A Mathematical Benchmark for Inductive Theorem Provers

Thibault Gauthier, Chad E. Brown,
Mikoláš Janota, Josef Urban

The 5th International Workshop on Automated (Co)inductive Theorem Proving (Nancy, France, on July 2, 2024)

Overview

Benchmark of **29687 problems** derived from the On-Line Encyclopedia of Integer Sequences (OEIS).

Problems require **arithmetic** and **inductive** reasoning.

They state the equivalence between the **smallest program** and the **fastest program** discovered by QSynt for an OEIS sequence.

Motivation

The OEIS provides finite number of terms for interesting integer sequences (e.g. prime numbers).

Our reinforcement learning system QSynt has discovered **at least 1 program** for approximately 100,000 OEIS sequences.

Motivation

The OEIS provides finite number of terms for interesting integer sequences (e.g. prime numbers).

Our reinforcement learning system QSynt has discovered **at least 1 program** for approximately 100,000 OEIS sequences.

For the same OEIS sequence a program Q may optimize (**be faster**) a program P (assumed to be correct).

Motivation

The OEIS provides finite number of terms for interesting integer sequences (e.g. prime numbers).

Our reinforcement learning system QSynt has discovered **at least 1 program** for approximately 100,000 OEIS sequences.

For the same OEIS sequence a program Q may optimize (**be faster**) a program P (assumed to be correct).

How can we **be sure** that they are actually equal?

Motivation

The OEIS provides finite number of terms for interesting integer sequences (e.g. prime numbers).

Our reinforcement learning system QSynt has discovered **at least 1 program** for approximately 100,000 OEIS sequences.

For the same OEIS sequence a program Q may optimize (**be faster**) a program P (assumed to be correct).

How can we **be sure** that they are actually equal?

Problems in the benchmark: $\forall x \in \mathbb{N}.\ f_P(x) = f_Q(x)$?

1) Program Synthesis for Integer Sequences

2) Benchmark for Inductive Theorem Provers

Program Synthesis for Integer Sequences

OEIS: $\geq$ 350000 Finite 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
(Formerly M0652 N0241) 10150

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.
The paper by Kaoru Motose starts as follows: "Let q be a prime divisor of a Mersenne number 2^p-1 where p is prime. Then p is the order of 2 (mod q). Thus p is a divisor of q - 1 and q > p. This shows that there exist infinitely many prime numbers." - Pieter Moree, Oct 14 2004

Generating programs for OEIS Sequences

$0, 1, 3, 6, 10, 15, 21, \ldots$

Generating programs for OEIS Sequences

$0, 1, 3, 6, 10, 15, 21, \ldots$

An undesirable **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 ...
```

Generating programs for OEIS Sequences

$0, 1, 3, 6, 10, 15, 21, \ldots$

An undesirable **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 ...
```

Small program:

$$\sum_{i=1}^{n} i$$

Generating programs for OEIS Sequences

$0, 1, 3, 6, 10, 15, 21, \ldots$

An undesirable **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 ...
```

Small program:

$$\sum_{i=1}^{n} i$$

Fast program:

$$\frac{n \times n + n}{2}$$

Web Interface

QSynt: Program Synthesis for Integer Sequences

Propose a sequence of integers:

`0 1 4 9 16 21 25 28 36 37 49`

Timeout (maximum 300s)

`10`

Generated integers (maximum 100)

`32`

Send Reset

A few sequences you can try:

0 1 1 0 1 0 1 0 0 0 1 0 1 0 0 0 1 0 1
0 1 4 9 16 21 25 28 36 37 49
0 1 3 6 10 15
2 3 5 7 11 13 17 19 23 29 31 37 41 43
1 1 2 6 24 120
2 4 16 256
1 1 2 3 5 8 13 21 34 55 89 144 233

Current version is version 3. Previous versions may find different solutions [Version 2, Version 1].

http://grid01.ciirc.cvut.cz/~thibault/qsynt.html

Web Interface

```
Generated sequence:
0 1 4 9 16 21 25 28 36 37 49 57 60 64 72 81 84 85 88 93 100 105 109 112 120 121 133 141 144 148 156 165
Matches best with: A10445(0-16), A155570(0-7), A10421(0-6)
Program found in cache after a failed search (10.02s):
f(x) := compr(\(x).(loop(\(x).x * x, 2, x) - x) mod (2 * loop(\(x).(x * x) + x, 2, 2)), x)
Execute the equivalent Python program here:
```

Programming Language

- Constants: $0, 1, 2$
- Variables: x, y
- Arithmetic: $+, -, \times, div, mod$
- Condition : if $\ldots \leq 0$ then ...else ...
- $loop(f, a, b) := u_a$ where

$$u_0 = b$$
$$u_n = f(u_{n-1}, n)$$

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

Programming Language

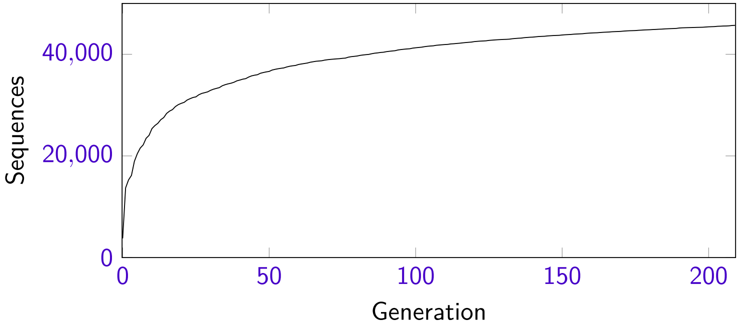
- Constants: $0, 1, 2$
- Variables: x, y
- Arithmetic: $+, -, \times, div, mod$
- Condition : if $\ldots \leq 0$ then ...else ...
- $loop(f, a, b) := u_a$ where

$$u_0 = b$$
$$u_n = f(u_{n-1}, n)$$

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

Example:

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



What does the neural network **learn**?

Rarely **add 0**, rarely **multiply by 1**, use correct arity for operators, and much more.

A45, the Fibonacci sequence: $\mathit{loop2}(x + y, x, x, 0, 1)$

A108, the Catalan numbers: $\binom{2x}{x}\frac{1}{x+1}$

A10051, prime characteristic function:

$$((x \times x!)\ \mathit{mod}\ (1 + x))\ \mathit{mod}\ 2$$

Solutions with Large Numbers

A10445, squares modulo 84:

$$0, 1, 4, 9, 16, 21, 25, 28, 36, 37, 49, 57, 60, 64, 72, 81$$

$$\{x \mid (x^4 - x) \ mod \ 84 = 0\}$$

with $84 = 2 \times f^2(2) \ and \ f(x) = x \times x + x$

A66298, *googol mod x*:

$$10^{10^2} \ mod \ (1 + x)$$

with $10 = 2 \times (2 + 2) + 2$

More Programs!

Inspect newest solutions at
https://github.com/Anon52MI4/oeis-alien

Benchmark for Inductive Theorem Provers

Benchmark of **29687 problems** derived from the On-Line Encyclopedia of Integer Sequences (OEIS).

Problems require **arithmetic** and **inductive** reasoning.

They state the equivalence between the **smallest program** and the **fastest program** discovered by QSynt for an OEIS sequence.

Motivation: A Manual Proof

Josef:

Now it got `https://oeis.org/A336614`, which people recently discussed here: `https://math.stackexchange.com/questions/4163446/number-of-n-times-n-0-1` Again, I am very curious if the invented program **corresponds** to their formulas.

Mirek:

The invented formula was a bit more complicated (two nested sums) but it is possible to mathematically also **convert** it to the formula in the Stack Exchange.

Benchmark

29687 sequences of with a small program P and a fast program Q.

Creation of 29687 problems of the form:

$$\forall x \in \mathbb{N}.\ f_P(x) = f_Q(x)$$

Checked on the first 100 natural numbers.

Can we prove that they hold on all natural numbers?

Problems in the Benchmark

- A217, triangular numbers:

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

- A537, sum of first n cubes:

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

- A79, powers of 2:

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

- A165, double factorial of even numbers,

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

Translation to SMT

We tried to make our translation **natural** (close to the original problem) and **efficient** (targeting the strength of the provers).

How do you **translate loops**?

Definition: $loop(f, a, b) := u_a$ where $u_0 = b, u_n = f(u_{n-1}, n)$

Idea: use **recursive functions** to translate loops

Translation to SMT of $\sum_{i=1}^{n} i := loop(X + Y, X, 1)$

$$u(x) = \text{if } x \leq 0 \text{ then } 1 \text{ else } u(x-1, y) + x$$

A Simple Problem (not in our benchmark)

A simple example for $0, 2, 4, 6, 8, \ldots$ with two programs f and g:

- $f(0) = 0, f(n) = 2 + f(n-1)$ if $n > 0$
- $g(n) = 2 \times n$
- conjecture: $\forall n \in \mathbb{N}.\ g(n) = f(n)$

```
(set-logic UFLIA)
(define-fun-rec f ((x Int)) Int (ite (<= x 0) 0 (+ 2 (f (- x 1)))))
(assert (exists ((c Int))
        (and (> c 0) (not (= (f c) (* 2 c))))))
(check-sat)
```

Results

Which provers supports arithmetic, **induction** and the SMT format?

Vampire and cvc5.

Supports for induction requires adding extra rules to the prover based on the **Peano's induction axiom schema**.
If you want to know more about it, ask Petra who is improving induction for Vampire.

Induction only recently being supported and we hope that our benchmark will create **further interest** for improving inductive reasoning in theorem provers.

Results: Provers Run for 60 seconds

	Z3	Vampire	cvc5
29,687 problems	4,757	2,195	2,428
Syntactic filtering: 23,163 problems	487	278	893
Semantic filtering: 16,197 problems	7	83	504

Can we filter à priori problems that do not **require induction**?

Results: Example

Solved only by cvc5 and Vampire:

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

Solved only by Vampire:

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

Conclusion

We created benchmark of **29,687 SMT problems**. It contains **inductive** and **arithmetical** problems automatically derived by a **program synthesis** system from **OEIS** sequences. This creates **interesting** problems of **various difficulties**.

Conclusion

We created benchmark of **29,687 SMT problems**. It contains **inductive** and **arithmetical** problems automatically derived by a **program synthesis** system from **OEIS** sequences. This creates **interesting** problems of **various difficulties**.

Want to get rich? Some problems have bounties in **ProofGold**. ProofGold is a both a proof checker and a cryptocurrency.

Possible inclusion of part of the benchmark in the **TPTP**. (when Thibault finds the time to send some problems to Geoff)