Automatic Programming: How Far Can Machines Go?

Hila Peleg
Technion
Laziness
Laziness

Impatience
Laziness  Impatience  Hubris
Automatic Programming

Clean up my spreadsheet!

Right away, boss!
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 to the rescue
Program Synthesis

\[ \forall p \in \text{Proc}, d \in \text{Disk} : \]
\[ (d \in \text{disksWritten}[p]) \land (\text{phase}[p] \in \{1, 2\}) \land \text{disk}[d][p] = \text{dblock}[p] \land (\text{phase}[p] = 2, 3) \Rightarrow (\text{blocksRead}[p][d] \neq \emptyset) \Rightarrow (d \in \text{disksWritten}[p]) \land \neg \text{hashRead}(p, d, p) \]

\[ \forall p \in \text{Proc} : \]
\[ (\text{phase}[p] = 0) \Rightarrow \text{dblock}[p] = \text{InitDB} \land \text{disksWritten}[p] = \emptyset \land \forall d \in \text{Disk} : \forall br \in \text{blocksRead} \land \text{br.proc} = p \land \text{br.block} = \text{disk}[d][p] \land (\text{phase}[p] \neq 1) \Rightarrow (\text{dblock}[p], \text{mbal} \in \text{Ballot}(p) \land \text{dblock}[p].\text{bal} \in \text{Ballot}(p) \cup \{0\} \land \forall d \in \text{Disk} : \forall br \in \text{blocksRead} \land \text{br.block.mbal} < \text{dblock}[p].\text{mbal}.
\]
\[ (\text{phase}[p] \in \{2, 3\}) \Rightarrow (\text{dblock}[p].\text{bal} = \text{dblock}[p].\text{mbal}, \text{output}[p] = \text{if phase}[p] = 3 \text{ THEN dblock}[p].\text{inp} \land \text{chosen} \in \text{allInput} \cup \{\text{NotAnInput}\} \land \forall p \in \text{Proc} : \text{input}[p] \in \text{allInput} \land (\text{chosen} = \text{NotAnInput}) \Rightarrow (\text{output}[p]) \]
What we really want

This is what I 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

Generalizing partial intent is hard

Build a program
But it’s ok!

Understand users

Solving HALT is hard*

Generalizing partial intent is hard

Build a program
private void demo() throws Exception {
    URLConnection s = new URL(url).openConnection();
    InputStreamReader content = ....
}

function chunkData(e, t) {
    var n = [];
    var r = e.length;
    var i = 0;
    for (; i < r; i += t) {
        if (i + t < r) {
            n.push(e.substring(i, i + t));
        } else {
            n.push(e.substring(i, r));
        }
    }
    return n;
}

// You can also use Codota

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

Array.prototype.map((e, i) => input.toString().slice(i, i + 1)).map((e, i) => input.toString())

User intent is hard

Clean up my spreadsheet!

Right away, boss!
User intent is hard

Clean up my spreadsheet!

Right away, boss!

What’s “clean”? Is this right? What am I doing with my life?
## Intent via examples

<table>
<thead>
<tr>
<th></th>
<th>First Name</th>
<th>Last Name</th>
<th>Time</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Simon</td>
<td>Raik-Allen</td>
<td>16:40</td>
<td>Hi, Simon, just a reminder your talk is at 16:40</td>
</tr>
<tr>
<td>2</td>
<td>Aino</td>
<td>Corry</td>
<td>16:55</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Michelle</td>
<td>Casbon</td>
<td>15:55</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Mikael</td>
<td>Vidstedt</td>
<td>10:30</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Sam</td>
<td>Aaron</td>
<td>16:40</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Anita</td>
<td>Sengupta</td>
<td>17:40</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Jessica</td>
<td>Kerr</td>
<td>09:00</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Dave</td>
<td>Thomas</td>
<td>11:30</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Chris</td>
<td>Richardson</td>
<td>14:35</td>
<td></td>
</tr>
</tbody>
</table>
## Intent via examples

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>First Name</td>
<td>Time</td>
<td>Message</td>
</tr>
<tr>
<td>2</td>
<td>Simon</td>
<td>16:40</td>
<td>Hi, Simon, just a reminder your talk is at 16:40</td>
</tr>
<tr>
<td>3</td>
<td>Aino</td>
<td>16:55</td>
<td>Hi, Aino, just a reminder your talk is at 16:55</td>
</tr>
<tr>
<td>4</td>
<td>Michelle</td>
<td>15:55</td>
<td>Hi, Michelle, just a reminder your talk is at 15:55</td>
</tr>
<tr>
<td>5</td>
<td>Mikael</td>
<td>10:30</td>
<td>Hi, Mikael, just a reminder your talk is at 10:30</td>
</tr>
<tr>
<td>6</td>
<td>Sam</td>
<td>16:40</td>
<td>Hi, Sam, just a reminder your talk is at 16:40</td>
</tr>
<tr>
<td>7</td>
<td>Anita</td>
<td>17:40</td>
<td>Hi, Anita, just a reminder your talk is at 17:40</td>
</tr>
<tr>
<td>8</td>
<td>Jessica</td>
<td>09:00</td>
<td>Hi, Jessica, just a reminder your talk is at 09:00</td>
</tr>
<tr>
<td>9</td>
<td>Dave</td>
<td>11:30</td>
<td>Hi, Dave, just a reminder your talk is at 11:30</td>
</tr>
<tr>
<td>10</td>
<td>Chris</td>
<td>14:35</td>
<td>Hi, Chris, just a reminder your talk is at 14:35</td>
</tr>
</tbody>
</table>
## Intent via examples

<table>
<thead>
<tr>
<th></th>
<th>First Name</th>
<th>Last Name</th>
<th>Time</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Simon</td>
<td>Raik-Allen</td>
<td>16:40</td>
<td>Hi, Simon, just a reminder your talk is at 16:40</td>
</tr>
<tr>
<td>3</td>
<td>Aino</td>
<td>Corry</td>
<td>16:55</td>
<td>Hi, Aino, just a reminder your talk is at 16:55</td>
</tr>
<tr>
<td>2</td>
<td>Michelle</td>
<td>Casbon</td>
<td>15:55</td>
<td>Hi, Michelle, just a reminder your talk is at 15:55</td>
</tr>
<tr>
<td>5</td>
<td>Mikael</td>
<td>Vidstedt</td>
<td>10:30</td>
<td>Hi, Mikael, just a reminder your talk is at 10:30</td>
</tr>
<tr>
<td>6</td>
<td>Sam</td>
<td>Aaron</td>
<td>16:40</td>
<td>Hi, Sam, just a reminder your talk is at 16:40</td>
</tr>
<tr>
<td>7</td>
<td>Anita</td>
<td>Sengupta</td>
<td>17:40</td>
<td>Hi, Anita, just a reminder your talk is at 17:40</td>
</tr>
<tr>
<td>8</td>
<td>Jessica</td>
<td>Kerr</td>
<td>09:00</td>
<td>Hi, Jessica, just a reminder your talk is at 09:00</td>
</tr>
<tr>
<td>9</td>
<td>Dave</td>
<td>Thomas</td>
<td>11:30</td>
<td>Hi, Dave, just a reminder your talk is at 11:30</td>
</tr>
<tr>
<td>10</td>
<td>Chris</td>
<td>Richardson</td>
<td>14:35</td>
<td>Hi, Chris, just a reminder your talk is at 14:35</td>
</tr>
</tbody>
</table>
## Intent via examples

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>First Name</td>
<td>Last Name</td>
<td>Time</td>
</tr>
<tr>
<td>2</td>
<td>Simon</td>
<td>Raik-Allen</td>
<td>16:40</td>
</tr>
<tr>
<td>3</td>
<td>Aino</td>
<td>Corry</td>
<td>16:55</td>
</tr>
<tr>
<td>4</td>
<td>Michelle</td>
<td>Casbon</td>
<td>15:55</td>
</tr>
<tr>
<td>5</td>
<td>Mikael</td>
<td>Vidstedt</td>
<td>10:30</td>
</tr>
<tr>
<td>6</td>
<td>Sam</td>
<td>Aaron</td>
<td>16:40</td>
</tr>
<tr>
<td>7</td>
<td>Anita</td>
<td>Sengupta</td>
<td>17:40</td>
</tr>
<tr>
<td>8</td>
<td>Jessica</td>
<td>Kerr</td>
<td>09:00</td>
</tr>
<tr>
<td>9</td>
<td>Dave</td>
<td>Thomas</td>
<td>11:30</td>
</tr>
<tr>
<td>10</td>
<td>Chris</td>
<td>Richardson</td>
<td>14:35</td>
</tr>
</tbody>
</table>
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

```scala
val routes : Route = ???
val bindingFuture = Http().bindAndHandle(routes, "localhost", 8080)
```
Commit Strip said it best...

Commit Strip said it best...

Commit Strip said it best...

Some day you coders are going to just write the program...

Oh wow, you're able to write a precise spec to need program...

And do you know the industry term for a project specification that is comprehensive and precise enough to generate a program?

Exactly!

Uh... no....

Commit Strip said it best...

Building a program is also hard*

Get me a program that takes a program and an input and tells me if that program stops on that input.
Building a program is also hard*

Get me a program that takes a program and an input and tells me if that program stops on that input. (i.e., the halting problem)
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

• 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
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:**

\[ p_1 \equiv p_2 \text{ i.f.f. for every possible input } i \text{ ever, } \llbracket p_1 \rrbracket(i) = \llbracket p_2 \rrbracket(i) \]
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, $\llbracket p_1 \rrbracket(i) = \llbracket p_2 \rrbracket(i)$

**Observational equivalence:**

$p_1 \equiv_{OE} p_2$ i.f.f. for every input $i$ the user cares about, $\llbracket p_1 \rrbracket(i) = \llbracket p_2 \rrbracket(i)$
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: examples!

1. \([1,2,3]\)→2
2. \([7,8,7,3]\)→7

Synthesis engine:

\[
\text{input}[\text{input.length}/2]
\]

User: It managed to find a single formula, let’s make a counterexample

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

- 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

input[input.length/2]
Programming Not Only by Example

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

Principle #2: Let developers be developers

input[input.length/2]
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

```scala
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

**Principle #3:**
Results must be explainable
Programmers aren’t as good as they think

![Bar Chart](chart.png)

- Percentage correct answer for:
  - Histogram
  - Number of lines with text
  - Most frequent word

- **PBE**
- **Only new operations**
- **All operations**
Read-Eval-Synth Loops

KEEP CALM
IT IS DEMO TIME
What if we have automatic programming?

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