Proofs

about programs, proofs as programs, programs as proofs!

Katherine Ye, !con 2014
**Introduction**

This paper contains a proof of the correctness of a simple compiling algorithm for compiling arithmetic expressions into machine language.

The definition of correctness, the formalism used to express the description of source language, object language and compiler, and the methods of proof all intended to serve as prototypes for the more complicated task of proving the correctness of usable compilers. The ultimate goal, as outlined in references [1], [2], [3] and [4] is to make it possible to use a computer to check proofs that compilers are correct.

The concepts of abstract syntax, state vector, the use of an interpreter or defining the semantics of a programming language, and the definition of correctness of a compiler are all the same as in [3]. The present paper, however, is the first in which the correctness of a compiler is proved.

The expressions dealt with in this paper are formed from constants and free variables. The only operation allowed is a binary + although no change in method would be required to include any other binary operations. An example of an expression that can be compiled is

\[(x + 3) + (x + (y + 2))\]

---

From a paper proof of correctness to a computer-checked proof of correctness of our optimizations. (I highly recommend checking out “Software Foundations.” It’s an interactive textbook.)
You’re a compiler.
Let \( x, c, \) end1, end2 be natural numbers.

\[
\text{if } !(x < c) \\
\text{then } \text{end1} \\
\text{else } \text{end2}
\]
Let $x$, $c$, $\text{end1}$, $\text{end2}$ be natural numbers.

Optimize?

```python
if !(x < c)
    then end1
else end2
```
Let \( x, c, \text{end1}, \text{end2} \) be natural numbers. Then for all \( x, c, \text{end1}, \text{end2} \), is this true?

\[
\begin{align*}
? &= \begin{cases} 
\text{end1} & \text{if } ! (x < c) \\
\text{end2} & \text{else}
\end{cases} \\
&= \begin{cases} 
\text{end2} & \text{if } (x < c) \\
\text{end1} & \text{else}
\end{cases}
\end{align*}
\]
Proof: 2 possibilities

Are we right? Prove that they are equal.
$$(x < c) == ?$$

1. if !(x < c) then end1 else end2
2. if (x < c) then end2 else end1

<table>
<thead>
<tr>
<th>Code used</th>
<th>Code output</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>end2</td>
</tr>
<tr>
<td>#2</td>
<td>end2</td>
</tr>
</tbody>
</table>

value of $(x < c)$
Code equality!
(compiler optimizations)
We’ve convinced our human-selves with the proof, but not our compiler-selves. Why not check the proof with a computer?
“Coq: the world’s best computer game”

http://golem.ph.utexas.edu/category/2012/06/the_gamification_of_higher_cat.html
Live-coding interlude

(see katherineye.com for the code)

Qed.
How does Coq work?
Coq = CoC
The CoC is a generalization of a correspondence that we’ll talk about now.
Math profs HATE it:

The one weird trick for interpreting your proofs as programs (and vice versa)
A type <-> a proposition

<->: corresponds to
a -> a
Given anything of type $a$, I can give you something of type $a$

- identity

$f \ x = x$

Interpretated as the type of a function
Given anything of type $a$, I can give you something of type $a$

identity

$f \ x = x$

Assuming hypothesis $a$, I can show that $a$ is true

Trivial: $a$ was assumed to be true!

Interpreted as a proposition for which a proof may or may not exist
Given anything of type a, I can give you something of type a

• identity

f x = x

Assuming hypothesis a, I can show that a is true

Trivial: a was assumed to be true!

Isomorphism
Program $\leftrightarrow$ proof

("value")
Examine our proof

Fancy name: Curry-Howard isomorphism

You can see that our proof, written in Coq’s tactic language, is internally represented as something that looks very much like a function. It’s written in Gallina, Coq’s internal language.
Lots of active research!

- verify compiler (CompCert)
- verify compiler optimizations (Vellvm, of LLVM)
- verify cryptographic code (SHA-256 in OpenSSL, Appel)
- create new foundations of mathematics: homotopy type theory
- writing the game 2048 in Coq (Laurent Théry)

I’d like to assert that the last application is the most important. (code not by me, but can be seen on my site: katherin eye.com)
Thanks!

Katherine Ye
@hypotext