Linear Second-Order Unification and Context Unification with Tree-Regular Constraints *

Jordi Levy[1]
Mateu Villaret[2]

[1] IIIA, CSIC
Campus de la UAB, Bellaterra, Barcelona, Spain.
http://www.iiia.csic.es/~levy
[2] IMA, UdG
Campus Montilivi, U. de Girona, Girona, Spain.
http://www.ima.udg.es/~villaret

Abstract. Linear Second-Order Unification and Context Unification are closely related problems. However, their equivalence was never formally proved. Context unification is a restriction of linear second-order unification. Here we prove that linear second-order unification can be reduced to context unification with tree-regular constraints.
Decidability of context unification is still an open question. We comment on the possibility that linear second-order unification is decidable, if context unification is, and how to get rid of the tree-regular constraints. This is done by reducing rank-bound tree-regular constraints to word-regular constraints.

1 Introduction

Context Unification (CU) [10, 11] is an extension of First-Order Unification where, in addition to the first-order variables, we also have variables that denote contexts. These *context variables* are applied to arguments, thus the term $F(t)$ denotes any term containing t as a subterm, and F denotes the context surrounding such subterm t. *Linear Second-Order Unification* (LSOU) [3, 5] is a restriction of unification in Second-Order Simply Typed λ-Calculus, where only *linear terms* are considered as possible instances of second-order variables. A linear term is a λ-term where the most external λ-bindings bound one and just one occurrence of the variable. CU is a restriction of LSOU, therefore both problems are between the decidable first-order unification problem and the undecidable second-order one. The common assumption is that CU is decidable. This is because various restrictions of this problem [1, 3, 13–15] make it decidable, while the same restrictions applied to second-order do not [4, 6]. It is also known that, like for the word theory, the context theory is undecidable beyong this existential fragment [9]. The natural question to ask is whether, if CU is decidable, then LSOU will be. This is the main topic of the present paper.

* The first author is partially supported by the project MODELOGOS founded by the CICYT.

CU is a restriction of LSOU where 1) third- or higher-order constants are not allowed, 2) second-order variables are unary, and 3) there are no internal λ-bindings, and external ones are only used to denote the parameter of a second-order variable. The common belief was that third- or higher-order constants do not play an important role w.r.t. the decidability of both problems, neither the use of λ-bindings. The restriction of using unary context variables is not a real restriction because we can replace binary (similarly for n-ary) variables like $F(t_1, t_2)$ by $F_0(f(F_1(t_1), F_2(t_2)))$ introducing a conjectured constant symbol f (see Subsection 3.2 and [12]). However, the equivalence of both problems was never formally proved.

The naive attempt to reduce LSOU to CU by replacing bound variables by new constant symbols does not work. This is because we have to ensure that substitutions avoid *variable capture*. For instance, the following LSOU problem

$$\lambda x.f(x) \doteq_{lsou} \lambda x.f(Y)$$

is not solvable. The substitution $\sigma = [Y \mapsto x]$ gives us:

$$\lambda x.f(x) \doteq_{lsou} \lambda y.f(x)$$

but both terms are not λ-equivalent, because an α-conversion is needed in order to avoid the capture of variable x. However, applying the naive reduction to this problem we get the following solvable CU problem:

$$f(c_x) \doteq_{cu} f(Y)$$

We can try to apply a more sophisticated reduction. Take the original LSOU problem and substitute the bound variables by two distinct constants. However, this method only works for the most external λ-bindings. Applying the reduction to the following solvable LSOU problem with internal λ-bindings:

$$f(g(\lambda x.x), a) \doteq_{lsou} f(Y, Z)$$

we get the following unsolvable CU problem:

$$\begin{array}{l} f(g(c_x), a) \doteq_{cu} f(Y, Z) \\ f(g(c'_x), a) \doteq_{cu} f(Y, Z) \end{array}$$

Bindings can transform free variables into bound variables at different depths. Somehow we have to ensure that if an instance of a (free) variable contains a bound variable, then it also contains its corresponding λ-binding. For instance, given the LSOU problem $F(X) \doteq_{lsou} g(\lambda y.y, a)$ and the following substitutions:

$$\begin{array}{lll} \sigma_1 = [\ X \mapsto a, & F \mapsto \lambda z.g(\lambda y.y, z)\ ] \\ \sigma_2 = [\ X \mapsto y, & F \mapsto \lambda z.g(\lambda y.z, a)\ ] \\ \sigma_3 = [\ X \mapsto g(\lambda y.y, a), & F \mapsto \lambda z.z\ \ \ ] \end{array}$$

only σ_1 and σ_3 are unifiers. As we will show, such a restriction can be ensured by means of tree automata [2], but it does not seem easy to be simply encoded in terms of context equations.

On the other hand, context unification and *word unification* are also closely related problems. Word unification is decidable [8], and can be enriched with regular restrictions without loosing decidability [16]. Tree-regular languages are to terms as regular languages to words. Therefore, if context unification turns out to be decidable, then it seems reasonable to think that context unification could be enriched with tree-regular restrictions without loosing decidability. To support this hypothesis, we would like to prove that membership equations on tree-regular languages can be reduced to membership equations on (word) regular languages, by encoding terms as sequences of symbols (the *traversal sequence*). Unfortunately, we can only prove this reduction for a certain subset of tree-regular languages (what we call *rank-bound tree-regular languages*).

There is a proof that, if context most general unifiers are rank-bound, then CU is decidable [7]. If most general context unifiers were proved to be rank-bound, then tree-regular restrictions would also be rank-bound, and we would also have the decidability of LSOU. We comment on this possibility at the beginning of Section 4.

This paper proceeds as follows. After introducing some basic definitions in Section 2, we reduce LSOU to CU with tree-regular constraints in Section 3. In Section 4 we reduce rank-bound tree-regular restrictions to word-regular restrictions. Finally, in Section 5 we discuss whether this results could be extended to linear third- or higher-order unification.

2 Preliminaries

Let Σ be a simply typed signature where first-order constants are denoted by $a, b, \ldots$, and higher-order constants by $f, g, h, \ldots$. Let $\mathcal{X}$ be an enumerable set of simply typed variables where first-order variables are denoted by capital letters $X, Y, Z, ...$, second-order variables by $F, G, H, ...$ and bound variables by lower-case letters $x, y, z, ...$ Types and their orders are defined as usual in the simply typed λ-calculus. For simplicity, we can assume that there is only one base type. Other types are built as $\tau_1 \to \tau_2$ where τ_1 and τ_2 are types. A term t is said to have arity n if $t : \tau_1 \to \cdots \tau_n \to \tau_0$ where τ_0 is a base type. Second-order terms are also standard: terms constructed using constants and bound variables of any order, but free variables of order at most two. Normal terms (β-reduced, η-expanded terms) have the form $\lambda \boldsymbol{x}.f(t_1, ..., t_n)$ where $\boldsymbol{x}$ is a list of bound variables, f is either a bound, a free variable or a constant, and $t_1, ..., t_n$ are normal terms. If $t_1 : \tau_1, ..., t_n : \tau_n$ then $f : \tau_1 \to ... \to \tau_n \to \tau_0$ where τ_0 is a base type; and, if $x_1 : \tau'_1, ..., x_m : \tau'_m$ then $\lambda \boldsymbol{x}.f(t_1, ..., t_n) : \tau'_1 \to ... \to \tau'_m \to \tau_0$. Notice that, as we are in second-order, if f is a free second-order variable then $\tau_1, ..., \tau_n$ are base types. A term $\lambda \boldsymbol{x}.f(t_1, ..., t_n)$ is said to be linear if, written in normal form, any bound variable $x_i \in \boldsymbol{x}$ occurs once and just once in $f(t_1, ..., t_n)$. Notice that $t_1, ..., t_n$ are not required to be linear.

A linear second-order unification (LSOU) problem is a pair[1] $t \doteq_{lsou} u$ of terms (not necessarily linear). A solution σ of a LSOU problem is a second-

[1] Notice that a set of equations is equivalent to just one equation.

order substitution such that $\sigma(t)$ and $\sigma(u)$ are λ-equivalents, and $\sigma(X)$ is a linear term for any free variable X.

A context unification (CU) problem is a pair $t \doteq_{cu} u$ of terms that does not contain λ-bindings neither constants of order higher than two, and where second-order variables are unary. A solution σ of a CU problem $t \doteq_{cu} u$ is a second-order substitution such that $\sigma(t) = \sigma(u)$, and $\sigma(X)$ is a linear term that does not contain n-ary ($n > 1$) variables, for any free variable X.

A CU problem with tree-regular constraints is a CU problem $t \doteq_{cu} u$ with a tree-regular constraint $v \in \mathcal{A}$.[2] A solution is a ground substitution σ solving the CU problem, and satisfying $\sigma(v) \in L(\mathcal{A})$.

For simplicity we assume that the signature of the problem allows us to ensure the existence of a ground solution whenever a solution exists. Notice that this fact can always be ensured if we extend the signature Σ ensuring that it contains at least a constant $a : \tau_0$ for any base type τ_0 and a binary function $f : \tau_1 \to \tau_2 \to \tau_0$ for any base types τ_0, τ_1 and τ_2.

3 From Linear Second-Order Unification to Context Unification with Tree-Regular Constraints

In this section, we prove that LSOU can be reduced to CU plus tree-regular constraints. This reduction is done in two steps. In subsection 3.1 we reduce the LSOU problem to the n-ary context unification problem by removing λ-bindings and constants with order higher than two. We obtain a context unification problem with n-ary contexts, i.e., second-order variables of arity n, plus tree-regular constraints. In subsection 3.2 we translate n-ary contexts to (1-ary) contexts.

3.1 Reducing Linear Second-Order Unification to n-ary Context Unification plus Tree-Regular Constraints

The translation from LSOU to n-ary CU has to remove λ-bindings from terms. Bound variables will be replaced by new constants. In second-order λ-calculus, λ-bindings of normal terms are always just below higher-order constants or bound variables, or are the most external symbol. They are never just below free variables. We can eliminate external λ-bindings by extending the signature Σ with an appropriate new unary constant o (if it does not contain any one) and translating the equation $\lambda \boldsymbol{x}.s \doteq_{lsou} \lambda \boldsymbol{y}.t$ into $o(\lambda \boldsymbol{x}.s) \doteq_{lsou} o(\lambda \boldsymbol{y}.t)$. This new problem does not have external λ-bindings and is equivalent to the original one.

The elimination of internal λ-bindings is performed in three steps:

First, we conjecture an α-conversion of bound variables in order to allow unification when they are later translated into constants in the following step. Notice that the second step of this translation procedure depends on the "names" of these bound variables.

[2] Notice that a set of tree-regular constraints is equivalent to just one constraint.

Second, let $B \subset \mathcal{X}$ be a finite set of variables and let $A \subset \mathcal{L}istsof\,(B)$ be a finite set of lists of variables from B. We define a translation function $trans^{A,B}$ that replaces any occurrence of a variable of B by a new first-order constant, and any occurrence of a λ-abstraction, whose list of bound variables is in A, also by a new constant. This set B will be the set of bound variables of the unification problem resulting from step 1, and A will be the set of lists of variables used in the λ-abstractions.

The signature Σ' of the resulting n-ary CU problem also depends on the set B of bound variables conjectured in the previous step, and on the set A of lists of bound variables of the λ-abstractions. It is defined as follows: Σ' contains the same constants as Σ, but every constant h or bound variable z, with order higher than two, is replaced in Σ' by a new second-order constant h' or c_z, respectively. The arity of h' (similarly for c_z) is equal to the arity of h plus its number of non-first-order arguments. Any non-first-order n-ary argument of h with type $\tau_1 \to \cdots \to \tau_n \to \tau_0$ is replaced by two first-order arguments, one with a new special type o, and the other with the base type τ_0. For instance, if $h : \tau_1 \to (\tau_2 \to \tau_3) \to \tau_4$ then $h' : \tau_1 \to o \to \tau_3 \to \tau_4$. The signature Σ' also contains a new constant symbol $b_{[x_1,...,x_n]}$ of type o, for every list $[x_1, ..., x_n] \in A$, and a new constant symbol c_x, for every variable $x \in B$. The set of variables of the resulting problem is $\mathcal{X}' = \mathcal{X} \backslash B$.

Let $t \in T(\Sigma, \mathcal{X})$ be a term, $B \subset \mathcal{X}$ a set of variables, and $A \subset \mathcal{L}istsof\,(B)$ a set of lists of variables from B. The term $trans^{A,B}(t) \in T(\Sigma', \mathcal{X}')$ is defined by:

$$trans^{A,B}(c) = c$$

$$trans^{A,B}(f(t_1, ..., t_n)) = f(trans^{A,B}(t_1), ..., trans^{A,B}(t_n))$$

$$trans^{A,B}(X) = \begin{cases} X & \text{if } X \notin B \\ c_X & \text{if } X \in B \end{cases}$$

$$trans^{A,B}(F(t_1, ..., t_n)) = \begin{cases} F(trans^{A,B}(t_1), ..., trans^{A,B}(t_n)) & \text{if } F \notin B \\ c_F(trans^{A,B}(t_1), ..., trans^{A,B}(t_n)) & \text{if } F \in B \end{cases}$$

$$trans^{A,B}(h(t_1, ..., t_n, \lambda \boldsymbol{x_1}.u_1, ..., \lambda \boldsymbol{x_m}.u_m)) = ...$$
$$= h'(trans^{A,B}(t_1), ..., trans^{A,B}(t_n), b_{\boldsymbol{x_1}}, trans^{A,B}(u_1), ..., b_{\boldsymbol{x_m}}, trans^{A,B}(u_m))$$

$$trans^{A,B}(z(t_1, ..., t_n, \lambda \boldsymbol{x_1}.u_1, ..., \lambda \boldsymbol{x_m}.u_m)) = ...$$
$$= c_z(trans^{A,B}(t_1), ..., trans^{A,B}(t_n), b_{\boldsymbol{x_1}}, trans^{A,B}(u_1), ..., b_{\boldsymbol{x_m}}, trans^{A,B}(u_m))$$

$$trans^{A,B}(\lambda \boldsymbol{x}.t) = \lambda \boldsymbol{x}.trans^{A',B\backslash \boldsymbol{x}}(t)$$

In the fifth and sixth case, for constants h and variables z with order higher than two, we assume for simplicity that non-first-order parameters are in the last positions. The constant h' is the second-order constant associated to h, c_z is the constant associated to the variable z, and $b_{\boldsymbol{x_i}}$ is the constant associated to the list of variables $\boldsymbol{x_i} \in A$ of the λ-binding $\lambda \boldsymbol{x_i}$. If, for some $i \in [1..m]$, $\boldsymbol{x_i} \notin A$, then the translation is undefined. In the last case, A' is the set of lists A where any list containing variables from $\boldsymbol{x}$ has been removed. Notice that most external λ-bindings are not removed by this translation.

Third, we introduce a set of tree-regular restrictions over the instantiations of variables to prevent them from containing constants associated to bound variables from B without its corresponding λ-bindings.

The tree automata $\mathcal{A}^{A,B} = \langle \Sigma, \mathcal{Q}, \mathcal{Q}_f, \Delta \rangle$ that characterises the set of terms that do not contain these bound variables from B in *free positions* is defined as follows.

- The signature contains the set of constants Σ'. Remember that Σ' allows us to ensure that, if a certain CU problem S is solvable then, S has a ground solution.
- The set of states is $\mathcal{Q} = \{q_X | X \subseteq B\} \cup \{p_X | X \in A\}$; where B is the set of bound variables and A is the set of λ-bindings.
- There is a single final state $\mathcal{Q}_f = \{q_\emptyset\}$
- The set of transitions Δ is defined as follows:
  - For any first-order constant $a \in \Sigma'$ not associated to a variable from B:

$$a \to q_\emptyset$$

  - For any first-order constant c_x associated to a bound variable $x \in B$ and any first-order constant $b_{\boldsymbol{y}}$ associated to a list of bound variables $\boldsymbol{y} \in A$:

$$c_x \to q_{\{x\}}$$
$$b_{\boldsymbol{y}} \to p_{\boldsymbol{y}}$$

  - For any second-order constant $f \in \Sigma'$ not associated to a variable from B and, for any constant c_x associated to a bound variable $x \in B$, and states $q_{A_1}, \ldots, q_{A_n}$:

$$f(q_{A_1}, ..., q_{A_n}) \to q_{A_1 \cup ... \cup A_n}$$
$$c_x(q_{A_1}, ..., q_{A_n}) \to q_{\{x\} \cup A_1 \cup ... \cup A_n}$$

  - For any second-order constant $h' \in \Sigma'$ associated to a higher-order constant $h \in \Sigma$ and, for any second-order constant $c_z \in \Sigma'$ associated to a higher-order bound variable $z \in B$:

$$h'(q_{A_1}, ...q_{A_n}, p_{B_1}, q_{C_1}, ..., p_{B_m}, q_{C_m}) \to q_D$$
$$c_z(q_{A_1}, ...q_{A_n}, p_{B_1}, q_{C_1}, ..., p_{B_m}, q_{C_m}) \to q_E$$

    where

$$D = \bigcup_{i \in [1..n]} A_i \cup \bigcup_{j \in [1..m]} (C_j \setminus B_j)$$

$$E = \{z\} \cup \bigcup_{i \in [1..n]} A_i \cup \bigcup_{j \in [1..m]} (C_j \setminus B_j)$$

    Notice that the B_k's are treated in the transitions as sets but they denote lists: $\lambda x, y$ is not the same λ-abstraction as $\lambda y, x$, so they have distinct associated constants but here are treated as the same set.

Then, we introduce a set of tree-regular restrictions over the solution σ of the translated problem.

- For any first-order variable X, the restriction $\sigma(X) \in L(\mathcal{A}^{A,B})$

– For any second-order variable F, the restriction $\sigma(F(a_1, ..., a_n)) \in L(\mathcal{A}^{A,B})$, where $a_i \in \Sigma'$ are first-order constants of the appropriate types.

Example 1. Given the problem $f(X, X) \doteq_{lsou} f(g(\lambda x.F(x)), F(g(\lambda y.y)))$ we can conjecture the following α-equivalent problem (this is the only solvable one) $f(X, X) \doteq_{lsou} f(g(\lambda x.F(x)), F(g(\lambda x.x)))$ and translate it into the following context unification problem with tree-regular constraints

$$\begin{array}{l} f(X, X) \doteq_{cu} f(g'(b_{[x]}, F(c_x)), F(g'(b_{[x]}, c_x))) \\ \sigma(X) \in L(\mathcal{A}^{\{[x]\},\{x\}}) \\ \sigma(F(a)) \in L(\mathcal{A}^{\{[x]\},\{x\}}) \end{array}$$

where the tree automata $\mathcal{A}^{\{[x]\},\{x\}}$ is defined by

$$\begin{array}{ll} a \to q_\emptyset & b_{[x]} \to p_{\{x\}} \\ f(q_A, q_B) \to q_{A \cup B} & c_x \to q_{\{x\}} \\ g'(p_A, q_B) \to q_{B \setminus A} & \end{array}$$

In the following lemmas we assume that all bound variables are in B and bindings in A.

Lemma 1. *For any second-order substitution σ satisfying $\sigma(X)$ does not contain variables of B in free positions, and the domain of σ neither contains variables of B, let $\tau = trans^{A,B}(\sigma)$ be the context substitution defined by $\tau(X) = trans^{A,B}(\sigma(X))$. Then, for any term t we have*

$$trans^{A,B}(\sigma(t)) = \tau(trans^{A,B}(t))$$

Lemma 2. *For any second-order term t the set of free variables of t and B are disjoint if and only if $trans^{A,B}(t) \in L(\mathcal{A}^{A,B})$.*

Theorem 1. *A LSOU problem $s \doteq_{lsou} t$ is unifiable if and only if there exists an α-equivalent unification problem $s' \doteq_{lsou} t'$ such that*

$$trans^{A,B}(s') \doteq_{cu} trans^{A,B}(t')$$

and the corresponding tree-regular constraints are solvable. Here, B is the set of bound variables of s' and t', and A is the set of lists of bound variables corresponding to λ-abstractions of s' and t'.

Corollary 1. *Linear second-order unification is reducible to n-ary context unification plus tree-regular constraints.*

3.2 Reducing n-ary context unification to (1-ary) context unification

In this subsection we reduce the n-ary CU problem to the (1-ary) CU problem. The same main ideas are also used in other previous papers, like [12]. Given an

n-ary context unification problem S over a signature Σ, if Σ does not contain an n-ary constant with $n \geq 2$ and a first-order constant, we enlarge it with them. We construct a new context unification problem S' by iteratively applying the following rule, until all non-unary context variables of the problem disappear.

For any n-ary context variable F with $n \geq 2$, we guess a p-ary constant symbol g, with $p \geq 2$, from the signature. Then, we guess a partition of $\{1, \ldots, n\}$ into $p \leq n$ many *disjoint* subsets such that $\bigcup_{i \in [1..p]} \{c_1^i, \ldots, c_{q_i}^i\} = \{1, \ldots, n\}$, and at least two of them are non-empty. We instantiate F by the following substitution:

$$[F \mapsto \lambda x_1 \cdots \lambda x_n . F_0(g(F_1(x_{c_1^1}, \ldots, x_{c_{q_1}^1}), \ldots, F_p(x_{c_1^p}, \ldots, x_{c_{q_p}^p})))]$$

where $F_0, \ldots, F_p$ are (maybe non-unary) context or first-order variables.

Example 2. Consider the following n-ary context unification problem:

$$X(Y(a, b)) \doteq_{lsou} Y(X(a), b) \tag{1}$$

where one of its infinitely many minimal solutions is (see Figure 1):

$$\begin{array}{l} \sigma = [\ X \mapsto \lambda x . Z(Z(x, b), b), \\ \qquad\ Y \mapsto \lambda x, y . Z(Z(Z(x, b), y), b), b)] \end{array} \tag{2}$$

where Z is a fresh context variable. We enlarge our signature to $\Sigma' = \{a, b, g\}$, where g is a new binary constant. Now, we can guess a partition of $\{1, 2\}$ into two disjoint subsets $\{1\}$ and $\{2\}$, where both are non-empty, and instantiate Y by:

$$\tau = [Y \mapsto \lambda x_1, x_2 . Y_0(g(Y_1(x_1), Y_2(x_2)))]$$

We obtain a new problem (see Figure 2):

$$X(Y_0(g(Y_1(a), Y_2(b)))) \doteq_{cu} Y_0(g(Y_1(X(a)), Y_2(b))) \tag{3}$$

which is also solvable, and only contains (unary) context variables.

Theorem 2. *n-ary context unification is NP-reducible to (1-ary) context unification.*

4 Translating Tree-Regular Constraints to Regular Constraints over Traversal Sequences

The decidability of context unification with tree-regular constraints, as well as the decidability of context unification, are still open problems. There is a proof that, if most general context unifiers are rank-bound, then CU is decidable [7]. However, it is not known if most general context unifiers are in general rank-bound. In this section we will show that, if this is the case, then the decidability proof of context unification could be extended to context unification with

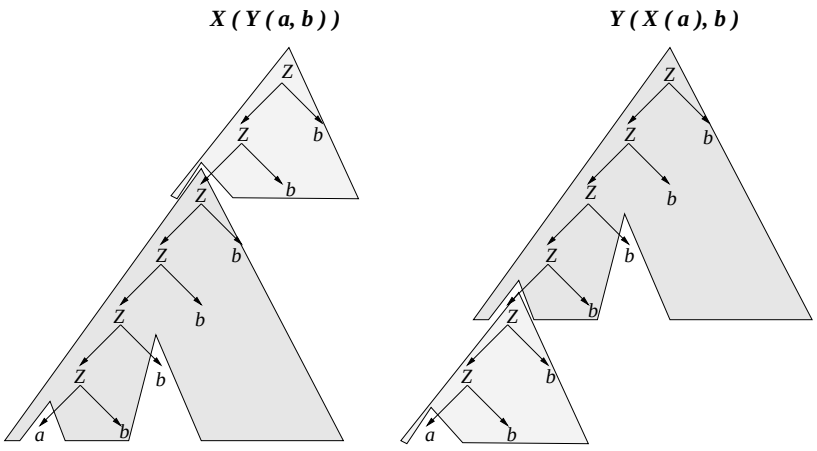


Fig. 1. A solution of the LSOU problem (1)

tree-regular constraints. Therefore, linear second-order unification would also be decidable.

This mentioned proof is based on a reduction of context unification to word unification with regular constraints [16], where terms are translated into sequences of symbols (traversal sequences). In the following we present the main ideas of this reduction.

Definition 1. *Given a signature* $\langle \Sigma, \mathcal{X} \rangle$*, we define the* extended signature

$$\Sigma^{\Pi} = \{ f^{\rho} \mid f \in \Sigma \cup \mathcal{X} \ \wedge \ \rho \in \Pi_{\mathrm{arity}(f)} \}$$

where Π_n *is the group of permutations over* n *elements.*
A sequence $s \in (\Sigma^{\Pi})^*$ *is said to be a* traversal sequence *of a term* t*, noted* $s \in \mathrm{trav}(t)$*, if:*

1. $s = t$ *when* $t = c$ *is a* 0*-ary symbol*
2. $s = f^{\rho} s_{\rho(1)} \cdots s_{\rho(n)}$ *when* $t = f(t_1, \ldots, t_n)$ *being* s_i *traversal sequences of* t_i *for any* $i \in [1..n]$*, and* $\rho \in \Pi_n$ *a permutation.*

Any traversal sequence of a term characterises this term. We use an extended signature with permutations in order to allow us the use of distinct traversals, i.e. the traversals of subterms in distinct possible orders.

Definition 2. *The* rank of a term, $\mathrm{rank}(t)$*, is defined by* $\mathrm{rank}(a) = 0$*, for any constant* a*, and* $\mathrm{rank}(f(t_1, ..., t_n)) = c$ *where* c *is the minimum integer satisfying: there exists a permutation* τ *of indices* $1, ..., n$ *such that, for any* $i \in [1..n]$*,* $\mathrm{rank}(t_{\tau_i}) \leq c - n + i$*.*

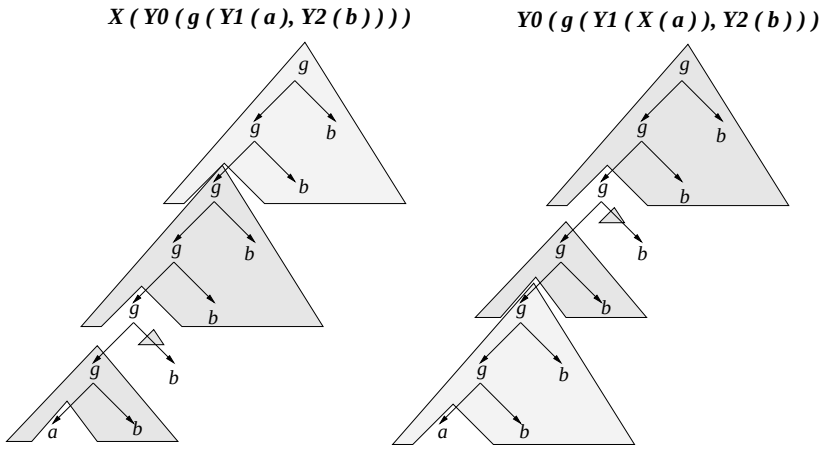


Fig. 2. A reduction of the LSOU problem (1) to context unification

This definition is bizarre, but it can be simplified for binary trees as follows:[3]

$$\mathrm{rank}(a) = 0$$
$$\mathrm{rank}(f(t_1, t_2)) = \begin{cases} \mathrm{rank}(t_1) + 1 & \text{if } \mathrm{rank}(t_1) = \mathrm{rank}(t_2) \\ \max\{\mathrm{rank}(t_1), \mathrm{rank}(t_2)\} & \text{otherwise} \end{cases}$$

The rank of a term allows us to define a normal traversal.

Definition 3. *Given a term t, its* normal traversal sequence $\mathrm{NF}(t)$ *is defined recursively as follows:*

1. *If $t = a$ then $\mathrm{NF}(t) = a$.*
2. *If $t = f(t_1, \ldots, t_n)$ then let be $\rho \in \Pi_n$ the permutation satisfying*

$$i < j \Rightarrow \big(\mathrm{rank}(t_{\rho(i)}) < \mathrm{rank}(t_{\rho(j)}) \\ \vee \mathrm{rank}(t_{\rho(i)}) = \mathrm{rank}(t_{\rho(j)}) \wedge \rho(i) < \rho(j)\big)$$

Then, $\mathrm{NF}(t) = f^{\rho}\ \mathrm{NF}(t_{\rho(1)})\ \cdots\ \mathrm{NF}(t_{\rho(n)})$.

Notice that a restriction on the rank of a tree does not imply a restriction on its size. The following conjecture states that the rank of any most general context unifier is bound:

Conjecture 1. For any solvable context unification problem $t \doteq u$ and m.g.u. σ, we have

$$\mathrm{rank}(\sigma(t)) \leq \phi(\mathrm{size}(t \doteq u))$$

where ϕ is a computable function.

[3] Alternatively, we can also define the rank of a binary tree t as the depth of the maximum complete tree t' (a tree where all leaves are at the same depth) such that there exist an injective morphism from t' to t.

Traversal sequences of rank-bound terms and regular language are related. For any signature Σ and bound n, there exists a regular language R^n_Σ such that, for any n-rank-bound term t there exists a traversal sequence $w \in \text{trav}(t)$ with $w \in R^n_\Sigma$. We can restrict the choice of the traversal sequence to the traversal normal form. Thus, the set of traversal normal forms of rank-bound terms is a regular language. For instance, for a signature with a constant a and a binary function f, any term satisfying $\text{rank}(t) \leq 1$ has a traversal belonging to:

$$R^1 \equiv ((f^{[1,2]} \mid f^{[2,1]})\ a)^*\ a$$

and those satisfying $\text{rank}(t) \leq 2$ have a traversal belonging to:

$$R^2 \equiv ((f^{[1,2]} \mid f^{[2,1]})\ R^1)^*\ a$$

This allows us to reduce any CU problem, like $X(f(Y(a), Z(b))) \doteq_{cu} Y(f(X(a), Z(b)))$ to some word unification problem plus *traversal equations*, like:

$$\begin{array}{l} X_0\ f^{[1,2]}\ Y_0\ a\ Y_1\ Z_0\ b\ Z_1\ X_1 \doteq_{wu} Y'_0\ f^{[1,2]}\ X'_0\ a\ X'_1\ Z'_0\ b\ Z'_1\ Y'_1 \\ X_0\ c\ X_1 \equiv X'_0\ c\ X'_1 \\ Y_0\ c\ Y_1 \equiv Y'_0\ c\ Y'_1 \\ Z_0\ c\ Z_1 \equiv Z'_0\ c\ Z'_1 \end{array}$$

where the words X_0 and X_1 encode a traversal sequence of the context X, and X'_0 and X'_1 another traversal of X. The intended meaning of $w_1 \equiv w_2$ is: w_1 and w_2 are *similar* traversal sequences of the same term. By similar we mean that we can bound the number of permutations in which w_1, w_2 and $\mathbb{NF}(t)$ differ, where $w_1, w_2 \in \text{trav}(t)$. If the rank of this term is bound, then we can non-deterministically reduce these traversal equations to word equations plus regular restrictions like

$$\begin{array}{ll} X_0\ c\ X_1 \doteq_{wu} X'_0\ c\ X'_1 & X_0\ c\ X_1 \in R^{\phi(\text{size})} \\ Y_0\ c\ Y_1 \doteq_{wu} Y'_0\ c\ Y'_1 & Y_0\ c\ Y_1 \in R^{\phi(\text{size})} \\ Z_0\ c\ Z_1 \doteq_{wu} Z'_0\ c\ Z'_1 & Z_0\ c\ Z_1 \in R^{\phi(\text{size})} \end{array}$$

The restriction $X_0\ c\ X_1 \in R^{\phi(\text{size})}$ ensures that the instances we find for these words are really traversal sequences.

In what follows we show how membership equations on tree-regular languages of rank-bound terms can be reduced to membership equations on (word) regular languages. We start by defining rank-bound tree automata.

Definition 4. *For any tree automata $\mathcal{A} = \langle \Sigma, \mathcal{Q}, \mathcal{Q}_f, \Delta \rangle$, and any state $q_i \in \mathcal{Q}$, we define*[4]

$$\text{rank}(q_i) = \max\{\text{rank}(t) \mid t \in L(\langle \Sigma, \mathcal{Q}, \{q_i\}, \Delta \rangle)\}$$

For any tree automata $\mathcal{A} = \langle \Sigma, \mathcal{Q}, \mathcal{Q}_f, \Delta \rangle$, we define

$$\text{rank}(\mathcal{A}) = \max\{\text{rank}(q_i) \mid q_i \in \mathcal{Q}_f\}$$

A tree automata $\mathcal{A}$ is said to be bound if $\text{rank}(\mathcal{A}) < \infty$.

[4] Notice that $\langle \Sigma, \mathcal{Q}, \{q_i\}, \Delta \rangle$ is similar to $\mathcal{A}$ but with a unique final state q_i.

Notice that the rank of the states of a tree automata satisfies the following property

- for any state q having only transitions like $c \to q$, where $c \in \Sigma$ is a 0-ary constant, $\text{rank}(q) = 0$,
- for any accessible state q_0 having transitions like $f(q_1, q_2) \to q_0$, where $f \in \Sigma$ is a binary function, we have:

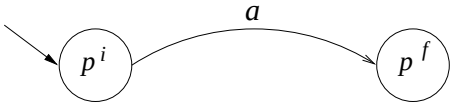
$$rank(q_0) \geq \begin{cases} max\{rank(q_1), rank(q_2)\} & \text{if } rank(q_1) \neq rank(q_2) \\ rank(q_1) + 1 & \text{if } rank(q_1) = rank(q_2) \end{cases}$$

We translate tree-regular restrictions to (word) regular restrictions over traversal sequences of terms. For simplicity assume that any symbol of the signature is, at most, binary. The result can easily be extended to any signature.

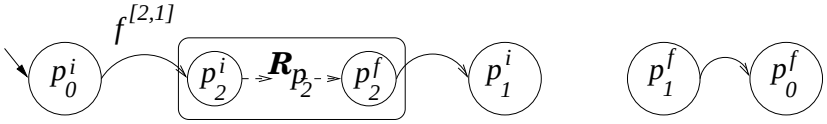
For any rank-bound tree automata $\mathcal{A} = \langle \Sigma, \mathcal{Q}, \mathcal{Q}_f, \Delta \rangle$, and any state $q \in \mathcal{Q}$, we define a regular language R_q satisfying $R_q \cap \text{trav}(t) \neq \emptyset$, for any term $t \in L(\langle \Sigma, \mathcal{Q}, \{q\}, \Delta \rangle)$, and $R_q \subseteq \bigcup\{\text{trav}(t) \mid t \in L(\langle \Sigma, \mathcal{Q}, \{q\}, \Delta \rangle)\}$.

We will construct the automata that recognises R_q using the following rules. Assume that $R_{q'}$ is already computed for any state $q' \in \mathcal{Q}$ with $\text{rank}(q') < \text{rank}(q)$. Let be $n = \text{rank}(q)$. The automata R_q has a pair of states p^i and p^f for any state p of the tree automata satisfying $\text{rank}(p) = n$, and some additional states that we will specify later. The initial state of R_q is q^i, and there is a single final state and it is q^f. The set of transitions of R_q is defined as follows.

- *Base case*, for any state $p \in \mathcal{Q}$ satisfying $\text{rank}(p) = n$, and any transition $a \to p \in \Delta$ we add a transition:



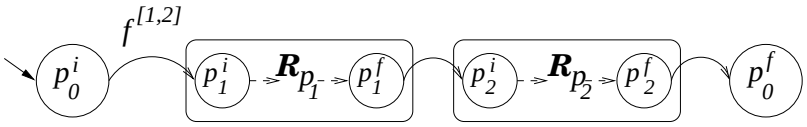
  from p^i to p^f labelled with a.
- *Inductive case 1*, for any state p_0 with $\text{rank}(p_0) = n$ and any transition $f(p_1, p_2) \to p_0$ satisfying $\text{rank}(p_2) < \text{rank}(p_1) \leq \text{rank}(p_0)$ [5]



  we can assume that R_{p_2} is already computed. We add a copy of the automata R_{p_2}, i.e. a copy of all its states and transitions (these are the unspecified additional states). We also add a transition from p_0^i to the initial state of the copy R_{p_2} labelled with $f^{[2,1]}$, a λ-transition from the final state of R_{p_2} to p_1^i, and another from p_1^f to p_0^f.
  For any transition $f(p_1, p_2) \to p_0$ satisfying $\text{rank}(p_1) < \text{rank}(p_2) \leq \text{rank}(p_0)$ we do something similar using the label $f^{[1,2]}$

[5] Notice that if $\text{rank}(p_1) = \text{rank}(p_2) = \text{rank}(p_0)$ then $\text{rank}(p_0) = \infty$ and the tree automata would be non-rank-bound. Thus the existence of a bound n for the rank of the tree automata is crucial in our translation.

– *Inductive case 2*, for any state p_0 with $\text{rank}(p_0) = n$ and any transition $f(p_1, p_2) \to p_0$ satisfying $\text{rank}(p_1) < \text{rank}(p_0)$ and $\text{rank}(p_2) < \text{rank}(p_0)$



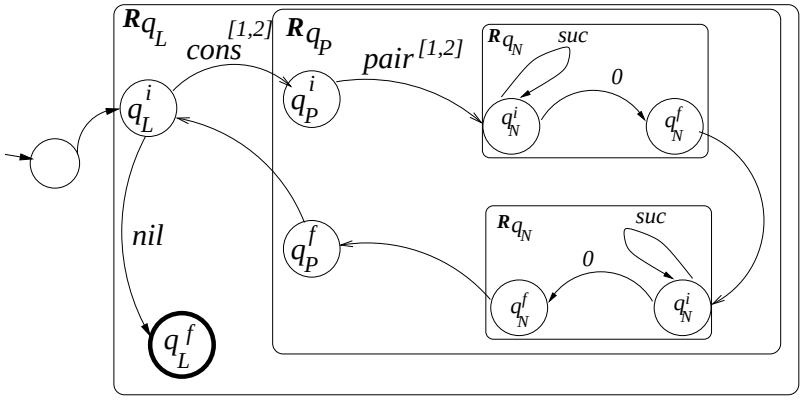
we can assume that R_{p_1} and R_{p_2} have been already computed. We add a copy of each one of these automata, a transition from p_0^i to the initial state of R_{p_1} labelled with $f^{[1,2]}$, a λ-transition from the final state of R_{p_1} to the initial state of R_{p_2}, and another from the final state of R_{p_2} to p_0^f.

Notice that with these cases, transitions like $f(q, q) \to q$ are not considered, because this means that $\text{rank}(q) = \infty$, and q can not lead to a final state. The final automata associated to $\mathcal{A}$ consists of an initial state q_0, a copy of R_q for any final state $q \in \mathcal{Q}_f$, a λ-transitions from q_0 to each one of the initial states of the R_q's. The set of final states is the set of final states of the R_q's.

Example 3. The tree automata defined by the following transitions

$0 \to q_N$, $\quad pair(q_N, q_N) \to q_P$, $\quad nil \to q_L$,
$s(q_N) \to q_N$, $\quad cons(q_P, q_L) \to q_L$

is translated into the following regular automata:



The term $cons(pair(suc(suc(0)), suc(0)), cons(pair(suc(0), 0), nil))$ recognised by the tree automata, has a traversal sequence

$$cons^{[1,2]}\ pair^{[1,2]}\ suc\ suc\ 0\ suc\ 0\ cons^{[1,2]}\ pair^{[1,2]}\ suc\ 0\ 0\ nil$$

recognised by the regular automata.

Theorem 3. *For any tree-regular language $L(\mathcal{A})$ of a rank-bound tree automata $\mathcal{A}$, let $L(B)$ be the regular language recognised by the automata B resulting from applying the previous translation. The following properties hold.*

1. *If $t \in L(\mathcal{A})$ then there exist a sequence $l \in L(B)$ such that $l \in trans(t)$.*

2. If $l \in L(B)$ then there exist a term $t \in L(\mathcal{A})$ such that $l \in trans(t)$.

The set of terms satisfying $\text{rank}(t) \leq n$ defines a tree-regular language. Moreover, this language can be recognised by a rank-bound tree automata $\mathcal{A}_n$. For instance, if $\Sigma = \{a, b, h(), f(,)\}$, then

$$\mathcal{A}_n = \begin{cases} \Sigma = \{a, b, h(), f(,)\},\ \mathcal{Q} = \{q_0, ..., q_n, q_{n+1}\},\ \mathcal{Q}_f = \{q_0, ..., q_n\}, \\ \Delta = \begin{cases} a \to q_0, \quad b \to q_0, & \\ h(q_i) \to q_i & \text{for any } i \in [0..n+1] \\ f(q_i, q_j) \to q_{max\{i,j\}} & \text{for any } i, j \in [0..n+1] \text{ with } i \neq j \\ f(q_i, q_i) \to q_{i+1} & \text{for any } i \in [0..n] \\ f(q_{n+1}, q_{n+1}) \to q_{n+1} & \end{cases} \end{cases}$$

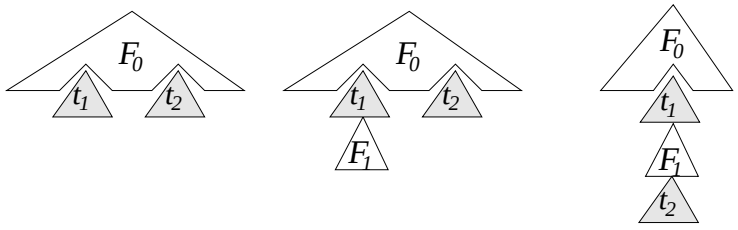
Definition 5. *A tree-regular language L is said to be n rank-bound if for any term $t \in L$ we have* $\text{rank}(t) \leq n$.

Theorem 4. *Any rank-bound regular language is recognised by a rank-bound tree automata. The language recognised by a rank-bound tree automata is a rank-bound tree-regular language.*

5 Extending the Results to Higher-Order Unification

In Section 3 we have shown how linear second-order unification can be reduced to context unification with tree-regular constraints. In this section we discuss whether this result could be extended to linear higher-order unification.

Higher-order unification can be defined as the problem of finding a substitution σ making the normal form of two instances of terms $\sigma(s)$ and $\sigma(t)$ equal. When we try to find such a substitution we have to take into account how this terms will β-reduce after being instantiated. The problem is simple in linear second-order. We know that any instance of $F(t_1, ..., t_n)$, after β-reduction, will contain $\sigma(t_i)$ as subterms, and representing $\sigma(F(t_1, ..., t_n))$ as a tree, all nodes corresponding to $\sigma(F)$ will be connected, forming a context. In third-order the situation is more complicate. First, we have to require instances of variables to be linear in all λ-bindings, i.e. not only in the most external λ-bindings. If we apply the substitution $F \mapsto \lambda y.\lambda z.g_1(y(g_2(z)))$ to $F(\lambda x.f(x), a)$ we get $g_1(f(g_2(a)))$. The nodes corresponding to F are no longer connected: $\sigma(F)$ is broken into pieces, and some of the arguments can also disappear. Each one of such pieces forms a kind of context. For instance, if $F : (o \to o) \to o \to o$, any instance of $F(t_1, t_2)$ has one of the following forms:



Each one of these situations is captured respectively by:

$$F \mapsto \lambda x.\lambda y.F_0(\lambda z.x(z), y)$$
$$F \mapsto \lambda x.\lambda y.F_0(\lambda \boldsymbol{z}.x(F_1(\boldsymbol{z})), y)$$
$$F \mapsto \lambda x.\lambda y.F_0(\lambda \boldsymbol{z}.x(F_1(y, \boldsymbol{z})))$$

In this example, F_0 is still a third-order typed variable. Moreover, the first instantiation is $F \mapsto F_0$ in normal form, so it subsumes the other two. The second one is equal to the first one, if $\boldsymbol{z}$ contains a single variable and we instantiate $F_1 \mapsto \lambda z.z$. In fact, this classification only makes sense if we translate F_0 into a context variable using the method described in Section 3 for higher-order constants. We would get:

$$F \mapsto \lambda x.\lambda y.F'_0(d_z, x(c_z), y)$$
$$F \mapsto \lambda x.\lambda y.F'_0(X_{\boldsymbol{z}}, x(F_1(\boldsymbol{c_z})), y)$$
$$F \mapsto \lambda x.\lambda y.F'_0(X_{\boldsymbol{z}}, x(F_1(y, \boldsymbol{c_z})))$$

The variable $X_{\boldsymbol{z}}$ encodes the binding $\lambda \boldsymbol{z}$. If we were able to know a priori how long and, with which types, can be these λ-bindings, then the translation would not seem much more complicate than in the second-order case.

Acknowledgements We thank M. Bonet and all the anonymous referees for their helpful comments on the paper.

References

1. H. Comon. Completion of rewrite systems with membership constraints. *J. of Symbolic Computation*, 25(4):397–453, 1998.
2. H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available on: http://www.grappa.univ-lille3.fr/tata, 1997.
3. J. Levy. Linear second-order unification. In *7th Int. Conf. on Rewriting Techniques and Applications, RTA'96*, volume 1103 of *LNCS*, pages 332–346, New Jersey, USA, 1996.
4. J. Levy. Decidable and undecidable second-order unification problems. In *9th Int. Conf. on Rewriting Techniques and Applications, RTA'98*, volume 1379 of *LNCS*, pages 47–60, Tsukuba, Japan, 1998.
5. J. Levy and J. Agustí. Bi-rewrite systems. *J. of Symbolic Computation*, 22:279–314, 1996.
6. J. Levy and M. Veanes. On the undecidability of second-order unification. *Information and Computation*, 2000. (to appear).
7. J. Levy and M. Villaret. On the decidability of context unification. Technical report, IIIA, CSIC, 2000.
8. G. S. Makanin. The problem of solvability of equations in a free semigroup. *Math. USSR Sbornik*, 32(2):129–198, 1977.
9. J. Niehren, M. Pinkal, and P. Ruhrberg. On equality up-to constraints over finite trees, context unification, and one-step rewriting. In *14th International Conference on Automated Deduction, CADE-14*, volume 1249 of *LNCS*, pages 34–48, Townsville, North Queensland, Australia, 1997.

10. M. Schmidt-Schauß. An algorithm for distributive unification. In *7th Int. Conf. on Rewriting Techniques and Applications, RTA'96*, volume 1103 of *LNCS*, pages 287–301, New Jersey, USA, 1996.
11. M. Schmidt-Schauß. A decision algorithm for distributive unification. *Theoretical Computer Science*, 208:111–148, 1998.
12. M. Schmidt-Schauß. Decidability of bounded second-order unification. Technical Report Frank-report-11, FB Informatik, J.W. Goethe Universität Frankfurt, 1999.
13. M. Schmidt-Schauß. A decision algorithm for stratified context unification. Technical Report Frank-report-12, FB Informatik, J.W. Goethe Universität Frankfurt, 1999.
14. M. Schmidt-Schauß and K.U. Schulz. On the exponent of periodicity of minimal solutions of context equations. In *9th Int. Conf. on Rewriting Techniques and Applications, RTA'98*, volume 1379 of *LNCS*, pages 61–75, Tsukuba, Japan, 1998.
15. M. Schmidt-Schaußand K.U. Schulz. Solvability of context equations with two context variables is decidable. In *Conference on Automated Deduction, CADE'99*, LNCS, pages 67–81, 1999.
16. K. U. Schulz. Makanin's algorithm, two improvements and a generalization. Technical Report CIS-Bericht-91-39, Centrum für Informations und Sprachverarbeitung, Universität München, 1991.