Prescience and time travel

Привет. Greetings.

Nervous

Advice from a friend

Be careful

jnthn:

"Don't wear a towel
on your head
from the airport
to the hotel. :-)"
-- jnthn

masak

Became active in 2008

Writing apps, submitting bugs

Exploring the frontiers of Perl 6

Increasing application size

A bit about Perl 6

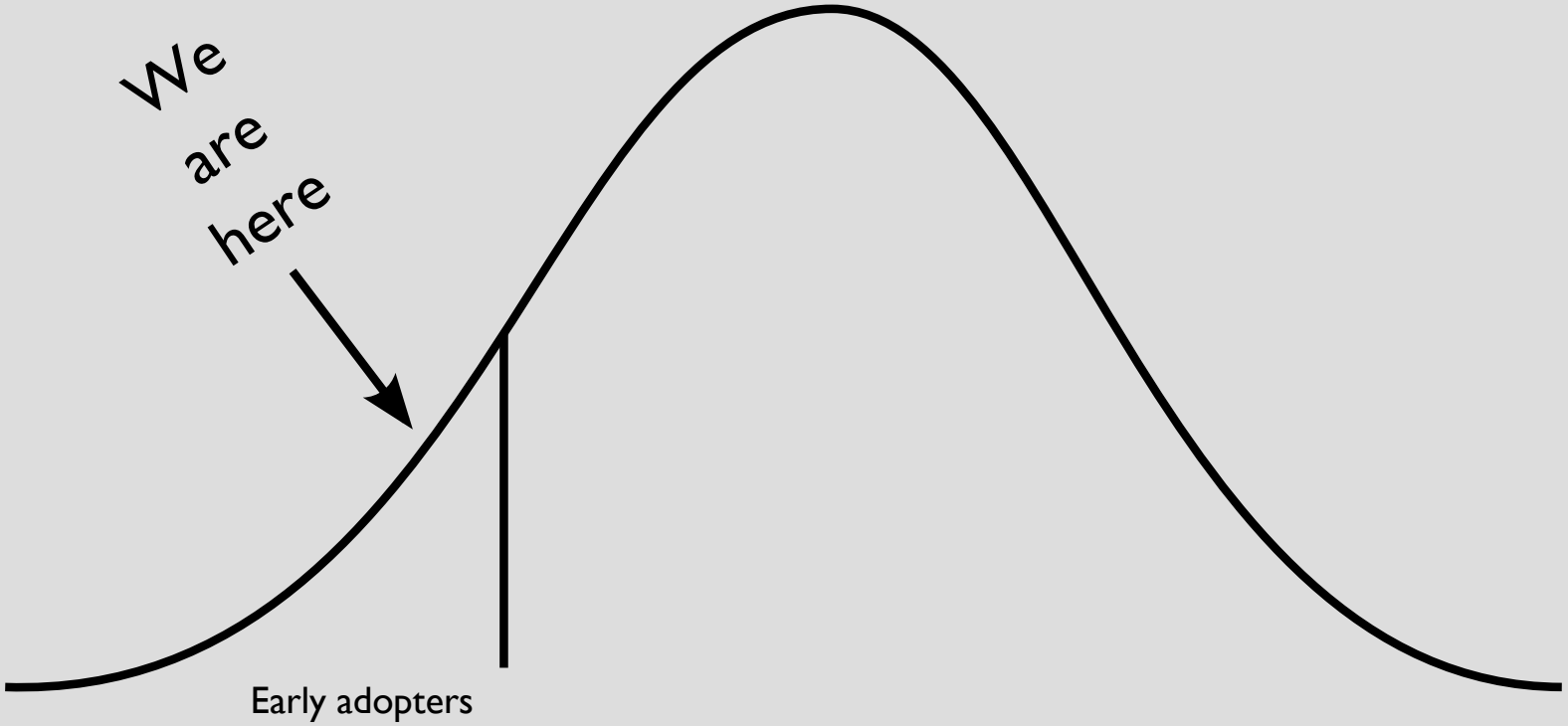
An experiment

A fairly controlled experiment

A "research project"?

A "skunkworks project"?

Early on the adopter curve


We
are
here
Early adopters

Sane defaults

Theory meeting practice

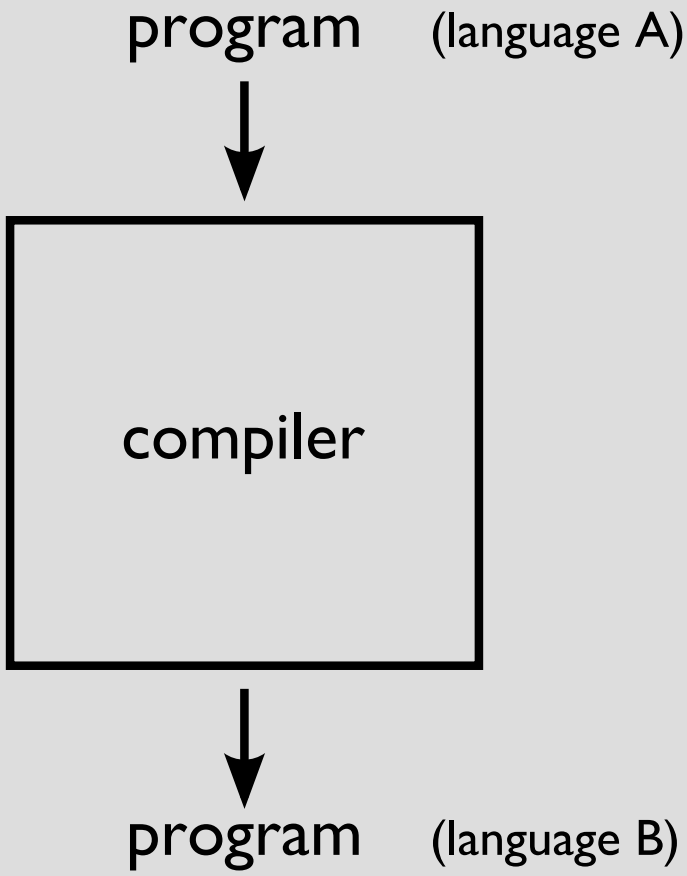
```
"In theory, there's no difference
between theory and practice.
..."
```

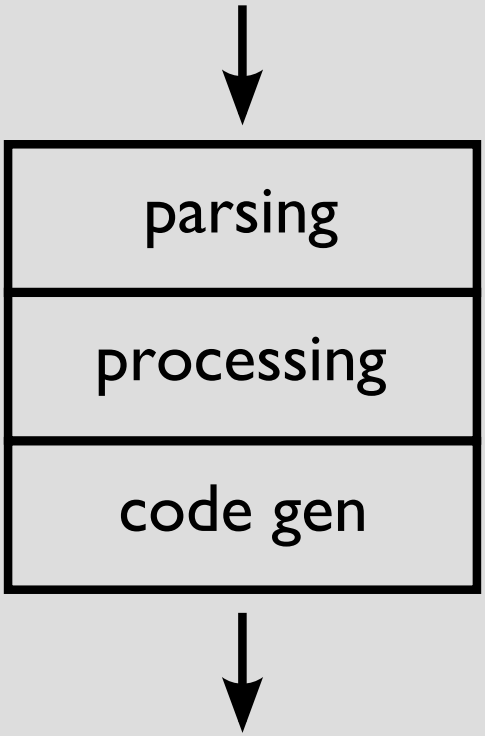
"In theory, there's no difference
between theory and practice.
In practice, there is."

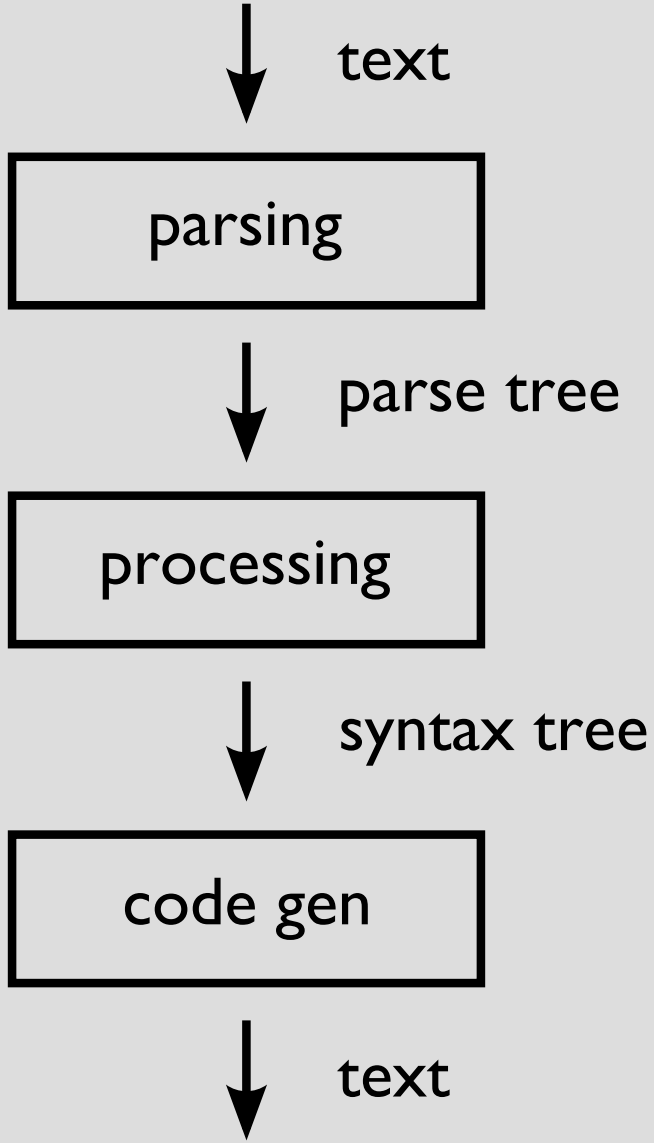
"In theory, there's no difference
between theory and practice,
even in practice." -- Scott Aaronson

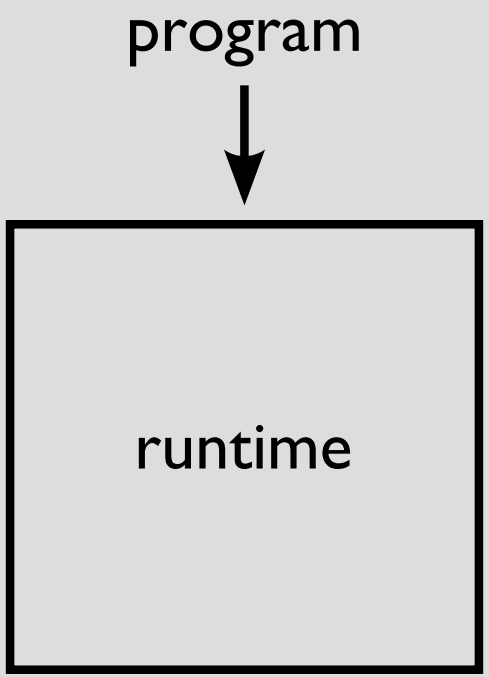
Yapsi

Yet Another Perl Six Implementation


program
(language A)
compiler
program
(language B)


parsing
processing
code gen

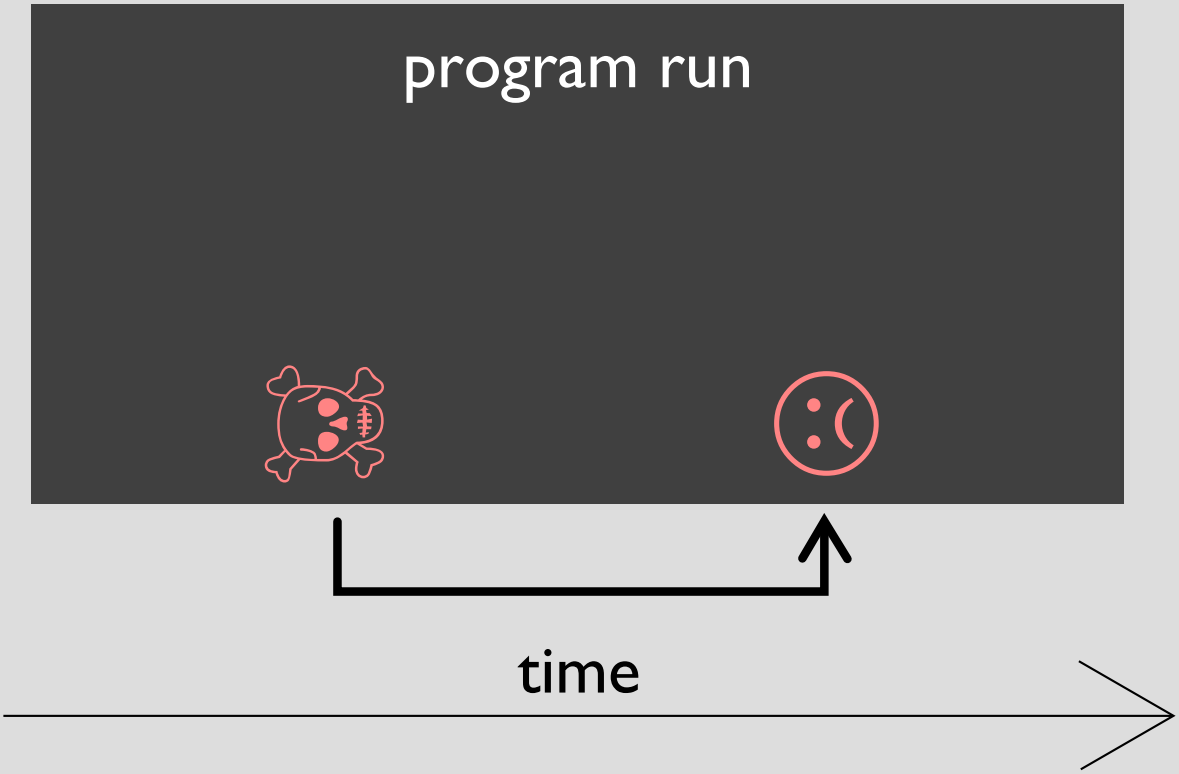

text
parsing
parse tree
processing
syntax tree
code gen
text

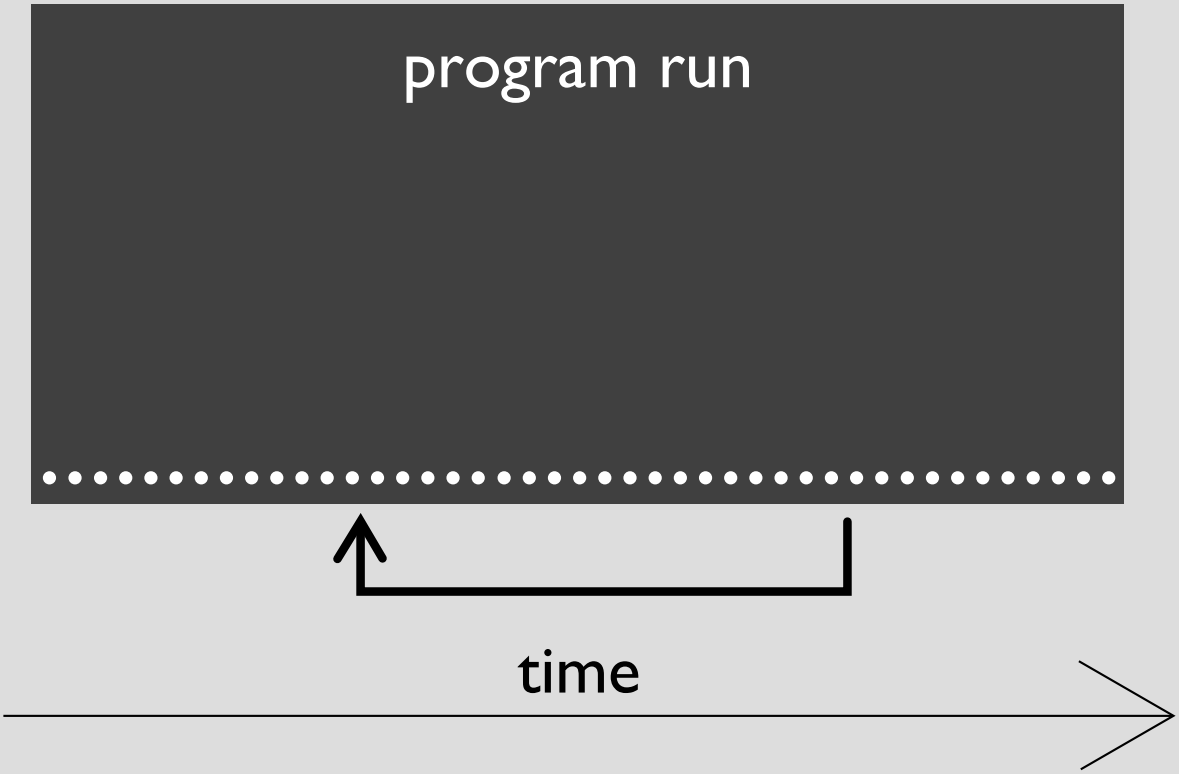

program
runtime

Tardis

Time-traveling debugger

Here's the problem:


program run
time


program run
time

Jump around in the program

Two models on time travel

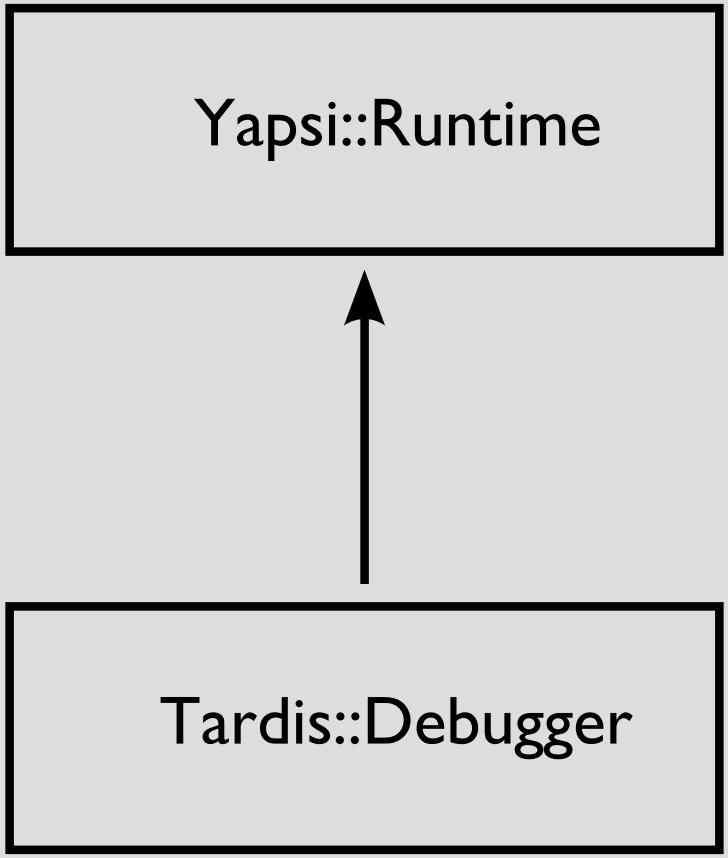
1. Can't kill grandfather

2. Can kill grandfather

If you kill a grandfather

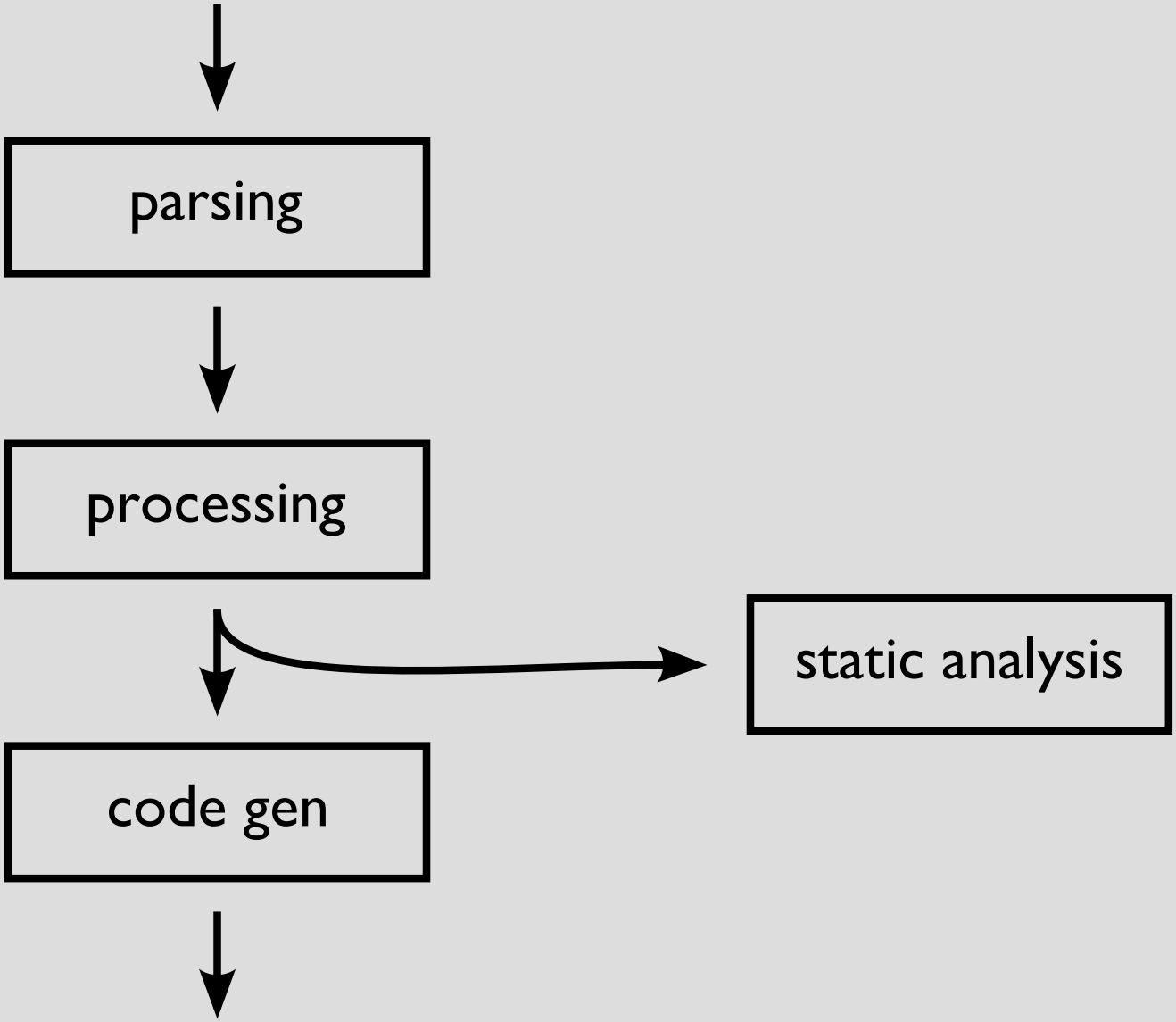
the future from that point is destroyed

Have to re-run from that point


Yapsi::Runtime
Tardis::Debugger

Sigmund

Static code analyzer


parsing
processing
static analysis
code gen

So many things blow up at runtime

Can be caught at CHECK time

```
my Int $a;
$a = 'OH HAI';
```

```
sub foo(Int $a) {
    ...
}

foo('OH HAI');
```

```
sub foo(Int $a) {
    $a = 42; # boom
}

foo(5);
```

```
sub foo() {
    ...
}

foo('OH HAI');
```

```
my $a;
say $a; # probably a mistake
$a = 42;
```

```
my $a = 42;

# $a is never used
```

```
if $a = 42 {
    ...
}
```

```
multi sub foo(Int $a) { ... }
multi sub foo(Int $b) { ... }

foo(42);    # boom
```

```
multi sub foo(Int $a?) { ... }
multi sub foo(Str $a?) { ... }

foo();       # boom
```

```
multi sub foo(Int $a, Int $b) { ... }
multi sub foo(Str $a, Str $b) { ... }

foo(42, 'OH HAI'); # boom
```

```
my ($a, $b) = 1, 2, 3;
# the 3 is lost!
```

```
our Int sub foo() {
    return 'OH HAI';
}
```

Halting problem

```
say 'OH HAI';
```

```
loop {
}
```

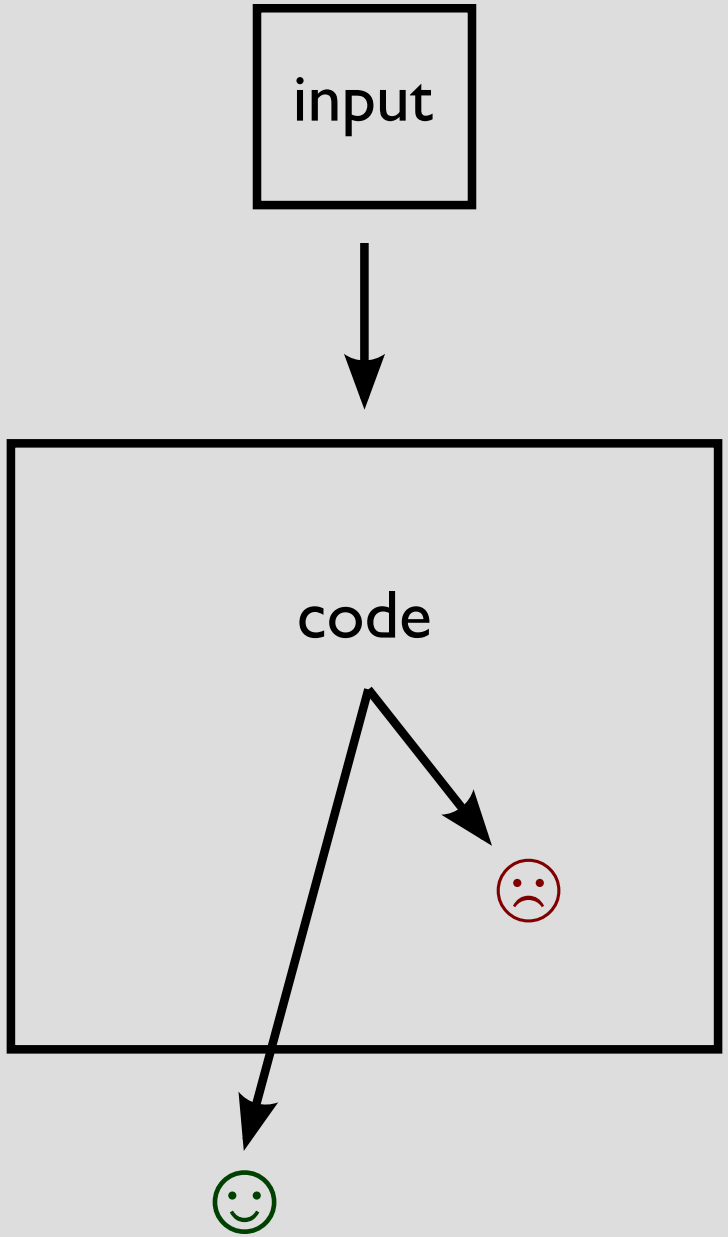
```
for 1..1337 {
    say 'OH HAI';
}
```

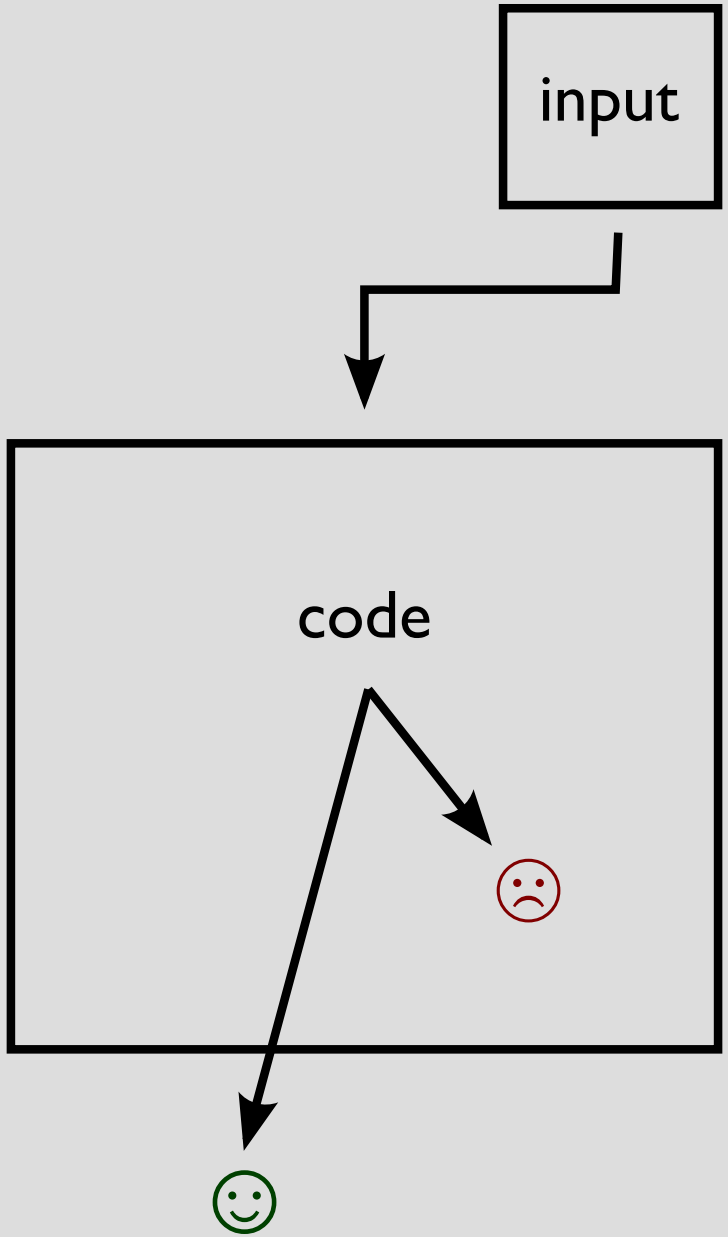
```
for 1..Inf {
    say 'OH HAI';
}
```

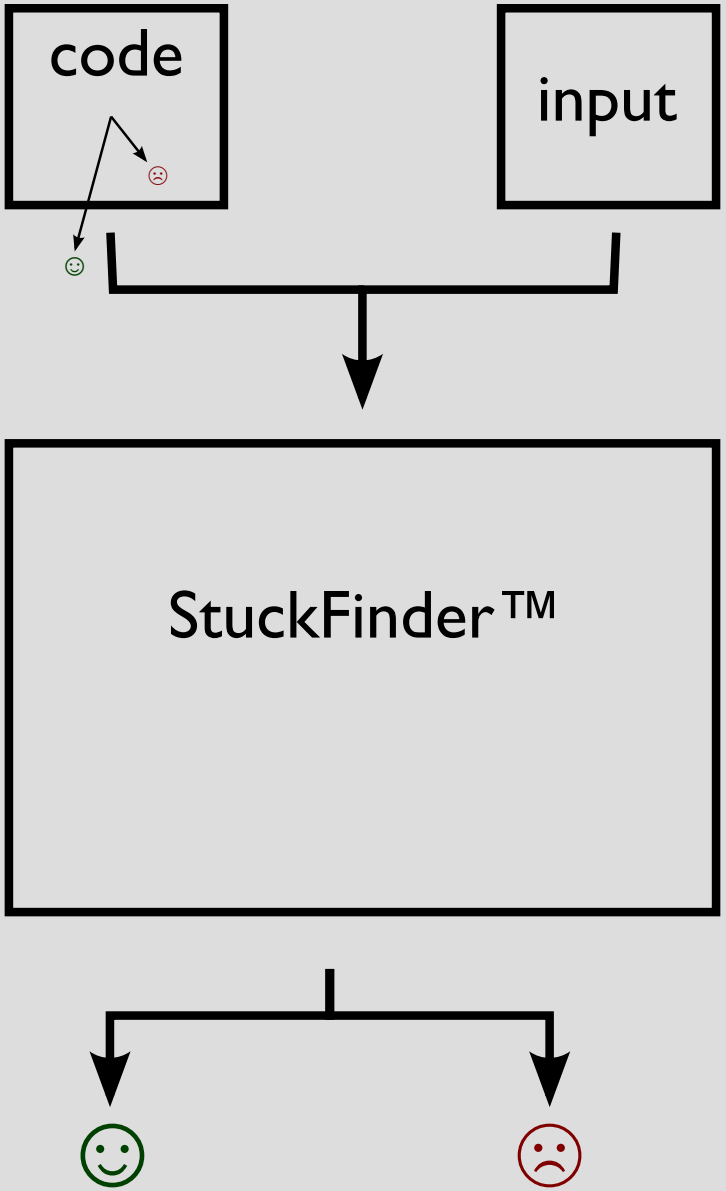
```
sub foo {
    foo;
}
foo;
```

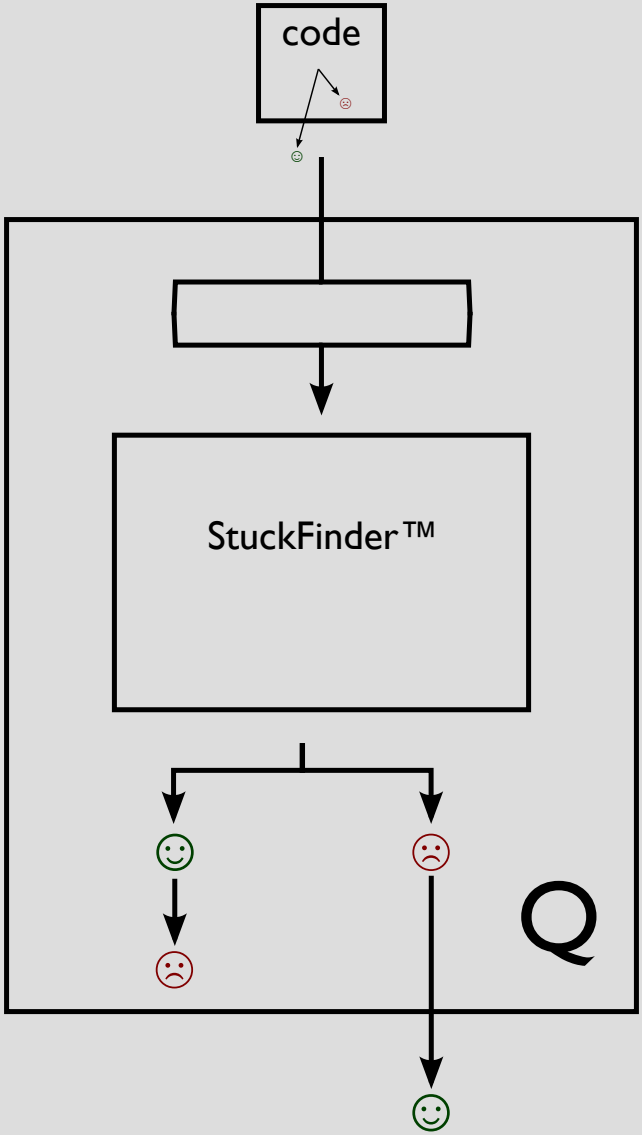
The limits of analysis

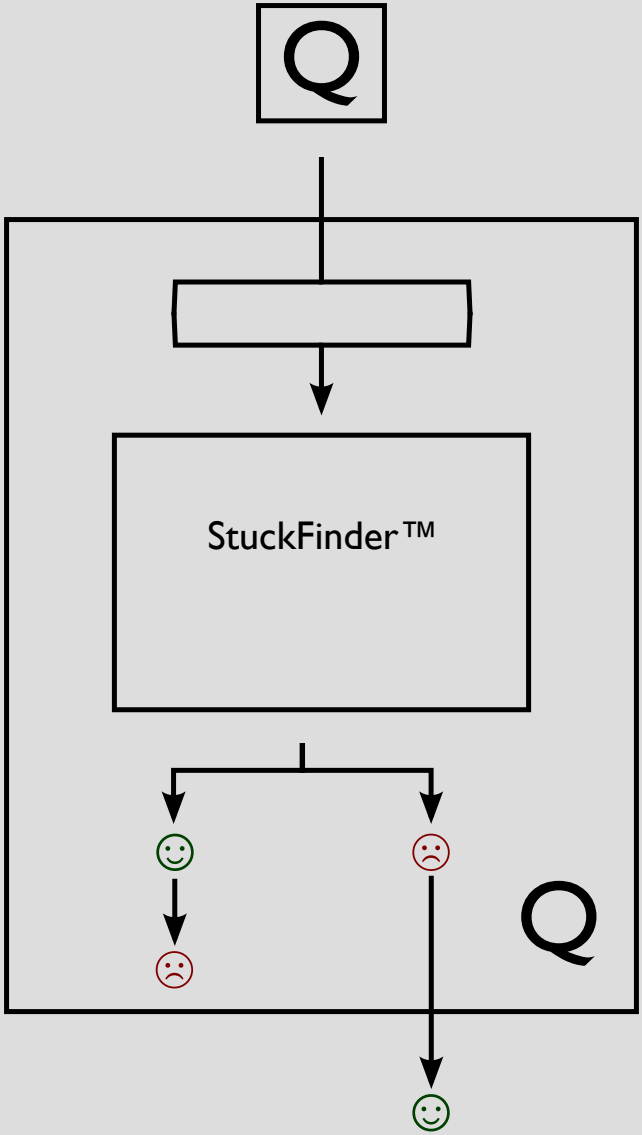
Tell me whether it halts!


input
code


input
code


code
input
StuckFinder™


code
StuckFinder™
Q


Q
StuckFinder™
Q

paradox

same as "I am lying"

same as "this statement is false"

"Scooping the Loop Snooper"

by Geoffrey K. Pullum

So, no StuckFinder™

Still, good analysis possible

Would help find errors early

Sigmund: finding errors ahead of time

Tardis: finding errors backwards in time

Tools are made of theory + practice

Let's build tools we really like

Спасибо.

Есть вопросы? Questions?