On the Unification of Term Schemata

David Cerna[1,2], Alexander Leitsch[3], and Anela Lolic[4]

[1] Institute for Formal Methods and Verification, JKU, Linz, Austria
[2] Research Institute for Symbolic Computation, JKU, Hagenberg, Austria
david.cerna@jku.at,david.cerna@risc.jku.at
[3] Institute of Logic and Computation, TU Wien, Vienna, Austria
leitsch@logic.at
[4] Institute of Logic and Computation, TU Wien, Vienna, Austria
anela@logic.at

Abstract. Term schemata are infinite sequences of terms which are defined inductively. We investigate the unification problem for term schemata and formulate some open problems. The solution of these problems is relevant to the analysis of proof schemata, in particular to schematic cut-elimination.

Keywords: schema · unification · induction.

1 Introduction

Recursive definitions of functions play a central role in computer science, particularly in functional programming. While recursive definitions of formulas and proofs are less common, they are of increasing importance in automated proof analysis. Proof schemata, i.e. recursively defined infinite sequences of proofs, serve as an alternative formulation of induction. Prior to the formalization of the concept, an analysis of Fürstenberg's proof of the infinitude of primes [1] suggested the need for a formalism quite close to the type of proof schemata defined in [6]. The underlying method for this analysis was CERES [2] (cut-elimination by resolution) which, unlike reductive cut-elimination, can be applied to recursively defined proofs by extracting a schematic unsatisfiable universal formula (the characteristic schema) and constructing a recursively defined refutation. Moreover, Herbrand's theorem can be extended to an expressive fragment of proof schemata, that is those formalizing k-induction [4,6]. Unfortunately, the construction of recursively defined refutations is a highly complex task. In previous work [6] a superposition calculus for certain types of formulas was used for the construction of refutation schemata, but only works for a weak fragment of arithmetic and is hard to use interactively. In [3] the schematic approach was substantially generalized; the new method is capable of handling several recursion parameters and thus can deal with nested inductions. To refute the corresponding characteristic schemata a new resolution calculus for universal formula schemata was developed (see also [3]). A crucial part of this calculus is unification which is needed to define single resolution steps. Unlike to ordinary first-order unification the problem here consists in *unifying term schemata*,

i.e. syntactic expressions representing infinite (recursively defined) sequences of terms. In [3] we introduced the new concept of s-unification which replaces ordinary unification in case of schemata. But s-unification is just one possibility to approach unification of term schemata. In this paper we describe the general problem of unifying term schemata and characterize different subclasses of schematic unification problems. Several models for unification of term schemata were developed in the 1990ties; we just mention [5] and [7]. However, our approach differs from those mentioned above; it is based on primitive recursive definition and the unification problem is undecidable in general.

2 Unification problems for term schemata

Below we define the general problem, give some basic definitions and formulate decision problems for schematic unification. In the most general sense a term schema is an arbitrary infinite sequence of first-order terms s_n.

Definition 1 (unification of term schemata). *Given two schemata of first order terms $\hat{s}\colon (s_n)_{n\in\mathbb{N}}$ and $\hat{t}\colon (t_n)_{n\in\mathbb{N}}$ we define $\hat{s}, \hat{t}$ as* unifiable *if there exists a sequence $(\lambda_n)_{n\in\mathbb{N}}$ of substitutions such that $s_n\lambda_n = t_n\lambda_n$ for all $n \in \mathbb{N}$.*

The definition above is just an informal one. For the formal definition of term schemata on which this paper is based see [3]. Here we define two different types of schematic unification problems, simple and global ones. We use basic definitions from [3], especially defined and undefined symbols and an ordering $<$ of the defined symbols.

For applications in computational logic only computable term schemata makes sense. Here we consider only schemata defined by primitive recursion. The so-called *simple term schemata* are schemata with a fixed number of variables defined via primitive recursion:

Definition 2 (simple term schema). *Let $\boldsymbol{x}$ be a tuple of first-order variables (variables of type ι)and n be a parameter (a variable of type ω). A* simple term schema *is defined by primitive recursive definitions of the form*

$$\hat{f}(\boldsymbol{x}, \mathbf{0}) = g(\boldsymbol{x}),$$
$$\hat{f}(\boldsymbol{x}, s(n)) = h(\boldsymbol{x}, n, z)\{z \leftarrow \hat{f}(\boldsymbol{x}, n)\}$$

where $g(\boldsymbol{x})$ is a term over the variables $\boldsymbol{x}$ and $h(\boldsymbol{x}, n, z)$ is a term over the variables $\boldsymbol{x}, z$ and the parameter n. If $\hat{f}$ is not a minimal defined symbol then both $g(\boldsymbol{x})$ and $h(\boldsymbol{x}, n, z)$ may contain defined symbols $\hat{u}$ with $\hat{u} < \hat{f}$.

Note that the general unfication problem for simple term schemata is undecidable (the equivalence problem of loop-programs can be reduced to it).

Example 1. Consider the following simple term schema:

$$\hat{f}(x, \mathbf{0}) \;=\; h(a, a) \qquad \hat{f}(x, s(n)) \;=\; h(x, \hat{f}(x, n))$$

$$\hat{f}_1(x,y,\mathbf{0}) \;=\; h(a,a) \qquad \hat{f}_1(x,y,s(n)) \;=\; h(x,\hat{f}(y,n))$$
$$\hat{g}(x,y,\mathbf{0}) \;=\; h(a,a) \qquad \hat{g}(x,y,s(n)) \;=\; h(\hat{g}(x,y,n),y)$$

Using these simple term schemata we can form the following four unification problems:

$$\hat{f}(x,s(n)) \stackrel{?}{=} \hat{g}(x,x,s(n)),\;\; \hat{f}(x,s(n)) \stackrel{?}{=} \hat{g}(x,y,s(n)),$$
$$\hat{f}(x,s(n)) \stackrel{?}{=} \hat{g}(y,y,s(n)),\;\; \hat{f}_1(x,y,s(n)) \stackrel{?}{=} \hat{g}(z,z,s(n)).$$

Notice that the first three problems fail due to the occurs-check while the fourth problem does not and is unifiable. Let us consider the first and the last problem in more detail:

$\hat{f}(x,s(n)) \stackrel{?}{=} \hat{g}(x,x,s(n))$ is solvable iff for all subsitutions of the number variable n by a numeral k the normal forms of $\hat{f}(x,s(k))$ and $\hat{g}(x,x,s(k))$ are unifiable (they are ordinary first-order terms). Therefore the first unification problem is unsolvable because $\hat{f}(x,s(1))$ and $\hat{g}(x,x,s(1))$ are not unifiable; note that

$$\hat{f}(x,2) = h(x,\hat{f}(x,1)) = h(x,h(x,\hat{f}(x,0))) = h(x,h(x,h(a,a))),$$
$$\hat{g}(x,x,2) = h(\hat{g}(x,x,1),x) = h(h(\hat{g}(x,x,0),x),x) = h(h(h(a,a),x),x).$$

The fourth problem is solvable. The infinite sequence of unifiers is given by the schematic substitution

$$\vartheta_n = \{x \leftarrow \hat{g}(\hat{f}(y,n),\hat{f}(y,n),n),\;\; z \leftarrow \hat{f}(y,n)\}.$$

Even though simple term schemata allow for recursive definitions, recursive variable occurrence causes unification to fail in most cases, thus, like the fourth example, unification is usually decided by analyzing the structure of the terms.

In contrast to simple term schemata *global term schemata* are based on primitive recursive definitions using *global* variables instead of ordinary first-order variables (see [3]). These schemata may contain increasing numbers of variables depending on the assignment of the parameter n.

Definition 3 (global term schema). *Let $\boldsymbol{X}$ be a tuple of global variables (variables of type $\omega \to \iota$) and n be a parameter (a variable of type ω). A* global term schema *is defined by primitive recursive definitions of the form*

$$\hat{f}(\boldsymbol{X},\mathbf{0}) = t(\boldsymbol{X}),$$
$$\hat{f}(\boldsymbol{X},s(n)) = s(\boldsymbol{X},n,z)\{z \leftarrow \hat{f}(\boldsymbol{X},n)\}$$

where $t(\boldsymbol{X})$ is a term over the global variables $\boldsymbol{X}$ and $s(\boldsymbol{X},n,z)$ is a term over the global variables $\boldsymbol{X}$, the individual variable z and the parameter n. If $\hat{f}$ is not a minimal defined symbol then both $t(\boldsymbol{X})$ and $s(\boldsymbol{X},n,z)$ may contain defined symbols $\hat{u}$ with $\hat{u} < \hat{f}$.

A formal definition of (the semantics of) objects of the form $\hat{f}(\boldsymbol{X}, n)$ can be found in [3]. While for simple term schemata the domain variables of the unification schema form a fixed finite set, the unifiers in global term schemata may have domains which may depend on the parameter n. Still it is possible that a unification schema if of the form

$$\vartheta_n \colon \{X(s_1) \leftarrow t_1, \ldots, X(s_k) \leftarrow t_k\}$$

where k is a fixed number and the domain varies with the value of the parameter n but the size of the domain is always k. In such a case we speak about s-unification schemata [3]. The following examples illustrates such an s-unification.

Example 2. Consider the following global term schema:

$$\begin{array}{lll} \hat{f}(X, \mathbf{0}) = h(a, X(0)) & \quad & \hat{f}(X, s(n)) = h(X(s(n)), \hat{f}(X, n)) \\ \hat{g}(X, \mathbf{0}) = h(X(0), a) & & \hat{g}(X, s(n)) = h(\hat{g}(X, n), X(s(n))) \end{array}$$

Using these schemata we can define the unification problem

$$\hat{f}(X, s(n)) \stackrel{?}{=} \hat{g}(X, s(n))$$

which has as an mgu, $\vartheta(n) \colon \left\{X(n) \leftarrow \hat{h}(n)\right\}$ where $\hat{h}(n)$ is as follows:

$$\hat{h}(\mathbf{0}) = a \qquad \hat{h}(s(n)) = h(\hat{h}(n), \hat{h}(n))$$

$\vartheta(n)$ is an s-unifier; its domains are different for every value of the parameter n but the domain size is always 1.

In the next example we define a global term schema and a corresponding unification problem which is solvable but unsolvable via s-unifiers .

Example 3. Consider the following two global schemata (where the second schema could also be defined as a simple one):

$$\begin{array}{l} \hat{f}(X, 0) = X(0) \;\; \hat{f}(X, n+1) = h(X(n+1), \hat{f}(X, n)), \\ \hat{g}(X, 0) = X(0) \;\; \hat{g}(X, n+1) = h(X(0), \hat{g}(X, n)). \end{array}$$

Note that $\hat{f}(X, 0), \hat{f}(X, 1), \hat{f}(X, 2) \ldots$ evaluate to

$$X(0), h(X(1), X(0)), h(X(2), h(X(1), X(0))), \ldots$$

so the number of different first-order variables is increasing. The unification problem

$$\hat{f}(X, n) \stackrel{?}{=} \hat{g}(X, n)$$

is solvable. The schematic unifier has the following recursive definition

$$\vartheta(0) = \{\}, \;\; \vartheta(n+1) = \{X(n+1) \leftarrow X(0)\} \cup \vartheta(n).$$

Note that $dom(\vartheta(n+1)) = \{X(n+1)\} \cup dom(\vartheta(n))$ and so the size of the domains increases with the value of n. Obviously there exists no s-unifier for this unification problem.

It is crucial to distinguish *free schemata* containing no equations between terms - except primitive recursive definitions - and *theory schemata*. In order to define the class of primitive recursive functions we need a theory schema containing equations defining projections and constant functions. We call such a theory schema the *standard schema*.

Definition 4 (standard schema). *A term schema (simple or global) is called a* standard schema *if it contains*

- *equations of the form* $\hat{g}[\alpha, i](x_1, \ldots, x_\alpha) = x_i$ *(where* $1 \leq i \leq \alpha$*) for every projection function* $I_i^\alpha \colon \iota^\alpha \to \iota$ *where* $I_i^\alpha(\beta_1, \ldots, \beta_n) = \beta_i$ *and*
- *equations of the form* $\hat{h}[\alpha, c](x_1, \ldots, x_\alpha) = c$ *for every constant function of type* $\iota^\alpha \to \iota$.

Here $\hat{g}[\alpha, i], \hat{h}[\alpha, c]$ *are* α*-ary defined function symbols.*

It is well known that the equivalence problem of standard schemata is undecidable and equivalent to the equivalence problem of LOOP programs. The problem is even undecidable when the recursion depth is ≤ 2 (correponding to the equivalence problem of LOOP-2 programs). As a consequence the unification problem for standard term schemata is undecidable as well. However, when we only consider standard term schemata of recursion depth ≤ 1 (corresponding to the LOOP-1 class) the equivalence problem becomes decidable. Based on the definitions above we can define the following 4 problems:

1. is the unification problem for simple standard schemata of recursion depth ≤ 1 decidable?
2. is the unification problem for global standard schemata of recursion depth ≤ 1 decidable?
3. is the unification problem for simple free schemata decidable?
4. is the unification problem for global free schemata decidable?

References

1. Matthias Baaz, Stefan Hetzl, Alexander Leitsch, Clemens Richter, and Hendrik Spohr. Ceres: An analysis of Fürstenberg's proof of the infinity of primes. *Theoretical Computer Science*, 403(2-3):160–175, August 2008.
2. Matthias Baaz and Alexander Leitsch. Cut-elimination and redundancy-elimination by resolution. *Journal of Symbolic Computation*, 29:149–176, 2000.
3. David Cerna, Alexander Leitsch, and Anela Lolic. Schematic refutations of formula schemata. *CoRR*, abs/1902.08055, 2019.
4. Cvetan Dunchev, Alexander Leitsch, Mikheil Rukhaia, and Daniel Weller. Cut-elimination and proof schemata. In *TbiLLC*, volume 8984 of *Lecture Notes in Computer Science*, pages 117–136. Springer, 2013.
5. Miki Hermann and Roman Galbavý. Unification of infinite sets of terms schematized by primal grammars. *Theoretical Computer Science*, 176(1):111 – 158, 1997.
6. Alexander Leitsch, Nicolas Peltier, and Daniel Weller. CERES for first-order schemata. *J. Log. Comput.*, 27(7):1897–1954, 2017.
7. Gernot Salzer. *Unification of Meta-Terms*. PhD thesis, Technical University of Vienna, 1991.