Bootstrapping

David May: February 27, 2015

Introduction

Bootstrapping is a technique that is widely used in compiler development. It has four main uses:

1. It enables new programming languages and compilers to be developed starting from existing ones.
2. It enables new features to be added to a programming language and its compiler.
3. It also allows new optimisations to be added to compilers.
4. It allows languages and compilers to be transferred between processors with different instruction sets

New programming languages

Let S_x^y be the source of a compiler for language y written in language x, and let E_x be an executable compiler for language x.

Suppose b is a new programming language and that there is a source S_a^b of a compiler for b written in a. An executable compiler for b can be created using the executable compiler for a:

$E_b = E_a(S_a^b)$

Assuming that language b is expressive enough to write a compiler, it can now be used to write its own compiler. The pair S_b^b, E_b can then be used to compile S_b^b into E_b in place of S_b^a, E_a and:

$E_b = E_b(S_b^b)$

This means that language a is no longer needed in the further development of language b.

Adding new features

The process of adding new features is similar to that of developing a new programming language.

Let S_x^y be the source of a compiler for language y written in language x, and let E_x be an executable compiler for language x. Assuming that the E_a and S_a^a are correct:

$E_a = E_a(S_a^a)$

Suppose language b is a superset of language a. An executable compiler for b can be created using the executable compiler for a:

$E_b = E_a(S_a^b)$

The pair S_b^b, E_b can then be used to compile S_b^b into E_b in place of S_b^a, E_a and:

$E_b = E_b(S_b^b)$

Notice that this process can be carried out repeatedly, starting from a very simple language and progressively adding syntactic or semantic features. Once included in the executable version of the compiler, a new feature can be used in the source of the compiler.

Compiler optimisation

Let S_i be the source of a compiler for language a written in language a, and let E_i be an executable compiler for language a. Assuming that the E_i and S_i are correct:

$E_i = E_i(S_i)$

Let S_{i+1} be the source of an optimising compiler for language a written in language a. An executable optimising compiler O_i can be created:

$O_i = E_i(S_{i+1})$

Note that O_i can optimise, but has not itself been optimised. However, it can now be used to create an optimised version of the executable optimising compiler:

$E_{i+1} = O_i(S_{i+1})$

Now the pair E_{i+1}, S_{i+1} can be used in place of E_i, S_i and:

$E_{i+1} = E_{i+1}(S_{i+1})$

Notice that this process can be carried out repeatedly, starting from a very sim-

ple non-optimising compiler and progressively introducing optimisations. As the process continues, the new optimisations are applied to the compiler itself. This means that the executable compiler may reduce in size even though the source becomes larger and more complex.

Transferring to a new instruction set

Let S_x be the source of a compiler for instruction set x, and let E_x^y be an executable compiler in instruction set x that compiles programs into instruction set y. Assuming that the E_a^a and S_a are correct:

$E_a^a = E_a^a(S_a)$

Suppose b is a new instruction set and that there is a source of a compiler for b. An executable compiler in instruction set b can be created using the executable compiler in a:

$E_a^b = E_a^a(S_b)$

Now E_a^b can be used to create an executable compiler in instruction set b that compiles programs into instruction set b:

$E_b^b = E_a^b(S_b)$

The pair S_b, E_b can then be used to compile S_b into E_b^b in place of S_b, E_a^a and:

$E_b^b = E_b^b(S_b)$

This means that the compiler has been transferred to the new instruction set b (and a is no longer needed).

Footnote

In using the above technique, it is essential to maintain a matching pair S, E that can be used to exactly reproduce E. The best way to do this is to perform

$E' = E(S)$

and check that the binary images of E' and E are identical.