A Formalisation of Java Strings for Program Specification and Verification

Richard Bubel¹, Reiner Hähnle¹, and Ulrich Geilmann²

¹ Department of Computer Science and Engineering
Chalmers University of Technology, Sweden
{bubel, reiner}@chalmers.se
² Department of Computer Science, KIT Karlsruhe, Germany
ugeilmann@googlemail.com

Abstract. We present a formalisation of Java Strings tailored to specification and verification of programs (using dynamic logic). The formalism allows to specify and verify properties about the content of strings—the most common use-case—in an easy and natural manner. Each instance of type String is related to an abstract data type representing the string content as an immutable sequence of characters. This avoids serious technicalities that would arise if the specification had to resort to Java arrays to represent sequences of characters. We also discuss advanced aspects of Java Strings including string literals and the string pool and support for regular expressions. The approach has been implemented in the KeY verification system. We demonstrate its practical applicability by case studies including the verification of a string sanitization function.

1 Introduction

Most Java programs that deal with user input and output make usage of strings. Formal verification of Java-like languages progressed rapidly regarding the size and complexity of verifiable programs in the past years, yet none of the state-of-art systems [1,2,3,4] offers a comparable degree of support and automation for strings as is on hand for other datatypes of central importance. Typically, strings are given less priority in formalisation by the designers of verification tools, because they deal with aspects of programs that are considered to be computationally trivial. Another reason for little work having been done on strings is that many Java verification systems concentrated on Java Card or other Java dialects that have no string support.

Nevertheless, missing or insufficient support for strings makes it impossible to verify many practically relevant programs or requires to rewrite them before verification. One should also keep in mind that many attacks on software security are based on injection of intentionally malformed input[1] which makes specification and verification of methods with string type much more security-relevant and less trivial than thought at first.

¹http://www.sans.org/top25-programming-errors/
In this paper we present a formalisation of Java strings within the program logic of the KeY verification system [3]. This includes a formal specification of a substantial part of the Java String API (Sect. 3.5) as well as automated reasoning on first-order string functions and predicates.

Strings in Java-like languages are non-trivial to model for several reasons: first of all, strings are (immutable) instances of the String class. This means that strings are objects with all the usual complications such as aliasing. It means also that several dozen API methods can be applied to Strings whose behaviour must be specified. In Java and C# immutability of strings is exploited in the String pool which implements caching of equal string literals.

In order to allow efficient automated reasoning we map Java String objects and methods into a first-order abstract data type for finite character sequences (Sect. 3.1). The mechanism used in KeY for symbolic state updates (described in Sect. 2.2) allows us to do this in a modular fashion, see Sect. 3.2. We also model the caching of string literals in the Java String pool (Sect. 3.3) and support for regular expression on character sequences (Sect. 3.4).

We demonstrate the practicability of our approach with three case studies that exhibit a high degree of automation that can be achieved with our formalisation when verifying Java programs with strings. Much of our work is also applicable to C# which has a very similar string concept than Java.

2 Background

2.1 Java Strings

In contrast to many other programming languages, strings in Java are not identified with an array of characters. Instead, a Java String is an object of class java.lang.String. Strings are immutable, i.e., once created their content—the encapsulated character sequence—cannot be changed.

While strings behave like ordinary objects, they are part of the core classes in the Java Language Specification (JLS) [5] and enjoy special support in the form of String literals and the String pool. String literals are Java expressions and consist of a possibly empty sequence of characters enclosed in double quotation marks (§3.10.5 [5]): StringLiteral := "(StringCharacter)"*

String literals are intended to behave like literals of primitive types. Consequently, they are immutable and refer always to the same instance of type String. Additionally, Java provides the following support for strings:

- The '+'-operator is overloaded to be defined on Strings: if one of its arguments is of type String, the second one is (if necessary) translated to its canonical String representation (§15.18.1 [5]). The result of the application of '+' is a (new) String object representing the concatenation of the argument strings.
- The notion of compile-time constants is extended to cover string literals. A compile-time constant (§15.28 [5]) is a Java expression adhering to a defined syntactical form that can be (and is) evaluated at compile-time to the resulting value.