Recompression: New Approach to Word Equations and Context Unification

Artur Jeż

Institute of Computer Science, University of Wrocław, Wrocław, Poland

Abstract

Word equations is given by two strings over disjoint alphabets of letters and variables and we ask whether there is a substitution that satisfies this equation. Recently, a new PSPACE solution to this problem was proposed, it is based on compressing simple substrings of the equation and modifying the equation so that such operations are sound. The analysis focuses on the way the equation is stored and changed rather than on the combinatorics of words. This approach greatly simplified many existing proofs and algorithms. In particular, unlike the previous solutions, it generalises to equations over contexts (known for historical reasons as context unification): contexts are terms with one special symbol that represent a missing argument and they can be applied on terms, in which case their argument replaces the special constant.

1998 ACM Subject Classification F.4.3 [Formal Languages] Decision Problems, F.4.2 Grammars and Other Rewriting Systems, F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases Word equations, exponent of periodicity, semantic unification, string unification, context unification, compression

Digital Object Identifier 10.4230/LIPIcs.STACS.2017.2

Category Invited Talk

1 Recompression for word equations

Word equations is given by two strings (U, V) over disjoint alphabets of letters (Σ) and variables $(\mathcal{X})$. We are to decide, whether there is a substitution $S : \mathcal{X} \to \Sigma^*$ which turns U and V into equal strings over Σ. This problem was first solved by Makanin [6], both the algorithm and its analysis employs nontrivial facts from word combinatorics. It took 20 years for a different solution to emerge, it was proposed by Plandowski [7]. It is based on a different approach, still uses nontrivial word combinatorics concepts and additionally uses compression; this algorithm works in PSPACE, which is the best computational complexity bound known till today, and the problem is only known to be NP-hard.

The recompression approach [5], which I will survey in this talk, takes a different approach. It focuses mostly on the compression and on the way equation is represented, and not at all on the word combinatorics. On a high level we want to perform two compression operations on the sides of the equation after applying the substitution to variables, which we denote by $S(U)$ and $S(V)$:

- the a-block compression replaces every maximal block a^ℓ with a (fresh) letter a_ℓ (an a-block is a subsequence consisting of letters a alone and that cannot be extended by a nor to the left, nor to the right);
- the ab pair compression, where $a \neq b$, replaces every occurrence of a pair ab with a fresh letter c.

For instance, given a word $aaababababaaabaa$ the ab pair compression returns $aacccaacaa$ while a block compression: $a_3bababa_3ba_2$.

In order to apply those operations, we may need to process the equation appropriately: given a solution S we say that ab is a *crossing pair* (with respect to S) if there is an occurrence of ab in $S(U)$ or $S(V)$ that does not fully come from the equation or from a substitution for one occurrence of a variable. A pair is *noncrossing* (with respect to S) otherwise. If ab is noncrossing with respect to a solution S of an equation $U = V$, then performing ab-compression on sides of the equation results in an equation $U' = V'$ that has a corresponding solution S', which is obtained by also replacing every ab in $S(X)$ by c. Then $S'(U')$ and $S'(V')$ are obtained from $S(U)$ and $S(V)$ by ab-pair compression, as desired.

If ab is crossing, we *uncross it*: for every variable X we nondeterministically guess, whether $S(X)$ begins with b and if so, we replace X with bX and whether $S(X)$ ends with a, in which case we replace X with Xa. It is easy to show that after doing so for all variables, the pair ab is noncrossing, and so it can be compressed.

We treat blocks similarly: when performing the a block compression, we look at maximal blocks of a in $S(U)$ and $S(V)$ that come partially from the equation and partially from substitution for variables. This time, though, adding one letter may be not enough, we replace X with $a^\ell X a^r$, where $S(X) = a^\ell w a^r$ and w does not begin, nor end, with a. Again, performing this for all variables makes it possible to perform the a-block compression.

It is easy to see that our procedure is sound and complete, what remains is a termination proof, we show it by proving a $4n^2$ bound on stored equation (for appropriate non-deterministic choices). To this end we give a strategy for choosing a pair/block to compress: If there is a non-crossing pair or a without crossing blocks, we perform the corresponding compression. If not, then we choose the pair/block after compression of which the equation is the shortest.

Observe that if occurrences of a pair/blocks of a letter use α letters in the equation, then compression of this pair/blocks of a letter removes at least $\alpha/2$ letters from the equation. On the other hand, each uncrossing introduces at most 2 letters per variable, so at most $2n$ in total. As each crossing pair/letter with a crossing block corresponds to a left or right side of an occurrence of a variable in the equation, in total there are at most $2n$ different crossing pairs and letters with crossing block. Thus if the equation has length m and it has no noncrossing pairs and no letters without crossing blocks, for some crossing pair/letter with a crossing block the length of the equation after corresponding the compression is at most $m' = \left(1 - \frac{1}{2n}\right) m + 2n$. It is easy to check that if $m \leq 4n^2$ then also $m' \leq 4n^2$.

2 Context unification

When one considers terms instead of words, equations become more involved and the first question is: What can the variables represent? If we allow the variables to represent only ground (closed) terms, then we end up with *first order unification*, for which the polynomial time algorithm is folklore. Allowing the variables to represent functions leads to second order unification, which is undecidable [3].

Context unification [1, 2, 8] is a fragment of second-order unification: A *context* is a ground term with exactly one occurrence of a special constant '•' that represents a missing argument; one should think of '•' as a 'hole' to be instantiated by a ground term later on: Contexts can be applied to ground terms, which results in a replacement of '•' by the given ground term; similarly we can define a composition of two contexts, which is again a context. In *context unification* we allow the variables to represent contexts; syntactically, the variables are treated as unary function symbols. A solution assigns to each (context) variable a context such that both sides of the equation evaluate to the same (ground) term. For ease of use, we also allow ground variables that represent ground terms, syntactically they are treated as constants and the solution substitutes them with ground terms.

The decidability status of context unification was unknown for years, as on one hand the undecidability proofs for second order unification essentially used the fact that argument can be used unbounded number of times, and on the other hand the combinatorial properties of strings used by algorithms of Makanin and Plandowski had no equivalent in the term case.

The recompression approach and the corresponding algorithm generalise naturally to the term case: the two compression operations can be readily applied to consecutive unary nodes. One still needs additional operation: leaves compression. The (f, i, c) *leaves compression* replaces each subterm $f(t_1, \ldots, t_{i-1}, c, t_{t+1}, \ldots, t_m)$ with $f(t_1, \ldots, t_{i-1}, t_{t+1}, \ldots, t_m)$. All notions from pairs and blocks generalise to this new compression operation: The crossing (f, i, c) is defined similarly, though the corresponding uncrossing is more involved: on one hand, if a (ground) variable x has a solution $S(x) = c$ and it has an occurrence in which x is the i-th child, then we replace x with c. On the other hand, when in $S(X)$ the symbol above '$\bullet$' is f and '$\bullet$' is the i-th child, we replace X with $X(f(x_1, \ldots, x_{i-1}, \bullet, x_{i+1}, \ldots, x_m))$, where $x_1, \ldots, x_{i-1}, x_{i+1}, x_m$ are fresh variables.

The algorithm is similar as before: we guess a compression, perform the uncrossing if needed and then compress. With appropriate choices the equation stays polynomial. Some additional analysis is needed to bound the number of introduced fresh variables. Thus the satisfiability of context unification is in PSPACE [4]; the lower bound is NP, as for words' case.

References

1 Hubert Comon. Completion of rewrite systems with membership constraints. Part I: Deduction rules. *J. Symb. Comput.*, 25(4):397–419, 1998. doi:10.1006/jsco.1997.0185.

2 Hubert Comon. Completion of rewrite systems with membership constraints. Part II: Constraint solving. *J. Symb. Comput.*, 25(4):421–453, 1998. doi:10.1006/jsco.1997.0186.

3 Warren D. Goldfarb. The undecidability of the second-order unification problem. *Theor. Comput. Sci.*, 13:225–230, 1981. doi:10.1016/0304-3975(81)90040-2.

4 Artur Jeż. Context unification is in PSPACE. In Elias Koutsoupias, Javier Esparza, and Pierre Fraigniaud, editors, *ICALP*, volume 8573 of *LNCS*, pages 244–255. Springer, 2014. full version at http://arxiv.org/abs/1310.4367. doi:10.1007/978-3-662-43951-7_21.

5 Artur Jeż. Recompression: a simple and powerful technique for word equations. *Journal of the ACM*, 63(1):4:1–4:51, Feb 2016. doi:10.1145/2743014.

6 Gennadiĭ Makanin. The problem of solvability of equations in a free semigroup. *Matematicheskii Sbornik*, 2(103):147–236, 1977. (in Russian).

7 Wojciech Plandowski. Satisfiability of word equations with constants is in PSPACE. *J. ACM*, 51(3):483–496, 2004. doi:10.1145/990308.990312.

8 Manfred Schmidt-Schauß. Unification of stratified second-order terms. Internal Report 12/94, Johann-Wolfgang-Goethe-Universität, 1994.