On a Problem from Set Constraints, Automata Theory, Model Checking and Program Analysis

Witold Charatonik Andreas Podelski
Jean-Marc Talbot
Max-Planck-Institut für Informatik
Im Stadtwald, D-66123 Saarbrücken, Germany
{witold;podelski;talbot}@mpi-sb.mpg.de

Set constraints. Set constraints denote relations between sets of trees. Syntactically, they are conjunctions of inclusions between expressions built over variables, constructors (constants and function symbols from a given alphabet) and a choice of set operators that defines the specific class of set constraints. Their main application domain is set-based program analysis.

Definite set constraints were introduced by Heintze and Jaffar in [6] and used in set-based program analysis. This class is strictly less expressive than the general class of set constraints (where all boolean set operators, that is, union, intersection and complement, as well as function symbols of arbitrary arity are allowed). In fact, it was observed in [4] that definite set constraints are exactly as expressive as set constraints with intersection as the only boolean operator, and that the satisfiability problem for constraints in this class is DEXPTIME-complete.

Unary set constraints were first introduced by Aiken, Kozen, Vardi and Wimmers in [1]. This is a class of set constraints, which does not allow the use of function symbols of arity greater than 1. Obviously, this class is also strictly less expressive than the general one. It is shown in [1] that the satisfiability problem for unary set constraints is DEXPTIME-complete, as opposed to the NEXPTIME-completeness of the non-unary case. The DEXPTIME-hardness proof presented there depends essentially on the complement operator, which is not allowed in definite constraints.

In both cases (definite or unary) the restriction in expressivity decreases the complexity of the satisfiability problem (see the left part of Figure 1). A similar decrease of complexity (to PSPACE) was expected if one takes the both restrictions at the same time. However, as we prove below, this is not the case: the satisfiability problem for unary definite set constraints is DEXPTIME-complete.

There are quite strong connections between definite or unary set constraints and tree automata. In particular, the DEXPTIME-hardness result for definite set constraints works by an encoding of the intersection emptiness problem for tree automata. Unary set constraints are equivalent to the existential fragment of the second-order theory of n successors, and thus can be solved using tree-automata techniques [8]. Since for many DEXPTIME-complete problems in the tree-automata theory (like intersection emptiness, language equivalence, universality) their corresponding problems for string automata are PSPACE-complete, it was expected that satisfiability of unary definite set constraints should also be PSPACE-complete. Surprisingly, as we show here, this problem is DEXPTIME-complete.

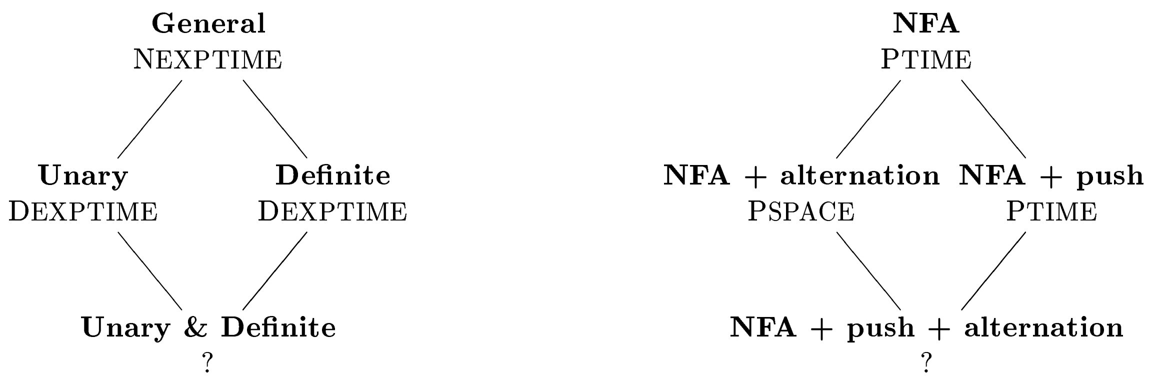

Figure 1: Complexity for problems in automata theory and set constraints

Automata theory. One can think of a nondeterministic finite automaton (NFA) as a machine equipped with a finite control and a stack, where the input word is written on the stack and the automaton is allowed to do only pop operations (it is not allowed to push anything on the stack). Büchi [3] extended word automata to canonical systems, which may be viewed as finite automata with both pop and push operations. It is well-known that the emptiness problem for such automata (often called pushdown processes) is decidable in PTIME. Another possible extension of finite automata is alternation. The emptiness problem for alternating word automata is PSPACE-complete. A natural question that arises here is what happens if we add both: alternation and push operations (see the right part of Figure 1). As we show in this paper, already the membership problem (and thus, also emptiness) for such automata is DEXPTIME-hard (the membership in DEXPTIME is easy by reduction to the tree case, so these problems are DEXPTIME-complete). These are the first problems that we are aware of that concern finite automata on words and are DEXPTIME-complete. It is worth noting here that a similar extension for alternating tree automata does not change the complexity (DEXPTIME) of the emptiness problem.

Model checking. In [2] Bouajjani, Esparza and Maler considered a problem of model checking for pushdown systems. They reduce this problem to a reachability problem for what they call alternating pushdown systems. These alternating pushdown systems are exactly the above mentioned finite automata extended with push operations and alternation. The problem whether a given alternating pushdown systems accepts a given word is a particular case of the reachability problem. The authors claim that the reachability problem can be solved in time polynomial in the size of the system and exponential in the size of the configuration that is to be reached (in the case of our acceptance problem this configuration is of constant size). The results of this paper show that the claim from [2] is not correct.

Program analysis. In [5] Frühwirth, Shapiro, Vardi and Yardeni proposed a type system for logic programs that is based on a syntactically restricted class of logic programs, so-called proper unary-predicate programs. The types inferred from a program coincide with the ones obtained by Heintze and Jaffar in [7] using set-based analysis. The basis of this analysis is to present types as sets of trees. It is shown in [5] that the type-checking problem for the set-based approximation (*i.e.* the membership problem for a ground atom to the least model of a proper unary-predicate program) is DEXPTIME-complete.

Frühwirth, Shapiro, Vardi and Yardeni mention in [5] another type system, based on a path-based approximation of the least model of a program. When set-based approximation refers to sets of trees, the path-based approximation refers to sets of words: the key idea of the path-based approximation is to view atoms (or sets of atoms) in terms of paths description (*i.e.* a set of words). Function symbols f of arity n strictly greater than 1 are replaced by n unary function symbols $f_1, \dots, f_n$. The term $f(a,b)$ is then considered as the sets of paths $\{f_1(a), f_2(b)\}$ and the set of terms $\{f(a,b), f(b,b)\}$ as the set $\{f_1(a), f_2(b), f_1(b)\}$.

Since this approximation is strictly "weaker" than the set-based one, the authors raised the question whether the type-checking problem for the path-based approximation would be simpler than for set-based approximation. We answer negatively to this question by showing that the type-checking problem for the path-based approximation is also DEXPTIME-complete.

References

[1] A. Aiken, D. Kozen, M. Vardi, and E. L. Wimmers. The complexity of set constraints. In *1993 Conference on Computer Science Logic*, LNCS 832, pages 1–17. Springer-Verlag, Sept. 1993.

[2] A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata: Application to model checking. In *CONCUR'97*, LNCS 1243, pages 135–150, 1997.

[3] J. R. Büchi. Regular canonical systems. *Archiv Mathematische Logik und Grundlagenforschung*, 6:91–111, 1964. Reprint in Saunders Mac Lane, Dirk Siefkes, editors, The collected works of J. Richard Büchi, Springer-Verlag, 1990.

[4] W. Charatonik and A. Podelski. Set constraints with intersection. In G. Winskel, editor, *Twelfth Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 362–372. IEEE, June 1997.

[5] T. Frühwirth, E. Shapiro, M. Vardi, and E. Yardeni. Logic programs as types for logic programs. In *Sixth Annual IEEE Symposium on Logic in Computer Science*, pages 300–309, July 1991.

[6] N. Heintze and J. Jaffar. A decision procedure for a class of set constraints (extended abstract). In *Fifth Annual IEEE Symposium on Logic in Computer Science*, pages 42–51, 1990.

[7] N. Heintze and J. Jaffar. A finite presentation theorem for approximating logic programs. In *Seventeenth Annual ACM Symposium on Principles of Programming Languages*, pages 197–209, January 1990.

[8] M. Vardi. An automata-theoretic approach to unary set constraints. unpublished manuscript.