Masarykova univerzita
Fakulta informatiky
Datum: 15 -01- 2021
Č.j./E.č.: MU-IS/5008/2021/1316846/FI-2
Počet listů dokumentu: .....
Počet příloh a listů/sv.: ...../.....
Počet a druh nelist. příloh: ...../.....

KANSAS STATE UNIVERSITY | Department of Computer Science

Report on ***String Analysis for Software Verification***

The objective of this thesis is to present a variety of useful, proved correct, analysis techniques for programs that construct and manipulate strings. Programs that work with strings are a key part of software that manipulate the internet, and these programs are particularly error prone and are therefore subject to exploitation and misuse. For this reason, the thesis addresses an important topic.

The work perfomed in the thesis is of theoretical and practical interest, it is substantial, and it is clearly adequate for a Doctor of Philosophy degree.

The central contribution of the thesis is an adaptation of a complex array analysis, FunArray, due to Cousot, et al., to string analysis on programs that use strings in a C-style implementation. This is a nontrival step and the resulting analysis, M-String, has a wide range of applications, which are themselves developed and applied over several chapters of the thesis.

It is important to note that the M-String algebra (its abstract domain and its abstract operations) are formulated as a Galois connection within the mathematical machinery of Cousot-Cousot abstract interpretation. This effort is nontrivial and significant because it ensures that the the algebra is sound (will not lead to false analysis results) and its abstract operations are as precise as possible with regards to the abstract domain.

The thesis also examines the completion of several abstract domains for string analysis so that domains' operations are backwards-(alpha)-complete, that is, there is no loss in information accumulated by the operations with regards to the level of precision expressed by the domain's elements. In addition, the reduced product (combination of) two abstract domains is studied. These listed techniques are nontrivial to develop and implement, and it is noteworthy that the material is provided here.

Finally, a last chapter presents work on software watermarking techniques. The material in this chapter stands somewhat independently from that of the other chapters. Since I do not have the expertise to evaluate software-watermarking techniques, I will only state that the approach presented in this chapter appears plausible. In any case, the thesis is sufficient for a doctor's degree even without this last chapter.

The thesis is well written; it does require of the reader prior expertise in the subjects of abstract interpretation and program flow analysis. It also demands of the reader a devotion to delve

into complex definitions and calculations. But the outcome is a proved-correct analysis and implementation for handling an important software problem. The literature search and list of bibliographic references are substantial. The structure of the thesis is appropriate.

What follows now is an overview of the thesis's technical development.

Chapter 3 formulates and proves sound the algebra (abstract domain and operations) of M-String. Abstract domain M-String approximates both the content and the "shape" of strings by representing them as abstractions of consecutive, non-overlapping, possibly empty segments. Each segment represents a sequence of characters which share the same abstract property (the abstract property is a value taken from a more primitive abstract domain) and is surrounded by the lower and upper segment bounds that abstract concrete array indexes.

A key insight to proving the soundness of this sophisticated technique is the decision to define a C-like "concrete string" not as a linear storage layout of characters but as a five-tuple that has an (isomorphic) mapping to the traditional linear layout. It is the domain of the concrete five-tuples that are abstracted via a Galois-connection into the elements of M-String. One complication of this definition of concrete strings is that the C-operations on strings must themselves be recoded to operate on the five-tuples. This is indeed done for the core C-operations of string embedding, extraction, comparison, and substitution.

The remainder of the chapter applies the M-String domain, which abstracts a (set of) concrete C-strings by a pair, where the first component abstracts the "string of interest" (the leading prefix of the concrete string up to first occurrence of null, that is, \0), and the second component abstracts the remainder of the string. Each component is a sequence of bounded elements that represent sequences of characters with a common property. Note that the bound is can be stated abstractly, say, as an interval. The development is necessarily complicated by the use of the array-unification algorithm of Cousot, et al., which must be adapted for the abstracted strings.

Following this technical development, the M-String domain is applied to several program analysis problems, where properties of individual characters or character sequences are embedded into M-String and then calculated for programs that manipulate strings. The chapter demonstrates that the analysis works well when characters are abstracted to the constant-propagation domain, that is, a string is either a constant value or *, meaning anything at all. The abstract domain and its operations is inserted into several analysis engines and applied to a variety of nontrivial C-libraries.

Chapter 4 is a case study of combining the FunArray domain with a second string-analysis domain and reducing the resulting pairs to obtain more precision than that available from using each domain independently. (This pairing is called a reduced product construction. It is constructed on-the-fly rather than with a fixed methodology, so reduced products must be

formulated on a case-by-case basis.) The sample string-analysis domains combined with FunArray are String Length, Character Inclusion and String Prefix. Examples are provided that show how analysis precision is improved within each of the reduced product constructions.

Chapter 5 is a study of completeness of abstract operations upon an abstract domain. The chapter begins with a review of Giacobazzi-style completeness and a motivating example of PHP-style arithmetic with strings that represent numbers. The bulk of the chapter develops two completed abstract domains for operations on strings: one domain, from the SAFE analyzer, determines whether a string represents a numeric value. The second, from the TAJS analyzer, determines whether a string can be used/cast into an array for indexing.

The first domain is analyzed with respect to its use by the toNum casting operation; the completion makes the operation backwards-(alpha)-complete. The second domain is analyzed with respect to the string concat operation. The domain's completion is more complex, but crucially, only a finite number of additional elements are needed to make the concat operation complete.

The pragmatic benefits of improving the two domains are discussed. Next, a metric is devised to measure the preciseness of an abstract element within a domain, and the metric is used to calculate measures for elements of abstract domains and their completions. The expected and obtained result is that a program analysis with a completed abstract domain yields answers whose precision measure are less-or-equal than the analysis employed with the original abstract domain.

Chapter 6 contains material that suggests one can perform relational analysis (that is, track information about the relationship between the values of two variables) on strings. The case study uses a less-than ordering on strings to remember which string value is less-than another, and the implementation mechanics are adopted from the octagon abstract domain for numerical variables.

The less-than ordering is first instantiated to approximate string length. Next the ordering is instantiated to sets-of-characters, that is, s1 <= s2 when all the characters that appear in string s1 also appear in s2. The third instance of the ordering is is-a-substring-of. The utility of the <= ordering is extended by <=*, which relates a string expression, perhaps built using string concatenation, to a string variable. The relation is used for the is-a-substring-of ordering. Galois connections and provided for each of the above orderings.

Chapter 7 is a proposal that applies string analysis to "watermarking" the data in a relational database (that is, mutating the data to encode ownership but in this case without altering substantively the results of data-base queries). This area falls outside of my expertise, so I cannot comment on its technical quality, but I appreciated the chapter's exposition and

technical development.

In summary, the thesis presents a comprehensive development of techniques for string analysis based on the abstract-interpretation methodology. The constructions are precisely defined, shown to be sound (and in some cases, complete), and are prototyped within existing program-analysis engines. The work is a substantial contribution to the field and its author is certainly deserving of a PhD degree.

Sincerely,

David A Schmidt

University Distinguished Professor Emeritus, Computer Science Department, Kansas State University, Manhattan, Kansas, USA

1 January 2021