INFORMATIK

$\forall\exists^*$-Equational Theory of Context Unification is Π_1^0-Hard

Sergei Vorobyov

MPI–I–98–2–008 April 1998

FORSCHUNGSBERICHT RESEARCH REPORT

MAX-PLANCK-INSTITUT
FÜR
INFORMATIK

Im Stadtwald 66123 Saarbrücken Germany

Author's Address

Sergei Vorobyov: Max-Planck Institut für Informatik, Im Stadtwald, D-66123, Saarbrücken, Germany, sv@mpi-sb.mpg.de, http://www.mpi-sb.mpg.de/~sv.

Publication Notes

Publication date: April 20, 1998

Acknowledgements

The author is grateful to Margus Veanes for suggesting improvements to an earlier draft of this paper.

Abstract

Context unification is a particular case of second-order unification, where all second-order variables are *unary* and only *linear* functions are sought for as solutions. Its decidability is an open problem. We present the simplest (currently known) undecidable quantified fragment of the theory of context unification by showing that for every signature containing a ≥ 2-ary symbol one can construct a *context equation* $\mathcal{E}\,[p, r, \overline{F}, \overline{w}]$ with parameter p, first-order variables r, $\overline{w}$, and context variables $\overline{F}$ such that the set of true sentences of the form

$$\forall r\ \exists\, \overline{F}\ \exists\, \overline{w}\ \mathcal{E}[p, r, \overline{F}, \overline{w}]$$

is Π^0_1-hard (i.e., every co-r.e. set is many-one reducible to it), as p ranges over finite words of a binary alphabet.

Moreover, the existential prefix above contains just 5 context and 3 first-order variables (this can be further improved).

It follows, in particular, that the $\forall\exists^8$-equational theory (without $\wedge$, $\vee$, $\neg$) of context unification is undecidable.

Keywords

Context unification, theory of free semigroup, undecidability, co-r.e. complete sets.

Contents

1 Introduction

The *Context Unification Problem* (CUP for short) is:

- A generalization of the celebrated Markov-Löb's problem of solvability of equations in a free semigroup proved decidable by (Makanin 1977); CUP coincides with the latter problem for monadic signatures.

- A specialization of the *Second-Order Unification* (SOU), known to be undecidable (Goldfarb 1981, Farmer 1991). CUP is almost SOU, with *only unary function variables* allowed and solutions required to be *linear*, i.e., of the form $\lambda x.t(x)$, where $t(x)$ contains *exactly one occurrence* of x.

Context unification is useful in different areas of Computer Science: term rewriting, automated theorem proving, equational unification, constraint solving, computational linguistics, software engineering (Schmidt-Schauß 1994, Niehren, Pinkal & Ruhrberg 1997, Schmidt-Schauß & Schulz 1997).
CUP is stated as follows:

> *Given a pair of terms t, t' built as usual from symbols of a signature Σ, first-order variables $\overline{w}$, and unary function variables $\overline{F}$, does there exist an assignment θ of terms to $\overline{w}$ and* linear *second-order functions to $\overline{F}$ such that $\theta(t) = \theta(t')$?*

Thus, CUP is a decidability problem for the existentially quantified equations ($\exists^*$-equational theory) of the form

$$\exists\, \overline{F}\, \exists\, \overline{w}\; t = t', \tag{1}$$

where the quantified context variables $\overline{F}$ range over *linear* functions.

Currently the decidability of CUP is an open problem (Schmidt-Schauß 1994, Niehren et al. 1997, Schmidt-Schauß & Schulz 1997). Most researchers conjecture and hope that CUP is decidable. All the above papers provide some approximations: either prove decidability of particular cases, or settle undecidability of some generalizations, or provide technical results towards decidability of CUP.

Presumably, CUP is very hard to settle, both in decidable and undecidable sense. This is because CUP lies between a technically difficult decidable case of equations in free semigroups (Markov-Löb's problem proved decidable by (Makanin 1977)), and the undecidable case of SOU settled by (Goldfarb 1981, Farmer 1991). Farmer's result is also technically quite difficult.

SOU is undecidable for second-order languages containing at least one $\geq$ 2-ary function constant and finitely many *unary* and *ternary* function variables (Goldfarb 1981). Later (Farmer 1991) improved it by showing that SOU remains undecidable in presence of *unary function variables only* (but, unlike CUP, substitutions looked for are *not required to be linear*; in fact, they are not linear in Farmer's proof). It follows from (Makanin 1977) that SOU is decidable when all variable and constant function symbols are unary. (Farmer 1988) improved it by showing that decidability is preserved if n-ary function variables are allowed in addition to constant function symbols of arity *at most one*.

Thus, CUP represents the only unknown remaining difficult intermediate case between decidable word equations and undecidable SOU (unary variables, n-ary constants, linear solutions). This explains why the progress on CUP has been quite slow. Indeed, decidability of CUP would considerably improve Makanin's result, whereas undecidability would considerably improve Goldfarb-Farmer's undecidability of SOU.

In this paper we show that adding just one outermost universal quantifier to a context equation (1) leads to the Π^0_1-hard class of formulas, where Π^0_1 is the class of all co-recursively enumerable sets.

For comparison, the following undecidability results are known about quantified fragments of context unification:

1. (Quine 1946) showed that the full first-order theory of free semigroups[1] is undecidable.

2. (Durnev 1973) improved it to undecidability of $\exists\forall\exists^3$-positive (without negation, but with $\wedge$ and $\vee$) theory of free semigroups.

3. (Marchenkov 1982) improved it to undecidability of $\forall\exists^4$-positive theory of free semigroups.

4. (Durnev 1997) improved it to undecidability of $\forall\exists^3$-positive theory of free semigroups.

5. (Niehren et al. 1997) claimed undecidability of the $\exists^*\forall^*\exists^*$-theory of context unification.

It should be noticed that all known methods to transform a positive formula of the theory of free semigroups into just one equation require a considerable number of *auxiliary existentially quantified variables*, (Büchi & Senger 1986), depending on the number of disjunctions involved. Thus

[1]This corresponds to context unification in unary signatures

the above undecidability results for $\forall\exists^4$- and $\forall\exists^3$-positive theories of free semigroups yield only undecidability of the $\forall\exists^n$-equational theories of free semigroups with a *very large* number n of existentially quantified variables.

In this paper we show that the situation with context equations is quite different, and *just two* extra existentially quantified context variables suffice to eliminate all disjunctions. This, together with a direct reduction from the halting problem for Turing machines, gives the undecidability of the $\forall\exists^8$-equational theory of context unification, with a reasonably simple quantifier prefix.

The main result of this paper may now be stated as follows.

Main Theorem. *For every signature containing a ≥ 2-ary symbol one can construct a* context equation $\mathcal{E}\,[p, r, C, F, F', G, H, x, y, z]$ *with parameter p, first-order variables r, x, y, z, and context variables C, F, F', G, H such that the set of true sentences of the form*

$$\forall r \; \exists\, C, F, F', G, H \; \exists\, x, y, z \; \mathcal{E}\,[p, r, C, F, F', G, H, x, y, z] \qquad (2)$$

is Π^0_1-hard (i.e., every co-r.e. set is many-one reducible to it), as p ranges over finite words of a binary alphabet. $\square$

It follows, in particular, that the $\forall\exists^8$-equational theory (without $\wedge$, $\vee$, $\neg$) of context unification is *undecidable.*

Warning. We would like to stress that in this paper we do not intend to improve the undecidability results of (Marchenkov 1982, Durnev 1997) for $\forall\exists^4$- and $\forall\exists^3$-*positive theories of free semigroups* as to *minimizing* the number of existential quantifiers for *positive theories.* However, we do get an improvement over (Marchenkov 1982, Durnev 1997) (of course in a different framework of context unification) as to simplicity of the existential prefix in the undecidable $\forall\exists^*$-*equational theory* of context unification. As we mentioned above, all known methods of eliminating disjunctions from positive formulas in free semigroups use a considerable number of auxiliary existentially quantified variables, proportional to the number of disjunctions to be eliminated. We show that in context unification just two extra variables are enough.

We would also like to stress that by no means the quantifier prefix we obtain is minimal. We rather try to keep proofs intuitive and transparent. We believe that applying the methods of disjunction elimination with a constant number of extra variables (described in Section 5) directly to the reductions of (Marchenkov 1982, Durnev 1997) will yield the undecidability of the $\forall\exists^5$-equational theory of context unification.

Outline. The remainder of this paper is organized as follows. In Section 2 we give necessary definitions concerning CUP, Turing machines, Π^0_1-sets. In Section 3 we describe the main construction of an $\forall\exists^*$-sentence from a deterministic TM with a complete Σ^0_1 domain, and in Section 4 implement it. Section 5 is devoted to eliminating disjunctions.

2 Preliminaries

2.1 Context Unification.

Let Σ be a fixed finite signature with each symbol assigned a fixed arity, *containing at least one constant.* Let X be a set of variables and $\mathcal{F} = \{F, G, \ldots\}$ be an infinite set of *function variables of arity one*, also called *context variables.*

Definition 2.1 (Terms) The set $\mathcal{T}(\Sigma, X)$ of *terms* of signature Σ with variables from X is defined as usual: variables from X are terms and for $f \in \Sigma$ of arity $n \geq 0$ an expression $f(t_1, \ldots, t_n)$ is a term whenever t_i's are terms. $\Box$

Writing terms with unary function symbols we usually drop parentheses to improve readability.

We assume all the standard definitions and conventions concerning λ-notation, like β-reduction, normalization, etc.

Definition 2.2 (Contexts) A *context* is an expression of the form $\lambda x.t(x)$, where $t(x) \in \mathcal{T}(\Sigma, \{x\})$ contains *exactly one* occurrence of the variable x. A context with β-normal form $\lambda x.x$ is called *empty.* All other contexts are *non-empty.* $\Box$

Remark. Note that the "*exactly one*" requirement, absent from the definition of SOU, is a specialty of CUP. $\Box$

Definition 2.3 (Context Terms) *Context terms* are defined inductively: if $\phi \in \Sigma \cup \mathcal{F}$ is of arity $n \geq 0$ and $t_1, \ldots, t_n$ are context terms, then $\phi(t_1, \ldots, t_n)$ is a context term. $\Box$

Writing context terms with unary function symbols and function variables we usually drop parentheses to improve readability.

Definition 2.4 (CUP, Context Equations) An instance of CUP, also called a *context equation* (CE for short), is an expression $\tau_1 \stackrel{?}{=} \tau_2$, where $\tau_{1,2}$ are context terms. A *solution* to a CE $\tau_1 \stackrel{?}{=} \tau_2$ is a substitution θ of contexts for context variables of $\tau_{1,2}$ such that $\theta(\tau_1) \equiv_\beta \theta(\tau_2)$ (equality modulo the usual β-reduction). $\Box$

Remark. Note that the requirement of replacement of functional (context) variables with *contexts*, as opposed to arbitrary second-order λ-terms, differs CUP from SOU. When this requirement is dropped, the problem becomes *undecidable* (Farmer 1991). $\Box$

2.2 Turing Machine with Complete R.E. Domain

A Σ_1^0-set is a recursively enumerable set. A Π_1^0-set is a complement of a Σ_1^0-set. An Σ_1^0-set (resp., Π_1^0-set) A is called *complete* iff every Σ_1^0-set (resp., Π_1^0-set) B is *many-one reducible* to A, i.e., there exists a total recursive function f such that $x \in B \Leftrightarrow f(x) \in A$.

It is well known that there exists a DTM M whose domain is a complete Σ_1^0-set, and, therefore, the set of elements it does not accept is a complete Π_1^0-set. We may assume, without loss of generality, that the tape alphabet B of M consists of two symbols, 0 and 1, the tape of M is infinite to the right, M never tries to move left from its leftmost tape cell, M may only extend its tape by writing new symbols on the right end, that the states of M are $Q = \{q_0, \ldots, q_f\}$, q_0 is the unique initial state, q_f is the unique final state of M, M always starts by observing its leftmost tape cell, and M immediately stops entering state q_f. Further we assume that M is a fixed such DTM.

The DTM M applies to a nonempty word $b_1 \ldots b_m \in B^+$ iff there exists a finite sequence of IDs (instantaneous descriptions) of M

$$id_0, \ldots, id_F \tag{3}$$

such that $id_0 \equiv q_0 b_1 \ldots b_m$, $id_F \equiv \gamma q_f \delta$ for some $\gamma, \delta \in B^*$, and such that for every pair of IDs id_i, id_{i+1} in the sequence id_{i+1} is obtained from id_i by application of some command of M. The corresponding definitions are well-known and we omit them here.

We will represent IDs of M by terms constructed from unary function symbols, starting from the constant ε for the empty word. We have a unary function symbol for every symbol in $B \cup Q$, and represent a word $q_0 10101010$ as a term $q_0(1(0(1(0(1(0(1(0(\varepsilon)))))))))$, which for simplicity will always be written as $q_0 10101010$ (i.e., with ( ) and ε omitted), if it does not lead to ambiguity.

We will represent a sequence (3) as a right-flattened term (list)

$$f(id_0, f(id_1, f \ldots, f(id_{F-1}, f(id_F, \varepsilon)) \ldots)), \tag{4}$$

where id_i's are term representations of words in (3) using unary functional symbols as described above. Thus the DTM M applies to an input if and only if such a term (4) exists, and does not apply, if and only if it does not exist.

Formally, we will use the following signature Σ:

1. ε - constant, for the empty word and list,
2. 0, 1 - unary for tape alphabet B,

3. $q_0, \ldots, q_f$ - unary for M's states in Q,
4. f - binary list constructor,
5. a - unary, auxiliary.

Notational Conventions. If not stated otherwise, below

1. s denotes a symbol from $B \cup Q$,
2. b denotes a symbol from B,
3. q denotes a symbol from Q.

3 Sentence Expressing Inapplicability

To prove the main claim of this paper we will write a sentence of the following form (with context variables C, F, F', G, H, and ordinary variables r, x, y, z):

$$\forall r \; \exists C, F, F' \; \exists x, y, z \; \exists G, H \; \Phi[f(q_0 b_{i_1} \ldots b_{i_p}, r), t_1, \ldots, t_m], \tag{5}$$

which expresses the fact that the DTM M *does not apply* to the input string $\bar{s} \equiv b_{i_1} \ldots b_{i_p}$, because for every r, the term $t \equiv f(q_0 b_{i_1} \ldots b_{i_p}, r)$ *is not* a correct run (4) of M, i.e., $\bar{s} \equiv b_{i_1} \ldots b_{i_p}$ is in the complement of the r.e. complete domain of M. This implies the main claim of the paper.

Technically, expressing that a term $t \equiv f(q_0 b_{i_1} \ldots b_{i_p}, r)$ *is not* a correct run amounts to saying that 'something goes wrong' in t. For example, t contains 'senseless' subterms like $f(f(\ldots, \ldots), \ldots)$, or $a(\ldots)$, etc. It turns out that this can be expressed by saying 'contains one of the *finitely* many wrong patterns' $t_1, \ldots, t_n$, which can be done by saying

$$t = Ct_1 \vee \ldots \vee t = Ct_m,$$

where C is a context variable used to say 'there exists a subterm of t. All the patterns $t_1, \ldots, t_n$ are expressed by using 2 context variables F, F' and 3 first-order variables x, y, z. We then show how to get rid of disjunction by using just two extra existentially quantified context variables (G and H in (5)), independently of the number m of patterns. Thus just one context equation with 5 context and 3 first-order existentially quantified variables as in (2), (5) suffices for undecidability.

In the next section we enumerate all forbidden patterns preventing a term to be a correct run. In Section 5 we show how to get rid of disjunctions.

4 Forbidden Patterns

4.1 Forbidden Structural Patterns

A term cannot represent a correct run of M if it contains one of the following subterms (for some context F and some terms x, y, z):

1. $f(F(a(x)))$.

   Reason: a is an auxiliary symbol and cannot occur in a run[2].

2. $f(f(x, y), z)$.

   Reason: a run is a right-flattened list of IDs, f cannot occur in the first argument position to itself.

3. $f(x, g(y))$

   for every function symbol $g \in B \cup Q$.

   Reason: a run is a right-flattened list of IDs.

4. $q(F(q'(x)))$

   for $q, q' \in Q$.

   Reason: an ID may contain at most one state symbol.

5. $g(f(x, y))$

   for $g \in B \cup Q$.

   Reason: IDs cannot contain f.

6. $f(F(q_f(x)), f(y, z))$

   Reason: entering q_f DTM stops.

7. $q(s(x))$

   for all $q \in Q \backslash \{q_f\}$ and $s \in B$ such that M has no commands $(q, s \to \ldots)$.

8. $q(\varepsilon)$

   for all $q \in Q \backslash \{q_f\}$ such that there are no commands to extend the tape on the right end in state q.

[2]For technical reasons we use this pattern instead of a more simple $a(x)$, see, Section 5.5.

4.2 Patterns for Incorrect ID Transitions

In this section we enumerate patterns that cannot occur in a correct run of the DTM M for the reason a term matching one of the patterns contains something 'senseless'. This can be expressed by existence of solutions to a *finite* number of 'local' context equations. As an easy and representative example, note that an ID α cannot be transformed into an ID β, if for some context F, state $q \in Q$, tape symbols $s_{1,2,3,4,5} \in B$, and x, y one has simultaneously $\alpha \equiv Fs_1qs_2x$ and $\beta \equiv Fs_3s_4s_4y$. This is because each TM's transition moves head either left or right, so in a correct transition either s_3 or s_5 should belong to Q (be a state). This illustrates the main idea. By routinely enumerating all possibilities we can assure that a transition is correct iff it does not match any of the patterns enlisted. Equivalently, a transition is incorrect iff it matches one of the patterns.

The reader is invited to check that all these patterns use only three first-order variables x, y, z, and two context variables F, F'. Recall that we use these patterns P_i disjunctively in

$$t = CP_1 \vee \ldots \vee t = CP_m$$

to say that t contains a subterm matching one of the patterns $P_1, \ldots, P_m$ (where the context variable C is used to say "there exists a subterm").

Since $\exists$ distributes over $\vee$, the variables may be reused and we need just three context C, F, F', and three first-order x, y, z variables. The two extra context variables G, H will be needed to transform a disjunction into an equation, see Section 5.

List of the patterns for incorrect transitions.

$f(Fs, f(Fs'y, z))$

for $s, s' \in B$, $s \neq s'$.

Reason: in two succeeding IDs the leftmost position in which they differ should necessarily contain a state from Q.

For example, for a right shift command $(q, s \to q', s', R)$ we have, in a correct ID transition:

$$\begin{array}{rcccccccc} ID_i & = & s_1 & \ldots & s_n & q & s & s_{n+1} & \ldots & s_{n+m} \\ ID_{i+1} & = & s_1 & \ldots & s_n & s' & q' & s_{n+1} & \ldots & s_{n+m} \end{array}$$

(the first disagreement is $q - s'$.)

For a left shift command $(q, s \rightarrow q', s', L)$ we have, in a correct ID transition:

$$\begin{array}{rclllllllll} ID_i & = & s_1 & \ldots & s_{n-1} & s_n & q & s & s_{n+1} & \ldots & s_{n+m} \\ ID_{i+1} & = & s_1 & \ldots & s_{n-1} & q' & s_n & s' & s_{n+1} & \ldots & s_{n+m} \end{array}$$

(the first disagreement is $s_n - q'$.)

For an 'extension on the right' command $(q, \varepsilon \rightarrow q', s')$ we have, in a correct ID transition:

$$\begin{array}{rcllllll} ID_i & = & s_1 & \ldots & s_n & q & \\ ID_{i+1} & = & s_1 & \ldots & s_n & q' & s' \end{array}$$

(the first disagreement is $q - q'$.)

$f(Fsx, f(F(\varepsilon), z))$

for all $s \in B \cup Q$.

Reason: the ID_{i+1} cannot be shorter than ID_i.

$f(qs_1x, f(s_2s_3y, z))$

for all $s_3 \in B$.

Reason: if the first symbol of the ID_i is a state, then the second symbol in the ID_{i+1} should be a state.

$f(Fs_1qs_2x, f(Fs_3s_4s_5y, z))$

for all $s_3, s_5 \in B$.

Reason: if the k-th symbol in ID_i is a state then either $k+1$-th or $k-1$-th in ID_{i+1} should be a state.

$f(Fqsx, f(Fs_1s_2y, z))$

for every command $(q, s \rightarrow q', s', R)$ and every pair of symbols $s_1, s_2 \in B \cup Q$ such that either $s_1 \neq s'$ or $s_2 \neq q'$.

Reason: an ID $Fqsx$ should yield $Fs'q'x$, thus each pair of consecutive IDs $Fqsx$, Fs_1s_2y is incorrect.

$f(Fqsx, f(Fs_1(\varepsilon), z))$

for every $s_1 \in B \cup Q$.

Reason: ID_{i+1} abruptly ends. Similar patterns must be written for the case of left shift commands below.

$f(Fbqsx, f(Fs_1s_2s_3y, z))$

for every command $(q, s \to q', s', L)$, every $b \in B$, and every triple of symbols $s_1, s_2, s_3 \in B \cup Q$ such that either $s_1 \neq q'$, or $s_2 \neq b$, or $s_3 \neq s'$.

Reason: an ID $Fbqsx$ should yield $Fq'bs'x$, thus each pair of consecutive IDs $Fbqsx$, $Fs_1s_2s_3y$ is incorrect.

... Similar patterns should be spelled out for the commands extending the tape on the right. We leave it as a straightforward exercise.

$f(FqsF's_1x, f(Fs'q'F's_2y, z))$

for every command $(q, s \to q', s', R)$ and every pair of different symbols $s_1, s_2 \in B$.

Reason: an ID transformation is 'almost' correct, but some symbol to the right of the head is copied erroneously.

... Similar pattern for an 'almost correct' left shift command We leave as a straightforward exercise.

Let us add a brief explanation of how the above patterns work. In a correct run no ID can contain two states (recall that we have a pattern $q(F(q'(x)))$. But can a correct run contain and ID with no state symbols at all? Let us show that this is impossible. For, if such a situation would be possible, there should existed a pair of neighboring IDs with the least index i such that id_i contains a state symbol and id_{i+1} does not. Then it is easy to see that one of the patterns responsible for the transition correctness should match, thus guaranteeing that a run candidate is incorrect.

We thus constructed a finite number of patterns $t_1, \ldots, t_m$ such that the DTM M *does not apply* to the word $\bar{s} = b_{i_1} \ldots b_{i_p}$ if and only if

$$\forall r\ \exists C, F, F'\ \exists x, y, z\ \Big(f(q_0 b_{i_1} \ldots b_{i_p}, r) = Ct_1 \vee \ldots \vee f(q_0 b_{i_1} \ldots b_{i_p}, r) = Ct_m\Big).$$

This already proves that the positive $\forall\exists^6$-theory of context unification is Π^0_1-hard (by the choice of M with the complete r.e. or Σ^0_1- set). In the next section we proceed to eliminating disjunctions from the latter sentence.

5 Trading Disjunctions for Equations

To finish the proof of the main claim of the paper, in this section we show how to represent a disjunction of context equations of the form

$$t = t_1 \vee \ldots \vee t = t_m$$

by *just one* context equation.

5.1 Expressing Conjunction of Context Equations by One Equation

In presence of function symbols of arity ≥ 2, a conjunction of context equations can be easily expressed by just one equation, since $s = s' \wedge t = t'$ if and only if $f(s,t) = f(s',t')$. One can similarly deal with many equations and symbols of arity greater than 2. [3]

5.2 Expressing $t = t_1 \vee t = t_2$ by one equation

Given a disjunction of context equations

$$t = t_1 \vee t = t_2, \tag{6}$$

we would like to transform it to just one equivalent context equation

$$s = s', \tag{7}$$

containing extra variables $\overline{W}$, such that

$$(t = t_1 \vee t = t_2) \quad \Leftrightarrow \quad \exists \overline{W} s = s' \tag{8}$$

(implicitly universally quantified) is true.

We will write (7) as conjunction (see Section 5.1) of equations

$$\begin{aligned} F(a(H(a(\varepsilon)))) &= f(a(f(a(\varepsilon), \varepsilon)), a(f(\varepsilon, a(\varepsilon)))), && (9)\\ S(t) &= f(t_1, t_2), && (10)\\ S(\varepsilon) &= H(z) && (11) \end{aligned}$$

with new context variables F, H, S, and first-order variable z. Thus the variable list $\overline{W}$ in (8) is F, H, S, z.

Let us show that (8) holds.

Our intention with (10) is to restrict solutions to S to either:

$$\begin{aligned} S &\leftarrow \lambda v. f(v, t_2) \text{ (meaning that } t = t_1\text{), or} && (12)\\ S &\leftarrow \lambda v. f(t_1, v) \text{ (meaning that } t = t_2\text{),} && (13) \end{aligned}$$

but exclude solutions like

$$S \leftarrow \lambda v. f(t_1, f(v)), \tag{14}$$

[3] When all function symbols are of arity 1, there exists a well-known trick to represent $s = s' \wedge t = t'$ as $satsbt = s'at's'bt'$, where a, b are different function symbols of arity 1.

(where, for the purposes of the example we suppose that the outermost symbol of t_2 is f). Solution (14) and many likewise should be excluded, since they do not imply that $t = t_1 \vee t = t_2$.

Notice that (9) has *exactly two* following solutions (because there are only two ways to match two occurrences of a on the left of (9) with two pairs of a's on the right):

1. $F \leftarrow \lambda x.f(x, a(f(\varepsilon, a(\varepsilon))))$, $H \leftarrow \lambda u.f(u, \varepsilon)$, or
2. $F \leftarrow \lambda x.f(a(f(a(\varepsilon), \varepsilon)), x)$, $H \leftarrow \lambda u.f(\varepsilon, u)$.

(We also exploit the fact that all possible functions substituted for context variables should use their arguments *exactly once*).

Therefore, (11) ensures that S is either (12) or (13), and thus, by (10), $t = t_1 \vee t = t_2$, as needed.

5.3 Expressing $t = t_1 \vee \ldots \vee t = t_m$ by one equation

The trick from the previous section easily generalizes as follows. Saying

$$t = t_1 \vee \ldots \vee t = t_m$$

is equivalent to saying

$$\exists x_2, x_3, \ldots, x_{m-1} \quad (t = t_1 \vee t = x_2) \wedge (x_2 = t_2 \vee x_2 = x_3) \wedge \ldots$$
$$\ldots \wedge (x_{m-1} = t_{m-1} \vee x_{m-1} = t_m).$$

Now every 2-disjunction from the last formula can be encoded by one equation, as described in Section 5.2 (each time using fresh F_i, H_i, S_i, z_i), and the resulting conjunction results in one equation, as described in Section 5.1.

5.4 Optimized translation with $O(\log m)$ new variables

The translation we described in the previous section adds $O(m)$ existentially quantified variables to eliminate m disjunctions. This is unreasonably much for long disjunctions, and we can do better with only $O(\log m)$ as follows.

Starting with (15), apply the trick of Section 5.2 simultaneously to each pair $t = t_{2i-1} \vee t = t_{2i}$. This will yield (since $\exists$ distributes over $\vee$)

$$\exists F, H, S, z(s_1 = s_1' \vee \ldots \vee s_p = s_p'), \tag{15}$$

where p is $\lceil m/2 \rceil$.

The idea is to repeat this transformation $O(\log m)$ times. There is only a small obstacle to overcome: we know how to eliminate disjunction from $t = t_1 \vee t = t_2$ (with the same t in both equalities), but not from $s_1 = t_1 \vee s_2 = t_2$. However, the latter is equivalent to $f(s_1, t_1) = f(t_1, t_1) \vee f(s_1, t_1) = f(s_1, t_2)$. Thus the term on the left becomes the same in both equalities, and we already know how to deal with this case.

Such an optimized translation introduces only approximately $4\log(m)$ new existentially quantified variables to get rid of m disjunctions.

5.5 Optimized translation with a fixed number of new variables

In this section we go even further and show that *just two* extra existentially quantified context variables suffice to eliminate all disjunctions, independently of their (finite) number. For that purpose we prove the following lemma. Let $[\,]$ be the empty list ε and $[t_0, t_1, \ldots, t_n] = f(t_0, [t_1, \ldots, t_n])$.

Lemma 5.1 Let Θ be the substitution $\Big\{Ct_i/x_i\Big\}_{i=1}^{m}$, where t_i's are the forbidden patterns enumerated in Section 4. Consider the system of two equations:

$$
\begin{aligned}
G(a(H(a))) \quad &= \quad \Big[a\Big((\lambda x_1.[x_1, \ldots, x_m])a\Big), \\
&\qquad \cdots \\
&\qquad a\Big((\lambda x_i.[x_1, \ldots, x_m])a\Big), \\
&\qquad \cdots \\
&\qquad a\Big((\lambda x_m.[x_1, \ldots, x_m])a\Big)\Big]\ \Theta,
\end{aligned}
\tag{16}
$$

$$
Hf(q_0\bar{s}, r) \quad = \quad [x_1, \ldots, x_m]\ \Theta. \tag{17}
$$

For every ground term r the following are equivalent:

1. the system (16), (17) has a solution,

2. $f(q_0\bar{s}, r)$ is an incorrect run.

Brief. Explanation. The term on the right of (16) is:

$$\begin{array}{lllllll} \Big[\; a \;[& a, & Ct_2, & \dots & Ct_i, & \dots & Ct_m \;], \\ \; a \;[& Ct_1, & a, & \dots, & Ct_i, & \dots, & Ct_m \;], \\ & & & & \dots & & \\ \; a \;[& Ct_1, & Ct_2 & \dots, & a, & \dots, & Ct_m \;], \\ & & & & \dots & & \\ \; a \;[& Ct_1, & Ct_2, & \dots, & Ct_i, & \dots, & a \;] \;\Big] \end{array} \tag{18}$$

(with a, or, to be more precise, $a(\varepsilon)$ on the diagonal.)

Proof. The direction 1. $\Leftarrow$ 2. is straightforward. Suppose for a ground r the term $f(q_0\bar{s}, r)$ is an incorrect run for the reason it contains one of the forbidden patterns t_i. Let

$$\begin{aligned} G_i &= \lambda u. \; \Big[a\Big((\lambda x_1.[x_1, \dots, x_m])a\Big), \\ &\qquad \dots \\ &\qquad a\Big((\lambda x_{i-1}.[x_1, \dots, x_m])a\Big), \\ &\qquad u, \\ &\qquad a\Big((\lambda x_{i+1}.[x_1, \dots, x_m])a\Big), \\ &\qquad \dots \\ &\qquad a\Big((\lambda x_m.[x_1, \dots, x_m])a\Big)\Big] \; \Theta, \end{aligned} \tag{19}$$

$$H_i = \Big(\lambda x_i.[x_1, \dots, x_i, \dots, x_m]\Big) \; \Theta. \tag{20}$$

Clearly, substituting such G_i, H_i in (16) yields the identity. Moreover, by substituting H_i in (17) we obtain

$$\begin{array}{llll} [Ct_1, \dots, Ct_{i-1}, & f(q_0\bar{s}, r), & Ct_{i+1}, \dots, Ct_m] = \\ [Ct_1, \dots, Ct_{i-1}, & Ct_i, & Ct_{i+1}, \dots, Ct_m], \end{array}$$

which, obviously, has a solution for C, since $f(q_0\bar{s}, r)$ contains a forbidden pattern t_i by assumption. Thus the system (16), (17) has a solution. □

For the opposite direction 1. $\Rightarrow$ 2. in Lemma 5.1 we prove the following (contrapositive) claim:

Lemma 5.2 Let $f(q_0\bar{s}, r)$ be a correct run. Then the system (16), (17) has no solutions.

Proof. (16), (17) cannot have solutions of the form (19), (20), because the latter would mean that $f(q_0\bar{s}, r)$ is incorrect (see the proof of the previous lemma).

The other 'possible' solutions correspond to the following cases.

1. Either the outermost a on the left of (16) matches one of the outermost a's on the right, but the innermost a on the left does not match the corresponding a on the right. In other words, either

$$\begin{array}{rcl} H' & = & \lambda z.[\ldots a, \ldots, Ct_j\Theta'[z/a(\varepsilon)], \ldots], \text{ or} \\ H'' & = & \lambda z.[\ldots, Ct_j\Theta'[z/a(\varepsilon)], \ldots, a, \ldots], \end{array}$$

   for some substitution Θ' and with a in the i-th ($i \neq j$) place in the list.

   It is easily seen that none of such solutions can satisfy (17), because $Ct_i\Theta'$ cannot be equal (for any C, Θ') to $a \equiv a(\varepsilon)$, because all the forbidden patterns t_i enumerated in Section 4 contain function symbols different from a [4].

2. Or the whole aHa on the left of (16) matches some subterm of $Ct_k\Theta'$, $1 \leq k \leq m$, for some substitution Θ' (i.e., neither of a's in aHa on the left matches a visible a on the right of (16), cf., also (18)).

   Since $q_0\bar{s}$ cannot match any of Ct_j, by construction[5], in order to satisfy (17), H should be substituted with

$$\lambda x.[q_1, \ldots, q_l(x), \ldots, q_m],$$

   i.e., $Ct_k\Theta'$ contains $a[q_1, \ldots, q_l(a), \ldots, q_m]$. Let us show that this cannot yield a solution.

   (a) Suppose, $k \neq l$. Then $Ct_k\Theta'$ *properly contains* q_k, and therefore (17) cannot be satisfied, because it requires $q_k = Ct_k\Theta'$.

   (b) Suppose, $k = l$. We have $Ct_k\Theta'$ contains $a[q_1, \ldots, q_k(a), \ldots, q_m]$. Thus $Ct_k\Theta'$ contains $p + 2$ occurrences of a. On the other hand, to satisfy (17) we should have

$$q_k(f(q_0\bar{s}, r)) = Ct_k\Theta'.$$

   Note that $q_k(f(q_0\bar{s}, r))$ contains at most p occurrences of a (compared with $p + 2$ on the right). Hence the above equality cannot hold.

[4] Recall that we decided to use a complicated pattern $f(F(a(x)))$ instead of the more intuitive $a(x)$ in the beginning of Section 4.1; now the hidden intention becomes clear.

[5] Because the ID $q_0\bar{s}$ is correct and does not contain any forbidden patterns.

Thus, the system (16), (17) has no solutions. This finishes the proof of Lemmas 5.1, 5.2, and the proof of the main claim of his paper. □

Remark. We described an *ad hoc* method to eliminate disjunctions from the positive formulas (of some particular structure) of context unification, which is completely sufficient for the purposes of this paper. It is clear that the method can be generalized so as to apply to *arbitrary* positive formulas. This, however, lies out of the scope of this paper.

References

Büchi, J. R. & Senger, S. (1986), 'Coding in the existential theory of concatenation', *Archive of Mathematical Logic* **26**, 101–106.

Durnev, V. (1997), Studying algorithmic problems for free semi-groups and groups, *in* 'Logical; Foundations of Computer Science (LFCS'97)', Vol. 1234 of *Lect. Notes Comput. Sci.*, Springer-Verlag, pp. 88–101.

Durnev, V. G. (1973), 'Positive theory of a free semigroup', *Soviet Math. Doklady* **211**(4), 772–774.

Farmer, W. (1988), 'A unification algorithm for second-order monadic terms', *Annals Pure Appl. Logic* **39**, 131–174.

Farmer, W. (1991), 'Simple second-order languages for which unification is undecidable', *Theor. Comput. Sci.* **87**, 25–41.

Goldfarb, W. D. (1981), 'The undecidability of the second-order unification problem', *Theor. Comput. Sci.* **13**, 225–230.

Makanin, G. S. (1977), 'The problem of solvability of equations in a free semigroup', *Math USSR Sbornik* **32**(2), 127–198.

Marchenkov, S. S. (1982), 'Undecidability of the positive $\forall\exists$-theory of a free semigroup', *Siberian Math. Journal* **23**(1), 196–198.

Niehren, J., Pinkal, M. & Ruhrberg, P. (1997), On equality up-to constraints over finite trees, context unification and one-step rewriting, *in* W. McCune, ed., 'CADE-14', Lect. Notes Comput. Sci., Springer-Verlag. to appear.

Quine, W. V. (1946), 'Concatenation as a basis for arithmetic', *J. Symb. Logic* **11**(4), 105–114.

Schmidt-Schauß, M. (1994), Unification of stratified second-order terms, Interner Bericht 12/94, University of Frankfurt am Main.

Schmidt-Schauß, M. & Schulz, K. U. (1997), On the exponent of periodicity of minimal solutions of context equations, *in* 'Rewriting Techniques and Applications'98', Lect. Notes Comput. Sci., Springer-Verlag. To appear.

INFORMATIK

Below you find a list of the most recent technical reports of the research group *Logic of Programming* at the Max-Planck-Institut für Informatik. They are available by anonymous ftp from our ftp server `ftp.mpi-sb.mpg.de` under the directory `pub/papers/reports`. Most of the reports are also accessible via WWW using the URL `http://www.mpi-sb.mpg.de`. If you have any questions concerning ftp or WWW access, please contact `reports@mpi-sb.mpg.de`. Paper copies (which are not necessarily free of charge) can be ordered either by regular mail or by e-mail at the address below.

Max-Planck-Institut für Informatik
Library
attn. Birgit Hofmann
Im Stadtwald
D-66123 Saarbrücken
GERMANY
e-mail: `library@mpi-sb.mpg.de`

MPI-I-98-2-007	S. Vorobyov	The Most Nonelementary Theory (A Direct Lower Bound Proof)
MPI-I-98-2-006	P. Blackburn, M. Tzakova	Hybrid Languages and Temporal Logic
MPI-I-98-2-005	M. Veanes	The Relation Between Second-Order Unification and Simultaneous Rigid *E*-Unification
MPI-I-98-2-004	S. Vorobyov	Satisfiability of Functional+Record Subtype Constraints is NP-Hard
MPI-I-98-2-003	R.A. Schmidt	*E*-Unification for Subsystems of *S4*
MPI-I-97-2-012	L. Bachmair, H. Ganzinger, A. Voronkov	Elimination of Equality via Transformation with Ordering Constraints
MPI-I-97-2-011	L. Bachmair, H. Ganzinger	Strict Basic Superposition and Chaining
MPI-I-97-2-010	S. Vorobyov, A. Voronkov	Complexity of Nonrecursive Logic Programs with Complex Values
MPI-I-97-2-009	A. Bockmayr, F. Eisenbrand	On the Chvátal Rank of Polytopes in the 0/1 Cube
MPI-I-97-2-008	A. Bockmayr, T. Kasper	A Unifying Framework for Integer and Finite Domain Constraint Programming
MPI-I-97-2-007	P. Blackburn, M. Tzakova	Two Hybrid Logics
MPI-I-97-2-006	S. Vorobyov	Third-order matching in $\lambda \rightarrow$-Curry is undecidable
MPI-I-97-2-005	L. Bachmair, H. Ganzinger	A Theory of Resolution
MPI-I-97-2-004	W. Charatonik, A. Podelski	Solving set constraints for greatest models
MPI-I-97-2-003	U. Hustadt, R.A. Schmidt	On evaluating decision procedures for modal logic
MPI-I-97-2-002	R.A. Schmidt	Resolution is a decision procedure for many propositional modal logics
MPI-I-97-2-001	D.A. Basin, S. Matthews, L. Viganò	Labelled modal logics: quantifiers
MPI-I-96-2-010	A. Nonnengart	Strong Skolemization
MPI-I-96-2-009	D.A. Basin, N. Klarlund	Beyond the Finite in Automatic Hardware Verification