Whiley: a Better C?

David J. Pearce
School of Engineering and Computer Science, Victoria University of Wellington, New Zealand
djp@ecs.vuw.ac.nz

Abstract

An ongoing challenge for computer science is the development of a tool which automatically verifies programs meet their specifications, and are free from runtime errors such as divide-by-zero, array out-of-bounds and null dereferences. We have been developing a programming language from scratch to simplify verification, called Whiley, and an accompanying verifying compiler. Like other modern programming languages (e.g. Go, Rust) Whiley eschews ideas from object orientation and is perhaps most similar in style to C. In this paper, we illustrate a short example in Whiley.

1. Introduction

The idea of verifying that a program meets a given specification for all possible inputs has been studied for a long time. Hoare's Verifying Compiler Grand Challenge was an attempt to spur new efforts in this area to develop practical tools [1]. A verifying compiler "*uses automated mathematical and logical reasoning to check the correctness of the programs that it compiles*" [1]. Commonly occurring errors that could be automatically eliminated include: *division-by-zero*, *integer overflow*, *buffer overruns* and *null dereferences*.

The Whiley programming language has been developed from the ground up to enable compile-time verification of its programs [2–5]. The Whiley Compiler (WyC) attempts to ensure that all functions in a program meet their specifications. When it succeeds in this endeavour, we know that: 1) all function post-conditions are met (assuming their pre-conditions held on entry); 2) all invocations meet their respective function's pre-condition; 3) runtime errors such as divide-by-zero, out-of-bounds accesses and null-pointer dereferences are impossible. Note, however, that such programs may still loop indefinitely and/or exhaust available resources (e.g. RAM).

2. Example: C Strings

As an example, let us consider representing the C strings in Whiley. This is useful as we can then try to verify properties about functions operating on C strings (e.g. `strlen()`, `strcpy()`, etc). The main points about C strings are:

- C Strings are arrays of 8bit ASCII characters (roughly speaking).
- C Strings are terminated by the special character '`\0`' (also called null terminated strings).
- C Strings do not carry any other length information (e.g. for the size of the containing memory chunk).

The interesting thing about Whiley is that we can encode these constraints within the language itself. Specifically, we're going to encode a C string as an array integers with appropriate invariants. The array will be constrained to ensure it is null terminated, whilst the contained integers will be constrained to ensure they are between 0 and 255.

Before giving our definition of a C string, we first need to define the notion of an ASCII character as follows:

```
type ASCII_char is (int n) where 0<=n && n<=255
```

We have defined an ASCII character to be an integer which is constrained between 0 and 255 (i.e. 8bits ASCII). Using this, we define our notion of a C string as follows.

```
type C_string is (ASCII_char[] chars)
where |chars| > 0  && chars[|chars|-1] == 0
```

Here, a `C_String` is an array of integers constrained to ensure it is always null terminated. Using this, we implement the well-known `strlen()` function:

```
function strlen(C_string str) → (int r)
ensures r >= 0:
  //
  int i = 0
  //
  while str[i] != 0 where i >= 0 && i < |str|:
     i = i + 1
  //
  return i
```

The Whiley compiler statically verifies this function does not overrun the string bounds. The loop invariant given by the **`where`** clause is needed as a hint to the verifier, but does not affect the function's execution in any way.

References

[1] C.A.R. Hoare. The verifying compiler: A grand challenge for computing research. *JACM*, 50(1):63–69, 2003.

[2] The Whiley Programming Language, http://whiley.org.

[3] D. J. Pearce and J. Noble. Implementing a language with flow-sensitive and structural typing on the JVM. *ENTCS*, 279(1):47–59, 2011.

[4] D. J. Pearce and L. Groves. Whiley: a platform for research in software verification. In *Proc. SLE*, pages 238–248, 2013.

[5] D. J. Pearce and Lindsay Groves. Reflections on verifying software with Whiley. In *Proc. FTSCS*, pages 142–159, 2013.