Electronic Notes in Theoretical Computer Science 66 No. 5 (2002)
URL: `http://www.elsevier.nl/locate/entcs/volume66.html` 18 pages

Pattern Unification with Sequence Variables and Flexible Arity Symbols

Temur Kutsia [1,2]

Research Institute for Symbolic Computation
Johannes Kepler University Linz
and
Software Competence Center Hagenberg
A-4232, Hagenberg, Austria

Abstract

A unification procedure for a theory with individual and sequence variables, free constants, free fixed and flexible arity function symbols and patterns is described. The procedure enumerates a set of substitution/constraint pairs which constitutes the minimal complete set of unifiers.

1 Introduction

The paper describes a unification procedure for a theory with individual and sequence variables, free constants, free fixed and flexible arity function symbols and patterns. We refer to the unification problem in this theory shortly as pattern unification with sequence variables and flexible arity symbols. Patterns abbreviate sequences of terms of unknown length, where the terms match a certain "common pattern". Sequence variables can be instantiated by arbitrary finite, possibly empty, sequences of terms. Flexible arity symbols can take arbitrary finite, possibly empty, number of arguments. In the literature the symbols with similar property are also referred to as "variable arity" or "variadic" symbols.

The subject of this research was proposed by Bruno Buchberger in [2] and in a couple of personal discussions [3]. He suggested also the term "flexible arity" instead of "variadic", mainly because of the following reason: variadic symbols, as they are understood in theorem proving or rewriting, are flattened associative symbols, i.e. flat symbols which take at least two arguments, while

[1] Supported by the Austrian Science Foundation (FWF) under Project SFB F1302 and by Software Competence Center Hagenberg (Austria) under ForMI project.
[2] Email: `kutsia@risc.uni-linz.ac.at`

flexible arity symbols can have zero or one argument as well and are not necessarily flat.

Sequence variables, patterns and flexible arity (variadic) symbols have been used in various areas:

- Knowledge management – Knowledge Interchange Format KIF [7] is an extension of first order language with (among other constructs) sequence variables and flexible arity symbols.
- Databases – Numerous formalisms involving sequence variables ([8], [15], [9]) have been developed for data representation and manipulation for genome or text databases.
- Rewriting – variadic symbols used in rewriting usually come from flattening terms with associative top function symbol. Patterns (sequence variables) used together with variadic symbols, make the syntax more flexible and expressive, and increase the performance of a rewriting system ([17], [10]).
- Programming languages – flexible arity symbols are supported by many of them. The programming language of Mathematica [18] is one of such examples, which uses the full expressive power of sequence variables and patterns. A relation of Mathematica programming language and rewrite languages, and the role of sequence variables in this relation is discussed in [2].
- Theorem proving – the Epilog package [6] can manipulate an information encoded in a subset of KIF language, containing sequence variables and flexible arity symbols. Among the other routines, Epilog includes various pattern matchers and an inference procedure based on model elimination.

Matching is the main solving technique in these applications. However, in some areas, like theorem proving or completion/rewriting, more powerful techniques (unification, for instance) are needed.

The problem whether Knuth-Bendix completion procedure [11] can be extended to handle term rewriting systems with variadic function symbols and patterns is stated as an open problem in [17]. The primary reason why it is an open problem is the absence of an appropriate unification algorithm.

In [13] we made the first step towards solving this problem, designing a unification procedure with sequence variables and flexible arity function symbols and giving a very brief informal overview of the pattern unification. Here we describe a unification procedure with sequence variables, flexible arity function symbols and patterns, thus moving further in solving the above mentioned problem. Sequence variables and pattern-terms are instances of the pattern construct of [17].

In the theorem proving context, quantification over sequence variables naturally introduces flexible arity symbols and patterns. For instance, skolemizing the expression $\forall\overline{x}\exists y\Phi[\overline{x}, y]$, where $\overline{x}$ is a sequence variable, y is an individual variable and $\Phi[\overline{x}, y]$ is a formula which depends on $\overline{x}$ and y, introduces a flexible arity Skolem function f: $\forall\overline{x}\Phi[\overline{x}, f(\overline{x})]$. On the other hand, Skolemizing the expression $\forall x \exists \overline{y}\Phi[x, \overline{y}]$ introduces a pattern $h_{1,n(x)}(x)$, which can be seen as an abbreviation of a sequence of terms $h_1(x), \dots, h_{n(x)}(x)$ of unknown length, where $h_1, \dots, h_{n(x)}$ are Skolem functions.

The problems like word equations (see e.g. [16]), equations over free semigroups [14], equations over lists of atoms with concatenation [5], pattern matching can be considered as particular pattern unification problems with sequence variables and flexible arity symbols.

We have implemented the unification procedure (without patterns and the decision algorithm) as a Mathematica package and incorporated it into the Theorema system [4], which aims at extending computer algebra systems by facilities for supporting proving.

The results in this paper are given without proofs. They can be found in [12].

2 Preliminaries

We consider an alphabet consisting of the following pairwise disjoint sets of symbols: the set of individual variables $\mathcal{V}_I$, the set of sequence variables $\mathcal{V}_S$, the set of object constants $\mathcal{C}_{Obj}$, the set of fixed arity function symbols $\mathcal{F}_{Fix}$, the set of flexible arity function symbols $\mathcal{F}_{Flex}$ and a singleton consisting of a binary predicate symbol $\simeq$ (equality). We define terms over $(\mathcal{V}, \mathcal{C})$, where $\mathcal{V}$ stands for $(\mathcal{V}_I, \mathcal{V}_S)$ and $\mathcal{C}$ - for $(\mathcal{C}_{Obj}, \mathcal{F}_{Fix}, \mathcal{F}_{Flex}, \simeq)$.

Definition 2.1 The set of terms (over $(\mathcal{V}, \mathcal{C})$) is the smallest set of strings over $(\mathcal{V}, \mathcal{C})$ that satisfies the following conditions:

- If $t \in \mathcal{V}_I \cup \mathcal{V}_S \cup \mathcal{C}_{Obj}$ then t is a term.
- If $f \in \mathcal{F}_{Fix}$, f is n-ary, $n \geq 0$ and $t_1, \dots, t_n$ are terms such that for all $1 \leq i \leq n$, $t_i \notin \mathcal{V}_S$, then $f(t_1, \dots, t_n)$ is a term. f is called the head of $f(t_1, \dots, t_n)$,
- If $f \in \mathcal{F}_{Flex}$ and $t_1, \dots, t_m$ $(m \geq 0)$ are terms, then so is $f(t_1, \dots, t_m)$. f is called the head of $f(t_1, \dots, t_m)$.

Let us have, in addition, a set of variables $\mathcal{V}_X$, called index variables, and the set $\mathcal{P}$ of linear polynomials with integer coefficients, whose variables are in $\mathcal{V}_X$. We assume that $\mathcal{V}_X$ and $\mathcal{P}$ are disjoint from any set from $\mathcal{V}$ and $\mathcal{C}$. If not otherwise stated, the following symbols, with or without indices, are used as metavariables: x, y and z – over individual variables, $\overline{x}$, $\overline{y}$ and $\overline{z}$ – over sequence variables, $\hat{x}, \hat{y}, \hat{z}$ – over index variables, v – over (individual, sequence or index) variables, a, b, c – over object constants, $\hat{a}, \hat{b}, \hat{c}$ – over integer constants, f, g and h – over (fixed or flexible arity) function symbols, s and t – over terms, $\hat{p}, \hat{q}, \hat{r}$ – over polynomials from $\mathcal{P}$.

We define the notion of pattern as follows:

Definition 2.2 The set of patterns $\mathcal{S}_{Pat}$ (over $(\mathcal{V}, \mathcal{C}, \mathcal{P})$) is the smallest set satisfying the following conditions:

- If $c \in \mathcal{C}_{Obj}$, $\hat{p}, \hat{q} \in \mathcal{P}$, then $c_{\hat{p},\hat{q}}$ is a pattern.
- If $f \in \mathcal{F}_{Fix}$, f is n-ary, $n \geq 0$ and $t_1, \ldots, t_n$ are terms such that for all $1 \leq i \leq n$, $t_i \notin \mathcal{V}_S$, and $\hat{p}, \hat{q} \in \mathcal{P}$, then $f_{\hat{p},\hat{q}}(t_1, \ldots, t_n)$ is a pattern.
- If $f \in \mathcal{F}_{Flex}$, each of $t_1, \ldots, t_m$ $(m \geq 0)$ is a term or pattern, and $\hat{p}, \hat{q} \in \mathcal{P}$, then $f_{\hat{p},\hat{q}}(t_1, \ldots, t_m)$ is a pattern.

A pattern of the form $c_{\hat{p},\hat{q}}$ is called a constant pattern. A pattern of the form $f_{\hat{p},\hat{q}}(t_1, \ldots, t_n)$ is called a functional pattern. c is called the head of $c_{\hat{p},\hat{q}}$ and f is called the head of $f_{\hat{p},\hat{q}}(t_1, \ldots, t_n)$. $f_{\hat{p},\hat{q}}$ is called a prefix of the pattern $f_{\hat{p},\hat{q}}(t_1, \ldots, t_n)$. A prefix of $c_{\hat{p},\hat{q}}$ is $c_{\hat{p},\hat{q}}$ itself. We say that a pattern $c_{\hat{p},\hat{q}}$ or $f_{\hat{p},\hat{q}}(t_1, \ldots, t_n)$ is explicit iff $\hat{p}$ and $\hat{q}$ are positive integers.

Definition 2.3 The set of quasi pattern-terms (over $(\mathcal{V}, \mathcal{C}, \mathcal{P})$), or, shortly, QP-terms, is the smallest set satisfying the following conditions:

- If $q \in \mathcal{V}_I \cup \mathcal{V}_S \cup \mathcal{C}_{Obj} \cup \mathcal{S}_{Pat}$ then q is a QP-term.
- If $f \in \mathcal{F}_{Fix}$, f is n-ary, $n \geq 0$ and $t_1, \ldots, t_n$ are QP-terms, for all $1 \leq i \leq n$, $t_i \notin \mathcal{V}_S \cup \mathcal{S}_{Pat}$, then $f(t_1, \ldots, t_n)$ is a QP-term.
- If $f \in \mathcal{F}_{Flex}$ and $t_1, \ldots, t_m$ are QP-terms, then $f(t_1, \ldots, t_m)$ is a QP-term $(m \geq 0)$.

f is called the head of $f(t_1, \ldots, t_n)$.

From now on, the letters s, t and r with or without indices are used as metavariables over QP-terms. The head of a QP-term t is denoted by $head(t)$.

Definition 2.4 The set of quasi pattern-equations (over $(\mathcal{V}, \mathcal{C}, \mathcal{P})$, or, shortly, QP-equations, is the smallest set satisfying the following condition: $\simeq (t_1,\, t_2)$ is a QP-equation over $(\mathcal{V}, \mathcal{C}, \mathcal{P})$ if

- t_1 and t_2 are QP-terms over $(\mathcal{V}, \mathcal{C}, \mathcal{P})$ and
- $t_1, t_2 \notin \mathcal{V}_S \cup \mathcal{S}_{Pat}$.

The symbol $\simeq$ is called the head of $\simeq (t_1, t_2)$. We write QP-equations in infix notation.

Let $t_1, \ldots, t_n$ $(n \geq 0)$ be a sequence of QP-terms. Then we denote by

- $ivars(t_1, \ldots, t_n)$ – the set of all individual variables occurring in the sequence $t_1, \ldots, t_n$.
- $svars(t_1, \ldots, t_n)$ – the set of all sequence variables in $t_1, \ldots, t_n$.
- $xvars(t_1, \ldots, t_n)$ – the set of all index variables in $t_1, \ldots, t_n$.
- $vars(t_1, \ldots, t_n) = ivars(t_1, \ldots, t_n) \cup svars(t_1, \ldots, t_n) \cup xvars(t_1, \ldots, t_n)$ – the set of all variables in $t_1, \ldots, t_n$.

For a sequence of QP-equations $s_1 \simeq t_1, \ldots, s_n \simeq t_n$ $(n \geq 0)$ we define $ivars(s_1 \simeq t_1, \ldots, s_n \simeq t_n) = ivars(s_1, t_1, \ldots, s_n, t_n)$. The other notions are defined in the same way.

Definition 2.5 A *QP*-term t (a *QP*-equation q) is called ground iff $vars(t) = \emptyset$ ($vars(q) = \emptyset$).

Definition 2.6 A quasi pattern-substitution, or, shortly, *QP*-substitution, is a finite set $\{x_1 \leftarrow s_1, \dots, x_n \leftarrow s_n, \overline{x}_1 \leftarrow t_1^1, \dots, t_{k_1}^1, \dots, \overline{x}_m \leftarrow t_1^m, \dots, t_{k_m}^m, \hat{x}_1 \leftarrow \hat{p}_1, \dots, \hat{x}_l \leftarrow \hat{p}_l\}$ where

- $n \geq 0$, $m \geq 0$ and for all $1 \leq i \leq m$, $k_i \geq 0$,
- $x_1, \dots, x_n$ are distinct individual variables,
- $\overline{x}_1, \dots, \overline{x}_m$ are distinct sequence variables,
- $\hat{x}_1, \dots, \hat{x}_l$ are distinct index variables,
- for all $1 \leq i \leq n$, s_i is a *QP*-term, $s_i \notin \mathcal{V}_S \cup \mathcal{S}_{Pat}$ and $s_i \neq x_i$,
- for all $1 \leq i \leq m$, $t_1^i, \dots, t_{k_i}^i$ is a sequence of *QP*-terms and if k_i=1 then $t_{k_i}^i \neq \overline{x}_i$,
- for all $1 \leq i \leq l$, $\hat{p}_i \in \mathcal{P}$ and $\hat{p}_i \neq \hat{x}_i$.

Each $x_i \leftarrow s_i$, $\overline{x}_i \leftarrow t_1^i, \dots, t_{k_i}^i$ and $\hat{x}_i \leftarrow \hat{p}_i$ is called a *QP*-binding respectively for x_i, $\overline{x_i}$ and $\hat{x}_i$.

A *QP*-substitution is called empty iff $n = 0$, $m = 0$ and $l = 0$. Greek letters are used to denote *QP*-substitutions. The letter ε denotes the empty *QP*-substitution. We define a notion of instance for index variables and polynomials from $\mathcal{P}$:

Definition 2.7 Let θ be a *QP*-substitution. Then:

- An instance of an index variable $\hat{x}$ with respect to θ, denoted $\hat{x}\theta$, is defined as
$$\hat{x}\theta = \begin{cases} \hat{p},\ if\ \hat{x} \leftarrow \hat{p} \in \theta, \\ \hat{x},\ otherwise \end{cases}$$
- An instance of a polynomial $\hat{p} = \hat{a}_1\hat{x}_1 + \cdots + \hat{a}_k\hat{x}_k \in \mathcal{P}$ with respect to θ, denoted $\hat{p}\theta$, is a polynomial in $\mathcal{P}$ obtained from $\hat{a}_1\hat{x}_1\theta + \cdots + \hat{a}_k\hat{x}_k\theta$ by arithmetic simplification.

Definition 2.8 Given a *QP*-substitution θ, we define an instance of a *QP*-term or *QP*-equation with respect to θ recursively as follows:

- $x\theta = \begin{cases} s,\ if\ x \leftarrow s \in \theta, \\ x,\ otherwise \end{cases}$
- $\overline{x}\theta = \begin{cases} s_1, \dots, s_m,\ if\ x \leftarrow s_1, \dots, s_m \in \theta,\ m \geq 0 \\ \overline{x}, \qquad otherwise \end{cases}$
- $c_{\hat{p},\hat{q}}\theta = c_{\hat{p}\theta,\hat{q}\theta}$.
- $f_{\hat{p},\hat{q}}(s_1, \dots, s_n)\theta = f_{\hat{p}\theta,\hat{q}\theta}(s_1\theta, \dots, s_n\theta)$.
- $f(s_1, \dots, s_n)\theta = f(s_1\theta, \dots, s_n\theta)$.

An instance of an equation is defined as follows: $(s_1 \simeq s_2)\theta = s_1\theta \simeq s_2\theta$.

Definition 2.9 The domain, codomain and range of a *QP*-substitution σ are defined respectively as

- $dom(\sigma) = \{v \mid v\sigma \neq v\}$,
- $cod(\sigma) = \{v\sigma \mid v \in dom(\sigma)\}$,
- $ran(\sigma) = \bigcup_{v \in dom(\sigma)} vars(v\sigma)$.

Definition 2.10 A substitution σ is called ground iff $ran(\sigma) = \emptyset$.

Definition 2.11 Let $\theta = \{x_1 \leftarrow t_1, \dots, x_n \leftarrow t_n, \overline{x_1} \leftarrow t^1_1, \dots, t^1_{k_1}, \dots, \overline{x_m} \leftarrow t^m_1, \dots, t^m_{k_m}, \hat{x}_1 \leftarrow \hat{p}_1, \dots, \hat{x}_k \leftarrow \hat{p}_k\}$ and $\lambda = \{y_1 \leftarrow s_1, \dots, y_l \leftarrow s_l, \overline{y_1} \leftarrow s^1_1, \dots, s^1_{q_1}, \dots, \overline{y_r} \leftarrow s^r_1, \dots, s^r_{q_r}, \hat{y}_1 \leftarrow \hat{q}_1, \dots, \hat{y}_q \leftarrow \hat{q}_q\}$ be two *QP*-substitutions. Then the composition of θ and λ is the *QP*-substitution, denoted by $\theta \circ \lambda$, obtained from the set

$$\begin{aligned}
\{\ & x_1 \leftarrow t_1\lambda, \dots, x_n \leftarrow t_n\lambda,\ \overline{x_1} \leftarrow t^1_1\lambda, \dots, t^1_{k_1}\lambda, \dots, \overline{x_m} \leftarrow t^m_1\lambda, \dots, t^m_{k_m}\lambda, \\
& \hat{x}_1 \leftarrow \hat{p}_1\lambda, \dots, \hat{x}_k \leftarrow \hat{p}_k\lambda,\ y_1 \leftarrow s_1, \dots, y_l \leftarrow s_l, \\
& \overline{y_1} \leftarrow s^1_1, \dots, s^1_{q_1}, \dots, \overline{y_r} \leftarrow s^r_1, \dots, s^r_{q_r},\ \hat{y}_1 \leftarrow \hat{q}_1, \dots, \hat{y}_q \leftarrow \hat{q}_q\}
\end{aligned}$$

by deleting

- all the elements $x_i \leftarrow t_i\lambda$ $(1 \leq i \leq n)$ for which $x_i = t_i\lambda$,
- all the elements $\overline{x_i} \leftarrow t^i_1\lambda, \dots, t^i_{k_i}\lambda$ $(1 \leq i \leq m)$ for which $k_i = 1$ and $\overline{x}_i = t^i_1\lambda$,
- all the elements $\hat{x}_i \leftarrow \hat{p}_i\lambda$ $(1 \leq i \leq k)$ for which $\hat{x}_i = \hat{p}_i\lambda$,
- all the elements $y_i \leftarrow s_i$ $(1 \leq i \leq l)$ such that $y_i \in \{x_1, \dots, x_n\}$,
- all the elements $\overline{y_i} \leftarrow s^i_1, \dots, s^i_{q_i}$ $(1 \leq i \leq r)$ such that $\overline{y_i} \in \{\overline{x_1}, \dots, \overline{x_m}\}$,
- all the elements $\hat{y}_i \leftarrow \hat{q}_i$ $(1 \leq i \leq q)$ such that $\hat{y}_i \in \{\hat{x}_1, \dots, \hat{x}_k\}$.

Example 2.12 Let $\theta = \{x \leftarrow f(y),\ \overline{x} \leftarrow \overline{y}, \overline{x},\ \overline{y} \leftarrow \overline{y}, \overline{z},\ \hat{x} \leftarrow 3\hat{x} + \hat{y},\ \hat{z} \leftarrow \hat{y} - 2\}$ and $\lambda = \{y \leftarrow g(c),\ \overline{x} \leftarrow c,\ \overline{z} \leftarrow,\ \hat{x} \leftarrow 2\hat{y} + 1,\ \hat{y} \leftarrow \hat{z} + 2\}$. Then $\theta \circ \lambda = \{x \leftarrow f(g(c)),\ y \leftarrow g(c),\ \overline{x} \leftarrow \overline{y}, c,\ \overline{z} \leftarrow,\ \hat{x} \leftarrow 6\hat{y} + \hat{z} + 5,\ \hat{y} \leftarrow \hat{z} + 2\}$.

Composition of *QP*-substitutions is associative (see [12]).

Definition 2.13 A restriction of a *QP*-substitution θ on a set of variables V, denoted $\theta|_V$, is a *QP*-substitution $\{v \leftarrow \tilde{r} \mid v \leftarrow \tilde{r} \in \theta \text{ and } v \in V\}$[3].

3 Equational Theory with Sequence Variables, Flexible Arity Symbols and Patterns

A set of *QP*-equations E (called presentation) defines a quasi-pattern equational theory, i.e. the equality of *QP*-terms induced by E. We use the term

[3] $\tilde{r}$ is either a single *QP*-term, a (possibly empty) sequence of *QP*-terms or a polynomial from $\mathcal{P}$.

QPE-theory for the *QP*-equational theory defined by E. We will write $s \simeq_E t$ for $s \simeq t$ modulo E. Some examples of E-theories are:

(i) Free theory ($\emptyset$): $E = \emptyset$;

(ii) Flat theory (F): $E = \{f(\overline{x}, f(\overline{y}), \overline{z}) \simeq f(\overline{x}, \overline{y}, \overline{z})\}$.

(iii) Orderless theory (O): $E = \{f(\overline{x}, x, \overline{y}, y, \overline{z}) \simeq f(\overline{x}, y, \overline{y}, x, \overline{z})\}$.

(iv) Flat-orderless theory (FO):
$E = \{f(\overline{x}, f(\overline{y}), \overline{z}) \simeq f(\overline{x}, \overline{y}, \overline{z}), f(\overline{x}, x, \overline{y}, y, \overline{z}) \simeq f(\overline{x}, y, \overline{y}, x, \overline{z})\}$.

Solving *QP*-equations in a *QPE*-theory is called *QPE*-unification. The fact that the *QP*-equation $s \simeq_E t$ has to be solved is written as $s \overset{?}{\simeq}_E t$.

Definition 3.1 A general quasi pattern E-unification, or, shortly, *QPE*-unification problem is a finite system of *QP*-equations $\langle s_1 \overset{?}{\simeq}_E t_1, \ldots, s_n \overset{?}{\simeq}_E t_n \rangle$.

Below by a *QP*-expression we mean either a *QP*-term, *QP*-equation, *QP*-substitution or *QPE*-unification problem.

Definition 3.2 Let Q be a *QP*-expression. The explicit pattern expansion in Q, denoted as $expex(Q)$, is a *QP*-expression obtained from Q by replacing each occurrence of an explicit pattern in Q with the sequence of *QP*-terms as long as possible in the following way:

- each occurrence of an explicit pattern $c_{\hat{a},\hat{a}}$ is replaced by $c_{\hat{a}}$;
- each occurrence of $c_{\hat{a},\hat{b}}$, $\hat{a} < \hat{b}$, is replaced by $c_{\hat{a}}, c_{\hat{a}+1}, \ldots, c_{\hat{b}}$;
- each occurrence of $f_{\hat{a},\hat{a}}(t_1, \ldots, t_n)$ is replaced by $f_{\hat{a}}(t_1, \ldots, t_n)$.
- each occurrence of an explicit pattern $f_{\hat{a},\hat{b}}(t_1, \ldots, t_n)$, $\hat{a} < \hat{b}$, is replaced by the sequence $f_{\hat{a}}(t_1, \ldots, t_n), f_{\hat{a}+1}(t_1, \ldots, t_n), \ldots, f_{\hat{b}}(t_1, \ldots, t_n)$.

Example 3.3 Let t be a *QP*-term $f(a, b, g_{1,3}(c_{1,2}, \overline{y}), h_{1,\hat{x}}(x))$. Then

$$expex(t) = f(a, b, g_1(c_1, c_2, \overline{y}), g_2(c_1, c_2, \overline{y}), g_3(c_1, c_2, \overline{y}), h_{1,\hat{x}}(x)).$$

Definition 3.4 Let Q be a *QP*-expression. Q is called

- a pattern-term or, shortly, *P*-term;
- a pattern-equation or, shortly, *P*-equation;
- a pattern-substitution or, shortly, *P*-substitution or
- a pattern-E-unification problem or, shortly, *PE*-unification problem

over $(\mathcal{V}, \mathcal{C}, \mathcal{P})$ iff there exist a substitution σ such that $dom(\sigma) \subset \mathcal{V}_X$ and $expex(Q\sigma)$ is respectively a term, an equation, a substitution or an E-unification problem over $(\mathcal{V}, \mathcal{C}, \mathcal{P})$.

Definition 3.5 For a *QP*-expression Q, the associated system of linear Diophantine constraints $ldc(Q)$ is defined in the following way:

- Q is a *QP*-term. Then
  · If $Q \in \mathcal{V}_I \cup \mathcal{V}_S \cup \mathcal{C}_{Obj}$ then $ldc(Q)$ is empty.

  - If Q is $c_{\hat{p},\hat{q}}$ or $f_{\hat{p},\hat{q}}(t_1,\dots,t_n)$, then $ldc(Q)$ is $1 \le \hat{p} \ \wedge\ \hat{p} \le \hat{q}$.
  - If Q is $f(t_1,\dots,t_n)$, where $f \in \mathcal{F}_{Fix} \cup \mathcal{F}_{Flex}$, then $ldc(Q)$ is $ldc(t_1) \wedge \dots \wedge ldc(t_n)$,
- Q is either *QP*-equation, *QP*-substitution or a *QPE*-unification problem. Then $ldc(Q)$ is $ldc(t_1) \wedge \dots \wedge ldc(t_m)$, where $t_1,\dots,t_m$ are all *QP*-terms occurring in Q.

By *P*-expression we mean either a *P*-term, *P*-equation, *P*-substitution or *PE*-unification problem. The theorem below shows how to decide whether a *QP*-expression is the corresponding *P*-expression.

Theorem 3.6 *A QP-expression Q is the corresponding P-expression if the constraint $ldc(Q)$ has a positive integer solution.*

Definition 3.7 Let $\mathfrak{U}$ be a *PE*-unification problem $\langle s_1 \overset{?}{\simeq}_E t_1, \dots, s_n \overset{?}{\simeq}_E t_n\rangle$. A *P*-substitution θ is called a *PE*-unifier of $\mathfrak{U}$ iff

- $xvar(\mathfrak{U}) \subseteq dom(\theta)$;
- for each $\hat{x} \in dom(\theta)$, $\hat{x}\theta$ is a positive integer;
- each pattern which occurs in *P*-terms in $cod(\theta)$ is explicit;
- for each $1 \le i \le n$, the *P*-equality $s_i\theta \simeq_E t_i\theta$ holds.

Example 3.8 Let $\mathfrak{U} = f(\overline{x},\overline{y}) \overset{?}{\simeq}_\emptyset f(h_{\hat{y},\hat{z}}(z))$. Then $\theta = \{\overline{x} \leftarrow h_{1,2}(z), \overline{y} \leftarrow h_{3,6}(z),\ \hat{y} \leftarrow 1, \hat{z} \leftarrow 6\}$ is one of $\emptyset$-unifiers of $\mathfrak{U}$.

Definition 3.9 A *P*-substitution θ is more general than a *P*-substitution σ on a finite set of variables V modulo a theory E (denoted $\theta \ll_E^V \sigma$) iff there exists a *P*-substitution λ such that

- for all $\overline{x} \in V$,
  - $\overline{x} \leftarrow\ \notin \lambda$;
  - there exist *P*-terms $t_1,\dots,t_n, s_1,\dots,s_n$, $n \ge 0$ such that $\overline{x}\sigma = t_1,\dots,t_n$, $\overline{x}\theta \circ \lambda = s_1,\dots,s_n$ and for each $1 \le i \le n$, either t_i and s_i are the same sequence variables or the equality $t_i \simeq_E s_i$ holds;
- for all $x \in V$, the equality $x\sigma \simeq_E x\theta \circ \lambda$ holds;
- for all $\hat{x} \in V$, $\hat{x}\sigma = \hat{x}\theta \circ \lambda$.

Example 3.10 $\{\overline{x} \leftarrow \overline{y}\} \ll_\emptyset^{\{\overline{x},\overline{y}\}} \{\overline{x} \leftarrow a,\overline{z},\ \overline{y} \leftarrow a,\overline{z}\}$, but not $\{\overline{x} \leftarrow \overline{y}\} \ll_\emptyset^{\{\overline{x},\overline{y}\}} \{\overline{x} \leftarrow, \overline{y} \leftarrow\}$.

Definition 3.11 Let $\mathfrak{U}$ be a *PE*-unification problem. The minimal complete set of *PE*-unifiers of $\mathfrak{U}$, denoted $mcu_E(\mathfrak{U})$, is a set of *P*-substitutions, satisfying the following conditions:

PE-correctness - for all $\theta \in mcu_E(\mathfrak{U})$, θ is an *PE*-unifier of $\mathfrak{U}$.

PE-completeness - for any *PE*-unifier σ of $\mathfrak{U}$ there exists $\theta \in mcu_E(\mathfrak{U})$ such that $\theta \ll_E^{vars(\mathfrak{U})} \sigma$.

PE-minimality - for all θ, $\sigma \in mcu_E(\mathfrak{U})$, $\theta \ll_E^{vars(\mathfrak{U})} \sigma$ implies $\theta = \sigma$.

Below in this paper we consider only the *QP*-$\emptyset$-theory, although the results valid for arbitrary *QPE*-theories are formulated in a general setting.

We represent $mcu_E(\mathfrak{U})$ as a set of *P*-substitution/constraint pairs. The constraints are linear Diophantine equations and/or inequalities. The representation must satisfy the following properties:

- for each pair $\{\theta, d\}$ in the representation and for each positive integer solution μ of d, the *P*-substitution $(\theta \circ \mu)|_{vars(\mathfrak{U})}$ is in $mcu_E(\mathfrak{U})$;
- for each substitution $\sigma \in mcu_E(\mathfrak{U})$ there is a pair $\{\theta, d\}$ in the representation such that $\sigma = (\theta \circ \mu)|_{vars(\mathfrak{U})}$ for a positive integer solution μ of d.

The following two examples give a demonstration of a representation of a minimal complete set of unifiers as a set of *P*-substitution/constraint pairs:

Example 3.12 Let $\mathfrak{U} = f(\overline{x}, \overline{y}) \overset{?}{\simeq}_\emptyset f(h_{\hat{x},\hat{y}}(z))$. Then we can represent $mcu_\emptyset(\mathfrak{U})$ as a finite set of *P*-substitution/constraint pairs:

$$
\begin{aligned}
S = \{\ &\{\{\overline{x} \leftarrow, \overline{y} \leftarrow h_{\hat{x},\hat{y}}(z)\}, 1 \le \hat{x} \wedge \hat{x} \le \hat{y}\},\\
&\{\{\overline{x} \leftarrow, h_{\hat{x},\hat{y}}(z), \overline{y} \leftarrow\}, 1 \le \hat{x} \wedge \hat{x} \le \hat{y}\},\\
&\{\{\overline{x} \leftarrow h_{\hat{x},\hat{z}}(z), \overline{y} \leftarrow h_{\hat{z}+1,\hat{y}}(z)\}, 1 \le \hat{x} \wedge \hat{x} \le \hat{z} \wedge \hat{z}+1 \le \hat{y}\}\ \}.
\end{aligned}
$$

In fact,

$$
\begin{aligned}
mcu_\emptyset(\mathfrak{U}) = \{\sigma \mid\ &\text{there exists } \{\theta, d\} \in \text{S and } \mu \text{ such that } \mu \text{ is a}\\
&\text{positive integer solution of } d \text{ and } \sigma = (\theta \circ \mu)|_{vars(\mathfrak{U})}\}.
\end{aligned}
$$

For instance, a solution $\{\hat{x} \leftarrow 1, \hat{r} \leftarrow 3, \hat{y} \leftarrow 4\}$ of the constraint $1 \le \hat{x} \wedge \hat{x} \le \hat{z} \wedge \hat{z}+1 \le \hat{y}$, applied on the substitution $\{\overline{x} \leftarrow h_{\hat{x},\hat{z}}(z), \overline{y} \leftarrow h_{\hat{z}+1,\hat{y}}(z)\}$ gives $\{\overline{x} \leftarrow h_{1,3}(z), \overline{y} \leftarrow h_{4,4}(z), \hat{x} \leftarrow 1, \hat{z} \leftarrow 3, \hat{y} \leftarrow 4\}$. The restriction of the latter to $vars(\mathfrak{U})$ is $\{\overline{x} \leftarrow h_{1,3}(z), \overline{y} \leftarrow h_{4,4}(z), \hat{x} \leftarrow 1, \hat{y} \leftarrow 4\}$, which belongs to $mcu_E(\mathfrak{U})$. In the expanded form the substitution looks like $\{\overline{x} \leftarrow h_1(z), h_2(z), h_3(z), \overline{y} \leftarrow h_4(z), \hat{x} \leftarrow 1, \hat{y} \leftarrow 4\}$.

Example 3.13 Let $\mathfrak{U} = f(\overline{x}, h_{\hat{x},\hat{y}}(z)) \overset{?}{\simeq}_\emptyset f(h_{\hat{x},\hat{y}}(z), \overline{x})$. Then the set S gives an infinite representation of $mcu_\emptyset(\mathfrak{U})$ as a set of substitution/constraint pairs:

$$
\begin{aligned}
S = \{\ &\{\{\overline{x} \leftarrow\}, 1 \le \hat{x} \wedge \hat{x} \le \hat{y}\},\\
&\{\{\overline{x} \leftarrow h_{\hat{x},\hat{y}}(z)\}, 1 \le \hat{x} \wedge \hat{x} \le \hat{y}\},\\
&\{\{\overline{x} \leftarrow h_{\hat{x},\hat{y}}(z), h_{\hat{x},\hat{y}}(z)\}, 1 \le \hat{x} \wedge \hat{x} \le \hat{y}\}\ \ldots\}.
\end{aligned}
$$

Again,

$$
\begin{aligned}
mcu_\emptyset(\mathfrak{U}) = \{\sigma \mid\ &\text{there exists } \{\theta, d\} \in \text{S and } \mu \text{ such that } \mu \text{ is a}\\
&\text{positive integer solution of } d \text{ and } \sigma = (\theta \circ \mu)|_{vars(\mathfrak{U})}\}.
\end{aligned}
$$

4 Unification Procedure

Next, we design a unification procedure for a P-$\emptyset$-unification problem. Note it is enough to consider single P-equations instead of systems of P-equations. The problem has a form of P-equation $t_1 \overset{?}{\simeq}_\emptyset t_2$. We design the unification procedure as a tree generation process based on three basic steps: projection, transformation and pattern-simplification. They are described in terms of "quasi-patterns" instead of "patterns".

4.1 Projection

Projection eliminates some sequence variables from the given QP-$\emptyset$-unification problem $\mathfrak{Q}_\emptyset$. Let $\Pi(\mathfrak{Q}_\emptyset)$ be the following set of substitutions: $\{\{\overline{x} \leftarrow \;|\; \overline{x} \in S\} \mid S \subseteq svars(\mathfrak{Q}_\emptyset)\}$. $\Pi(\mathfrak{Q}_\emptyset)$ is called the set of projecting substitutions for $\mathfrak{Q}_\emptyset$. Each $\pi \in \Pi$ replaces some sequence variables from $\mathfrak{Q}_\emptyset$ with the empty sequence. The projection rule is shown in Figure 1.

$$\begin{array}{lll} \mathsf{Projection:} & s \overset{?}{\simeq}_\emptyset t \rightsquigarrow \langle\langle s\pi_1 \overset{?}{\simeq}_\emptyset t\pi_1,\ \pi_1,\ d\rangle, & \text{where } \{\pi_1, \ldots, \pi_k\} = \Pi(s \overset{?}{\simeq}_\emptyset t) \\ & \quad \ldots, \langle s\pi_k \overset{?}{\simeq}_\emptyset t\pi_k,\ \pi_k,\ d\rangle\rangle & \quad \text{and } d = ldc(s \overset{?}{\simeq}_\emptyset t). \end{array}$$

Fig. 1. Projection rule for QP-$\emptyset$-unification.

4.2 Transformation

Each of the transformation rules for QP-$\emptyset$-unification have one of the following forms: $\mathfrak{Q}_\emptyset \rightsquigarrow \bot$ or $\mathfrak{Q}_\emptyset \rightsquigarrow \langle\langle \mathfrak{S}_1, \sigma_1, d_1\rangle, \ldots, \langle \mathfrak{S}_n, \sigma_n, d_n\rangle\rangle$ where each of the successors $\mathfrak{S}_i$ is either $\top$ or a new unification problem, σ-s are substitutions and d-s are linear Diophantine constraints.

Transformation rules are success, failure, elimination and splitting rules given on the figures 2, 3, 4 and 5 below.

$$\begin{array}{lll} \mathsf{SuccessT:} & t \overset{?}{\simeq}_\emptyset t \rightsquigarrow \langle\langle \top,\ \varepsilon,\ true\rangle\rangle. & \\ & x \overset{?}{\simeq}_\emptyset t \rightsquigarrow \langle\langle \top,\ \{x \leftarrow t\},\ true\rangle\rangle, & \text{if } x \notin ivars(t). \\ & t \overset{?}{\simeq}_\emptyset x \rightsquigarrow \langle\langle \top,\ \{x \leftarrow t\},\ true\rangle\rangle, & \text{if } x \notin ivars(t). \end{array}$$

Fig. 2. Success rules for transformation for QP-$\emptyset$-unification.

4.3 Pattern Simplification

Like the transformation rules, QP-$\emptyset$-simplification rules for patterns have one of the following forms: $\mathfrak{Q}_\emptyset \rightsquigarrow \bot$ or $\mathfrak{Q}_\emptyset \rightsquigarrow \langle\langle \mathfrak{S}_1, \sigma_1, d_1\rangle, \ldots, \langle \mathfrak{S}_n, \sigma_n,\ d_n\rangle\rangle$

FailureT:	$c_1 \overset{?}{\simeq}_\emptyset c_2 \rightsquigarrow \bot,$	if $c_1 \neq c_2$.
	$x \overset{?}{\simeq}_\emptyset t \rightsquigarrow \bot,$	if $t \neq x$ and $x \in ivars(t)$.
	$t \overset{?}{\simeq}_\emptyset x \rightsquigarrow \bot,$	if $t \neq x$ and $x \in ivars(t)$.
	$f_1(\tilde{t}) \overset{?}{\simeq}_\emptyset f_2(\tilde{s}) \rightsquigarrow \bot,$	if $f_1 \neq f_2$.
	$f() \overset{?}{\simeq}_\emptyset f(t_1, \tilde{t}) \rightsquigarrow \bot.$	
	$f(t_1, \tilde{t}) \overset{?}{\simeq}_\emptyset f() \rightsquigarrow \bot.$	
	$f(\overline{x}, \tilde{t}) \overset{?}{\simeq}_\emptyset f(s_1, \tilde{s}) \rightsquigarrow \bot,$	if $s_1 \neq \overline{x}$ and $\overline{x} \in svars(s_1)$.
	$f(s_1, \tilde{s}) \overset{?}{\simeq}_\emptyset f(\overline{x}, \tilde{t}) \rightsquigarrow \bot,$	if $s_1 \neq \overline{x}$ and $\overline{x} \in svars(s_1)$.
	$f(t_1, \tilde{t}) \overset{?}{\simeq}_\emptyset f(s_1, \tilde{s}) \rightsquigarrow \bot,$	if $t_1 \overset{?}{\simeq}_\emptyset s_1 \rightsquigarrow \bot$.

Fig. 3. Failure rules for transformation for *QP*-$\emptyset$-unification. $\tilde{t}$ and $\tilde{s}$ are possibly empty sequences of *QP*-terms. $f, f_1, f_2 \in \mathcal{F}_{Fix} \cup \mathcal{F}_{Flex}$.

where each of the successors $\mathfrak{S}_i$ is either $\top$ or a new unification problem, σ-s are substitutions and d-s are linear Diophantine constraints.

The full set of pattern simplification rules consists of failure, contraction and separation rules for constants patterns (given on the figures 6, 7 and 8) and for functional patterns (given on the figures 9, 10 and 11).

4.4 Tree Generation

Projection, transformation and pattern simplification can be seen as single steps in a tree generation process. Each node of the tree is labeled either with a *QP*-$\emptyset$-unification problem, $\top$ or $\bot$. The edges of the tree are labeled by substitutions and linear Diophantine constraints. The nodes labeled with $\top$ or $\bot$ are terminal nodes. The nodes labeled with *QP*-$\emptyset$-unification problems are non-terminal nodes. The children of a non-terminal node are constructed in the following way:

Let $\mathfrak{Q}$ be a *QP*-$\emptyset$-unification problem attached to a non-terminal node and $c_{\mathfrak{Q}}$ be a conjunction of linear Diophantine constraints attached to the edges in the branch, from the root of the tree till the current node. First, we check whether $c_{\mathfrak{Q}}$ is satisfiable. If it is not, we replace $\mathfrak{Q}$ with the new label $\bot$. Otherwise we proceed as follows: If we can decide whether $\mathfrak{Q}$ is not unifiable, then we replace $\mathfrak{Q}$ with the new label $\bot$. Otherwise we apply projection, transformation or pattern simplification on $\mathfrak{Q}$ and get $\langle\langle \mathfrak{S}_1, \sigma_1, d_1\rangle, \ldots, \langle \mathfrak{S}_n, \sigma_n, d_n\rangle\rangle$. Then the node $\mathfrak{Q}$ has n children, labeled respectively with $\mathfrak{S}_1, \ldots, \mathfrak{S}_n$ and the edge to the $\mathfrak{S}_i$ node is labeled with σ_i and d_i ($1 \leq i \leq n$). The set $\{\sigma_1, \ldots, \sigma_n\}$ is denoted by $sub(\mathfrak{Q})$. The set $\{d_1, \ldots, d_n\}$ is denoted by $con(\mathfrak{Q})$.

Satisfiability of $c_{\mathfrak{Q}}$ can be checked by one of known algorithms for solving linear Diophantine equational and inequational systems, e.g. [1].

$$
\begin{array}{lll}
\textsf{EliminationT:} & f(t_1,\tilde{t})\overset{?}{\simeq}_{\emptyset}f(s_1,\tilde{s}) \leadsto & \text{if } t_1\overset{?}{\simeq}_{\emptyset}s_1 \leadsto \\
& \quad \langle\langle g(\tilde{t}\sigma)\overset{?}{\simeq}_{\emptyset}g(\tilde{s}\sigma),\ \sigma,\ \mathit{true}\rangle\rangle, & \quad \langle\langle \top,\ \sigma,\ \mathit{true}\rangle\rangle. \\
& f(\overline{x},\tilde{t})\overset{?}{\simeq}_{\emptyset}f(\overline{x},\tilde{s}) \leadsto & \\
& \quad \langle\langle g(\tilde{t})\overset{?}{\simeq}_{\emptyset}g(\tilde{s}),\ \varepsilon,\ \mathit{true}\rangle\rangle. & \\
& f(\overline{x},\tilde{t})\overset{?}{\simeq}_{\emptyset}f(s_1,\tilde{s}) \leadsto & \text{if } \overline{x}\notin \mathit{svars}(s_1) \text{ and} \\
& \quad \langle\langle g(\tilde{t}\sigma_1)\overset{?}{\simeq}_{\emptyset}g(\tilde{s}\sigma_1),\ \sigma_1,\ \mathit{true}\rangle, & \quad \sigma_1=\{\overline{x}\leftarrow s_1\}, \\
& \quad \langle g(\overline{x},\tilde{t}\sigma_2)\overset{?}{\simeq}_{\emptyset}g(\tilde{s}\sigma_2),\ \sigma_2,\ \mathit{true}\rangle\rangle, & \quad \sigma_2=\{\overline{x}\leftarrow s_1,\overline{x}\}. \\
& f(t_1,\tilde{t})\overset{?}{\simeq}_{\emptyset}f(\overline{x},\tilde{s}) \leadsto & \text{if } \overline{x}\notin \mathit{svars}(t_1) \text{ and} \\
& \quad \langle\langle g(\tilde{t}\sigma_1)\overset{?}{\simeq}_{\emptyset}g(\tilde{s}\sigma_1),\ \sigma_1,\ \mathit{true}\rangle, & \quad \sigma_1=\{\overline{x}\leftarrow t_1\}, \\
& \quad \langle g(\tilde{t}\sigma_2)\overset{?}{\simeq}_{\emptyset}g(\overline{x},\tilde{s}\sigma_2),\ \sigma_2,\ \mathit{true}\rangle\rangle, & \quad \sigma_2=\{\overline{x}\leftarrow t_1,\overline{x}\}. \\
& f(\overline{x},\tilde{t})\overset{?}{\simeq}_{\emptyset}f(\overline{y},\tilde{s}) \leadsto & \text{where} \\
& \quad \langle\langle g(\tilde{t}\sigma_1)\overset{?}{\simeq}_{\emptyset}g(\tilde{s}\sigma_1),\ \sigma_1,\ \mathit{true}\rangle, & \quad \sigma_1=\{\overline{x}\leftarrow \overline{y}\}, \\
& \quad \langle g(\overline{x},\tilde{t}\sigma_2)\overset{?}{\simeq}_{\emptyset}g(\tilde{s}\sigma_2),\ \sigma_2,\ \mathit{true}\rangle, & \quad \sigma_2=\{\overline{x}\leftarrow \overline{y},\overline{x}\}, \\
& \quad \langle g(\tilde{t}\sigma_3)\overset{?}{\simeq}_{\emptyset}g(\overline{y},\tilde{s}\sigma_3),\ \sigma_3,\ \mathit{true}\rangle\rangle, & \quad \sigma_3=\{\overline{y}\leftarrow \overline{x},\overline{y}\}.
\end{array}
$$

Fig. 4. Elimination rules for transformation for QP-$\emptyset$-unification. $t_1, s_1 \notin \mathcal{V}_S \cup \mathcal{S}_{Pat}$. $\tilde{t}$ and $\tilde{s}$ are possibly empty sequences of QP-terms. $g \in \mathcal{F}_{Flex}$ is a new symbol, if in the same rule $f \in \mathcal{F}_{Fix}$. Otherwise $g = f$.

$$
\begin{array}{lll}
\textsf{SplittingT:} & f(t_1,\tilde{t})\overset{?}{\simeq}_{\emptyset}f(s_1,\tilde{s}) \leadsto & \text{if } t_1\overset{?}{\simeq}_{\emptyset}s_1 \leadsto \\
& \langle\langle f(r_1,\tilde{t}\sigma_1)\overset{?}{\simeq}_{\emptyset}f(q_1,\tilde{s}\sigma_1),\ \sigma_1,\ d_1\rangle, & \langle\langle r_1\overset{?}{\simeq}_{\emptyset}q_1,\ \sigma_1,\ d_1\rangle, \\
& \dots, & \dots, \\
& \langle f(r_k,\tilde{t}\sigma_k)\overset{?}{\simeq}_{\emptyset}f(q_k,\tilde{s}\sigma_k),\ \sigma_k,\ d_k\rangle\rangle & \langle r_k\overset{?}{\simeq}_{\emptyset}q_k,\ \sigma_k,\ d_k\rangle\rangle.
\end{array}
$$

Fig. 5. Splitting rules for transformation for QP-$\emptyset$-unification. $t_1, s_1 \notin \mathcal{V}_I \cup \mathcal{V}_S \cup \mathcal{S}_{Pat}$. $\tilde{t}$ and $\tilde{s}$ are possibly empty sequences of terms.

We design the general P-$\emptyset$-unification procedure for $\mathfrak{Q}_{\emptyset}$ as a breadth first (level by level) tree generation process. The root of the tree is labeled with $\mathfrak{Q}_{\emptyset}$ (zero level). First level nodes (the children of the root) of the tree are obtained from $\mathfrak{Q}_{\emptyset}$ by projection[4]. Starting from the second level, we apply only a transformation or pattern simplification step to a QP-$\emptyset$-unification problem of

[4] Starting from the first level, the unification problems attached to the nodes in the tree might not be P-$\emptyset$-unification problems, but they are, of course, QP-$\emptyset$-unification problems.

$$\begin{array}{lll}
\mathsf{FailureC}: & f(t_1,\tilde{t})\overset{?}{\simeq}_{\emptyset} f(c_{\hat{p},\hat{q}},\tilde{s}) \rightsquigarrow \bot, & \text{if } t_1 \notin \mathcal{V}_I \cup \mathcal{V}_S,\ head(t_1) \neq c. \\
 & f(h_{\hat{p},\hat{q}},\tilde{t})\overset{?}{\simeq}_{\emptyset} f(s_1,\tilde{s}) \rightsquigarrow \bot, & \text{if } s_1 \notin \mathcal{V}_I \cup \mathcal{V}_S,\ head(s_1) \neq c.
\end{array}$$

Fig. 6. Failure rules for constant pattern simplification for *QP*-$\emptyset$-unification. $\tilde{t}$ and $\tilde{s}$ are possibly empty sequences of *QP*-terms. $f \in \mathcal{F}_{Flex}$.

$$\begin{array}{lll}
\mathsf{ContractionC}: & f(x,\tilde{t})\overset{?}{\simeq}_{\emptyset} f(c_{\hat{p},\hat{q}},\tilde{s}) \rightsquigarrow & \text{where} \\
 & \quad \langle\langle f(\tilde{t})\sigma_1 \overset{?}{\simeq}_{\emptyset} f(\tilde{s})\sigma_1, \sigma_1, \hat{p}=\hat{q}\rangle, & \quad \sigma_1=\{x\leftarrow c_{\hat{q}}\}, \\
 & \quad \langle f(\tilde{t})\sigma_2 \overset{?}{\simeq}_{\emptyset} f(c_{\hat{p}+1,\hat{q}},\tilde{s})\sigma_2, & \quad \sigma_2=\{x\leftarrow c_{\hat{p}}\} \\
 & \qquad \sigma_2, \hat{p}+1\leq \hat{q}\rangle\rangle. & \\
 & f(c_{\hat{p},\hat{q}},\tilde{t})\overset{?}{\simeq}_{\emptyset} f(x,\tilde{s}) \rightsquigarrow & \text{where} \\
 & \quad \langle\langle f(\tilde{t})\sigma_1 \overset{?}{\simeq}_{\emptyset} f(\tilde{s})\sigma_1, \sigma_1, \hat{p}=\hat{q}\rangle, & \quad \sigma_1=\{x\leftarrow c_{\hat{q}}\}, \\
 & \quad \langle f(c_{\hat{p}+1,\hat{q}},\tilde{t})\sigma_2 \overset{?}{\simeq}_{\emptyset} f(\tilde{s})\sigma_2, & \quad \sigma_2=\{x\leftarrow c_{\hat{p}}\}. \\
 & \qquad \sigma_2, \hat{p}+1\leq \hat{q}\rangle\rangle, & \\
 & f(\overline{x},\tilde{t})\overset{?}{\simeq}_{\emptyset} f(c_{\hat{p},\hat{q}},\tilde{s}) \rightsquigarrow & \text{where} \\
 & \quad \langle\langle f(\tilde{t})\sigma_1 \overset{?}{\simeq}_{\emptyset} f(c_{\hat{x}+1,\hat{q}},\tilde{s})\sigma_1, & \quad \sigma_1=\{\overline{x}\leftarrow c_{\hat{p},\hat{x}}\}, \\
 & \qquad \sigma_1, \hat{p}\leq\hat{x}\wedge\hat{x}+1\leq\hat{q}\rangle, & \\
 & \quad \langle f(\overline{x},\tilde{t})\sigma_2 \overset{?}{\simeq}_{\emptyset} f(\tilde{s})\sigma_2, \sigma_2, true\rangle, & \quad \sigma_2=\{\overline{x}\leftarrow c_{\hat{p},\hat{q}},\overline{x}\}, \\
 & \quad \langle f(\tilde{t})\sigma_3 \overset{?}{\simeq}_{\emptyset} f(\tilde{s})\sigma_3, \sigma_3, true\rangle\rangle, & \quad \sigma_3=\{\overline{x}\leftarrow c_{\hat{p},\hat{q}}\}. \\
 & f(c_{\hat{p},\hat{q}},\tilde{t})\overset{?}{\simeq}_{\emptyset} f(\overline{x},\tilde{s}) \rightsquigarrow & \text{where} \\
 & \quad \langle\langle f(c_{\hat{x}+1,\hat{q}},\tilde{t})\sigma_1 \overset{?}{\simeq}_{\emptyset} f(\tilde{s})\sigma_1, & \quad \sigma_1=\{\overline{x}\leftarrow c_{\hat{p},\hat{x}}\}, \\
 & \qquad \sigma_1, \hat{p}\leq\hat{x}\wedge\hat{x}+1\leq\hat{q}\rangle, & \\
 & \quad \langle f(\tilde{t})\sigma_2 \overset{?}{\simeq}_{\emptyset} f(\overline{x},\tilde{s})\sigma_2, \sigma_2, true\rangle, & \quad \sigma_2=\{\overline{x}\leftarrow c_{\hat{p},\hat{q}},\overline{x}\}, \\
 & \quad \langle f(\tilde{t})\sigma_3 \overset{?}{\simeq}_{\emptyset} f(\tilde{s})\sigma_3, \sigma_3, true\rangle\rangle, & \quad \sigma_3=\{\overline{x}\leftarrow c_{\hat{p},\hat{q}}\}.
\end{array}$$

Fig. 7. Contraction rules for constant pattern simplification for *QP*-$\emptyset$-unification. $\tilde{t}$ and $\tilde{s}$ are possibly empty sequences of *QP*-terms. $f \in \mathcal{F}_{Flex}$.

each node, thus getting new successor nodes. The branch which ends with a node labeled by $\top$ is called a successful branch. The branch which ends with a node labeled by $\bot$ is a failed branch. All *QP*-$\emptyset$-unification problems attached to the nodes of a successful branch are in fact *P*-$\emptyset$-unification problems.

For each node in the tree, we compose substitutions (top-down) displayed on the edges of the branch which leads to this node and attach the obtained

$$\begin{array}{lll}
\textsf{SeparationC:} & f(c_{\hat r},\tilde t)\overset{?}{\simeq}_{\emptyset}f(c_{\hat p,\hat q},\tilde s)\rightsquigarrow & \\
& \quad\langle\langle f(\tilde t)\overset{?}{\simeq}_{\emptyset}f(c_{\hat p+1,\hat q},\tilde s), & \\
& \qquad \varepsilon, \hat p=\hat r\wedge \hat p+1\leq \hat q\rangle\rangle. & \\
& f(c_{\hat p,\hat q},\tilde t)\overset{?}{\simeq}_{\emptyset}f(c_{\hat r},\tilde s)\rightsquigarrow & \\
& \quad\langle\langle f(c_{\hat p+1,\hat q},\tilde t)\overset{?}{\simeq}_{\emptyset}f(\tilde s), & \\
& \qquad \varepsilon, \hat p=\hat r\wedge \hat p+1\leq \hat q\rangle\rangle. & \\
& f(c_{\hat p_1,\hat p_2},\tilde t)\overset{?}{\simeq}_{\emptyset}f(c_{\hat q_1,\hat q_2},\tilde s)\rightsquigarrow & \text{where} \\
& \quad\langle\langle f(\tilde t)\overset{?}{\simeq}_{\emptyset}f(c_{\hat p_2+1,\hat q_2},\tilde s), & \quad d_1=(\hat p_1=\hat q_1\wedge \hat p_2+1\leq \hat q_2) \\
& \qquad \varepsilon, d_1\rangle, & \\
& \quad\langle f(c_{\hat q_2+1,\hat p_2},\tilde t)\overset{?}{\simeq}_{\emptyset}f(\tilde s), & \quad d_2=(\hat p_1=\hat q_1\wedge \hat q_2+1\leq \hat p_2) \\
& \qquad \varepsilon, d_2\rangle, & \\
& \quad\langle f(\tilde t)\overset{?}{\simeq}_{\emptyset}f(\tilde s),\varepsilon,d_3\rangle\rangle, & \quad d_3=(\hat p_1=\hat q_1\wedge \hat p_2=\hat q_2).
\end{array}$$

Fig. 8. Separation rules for constant pattern simplification for QP-$\emptyset$-unification. $\tilde t$ and $\tilde s$ are possibly empty sequences of QP-terms. $f\in\mathcal{F}_{Flex}$.

$$\begin{array}{lll}
\textsf{FailureF:} & f(t_1,\tilde t)\overset{?}{\simeq}_{\emptyset}f(h_{\hat p,\hat q}(\tilde r),\tilde s)\rightsquigarrow\bot, & \text{if } t_1\notin\mathcal{V}_I\cup\mathcal{V}_S,\ head(t_1)\neq h. \\
& f(h_{\hat p,\hat q}(\tilde r),\tilde t)\overset{?}{\simeq}_{\emptyset}f(s_1,\tilde s)\rightsquigarrow\bot, & \text{if } s_1\notin\mathcal{V}_I\cup\mathcal{V}_S,\ head(s_1)\neq h.
\end{array}$$

Fig. 9. Failure rules for functional pattern simplification for QP-$\emptyset$-unification. $\tilde t$, $\tilde s$ and $\tilde r$ are possibly empty sequences of QP-terms. $f\in\mathcal{F}_{Fix}\cup\mathcal{F}_{Flex}$.

substitution to the node. The empty substitution is attached to the root. For a node N, the substitution attached to N in such a way is called the associated substitution of N.

Similarly, for each node in the tree, we take a conjunction of the linear Diophantine constraints displayed on the edges of the branch which leads to this node and attach the obtained constraint to the node. The linear Diophantine constraint $ldc(\mathfrak{Q}_{\emptyset})$ is attached to the root. For a node N, the constraint attached to N in such a way is called the associated constraint of N.

We call the tree a P-$\emptyset$-unification tree for $\mathfrak{Q}_{\emptyset}$ and denote it $putree(\mathfrak{Q}_{\emptyset})$.

Let $\Delta(\mathfrak{Q}_{\emptyset})$ be the set of all P-substitution/constraint pairs associated with the $\top$ nodes. Then we define the set $\Sigma(\mathfrak{Q}_{\emptyset})$ as follows:

$$\Sigma(\mathfrak{Q}_{\emptyset})=\{\sigma\mid \text{ there exists } \{\theta,d\}\in\Delta(\mathfrak{Q}_{\emptyset}) \text{ and } \mu \text{ such that } \mu \text{ is a positive integer solution of } d \text{ and } \sigma=(\theta\circ\mu)|_{vars(\mathfrak{U})}\}.$$

$$
\begin{array}{lll}
\mathsf{ContractionF:} & f(x,\tilde{t})\overset{?}{\simeq}_{\emptyset} f(h_{\hat{p},\hat{q}}(\tilde{r}),\tilde{s}) \leadsto & \text{if } x \notin ivars(h_{\hat{p},\hat{q}}(\tilde{r})),\\
& \quad \langle\langle f(\tilde{t})\sigma_1 \overset{?}{\simeq}_{\emptyset} f(\tilde{s})\sigma_1, \sigma_1, \hat{p}=\hat{q}\rangle, & \quad \sigma_1 = \{x \leftarrow h_{\hat{q}}(\tilde{r})\},\\
& \quad \langle f(\tilde{t})\sigma_2 \overset{?}{\simeq}_{\emptyset} f(h_{\hat{p}+1,\hat{q}}(\tilde{r}),\tilde{s})\sigma_2, & \quad \sigma_2 = \{x \leftarrow h_{\hat{p}}(\tilde{r})\}\\
& \qquad \sigma_2, \hat{p}+1 \le \hat{q}\rangle\rangle, & \\
& f(h_{\hat{p},\hat{q}}(\tilde{r}),\tilde{t})\overset{?}{\simeq}_{\emptyset} f(x,\tilde{s}) \leadsto & \text{if } x \notin ivars(h_{\hat{p},\hat{q}}(\tilde{r})),\\
& \quad \langle\langle f(\tilde{t})\sigma_1 \overset{?}{\simeq}_{\emptyset} f(\tilde{s})\sigma_1, \sigma_1, \hat{p}=\hat{q}\rangle, & \quad \sigma_1 = \{x \leftarrow h_{\hat{q}}(\tilde{r})\},\\
& \quad \langle f(h_{\hat{p}+1,\hat{q}}(\tilde{r}),\tilde{t})\sigma_2 \overset{?}{\simeq}_{\emptyset} f(\tilde{s})\sigma_2, & \quad \sigma_2 = \{x \leftarrow h_{\hat{p}}(\tilde{r})\}.\\
& \qquad \sigma_2, \hat{p}+1 \le \hat{q}\rangle\rangle, & \\
& f(\overline{x},\tilde{t})\overset{?}{\simeq}_{\emptyset} f(h_{\hat{p},\hat{q}}(\tilde{r}),\tilde{s}) \leadsto & \text{if } \overline{x} \notin svars(h_{\hat{p},\hat{q}}(\tilde{r})),\\
& \quad \langle\langle f(\tilde{t})\sigma_1 \overset{?}{\simeq}_{\emptyset} f(h_{\hat{x}+1,\hat{q}}(\tilde{r}),\tilde{s})\sigma_1, & \quad \sigma_1 = \{\overline{x} \leftarrow h_{\hat{p},\hat{x}}(\tilde{r})\},\\
& \qquad \sigma_1, \hat{p} \le \hat{x} \wedge \hat{x}+1 \le \hat{q}\rangle, & \\
& \quad \langle f(\overline{x},\tilde{t})\sigma_2 \overset{?}{\simeq}_{\emptyset} f(\tilde{s})\sigma_2, \sigma_2, true\rangle, & \quad \sigma_2 = \{\overline{x} \leftarrow h_{\hat{p},\hat{q}}(\tilde{r}), \overline{x}\},\\
& \quad \langle f(\tilde{t})\sigma_3 \overset{?}{\simeq}_{\emptyset} f(\tilde{s})\sigma_3, \sigma_3, true\rangle\rangle, & \quad \sigma_3 = \{\overline{x} \leftarrow h_{\hat{p},\hat{q}}(\tilde{r})\}.\\
& f(h_{\hat{p},\hat{q}}(\tilde{r}),\tilde{t})\overset{?}{\simeq}_{\emptyset} f(\overline{x},\tilde{s}) \leadsto & \text{if } \overline{x} \notin svars(h_{\hat{p},\hat{q}}(\tilde{r})),\\
& \quad \langle\langle f(h_{\hat{x}+1,\hat{q}}(\tilde{r}),\tilde{t})\sigma_1 \overset{?}{\simeq}_{\emptyset} f(\tilde{s})\sigma_1, & \quad \sigma_1 = \{\overline{x} \leftarrow h_{\hat{p},\hat{x}}(\tilde{r})\},\\
& \qquad \sigma_1, \hat{p} \le \hat{x} \wedge \hat{x}+1 \le \hat{q}\rangle, & \\
& \quad \langle f(\tilde{t})\sigma_2 \overset{?}{\simeq}_{\emptyset} f(\overline{x},\tilde{s})\sigma_2, \sigma_2, true\rangle, & \quad \sigma_2 = \{\overline{x} \leftarrow h_{\hat{p},\hat{q}}(\tilde{r}), \overline{x}\},\\
& \quad \langle f(\tilde{t})\sigma_3 \overset{?}{\simeq}_{\emptyset} f(\tilde{s})\sigma_3, \sigma_3, true\rangle\rangle, & \quad \sigma_3 = \{\overline{x} \leftarrow h_{\hat{p},\hat{q}}(\tilde{r})\}.
\end{array}
$$

Fig. 10. Contraction rules for functional pattern simplification for *QP*-$\emptyset$-unification. $\tilde{t}$, $\tilde{s}$ and $\tilde{r}$ are possibly empty sequences of *QP*-terms. $f \in \mathcal{F}_{Flex}$.

The next theorem shows that $\Sigma(\mathfrak{Q}_\emptyset)$ is a minimal complete set of *P*-$\emptyset$-unifiers of $\mathfrak{Q}_\emptyset$. This is the main result of this paper:

Theorem 4.1 $\Sigma(\mathfrak{Q}_\emptyset) = mcu_\emptyset(\mathfrak{Q}_\emptyset)$.[5]

We can observe that unification procedure terminates if one of the *P*-terms to be unified is ground. This yields to the following result:

Theorem 4.2 *Let* $\mathfrak{M}_\emptyset$ *be a general* P-$\emptyset$*-matching problem. Then the set* $\Delta(\mathfrak{M}_\emptyset)$ *is finite. If* $\mathfrak{M}_\emptyset$ *contains no patterns, then* $\Sigma(\mathfrak{M}_\emptyset)$ *is finite.*

The termination condition given in the theorem below requires for a prob-

[5] In fact, in [12] we have proved a stronger result: $\Sigma(\mathfrak{Q}_\emptyset)$ is a disjoint complete set of free unifiers of $\mathfrak{Q}_\emptyset$.

SeparationF:	$f(h_{\hat{r}}(\tilde{q}), \tilde{t}) \overset{?}{\simeq}_\emptyset f(h_{\hat{p},\hat{q}}(\tilde{r}), \tilde{s}) \rightsquigarrow$	where
	$\langle\langle f(h_{\hat{p}}(\tilde{q}), \tilde{t}) \overset{?}{\simeq}_\emptyset$	$d = (\hat{p} = \hat{r} \wedge$
	$f(h_{\hat{p}}(\tilde{r}), h_{\hat{p}+1,\hat{q}}(\tilde{r}), \tilde{s}), \varepsilon, d\rangle,$	$\hat{p} + 1 \le \hat{q}).$
	$f(h_{\hat{p},\hat{q}}(\tilde{r}), \tilde{t}) \overset{?}{\simeq}_\emptyset f(h_{\hat{r}}(\tilde{q}), \tilde{s}) \rightsquigarrow$	where
	$\langle\langle f(h_{\hat{p}}(\tilde{r}), h_{\hat{p}+1,\hat{q}}(\tilde{r}), \tilde{t}) \overset{?}{\simeq}_\emptyset$	$d = (\hat{p} = \hat{r} \wedge$
	$f(h_{\hat{p}}(\tilde{q}), \tilde{s}), \varepsilon, d\rangle,$	$\hat{p} + 1 \le \hat{q}).$
	$f(h_{\hat{p}_1,\hat{p}_2}(\tilde{r}), \tilde{t}) \overset{?}{\simeq}_\emptyset f(h_{\hat{q}_1,\hat{q}_2}(\tilde{q}), \tilde{s}) \rightsquigarrow$	where
	$\langle\langle f(h_{\hat{p}_2}(\tilde{r}), \tilde{t}) \overset{?}{\simeq}_\emptyset$	$d_1 = (\hat{p}_1 = \hat{q}_1 \wedge$
	$f(h_{\hat{p}_2}(\tilde{q}), h_{\hat{p}_2+1,\hat{q}_2}(\tilde{q}), \tilde{s}), \varepsilon, d_1\rangle,$	$\hat{p}_2 + 1 \le \hat{q}_2),$
	$\langle f(h_{\hat{q}_2}(\tilde{r}), h_{\hat{q}_2+1,\hat{p}_2}(\tilde{r}), \tilde{t}) \overset{?}{\simeq}_\emptyset$	$d_2 = (\hat{p}_1 = \hat{q}_1 \wedge$
	$f(h_{\hat{q}_2}(\tilde{q}), \tilde{s}), \varepsilon, d_2\rangle,$	$\hat{q}_2 + 1 \le \hat{p}_2),$
	$\langle f(h_{\hat{q}_2}(\tilde{r}), \tilde{t}) \overset{?}{\simeq}_\emptyset f(h_{\hat{q}_2}(\tilde{q}), \tilde{s}),$	$d_3 = (\hat{p}_1 = \hat{p}_2 \wedge$
	$\varepsilon, d_3\rangle\rangle,$	$\hat{q}_1 = \hat{q}_2).$

Fig. 11. Separation rules for functional pattern simplification for *QP*-$\emptyset$-unification. $\tilde{t}$, $\tilde{s}$, $\tilde{r}$ and $\tilde{q}$ are possibly empty sequences of *QP*-terms, $f \in \mathcal{F}_{Flex}$.

lem of the form $f(\overline{x}) \overset{?}{\simeq}_\emptyset f(t_1, \ldots, t_n)$, $n > 1$, to check whether $\overline{x}$ occurs in $f(t_1, \ldots, t_n)$. We call it *the last sequence variable occurrence checking* (*lsvoc*). We can tailor *lsvoc* into the tree generation process as follows: if in the tree a successor of the *QP*-$\emptyset$-unification problem of the form $f(\overline{x}) \overset{?}{\simeq}_\emptyset f(t_1, \ldots, t_n)$, $n > 1$, has to be generated, perform *lsvoc.* If $\overline{x}$ occurs in $f(t_1, \ldots, t_n)$, label the node with $\perp$, otherwise proceed in the usual way.

Theorem 4.3 *If $\mathfrak{Q}_\emptyset$ is a unification problem such that all sequence variables occurring in $\mathfrak{Q}_\emptyset$ are only the last arguments of the term they occur, then the unification procedure with lsvoc terminates.*

The fact that in most of the applications sequence variables occur precisely only at the last position in terms, underlines the importance of Theorem 4.3.

5 Conclusion

We considered a unification problem for an equational theory with sequence and individual variables, free constants, fixed and flexible arity function symbols and patterns and described a minimal complete unification procedure. Patterns abbreviate sequences of unknown lengths of terms matching certain "pattern". The unification procedure enumerates substitution/constraint pairs which constitute the minimal complete set of solutions of the problem. Two sufficient termination conditions have been established.

6 Acknowledgments

I wish to thank Prof. Bruno Buchberger for supervision and for many helpful discussions.

References

[1] F. Ajili and E. Contejean. Complete solving of linear Diophantine equations and inequations without adding variables. In U. Montanari and F. Rossi, editors, *Proceedings of the First International Conference on Principles and Practice of Constraint Programming*, volume 1949 of *Lecture Notes in Computer Science*, pages 1–17, Cassis, France, 1995. Springer Verlag.

[2] B. Buchberger. Mathematica as a rewrite language. In T. Ida, A. Ohori, and M. Takeichi, editors, *Proceedings of the 2nd Fuji International Workshop on Functional and Logic Programming)*, pages 1–13, Shonan Village Center, Japan, 1–4 November 1996. World Scientific.

[3] B. Buchberger. Personal communication, 2001.

[4] B. Buchberger, C. Dupre, T. Jebelean, F. Kriftner, K. Nakagawa, D. Vasaru, and W. Windsteiger. The Theorema project: A progress report. In M. Kerber and M. Kohlhase, editors, *Symbolic Computation and Automated Reasoning. Proceedings of Calculemus'2000*, pages 98–113, St. Andrews, UK, 6–7 August 2000.

[5] A. Colmerauer. An introduction to Prolog III. *CACM*, 33(7):69–91, 1990.

[6] M. R. Genesereth. Epilog for Lisp 2.0 Manual. Epistemics Inc., Palo Alto, US, 1995.

[7] M. R. Genesereth and R. E. Fikes. Knowledge Interchange Format, Version 3.0 Reference Manual. Technical Report Logic-92-1, Computer Science Department, Stanford University, Stanford, US, June 1992.

[8] S. Ginsburg and X. S. Wang. Pattern matching by Rs-operations: Toward a unified approach to querying sequenced data. In *Proceedings of the 11th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 293–300, San Diego, US, 2–4 June 1992.

[9] G. Grahne and E. Waller. How to make SQL stand for string query language. In R. Connor and A. Mendelzon, editors, *Research Issues in Structured and Semistructured Database Programming. Proceedings of the 7th International Workshop on Database Programming Languages, DBPL'99*, volume 1949 of *Lecture Notes in Computer Science*, pages 61–79, Kinloch Rannoch, UK, 1–3 September 2000. Springer Verlag.

[10] M. Hamana. Term rewriting with sequences. In *Proceedings of the First International Theorema Workshop*, Hagenberg, Austria, 9–10 June 1997.

[11] D. E. Knuth and P. B. Bendix. Simple word problems in universal algebras. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–298, Oxford, 1967. Pergamon Press. Appeared 1970.

[12] T. Kutsia. Solving and proving in equational theories with sequence variables and flexible arity symbols. PhD Thesis. Available from `http://www.risc.uni-linz.ac.at/people/tkutsia/pub/Thesis.pdf`, 2002.

[13] T. Kutsia. Unification with sequence variables and flexible arity symbols and its extension with pattern-terms. In J. Calmet, B. Benhamou, O. Caprotti, L. Henocque, and V. Sorge, editors, *Artificial Intelligence, Automated Reasoning and Symbolic Computation. Proceedings of Joint AICS'2002 and Calculemus'2002 Conferences*, volume 2385 of *Lecture Notes in Artificial Intelligence*, pages 290–304, Marseille, France, 1–5 July 2002. Springer Verlag.

[14] G. S. Makanin. The problem of solvability of equations on a free semigroup. *Math. USSR Sbornik*, 32(2), 1977.

[15] G. Mecca and A. J. Bonner. Sequences, Datalog and transducers. In *Proceedings of the Fourteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 23–35, San Jose, US, 22–25 May 1995.

[16] K. U. Schulz. Word unification and transformation of generalized equations. *J. Automated Reasoning*, 11(2):149–184, 1993.

[17] M. Widera and C. Beierle. A term rewriting scheme for function symbols with variable arity. Technical Report 280, Prakt. Informatik VIII, FernUniversität Hagen, Germany, 2001.

[18] S. Wolfram. *The Mathematica Book*. Cambridge University Press and Wolfram Research, Inc., fourth edition, 1999.