Compiler as theorems

A mindset about compilers, informed by logic

Mei Matteo, march 2024

Disclaimer

I know very little about mathematical logic and compilers, but...

Since I have a foot in two shoes, I risk making this crossover

Let’s go, please correct my assumptions

Think about a computational model

Let’s take for example the Brainf*** computational model:

- 32 KiB of contiguous memory
- Instructions modify the content of the selected memory cell and the position of the head
- Basic branching (of course we want to do something useful with this language)

Hello world example

```
+[-->-[>>+>-----<<]<--<---]>-.>>>+.>>..+++
[.>]<<<<.+++.------.<<-.>>>>+.
```

Why playing with this toy language?

- It’s sufficiently simple to reason about
- It’s sufficiently simple to **programmatically** reason about

What is the goal here

- Transpile a bf program to a C one
- Potentially produce native machine code

=> **Compiling a bf program for a more powerful computational model (actual CPUs)**

Core of this proposal

An optimizing compiler for bf can optimize this expressions:

1) ++++ => add 4 to the currently pointed cell

This optimization is valid because the two instructions produce the same effect on the superset of computational models

Isn’t (1) a theorem??

Theorem zoo

Theorems can be combined to produce correct results not known in advance

What if this (automatic) theorem construction yields some optimizations we could not even imagine?

For example: [-] => ??

My ideal world

- Given a simple computational model, provide a small set of theorems for implementing the operations on a more powerful model
- Automatically deduce new theorems that provide useful optimizations **(and correct ones too!!)**

A hope

- Most of the code I wrote in the last months is conceptually simple…
- Scientific computing code is not demanding in terms of computational expressiveness (my take)
- There is chance for this proposal to find its place (and it won’t be the garbage bin)

If we remove turing completeness...

Heavily inspired on:

- **https://www.youtube.com/watch?v=wE1ZoMGIZHM&ab_channel=LexClips**
- **https://tinygrad.org/**