Automatic Programming: How Far Can Machines Go?

Hila Peleg

Technion

UNIX Programming

Programming

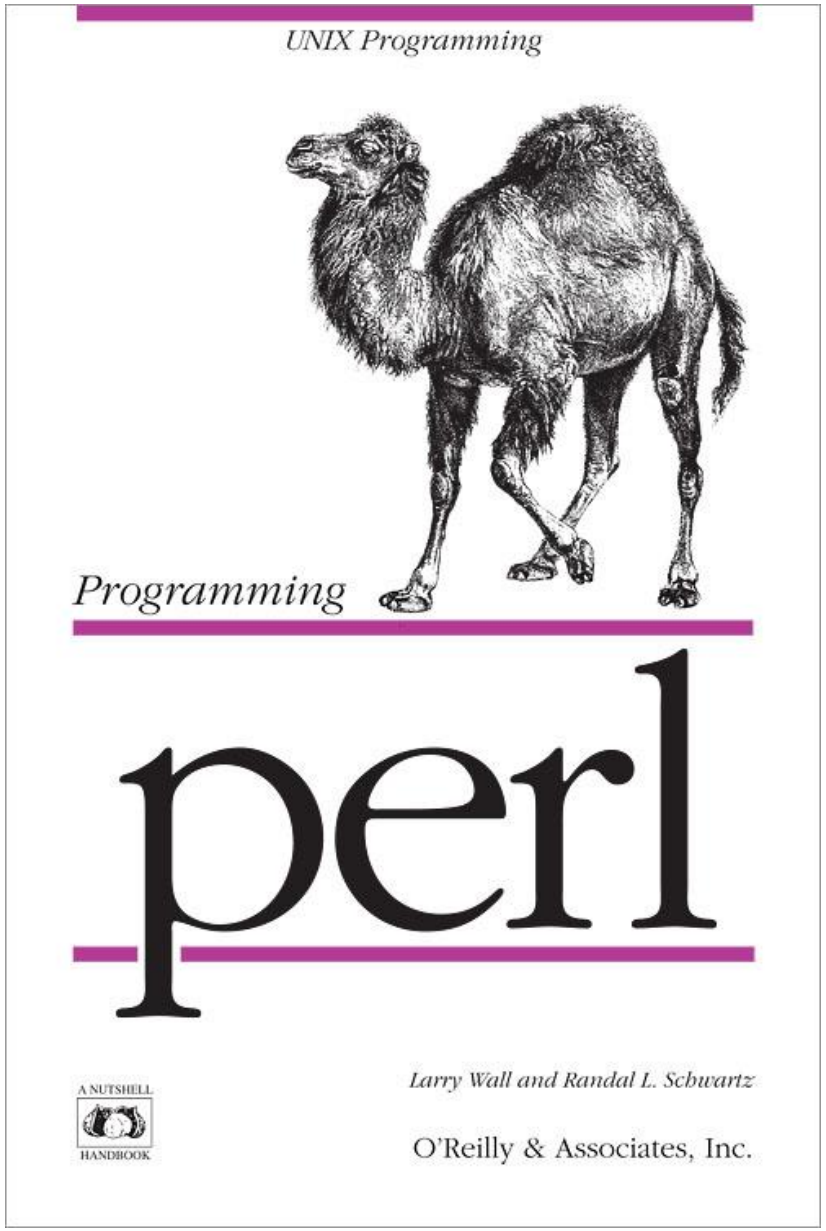
perl

A NUTSHELL HANDBOOK

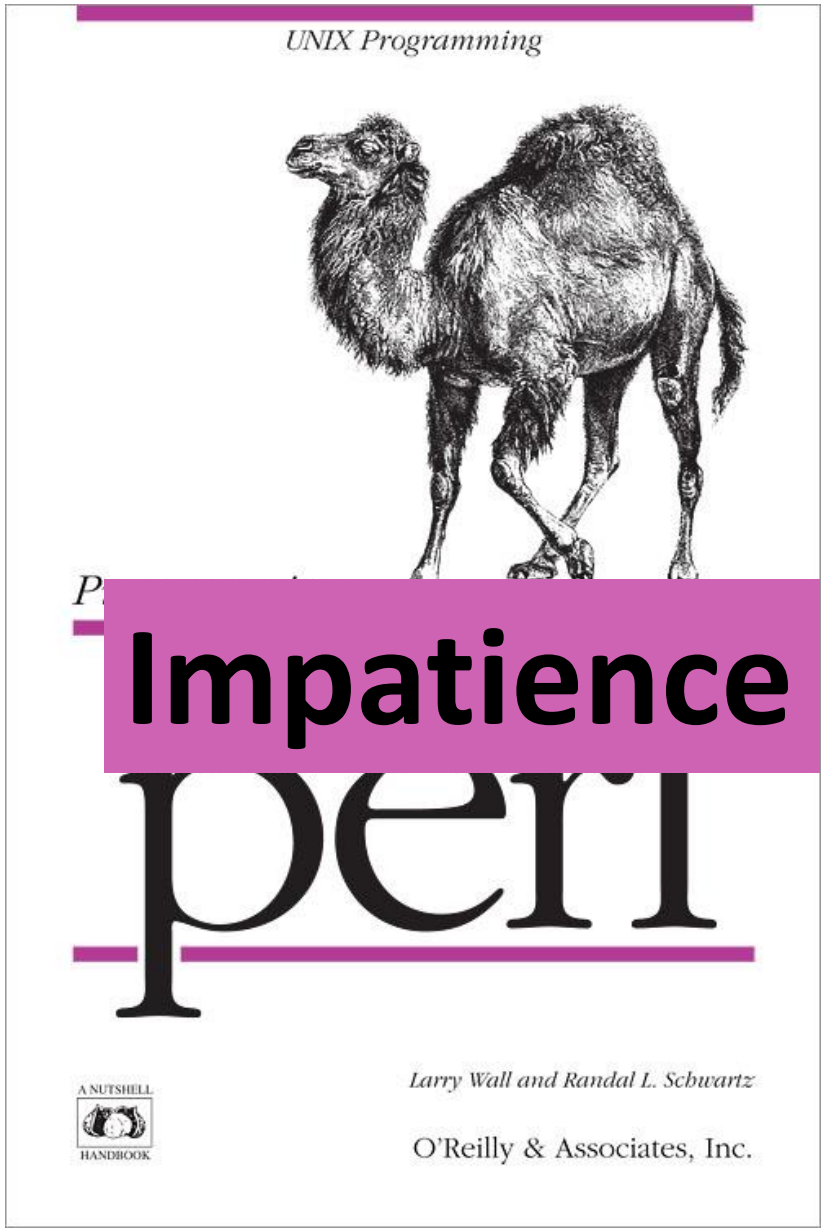
Larry Wall and Randal L. Schwartz

O'Reilly & Associates, Inc.

Laziness

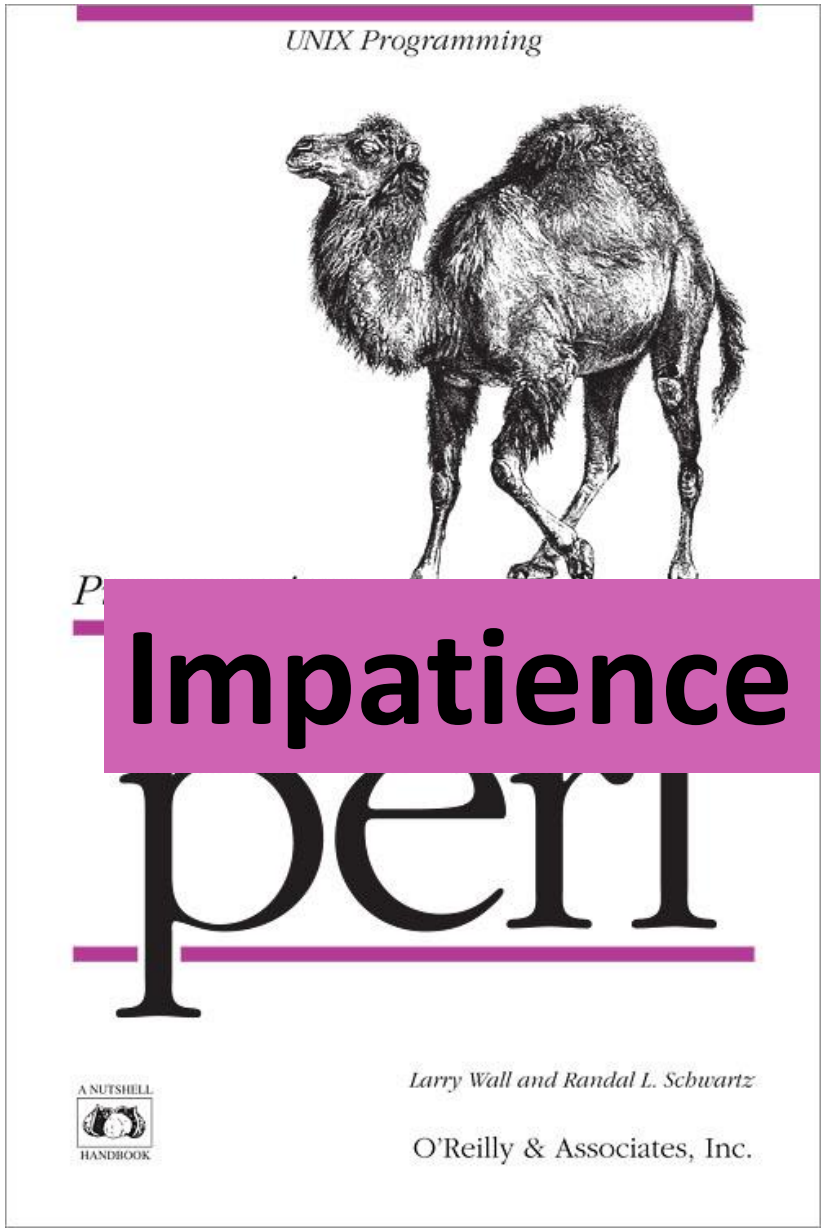

UNIX Programming
Programming
perl
A NUTSHELL
HANDBOOK
Larry Wall and Randal L. Schwartz
O'Reilly & Associates, Inc.

Laziness



Impatience

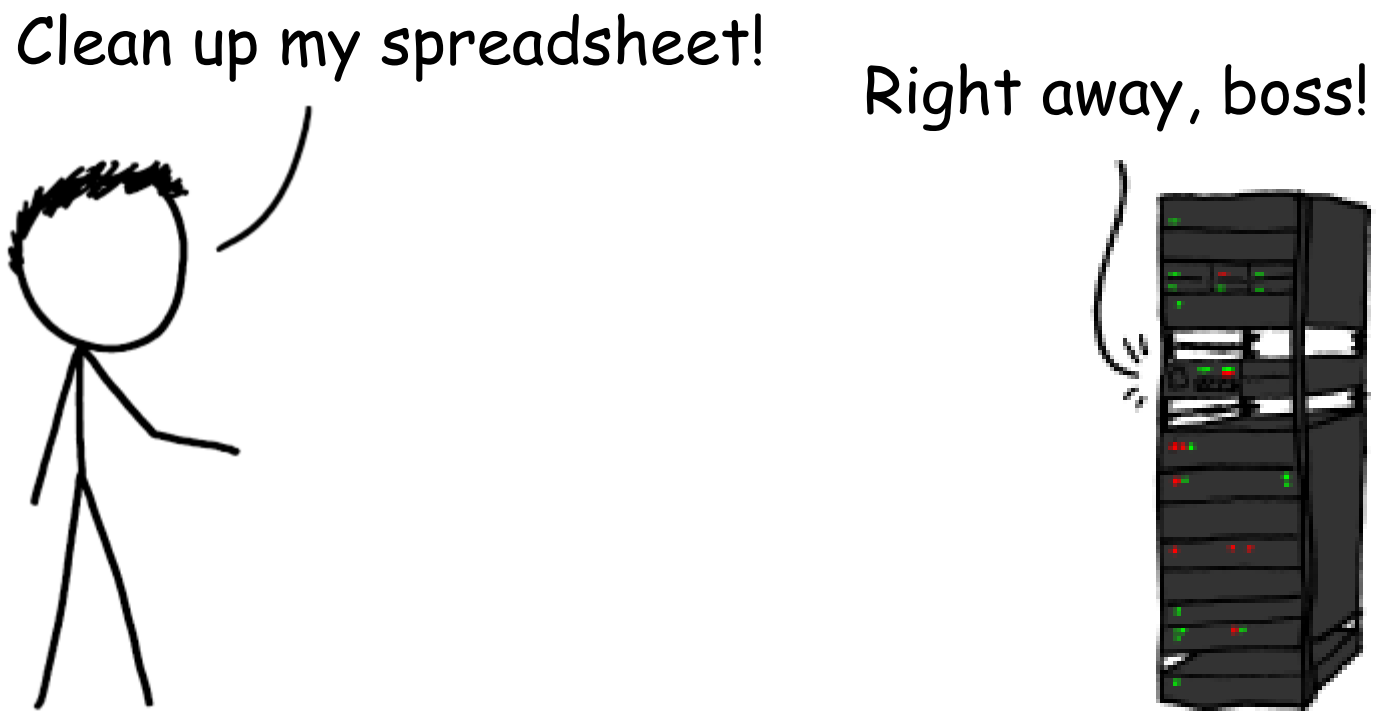
Laziness



Impatience

Hubris

Automatic Programming



Program synthesis to the rescue

APPLICATION OF RECURSIVE ARITHMETIC TO THE
PROBLEM OF CIRCUIT SYNTHESIS

by Alonzo Church

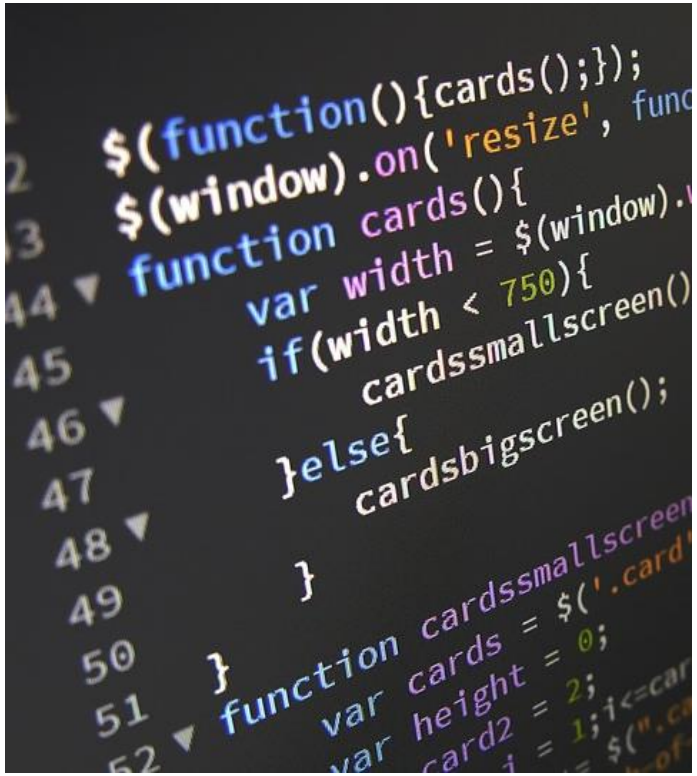
A paper presented at the Summer Institute of Symbolic Logic at Ithaca, N. Y. , in July, 1957 - with revisions made in August , 1957.

Program Synthesis

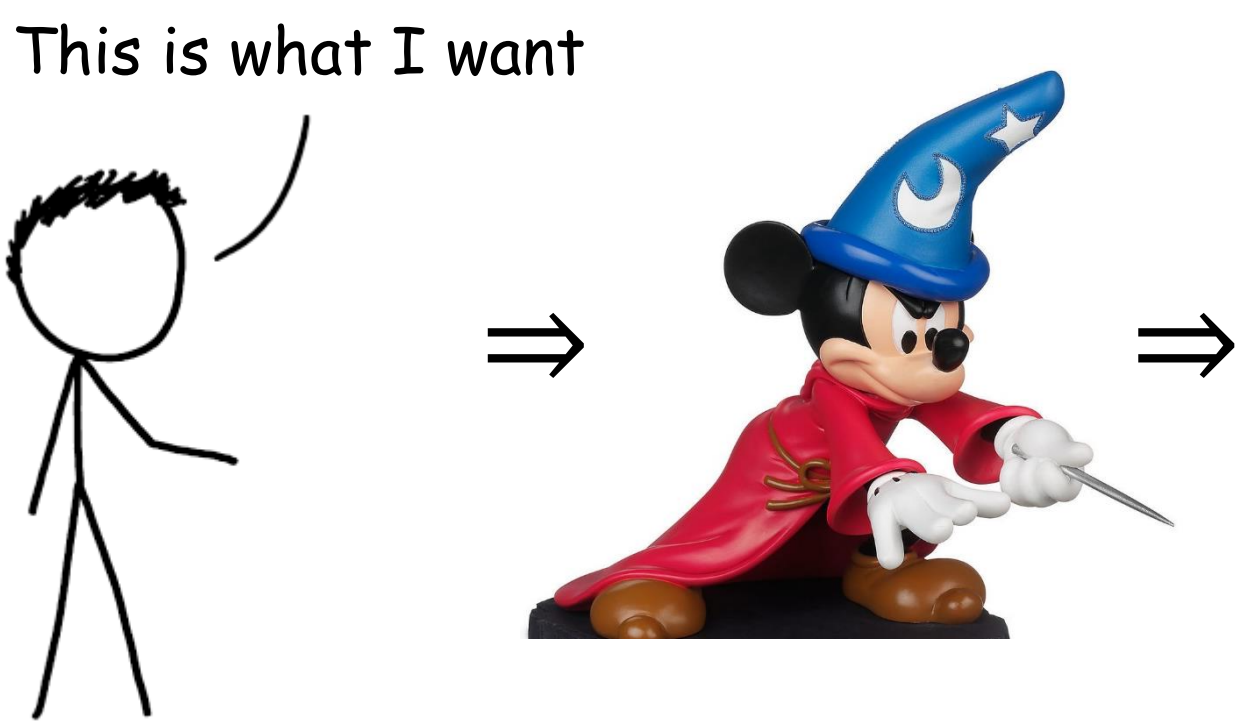
$$
\begin{array}{l}
\wedge\ \forall p \in Proc, d \in Disk\ : \\
\quad \wedge\ (d \in disksWritten[p]) \Rightarrow \wedge\ phase[p] \in \{1,2\} \\
\qquad\qquad\qquad\qquad \wedge\ disk[d][p] = dblock[p] \\
\quad \wedge\ (phase[p] \in 1,2) \Rightarrow \wedge\ (blocksRead[p][d] \neq \{\}) \Rightarrow \\
\qquad\qquad\qquad\qquad\qquad\qquad (d \in disksWritten[p]) \\
\qquad\qquad\qquad\qquad\quad \wedge\ \neg hasRead(p, d, p) \\
\wedge\ \forall p \in Proc\ : \\
\quad \wedge\ (phase[p] = 0) \Rightarrow \wedge\ dblock[p] = InitDB \\
\qquad\qquad\qquad\qquad\quad \wedge\ disksWritten[p] = \{\} \\
\qquad\qquad\qquad\qquad\quad \wedge\ \forall d \in Disk\ :\ \forall br \in blocksReaa \\
\qquad\qquad\qquad\qquad\qquad\quad \wedge\ br.proc = p \\
\qquad\qquad\qquad\qquad\qquad\quad \wedge\ br.block = disk[d][p] \\
\quad \wedge\ (phase[p] \neq 0) \Rightarrow \wedge\ dblock[p].mbal \in Ballot(p) \\
\qquad\qquad\qquad\qquad\quad \wedge\ dblock[p].bal \in Ballot(p) \cup \{0\} \\
\qquad\qquad\qquad\qquad\quad \wedge\ \forall d \in Disk\ :\ \forall br \in blocksReaa \\
\qquad\qquad\qquad\qquad\qquad\quad br.block.mbal < dblock[p]. \\
\quad \wedge\ (phase[p] \in \{2,3\}) \Rightarrow (dblock[p].bal = dblock[p].mt \\
\quad \wedge\ output[p] = \text{IF } phase[p] = 3 \text{ THEN } dblock[p].inp \text{ E} \\
\wedge\ chosen \in allInput \cup \{NotAnInput\} \\
\wedge\ \forall p \in Proc\ :\ \wedge\ input[p] \in allInput \\
\qquad\qquad\qquad\quad \wedge\ (chosen = NotAnInput) \Rightarrow (output[p]
\end{array}
$$

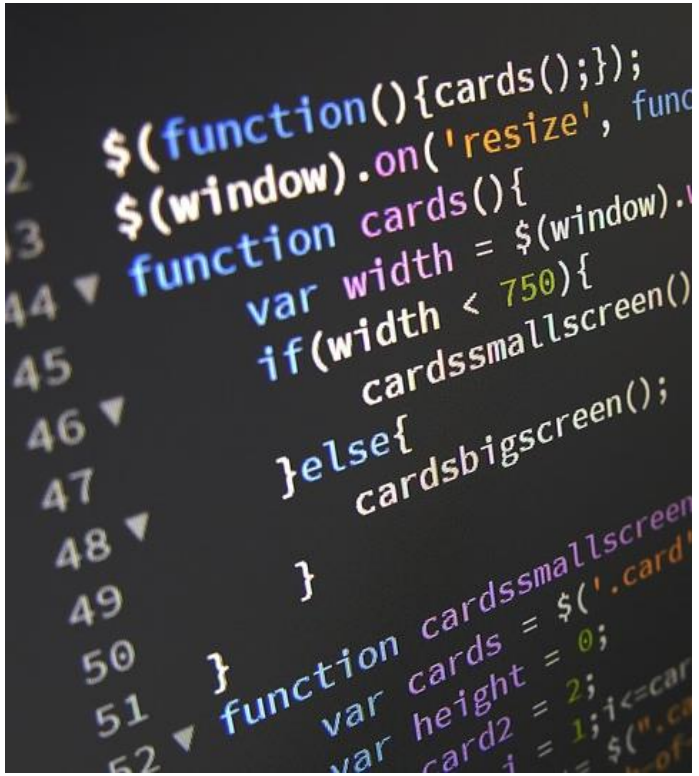
$\Rightarrow$

$\Rightarrow$



What we really want





What if we have automatic programming?

- Hyper-intelligent program generation for your every need

What if we have automatic programming?

- Hyper-intelligent program generation for your every need
- Self-aware, self-augmenting AI

What if we have automatic programming?

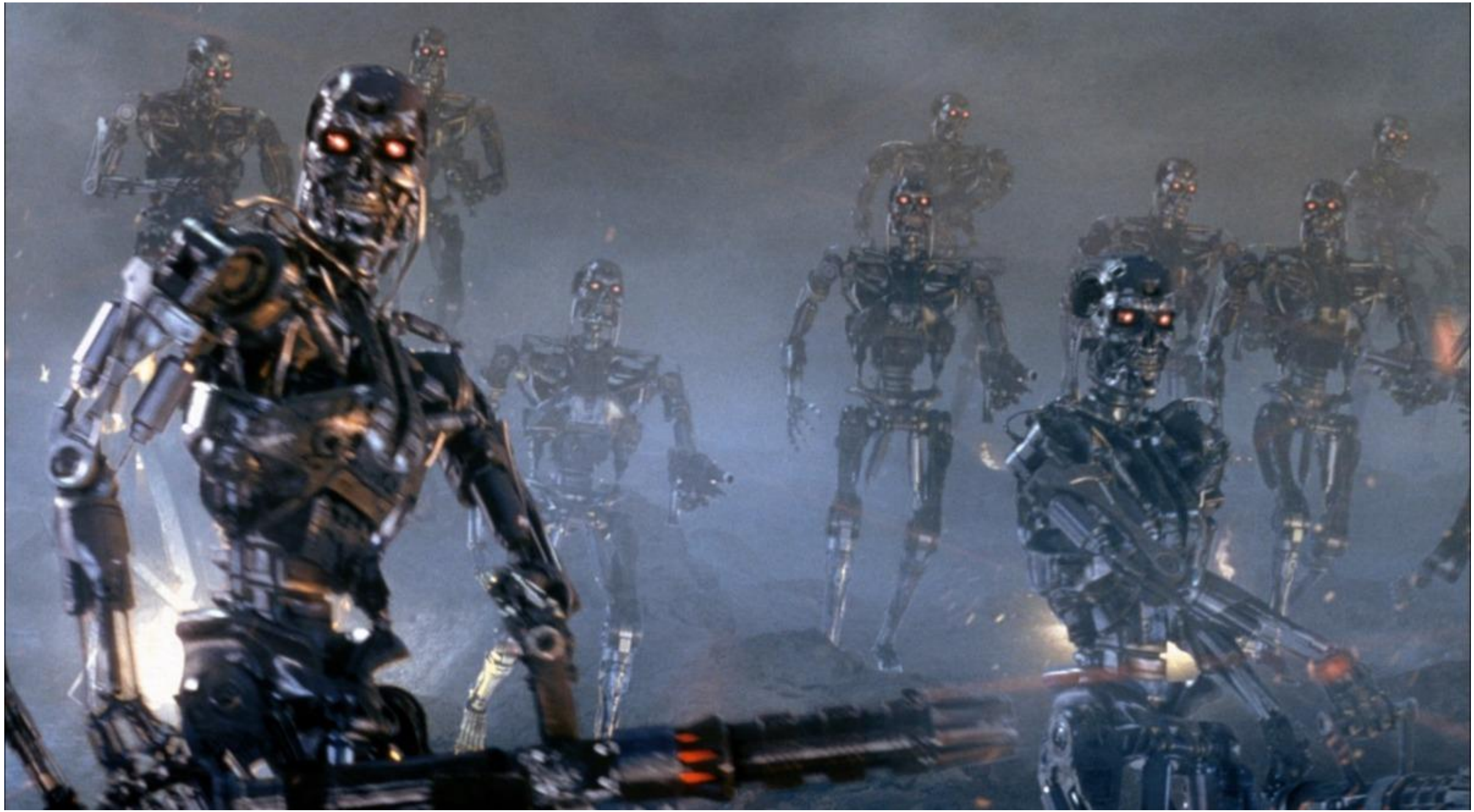
- Hyper-intelligent program generation for your every need
- Self-aware, self-augmenting AI
- The singularity

What if we have automatic programming?

- Hyper-intelligent program generation for your every need
- Self-aware, self-augmenting AI
- The singularity
- ???

What if we have automatic programming?

- Hyper-intelligent program generation for your every need
- Self-aware, self-augmenting AI
- The singularity
- ???
- THE ROBOT APOCALYPSE



But it’s ok!

But it’s ok!

Understand users

Build a program

But it’s ok!

Understand users

Build a program

Generalizing partial intent is hard

But it’s ok!

Understand users

Build a program

Generalizing partial intent is hard

Solving HALT is hard*

```
() {
private void demo() throws Exception {

    URLConnection s = new URL(url).openConnection();
    InputStreamReader content =
```

- `new InputStreamReader(s.getInputStream());`
- `…new java.io.InputStreamReader(dstream,url);`
- `…new InputStreamReader(inputStream,url);`

Completions by Codota

codota

ENTER JAVASCRIPT

NICIFY JAVASCRIPT

```
1  // Put your JavaScript here that you want to rename,
   deobfuscate,
2  // or infer types for:
3  function chunkData(e, t) {
4    var n = [];
5    var r = e.length;
6    var i = 0;
7    for (; i < r; i += t) {
8      if (i + t < r) {
9        n.push(e.substring(i, i + t));
10     } else {
11       n.push(e.substring(i, r));
12     }
13   }
14   return n;
15 }
16 // You can also use s
```

ETH zürich

```
dog.drink(h0);
}
if (dog == null)
    return;
dog.drink(h0);
}
```

1

```
{
  dog.drink(milk);
}
```

```
{
  if (dog == null)
    return;
  dog.drink(milk);
}
```

errorVar: dog

1

```
{
  bowl.fill();
  dog.drink(water);
}
```

```
{
  bowl.fill();
  if (dog == null)
    return;
  dog.drink(water);
}
```

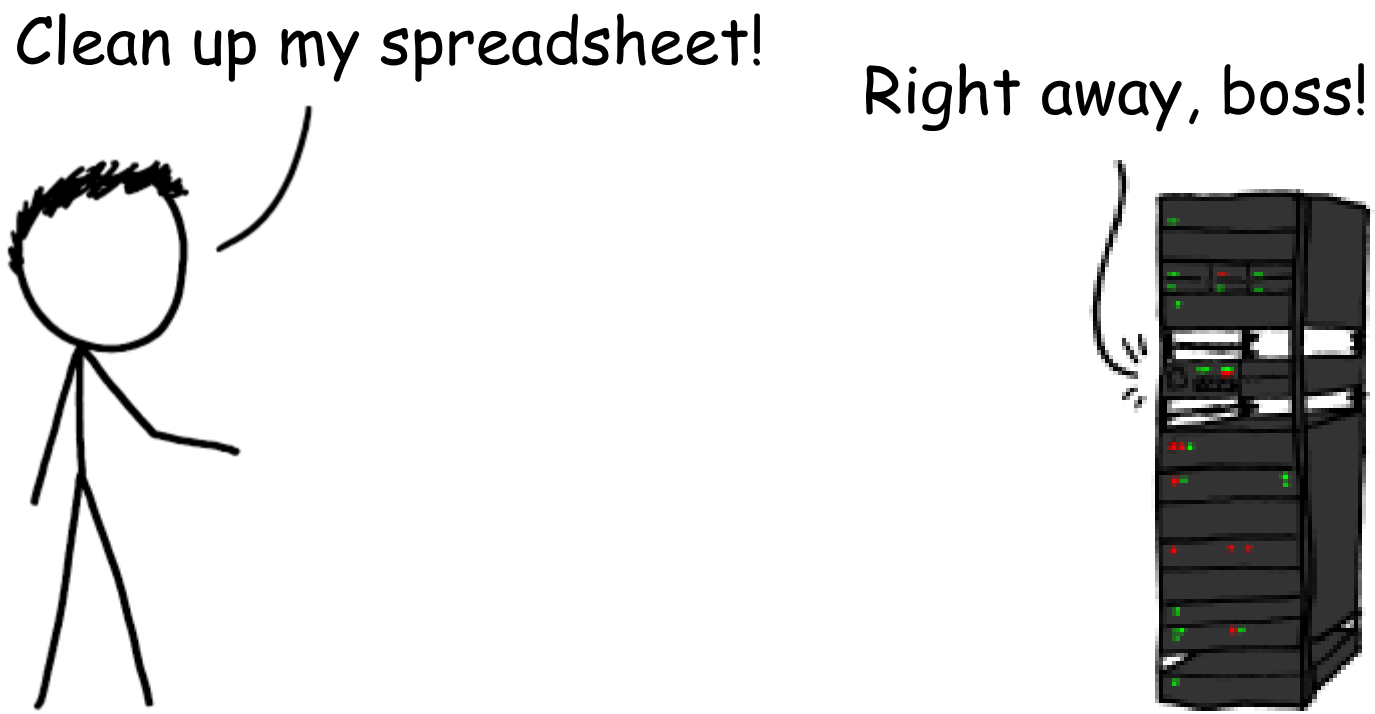
errorVar: dog

Merge edits which check `dog` for null before calling `dog.drink()` (either `milk` or `water`)

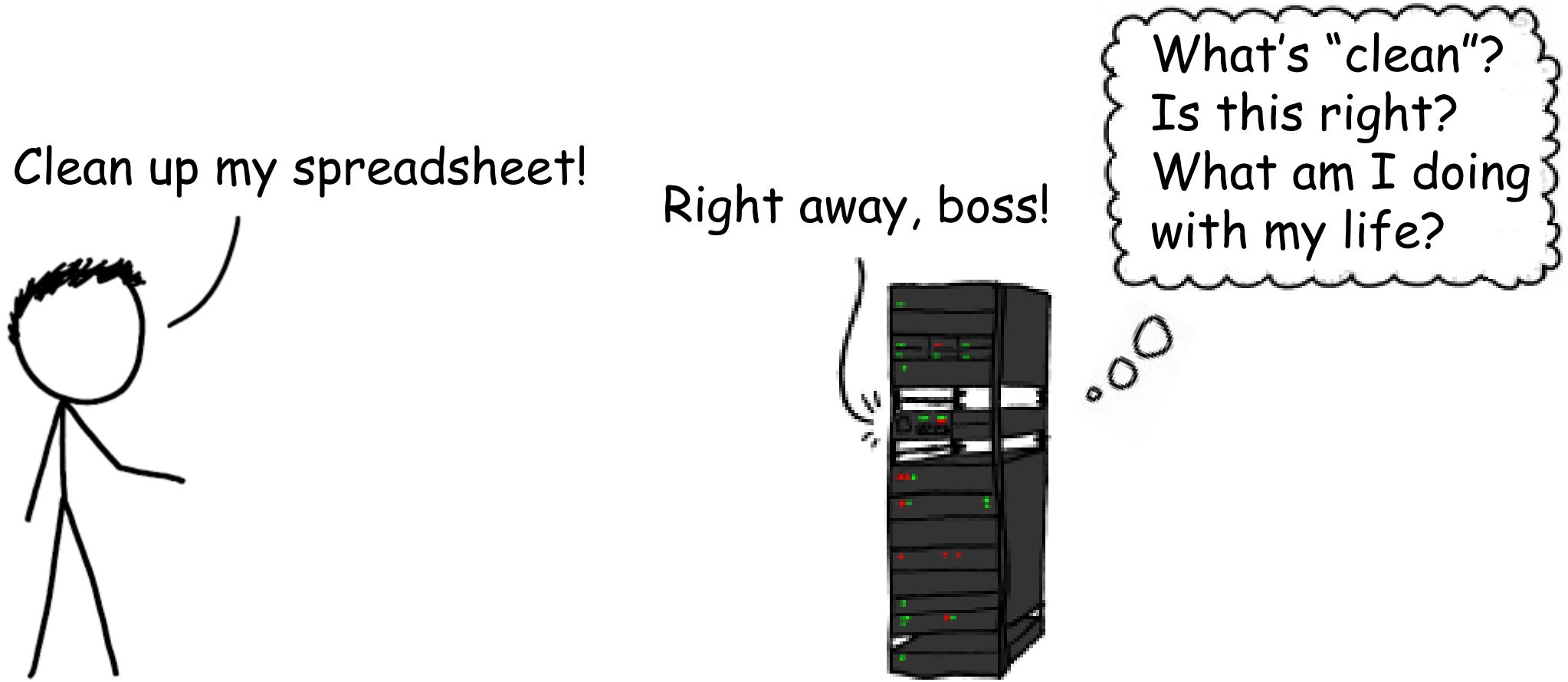
```
input: 2017   output: 2017   (["2", "20", "201", "2017"])

[0,1,2].map(i => input.toString().slice(0,i))
input: 420    output: ["", "4", "42"]   (["4", "42", "420"])
input: 2017   output: ["", "2", "20"]   (["2", "20", "201", "2017"])

[1,input.toString().length].map(i => input.toString().slice(0,i))
input: 420    output: ["4", "420"]    (["4", "42", "420"])
input: 2017   output: ["2", "2017"]   (["2", "20", "201", "2017"])

Array.apply(null,{'length'
ing().length}).map((e,i) => i + 1).map(i => input.toString()
input: 420    output: ["4",
input: 2017   output: ["2",
]
```

Retain
Exclude
Fix this
Add input

User intent is hard



User intent is hard



Intent via examples

	A	B	C	D
1	First Name	Last Name	Time	Message
2	Simon	Raik-Allen	16:40	Hi, Simon, just a reminder your talk is at 16:40
3	Aino	Corry	16:55	
4	Michelle	Casbon	15:55	
5	Mikael	Vidstedt	10:30	
6	Sam	Aaron	16:40	
7	Anita	Sengupta	17:40	
8	Jessica	Kerr	09:00	
9	Dave	Thomas	11:30	
10	Chris	Richardson	14:35	

Intent via examples

	A	B	C	D
1	First Name	Last Name	Time	Message
2	Simon	Raik-Allen	16:40	Hi, Simon, just a reminder your talk is at 16:40
3	Aino	Corry	16:55	Hi, Aino, just ceminder your talk is at 16:55
4	Michelle	Casbon	15:55	Hi, Michelle, just ceminder your talk is at 15:55
5	Mikael	Vidstedt	10:30	Hi, Mikael, just veminder your talk is at 10:30
6	Sam	Aaron	16:40	Hi, Sam, just aeminder your talk is at 16:40
7	Anita	Sengupta	17:40	Hi, Anita, just seminder your talk is at 17:40
8	Jessica	Kerr	09:00	Hi, Jessica, just keminder your talk is at 09:00
9	Dave	Thomas	11:30	Hi, Dave, just teminder your talk is at 11:30
10	Chris	Richardson	14:35	Hi, Chris, just reminder your talk is at 14:35

Intent via examples

	A	B	C	D
1	First Name	Last Name	Time	Message
2	Simon	Raik-Allen	16:40	Hi, Simon, just a reminder your talk is at 16:40
3	Aino	Corry	16:55	Hi, Aino, just ceminder your talk is at 16:55
4	Michelle	Casbon	15:55	Hi, Michelle, just ceminder your talk is at 15:55
5	Mikael	Vidstedt	10:30	Hi, Mikael, just veminder your talk is at 10:30
6	Sam	Aaron	16:40	Hi, Sam, just aeminder your talk is at 16:40
7	Anita	Sengupta	17:40	Hi, Anita, just seminder your talk is at 17:40
8	Jessica	Kerr	09:00	Hi, Jessica, just keminder your talk is at 09:00
9	Dave	Thomas	11:30	Hi, Dave, just teminder your talk is at 11:30
10	Chris	Richardson	14:35	Hi, Chris, just reminder your talk is at 14:35

Intent via examples

	A	B	C	D
1	First Name	Last Name	Time	Message
2	Simon	Raik-Allen	16:40	Hi, Simon, just a reminder your talk is at 16:40
3	Aino	Corry	16:55	Hi, Aino, just ceminder your talk is at 16:55
4	Michelle	Casbon	15:55	Hi, Michelle, just ceminder your talk is at 15:55
5	Mikael	Vidstedt	10:30	Hi, Mikael, just veminder your talk is at 10:30
6	Sam	Aaron	16:40	Hi, Sam, just aeminder your talk is at 16:40
7	Anita	Sengupta	17:40	Hi, Anita, just seminder your talk is at 17:40
8	Jessica	Kerr	09:00	Hi, Jessica, just keminder your talk is at 09:00
9	Dave	Thomas	11:30	Hi, Dave, just teminder your talk is at 11:30
10	Chris	Richardson	14:35	Hi, Chris, just reminder your talk is at 14:35

Query knowledge about (some kind of) code

I want to apply foo() to
requests to my http server

Query knowledge about (some kind of) code

I want to apply foo() to requests to my http server


```
val routes :  Route = ???
val bindingFuture = Http().
    bindAndHandle(routes,
                  "localhost",
                  8080)
```

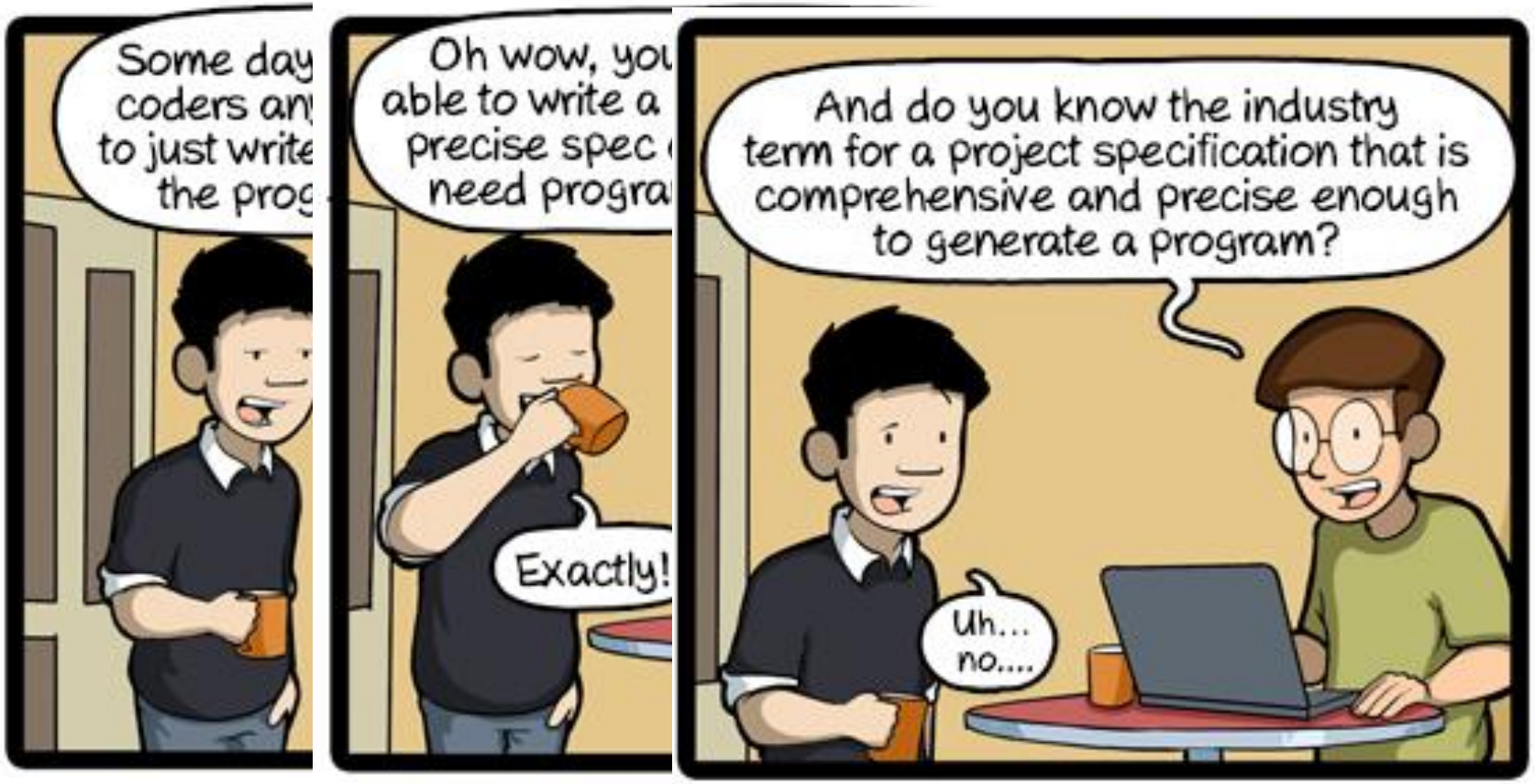
Commit Strip said it best…

http://www.commitstrip.com/en/2016/08/25/a-very-comprehensive-and-precise-spec/

Commit Strip said it best…

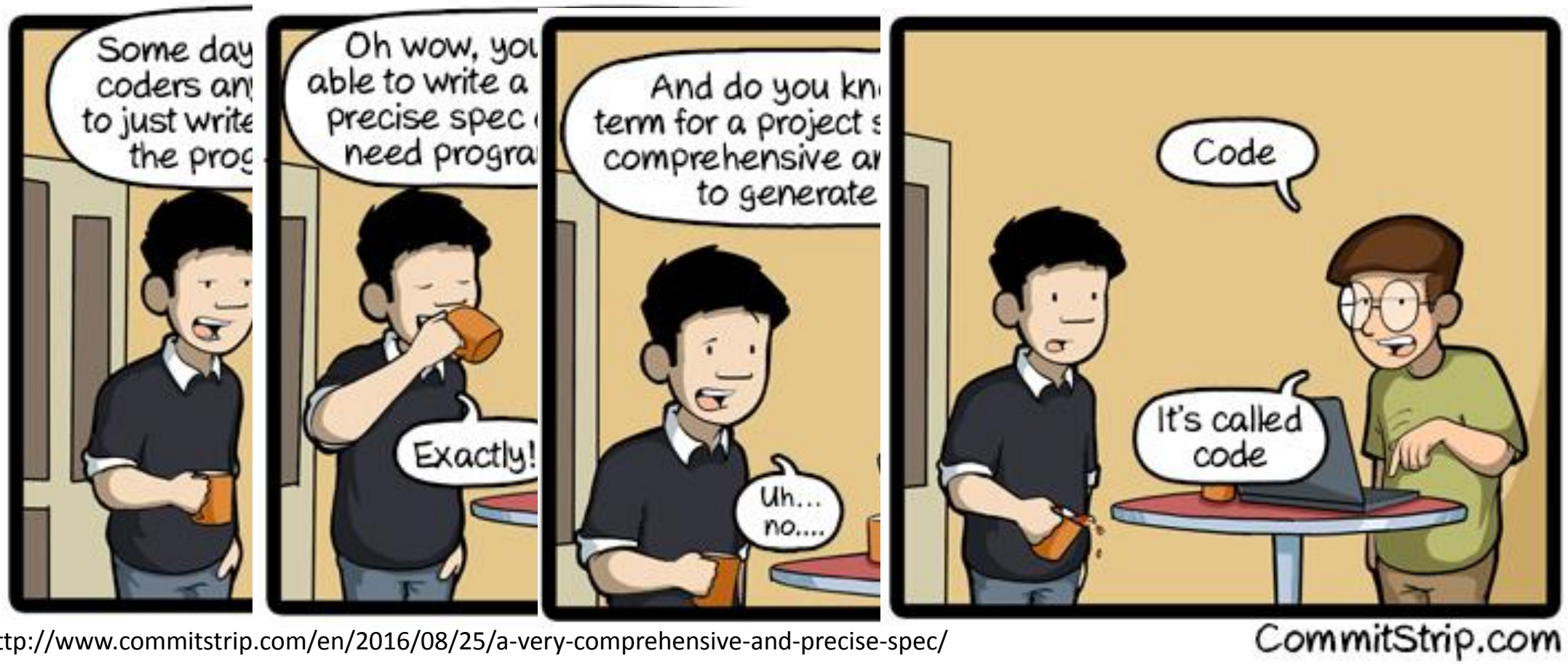
http://www.commitstrip.com/en/2016/08/25/a-very-comprehensive-and-precise-spec/

Commit Strip said it best…



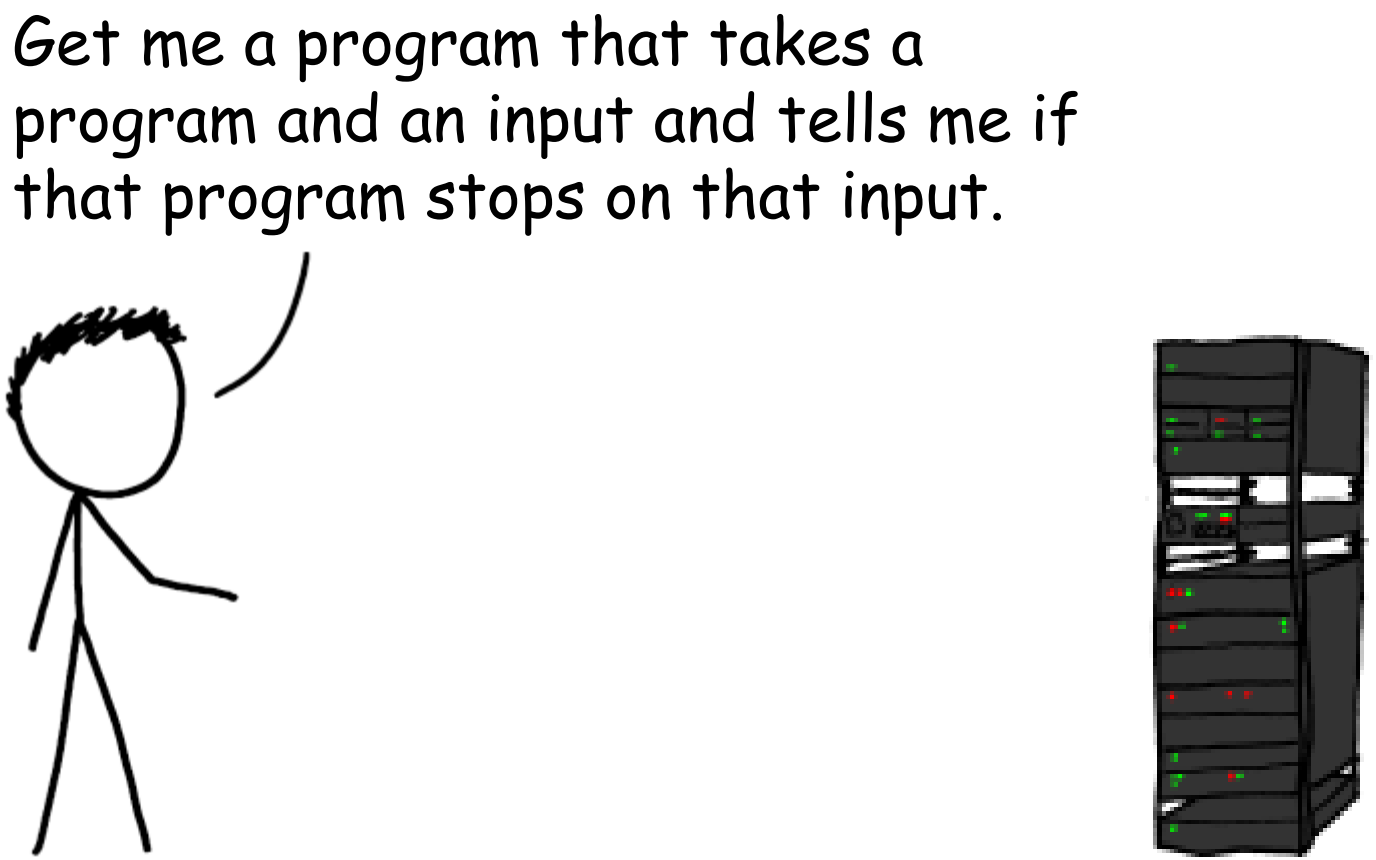
http://www.commitstrip.com/en/2016/08/25/a-very-comprehensive-and-precise-spec/

Commit Strip said it best…

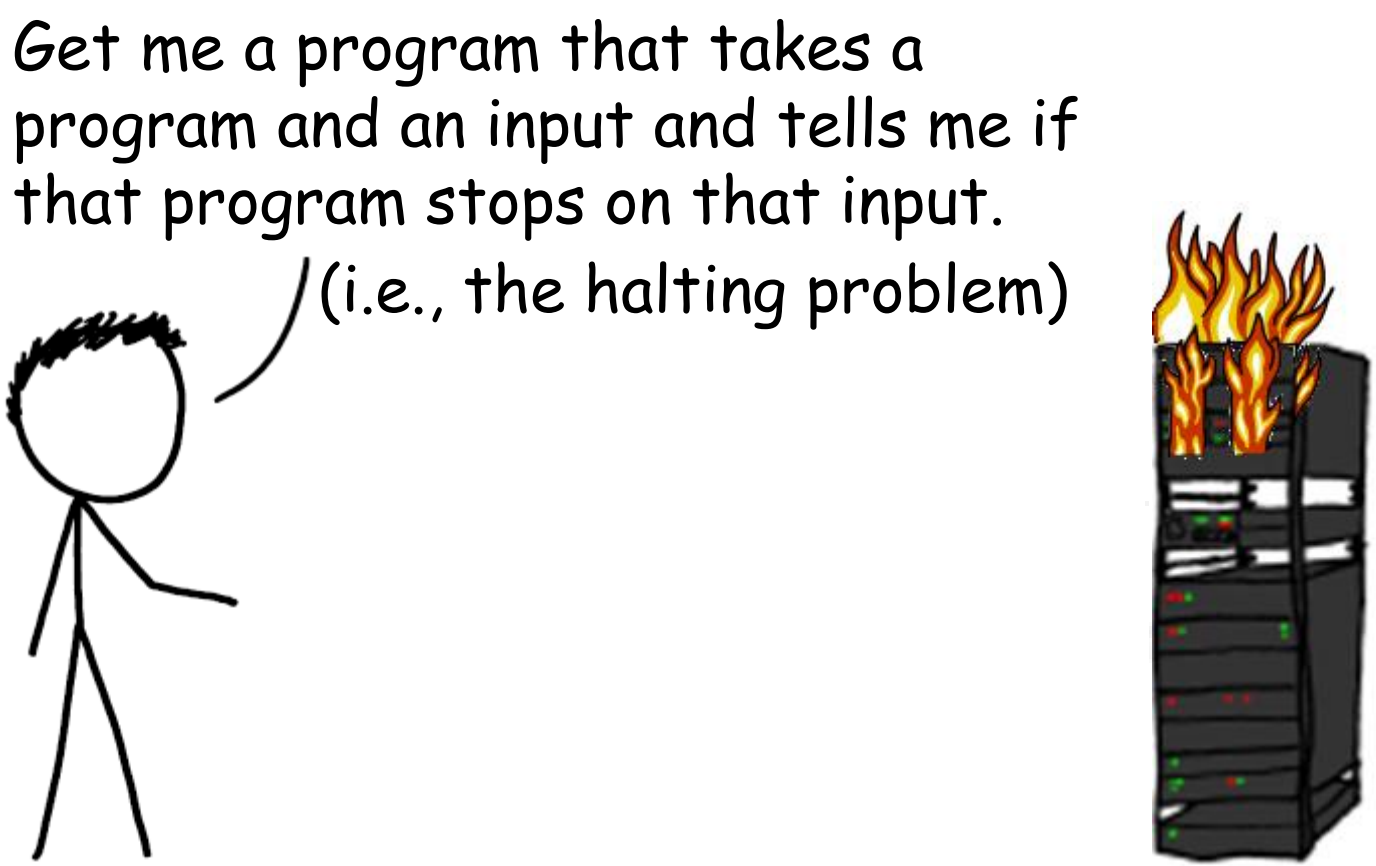


http://www.commitstrip.com/en/2016/08/25/a-very-comprehensive-and-precise-spec/

Building a program is also hard*



Building a program is also hard*

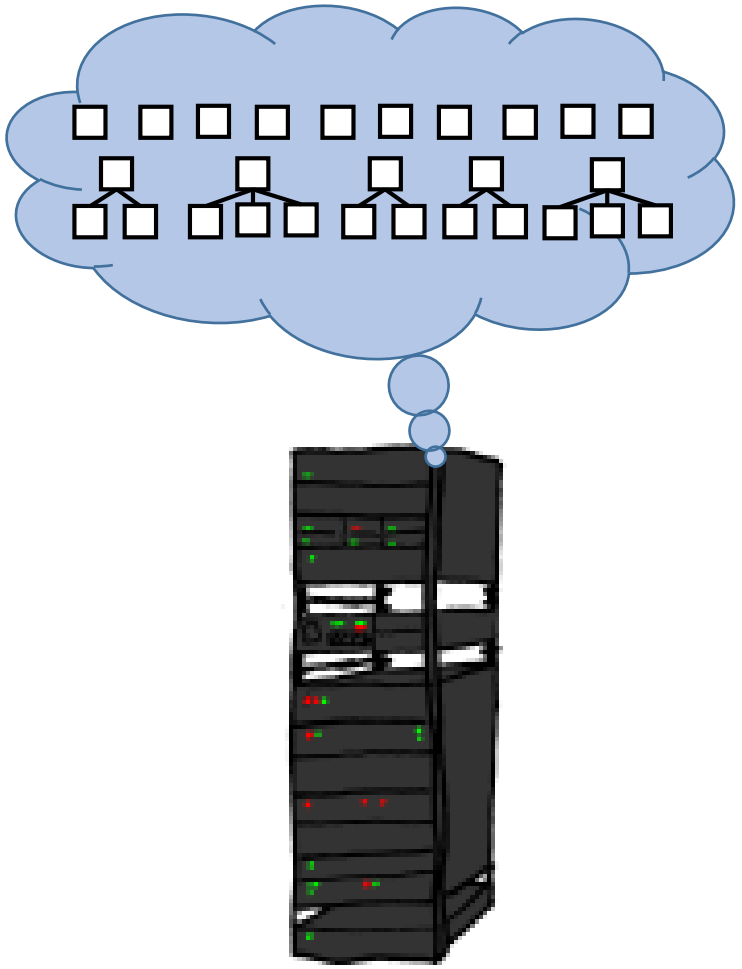


Adjusting our expectations

Here's a grammar of 20 functions and 10 constants, get me a program that I'm *certain* is in this space.

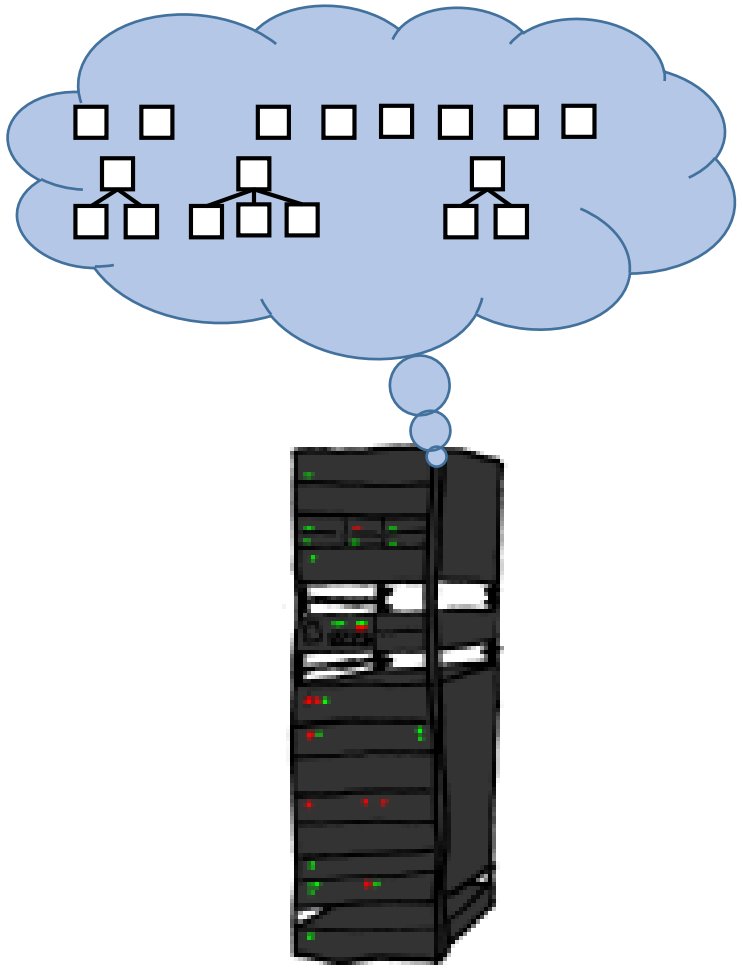
Adjusting our expectations

Here's a grammar of 20 functions and 10 constants, get me a program that I'm *certain* is in this space.



Adjusting our expectations

Here's a grammar of 20 functions and 10 constants, get me a program that I'm *certain* is in this space.



What does this all mean?

What does this all mean?

- Generally, “find me a program that—” cannot be solved

What does this all mean?

- Generally, “find me a program that—” cannot be solved
- Still, we’re not giving up

What does this all mean?

- Generally, “find me a program that—” cannot be solved
- Still, we’re not giving up
- Realistic expectations for realistic program synthesis

Realistic expectations for realistic synthesis

We still want

- Partial specifications
- To not have to know everything
- A result!

But we'll have to live without

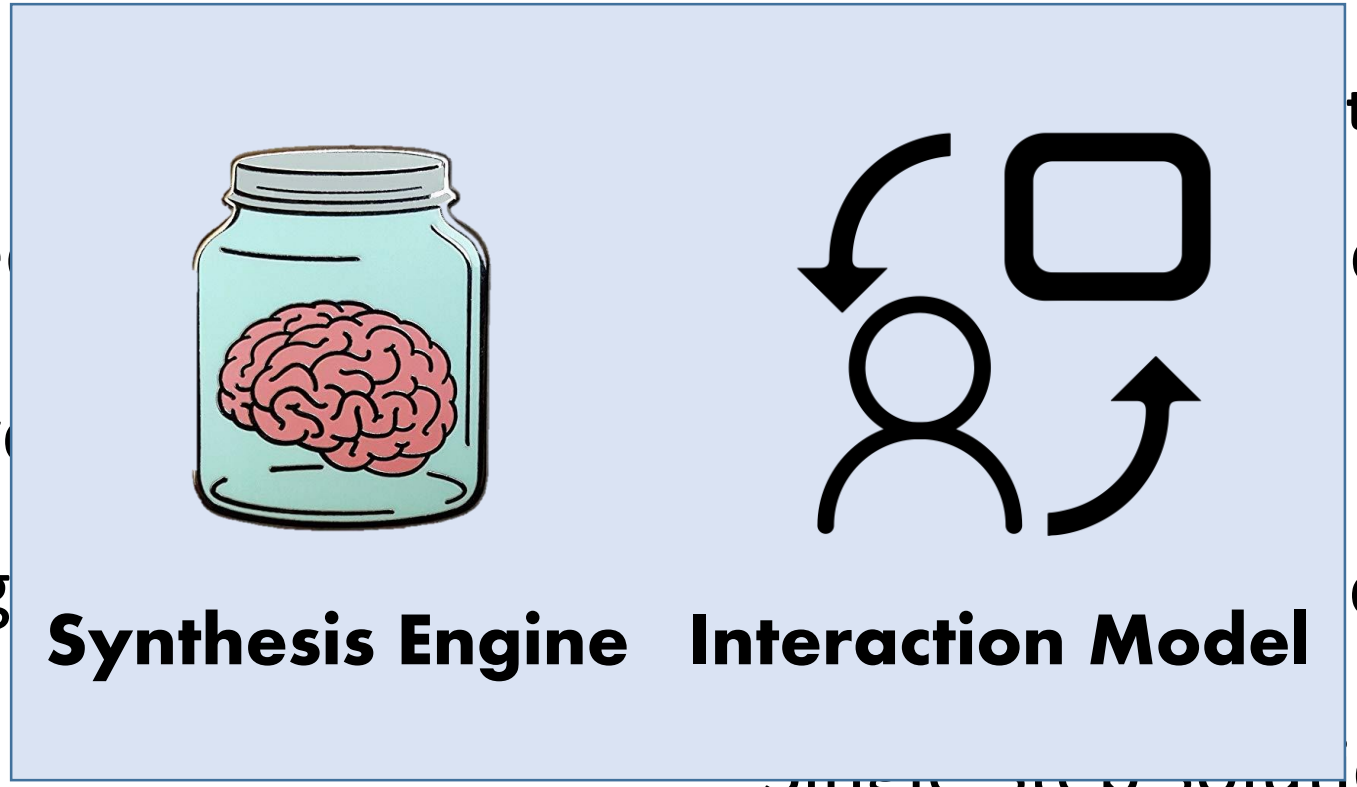
- Checking every possible program
- Fully automatic solution
- Single-step solution

Realistic expectations for realistic synthesis

We still want **thout**

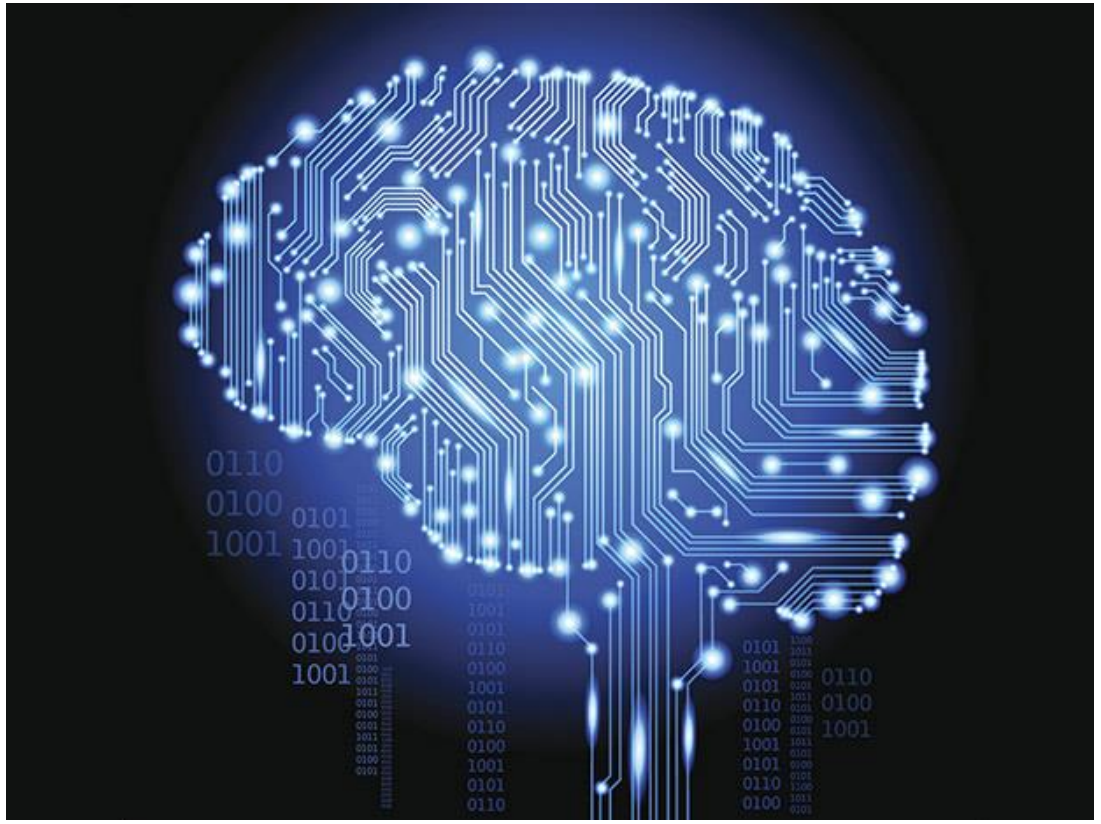
- Partial spe ossible
- To not hav

  everything olution
- A result! Single step solution



The synthesis engine

- Predicts code for intent
- Draws its understanding from language syntax
  - and/or crowd wisdom
  - and/or semantic specifications
- Reduce the number of programs seen



Reducing equivalent programs

- We've seen `x+y`, so we don't want `y+x`

Reducing equivalent programs

- We've seen `x+y`, so we don't want `y+x`
- But how do we know they're the same?

Reducing equivalent programs

- We've seen `x+y`, so we don't want `y+x`
- But how do we know they're the same?

1. Heuristics

Reducing equivalent programs

- We've seen `x+y`, so we don't want `y+x`
- But how do we know they're the same?

1. Heuristics
2. Solvers (e.g., Z3)

Reducing equivalent programs

- We've seen `x+y`, so we don't want `y+x`
- But how do we know they're the same?

1. Heuristics
2. Solvers (e.g., Z3)
3. *Observational* Equivalence

Reducing equivalent programs

- We've seen `x+y`, so we don't want `y+x`
- But how do we know they're the same?

1. Heuristics
2. Solvers (e.g., Z3)
3. *Observational* Equivalence

Equivalence:

$p_1 \equiv p_2$ i.f.f. for every possible input i ever, $[\![p_1]\!](i) = [\![p_2]\!](i)$

Reducing equivalent programs

- We've seen `x+y`, so we don't want `y+x`
- But how do we know they're the same?

1. Heuristics
2. Solvers (e.g., Z3)
3. *Observational* Equivalence

Equivalence:

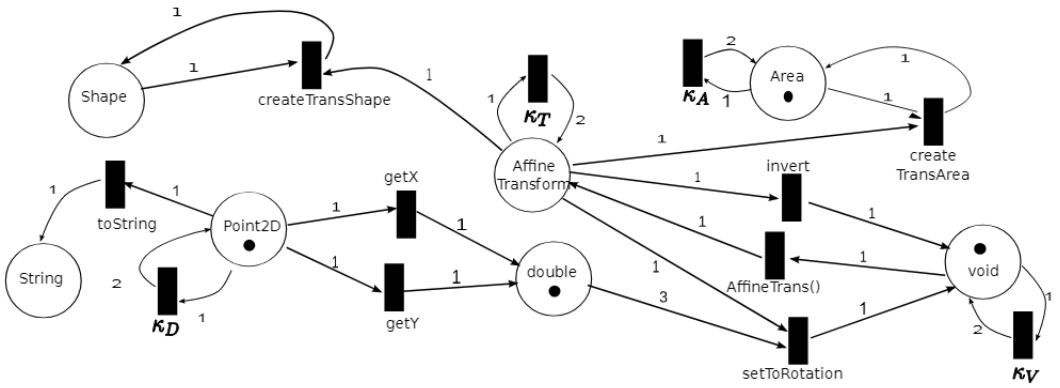
$p_1 \equiv p_2$ i.f.f. for every possible input i ever, $[\![p_1]\!](i) = [\![p_2]\!](i)$

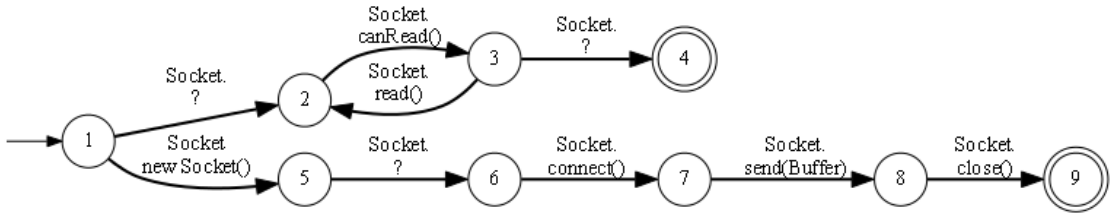
Observational equivalence:

$p_1 \equiv_{OE} p_2$ i.f.f. for every input i the user cares about, $[\![p_1]\!](i) = [\![p_2]\!](i)$

Trying a different strategy altogether

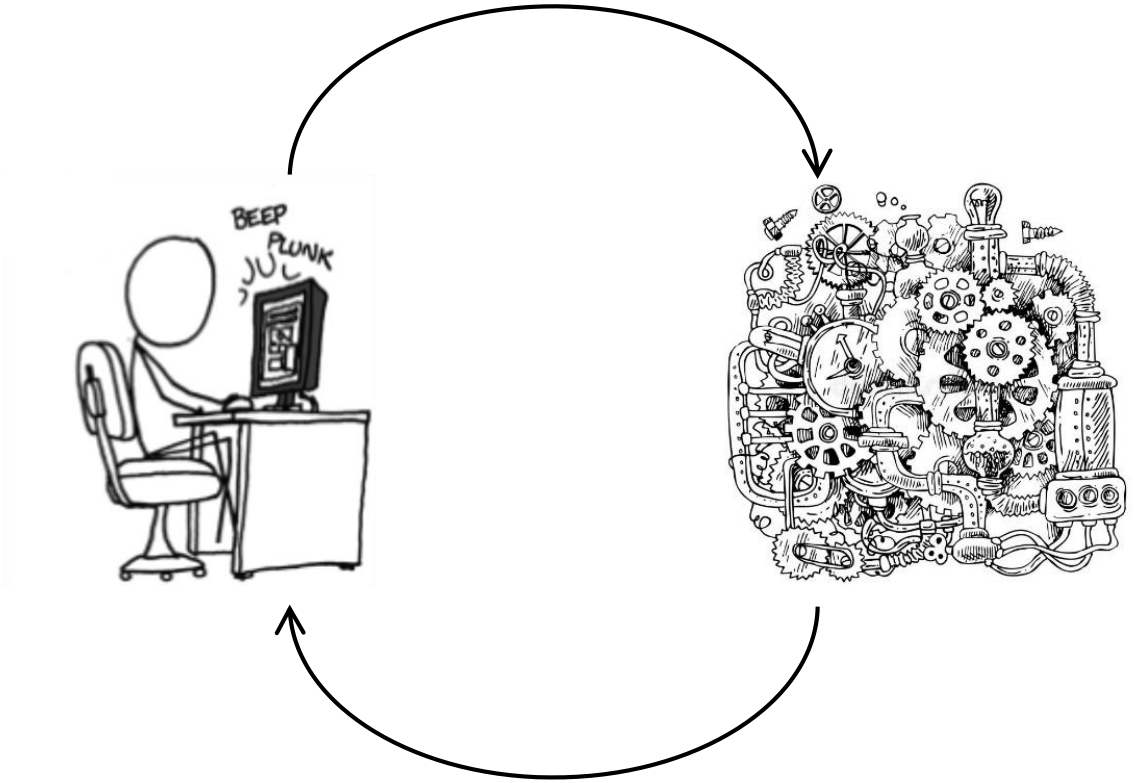
- Use knowledge bases instead of the language grammar
- They no longer contain every program (neither limited grammars)
- Searchable via graph algorithms or probability equations





The Interaction Model

- Aimed at programmers
  - Specify intent
  - Express yourself
  - Think like a programmer



Specifying (and re-specifying) intent

Task: find the median of a list

User: examples!

1. [1,2,3]→2
2. [7,8,7,3]→7

Synthesis engine:

```
input[input.length/2]
```

User: 

Specifying (and re-specifying) intent

Task: find the median of a list

User: examples!

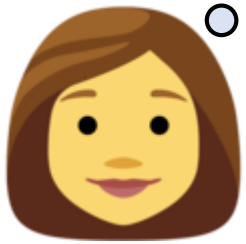
1. [1,2,3]→2
2. [7,8,7,3]→7

Synthesis engine:

```
input[input.length/2]
```

User:

It managed to find a single formula, let's make a counterexample



Specifying (and re-specifying) intent

Task: find the median of a list

User: examp

1. [1,2,3]→
2. [7,8,7,3]-

Synthesis er

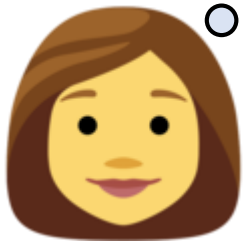
`input[inp`

User:

find a
let's
xample

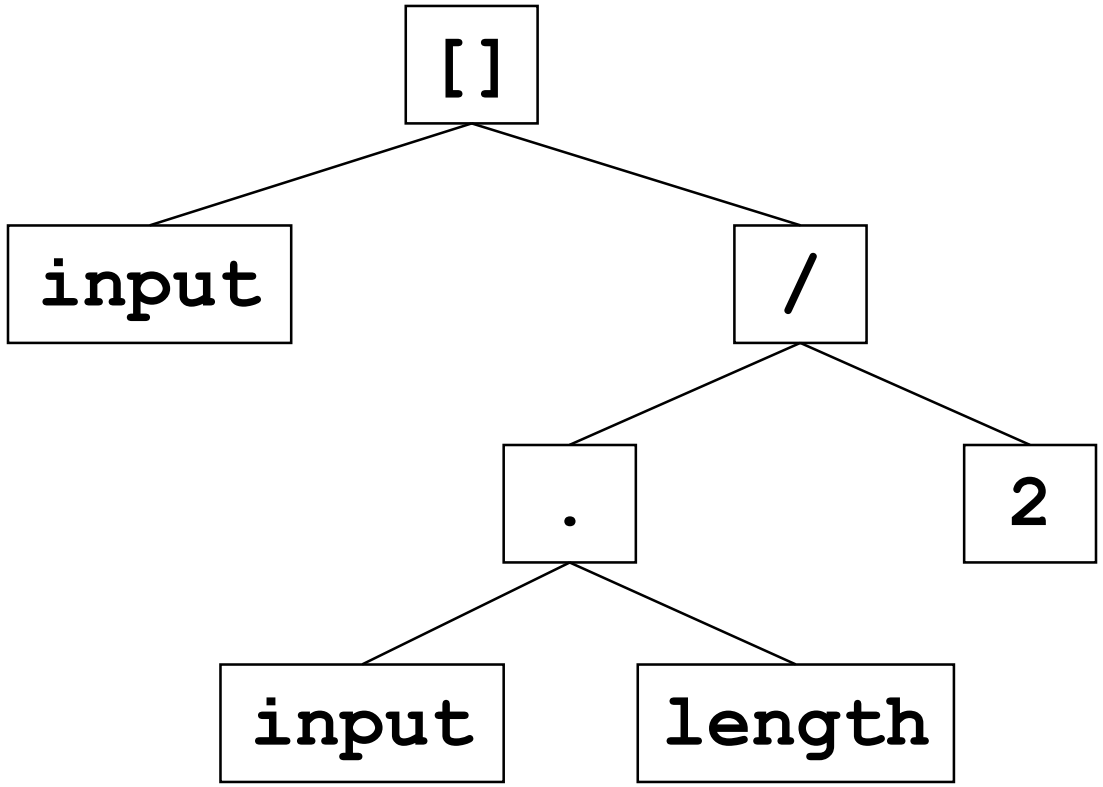
Principle #1:

Cost of communicating intent + consuming result << cost of manually performing the task



Programming Not Only by Example

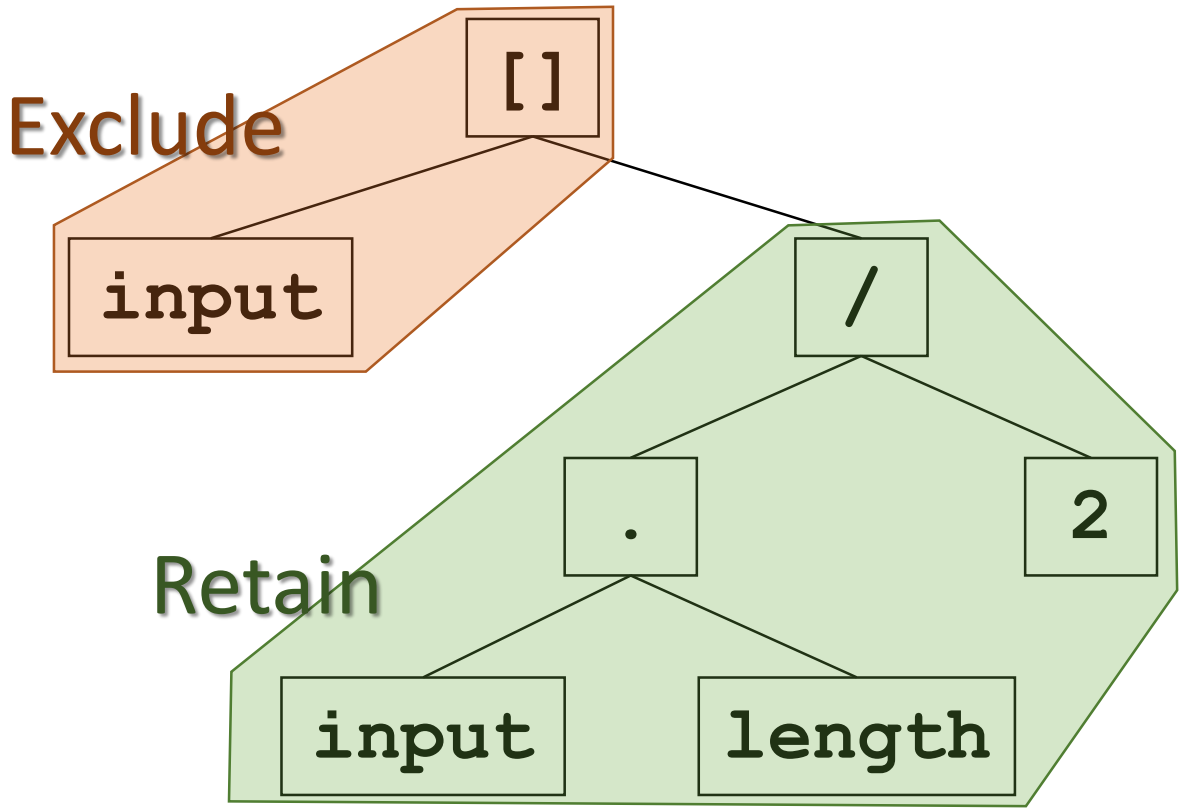
`input[input.length/2]`



- A programmer can talk at the level of the program
- Read debug info
- Reason about subtrees or sequences of methods
- Even rewrite the program
- But also give examples, if those happen to be easier

Programming Not Only by Example

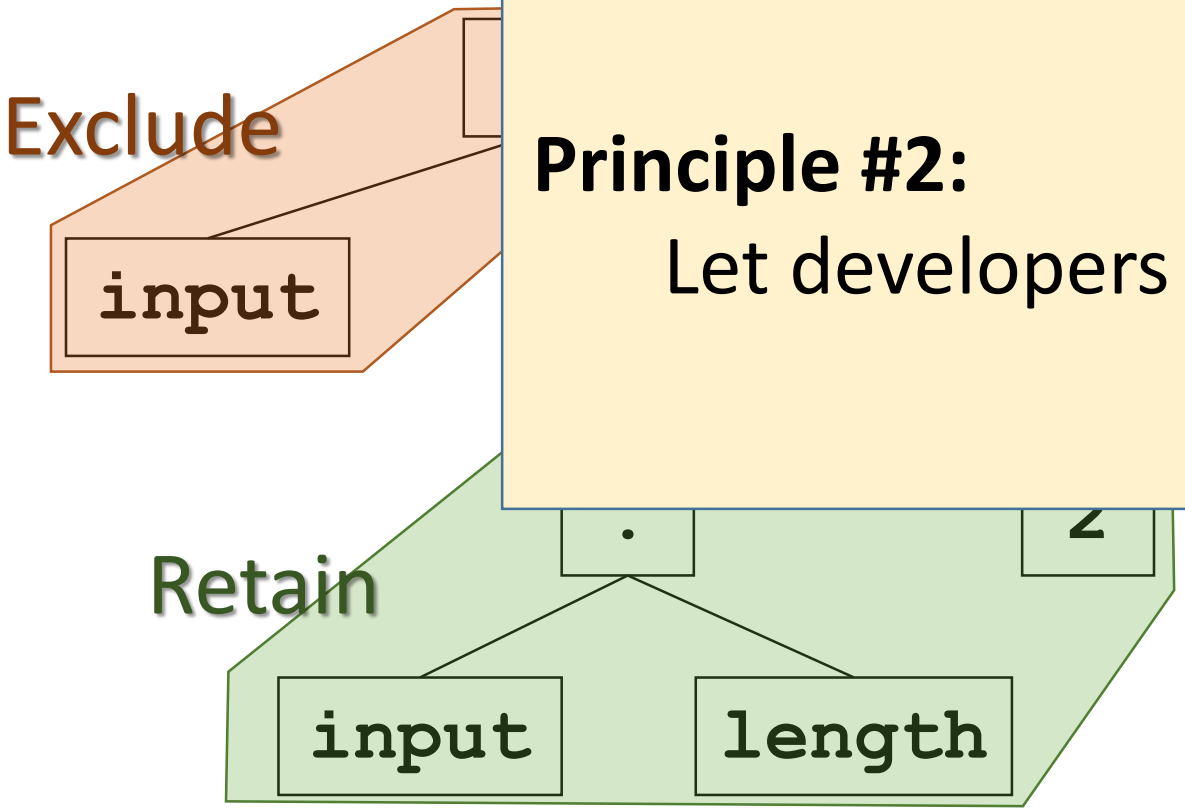
```
input[input.length/2]
```



- A programmer can talk at the level of the program
- Read debug info
- Reason about subtrees or sequences of methods
- Even rewrite the program
- But also give examples, if those happen to be easier

Programming Not Only by Example

```
input[input.length/2]
```



Principle #2:

Let developers be developers

an talk at
program

btrees or
ethods

program

- But also give examples, if those happen to be easier

When models “out-think” the programmer

```
def counts(l : List[String]) : Map[String,Int]=
    l.
```

When models “out-think” the programmer

```
def counts(l : List[String]) : Map[String,Int]=
    l.
      groupBy(
      map(
      fold(
```

When models “out-think” the programmer

```
def counts(l : List[String]) : Map[String,Int]=
    l.groupBy(
                identity)
                x => x.length)
                x => x[0])
```

When models “out-think” the programmer

```
def counts(l : List[String]) : Map[String,Int]=
    l.groupBy(identity).
```

```
map(x => x._1 -> x._2
x => x.length)
x => x[0])
```

When models “out-think” the programmer

```
def counts(l : List[String]) : Map[String,Int]=
    l.groupBy(identity).
    map(x => x._1 -> x._2.
```

```
filter(y => y.startsWith(
x._2.length
x._2[0])
```

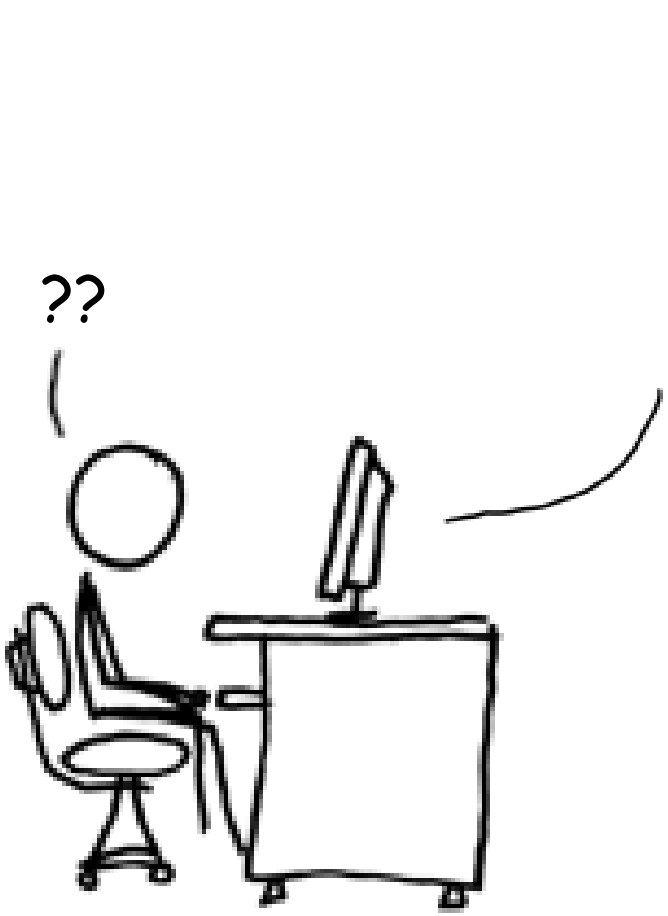
When models “out-think” the programmer

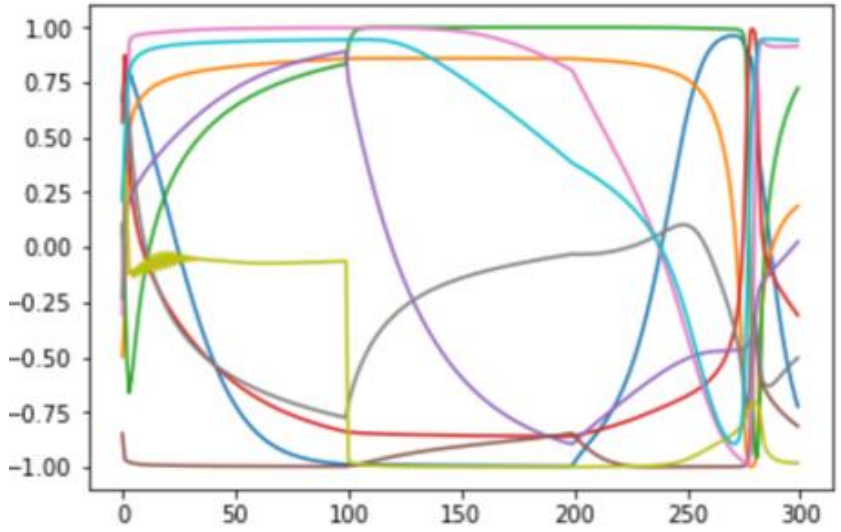
```
def counts(l : List[String]) : Map[String,Int]=
    l.groupBy(identity).
    map(x => x._1 -> x._2.
    filter(y => y.startsWith(
```

When models “out-think” the programmer

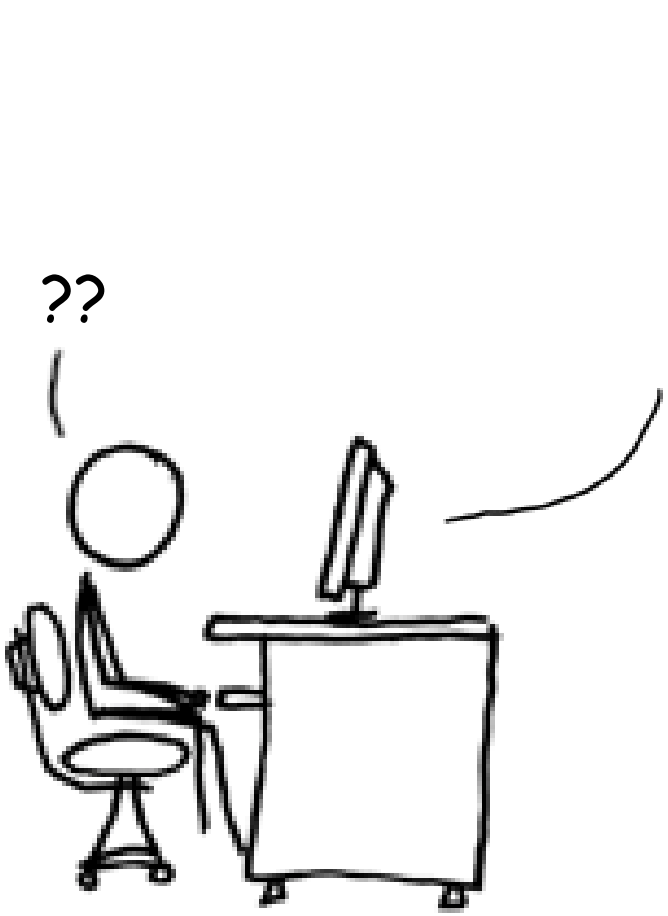
```
def counts(l : List[String]) : Map[String,Int]=
    l.groupBy(identity).
    map(x => x._1 -> x._2.
    filter(y => y.startsWith(
```

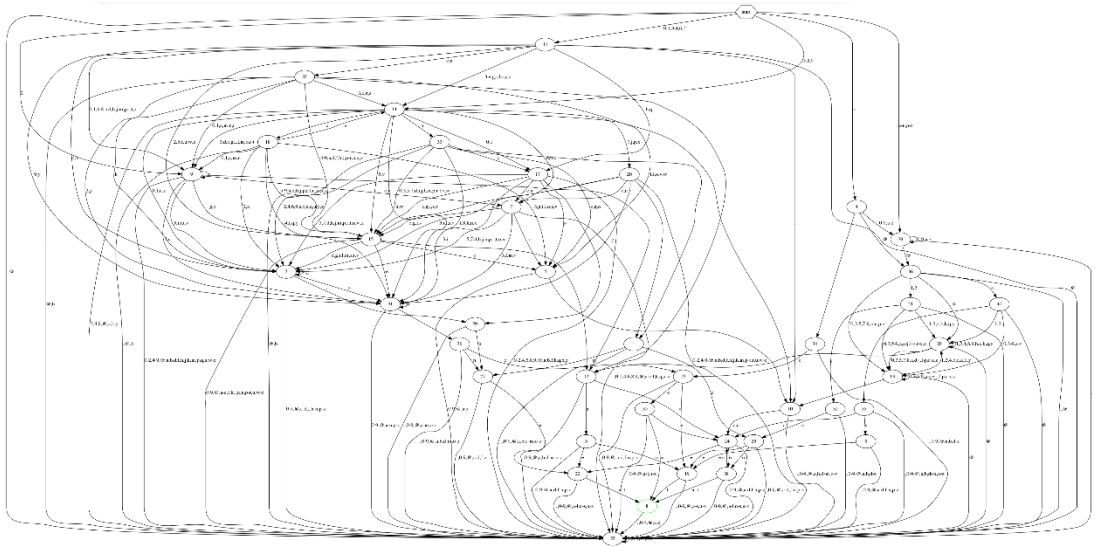
Keep understanding what’s going on



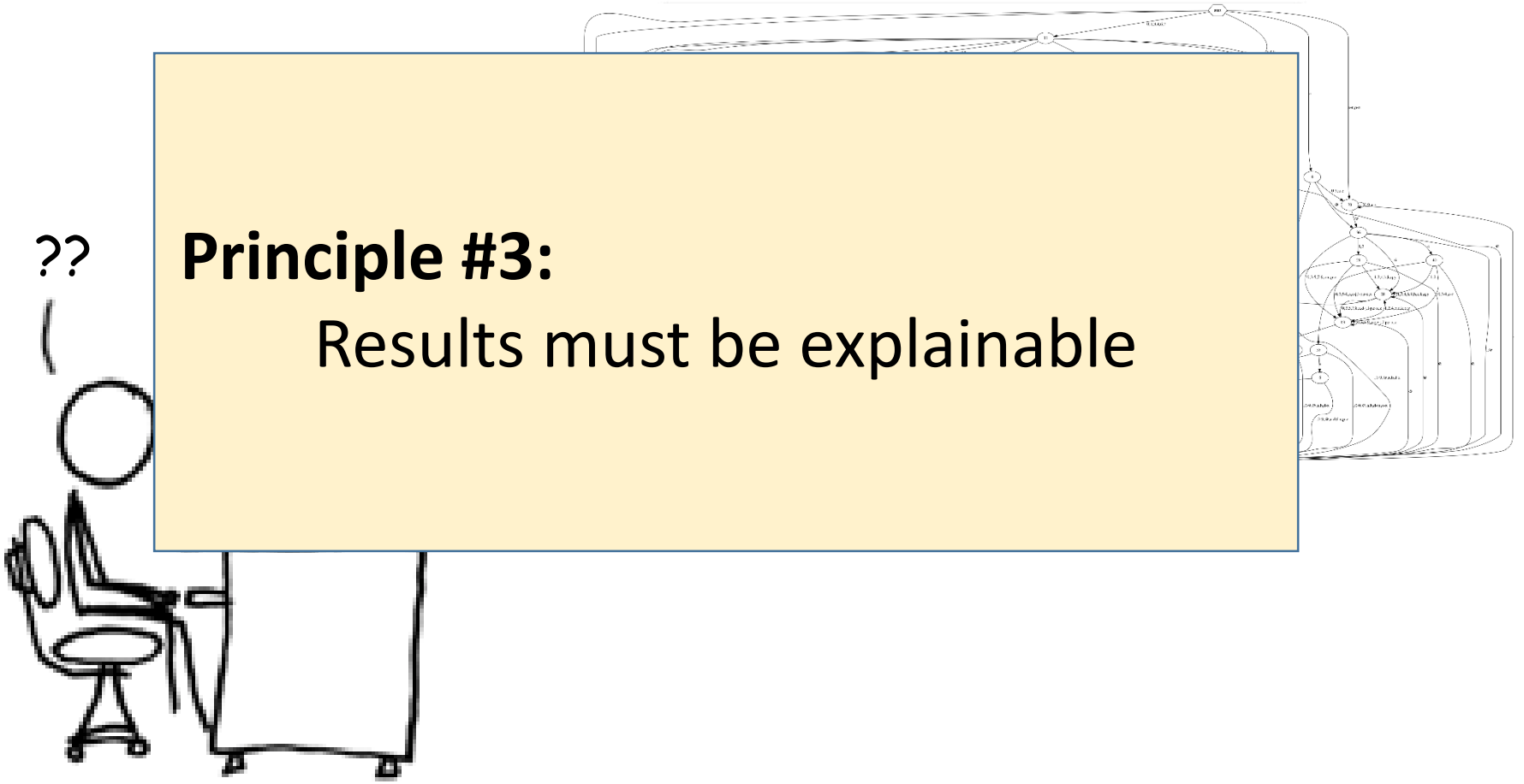


Keep understanding what’s going on

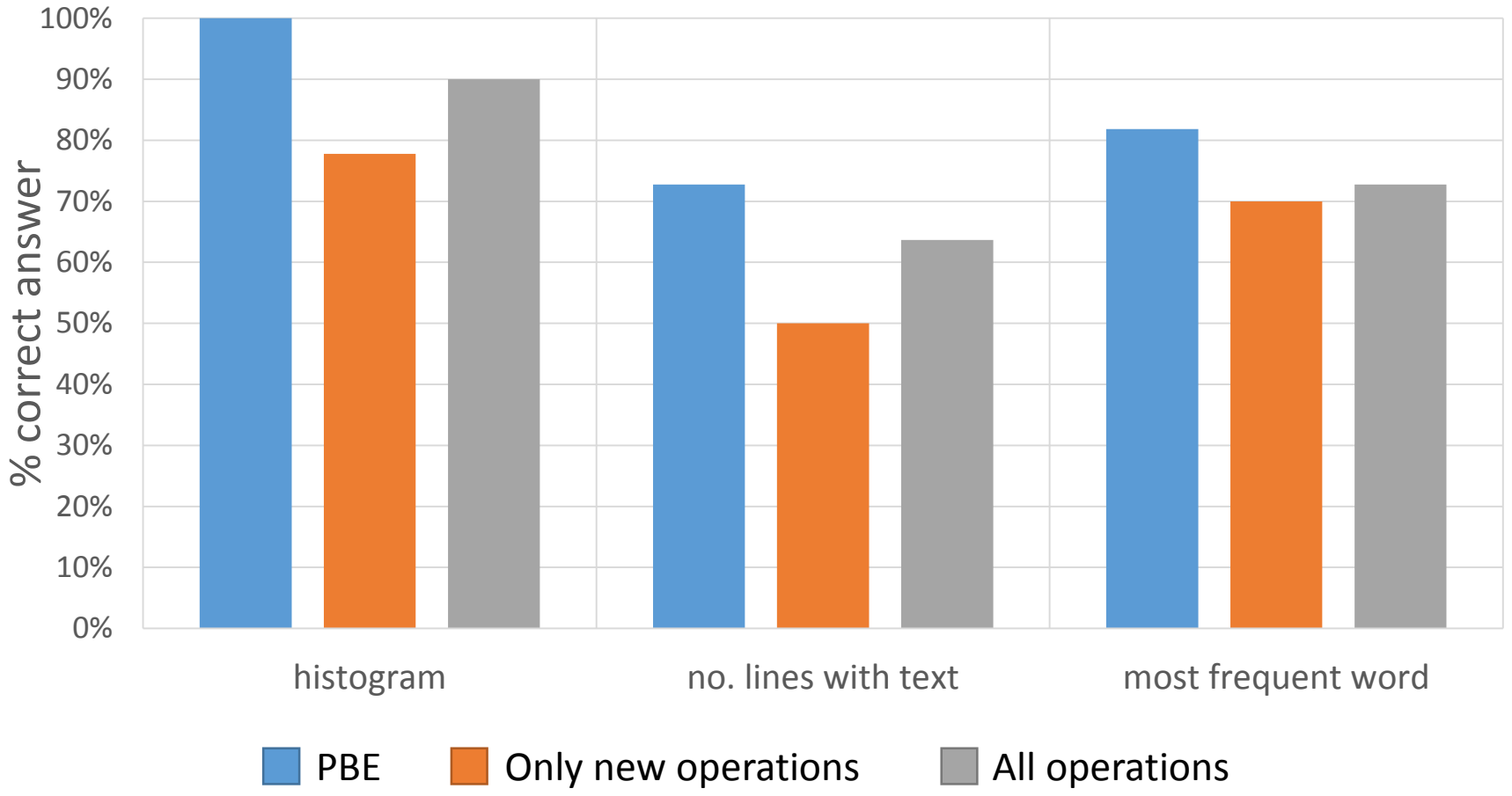




Keep understanding what's going on



Programmers aren’t as good as they think



Read-Eval-Synth Loops

What if we have automatic programming?

- ~~Hyper-~~intelligent program generation for ~~your every~~ need
- ~~Self-aware, self-augmenting AI~~
- ~~The singularity~~
- ???
- ~~THE ROBOT APOCALYPSE~~ PROFIT