Decidability of Bounded Higher-Order Unification

Manfred Schmidt-Schauß[1] & Klaus U. Schulz[2]

[1] Institut für Informatik, J.-W.-Goethe-Universität, Postfach 11 19 32,
D-60054 Frankfurt,Germany
Tel: (+49)69-798-28597, Fax: (+49)69-798-28919,
E-mail: schauss@ki.informatik.uni-frankfurt.de
[2] CIS, University of Munich, Oettingenstr 67, D-80538 München, Germany
Tel: (+49)89-2178-2700, Fax: (+49)89-2178-2701,
E-mail: schulz@cis.uni-muenchen.de

Abstract. It is shown that unifiability of terms in the simply typed lambda calculus with β and η rules becomes decidable if there is a bound on the number of bound variables and lambdas in a unifier in η-long β-normal form.

1 Introduction

First-order unification [BS94] is a fundamental operation in several areas of computer science, e.g. automated deduction, term rewriting, logic programming and type-checking. The generalization to higher-order unification increases the expressiveness, the applicability and improves the level of abstraction. This explains the interest in higher-order systems such as higher-order logics and higher-order deduction systems [And86,Pau94,GLM97,And01,Pfe01], higher-order (functional) programming languages [BMS80,Tur85,Pau91,Bar90,Bir98], higher-order logic programming languages [Mil91,HKMN95], higher-order rewriting [Nip91,Klo92,DJ90] and higher-order unification [Hue75,Dow01].

It is well-known that second-order unification — hence higher-order unification — is undecidable ([Gol81,Far91,LV00a]). Higher-order unification procedures were already described by in [Hue75,JP76]. The undecidability results triggered research into restrictions to make unification decidable. A special syntactic restriction is the unification of higher-order patterns [Mil91], which is decidable. In [SS99a,SS01] it was shown that second-order unification becomes decidable if an upper bound on the number of occurrences of bound variables in the substitution terms is fixed, which has as a corollary the well-known result that second-order unification with monadic function symbols is decidable [Hue75,Zhe79,Far88]. The monadic restriction fails to yield a decidable unification problem, if generalized to all types. Restricting third-order unification to monadic types, i.e. every function has at most one argument, was shown to be undecidable in [Nar90]. A further restriction of second-order unification that restricts the number of bound variables in the substitution terms to be

one is context unification. The conjecture is that context unification is decidable. There are several results on decidability of fragments of context unification [Com98,SS94,SS99b,SSS00,Lev96] or variants of context unification [CP97].

In this paper we generalize the result on decidability of bounded second-order unification to higher-order unification in the simply typed lambda calculus with β and η rules [Bar84,HS86]. We show that solvability of unification problems is decidable if for any variable a bound on the number of lambda-binders and occurrences of bound variables in the image of the variable under a unifier is given. Here each image is assumed to be in η-expanded β-normal form.

The result implies that undecidability proofs for higher-order unification require an unbounded number of lambda-bound variables or lambdas in a unifier in η-expanded normal form. It can be used to define a semi-decision procedure for ordinary higher-order unification where we start with given bounds for the variables in the problem and increase the bounds as long as we have an unsolvable problem.

The proof technique uses a lemma on an upper bound for the exponent of periodicity for a minimal unifier for context unification from [SSS98] which is a generalization of a lemma that appeared in the decidability proof of word unification by Makanin [Mak77]. An improvement of the latter result was given in [KP96]. This link to word unification ([Mak77,Sch90,Sch93,Gut98,Pla99]) is not accidental. The relationship between word unification and bounded higher-order unification is indicated in Figure 1 where we also mention some other problems in order to position the results of this paper.

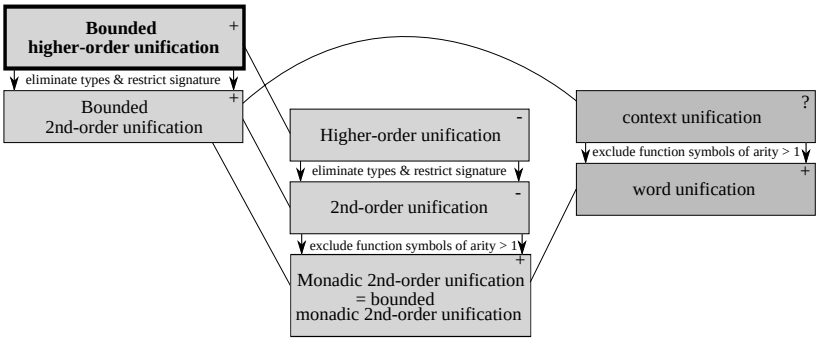


Fig. 1. Decidable (+) and undecidable (-) unification problems and their relationship.

The problems mentioned in the figure can be divided into two classes. The class of "complete structural alignment problems" comprehends word unification and context unification. The class of "functional equality problems" contains all other problems. The principal difference between both classes may be exemplified

using the equation

$$xa \doteq xb.$$

When treating the equation as a complete structural alignment problem, as it is done in word unification, we ask for a word W, representing the solution value for x, such that the complete strings Wa and Wb are identical. Obviously, this is not possible and the equation is unsolvable. From the functional equality perspective, where the equation can be rewritten in the form $x(a) = x(b)$, we ask for a function F such that $F(a) = F(b)$. Assuming that the constant function $\lambda x.a$ represents a possible value for x, the equation has a trivial solution. More generally, in complete structural alignment problems the value of each subterm occurring in an equation influences the final identity of both sides under a solution. In contrast, when solving functional equality problems, constant functions can be used and the values of arguments may become irrelevant.

The possible values for functional variables in second-order monadic unification are unary (resp. constant) term functions of the form $f(\ldots g([\cdot])\ldots)$ (resp. $f(\ldots g(a)\ldots)$) composed of unary first-order function symbols and individual constants. From one perspective, all these functions have *one argument* that has *at most one* occurrence/position. Second-order and higher-order unification problems are more general in the sense that the functions assigned to variables may have an arbitrary number of arguments, and each argument may have an arbitrary number of argument positions/occurrences. For the bounded versions, an upper limit for the number of arguments and positions is fixed. Still, since for each argument the number of occurrences/positions may be zero, solution values may be constant functions, which means that equations where both sides start with variables are always trivially solvable. This effect is heavily used in the decidability results for bounded second-order/higher-order unification given in [SS99a,SS01] and in the present paper.

In context unification [NPR97,Vor98,NTT00,SS99b,SSS00,LV00b], solution values of context variables are tree functions with one argument that has *exactly one* occurrence/position. Hence solving a context equation leads to a complete structural alignment of both sides as trees.

As we explain in Section 10.3, (bounded) second-order unification can be considered as a special case of (bounded) higher-order unification where characteristic restrictions on signatures and types are imposed. From (bounded) second-order unification we get monadic second-order unification by exclusion of function symbols of arity > 1. In the same way, word unification is obtained from context unification.

Lemmas on the exponent of periodicity, roughly, give a bound on the maximal number of periodical repetitions of words/trees that occur in the solution values of "minimal" solutions. All known decision procedures for word unification are based on periodicity bounds. For the above-mentioned "functional equality problems", periodicity bounds are used for simplifying equations of a particular kind. The idea can be illustrated using a monadic second-order equation $X(s) \doteq f(X(t))$. Obviously, under any solution X is mapped to a function of the form $f(f(\ldots([\cdot])\ldots))$. If we have an upper bound for the number of peri-

odical repetitions of f's in a minimal solution, it is possible to eliminate X in the equation, using a finite subcase analysis. A more complex, but similar argument is used in the present paper for simplifying bounded higher-order unification problems of a specific syntactic form.

The structure of the paper is as follows. We start with a brief introduction to the simply types lambda-calculus with β and η rules in Section 2. In Section 3 we formally introduce bounded higher-order unification problems, which represent the decision problems studied in this paper. Section 4 describes properties of minimal solutions of bounded higher-order unification problems that are needed for proving correctness of the decision algorithm. Section 5 introduced the notions that are needed for the decision algorithm, describes the termination order and the structure of the algorithm. The remaining Sections 6–9 introduce the specific reduction and transformation rules that are used in the decision algorithm for different types of bounded higher-order unification problems. In Section 10 we summarize the obtained results.

2 Simply Typed Lambda Calculus

We present the simply typed lambda-calculus (see [Bar84,Wol93,Hin97]).

2.1 Types and Terms

Definition 2.1. *The language of types is defined according to the grammar*

$$T ::= T_0 \mid (T \to T)$$

where $T_0 \neq \emptyset$ is the set of elementary types. *The symbols α, τ range over types, and ι ranges over elementary types.*

A shorter notation for types of the form $\tau = (\alpha_1 \to (\alpha_2 \ldots (\alpha_n \to \iota) \ldots))$ is $(\alpha_1 \to \alpha_2 \to \ldots \to \alpha_n \to \iota)$. The number n is called the *arity* of the type τ, denoted $ar(\tau)$, and ι is called the *target type* of τ.

The background signature Σ for building higher-order terms is a set of *function symbols*, where every function symbol f comes with a type $type(f)$. The arity of f is defined as $ar(f) := ar(type(f))$. Function symbols f of elementary type (i.e., $ar(f) = 0$) are called *elementary constant symbols*. We assume that Σ contains for every type τ a countably infinite set of function symbols. For every type τ there is in addition a (countably) infinite set of variables V_τ. The union of all these sets is denoted as $\mathcal{V}$. Variables are denoted as x, y, z, expressions $\vec{x}$ denote finite sequences of variables. As for function symbols, with $type(x)$ we denote the type of x. The arity $ar(x)$ of x is $ar(x) := ar(type(x))$. If necessary, the type is indicated as a superscript. Since for every type there are infinitely many variables (or function symbols, respectively), we can always use fresh variables (or function symbols, respectively). A variable of elementary type is also called a *first-order variable*.

Definition 2.2. *For every type τ we define the set $Term^\tau$ of terms of type τ according to the grammar*

$$Term^\tau ::= f^\tau \mid x^\tau \mid (Term^{\tau'\to\tau}\ Term^{\tau'}) \mid \lambda x^{\tau_1}.Term^{\tau_2}$$

where f is a function symbol of type τ, and x is a variable of type τ. The term $\lambda x^{\tau_1}.Term^{\tau_2}$ is only valid if $\tau_1 \to \tau_2 = \tau$.

If $t \in Term^\tau$, we say t *has type* τ and denote this by $type(t) = \tau$. Terms of the form $(t_1\ t_2)$ are called *applications*, and terms of the form $\lambda x.t$ are called *abstractions*. The *target type* of a term t is the target type of $type(t)$.

The notions of bound and free variables in a term and open and closed (or ground) terms are as usual. The set of free variables in a term t is denoted as $FV(t)$. We say that s, t are α-equal ($=_\alpha$), if s and t differ only by a sequence of renamings of bound variables of equal types. To avoid too clumsy notation and to avoid distraction from the essential, we assume the *disjoint variable convention*: all bound variables in terms are distinct, and whenever an operation makes it necessary to rename bound variables, this is performed.

To avoid excessive bracketing, we write applications in flat form: $(t_1\ t_2\ \ldots t_n)$ means the term $(\ldots((t_1\ t_2)\ t_3)\ldots t_n)$. If a term is of the form $(f\ t_1 \ldots t_n)$, where $n = ar(f)$, then we may write this term as $f(t_1, \ldots, t_n)$. We also write nested lambda-expressions in a shorter form. $\lambda x_1, x_2 \ldots, x_n.t$ means $\lambda x_1.(\lambda x_2.\ldots(\lambda x_n.t)\ldots)$. Expressions $\lambda\vec{x}.t$ are shorthand for $\lambda x_1, x_2 \ldots, x_n.t$. We will use positions in terms, which are tree addresses corresponding to occurrences of subterms.

A *maximal application* in a term is a subterm of the form $(t_1\ t_2\ \ldots t_n)$ $(n \geq 1)$ that is not an abstraction and not the left subterm of an application.

The *head* of an application is the subterm that is in the leftmost position in the flat representation. For example, f is the head of $(f\ t_1\ \ldots\ t_n)$.

For a set of types T, let $subt(T)$ be the smallest superset of T with the condition: If $\alpha \to \beta \in subt(T)$, then $\alpha, \beta \in subt(T)$. For a term t, let $types(t)$ be the set of all the types of subterms, and let $subt(t)$ be $subt(types(t))$.

2.2 First-order Terms and First-Order Contexts

For each type τ let $[\cdot]^\tau$ denote a new function symbol of type τ that does not belong to Σ. $[\cdot]^\tau$ is called the *hole* of type τ. *Contexts* are terms built over the enlarged signature that have exactly one occurrence of a hole. The expression $C[t]^\tau$ for a context C and a term/context t of type τ denotes the term/context that is constructed from C by replacing the hole $[\cdot]^\tau$ with t. Since contexts are used as a kind of meta syntax to describe a term, it is justified that variable capture is permitted for contexts. A context is *trivial* iff it has the form $[\cdot]^\tau$. A context B is a *prefix of a context* C iff there is a context B' such that $C = B[B']$. A context B is a *suffix of a context* C iff there is a context B' such that $C = B'[B]$. A context B is a *subcontext of a context* C iff there are contexts B', B'' such that $C = B'[B[B'']]$. A context B is a *subcontext of a term* t iff there is a context B' and a term t' such that $t = B'[B[t']]$.

Definition 2.3. *A* first-order function symbol *is either an elementary constant, or a function symbol of type* $\iota_1 \to \iota_2 \to \ldots \to \iota_m \to \iota$. *A* first-order term *is a term generated by the grammar:*

$$FOT ::= x^\iota \mid f(FOT_1, \ldots, FOT_n)$$

where f *denotes a first-order function symbol of arity* $n \geq 0$. *A* first-order context *is defined using the grammar:*

$$FOC ::= [\cdot]^\iota \mid f(t_1, \ldots, t_{i-1}, FOC, t_{i+1}, \ldots, t_n)$$

where f *is a first-order function symbol of arity* $n \geq 1$ *and the* t_i *are first-order terms.*

For first-order terms/contexts we will use positions as for conventional terms of first-order logic and call them *first-order positions*. E.g., i is the first-order position of t_i in $f(t_1, \ldots, t_n)$. The length of the first-order position of the hole of a first-order context C is called *main depth* of C, denoted as $|C|$. The first digit of the position of the hole of a nontrivial first-order context C is denoted as $\mathit{firstdpos}(C)$. If n is a nonnegative integer and C is a first-order context of type ι with hole of type ι, then $C^0 := [\cdot]^\iota$, and $C^{n+1} := C[C^n]$. Note that C^n is of type ι.

2.3 Measures

For the following proofs, several measures are needed.

Definition 2.4.

- *The* order $ord(\tau)$ *of a type* τ *is defined as follows:*
  - $ord(\iota) = 1$.
  - *If* $\tau = \alpha_1 \to \ldots \alpha_n \to \iota$, *then* $ord(\tau) = 1 + max\{ord(\alpha_1), \ldots, ord(\alpha_n)\}$.

  The degree *of a term* t *is* $deg(t) := max\{(ord(\tau) - 1) \mid \tau \in subt(t)\}$. [1]
- *The* size *of type* τ *is defined as follows:*
  - $size(\iota) = 1$
  - *If* $\tau = \alpha_1 \to \ldots \alpha_n \to \iota$, *then* $size(\tau) = 1 + n + \sum_{i=1}^{n} size(\alpha_i)$.
- *The* size *of a term* t *is defined as follows:* $size(x) = 1, size(f) = 1, size(\lambda x.t) = size(t) + 2$, *and* $size(s\ t) = 1 + size(s) + size(t)$.
- *The* length $len(t)$ *of a term* t *is defined as follows ([Bec01]):* $len(x) = 1, len(f) = 1, len(\lambda x.t) = len(t) + 1$, *and* $len(s\ t) = len(s) + len(t)$. *Note that* $len(t) \leq size(t)$.

The last measure is used to formally define bounded higher-order unification problems — it plays a central role:

Definition 2.5. *For a term* t, *we define* $\#bvl(t)$ *to be the number of occurrences of bound variables in* t *plus the number of lambda-binders in* t. *If* $\overrightarrow{y}$ *is a finite sequence of variables, then* $\#bvl_{\overrightarrow{y}}(t)$ *is defined as* $\#bvl(t)$, *plus the number of free occurrences of variables from* $\overrightarrow{y}$ *in* t.

For example, $\#bvl(\lambda x.f(\lambda y.(x\ y\ z))) = 4$, and $\#bvl_y(\lambda x_1.\lambda x_2.\lambda x_3.y) = 4$.

[1] In [Bec01], the degree of a term is defined similarly as the order in papers on unification, however, $degree = order - 1$.

2.4 Instantiation and Substitutions

An *instantiation* of a variable x^τ in s by the term t^τ, written $s[t/x]$, replaces free occurrences of the variable x in s by t, where before replacement, the bound variables in s have to be renamed to avoid variable capture. See, e.g., [HS86] for a precise definition. After the replacement, it may be necessary to rename bound variables in the different copies of t, since we use the disjoint variable convention. The notation $s[t/x]$ is only used if t and x have the same type.

A *closed substitution* is a mapping from terms to closed terms. In the sequel, with a substitution we always mean a closed substitution. A substitution σ can be represented as $\{x_i \to t_i \mid i = 1, \ldots, n\}$, where t_i for $i = 1, \ldots, n$ is a closed term with $type(x_i) = type(t_i)$ for $i = 1, \ldots, n$. To apply the substitution σ to the term s means to simultaneously instantiate each variable x_i by t_i $(1 \leq i \leq n)$. The *domain* of σ is the set $\{x_i \mid i = 1, \ldots, n\}$, and the *codomain* of σ (denoted $cod(\sigma)$) is the set $\{t_i \mid i = 1, \ldots, n\}$. If all function symbols occurring as subterms in $cod(\sigma)$ are in the set $\Sigma_0 \subseteq \Sigma$, then σ is called a Σ_0*-substitution*.

We tacitly assume that a substitution σ is only applied to terms t where $FV(t)$ is a subset of the domain of σ, hence we assume that the result of applying a closed substitution to a term results in a closed term.

2.5 Reduction and Equality

Since we use the $\beta\eta$ rules for the simply typed lambda-calculus, there are the following equations between terms:

(α)	$\lambda x.t$	$= \lambda y.t[y/x]$	y is a fresh variable
(β)	$((\lambda x.t)\ s)$	$= t[s/x]$	
(η)	t	$= \lambda x^\tau.(t\ x)$	if $type(t) = \tau \to \tau_1$ and $x \notin FV(t)$.

Of course we also assume that the thus defined equality $=_{\beta\eta}$ is an equivalence relation and a congruence, i.e. $s =_{\beta\eta} t \Rightarrow C[s] =_{\beta\eta} C[t]$. Note that $s =_\alpha t \Rightarrow s =_{\beta\eta} t$.

Usually, the equations for $(\beta), (\eta)$ are directed. We will employ η-expansion, denoted as $\overline{\eta}$.

(β)	$C[(\lambda x.t)\ s] \to C[t[s/x]]$	for all contexts C.
(η)	$C[\lambda y.(t\ y)] \to C[t]$	If $y \notin FV(t)$. The rule is applicable for all contexts $C[]$.
$(\overline{\eta})$	$C[t] \to C[\lambda y.(t\ y)]$	if t is not an abstraction, $type(t)$ is not an elementary type, and t in $C[t]$ is a maximal application. The variable y must be a fresh variable of appropriate type. This reduction is valid for all contexts C.

Since there are wrong definitions in the literature, we give some examples to clarify what we mean by $(\overline{\eta})$-reduction.

Example 2.6. The term $x^{\iota\to\iota}$ can be $(\overline{\eta})$-reduced (η-expanded) to $\lambda y^{\iota}.(x^{\iota\to\iota}\ y)$. The term $\lambda x_1^{(\iota\to\iota)\to\iota\to\iota}, x_2^{\iota\to\iota}.(x_1\ \ x_2)$ will be $(\overline{\eta})$-reduced in two steps to $\lambda x_1, x_2, y_1^{\iota}.(x_1\ (\lambda y_2^{\iota}.(x_2\ y_2))\ y_1)$.

If a term cannot be further reduced by $\beta\overline{\eta}$ (resp. $\overline{\eta}$), then it is in $\beta\overline{\eta}$-normal form (resp. $\overline{\eta}$-normal form). It is well-known that the reduction relation defined by $\beta, \overline{\eta}$ is strongly terminating and Church-Rosser [Wol93,Hue76,Bar84]. Hence for every term t, there is a $\beta\overline{\eta}$-normal form $t{\downarrow_{\beta\overline{\eta}}}$, which is unique up to $=_\alpha$.

Remark 2.7. Let t be a term in $\overline{\eta}$-normal form. Let t' result from t by a series of β-reductions. Then t' is in $\overline{\eta}$-normal form.

Remark 2.8. Let t be a term of type τ in $\beta\overline{\eta}$-normal form. Let $m = ar(\tau)$. Then t has the form $\lambda y_1, \ldots, y_m.t'$. In particular $\#bvl(t) \geq m$. The head of any maximal application in t' is either a variable or a function symbol $f \in \Sigma$. Hence maximal applications can be written in the form $x(t_1, \ldots, t_n)$ or $f(t_1, \ldots, t_n)$. This leads to a tree representation of terms in $\beta\overline{\eta}$-normal form that closely resembles the usual tree representation of terms in first-order logic. Cf. Figure 2 below.

Proposition 2.9. *The following equivalence holds:*

$$s =_{\beta\eta} t \Leftrightarrow s{\downarrow_{\beta\overline{\eta}}} =_\alpha t{\downarrow_{\beta\overline{\eta}}} \Leftrightarrow s{\downarrow_{\beta\eta}} =_\alpha t{\downarrow_{\beta\eta}}$$

Lemma 2.10. *A term t is in $\beta\overline{\eta}$-normal form, iff the following holds:*

- *it is in β-normal form, and*
- *every proper subterm s of t such that s is not an abstraction and s has a non-elementary type is embedded in a superterm of the form $(s\ s')$.*

Example 2.11. The $\beta\overline{\eta}$-normal form of the term $\lambda x^{\iota\to\iota}.x$ is the term $\lambda x^{\iota\to\iota}.(\lambda y^{\iota}.(x\ y))$. The $\beta\overline{\eta}$-normal form of the function symbol $f^{\iota\to\iota\to\iota}$ is the term $\lambda x^{\iota}, y^{\iota}.(f\ x\ y)$.

Proposition 2.12. *Let s, t, be terms of equal type, and f be a function symbol. Then*

- $f\ s =_{\beta\eta} f\ t \Leftrightarrow s =_{\beta\eta} t$
- *If $type(s) = type(t) = \alpha_1 \to \ldots$ is not elementary, and f a fresh function symbol of type α_1, then $s =_{\beta\eta} t \Leftrightarrow s\ f =_{\beta\eta} t\ f$.*

Proof. The first claim follows from reduction to $\beta\eta$-normal form. In the second claim, the direction $s =_{\beta\eta} t \Rightarrow s\ f =_{\beta\eta} t\ f$ is trivial, since it follows from the congruence property. To prove the other direction, let $s\ f =_{\beta\eta} t\ f$, where f is a fresh function symbol. Since reduction makes no difference between function symbols and free variables, this implies that $s\ x =_{\beta\eta} t\ x$, where x is fresh variable. From congruence it follows that $\lambda x.(s\ x) =_{\beta\eta} \lambda x.(t\ x)$. Then we can use (η) on both sides of the equation and obtain $s =_{\beta\eta} t$. □

Lemma 2.13. *Let x be a variable of type τ. Then the $\beta\overline{\eta}$-normal form of x has size at most $3 * size(\tau)$.*

Proof. We use induction on the size of τ. If x has type ι, then we are ready. If x has type $\tau = \alpha_1 \to \ldots \to \alpha_n \to \iota$, then $(\overline{\eta})$-reductions transform x into $\lambda x_1^{\alpha_1}, \ldots, x_n^{\alpha_n}.(x\ x_1 \ldots\ x_n)$. The size is $4n + 1$. The final $\beta\overline{\eta}$-normal form is

$$\lambda x_1^{\alpha_1}, \ldots, x_n^{\alpha_n}.(x\ x_1{\downarrow_{\beta\overline{\eta}}} \ldots\ x_n{\downarrow_{\beta\overline{\eta}}}).$$

This gives a size of $3n + 1 + \sum_i (size(x_i{\downarrow_{\beta\overline{\eta}}}))$. By induction this is smaller than $3 \cdot (n + 1 + \sum(size(\alpha_i))) = 3 \cdot size(\tau)$. □

Lemma 2.14. *If a ground term t is in $\beta\overline{\eta}$-normal form, and $\#bvl(t) = 0$, then t is ground first-order term and of elementary type.*

2.6 Upper Bounds on Sizes of Normal Forms

There are estimations on the length of reduction sequences for various lambda-calculi (see [Bec01,Gan80,Sch82,Sch91]). We adapt this to our purposes and argue that there is a computable upper bound on the size of a $\beta\overline{\eta}$-normal form of a term t. Note that there are also lower bounds for the complexity [Sta79,Bec01].

We need an estimation on the size of normal forms depending on the starting term. Let $2_0(n) := n$ and $2_m(n) = 2^{2_{m-1}(n)}$ for $m > 0$. Let $maxtypesize(t)$ be the maximal size of the types of subterms of t, i.e. $maxtypesize(t) := max\{size(\tau) \mid \tau \in types(t)\}$.

Lemma 2.15. *Let t be a term. Then the size of the $\overline{\eta}$-normal form of t is at most $seqnf(t) := 3 \cdot size(t) \cdot maxtypesize(t)$.*

Proof. Lemma 2.13 shows the claim for variables. For each subterm of t that is a non-maximal application in the sense that some arguments are not made explicit, we may add $\overline{\eta}$-normal forms of appropriate variables as arguments and corresponding lambda-binders, as in the proof of Lemma 2.13. The enlargement of the size that results from treating one subterm in this way is bound by $3 \cdot maxtypesize(t) - 1$. Since the total number of subterms that represent non-maximal applications in the above sense does not exceed $size(t)$, the size of the final term is bound by $size(t) \cdot (3 \cdot maxtypesize(t) - 1) + size(t) = seqnf(t)$. □

Theorem 2.16. *Let t be a term. Then the size of the $\beta\overline{\eta}$-normal form of t is not greater than $sbeqnf(t) := seqnf(t)^{2^{deg(t)+1}(seqnf(t))}$.*

Proof. We first transform t into its $\overline{\eta}$-normal form t'. Lemma 2.15 shows that the size of t' is bound by $seqnf(t)$. It is simple to see that $deg(t) = deg(t')$. Remark 2.7 shows that only β-reductions are needed to reach the $\beta\overline{\eta}$-normal form t'' of t. Using the result in ([Bec01]), who shows that the number of reductions of a term r is at most $2_{deg(r)}(len(r))$, we obtain an upper bound $2_{deg(t)}(seqnf(t))$ for the number of β-reductions that are needed to reach t''. Since every β-reduction step may increase the size of a term at most by squaring it it follows that $size(t'') \leq sbeqnf(t)$. □

3 Bounded Higher-Order Unification Problems

In this section we formally introduce the decision problems that are studied in this paper. In the sequel, let $\Sigma_0 \subseteq \Sigma$ denote a subsignature.

Definition 3.1. *A* higher-order unification problem (HOUP) *is a finite set S of (symmetric) equations $\{s_1 \doteq t_1, \ldots, s_n \doteq t_n\}$ where s_i, t_i are terms with $type(s_i) = type(t_i)$ for all i. A closed (Σ_0-) substitution σ such that $\sigma(s_i) =_{\beta\eta} \sigma(t_i)$ for $1 \leq i \leq n$ is called a (Σ_0-)* unifier *of S.*

The symmetry of equations is mainly used for avoiding clumsy statements such as: "if $s \doteq t$ or $t \doteq s$ is in S". Thus if $S = \{a \doteq b, c \doteq d\}$, then by symmetry, it is correct to say that $b \doteq a$ is in S.

Definition 3.2. *Let S be a HOUP, let $b : FV(S) \to \mathbb{N}_0$ be a function. Then the pair (S, b) is called a* bounded HOUP (BHOUP). *A (Σ_0-) substitution σ is a (Σ_0-)* unifier *of (S, b) iff all terms in the codomain of σ are in $\beta\overline{\eta}$-normal form, σ is a (Σ_0-) unifier of S and for every variable $x \in FV(S)$ the inequation $\#bvl(\sigma(x)) \leq b(x)$ holds.*

Note that in a BHOUP the size of unifiers is *not* bounded, since for example for $t = \lambda x.\underbrace{f(\ldots(f}_{k}\ x)\ldots)$ we have $\#bvl(t) = 2$, but the size of t grows with k.
We remark that the upper bound b also provides an (implicit) upper bound on the size of types in $subt(t)$ for terms in $cod(\sigma)$ for unifiers σ.

Lemma 3.3. *Let (S, b) be BHOUP with unifier σ. Then for every variable x with $b(x) = 0$, the variable x has elementary type and $\sigma(x)$ is a ground first-order term.*

Proof. By assumption, the terms in the codomain of σ are ground and in $\beta\overline{\eta}$-normal form. The result follows from Lemma 2.14. □

4 Properties of Minimal Unifiers

A *minimal* unifier σ of a BHOUP (S, b) is a unifier such that the sum $\sum_{x \in FV(S)} size(\sigma(x))$ is minimal with respect to all unifiers of the problem. In this section we describe two properties of minimal unifiers of BHOUPs.

4.1 Sufficient Signatures

First it is shown that for any given BHOUP a finite signature can be described that suffices to unify the BHOUP if there is any unifier. This result will be important when formulating non-deterministic transformation rules where we "guess" function symbols occurring in unifiers. It will lead to a finite branching of the search tree.

Lemma 4.1. *Let σ be a minimal unifier of the BHOUP (S, b). Then the following holds:*

1. *Every function symbol that is not an elementary constant occurring as a subterm in the codomain of σ also occurs in S.*
2. *All types of elementary constants, variables and applications occurring as subterms in the codomain of σ are in $subt(S)$.*
3. *The maximal arity of types of variables occurring as subterms in $cod(\sigma)$ is not greater than the maximal arity of types in $subt(S)$.*
4. *If $\Sigma_0 \subseteq \Sigma$ is any subsignature that contains all function symbols occurring in S and in addition at least one elementary constant a^ι for each (elementary) target type ι in $subt(S)$, then there exists a minimal unifier σ' of (S, b) that is a Σ_0-unifier.*

Proof. 1. Let (S, b) be a BHOUP and σ be a minimal unifier of (S, b). Assume there exists a non-elementary function symbol f in the codomain of σ that does not occur in S. Since the terms in the codomain of σ are in $\beta\overline{\eta}$-normal form each occurrence of f is the head of a term $f(t_1, \ldots, t_{ar(f)})$. Let ι be the target type of f, let a^ι be an elementary constant. Then replace every occurrence of a term $f(t_1, \ldots, t_{ar(f)}$ in the codomain of σ by a. The constructed substitution σ' remains in $\beta\overline{\eta}$-normal form. If $s \doteq t$ is an equation of S, then $\sigma'(s)\downarrow_{\beta\overline{\eta}}$ is obtained from $\sigma(s)\downarrow_{\beta\overline{\eta}}$ by replacing each subterm $f(t_1, \ldots, t_{ar(f)})$ by a, and similarly for $\sigma'(t)\downarrow_{\beta\overline{\eta}}$ and $\sigma(t)\downarrow_{\beta\overline{\eta}}$. This follows from the fact that f does not occur in s, t. Since $\sigma(s)\downarrow_{\beta\overline{\eta}} =_\alpha \sigma(t)\downarrow_{\beta\overline{\eta}}$ also $\sigma'(s)\downarrow_{\beta\overline{\eta}} =_\alpha \sigma'(t)\downarrow_{\beta\overline{\eta}}$. This shows that σ is a unifier for (S, b). Since σ' has a smaller size, this is a contradiction.
2. First of all, the top terms in the codomain have a type in $subt(S)$.
We use induction on the structure of the terms in the codomain. Suppose a subterm $\lambda x_1, \ldots, x_m.f(t_1, \ldots, t_n)$ of a term in the codomain has a type in $subt(S)$. Part 1 shows that for $n > 0$ we may assume that the type of f, hence the type of any t_i is in $subt(S)$ $(1 \leq i \leq n)$. Suppose a subterm $\lambda x_1, \ldots, x_m.x_i(t_1, \ldots, t_n)$ has a type in $subt(S)$ where $1 \leq i \leq n$. Then the type of x_i, hence the type of any argument t_i is in $subt(S)$ $(1 \leq i \leq n)$. The result follows by induction.
3. Follows from the previous part.
4. Let a^ι be an elementary constant occurring in the codomain of σ that does not occur in S. Part 2 shows that ι is a target type in $subt(S)$. Hence, as in Part 1 of the proof a^ι can be replaced by an elementary constant $b^\iota \in \Sigma_0$. □

4.2 The Exponent of Periodicity

The *exponent of periodicity* (see also [SSS98]) of a unifier σ of (S, b) is the maximal number n such that for some variable x occurring in S the image $\sigma(x)$ contains a subterm of the form $C^n[t]$, where C is a nontrivial ground first-order context.

Definition 4.2. *Let t be a ground term in $\beta\overline{\eta}$-normal form. Assume that we color in t the positions of*

1. *each of the n lambda-binders in expressions $\lambda x_1, \ldots, x_n$ occurring in t,*
2. *each occurrence of a bound variable in t,*
3. *each occurrence of a function symbol f in an expression $f(t_1, \ldots, t_n)$ where either*
   (a) *f contains an argument of non-elementary type, i.e. f is not first-order, or*
   (b) *there are at least two subterms t_{i_1}, t_{i_2} $(i_1 \neq i_2)$ such that t_{i_j} for $j = 1, 2$ contains an occurrence of a variable or a lambda.*

The uncolored positions of t can be considered as the nodes of a graph where links correspond to immediate subterm relationship. Each maximal connected uncolored component either defines a ground first-order term or a ground first-order context. They are called the maximal first-order subterms/subcontexts *of t. The* representation size *of t, repsize(t) is defined similarly as the size of t, but each maximal first-order subterm/subcontext yields a uniform contribution of 1.*

Intuitively, in the *repsize*-measure, maximal first-order subterms/subcontexts are treated as primitive symbols.

Example 4.3. The ground term t depicted in Figure 2 is colored in the above sense. There are two occurrences of the maximal first-order subterm $f(a, a, a)$, one occurrence of the maximal first-order subterm a, one occurrence of the maximal first-order context $f(a, f(a, [\cdot], c), c)$, and one occurrence of the maximal first-order context $f(a, [\cdot], c)$. Hence the maximal first-order subterms/subcontexts yield a contribution of 5 to *repsize*(t).

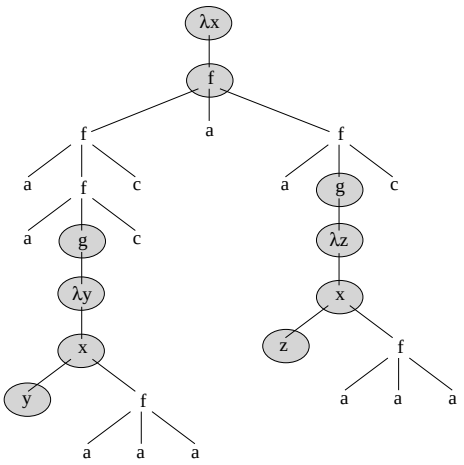


Fig. 2. Colored positions and maximal first-order subterms and subcontexts.

Lemma 4.4. *Let t be a ground term in $\beta\overline{\eta}$-normal form, colored as above. Let $k := \#bvl(t)$. Then there are at most $3k$ colored positions. The number of maximal first-order (uncolored) subcontexts of t does not exceed $3k$. The number of maximal first-order (uncolored) subterms of t does not exceed $1 + 3k \cdot (1 + maxar(t))$ where $maxar(t)$ is the maximal arity of a type in $subt(t)$. The representation size $repsize(t)$ does not exceed $2 + 22k + 6k \cdot maxar(t)$.*

Proof. In t we have at most k colored positions corresponding to lambda-binders or occurrences of bound variables. If f has an argument of non-elementary type, then this argument starts with a lambda-binder. Hence there are at most k colored positions of type 3 (a). In addition there are at most k colored positions of type 3 (b). This gives a total of at most $3k$ colored positions. For each colored position, there is at most one maximal first-order subcontext of t that ends at the position. Hence the number of maximal first-order subcontexts of t does not exceed $3k$. The number of maximal first order subterms of t is at most 1 (for the root) plus the sum of the number of immediate subterms of colored positions. It can be estimated by $1+3k\cdot(1+maxar(t))$. As to $repsize(t)$, the total contribution of maximal first-order subterms/subcontexts is $3k + 1 + 3k \cdot (1 + maxar(t)) = 3k\cdot maxar(t)+6k+1$. The maximal contribution of λ-binders and bound variables is $\leq 2k$ (recall that a binder λx yields a size contribution of 2). The function symbols at colored positions can contribute $3k$. Ignoring applications, this yields a total bound of $3k\cdot maxar(t)+11k+1$. Since each application yields an additional contribution of 1, a bound for $repsize(t)$ is $6k \cdot maxar(t) + 22k + 2$. □

Definition 4.5. *Let (S, b) be a BHOUP. Let $maxar(S)$ denote the maximal arity of a type in $subt(S)$, let $maxb$ be the maximal value $b(x)$ for variables in $FV(S)$. Then the number*

$$repn(S, b) := 6 \cdot maxb \cdot maxar(S) + 22 \cdot maxb + 2$$

is called the representation number *of (S, b).*

Lemma 4.6. *Let (S, b) be a BHOUP, and σ be a minimal unifier of (S, b). Then the representation size of any term in the codomain of σ is at most $repn(S, b)$.*

Proof. This follows from lemma 4.4 and lemma 4.1. □

The important point to note is that the above estimate for the representation size does not depend on σ. In the sequel we use some of the previously introduced measuring functions also for HOUPs S as follows. If $S = \{s_1 \doteq t_1, \ldots, s_n \doteq t_n\}$, then $terms(S)$ is the multiset of all terms s_i and t_i $(i = 1, \ldots, n)$. Now we can use the functions $ord, deg, size, maxtypesize, subt, seqnf, sbeqnf$ also for S by applying them to $terms(S)$, and use the obvious operators for extending the functions to multisets.

Lemma 4.7. *There is a positive real constant c_0 such that for every unifiable BHOUP (S, b) the exponent of periodicity of a minimal unifier of (S, b) is less than $2^{(c_0+2,14\cdot finsize(S))}$, where*

$$finsize(S) := 2_{deg(S)+1}(repn(S, b) \cdot sbeqnf(S)).$$

Proof. Let (S, b) be a BHOUP and let σ be a minimal unifier of (S, b). Let $\mathit{terms}(\sigma(S))$ denote the multiset of image terms $\{\sigma(s) \mid s \in \mathit{terms}(S)\}$. In each term $\sigma(s)$ we consider the occurrences of codomain terms $\sigma(x)$ that represent the images of the variables occurring in s under σ. For each such occurrence we consider the maximal first-order subterms/subcontexts of the respective codomain term as primitive symbols. Each such subterm/subcontext will be called an *inner codomain subterm/subcontext*, stressing their origin in codomain terms. By Lemma 4.6, the sum of the sizes of all terms in $\mathit{terms}(\sigma(S))$ with respect to this representation is bound by $\mathit{size}(S) \cdot \mathit{repn}(S, b)$.

When we compute the $\beta\overline{\eta}$-normal form of the terms in $\mathit{terms}(\sigma(S))$, the inner codomain subterms/subcontexts are not destroyed. For the reduction they can be considered as primitive symbols as well. Hence it follows from Theorem 2.16 that the corresponding representation for the normalized image terms in the set $\{\sigma(s)\downarrow_{\beta\overline{\eta}} \mid s \in \mathit{terms}(S)\}$ has representation size not exceeding $\mathit{finsize}(S)$ as defined in the lemma.

Now we use the fact that the $\beta\overline{\eta}$-normal forms of the left- and right-hand side of equations are α-equal to extract a context unification problem [SS99b,SSS00]. This can be done by equating the following:

- the maximal ground first-order terms in equations $\sigma(s)\downarrow_{\beta\overline{\eta}} =_\alpha \sigma(t)\downarrow_{\beta\overline{\eta}}$ at corresponding positions,
- the maximal ground first-order contexts in equations $\sigma(s)\downarrow_{\beta\overline{\eta}} =_\alpha \sigma(t)\downarrow_{\beta\overline{\eta}}$ at corresponding positions,

for $s \doteq t \in S$. Note that all the inner codomain subterms/subcontexts are contained in some maximal first-order term/context. The context unification problem CUP is formed from the equations $\sigma(s)\downarrow_{\beta\overline{\eta}} =_\alpha \sigma(t)\downarrow_{\beta\overline{\eta}}$ $(s \doteq t \in S)$ as follows: The inner codomain contexts are replaced consistently by context variables, and the inner codomain terms are consistently replaced by first-order variables.

The total number of occurrences of variables and function symbols in CUP does not exceed $\mathit{finsize}(S)$. The results in [SSS98] show that there exists a fixed real constant c_0 such that the exponent of periodicity of a minimal unifier for CUP is smaller than $2^{c_0+2.14 \cdot finsize(S)}$.

We consider the unifier σ' of CUP that assigns to each context variable the corresponding inner codomain context, and to each first-order variable the corresponding inner codomain term. We show that σ' is a minimal unifier for CUP. It then follows that the exponent of periodicity of σ', hence the exponent of periodicity of σ, does not exceed $2^{c_0+2.14 \cdot finsize(S)}$.

Assume that σ' is not a minimal unifier for CUP. In [SSS98], in order to turn a non-minimal unifier for CUPs into a minimal one, subcontexts of the form CC^mC occurring in the images of variables are replaced by similar subcontexts of the form CC^nC. An appropriate selection of the numbers n guarantees that the new values define a unifier for the CUP that satisfies the above bound. Since for a context C of type ι always C^k has type ι for $k \geq 0$ this shows that the same techniques for modifying non-minimal unifiers can be applied in situation of bounded higher-order unification, where types have to be respected. This

shows that a smaller unifier of CUP could be retranslated into a smaller unifier for (S, b) and yields a contradiction. □

5 The Decision Algorithm

In this section we first introduce the concepts that yield the background for the transformation rules of the decision algorithm for bounded higher-order unification. The structure of the algorithm is explained in the last subsection.

5.1 Surface Positions and Cycles

In order to describe the main reduction techniques, the following notions play a central role.

Definition 5.1. *Let t be a term. The* surface positions *of t are defined as follows:*

- *ε, if t is elementary*[2]*.*
- *if t is elementary, and $t = f(t_1, \ldots, t_n)$, then for every surface position p of t_i: $i.p$ is a surface position of t.*
  In this case we also say f is on the surface of t.
- *if t is elementary, and $t = x\ t_1\ \ldots\ t_{ar(x)}$, then 0, the position of x, is a surface position of t.*

The depth *of a surface position p is the length of p.*

We use the notation $t\lfloor s \rfloor$ to indicate that t has a surface occurrence of the term s.

Example 5.2. Let f be of type $(\iota \to (\iota \to \iota) \to \iota)$ and g be of type $\iota \to \iota$. Then the surface positions of $f(x^\iota, y^{\iota \to \iota})$ are $\{\varepsilon, 1\}$, the term $f\ x^\iota$ has no surface positions, the term $f(x^\iota, g)$ has $\{\varepsilon, 1\}$ as surface positions, $f(g(x^\iota), g)$ has $\{\varepsilon, 1, 1.1\}$, and $f(y^{\iota \to \iota}\ x^\iota, g)$ has $\{\varepsilon, 1, 1.0\}$ as surface positions.

Remark 5.3. Assume that the variable x occurring on the surface of t is replaced by a term s of type $type(x)$ where the variable y occurs on the surface of s. Then y occurs on the surface of $t[s/x]$.

Note that every position in a first-order term is a surface position, and that every surface position is elementary or the position of a variable representing the head of an elementary term. Moreover, in the latter case, each node on the path from the root to the variable is labeled by a function symbol, and the argument positions determined by the direction of this path are of elementary type. In the sequel, to simplify index notation for cycles of equations we use expressions $i\ \mathtt{mod}*\ n$ where

$$i\ \mathtt{mod}*\ n = \begin{cases} i\ \mathtt{mod}\ n & \text{if } i\ \mathtt{mod}\ n \neq 0, \\ n & \text{if } i\ \mathtt{mod}\ n = 0. \end{cases}$$

[2] ε denotes the empty sequence.

The algorithm that is used to analyze BHOUPs does not operate on single equations. Rather it tries to reduce combinations of equations of a particular form, called cycles.

Definition 5.4. *Let S be a BHOUP. A* cycle *is a sequence $s_1 \doteq t_1, \ldots, s_h \doteq t_h$ of length $h \geq 1$ of equations from S, such that for all $1 \leq i \leq h$: $s_i \equiv x_i\ r_{i,1}\ \ldots\ r_{i,m_i}$, and x_i occurs on the surface of $t_{i-1 \bmod^* h}$. Moreover, there should be at least one term t_i of the form $f(t_{i,1}, \ldots, t_{i,n})$ and at least one term s_i of the form $x_i\ r_{i,1}\ \ldots\ r_{i,m_i}$ with $\mathrm{ar}(x_i) \geq 1$.*

A cycle is path-unique *if for every $1 \leq i \leq h$ there is only one occurrence of x_i on the surface of $t_{(i-1) \bmod^* h}$.*

Let L be a cycle in S of the form $s_1 \doteq t_1, \ldots, s_h \doteq t_h$, For each of the terms t_i, $1 \leq i \leq h$, let C_i be the context determined as follows: Let q_i be the smallest subterm of t_i such that all surface occurrences of $x_{(i+1 \bmod^* h)}$ from t_i are also contained in q_i. The relevant context *C_i of equation i is uniquely determined by $t_i = C_i\lfloor q_i \rfloor$.*

The length of a cycle *is the number of equations in it. If for some cycle L, there is no other cycle in S with a smaller length, then we say L is a* minimal-length *cycle.*

A cycle $s_1 \doteq t_1, \ldots, s_h \doteq t_h$ is called compressed, *iff there is no i such that s_i or t_i is a first-order variable.*

Example 5.5. We give some examples for cycles and non-cycles.
The sequence $x \doteq h(y\ s_1), y\ s_2 \doteq x$ is a (non-compressed) cycle of length 2, provided $x, y\ s_1, y\ s_2$ are terms of elementary type. When instantiating x by $h(y\ s_1)$ we receive from the second equation a shorter cycle of the form $y\ s_2' \doteq h(y\ s_1)$ which is compressed. The sequence $x_1\ s_1 \doteq f(x_1\ s_2, x_2(x_1\ s_3))$ is a path-unique and compressed cycle of length 1, provided that $x_1\ s_1, x_1\ s_2$ are elementary. The sequence $x_1\ y_1 \doteq x_2\ x_1\ y_1$ is not a cycle.

5.2 Definitions of Soundness and Completeness

Definition 5.6. *Let $\Sigma_0 \subseteq \Sigma$ be a fixed subsignature, let E be a natural number. A non-deterministic transformation rule $\mathcal{T}$ that transforms a BHOUP (S, b) into another BHOUP (S', b'), offering a finite number of alternatives, is called*

- sound (for subsignature Σ_0), *if whenever (S, b) is transformed by $\mathcal{T}$ into (S', b'), and (S', b') is unifiable using a Σ_0-unifier, then (S, b) is unifiable using a Σ_0-unifier.*
- complete (for bound E and subsignature Σ_0), *iff the following holds: If (S, b) has a Σ_0-unifier σ with exponent of periodicity not greater than E, then $\mathcal{T}$ can transform (S, b) into a BHOUP (S', b') that has a Σ_0-unifier with exponent of periodicity not greater than E.*

In the following sections, the upper bound E and the signature Σ_0 are often fixed. In such a context we may simply talk about "soundness" and "completeness".

5.3 A Well-Founded Measure for Termination

We now introduce the measure that is used to prove termination of the decision algorithm.

Definition 5.7. *The lexicographic measure* $\psi(L) = (\psi_1(L), \psi_2(L), \psi_3(L))$ *of a cycle* L *of a BHOUP* (S, b) *has the following three components:*

$\psi_1 =$ *the length* h *of* L.

$\psi_2 =$ 0, *if* L *is non-path-unique,* 1, *if* L *is path-unique.*

$\psi_3 =$ – *if* L *is non-path-unique, then the minimal main depth of the relevant contexts* C_j *of* L *where* t_j *contains at least two different surface-occurrences of* $x_{(j+1) \bmod^* h}$.

– *if* L *is path-unique, then the number of indices* $1 \leq i \leq h$ *such that* C_i *is not trivial.*

Definition 5.8. *The measure* μ *of a BHOUP* (S, b) *is a lexicographic one with the components* $\mu_1, \mu_2, \mu_3, \mu_4, \mu_5, \mu_6$:

$\mu_1 =$ *the multiset* $\{b(x) \mid x \in FV(S), b(x) > \mathrm{ar}(x)\}$. *This component is ordered by the multiset ordering (see [DM79,BN98]).*

$\mu_2 =$ *the multiset* $\{b(x) - \mathrm{ar}(x) \mid x \in FV(S), b(x) > \mathrm{ar}(x)\}$. *This component is ordered by the multiset ordering.*

$\mu_3 =$ *if there is a cycle in* S, *then* $min\{\psi(L) \mid L$ *is a cycle in* $S\}$; *Otherwise,* ∞.

$\mu_4 =$ *the multiset* $\{size(t) \mid t$ *is a top-level term in* S *that is not a first order variable*$\}$. *This component is ordered by the multiset ordering.*

$\mu_5 =$ *the number of occurrences of function symbols in* S *on surface positions.*

$\mu_6 =$ *the number of first-order variables in* S.

Lemma 5.9. *The measure* μ *for BHOUPs is well-founded.*

5.4 Decomposition Rules

The decision algorithm for BHOU operates on systems of a special kind, called "decomposed" BHOUPs. The decomposition of an intermediate system constitutes the final step of each transformation rule. The decomposition rules are described in Table 1. In every application of decomposition the failure rules have highest priority. For the application of the rules we presuppose that all terms in the BHOUP are in $\beta\overline{\eta}$-normal form.

Note that the selection of the function symbol in rule (extend) is "don't care".

Definition 5.10. *A BHOUP* (S, b) *is* decomposed *if no decomposition rule and no failure rule is applicable.*

Remark 5.11. Let S be a decomposed BHOUP and L be a minimal-length cycle in S of the form $x_1 \doteq t_1, \ldots, x_h \doteq t_h$. Then the relevant context C_i of equation i does not have any surface-occurrence of a variable x_j for $j = 1, \ldots, h$. In fact, by definition, C_i cannot contain a surface occurrence of x_{i+1}. If C_i would have a surface occurrence of a variable $x_j \neq x_{i+1}$, then L cannot be length-minimal.

(decomp) $\dfrac{\{f\ s_1\ \ldots\ s_n \doteq f\ t_1\ \ldots\ t_n\} \cup S}{\{s_1 \doteq t_1, \ldots, s_n \doteq t_n\} \cup S}$

(extend) $\dfrac{\{\lambda u^\tau.s \doteq \lambda u^\tau.t\} \cup S}{\{s[f/u] \doteq (t[f/u])\} \cup S}$ where $f \in \Sigma \setminus \Sigma_0$ is a fresh function symbol of type τ.

(constantify) $\dfrac{S}{S'}$ Where S' is constructed from S as follows: Let x be a variable with $ar(x) \geq 1$ and $b(x) = ar(x)$. Let x' be a fresh first-order variable. Replace all subterms $(x\ t_1\ \ldots\ t_{ar(x)})$ by x'. Let $b'(x') := 0$.

(repvv) $\dfrac{\{x \doteq y\} \cup S}{S'}$ If x, y are first-order variables. S' is constructed from S by replacing all occurrences of x by y. $b'(y)$ is defined as $min\{b(x), b(y)\}$.

(decomp-repvt) $\dfrac{\{x \doteq t\} \cup S}{\{x \doteq t\} \cup S'}$ If x is a first-order variable. S' is constructed from S by replacing all surface occurrences of x by t. Conditions for applications are: There must be a minimal-length cycle L that is not compressed, and $x \doteq t$ must be an equation in the cycle L.

Failure rules:

(clash) $\dfrac{\{f\ s_1\ \ldots\ s_n \doteq g\ t_1\ \ldots\ t_m\} \cup S}{Fail}$ if $f \neq g$

(occurs-check) $\dfrac{S}{Fail}$ if there is a chain of equations $x_1 \doteq t_1\lfloor x_2 \rfloor, \ldots, x_{n-1} \doteq t_{n-1}\lfloor x_n \rfloor, x_n \doteq t_n\lfloor x_1 \rfloor$, such that for all $i = 1, \ldots, n$, x_i is a first-order variable, and x_i occurs on the surface of $t_{(i-1)\ \mathtt{mod*}\ n}$; and for some $i = 1, \ldots, n$, the term t_i is of the form $f\ t_{i,1}\ \ldots\ t_{i,ar(f)}$.

(inconsistent-bound) $\dfrac{S}{Fail}$ if there is a variable x with $b(x) < ar(x)$.

Table 1. The decomposition rules

Lemma 5.12. *The decomposition rules are sound and complete.*

Proof. Soundness of (decomp), (repvv) and (decomp-repvt) is trivial. Let (S', b') result from (S, b) by an application of (extend), and let σ be a Σ_0-unifier for (S', b'). Then the substitution does not use the symbol f. Since σ is ground it does not use u. We have $\sigma(s[f/u]) =_{\beta\eta} \sigma(t[f/u])$. We can replace all symbols f by u, and obtain $\sigma(s) =_{\beta\eta} \sigma(t)$, hence $\sigma(\lambda u.s) =_{\beta\eta} \lambda u.\sigma(s) =_{\beta\eta} \lambda u.\sigma(t) =_{\beta\eta} \sigma(\lambda u.t)$.

Let (S', b') result from (S, b) by an application of (constantify). Let σ' be a Σ_0-unifier for (S', b'). Then $t := \sigma'(x')$ is a first-order ground term, by Lemma 3.3. We define $\sigma(x) := \lambda y_1, \ldots, y_{ar(x)}.t$ and $\sigma(y) := \sigma'(y)$ for all free variables $y \neq x$ in S. Obviously σ is a Σ_0-unifier for (S, b).

For *completeness* first note that if a failure rule applies, then the input system does not have a unifier. For (inconsistent-bound) recall Remark 2.8. Completeness of rules (decomp), (repvv) and (decomp-repvt) is trivial. Let σ be a Σ_0-unifier of the input system.

First assume that (extend) is applied in the form described in Table 1. We have $\sigma(s) =_{\beta\eta} \sigma(t)$, where u is a free variable in $\sigma(s)$ and $\sigma(t)$. Since σ is ground, variable u does not occur in its codomain. It follows that $\sigma(s[f/u]) =_{\beta\eta} \sigma(t[f/u])$. This shows that σ unifies the system reached with (extend). Completeness of (extend) follows.

Assume now that (constantify) is applied. In the situation of the rule, $\sigma(x)$ has the form $\lambda y_1, \ldots, y_{ar(x)}.t$ where $\#bvl^{\rightarrow}_{y}(t) = 0$ (cf. Remark 2.8). By Lemma 2.14, t is a ground first-order term. Hence, for all subterms $x\ t_1 \ldots t_n$ occurring in the input system we have $\sigma(x\ t_1 \ldots t_n) = t$. Let $\sigma'(y) := \sigma(y)$ for all $y \neq x$ occurring in S, let $\sigma'(x') := t$. Then σ' satisfies the bounds b'. Since σ' is a Σ_0-unifier of S' and the exponent of periodicity of σ' does not exceed the exponent of periodicity of σ completeness of (constantify) follows. □

Example 5.13. Soundness of the algorithm is an issue. In particular the rule (extend) enforces a careful usage of signatures. Consider the equation

$$\lambda x.y \doteq \lambda x.fx,$$

where y is a free (first-order) variable, and f is a function symbol. This equation has no unifier, since y cannot be instantiated with a term containing the free variable x, since instantiation is capture-free.

After applying (extend), the new equation is

$$y \doteq fg$$

where g is a function symbol from $\Sigma \setminus \Sigma_0$.

After an imitation instantiation $y \to f\ y'$ and a subsequent decomposition the system is:

$$y' \doteq g$$

This is unifiable as a first-order unification problem, but there is no unifier using symbols from Σ_0.

Hence soundness of decomposition requires restricting imitation instantiations to symbols from Σ_0.

Lemma 5.14. *If in (S, b) all terms are in $\beta\overline{\eta}$-normal form, then each non-failing decomposition rule reduces the measure μ. Thus the repeated application of decomposition rules terminates.*

Proof. Rule (decomp) does not modify μ_1, μ_2. Since equations $f\ s_1 \ldots s_n \doteq f\ t_1 \ldots t_n$ do not occur in cycles, μ_3 can only be decreased. The rule strictly reduces μ_4, μ_5.
Rule (extend) does not modify μ_1, μ_2. It strictly reduces μ_4. It may also decrease μ_3.

Rule (constantify) does not modify μ_1, μ_2. In a term $x\ t_1\ \ldots\ t_{ar(x)}$, variable occurrences in the subterms t_i are not on the surface and do not participate to cycles. This shows that also μ_3 is not modified. The size of some term is strictly reduced, hence μ_4 is strictly decreased.

Rule (repvv) does not modify $\mu_1, \mu_2, \mu_4, \mu_5$. It may reduce μ_3 in different ways: A minimal-length cycle may become shorter after application, a path-unique cycle may become non path-unique. Rule (repvv) also reduces the number of first-order variables, i.e. μ_6.

Rule (decomp-repvt) does not modify μ_1, μ_2. If (decomp-repvt) is applied, then $x \doteq t$ is an equation of a minimal-length cycle L. The term t has a surface occurrence of a variable y which is distinct from x since otherwise (occurs-check) would lead to fail. Looking at the predecessor and successor equations of $x \doteq t$ in L it is obvious that S' has a shorter cycle, cf. Remark 5.3. Hence μ_3 strictly decreases. □

Lemma 5.15. *Let (S', b') result from (S, b) by applying one non-failing decomposition rule. If S contains a cycle L, then also (S', b') contains a cycle L' such that $\psi(L') \leq \psi(L)$.*

Proof. The equations that are manipulated in (decomp) and (extend) do not contribute to cycles. For applications of (constantify) note that the subterms t_i of a term of the form $x\ t_1\ \ldots\ t_{ar(x)}$ do not contain surface occurrences of terms. Hence replacing x by x' cannot discard surface occurrences of variables $y \neq x$. Since (constantify) replaces all surface occurrences of x (together with the arguments) by x' it follows that any existing cycle L is preserved with the same ψ-measure. Applications of (repvv) preserve cycles. A path-unique cycle may be shortened, and a path-unique cycle may be transformed into path that is non path-unique. Both effects cannot increase the measure ψ. The situation where (decomp-repvt) is applied leads to a shorter cycle, as we have seen in the previous proof. □

Lemma 5.16. *Let the BHOUP (S', b') result from the BHOUP in $\beta\overline{\eta}$-normal form (S, b) by applying one non-failing decomposition rule. Then all terms of (S', b') are in $\beta\overline{\eta}$-normal form.*

Proof. It is simple to check that the non-failing rules do not enable any new application of β-reduction or η-expansion. For (decomp-repvt) note that t cannot be an abstraction since x has elementary type. □

The following example shows that a unifiable BHOUP may have surface occurrences of a variable and also an occurrence on a non-surface position. Moreover, it shows that the rule (decomp-repvt) does not necessarily eliminate all occurrences of a variable x.

Example 5.17. The BHOUP $(\{x^\iota \doteq y^{\iota\to\iota}\ x, \ldots\}, b)$ with appropriate b is unifiable. A unifier is: $\{x \to a, y \to \lambda u_2^\iota.a\}$, where a is a constant of type ι. In this BHOUP it is not possible to eliminate all the occurrences of x by a variable replacement using $x \doteq y\ x$.

5.5 Different Types of BHOUPs

Definition 5.18. *A decomposed BHOUP (S, b) is of*

- type "xy" *if S does not have any cycles, and if there is no function symbol f on the surface of S, (also called pre-unified in the literature on higher-order unification)*[3]
- type "nocycle" *if S does not have any cycles, and if there exists a function symbol f on the surface of S,*
- type "amb" *if S contains a cycle and if there is a ψ-minimal cycle that is non-path-unique,*
- type "unique" *if S contains a cycle and if all ψ-minimal cycles are path-unique.*

Lemma 5.19. *Let (S, b) be a decomposed BHOUP (S, b) of type "xy", and let Σ_0' be a signature such that Σ_0' contains only constants and for every elementary type $\iota \in subt(S)$ there is a constant of type ι in Σ_0'. Then (S, b) is unifiable by a Σ_0'-unifier.*

Proof. Let (S, b) be decomposed and of type "xy". Since decomposition rule (extend) replaces equations between abstractions, all equations of S are of the form $x\ t_1 \ldots t_n \doteq y\ s_1 \ldots s_n$ where x and y have the same target type. Instantiate every variable x with a constant function of the form $\lambda y_1, \ldots, y_{ar(x)}.a^\iota$ where ι is the target type of x. This is a unifier since it transforms every equation into an identity between elementary constants. The bound b is also satisfied since the rule (inconsistent-bound) is not applicable, which implies $ar(x) \leq b(x)$ for all variables x. □

Lemma 5.20. *Let (S, B) be a BHOUP in $\beta\overline{\eta}$-normal form. After decomposition, the resulting BHOUP (S', b') has either type "xy", or type "nocycle", or type "amb", or type "unique".*

[3] Since constant symbols count as function symbols, this includes that there is also no constant symbol on the surface of S.

5.6 The Algorithm BHU

The main backbone of our algorithm is the following observation.

Proposition 5.21. *For each of the types "nocycle", "amb", or "unique", a transformation rule can be given that accepts a BHOUP (S, b) of the given type and nondeterministically computes a successor system (S', b') such that the following properties hold:*

1. *The rule leads to a finite branching, i.e., for each input BHOUP there is only a finite set of possible successor systems.*
2. *Each successor BHOUP is decomposed and in $\beta\overline{\eta}$-normal form.*
3. *For every successor BHOUP (S', b') we have $\mu(S', b') < \mu(S, b)$,*
4. *The transformation rules are sound and complete, for any finite signature Σ_0 that contains all function symbols from S and an elementary constant of type ι for every elementary type $\iota \in subt(S)$ and any bound E for the exponent of periodicity,*
5. *For every successor BHOUP (S', b') we have $subt(S') \subseteq subt(S)$.*

The transformation rule for BHOUPs of type "nocycle" is described in Section 7. The above properties for the rule are proved in Lemma 7.3, Lemma 7.5 and Lemma 7.4. The transformation rule for BHOUPs of type "amb" is given in Section 8. The above properties are shown in Lemmas 8.2, 8.3 and 8.4. The transformation rule for BHOUPs of type "unique" is given in Section 8. The above properties are shown in Lemmas 9.9, 9.11, 9.12 and 9.14.

On the basis of the Proposition we have the following algorithm for deciding unifiability of BHOUPs.

Definition 5.22. *(**Algorithm BHU** for finite signature Σ_0)*
The input *is a BHOUP (S_{inp}, b_{inp}), and a finite signature Σ_0 such that Σ_0 contains at least one elementary constant a^ι for each (elementary) target type ι in $subt(S_{inp})$, and every function symbol from S_{inp} is in Σ_0.*
The first step *in the algorithm is to apply β-reduction and $\overline{\eta}$-reduction to $\beta\overline{\eta}$-normal form resulting in (S_0, b_0) where $b_0 = b_{inp}$.*
The second step *is to decompose the BHOUP (S_0, b_0). We fix a number E for (S_0, b_0) according to Lemma 4.7 that is used as a bound for the exponent of periodicity. Then the following steps are performed:*

1. *Iteratively transform the current BHOUP (S, b) using the appropriate transformation rule as described in Sections 7–9 into a successor problem (S^*, b^*), which is again in $\beta\overline{\eta}$-normal form.*
2. *The repetition stops if either a fail occurs or it signals success: a BHOUP of type "xy" is generated.*

The input (S_{inp}, b_{inp}) is recognized as unifiable iff there is an execution possibility of BHU such that success results.

Theorem 5.23. *Unifiability of BHOUPs is decidable.*

Proof. We claim that BHU yields "success" iff the input problem (S_{inp}, b_{inp}) is unifiable. First note that (S_{inp}, b_{inp}) is unifiable iff (S_0, b_0) is unifiable: this follows, since β and $\overline{\eta}$-reduction do not change the equality, using Lemma 5.12.

If (S_0, b_0) is unifiable, then there exists a Σ_0-unifier σ_0 with exponent of periodicity not exceeding E (cf. Part 4 of Lemma 4.1 and Lemma 4.7). Since the transformation rules are complete w.r.t. E and Σ_0 and reduce the well-founded measure μ there exists an execution possibility where after a finite number of steps a BHOUP of type "xy" is reached. Hence BHU yields "success". Every transition sequence terminates and each transformation rule is finitely branching, hence by König's Lemma the search tree is finite and the execution strategy that yields success is effectively found.
Conversely, if BHU yields "success", then a BHOUP (S_{fin}, b_{fin}) of type "xy" has been found. Since we have $subt(S_{fin}) \subseteq subt(S_0) \subseteq subt(S_{inp})$ it follows from Lemma 5.19 that (S_{fin}, b_{fin}) has a Σ_0-unifier. Soundness of the transformation rules shows that (S_0, b_0) has a Σ_0-unifier.□

6 Reduction Rules

Before we discuss the treatment of BHOUPs of specific types we describe some rules that represent possible alternatives in various situations and immediately lead to a reduced μ-measure of the resulting BHOUP. The background signature Σ_0 is assumed to be finite. We start with a remark on the possible values of variables under Σ_0-unifiers.

Remark 6.1. Consider the value $\sigma(x)$ of a variable x of arity m under a Σ_0-unifier σ of a decomposed BHOUP (S, b). As always we assume $\sigma(x)$ to be in $\beta\overline{\eta}$-normal form. As a simple consequence of Lemma 3.3, the following four cases represent an exhaustive subcase analysis.

(1) $\sigma(x)$ has the form $\lambda y_1, \ldots, y_m.y_i(t_1, \ldots, t_k)$ for some $1 \leq i \leq m$. This enforces $b(x) > ar(x)$.
(2) $\sigma(x)$ has the form $\lambda y_1, \ldots, y_m.f(t_1, \ldots, t_k)$ where $f \in \Sigma_0$ and there are two subterms t_i, t_j such that $\#bvl_{\overrightarrow{y}}(t_i) \neq 0 \neq \#bvl_{\overrightarrow{y}}(t_j)$ $(i, j \in \{1, \ldots, k\}, i \neq j)$. Here again $b(x) > ar(x)$.
(3) $\sigma(x)$ has the form $\lambda y_1, \ldots, y_m.f(t_1, \ldots, t_k)$ for some $f \in \Sigma_0$ and there exists a unique subterm t_i $(1 \leq i \leq k)$ of non-elementary type. All subterms t_j for $j \neq i$ are ground first-order terms. Here t_i must have the form $\lambda \overrightarrow{z}.t_i^*$ where $\overrightarrow{z}$ is non-empty. Again we have $b(x) > ar(x)$.
(4) $\sigma(x)$ has the form $\lambda y_1, \ldots, y_m.f(t_1, \ldots, t_k)$ where f is an elementary constant or $f \in \Sigma_0$, all arguments of f have elementary type, and there exists at most one index $i \in \{1, \ldots, k\}$ such that $\#bvl_{\overrightarrow{y}}(t_i) \neq 0$. All arguments t_j for $j \neq i$ are ground first-order terms.

The following three reduction rules respectively refer to situations (1)-(3).

Definition 6.2. *(reduce-bv)*
The input is a decomposed BHOUP (S, b) together with a variable $x \in FV(S)$ with $b(x) > ar(x) = m$.

(a) Select some $1 \leq i \leq m$ and instantiate x by the $\beta\overline{\eta}$-normal form of

$$\lambda y_1, \ldots, y_m.y_i\ (x_1\ y_1 \ldots y_m)\ \ldots\ (x_k\ y_1\ \ldots\ y_m)$$

where x_j for $j = 1, \ldots, k = \mathrm{ar}(y_i)$ are fresh variables of the appropriate type.
(b) Select bounds $b'(x_j)$ for $j = 1, \ldots, k$ such that $m \leq b'(x_j) < b(x)$ for all variables x_j and furthermore $\sum_{j=1}^{k} (b'(x_j) - m) \leq b(x) - m - 1$.
(c) Beta-reduce the terms until a $\beta\overline{\eta}$-normal form is reached.
(d) Decompose the resulting BHOUP.

Definition 6.3. *(reduce-split)*
The input is a decomposed BHOUP (S, b) together with a variable $x \in FV(S)$ with $b(x) > \mathrm{ar}(x) = m$.

(a) select some $f \in \Sigma_0$, and instantiate x by the $\beta\overline{\eta}$-normal form of

$$\lambda y_1, \ldots, y_m.f((x_1\ y_1 \ldots y_m), \ldots, (x_k\ y_1\ \ldots\ y_m))$$

where x_j for $j = 1, \ldots, k$ are fresh variables of the appropriate type.
(b) Select bounds $b'(x_j)$ for $j = 1, \ldots, k$ such that $m \leq b'(x_j) < b(x)$ for all variables x_j and furthermore $\sum_{j=1}^{k} (b'(x_j) - m) \leq b(x) - m$.
(c) Beta-reduce the terms until a $\beta\overline{\eta}$-normal form is reached.
(d) Decompose the resulting BHOUP.

Definition 6.4. *(reduce-binder)*
The input is a decomposed BHOUP (S, b) together with a variable $x \in FV(S)$ with $b(x) > \mathrm{ar}(x) = m$.

(a) select some $f \in \Sigma_0$ with a unique non-elementary argument position, say, position i, and instantiate x by the $\beta\overline{\eta}$-normal form of

$$\lambda y_1, \ldots, y_m.f(z_0, \ldots, z_{i-1}, (x'\ y_1\ \ldots\ y_m), z_{i+1}, \ldots, z_k)$$

where z_j for $j = 1, \ldots, i-1, i+1, \ldots k$ are fresh first-order variables and x' is a fresh variable of the appropriate type.
(b) Define $b'(x') := b(x)$ and $b'(z_j) := 0$ for $j = 1, \ldots, i-1, i+1, \ldots k$.
(c) Beta-reduce the terms until a $\beta\overline{\eta}$-normal form is reached.
(d) Decompose the resulting BHOUP.

For proving soundness of the three rules (reduce-bv), (reduce-split), and (reduce-binder), the following lemma is needed.

Lemma 6.5. *Let s be a term in $\beta\overline{\eta}$-normal form of type $\alpha_1 \to \ldots \to \alpha_m \to \iota$, for $1 \leq i \leq m$ let $y_i^{\alpha_i}$ be a fresh variable. Then $\#bvl_{\overrightarrow{y}}((s\ y_1{\downarrow_{\beta\overline{\eta}}} \ldots y_m{\downarrow_{\beta\overline{\eta}}}){\downarrow_{\beta\overline{\eta}}}) = \#bvl(s) - m$.*

Proof. It is sufficient to show this for $m = 1$, and then use induction. Assume that $s = \lambda u_1.s'$. Then $\#bvl(s) - 1 = \#bvl_{u_1}(s')$. We have $[s\ y_1{\downarrow_{\beta\overline{\eta}}}]{\downarrow_{\beta\overline{\eta}}} = (s'[y_1{\downarrow_{\beta\overline{\eta}}}/u_1]){\downarrow_{\beta\overline{\eta}}}$. Since s' is in $\beta\overline{\eta}$-normal form, each occurrence of u_1 represents the head of a maximal application. The replacements of u_1 by $y_1{\downarrow_{\beta\overline{\eta}}}$ and the subsequent beta-reductions have the effect of exactly replacing the variable u_1 by the variable y_1, i.e., $s'[y_1{\downarrow_{\beta\overline{\eta}}}/u_1]{\downarrow_{\beta\overline{\eta}}} =_\alpha s'[y_1/u_1]$. Hence $\#bvl_{y_1}((s\ y_1{\downarrow_{\beta\overline{\eta}}}){\downarrow_{\beta\overline{\eta}}}) = \#bvl_{y_1}(s'[y_1/u_1]) = \#bvl_{u_1}(s') = \#bvl(s) - 1$. □

Example 6.6. Let $s = x^{(\iota\to\iota)\to\iota}{\downarrow_{\beta\overline{\eta}}} = \lambda u^{\iota\to\iota}.(x\ \lambda z^\iota.(u^{\iota\to\iota}\ z))$. We have $\#bvl(s) = 4$. Let $y^{\iota\to\iota}$ be a fresh variable. Applying s to $y{\downarrow_{\beta\overline{\eta}}} = \lambda w^\iota.(y\ w)$ yields after a first β-reduction $x\ \lambda z^\iota.((\lambda w^\iota.(y\ w))\ z)$. A second β-reduction gives $(s\ y{\downarrow_{\beta\overline{\eta}}}){\downarrow_{\beta\overline{\eta}}} = x\ \lambda z.(y\ z)$. We have $\#bvl_y((s\ y{\downarrow_{\beta\overline{\eta}}}){\downarrow_{\beta\overline{\eta}}}) = 3$.

Lemma 6.7. *(a) The three reduction rules (reduce-bv), (reduce-split), and (reduce-binder) are sound.*
(b) Each reduction rule (reduce-bv), (reduce-split), and (reduce-binder) fails or leads to a decomposed BHOUP (S^*, b^*) such that $\mu(S^*, b^*) < \mu(S, b)$.

Proof. (a) Let σ^* be a Σ_0-unifier for the output system (S^*, b^*). Soundness of decomposition shows that there exists a Σ_0-unifier σ' for the system (S', b') reached before decomposition. Obviously, if t denotes the substitute for x as defined in Step (a) of the respective rule, then $\sigma(x) := \sigma'(t){\downarrow_{\beta\overline{\eta}}}$ and $\sigma(y) := \sigma'(y)$ for all free variables $y \neq x$ of S defines a Σ_0-unifier for S. To prove soundness it remains to show that σ respects bound b.

First assume that rule (reduce-bv) is used. Then x is replaced by the term

$$\lambda\overrightarrow{y}.y_i\ (\lambda\overrightarrow{u_1}.x_1\ \overrightarrow{y{\downarrow_{\beta\overline{\eta}}}}\ \overrightarrow{u_1{\downarrow_{\beta\overline{\eta}}}})\ \ \ldots\ \ (\lambda\overrightarrow{u_k}.x_k\ \overrightarrow{y{\downarrow_{\beta\overline{\eta}}}}\ \overrightarrow{u_k{\downarrow_{\beta\overline{\eta}}}}),$$

the length l_j of the lambda-binders $\lambda\overrightarrow{u_j}$ depending on the arity of the j-th argument type of y_i. Hence

$$\sigma(x) = \lambda\overrightarrow{y}.y_i\ (\lambda\overrightarrow{u_1}.[\sigma'(x_1)\ \overrightarrow{y{\downarrow_{\beta\overline{\eta}}}}\ \overrightarrow{u_1{\downarrow_{\beta\overline{\eta}}}}]{\downarrow_{\beta\overline{\eta}}})\ \ \ldots\ \ (\lambda\overrightarrow{u_k}.[\sigma'(x_k)\ \overrightarrow{y{\downarrow_{\beta\overline{\eta}}}}\ \overrightarrow{u_k{\downarrow_{\beta\overline{\eta}}}}]{\downarrow_{\beta\overline{\eta}}}).$$

Lemma 6.5 shows that $\#bvl_{\overrightarrow{y}\,\overrightarrow{u_j}}([\sigma'(x_j)\ \overrightarrow{y{\downarrow_{\beta\overline{\eta}}}}\ \overrightarrow{u_j{\downarrow_{\beta\overline{\eta}}}}]{\downarrow_{\beta\overline{\eta}}}) = \#bvl(\sigma'(x_j) - m - l_j)$. It follows that $\#bvl(\sigma(x)) = m + 1 + \sum_{j=1}^{k}(\#bvl_{\overrightarrow{y}}(\sigma'(x_j)) - m) \leq m + 1 + \sum_{j=1}^{k}(b'(x_j) - m)$. Now the condition in Part (b) of the rule shows $\#bvl(\sigma(x)) \leq b(x)$. The proof for rule (reduce-split) is analogous. For rule (reduce-binder) note that $\sigma'(z_j)$ is a ground first-order term, for $j = 1, \ldots, i_0 - 1, i_0 + 1, \ldots k$. The rest is as before.

(b) Assume that the final decomposition step in Part (d) does not lead to failure. Rules (reduce-bv) and (reduce-split) reduce μ_1 since x is replaced by a finite number of variables with smaller bound. Rule (reduce-binder) does not modify μ_1. Since $ar(x') > ar(x)$ it follows that μ_2 is decreased. □

Lemma 6.8. *[Weak completeness of reduction] Let (S, b) be a decomposed BHOUP with a Σ_0-unifier σ with exponent of periodicity e. Let $M \neq \emptyset$ be any subset of $FV(S)$. Then either*

1. *it is possible to reach via application of one of the rules (reduce-bv), (reduce-split) or (reduce-binder) to some $x \in M$ a decomposed BHOUP (S^*, b^*) such that $\mu(S^*, b^*) < \mu(S, b)$ and (S^*, b^*) has a Σ_0-unifier σ^* with exponent of periodicity $\leq e$, or*
2. *each value $\sigma(x)$ for $x \in M$ has the form (4) described in Remark 6.1.*

Proof. Case 1. If there is any variable $x \in M$ where $\sigma(x)$ has the form (1) described in Remark 6.1, then we apply (reduce-bv) using x (Alternative 1). For the new variables we define $b'(x_j) := m + \#bvl_{\overrightarrow{y}}(t_j)$ $(1 \leq j \leq k)$. This choice is consistent with the condition in Step (b) of (reduce-bv) since $\sum_{j=1}^{k}(b'(x_j) - m) = \sum_{j=1}^{k} \#bvl_{\overrightarrow{y}}(t_j) = \#bvl(\sigma(x)) - 1 - m \leq b(x) - 1 - m$, which also implies that $b'(x_j) < b(x)$ for $1 \leq j \leq k$. We define $\sigma'(x_j) := \lambda y_1, \ldots, y_m.t_j$ and $\sigma'(y) := \sigma(y)$ for $x \neq y \in FV(S)$. Obviously this definition is compatible with bound b' defined in Step (b). It is trivial to verify that σ' is a Σ_0-unifier for the system reached after Step (c). Clearly the exponent of periodicity of σ' does not exceed the exponent of periodicity of σ. Using completeness of decomposition we are done.

In the remaining cases, some variable $x \in M$ has a value $\sigma(x)$ of the form $\lambda y_1, \ldots, y_m.f\ (t_1, \ldots, t_k)$ where $f \in \Sigma_0$.

Case 2. If there exists a variable $x \in M$ where $\sigma(x)$ has the form (2) described in Remark 6.1, then we apply (reduce-split) using x (Alternative 2). For the new variables we define $b'(x_j) := m + \#bvl_{\overrightarrow{y}}(t_j)$ $(1 \leq j \leq k)$. The assumptions on form (2) show that $b'(x_j) < m + \#bvl(\sigma(x)) - m = \#bvl(\sigma(x)) \leq b(x)$ for $j = 1, \ldots, k$. In addition we have $\sum_{j=1}^{k}(b'(x_j) - m) = \sum_{j=1}^{k} \#bvl_{\overrightarrow{y}}(t_j) = \#bvl(\sigma(x)) - m \leq b(x) - m$. Defining $\sigma'(x_j)$ as $\lambda\overrightarrow{y}.t_j$ is consistent with b'. It is trivial to verify that σ' is a Σ_0-unifier for the system reached after Step (c). The exponent of periodicity of σ' does not exceed the exponent of periodicity of σ. Using completeness of decomposition we are done.

In the remaining case, some variable $x \in M$ has a value $\sigma(x)$ of the form $\lambda y_1, \ldots, y_m.f\ (t_1, \ldots, t_k)$ where $f \in \Sigma_0$, and for at most one subterm t_i we have $\#bvl_{\overrightarrow{y}}(t_i) \neq 0$.

Case 3. If there exists a variable $x \in M$ where $\sigma(x)$ has the form (3) described in Remark 6.1, then we apply (reduce-binder) using x (Alternative 3). Here f and i are as above. We define $\sigma'(x') := \lambda\overrightarrow{y}.t_i$ and $\sigma'(z_j) := t_j$ for $j = 1, \ldots, i-1, i+1, \ldots, k$. The definition respects b'. It is trivial to verify that σ' is a Σ_0-unifier for the system reached after Step (c). The exponent of periodicity of σ' does not exceed the exponent of periodicity of σ. Using completeness of decomposition we are done.

In the remaining *Case 4*, every value $\sigma(x)$ for $x \in M$ is of the form (4) described in Remark 6.1. The result follows.□

7 Rules for Type "nocycle"

Let (S, b) denote a BHOUP of type "nocycle", with a set of variables $\mathcal{V}_S := FV(S)$. Let the relations "$\sim_1$" and "$>_1$" on $\mathcal{V}_S$ be defined as follows: if there

exists an equation $x\ s_1\ \ldots s_n \doteq y\ t_1 \ldots\ t_m \in S$, then $x \sim_1 y$. If there exists an equation $x\ s_1\ \ldots s_n \doteq t \in S$, and t has some function symbol f as head, y is on the surface of t, then $x >_1 y$.

Let "$\sim$" denote the equivalence relation in $\mathcal{V}_S$ generated by $\sim_1$. Denote the equivalence class of a variable x by $[x]_\sim$. For equivalence classes D_1, D_2 of $\mathcal{V}_S/\sim$ define $D_1 \rhd_1 D_2$ if there exist $x_i \in D_i$ for $i = 1, 2$ such that $x_1 >_1 x_2$. Let "$\rhd$" denote the transitive closure of "$\rhd_1$".

Lemma 7.1. *If the decomposed BHOUP (S, b) is of type "nocycle", then the relation "$\rhd$" is an irreflexive partial order on $\mathcal{V}_S/\sim$.*

Proof. Assume that "$\rhd$" is not irreflexive. Then there exists a sequence $x_1\ \overrightarrow{s_1} \doteq t_1, \ldots, x_h\ \overrightarrow{s_h} \doteq t_h$ of length $h \geq 1$ of equations from S such that x_i occurs on the surface of $t_{i-1 \bmod^* h}$ for $1 \leq i \leq h$. Moreover, there is at least one term t_i of the form $f(t_{i,1}, \ldots, t_{i,n})$. Since the sequence does not represent a cycle, all x_i have arity 0. But then decomposition rule (occurs-check) would lead to failure, a contradiction.□

Definition 7.2. (Imitation) *Let (S, b) be a decomposed BHOUP of type "nocycle". Select a $\rhd$-maximal $\sim$-equivalence class D and a function symbol f according to the following conditions: There must be an equation $z\ \ldots \doteq f \ldots$ in S where $z \in D$. Let $k := \mathrm{ar}(f)$. Select one of the following two alternatives. The second alternative is only possible if $f \in \Sigma_0$ has arity $k \geq 1$ and if all arguments of f have elementary type, i.e. f is a first-order function symbol.*

1. *Apply (reduce-bv), (reduce-split) or (reduce-binder) using a variable $x \in D$ with $b(x) > \mathrm{ar}(x)$.*
2. *Apply the following steps:*
   (a) *For every variable $x \in D$ select an index j_x with $1 \leq j_x \leq k$. Instantiate x by the $\beta\overline{\eta}$-normal form of*

$$\lambda y_1, \ldots, y_{ar(x)}.f(z_1, \ldots, z_{j_x-1}, (x'\ y_1 \ldots y_{ar(x)}), z_{j_x+1} \ldots, z_k)$$

   where the $z_i, i = 1, \ldots k, i \neq j_x$ are fresh first-order variables and x' is a new variable of appropriate type. Define $b'(x') := b(x)$ and $b(z_i) = 0$ for $i \in \{1, \ldots, k\}, i \neq j_x$.
   (b) *Use (β)-reduction to transform the terms into $\beta\overline{\eta}$-normal form.*
   (c) *Decompose the resulting BHOUP.*

Lemma 7.3. *Application of the the rule (imitation) to a decomposed BHOUP (S, b) of type "nocycle" either fails or results in a BHOUP (S^*, b^*), such that $\mu(S^*, b^*) < \mu(S, b)$.*

Proof. A $\rhd$-maximal equivalence class with the required properties exists, since there are no cycles, there is no occurs-check failure, and the BHOUP is not of type xy.

If Alternative 1 is selected, then the result follows from Lemma 6.7. If Alternative 2 is selected, since all arguments of f have elementary type, each

$x \in D$ is instantiated with a term $\lambda\overrightarrow{y}.f(z_1,\ldots,z_{j_x-1},x'\ \overrightarrow{y\downarrow_{\beta\overline{\eta}}},z_{j_x+1},\ldots,z_n)$. Since $b'(x')=b(x)$ and $ar(x')=ar(x)$ neither μ_1 nor μ_2 are affected. If S^* contains a cycle, then μ_3 is reduced. Hence we may assume that S^* does not have a cycle. Then none of the systems reached before or during Step (d) has a cycle, by Lemma 5.15. Hence rule (decomp-repvt) is not applied.

Note that all surface occurrences of variables $x \in D$ are the distinguished occurrences in equations $x\overrightarrow{r} \doteq y\overrightarrow{s}$ or $x\overrightarrow{r} \doteq f\overrightarrow{s}$. We consider the modifications of μ_4,μ_5 that result from the treatment of each equation. We first consider the equations of the form $x\overrightarrow{r} \doteq y\overrightarrow{s}$ in S for $x,y \in D$. The instantiation of x,y, after β-reduction yields equations

$$f(z_1,\ldots,x'\ \overrightarrow{r'},\ldots,z_n) \doteq f(z'_1,\ldots,y'\ \overrightarrow{s'},\ldots,z'_n).$$

Via decomposition both occurrences of f are removed. The decomposition of the successor equations only uses (repvv). We do not obtain new function symbols on the surface, perhaps some first-order variables are added. Replacements of variable occurrences in $x\overrightarrow{r} \doteq y\overrightarrow{s}$ that are not on the surface do not modify μ_4,μ_5. Hence, after Step (d), we do not have any new contribution to measure components μ_4,μ_5 from equations $x\overrightarrow{r} \doteq y\overrightarrow{s}$.

For the equation(s) $x\overrightarrow{r} \doteq f\overrightarrow{s}$ in S the instantiation of x after β-reduction yields

$$f(z_1,\ldots,x'\ \overrightarrow{r'},\ldots,z_n) \doteq f(s_1,\ldots,s_{i_0},\ldots,s_n).$$

Via decomposition both occurrences of f are removed. The rest is as above. Hence, after Step (d), there is one term that is replaced by some terms of a smaller size, and at least one surface occurrence of f is removed. Hence μ_4 and μ_5 are strictly decreased. □

Lemma 7.4. *The rule (imitation) is sound and complete.*

Proof. Let (S,b) be a BHOUP before application of the rule.
Soundness. Assume there is a Σ_0-unifier σ^* of the BHOUP (S^*,b^*) reached after the transformation. If Alternative 1 is selected, then Lemma 6.7 shows that (S,b) has a Σ_0-unifier.

If Alternative 2 is used, soundness of decomposition shows that there exists a Σ_0-unifier σ' of the BHOUP (S',b') reached before the final decomposition. We now show that there is a Σ_0-unifier for (S,b). For each $x \in D$, let $m=ar(x)$ and define $\sigma(x)$ as the $\beta\overline{\eta}$-normal form of the σ'-image of the $\beta\overline{\eta}$-normal form of $\lambda y_1,\ldots,y_m.f(z_1,\ldots,z_{j_x-1},(x'\ y_1\ldots y_m),z_{j_x+1}\ldots,z_n)$. Is is trivial to see that σ is a Σ_0-unifier for S. Since for $i \neq j_x$ always $\sigma'(z_i)$ is a ground first-order term (cf. Lemma 3.3) we see as in the proof of Lemma 6.7 (Soundness) that $\#bvl(\sigma(x)) \leq m+b'(x')-m=b(x)$. It follows that σ is a Σ_0-unifier for (S,b).

Completeness. Let σ be a Σ_0-unifier of (S,b). It follows from Lemma 6.8 that it suffices to consider the case where each variable $x \in D$ has a value $\sigma(x)$ of the form $\lambda y_1,\ldots,y_{ar(x)}.f\ (t_1,\ldots,t_k)$ where for at most one subterm t_i we have $\#bvl_{\overrightarrow{y}}(t_i) \neq 0$ and every argument of f has elementary type. Obviously f must be the function symbol mentioned in the rule (Imitation). Here we use

Alternative 2. Let $x \in D$ and let $\sigma(x) = \lambda\overrightarrow{y}.f(t_1, \ldots, t_n)$. In Step (a), if there exists a term t_i such that $\#bvl_{\overrightarrow{y}}(t_i) \neq 0$, then select $j_x := i$, in the other case the selection of j_x is arbitrary. We define $\sigma'(x') := \lambda\overrightarrow{y}.t_i$ and $\sigma'(x) := \sigma(y)$ for $x \neq y$. Images of fresh first-order variables are obvious. The definition respects b'. It is trivial to verify that σ' is a Σ_0-unifier for the system reached after Step (b). The exponent of periodicity of σ' does not exceed the exponent of periodicity of σ. Since decomposition is complete we are done. □

Lemma 7.5. *The rule (Imitation) either fails or transforms a decomposed BHOUP (S, b) in $\beta\overline{\eta}$-normal form into a decomposed BHOUP (S', b') in $\beta\overline{\eta}$-normal form. Moreover, $subt(S') \subseteq subt(S)$.*

Proof. By inspecting the rule. □

8 Rules for Type "amb"

Before we describe the treatment of BHOUPs of type "amb" we introduce a rule that is used to replace surface occurrences of first-order variables by a term t, if the equation $x \doteq t$ is in the problem set. It is not used for decomposition, since for BHOUPs of type "nocycle" it would in general increase the measure.

(repvt) $\dfrac{\{x \doteq t\} \cup S}{\{x \doteq t\} \cup S'}$ S' is constructed from S by replacing all surface occurrences of x by t. The variable x must be a first-order variable.

Now let (S, b) denote a problem of type "amb". Recall that S is decomposed and has a ψ-minimal cycle L that is non-path-unique. We may assume that L has the form $x_1\ \overrightarrow{s_1} \doteq t_1, \ldots, x_h\ \overrightarrow{s_h} \doteq t_h$. The cycle could as well be represented as $x_1\ \overrightarrow{s_1} \doteq C_1[t'_1], \ldots, x_h\ \overrightarrow{s_h} \doteq C_h[t'_h]$, where C_i are the relevant contexts (see Definition 5.4).

Definition 8.1. **(solve-ambiguous-cycle)**. *The input is the decomposed BHOUP (S, b) of type "amb" with a ψ-minimal cycle L as described above. Select one of the following two alternatives.*

1. *Apply (reduce-bv), (reduce-split) or (reduce-binder) using a variable $x \in \{x_1, \ldots, x_h\}$ with $b(x) > \mathrm{ar}(x)$.*
2. *Select an index j such that $x_j\overrightarrow{s_j} \doteq t_j$ is an equation in L where $x_{(j+1 \bmod^* h)}$ occurs at least twice on the surface of $t_j = f\ t_{j,1}\ \ldots\ t_{j,k}$ and the main depth of the relevant context C_j is minimal in L. If f has an argument with non-elementary type, then fail.*
   Now apply the following steps:
   (a) *Select an index $r \in \{1, \ldots, k\}$. In the special situation where $h = 1$, the selection of r is subject to the following condition: all surface occurrences of x_1 in $f(t_{1,1}, \ldots, t_{1,k})$ have to be in $t_{1,r}$. If this is not possible since C_1 is trivial, then stop with fail.*

(b) Instantiate x_j *by* $\lambda\overrightarrow{y}.f(z_1,\ldots,z_{r-1},x'_j\ \overrightarrow{y\downarrow_{\beta\overline{\eta}}},z_{r+1}\ \ldots\ z_k)$ *where the* z_i *are fresh first-order variables* $(1 \leq i \leq k, i \neq r)$*, and* x'_j *is a fresh variable.*

(c) Define $b'(z_i) := 0$ *and* $b'(x'_j) := b(x_j)$*.*

(d) Use β*-reduction until a* $\beta\overline{\eta}$*-normal form is reached for every term in the system.*

(e) Apply rule (decomp) to the equation that is obtained from the equation $x_j\overrightarrow{s_j} \doteq t_j$ *in Step (d).*

(f) Apply (repvt) for all the new equations $z_i \doteq t_{j,i}$ $(1 \leq i \leq k, i \neq r)$ *that are obtained from the previous step.*

(g) Then decompose the resulting BHOUP.

Lemma 8.2. *Application of the the rule (solve-ambiguous-cycle) to a BHOUP* (S,b) *of type "amb" either fails or results in a BHOUP* (S^*,b^*)*, such that* $\mu(S^*,b^*) < \mu(S,b)$*.*

Proof. If Alternative 1 is selected, then the result follows from Lemma 6.7.

Assume that Alternative 2 is selected. By Lemma 5.14 it suffices to show that the system (S',b') reached after Step (f) satisfies $\mu(S',b') < \mu(S,b)$. Obviously Steps (a)-(f) do not affect the measures μ_1 and μ_2. Note that in (b) we have $ar(x'_j) = ar(x_j)$. We now show that μ_3 is reduced.

We first assume that $h > 1$ and consider the relevant equation, its predecessor and successor equation (for $h = 2$, the first and the third equation are identical).

$$\begin{aligned} x_{j-1}\overrightarrow{q} &\doteq t_{j-1}\lfloor x_j\rfloor \\ x_j\overrightarrow{r} &\doteq f(t_{j,1},\ldots,t_{j,k})\lfloor x_{j+1}\rfloor \\ x_{j+1}\overrightarrow{s} &\doteq t_{j+1} \end{aligned}$$

Instantiating x_j plus beta-reductions yields the equations

$$\begin{aligned} x_{j-1}\overrightarrow{q'} &\doteq t'_{j-1}\lfloor f(z_1,\ldots,z_{r-1},x'_j\ \overrightarrow{\cdot\cdot},z_{r+1}\ \ldots\ z_k)\rfloor \\ f(z_1,\ldots,z_{r-1},x'_j\ \overrightarrow{r'},z_{r+1}\ \ldots\ z_k) &\doteq f(t'_{j,1},\ldots,t'_{j,k})\lfloor x_{j+1}\rfloor \\ x_{j+1}\overrightarrow{s'} &\doteq t'_{j+1}. \end{aligned}$$

Here primed terms are obtained from unprimed predecessors via instantiation. The arguments represented as $\overrightarrow{\cdot\cdot}$ depend on the arguments of the respective surface occurrence(s) of x_j in t_{j-1}. Decomposition of the central equation in Step (e) yields

$$\begin{aligned} x_{j-1}\overrightarrow{q'} &\doteq t'_{j-1}\lfloor f(z_1,\ldots,z_{r-1},x'_j\ \overrightarrow{\cdot\cdot},z_{r+1}\ \ldots\ z_k)\rfloor \\ x'_j\ \overrightarrow{r'} &\doteq t'_{j,r} \\ x_{j+1}\overrightarrow{s'} &\doteq t'_{j+1}. \end{aligned}$$

plus the equations $z_i \doteq t'_{j,i}$ $(i \neq r)$. Replacing the z_i by $t'_{j,i}$ in Step (f) now yields

$$\begin{aligned} x_{j-1}\overrightarrow{q''} &\doteq t''_{j-1}\lfloor f(t'_{j,1},\ldots,t'_{j,r-1},x'_j\ \overrightarrow{\cdots},t'_{j,r+1}\ \ldots\ t'_{j,k})\rfloor \\ x'_j\ \overrightarrow{r''} &\doteq t''_{j,r} \\ x_{j+1}\overrightarrow{s''} &\doteq t''_{j+1}. \end{aligned}$$

First assume that there exists at least one index $l \neq r$ such that $t_{j,l}$ has a surface occurrence of x_{j+1}. Since the cycle has minimal length and $h > 1$ we have $x_{j+1} \neq x_j$. It follows that $t'_{j,l}$ has a surface occurrence of x_{j+1}. This shows that the equations

$$\begin{aligned} x_{j-1}\overrightarrow{q''} &\doteq t''_{j-1}\lfloor f(t'_{j,1},\ldots,t'_{j,r-1},x'_j\ \overrightarrow{\cdots},t'_{j,r+1}\ \ldots\ t'_{j,k})\rfloor \\ x_{j+1}\overrightarrow{s''} &\doteq t''_{j+1}. \end{aligned}$$

together with the images of the remaining equations of L represent a cycle of length $h-1$. Note that the conditions for a cycle are satisfied since $t''_{j-1}\lfloor f(t'_{j,1},\ldots,t'_{j,r-1},x'_j\ \overrightarrow{\cdots},t'_{j,r+1}\ \ldots\ t'_{j,k})\rfloor$ contains a function symbol as head. Hence, after Steps (a)-(f) we reach a system with smaller ψ_1-measure.
Now assume that all surface occurrences of x_{j+1} belong to $t_{j,r}$. This means that x_{j+1} occurs at least twice on the surface of $t'_{j,r}$. Together with the images of the remaining equations of L the equations

$$\begin{aligned} x_{j-1}\overrightarrow{q''} &\doteq t''_{j-1}\lfloor f(t'_{j,1},\ldots,t'_{j,r-1},x'_j\ \overrightarrow{\cdots},t'_{j,r+1}\ \ldots\ t'_{j,k})\rfloor \\ x'_j\ \overrightarrow{r''} &\doteq t''_{j,r}\lfloor x_{j+1}\rfloor \\ x_{j+1}\overrightarrow{s''} &\doteq t''_{j+1}. \end{aligned}$$

represent a cycle of the system reached after Step (d). The main depth of the relevant context of the equation with index j is decreased. Note that if x_j has two surface occurrences in t_{j-1}, the main depth of the relevant context of equation $j-1$ is not affected. The new cycle is non-path-unique, hence it has smaller ψ-measure than L. Hence after Steps (a)-(d) we reach a system with smaller ψ-measure.

It remains to consider the case $h = 1$. Let the equation be

$$x_1\overrightarrow{r} \doteq f(t_1,\ldots,t_m)\lfloor x_1\rfloor.$$

Here x_1 has at least two surface occurrences in t_r. Instantiation and beta-reductions yield

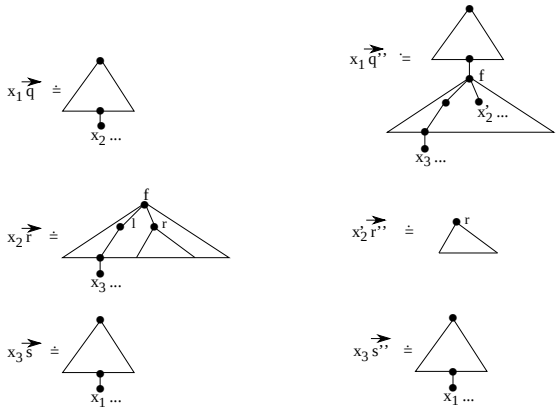
$$\begin{aligned} & f(z_1,\ldots,z_{r-1},(x'_1\ \overrightarrow{r'}),z_{r+1}\ \ldots\ z_k) \\ \doteq\ & f(t'_1,\ldots,t'_{r-1},t'_r\lfloor f(z_1,\ldots,z_{r-1},(x'_1\ \overrightarrow{\cdots}),z_{r+1}\ \ldots\ z_m)\rfloor,t'_{r+1},\ldots,t'_m). \end{aligned}$$

Decomposition gives

$$x'_1\ \overrightarrow{r'} \doteq t'_r\lfloor f(z_1,\ldots,z_{r-1},(x'_1\ \overrightarrow{\cdots}),z_{r+1}\ \ldots\ z_k)\rfloor.$$

We have a non-path-unique cycle where the depth of the main context is smaller than before. As above it follows that we reach a system with smaller ψ-measure. □

The following figure illustrates the first situation considered in the preceding proof where $h = 3$ and $j = 2$.



Lemma 8.3. *The rule (solve-ambiguous-cycle) is sound and complete.*

Proof. Let (S, b) be a BHOUP before application of the rule.
Soundness. Assume there is a Σ_0-unifier σ^* of the BHOUP (S^*, b^*) reached after the transformation. If Alternative 1 is selected, then Lemma 6.7 shows that (S, b) has a Σ_0-unifier.

If Alternative 2 is used, soundness of decomposition shows that there exists a Σ_0-unifier σ' of the BHOUP (S', b') reached before the final decomposition. Obviously, if $t = \lambda\overrightarrow{y}.f(z_1, \ldots, z_{r-1}, x'_j\ \overrightarrow{y{\downarrow_{\beta\overline{\eta}}}}, z_{r+1}\ \ldots\ z_k)$ denotes the substitute for x_j as defined in Step (b) of Alternative 2, then $\sigma(x_j) := \sigma'(t){\downarrow_{\beta\overline{\eta}}}$ and $\sigma(y) := \sigma'(y)$ for all free variables $y \neq x_j$ of S defines a Σ_0-unifier for S. To prove soundness it remains to show that σ respects bound b. We have

$$\sigma(x_j) = \lambda\overrightarrow{y}.f(\sigma'(z_1), \ldots, \sigma'(z_{r-1}), [\sigma'(x'_j)\ \overrightarrow{y{\downarrow_{\beta\overline{\eta}}}}]{\downarrow_{\beta\overline{\eta}}}, \sigma'(z_{r+1})\ \ldots\ \sigma'(z_k))$$

where $\#bvl_{\overrightarrow{y}}(\sigma'(z_i)) = 0$ for $i \in \{1, \ldots, k\}, i \neq r$. Lemma 6.5 shows that $\#bvl_{\overrightarrow{y}}([\sigma'(x'_j)\ \overrightarrow{y{\downarrow_{\beta\overline{\eta}}}}]{\downarrow_{\beta\overline{\eta}}}) = \#bvl(\sigma'(x'_j)) - m_j$ where m_j is the length of $\overrightarrow{y}$. It follows that $\#bvl(\sigma(x_j)) = m_j + \#bvl(\sigma'(x'_j)) - m_j = \#bvl(\sigma'(x'_j)) \leq b'(x'_j) = b(x_j)$, which shows that σ respects b.

Completeness. Let σ be a Σ_0-unifier of (S, b). It follows from Lemma 6.8 that it suffices to consider the case where any variable $x \in \{x_1, \ldots, x_h\}$ has a value $\sigma(x)$ of the form $\lambda y_1, \ldots, y_m.f(t_1, \ldots, t_k)$ where for at most one subterm t_i we have $\#bvl_{\overrightarrow{y}}(t_i) \neq 0$ and every argument of f has elementary type. Here we use Alternative 2. We select an index j such that $x_j\overrightarrow{s_j} \doteq t_j$ is an equation in L where $x_{(j+1 \bmod^* h)}$ occurs at least twice on the surface of $t_j = f(t_{j,1}, \ldots, t_{j,k})$ and the main depth of the relevant context C_j is minimal in L. Let $\sigma(x_j) =$

$\lambda y_1, \ldots, y_m.f(t_1, \ldots, t_k)$, let r be an index such that $\#bvl_{\overrightarrow{y}}(t_r) = \#bvl(\sigma(x)) - m$. We select this index in Step (a).

We have to show that for $h = 1$ all surface occurrences of x_1 in $f(t_{1,1}, \ldots, t_{1,k})$ are in $t_{1,r}$. For index $i \neq r$, the subterms t_i of $\sigma(x_1) = \lambda y_1, \ldots, y_m.f(t_1, \ldots, t_k)$ are ground first-order terms. Hence we have $\sigma(x_1 \overrightarrow{s_1}) = f(t_1, \ldots, t_{r-1}, t'_r, t_{r+1}, \ldots, t_k) = f(\sigma(t_{1,1}), \ldots, \sigma(t_{1,k}))$. Assume that for some $i \neq r$ variable x_1 occurs on the surface of $t_{1,i}$. Since t_i is a proper subterm of $\sigma(x_1)$ and $\sigma(x_1)$ is a subterm of $\sigma(t_{1,i}) = t_i$ we obtain a contradiction.

Returning to the general case with j, r as above it is simple to see that $\sigma'(x'_j) := \lambda \overrightarrow{y}.t_r$ and $\sigma(z_i) := t_i$ for $i \neq r, 1 \leq i \leq k$ defines a Σ_0-unifier for (S', b'). It obviously satisfies bound b' and has an exponent of periodicity that does not exceed the exponent of periodicity of σ. From completeness of decomposition rules it follows that we are done. □

Lemma 8.4. *The rule (solve-ambiguous-cycle) either fails or transforms a decomposed BHOUP (S, b) in $\beta\overline{\eta}$-normal form into a decomposed BHOUP (S', b') in $\beta\overline{\eta}$-normal form. Moreover, $subt(S') \subseteq subt(S)$.*

Proof. By inspecting the rule. □

9 Rules for Type "unique"

In this section we distinguish between two types of path-unique cycles.

Definition 9.1. *Let (S, b) be a BHOUP. A term t occurring in S is* b-constrained *iff either $t \in FV(S)$ is a first-order variable with $b(t) = 0$, or there exists an equation $z \doteq t$ in S such that $b(z) = 0$. A context $f(t_1, \ldots, t_{r-1}, [\cdot], t_{r+1}, \ldots, t_k)$ appearing in S is called* b-constrained *iff f is a first-order function symbol, and every subterm t_i $(i = 1, \ldots, r-1, r+1, \ldots, r_k)$ is b-constrained. A non-empty context C is* b-constrained *iff every subcontext of main depth 1 is b-constrained.*

Definition 9.2. *A cycle L of length h is* special path-unique *if the following holds:*

- *it is path-unique,*
- *only the relevant context of the last equation C_h is non-trivial,*
- *the context C_h is b-constrained.*

The following rule (shuffle) does not necessarily decrease μ; it may increase the ψ_3- component of the measure component μ_3. It can be applied to a BHOUP and a cycle L only in the context of a larger procedure where we will be able to decrease μ eventually.

Definition 9.3. Sub-rule (shuffle) *The input is a decomposed BHOUP (S, b) and a path-unique cycle L in (S, b) of length $h \geq 2$ of the form $x_1 \overrightarrow{s_1} \doteq C_1[x_2 \overrightarrow{t_1}], \ldots, x_h \overrightarrow{s_h} \doteq C_h[x_1 \overrightarrow{t_h}]$ where L contains at least two non-trivial relevant contexts C_j and $C_{j'}$. Select one of the following alternatives.*

1. *Apply (reduce-bv), (reduce-split) or (reduce-binder) using a variable* $x \in \{x_1, \ldots, x_h\}$ *with* $b(x) > \mathrm{ar}(x)$.
2. *Let* $C_j[x_{j+1}\overrightarrow{t_j}]$ *have the form* $f(t_{j,1}, \ldots, t_{j,s-1}, t_{j,s}\lfloor x_{j+1}\rfloor, t_{j,s+1}, \ldots, t_{j,k})$.
   (a) *Fail, if* f *has an argument position of non-elementary type, or if* $f \notin \Sigma_0$. *Select an index* $1 \leq r \leq k$.
   (b) *Instantiate* x_j *by* $\lambda\overrightarrow{y}.f(z_1, \ldots, z_{r-1}, x'_j\ \overrightarrow{y\downarrow_{\beta\overline{\eta}}}, z_{r+1}, \ldots, z_k)$, *where the* z_i $(1 \leq i \leq k, i \neq r)$ *are fresh first-order variables and* x'_j *is a fresh variable.*
   (c) *Define* $b'(z_i) := 0$ $(1 \leq i \leq k, i \neq r)$ *and* $b'(x'_j) := b(x_j)$.
   (d) *Use* β*-reduction to reach a* $\beta\overline{\eta}$*-normal form.*
   (e) *Apply (decomp) to the* j^{th} *equation of* L *after the instantiation, i.e. to*

$$f(z_1, \ldots, z_{r-1}, x'_j\ \overrightarrow{s_j}, z_{r+1}, \ldots, z_k) \doteq f(t_{j,1}, \ldots, t_{j,s-1}, t_{j,s}\lfloor x_{j+1}\rfloor, t_{j,s+1}, \ldots, t_{j,k}).$$

   (f) *Apply (repvt) or (repvv) for all equations* $z_i \doteq t_{j,i}$ *for* $i \neq r$ *added by the last step.*
   (g) *Then decompose the resulting BHOUP.*

Lemma 9.4. *The rule (shuffle) is sound and complete.*

Proof. The reader should note that the procedure in Alternative 2 is the same as in Alternative 2 of the rule (solve-ambiguous-cycle), modulo the irrelevant origin of the variable x_j. For (shuffle), the situation $h = 1$ does not occur by assumption. Hence soundness and completeness of (shuffle) can be shown exactly as in the proof of Lemma 8.3. □

In the following, let $\overline{\mu}$ be the measure with the three components μ_1, μ_2, and as third component, the minimal length of a cycle in (S, b).

Lemma 9.5. *Let* (S, b) *be a decomposed BHOUP with a path* L *of minimal length* $h \geq 2$ *that is path-unique. Let* (S', b') *be obtained from* (S, b) *by an application of (shuffle) using* L. *Then one of the following cases holds:*

1. $\overline{\mu}(S', b') < \overline{\mu}(S, b)$,
2. $r = s$, *the system* (S', b') *is decomposed and contains a path-unique cycle* L' *of length* h *such that the relevant contexts* $C'_1, \ldots, C'_h$ *have main depth corresponding to* $C_1, \ldots, C_h$ *except for indices* j *and* $j - 1$ $\mathtt{mod}*$ h. *We have* $|C'_{j-1 \bmod* h}| = |C_{j-1 \bmod* h}| + 1$ *and* $|C'_j| = |C_j| - 1$. *In* $C'_{j-1 \bmod* h}$, *the suffix subcontext of main depth* 1 *is* b*-constrained.*

Proof. If Selection 1 is used, then $\overline{\mu}(S', b') < \overline{\mu}(S, b)$ (cf. Lemma 6.7).

Assume that Selection 2 is used. Then μ_1 and μ_2 are not affected since $\mathrm{ar}(x_j) = \mathrm{ar}(x'_j)$. First consider the case where $r \neq s$. Let the equations with indices $j - 1$ $\mathtt{mod}*$ h, j and $j + 1$ $\mathtt{mod}*$ h be

$$\begin{aligned} x_{j-1}\ \overrightarrow{r_{j-1}} &\doteq C_{j-1}\lfloor x_j\ \overrightarrow{t_{j-1}}\rfloor \\ x_j\ \overrightarrow{r_j} &\doteq f(t_{j,1}, \ldots, t_{j,s-1}, t_{j,s}\lfloor x_{j+1}\rfloor, t_{j,s+1}, \ldots, t_{j,k}) \\ x_{j+1}\ \overrightarrow{r_{j+1}} &\doteq t. \end{aligned}$$

Path uniqueness and length-minimality imply that L contains only the explicitly indicated surface occurrences of x_j. Instantiation (b) and β-reductions give successor equations of the form

$$x_{j-1}\ \overrightarrow{r'_{j-1}} \doteq C'_{j-1}\lfloor f(z_1,\ldots,z_{r-1},(x'_j\ \overrightarrow{t'_{j-1}}),z_{r+1},\ldots,z_k)\rfloor$$
$$f(z_1,\ldots,z_{r-1},(x'_j\ \overrightarrow{r'_j}),z_{r+1},\ldots,z_k) \doteq f(t'_{j,1},\ldots,t'_{j,s-1},t'_{j,s}\lfloor x_{j+1}\rfloor,t'_{j,s+1},\ldots,t'_{j,k})$$
$$x_{j+1}\ \overrightarrow{r'_{j+1}} \doteq t'.$$

Applying (decomp) to the middle equation yields (among others) the equation $z_s \doteq t'_{j,s}\lfloor x_{j+1}\rfloor$. Applying (repvt) or (repvv) we obtain a cycle L_1 of length $h-1$:

$$x_{j-1}\ \overrightarrow{r'_{j-1}} \doteq C'_{j-1}\lfloor f(z_1,\ldots,z_{r-1},(x'_j\ \overrightarrow{t'_{j-1}}),z_{r+1},\ldots,z_{s-1},t'_{j,s}\lfloor x_{j+1}\rfloor,z_{s+1},\ldots,z_k)\rfloor$$
$$x_{j+1}\ \overrightarrow{r'_{j+1}} \doteq t'.$$

After decomposition we have a successor cycle L' of length $h-1$ in the system (S',b') that is reached (cf. Lemma 5.15). Hence $\overline{\mu}(S',b') < \overline{\mu}(S,b)$.

It remains to consider the case where $r=s$. If $\overline{\mu}(S',b') < \overline{\mu}(S,b)$ we are done. Assume that $\overline{\mu}(S',b') \geq \overline{\mu}(S,b)$. Since $r=s$, instantiation (c) and β-reductions lead to

$$x_{j-1}\ \overrightarrow{r'_{j-1}} \doteq C'_{j-1}\lfloor f(z_1,\ldots,z_{r-1},(x'_j\ \overrightarrow{t'_{j-1}}),z_{r+1},\ldots,z_k)\rfloor$$
$$f(z_1,\ldots,z_{r-1},(x'_j\ \overrightarrow{r'_j}),z_{r+1},\ldots,z_k) \doteq f(t'_{j,1},\ldots,t'_{j,r-1},t'_{j,r}\lfloor x_{j+1}\rfloor,t'_{j,r+1},\ldots,t'_{j,k})$$
$$x_{j+1}\ \overrightarrow{r'_{j+1}} \doteq t'.$$

Applying (decomp) to the middle equation yields a variant L_1 of L of length h with equations

$$x_{j-1}\ \overrightarrow{r'_{j-1}} \doteq C'_{j-1}\lfloor f(z_1,\ldots,z_{r-1},(x'_j\ \overrightarrow{t'_{j-1}}),z_{r+1},\ldots,z_k)\rfloor$$
$$x'_j\ \overrightarrow{r'_j} \doteq t'_{j,r}\lfloor x_{j+1}\rfloor$$
$$x_{j+1}\ \overrightarrow{r'_{j+1}} \doteq t'.$$

(S',b') is the system reached after decomposition. As we have seen in Lemma 5.15, S' contains a cycle L' of length h corresponding to L_1. Note that the variables $x_1,\ldots,x_h$ cannot be first-order since (S,b) is decomposed. Hence possible application of decomposition rules (repvv) and (decomp-repvt) do not affect path-uniqueness and L' is again path-unique. Obviously L' has the properties demanded in Situation 2 above. Note that $f(z_1,\ldots,z_{r-1},[\cdot],z_{r+1},\ldots,z_k)$ is b-constrained. Decomposition does not affect this property. Hence the result follows. □

Definition 9.6. (shuffle*) *Let (S_0,b_0) be a BHOUP and let L be a ψ-minimal path-unique cycle of length h with at least two non-trivial relevant contexts C_j and $C_{j'}$. Let $(S,b)=(S_0,b_0)$. Iterate (shuffle) as follows:*

1. *First select an index j in the cycle L such that C_j is non-trivial.*
2. *Apply (shuffle) for index j, yielding the BHOUP (S', b').*
3. *If $\mu(S', b') < \mu(S_0, b_0)$, then return (S', b').*
4. *Otherwise, let L' be the path-unique cycle obtained from L. If C'_j is nontrivial, then go to 2 using the same index.*
   If C'_j is trivial, but still L' has at least two non-trivial relevant contexts, then go to 2, replacing (S, b) by (S', b') and using the index $j - 1 \bmod* h$ instead of j. Otherwise stop.

Definition 9.7. (shuffle)** *Let (S, b) be a BHOUP, let L be a minimal-length cycle of S of length h that is path-unique and contains exactly one non-trivial relevant context, say C_h. If L is not special path-unique, then iterate (shuffle) as follows:*

1. *Apply (shuffle) for index h, yielding the BHOUP (S', b').*
2. *If $\mu(S', b') < \mu(S, b)$, then return (S', b').*
3. *Otherwise, let L' be the cycle obtained from L.*
   If C'_h is nontrivial, then go to 1 using the same index.
   If C'_h is trivial, then return (S', b').

Remark 9.8. (shuffle*) is intended to operate on a ψ-minimal path-unique cycle L, where several relevant contexts are non-trivial. The application of (shuffle) shuffles subcontexts of main depth 1 of a relevant context to another index, until two relevant contexts are merged into one.

The rule (shuffle**) should operate in the situation as above, when exactly one relevant context is in the cycle. The operation shuffles the relevant context to the next index, with the intention to clean it, such that after the shuffle, every subcontext of the relevant context is b-constrained.

Lemma 9.9. *Given a BHOUP (S, b) of type "unique" with a ψ-minimal cycle L that is not special path-unique, it is possible using (shuffle*) and (shuffle**) to either reach failure or a BHOUP (S', b') with $\mu(S', b') < \mu(S, b)$, or a BHOUP (S', b') with a ψ-minimal and special path-unique cycle. The whole transformation is sound and complete.*

Proof. If L has at least two non-empty relevant contexts we first apply (shuffle*). If there is no failure and measure μ is not decreased, then we reach the BHOUP (S^*, b^*) with a path-unique cycle L^* with the same length as L that has less non-empty relevant contexts. (If L has only one non-empty relevant context, then let $S^* := S$, $b^* := b$ and $L^* := L$.) If L^* is already special path-unique, then we are ready. In the other case we apply (shuffle**). If there is no failure and measure μ is not decreased, then we eventually reach the BHOUP (S', b') with a special path-unique cycle L' with the same length as L. This holds, since after application of (shuffle**) the new relevant context has only b-constrained subcontexts, and is thus itself b-constrained. Soundness and completeness of the complete procedure directly follow from Lemma 9.4. □

9.1 The Rule for Special Path-unique Cycles

Now we consider the case that there is a ψ-minimal special path-unique cycle. I.e. there is a ψ-minimal cycle with exactly one non-trivial relevant context C_h of the form

$$x_1\overrightarrow{s_1} \doteq x_2\overrightarrow{t_1}, \ldots, x_{h-1}\overrightarrow{s_{h-1}} \doteq x_h\overrightarrow{t_{h-1}}, x_h\overrightarrow{s_h} \doteq C_h[x_1\overrightarrow{t_h}],$$

where C_h is a b-constrained. Since the BHOUP is decomposed it follows that for the variables $x \in \{x_1, \ldots, x_h\}$ we have $b(x) > ar(x)$.

Recall that E is the upper bound on the exponent of periodicity fixed in the algorithm BHU. Recall also that C^e for a context C and a positive integer e means the expanded form $\underbrace{C \ldots C}_{e}$.

Definition 9.10. (solve-special-cycle) *The input is a decomposed BHOUP (S, b) with a ψ-minimal special path-unique cycle L of the form described above. Select one of the following alternatives.*

1. *Apply (reduce-bv), (reduce-split) or (reduce-binder) using a variable $x \in \{x_1, \ldots, x_h\}$.*
2. *(a) Select some $0 \leq e \leq E$ and some (possibly trivial) prefix $C_{h,1}$ of C_h. Let $C_h = C_{h,1}C_{h,2}$.*
   (b) For $i = 1, \ldots, h$, replace x_i by $\lambda\overrightarrow{y_i}.C_h^e C_{h,1}[x_i' \ \overrightarrow{y_i\downarrow_{\beta\overline{\eta}}}]$ where x_i' is new, define $b'(x_i') := b(x_i)$.
   (c) Use β-reduction to transform the system into $\beta\overline{\eta}$-normal form.
   (d) Select an index $1 \leq j \leq h$ and apply (reduce-bv), (reduce-split) or (reduce-binder) using x_j'.
3. *This selection is only applicable if $h > 1$.*
   (a) Select $e \leq E$ and some (possibly trivial) prefix $C_{h,1} \neq C_h$ of C_h, such that $C_h = C_{h,1}C_{h,2}$ and $C_{h,2}$ has a top level function symbol $f \in \Sigma_0$ of arity $n > 1$. If this is not possible, then fail.
   (b) For $i = 1, \ldots, h$ select an index k_i with $1 \leq k_i \leq n$ and instantiate x_i by $\lambda\overrightarrow{y_i}.C_h^e C_{h,1}[f(z_{i,1}, \ldots, z_{i,k_i-1}, x_i' \ \overrightarrow{y_i\downarrow_{\beta\overline{\eta}}}, z_{i,k_i+1}, \ldots, z_{i,n})]$ with new first-order variables $z_{i,l}$. At least one index k_j should be different from $firstdpos(C_{h,2}C_{h,1})$. Define $b'(x_i') := b(x_i)$ for $i = 1, \ldots, h$ and $b'(z_{i,l}) := 0$ for the new first-order variables. Use β-reduction to reach a $\beta\overline{\eta}$-normal form of all terms in S.
   (c) Apply (decomp) to the equations obtained from the equations of L by instantiation.
   (d) Apply (repvt) or (repvv) to all the equations $z_{i,l} \doteq t_{i,l}$ obtained from repeated (decomp) in (c) for the first $h-1$ equations of L,
   (e) Then decompose the resulting BHOUP.

Lemma 9.11. *Application of the the rule (solve-special-cycle) to a BHOUP (S, b) with a ψ-minimal special path unique cycle either fails or results in a BHOUP (S^*, b^*) such that $\mu(S^*, b^*) < \mu(S, b)$.*

Proof. If Alternative 1 is selected, then the result follows from Lemma 6.7.

First assume that Alternative 2 is selected. Since the hole $[\cdot]$ of the context $C_h^e C_{h,1}$ has elementary type it follows that each variable x_i' has the same arity as its predecessor x_i. Since $b'(x_i') = b(x_i)$ for $i = 1, \ldots, h$ it follows that Steps (a)-(c) do not affect the measure components μ_1 and μ_2. The application of a reduction rule in (d) decreases $\langle \mu_1, \mu_2 \rangle$, hence we arrive at a BHOUP (S^*, b^*) such that $\mu(S^*, b^*) < \mu(S, b)$.

Assume now that Alternative 3 is selected. As above we see that the steps do not affect the measure components μ_1 and μ_2. From the cycle equations

$$x_1 \overrightarrow{s_1} \doteq x_2 \overrightarrow{t_1}, \ldots, x_{h-1} \overrightarrow{s_{h-1}} \doteq x_h \overrightarrow{t_{h-1}}, x_h \overrightarrow{s_h} \doteq C_h[x_1 \overrightarrow{t_h}]$$

we obtain after Step (b)

$$\begin{aligned}
& C_h^e C_{h,1}[f(z_{1,1}, \ldots, z_{1,k_1-1}, x_1' \ \overrightarrow{s_1}, z_{1,k_1+1}, \ldots, z_{1,n})] \\
\doteq\ & C_h^e C_{h,1}[f(z_{2,1}, \ldots, z_{2,k_2-1}, x_2' \ \overrightarrow{t_1}, z_{2,k_2+1}, \ldots, z_{2,n})], \\
& \ldots \\
& C_h^e C_{h,1}[f(z_{h-1,1}, \ldots, z_{h-1,k_{h-1}-1}, x_{h-1}' \ \overrightarrow{s_{h-1}}, z_{h-1,k_{h-1}+1}, \ldots, z_{h-1,n})] \\
\doteq\ & C_h^e C_{h,1}[f(z_{h,1}, \ldots, z_{h,k_h-1}, x_h' \overrightarrow{t_{h-1}}, z_{h,k_h+1}, \ldots, z_{h,n})], \\
& C_h^e C_{h,1}[f(z_{h,1}, \ldots, z_{h,k_h-1}, x_h' \overrightarrow{s_h}, z_{h,k_h+1}, \ldots, z_{h,n})] \\
\doteq\ & C_h^{e+1} C_{h,1}[f(z_{1,1}, \ldots, z_{1,k_1-1}, x_1' \overrightarrow{t_h}, z_{1,k_1+1}, \ldots, z_{1,n})].
\end{aligned}$$

Decomposition yields (among others) the equations

$$\begin{aligned}
& f(z_{1,1}, \ldots, z_{1,k_1-1}, x_1' \ \overrightarrow{s_1}, z_{1,k_1+1}, \ldots, z_{1,n}) \\
\doteq\ & f(z_{2,1}, \ldots, z_{2,k_2-1}, x_2' \ \overrightarrow{t_1}, z_{2,k_2+1}, \ldots, z_{2,n}), \\
& \ldots \\
& f(z_{h-1,1}, \ldots, z_{h-1,k_{h-1}-1}, x_{h-1}' \ \overrightarrow{s_{h-1}}, z_{h-1,k_{h-1}+1}, \ldots, z_{h-1,n}) \\
\doteq\ & f(z_{h,1}, \ldots, z_{h,k_h-1}, x_h' \overrightarrow{t_{h-1}}, z_{h,k_h+1}, \ldots, z_{h,n}), \\
& f(z_{h,1}, \ldots, z_{h,k_h-1}, x_h' \overrightarrow{s_h}, z_{h,k_h+1}, \ldots, z_{h,n}) \\
\doteq\ & C_{h,2} C_{h,1}[f(z_{1,1}, \ldots, z_{1,k_1-1}, x_1' \overrightarrow{t_h}, z_{1,k_1+1}, \ldots, z_{1,n})].
\end{aligned}$$

Let $k = \mathit{firstdpos}(C_{h,2} C_{h,1})$. Decompose the above equations and collect the equations that result from pairing terms at index k. Let $C_{h,2} C_{h,1} = C'C''$ where C' has main depth 1. Then in the interval $2 \leq j < h$ all pairs of consecutive equations have either the form

$$\ldots \doteq z_{j,k}, \quad z_{j,k} \doteq \ldots$$

or

$$\ldots \doteq x_j' \ \overrightarrow{t_{j-1}}, \quad x_j' \overrightarrow{s_j} \doteq \ldots$$

The final equation is either

$$z_{h,k} \doteq C''[f(z_{1,1}, \ldots, z_{1,k_1-1}, x_1' \overrightarrow{t_h}, z_{1,k_1+1}, \ldots, z_{1,n})]$$

or

$$x'_h \overrightarrow{s_h} \doteq C''[f(z_{1,1}, \ldots, z_{1,k_1-1}, x'_1 \overrightarrow{t_h}, z_{1,k_1+1}, \ldots, z_{1,n})].$$

If there are only equations $z_{j,k} \doteq z_{j+1,k}$ for all $1 \leq j < h$, then there is a fail due to occurs-check. In the remaining case there occurs at least one pair

$$\ldots \doteq x'_j \ \overrightarrow{t_{j-1}}, \quad x'_j \overrightarrow{s_j} \doteq \ldots$$

for some $2 \leq j < h$. Then the series of equations represents a cycle of length h. Moreover, there is at least one pair

$$\ldots \doteq z_{j,k}, \quad z_{j,k} \doteq \ldots$$

in the cycle since for at least one index j we have $k_j \neq k$ (cf. (b)). Using (repvt) for indices $j < h$ this yields a cycle of length $h-1$. Then also the BHOUP (S^*, b^*) reached after decomposition has a cycle of length $h-1$. This shows that $\mu(S^*, b^*) < \mu(S, b)$. □

Lemma 9.12. *The rule (solve-special-cycle) is sound and complete.*

Proof. Soundness. Assume there is a Σ_0-unifier σ^* of the BHOUP (S^*, b^*) reached after the transformation. If Alternative 1 is selected, then Lemma 6.7 shows that (S, b) has a Σ_0-unifier. Assume that Alternative 2 is selected. By Part (a) of Lemma 6.7 the BHOUP (S', b') reached after Step (c) has a Σ_0-unifier σ'. For each $x_i \in \{x_1, \ldots, x_h\}$ define $\sigma(x_i)$ as the $\beta\overline{\eta}$-normal form of $\lambda\overrightarrow{y}.\sigma'(C^e_h C_{h,1})[\sigma'(x'_i)\overrightarrow{y{\downarrow_{\beta\overline{\eta}}}}]$. For the remaining free variables z of S let $\sigma(z) := \sigma'(z)$. It is simple to show that σ' is a Σ_0-unifier for S. Since the context $C^e_h C_{h,1}$ is b-constrained, the ground context $\sigma'(C^e_h C_{h,1})$ does not contribute to the $\#bvl_{\overrightarrow{y}}$-measure. Lemma 6.5 shows that $\#bvl_{\overrightarrow{y}}([\sigma'(x'_i)\ \overrightarrow{y{\downarrow_{\beta\overline{\eta}}}}]{\downarrow_{\beta\overline{\eta}}}) = \#bvl(\sigma'(x'_i)) - m$ where m is the length of $\overrightarrow{y}$. It follows that $\#bvl(\sigma(x_i)) = \#bvl(\sigma'(x'_i)) \leq b'(x_i) = b(x_i)$ for $i = 1, \ldots, h$. Hence σ is a Σ_0-unifier of (S, b). If Alternative 3 is selected, the proof is analogous.

Completeness. Let σ be a Σ_0-unifier for (S, b) with exponent of periodicity not exceeding E. For $i = 1, \ldots, h$ let $\sigma(x_i) = \lambda\overrightarrow{y_i}.t_i$ be represented in the form $\lambda\overrightarrow{y_i}.D_i[s^*_i]$ where s^*_i is the minimal subterm of t_i that contains all subterms t' such that $\#bvl_{\overrightarrow{y_i}}(t') \neq 0$ or t' has the form $f(\ldots)$ where f has an argument of non-elementary type. In the case where t_i is a ground first-order term let D_i denote the maximal prefix of $\sigma(C_h)^{E+1}$ such that t_i can be represented in the form $D_i[s^*_i]$ for a ground first-order term s^*_i.

Looking at Alternative 1 it follows from Lemma 6.8 that it suffices to consider the case where any variable $x_i \in \{x_1, \ldots, x_h\}$ has a value $\sigma(x_i)$ of the form $\lambda\overrightarrow{y_i}.f(t_{i,1}, \ldots, t_{i,k_i})$ where for at most one subterm $t_{i,l}$ we have $\#bvl_{\overrightarrow{y_i}}(t_{i,l}) \neq 0$ and every argument of f has elementary type. Hence D_i is non-empty for all $1 \leq i \leq h$. Let D_0 denote the maximal common prefix of all contexts D_i and $\sigma(C_h)^{E+1}$.

If $D_0 = D_j$ for some $j, 1 \leq j \leq h$, then we select Alternative 2. The number e and the prefix $C_{h,1}$ are chosen in such a way that the main depth of D_0 and

$C^e C_{h,1}$ are identical. Reduction rules in (d) use variable x'_j. The selection of the appropriate reduction rule is described below. For $i = 1, \ldots, h$ let $\sigma(x_i) = \lambda\overrightarrow{y_i}.D_0[s_i^{(0)}]$. Define $\sigma'(x'_i) := \lambda\overrightarrow{y_i}.s_i^{(0)}$. It is trivial to verify that σ' is a Σ_0-unifier of the BHOUP (S', b') reached after Step (c). The exponent of periodicity of σ' does not exceed the exponent of periodicity of σ. Moreover, by choice of the D_i, for index j we know $\sigma'(x'_j)$ has one of the types (1)-(3) described in Remark 6.1. Hence, applying (reduce-bv), (reduce-split) or (reduce-binder) we reach the BHOUP (S^*, b^*). As in the proof of Lemma 6.8 (Cases 1,2,3) it follows that (S^*, b^*) has a Σ_0-unifier σ^* such that the exponent of periodicity of σ^* does not exceed the exponent of periodicity of σ'.

In the remaining case, D_0 is a proper prefix of all $D_1, \ldots, D_h$. We first verify that $h \neq 1$. Assume otherwise. Let D_0 be represented in the form $D_1^e D_{1,1}$ where $D_h = D_1 = \sigma(C_1)$, let $D_1 = D_{1,1}D_{1,2}$. Then there must be an index $k \neq \mathit{firstdpos}(D_{1,2}D_{1,1})$ such that $\sigma(x_1) = \lambda\overrightarrow{y_1}.D_1^e D_{1,1}[f(r_1, \ldots, r_k\lfloor s_1^*\rfloor, \ldots, r_n)]$. The equation $x_1\overrightarrow{s_1} \doteq C_1[x_1\overrightarrow{t_1}]$ after applying σ plus beta-reductions has the form

$$D_1^e D_{1,1}[f(r_1, \ldots, r_k\lfloor s_1^{*\prime}\rfloor, \ldots, r_n)] \doteq D_1 D_1^e D_{1,1}[f(r_1, \ldots, r_k\lfloor s_1^{*\prime\prime}\rfloor, \ldots, r_n)].$$

This implies that

$$f(r_1, \ldots, r_k\lfloor s_1^{*\prime}\rfloor, \ldots, r_n) \doteq D_{1,2}D_{1,1}[f(r_1, \ldots, r_k\lfloor s_1^{*\prime\prime}\rfloor, \ldots, r_n)].$$

By assumption we have $D_{1,2}D_{1,1} = f(a_1, \ldots, a_{j-1}, E\lfloor[\cdot]\rfloor, a_{j+1}, \ldots, a_n)$ where $k_1 \neq j$. Then by decomposition we get that r_j has a subterm $f(r_1, \ldots, r_k\lfloor s_1^{*\prime\prime}\rfloor, \ldots, r_n)$ on the surface, which is impossible.

Now assume that $h \geq 2$. Then each value $\sigma(x_i)$ can be represented in the form $\lambda\overrightarrow{y_i}.D_0[f(r_{i,1}, \ldots, r_{i,k_i-1}, r_{i,k_i}\lfloor s_i^*\rfloor, r_{i,k_i+1}, \ldots, r_{i,n})]$ where $k_j \neq \mathit{firstdpos}(D_{1,2}D_{1,1})$ for at least one index k_j. Here f is the topmost function symbol of $D_{1,2}$ which has arity > 1. We select Alternative 3. The choice of e is as in the previous case. The indices k_i in Step (b) are triggered by the above representation. The choice of s_i^* shows that the terms $r_{i,l}$ for $l \neq k_i$ do not contribute to the $\#bvl_{\overrightarrow{y_i}}$-measure. It is now straightforward to see that the BHOUP (S', b') reached after Step (d) has a Σ_0-unifier σ' where the exponent of periodicity does not exceed E. The rest is standard. □

Remark 9.13. With a more detailed analysis it can be shown that the second alternative in the rule (solve-special-cycle) can be simplified in the sense that in Step (d) we only apply (reduce-bv) or (reduce-split). To see this one has to reconsider in the above proof the situation where $D_0 = D_j$ for some $1 \leq j \leq h$. It can be seen, using appropriate sequences of imitation steps, that for at least one index j with $D_0 = D_j$ the value $\sigma_(x_j)$ has the form $\lambda\overrightarrow{y_j}.D_j[s_j^{(0)}]$ where $\lambda\overrightarrow{y_j}.s_j^{(0)}$ has one of the types (1)-(2) described in Remark 6.1. Hence, applying (reduce-bv) and (reduce-split) suffices.

Lemma 9.14. *Every application of the rules (shuffle*), (shuffle**), or (solve-special-cycle) either fails or transforms a decomposed BHOUP (S, b) in $\beta\overline{\eta}$-normal form into a decomposed BHOUP (S', b') in $\beta\overline{\eta}$-normal form. Moreover, $subt(S') \subseteq subt(S)$.*

Proof. By inspecting the rule. □

10 Results and Corollaries

We summarize the decidability results and also describe some improvements and reformulations of the theorems.

10.1 Higher-Order Unification

The main goal of the paper is to prove Theorem 5.23:

Theorem 10.1. *Unifiability of BHOUPs is decidable.*

The decision procedure BHUP is very careful with the origin of used function symbols. If unifiability in a given signature is an issue, then we can specialize the claim:

Theorem 10.2. *Let Σ_0 be a signature that contains at least one constant symbol for every elementary type from T_0. Then unifiability of BHOUPs w.r.t. Σ_0 is decidable.*

Proof. We have to inspect the decision algorithm BHUP, in particular the rules that instantiate variables with terms containing function symbols and/or constants. It can be seen, that given a unifiable BHOUP (S, b), the algorithm BHUP constructs only instantiations for the free variables using symbols from Σ_0. The completeness proof shows that there are only final systems of type "xy". The soundness proofs show that we can then reconstruct a unifier of the input system that uses only Σ_0-symbols. □

Note that the theorem above holds also for BHOUPs containing symbols not in Σ_0.

A further issue is to avoid the condition that unifiers must be in $\beta\overline{\eta}$-normal form. Since the algorithm BHUP is not complete in the sense that it computes all unifiers, only a sufficient condition for non-unifiability can be derived: This can be avoided at the cost of slightly increasing the bounds.

Proposition 10.3. *Let S be a higher-order unification problem and b be a bounding function. Let a be the maximal size of types in $subt(S)$, let b' be the function $x \mapsto 3 * a * b(x)$. Then the following holds: If (S, b') is not unifiable as a BHOUP, then there is no unifier σ of S (as a HOUP)[4] such that for all variables $x \in FV(S) : \#bvl(\sigma(x)) \leq b'(x)$.*

[4] The unifier σ is not forced to be in $\beta\overline{\eta}$-normal form.

Proof. This follows from Theorem 5.23, and Lemma 2.13. □

The other direction does not follow from the results and techniques in this paper, since the algorithm BHUP is not designed to compute a complete set of unifiers of a BHOUP.

Remark 10.4. Whether it is possible to use a binding function that only refers to the number of bound variables (in contrast to the sum of the number of bound variables and the number of lambdas) is an open question. One obstacle is the construction in Lemma 4.4, which constructs first-order contexts. It is not obvious how to generalize this construction to a measure ignoring the number of lambdas, since a context of the form $\lambda x_1.f_1(\lambda x_2.f_2(\ldots\ (\lambda x_n.f_n(\ldots[\cdot])))))$ may be constructed. However, this context is not a first-order context, and moreover, it may be destroyed during reduction of terms $\sigma(s)$ to their normal form.

10.2 Higher-Order Matching

Currently, it is not known whether higher-order matching is decidable, however there is some knowledge about decidability and complexity of special cases [Wol93,Dow92,CJ97,Wie99,Pad00]. The techniques in this paper permit to show that a variant of higher-order matching with a bound on the instantiation is decidable:

Let S be a HOUP in $\beta\overline{\eta}$-normal form, such that in every equation $s \doteq t$ in S, the right hand side t has no occurrences of free variables. Then S is called a higher-order matching problem. Let b be a function from free variables to $\mathbb{N}$. Then (S, b) is called a bounded higher-order matching problem (BHOMP). A substitution σ in $\beta\overline{\eta}$-normal form is a solution of a (BHOMP), iff σ is a unifier of S, and furthermore, for every free variable x in S, the number of bound variables in $\sigma(x)$ is not greater than $b(x)$.

Theorem 10.5. *Bounded higher-order matching is decidable*

Proof. Using a similar technique as in the proof of soundness and completeness of (constantify) (see 5.12), it is easy to prove that in a minimal unifier σ the number of occurrences of function symbols is not greater than the number of occurrences of function symbols in the right hand side of S. Lemma 4.1 shows that the types of subterms of terms in the codomain of σ are already in $subt(S)$. Hence, lambda-prefixes in the codomain are bounded by the maximal arity of types in $subt(S)$. Since codomain terms are in $\beta\overline{\eta}$- normal form, we conclude that the following holds: there is a constant $c(S)$, such that $size(\sigma(x)) \leq c(S) * b(x)$. In summary, decidability follows, since it is only necessary to test a finite number of potential unifiers, which are effectively enumerable. □

10.3 Bounded Second-Order Unification

The results in this paper are a generalization of the decidability result for bounded second-order unification [SS99a,SS01]. The specializations for second-order are:

- There is exactly one elementary type ι.
- All function symbols in the signature have type of the form $\iota \to \ldots \to \iota$.
- In unification problems, every type of subterms is either ι or a function type of the form $\iota \to \ldots \to \iota$. In particular, every free variable has type ι or $\iota \to \ldots \to \iota$, which corresponds to the distinction between first-order variables and second-order variables. The only subterms of function type are the second-order variables. It follows also that every bound variable has type ι.

It is now easy to see that second-order unifiers may either instantiate variables by a ground first-order term, or by a term with a lambda-prefix and a first-order term as body.

The bound on the number of bound variables in the codomain terms can easily be translated into an equivalent bound for a higher-order unification problem. We obtain as corollary of Theorem 5.23.

Corollary 10.6. *Bounded second-order unification is decidable.*

References

[And86] Peter Andrews. *An introduction to mathematical logic and type theory: to truth through proof.* Academic Press, 1986.

[And01] Peter Andrews. Classical type theory. In Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, volume 2, chapter 15, pages 965–1007. North-Holland, 2001.

[Bar84] Henk P. Barendregt. *The Lambda Calculus. Its Syntax and Semantics.* North-Holland, Amsterdam, New York, 1984.

[Bar90] Henk P. Barendregt. Functional programming and lambda calculus. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science: Formal Models and Semantics*, volume B, chapter 7, pages 321–363. Elsevier, 1990.

[Bec01] Arnold Beckmann. Exact bounds for lengths of reductions in typed λ-calculus. *J. Symbolic Logic*, 66:1277–1285, 2001.

[Bir98] Richard Bird. *Introduction to Functional Programming using Haskell.* Prentice Hall, 1998.

[BMS80] R. Burstall, D. MacQueen, and D.T. Sanella. Hope: an experimental applicative language. In *Proc. LISP Conference*, pages 1363–143, 1980.

[BN98] Franz Baader and Tobias Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.

[BS94] Franz Baader and Jörg Siekmann. Unification theory. In D.M. Gabbay, C.J. Hogger, and J.A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, pages 41–125. Oxford University Press, 1994.

[CJ97] Hubert Comon and Yan Jurski. Higher-order matching and tree automata. In *Proc. of CSL 97*, volume 1414 of *Lecture Notes in Computer Science*, pages 157–176, 1997.

[Com98] Hubert Comon. Completion of rewrite systems with membership constraints. Part I: Deduction rules. *J. of Symbolic Computation*, 25(4):397–419, 1998.

[CP97] I. Cervesato and Frank Pfenning. Linear higher-order pre-unification. In *Proc. 12th LICS*, pages 422–433, 1997.

[DJ90] Nachum Dershowitz and Jean-Pierre Jouannaud. Rewrite systems. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science: Formal Models and Semantics*, volume B, chapter 6, pages 243–320. Elsevier, 1990.

[DM79] Nachum Dershowitz and Z. Manna. Proving termination with multiset orderings. *Communications of the ACM*, 22:465–476, 1979.

[Dow92] Gilles Dowek. Third order matching is decidable. In *Proceedings of the 7th Annual IEEE Symposium on Logic in Computer Science*, pages 2–10, 1992.

[Dow01] Gilles Dowek. Higher-order unification and matching. In Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, volume 2, chapter 16, pages 1009–1062. North-Holland, 2001.

[Far88] W.A. Farmer. A unification algorithm for second order monadic terms. *Annals of Pure and Applied Logic*, 39:131–174, 1988.

[Far91] W.A. Farmer. Simple second-order languages for which unification is undecidable. *J. Theoretical Computer Science*, 87:173–214, 1991.

[Gan80] Robin O. Gandy. Proofs of strong normalization. In J.P. Seldin and J.R. Hindley, editors, *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 457–477. Academic Press, 1980.

[GLM97] J. Goubault-Larrecq and I. Mackie. *Proof Theory and Automated Deduction*, volume 6 of *Applied Logic Series*. Kluwer, may 1997. ISBN 0-7923-4593-2.

[Gol81] Warren. D. Goldfarb. The undecidability of the second-order unification problem. *Theoretical Computer Science*, 13:225–230, 1981.

[Gut98] C. Gutierrez. Satisfiability of word equations with constants is in exponential space. In *Proceedings FOCS'98*, pages 112–119, Palo Alto, California, 1998. IEEE Computer Society Press.

[Hin97] J.Roger Hindley. *Basic simple type theory*. Cambridge tracts in theoretical computer science. Cambridge University Press, 1997.

[HKMN95] M. Hanus, H. Kuchen, and J.J. Moreno-Navarro. Curry: A truly functional logic language. In *Proc. ILPS'95 Workshop on Visions for the Future of Logic Programming*, pages 95–107, 1995.

[HS86] J.Roger Hindley and Jonathan P. Seldin. *Introduction to combinators and λ-calculus*. Cambridge University Press, 1986.

[Hue75] Gérard Huet. A unification algorithm for typed λ-calculus. *Theoretical Computer Science*, 1:27–57, 1975.

[Hue76] Gérard Huet. *Résolution d'équations dans des langages d'ordre 1,2,...ω*. Thèse de doctorat d'etat, Université Paris VII, 1976. In French.

[JP76] D. Jensen and Tomasz Pietrzykowski. Meachanizing ω-order type theory through unification. *Theoretical Computer Science*, 3(2):123–171, 1976.

[Klo92] Jan Willem Klop. Term rewriting systems. In S. Abramsky, D.M. Gabbay, and T.S.E.Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2, pages 2–116. Oxford University Press, 1992.

[KP96] Antoni Kościelski and Leszek Pacholski. Complexity of Makanin's algorithms. *Journal of the Association for Computing Machinery*, 43:670–684, 1996.

[Lev96] Jordi Levy. Linear second order unification. In *Proceedings of the 7th International Conference on Rewriting Techniques and Applications*, volume 1103 of *Lecture Notes in Computer Science*, pages 332–346, 1996.

[LV00a] Jordi Levy and Margus Veanes. On the undecidability of second-order unification. *Information and Computation*, 159:125–150, 2000.

[LV00b] Jordi Levy and Mateu Villaret. Linear second-order unification and context unification with tree-regular constraints. In *Proceedings of the 11th Int. Conf. on Rewriting Techniques and Applications*, volume 1833 of *Lecture Notes in Computer Science*, pages 156–171, 2000.

[Mak77] G.S. Makanin. The problem of solvability of equations in a free semigroup. *Math. USSR Sbornik*, 32(2):129–198, 1977.

[Mil91] Dale Miller. A logic programming language with lambda-abstraction, function variables and simple unification. *J. of Logic and Computation*, 1(4):497–536, 1991.

[Nar90] Paliath Narendran. Some remarks on second order unification. Technical report, Inst. of Programming and Logics, Department of Computer Science, Univ. of NY at Albany, 1990.

[Nip91] Tobias Nipkow. Higher-order critical pairs. In *Proc. 6th IEEE Symp. LICS*, pages 342–349, 1991.

[NPR97] Joachim Niehren, Manfred Pinkal, and Peter Ruhrberg. On equality up-to constraints over finite trees, context unification, and one-step rewriting. In *Proceedings of the International Conference on Automated Deduction*, volume 1249 of *Lecture Notes in Computer Science*, pages 34–48, 1997.

[NTT00] Joachim Niehren, Sophie Tison, and Ralf Treinen. On rewrite constraints and context unification. *Information Processing Letters*, 74:35–40, 2000.

[Pad00] Vincent Padovani. Decidability of fourth-order matching. *Mathematical Structures in Computer Science*, 10(3):361–372, 2000.

[Pau91] Lawrence C. Paulson. *ML for the working programmer*. Cambridge University Press, 1991.

[Pau94] Lawrence C. Paulson. *Isabelle*, volume 828 of *Lecture Notes in Computer Science*. Springer-Verlag, 1994.

[Pfe01] Frank Pfenning. Logical frameworks. In Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, volume 2, chapter 17, pages 1063–1147. North-Holland, 2001.

[Pla99] Wojciech Plandowski. Satisfiability of word equations with constants is in PSPACE. In *FOCS 99*, pages 495–500, 1999.

[Sch82] Helmut Schwichtenberg. Complexity of normalization in the pure typed λ-calculus. In A.S. Troelstra and D. van Dalen, editors, *The L.E.J. Brouwer Centenary Symposium. Proceedings of the Conference hold in Noordwijkerhout, 8–13 June, 1981*, volume 110 of *Studies in Logic and the Foundations of Mathematics*, pages 453–458. North Holland, 1982.

[Sch90] Klaus U. Schulz. Makanin's algorithm - two improvements and a generalization. In *Proc. of IWWERT 1990*, volume 572 of *Lecture Notes in Computer Science*, pages 85–150. Springer-Verlag, 1990.

[Sch91] Helmut Schwichtenberg. An upper bound for reduction sequences in the typed λ-calculus. *Archive for Mathematical Logic*, 30:405–408, 1991. Dedicated to Kurt Schütte on the occasion of his 80th birthday.

[Sch93] Klaus U. Schulz. Word unification and transformation of generalized equations. *J. Automated Reasoning*, pages 149–184, 1993.

[SS94] Manfred Schmidt-Schauß. Unification of stratified second-order terms. Internal Report 12/94, Fachbereich Informatik, J.W. Goethe-Universität Frankfurt, Frankfurt, Germany, 1994.

[SS99a] Manfred Schmidt-Schauß. Decidability of bounded second order unification. Frank report 11, FB Informatik, J.W. Goethe-Universität Frankfurt am Main, 1999. available at `http://www.ki.informatik.uni-frankfurt.de/papers/articles.html`.

[SS99b] Manfred Schmidt-Schauß. A decision algorithm for stratified context unification. Frank-Report 12, Fachbereich Informatik, J.W. Goethe-Universität Frankfurt, Frankfurt, Germany, 1999. accepted for publication in J. Logic and Computation, available at `http://www.ki.informatik.uni-frankfurt.de/papers/articles.html`.

[SS01] Manfred Schmidt-Schauß. Decidability of bounded second order unification, 2001. submitted for publication.

[SSS98] Manfred Schmidt-Schauß and Klaus U. Schulz. On the exponent of periodicity of minimal solutions of context equations. In *Proceedings of the 9th Int. Conf. on Rewriting Techniques and Applications*, volume 1379 of *Lecture Notes in Computer Science*, pages 61–75, 1998.

[SSS00] Manfred Schmidt-Schauß and Klaus U. Schulz. Solvability of context equations with two context variables is decidable. *Journal of Symbolic Computation*, 2000. accepted for publication.

[Sta79] Richard Statman. The typed λ-calculus is not elementary recursive. *Theoretical Computer Science*, 9:73–81, 1979.

[Tur85] D. A. Turner. Miranda: A non-strict functional language with polymorphic types. In *Functional Programming Languages and Computer Architecture*, number 201 in Lecture Notes in Computer Science, pages 1–16. Springer, 1985.

[Vor98] Sergei Vorobyov. $\forall\exists^*$-equational theory of context unification is Π_1^0-hard. In *MFCS 1998*, volume 1450 of *Lecture Notes in Computer Science*, pages 597–606. Springer-Verlag, 1998.

[Wie99] Tomasz Wierzbicki. Complexity of the higher-order matching. In *Proc.* 16^{th} *CADE*, Lecture Notes in Computer Science, pages 82–96. Springer-Verlag, 1999.

[Wol93] David A. Wolfram. *The clausal theories of types*. Number 21 in Cambridge tracts in theoretical computer science. Cambridge University Press, 1993.

[Zhe79] A.P Zhezherun. Decidability of the unification problem for second order languages with unary function symbols. *Kibernetika (Kiev)*, 5:120–125, 1979. Translated as *Cybernetics* 15(5): 735-741,1980.