Science of Computer Programming 97 (2015) 405–425

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

Automatic generation of valid and invalid test data for string validation routines using web searches and regular expressions

Muzammil Shahbaz *, Phil McMinn, Mark Stevenson

University of Sheffield, UK

HIGHLIGHTS

- An approach for finding valid values for string data types on the Internet.
- A mutation algorithm for regular expressions to produce invalid values for string data types.
- A testing procedure to identify program errors using the valid and invalid values.
- An empirical study of the approach on 24 open source case studies.
- An analysis of the approach against two contemporary test data generation tools.

ARTICLE INFO

Article history:
Received 26 May 2013
Received in revised form 3 April 2014
Accepted 8 April 2014
Available online 24 April 2014

Keywords:
Test data generation
Web searches
Regular expressions

ABSTRACT

Classic approaches to automatic input data generation are usually driven by the goal of obtaining program coverage and the need to solve or find solutions to path constraints to achieve this. As inputs are generated with respect to the structure of the code, they can be ineffective, difficult for humans to read, and unsuitable for testing missing implementation. Furthermore, these approaches have known limitations when handling constraints that involve operations with string data types.

This paper presents a novel approach for generating string test data for string validation routines, by harnessing the Internet. The technique uses program identifiers to construct web search queries for regular expressions that validate the format of a string type (such as an email address). It then performs further web searches for strings that match the regular expressions, producing examples of test cases that are both valid and realistic. Following this, our technique mutates the regular expressions to drive the search for invalid strings, and the production of test inputs that should be rejected by the validation routine.

The paper presents the results of an empirical study evaluating our approach. The study was conducted on 24 string input validation routines collected from 10 open source projects. While dynamic symbolic execution and search-based testing approaches were only able to generate a very low number of values successfully, our approach generated values with an accuracy of 34% on average for the case of valid strings, and 99% on average for the case of invalid strings. Furthermore, whereas dynamic symbolic execution and search-based testing approaches were only capable of detecting faults in 8 routines, our approach detected faults in 17 out of the 19 validation routines known to contain implementation errors.

* Corresponding author.
E-mail addresses: muzammil.shahbaz@gmail.com (M. Shahbaz), p.mcminn@sheffield.ac.uk (P. McMinn), m.stevenson@dcs.shef.ac.uk (M. Stevenson).

http://dx.doi.org/10.1016/j.scico.2014.04.008
0167-6423/

1. Introduction

There has been much work in the literature of late devoted to automated test input generation [1], however handling string input types remains a challenging task [2,3]. This is due to the inherent complexity of real-world data that is naturally encoded as strings—e.g., dates of different formats, banking codes, registration numbers, etc.—which have very large input domains, and consequently, involve a huge search space for test data generation.

To date, a number of approaches have been investigated, including symbolic execution [4] and search-based testing [5]. However, since they are driven by the need to obtain high levels of structural coverage—e.g., branch coverage—the test suited produced have the following deficiencies:

Low test effectiveness: Test suites that achieve high coverage are not necessarily effective, particularly where string data types are concerned, since it is possible to cover program structure without generating any inputs similar to those actually supplied in practice when the software is deployed. For example, the Java method below—`isMonth` (from the open source project *TMG*[1])—validates whether a given string input is a month name, i.e. '`January`' to '`December`'. However, the method can be fully "covered" without an actual month name being supplied, through execution of the method with an arbitrary (possibly empty) string:

```
// declaration of a set          // method body
Set months = new HashSet();      boolean isMonth(String month) {
// initialisation                    return months.contains(month);
months.add("January");           }
...
months.add("December");
```

Difficult-to-read test inputs: Automatically generated test inputs tend to be hard for human testers to read and understand. Since a formal specification is frequently unavailable, a tester often assumes the role of a *human oracle* [6]—that is, manually determining whether the right outputs were produced for the generated inputs. This task is made harder when test inputs are not easy to read [7]. For instance, it is harder for a human to distinguish between arbitrary email addresses such as '`"b\2@3#t"@s3t`' (valid) and '`"b\2@3#"t@s3t`' (invalid[2]), than '`bob@mail.com`' (valid) and '`bob@mailcom.`' (invalid[3]).

Inability to test missing implementation: Roughly 35% of program implementation errors result from missing functionality [8]. One way to detect such errors is to test programs with invalid values. However, automated techniques guided by program structures cannot produce such values due to missing logic paths, or so-called "sins of omissions". For example, an email validation program in the open source project *LGOL*,[4] misses a check for rejecting values containing more than one '@' symbol. Hence, the address '`i.am@invalid@for.sure.com`' passes the validation test.

This paper builds upon our previous work [3] that proposed an approach for generating valid values using tailored web searches and regular expressions (which were also sought dynamically from web sources). The web searches are conducted through web queries that are generated using information extracted from program identifiers following the application of natural language processing techniques.

In this paper, we extend the approach for generating invalid values using regular expression mutation, and further define a testing procedure using the generated valid and invalid values to find potential program errors—in particular missing logic paths. The paper furnishes an empirical study conducted on 24 string input validation routines collected from 10 open source projects. The results of the study show that the approach was capable of finding a number of valid and invalid values for different string types, with an average accuracy of approximately 34% for valid values and 99% for invalid values. The approach also detected that 17 out of the 19 routines contained implementation errors when using the values generated. The approach has been analysed against two contemporary test data generation tools implementing dynamic symbolic execution [9] and search-based testing [2,10] techniques. These tools were only able to generate a very low number of values, and detected errors in only 8 routines.

The rest of the paper is organised as follows. Section 2 provides an overview of the proposed approach. Sections 3–7 explain different steps of our approach in detail. Section 8 then reports the empirical and comparative study of the approach, while Section 9 discusses potential inherent threats to validity in our evaluation. Section 10 details related work, and finally Section 11 concludes the paper with directions for future work.

[1] http://tmgerman.sf.net.
[2] Quotes must be separated by '.', or they must be the outer characters of the local-part.
[3] '.' must not be the last character in the domain-part.
[4] http://lgol.sf.net/.