Bounded Higher-order Unification Using Regular Terms

Tomer Libal

Microsoft Research - Inria Joint Center, École Polytechnique, Palaiseau, France
tomer.libal@inria.fr

Abstract

We present a procedure for the bounded unification of higher-order terms [22]. The procedure extends G. P. Huet's pre-unification procedure [11] with rules for the generation and folding of regular terms. The concise form of the procedure allows the reuse of the pre-unification correctness proof. Furthermore, the regular terms can be restricted in order to decide the unifiability problem. Finally, the procedure avoids re-computation of terms in a non-deterministic search which leads to a better performance in practice when compared to other bounded unification algorithms.

Contents

1 Introduction

The unification principle has many uses in Computer Science. Due to the undecidability of the higher-order unification problem, many applications find it necessary to restrict the use of unification to decidable classes only. This can either be achieved by applying unification on fragments of higher-order logic problems, whose unifiability is known to be decidable or by restricting unification procedures to search for an incomplete set of unifiers. Among the fragments of the first we can find higher-order pattern unification [16, 18] and decidable subclasses of context unification [19, 6, 14, 21]. When we need to consider arbitrary higher-order unification problems, as is the case in higher-order resolution, we must search for an incomplete set of unifiers.

Most higher-order theorem provers, such as Isabelle [17], TPS [2] and LEO II [4], rely on Huet's pre-unification procedure [11] for the unification of higher-order terms. Since the

procedure does not terminate, these theorem provers must search for incomplete finite sets of unifiers only. The most common way to obtain such a set is by bounding the depth of the terms in the co-domain of the unifiers. The next example, based on a similar example in [12] , gives a family of simple unification problems where the depth of terms in the co-domain of unifiers grows exponentially while the size of the problems grows linearly.

Example 1.1. *The following monadic (right-associative) second-order equation has a unique unifier σ such that $\sigma(X_1) = a^{3^n}$ but the size of the problem is only $6n + 6$.*

$$\{X_1 a b X_1 b X_2 b \dots b X_n c \doteq a X_1 b X_2 X_2 X_2 b \dots b X_n X_n X_n b a a a c\} \tag{1}$$

Another approach for obtaining incomplete sets of unifiers is by bounding the number of occurrences of bound variables in the co-domains of unifiers. This approach, called bounded higher-order unification [22], gives a more refined incompleteness as can be shown with regard to the example above, which has only two bound variable occurrences per variable (one in the binder and one in the term). The drawback of this method is that the computed (incomplete) set of pre-unifiers is now infinite although it was shown that the unifiability problem is decidable [22].

When we are concerned not only with the unifiability problem, but with all most general unifiers, such an approach is not enough and we have to revert to the non-terminating pre-unification procedure. This problem arises in some applications of unification, most notably in the resolution calculus, where both a search for all most general unifiers and a terminating unification procedure are desired properties.

In this paper we present a procedure that captures these two properties. First, we enumerate all pre-unifiers by the use of regular terms using the Kleene star. Second, we can force the procedure to terminate by fixing the number of iterations allowed over the Kleene star, thus obtaining, not only minimal unifiers, but in practice any possible pre-unifier.

Even when we are concerned with unifiability only, the procedure presented in this paper is of a compact form, that smoothly extends Huet's pre-unification procedure. This allows us, not only to give a simpler procedure than current existing ones, but also simpler correction proofs.

The procedure presented in this paper takes many ideas from the algorithm for bounded higher-order unification presented in [22] but differs both in its ability to enumerate all pre-unifiers and its simpler correctness proofs and in the use it makes of regular expressions in order to replace cycles. This characteristic of the procedure will be discussed further in the conclusion.

Other similar works includes the finite representation of all unifiers for sub-classes of the string unification problem by using graphs and regular expressions [1] and the description of first-order cycles using finite automata [13].

The paper is organized as follows. In the first section we give some definitions and notations which will be used throughout the paper. At the end of this section we give an adaptation of Huet's pre-unification procedure to bounded higher-order unification problems. The second section is dedicated to the presentation of the regular terms and a pre-unification procedure based on them. The correctness of the procedure is given in the third section. In the fourth section we present a terminating variation of the procedure from the previous section. In the last section we conclude the paper and suggest some future work.

2 Preliminaries

2.1 Typed Lambda Calculus

In this section we will present the logical language that will be used throughout the paper. The language is a version of Church's simple theory of types [5] with an η-conversion rule as presented in [3] and [23] and with implicit α-conversions. Most of the definitions in this section are adapted from [23].

Let $\mathfrak{T}_o$ be a set of basic types, then the set of types $\mathfrak{T}$ is generated by $\mathfrak{T} := \mathfrak{T}_o | \mathfrak{T} \to \mathfrak{T}$. Let Σ be a signature of function symbols and let $\mathfrak{V}$ be a countably infinite set of variable symbols. The function $\mathtt{ar}$ denotes the *arity* of each function symbol and variable according to its type in the usual way. *Variables* are normally denoted by the letters x, y, z and *function symbols* by the letters f, g, h. We sometimes use subscripts and superscript as well. We sometimes add a superscript to symbols in order to specify their type. In examples containing at most second-order variables we will denote those variable using the capitalized letters X, Y, Z. The set $\mathtt{Term}^\alpha$ of terms of type α is generated by $\mathtt{Term}^\alpha := f^\alpha | x^\alpha | \lambda x^\beta.\mathtt{Term}^\gamma | \mathtt{Term}^{\beta\to\alpha}(\mathtt{Term}^\beta)$ where $f \in \Sigma, x \in \mathfrak{V}$ and $\alpha = \beta \to \gamma$. We will sometimes omit brackets in applications when the meaning is clear. The set $\mathtt{Term}$ denotes the set of all terms. The function τ maps terms to their types. *Subterms* and *positions* are defined as usual. We denote the subterm of t at position p by $t|_p$. *Bound* and *free variables* are defined as usual. Given a term t, we denote by $\mathtt{hd}(t)$ its *head symbol* and distinct between *flex* terms, whose head is a free variable and *rigid* terms, whose head is a function symbol or a bound variable. *Rigid positions* are positions such that no flex subterm is in a strict prefix position. We denote by $t[s]_p$ the term obtained from term t by replacing its subterm at position p with the term s. We sometimes denote the fact that s is a subterm of t by $t[s]$. The *depth* of a term t, denoted by $\mathtt{d}(t)$, is the size of the maximal rigid position in t.

Substitutions and their *composition* ($\circ$) are defined as usual. We denote by $\sigma|_W$ the substitution obtained from substitution σ by restricting its domain to variables in W. We extend the application of substitutions to terms in the usual way and denote it by postfix notation. Variables capture is avoided by implicitly renaming variables to fresh names upon binding. A substitution σ is *more general* than a substitution θ if there is a substitution δ such that $\sigma \circ \delta = \theta$.

A term t *subsumes* a term s, denoted $t \leq_s s$ if there is a substitution σ such that $t\sigma = s$. Given a term t and a set of terms S, we denote by $t \in_s S$, the fact that there is a term in S which subsumes t.

We assume that all the terms considered in this paper, unless specified otherwise, are in β-normal and η-expanded forms [23]. We further assume that all substitutions are idempotent [24] and contain only terms in β-normal and η-expanded forms in their co-domain. This allows us to deal with normal forms implicitly (see [23] for more information). Equality between terms is always assumed to be α-equality.

We introduce also a vector notation $\overline{t_n}$ for the sequence of terms $t_1, \ldots, t_n$. We will sometimes refer to the position i of a term s in the sequence by $t_1, \ldots, s_{@i}, \ldots, t_n$.

2.2 Contexts and Pre-unification

The majority of the definitions in this section are taken from [23] and [22]. Since some of the definitions in this section are given in an intuitive rather than formal presentation, we recommend to refer to these sources for a better understanding of these definitions.

The set $\texttt{Context}_\alpha \subset \texttt{Term}^{\alpha\to\alpha}$ of contexts of type α contains all terms $\lambda z^\alpha.s^\alpha[z]_p$ where z occurs in s exactly once. We call such terms *contexts* and denote them by $s[[.]]_p$ where [.] is considered as the "hole" of the term. We denote by $\texttt{mpath}(C)$ the *main path* of the context C which is the position of the hole in the context C. Contexts of the same type can be composed and can denote prefixes of terms. For example, the repeated composition of the context $C = ff([.])$, denoted by C^k, is a prefix of the term $\lambda z.fffffgz$ for $0 \leq k < 3$

Unification problems (or systems) are sets of terms $t \doteq s$, called *equations*, where t and s are of the same type. Based on whether the terms in an equation are flex or rigid, we make a distinction between *flex-flex*, *flex-rigid* and *rigid-rigid* equations. Systems are considered closed under symmetry of $\doteq$.

A substitution σ *unifies* an equation $t \doteq s$ if $t\sigma = s\sigma$. It unifies a system if it unifies all its equations. We denote the *set of all unifiers* of a system S by $\texttt{Unifiers}(S)$. Let $\widetilde{=}$ be the least congruence relation on $\texttt{Term}$ which contains $\{(t,s) \mid \texttt{hd}(t), \texttt{hd}(s) \in \mathfrak{V}\}$. A substitution σ *pre-unifies* an equation $t \doteq s$ if $t\sigma \widetilde{=} s\sigma$. It pre-unifies a system if it pre-unifies all its equations. The completing substitution ξ_S for a system S maps every two variables in S of the same type to the same fresh variable. It is simple to prove that if σ pre-unifies a system S, then $\sigma \circ \xi$ unifies S [23]. A *complete set of pre-unifiers* for a system S, denoted by $\texttt{PreUnifiers}(S)$, is a set of substitutions such that $\{\sigma \circ \xi_S \mid \sigma \in \texttt{PreUnifiers}(S)\} \subseteq \texttt{Unifiers}(S)$ and for every $\theta \in \texttt{Unifiers}(S)$ there exists $\sigma \in \texttt{PreUnifiers}(S)$ such that $\sigma|_{dom(\theta)} \leq \theta$.

Cycles in systems are, informally, sequences of flex-flex or flex-rigid equations where the same variable occurs in head position and in some rigid position in any two sequential equations and in the last and first equations. These variables are called the cycle's variables. An example is the cycle $xa \doteq y, y \doteq zb, za \doteq fxa$ whose cycle's variables are x, y and z. A formal definition is given in [22]. A cycle having exactly one none flex-flex equation is called a *standard cycle*. A standard cycle having at most one occurrence of any of its cycle's variables in rigid positions in each side of an equation is called a *unique standard cycle*. We can regard the position of the cycle's variable in the rigid term as the hole of a context. The cycle given above is a unique standard cycle containing the rigid term fxa. We can consider this term as the context $f([.])$ (i.e. $\lambda z.fz$). For each unique standard cycle, there is exactly one such context, which we call the *cycle's context* and denote it by $\texttt{ccon}(c)$ for a cycle c. The *number of equations* in a cycle is denoted by $\texttt{size}(c)$.

Let $\lambda\texttt{size}(t)$ be the number of occurrences of λ-binders and bounded variables in t. The function $\hat{b}$ maps the variables of a system to natural numbers, a $\hat{b}$-bounded (pre-)unifier of a system S is a (pre-)unifier σ of S such that for all $x \in \texttt{dom}(\sigma)$, $\lambda\texttt{size}(\sigma(x)) \leq \hat{b}(x)$. Bounded (pre-)unification is the search for $\hat{b}$-bounded (pre-)unifiers for a given $\hat{b}$. A procedure which decides the unifiability of bounded higher-order unification problems can be found in [22]. For a given bounding function $\hat{b}$, we denote the complete set of bounded pre-unifiers of S by $\texttt{PreUnifiers}^b(S)$.

An equation $x \doteq t$ in η-normal form is called *solved* in system S if x does not occur elsewhere in S and $\lambda\texttt{size}(t) \leq \hat{b}(x)$. We call x a solved variable in S. An equation is *pre-solved* in a system S if it is either solved in S or flex-flex. We denote by σ_S the substitution obtained from mapping x to t in all solved equations in S.

Before presenting a version of Huet's procedure for bounded unification and the notion of bounded partial bindings on which it depends, we will repeat the definition of partial bindings as given in [23].

Definition 2.1 (Partial bindings)**.** A partial binding of type $\alpha_1 \to \ldots \to \alpha_n \to \beta$ where $\beta \in \mathfrak{T}_o$ is a term of the form
$\lambda\overline{y_n}.a(\lambda\overline{z^1_{p_1}}.x_1(\overline{y_n}, \overline{z^1_{p_1}}), \ldots, \lambda\overline{z^m_{p_m}}.x_m(\overline{y_n}, \overline{z^m_{p_m}}))$ for some atom a where

- $\tau(y_i) = \alpha_i$ for $0 < i \leq n$.
- $\tau(a) = \gamma_1 \to \ldots \to \gamma_m \to \beta$ where $\gamma_i = \delta_1^i \to \ldots \to \delta_{p_i}^i \to \gamma_i'$ for $0 < i \leq m$.
- $\tau(z_j^i) = \delta_j^i$ for $0 < i \leq m$ and $0 < j \leq p_i$.
- x_i is a fresh variable and $\tau(x_i) = \alpha_1 \to \ldots \to \alpha_n \to \delta_1^i \to \ldots \to \delta_{p_i}^i \to \gamma_i'$ for $0 < i \leq m$.
- $\gamma_1', \ldots, \gamma_m' \in \mathfrak{T}_o$.

Partial bindings fall into two categories, imitation bindings, which for a given atom a and type α, are denoted by $\mathtt{PB}(a, \alpha)$ and projection bindings, which for a given index $0 < i \leq n$ and a type α, are denoted by $\mathtt{PB}(i, \alpha)$ and in which the atom a is equal to the bound variable y_i. Since partial bindings are uniquely determined by a type and an atom (up to renaming of the fresh variables $\overline{x_m}$), this defines a particular term.

Definition 2.2 (Bounded partial bindings)**.** Let $\lambda\overline{y_n}.a(\lambda\overline{z_{p_1}^1}.x_1(\overline{y_n}, \overline{z_{p_1}^1}), \ldots, \lambda\overline{z_{p_m}^m}.x_m(\overline{y_n}, \overline{z_{p_m}^m}))$ be either the imitation binding $\mathtt{PB}(f, \alpha)$ for function symbol f ($a = f$) and type α or the i^{th} projection binding $\mathtt{PB}(i, \alpha)$ ($a = y_i$) for type α. Then, the bounded imitation ($\mathtt{PB}^b(f, \alpha)$) and i^{th} projection ($\mathtt{PB}^b(i, \alpha)$) bindings for a given function $\hat{b}$, are the respective imitation and i^{th} projection bindings where in addition we update $\hat{b}$ for the newly introduced variables $x_1, \ldots, x_m$ (and we denote the replaced variable by x) as follows:

- for all $0 < i \leq m$, $n \leq \hat{b}(x_i) \leq \hat{b}(x)$.
- if a is a function symbol or a free variable (an imitation binding), then $\Sigma_{0<i\leq m}(\hat{b}(x_i) - n) \leq \hat{b}(x) - n$.
- if $a = y_i$ for $0 < i \leq n$ is a bound variable (a projection binding), then $\Sigma_{0<i\leq m}(\hat{b}(x_i) - n) \leq \hat{b}(x) - n - 1$.

Given a function $\hat{b}$, the bounded pre-unification procedure $\mathtt{PUA}_B$ is given in Fig. 1.

$$\frac{S \cup \{A \doteq A\}}{S}\ \texttt{(Delete)} \qquad \frac{S \cup \{\lambda\overline{z_k}.f(\overline{s_n}) \doteq \lambda\overline{z_k}.f(\overline{t_n})\}}{S \cup \{\lambda\overline{z_k}.s_1 \doteq \lambda\overline{z_k}.t_1, \ldots, \lambda\overline{z_k}.s_n \doteq \lambda\overline{z_k}.t_n\}}\ \texttt{(Decomp)}$$

$$\frac{S \cup \{\lambda\overline{z_k}.x(\overline{z_k}) \doteq \lambda\overline{z_k}.t\} \qquad x \notin \mathtt{FV}(t), \sigma = [\lambda\overline{z_k}.t/x]}{S\sigma \cup \{x \doteq \lambda\overline{z_k}.t\}}\ \texttt{(Bind)}$$

$$\frac{S \qquad \lambda\overline{z_k}.x^\alpha(\overline{s_n}) \doteq \lambda\overline{z_k}.f(\overline{t_m}) \in S, u \in \mathtt{PB}^b(f, \alpha), \sigma = [u/x]}{S\sigma \cup \{x \doteq u\}}\ \texttt{(Imitate)}$$

$$\frac{S \qquad \lambda\overline{z_k}.x^\alpha(\overline{s_n}) \doteq \lambda\overline{z_k}.a(\overline{t_m}) \in S, 0 < i \leq k, u = \mathtt{PB}^b(i, \alpha), \sigma = [u/x]}{S\sigma \cup \{x \doteq u\}}\ \texttt{(Project)}$$

Figure 1: $\mathtt{PUA}_B$- Huet's pre-unification rules for bounded unification

This procedure is a straightforward adaptation to bounded problems of Huet's pre-unification procedure as presented in [23] with the additional modification that the application of `(Bind)` after calling `(Imitate)` and `(Project)` is done implicitly within the rules `(Imitate)` and `(Project)` instead of being called explicitly. Therefore, the following theorems from [23] apply also to this procedure.

Theorem 2.3 (Soundness)**.** If S' is obtained from a unification system S using $\texttt{PUA}_B$ and $\hat{b}$ and is in pre-solved form, then $\sigma_{S'}|_{\texttt{FV}(S)} \in \texttt{PreUnifiers}^b(S)$.

Theorem 2.4 (Completeness)**.** If θ is a $\hat{b}$-bounded pre-unifier of a unification system S, then there exists a pre-solved system S', which is obtainable from S using $\texttt{PUA}_B$ and $\hat{b}$ such that $\sigma_{S'}|_{\texttt{FV}(S)} \leq \theta$.

Remark 2.5. The procedure $\texttt{PUA}_B$ contains two kinds of non-determinism. On the one hand, we need to choose an equation at each step and on the other, we need to choose which rule to apply to it. In [23] it is argued that completeness is only affected by the second kind of non-determinism and more precisely, by the choice between the `(Imitate)` and `(Project)` rules. We will use this fact in the rest of the paper and allow ourselves to choose specific equations to process without harming completeness.

3 Pre-unification

The procedure $\texttt{PUA}_B$ given in Fig. 1 may not terminate. A classic example is the unification problem $xfa \doteq gxa$, which is non-unifiable for any $\hat{b}$ but the execution of $\texttt{PUA}_B$ will not terminate.

In this section we will introduce two kinds of tools for dealing with bounded higher-order unification problems. We will introduce bounds for dealing with "acyclic" components of the problem and we will introduce regular terms for dealing with the "cyclic" components of the problem.

Once we have introduced the bounds and regular terms, we would like to prove the soundness and completeness of our procedure (which will be introduced later in this section). The proof of completeness will be by induction on a measure over systems. We will introduce this measure now.

Definition 3.1 (Bounding measure [22])**.** Let S be a system. The bounding measure of S for $\hat{b}$, which is denoted by `b-measure`, is the multiset $\{\hat{b}(x) - \texttt{ar}(x) | x \in V\}$ where V is the set of unsolved variables in S. Multi-sets are ordered according to multi-sets ordering [7].

By considering the function $\texttt{PB}^b$ we can see that any choice of a projection partial binding strictly decreases the bounding measure while the application of no other rule increases it. By considering also the bounds defined next, we will be able to show the decrease of the bounding measure.

3.1 First-order Bounds

When we disallow the application of the `(Project)` rule on an acyclic problem, any run of the procedure can be simulated, in a sense, by a run of a first-order unification algorithm. The following definition is motivated by the fact that the depth of terms generated during the execution of first-order unification algorithms is bounded.

Definition 3.2 (Maximum depths)**.** Given a system S, we denote the maximum size $\texttt{d}(t)$ for all terms t in S by $\texttt{md}(S)$.

Definition 3.3 (First-order bound)**.** Given a system S, its first-order bound is $(k+1) \cdot \texttt{md}(S)$, where k is the number of unsolved variables in S. It is denoted by $\texttt{fbound}(S)$.

3.2 Regular Terms and Contexts

Using the first-order bound, we can simulate a first-order unification algorithm on acyclic higher-order systems. This approach will not work for cyclic problems and therefore we would like to eliminate cycles from systems. We will introduce next regular contexts and terms which will serve as bounds to the possible forms of contexts and terms.

Definition 3.4 (Regular contexts)**.** The set $\texttt{Context}_r^\alpha$ extends the set $\texttt{Context}_\alpha$ and is defined inductively by

- if $C \in \texttt{Context}_\alpha$ then $C \in \texttt{Context}_r^\alpha$.
- if $C \in \texttt{Context}_\alpha$ and $D \in \texttt{Context}_r^\alpha$ then $C(D) \in \texttt{Context}_r^\alpha$.
- if $C \in \texttt{Context}_\alpha$ and $D \in \texttt{Context}_r^\alpha$ then $C^*(D) \in \texttt{Context}_r^\alpha$ where $*$ is the Kleene star.

The set of all regular contexts is defined as the union of $\texttt{Context}_r^\alpha$ for all $\alpha \in \mathfrak{T}$.

Definition 3.5 (Regular terms)**.** The set $\texttt{Term}_r^n$ for a given arity n is defined as $\lambda\overline{z_n}.C(t)$ where

- C is a regular context.
- the bound variables z_i for $0 < i \leq n$ cannot occur in the scope of starred sub-contexts of C.
- t is a term which may contain occurrences of the bound variables $z_1, \ldots, z_n$.

Example 3.6. *The following term is regular:* $\lambda z.(f([.], w))^*(f([.], z))(f([.], a))^*(f(z, a))$

We will now show that some variables occurring in cycles must be mapped, in pre-unifiers, to a form which can be described by regular terms. For the rest of this section we deal only with unique standard cycles. We will show how one can transform a arbitrary cycle to this form in Sec. 4.

We first define the set of regular terms for a given context and variable.

Definition 3.7 ($\texttt{reg}$)**.** Given a variable x and a context C, the finite set of regular terms is defined as $\texttt{reg}(x, C) = \{\lambda\overline{z_m}.C^*C'(t(\overline{z_m}))\}$ such that

1. $C = C'f(v_1, \ldots, C''_{@k}, \ldots, v_n)$ for some $C'', f, v_1, \ldots, v_n$.
2. $\tau(x) = \alpha_1 \to \ldots \to \alpha_m \to \beta$ for some β and $\tau(z_i) = \alpha_i$ for $0 < i \leq m$.
3. we have one of the followings:
   (a) $t \in \texttt{PB}^b(f, \tau(x))$ and $\hat{b}(x_i) < \hat{b}(x)$ for all new variables x_i which were introduced by the function $\texttt{PB}^b$.
   (b) there is $0 < i \leq m$ such that $t \in \texttt{PB}^b(i, \tau(x))$.

Example 3.8. *The members of the set* $\texttt{reg}(x, hg(ef(y, [.]), b))$ *for* $\tau(x) = \alpha \to \alpha$ *are given in the following list:*

$\lambda z.(hg(ef(y,[.]),b))^*h(x_1(z))$
$\lambda z.(hg(ef(y,[.]),b))^*z$
$\lambda z.(hg(ef(y,[.]),b))^*(hg(x_1(z),x_2(z)))$
$\lambda z.(hg(ef(y,[.]),b))^*hz$
$\lambda z.(hg(ef(y,[.]),b))^*(hg(e(x_1(z)),b))$
$\lambda z.(hg(ef(y,[.]),b))^*(hg(z,b))$
$\lambda z.(hg(ef(y,[.]),b))^*(hg(ef(x_1(z),x_2(z)),b))$
$\lambda z.(hg(ef(y,[.]),b))^*(hg(e(z),b))$

where $0 < \hat{b}(x_1), \hat{b}(x_2) < \hat{b}(x)$.

For a given context and variable, the following infinite set contains all the instantiations of their regular terms.

Definition 3.9 (Instantiations of regular terms ($\mathtt{insts}_1$))**.** Given a variable x and a context C, we define the infinite set $\mathtt{insts}_1(x,C) = \{t' | t \in \mathtt{reg}(x,C)\}$ such that

- t' is obtained from t by replacing each Kleene star $*$ with some $k \geq 0$.

An important proof idea which is taken from [22] is that of a maximal context with no bound variables occurrences.

Definition 3.10 (Maximal contexts)**.** Let x be a variable and σ a substitution, then D is called a maximal context of $\sigma(x)$ if D is a maximal prefix of $\sigma(x)$ such that it does not contain bound variable occurrences. If there is a context C such that D is of the form $(C\sigma)^l C'$ for some $l \geq 0$ and C' a prefix of $C\sigma$ then D is called a maximal context of $\sigma(x)$ for C.

Example 3.11. *Let* $\sigma = [\lambda z.f(g(f(z,f(z,a))),f(a,a))/x,\ f(a,a)/y]$ *then the maximal context for* x *and* σ *is* $f(g([.]),f(a,a))$*. This is a maximal context for the context* $f(g([.]),y)$ *but not for* $f([.],f(a,a))$ *or* $f(g(f([.],a)),f(a,a))$*.*

The first property we will prove for the case $\sigma(x)$ has a maximal context for a context C is that $\sigma(x)$ is subsumed by $\mathtt{insts}_1(x,C)$.

Lemma 3.12. Let $\lambda\overline{z_{m_1}}.x_1(\overline{t^1_{n_1}}) \doteq \lambda\overline{z_{m_1}}.x_2(\overline{s^1_{n_2}}), \ldots, \lambda\overline{z_{m_k}}.x_k(\overline{t^k_{n_k}}) \doteq \lambda\overline{z_{m_k}}.C[x_1(\overline{s^k_{n_1}})]$, be a cycle in system S with cycle context C and σ a pre-unifier for S. Let $\sigma(x_i) = \lambda\overline{z_n}.Dt$ for some index $0 < i \leq k$, a context D and a term t. If D is a maximal context for C then $\sigma(x_i) \in_s \mathtt{insts}_1(x_i,C)$.

Proof. Since D is a maximal context for C, we have $D = (C^l C')\sigma$ for $l \geq 0$. We need now to show that the requirements of Def. 3.7 hold. The first two requirements hold by assumption. We now consider the following three cases:

- $t = f(v_1,\ldots,v_r)$ and there are at least two indices $0 < i_1, i_2 \leq r$ such that $\lambda\mathtt{size}_r(v_{i_1})$, $\lambda\mathtt{size}_r(v_{i_2}) > 0$. Let $t' = \mathtt{PB}^b(f,\tau(x_i))$ such that $t'\theta = \lambda\overline{z_n}.t$ for some θ, then t' must contain at least two new variables, y_{i_1}, y_{i_2}, such that $\hat{b}(y_{i_1}), \hat{b}(y_{i_2}) > n$ and we satisfy requirement 3a. Therefore, $\lambda\overline{z_n}.Dt'(\overline{z_n}) \in \mathtt{insts}_1(x_i,C)$ and $\sigma(x_i) \in_s \mathtt{insts}_1(x_i,C)$.
- $\mathtt{hd}(t) = z_r$ for $0 < r \leq k$ and let $t' \in \mathtt{PB}^b(r,\tau(t))$ such that $t'\theta = \lambda\overline{z_n}.t$ for some θ and we satisfy requirement 3b and have $\sigma(x_i) \in_s \mathtt{insts}_1(x_i,C)$.
- otherwise $t = f(v_1,\ldots,v_r)$ where we have only one index $0 < j \leq r$ such that $\lambda\mathtt{size}_r(v_j) > 0$. In this case D is not maximal as it should include f as well and we get a contradiction.

Note that we did not consider the case $\lambda\mathtt{size}_r(t_i) = 0$ for all $0 < j \leq \mathtt{ar}(f)$. A simple counting of the symbols occurring on each side of the unification constraint will give us that σ is not a pre-unifier in this case. □

We will now prove, for unique standard cycles containing only one equation, that we can give a regular term which describes the mappings of the cyclic variable in all ground unifiers of the system.

Lemma 3.13. Given a unique standard cycle $\{\lambda\overline{y_m}.x(\overline{t_n}) \doteq \lambda\overline{y_m}.C[x(\overline{s_n})]\}$ in system S, then for every ground unifier σ of S, $\sigma(x) \in_s \mathtt{insts}_1(x, C)$.

Proof. Let $\sigma(x) = \lambda\overline{z_n}.C^0[t]$ such that C^0 is the maximal context. Let k be the depth of $\sigma(x)$ and let A be the greatest common prefix of $(C^{k+1})\sigma$ and C^0. Then $A = (C\sigma)^l(C')$ for some $l \leq k$ where C' is a proper prefix of $C\sigma$ and let C'' be a context such that $C'(C'') = C\sigma$. We will first prove that $A = C^0$. Assume otherwise, then A must be a proper prefix of C^0 and therefore, $C^0 = A(f(t_1, \ldots, D_{@k}, \ldots, t_n))$ for some context D where $A(f(t_1, \ldots, [.]_{@k}, \ldots, t_n))$ is not a prefix of $(C^{k+1})\sigma$. Applying σ to the unification constraint, we get

$$\lambda\overline{y_m}.A(f(t_1, \ldots, t'_k, \ldots, t_n)) = \lambda\overline{y_m}.(C\sigma)(A(f(t_1, \ldots, t''_k, \ldots, t_n))),$$

where t'_k, t''_k are terms. Applying `(Decomp)`s we get

$$\lambda\overline{y_m}.f(t_1, \ldots, t'_k, \ldots, t_n) \doteq \lambda\overline{y_m}.C''(C'(f(t_1, \ldots, t''_k, \ldots, t_n)))$$

Now we consider the head component of the main path of C''. If it is k, then we get a contradiction to the maximality of A as σ is a ground unifier and A should include f as well. Otherwise, it is $l \neq k$ and let $C''[.] = f(s_1, \ldots, D'_{@l}, \ldots, s_n)$, then we get from the constraint, after one `(Decomp)`, that

$$t_l \doteq D'(C'(f(t_1, \ldots, t_l, \ldots, t_n)))$$

which is a contradiction to the unifiability of the pair as the positions of the holes in D' and C' are rigid (according to the definition of standard cycles). We therefore assume that $A = C^0$ and by Lemma 3.12, we get that $\sigma(x) \in_s \mathtt{insts}_1(x, C)$ □

The case is more complex for bigger standard cycles. The following example for $\hat{b}(x_1) = \hat{b}(x_2) = 3$:

$$\{x_1 f(a, a) \doteq x_2 a, x_2 b \doteq f(x_1 a, f(b, b))\} \tag{2}$$

demonstrates this as the ground unifier given next maps the two variables to terms which are not described as above.

$$[\lambda z.f(f(z, z), f(a, a))/x_1, \lambda z.f(f(f(a, a), f(a, a)), f(z, z))/x_2] \tag{3}$$

The reason for that is the possible "derailing" [22] of the bound variables in the ground unifiers from the hole in the cycle's context.

The following definitions will take this derailing into account when considering larger standard cycles.

Definition 3.14 (Derailing)**.** Given a context C let p be a prefix of its main path such that $\mathtt{hd}(C|_p) = f$ and $\mathtt{ar}(f) = n > 1$ and let $0 < k \leq n$ be the head component of the main path of $C|_p$. Then, the context $C[f(y_1, \ldots D_{@k}, \ldots, y_n)]_p$ is called a derailing of C where $D = C|_{p.k}$ and the y_i for $0 < i \leq n$ are new first-order variables. We call the context $C[f(y_1, \ldots, [.]_{@k}, \ldots, y_n)]_p$ the pre-context of the derailing and the context D the post-context of the derailing. We denote by $\mathtt{one\text{-}derail}(C)$ the set of all possible derailings of a context C. This set always includes also the empty derailing, i.e. the context C itself. In this case the pre-context is empty and the post-context is equal to C. Note that this set is finite and its size is equal to the number of non-unary symbols occurring at prefix positions of the main path. Given a derailed context D, we sometimes denote by D_{pre} and D_{post} its pre and post-contexts.

Example 3.15. *The set of all derailings for the context $hg([.], b)$ is $\{hg([.], b), hg([.], y)\}$. The context $hg([.], y)$ is the pre-context and $[.]$ is the post-context of the non-trivial element in the set.*

Definition 3.16 (Iterated derailing)**.** Let $\lambda\overline{z_{m_1}}.x_1(\overline{t^1_{n_1}}) \doteq \lambda\overline{z_{m_1}}.x_2(\overline{s^1_{n_2}}), \ldots, \lambda\overline{z_{m_k}}.x_k(\overline{t^k_{n_k}}) \doteq \lambda\overline{z_{m_k}}.C[x_1(\overline{s^k_{n_1}})]$ be a unique standard cycle with a context C, then the set $\mathtt{derail}(x_i, m, C)$ of the iterated derailed regular terms for the cycle context C and an index $0 < i \leq k$ is defined as follows:

- if $m = 1$ then $\mathtt{derail}(x_i, 1, C) = \mathtt{reg}(x_i, C)$.
- if $m > 1$ then $\mathtt{derail}(x_i, m, C) = \{\lambda\overline{z_n}.C^*(D_{pre}(C^r)) \mid$
  $D \in \mathtt{one\text{-}derail}(C), \lambda\overline{z_n}.C^r \in \mathtt{derail}(x_i, m-1, D_{post}(D_{pre}))\}$

Example 3.17. *The following table shows the construction of all (non-trivial) iterated derailings for the cycle context $hg(ef(a, [.]), b)$ of the system:*

$$\{x_1 t_1 \doteq x_2 s_1, x_2 t_2 \doteq hg(ef(a, x_1 s_2), b)\} \tag{4}$$

In order to simplify the example, we consider the types of both variables to be equal. For $m = 1$ we follow Ex. 3.8. For $m = 2$ we have

C	D_{pre}	D_{post}	$\lambda z.C^r z$
$hg(ef(a, [.]), b)$	$hg([.], y)$	$ef(a, [.])$	$\in \mathtt{derail}(x_i, 1, ef(a, hg([.], y))$
	$hg(ef(y, [.]), b)$	$[.]$	$\in \mathtt{derail}(x_i, 1, hg(ef(y, [.]), b)$

Definition 3.18 (Instantiations of regular terms)**.** Let C be a context and x a variable, then the infinite set $\mathtt{insts}(x, C, m) = \{t' | t \in \mathtt{derail}(x, m, C)\}$ such that t' is obtained from t by replacing each of the n occurrences of the Kleene stars in t with the natural numbers $k_1, \ldots, k_n$ respectively.

Next, we will prove some properties of instantiations of regular terms.

The first property is that if a term is subsumed by an instantiation using n iterations, then it also does so using $m > n$ iterations.

Lemma 3.19. Let S be a system containing a standard cycle with a cycle context C and variable x and assume that for some pre-unifier σ of S, $\sigma(x) \in_s \mathtt{insts}(x, C, l)$ for $l > 0$ then $\sigma(x) \in_s \mathtt{insts}(x, C, k)$ for all $k > l$.

Proof. For the corresponding iterations we replace the Kleene star with 0 and choose an empty prefix. □

The next property asserts that any term subsumed by $\texttt{insts}_1$ is also subsumed by $\texttt{insts}$ after one iteration.

Lemma 3.20. Let S be a system containing a standard cycle with a cycle context C and let σ be a pre-unifier of S, if $\sigma(x) \in_s \texttt{insts}_1(x, C)$ then $\sigma(x) \in_s \texttt{insts}(x, C, 1)$.

Proof. Clear from the definitions of $\texttt{insts}$ and $\texttt{insts}_1$. □

The following lemma unfolds one iteration and relate terms obtained using different iterations of $\texttt{insts}$.

Lemma 3.21. Let C be a context, C' and C'' contexts such that $C'C'' = C$ and $C'' = f(v_1, \ldots, D'_{@k}, \ldots, v_n)$. Assume that

- $\lambda\overline{z_m}.t_0 \in_s \texttt{insts}(x, D'C'f(y_1, \ldots, [.], \ldots, y_n), m)$ for some variable x and
- $\lambda\overline{z_m}.t_1 \geq_s \lambda\overline{z_m}.(C^lC'f(y_1, \ldots, t_0, \ldots, y_n))$ such that C^lC' does not contain any bound variable.

Then, $\lambda\overline{z_m}.t_1 \in_s \texttt{insts}(x, C, m+1)$.

Proof. From the assumptions we know that there are substitutions θ_1 and θ_2 and a term t' such that

- $t' \in \texttt{insts}(x, D'C'f(y_1, \ldots, [.], \ldots, y_n), m)$.
- $\lambda\overline{z_m}.t_0 = t'\theta_1$.
- $\lambda\overline{z_m}.t_1 = (\lambda\overline{z_m}.(C^lC'f(y_1, \ldots, t_0, \ldots, y_n)))\theta_2$.

Furthermore, since $t' \in \texttt{insts}(x, D'C'f(y_1, \ldots, [.], \ldots, y_n), m)$, we know that there is a regular term $\lambda\overline{z_m}.t^1_0 \in \texttt{derail}(x, m, D'C'f(y_1, \ldots, [.], \ldots, y_n))$. Let $D_{pre} = C'f(y_1, \ldots, [.], \ldots, y_n)$ and $D_{post} = D'$, then $D_{pre}D_{post} \in \texttt{one-derail}(C)$ and $\lambda\overline{z_m}.C^*C'f(y_1, \ldots, t^1_0, \ldots, y_n) \in \texttt{derail}(x, m+1, C)$. Let $t'' = \lambda\overline{z_m}.(C^lC'f(y_1, \ldots, t', \ldots, y_n))$, then $t'' \in \texttt{insts}(x, C, m+1)$. We now have that $t_1 = t''\theta_2$ and therefore, that $t_1 \in_s \texttt{insts}(x, C, m+1)$. □

We can now prove the main lemma in this section, which contains an essential step from [22].

Lemma 3.22. Let S be a system containing a unique standard cycle $\lambda\overline{z_{m_1}}.x_1(\overline{t^1_{n_1}}) \doteq \lambda\overline{z_{m_1}}.x_2(\overline{s^1_{n_2}}), \ldots, \lambda\overline{z_{m_k}}.x_k(\overline{t^k_{n_k}}) \doteq \lambda\overline{z_{m_k}}.C[x_1(\overline{s^k_{n_1}})]$, then for any ground unifier σ of S there is an index $0 < i \leq n$ such that $\sigma(x_i) \in_s \texttt{insts}(x_i, C, k)$.

Proof. First, we compute the maximal contexts D_i of $\sigma(x_i)$ for $0 < i \leq k$. Let A be the greatest common prefix of D_i and $(\sigma(C))^h$ for $0 < i \leq k$ where h is the minimal depth of all D_i. Then, $A = \sigma(C)^q(C')$ for $q \leq h$ where C' is a proper prefix of $\sigma(C)$ and let C'' be a context such that $C'(C'') = \sigma(C)$. By induction on the number of constraints in the cycle.

- for $k = 1$ we first use Lemma 3.13 in order to obtain that $\sigma(x_1) \in_s \texttt{insts}_1(x_1, C)$ and then Lemma 3.20 in order to show that $\sigma(x_1) \in_s \texttt{insts}(x_1, C, 1)$.
- for $k > 0$, if there is $0 < i \leq k$ such that $D_i = A$, then we can use lemmas 3.12 and 3.20 in order to prove that $\sigma(x_i) \in_s \texttt{insts}(x_i, C, 1)$ and by using Lemma 3.19 we have $\sigma(x_i) \in_s \texttt{insts}(x_i, C, k)$. We now assume that $D_i \neq A$ for all $0 < i \leq k$, i.e. that $D_i = Af(v^i_1, \ldots, v^i_n)$ for all $0 < i \leq k$. Clearly, there are at least two terms v^q_i and v^r_j for $0 < i, j \leq n$, $0 < q, r \leq k$ and $i \neq j$ such that both contain bound variables as otherwise f will be common to all D_i for $0 < i \leq k$ (note that σ is a ground unifier). Now consider two cases

– there is an index $0 < p \leq k$ such that v_i^p and v_j^p contain bound variables for $i \neq j$. In this case $\sigma(x_p)$ can be written as $\lambda\overline{z_{n_p}}.Af(\theta(x'_1)(\overline{z_{n_p}}), \ldots, \theta(x'_n)(\overline{z_{n_p}}))$ for $\theta = [v_1^p/x'_1, \ldots, v_n^p/x'_n]$ such that $\hat{b}(x'_i), \hat{b}(x'_j) < \hat{b}(x_p)$ and therefore we satisfy requirement 3a in Def. 3.7 and have that $\sigma(x_p) \in_s \texttt{insts}_1(x_p, C)$. We can now use lemmas 3.20 and 3.19 to have $\sigma(x_p) \in_s \texttt{insts}(x_p, C, k)$.

– in this case for each $0 < i \leq k$, there is only one index $0 < l_i \leq n$ such that $v_{l_i}^i$ contains bound variables and v_j^i does not contain any bound variable for $0 < j \leq n$ and $j \neq l_i$. The standard cycle, after the application of σ, can now be represented as

$$\lambda\overline{z_{m_1}}.Af(v_1^1, \ldots, D^1_{@l_1}(\overline{t^1_{n_1}}), \ldots, v_n^1) \doteq \lambda\overline{z_{m_1}}.Af(v_1^2, \ldots, D^2_{@l_2}(\overline{s^1_{n_2}}), \ldots, t_n^2), \ldots,$$
$$\lambda\overline{z_{m_k}}.Af(v_1^k, \ldots, D^k_{@l_k}(\overline{t^k_{n_k}}), \ldots, v_n^k) \doteq \lambda\overline{z_{m_k}}.C(Af(v_1^1, \ldots, D^1_{@l_1}(\overline{s^k_{n_1}}), \ldots, v_n^1))$$

where $v_{l_i}^i = \lambda\overline{z_{n_i}}.D^i(\overline{z_{n_i}})$ are new terms and are the only terms containing bound variables. Let $C'' = f(t_1, \ldots, D'_{@l}, \ldots, t_n)$ and let $I = \{i_1, \ldots, i_p\}$ contain all the indices $0 < i \leq k$ such that $l_i = l$. (i.e. the head component of the position of the hole in f is l). Consider now the cycle after applications of the `(Decomp)` rule only

$$\lambda\overline{z_{m_1}}.u_1 \doteq \lambda\overline{z_{m_1}}.u'_1, \ldots, \lambda\overline{z_{m_k}}.u_k \doteq \lambda\overline{z_{m_k}}.u'_k \tag{5}$$

where

* for every $0 < i \leq k$, $u_i = D^i(\overline{t^i_{n_i}})$ if $i \in I$ and $u_i = v_l^i$ otherwise.
* for every $0 < i < k$, $u'_i = D^{i+1}(\overline{s^i_{n_{i+1}}})$ if $i + 1 \in I$ and $u'_i = v_l^{i+1}$ otherwise.
* $u'_k = D'C'f(v_1^1, \ldots, D^1_{@l_1}(\overline{s^k_{n_1}}), \ldots, v_n^1)$.

Now let us consider the substitution δ such that $\delta(y_i) = v_l^i$ for $0 < i \leq k$ where y_i are fresh variables and consider the equations

$$\lambda\overline{z_{m_1}}.r_1 \doteq \lambda\overline{z_{m_1}}.r'_1, \ldots, \lambda\overline{z_{m_k}}.r_k \doteq \lambda\overline{z_{m_k}}.r'_k \tag{6}$$

where

* for every $0 < i \leq k$, $r_i = D^i(\overline{t^i_{n_i}})$ if $i \in I$ and $r_i = y_i$ otherwise.
* for every $0 < i < k$, $r'_i = D^{i+1}(\overline{s^i_{n_{i+1}}})$ if $i + 1 \in I$ and $r'_i = y_{i+1}$ otherwise.
* $r'_k = D'C'f(v_1^1, \ldots, D^1_{@l_1}(\overline{s^k_{n_1}}), \ldots, v_n^1)$.

Clearly, this cycle is pre-unifiable by $\sigma \circ \delta$. After applying `(Bind)` on all equations containing y_i, we get

$$\lambda\overline{z_{m_{i_1}}}.D^{i_1}(\overline{t^{i_1}_{n_{i_1}}}) \doteq \lambda\overline{z_{m_{i_1}}}.D^{i_2}(\overline{s^{i_1}_{n_{i_2}}}), \ldots,$$
$$\lambda\overline{z_{m_{i_p}}}.D^{i_p}(\overline{t^{i_p}_{n_{i_p}}}) \doteq \lambda\overline{z_{m_{i_p}}}.D'C'f(v_1^1, \ldots, D^{i_1}_{@l}(\overline{s^{i_p}_{n_{i_1}}}), \ldots, v_n^1) \tag{7}$$

where, $i_1, \ldots, i_p$ is some permutation of $1, \ldots, p$, which is forced by the application of `(Bind)`. Let us take now the substitution θ such that $\theta(w^{i_j}) = \lambda\overline{z_{n_{i_j}}}.D^{i_j}(\overline{z_{n_{i_j}}})$ for $0 < j \leq p$ where w^{i_j} are fresh variables and consider the standard cycle

$$\lambda\overline{z_{m_{i_1}}}.w^{i_1}(\overline{t^{i_1}_{n_{i_1}}}) \doteq \lambda\overline{z_{m_{i_1}}}.w^{i_2}(\overline{s^{i_1}_{n_{i_2}}}), \ldots,$$
$$\lambda\overline{z_{m_{i_p}}}.w^{i_p}(\overline{t^{i_p}_{n_{i_p}}}) \doteq \lambda\overline{z_{m_{i_p}}}.D'C'f(v_1^1, \ldots, w^{i_1}_{@l}(\overline{s^{i_p}_{n_{i_1}}}), \ldots, v_n^1) \tag{8}$$

which is clearly pre-unifiable by $\sigma \circ \theta$. Assume further that we have the substitution η such that $\eta(z_i) = v_i^1$ for $0 < i \leq n$, then the standard cycle

$$\begin{aligned} \lambda\overline{z_{m_{i_1}}}.w^{i_1}(\overline{t_{n_{i_1}}^{i_1}}) &\doteq \lambda\overline{z_{m_{i_1}}}.w^{i_2}(\overline{s_{n_{i_2}}^{i_1}}), \ldots, \\ \lambda\overline{z_{m_{i_p}}}.w^{i_p}(\overline{t_{n_{i_p}}^{i_p}}) &\doteq \lambda\overline{z_{m_{i_p}}}.D'C'f(z_1, \ldots, w_{@l}^{i_1}(\overline{s_{n_{i_1}}^{i_p}}), \ldots, z_n) \end{aligned} \tag{9}$$

is pre-unifiable by $\sigma \circ \theta \circ \eta$. Since $p < k$, we can apply the induction hypothesis in order to obtain that there is and index $0 < j \leq p$ such that $\theta(w^{i_j}) = \lambda\overline{z_{n_{i_j}}}.D^{i_j}(\overline{z_{n_{i_j}}}) \in_s \mathtt{insts}(w^{i_j}, D'C'f(z_i, \ldots, [.]_{@l}, \ldots, z_n)), p)$. Since $\sigma(x_{i_j}) = \lambda\overline{z_{n_{i_j}}}.Af(v_1^{i_j}, \ldots, D^{i_j}(\overline{z_{n_{i_j}}}), \ldots, v_n^{i_j}) = \lambda\overline{z_{n_{i_j}}}.(C\sigma)^q C'f(v_1^{i_j}, \ldots, D^{i_j}(\overline{z_{n_{i_j}}}), \ldots, v_n^{i_j})$, we can use Lemma 3.21 in order to obtain that $\sigma(x_{i_j}) \in_s \mathtt{insts}(x_{i_j}, C, p+1)$ and therefore, using Lemma 3.19, that $\sigma(x_{i_j}) \in_s \mathtt{insts}(x_{i_j}, C, k)$.

□

3.3 Pre-unification using Regular Terms

In this section we will give a procedure for the pre-unification of bounded higher-order problems, which will be based on $\mathtt{PUA}_B$.

The procedure will differ from $\mathtt{PUA}_B$ by using an environment in order to restrict the possible rule applications.

Definition 3.23 (Environments and constraints)**.** A bounding constraint is an equation of the form $\hat{b}(x) = n$ and allows us to represent and monitor the $\hat{b}$ values of the unification problem using the environments. Given an environment E, we will use the following notation for denoting constraints in E:

- $E[d(x) = n]$ restricts mappings for x to be of maximal depth of n. These constraints are called depth constraints.
- $E[r(x) = t]$ restricts mappings for x to be subsumed by the set described by the regular term t. These constraints are called binding constraints.
- $E[\hat{b}(x) = n]$ stands for $\hat{b}(x) = n$.
- let $\rho \in \{d, r, \hat{b}\}$ then
  - $E[\rho(x) = \epsilon]$ means there is no such constraint for x in E.
  - when we write $E[\rho(x) = u]$ we also mean the environment obtained from E by replacing the constraint ρ for x with the new one. If there was no old constraint, we just insert a new constraint into the new environment.
  - $E[\rho_1(x_1) = v_1, \ldots, \rho_n(x_n) = v_n]$ stands for $E[\rho_1(x_1) = v_1] \ldots [\rho_n(x_n) = v_n]$.

Since we have at most one constraint of each type for each variable in the environment, the new notation is well defined.

Definition 3.24 (The set of rules BUA (Bounded unification procedure))**.** Let S be a unification system and E an environment, then the set of rules BUA are defined in Fig. 2. The $\mathtt{reset}(E, S)$ function used in the procedure returns an environment after adding to the environment E a depth constraint containing the value $2 \cdot \mathtt{fbound}(S)$ for each unsolved variable in S. The initial environment is equal to $\mathtt{reset}(\hat{b}, S)$. The function scy returns all unique standard cycles in S.

The following lemma, which will be used in the completeness proof, confirms that cyclic variables can indeed be mapped by BUA to terms subsumed by sets obtained from the cycles' contexts.

Lemma 3.25. Let x be a variable in a cycle in system S with cycle context C and cycle size n and let $t \in \mathtt{insts}(x, C, n)$, then we can derive using BUA a system S' such that $\sigma_{S'}(x) = t$.

Proof. According to the definition of insts there is a regular term t_r corresponding to t. By applying the four cyclic rules (Skip), ($\mathtt{Imitate}_0$), ($\mathtt{Imitate}_*$) and at the end (Project) and by choosing the right partial bindings, we can simulate any instantiation of t_r (by insts). □

The following example shows how the unification procedure unfolds cyclic problems.

Example 3.26. *Let S be the following monadic (right-associative) problem.*

$$\{X_1abX_1bX_2bX_3bX_4c \doteq aX_1bX_2X_2X_2bX_3X_3X_3bX_4X_4X_4baaac\} \tag{10}$$

Fig. 3 shows how we can obtain the unifier

$$[\lambda z.a^{81}z/X_1, \lambda z.a^{27}z/X_2, \lambda z.a^{9}z/X_3, \lambda z.a^{3}z/X_4] \tag{11}$$

Please note that some constraints are removed in order to simplify presentation. We also remove the bounding constraints from the environment as clearly they will be satisfied, in monadic problems, for $\hat{b}(X) > 1$ for all variables X in the problem.

4 Correctness and Termination

4.1 Soundness and Completeness

In this section we will prove the soundness and completeness of BUA with regard to $\mathtt{PUA}_B$.

Theorem 4.1 (Soundness)**.** If S' is obtained from a unification system S using BUA and $\hat{b}$ and is in pre-solved form then $\sigma_{S'}|_{\mathtt{FV}(S)} \in \mathtt{PreUnifiers}^b(S)$.

Proof. The rules in BUA are the same rules as in $\mathtt{PUA}_B$ but pose more restrictions on the generated substitutions. First, we restrict the depth of terms using the depth constraints and second, we generated partial bindings which are less general than the ones generated in $\mathtt{PUA}_B$ due to the use of the binding constraints. Therefore, each run of the procedure can be simulated by $\mathtt{PUA}_B$. □

Our first result towards the completeness of the procedure is the relationship between the depth of terms mapped to variables and the bound fbound in acyclic systems. Before we prove that, though, we need some further definitions.

Definition 4.2 (Relation on variables)**.** Given a system S, the relation $<_c^0$ for S is a relation on the variables in S such that for all constraints $t \doteq s \in S$ and positions p_1 and p_2 such that $\mathtt{hd}(t|_{p_1}) = x$ and $\mathtt{hd}(s|_{p_2}) = y$ and p_1 is a proper prefix of p_2, then $y <_c^0 x$. We define $=_c^0$ in a similar way but require that $p_1 = p_2$. $=_c$ is the symmetric, transitive closure of $=_c^0$. We now define the relation $<_c^T$ inductively

- if $x <_c^0 y$ then $x <_c^T y$.
- if $x =_c z$ and $z <_c^T y$ then $x <_c^T y$.
- if $<_c^T z$ and $z <_c^T y$ then $x <_c^T y$.

$<_c$ is any arbitrary extension of $<_c^T$ into a total order, such that if $<_c^T$ is acyclic, so is $<_c$ (otherwise, the relation is not an ordering).

$$\frac{E \vdash S \cup \{A \doteq A\}}{E \vdash S}\ \texttt{(Delete)} \qquad \frac{E \vdash S \cup \{\lambda\overline{z_k}.f(\overline{s_n}) \doteq \lambda\overline{z_k}.f(\overline{t_n})\}}{E \vdash S \cup \{\lambda\overline{z_k}.s_1 \doteq \lambda\overline{z_k}.t_1, \ldots, \lambda\overline{z_k}.s_n \doteq \lambda\overline{z_k}.t_n\}}\ \texttt{(Decomp)}$$

$$\frac{E[r(x) = \lambda\overline{z_k}.C^*(C_r)] \vdash S}{E[r(x) = \lambda\overline{z_k}.C_r] \vdash S}\ \texttt{(Skip)} \qquad \frac{E \vdash S \cup \{\lambda\overline{z_k}.x(\overline{z_k}) \doteq \lambda\overline{z_k}.t\} \qquad x \notin \texttt{FV}(t), \sigma = [\lambda\overline{z_k}.t/x]}{E\sigma \vdash S\sigma \cup \{x \doteq \lambda\overline{z_k}.t\}}\ \texttt{(Bind)}$$

$$\frac{E \vdash S \qquad \lambda\overline{z_k}.x^\alpha(\overline{s_n}) \doteq \lambda\overline{z_k}.a(\overline{t_m}) \in S, 0 < i \leq k, u = \texttt{PB}^b(i, \alpha), \sigma = [u/x]}{\texttt{reset}(E, S\sigma) \vdash S\sigma \cup \{x \doteq u\}}\ \texttt{(Project)}^2$$

$$\frac{E[r(x) = \epsilon] \vdash S \qquad c \in \texttt{scy}(S, E), x \in c, t \in \texttt{derail}(x, \texttt{size}(c), \texttt{ccon}(c))}{E[r(x) = t] \vdash S}\ \texttt{(Rec)}$$

$$\frac{E[d(x) = m > 0, r(x) = \epsilon] \vdash S \qquad \lambda\overline{z_k}.x^\alpha(\overline{s_n}) \doteq \lambda\overline{z_k}.f(\overline{t_m}) \in S, u = \texttt{PB}^b(f, \alpha), \sigma = [u/x]}{E[d(x_1) = m - 1, \ldots, d(x_1) = m - 1]\sigma \vdash S\sigma \cup \{x \doteq u\}}\ \texttt{(Imitate}_{pb}\texttt{)}^1$$

$$\frac{E[r(x) = \lambda\overline{z_n}.t^*(tr)] \vdash S \qquad \lambda\overline{z_k}.x^\alpha(\overline{s_n}) \doteq \lambda\overline{z_k}.f(\overline{v_p}) \in S, t = f(t_1, \ldots, t'([.]), \ldots, t_p), u = \texttt{PB}^b(f, \alpha), \sigma = [u/x]}{E[r(x_1) = t_1, \ldots, r(x_k) = t'(t^*(tr)), \ldots, r(x_1) = t_p]\sigma \vdash S\sigma \cup \{x \doteq u\}}\ \texttt{(Imitate}_*\texttt{)}^1$$

$$\frac{E[r(x) = \lambda\overline{z_n}.t] \vdash S \qquad \lambda\overline{z_k}.x^\alpha(\overline{s_n}) \doteq \lambda\overline{z_k}.f(\overline{v_p}) \in S, t = f(t_1, \ldots, t_p), u = \texttt{PB}^b(f, \alpha), \sigma = [u/x]}{E[r(x_1) = t_1, \ldots, r(x_1) = t_p]\sigma \vdash S\sigma \cup \{x \doteq u\}}\ \texttt{(Imitate}_0\texttt{)}^1$$

1. $x_1, \ldots, x_p$ are all fresh variables introduced by $\texttt{PB}^b$.

2. a is either a function symbol or a bound variable.

Figure 2: BUA - Bounded unification rules

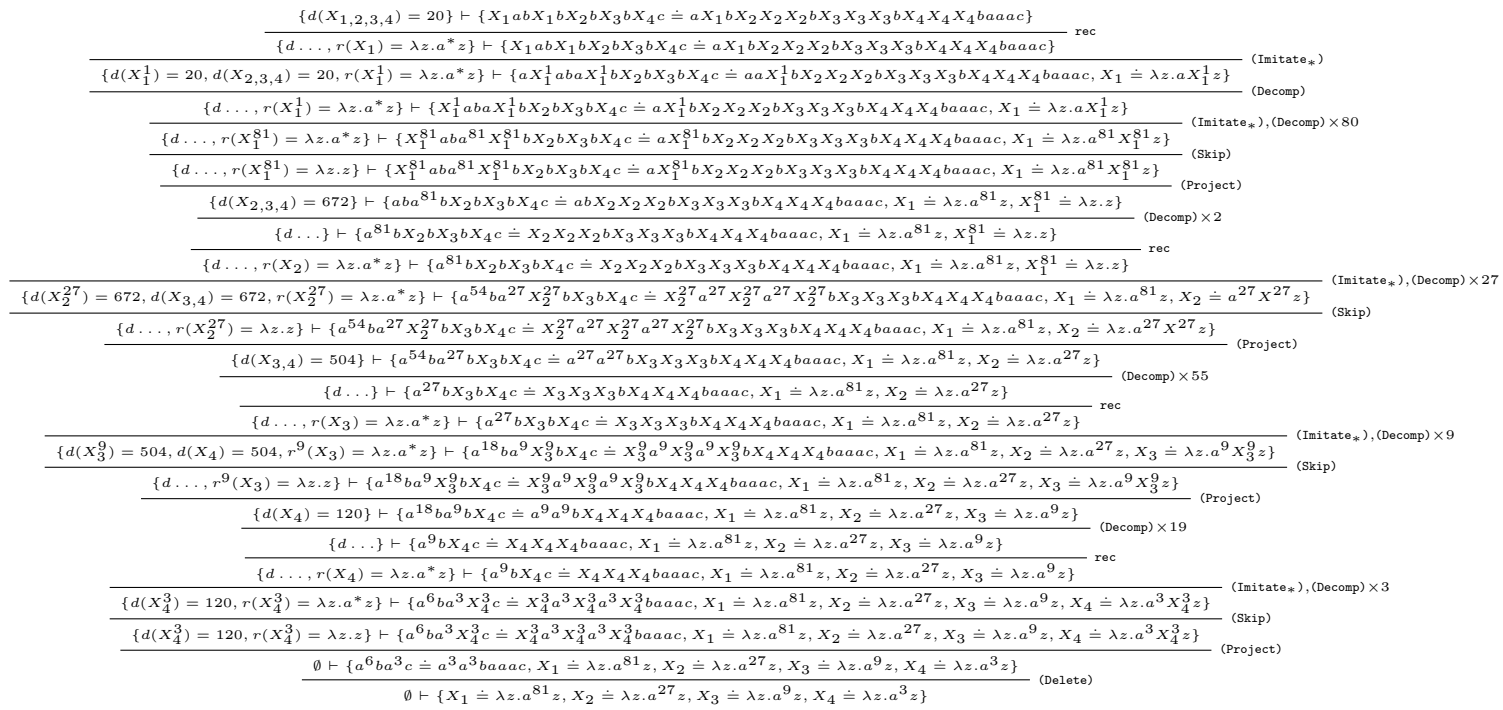

Figure 3: Running BUA on Eq. 10

Example 4.3. *Given the system* $\{f(Xa, g(w, y)) \doteq f(h(z), Xb), y \doteq Xc, y \doteq w\}$*, then* $<_c^0 = \{(z, X), (w, X), (y, X)\}$ *and* $=_c^0 = \{(y, X), (y, w)\}$*. The relation* $<_c^T$ *then is a superset of the set* $\{(z, X), (w, X), (y, X), (X, X)\}$*.*

Definition 4.4 (Repeated variables)**.** Let S be a system. A variable x in S is called a repeated variable in S if there exists a system S' obtainable from S using $\mathtt{PUA}_B$ via any sequence of transformations not including `(Project)` such that $\mathtt{d}(\sigma_{S'}(x)) > \mathtt{fbound}(S)$.

Example 4.5. *Assume we can obtain, from the system* S *containing the constraint* $Xa \doteq t$ *and having* $\mathtt{fbound} = 4$*, a system with the solved constraint* $X \doteq f^5([.])$*, then* X *is called a repeated variable in* S*.*

Lemma 4.6. Let S be a system such that all variables in it are acyclic according to $<_c$ and x be a variable in S. Then, for any system S' obtainable from S by using $\mathtt{PUA}_B$ without the application of a `(Project)`, we have $\mathtt{d}(\sigma_{S'}(x)) \leq \mathtt{fbound}(S)$.

Proof. By induction on the number m of variables, ordered by $<_c$. If $m = 1$, we can apply `(Decomp)` exhaustively until we get a constraint $t \doteq s$ in a system S' such that $\mathtt{hd}(t) = x$ and clearly, for all systems S'' obtainable from S without an application of `(Project)` we have $\mathtt{d}(\sigma_{S''}(x)) \leq \mathtt{md}(S') \leq \mathtt{md}(S) < \mathtt{fbound}(S)$. Otherwise, let x be a maximal variable according to $\leq_c$. Then, there is no other variable y in S, such that there is an equation $t \doteq s \in S$, $\mathtt{hd}(t|_{p_1}) = y$, $\mathtt{hd}(s|_{p_2}) = x$ with p_1 a proper prefix of p_2. Let S_0 be the problem after the removal of all equations containing x in rigid positions. By induction hypothesis, for any system S_0' obtainable from S_0 without the application of `(Project)` and for all variables y in S other than x, $\mathtt{d}(\sigma_{S_0'}(y)) \leq \mathtt{fbound}(S_0)$. Since $\mathtt{md}(S) \geq \mathtt{md}(S_0)$, we get that for all variables y other than x, $\mathtt{d}(\sigma_{S_0'}(y)) \leq \mathtt{fbound}(S_0) \leq \mathtt{fbound}(S) - \mathtt{md}(S)$. Since we can choose the order of equations to be processed in $\mathtt{PUA}_B$ freely (see Remark 2.5), then also $\mathtt{d}(\sigma_{S'}(y)) \leq \mathtt{fbound}(S) - \mathtt{md}(S)$. Now, let $t \doteq s \in S$ be an equation containing x and $t' \doteq s'$ the equation after applying several `(Decomp)`s such that $\mathtt{hd}(t') = x$. Let V_0 be the set of all variables in s', then $\mathtt{d}(\sigma_{S'}(x)) \leq \mathtt{d}(s') + \max_{y \in V_0}(\mathtt{d}(\sigma_{S'}(y)))$. Since $\mathtt{d}(s') \leq \mathtt{md}(S)$ and $\mathtt{d}(\sigma_{S'}(y)) \leq \mathtt{fbound}(S) - \mathtt{md}(S)$, we get that $\mathtt{d}(\sigma_{S'}(x)) \leq \mathtt{fbound}(S)$. □

Definition 4.7 (Sub-equations)**.** Given a rigid-rigid constraint $t \doteq s$, its set of sub-equations is $\{t' \doteq s' \mid t|_p = t', s|_p = s', p \in \mathtt{rigid\text{-}pos}(t) \cap \mathtt{rigid\text{-}pos}(s)\}$ where $\mathtt{rigid\text{-}pos}(t)$ is the set of all rigid positions in t. For a given constraint e, we denote its set of sub-equations by $\mathtt{sub}(e)$.

Example 4.8. $\mathtt{sub}(f(XgXa, gb) \doteq f(ga, gz))$ *is* $\{f(XgXa, gb) \doteq f(ga, gz),\ XgXa \doteq ga, gb \doteq gz, b \doteq z\}$*.*

Definition 4.9 (Problem Restriction)**.** Let S be a system, then a restriction of S with regard to a variable set $V^0 \subseteq \mathtt{FV}(S)$ is the set of all sub-equations of equations in S such that a variable from V^0 is the head of one of the terms in the sub-equation.

Example 4.10. *The restriction to* $V^0 = \{X\}$ *of the above system is the problem* $\{XgXa \doteq ga\}$*.*

Lemma 4.11. If σ unifies a system S, then it unifies a problem restriction S' of S.

Proof. S' is just a subset of the equations generated from S after the application of `(Decomp)` transformations, which preserves the set of unifiers [23]. □

The next lemma states that if a variable is mapped in some pre-unifier to a large term, then this variable or a smaller one can be related to a standard cycle.

Lemma 4.12. If a variable x in a system S is repeated, then it is either cyclic or there is a smaller variable in S, according to $<_c$, which is cyclic.

Proof. Let V_0 be the set of all smaller variables including x and assume none of them is cyclic, then the order $<_c$ is well-founded over the set V^0 and we consider the problem restriction S' of S with regard to V^0. We know that there is a derivation φ in $\mathtt{PUA}_B$ such that we obtain S' with $\mathtt{d}(\sigma_{S'}(x)) > \mathtt{fbound}(S) \geq \mathtt{fbound}(S')$. We would like to get a contradiction to the existence of such a unifier which will imply that no such (extension of a) unifier also exists for S using Lemma 4.11. We choose φ such that it does not contain a `(Project)` call. We can do so as `(Project)` resets repeatability so we can choose φ starting after the last `(Project)` before obtaining S'. Now, as all the variables in S' are acyclic, we can use Lemma 4.6 to get a contradiction. □

The next lemmas show that if we have a cyclic variable, a unique standard cycle can be obtained using `BUA`.

Lemma 4.13. Given a system S with an environment E and assume S contains a cyclic variable and for all unsolved higher-order variables $x \in \mathtt{FV}(S)$ there is $v \geq \mathtt{fbound}(S)$ such that $[d(x) = v]$, then we can obtain a system S' and environment E' using the rules `(Delete)`, `(Decomp)`, `(Bind)` and $(\mathtt{Imitate}_{pb})$ such that S' contains a cycle over the variables $x_1, ., , x_n$ and $E'[d(x_i) = v]$ for $0 < i \leq n$.

Proof. We can obtain such a cycle with the application of `(Decomp)` and `(Bind)` only and these two rules do not affect depth constraints. □

Lemma 4.14. Given a system S with an environment E and assume S contains a cycle $\lambda\overline{z_{m_1}}.x_1(\overline{t^1_{n_1}}) \doteq \lambda\overline{z_{m_1}}.t_1, \ldots, \lambda\overline{z_{m_k}}.x_k(\overline{t^k_{n_k}}) \doteq \lambda\overline{z_{m_k}}.t_k$, and assume further that for all $0 < i \leq k$ there is $v_i \geq \mathtt{fbound}(S)$ such that$E[d(x_i) = v_i]$, then we can obtain a system S' and environment E' using the rules `(Delete)`, `(Decomp)`, `(Bind)` and $(\mathtt{Imitate}_{pb})$ such that S' contains a standard cycle over the variables $y_1, \ldots, y_k$ and such that $E'[d(y_i) = u_i]$ for $0 < i \leq k$ and $u_i > \mathtt{md}(S)$.

Proof. We prove this by induction on $l = \Sigma_{i=1}^{k-1} i * M_i$ where M_i is the size of the minimal position of x_{i+1} in t_i for $0 < i \leq k-1$. If $l = 0$, then we are done as $X_1 \doteq t_1$ cannot be a flex-flex constraint (see next) and we already have a standard cycle. If $l > 0$, then we apply $(\mathtt{Imitate}_{pb})$ on an equation $\lambda\overline{z_{m_j}}.x_j(\overline{t^j_{n_j}}) \doteq \lambda\overline{z_{m_j}}.t_j$ with $0 < j < m$ maximal such that $\mathtt{hd}(t_j) \notin \mathfrak{V}$. Assume further that $t_j = f(t'_1, \ldots, t'_p)$ and that x_{j+1} occurs in t'_q for $0 < q \leq p$. The result, after applying `(Decomp)`, is again a cycle with a new variable x'_j instead of x_j. l is decreased in the new cycle as either $j > 1$ and then we get that M_j is decreased by 1 and M_{j-1} is increased by 1, or $j = 1$ and then M_1 is decreased by 1. In the second case, M_m is increased but we don't count it. Therefore, we can apply the induction hypothesis in order to obtain a standard cycle. The reason E' is as above is that we apply at most l $(\mathtt{Imitate}_{pb})$ steps and each one of these steps decreases one depth constraint by 1, so at worst case one constraint will be decreased by l. As we assumed the constraints to be of the form $E'[d(x_i) = v_i]$ before we start, we will obtain, in the worst case, one constraint of the form $E'[d(x'_i) = v]$ with $v \geq \mathtt{fbound}(S) - l$. Since $M_i \leq \mathtt{md}(S)$ for $0 < i \leq k$, $\mathtt{fbound}(S) = (K+1) \cdot \mathtt{md}(S)$ $(K = \mathtt{size}(\mathtt{FV}(S)))$ and $k \leq K$ we obtain that $v > \mathtt{md}(S)$. □

Lemma 4.15. Given a system S with an environment E and assume S contains a standard cycle $\lambda\overline{z_{m_1}}.x_1(\overline{t^1_{n_1}}) \doteq \lambda\overline{z_{m_1}}.x_2(\overline{s^1_{n_2}}), \ldots, \lambda\overline{z_{m_k}}.x_k(\overline{t^k_{n_k}}) \doteq \lambda\overline{z_{m_k}}.t_k$, and assume further that $E[d(x_i) = v]$ for $0 < i \leq k$ and $v > v_0$ where v_0 is the size of the minimal position of x_1 in t_k, then we can

obtain a system S' with a unique standard cycle using the rules (Delete), (Decomp), (Bind) and (Imitate$_{pb}$).

Proof. First, if the standard cycle is also unique, then we are done. Otherwise, let p_m be the minimal position in t_k of x_1. The way to achieve a unique standard cycle is similar to what was done in the proof of the previous lemma. By applying $\mathtt{size}(p_m)$ times the rule (Imitate$_{pb}$) on the last equation we will obtain a unique standard cycle. Applying the rule $\mathtt{size}(p_m)$ times is possible according to the depth constraints. □

Lemma 4.16. Let S_0 be a system with a cyclic variable, then there is a system S with a cyclic variable and with an environment E such that S_0 is obtainable from S using no application of the rule (Project) and for all unsolved variables x in S, $E[d(x) = v]$ where $v \geq \mathtt{fbound}(S)$.

Proof. Let φ be the derivation of S_0 and let S_1 be the last system in the derivation which is either an initial system or immediately after the application of a (Project). If there is a cyclic variable in S_1, then we choose $S = S_1$ and have $E[d(x) = 2 \cdot \mathtt{fbound}(S)]$ for all unsolved variables x in S_1 and we are done. Otherwise, since S_1 is acyclic, let $S = S_0$ and we can use Lemma 4.6 in order to obtain S such that $E[d(x) = v]$ for all unsolved variables x in S where $v \geq \mathtt{fbound}(S)$. □

So far we considered all cycle's contexts to be indeed contexts, i.e. terms of type $\alpha \to \alpha$. Clearly, we might have cycles where this term is of type $\alpha \to \beta$ for $\beta \neq \alpha$. This means that the cyclic variable at this position is preceded by a λ-binder, a fact which strictly reduces the $\lambda\mathtt{size}$ of the term mapped to the variable and therefore the $\hat{b}$ values in the partial binding. Let us call the first kind of cycles pure, we deal with both kinds in the following lemma.

Lemma 4.17. Given a system S with an environment E and assume it contains a cyclic variable, then we can obtain either

- a pure unique standard cycle using BUA while applying only the rules (Delete), (Decomp), (Bind) and (Imitate$_{pb}$) or
- for every pre-unifier σ of S, we can obtain a system S' such that $\mathtt{b\text{-}measure}(S', E'[\hat{b}]) < \mathtt{b\text{-}measure}(S, E[\hat{b}])$ and $\sigma \circ \theta$ is a pre-unifier of S' for some substitution θ.

Proof. We first use Lemma 4.16 in order to obtain a system with a cyclic variable such that for all $x \in \mathtt{FV}(S)$ there is $v \geq \mathtt{fbound}(S)$ such that $E[d(x) = v]$. We use now Lemma 4.13 in order to obtain a cycle without having the environment changed. We now consider two cases

- if the cycle is pure, then we obtain a standard cycle over the variables $x_1, \ldots, x_k$ such that $E'[d(x_i) = v]$ for $0 < i \leq n$ and $v > \mathtt{md}(S)$ using Lemma 4.14. The last step is to obtain a standard cycle using Lemma 4.15 and here we note that the size of the minimal position of x_1 in t_k must be smaller than $\mathtt{md}(S)$. This is because the rigid positions of variables cannot become deeper by applying the (Imitate$_{pb}$) rule.
- if the cycle is impure, then we consider the constraint $\lambda\overline{z_{m_j}}.x_j(\overline{t^j_{n_j}}) \doteq \lambda\overline{z_{m_j}}.t_j$ where t_j is impure for some $0 < j \leq k$. In order to preserve completeness, we must consider both applications of (Project) and (Imitate$_{pb}$). Since an application of (Project) will decrease the bounding measure, we assume we apply (Imitate$_{pb}$) only, but after at most $\mathtt{d}(t_j)$ applications, where $\mathtt{d}(t_j) \leq \mathtt{md}(S) \leq \mathtt{fbound}(S)$, we will get a solved constraint $x_j \doteq t'_j$ where t'_j contains unsolved variables $x'_1, \ldots, x'_l$ and since t_j was impure, $\hat{b}(x'_i) < \hat{b}(x_j)$ for $0 < i \leq l$ and therefore $\mathtt{b\text{-}measure}(S', E'[\hat{b}]) < \mathtt{b\text{-}measure}(S, E[\hat{b}])$.

□

In the following lemma we show that for any pre-unifier σ of a problem containing a pure standard cycle, we can derive a problem with a reduced bounded measure which is unifiable by σ.

Lemma 4.18. Given a system S with an environment E and a pure unique standard cycle, then for any ground unifier σ of S, there is a derivation S' of S using `BUA` such that S' is unifiable by σ and $\texttt{b-measure}(S', E'[\hat{b}]) < \texttt{b-measure}(S, E[\hat{b}])$.

Proof. We first use Lemma 3.22 in order to obtain that there is a variable x such that $\sigma(x) \in_s \texttt{insts}(x, C', n)$ for C' the standard cycles' context and n its size. Assume there is a term $t \in \texttt{insts}(x, C', n)$ and a substitution θ such that $\sigma(x) = t\theta$. We now use Lemma 3.25 in order to obtain a system S' such that $\sigma_{S'}(x) = t$ and therefore $\sigma_{S'} \leq \sigma$. Since the definition of `insts` is based on `reg` which strictly reduces the `b-measure`, we get that $\texttt{b-measure}(S', E'[\hat{b}]) < \texttt{b-measure}(S, E[\hat{b}])$. □

Theorem 4.19 (Completeness)**.** If a bounded unification system S is pre-unifiable by a $\hat{b}$-bounded substitution θ, then there exists a pre-solved system S', which is obtainable from S using `BUA` and $\hat{b}$ such that $\sigma_{S'}|_{\texttt{FV}(S)} \leq \theta$.

Proof. We will prove by induction over the bounding measure `b-measure`, that each pre-solved form obtainable using $\texttt{PUA}_B$ can be also obtained by `BUA`. The induction hypothesis is therefore for a given system S having bounding measure m, if it is possible to obtain a pre-unifier of S using $\texttt{PUA}_B$, then it is possible to obtain the same pre-unifier using `BUA`. Induction base ($m = \emptyset$) - we can replace all variables by first-order variables and clearly cyclic systems are not unifiable. Therefore, we can simulate any run of the complete $\texttt{PUA}_B$ with `BUA`. Induction step - once we apply `(Project)` in $\texttt{PUA}_B$, we can use the induction hypothesis so we assume we need to simulate, using `BUA`, the remaining rules only. We notice that all rules except `(Imitate)` are the same. `(Imitate)` differs from $\texttt{(Imitate}_{pb}\texttt{)}$ in `BUA` with regard to cyclic variables only. We consider the following two cases. If `(Imitate)` is applied on a variable which is not repeated and is not cyclic, then we can use Lemma 4.6 and obtain the same system using the rule $\texttt{(Imitate}_{pb}\texttt{)}$ of `BUA`. Now, assume we apply `(Imitate)` in $\texttt{PUA}_B$ on a variable that is either cyclic or repeated. Using Lemma 4.12 we know that if the variable is repeated, then there is a cyclic variable in the system. We now show that without losing any pre-unifier, we can reduce the bounding measure and therefore apply the induction hypothesis. Since the only two non-deterministic rules to apply are $\texttt{(Imitate}_{pb}\texttt{)}$ and `(Project)` and an application of `(Project)` will allow us to use the induction hypothesis, we can use Lemma 4.17, without losing any pre-unifier, in order to obtain either a system with a smaller bounding measure or a system with a pure standard cycle. We use Lemma 4.18 and the fact that a pre-unifier can be extended easily into a ground unifier in order to derive, in the second case, a system using `BUA` which is unifiable by θ and which has a smaller bounding measure. Either way, we can use the induction hypothesis. □

4.2 Termination

In this section we give a proof, based on our procedure, of the decidability of the unifiability question of bounded higher-order problems [22]. This is possible due to the following result [20].

Definition 4.20 (Minimal unifiers)**.** Given a system S, a unifier σ of S is called minimal if there is no other unifier σ' of S with $\Sigma_{x\in \mathtt{FV}(S)}\mathtt{size}(\sigma'(x)) < \Sigma_{x\in \mathtt{FV}(S)}\mathtt{size}(\sigma(x))$.

Definition 4.21 (Exponent of periodicity)**.** A ground unifier σ has an exponent of periodicity n iff n is the maximal number such that there is some variable x and ground contexts A and B as well as a term t such that $\sigma(x) = \lambda\overline{z_m}.AB^n t$ for $m \geq 0$.

We define $\mathtt{eop}(S)$ for a system S to be some value based on S which satisfies the next lemma. The precise value can be found in [22].

Lemma 4.22 ([22])**.** For every unifiable system S and for every minimal unifier σ of S, its exponent of periodicity is less than $\mathtt{eop}(S)$.

The exponent computed in the lemma above allows us to replace the Kleene stars in the regular terms with a concrete value.

Definition 4.23 (Restricted `derail`)**.** Given a system S, the restricted `derail` function for S ($\mathtt{derail}_S$) is defined as `derail` but instead of introducing the Kleene star, the function introduce the number $\mathtt{eop}(S)$. The produced terms are called restricted regular terms or just regular terms.

Definition 4.24 (Restricted instantiations)**.** Similarly to Def 3.18, we define a restricted instantiation of a restricted regular term t as the term obtained by replacing the n occurrences of the exponent e by values $k_1, \ldots, k_n$ such that $k_i \leq e$ for $0 < i \leq n$. In addition we define the set of all instantiations of a restricted regular term t as the finite set containing all possible restricted instantiations. This set is finite as the set of iterated derailings is finite and once we put a bound on the maximal number of repetitions of the Kleene star, the number of instantiations is finite as well. In the remaining of this section `insts` will refer to its restricted version.

Example 4.25. *The term $\lambda z.(f([.], w))^4 f(x_1(z), x_2(z))$ is a restricted instantiation of the restricted regular term $\lambda z.(f([.], w))^e f(x_1(z), x_2(z))$ for $e = 8$.*

Definition 4.26 (Environments and constraints)**.** The notions of environments and of binding and depth constraints are the same as for `BUA`. The only difference is that we identify each binding constraint over a restricted regular term t with a natural number such that this number is the maximal size of an `mpath` of all maximal contexts of terms contained in $\mathtt{insts}(t)$ $+2$. The intuition behind this value is to describe the maximal depth of a bound variable in the regular term. Since the language is (now) finite, it is possible to compute this value. We call this number the value of the constraint.

Example 4.27. *Assume we have a constraint $E[r(x) = t]$ where t is the restricted regular context from the previous example, then the value of this constraint is $8 + 2 = 10$.*

Definition 4.28 (Restricted `BUA` (`RBUA`))**.** Let $(S_0, \hat{b}_0)$ be the initial system, then the restricted `BUA` (`RBUA`) consists of the rules `(Delete)`, `(Decomp)`, `(Bind)`, $\mathtt{(Imitate}_{pb}\mathtt{)}$, `(Project)` and $\mathtt{(Imitate}_0\mathtt{)}$ from Fig. 2 together with the three rules in Fig. 4.

Example 4.29. *By replacing all the Kleene stars in Ex. 3.26 with the exponent of periodicity of the problem we can obtain exactly the same derivation using `RBUA`.*

Lemma 4.30 (Soundness)**.** If S' is obtained from a unification system S using `RBUA` then $\mathtt{PreUnifiers}(S') \subseteq \mathtt{PreUnifiers}(S)$.

Proof. Following from Thm. 4.1. □

$$\frac{E[r(x) = \lambda\overline{z_k}.C^l(C_r)] \vdash S}{E[r(x) = \lambda\overline{z_k}.C_r] \vdash S}\ \texttt{(Skip)}$$

$$\frac{E[r(x) = \epsilon] \vdash S \qquad c \in \texttt{scy}(S, E), x \in c, t \in \texttt{derail}_{S_0}(x, \texttt{size}(c), \texttt{ccon}(c))}{E[r(x) = t] \vdash S}\ \texttt{(Rec)}$$

$$\frac{E[r(x) = \lambda\overline{z_n}.t^l(tr)] \vdash S \qquad \lambda\overline{z_k}.x^\alpha(\overline{s_n}) \doteq \lambda\overline{z_k}.f(\overline{v_p}) \in S, t = f(t_1, \ldots, t'([.]), \ldots, t_p), u = \texttt{PB}^b(f, \alpha), \sigma = [u/x]}{E[r(x_1) = t_1, \ldots, r(x_k) = t'(t^{l-1}(tr)), \ldots, r(x_1) = t_p]\sigma \vdash S\sigma \cup \{x \doteq u\}}\ \texttt{(Imitate}_*\texttt{)}^1$$

1. $x_1, \ldots, x_p$ are all fresh variables introduced by $\texttt{PB}^b$.

Figure 4: RBUA - Restricted bounded unification rules

Lemma 4.31. If θ is a minimal unifier of a unification system S, then there exists a pre-solved system S', which is obtainable from S using `RBUA` such that $\sigma_{S'}|_{\mathtt{FV}(S)} \leq \theta$.

Proof. From the completeness of `BUA` we know that we can obtain such a pre-unifier for each unifier of S. By using Lemma 4.22 we can show that we do not need to seek pre-unifiers with term depth bigger than the exponent of periodicity, which is exactly the bound we use in the algorithm. □

Definition 4.32 (Regular measure)**.** Let E be an environment and let $d_1, \ldots, d_n$ be the values of the binding constraints in E for all unsolved variables in S, then the regular measure of E is the sum $\Sigma_{0<i\leq n} d_i$.

Definition 4.33 (Depth measure)**.** Let E be an environment and let $m_1, \ldots, m_n$ be all the numbers occurring in depth constraints in E for all unsolved variables in S, then the depth measure of E is the sum $\Sigma_{0<i\leq n} m_i$.

Theorem 4.34. Given a system S and environment E, `RBUA` terminates on S.

Proof. The algorithm is finitely branching. We will show termination of a specific run by taking the lexicographic ordering of the following measure $\mu =< m_1, m_2, m_3, m_4, m_5, m_6 >$ where

- m_1 is the bounding measure $\mathtt{b\text{-}measure}(S, E[\hat{b}])$,
- m_2 is is the multiset $\{\hat{b}(x) - \mathtt{ar}(x) | x \in V\}$ where V contains all variables which do not occur also in a binding constraint in E.
- m_3 is the regular measure,
- m_4 is the depth measure,
- m_5 is the number of unsolved variables x with $\hat{b}(x) = 0$ and
- m_6 is the number of symbols other than $\doteq$ in the problem.

We prove that the measure μ is decreased after any application of `RBUA`.

- An application of `(Delete)` or `(Decomp)` decreases m_6 and does not increase any other measure.
- An application of `(Bind)` either decreases m_1 and m_2 if $\hat{b}(x) > 0$ or decreases m_5 if $\hat{b}(x) = 0$. It does not increase m_1, m_2 or m_4. It also does not increase m_3 since the size of `mpath` of maximal contexts is not affected by `(Bind)`.
- An application of $\mathtt{(Imitate}_{pb}\mathtt{)}$ decreases m_4 and does not increase m_1 or m_2 (although it might decrease them). It does not increase m_3 following the previous argument.
- An application of `(Skip)`, $\mathtt{(Imitate}_0\mathtt{)}$ and $\mathtt{(Imitate}_*\mathtt{)}$ decreases m_3 as we decrease the size of the `mpath` of the maximal context in the new constraint by at least 1. It does not increase m_1 or m_2.
- An application of a `(Project)` decreases m_1.
- An application of a `(Rec)` decreases m_2 as it introduces a binding constraint for a variable and is applicable only if one did not exists. It does not increase m_1.

Therefore, the measure μ decreases after each application of `RBUA`. □

Theorem 4.35. The unifiability question of bounded unification problems is decidable.

Proof. Following lemmas 4.30 and 4.31 and Thm. 4.34. □

Corollary 4.36. The monadic second-order unification problem [8] is decidable.

Proof. Since the problem is second-order, bounded variables must be of basic type only and as the signature contains only monadic function symbols and constants, any ground unifier of the problem will map the variables of the problem to terms with at most one bounded variable occurrence. The maximal number of λ-binders can be deduced from the original variables in the problem and hence a fixed bound can be given for such a problem and we can use Thm. 4.35 in order to decide its unifiability. □

5 Conclusion

The general higher-order unification question is undecidable [9]. One of the two ways to deal with this undecidability is to consider the unifiability question instead. This was done for several sub-classes of higher-order logic from string unification [15] to bounded higher-order unification [22]. In this paper a new procedure for bounded higher-order unification is introduced. The three main differences between this procedure and the one introduced in [22] are the enumeration of all pre-unifiers, the similarity to Huet's procedure, which implies a simpler set of rules and the re-use of its correctness proofs, and the use of regular terms for replacing cycles in the unification problems. We will discuss each of these points in the remaining of this section.

The fact the procedure enumerates all pre-unifiers, in the same way as the Huet's procedure, has the obvious benefit that this procedure is complete. On the other hand, as is shown in the paper, it is possible to sacrifice completeness for gaining termination, in the same way as is done in [22]. This study of the relationship between completeness and termination might allow for specialized procedures which sacrifice the termination on some classes for greater completeness and vice-versa. This study also has implications on understanding possible non-terminating searches for unifiers in the Huet's procedure.

The similarity to Huet's procedure and the (relative) simplicity of the correctness proof suggest an easier implementation and accessibility to the procedure outside the field of higher-order unifiability algorithms.

The most interesting difference lies in the encoding of cycles using regular expressions. The most obvious advantage is when used in automated deduction, where the search for proof requires back-tracking. We might need not recompute the encodings in cases the cycles do not change. This is the case when we only add equations to the problems, as happens in the constrained resolution calculus [10].

There is a more interesting result of this encoding, which is the one-to-one mapping between regular expressions and (cycles in) unification problems. The definitions and results in this paper show how to obtain a regular expression for each possible cycles in higher-order unification. Moreover, one can use the results to obtain a unification problem for a given regular expression. This direction, which is planned as future work, might have implications in numerous fields, like text compression, automated deduction, the proving of undecidability problems by reductions from higher-order unification and decidability problems by reduction to bounded higher-order unification.

References

[1] Habib Abdulrab, Pavel Goralcik, and G. S. Makanin. Towards parametrizing word equations. *J. Info. Theor. Appl.*, 35(4):331–350, 2001.

[2] Peter Andrews, Sunil Issar, Daniel Nesmith, and Frank Pfenning. The tps theorem proving system. volume 310 of *Lect. Not. Comput. Sci.*, pages 760–761. 1988.

[3] Hendrik Pieter Barendregt. *The Lambda Calculus – Its Syntax and Semantics*, volume 103 of *Stud. Log. Found. Math.* North-Holland, 1984.

[4] Christoph Benzmüller, Larry Paulson, Frank Theiss, and Arnaud Fietzke. The LEO-II project. In *Proc. Workshop Auto. Reason.*, 2007.

[5] Alonzo Church. A formulation of the simple theory of types. *J. Symb. Log.*, 5(2):56–68, 1940.

[6] Hubert Comon. Completion of rewrite systems with membership constraints. part i: Deduction rules. *J. Symb. Comput.*, 25(4):397–419, 1998.

[7] Nachum Dershowitz and Zohar Manna. Proving termination with multiset orderings. *Commun. ACM*, 22(8):465–476, August 1979.

[8] William M. Farmer. A unification algorithm for second-order monadic terms. *Ann. Pure Appl. Logic*, 39(2):131–174, 1988.

[9] Warren D. Goldfarb. The undecidability of the second-order unification problem. *Theor. Comput. Sci.*, 13:225–230, 1981.

[10] Gérard P. Huet. *Constrained Resolution: A Complete Method for Higher Order Logic.* PhD thesis, Case Western Reserve University, 1972.

[11] Gérard P. Huet. A unification algorithm for typed lambda-calculus. *Theor. Comput. Sci.*, 1(1):27–57, 1975.

[12] A. Koscielski and L. Pacholski. Complexity of unification in free groups and free semi-groups. In *Symp. Found. Comput. Sci.*, pages 824–829 vol.2, Washington, DC, USA, 1990. IEEE Computer Society.

[13] Philippe Le Chenadec. The finite automaton of an elementary cyclic set. Technical Report RR-0824, INRIA, April 1988.

[14] Jordi Levy. Linear second-order unification. volume 1103 of *Lect. Not. Comput. Sci.*, 1996.

[15] G S Makanin. The problem of solvability of equations in a free semigroup. *Math. USSR-Sbornik*, 32(2):129, 1977.

[16] Dale Miller. Unification of simply typed lambda-terms as logic programming. In *Proc. Log. Prog.*, pages 255–269. MIT Press, 1991.

[17] Lawrence Paulson. Isabelle: The next seven hundred theorem provers. volume 310 of *Lect. Not. Comput. Sci.*, pages 772–773. Springer Berlin / Heidelberg, 1988.

[18] Christian Prehofer. Decidable higher-order unification problems. volume 814 of *Lect. Not. Comput. Sci.*, pages 635–649, London, UK, UK, 1994. Springer-Verlag.

[19] Manfred Schmidt-Schauß. A decision algorithm for stratified context unification. *J. Log. Comput.*, 12(6):929–953, 2002.

[20] Manfred Schmidt-Schauß and Klaus U. Schulz. On the exponent of periodicity of minimal solutions of context equation. volume 1379 of *Lect. Not. Comput. Sci.*, pages 61–75. Springer, 1998.

[21] Manfred Schmidt-Schauß and Klaus U. Schulz. Solvability of context equations with two context variables is decidable. *J. Symb. Comput.*, 33(1):77–122, 2002.

[22] Manfred Schmidt-Schauß and Klaus U. Schulz. Decidability of bounded higher-order unification. *J. Symb. Comput.*, 40(2):905–954, August 2005.

[23] Wayne Snyder and Jean H. Gallier. Higher-order unification revisited: Complete sets of transformations. *J. Symb. Comput.*, 8(1/2):101–140, 1989.

[24] Wayne S. Snyder. *Complete sets of transformations for general unification.* PhD thesis, Philadelphia, PA, USA, 1988. AAI8824793.