Compilers and Formal Languages (9)

Email: christian.urban at kcl.ac.uk
Office: N7.07 (North Wing, Bush House)
Slides: KEATS (also homework is there)

While Language

$$
\begin{array}{lcl}
\textit{Stmt} & ::= & \texttt{skip} \\
 & | & \textit{Id}\ \texttt{:=}\ \textit{AExp} \\
 & | & \texttt{if}\ \textit{BExp}\ \texttt{then}\ \textit{Block}\ \texttt{else}\ \textit{Block} \\
 & | & \texttt{while}\ \textit{BExp}\ \texttt{do}\ \textit{Block} \\
 & | & \texttt{read}\ \textit{Id} \\
 & | & \texttt{write}\ \textit{Id} \\
 & | & \texttt{write}\ \textit{String} \\
\textit{Stmts} & ::= & \textit{Stmt}\ \ ;\ \ \textit{Stmts} \mid \textit{Stmt} \\
\textit{Block} & ::= & \{\ \textit{Stmts}\ \} \mid \textit{Stmt} \\
\textit{AExp} & ::= & \ldots \\
\textit{BExp} & ::= & \ldots
\end{array}
$$

Fibonacci Numbers

```
write "Fib";
read n;
minus1 := 0;
minus2 := 1;
while n > 0 do {
        temp := minus2;
        minus2 := minus1 + minus2;
        minus1 := temp;
        n := n - 1
};
write "Result";
write minus2
```

BF***

some big array, say `a`; 7 (8) instructions:

- `>` move `ptr++`
- `<` move `ptr--`
- `+` add `a[ptr]++`
- `-` subtract `a[ptr]--`
- `.` print out `a[ptr]` as ASCII
- `[` if `a[ptr] == 0` jump just after the corresponding `]`; otherwise `ptr++`
- `]` if `a[ptr] != 0` jump just after the corresponding `[`; otherwise `ptr++`

Arrays in While

- `new arr[15000]`
- `x := 3 + arr[3 + y]`
- `arr[42 * n] := ...`

New Arrays

```
new arr[number]
```

```
ldc number
newarray int
astore loc_var
```

Array Update

```
arr[...] :=
```

```
aload loc_var
index_aexp
value_aexp
iastore
```

Array Lookup in AExp

```
...arr[...]...
```

```
aload loc_var
index_aexp
iaload
```

Dijkstra on Testing

"Program testing can be a very effective way to show the presence of bugs, but it is hopelessly inadequate for showing their absence."

What is good about compilers: the either seem to work, or go horribly wrong (most of the time).

Proving Programs to be Correct

Theorem: There are infinitely many prime numbers.

Proof ...

similarly

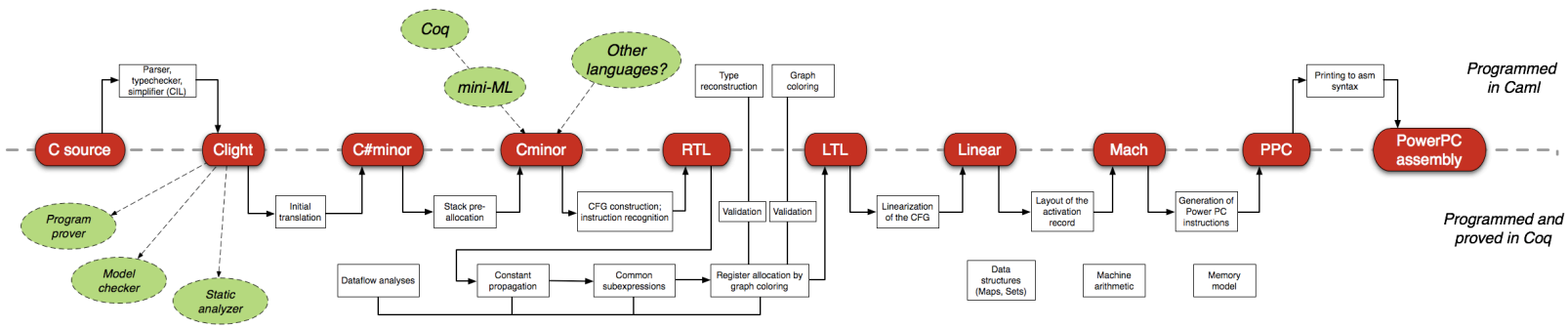
Theorem: The program is doing what it is supposed to be doing.

Long, long proof ...

This can be a gigantic proof. The only hope is to have help from the computer. 'Program' is here to be understood to be quite general (compiler, OS, ...).

Can This Be Done?

- in 2008, verification of a small C-compiler
  - "if my input program has a certain behaviour, then the compiled machine code has the same behaviour"
  - is as good as `gcc -O1`, but much, much less buggy



Fuzzy Testing C-Compilers

- tested GCC, LLVM and others by randomly generating C-programs
- found more than 300 bugs in GCC and also many in LLVM (some of them highest-level critical)
- about CompCert:

> "The striking thing about our CompCert results is that the middle-end bugs we found in all other compilers are absent. As of early 2011, the under-development version of CompCert is the only compiler we have tested for which Csmith cannot find wrong-code errors. This is not for lack of trying: we have devoted about six CPU-years to the task."