A Practical String Analyzer by the Widening Approach*

Tae-Hyoung Choi, Oukseh Lee, Hyunha Kim, and Kyung-Goo Doh**

Department of Computer Science and Engineering, Hanyang University, Ansan, Korea
{thchoi, oukseh, hhkim, doh}@pllab.hanyang.ac.kr

Abstract. The static determination of approximated values of string expressions has many potential applications. For instance, approximated string values may be used to check the validity and security of generated strings, as well as to collect the useful string properties. Previous string analysis efforts have been focused primarily on the maxmization of the precision of regular approximations of strings. These methods have not been completely satisfactory due to the difficulties in dealing with heap variables and context sensitivity. In this paper, we present an abstract-interpretation-based solution that employs a heuristic widening method. The presented solution is implemented and compared to JSA. In most cases, our solution gives results as precise as those produced by previous methods, and it makes the additional contribution of easily dealing with heap variables and context sensitivity in a very natural way. We anticipate the employment of our method in practical applications.

1 Introduction

Strings are used in many applications to build SQL queries, construct semi-structured Web documents, create XPath and JavaScript expressions, and so on. After being dynamically generated from user inputs, strings are sent to their respective processors. However, strings are not evaluated for their validity or security despite the potential usefulness of such metrics [5,7,6]. Hence, this paper aims to establish a method for statically determining the approximated values of string expressions in a string-generating program.

1.1 Related Works

Previous efforts to statically determine the approximated values of string expressions have attempted to maximize the precision of string approximations.

Christensen, Møller and Schwartzbach [2] developed a Java string analyzer (JSA) that approximates the values of string expressions using regular language. An interprocedural data-flow analysis is first used to extract context-free grammar from a Java program such that each string expression is represented as a

* This work was supported in part by grant No.R01-2006-000-10926-0 from the Basic Research Program of the Korea Science and Engineering Foundation, and in part by Brain Korea 21.

** Corresponding author.

N. Kobayashi (Ed.): APLAS 2006, LNCS 4279, pp. 374–388, 2006.

nonterminal symbol. Then, Mohri and Nederhof's algorithm [8] is applied to approximate the context-free grammar with regular grammar. Eventually, the string analysis produces a finite state automaton that conservatively approximates the set of possible strings for each specified string expression. JSA tends to be adequate when every string value is stored in a local variable, but it falters when dealing with strings stored in heap variables. Perhaps the method could be extended to deal with such variables, but not in a straightforward and immediate manner.

To conduct string analysis based on regular expressions, Tabuchi, Sumii, and Yonezawa [11] created a type system for a minimally functional language equipped with string concatenation and pattern matching over strings. However, they failed to provide a type inference algorithm due to a technical problem with recursive constraint solving. Our analysis can be thought of as a solution to their problem based on a carefully crafted widening.

Thiemann [12] presented a type system for string analysis based on context-free grammar and provided a type inference algorithm derived from Earley's parsing algorithm. His analysis is more precise than those based on regular expressions, and though sound, his inference algorithm is incomplete because its context-free language inclusion problem cannot be solved. The weak point is that the grammar must be written in terms of single characters rather than tokens.

Minamide [7] also developed a static program analyzer that approximates string output using context-free grammar. His analyzer, which uses a variant of the JSA approach to produce context-free grammar from a PHP program, validates HTML documents generated by a PHP program either by extracting and testing sample documents or by considering documents with a bounded depth only.

1.2 Our Approach

Our work is motivated by a desire to statically determine which database application program accesses and updates which database tables and fields. Such information is particularly useful in maintaining huge enterprise software systems. To obtain this information statically, all possible SQL queries must be extracted from database application programs as strings.

Strings may be stored as field variables in object-oriented applications, thus a string analysis must be able to determine their value. For example, the Java application in Fig. 1 uses a field variable to construct strings. The class `SQLBuffer` is defined as a gateway for connecting to a database server. In this example, two `SQLBuffer` objects are allocated and each object has a separate string field, `buf`. To prevent the clouding of analysis precision, independent string fields should be maintained as such. Thus, heap memory analysis is required. Furthermore, the methods `add` and `set` are called multiple times in different contexts. As such, precise string analysis must also be context-sensitive. For the example in Fig. 1, our analyzer is able to distinguish possible queries as `SELECT .* FROM .*` and `UPDATE .* SET .* = .*`, while JSA is unable to do so and only gives `.*` that means any string.