Unification Procedure for Terms with Sequence Variables and Sequence Functions (Extended Abstract)

Temur Kutsia[1]* and Mircea Marin[2]

[1] Research Institute for Symbolic Computation,
Johannes Kepler University Linz,
A-4040 Linz, Austria
kutsia@risc.uni-linz.ac.at

[2] Johann Radon Institute for Computational and Applied Mathematics
Austrian Academy of Sciences
A-4040 Linz, Austria
Mircea.Marin@oeaw.ac.at

1 Introduction

We study term equations with sequence variables and sequence function symbols. A sequence variable can be instantiated by any finite sequence of terms, including the empty sequence. A sequence function abbreviates a finite sequence of functions all having the same argument lists[3]. An instance of such a function is $\mathsf{IntegerDivision}(\mathsf{x},\mathsf{y})$ that abbreviates the sequence $\mathsf{Quotient}(\mathsf{x},\mathsf{y}), \mathsf{Remainder}(\mathsf{x},\mathsf{y})$.

Bringing sequence functions in the language naturally allows Skolemization over sequence variables: let x, y be individual variables, $\overline{x}$ be a sequence variable, and p be a flexible arity predicate symbol. Then $\forall x \forall y \exists \overline{x}. p(x, y, \overline{x})$ Skolemizes to $\forall x \forall y. p(x, y, \overline{f}(x, y))$, where $\overline{f}$ is a binary Skolem sequence function symbol. Another example, $\forall \overline{y} \exists \overline{x}. p(\overline{y}, \overline{x})$, where $\overline{y}$ is a sequence variable, after Skolemization introduces a flexible arity sequence function symbol $\overline{g}$: $\forall \overline{y}. p(\overline{y}, \overline{g}(\overline{y}))$.

In this paper we introduce a unification procedure for solving equations in the free theory with individual and sequence variables, and individual and sequence function symbols. Function symbols can have either fixed or flexible arity. Unification in such a theory is decidable. The procedure enumerates an almost minimal complete set of solutions, and terminates if the set is finite. This work is an extension and refinement of our previous results [12].

We implemented the procedure (without the decision algorithm) in Mathematica [25] on the base of a rule-based programming system ρLog[4] [15].

Equation solving with sequence variables plays an important role in various applications in automated reasoning, artificial intelligence, and programming. Historically, probably the first attempt to implement unification with sequence

* Supported by the Austrian Science Foundation (FWF) under Project SFB F1302 and the European Union "Calculemus" project (HPRN-CT-2000-00102).

[3] Semantically, sequence functions can be interpreted as multi-valued functions.

[4] Available from http://www.ricam.oeaw.ac.at/people/page/marin/RhoLog/.

variables (without sequence functions) was made in the system MVL [7]. It was incomplete because of restricted use of widening technique. The restriction was imposed for the efficiency reasons. No theoretical study of the unification algorithm of MVL, to the best of our knowledge, was undertaken.

Richardson and Fuchs [20] describe another unification algorithm with sequence variables that they call vector variables. Vector variables come with their length attached, that makes unification finitary. The algorithm was implemented but its properties have never been investigated.

Implementation of first-order logic in ISABELLE [17] is based on the sequent calculus formulated using sequence variables (on the meta level). Sequence meta-variables are used to denote sequences of formulae, and individual meta-variables denote single formulae. Since in every such unification problem no sequence meta-variable occurs more that once, and all of them occur only on the top level, ISABELLE, in fact, deals with a finitary case of sequence unification.

Word equations [1, 8, 23] and associative unification [18] can be modelled by syntactic sequence unification using constants, sequence variables and one flexible arity function symbol. In a similar way we can imitate the unification algorithm for path logics closed under right identity and associativity [21].

The SET-VAR prover [4] has a construct called vector of (Skolem) functions that resembles our sequence functions. However, unification does not allow to split vectors of functions between variables: such a vector of functions either entirely unifies with a variable, or with another vector of functions.

The programming language of MATHEMATICA uses pattern matching that supports sequence variables (represented as identifiers with "triple blanks", e.g., `x___`) and flexible arity function symbols. Our procedure (without sequence function symbols) can imitate the behavior of MATHEMATICA matching algorithm.

Buchberger introduced sequence functions in the THEOREMA system [6] to Skolemize quantified sequence variables. In the equational prover of THEOREMA [13] we implemented a special case of unification with sequence variables and sequence functions: sequence variables occurring only in the last argument positions in terms. It makes unification unitary. A similar restriction is imposed on sequence variables in the RELFUN system [5] that integrates extensions of logic and functional programming. RELFUN allows multiple-valued functions as well.

In [12] we described unification procedures for free, flat, restricted flat and orderless theories with sequence variables, but without sequence functions.

The paper is organized as follows. In Section 2 basic notions are introduced. In Section 3 the relation with order-sorted higher-order E-unification is discussed. In Section 4 the unification procedure is defined and its properties are studied. Section 5 is about implementation. Section 6 concludes.

The missing proofs can be found in the report [14].

2 Preliminaries

We assume that the reader is familiar with the standard notions of unification theory [3]. We consider an alphabet consisting of the following pairwise disjoint sets of symbols: individual variables $\mathcal{V}_{\mathrm{Ind}}$, sequence variables $\mathcal{V}_{\mathrm{Seq}}$, fixed arity individual function symbols $\mathcal{F}_{\mathrm{Ind}}^{\mathrm{Fix}}$, flexible arity individual function symbols $\mathcal{F}_{\mathrm{Ind}}^{\mathrm{Flex}}$, fixed arity sequence function symbols $\mathcal{F}_{\mathrm{Seq}}^{\mathrm{Fix}}$, flexible arity sequence function symbols $\mathcal{F}_{\mathrm{Seq}}^{\mathrm{Flex}}$. Each set of variables is countable. Each set of function symbols is finite or countable. Besides, the alphabet contains the parentheses '(', ')' and the comma ','. We will use the following denotations: $\mathcal{V} := \mathcal{V}_{\mathrm{Ind}} \cup \mathcal{V}_{\mathrm{Seq}}$; $\mathcal{F}_{\mathrm{Ind}} := \mathcal{F}_{\mathrm{Ind}}^{\mathrm{Fix}} \cup \mathcal{F}_{\mathrm{Ind}}^{\mathrm{Flex}}$; $\mathcal{F}_{\mathrm{Seq}} := \mathcal{F}_{\mathrm{Seq}}^{\mathrm{Fix}} \cup \mathcal{F}_{\mathrm{Seq}}^{\mathrm{Flex}}$; $\mathcal{F}^{\mathrm{Fix}} := \mathcal{F}_{\mathrm{Ind}}^{\mathrm{Fix}} \cup \mathcal{F}_{\mathrm{Seq}}^{\mathrm{Fix}}$; $\mathcal{F}^{\mathrm{Flex}} := \mathcal{F}_{\mathrm{Ind}}^{\mathrm{Flex}} \cup \mathcal{F}_{\mathrm{Seq}}^{\mathrm{Flex}}$; $\mathcal{F} := \mathcal{F}_{\mathrm{Ind}} \cup \mathcal{F}_{\mathrm{Seq}} = \mathcal{F}^{\mathrm{Fix}} \cup \mathcal{F}^{\mathrm{Flex}}$. The *arity* of $f \in \mathcal{F}^{\mathrm{Fix}}$ is denoted by $\mathcal{A}r(f)$. A function symbol $c \in \mathcal{F}^{\mathrm{Fix}}$ is called a *constant* if $\mathcal{A}r(c) = 0$.

Definition 1. *A* term *over $\mathcal{F}$ and $\mathcal{V}$ is either an* individual *or a* sequence *term defined as follows:*

1. *If $t \in \mathcal{V}_{\mathrm{Ind}}$ (resp. $t \in \mathcal{V}_{\mathrm{Seq}}$), then t is an individual (resp. sequence) term.*
2. *If $f \in \mathcal{F}_{\mathrm{Ind}}^{\mathrm{Fix}}$ (resp. $f \in \mathcal{F}_{\mathrm{Seq}}^{\mathrm{Fix}}$), $\mathcal{A}r(f) = n$, $n \geq 0$, and $t_1, \ldots, t_n$ are individual terms, then $f(t_1, \ldots, t_n)$ is an individual (resp. sequence) term.*
3. *If $f \in \mathcal{F}_{\mathrm{Ind}}^{\mathrm{Flex}}$ (resp. $f \in \mathcal{F}_{\mathrm{Seq}}^{\mathrm{Flex}}$) and $t_1, \ldots, t_n$ $(n \geq 0)$ are individual or sequence terms, then $f(t_1, \ldots, t_n)$ is an individual (resp. sequence) term.*

The *head* of a term $t = f(t_1, \ldots, t_n)$, denoted by $\mathcal{H}ead(t)$, is the function symbol f. We denote by $\mathcal{T}_{\mathrm{Ind}}(\mathcal{F}, \mathcal{V})$, $\mathcal{T}_{\mathrm{Seq}}(\mathcal{F}, \mathcal{V})$, and $\mathcal{T}(\mathcal{F}, \mathcal{V})$, respectively, the sets of all individual terms, all sequence terms, and all terms over $\mathcal{F}$ and $\mathcal{V}$. An *equation* over $\mathcal{F}$ and $\mathcal{V}$ is a pair $\langle s, t \rangle$, denoted by $s \approx t$, where $s, t \in \mathcal{T}_{\mathrm{Ind}}(\mathcal{F}, \mathcal{V})$.

Example 1. Let $x, y \in \mathcal{V}_{\mathrm{Ind}}$, $\overline{x} \in \mathcal{V}_{\mathrm{Seq}}$, $f \in \mathcal{F}_{\mathrm{Ind}}^{\mathrm{Flex}}$, $g \in \mathcal{F}_{\mathrm{Ind}}^{\mathrm{Fix}}$, $\overline{f} \in \mathcal{F}_{\mathrm{Seq}}^{\mathrm{Flex}}$, $\overline{g} \in \mathcal{F}_{\mathrm{Seq}}^{\mathrm{Fix}}$, $\mathcal{A}r(g) = 2$, and $\mathcal{A}r(\overline{g}) = 1$. Then $f(\overline{x}, g(x, y))$ and $f(\overline{x}, \overline{f}(x, \overline{x}, y))$ are individual terms; $\overline{f}(\overline{x}, \overline{f}(x, \overline{x}, y))$ and $\overline{g}(f(x, \overline{x}, y))$ are sequence terms; $f(\overline{x}, \overline{g}(\overline{x}))$, $f(\overline{x}, g(\overline{x}, y))$ and $f(\overline{x}, \overline{g}(x, y))$ are not terms; $f(\overline{x}, g(x, y)) \approx g(x, y)$ is an equation; $f(\overline{x}, g(\overline{x}, y)) \approx g(x, y)$, $\overline{x} \approx f(\overline{x})$ and $\overline{g}(x) \approx f(\overline{x})$ are not equations.

If not otherwise stated, the following symbols, maybe with indices, are used as metavariables: x and y – over individual variables; $\overline{x}$, $\overline{y}$, $\overline{z}$ – over sequence variables; v – over (individual or sequence) variables; f, g, h – over individual function symbols; $\overline{f}$, $\overline{g}$, $\overline{h}$ – over sequence function symbols; a, b, c – over individual constants; $\overline{a}$, $\overline{b}$, $\overline{c}$ – over sequence constants; s, t, r, q – over terms.

Let T be a term, a sequence of terms, or a set of terms. Then we denote by $\mathcal{IV}ar(T)$ (resp. by $\mathcal{SV}ar(T)$) the set of all individual (resp. sequence) variables in T; by $\mathcal{V}ar(T)$ the set $\mathcal{IV}ar(T) \cup \mathcal{SV}ar(T)$; by $\mathcal{IF}un(T)$ (resp. by $\mathcal{SF}un(T)$) the set of all individual (resp. sequence) function symbols in T; by $\mathcal{F}ix(T)$ (resp. by $\mathcal{F}lex(T)$) the set of all fixed (resp. flexible) arity function symbols in T.

Definition 2. *A* variable binding *is either a pair $x \mapsto t$ where $t \in \mathcal{T}_{\mathrm{Ind}}(\mathcal{F}, \mathcal{V})$ and $t \neq x$, or an expression $\overline{x} \mapsto \ulcorner t_1, \ldots, t_n \urcorner$[5] where $n \geq 0$, for all $1 \leq i \leq n$ we have $t_i \in \mathcal{T}(\mathcal{F}, \mathcal{V})$, and if $n = 1$ then $t_1 \neq \overline{x}$.*

[5] To improve readability, we write sequences that bind sequence variables between $\ulcorner$ and $\urcorner$.

Definition 3. *A* sequence function symbol binding *is an expression of the form* $\overline{f} \mapsto \ulcorner \overline{g_1}, \ldots, \overline{g_m} \urcorner$*, where* $m \geq 1$*, if* $m = 1$ *then* $\overline{f} \neq \overline{g_1}$*, and either* $\overline{f}, \overline{g_1}, \ldots, \overline{g_m} \in \mathcal{F}_{\mathrm{Seq}}^{\mathrm{Fix}}$*, with* $\mathcal{A}r(\overline{f}) = \mathcal{A}r(\overline{g_1}) = \cdots = \mathcal{A}r(\overline{g_m})$*, or* $\overline{f}, \overline{g_1}, \ldots, \overline{g_m} \in \mathcal{F}_{\mathrm{Seq}}^{\mathrm{Flex}}$.

Definition 4. *A* substitution *is a finite set of bindings* $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n, \overline{x}_1 \mapsto \ulcorner s_1^1, \ldots, s_{k_1}^1 \urcorner, \ldots, \overline{x}_m \mapsto \ulcorner s_1^m, \ldots, s_{k_m}^m \urcorner, \overline{f_1} \mapsto \ulcorner \overline{g_1^1}, \ldots, \overline{g_{l_1}^1} \urcorner, \ldots, \overline{f_r} \mapsto \ulcorner \overline{g_1^r}, \ldots, \overline{g_{l_r}^r} \urcorner\}$ *where* $n, m, r \geq 0$, $x_1, \ldots, x_n, \overline{x}_1, \ldots, \overline{x}_m$ *are distinct variables and* $\overline{f_1}, \ldots, \overline{f_r}$ *are distinct sequence function symbols.*

Lower case Greek letters are used to denote substitutions. The empty substitution is denoted by ε.

Definition 5. *The* instance *of a term* t *with respect to a substitution* σ*, denoted* $t\sigma$*, is defined recursively as follows:*

1. $x\sigma = \begin{cases} t, & \text{if } x \mapsto t \in \sigma, \\ x, & \text{otherwise.} \end{cases}$
2. $\overline{x}\sigma = \begin{cases} t_1, \ldots, t_n, & \text{if } \overline{x} \mapsto \ulcorner t_1, \ldots, t_n \urcorner \in \sigma,\ n \geq 0, \\ \overline{x}, & \text{otherwise.} \end{cases}$
3. $f(t_1, \ldots, t_n)\sigma = f(t_1\sigma, \ldots, t_n\sigma)$.
4. $\overline{f}(t_1, \ldots, t_n)\sigma =$
$$\begin{cases} \overline{g_1}(t_1\sigma, \ldots, t_n\sigma), \ldots, \overline{g_m}(t_1\sigma, \ldots, t_n\sigma), & \text{if } \overline{f} \mapsto \ulcorner \overline{g_1}, \ldots, \overline{g_m} \urcorner \in \sigma, \\ \overline{f}(t_1\sigma, \ldots, t_n\sigma), & \text{otherwise.} \end{cases}$$

Example 2. Let $\sigma = \{x \mapsto a, y \mapsto f(\overline{x}), \overline{x} \mapsto \ulcorner\urcorner, \overline{y} \mapsto \ulcorner a, \overline{x} \urcorner, \overline{g} \mapsto \ulcorner \overline{g_1}, \overline{g_2} \urcorner\}$. Then $f(x, \overline{x}, \overline{g}(y, \overline{g}()), \overline{y})\sigma = f(a, \overline{g_1}(f(\overline{x}), \overline{g_1}(), \overline{g_2}()), \overline{g_2}(f(\overline{x}), \overline{g_1}(), \overline{g_2}()), a, \overline{x})$.

Definition 6. *The* application *of* σ *on* $\overline{f}$*, denoted* $\overline{f}\sigma$*, is a sequence of function symbols* $\overline{g_1}, \ldots, \overline{g_m}$ *if* $\overline{f} \mapsto \ulcorner \overline{g_1}, \ldots, \overline{g_m} \urcorner \in \sigma$*. Otherwise* $\overline{f}\sigma = \overline{f}$.

Applying a substitution θ on a sequence of terms $\ulcorner t_1, \ldots, t_n \urcorner$ gives a sequence of terms $\ulcorner t_1\theta, \ldots, t_n\theta \urcorner$.

Definition 7. *Let* σ *be a substitution. (1) The* domain *of* σ *is the set* $\mathcal{D}om(\sigma) = \{l \mid l\sigma \neq l\}$ *of variables and sequence function symbols. (2) The* codomain *of* σ *is the set* $\mathcal{C}od(\sigma) = \{l\sigma \mid l \in \mathcal{D}om(\sigma)\}$ *of terms and sequence function symbols*[6]*. (3) The* range *of* σ *is the set* $\mathcal{R}an(\sigma) = \mathcal{V}ar(\mathcal{C}od(\sigma))$ *of variables.*

Definition 8. *Let* σ *and* ϑ *be two substitutions:*

$$\begin{aligned} \sigma = \{\, & x_1 \mapsto t_1, \ldots, x_n \mapsto t_n, \overline{x_1} \mapsto \ulcorner s_1^1, \ldots, s_{k_1}^1 \urcorner, \ldots, \overline{x_m} \mapsto \ulcorner s_1^m, \ldots, s_{k_m}^m \urcorner, \\ & \overline{f_1} \mapsto \ulcorner \overline{f_1^1}, \ldots, \overline{f_{l_1}^1} \urcorner, \ldots, \overline{f_r} \mapsto \ulcorner \overline{f_1^r}, \ldots, \overline{f_{l_r}^r} \urcorner\}, \\ \vartheta = \{\, & y_1 \mapsto r_1, \ldots, y_{n'} \mapsto r_{n'}, \overline{y_1} \mapsto \ulcorner q_1^1, \ldots, q_{k'_1}^1 \urcorner, \ldots, \overline{y_{m'}} \mapsto \ulcorner q_1^{m'}, \ldots, q_{k'_{m'}}^{m'} \urcorner, \\ & \overline{g_1} \mapsto \ulcorner \overline{g_1^1}, \ldots, \overline{g_{l'_1}^1} \urcorner, \ldots, \overline{g_{r'}} \mapsto \ulcorner \overline{g_1^{r'}}, \ldots, \overline{g_{l'_{r'}}^{r'}} \urcorner\}. \end{aligned}$$

[6] Note that the codomain of a substitution is a set of terms and sequence function symbols, not a set consisting of terms, sequences of terms, sequence function symbols, and sequences of sequence function symbols. For instance, $\mathcal{C}od(\{x \mapsto f(a), \overline{x} \mapsto \ulcorner a, a, b \urcorner, \overline{c} \mapsto \ulcorner \overline{c_1}, \overline{c_2} \urcorner\}) = \{f(a), a, b, \overline{c_1}, \overline{c_2}\}$.

Then the composition *of σ and ϑ, $\sigma\vartheta$, is the substitution obtained from the set*

$$\{ x_1 \mapsto t_1\vartheta, \dots, x_n \mapsto t_n\vartheta, \overline{x_1} \mapsto \ulcorner s^1_1\vartheta, \dots, s^1_{k_1}\vartheta\urcorner, \dots, \overline{x_m} \mapsto \ulcorner s^m_1\vartheta, \dots, s^m_{k_m}\vartheta\urcorner,$$
$$\overline{f_1} \mapsto \ulcorner \overline{f^1_1}\theta, \dots, \overline{f^1_{l_1}}\theta\urcorner, \dots, \overline{f_r} \mapsto \ulcorner \overline{f^r_1}\theta, \dots, \overline{f^r_{l_r}}\theta\urcorner,$$
$$y_1 \mapsto r_1, \dots, y_{n'} \mapsto r_{n'}, \overline{y_1} \mapsto \ulcorner q^1_1, \dots, q^1_{k'_1}\urcorner, \dots, \overline{y_{m'}} \mapsto \ulcorner q^{m'}_1, \dots, q^{m'}_{k'_{m'}}\urcorner,$$
$$\overline{g_1} \mapsto \ulcorner \overline{g^1_1}, \dots, \overline{g^1_{l'_1}}\urcorner, \dots, \overline{g_{r'}} \mapsto \ulcorner \overline{g^{r'}_1}, \dots, \overline{g^{r'}_{l'_{r'}}}\urcorner\}$$

by deleting

1. *all the bindings $x_i \mapsto t_i\vartheta$ $(1 \le i \le n)$ for which $x_i = t_i\vartheta$,*
2. *all the bindings $\overline{x_i} \mapsto \ulcorner s^i_1\vartheta, \dots, s^i_{k_i}\vartheta\urcorner$ $(1 \le i \le m)$ for which the sequence $s^i_1\vartheta, \dots, s^i_{k_i}\vartheta$ consists of a single term $\overline{x_i}$,*
3. *all the sequence function symbol bindings $\overline{f_i} \mapsto \ulcorner \overline{f^i_1}\vartheta, \dots, \overline{f^i_{l_i}}\vartheta\urcorner$ $(1 \le i \le r)$ such that the sequence $\overline{f^i_1}\vartheta, \dots, \overline{f^i_{l_r}}\vartheta$ consists of a single function symbol $\overline{f_i}$,*
4. *all the bindings $y_i \mapsto r_i$ $(1 \le i \le n')$ such that $y_i \in \{x_1, \dots, x_n\}$,*
5. *all the bindings $\overline{y_i} \mapsto \ulcorner q^i_1, \dots, q^i_{k'_i}\urcorner$ $(1 \le i \le m')$ with $\overline{y_i} \in \{\overline{x_1}, \dots, \overline{x_m}\}$,*
6. *all the sequence function symbol bindings $\overline{g_i} \mapsto \ulcorner \overline{g^i_1}, \dots, \overline{g^i_{l'_i}}\urcorner$ $(1 \le i \le r')$ such that $\overline{g_i} \in \{\overline{f_1}, \dots, \overline{f_r}\}$.*

Example 3. Let $\sigma = \{x \mapsto y, \overline{x} \mapsto \ulcorner \overline{y}, \overline{x}\urcorner, \overline{y} \mapsto \ulcorner f(a,b), y, \overline{g}(\overline{x})\urcorner, \overline{f} \mapsto \ulcorner \overline{g}, \overline{h}\urcorner\}$ and $\vartheta = \{y \mapsto x, \overline{y} \mapsto \overline{x}, \overline{x} \mapsto \ulcorner\urcorner, \overline{g} \mapsto \ulcorner \overline{g_1}, \overline{g_2}\urcorner\}$ be two substitutions. Then $\sigma\vartheta = \{y \mapsto x, \overline{y} \mapsto \ulcorner f(a,b), x, \overline{g_1}(), \overline{g_2}()\urcorner, \overline{f} \mapsto \ulcorner \overline{g_1}, \overline{g_2}, \overline{h}\urcorner, \overline{g} \mapsto \ulcorner \overline{g_1}, \overline{g_2}\urcorner\}$.

Definition 9. *A substitution σ is called* linearizing away *from a finite set of sequence function symbols $\mathcal{Q}$ iff the following three conditions hold: (1) $\mathcal{C}od(\sigma) \cap \mathcal{Q} = \emptyset$. (2) For all $\overline{f}, \overline{g} \in \mathcal{D}om(\sigma) \cap \mathcal{Q}$, if $\overline{f} \neq \overline{g}$, then $\{\overline{f}\sigma\} \cap \{\overline{g}\sigma\} = \emptyset$. (3) If $\overline{f} \mapsto \ulcorner \overline{g_1} \dots, \overline{g_n}\urcorner \in \sigma$ and $\overline{f} \in \mathcal{Q}$, then $\overline{g_i} \neq \overline{g_j}$ for all $1 \le i < j \le n$.*

Intuitively, a substitution linearizing away from $\mathcal{Q}$ either leaves a sequence function symbol in $\mathcal{Q}$ "unchanged", or "moves it away from" $\mathcal{Q}$, binding it with a sequence of distinct sequence function symbols that do not occur in $\mathcal{Q}$, and maps different sequence function symbols to disjoint sequences.

Let E be a set of equations over $\mathcal{F}$ and $\mathcal{V}$. By $\approx_E$ we denote the least congruence relation on $\mathcal{T}(\mathcal{F}, \mathcal{V})$ that is closed under substitution application and contains E. More precisely, $\approx_E$ contains E, satisfies reflexivity, symmetry, transitivity, congruence, and a special form of substitutivity: For all $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$, if $s \approx_E t$ and $s\sigma, t\sigma \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ for some σ, then $s\sigma \approx_E t\sigma$. Substitutivity in this form requires that $s\sigma$ and $t\sigma$ must be single terms, not arbitrary sequences of terms. The set $\approx_E$ is called the *equational theory* defined by E. In the sequel, we will also call the set E an equational theory, or E-theory. The *signature* of E is the set $\mathcal{S}ig(E) = \mathcal{IF}un(E) \cup \mathcal{SF}un(E)$. Solving equations in an E-theory is called E*-unification*. The fact that the equation $s \approx t$ has to be solved in an E-theory is written as $s \approx^?_E t$.

Definition 10. *Let E be an equational theory with $\mathcal{S}ig(E) \subseteq \mathcal{F}$. An* E-unification problem *over $\mathcal{F}$ is a finite multiset $\Gamma = \{s_1 \approx^?_E t_1, \ldots, s_n \approx^?_E t_n\}$ of equations over $\mathcal{F}$ and $\mathcal{V}$. An* E-unifier *of Γ is a substitution σ such that σ is linearizing away from $\mathcal{SF}un(\Gamma)$ and for all $1 \leq i \leq n$, $s_i\sigma \approx_E t_i\sigma$. The set of all E-unifiers of Γ is denoted by $\mathcal{U}_E(\Gamma)$, and Γ is* E-unifiable *iff $\mathcal{U}_E(\Gamma) \neq \emptyset$.*

If $\{s_1 \approx^?_E t_1, \ldots, s_n \approx^?_E t_n\}$ is a unification problem, then $s_i, t_i \in \mathcal{T}_{\mathrm{Ind}}(\mathcal{F}, \mathcal{V})$ for all $1 \leq i \leq n$.

Example 4. Let $\Gamma = \{f(\overline{g}(\overline{x}, \overline{y}, a)) \approx^?_\emptyset f(\overline{g}(\overline{c}, b, x))\}$. Then $\{\overline{x} \mapsto \overline{c_1}, \overline{y} \mapsto \ulcorner \overline{c_2}, b \urcorner, x \mapsto a, \overline{c} \mapsto \ulcorner \overline{c_1}, \overline{c_2} \urcorner\} \in \mathcal{U}_\emptyset(\Gamma)$.

Let $\Gamma = \{f(\overline{g}(\overline{x}, \overline{y}, a)) \approx^?_\emptyset f(\overline{h}(\overline{c}, x))\}$. Then $\mathcal{U}_\emptyset(\Gamma) = \emptyset$. If we did not require the E-unifiers of a unification problem to be linearizing away from the sequence function symbol set of the problem, then Γ would have $\emptyset$-unifiers, e.g., $\{\overline{x} \mapsto \overline{c_1}, \overline{y} \mapsto \ulcorner \overline{c_2}, b \urcorner, x \mapsto a, \overline{g} \mapsto \overline{h}, \overline{c} \mapsto \ulcorner \overline{c_1}, \overline{c_2} \urcorner\}$ would be one of them.

In the sequel, if not otherwise stated, E stands for an equational theory, $\mathcal{X}$ for a finite set of variables, and $\mathcal{Q}$ for a finite set of sequence function symbols.

Definition 11. *A substitution σ is called* erasing *on $\mathcal{X}$ modulo E iff either $f(v)\sigma \approx_E f()$ for some $f \in \mathcal{S}ig(E)$ and $v \in \mathcal{X}$, or $\overline{x} \mapsto \ulcorner\urcorner \in \sigma$ for some $\overline{x} \in \mathcal{X}$. We call σ* non-erasing *on $\mathcal{X}$ modulo E iff σ is not erasing on $\mathcal{X}$ modulo E.*

Example 5. Any substitution containing $\overline{x} \mapsto \ulcorner\urcorner$ is erasing modulo $E = \emptyset$ on any $\mathcal{X}$ that contains $\overline{x}$.

Let $E = \{f(\overline{x}, f(\overline{y}), \overline{z}) \approx f(\overline{x}, \overline{y}, \overline{z})\}$ and $\mathcal{X} = \{x, \overline{x}\}$. Then any substitution that contains $x \mapsto f()$, or $\overline{x} \mapsto \ulcorner\urcorner$, or $\overline{x} \mapsto \ulcorner t_1, \ldots, t_n \urcorner$ with $n \geq 1$ and $t_1 = \cdots = t_n = f()$, is erasing on $\mathcal{X}$ modulo E. For instance, the substitutions $\{x \mapsto f()\}$, $\{\overline{x} \mapsto f()\}$, $\{\overline{x} \mapsto \ulcorner\urcorner\}$, $\{\overline{x} \mapsto \ulcorner f(), f(), f(), f() \urcorner\}$ are erasing on $\mathcal{X}$ modulo E.

Definition 12. *A substitution σ* agrees *with a substitution ϑ on $\mathcal{X}$ and $\mathcal{Q}$ modulo E, denoted $\sigma =^{\mathcal{X},\mathcal{Q}}_E \vartheta$, iff (1) for all $x \in \mathcal{X}$, $x\sigma \approx_E x\vartheta$; (2) for all $\overline{f} \in \mathcal{Q}$, $\overline{f}\sigma = \overline{f}\vartheta$; (3) for all $\overline{x} \in \mathcal{X}$, there exist $t_1, \ldots, t_n, s_1, \ldots, s_n \in \mathcal{T}(\mathcal{F}, \mathcal{V})$, $n \geq 0$, such that $\overline{x}\sigma = \ulcorner t_1, \ldots, t_n \urcorner$, $\overline{x}\vartheta = \ulcorner s_1, \ldots, s_n \urcorner$ and $t_i \approx_E s_i$ for each $1 \leq i \leq n$.*

Example 6. Let $\sigma = \{\overline{x} \mapsto \overline{a}\}$, $\vartheta = \{\overline{x} \mapsto \ulcorner \overline{b}, \overline{c} \urcorner, \overline{a} \mapsto \ulcorner \overline{b}, \overline{c} \urcorner\}$, and $\varphi = \{\overline{x} \mapsto \ulcorner \overline{b}, \overline{c} \urcorner, \overline{a} \mapsto \ulcorner \overline{b}, \overline{c} \urcorner\}$. Let also $\mathcal{X} = \{\overline{x}\}$, $\mathcal{Q} = \{\overline{a}\}$, and $E = \emptyset$. Then $\sigma\varphi =^{\mathcal{X},\mathcal{Q}}_E \vartheta$.

Definition 13. *A substitution σ is* more general *(resp.* strongly more general*) than ϑ on $\mathcal{X}$ and $\mathcal{Q}$ modulo E, denoted $\sigma \lessdot^{\mathcal{X},\mathcal{Q}}_E \vartheta$ (resp. $\sigma \preceq^{\mathcal{X},\mathcal{Q}}_E \vartheta$), iff $\sigma\varphi =^{\mathcal{X},\mathcal{Q}}_E \vartheta$ for some substitution (resp. substitution non-erasing on $\mathcal{X}$ modulo E) φ.*

Example 7. Let $\sigma = \{\overline{x} \mapsto \overline{y}\}$, $\vartheta = \{\overline{x} \mapsto \ulcorner a, b \urcorner, \overline{y} \mapsto \ulcorner a, b \urcorner\}$, $\eta = \{\overline{x} \mapsto \ulcorner\urcorner, \overline{y} \mapsto \ulcorner\urcorner\}$, $\mathcal{X} = \{\overline{x}, \overline{y}\}$, $\mathcal{Q} = \emptyset$, $E = \emptyset$. Then $\sigma \lessdot^{\mathcal{X},\mathcal{Q}}_E \vartheta$, $\sigma \preceq^{\mathcal{X},\mathcal{Q}}_E \vartheta$, $\sigma \lessdot^{\mathcal{X},\mathcal{Q}}_E \eta$, $\sigma \not\preceq^{\mathcal{X},\mathcal{Q}}_E \eta$.

A substitution ϑ is an *E-instance* (resp. *strong E-instance*) of σ on $\mathcal{X}$ and $\mathcal{Q}$ iff $\sigma \lessdot^{\mathcal{X},\mathcal{Q}}_E \vartheta$ (resp. $\sigma \preceq^{\mathcal{X},\mathcal{Q}}_E \vartheta$). The equivalence associated with $\lessdot^{\mathcal{X},\mathcal{Q}}_E$ (resp. with $\preceq^{\mathcal{X},\mathcal{Q}}_E$) is denoted by $\doteq^{\mathcal{X},\mathcal{Q}}_E$ (resp. by $\dot{\approx}^{\mathcal{X},\mathcal{Q}}_E$). The strict part of $\lessdot^{\mathcal{X},\mathcal{Q}}_E$ (resp. $\preceq^{\mathcal{X},\mathcal{Q}}_E$) is denoted by $<^{\mathcal{X},\mathcal{Q}}_E$ (resp. $\prec^{\mathcal{X},\mathcal{Q}}_E$). Definition 13 implies $\preceq^{\mathcal{X},\mathcal{Q}}_E \subseteq \lessdot^{\mathcal{X},\mathcal{Q}}_E$.

Definition 14. *A set of substitutions $\mathcal{S}$ is called* minimal *(resp.* almost minimal*) with respect to $\mathcal{X}$ and $\mathcal{Q}$ modulo E iff two distinct elements of $\mathcal{S}$ are incomparable with respect to $\leq_E^{\mathcal{X},\mathcal{Q}}$ (resp. $\preceq_E^{\mathcal{X},\mathcal{Q}}$).*

Minimality implies almost minimality, but not vice versa: A counterexample is the set $\{\sigma, \eta\}$ from Example 7.

Definition 15. *A* complete set of E-unifiers *of an E-unification problem Γ is a set $\mathcal{S}$ of substitutions such that (1) $\mathcal{S} \subseteq \mathcal{U}_E(\Gamma)$, and (2) for each $\vartheta \in \mathcal{U}_E(\Gamma)$ there exists $\sigma \in \mathcal{S}$ such that $\sigma \leq_E^{\mathcal{X},\mathcal{Q}} \vartheta$, where $\mathcal{X} = \mathcal{V}ar(\Gamma)$ and $\mathcal{Q} = \mathcal{SF}un(\Gamma)$.*

The set $\mathcal{S}$ is a minimal *(resp.* almost minimal*)* complete set of E-unifiers *of Γ, denoted $mcu_E(\Gamma)$ (resp. $amcu_E(\Gamma)$) iff it is a complete set that is minimal (resp. almost minimal) with respect to $\mathcal{X}$ and $\mathcal{Q}$ modulo E.*

Proposition 1. *An E-unification problem Γ has an almost minimal complete set of E-unifiers iff it has a minimal complete set of E-unifiers.*

If Γ is not E-unifiable, then $mcu_E(\Gamma) = amcu_E(\Gamma) = \emptyset$. A minimal (resp. almost minimal) complete set of E-unifiers of Γ, if it exists, is unique up to the equivalence $\doteq_E^{\mathcal{X},Q}$ (resp. $\approx_E^{\mathcal{X},Q}$), where $\mathcal{X} = \mathcal{V}ar(\Gamma)$ and $\mathcal{Q} = \mathcal{SF}un(\Gamma)$.

Example 8. 1. $\Gamma = \{f(\overline{x}) \approx_\emptyset^? f(\overline{y})\}$. Then $mcu_\emptyset(\Gamma) = \{\{\overline{x} \mapsto \overline{y}\}\}$, $amcu_\emptyset(\Gamma) = \{\{\overline{x} \mapsto \overline{y}\}, \{\overline{x} \mapsto \ulcorner\urcorner, \overline{y} \mapsto \ulcorner\urcorner\}\}$.
2. $\Gamma = \{f(\overline{x}, x, \overline{y}) \approx_\emptyset^? f(f(\overline{x}), x, a, b)\}$. Then $mcu_\emptyset(\Gamma) = amcu_\emptyset(\Gamma) = \{\{x \mapsto f(), \overline{x} \mapsto \ulcorner\urcorner, \overline{y} \mapsto \ulcorner f(), a, b\urcorner\}\}$.
3. $\Gamma = \{f(a, \overline{x}) \approx_\emptyset^? f(\overline{x}, a)\}$. Then $mcu_\emptyset(\Gamma) = amcu_\emptyset(\Gamma) = \{\{\overline{x} \mapsto \ulcorner\urcorner\}, \{\overline{x} \mapsto a\}, \{\overline{x} \mapsto \ulcorner a, a\urcorner\}, \ldots\}$.
4. $\Gamma = \{f(\overline{x}, \overline{y}, x) \approx_\emptyset^? f(\overline{c}, a)\}$. Then $mcu_\emptyset(\Gamma) = amcu_\emptyset(\Gamma) = \{\{\overline{x} \mapsto \ulcorner\urcorner, \overline{y} \mapsto \overline{c}, x \mapsto a\}, \{\overline{x} \mapsto \overline{c}, \overline{y} \mapsto \ulcorner\urcorner, x \mapsto a\}, \{\overline{x} \mapsto \overline{c_1}, \overline{y} \mapsto \overline{c_2}, x \mapsto a, \overline{c} \mapsto \ulcorner \overline{c_1}, \overline{c_2}\urcorner\}\}$.

Definition 16. *A set of substitutions $\mathcal{S}$ is* disjoint *(resp.* almost disjoint*) wrt $\mathcal{X}$ and $\mathcal{Q}$ modulo E iff two distinct elements in $\mathcal{S}$ have no common E-instance (resp. strong E-instance) on $\mathcal{X}$ and $\mathcal{Q}$, i.e., for all $\sigma, \vartheta \in \mathcal{S}$, if there exists φ such that $\sigma \leq_E^{\mathcal{X},\mathcal{Q}} \varphi$ (resp. $\sigma \preceq_E^{\mathcal{X},\mathcal{Q}} \varphi$) and $\vartheta \leq_E^{\mathcal{X},\mathcal{Q}} \varphi$ (resp. $\vartheta \preceq_E^{\mathcal{X},\mathcal{Q}} \varphi$), then $\sigma = \vartheta$.*

Disjointness implies almost disjointness, but not vice versa: Consider again the set $\{\sigma, \eta\}$ in Example 7.

Proposition 2. *If a set of substitutions $\mathcal{S}$ is disjoint (almost disjoint) wrt $\mathcal{X}$ and $\mathcal{Q}$ modulo E, then it is minimal (almost minimal) wrt $\mathcal{X}$ and $\mathcal{Q}$ modulo E.*

However, almost disjointness does not imply minimality: Again, take the set $\{\sigma, \eta\}$ in Example 7. On the other hand, minimality does not imply almost disjointness: Let $\sigma = \{x \mapsto f(a, y)\}$, $\vartheta = \{x \mapsto f(y, b)\}$, $\mathcal{X} = \{x\}$, $\mathcal{Q} = \emptyset$, and $E = \emptyset$. Then $\{\sigma, \vartheta\}$ is minimal but not almost disjoint with respect to $\mathcal{X}$ and $\mathcal{Q}$ modulo E, because $\sigma \preceq_E^{\mathcal{X},\mathcal{Q}} \varphi$ and $\vartheta \preceq_E^{\mathcal{X},\mathcal{Q}} \varphi$, with $\varphi = \{x \mapsto f(a, b)\}$, but $\sigma \neq \vartheta$. The same example can be used to show that almost minimality does not imply almost disjointness either. From these observations we can also conclude that neither minimality nor almost minimality imply disjointness.

The equational theory $E = \emptyset$ is called the *free theory with sequence variables and sequence function symbols.* Unification in the free theory is called the *syntactic sequence unification.* The theory $E = \{f(\overline{x}, f(\overline{y}), \overline{z}) \approx f(\overline{x}, \overline{y}, \overline{z})\}$ that we first encountered in Example 5 is called *the flat theory*, where f is the flat flexible arity individual function symbol.

General syntactic sequence unification is decidable. We just sketch the proof idea here, details can be found in [14]. We show decidability in three steps: First, we reduce the general syntactic sequence unification problem by a unifiability preserving transformation to a unification problem containing no sequence function symbols. Second, applying yet another unifiability preserving transformation we get rid of all free flexible arity (individual) function symbols, obtaining a unification problem whose signature consists of fixed arity individual function symbols and one flat flexible arity individual function symbol. Finally, we show decidability of the reduced problem using Baader-Schulz combination method [2] and decidability results for word equations and Robinson unification, both with linear constant restrictions.

3 Relation with Order-Sorted Higher-Order Unification

In this section we relate syntactic sequence unification with order-sorted unification. Unification in order-sorted theories has been studied by many authors (see, e.g., [9–11, 16, 19, 24, 22]). Syntactic sequence unification can be considered as a special case of order-sorted higher-order E-unification. Here we show the corresponding encoding. We consider simply typed λ-calculus with the types i and o. The set of base sorts consists of $\mathtt{ind}, \mathtt{seq}, \mathtt{seqc}, \mathtt{o}$ such that the type of $\mathtt{o}$ is o and the type of the other sorts is i. We will treat individual and sequence variables as first order variables, sequence functions as second order variables and define a context Γ such that $\Gamma(x) = \mathtt{ind}$ for all $x \in \mathcal{V}_{\mathrm{Ind}}$, $\Gamma(\overline{x}) = \mathtt{seq}$ for all $\overline{x} \in \mathcal{V}_{\mathrm{Seq}}$, $\Gamma(\overline{f}) = \mathtt{seq} \to \mathtt{seqc}$ for each $\overline{f} \in \mathcal{F}^{\mathrm{Flex}}_{\mathrm{Seq}}$, and $\Gamma(\overline{f}) = \underbrace{\mathtt{ind} \to \cdots \to \mathtt{ind}}_{n \text{ times}} \to \mathtt{seqc}$ for each $\overline{f} \in \mathcal{F}^{\mathrm{Fix}}_{\mathrm{Seq}}$ with $\mathcal{A}r(f) = n$. Individual function symbols are treated as constants. We assign to each $f \in \mathcal{F}^{\mathrm{Flex}}_{\mathrm{Ind}}$ a functional sort $\mathtt{seq} \to \mathtt{ind}$ and to each $f \in \mathcal{F}^{\mathrm{Fix}}_{\mathrm{Ind}}$ with $\mathcal{A}r(f) = n$ a functional sort $\underbrace{\mathtt{ind} \to \cdots \to \mathtt{ind}}_{n \text{ times}} \to \mathtt{ind}$. We assume equality constants $\approx_{\mathtt{s}}$ for every sort $\mathtt{s}$. In addition, we have two function symbols: binary $\ulcorner\urcorner$ of the sort $\mathtt{seq} \to \mathtt{seq} \to \mathtt{seq}$ and a constant $[]$ of the sort $\mathtt{seq}$. Sorts are partially ordered as $[\mathtt{ind} \leq \mathtt{seqc}]$ and $[\mathtt{seqc} \leq \mathtt{seq}]$. The equational theory is an AU-theory, asserting associativity of $\ulcorner\urcorner$ with $[]$ as left and right unit. We consider unification problems for terms of the sort $\mathtt{ind}$ where terms are in $\beta\eta$-normal form containing no bound variables, and terms whose head is $\ulcorner\urcorner$ are flattened. For a given unification problem Γ in this theory, we are looking for unifiers that obey the following restrictions: If a unifier σ binds a second order variable $\overline{f}$ of the sort $\mathtt{seq} \to \mathtt{seqc}$, then $\overline{f}\sigma = \lambda\overline{x}.\ulcorner\overline{g_1}(\overline{x}), \ldots, \overline{g_m}(\overline{x})\urcorner$

and if σ binds a second order variable $\overline{f}$ of the sort $\underbrace{\mathtt{ind} \to \cdots \to \mathtt{ind}}_{n \text{ times}} \to \mathtt{seqc}$, then $\overline{f}\sigma = \lambda x_1 \dots x_n . \ulcorner \overline{g_1}(x_1, \dots, x_n), \dots, \overline{g_m}(x_1, \dots, x_n) \urcorner$, where $m > 1$ and $\overline{g_1}, \dots, \overline{g_m}$ are fresh variables of the same sort as f.

Hence, syntactic sequence unification can be considered as order-sorted second-order AU-unification with additional restrictions. Order-sorted higher-order syntactic unification was investigated in [16, 9–11, 19], but we are not aware of any work done on order-sorted higher-order equational unification.

4 Unification Procedure

In the sequel we assume that Γ, maybe with indices, and Γ' denote syntactic sequence unification problems. A *system* is either the symbol $\perp$ (representing failure), or a pair $\langle \Gamma; \sigma \rangle$. The inference system $\mathfrak{U}$ consists of the transformation rules on systems listed below. The function symbol $g \in \mathcal{F}_{\mathrm{Ind}}^{\mathrm{Flex}}$ in the rule PD2 is new. In the Splitting rule $\overline{f_1}$ and $\overline{f_2}$ are new sequence function symbols of the same arity as $\overline{f}$ in the same rule. We assume that the indices $n, m, k, l \geq 0$.

Projection (P):

$$\langle \Gamma;\ \sigma \rangle \Longrightarrow \langle \Gamma\vartheta;\ \sigma\vartheta \rangle, \text{ where } \vartheta \neq \varepsilon,\ \mathcal{D}om(\vartheta) \subseteq \mathcal{SV}ar(\Gamma) \text{ and } \mathcal{C}od(\vartheta) = \emptyset.$$

Trivial (T):

$$\langle \{s \approx_{\emptyset}^{?} s\} \cup \Gamma';\ \sigma \rangle \Longrightarrow \langle \Gamma';\ \sigma \rangle.$$

Orient 1 (O1):

$$\langle \{s \approx_{\emptyset}^{?} x\} \cup \Gamma'; \sigma \rangle \Longrightarrow \langle \{x \approx_{\emptyset}^{?} s\} \cup \Gamma'; \sigma \rangle, \text{ if } s \notin \mathcal{V}_{\mathrm{Ind}}.$$

Orient 2 (O2):

$$\langle \{f(s, s_1, \dots, s_n) \approx_{\emptyset}^{?} f(\overline{x}, t_1, \dots, t_m)\} \cup \Gamma';\ \sigma \rangle \Longrightarrow$$
$$\langle \{f(\overline{x}, t_1, \dots, t_m) \approx_{\emptyset}^{?} f(s, s_1, \dots, s_n)\} \cup \Gamma';\ \sigma \rangle, \qquad \text{if } s \notin \mathcal{V}_{\mathrm{Seq}}.$$

Solve (S):

$$\langle \{x \approx_{\emptyset}^{?} t\} \cup \Gamma'; \sigma \rangle \Longrightarrow \langle \Gamma'\vartheta;\ \sigma\vartheta \rangle, \text{ if } x \notin \mathcal{IV}ar(t) \text{ and } \vartheta = \{x \mapsto t\}.$$

Total Decomposition (TD):

$$\langle \{f(s_1, \dots, s_n) \approx_{\emptyset}^{?} f(t_1, \dots, t_n)\} \cup \Gamma';\ \sigma \rangle \Longrightarrow$$
$$\langle \{s_1 \approx_{\emptyset}^{?} t_1, \dots, s_n \approx_{\emptyset}^{?} t_n\} \cup \Gamma';\ \sigma \rangle$$

if $f(s_1, \dots, s_n) \neq f(t_1, \dots, t_n)$, and $s_i, t_i \in \mathcal{T}_{\mathrm{Ind}}(\mathcal{F}, \mathcal{V})$ for all $1 \leq i \leq n$.

Partial Decomposition 1 (PD1):

$$\langle \{f(s_1, \dots, s_n) \approx_{\emptyset}^{?} f(t_1, \dots, t_m)\} \cup \Gamma';\ \sigma \rangle \Longrightarrow$$
$$\langle \{s_1 \approx_{\emptyset}^{?} t_1, \dots, s_{k-1} \approx_{\emptyset}^{?} t_{k-1}, f(s_k, \dots, s_n) \approx_{\emptyset}^{?} f(t_k, \dots, t_m)\} \cup \Gamma';\ \sigma \rangle$$

if $f(s_1, \dots, s_n) \neq f(t_1, \dots, t_m)$, for some $1 < k \leq min(n, m)$,
$s_k \in \mathcal{T}_{\mathrm{Seq}}(\mathcal{F}, \mathcal{V})$ or $t_k \in \mathcal{T}_{\mathrm{Seq}}(\mathcal{F}, \mathcal{V})$, and $s_i, t_i \in \mathcal{T}_{\mathrm{Ind}}(\mathcal{F}, \mathcal{V})$ for all $1 \leq i < k$.

Partial Decomposition 2 (PD2):

$$\langle \{f(\overline{f}(r_1, \dots, r_k), s_1, \dots, s_n) \approx_{\emptyset}^{?} f(\overline{f}(q_1, \dots, q_l), t_1, \dots, t_m)\} \cup \Gamma';\ \sigma \rangle \Longrightarrow$$
$$\langle \{g(r_1, \dots, r_k) \approx_{\emptyset}^{?} g(q_1, \dots, q_l), f(s_1, \dots, s_n) \approx_{\emptyset}^{?} f(t_1, \dots, t_m)\} \cup \Gamma';\ \sigma \rangle.$$

if $f(\overline{f}(r_1, \dots, r_k), s_1, \dots, s_n) \neq f(\overline{f}(q_1, \dots, q_l), t_1, \dots, t_m)$.

Sequence Variable Elimination 1 (SVE1):

$$\langle\{f(\overline{x}, s_1, \ldots, s_n) \approx^?_\emptyset f(\overline{x}, t_1, \ldots, t_m)\} \cup \Gamma';\ \sigma\rangle \Longrightarrow$$
$$\langle\{f(s_1, \ldots, s_n) \approx^?_\emptyset f(t_1, \ldots, t_m)\} \cup \Gamma';\ \sigma\rangle$$

if $f(\overline{x}, s_1, \ldots, s_n) \neq f(\overline{x}, t_1, \ldots, t_m)$.

Sequence Variable Elimination 2 (SVE2):

$$\langle\{f(\overline{x}, s_1, \ldots, s_n) \approx^?_\emptyset f(t, t_1, \ldots, t_m)\} \cup \Gamma';\ \sigma\rangle \Longrightarrow$$
$$\langle\{f(s_1, \ldots, s_n)\vartheta \approx^?_\emptyset f(t_1, \ldots, t_m)\vartheta\} \cup \Gamma'\vartheta;\ \sigma\vartheta\rangle$$

if $\overline{x} \notin \mathcal{SV}ar(t)$ and $\vartheta = \{\overline{x} \mapsto t\}$.

Widening 1 (W1):

$$\langle\{f(\overline{x}, s_1, \ldots, s_n) \approx^?_\emptyset f(t, t_1, \ldots, t_m)\} \cup \Gamma';\ \sigma\rangle \Longrightarrow$$
$$\langle\{f(\overline{x}, s_1\vartheta, \ldots, s_n\vartheta) \approx^?_\emptyset f(t_1\vartheta, \ldots, t_m\vartheta)\} \cup \Gamma'\vartheta;\ \sigma\vartheta\rangle$$

if $\overline{x} \notin \mathcal{SV}ar(t)$ and $\vartheta = \{\overline{x} \mapsto \ulcorner t, \overline{x}\urcorner\}$.

Widening 2 (W2):

$$\langle\{f(\overline{x}, s_1, \ldots, s_n) \approx^?_\emptyset f(\overline{y}, t_1, \ldots, t_m)\} \cup \Gamma';\ \sigma\rangle \Longrightarrow$$
$$\langle\{f(s_1\vartheta, \ldots, s_n\vartheta) \approx^?_\emptyset f(\overline{y}, t_1\vartheta, \ldots, t_m\vartheta)\} \cup \Gamma'\vartheta;\ \sigma\vartheta\rangle$$

where $\vartheta = \{\overline{y} \mapsto \ulcorner \overline{x}, \overline{y}\urcorner\}$.

Splitting (Sp):

$$\langle\{f(\overline{x}, s_1, \ldots, s_n) \approx^?_\emptyset f(\overline{f}(r_1, \ldots, r_k), t_1, \ldots, t_m)\} \cup \Gamma';\ \sigma\rangle \Longrightarrow$$
$$\langle\{f(s_1, \ldots, s_n)\vartheta \approx^?_\emptyset f(\overline{f_2}(r_1, \ldots, r_k), t_1, \ldots, t_m)\vartheta\} \cup \Gamma'\vartheta;\ \sigma\vartheta\rangle$$

if $\overline{x} \notin \mathcal{SV}ar(\overline{f}(r_1, \ldots, r_k))$ and $\vartheta = \{\overline{x} \mapsto \overline{f_1}(r_1, \ldots, r_k)\}\{\overline{f} \mapsto \ulcorner \overline{f_1}, \overline{f_2}\urcorner\}$.

We may use the rule name abbreviations as subscripts, e.g., $\langle\Gamma_1;\ \sigma_1\rangle \Longrightarrow_{\mathsf{P}} \langle\Gamma_2;\ \sigma_2\rangle$ for Projection. We may also write $\langle\Gamma_1;\ \sigma_1\rangle \Longrightarrow_{\mathsf{BT}} \langle\Gamma_2;\ \sigma_2\rangle$ to indicate that $\langle\Gamma_1;\ \sigma_1\rangle$ was transformed to $\langle\Gamma_2;\ \sigma_2\rangle$ by some *basic transformation* (i.e., non-projection) rule. **P**, **SVE2**, **W1**, **W2**, and **Sp** are non-deterministic rules.

A *derivation* is a sequence $\langle\Gamma_1;\sigma_1\rangle \Longrightarrow \langle\Gamma_2;\sigma_2\rangle \Longrightarrow \cdots$ of system transformations. A derivation is *fair* if any transformation rule which is continuously enabled is eventually applied. Any finite fair derivation $S_1 \Longrightarrow S_2 \Longrightarrow \cdots \Longrightarrow S_n$ is maximal, i.e., no further transformation rule can be applied on S_n.

Definition 17. *A* syntactic sequence unification procedure *is any program that takes a system $\langle\Gamma;\ \varepsilon\rangle$ as an input and uses the rules in $\mathfrak{U}$ to generate a tree of fair derivations, called the* unification tree for Γ, $\mathcal{UT}(\Gamma)$, *in the following way:*

1. *The root of the tree is labeled with $\langle\Gamma;\ \varepsilon\rangle$;*
2. *Each branch of the tree is a fair derivation either of the form $\langle\Gamma;\ \varepsilon\rangle \Longrightarrow_{\mathsf{P}} \langle\Gamma_1;\ \sigma_1\rangle \Longrightarrow_{\mathsf{BT}} \langle\Gamma_2;\ \sigma_2\rangle \Longrightarrow_{\mathsf{BT}} \cdots$ or $\langle\Gamma;\ \varepsilon\rangle \Longrightarrow_{\mathsf{BT}} \langle\Gamma_1;\ \sigma_1\rangle \Longrightarrow_{\mathsf{BT}} \langle\Gamma_2;\ \sigma_2\rangle \Longrightarrow_{\mathsf{BT}} \cdots$. The nodes in the tree are systems.*
3. *If several transformation rules, or different instances of the same transformation rule are applicable to a node in the tree, they are applied concurrently.*
4. *The decision procedure is applied to the root and to each node generated by a non-deterministic transformation rule, to decide whether the node contains a solvable unification problem. If the unification problem Δ in a node $\langle\Delta;\ \delta\rangle$ is unsolvable, then the branch is extended by $\langle\Delta;\ \delta\rangle \Longrightarrow_{\mathsf{DP}} \bot$.*

The leaves of $\mathcal{UT}(\Gamma)$ are labeled either with the systems of the form $\langle\emptyset;\ \sigma\rangle$ or with $\bot$. The branches of $\mathcal{UT}(\Gamma)$ that end with $\langle\emptyset;\ \sigma\rangle$ are called *successful branches*, and those with the leaves $\bot$ are *failed branches*. We denote by $\mathcal{S}ol_\emptyset(\Gamma)$ the solution set of Γ, i.e., the set of all σ-s such that $\langle\emptyset;\ \sigma\rangle$ is a leaf of $\mathcal{UT}(\Gamma)$.

The calculus is sound and complete:

Theorem 1 (Soundness). *If* $\langle\Gamma;\ \varepsilon\rangle \Longrightarrow^+ \langle\emptyset;\ \vartheta\rangle$, *then* $\vartheta \in \mathcal{U}_\emptyset(\Gamma)$.

Theorem 2 (Completeness). $\mathcal{S}ol_\emptyset(\Gamma)$ *is a complete set of unifiers of* Γ.

The set $\mathcal{S}ol_\emptyset(\Gamma)$, in general, is not minimal with respect to $\mathcal{V}ar(\Gamma)$ and $\mathcal{SF}un(\Gamma)$ modulo the free theory. Just consider $\Gamma = \{f(\overline{x}) \approx^?_\emptyset f(\overline{y})\}$, then $\mathcal{S}ol_\emptyset(\Gamma) = \{\{\overline{x} \mapsto \overline{y}\},\ \{\overline{x} \mapsto \ulcorner\urcorner, \overline{y} \mapsto \ulcorner\urcorner\}\}$. However, it can be shown that $\mathcal{S}ol_\emptyset(\Gamma)$ is almost minimal[7], that implies the main result of this section:

Theorem 3 (Main Theorem). $\mathcal{S}ol_\emptyset(\Gamma) = amcu_\emptyset(\Gamma)$.

5 Implementation

We implemented the procedure (without the decision algorithm) in Mathematica [25] using the rule-based programming system ρLog [15]. ρLog provides full support for programming with primitive operators for defining elementary rules and for computing with unions, compositions, reflexive-transitive closures, and normal forms of transformation rules. Rules are specifications of partially defined and possibly non-deterministic computations which describe the calculation of a new object from another object described by a pattern. Strategies are built with a minimal number of rule combinators: composition, alternatives, repetition, normalization, and first commitment. A version of applicative higher-order matching algorithm with sequence variables is built-in. In such a language it is pretty straightforward to implement calculi like the unification procedure described in this paper. For instance, the **SVE2** and **O2** rules can be implemented as a single ρLog rule "O2SVE2" as follows:

```
DeclareRule[ ⟨{(f_[x_?SVarQ, s___] ≈ f_[r_, t___]) |
             (f_[r_/; Not[SVarQ[r]], t___] ≈ f_[x_?SVarQ, y___]), rest___}, σ_⟩ →"O2SVE2"
             ⟨{f[s] ≈ f[t], rest}/. x → r, σ ∘ {x → r}⟩];
```

The basic transformation rules of the procedure are combined in one rule called "BT" using the combinator for alternatives, that (on backtracking) computes the union of the results of each rule. Instead of costly decision algorithm we implemented simpler rules that provide sufficient conditions to detect failure. These rules are combined with each other by the combinator for alternatives, and the obtained rule is combined with the "BT" by the first commitment combinator, getting a rule called "Failure+BT". Then the procedure itself is nothing else than the strategy that composes the projection rule with the NF["Failure+BT"] rule,

[7] In fact, $\mathcal{S}ol_\emptyset(\Gamma)$ has a stronger property: almost disjointness.

where NF is the rule combinator for the normal form. Then ρLog performs depth-first search with bounded depth and backtracking, thus generating a finite subset of the set of solutions.

6 Conclusions

We developed a complete almost minimal unification procedure for general syntactic unification with sequence variables and sequence functions.

Syntactic sequence unification is decidable but infinitary. Under certain restrictions the unification problems have at most finitely many solutions: sequence variables in the last argument positions, unification problems with at least one ground side (matching as a particular instance), all sequence variables on the top level with maximum one occurrence. It would be interesting to identify more cases with finite or finitely representable solution sets.

References

1. H. Abdulrab and J.-P. Pécuchet. Solving word equations. *J. Symbolic Computation*, 8(5):499–522, 1990.
2. F. Baader and K. U. Schulz. Unification in the union of disjoint equational theories: Combining decision procedures. *J. Symbolic Computation*, 21(2):211–244, 1996.
3. F. Baader and W. Snyder. Unification theory. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 8, pages 445–532. Elsevier Science, 2001.
4. W. W. Bledsoe and Guohui Feng. Set-Var. *J. Automated Reasoning*, 11(3):293–314, 1993.
5. H. Boley. *A Tight, Practical Integration of Relations and Functions*, volume 1712 of *LNAI*. Springer, 1999.
6. B. Buchberger, C. Dupré, T. Jebelean, F. Kriftner, K. Nakagawa, D. Vasaru, and W. Windsteiger. The Theorema project: A progress report. In M. Kerber and M. Kohlhase, editors, *Proc. of Calculemus'2000 Conference*, pages 98–113, St. Andrews, UK, 6–7 August 2000.
7. M. L. Ginsberg. User's guide to the MVL system. Technical report, Stanford University, Stanford, California, US, 1989.
8. J. Jaffar. Minimal and complete word unification. *J. ACM*, 37(1):47–85, 1990.
9. P. Johann. A combinator-based order-sorted higher-order unification algorithm. Technical Report SR-93-16, Universität des Saarlandes. Saarbrücken, Germany, 1993.
10. P. Johann and M. Kohlhase. Unification in an extensional lambda calculus with ordered function sorts and constant overloading. In *Proc. of the 12th Int. Conference on Automated Deduction, CADE'94*, volume 814 of *LNAI*, pages 620–634, Nancy, France, 1994. Springer.
11. M. Kohlhase. A mechanization of sorted higher-order logic based on the resolution principle. PhD Thesis. Universität des Saarlandes. Saarbrücken, Germany, 1994.
12. T. Kutsia. Solving and proving in equational theories with sequence variables and flexible arity symbols. Technical Report 02-31, RISC, Johannes Kepler University, Linz, Austria, 2002.

13. T. Kutsia. Equational prover of THEOREMA. In R. Nieuwenhuis, editor, *Proc. of the 14th Int. Conference on Rewriting Techniques and Applications, RTA'03*, volume 2706 of *LNCS*, pages 367–379, Valencia, Spain, 9–11 June 2003. Springer.
14. T. Kutsia. Solving equations involving sequence variables and sequence functions. Technical report, RISC, Johannes Kepler University, Linz, Austria, 2004. Available from: ftp://ftp.risc.uni-linz.ac.at/pub/people/tkutsia/papers/sequnif.ps.
15. M. Marin and T. Kutsia. On the implementation of a rule-based programming system and some of its applications. In B. Konev and R. Schmidt, editors, *Proc. of the 4th Int. Workshop on the Implementation of Logics (WIL'03)*, pages 55–68, Almaty, Kazakhstan, 2003.
16. Tobias Nipkow. Higher-order unification, polymorphism, and subsorts. In S. Kaplan and M. Okada, editors, *Proc. of the 2nd Int. Workshop Conditional and Typed Rewriting Systems*, volume 516 of *LNCS*. Springer, 1991.
17. L. Paulson. Isabelle: the next 700 theorem provers. In P. Odifreddi, editor, *Logic and Computer Science*, pages 361–386. Academic Press, 1990.
18. G. Plotkin. Building in equational theories. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, volume 7, pages 73–90, Edinburgh, UK, 1972. Edinburgh University Press.
19. Z. Qian and T. Nipkow. Reduction and unification in lambda calculi with a general notion of subtype. *J. of Automated Reasoning*, 12:389–406, 1994.
20. J. Richardson and N. E. Fuchs. Development of correct transformation schemata for prolog programs. In N. E. Fuchs, editor, *Proc. of the 7th Int. Workshop on Logic Program Synthesis and Transformation, LOPSTR'97*, volume 1463 of *LNCS*, pages 263–281, Leuven, Belgium, 10–12 July 1997. Springer.
21. R. Schmidt. E-Unification for subsystems of S4. In T. Nipkow, editor, *Proc. of the 9th Int. Conference on Rewriting Techniques and Applications, RTA'98*, volume 1379 of *LNCS*, pages 106–120, Tsukuba, Japan, 1998. Springer.
22. M. Schmidt-Schauss. *Computational aspects of an order-sorted logic with term declarations*, volume 395 of *LNAI*. Springer, 1989.
23. J. Siekmann. String unification. Research paper, Essex University, 1975.
24. C. Walther. *A many sorted calculus based on resolution and paramodulation*. Pitman, London, 1987.
25. S. Wolfram. *The* MATHEMATICA *Book*. Cambridge University Press and Wolfram Research, Inc., fourth edition, 1999.