PASS: String Solving with Parameterized Array and Interval Automaton

Guodong Li and Indradeep Ghosh
Fujitsu Labs of America, CA
{gli,ighosh}@fla.fujitsu.com

Abstract. The problem of solving string constraints together with numeric constraints has received increasing interest recently. Existing methods use either bit-vectors or automata (or their combination) to model strings, and reduce string constraints to bit-vector constraints or automaton operations, which are then solved in the respective domain. Unfortunately, they often fail to achieve a good balance between efficiency, accuracy, and comprehensiveness. In this paper we illustrate a new technique that uses parameterized arrays as the main data structure to model strings, and converts string constraints into quantified expressions that are solved through quantifier elimination. We present an efficient and sound quantifier elimination algorithm. In addition, we use an automaton model to handle regular expressions and reason about string values faster. Our method does not need to enumerate string lengths (as bit-vector based methods do), or concrete string values (as automaton based methods do). Hence, it can achieve much better accuracy and efficiency. In particular, it can identify unsatisfiable cases quickly. Our solver (named PASS) supports most of the popular string operations, including string comparisons, string-numeric conversions, and regular expressions. Experimental results demonstrate the advantages of our method.

1 Introduction

A string solver is used to determine the satisfiability of a set of constraints involving string operations. These constraints can be mixed with numeric constraints, in which case we call them hybrid constraints. This paper is about how to solve hybrid constraints efficiently using SMT solving and automaton approximation.

Hybrid constraints may be produced by a static analyzer or a symbolic executor. For example, a symbolic executor for web applications may produce thousands of path conditions containing non-trivial hybrid constraints. Solving these constraints efficiently is the key for the tool to be scalable and practical. A typical web application takes string inputs on web pages and performs a lot of string operations such as concatenation, substring, >, and matches. There are also three more typical requirements: (1) strings are converted into numeric values for back-end computations; (2) string values are constrained through regular expressions; and (3) unsatisfiable hybrid constraints should be identified quickly. This poses unique challenges to many symbolic execution tools which usually handle only numeric constraints well.
While there are some external string solvers available, none of them meet our need to obtain a good balance between efficiency, accuracy, and comprehensiveness. Roughly, existing solvers can be divided into two categories: (1) bit-vector (BV) based methods, which model a string with a fixed-length bit-vector; and (2) automaton based methods, which model a string with an automaton. A BV method needs to compute the lengths of all strings before constructing bit-vectors, hence it may enumerate all possible length values in order to prove or disprove a set of constraints. Such enumeration often leads to exponential numbers of fruitless trials. In contrast, an automaton method models a string with an automaton capturing all possible values of this string. String automata can be refined according to the relation of the strings. Essentially, an automaton is an over-approximation of string values, and the refinement is often insufficient, requiring the enumeration of concrete string values and/or string sequences to find out a valid solution. The methods combining these two models inherit many of the disadvantages while circumvent some.

In this paper we propose a new way to model strings so as to avoid brute-force enumeration of string lengths or values. We model a string with a parameterized array (parray for short) such that (1) the array maps indices to character values, (2) both the indices and the characters can be symbolic, and (3) the string length is pure symbolic. With this model, string constraints are converted into quantified constraints (e.g. $\exists$ and $\forall$ expressions) which are then handled through our quantifier elimination scheme. Our conversion scheme follows a declarative and non-recursive style. The produced quantified constraints are often beyond the capacity of modern SMT solvers such as Yices [19] and CVC [1]. To handle them, we propose an efficient quantifier elimination algorithm. This conversion scheme is our first contribution. It precisely models string operations and string-numeric conversions. The quantifier eliminator is our second contribution.

Our third contribution is to use interval automata to build an extra model for strings, and reason about string values via automata. We use automata to not only handle regular expressions (RegExps), but also enhance the solving of non-RegExp cases. We demonstrate how to refine the automata through deductive reasoning and fixed-point calculation.

Our fourth contribution is to combine the parray and automaton model to determine satisfiability efficiently. For example, when the automaton domain finds unsat, the solver can safely claim unsat. While the automaton model is mandatory in modeling RegExps, we can use the automata to refine the parray model for locating a solution fast.

We perform preliminary experiments to compare different methods, and show that our method outperforms existing ones in general.

As far as we know, our P-Array based String Solver (PASS) is the first to explicitly use parameterized arrays to model strings and apply quantifier elimination to solve string constraints. It is also the first to combine interval automaton and parray for fast string solving. As for comprehensiveness, it handles virtually all Java string operations, regular expressions, and string-numeric conversions.