Solving One-Third of the OEIS from Scratch

Thibault Gauthier, Josef Urban

January 23, 2024

Guessing/intuition/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.

"Hypothesen sind Netze; nur der fängt, wer auswirft."
(Hypotheses are nets: only he who casts will catch.)

– Novalis, quoted by Popper – The Logic of Scientific Discovery

Certainly, let us learn proving, but also let us learn guessing.

– G. Polya - Mathematics and Plausible Reasoning

*Galileo once said, "Mathematics is the language of Science." Hence, facing the same laws of the physical world, **alien mathematics** must have a good deal of similarity to ours.*

– R. Hamming - Mathematics on a Distant Planet

How to conjecture?

$$f(0) = 1, f(1) = 2, f(3) = 4, f(4) = 8, f(5) = 16, f(6) = ?$$

The answer is 32.

How to conjecture?

$$f(0) = 1, f(1) = 2, f(3) = 4, f(4) = 8, f(5) = 16, f(6) = ?$$

The answer is 32.

Objection! $f(6)$ could be anything.
There is a degree 6 polynomial such that:

$$f(0) = 1, f(1) = 2, \ldots, f(6) = 8892$$

How to conjecture?

$$f(0) = 1, f(1) = 2, f(3) = 4, f(4) = 8, f(5) = 16, f(6) = ?$$

The answer is 32.

Objection! $f(6)$ could be anything.
There is a degree 6 polynomial such that:

$$f(0) = 1, f(1) = 2, \ldots, f(6) = 8892$$

What is the simplest explanation/program for that sequence?

OEIS: $\geqslant$ 350000 sequences

The OEIS is supported by the many generous donors to the OEIS Foundation.

THE ON-LINE ENCYCLOPEDIA OF INTEGER SEQUENCES ®

founded in 1964 by N. J. A. Sloane

2 3 5 7 11 Search Hints

(Greetings from The On-Line Encyclopedia of Integer Sequences!)

Search: **seq:2,3,5,7,11**

Displaying 1-10 of 1163 results found. page 1 2 3 4 5 6 7 8 9 10 ... 117

Sort: relevance | references | number | modified | created Format: long | short | data

A000040 The prime numbers. +30 10150

(Formerly M0652 N0241)

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271 (list; graph; refs; listen; history; text; internal format)

OFFSET 1,1

COMMENTS See A065091 for comments, formulas etc. concerning only odd primes. For all information concerning prime powers, see A000961. For contributions concerning "almost primes" see A002808.

A number p is prime if (and only if) it is greater than 1 and has no positive divisors except 1 and p.

A natural number is prime if and only if it has exactly two (positive) divisors.

A prime has exactly one proper positive divisor, 1.

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: $+, -, \times, div, mod$
- Condition: if $\ldots \leqslant 0$ then $\ldots$ else $\ldots$
- $loop(f, a, b) := u_a$ where $u_0 = b$,

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

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

Example:

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

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

Synthesizing programs from scratch

OEIS sequence

$$0, 1, 3, 6, 10, 15, \ldots, 120$$

Synthesized program

$$(_{0.2}$$

Synthesizing programs from scratch

OEIS sequence

$$0, 1, 3, 6, 10, 15, \ldots, 120$$

Synthesized program

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

Synthesizing programs from scratch

OEIS sequence

$$0, 1, 3, 6, 10, 15, \ldots, 120$$

Synthesized program

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

Synthesizing programs from scratch

OEIS sequence

$$0, 1, 3, 6, 10, 15, \ldots, 120$$

Synthesized program

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

Synthesizing programs from scratch

OEIS sequence

$$0, 1, 3, 6, 10, 15, \ldots, 120$$

Synthesized program

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

Synthesizing programs from scratch

OEIS sequence

$$0, 1, 3, 6, 10, 15, \ldots, 120$$

Synthesized program

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

Synthesizing programs from scratch

OEIS sequence

$$0, 1, 3, 6, 10, 15, \ldots, 120$$

Synthesized program

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

Synthesizing programs from scratch

OEIS sequence

$$0, 1, 3, 6, 10, 15, \ldots, 120$$

Synthesized program

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

Synthesizing programs from scratch

OEIS sequence

$$0, 1, 3, 6, 10, 15, \ldots, 120$$

Synthesized program

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

Synthesizing programs from scratch

OEIS sequence

$$0, 1, 3, 6, 10, 15, \ldots, 120$$

Synthesized program

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

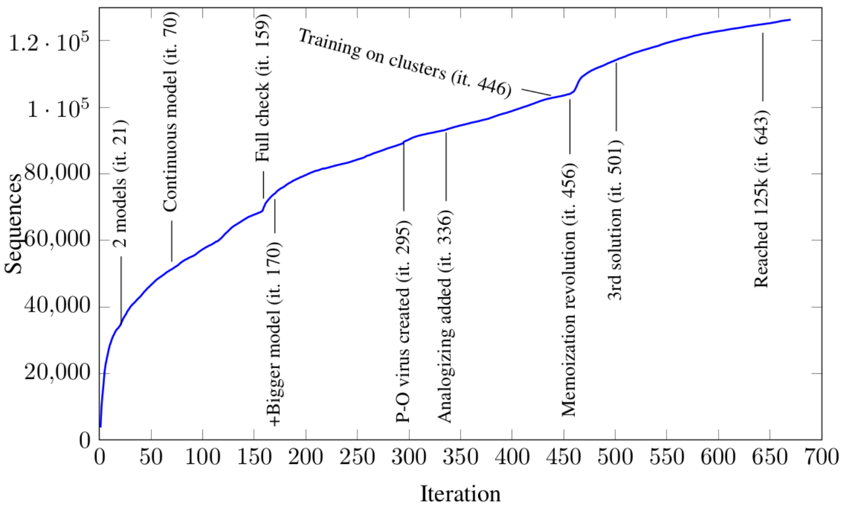
The probability of generating this program is:
$0.2 \times 0.3 \times 0.12 \times 0.99 \times 0.1 \times 0.25 \times 0.48 \times 0.02 \times 0.09 = 1.54... \times 10^{-7}$

A self-learning language model

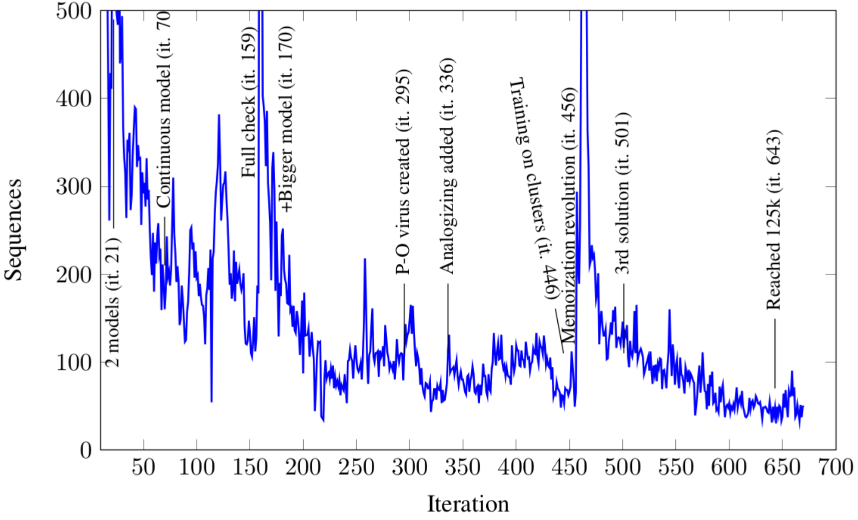
Repeat the following steps:

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

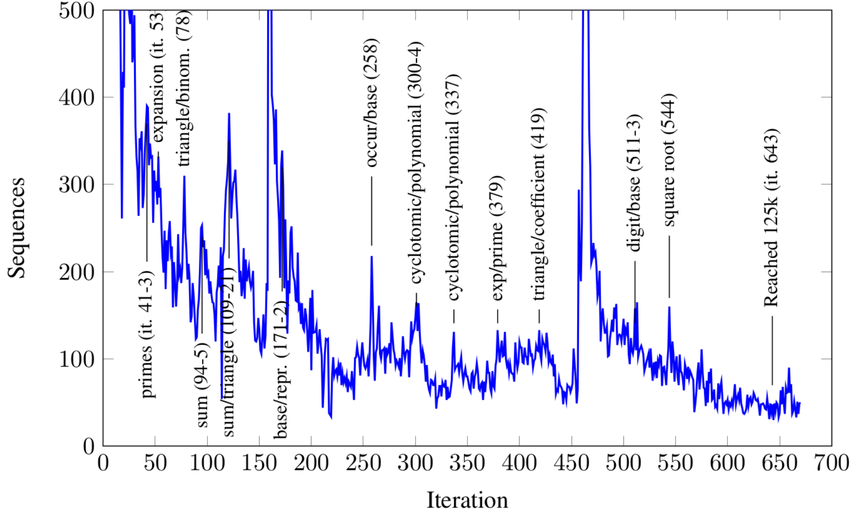
2.5 years of program evolution



2.5 years of program evolution



2.5 years of program evolution



Generalization of the programs to extra terms

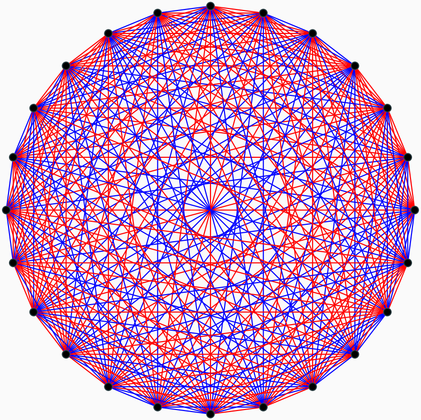
- 90.57% of the slow programs are correct on 100 extra terms.
- 77.51% of the fast programs are correct on 100 extra terms.

Mirek Olšák proved that some programs are correct on all natural numbers.

Generalization to other tasks

- LMFB
- ARC-AGI
- Ramsey graphs
- Inductive theorem proving

Ramsey graphs



Synthesize adjacency matrices of Ramsey graphs (more on Thursday).

A benchmark for inductive theorem provers

- 29687 sequences of with a fast program P and a fast program Q.
- Creation of 29687 SMT problems of the form $\forall x \in \mathbb{N}.\ f_P(x) = f_Q(x)$.

A benchmark for inductive theorem provers

- 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:

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

Conclusion

A general and effective approach to program synthesis. To AGI?

Continued self-improvement from scratch over 2.5 years on the OEIS.

Manual improvements:

- better programming language
- better language model: (size, training regime, training data)
- creation of side tasks, side objectives

Future works:

- create useful definitions/macros
- interleave neural network calls and program calls
- avoid near-misses by multiplying targets