LEARNING CONJECTURING FROM SCRATCH[1]

Thibault Gauthier, Josef Urban

Czech Technical University in Prague, Czech Republic

[1] This work was supported by the Czech Science Foundation grant no. 25-17929X, Czech Ministry of Education, Youth and Sports within the ERC CZ program under the project POSTMAN no. LL1902, and by Amazon Research Awards.

What is conjecturing?

*“C’est par la logique qu’on démontre, c’est par **l’intuition** qu’on invente.”*
(It is by logic that we prove, but by intuition that we discover.)

– Henri Poincaré, Mathematical Definitions and Education.

A simple form of conjecturing: what comes next?

$$2, 1, 3, 7, 5, 31, 3, 127, 17, 73, 11, 2047, 13, 8191, 43, 151, 257, ?$$

A **synthesize and test** approach

OEIS sequence

$$0, 1, 3, 6, 10, 15, \ldots, 1431$$

Synthesized program:

$$f(x) = (x \times x + x) \div 2$$

Test/Filter:

$$f(0) = 0,\ f(1) = 1,\ f(2) = 3,\ f(3) = 6,\ \ldots,\ f(53) = 1431$$

Simple explanations are often better (Occam's razor).

OEIS sequence

$$0, 1, 3, 6, 10, 15, \ldots, 1431$$

Large program

```
if x = 0 then 0 else
if x = 1 then 1 else
if x = 2 then 3 else
if x = 3 then 6 else ...
if x >= 53 then 1431
```

Small program

$$f(x) = \sum_{i=1}^{x} i$$

Fast program

$$f(x) = (x \times x + x) \div 2$$

Synthesize: extended language of recursive functions

- Constants: $0, 1, 2$
- Variables: x, y
- Arithmetical operators: $+, -, \times, \mathit{div}, \mathit{mod}$
- Condition: if $\ldots \leqslant 0$ then $\ldots$ else $\ldots$
- $\mathit{loop}(f, a, b) := u_a$ where $u_0 = b$,

$$u_n = f(u_{n-1}, n)$$

- Two other loop constructs: *loop2*, a while loop

Example:

$2^{\mathbf{x}} = \prod_{y=1}^{x} 2 = \mathit{loop}(2 \times x, \mathbf{x}, 1)$

$\mathbf{x}! = \prod_{y=1}^{x} y = \mathit{loop}(y \times x, \mathbf{x}, 1)$

How hard is it to synthesize programs from scratch?

OEIS sequence

$$0, 1, 3, 6, 10, 15, \ldots, 1431$$

Synthesized program

$$(0.2$$

How hard is it to synthesize programs from scratch?

OEIS sequence

$$0, 1, 3, 6, 10, 15, \ldots, 1431$$

Synthesized program

$$(_{0.2} \quad x_{0.3}$$

How hard is it to synthesize programs from scratch?

OEIS sequence

$$0, 1, 3, 6, 10, 15, \ldots, 1431$$

Synthesized program

$$(_{0.2} \quad x_{0.3} \quad \times_{0.12}$$

How hard is it to synthesize programs from scratch?

OEIS sequence

$$0, 1, 3, 6, 10, 15, \ldots, 1431$$

Synthesized program

$$(_{0.2} \quad x_{0.3} \quad \times_{0.12} \quad x_{0.99}$$

How hard is it to synthesize programs from scratch?

OEIS sequence

$$0, 1, 3, 6, 10, 15, \ldots, 1431$$

Synthesized program

$$(_{0.2} \quad x_{0.3} \quad \times_{0.12} \quad x_{0.99} \quad +_{0.1}$$

How hard is it to synthesize programs from scratch?

OEIS sequence

$$0, 1, 3, 6, 10, 15, \ldots, 1431$$

Synthesized program

$$(_{0.2} \quad x_{0.3} \quad \times_{0.12} \quad x_{0.99} \quad +_{0.1} \quad x_{0.25}$$

How hard is it to synthesize programs from scratch?

OEIS sequence

$$0, 1, 3, 6, 10, 15, \ldots, 1431$$

Synthesized program

$$(_{0.2} \quad x_{0.3} \quad \times_{0.12} \quad x_{0.99} \quad +_{0.1} \quad x_{0.25} \quad)_{0.48}$$

How hard is it to synthesize programs from scratch?

OEIS sequence

$$0, 1, 3, 6, 10, 15, \ldots, 1431$$

Synthesized program

$$(_{0.2} \quad x_{0.3} \quad \times_{0.12} \quad x_{0.99} \quad +_{0.1} \quad x_{0.25} \quad)_{0.48} \quad \div_{0.02}$$

How hard is it to synthesize programs from scratch?

OEIS sequence

$$0, 1, 3, 6, 10, 15, \ldots, 1431$$

Synthesized program

$$(_{0.2} \quad x_{0.3} \quad \times_{0.12} \quad x_{0.99} \quad +_{0.1} \quad x_{0.25} \quad)_{0.48} \quad \div_{0.02} \quad 2_{0.09}$$

How hard is it to synthesize programs from scratch?

OEIS sequence

$$0, 1, 3, 6, 10, 15, \ldots, 1431$$

Synthesized program

$$(_{0.2} \quad x_{0.3} \quad \times_{0.12} \quad x_{0.99} \quad +_{0.1} \quad x_{0.25} \quad)_{0.48} \quad \div_{0.02} \quad 2_{0.09}$$

The probability of generating this program is:
$0.2 \times 0.3 \times 0.12 \times 0.99 \times 0.1 \times 0.25 \times 0.48 \times 0.02 \times 0.09 = 1.54... \times 10^{-7}$

A self-learning language model

Repeat the following steps:

- **Synthesize** candidate programs for each OEIS sequence ($> 350,000$).
- **Test** if the candidate programs generate **any** OEIS sequence.
- **Train** on previously discovered pairs (sequence,program).

Result

Discovering new programs at each iteration for more than 3 years.

Programs for more than one-third of the OEIS (134,000 sequences)

Example

OEIS description: Cyclotomic polynomials at x=2

$$2, 1, 3, 7, 5, 31, 3, 127, 17, 73, 11, 2047, 13, 8191, 43, 151, 257, 131071, \ldots?$$

We discovered a program for that sequence, but is it predicting correctly the next terms?

```
def f(X):
  j = len(X)
  x = 2**(j+1)-1
  for d in X:
    x = x // d if x % d == 0 else x
return x

def g(n):
  x = [1]
  for _ in range(n):
    x = [f(x)] + x
  return x[::-1]
```

Yes, by the Bang–Zsigmondy theorem according to Tom Hales.

Can we automatically prove those programs correct?

We would like to prove:

$$\forall x \in \mathbb{N}.\ f_{original}(x) = f_{synthesized}(x)$$

Instead, we prove:

$$\forall x \in \mathbb{N}.\ f_{smallest}(x) = f_{fastest}(x)$$

A benchmark of 29687 SMT problems.

Problems in the benchmark

- A217, triangular numbers:

$$\sum_{i=0}^{n} i = \frac{n \times n + n}{2}$$

- A537, sum of first n cubes:

$$\sum_{i=0}^{n} i^3 = (\frac{n \times n + n}{2})^2$$

- A79, powers of 2:

$$2^x = 2^{(x \ mod \ 2)} \times (2^{(x \ div \ 2)})^2$$

- A165, double factorial of even numbers:

$$\prod_{i=1}^{n} 2i = 2^n \times n!$$

A translation to SMT

Loops are translated to SMT using uninterpreted functions.

Program:

$$f(n) = \sum_{i=0}^{n} i$$

Translation:

$$f(n) = \text{if } n \leqslant 0 \text{ then } 0 \text{ else } n + f(n-1)$$

What would be the best theorem prover for our task?

These problems usually require inductive reasoning.

Vampire and CVC5 have an inductive reasoning mode.

Z3 performs much better than Vampire and CVC5 on arithmetical problems.

How do we help Z3 with inductive reasoning?

Our approach: synthesizing predicates

Help Z3 by adding instances of a second-order induction axiom:

$$\forall P.\ (P(0) \wedge (\forall x.\ P(x) \Rightarrow P(x+1)) \Rightarrow (\forall x.\ 0 \leqslant x \Rightarrow P(x))$$

For example, if we automatically conjecture the lemma/predicate $f(x) = g(x)$ and we add the following axiom to the problem:

$$f(0) = g(0) \wedge (\forall x.\ f(x) = g(x) \Rightarrow f(x+1) = g(x+1))$$

$$\Rightarrow \forall x.\ 0 \leqslant x \Rightarrow f(x) = g(x)$$

A language for conjectures (induction predicates)

- $0, 1, 2, x \in \mathbb{T}$.
- If $t_1, t_2 \in \mathbb{T}$ then $t_1 + t_2,\ t_1 - t_2,\ t_1 \times t_2,\ t_1\ \mathit{div}\ t_2,\ t_1\ \mathit{mod}\ t_2 \in \mathbb{T}$.
- if $t_1, t_2, t_3 \in \mathbb{T}$ then if $t_1 \leqslant 0$ then t_2 else $t_3 \in \mathbb{T}$.
- if f is a n-ary *function* appearing in the problem and $t_1, \ldots, t_n \in \mathbb{T}$ then $f(t_1, \ldots, t_n) \in \mathbb{T}$.
- if $t_1, t_2 \in \mathbb{T}$ then $t_1 = t_2,\ \neg(t_1 = t_2),\ t_1 \leqslant t_2, \neg(t_1 \leqslant t_2) \in \mathbb{L}$.
- if $l_1, l_2 \in \mathbb{L}$ and $f_1, f_2 \in \mathbb{F}$ then $l_1, l_2, f_1 \wedge f_2,\ f_1 \Rightarrow f_2 \in \mathbb{F}$.

We synthesize predicates probabilistically:

$$f_{0.2}\ (_{0.3}\ x_{0.02}\ )_{0.1}\ =_{0.2}\ g_{0.8}\ (_{0.54}\ x_{0.04}\ )_{0.11}$$

A self-learning language model

Repeat the following steps:

- **Synthesize** candidate predicates for each problem.
- **Attempt to prove** the modified problem using Z3 for 200ms.
- **Train** on previously discovered pairs (problem,predicates).

Result

The self-learning system initially discovers solutions (helping predicates) for more and more problems at each iteration but plateau after a few months.

The system outperforms the performance of Z3, Vampire and CVC5 run with a timeout of 10 seconds (even with manually defined inductive predicates).

Example (only solved automatically by self-learning)

A26532: $1, 3, 6, 18, 36, 108, 216, \ldots$

Problem in math form:

$$\prod_{k=1}^{x}(2 + k \bmod 2) = 6^{\lfloor \frac{x}{2} \rfloor} \times \begin{cases} 1 & \text{if } x \bmod 2 = 0 \\ 3 & \text{otherwise} \end{cases}$$

Problem in program form:

$$\underbrace{\underbrace{loop(2 + (Y \ mod \ 2) \times X, X}_{v}, 1)}_{u} =$$

$$\underbrace{\underbrace{loop2(X \times Y, Y, X \ div \ 2}_{w}, loop(3, X \ mod \ 2, 1), 6)}_{s}$$

Helpful predicate:

$$u(1 + x, 1) = w(1 + x) \wedge v(x) = w(x) \wedge w(2) = s(x) \wedge s(1 + x) = s(x)$$

Conclusion

Our self-learning system **synthesizes programs** for *hard* OEIS sequences

Our self-learning system **synthesizes predicates** and help Z3 prove *simple* equivalences between such programs.

Future work:

- better automated theorem prover
- better language model