

# Regular Patterns in Second-order Unification

Tomer Libal<sup>1</sup>

INRIA

`tomer.libal@inria.fr`

**Abstract.** The second-order unification problem is undecidable. While unification procedures, like Huet’s pre-unification, terminate with success on unifiable problems, they might not terminate on non-unifiable ones. There are several decidability results for infinitary unification, such as for monadic second-order problems. These results are based on the regular structure of the solutions of these problems and by computing minimal unifiers. In this paper we describe a refinement to Huet’s pre-unification procedure for arbitrary second-order signatures which, in some cases, terminates on problems on which the original pre-unification procedure fails to terminate. We show that the refinement has, asymptotically, the same complexity as the original procedure. Another contribution of the paper is the identification of a new decidable class of second-order unification problems.

## 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 kind we can find Miller’s higher-order pattern unification [15],[17] and decidable sub-classes of context unification [6],[12],[18],[20]. When we need to consider arbitrary higher-order unification problems, we must search for an incomplete set of unifiers.

Most higher-order theorem provers, such as Isabelle [16], TPS [2] and LEO II [4] and III [24], rely on Huet’s pre-unification procedure [9] 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. When one is interested in complete sets of unifiers, one must accept non-termination.

---

<sup>1</sup> Funded by the ERC Advanced Grant ProofCert.

The common practice of establishing the decidability of a new class of infinitary unification problems is by proving that their complete sets of unifiers can be described by a finite regular expression. One can then use the exponent of periodicity theorem [10], [19], in order to prove the existence of minimal unifiers. Among the classes of unification problems decided by this technique are not only those over monadic signatures [7],[14],[25] but also their extensions to problems over arbitrary signatures for which the unifiers are restricted to have a limited number of occurrences of the bound variables [12],[13],[18],[20],[21]. The common property of all these classes is that complete sets of unifiers can be described by regular terms. By regularity we mean the ability to describe infinite sets using finite descriptions.

Unfortunately, unrestricted unification over arbitrary signatures does not enjoy this property, even when restricted to very simple second-order languages, as was shown by Farmer [8].

Many interesting problems, among them unification problems generated in the search of theorems of second-order arithmetic, do not fall within these classes. For these problems, non-termination of unification seems inevitable.

In this paper we present a procedure for second-order pre-unification which terminates on more classes of unification problems than Huet's pre-unification procedure while keeping to the same complexity class. This is achieved by a new technique of extending the non-regular complete sets of unifiers of these problems into regular complete supersets of unifiers. We then prove the existence of minimal members in these supersets. Empty complete supersets of unifiers will imply the emptiness of the respective complete sets of unifiers and those, will prove the non-unifiability of the respective problems. We prove the soundness and completeness of this procedure.

As a second contribution, we use the structures developed in this paper in order to recognize a new class of second-order unification problems, whose complete sets of unifiers are regular and which can be decided by the new pre-unification procedure. We believe that this approach can lead to more decidable classes of second-order unification problems.

Other similar works includes the work of Abdulrab et al. for the finite representation of all unifiers for sub-classes of the string unification problem by using graphs and regular expressions [1], that of Zaionc for the regular expression description of complete sets of unifiers for monadic second-order unification [25] and the work of Le Chenadec for the description of first-order cycles using finite automata [11]. These works differ from the current one in that they cover problems whose complete sets of unifiers are regular.

The paper is organized as follows. In the next section we give some definitions and notations which will be used throughout the paper. The main section is dedicated to the construction of complete supersets of unifiers, the establishment of some of their properties and the presentation of the pre-unification procedure. We then prove the correctness of the procedure and its improved termination over classes of problems when compared to Huet's procedure. We conclude by proving the asymptotic equivalence of the complexity of the two procedures.

Due to space considerations, we omit the proofs of all the theorems and lemmas appearing in the paper. The interested reader can find these proofs on the author’s website<sup>1</sup>.

## 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  $\eta$ -conversion rule as presented in [3] and [22] and with implicit  $\alpha$ -conversions. Most of the definitions in this section are adapted from [22].

Let  $\mathfrak{T}_\circ$  be a set of basic types, then the set of types  $\mathfrak{T}$  is generated by  $\mathfrak{T} := \mathfrak{T}_\circ | \mathfrak{T} \rightarrow \mathfrak{T}$ . Let  $\Sigma$  be a signature of function symbols and let  $\mathfrak{V}$  be a countably infinite set of variable symbols. The function **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. The set  $\mathbf{Term}^\alpha$  of terms of type  $\alpha$  is generated by  $\mathbf{Term}^\alpha := f^\alpha | x^\alpha | \lambda x^\beta. \mathbf{Term}^\gamma | \mathbf{Term}^{\beta \rightarrow \alpha}(\mathbf{Term}^\beta)$  where  $f \in \Sigma, x \in \mathfrak{V}$  and  $\alpha \in \mathfrak{T}$  (in the abstraction,  $\alpha = \beta \rightarrow \gamma$ ). Applications throughout the paper will be associated to the right. We will sometimes omit brackets in applications when the meaning is clear. The set **Term** denotes the set of all terms. *Subterms*, *positions* and *position prefixes* are defined as usual. *Sizes of positions* denote the length of the path to the position. 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  $\mathbf{hd}(t)$  its *head symbol* and distinguish 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 prefix position. The *depth* of a term  $t$ , denoted by  $\mathbf{d}(t)$ , is the size of the maximal rigid position in  $t$ . The *order* of types are denoted by **order** and are defined as usual. The order of a term, denoted using the same symbol, is the order of its type.

*Substitutions* and their *composition* ( $\circ$ ) are defined as usual. We denote by  $\sigma|_W$  the substitution obtained from substitution  $\sigma$  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. Variable capture is avoided by implicitly renaming variables to fresh names upon binding. A substitution  $\sigma$  is *more general* than a substitution  $\theta$ , denoted  $\sigma \leq_s \theta$ , if there is a substitution  $\delta$  such that  $\sigma \circ \delta = \theta$ .

We assume that all the terms considered in this paper are in  $\beta$ -normal and  $\eta$ -expanded forms [22]. We further assume that all substitutions are idempotent [23] and contain only terms in  $\beta$ -normal and  $\eta$ -expanded forms in their co-domain. This allows us to deal with normal forms implicitly (see [22] for more

<sup>1</sup> [http://logic.at/staff/shaolin/papers/holunif\\_proofs.pdf](http://logic.at/staff/shaolin/papers/holunif_proofs.pdf)

information). Equality between terms is always assumed to be  $\alpha$ -equality. Each application of a  $\lambda$ -term to another is always converted implicitly into  $\beta$ -normal form.

We introduce also a vector notation  $\overline{t_n}$  for the sequence of terms  $t_1, \dots, t_n$ . Currying and uncurrying is applied implicitly as well.

We will sometimes refer to the position  $0 < i \leq n$  of a term  $s$  in the sequence by  $t_1, \dots, s_{@i}, \dots, t_n$ .

## 2.2 Contexts and Pre-unification

The majority of the definitions in this section are taken from [21,22].

Terms of the form  $\lambda z^\alpha. s^\alpha$  where  $z$  occurs in  $s$  exactly once are called *contexts* and are denoted by  $s([\cdot])$  where  $[\cdot]$  is considered as the “hole” of the term. We denote by  $\text{mpath}(C)$  the *main path* of the context  $C$  which is the position of the hole in the context  $C$ .

*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  $t$  and  $s$  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  $\sigma$  *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  $\text{Unifiers}(S)$ . Let  $\cong$  be the least congruence relation on  $\text{Term}$  which contains  $\{(t, s) \mid \text{hd}(t), \text{hd}(s) \in \mathfrak{V}\}$ . A substitution  $\sigma$  *pre-unifies* an equation  $t \doteq s$  if  $t\sigma \cong s\sigma$ . It pre-unifies a system if it pre-unifies all its equations. The completing substitution  $\xi_S$  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  $\sigma$  pre-unifies a system  $S$ , then  $\sigma \circ \xi$  unifies  $S$  [22]. A *complete set of pre-unifiers* for a system  $S$ , denoted by  $\text{PreUnifiers}(S)$ , is a set of substitutions such that  $\{\sigma \circ \xi_S \mid \sigma \in \text{PreUnifiers}(S)\} \subseteq \text{Unifiers}(S)$  and for every  $\theta \in \text{Unifiers}(S)$  there exists  $\sigma \in \text{PreUnifiers}(S)$  such that  $\sigma|_{\text{dom}(\theta)} \leq \theta$ .

An equation  $x \doteq t$  in  $\eta$ -normal form is called *solved* in system  $S$  if  $x$  does not occur elsewhere in  $S$ . 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. A system is solved (pre-solved) if all its equations are solved (pre-solved). We denote by  $\sigma_S$  the substitution obtained from mapping  $x$  to  $t$  in all solved equations  $x \doteq t$  in  $S$ .

*Imitation partial bindings* and *projection partial bindings* are defined in [22] and are denoted, respectively, by  $\text{PB}(f, \alpha)$  and  $\text{PB}(i, \alpha)$  where  $\alpha \in \mathfrak{T}$ ,  $f \in \Sigma$  and  $0 < i$ . Briefly, partial bindings are substitutions which are used in order to approximate the (possibly infinite) number of final mappings for variables occurring in flex-rigid equations. By either imitating the head symbol of the rigid equation or by projecting one of the bound variables of the mapping for the variable, the set of partial bindings is always finite.

Huet’s pre-unification procedure PUA, as presented by Snyder and Gallier [22], is given in Fig. 1.

**Theorem 1 (Soundness of PUA [9]).** *If  $S'$  is obtained from a unification system  $S$  using PUA and is in pre-solved form, then  $\sigma_{S'}|_{\text{FV}(S)} \in \text{PreUnifiers}(S)$ .*

$$\begin{array}{c}
\frac{S}{S \cup \{A \doteq A\}} \text{ (Delete)} \quad \frac{S \cup \{\lambda \bar{z}_k.s_1 \doteq \lambda \bar{z}_k.t_1, \dots, \lambda \bar{z}_k.s_n \doteq \lambda \bar{z}_k.t_n\}}{S \cup \{\lambda \bar{z}_k.f(\bar{s}_n) \doteq \lambda \bar{z}_k.f(\bar{t}_n)\}} \text{ (Decomp)} \\
\frac{S \sigma \cup \{x \doteq \lambda \bar{z}_k.t\} \quad x \notin \mathbf{FV}(t) \wedge \sigma = [\lambda \bar{z}_k.t/x]}{S \cup \{\lambda \bar{z}_k.x(\bar{z}_k) \doteq \lambda \bar{z}_k.t\}} \text{ (Bind)} \\
\frac{S \cup \{x \doteq u, \lambda \bar{z}_k.x^\alpha(\bar{s}_n) \doteq \lambda \bar{z}_k.f(\bar{t}_m)\} \quad u \in \mathbf{PB}(f, \alpha)}{S \cup \{\lambda \bar{z}_k.x^\alpha(\bar{s}_n) \doteq \lambda \bar{z}_k.f(\bar{t}_m)\}} \text{ (Imitate)}^1 \\
\frac{S \cup \{x \doteq u, \lambda \bar{z}_k.x^\alpha(\bar{s}_n) \doteq \lambda \bar{z}_k.a(\bar{t}_m)\} \quad 0 < i \leq k, u = \mathbf{PB}(i, \alpha)}{S \cup \{\lambda \bar{z}_k.x^\alpha(\bar{s}_n) \doteq \lambda \bar{z}_k.a(\bar{t}_m)\}} \text{ (Project)}^2 \\
\frac{\perp}{S \cup \{\lambda \bar{z}_k.f(\bar{s}_n) \doteq \lambda \bar{z}_k.g(\bar{t}_n)\}} \text{ (Symbol-Clash)}^{3,4}
\end{array}$$

1. where  $f \in \Sigma$ .
2. where either  $a \in \Sigma$  or  $a = z_i$  for some  $0 < j \leq k$ .
3. where  $f, g \in \Sigma$  and  $f \neq g$ .
4. this rule is redundant with regard to soundness and completeness but has implications with regard to termination and appears in [9].

**Fig. 1.** PUA- Huet's pre-unification procedure

**Theorem 2 (Completeness of PUA [9]).** *If  $\theta \in \mathbf{PreUnifiers}(S)$  for a unification system  $S$ , then there exists a pre-solved system  $S'$ , which is obtainable from  $S$  using PUA such that  $\sigma_{S'}|_{\mathbf{FV}(S)} \leq_s \theta$ .*

*Remark 3.* The procedure PUA 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 [22] 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. The first case is a "don't-care" non-determinism while the second is a "don't-know" non-determinism. We will use this fact in the rest of the paper and allow ourselves to choose specific equations to process without harming completeness.

### 3 The refinement procedure

In order to simplify definitions and proofs, we consider only first-order functions symbols of arbitrary arity and second-order unary variables. Note that even very simple second-order unification classes are undecidable [8]. An extension to the general second-order case is straightforward. We discuss the possibility to extend the method to the general higher-order case in the conclusion.

In this section we will be interested in trying to obtain failure information from cyclic equations.

### 3.1 Cyclic equations and their properties

Let the relations  $x < y$  and  $x = y$  be defined for equations  $C(\overline{x t_n}) \doteq D(\overline{y s_m})$  where  $C$  and  $D$  are contexts and where  $\text{mpath}(C) < \text{mpath}(D)$  and  $\text{mpath}(C) = \text{mpath}(D)$  respectively. Define a partial order over variables by the transitive closure of the union of the two relations (under further restrictions on the symmetry of  $=$ , see [13],[21] for a full definition). A set of equations is *cyclic* if the partial order generated over the set contains the relation  $x < x$  for some variable  $x$  occurring in the set. An example of a cycle is the set  $\{xa \doteq f(yc, b), g(yd) \doteq g(ze), zb \doteq xb\}$ . Cycles capture the idea that using PUA, one can, in some cases, obtain again (a variation of) the original set of equations.

The next result exemplifies the role of cycles in the non-termination of second-order unification.

**Theorem 4 (Levy [13]).** *It is decidable whether a second-order unification problem not containing cycles has a unifier.*

In this paper we will focus on a certain kind of second-order cycles of the following form.

**Definition 5 (Cyclic equations).** *Let  $e$  be an equation of the form  $\lambda \overline{z_n}.x_0 t \doteq \lambda \overline{z_n}.C(x_0 s)$ .  $e$  is called a cyclic equation where  $C$  is a context.  $t, s$  and  $C$  may contain the variables  $\overline{z_n}$  but not the variable  $x_0$ . We denote the fact that  $e$  is cyclic by the predicate  $\text{cyclic}(e)$ .*

We next prove that the restriction on  $C$  not to contain  $x_0$  can be avoided.

**Lemma 6.** *Let  $e$  be an equation  $\lambda \overline{z_n}.x_0 t \doteq \lambda \overline{z_n}.C(x_0 s)$  and assume further, without loss of generality, that for all occurrences of  $x_0$  in  $C$ , the sizes of their positions are not smaller than the size of the position of the hole in  $C$  (otherwise, define the hole to be the position of the minimal such occurrence). Then we can obtain, using the rules of PUA, an equation  $\lambda \overline{z_n}.w_0 t \doteq \lambda \overline{z_n}.C'(w_0 s)$  where  $w_0$  does not occur in  $C'$  for some context  $C'$ .*

**Definition 7 (Progressive context).** *Given a cyclic equation  $e$ , where  $C = C_1 \dots C_m$  such that for all  $0 < i \leq m$ ,  $C_i = f_i(r_i^1, \dots, [\cdot], \dots, r_i^{n_i})$  where  $n_i = \text{ar}(f_i) - 1$ . Define also, for all  $m < i$ ,  $C_i = f_k(y_{i-m}^1 s, \dots, [\cdot], \dots, y_{i-m}^{n_k} s)$  where  $k = ((i - 1) \bmod m) + 1$  and  $y_{i-m}^j$  for  $0 < j \leq n_k$  are new variables. We define the progressive context  $D_i^e$  for all  $0 \leq i$  as follows:*

- for all  $0 \leq i$ ,  $D_i^e = C_{i+1} \dots C_{i+m}$ .

We will use the cycle  $x_0 t \doteq f(r_1, g(x_0 s, r_2))$  as a running example. This cycle is interesting as it has instances which are unifiable and instances which are not unifiable. For the unifiable ones, both Huet's procedure and the one presented here will compute a complete set of pre-unifiers. For the non-unifiable ones, Huet's will fail to terminate while our procedure, as proved in Theorem 36 and under some additional restrictions as defined in Definition 34, will terminate with failure. An example for a unifiable instance is for  $t = f(a, g(f(a, a), a))$ ,  $r_1 = a$ ,  $r_2 = a$  and  $s = f(a, a)$ . For obtaining a non-unifiable instance which corresponds to Definition 34, just replace  $s$  from the previous instance with  $f(a, b)$ .

*Example 8.* Given the cycle  $x_0 t \doteq f(r_1, g(x_0 s, r_2))$  (having  $m = 2$ ), its progressive contexts for  $0 \leq i \leq 2$  are

$$- D_0 = C_1 C_2, D_1 = C_2 C_3 \text{ and } D_2 = C_3 C_4$$

where

$$\begin{aligned} - C_1 &= f(r_1, [.]) \\ - C_2 &= g([.], r_2) \\ - C_3 &= f(y_1, [.]) \\ - C_4 &= g([.], y_2) \end{aligned}$$

In the rest of this paper,  $e$  will refer to equations of this form and  $t, s, C, m, k, n_i, r_i^j$  and  $y_i^j$  will refer to the corresponding values in  $e$ .

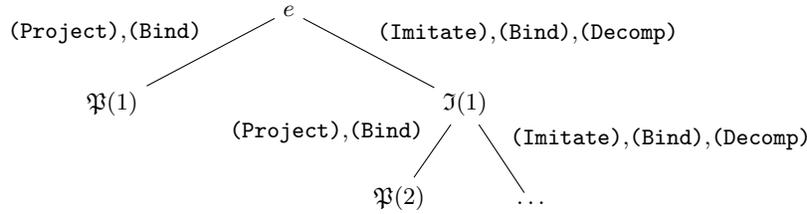
As mentioned in remark 3, the "don't-know" non-determinism affects the completeness of PUA and it is not hard to see that it is also the cause of its non-termination. The way to improve termination and to define additional decidable classes will depend, therefore, on refining the possible "don't-know" choices allowed in the search.

We will first define the result of applying (Imitate) and (Project) (plus some additional deterministic rules) on cyclic equations.

**Definition 9 ( $\mathfrak{I}$  and  $\mathfrak{P}$ ).** Given a cyclic equation  $e$ , for all  $0 \leq i$ , we define  $\mathfrak{I}(i)$ ,  $\mathfrak{I}^*(i)$  and  $\mathfrak{P}(i)$  inductively as follows:

- $\mathfrak{P}(0) = \mathfrak{I}(0) = \mathfrak{I}^*(0) = \emptyset$ .
- if  $0 < i \leq m$  then  $\mathfrak{I}^*(i) = \mathfrak{I}^*(i-1) \cup \{\lambda \bar{z}_n. y_i^j t \doteq \lambda \bar{z}_n. r_i^j \mid 1 \leq j \leq n_i\}$ .
- if  $m < i$  then  $\mathfrak{I}^*(i) = \mathfrak{I}^*(i-1) \cup \{\lambda \bar{z}_n. y_i^j t \doteq \lambda \bar{z}_n. y_{i-m}^j s \mid 1 \leq j \leq n_i\}$ .
- for all  $0 < i$ ,  $\mathfrak{I}(i) = \mathfrak{I}^*(i) \cup \{\lambda \bar{z}_n. x_i t \doteq \lambda \bar{z}_n. D_i^e(x_i s)\}$ .
- for all  $0 < i$ ,  $\mathfrak{P}(i) = \mathfrak{I}^*(i-1) \cup \{\lambda \bar{z}_n. t \doteq \lambda \bar{z}_n. D_{i-1}^e(s)\}$ .

Using these definitions, one can now describe the search conducted by PUA graphically as can be seen in Figure 2.



**Fig. 2.** The "don't-know" non-determinism in PUA

*Example 10.* Extending Example 8, we get the following values for  $0 < i \leq 3$ :

- $\mathfrak{P}(1)$  is  $t \doteq f(r_1, g(s, r_2))$  which is equivalent to  $t \doteq D_0(s)$ .
- $\mathfrak{J}(1)$  is  $\{x_1t \doteq g(f(y_1s, x_1s), r_2), y_1t \doteq r_1\}$  which is equivalent to  $\{x_1t \doteq D_1(x_1s), y_1t \doteq r_1\}$ .
- $\mathfrak{P}(2)$  is  $\{t \doteq g(f(y_1s, s), r_2), y_1t \doteq r_1\}$  which is equivalent to  $\{t \doteq D_1(s), y_1t \doteq r_1\}$ .
- $\mathfrak{J}(2)$  is  $\{x_2t \doteq f(y_1s, g(x_2s, y_2s)), y_2t \doteq r_2, y_1t \doteq r_1\}$  which is equivalent to  $\{x_2t \doteq D_2(x_2s), y_2t \doteq r_2, y_1t \doteq r_1\}$ .
- $\mathfrak{P}(3)$  is  $\{t \doteq D_2(s), y_2t \doteq r_2, y_1t \doteq r_1\}$
- $\mathfrak{J}(3)$  is  $\{x_3t \doteq D_3(x_3s), y_3t \doteq y_1s, y_2t \doteq r_2, y_1t \doteq r_1\}$

The correctness of this description is proved next.

**Lemma 11.** *Let  $e$  be a cyclic equation, then, up to the renaming of the free variables and for all  $0 \leq i$ , the application of **(Imitate)**, **(Bind)** and **(Decomp)** on  $\lambda \bar{z}_n. x_i t \doteq \lambda \bar{z}_n. D_i^e(x_i s)$  results in a set of equations containing  $\lambda \bar{z}_n. x_{i+1} t \doteq \lambda \bar{z}_n. D_{i+1}^e(x_{i+1} s)$ .*

We call this cycle the *principle cycle* of the application of **(Imitate)**.

**Lemma 12.** *Let  $S \cup \{e\}$  be a unification problem where  $e$  is a cyclic equation. Then, there is a substitution  $\tau$  such that  $\text{FV}(S \cup \{e\}) \cap \text{dom}(\tau) = \{x_0\}$  and such that the following holds, up to the renaming of the free variables:*

- assume we repeatedly apply  $i$  times **(Imitate)**, **(Bind)** and **(Decomp)** on  $e$  and the generated principal cycles, then the obtained unification problem is  $(S \cup \mathfrak{J}(i))\tau$ .
- assume we apply a **(Project)** and **(Bind)** after  $i - 1$  applications of **(Imitate)**, **(Bind)** and **(Decomp)** on  $e$  and the generated principal cycles, then the obtained problem is  $(S \cup \mathfrak{P}(i))\tau$ .

A simple but crucial fact that will enable us to enlarge the non-regular sets of solutions of PUA into regular supersets is the following.

**Proposition 13.** *Let  $S$  be a unification problem and let  $S' \subset S$ , then:*

- $S$  is unified by a substitution  $\sigma$  only if  $S'$  is unified by  $\sigma$ .
- $\text{PreUnifiers}(S) \subseteq \text{PreUnifiers}(S')$ .

We can now prove that each derivation of  $e$  must, at some point, use the above sequence of rules.

**Lemma 14.** *For any  $S$ ,  $\sigma \in \text{PreUnifiers}(S \cup \{e\})$  iff there is  $0 < i$  and substitutions  $\theta$  and  $\tau$  such that  $\theta \in \text{PreUnifiers}((S \cup \{e\} \cup \mathfrak{P}(i))\tau)$ ,  $\theta|_{\text{FV}(S \cup \{e\})} \leq_s \sigma$  and  $\tau|_{\text{FV}(S \cup \{e\})} \leq_s \sigma$ .*

By taking  $\theta \circ \tau$ , the next corollary follows immediately.

**Corollary 15.** *For any  $S$ ,  $\sigma \in \text{PreUnifiers}(S \cup \{e\})$  only if there is  $0 < i$  and a substitution  $\theta$  such that  $\theta \in \text{PreUnifiers}(S \cup \{e\} \cup \mathfrak{P}(i))$ , and  $\theta|_{\text{FV}(S \cup \{e\})} \leq_s \sigma$ .*

The definitions of the generated sets  $\mathfrak{P}(i)$  for all  $0 < i$ , are given inductively. We notice that the sets, for  $i > m$ , are made of two components:

- the inductive part which includes all equations  $\{\lambda\bar{z}_n.y_l^j t \doteq \lambda\bar{z}_n.y_{l-m}^j s \mid 1 \leq j \leq n_k\}$ , for all  $m < l \leq i$ .
- the base part which includes the equations  $\{\lambda\bar{z}_n.t \doteq \lambda\bar{z}_n.D_{i-1}^e(s)\}$  and  $\{\lambda\bar{z}_n.y_l^j t \doteq \lambda\bar{z}_n.r_l^j \mid 1 \leq j \leq n_k\}$  for just  $0 < l \leq m$ .

We will use this distinction in the next section.

### 3.2 The refinement procedure

In this section we will show how to obtain a superset of all unifiers of a cycle such that this superset will not be defined inductively. This will allow us to give a finite representation of this set which will be used in order to improve termination.

The next sets are constructed without the inductive part mentioned earlier.

**Definition 16** ( $\mathfrak{P}^-$ ). *Given a cyclic equation  $e$ , we define  $\mathfrak{P}^-$  for all  $0 < i$  as follows:*

- if  $0 < i \leq m + 1$  then  $\mathfrak{P}^-(i) = \mathfrak{P}(i)$ .
- if  $m + 1 < i$  then  $\mathfrak{P}^-(i) = \mathfrak{I}^*(m) \cup \{\lambda\bar{z}_n.t \doteq \lambda\bar{z}_n.D_{i-1}^e(s)\}$ .

*The equation  $\lambda\bar{z}_n.t \doteq \lambda\bar{z}_n.D_{i-1}^e(s)$  is called the projected equation of  $\mathfrak{P}^-(i)$ .*

As can be seen from the definition, for  $i > m$ ,  $\mathfrak{P}^-$  is defined in a non-inductive way as it depends on a fixed set  $\mathfrak{I}^*(m)$ .

*Example 17.* The values for the equation from Example 8 for  $i = 3, 4, 5, 6, 7$  are:

- $\mathfrak{P}^-(3) = \{t \doteq D_2(s), y_2 s \doteq r_2, y_1 s \doteq r_1\} = \mathfrak{P}(3)$ .
- $\mathfrak{P}^-(4) = \{t \doteq D_3(s), y_2 s \doteq r_2, y_1 s \doteq r_1\} \subseteq \mathfrak{P}(4)$ .
- $\mathfrak{P}^-(5) = \{t \doteq D_4(s), y_2 s \doteq r_2, y_1 s \doteq r_1\} \subseteq \mathfrak{P}(5)$ .
- $\mathfrak{P}^-(6) = \{t \doteq D_5(s), y_2 s \doteq r_2, y_1 s \doteq r_1\} \subseteq \mathfrak{P}(6)$ .
- $\mathfrak{P}^-(7) = \{t \doteq D_6(s), y_2 s \doteq r_2, y_1 s \doteq r_1\} \subseteq \mathfrak{P}(7)$ .

Together with Proposition 13 and Corollary 15, we can now prove two lemmas asserting that these new sets are indeed complete supersets of unifiers.

**Lemma 18.** *For all  $0 < i$ ,  $\mathfrak{P}^-(i) \subseteq \mathfrak{P}(i)$ .*

**Lemma 19.** *For all  $S$  and for all  $0 < i$ ,  $\text{PreUnifiers}(S \cup \mathfrak{P}(i)) \subseteq \text{PreUnifiers}(S \cup \mathfrak{P}^-(i))$ .*

The fact that the sets  $\mathfrak{P}^-(i)$ , for  $i > m$ , are not defined in an inductive way, will enable us to simplify the description of their pre-unifiers. In the next lemma we will prove that iterating the **(Imitate)** rule beyond the first  $3m$  iterations gives no further information about the unifiability of the set.

**Lemma 20.** *For all  $S$  and for all  $3m < i$ ,  $S \cup \mathfrak{P}^-(i)$  is unifiable iff  $S \cup \mathfrak{P}^-(i-m)$  is unifiable. Moreover, if  $\sigma$  is a pre-unifier of  $S \cup \mathfrak{P}^-(i-m)$ , then  $\sigma'$  is a pre-unifier of  $S \cup \mathfrak{P}^-(i)$  where  $\text{dom}(\sigma') = \text{dom}(\sigma) \setminus \{y_l^j \mid 0 < j \leq n_k, i-2m \leq l < i-m\} \cup \{y_l^j \mid 0 < j \leq n_k, i-m \leq l < i\}$  and  $\sigma'(y_l^j) = \sigma(y_{l-m}^j)$  for all  $i-m \leq l < i$  and  $0 < j \leq n_k$  where  $k = ((i-1) \bmod m) + 1$ .*

The intuition behind this lemma is demonstrated in the following example.

*Example 21.* Take  $\mathfrak{P}^-(5)$  and  $\mathfrak{P}^-(7)$  (remember that  $m = 2$ ) from Example 17:

$$\begin{aligned} - \mathfrak{P}^-(5) &= \{t \doteq f(y_3s, g(s, y_4s)), y_2s \doteq r_2, y_1s \doteq r_1\}. \\ - \mathfrak{P}^-(7) &= \{t \doteq f(y_5s, g(s, y_6s)), y_2s \doteq r_2, y_1s \doteq r_1\}. \end{aligned}$$

The two pairs of variables  $y_3, y_4$  and  $y_5, y_6$  occur only once in both sets.

Next, we prove that the supersets of pre-unifiers for  $e$  can be restricted by computing pre-unifiers for the problems  $\mathfrak{P}^-(i)$  for  $0 < i \leq 3m$ . This will establish the minimality property which is required for proving termination.

**Lemma 22.** *For any  $S$  and for any  $\sigma \in \text{PreUnifiers}(S \cup \{e\})$ , there is  $0 < i \leq 3m$  and  $\theta \in \text{PreUnifiers}(S \cup \{e\} \cup \mathfrak{P}^-(i))$  such that  $\theta|_{\text{FV}(S \cup \{e\})} \leq_s \sigma$ .*

The following is a corollary of the previous lemma. This result states that termination can be achieved on some problems, even if their sets of solutions is irregular.

**Corollary 23.** *Given a set  $S$  and a cycle  $e$ . If, for all  $0 < i \leq 3m$ ,  $\text{PreUnifiers}(\mathfrak{P}^-(i)) = \emptyset$ , then  $S \cup \{e\}$  is not unifiable.*

As an example of applying the above corollary, consider the following instance of our running example.

*Example 24.* Given the cycle  $x_0(f(a, g(a, a), a)) \doteq f(a, g(x_0(f(a, b)), a))$ , none of the  $\mathfrak{P}^-(i)$  for  $0 < i \leq 6$  are unifiable. Using the above corollary, we can conclude that this problem is not unifiable.

We will proceed next to the refinement of PUA but first, we need to modify unification equations and the predicate `cyclic`. This modification is required in order to apply the refinement at most once per cyclic equation.

**Definition 25 (Marked equations).** *Given a unification equation  $\lambda \bar{z}_n.t \doteq \lambda \bar{z}_n.s$ , let  $\lambda \bar{z}_n.t \doteq^{\bullet} \lambda \bar{z}_n.s$  be its marked version. The function `cyclic` now fails if  $e$  is marked.*

The idea of the following procedure is the following. When running on a unifiable problem, the extra equations added by the `(Cycle)` rule will also be unifiable for some  $0 < i \leq 3m$  according to Lemma 22. On the contrary, when a problem is not unifiable, the generated sets  $\mathfrak{P}^-(i)$  must all be processed before any rule is applied to  $e$ . If none is unifiable, we get on all branches of the search `(Symbol-Clash)` failure nodes and therefore will not apply any further rule to

$e$  and the procedure will terminate. Corollary 23 also tells us that in that case the problem is indeed not unifiable. In the case the problem is not unifiable but some set  $\mathfrak{P}^-(i)$  is, we will proceed with the unifiability of  $e$ , which might not terminate.

**Definition 26 (RPUA).** *The procedure RPUA has the same set of rules as PUA (see Fig. 1) but has, in place of (Imitate) and (Project), the rules in Fig. 3. In addition, all rules apply to marked and unmarked equations in the same way.*

$$\frac{S \cup \{x \doteq u, \lambda \bar{z}_k . x^\alpha(\bar{s}_n) \doteq \lambda \bar{z}_k . f(\bar{t}_m)\}}{S \cup \{\lambda \bar{z}_k . x^\alpha(\bar{s}_n) \doteq \lambda \bar{z}_k . f(\bar{t}_m)\}} \quad u \in \text{PB}(f, \alpha) \wedge \neg \text{cyclic}(e) \quad (\text{Imitate})^1$$

$$\frac{S \cup \{x \doteq u, \lambda \bar{z}_k . x^\alpha(\bar{s}_n) \doteq \lambda \bar{z}_k . a(\bar{t}_m)\}}{S \cup \{\lambda \bar{z}_k . x^\alpha(\bar{s}_n) \doteq \lambda \bar{z}_k . a(\bar{t}_m)\}} \quad 0 < i \leq k, u = \text{PB}(i, \alpha) \wedge \neg \text{cyclic}(e) \quad (\text{Project})^2$$

$$\frac{S \cup \{\lambda \bar{z}_k . x^\alpha(\bar{s}_n) \doteq \bullet \lambda \bar{z}_k . a(\bar{t}_m)\} \cup \mathfrak{P}^-(i)}{S \cup \{\lambda \bar{z}_k . x^\alpha(\bar{s}_n) \doteq \lambda \bar{z}_k . a(\bar{t}_m)\}} \quad 0 < i \leq 3m \wedge \text{cyclic}(e) \quad (\text{Cycle})^2$$

1. where  $f \in \Sigma$  and  $e = \lambda \bar{z}_k . x^\alpha(\bar{s}_n) \doteq \lambda \bar{z}_k . f(\bar{t}_m)$ .
2. where either  $a \in \Sigma$  or  $a = z_i$  for some  $0 < j \leq k$  and  $e = \lambda \bar{z}_k . x^\alpha(\bar{s}_n) \doteq \lambda \bar{z}_k . a(\bar{t}_m)$

**Fig. 3.** RPUA- Pre-unification with refined termination

### 3.3 The correctness of the refinement

In this section we prove the soundness and completeness of the procedure. Both are proved relatively to PUA.

**Theorem 27 (Soundness of RPUA).** *If  $S'$  is obtained from a unification system  $S$  using RPUA and is in pre-solved form, then  $\sigma_{S'}|_{\text{FV}(S)} \in \text{PreUnifiers}(S)$ .*

For proving the completeness of RPUA, we need one more definition.

**Definition 28 (Imitation blocks).** *Let  $D$  be a derivation in PUA, and let  $e$  be an unmarked cyclic equation. The imitation block for  $e$  in  $D$  is the following inductive set:*

- $e$  is in the imitation block.
- if there is an application of (Imitate) on an equation in the block, then its principal cycle is also in the block.

The size of the block is the size of the set plus 1.

The intuition behind this definition is that imitation blocks help us reconstruct, out of some arbitrary derivation, the exact  $i$  for constructing  $\mathfrak{P}^-(i)$ .

**Theorem 29 (Completeness of RPUA).** *If  $\theta$  is a pre-unifier of a unification system  $S$ , then there exists a pre-solved system  $S'$ , which is obtainable from  $S$  using RPUA such that  $\sigma_{S'}|_{\text{FV}(S)} \leq_s \theta$ .*

### 3.4 Termination and decidability results

The most interesting property of RPUA is that it terminates on some cases where PUA does not and, at the same time, has no additional asymptotic complexity. We will investigate these two claims next.

We first prove that RPUA terminates on at least all problems on which PUA terminates.

**Theorem 30.** *Let  $S$  be a unification system, then PUA terminates on it only if RPUA does.*

We now prove that RPUA terminates, in contrast to PUA, on more classes of problems.

As noted above, in order for RPUA to terminate on problems on which PUA does not terminate, one must use the eager strategy of, upon calling (Cycle), attempting to unify all generated sets  $\mathfrak{P}^-(i)$  before applying any rule to  $e$ . In order for RPUA not to compute unnecessary steps, we will also add a constraint on the calls to (Imitate) and (Project).

**Definition 31 (Possible pairs).** *Given a problem  $S \cup \{e\} \cup \mathfrak{P}^-(i)$ , an equation  $e'$  derived from  $e$  and an equation  $e''$  derived from  $\mathfrak{P}^-(i)$  are paired if  $e'$  was derived from the set generated by applying (Imitate)  $i$  times on  $e$  and the generated principle cycles, then applying one (Project) and then following the rule applications used for deriving  $e''$  from  $\mathfrak{P}^-(i)$ .*

The intuition behind possible pairs, as demonstrated in the following example, is that one can optimize the execution of the procedure by applying the same rules to pairs of equations.

*Example 32.* Consider the equation from previous examples and consider the application of (Cycle) with  $i = 4$ , so  $\mathfrak{P}^-(4) = \{t \doteq D_3(s), y_2s \doteq r_2, y_1s \doteq r_1\}$ . After applying 3 times (Imitate) and a (Project) on  $e$  and its generated principal cycles (among other rules), we obtain  $\{t \doteq D'_3(s), y'_2s \doteq r_2, y'_1s \doteq r_1, y'_3t \doteq y'_1s\}$  where  $D'_3$  is equal to  $D_3$  except for the renaming of the free variables. Then the following are possible pairs:

- $t \doteq D'_3(s)$  and  $t \doteq D_3(s)$ .
- $y'_2s \doteq r_2$  and  $y_2s \doteq r_2$ .
- $y'_1s \doteq r_1$  and  $y_1s \doteq r_1$ .

Note that the equation  $y'_3t \doteq y'_1s$ , has no possible pair.

**Definition 33 (RPUA strategy).** *When running RPUA, we require the following strategies:*

- Given an unmarked cyclic equation, do the following:
  - let  $i = 1$ .
  - apply (Cycle) with  $i$ .
  - exhaustively apply RPUA on the equations in  $\mathfrak{P}^-(i)$ .

- if a pre-solved form is found, break. Otherwise increment  $i$  by 1 (as long as  $i \leq 3m$ ).
  - try to apply (*Symbol-Clash*) on the current problem.
- Always apply the same (*Project*) or (*Imitate*) on both equations in a possible pair.

**Theorem 34 (Correctness of the strategy).** *RPUA with the strategy is sound and complete.*

We can now define a new class of second-order unification problems and show it to be decidable when using RPUA, in contrast to PUA.

**Definition 35 (Projected cycles).** *A cycle  $x_0t \doteq C(x_0s)$  is called a projected cycle if:*

1.  $t$  is ground.
2. for all positions  $p$  in  $C$  which are not on the main path of  $C$ :
  - (a)  $d(t|_p) < d(s)$ .
  - (b)  $t|_p = C|_p$ .

**Theorem 36.** *PUA does not terminate on problems containing projected cycles.*

In the next theorem, we assert that the unifiability of problems in this class can be decided using RPUA. The idea behind the proof is that, if the problem is unifiable, it is unifiable only by substitutions which map each of the variables  $y_i^j$  to terms  $\lambda z.s_i^j$  where  $z$  does not occur in  $s_i^j$ . Such substitutions will always unify the equations  $y_i^j t \doteq y_{i-m} s$  and therefore, our computed supersets are actually complete sets of unifiers.

**Theorem 37.** *RPUA decides the unification problem of projected cycles.*

### 3.5 Asymptotic analysis

In the last part of the paper we discuss the complexity of RPUA. We will measure the complexity of both procedures in the number of "don't-know" non-deterministic calls done along the derivation. A naive consideration of RPUA might suggest that it has an asymptotically exponential slow-down, in the number and size of the cyclic equations, on problems on which PUA terminates. We will show next that both procedures have the same complexity.

**Theorem 38.** *The number of "don't know" non-deterministic choices in runs of RPUA on some problem  $S$  when using the strategy is the same as in runs of PUA on  $S$ .*

## 4 Conclusion

Second-order unification problems play an important role within general higher-order unification. Many important theorems, like those in arithmetic which can be finitely axiomatized in second-order logic, require only unification over second-order formulas. Nevertheless, except for few results like the one by Levy [13] for deciding acyclic second-order unification problems, these problems are treated within the general procedures for higher-order problems. In this paper we have attempted to show that these problems are inherently simpler than general higher-order problems and that one can design for them (theoretically) improved unification procedures. We showed that for these problems, one can compute information which is static for arbitrarily long runs and which can be used in order to improve termination.

The fact that the procedure has the same asymptotic complexity does not mean that it is as efficient as Huet's. Indeed, even in the most efficient implementation where different computations of the  $\mathfrak{P}$  sets are done using the previous set information, care should be taken to back-track at the right points and those, extra machinery is required. On the other hand, this procedure might also be implemented in a more efficient way than Huet's. This can be achieved by taking advantage of the natural parallelism which is inherent in the procedure in the form of the separate computation of the  $3m$   $\mathfrak{P}$  sets. The claims in this paragraph have still to be demonstrated and the implementation of this procedure, both in a sequential form and in a parallel form, is planned using the multi-agent architecture of LEO-III [24].

An extension to higher-order logic is far from being trivial. The main difficulty is that the number of higher-order variables does not decrease when applying projections. One of the consequences is that, in contrary to the second-order case where infinite sequences of cyclic problems can only be generated by applying imitations, such sequences can also be generated using projections. Even if this obstacle can be overcome by detecting these cycles, the fact that the total number of higher-order variables does not decrease at each  $\mathfrak{P}$  set, renders our procedure ineffective.

An interesting extension to the work presented in the paper is to consider also the equations in the set  $\mathfrak{P} \setminus \mathfrak{P}^-$ . This set was considered in this paper only with regard to deciding the unification problem of projected cycles. We have started a very promising work on using this set in order to build finite tree automata which, together with unifiers of the finitely many  $\mathfrak{P}^-$  sets, can be used in order to decide the unification problem of far more complex cases than projected cycles.

## References

1. Habib Abdulrab, Pavel Goralcik, and G. S. Makanin. Towards parametrizing word equations. *ITA*, 35(4):331–350, 2001.
2. Peter Andrews, Sunil Issar, Daniel Nesmith, and Frank Pfenning. The tps theorem proving system. In Ewing Lusk and Ross Overbeek, editors, *9th International*

- Conference on Automated Deduction*, volume 310 of *Lecture Notes in Computer Science*, pages 760–761. Springer Berlin / Heidelberg, 1988. 10.1007/BFb0012885.
3. Hendrik Pieter Barendregt. *The Lambda Calculus – Its Syntax and Semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, 1984.
  4. Christoph Benzmüller, Larry Paulson, Frank Theiss, and Arnaud Fietzke. The LEO-II project. In *Proceedings of the Fourteenth Workshop on Automated Reasoning, Bridging the Gap between Theory and Practice*. Imperial College, London, England, 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. William M. Farmer. A unification algorithm for second-order monadic terms. *Annals of Pure and Applied Logic*, 39(2):131–174, 1988.
  8. William M. Farmer. Simple second-order languages for which unification is undecidable. *Theor. Comput. Sci.*, 87(1):25–41, 1991.
  9. Gérard P. Huet. A unification algorithm for typed lambda-calculus. *Theor. Comput. Sci.*, 1(1):27–57, 1975.
  10. Joxan Jaffar. Minimal and complete word unification. *Journal of the ACM (JACM)*, 37(1):47–85, 1990.
  11. Philippe Le Chenadec. The finite automaton of an elementary cyclic set. Technical Report RR-0824, INRIA, April 1988.
  12. Jordi Levy. Linear second-order unification. In *RTA*, pages 332–346, 1996.
  13. Jordi Levy. Decidable and undecidable second-order unification problems. In *RTA*, pages 47–60, 1998.
  14. G. S. Makanin. On the decidability of the theory of free groups (in russian). In *FCT*, pages 279–284, 1985.
  15. Dale Miller. Unification of simply typed lambda-terms as logic programming. In *In Eighth International Logic Programming Conference*, pages 255–269. MIT Press, 1991.
  16. Lawrence Paulson. Isabelle: The next seven hundred theorem provers. In Ewing Lusk and Ross Overbeek, editors, *9th International Conference on Automated Deduction*, volume 310 of *Lecture Notes in Computer Science*, pages 772–773. Springer Berlin / Heidelberg, 1988. 10.1007/BFb0012891.
  17. Christian Prehofer. Decidable higher-order unification problems. In *Proceedings of the 12th International Conference on Automated Deduction, CADE-12*, pages 635–649, London, UK, UK, 1994. Springer-Verlag.
  18. Manfred Schmidt-Schauß. A decision algorithm for stratified context unification. *J. Log. Comput.*, 12(6):929–953, 2002.
  19. Manfred Schmidt-Schauß and Klaus U. Schulz. On the exponent of periodicity of minimal solutions of context equation. In *RTA*, pages 61–75, 1998.
  20. 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.
  21. Manfred Schmidt-Schauß and Klaus U. Schulz. Decidability of bounded higher-order unification. *J. Symb. Comput.*, 40(2):905–954, August 2005.
  22. Wayne Snyder and Jean H. Gallier. Higher-order unification revisited: Complete sets of transformations. *J. Symb. Comput.*, 8(1/2):101–140, 1989.
  23. Wayne S. Snyder. *Complete sets of transformations for general unification*. PhD thesis, Philadelphia, PA, USA, 1988. AAI8824793.

24. Max Wisniewski, Alexander Steen, and Christoph Benzmüller. The Leo-III project. In Alexander Bolotov and Manfred Kerber, editors, *Joint Automated Reasoning Workshop and Deduktionstreffen*, page 38, 2014.
25. Marek Zaionc. The regular expression descriptions of unifier sets in the typed lambda calculus. In *Fundamenta Informaticae X*, pages 309–322. North-Holland, 1987.

## Appendix: Proofs of the theorems and lemmas stated in the paper

**Lemma 6.** *Let  $e$  be an equation  $\lambda\bar{z}_n.x_0t \doteq \lambda\bar{z}_n.C(x_0s)$  and assume further, without loss of generality, that for all occurrences of  $x_0$  in  $C$ , the sizes of their positions are not smaller than the size of the position of the hole in  $C$  (otherwise, define the hole to be the position of the minimal such occurrence). Then we can obtain, using the rules of PUA, an equation  $\lambda\bar{z}_n.w_0t \doteq \lambda\bar{z}_n.C'(w_0s)$  where  $w_0$  does not occur in  $C'$  for some context  $C'$ .*

*Proof.* Let  $k$  be the size of the position of the smallest subterm containing all occurrences of  $x_0$  and the hole in  $C$  and let  $l$  be the number of these occurrences. We prove by induction on the lexicographic ordering of  $\langle l, k \rangle$ . If  $l = 0$  we are done. Otherwise, apply **(Imitate)**, **(Bind)** and **(Decomp)** in order to obtain a set of equations containing a cycle  $\lambda\bar{z}_n.x_it \doteq \lambda\bar{z}_n.C'(f(x_1s, \dots, x_is, \dots, x_{\text{ar}(f)}s))$  where  $C = f(t_1, \dots, C'_{\text{@}_i}, \dots, t_{\text{ar}(f)})$  and  $0 < i \leq \text{ar}(f)$ . If  $k = 0$ , then it means that at least one occurrence was inside  $t_j$  for  $0 < j \leq \text{ar}(f) \wedge i \neq j$  and therefore does not occur in the generated cycle and we have less occurrences. Otherwise, we see that the size of the position of the smallest subterm was decreased by one. In both cases,  $\langle l, k \rangle$  is smaller and we can apply the induction hypothesis.  $\square$

**Lemma 11.** *Let  $e$  be a cyclic equation, then, up to the renaming of the free variables and for all  $0 \leq i$ , the application of **(Imitate)**, **(Bind)** and **(Decomp)** on  $\lambda\bar{z}_n.x_it \doteq \lambda\bar{z}_n.D_i^e(x_is)$  results in a set of equations containing  $\lambda\bar{z}_n.x_{i+1}t \doteq \lambda\bar{z}_n.D_{i+1}^e(x_{i+1}s)$ .*

*Proof.* By induction on  $i$  and according to the definition of **(Imitate)**.  $\square$

**Lemma 12.** *Let  $S \cup \{e\}$  be a unification problem where  $e$  is a cyclic equation. Then, there is a substitution  $\tau$  such that  $\text{FV}(S \cup \{e\}) \cap \text{dom}(\tau) = \{x_0\}$  and such that the following holds, up to the renaming of the free variables:*

- assume we repeatedly apply  $i$  times **(Imitate)**, **(Bind)** and **(Decomp)** on  $e$  and the generated principal cycles, then the obtained unification problem is  $(S \cup \mathfrak{I}(i))\tau$ .
- assume we apply a **(Project)** and **(Bind)** after  $i - 1$  applications of **(Imitate)**, **(Bind)** and **(Decomp)** on  $e$  and the generated principal cycles, then the obtained problem is  $(S \cup \mathfrak{P}(i))\tau$ .

*Proof.* By induction on  $i$  using Lemma 11 and according to the definition of PUA.  $\square$

**Lemma 14.** For any  $S$ ,  $\sigma \in \text{PreUnifiers}(S \cup \{e\})$  iff there is  $0 < i$  and substitutions  $\theta$  and  $\tau$  such that  $\theta \in \text{PreUnifiers}((S \cup \{e\} \cup \mathfrak{P}(i))\tau)$ ,  $\theta|_{\text{FV}(S \cup \{e\})} \leq_s \sigma$  and  $\tau|_{\text{FV}(S \cup \{e\})} \leq_s \sigma$ .

*Proof.* First, if  $\sigma \in \text{PreUnifiers}(S \cup \{e\})$ , then  $\sigma \in \text{PreUnifiers}(S \cup \{e, e\})$ . Using Lemma 12, remark 3 and the fact that only **(Imitate)** and **(Project)** can be applied to  $e$ , we know that there is an  $0 < i$  such that a solved form can only be derived via a system  $(S \cup \{e\} \cup \mathfrak{P}(i))\tau$  or  $(S \cup \{e\} \cup \mathfrak{J}(i))\tau$  for some substitution  $\tau$ . These two sets are unifiable as well and therefore we can choose a substitution  $\iota$ , such that  $\sigma = \iota \circ \tau|_{S \cup \{e\}}$  and therefore,  $\tau|_{S \cup \{e\}} \leq_s \sigma$ . Assume now, on the contrary, that the derivation never reaches  $(S \cup \{e\} \cup \mathfrak{P}(i))\tau$ . Lemma 11 tells us that the application of **(Imitate)** on a principal cycle always results in a set containing a principal cycle. Without applying a **(Project)** step on the cycle, it will never reach a pre-solved form and we get a contradiction. So, we know that if there is a derivation of a pre-solved form, it must pass through  $\mathfrak{P}(i)$  for some  $0 < i$ . Using Theorem 2, we obtain that there is actually such a derivation ending in  $S'$  such that  $\sigma_{S'}|_{S \cup \{e\}} \leq_s \sigma$ . Using Theorem 1, we obtain that, since the derivation is also a derivation of  $(S \cup \{e\} \cup \mathfrak{P}(i))\tau$ , there is a substitution  $\theta = \sigma_{S'}|_{\text{FV}((S \cup \{e\} \cup \mathfrak{P}(i))\tau)}$ , such that  $\theta \in \text{PreUnifiers}((S \cup \{e\} \cup \mathfrak{P}(i))\tau)$  and therefore  $\theta \circ \tau \in \text{PreUnifiers}(S \cup \{e\} \cup \mathfrak{P}(i))$ . Since  $\text{FV}(S \cup \{e\}) \subseteq \text{FV}(S \cup \{e\} \cup \mathfrak{P}^-(i))$ , we obtain that  $\theta \circ \tau|_{\text{FV}(S \cup \{e\})} \leq_s \sigma$  and therefore  $\theta|_{\text{FV}(S \cup \{e\})} \leq_s \sigma$ . For the other direction, if  $\theta \in \text{PreUnifiers}((S \cup \{e\} \cup \mathfrak{P}(i))\tau)$ , then  $\theta \circ \tau$  unifies  $S \cup \{e\} \cup \mathfrak{P}(i)$  and according to Proposition 13,  $\theta \circ \tau$  unifies also  $S \cup \{e\}$ . Clearly,  $\theta|_{\text{FV}(S \cup \{e\})} \leq_s \theta \circ \tau$  as well as  $\tau|_{\text{FV}(S \cup \{e\})} \leq_s \theta \circ \tau$ .  $\square$

**Lemma 18.** For all  $0 < i$ ,  $\mathfrak{P}^-(i) \subseteq \mathfrak{P}(i)$ .

*Proof.* Clear from the definition of  $\mathfrak{P}^-$ .  $\square$

**Lemma 19.** For all  $S$  and for all  $0 < i$ ,  $\text{PreUnifiers}(S \cup \mathfrak{P}(i)) \subseteq \text{PreUnifiers}(S \cup \mathfrak{P}^-(i))$ .

*Proof.* Follows from Proposition 13 and Lemma 18.  $\square$

**Lemma 20.** For all  $S$  and for all  $3m < i$ ,  $S \cup \mathfrak{P}^-(i)$  is unifiable iff  $S \cup \mathfrak{P}^-(i-m)$  is unifiable. Moreover, if  $\sigma$  is a pre-unifier of  $S \cup \mathfrak{P}^-(i-m)$ , then  $\sigma'$  is a pre-unifier of  $S \cup \mathfrak{P}^-(i)$  where  $\text{dom}(\sigma') = \text{dom}(\sigma) \setminus \{y_l^j \mid 0 < j \leq n_k, i-2m \leq l < i-m\} \cup \{y_l^j \mid 0 < j \leq n_k, i-m \leq l < i\}$  and  $\sigma'(y_l^j) = \sigma(y_{l-m}^j)$  for all  $i-m \leq l < i$  and  $0 < j \leq n_k$  where  $k = ((i-1) \bmod m) + 1$ .

*Proof.* The only difference between  $S \cup \mathfrak{P}^-(i-m)$  and  $S \cup \mathfrak{P}^-(i)$  is that the variables  $y_{l-m}^j$  for  $0 < j \leq n_k$  and  $i-m \leq l < i$  are named  $y_l^j$ . This can be seen as both sets of variables do not occur in the other set nor in the  $\mathfrak{J}^*(m)$  component of  $\mathfrak{P}^-(i)$  of their own respective set. Therefore,  $\sigma'$  is a pre-unifier of  $S \cup \mathfrak{P}^-(i)$ .  $\square$

**Lemma 22.** For any  $S$  and for any  $\sigma \in \text{PreUnifiers}(S \cup \{e\})$ , there is  $0 < i \leq 3m$  and  $\theta \in \text{PreUnifiers}(S \cup \{e\} \cup \mathfrak{P}^-(i))$  such that  $\theta|_{\text{FV}(S \cup \{e\})} \leq_s \sigma$ .

*Proof.* Let  $\sigma \in \text{PreUnifiers}(S \cup \{e\})$ , then Corollary 15 says that there is  $0 < i$  and  $\theta \in \text{PreUnifiers}(S \cup \{e\} \cup \mathfrak{P}(i))$  such that  $\theta|_{\text{FV}(S \cup \{e\})} \leq_s \sigma$ . Lemma 19 says that  $\theta \in \text{PreUnifiers}(S \cup \{e\} \cup \mathfrak{P}^-(i))$ . We next prove that there is  $0 < j \leq 3m$  and  $\theta' \in \text{PreUnifiers}(S \cup \{e\} \cup \mathfrak{P}^-(j))$  such that  $\theta'|_{\text{FV}(S \cup \{e\})} = \theta|_{\text{FV}(S \cup \{e\})}$ . We proceed by induction on  $i$ . If  $0 < i \leq 3m$ , then we are done. Otherwise, we use the induction hypothesis to assume  $\theta'' \in \text{PreUnifiers}(S \cup \{e\} \cup \mathfrak{P}^-(i-m))$  for  $\theta''|_{\text{FV}(S \cup \{e\})} = \theta'|_{\text{FV}(S \cup \{e\})}$  and then Lemma 20 to conclude the induction. Since  $\theta'|_{\text{FV}(S \cup \{e\})} = \theta|_{\text{FV}(S \cup \{e\})}$ , then  $\theta'|_{\text{FV}(S \cup \{e\})} \leq_s \sigma$  and we can conclude the proof.  $\square$

**Theorem 26 (Soundness of RPUA).** *If  $S'$  is obtained from a unification system  $S$  using RPUA and is in pre-solved form, then  $\sigma_{S'}|_{\text{FV}(S)} \in \text{PreUnifiers}(S)$ .*

*Proof.* We prove it relatively to the soundness of PUA. Let  $S'$  be a pre-solved form obtained from  $S$  using RPUA. We proceed by the following induction on the size of the derivation. For each such  $S'$ , we can obtain a derivation, from  $S$ , of a pre-solved form in PUA. The rest follows from Theorem 1. If the size is 0, then  $S = S'$  and we are done according to the definition of pre-solved forms. Otherwise, we distinguish between the possible rule applications applied to  $S$ . For all cases when the equation is not an unmarked cyclic equation, we proceed as in PUA and can apply the induction hypothesis using the remaining, smaller, derivation in order to obtain a pre-solved form using PUA. In the case where we have an unmarked cyclic equation, the first application is (Cycle), obtaining  $S \cup \mathfrak{P}^-(i)$  for some  $0 < i$ , for which there is a smaller derivation in RPUA. We use the induction hypothesis in order to obtain a derivation of some  $S'$  from  $S \cup \mathfrak{P}^-(i)$  in PUA. According to Theorem 1,  $\sigma_{S'}|_{\text{FV}(S \cup \mathfrak{P}^-(i))} \in \text{PreUnifiers}(S \cup \mathfrak{P}^-(i))$  and according to Proposition 13,  $\sigma_{S'}|_{\text{FV}(S)} \in \text{PreUnifiers}(S)$ .  $\square$

**Theorem 28 (Completeness of RPUA).** *If  $\theta$  is a pre-unifier of a unification system  $S$ , then there exists a pre-solved system  $S'$ , which is obtainable from  $S$  using RPUA such that  $\sigma_{S'}|_{\text{FV}(S)} \leq_s \theta$ .*

*Proof.* We will prove completeness by the following inductive argument on  $n$ . If  $S$  is pre-unifiable by  $\theta$  and we can obtain a pre-solved form  $S'$  from  $S$  using a derivation of size  $n$  in PUA, then we can derive a pre-solved form  $S''$  from  $S$  using RPUA such that  $\sigma_{S''}|_{\text{FV}(S)} \leq_s \theta$ . If  $n = 0$  then  $S' = S$  and we are done. Otherwise, assume the above holds for all derivations of size  $n$  and let us obtain, using Theorem 2, a derivation in PUA of size  $n + 1$ . If the top rule application is not an (Imitate) or a (Project) on an unmarked cyclic equation, we can apply the same rule using RPUA and apply the induction hypothesis in order to obtain the rest of the derivation. Otherwise, the top rule is an (Imitate) or (Project) on an unmarked cyclic equation  $e$  with  $m$  defined as usual and let the imitation block containing  $e$  be of size  $i$ . We first apply (Cycle) such that if  $0 < i \leq 3m$  then  $j = i$ , otherwise  $j = 2m + ((i-1) \bmod m) + 1$ . Note, that  $j$  is chosen according to the proof of Lemma 22. We know, according to this lemma, that there is a substitution  $\tau \in \text{PreUnifiers}(S \cup \mathfrak{P}^-(j))$  such that  $\tau|_{\text{FV}(S)} \leq_s \theta$ . Let the first derivation of  $D$  be  $S \rightarrow S^1$  where the rule is either (Imitate) or (Project) on

e. Apply this rule in order to obtain  $S^1 \cup \mathfrak{P}^-(j)$ . Now use the induction hypothesis in order to obtain a derivation  $S^1 \rightarrow S'$  in RPUA and apply this derivation to  $S^1 \cup \mathfrak{P}^-(j)$  in order to obtain  $S' \cup \mathfrak{P}^-(j) \sigma_{S'}$  and as  $\tau \in \text{PreUnifiers}(S^1)$  (Proposition 13), we have that  $\sigma_{S'}|_{\text{FV}(S^1 \cup \mathfrak{P}^-(j))} \leq_s \tau$ . This implies, since (Imitate) and (Project) only add variables, that  $\sigma_{S'}|_{\text{FV}(S \cup \mathfrak{P}^-(j))} \leq_s \tau$ . Since  $S \sigma_{S'}$  can be obtained from  $S'$  by applying (Decomp) and (Delete) only ( $S'$  is a pre-solved form of  $S$ ), we have  $\text{PreUnifiers}((S \cup \mathfrak{P}^-(j)) \sigma_{S'}) = \text{PreUnifiers}(S' \cup \mathfrak{P}^-(j) \sigma_{S'})$ . Because  $S \cup \mathfrak{P}^-(j)$  is unifiable by  $\tau$  and  $\sigma_{S'}|_{\text{FV}(S \cup \mathfrak{P}^-(j))} \leq_s \tau$ , there is substitution  $\delta$  such that  $\delta \in \text{PreUnifiers}((S \cup \mathfrak{P}^-(j)) \sigma_{S'})$  and  $\delta \circ \sigma_{S'}|_{\text{FV}(S \cup \mathfrak{P}^-(j))} \leq_s \tau$ . Applying Theorem 2 we get a pre-solved form using PUA. This derivation is also shorter than  $D$  as all the equations in  $\mathfrak{P}^-(j)$  were part of  $D$  and in addition we already applied one (Project). We therefore apply the induction hypothesis in order to obtain a derivation  $S''$  in RPUA such that  $\sigma_{S''}|_{\text{FV}((S \cup \mathfrak{P}^-(j)) \sigma_{S'})} \leq_s \delta$ . Since  $\leq_s$  is closed under composition, we get  $\sigma_{S''} \circ \sigma_{S'}|_{\text{FV}((S \cup \mathfrak{P}^-(j)) \sigma_{S'})} \leq_s \delta \circ \sigma_{S'}$  and therefore,  $\sigma_{S''} \circ \sigma_{S'}|_{\text{FV}(S \cup \mathfrak{P}^-(j))} \leq_s \delta \circ \sigma_{S'}$ . Now, by transitivity of  $\leq_s$  and the fact that  $\text{FV}(S) \subseteq \text{FV}(S \cup \mathfrak{P}^-(j))$ , we get  $\sigma_{S''} \circ \sigma_{S'}|_{\text{FV}(S)} \leq_s \tau|_{\text{FV}(S)} \leq_s \theta$  and we are done.  $\square$

**Theorem 29.** *Let  $S$  be a unification system, then PUA terminates on it only if RPUA does.*

*Proof.* Let  $D$  be an infinite derivation from  $S$  in RPUA. We want to show that starting with  $S$ , one can obtain an infinite derivation in PUA as well. This will show that if a run does not terminate in RPUA, we can simulate it by PUA and therefore, the run does not terminate there as well. We can clearly simulate all rule applications in RPUA except (Cycle). Assume we have a derivation  $S \rightarrow S' \cup \{e\} \xrightarrow{(\text{Cycle})} S' \cup \{e\} \cup \mathfrak{P}^-(i) \rightarrow \dots$  in RPUA. We prove by induction on the number of unmarked cyclic equations in  $S' \cup \{e\}$  that if there is an infinite derivation from this set using RPUA, there is also one using PUA. We consider four cases:

- $e$  is unifiable - then, according to the strategy, we know that  $\mathfrak{P}^-(i)$  is unifiable and we derive  $S' \tau \cup \mathfrak{P}(i) \cup \mathfrak{P}^-(i)$  according to Lemma 12. Continuing the derivation on the two sets, we get  $S' \tau_2$ , for some  $\tau_2$ . This set can also be derived from  $S' \cup \{e\}$  using PUA and has one less unmarked cyclic equation than  $S' \cup \{e\}$ . Therefore, we can apply the induction hypothesis in order to obtain an infinite derivation in PUA.
- there is an infinite derivation from  $e$  - then we just continue the derivation on  $S' \cup \{e\}$  in PUA.
- there is an infinite derivation from  $\mathfrak{P}^-(i)$  - which means, according to the strategy, that for all  $0 < j \leq i$ ,  $\mathfrak{P}^-(j)$  terminated and was not unifiable. Therefore, we can derive, in PUA,  $S' \tau \cup \mathfrak{P}(i)$  (as it is also being done incrementally) and since  $\mathfrak{P}^-(i) \subseteq \mathfrak{P}(i)$ , this derivation will not terminate as well.
- $\mathfrak{P}^-(i)$  is not unifiable for all  $0 < i \leq 3m$  - then RPUA terminates.

**Theorem 33 (Correctness of the strategy).** *RPUA with the strategy is sound and complete.*

*Proof.* Soundness is clear as we do not change the rules. Completeness for the first part of the strategy is also clear as we only change the order of applying equations. The completeness of the second part is based on the proof of Lemma 14.  $\square$

**Theorem 35.** *PUA does not terminate on problems containing projected cycles.*

*Proof.* PUA does not terminate on  $xf(a, a) \doteq f(xga, a)$ , which is a projected cycle.  $\square$

**Theorem 36.** *RPUA decides the unification problem of projected cycles.*

*Proof.* We prove that  $\sigma \in \text{PreUnifiers}(\mathfrak{P}^-(i))$  iff  $\sigma \in \text{PreUnifiers}(\mathfrak{P}(i))$  for all  $0 < i$ . The rest follows from Lemma 14. Lemma 19 gives us the easy direction. For the reverse one, since  $\mathfrak{P}(i) = \mathfrak{P}^-(i) \cup \{\lambda \bar{z}_n. y_l^j t \doteq \lambda \bar{z}_n. y_{l-m}^j s \mid 1 \leq j \leq n_l, m < l \leq i\}$ , we only need to prove that if  $\sigma$  unifies  $\mathfrak{P}^-(i)$  then it unifies

$$\{\lambda \bar{z}_n. y_l^j t \doteq \lambda \bar{z}_n. y_{l-m}^j s \mid 1 \leq j \leq n_l, m < l \leq i\} \quad (1)$$

Consider the equation:

$$\lambda \bar{z}_n. t \doteq \lambda \bar{z}_n. D_{i-1}^e(s) \in \mathfrak{P}^-(i) \quad (2)$$

We first note that  $\mathfrak{P}^-(i)$  contains only the following new variables  $y_1^j, \dots, y_m^j$  and  $y_{i-m}^j, \dots, y_{i-1}^j$ . Let  $y_l^j$  be all the new variables occurring in  $D_{i-1}^e$  for  $i-m-1 < l \leq i-1$  and  $0 < l$ . By applying (Decomp) multiple times on Equation 2, we obtain the set of problems  $t_l^j \doteq y_l^j s$ , where  $t_l^j$  are subterms of  $t$ . Since,  $t_l^j$  is ground and  $d(t_l^j) < d(s)$ , the only possible unifier for Equation 2, is  $\sigma(y_l^j) = \lambda \bar{z}_n. t_l^j$ , where  $t_l^j$  has no occurrence of  $z_k$  for  $0 < k \leq n$ . For all  $i-m \leq l < i$  we prove by induction on  $n = (l-1)/m$  ( $l-1$  divided by  $m$  and let  $k = (l-1 \bmod m) + 1$ ) that  $\sigma$  can be extended to a unifier  $\theta$  for  $\mathfrak{P}(i)$  such that:  $\sigma(y_k^j) = \theta(y_k^j) = \theta(y_{m+k}^j) = \theta(y_{2*m+k}^j) = \dots = \theta(y_{n*m+k}^j) = \theta(y_l^j) = \sigma(y_l^j) = \lambda \bar{z}_n. t_l^j$  as  $n * m + k = l$ . All that is left to prove is that  $\sigma(y_k^j) = \lambda \bar{z}_n. t_l^j$ . Property 2b of Definition 34 tells us that  $t_{i-m}^j = r_1^j, \dots, t_{i-1}^j = r_m^j$ . We prove that  $\sigma(y_k^j) = \lambda \bar{z}_n. r_k^j$  and we are done. Assume  $d(C) > 1$ , since  $d(D_{i-1}^e) = d(C)$  and  $d(t) > d(D_{i-1}^j)$  (Property 1 and the fact that  $\sigma$  unifies Eq. 2), we get  $d(t) > d(C)$ . Now since  $d(r_k^j) < d(C)$  ( $r_k^j$  is a strict subterm of  $C$ ), we get that  $d(r_k^j) < d(t)$ . Considering the equations  $y_k^j t = r_k^j \in \mathfrak{P}^-(i)$ , we get that they are only unified if  $\sigma(y_k^j) = \lambda \bar{z}_n. r_k^j$  (same argument as for the equations  $y_l^j s = t_l^j$  above). If  $d(C) = 0$ , then the claim is true since there are no variables  $y_n^j$  in  $\mathfrak{P}(i)$ .  $\square$

**Theorem 37.** *The number of "don't know" non-deterministic choices in runs of RPUA on some problem S when using the strategy is the same as in runs of PUA on S.*

*Proof.* We will map each such choice in such a run of RPUA to a choice in the equivalent run in PUA (see the proof of Theorem 29 for the existence of such an equivalent run). There are two kinds of "don't-know" choices in RPUA:

1. choosing which  $0 < i \leq 3m$  to use when applying **(Cycle)**. Since we choose in an incremental way and since we stop once  $\mathfrak{P}^-(i)$  is unifiable, this corresponds to choosing in PUA, how many **(Imitate)** to apply on  $e$  before applying **(Project)**, which is also attempted incrementally.
2. choosing between applying **(Imitate)** and **(Project)** on an equation  $e'$ . We distinguish these choices according to the equation used to derive  $e'$  in RPUA:
  - (a) one of the equations in  $\mathfrak{P}^-(i)$  for some  $0 < i \leq 3m$ .  $e'$  forms a possible pair with an equation  $e''$  derived from the equation  $e$ , which also occurs in the derivation of PUA. In RPUA we apply each choice once per pair, which corresponds exactly to the choices in PUA.
  - (b)  $e$  - this case can be further divided according to whether  $e$  was cyclic or not:
    - i. not cyclic - in this case the rule applications in RPUA and PUA are identical.
    - ii. cyclic and let **(Cycle)** be called with parameter  $i$ . Our strategy tells us that we already managed to find a pre-solved form from  $\mathfrak{P}^-(j)$  for some  $0 < j \leq 3m$ . If  $e'$  has an optional pair among those, see case 2a. Otherwise, this equation was not among the ones derived from  $\mathfrak{P}^-(j)$  and we choose on it for the first time, which corresponds to choosing on it in the equivalent derivation in PUA.

□