32nd International Workshop on Unification (UNIF 2018), Informal Proceedings
Oxford, 7th July, 2018

Compressed Term Unification: Results, applications, open problems, and hopes.

Adrià Gascón[1]

Warwick University and The Alan Turing Institute, UK
agascon@turing.ac.uk

Abstract

Already in the classic first-order unification problem, the choice of a suitable formalism for term representation has a significant impact in computational efficiency. One can find other instances of this situation in some variants of second-order unification, where representing partial solutions efficiently leads to better algorithms. In this talk I will present some compression schemes for terms and discuss computational complexity results for variants of first and second-order unification on compressed terms. I'll show how these results build up on each other and discuss open problems in compressed term unification, as well as potential approaches to solutions.

1 Introduction

The problem of checking satisfiability of a set of equations plays a central role in any mathematical science. From the perspective of computer science, a lot of effort is devoted to finding efficient decision procedures for different families of equations. The problem of *satisfiability of word equations*, also known as *Word Unification (WU)*, figures prominently as one of the most intriguing problems of that form. The first algorithm for that problem was given by Makanin [26], and the best known upper bound (PSPACE) is due to Plandowski [31]. On the other hand, its PSPACE-hardness is an open question. Several particular cases of that problem, such as the ones that result from fixing the number of variables in the equations to a constant, have also been studied. For instance, efficient algorithms for satisfiability of word equations with one [6, 29, 18] and two [5] variables have been discovered.

Another fundamental operation in symbolic computation systems is the well-known *first-order unification problem*. This problem consists of solving equations of the form $s \doteq t$, where s and t are terms with first-order variables. The goal is to find a mapping from variables to (first-order) terms that would make the terms s and t syntactically equal. This problem was firstly introduced as such in the work by J.A. Robinson, which established the foundations of automated theorem proving and logic programming. More concretely, Robinson presented in [33] a procedure to determine the validity of a first-order sentence that has term unification as its main ingredient. Later, term unification was also used by Knuth and Bendix as a key component of their critical pairs method to determine local confluence of term rewrite systems (see [1] for a general survey on unification theory). The syntactic unification and matching

problems were deeply investigated in the last century. Among other results, linear time algorithms were discovered [27, 30]. Moreover, more expressive variants of term unification such as *unification modulo theories* have also drawn a lot of attention. In this notion of term unification, equality between terms is interpreted under equational theories such as associativity, commutativity, and distributivity, among others [1].

An interesting connection between word and term unification is the *Context Unification (CU)* problem. In CU, the terms s, t in the equation $s \doteq t$ may contain context variables. For example, consider the equation $F(f(x, b)) \doteq f(a, F(y))$. where x, y are first-order variables ranging over terms and F is a context variable that can be replaced by any context. One of the possible solutions of this instance is the substitution $\{F \mapsto f(a, \bullet), x \mapsto a, y \mapsto b\}$. Note that when we instantiate F by $f(a, \bullet)$ in the equation, replacing the occurrence of $\bullet$ by the argument of F in each of its occurrences, we get $f(a, f(x, b)) \doteq f(a, f(a, y))$, and thus both sides of the equations become equal after applying $\{x \mapsto a, y \mapsto b\}$. Note that, simply using a unary signature, WU reduces to CU. On the other hand, CU is a particular case of second-order unification, which is undecidable [15]. The decidability of CU remained open for a long time, until recently a PSPACE algorithm was presented by Jeż [19].

The example above is in fact an instance of the so-called *one context unification* problem, denoted 1-CU. In 1-CU, only one context variable, possibly with many occurrences, may appear in the input terms. One of the motivations for the study of this problem is its close relationship to interprocedural program analysis [17], whose goal is to compute all simple invariants of imperative procedural programs. Although it is known that 1-CU can be solved in non deterministic polynomial time [10], whether the problem is NP-hard or a polynomial time algorithm exists is an open problem.

Another particular case of context unification is *context matching*. The input of context matching is an equation $s \doteq t$ such that s may contain context variables and first-order variables, and t does not contain variables of any kind. Although it is known to be NP-complete, there are several subcases of context matching that can be solved efficiently [9, 36].

Interesting applications of one context unification and matching arise in the search/extraction of information from tree data structures. For example, a simple matching equation of the form $F(s) \doteq t$, where F is the context variable, t is ground, and s may contain first-order variables but it does not contain occurrences of F, corresponds to searching instances of s within t. Context matching also captures, for example, a conjunctive search of the form $F_1(s_1) \doteq t \wedge \cdots \wedge F_n(s_n) \doteq t$, where the F_is are pairwise different and do not occur elsewhere. These equations correspond to searching for a subterm u_i of t that can be matched by s_i, for every $i \in \{1, \ldots, n\}$; with the additional constraint that variables within the s_is must have a common instance in t, see [16] for the analysis of conjunctive query mechanisms over trees. More generally, multiple occurrences of the same context variable in the term s of a context equation $s \doteq t$, correspond to searching for instances of subterms of t that differ at, at most, one position. This has applications in computational linguistics [28]. It is also easy to encode questions that ask for subtrees that are equal up to several positions.

Term representation

To give a complete description of the problems stated above, one has to precisely state how is the input represented. Besides an *explicit* tree representation for terms, in this talk we will consider two more succinct representations: *Directed Acyclic Graphs* (DAGs) and *Singleton Tree Grammars* (STGs), also known as *tree straight-line program*. Similarly as DAGs allow compression by exploiting the reuse of repeated instances of a subterm in a term, STGs are

Problem	Term Representation Formalism		
	Explicit	*DAG*	*STG*
Context Unification	PSPACE [19]	?	?
Context Matching	NP-Complete	NP-Complete	NP-Complete
One-context Unification	NP [10]	NP [4]	NP [4]
Left-linear One-context Unification (and similar variants)	Ptime	Ptime [14]	?
2-*restricted context Unification*	Ptime	Ptime [13]	?
k-context Matching	Ptime	Ptime	Ptime [7]
First-order Unification	Ptime	Ptime	Ptime [9, 8]
First-order Matching	Ptime	Ptime	Ptime [7]
Equality of unordered ranked trees	Ptime	Ptime	Ptime [24]
Equality of unordered unranked trees	Ptime	Ptime	Ptime [11]

Figure 1: Some recent and classical results in first-order and context unification, for different variants and formalisms for term representation.

a grammar-based compression mechanism based on the reuse of repeated (multi)contexts. An interesting property of the STG formalism is that many operations on terms can be efficiently performed directly in their compressed representation (see [21] for a survey). Some examples are linear subpattern matching [34], i.e. finding instances of a linear terms s within a ground term t, first-order and unification and matching and 1-CU [9, 12, 4], equivalence checking [3], congruence closure [35], and membership in several classes of languages represented by tree automata [22, 25]. Moreover, STG compressors have already been developed and proven useful in practice for XML representation and processing [3, 23], and termination analysis of Term Rewrite Systems [2]. More generally, STGs are a very useful concept for the analysis of unification problems since, roughly speaking, allow to represent solutions in a succinct but still efficiently verifiable form. This observation is related to Jeż's *recompression technique*, which led to many results including the positive answer to decidability of Context Unification mentioned above. In fact, the relationship between compression and solvability of work equations goes back to the work of Plandowski and Rytter [32] and Plandowski's work on word unification [31].

2 Selected Results & Open Problems

Figure 1 collects some classical results in unification. We consider three variants for each problem, depending of whether the input equations are represented explicitly as trees, DAGs, or STGs. The main major open problem in this space has to do with the exact complexity of the context unification problem. While the problem is known to be in PSPACE, as mentioned above, it is only known to be NP-hard. As in fact this is not even known for the simpler case of word unification, that problem is the natural starting point.

Open Problem 1. *Is there a non-deterministic polynomial time decision procedure for WU?*

For the particular case of 1-CU, the main open question is whether there is a polynomial time decision procedure for this problem, as it is only known to be in NP. The problem, for the case where the input is given a DAG, was conjectured to be in P in [14], where many particular

cases were shown to be solvable in polynomial time. In fact, given the results from [14], it suffices to find a polynomial time algorithm to the following reduced form.

Definition 2.1. *An 1-CU instance $\mathcal{I}$ is called* reduced *if it is of the form*

$$\{F(u_i) \doteq x_i \mid i = 1, 2, \ldots\} \cup \{F(v_j) \doteq s \mid j = 1, 2, \ldots\} \\ \cup \ \{F(w_k) \doteq t \mid k = 1, 2, \ldots\} \tag{1}$$

where s, t do not contain F; that is, the right hand-side of the equations have at most two non-variable terms.

Open Problem 2. *Is there a polynomial time decision procedure for reduced 1-CU?*

A particular case of 1-CU that is instrumental in solving the particular cases of [14] is the so-called 2-restricted 1-CU. In this particular case, the context variable occurs at most twice in the input. While we have a polynomial time algorithm for this problem [13] even in the case where the input is given as a DAG, it is not known whether the results in [14] and [13] extend to the case where the input is a STG. In fact, before tackling that more general problem, one should consider a simpler case that was left open in [34]: whether STG-compressed submatching (of nonlinear terms s in t) can be solved in polynomial time or not. This in fact corresponds to solving the 1-CU equation $F(s) = t$, for terms s and t containing arbitrary first-order variables, but not the context variable F.

Open Problem 3. *Is there a polynomial time algorithm STG-compressed term submatching?*

Note that this problem is related with finding a redex of a term rewriting rule in a term. Another open problem with a similar flavour is the one of deciding whether, for STG-compressed terms s and t, and an order $<$ on terms, $s < t$ holds. This problem can be parameterized by the class of the ordering $<$, e.g. lexicographic path orderings or knuth-Bendix orderings. This brings us to our next open problem.

Open Problem 4. *Let s and t be STG-compressed terms, and let $<$ be a KBO ordering on terms, and thus parameterized by a weight function and a precedence order on the signature of s and t. Is there a polynomial time decision procedure for $s < t$?*

All the results regarding STGs from Figure 1, as well as the general unification results that rely on STGs for a succinct representation of sets of solutions of potentially exponential size, critically rely on a polynomial time algorithm for equivalence checking of STG-compressed terms. This classical result by Plandowski, has been recently extended to unordered and unranked trees [11]. A potential extension of such works could consider grammar-based tree compression mechanisms more general than STGs. In particular, STGs do not allow repeated occurrences of parameters and hence, roughly speaking, do not allow *copying*. If we drop that restriction, we obtain nonlinear STGs, a formalism equivalent to Lamping's sharing graphs [20] that allows for a doubly exponential compression ratio [3]. However, the existence of an efficient equivalence checking procedure for nonlinear STGs is open, since the best known complexity upper bound for this problem in PSPACE [3]. I believe that this bound can be improved to co-NP, but the existence of a polynomial time algorithm does not seem plausible since, in the worst-case, $O(2^n)$ bits are needed to simply store the size of a term represented with a non-linear STG of size n. In any case, no hardness result is known.

Open Problem 5. *Is there a non-deterministic polynomial time decision procedure for equivalence checking of nonlinear STGs?*

3 Conclusion

In this talk I will review some of the results from Figure 1 and related techniques, and discuss potential approaches to solve the above open questions.

References

[1] Franz Baader and Wayne Snyder. Unification theory. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, pages 445–532. Elsevier and MIT Press, 2001.

[2] Alexander Bau, Markus Lohrey, Eric Nöth, and Johannes Waldmann. Compression of rewriting systems for termination analysis. In Femke van Raamsdonk, editor, *RTA*, volume 21 of *LIPIcs*, pages 97–112. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013.

[3] Giorgio Busatto, Markus Lohrey, and Sebastian Maneth. Efficient memory representation of XML document trees. *Inf. Syst.*, 33(4-5):456–474, 2008.

[4] Carles Creus, Adrià Gascón, and Guillem Godoy. One-context Unification with STG-Compressed Terms is in NP. In *RTA*, pages 149–164, 2012.

[5] Robert Dabrowski and Wojciech Plandowski. Solving two-variable word equations (extended abstract). In *Automata, Languages and Programming: 31st International Colloquium, ICALP 2004, Turku, Finland, July 12-16, 2004. Proceedings*, pages 408–419, 2004.

[6] Robert Dabrowski and Wojciech Plandowski. On word equations in one variable. *Algorithmica*, 60(4):819–828, 2011.

[7] Adrià Gascón, Guillem Godoy, and Manfred Schmidt-Schauß. Context matching for compressed terms. In *LICS*, pages 93–102, 2008.

[8] Adrià Gascón, Guillem Godoy, and Manfred Schmidt-Schauß. Unification with singleton tree grammars. In *RTA*, pages 365–379, 2009.

[9] Adrià Gascón, Guillem Godoy, and Manfred Schmidt-Schauß. Unification and Matching on Compressed Terms. *ACM Trans. Comput. Log.*, 12(4):26, 2011.

[10] Adrià Gascón, Guillem Godoy, Manfred Schmidt-Schauß, and Ashish Tiwari. Context Unification with One Context Variable. *J. Symb. Comput.*, 45(2):173–193, 2010.

[11] Adrià Gascón, Markus Lohrey, Sebastian Maneth, Carl Philipp Reh, and Kurt Sieber. Grammar-based compression of unranked trees. In *CSR*, volume 10846 of *Lecture Notes in Computer Science*, pages 118–131. Springer, 2018.

[12] Adrià Gascón, Sebastian Maneth, and Lander Ramos. First-order unification on compressed terms. In *RTA*, pages 51–60, 2011.

[13] Adrià Gascón, Manfred Schmidt-Schauß, and Ashish Tiwari. Two-restricted one context unification is in polynomial time. In *CSL*, volume 41 of *LIPIcs*, pages 405–422. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.

[14] Adrià Gascón, Ashish Tiwari, and Manfred Schmidt-Schauß. One context unification problems solvable in polynomial time. In *LICS*, pages 499–510. IEEE Computer Society, 2015.

[15] Warren D. Goldfarb. The undecidability of the second-order unification problem. *Theor. Comput. Sci.*, 13:225–230, 1981.

[16] Georg Gottlob, Christoph Koch, and Klaus U. Schulz. Conjunctive queries over trees. *J. ACM*, 53(2):238–272, 2006.

[17] Sumit Gulwani and Ashish Tiwari. Computing procedure summaries for interprocedural analysis. In *16th European Symposium on Programming Languages and Systems, ESOP 2007, part of the Joint European Conferences on Theory and Practics of Software, ETAPS 2007, Braga, Portugal, March 24 - April 1, 2007*, pages 253–267, 2007.

[18] Artur Jeż. One-variable word equations in linear time. In *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part II*, pages 324–335, 2013.

[19] Artur Jeż. Context unification is in PSPACE. In *Proc. ICALP 2014, Part II*, volume 8573 of *Lecture Notes in Computer Science*, pages 244–255. Springer, 2014.

[20] John Lamping. An algorithm for optimal lambda calculus reduction. In *POPL*, pages 16–30, 1990.

[21] Markus Lohrey. Grammar-based tree compression. In *DLT*, volume 9168 of *Lecture Notes in Computer Science*, pages 46–57. Springer, 2015.

[22] Markus Lohrey and Sebastian Maneth. The complexity of tree automata and XPath on grammar-compressed trees. *Theor. Comput. Sci.*, 363(2):196–210, 2006.

[23] Markus Lohrey, Sebastian Maneth, and Roy Mennicke. XML tree structure compression using repair. *Inf. Syst.*, 38(8):1150–1167, 2013.

[24] Markus Lohrey, Sebastian Maneth, and Fabian Peternek. Compressed tree canonization. In *ICALP (2)*, volume 9135 of *Lecture Notes in Computer Science*, pages 337–349. Springer, 2015.

[25] Markus Lohrey, Sebastian Maneth, and Manfred Schmidt-Schauß. Parameter reduction and automata evaluation for grammar-compressed trees. *J. Comput. Syst. Sci.*, 78(5):1651–1669, 2012.

[26] G. S. Makanin. On the decidability of the theory of free groups (in russian). In *Fundamentals of Computation Theory, FCT '85, Cottbus, GDR, September 9-13, 1985*, pages 279–284, 1985.

[27] Alberto Martelli and Ugo Montanari. An efficient unification algorithm. *ACM Trans. Program. Lang. Syst.*, 4(2):258–282, 1982.

[28] Joachim Niehren, Manfred Pinkal, and Peter Ruhrberg. A uniform approach to underspecification and parallelism. In *ACL*, pages 410–417, 1997.

[29] S. Eyono Obono, Pavel Goralcik, and M. N. Maksimenko. Efficient solving of the word equations in one variable. In *Mathematical Foundations of Computer Science 1994, 19th International Symposium, MFCS'94, Kosice, Slovakia, August 22 - 26, 1994, Proceedings*, pages 336–341, 1994.

[30] Mike Paterson and Mark N. Wegman. Linear unification. *J. Comput. Syst. Sci.*, 16(2):158–167, 1978.

[31] Wojciech Plandowski. Satisfiability of word equations with constants is in PSPACE. *J. ACM*, 51(3):483–496, 2004.

[32] Wojciech Plandowski and Wojciech Rytter. Application of lempel-ziv encodings to the solution of words equations. In *ICALP*, volume 1443 of *Lecture Notes in Computer Science*, pages 731–742. Springer, 1998.

[33] John Alan Robinson. A Machine-Oriented Logic Based on the Resolution Principle. *J. ACM*, 12(1):23–41, 1965.

[34] Manfred Schmidt-Schauß. Linear Compressed Pattern Matching for Polynomial rewriting (extended abstract). In *TERMGRAPH*, pages 29–40, 2013.

[35] Manfred Schmidt-Schauß, David Sabel, and Altug Anis. Congruence closure of compressed terms in polynomial time. In *FroCoS*, pages 227–242, 2011.

[36] Manfred Schmidt-Schauß and Jürgen Stuber. The complexity of linear and stratified context matching problems. *Theory Comput. Syst.*, 37(6):717–740, 2004.