $\mathcal{R}$, i.e., the free algebra generated by $\mathcal{R}$. The possibilities to define that s subsumes t are: (1) the formula $\exists \mathbf{m} \exists \mathbf{x}(s \doteq t)$ is valid; (2) the formula $\forall \mathbf{n} \forall \mathbf{y} \exists \mathbf{m} \exists \mathbf{x}(s \doteq t)$ is valid; (3) the formula $\forall \mathbf{n} \exists \mathbf{m}(s \doteq t)$ is valid; (4) the formula $\forall \mathbf{n} \exists \mathbf{m} \exists \mathbf{x}(s \doteq t)$ is valid. The first two approaches are straightforward extensions of the first-order concept. The second approach does not meet a natural requirement for subsumption, namely independence of the underlying signature. Subsumption should be a local test on two terms independent of other elements. There exist two terms s, t, such that s subsumes t (according to the second definition) over a signature $\mathcal{F}$, but not over an extended signature $\mathcal{F}' \supset \mathcal{F}$ [AHL97, Example 14]. The same terms also show that the first two subsumption concepts are not equivalent, since there is no substitution σ, such that $s\sigma \doteq t$, as required by the first concept. The problems with the second concept originate from quantification over first-order variables. One possibility to avoid them is to quantify only the counter variables, as in the third approach. This concept is not satisfactory either, since it does not capture usual first-order subsumption. When we extend the third concept with usual equational first-order subsumption, we get the fourth concept.

Hence, we have two suitable concepts for subsumption: the first and the last one. Intuitively, the first concept expresses that there is a uniform mapping σ, relating the term s and t in the equational theory of the schematization. In particular, for the counter variable vectors $\mathbf{m}$ and $\mathbf{n}$, this means that $\mathbf{m}$ is a linear expression of $\mathbf{n}$. In contrast, the fourth concept requires this uniformity only on the first-order level; the vectors $\mathbf{m}$ and $\mathbf{n}$ need not be related by a linear function. Clearly, the first concept implies the fourth concept. The converse is not true.

The last subsumption concept encompasses the first one. Moreover, the last concept corresponds to the natural view that schematizations are just a finite representation of infinite sets of first-order terms: s subsumes t if every term represented by t is subsumed by a term represented by s. Therefore we adopt the last concept of subsumption.

Definition 2. Let s and t be primal terms, where $\mathbf{m} = \mathcal{CV}ar(s)$, $\mathbf{n} = \mathcal{CV}ar(t)$, and $\mathbf{x} = \mathcal{V}ar(s)$. The term s **subsumes** t if the formula $\forall \mathbf{n} \exists \mathbf{m} \exists \mathbf{x}(s \doteq t)$ is valid. A set S subsumes a set T if for each term $t' \in T$ there exists a term $s' \in S$, such that s' subsumes t'.

A primal term s subsumes a primal term t iff the set $L(s)$ subsumes the set $L(t)$.

Similar to the word problem, we want to reduce subsumption to unification. We proceed in four steps: we replace certain first-order variables by new constants, apply the unification algorithm, simplify the resulting formula, and check its validity in Presburger arithmetic.

1. *Elimination of first-order variables in t:* replace all first-order variables in t by new constants, producing the term t^*. The formula $\forall\mathbf{n}\exists\mathbf{m}\exists\mathbf{x}(s \doteq t)$ is valid iff $\forall\mathbf{n}\exists\mathbf{m}\exists\mathbf{x}(s \doteq t^*)$ holds by the way how we interpret free variables.
2. *Unification:* solve the equation $s = t^*$ by means of a unification algorithm. Its output can be written as the finite formula $\phi(\mathbf{m},\mathbf{n},\mathbf{x}) = \exists\mathbf{k}\bigvee_i(\mathbf{x} = \mathbf{u}_i(\mathbf{k}) \wedge \mathbf{m} = \mathbf{M}_i\mathbf{k}+\mathbf{c}_i \wedge \mathbf{n} = \mathbf{N}_i\mathbf{k}+\mathbf{d}_i)$, where $\mathbf{k}$ are the new counter variables introduced during unification, $\mathbf{M}_i, \mathbf{N}_i$ are matrices of non-negative integers, and $\mathbf{c}_i$, $\mathbf{d}_i$ are vectors of non-negative integers.
3. *Simplification:* remove the equations $\mathbf{x} = \mathbf{u}_i(\mathbf{k})$ and $\mathbf{m} = \mathbf{M}_i\mathbf{k} + \mathbf{c}_i$ from the formula $\phi(\mathbf{m},\mathbf{n},\mathbf{x})$, producing $\phi'(\mathbf{n})$. Note that $\exists\mathbf{m}\exists\mathbf{x}\ \phi(\mathbf{m},\mathbf{n},\mathbf{x})$ is equivalent to $\phi'(\mathbf{n})$, since the variables $\mathbf{m}$ and $\mathbf{x}$ are existentially quantified and appear only once and separated on the left-hand side of equations.
4. *Validity check:* check if $\forall\mathbf{n}\ \phi'(\mathbf{n})$ is valid. The result $\forall\mathbf{n}\exists\mathbf{k}\bigvee_i(\mathbf{n} = \mathbf{N}_i\mathbf{k}+\mathbf{d}_i)$ belongs to the Π_2-fragment of Presburger arithmetic.

3.3 Complexity issues

Both the word problem and the subsumption problem reduce in the last step to a Π_2-formula in Presburger arithmetic. While the complexity of full Presburger arithmetic is at least doubly exponential and Cooper presents in [Coo72] an algorithm of triple exponential complexity, the Π_2-fragment is only coNP-complete, as it was proved by Grädel [Grä88] and Schöning [Sch97]. Our formulas are quite simple and do not cover the whole Π_2-fragment: they are of the form $\forall\mathbf{n}\exists\mathbf{k}\bigvee_i(\mathbf{n} = \mathbf{N}_i\mathbf{k} + \mathbf{d}_i)$, i.e., the formula is in disjunctive normal form and the variables $\mathbf{n}$ appear only once separated on the left-hand side. Therefore we can ask whether our special problems are still coNP-complete. The lower bound reductions used by Grädel and Schöning require more complex formulas. However, following an idea in [Sch97], due to Grädel, we can prove the coNP-hardness of our problems by a reduction from SIMULTANEOUS INCONGRUENCES [GJ79]. This NP-complete problem is defined as follows: given a set $\{(a_1,b_1),\ldots,(a_p,b_p)\}$ of ordered pairs of positive integers, with $a_i \leq b_i$, the problem asks whether there is an integer n such that $n \not\equiv a_i \pmod{b_i}$ holds for all i. We use the dual problem to show coNP-hardness. Encoding $n \equiv a_i \pmod{b_i}$ as $\exists k(n = b_ik + a_i)$, we obtain the disjunction $\exists k\bigvee_{i=1}^{p}(n = b_ik+a_i)$. The final formula is $\forall n\exists k\bigvee_i(n = b_ik+a_i)$, which is of the same type as the formulas obtained from word and subsumption problems. Note that in both cases only the problem solved in the last step is coNP-complete. The overall complexity of our algorithms is determined by the complexity of unification. In particular, the cardinality of a minimal complete set of unifiers can be at least exponential [Sal91]; and we have to compute all solutions to obtain the formula. Hence, the formula in the last step can be exponentially longer than the input of the original problem.

4 Complement problem

If t is a first-order term, its Herbrand universe is $\mathcal{H}(t) = \{t\sigma \mid \sigma{:}\,\mathcal{X} \longrightarrow \mathcal{T}(\mathcal{F})\}$, the set of the ground instances of t with respect to the underlying signature $\mathcal{F}$.

Similarly, if T is a set of first-order terms, its Herbrand universe $\mathcal{H}(T)$ is the union of the Herbrand universes $\mathcal{H}(t)$ for each $t \in T$. For a primal term t, its Herbrand universe is the set $\mathcal{H}(L(t))$, i.e., the Herbrand universe of the schematized set. Finally, the Herbrand universe of a set of primal terms T is obtained as the union of the Herbrand universes $\mathcal{H}(t)$ for each $t \in T$.

Given a set of first-order or primal terms T, its *complement* is the set $T^c = \mathcal{T}(\mathcal{F}) \setminus \mathcal{H}(T)$. A class $\mathbb{C}$ is a collection of sets of terms satisfying a common property. For a given class $\mathbb{C}$, the *complement problem* is the question whether for each finite set of terms $T \in \mathbb{C}$ there exists a finite set of terms $T' \in \mathbb{C}$, such that $\mathcal{H}(T') = T^c$ holds. The set T' is called a finite complement representation.

For first-order terms, Lassez and Marriott proved that finite sets of linear terms always have a finite complement representation [LM87]. On the other hand, they showed that this is not true for arbitrary finite sets of first-order terms. Since schematizations were introduced to increase the expressive power of first-order terms, we might expect to be able to represent the complements of non-linear terms by a finite set of primal terms. However, as we show in the sequel, already the very simple non-linear term $f(x, x)$ has no finite complement representation by primal terms.

The potential of primal terms resides in the possibility to generate arbitrarily deep terms by iterating contexts. The expressive power of iteration is limited by the fact that the number of contexts must be finite. The maximal number of consecutive iterations during a reduction of a primal term is measured by the iteration depth. Each iteration terminates with the application of the base rule $\hat{f}(0, \ldots) \to r_1^{\hat{f}}$ for some defined symbol $\hat{f}$. Therefore we can determine the iteration depth by counting the occasions when a variable gets decremented to 0. The iteration depth of a primal term is then the maximum over all reductions. Inspection of the rewrite system $\mathcal{R}$ reveals that there is a correspondence between the application of base rules and the number of counter positions present in the primal term: each iteration consumes a counter position.

Definition 3. The **iteration depth** of a primal term is the function τ defined recursively as follows:

- $\tau(x) = \tau(a) = 0$ for a first-order variable x and a constant a,
- $\tau(f(t_1, \ldots, t_n)) = \max\{\tau(t_i) \mid i = 1, \ldots, n\}$ for an n-ary function symbol f,
- $\tau(\hat{f}(\mathbf{c}; t_1, \ldots, t_n)) = |\mathbf{c}| + \max\{\tau(t_i) \mid i = 1, \ldots, n\}$ for a defined symbol $\hat{f}$.

The iteration depth naturally extends to a set of primal terms T, defined by $\tau(T) = \max\{\tau(t) \mid t \in T\}$.

This definition emphasizes the static aspect by looking at the primal term only. The operational aspect, namely counting the occasions when a variable is decremented to 0, is expressed by the equalities $\tau(\hat{f}(0, \ldots)\theta) = 1 + \tau(r_1^{\hat{f}}\theta)$ and $\tau(\hat{f}(n+1, \ldots)\theta) = \tau(r_2^{\hat{f}}\theta)$ for each defined symbol $\hat{f}$ and substitution θ.

Iteration of contexts consumes resources of the primal term. On one hand, a single iteration can produce an arbitrarily deep term. On the other hand,

there are ground first-order terms that require a certain iteration depth. We use two different contexts, $f(\circ, a)$ and $f(a, \circ)$, to force a consumption of resources. Consider the ground term $s = f(\circ, a)^m \cdot a$. If the value of m is sufficiently large, then a primal term t representing s must contain a defined symbol through which we iterate the context $f(\circ, a)$, and the iteration depth of t must be at least 1. If we simply concatenate two blocks of the same context, like in $f(\circ, a)^m \cdot f(\circ, a)^m \cdot a$, we do not necessarily need to increase the iteration depth of the primal term. However, if we insert the context $f(a, \circ)$ between the two blocks, producing the term $s = f(\circ, a)^m \cdot f(a, \circ) \cdot f(\circ, a)^m \cdot a$, we force a primal term t representing s to have an iteration depth of at least 2. Repeating the step, this idea leads to an upper bound on the number of context blocks $f(\circ, a)^m \cdot f(a, \circ)$ that can be represented by a given primal term t.

Lemma 4. *Let t be a primal term without first-order variables and let $s = w \cdot (f(\circ, a)^m \cdot f(a, \circ))^n \cdot a$ be a ground first-order term, where w is a proper subcontext of $f(\circ, a)^m \cdot f(a, \circ)$. If $s \in L(t)$ and $m > \tau(t) \times depth(\mathcal{R}) + depth(t)$ then $n \leq \tau(t)$.*

The lemma indicates that if we choose the value of n in the term s larger than the iteration depth $\tau(t)$ of the primal term t, then we cannot represent s by t using iteration only. Therefore, the term t must contain variables.

Corollary 5. *If $s = (f(\circ, a)^m \cdot f(a, \circ))^n \cdot a$ is an instance of a primal term t with $\tau(t) < n$ and $m > \tau(t) \times depth(\mathcal{R}) + depth(t)$, then t must end with a variable.*

We show by contradiction that the complement of the first-order term $f(x, x)$ has no finite representation. The underlying idea is to choose a ground term $s = f(s_1, s_2)$ from the complement, such that both s_1 and s_2 are too complex to be produced by iteration alone, and s_2 is twice as deep as s_1. Therefore a term representing s must be of the form $f(u, v)$, where both u and v end with variables y and z, respectively. If $y \neq z$ then the terms $f(u, v)$ and $f(x, x)$ are unifiable, contradicting the assumption that $f(u, v)$ represents (part of) the complement of $f(x, x)$. If $y = z$, then there is no substitution σ, such that $u\sigma{\downarrow_{\mathcal{R}}} = s_1$ and $v\sigma{\downarrow_{\mathcal{R}}} = s_2$ hold.

Theorem 6. *The complement of a finite set of first-order terms cannot be represented in general by a finite set of primal terms.*

5 Conclusion

We presented general algorithms for solving the word and the subsumption problem for primal terms that also work for ρ-terms, I-terms, and R-terms. The algorithms require a finitary unification algorithm for the schematization formalisms, as well as a solver for the Π_2-fragment of Presburger arithmetic. Still, there are some problems left, especially concerning efficiency. For the word problem, it would be interesting to have an algorithm that computes first a suitable normal

form of primal terms, followed by a syntactic comparison. Algebraically, this amounts to axiomatizing the theory of primal terms.

We also showed that equations and primal terms are not sufficient for describing in general the complement of first-order terms. This result trivially extends to recurrent term schematizations, since first-order terms are just a special case. On the other hand, the complement problem is easily solvable if we extend the language by negation and quantification. Then the complement can be expressed by a formula in the first-order theory of term schematizations. In this context, we are interested in deciding the validity of formulas and in obtaining solved forms, e.g., by quantifier elimination. Peltier showed in [Pel97] that the first-order theory of R-terms is decidable by quantifier elimination. The decidability of the first-order theory of primal terms is still an open problem.

References

[AHL97] A. Amaniss, M. Hermann, and D. Lugiez. Set operations for recurrent term schematizations. In M. Bidoit and M. Dauchet, editors, *Proc. 7th TAPSOFT Conference, Lille (France)*, LNCS 1214, pages 333–344. Springer, 1997.

[CH95] H. Chen and J. Hsiang. Recurrence domains: Their unification and application to logic programming. *Information and Computation*, 122:45–69, 1995.

[Com95] H. Comon. On unification of terms with integer exponents. *Mathematical Systems Theory*, 28(1):67–88, 1995.

[Coo72] D.C. Cooper. Theorem proving in arithmetic without multiplication. In B. Meltzer and D. Mitchie, editors, *Machine Intelligence*, volume 7, pages 91–99. Edinburgh University Press, 1972.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and intractability: A guide to the theory of NP-completeness*. W.H. Freeman and Co, 1979.

[Grä88] E. Grädel. Subclasses of Preburger arithmetic and the polynomial-time hierarchy. *Theoretical Computer Science*, 56(3):289–301, 1988.

[HG97] M. Hermann and R. Galbavý. Unification of infinite sets of terms schematized by primal grammars. *Theoretical Computer Science*, 176(1-2):111–158, 1997.

[LM87] J.-L. Lassez and K. Marriott. Explicit representation of terms defined by counter examples. *J. Automated Reasoning*, 3(3):301–317, 1987.

[Pel97] N. Peltier. Increasing model building capabilities by constraint solving on terms with integer exponents. *J. Symbolic Computation*, 24(1):59–101, 1997.

[Sal91] G. Salzer. Deductive generalization and meta-reasoning, or how to formalize Genesis. In *Österreichische Tagung für Künstliche Intelligenz, Informatik-Fachberichte* 287, pages 103–115. Springer, 1991.

[Sal92] G. Salzer. The unification of infinite sets of terms and its applications. In A. Voronkov, editor, *Proc. 3rd LPAR Conference, St. Petersburg (Russia)*, LNCS (LNAI) 624, pages 409–420. Springer, 1992.

[Sal94] G. Salzer. Primal grammars and unification modulo a binary clause. In A. Bundy, editor, *Proc. 12th CADE Conference, Nancy (France)*, LNCS (LNAI) 814, pages 282–295. Springer, 1994.

[Sch97] U. Schöning. Complexity of Presburger arithmetic with fixed quantifier dimension. *Theory of Computing Systems*, 30(4):423–428, 1997.