Solving Classes of Set Constraints with Tree Automata

P. Devienne, JM. Talbot and S. Tison *
{devienne,talbot,tison}@lifl.fr

Laboratoire d'Informatique Fondamentale de Lille
CNRS - URA 369
Université des Sciences et Technologies de Lille

Abstract. Set constraints is a suitable formalism for static analysis of programs. However, it is known that the complexity of set constraint problems in the most general cases is very high (NEXPTIME-completeness of the satisfiability test). Lots of works are involved in finding more tractable subclasses.

In this paper, we investigate two classes of set constraints shown to be useful for program analysis: the first one is an extension of definite set constraints including the main feature of quantified set expressions. We will show that the satisfiability problem for this class is EXPTIME-complete.

The second one concerns constraints of the form $X \subseteq \exp$, where $\exp$ is built with function symbols, the intersection and union connectives and projection operators.

The dual aspects of those two classes allows to find a common approach for solving both of them. This approach uses as basic tool tree automata, which are suitable both for computation and representing the solution of those solving problems. It leads also to simple algorithms and an easy characterization of complexity.

1 Introduction

Set constraints allow to express relations between sets of (ground) terms. They can be defined as inclusion or non-inclusion between expressions built over variables, function symbols and set operators. The set operators that are encountered in most works are intersection, union, complementation and projection. The main problems that have been addressed concerning set constraints are the classical ones of the constraint paradigm, that is, satisfiability [AW92] [GTT93][MNP97], entailment [CP97a] [MN97] and solving [AM91] [Hei92b]. They have been shown very useful for program analysis [Hei92b][Hei94][MNP97]. Satisfiability problem for the largest class of set constraints (where all the set operations cited above can occur) has been proved to be NEXPTIME-complete [CP94] and the complexity remains the same if projection is omitted [Ste94]

* This work was supported in part by the HCM project CONSOLE (CHRXCT940495)
This leads to a wide interest for more tractable subclasses, defined according to the different considered set operations and/or other syntactic restrictions.

In this paper, we extend the class of definite set constraints [Hei92a] by allowing a “membership expression” operator in the left hand-side of inclusion. This operator is a limited form of the quantified set expressions introduced in [Hei92b].

It looks like \{x \mid f(x, x) \in Y \land g(x, y) \in Z\}. Such an expression would be interpreted as the set \{a\} if for instance \(Y = \{f(a, a), f(b, c), f(c, b), f(d, d)\}\) and \(Z = \{f(a, b), g(a, b)\}\). We propose an algorithm for testing satisfiability of a system of such set constraints and prove that this problem is EXPTIME-complete.

Podelski and Charatonik proposed in [CP97b] a method à la Heintze for approximating non-failure semantics of logic programs. This analysis amounts to computing the greatest solution (over sets of finite or infinite trees) of a class of set constraints, symmetric of the one used by Heintze [Hei92a], that is of the form \(X \subseteq \text{exp}\), where \text{exp} is built over function symbols, intersection, union and projection. Their method, dealing with sets of finite or infinite trees, combines syntactic transformations of the constraints with tree automata for testing emptiness of variables and representing the greatest solution.

We propose in this paper an elegant and homogeneous framework for solving definite set constraints with "membership expression" operator and the second class (over finite trees). This framework is based on tree automata providing a tool both for computation and representation of the solution of the constraints. The solving algorithms are described as relations (defined as inference rules) over tree automata. This leads to a very simple and uniform approach, in which the dual aspects of those two classes are revealed. Roughly speaking, for set constraints with intersection, we start with an "empty" automaton (i.e. each variable is interpreted as the empty set) and we modify this automaton (by adding terms into the interpretation of the variables) according to the constraints. For the second class, we start from the "universe" automaton (i.e. each variable is interpreted as the set of all terms) and we remove terms from the interpretation of the variables according to the constraints.

After giving a few definitions and properties about set constraints in the next section, we present our basic tool, that is tree automata in section 3. Section 4 is devoted to an algorithm for deciding satisfiability of definite set constraints with "membership expression" operator. Finally, section 5 deals with an algorithm for computing the greatest solution (over finite trees) for the class of set constraints introduced in [CP97b].

2 Preliminaries

We assume given a finite set of function symbols \(\Sigma\) and \(\mathcal{V}\) a countable set of (set) variables (denoted \(X, Y, Z, X_1, X_2, \ldots\)). \(\text{TERM}(\Sigma)\) is the set of ground terms built over \(\Sigma\).