Towards Deciding Second-order Unification Problems Using Regular Tree Automata

Tomer Libal

INRIA
`tomer.libal@inria.fr`

The second-order unification problem is undecidable. While unification procedures, like Huet's pre-unification, terminate with success on unifiable problems, they might not terminate on non-unifiable ones. There are several decidability results for infinitary unification, such as for monadic second-order problems. These results are based on the regular structure of the solutions of these problems and by computing minimal unifiers.

Beyond the importance of the knowledge that searching for unifiers of decidable problems always terminates, one can also use this information in order to optimize unification algorithms, such as in the case for pattern unification [6].

Nevertheless, being able to prove that the unification problem of a certain class of unification constraints is decidable is far from easy. Some results were obtained for certain syntactic restrictions on the problems (see Levy [4] for some results and references) or on the unifiers (see Schmidt-Schauß [7] and Schmidt-Schauß and Schulz [8, 9] for some results).

Infinitary unification problems, like the ones we are considering, might suggest that known tools for dealing with the infinite might be useful. One such tool is the regular tree automaton. The drawback of using regular automata for unification is, of course, their inability to deal with variables. In this talk we try to overcome this obstacle and describe an on-going work about using regular tree automata [1] in order to decide more general second-order unification problems.

The second-order unification problems we will consider are of the form $\lambda\overline{z_n}.x_0 t \doteq \lambda\overline{z_n}.C(x_0 s)$ where C is a context [2] and x_0 does not occur in t or s. We will call such problems *cyclic problems*. A sufficient condition for the decidability of second-order unification problems was given by Levy [4]. This condition states that if we can never encounter, when applying Huet's pre-unification procedure [3] to a problem, a cyclic equation, then the unification problem is decidable.

It follows from this result that deciding second-order unification problems depends on the ability to decide cyclic problems. The rules of Huet's procedure ($\mathtt{PUA}$) are given in Fig. 1. Imitation partial bindings and projection partial bindings are defined in [10] and are denoted, respectively, by $\mathtt{PB}(f,\alpha)$ and $\mathtt{PB}(i,\alpha)$ where α is a type, Σ a signature $f \in \Sigma$ and $0 < i$.

The following technical definitions, taken from our previous work on extending $\mathtt{PUA}$ to deal with some non-termination [5], describe the change in the unification constraints set when we start with a cyclic problem and execute certain rules of $\mathtt{PUA}$.

Let e be a cyclic equation as above where $C = C_1 \dots C_m$ such that for all $0 < i \leq m$, $C_i = f_i(r_i^1, \dots, [.], \dots, r_i^{n_i})$ where $n_i = \mathtt{arity}(f_i) - 1$. Define also, for all $m < i$, $C_i = f_k(y_{i-m}^1 s, \dots, [.], \dots, y_{i-m}^{n_k} s)$ where $k = ((i-1) \mod m) + 1$ and y_{i-m}^j for $0 < j \leq n_k$ are new variables. We define the progressive context D_i^e for all $0 \leq i$ as $D_i^e = C_{i+1} \dots C_{i+m}$.

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

In order to clarify the definitions, we will use the following (non-unifiable) cyclic equation as an example: $x_0 f(a,a) \doteq f(x_0 a, f(f(a,a),b))$. Note that $\mathtt{PUA}$ does not terminate on this problem.

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

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

$$\frac{S \cup \{x \doteq u, \lambda\overline{z_k}.x^{\alpha}(\overline{s_n}) \doteq \lambda\overline{z_k}.f(\overline{t_m})\} \qquad u \in \texttt{PB}(f,\alpha)}{S \cup \{\lambda\overline{z_k}.x^{\alpha}(\overline{s_n}) \doteq \lambda\overline{z_k}.f(\overline{t_m})\}}\ \texttt{(Imitate)}^1$$

$$\frac{S \cup \{x \doteq u, \lambda\overline{z_k}.x^{\alpha}(\overline{s_n}) \doteq \lambda\overline{z_k}.a(\overline{t_m})\} \qquad 0 < i \leq k, u = \texttt{PB}(i,\alpha)}{S \cup \{\lambda\overline{z_k}.x^{\alpha}(\overline{s_n}) \doteq \lambda\overline{z_k}.a(\overline{t_m})\}}\ \texttt{(Project)}^2$$

1. where $f \in \Sigma$.
2. where either $a \in \Sigma$ or $a = z_i$ for some $0 < j \leq k$.

Figure 1: `PUA`- Huet's pre-unification procedure

For the example, $t = f(a,a), s = a, C = f([.], f(f(a,a),b)), m = 1, k = 1, n_i = 1$, and $r_i^1 = f(f(a,a),b)$ for all $0 < i$. The progressive contexts for this example are: $D_0 = C, D_1 = f([.], y_1 a), D_2 = f([], y_2 a)$, etc.

Since the only "don't know" non-determinism in `PUA` is due to the choices in the search between the rules `(Imitate)` and `(Project)` [10], we can follow the execution of `PUA` on the cyclic equation e using the search tree in Figure 2 and using the following definitions. Given a cyclic equation e, for all $0 \leq i$, we define $\mathfrak{I}(i)$, $\mathfrak{I}^*(i)$ and $\mathfrak{P}(i)$ inductively as follows:

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

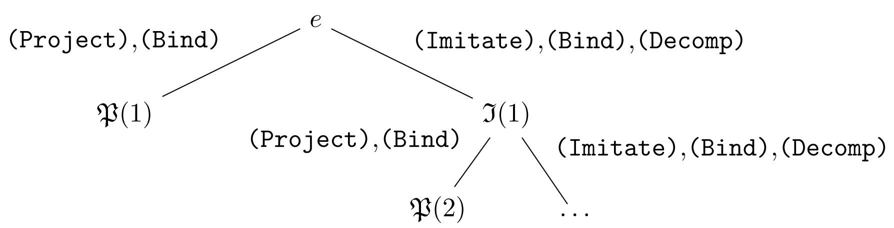

Figure 2: The "don't-know" non-determinism in `PUA`

From the fact that `PUA` is complete for higher-order unification [3] and from the fact that e is a cyclic problem, it follows that e is unifiable iff there is $0 < i$ such that $\mathfrak{P}(i)$ is unifiable [5].

For the example, we have $\mathfrak{P}(1) = \{f(a,a) \doteq f(f(a,a),b)\}$, $\mathfrak{P}(2) = \{y_1 f(a,a) \doteq f(f(a,a),b), f(a,a) \doteq f(a, y_1 a)\}$, $\mathfrak{P}(3) = \{y_1 f(a,a) \doteq f(f(a,a),b), y_2 f(a,a) = y_1 a, f(a,a) \doteq f(a, y_2 a)\}$, etc.

Let $\mathfrak{P}^-(i) \subseteq \mathfrak{P}(i)$ be the following set of equations.

- if $0 < i \le m+1$ then $\mathfrak{P}^-(i) = \mathfrak{P}(i)$.
- if $m+1 < i$ then $\mathfrak{P}^-(i) = \mathfrak{I}^*(m) \cup \{\lambda\overline{z_n}.t \doteq \lambda\overline{z_n}.D^e_{i-1}(s)\}$.

and let $\mathfrak{P}^*(i) = \mathfrak{P}(i) \setminus \mathfrak{P}^-(i)$.

The corresponding values for our example are $\mathfrak{P}^-(1) = \mathfrak{P}(1), \mathfrak{P}^-(2) = \mathfrak{P}(2)$, $\mathfrak{P}^-(3) = \{y_1 f(a,a) \doteq f(f(a,a),b), f(a,a) \doteq f(a, y_2 a)\}$ and $\mathfrak{P}^*(3) = \{y_2 f(a,a) = y_1 a\}$.

Clearly, $\mathfrak{P}(i)$ is unifiable only if both $\mathfrak{P}^-(i)$ and $\mathfrak{P}^*(i)$ are (using the same substitution). In [5] we proved that $\mathfrak{P}^-(i)$ is unifiable only if there is $0 < j \le 3m \le i$ such that $\mathfrak{P}^-(j)$ is unifiable. I.e. that we can always decide if there is an $0 < i$ such that $\mathfrak{P}^-(i)$ is unifiable.

In the present talk, we will describe how to decide the unification problems of these cycles under a very strong restriction. We require that the generated sets $\mathfrak{P}^-$ are each finitary unification problems, meaning that a complete set of their unifiers cannot be infinite. This condition can be achieved, for example, if we require these sets to be acyclic. In this case this restriction holds following the proof of the decidability of acyclic problems [4]. The cyclic problems in our case are composed of exactly one cycle and we will show that one can decide the unification problem in this case.

In order to decide if such cyclic equations are unifiable, we will first investigate the equations in $\mathfrak{P}^-(i)$ and $\mathfrak{P}^*(i)$ and the relation between them.

Let us consider $\mathfrak{P}(i)$ for some $0 < i$ and let us pick an arbitrary equation $\lambda\overline{z_{n_1}}.y^j_k t \doteq \lambda\overline{z_{n_1}}.r^j_k \in \mathfrak{P}^-(i)$ where $0 < k \le m$ and j is some index depending on the arity of the enclosing function symbol in C. This equation is connected to the following set of equations in $\mathfrak{P}^*(i)$: $\{\lambda\overline{z_{n_2}}.y^j_{k+m} t \doteq \lambda\overline{z_{n_2}}.y^j_k s, \lambda\overline{z_{n_3}}.y^j_{k+2m} t \doteq \lambda\overline{z_{n_3}}.y^j_{k+m} s, \ldots, \lambda\overline{z_{n_l}}.y^j_{\frac{i-1}{m}} t \doteq \lambda\overline{z_{n_l}}.y^j_{\frac{i-1}{m}-m} s\}$. The last occurrence of a variable in this chain is the occurrence of $y^j_{\frac{i-1}{m}} s$ in the equation $\lambda\overline{z_{n_{l+1}}}.t \doteq \lambda\overline{z_{n_{l+1}}}.D^e_{i-1}(s) \in \mathfrak{P}^-(i)$. Call the equations from $\mathfrak{P}^-(i)$ base equations and the ones from $\mathfrak{P}^*(i)$ inductive equations. In the following algorithm we will consider first the finitely-many $\mathfrak{P}(j)$ problems for $0 < j \le 3m$ and will then consider the problems $\mathfrak{P}(i)$ such that $\mathfrak{P}^-(i)$ is unifiable iff $\mathfrak{P}^-(j)$. We will call the infinitely-many such problems $\mathfrak{P}(i)$ the extensions of $\mathfrak{P}(j)$.

Since in our example $m = 1$, there can be only one chain. For $\mathfrak{P}(i)$ the chain is just the sequence of equations in $\mathfrak{P}(i)$.

In the current talk we will consider only the case when there is one chain and renumber the indices of the y variables with $1, \ldots, p$. Since the chain and the problem $\mathfrak{P}(i)$ are the same, we will consider both the problems $\mathfrak{P}(i)$ as extensions of $\mathfrak{P}(j)$ and the (single) chains of $\mathfrak{P}(i)$ as extensions to the chain in $\mathfrak{P}(j)$.

In order to define the algorithm, we need first to define how to construct the regular tree automaton based on three terms. The first term will be of the form $\lambda z.u$ where u can contains z but no subterm of the form v, the second term will be v and the third term will be w such that w contains occurrences of z and has no subterm of the form v. Using these three terms, we define the following tree automaton $A = (Q, Q_f, \Delta)$ where $Q = \{q_w, q_u\}, Q_f = \{q_u\}$ and Δ is defined as follows:

- $\lambda z.u_1 \to q_u(\lambda z.u_2)$ where u_1 is obtained from u by replacing each occurrence of z with $q_w(x_l)$ where l is a new index for each occurrence. u_2 is obtained in the same way but we replace each occurrence of z with x_l.
- $w_1 \to q_w(w_2)$ where w_1 and w_2 are obtained from w in the same way u_1 and u_2 were obtained from u,
- $v \to q_w(v)$.

- $z \to q_w(z)$.

We describe this automaton using $\texttt{aut}(u, v, w)$.

The idea behind this construction is that the language accepted by this automaton is exactly the one containing all the possible mappings for y_1 in the chain according to the arbitrarily-many constraints in the inductive part of the chain and based on a given unifier σ for the base part of the chain. Given this automaton, we need just to test if $\sigma(y_1)$ is recognized by it. Note that the base equations contain only occurrences of the variables y_1 and y_p and therefore, a unifier for the base part poses no constraints on the values for the variables $y_2, \ldots, y_{p-1}$ in unifiers for the inductive part and we can freely generate all possible substitutions.

We describe next an algorithm for deciding our cyclic problems.

1. given a cyclic problem e.
2. compute the set of unifiers for some $\mathfrak{P}^-(j)$ where $0 < j \leq 3m$.
3. let σ be such a unifier and for the single chain in $\mathfrak{P}^-(j)$ do the following:
   (a) let $\lambda z.u$ be obtained from $\sigma(y_p)$ by replacing all occurrences of $s\sigma$ with z..
   (b) let $v = s\sigma$.
   (c) let w be obtained from $t\sigma$ by replacing all occurrences of $s\sigma$ in it with z.
   (d) fail if $\sigma(y_1)$ is not recognized by $\texttt{aut}(u, v, w)$.

The algorithm tries to find one unifier of a $\mathfrak{P}^-(j)$ which can be extended in order to unify some $\mathfrak{P}^*(i)$ for an arbitrary $0 < i$.

The correctness of the above algorithm is based on the following theorem.

Theorem 1. Given a problem e, $0 < j \leq 3m$, a chain in $\mathfrak{P}(j)$ over variables $y_1, \ldots, y_p$ and a unifier σ of $\mathfrak{P}^-(j)$, $\sigma(y_1)$ is recognized by $\texttt{aut}(u, v, w)$ as above iff there is an extension $\mathfrak{P}(i)$ of $\mathfrak{P}(j)$ and a substitution θ, such that θ unifies $\mathfrak{P}(i)$.

Proof. Proof sketch: the automata for the chain and all its extensions are the same.

- if - by induction on the number of the variables $y_1, \ldots, y_q$ in the extension $\mathfrak{P}(i)$ of $\mathfrak{P}(j)$. For the step we need to prove that the first equation in the chain (which is determined last since the base of the terms in the language are determined by $\theta(y_q)$) is unifiable only if $\sigma(y_1)$ is recognized by the automaton. Since θ extends σ, we have $\theta(y_1) = \sigma(y_1)$ and by assuming that $\theta(y_2)$ is recognized, the rest follows from the definition of the automaton.
- only if - for this direction, we need to choose the extension $\mathfrak{P}(i)$ and build the substitution θ. This is computed by considering the accepting sequence of transitions for $\sigma(y_1)$. The maximal number of nested transitions determines the number of equations in the chain while the transitions themselves determines the values $\theta(y_2), \ldots, \theta(y_{q-1})$ in the chain. In addition, $\theta(y_1) = \sigma(y_1)$ and $\theta(y_q) = \sigma(y_p)$.

□

We will now demonstrate this idea on the example. A unifier σ for $\mathfrak{P}^-(2)$ is $[y_2 \mapsto \lambda z.z, y_1 \mapsto \lambda z.f(z, b)]$. Note that there is no $0 < i$ such that we can extend this unifier to a unifier of $\mathfrak{P}^*(i)$. Let $u = \lambda z.z, v = a$ and $w = f(z, z)$. $\texttt{aut}(u, v, w) = (Q, Q_f, \Delta)$ where $Q = \{q_w, q_u\}, Q_f = \{q_u\}$ and Δ is defined as follows:

- $\lambda z.q_w(x) \to q_u(\lambda z.x)$.
- $f(q_w(x_1), q_w(x_2)) \to q_w(f(x,y))$.
- $a \to q_w(a)$.
- $z \to q_w(z)$.

Clearly $\sigma(y_1)$ cannot be generated by $\texttt{aut}(u,v,w)$ and therefore we have proved that there is no possible extension of σ. By doing the same to all unifiers of $\mathfrak{P}^-(1), \mathfrak{P}^-(2)$ and $\mathfrak{P}^-(3)$, we can prove that the example is not unifiable.

To summarize, we have described a decision algorithm for cyclic second-order unification problems. The main novelty of the method is its use of tree automata in order to decide unification problems. At the same time, the exact form of the cycles (beyond the simple ones discussed here) and the possibility to treat problems with more than one chain are still investigated.

References

[1] H Comon, M Dauchet, R Gilleron, F Jacquemard, D Lugiez, S Tison, and M Tommasi. Tree automata techniques and applications, 1999.

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

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

[4] Jordi Levy. Decidable and undecidable second-order unification problems. In *RTA*, pages 47–60, 1998.

[5] Tomer Libal. Regular patterns in second-order unification. 2015. to appear. http://logic.at/staff/shaolin/papers/holunif.pdf.

[6] Dale Miller. Unification of simply typed lambda-terms as logic programming. In *In Eighth International Logic Programming Conference*, pages 255–269. MIT Press, 1991.

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

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

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

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