On the Complexity of Bounded Second Order Unification

Manfred Schmidt-Schauß

Fachbereich Informatik, Johann Wolfgang Goethe-Universität,
Postfach 11 19 32, D-60054 Frankfurt,Germany
Tel: (+49)69-798-28597, Fax: (+49)69-798-28919,
E-mail: `schauss@ki.informatik.uni-frankfurt.de`
URL: `www.ki.informatik.uni-frankfurt.de/`

1 Overview

It is well-known that first order unification is decidable in linear time (for further information see the overview article [BS94]) whereas second order (and higher-order) unification is undecidable [Gol81]. This result was sharpened in [Far91] for a restricted signature and in [Lev98, LV00] for severe restrictions on occurrences of second order variables.

Bounded second order unification (BSOU) is second order unification under the restriction that only a bounded number of holes in the instantiating terms for second order variables is permitted, however, the size of the instantiation is not restricted.

We report on work in progress and argue that the decision algorithm in [SS99] for bounded second order unification can be modified such that it runs in nondeterministic polynomial time. Together with the result that it is $\mathcal{NP}$-hard, this show that bounded second order unification is $\mathcal{NP}$-complete.

This also shows that monadic second order unification (MSOU), which was also treated in [Far88] is $\mathcal{NP}$-complete. Since MSOU can be easily encoded as word unification, this compares favorably to the best known upper bounds NEXPTIME [Gut98], and PSPACE [Pla99] for word unification.

This supports the claim that bounded second order unification is easier than context unification, whose decidability is currently an open question.

There are two key results to obtain this sharp bound for the complexity of bounded second order unification:
One is the result on the exponential upper bound of the exponent of periodicity for a minimal unifier of a word-unification problem and of a context unification problem, respectively [Mak77, KP96, SSS98]. This upper bound can be represented in polynomial space.
The other key result is in a paper of Plandowski [Pla94], where it is shown that the following equality test is polynomial: Given two context free grammars G_1, G_2 where $L(G_1) = \{t_1\}, L(G_2) = \{t_2\}$. The question whether $t_1 = t_2$ can be answered in polynomial time depending on the size of the grammar (not the words). This for example shows that the equality test of two words with nested natural exponents can be done in polynomial time. Note that the represented words may be of exponential size.

For example, this allows a polynomial test for checking the validity of

$$((ab)^{1001}a(ba)^{2001}b)^3 = ((aba(ba)^3b(ab)^4)^7)^{143}.$$

The idea for applying Plandowskis theorem is to encode the powers as a series of non-terminals. E.g. a^{15} would be encoded as $E_1 = a, E_2 = E_1E_1, E_3 = E_2E_2, E_4 = E_3E_3, E = E_4E_3E_2E_1$.

The exploitation of similar ideas led to a PSPACE upper bound for stratified context unification [SS01b].

2 Modifications of the Algorithm and Complexity

Bounded second order unification is the following question: Given equations $s_i \doteq t_i, i = 1, \ldots, m$ where s_i, t_i are second order terms, and an integer n, are the equations $s_i \doteq t_i, i = 1, \ldots, m$ unifiable, under the restriction that for every second order variable X, the sum of the occurrences of arguments used in the instantiations of X in a unifier is bounded by n.

As shown in [SS99], this can easily be transformed in non-deterministic polynomial time into a similar question, where the condition is sharpened such that every second order variable has arity one and the argument is used at most once in the instantiations. The second order variables under this restriction are called *Z-context variables* (zero or one instantiation).

The main modification of the algorithm in [SS99] is to permit a kind of nested powers: like $(f([], a)^4 g(b, []))^3)^2$. This avoids the explicit generation of large terms and in particular an exponential explosion of space usage. Instead it represents large terms implicitly. The price to pay is an algorithm with more rules which, unfortunately, are also more complex.

Furthermore there are two kinds of equations, term equations, and context equations $U := S$, where U is a context variable, and the context equations are only used for sharing purposes. There is also sharing on the term level, such that every subterm is represented by a first order variable. Instead of simple powers we permit expressions $pow(U, n_1, n_2)$, where $n_2 \geq n_1$ are nonnegative integers and U is a context variable for which there is a sharing equation. The reason is that several rules are easier to express using the generalized power expressions.

There are basically three situations for the non-deterministic algorithm in [SS99]:

1. Every term in an equation is either a variable or the top symbol is a Z-context variable. This is a presolved form, and the set of equations is unifiable. This situation is the same for the new algorithm.
2. There are so-called surface cycles like $X(a) \doteq y_1, y_1 \doteq f(x, y), y \doteq X(b)$. In this case one possibility of the algorithm is to replace X by $f(x, [])^k$, where k is smaller than the upper bound for the exponent of periodicity for the input problem. The modified algorithm adds $U_1 := f(x, []), U_2 := pow(U_1, 0, k)$. The equations above are replaced by $pow(U_1, 0, k)(a) \doteq y_1, y_1 \doteq f(x, y), y = pow(U_1, 0, k)(b)$.
   Note that there may be longer surface cycles and that there are further selection possibilities.
3. There are no surface cycles. Then it is necessary e.g. to make a generalized decomposition of terms, in particular those that have power-expressions at the top. The theorem of Plandowski is of help here to show that the macro decomposition steps can be done in non-deterministic polynomial time. This means, there is a guessing step, and a subsequent test that requires polynomial time. If we only used a top-down, single-step decomposition as in [SS99], then the number of steps would be exponential in the worst case.

Let us make an example for this decomposition macro step. The macro step may require a variable identification before application. Assume there are equations

$$
\begin{aligned}
&y_1 \doteq pow(U_1, 0, 1000)(x_1), y_1 \doteq pow(U_3, 1, 2000)(x_2), y_1 \doteq X(x_3),\\
&\ldots,\\
&U_1 := pow(U_2, 2, 100), U_2 := f(x_5, [])(g([], x_6))\\
&U_3 := pow(U_4, 0, 200), U_4 := g([], x_6)f(x_5, [])
\end{aligned}
$$

Then a guessing step consists of guessing the depth up to which the same path can be used, and then of guessing the different continuations of the path. The representation of the guess may be as follows:

$$X \to pow(U_3, 1, 500)f([], x_7)X'$$

Then the following algorithmic steps have to be performed:
A test whether the paths are indeed the same. I.e., whether $pow(U_1, 0, 499)$ and $pow(U_3, 1, 500)$ are the same contexts. Here the polynomial upper bound on the time complexity can be used.
The second step is to throw away the start of the path, and to compute the tail. The interesting part of the resulting system is:

$$
\begin{aligned}
&y_2 \doteq pow(U_1, 499, 1000)(x_1), y_2 \doteq pow(U_3, 500, 2000)(x_2),\\
&y_2 \doteq f(x_8, x_7), x_8 \doteq X'(x_3), \ldots
\end{aligned}
$$

The next decomposition step will reveal that $x_8 = x_5$.

Further changes to the algorithm are more or less straightforward adaptions to the modified data structures.

3 Conclusion

The claim is that bounded second order unification is $\mathcal{NP}$-complete based on extending the decision algorithm by nested power expressions and using Plandowskis result.

The author is sure that the same method of modifying the decision algorithm in [SS01a] will prove that stratified context unification is in $\mathcal{NP}$.

References

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

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

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

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

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

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

[Lev98] Jordi Levy. Decidable and undecidable second order unification problems. In *Proceedings of the 9th Int. Conf. on Rewriting Techniques and Applications*, volume 1379 of *Lecture Notes in Computer Science*, pages 47–60, 1998.

[LV00] Jordi Levy and Margus Veanes. On the undecidability of second-order unification. *Information and Computation*, 159:125–150, 2000.

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

[Pla94] W. Plandowski. Testing equivalence of morphisms in context-free languages. In *ESA 94*, volume 855 of *Lecture Notes in Computer Science*, pages 460–470, 1994.

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

[SS99] Manfred Schmidt-Schauß. Decidability of bounded second order unification. Technical Report Frank-report-11, FB Informatik, J.W. Goethe-Universität Frankfurt am Main, 1999. submitted for publication.

[SS01a] Manfred Schmidt-Schauß. A decision algorithm for stratified context unification. *Journal of Logic and Computation*, 2001. accepted for publication.

[SS01b] Manfred Schmidt-Schauß. Stratified context unification is in pspace. In *Proceedings of CSL'01*, 2001. to appear.

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