ALIEN CODING: LEARNING SYNTHESIS OF OEIS SEQUENCES

Thibault Gauthier, Miroslav Olšák, Josef Urban

Czech Technical University in Prague

Dagstuhl seminar
Automated mathematics: integrating proofs, algorithms and data
October 1 – 6, 2023

https://github.com/Anon52MI4/oeis-alien

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

What Are the Current AI/TP TODOs/Bottlenecks?

- AI/TP main agenda in 2011-20: high/low/mid-level ML/RL guidance of TP:
- 2017/18: TacticToe (Gauthier et al): 70% of HOL proved automatically
- 2021: ENIGMA (Jakubuv et al): 60/75% of Mizar proved automatically
- More AITP agenda/propaganda – AGI'18: `https://bit.ly/3qifhg4`
- Is that all? What are the current bottlenecks?
- High-level structuring of proofs - proposing good lemmas
- Proposing new concepts, definitions and theories
- Proposing new targeted algorithms, decision procedures, tactics
- Proposing good witnesses for existential proofs
- All these problems involve synthesis of some mathematical objects
- Btw., constructing proofs is also a synthesis task
- This talk: explore learning-guided synthesis for OEIS
- Interesting research topic and tradeoff in learning/AI/proving:
- Learning direct guessing of objects (this talk) vs guidance for search procedures (ENIGMA and friends)
- Start looking also at semantics rather than just syntax of the objects

Quotes: Learning vs. Reasoning vs. Guessing

"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

QSynt: Semantics-Aware Synthesis of Math Objects

- Gauthier (et al) 2019-23
- Synthesize math expressions based on semantic characterizations
- i.e., not just on the syntactic descriptions (e.g. proof situations)
- Tree Neural Nets and Monte Carlo Tree Search (a la AlphaZero)
- Recently also various (small) *language models* with their search methods
- Invent programs for OEIS sequences FROM SCRATCH (no LLM cheats)
- 100k OEIS sequences (out of 350k) solved so far (411 iterations):
  `https://www.youtube.com/watch?v=24oejR9wsXs`,
  `http://grid01.ciirc.cvut.cz/~thibault/qsynt.html`
- 2.4M explanations invented: 50+ different characterizations of primes
- Non-neural (Turing complete) symbolic computing and semantics collaborate with the statistical/neural learning
- Program evolution governed by high-level criteria (Occam, efficiency)

Connections to this seminar topics

- Generating examples and counterexamples?
- Creating concepts?
- Finding explanations for patterns?
- In particular symbolic ones? (no just some ugly big transformer)
- How do we make conjectures, synthesize math objects?
- When generating conjectures, how do we compare/order them?
- How do we combine statistical and symbolic ML over math?
- Designing long-running exploration/evolution systems and feedback loops combining search/guessing, verification, learning, proving ... ?

OEIS: ⩾ 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.

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 (Occam's Razor):

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

Fast program (efficiency criteria):

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

Programming language

- Constants: $0, 1, 2$
- Variables: x, y
- Arithmetic: $+, -, \times, \mathit{div}, \mathit{mod}$
- 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:

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

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

QSynt: synthesizing the programs/expressions

- Inductively defined set P of our *programs and subprograms*,
- and an auxiliary set F of binary functions (higher-order arguments)
- are the smallest sets such that $0, 1, 2, x, y \in P$, and if $a, b, c \in P$ and $f, g \in F$ then:

$$a + b, a - b, a \times b, a \; div \; b, a \; mod \; b, cond(a, b, c) \in P$$

$$\lambda(x, y).a \in F, \; loop(f, a, b), loop2(f, g, a, b, c), compr(f, a) \in P$$

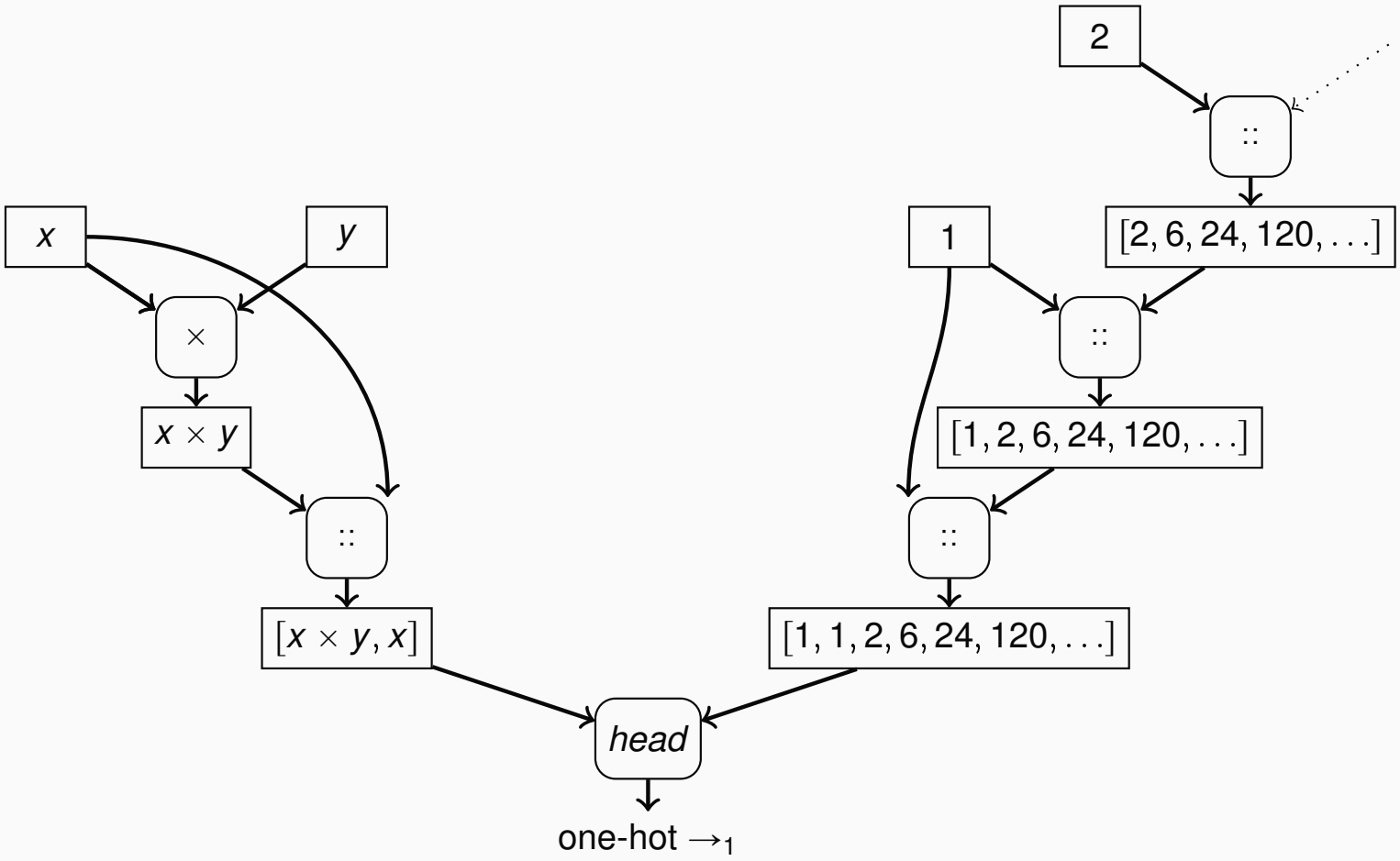
- Programs are built in reverse polish notation
- Start from an empty stack
- Use ML to repeatedly choose the next operator to push on top of a stack
- Example: Factorial is $loop(\lambda(x, y).\ x \times y, x, 1)$, built by:

$$[\,] \rightarrow_x [x] \rightarrow_y [x, y] \rightarrow_\times [x \times y] \rightarrow_x [x \times y, x]$$

$$\rightarrow_1 [x \times y, x, 1] \rightarrow_{loop} [loop(\lambda(x, y).\ x \times y, x, 1)]$$

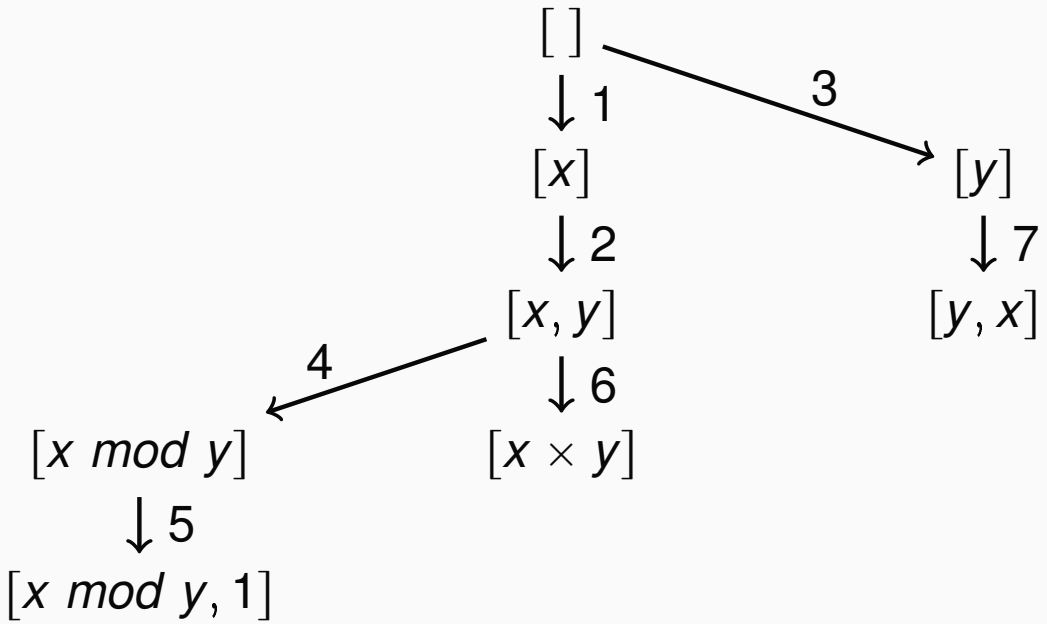
QSynt: Training of the Neural Net Guiding the Search

- The triple $((\mathit{head}([x \times y, x], [1, 1, 2, 6, 24, 120 \ldots]),\ \rightarrow_1)$ is a training example extracted from the program for factorial $\mathit{loop}(\lambda(x, y).\ x \times y, x, 1)$
- $\rightarrow_1$ is the action (adding 1 to the stack) required on $[x \times y, x]$ to progress towards the construction of $\mathit{loop}(\lambda(x, y).\ x \times y, x, 1)$.



QSynt program search - Monte Carlo search tree

7 iterations of the tree search gradually extending the search tree. The set of the synthesized programs after the 7th iteration is $\{1, x, y, x \times y, x \ mod \ y\}$.

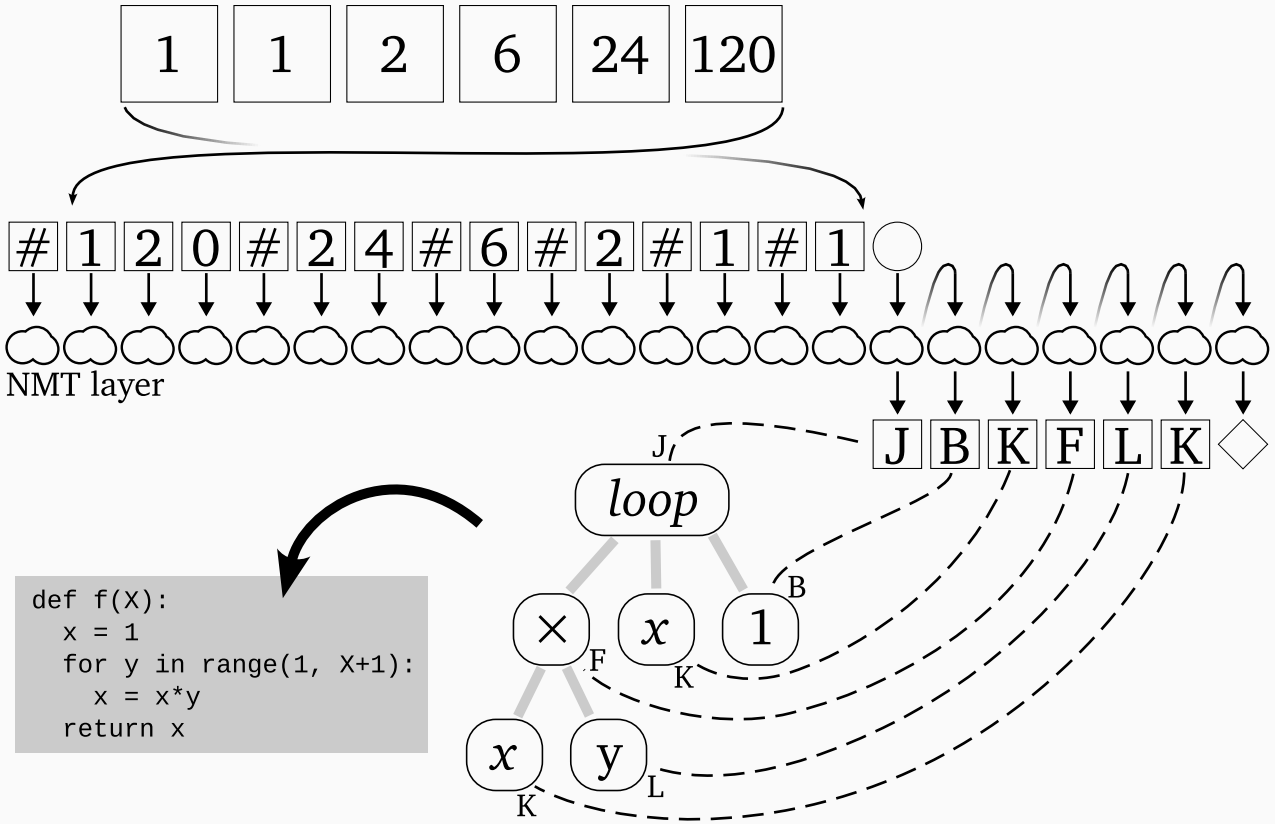


Using Language Models for Math Tasks

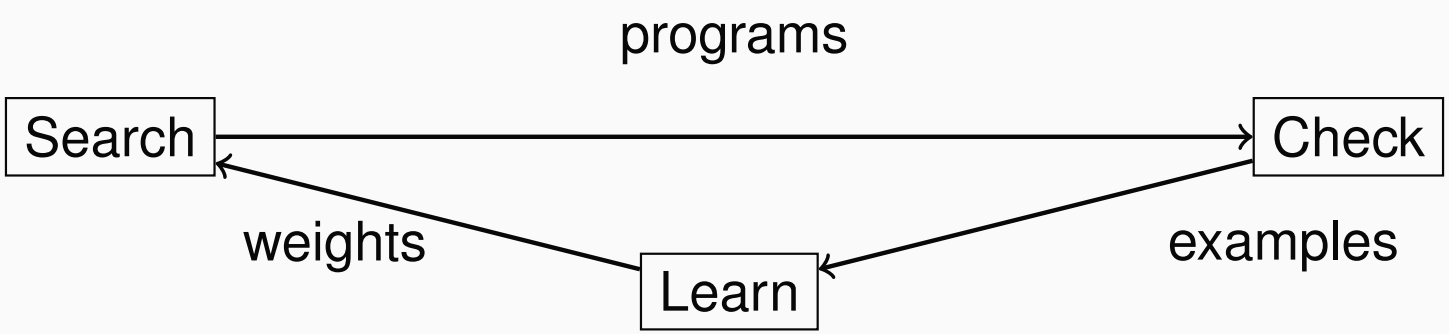
- Recurrent neural networks (RNNs) with attention (NMT)
- Transformers (BERT, GPT)
- Since 2018/19 LMs applied to symbolic/mathematical tasks:
- ... autoformalization - translation from informal to formal (Wang et al'18)
- ... rewriting (Piotrowski et al'19), conjecturing (Chvalovsky et al'19), etc.
- Formulated as sequence-to-sequence translation tasks
- Efficient training and inference on GPUs, many toolkits
- Can get expensive for large LMs (LLMs) ($5M for GPT-3)
- We use small models on old HW, our total energy bill is below $1000

Encoding OEIS for Language Models

- Input sequence is a series of digits
- Separated by an additional token # at the integer boundaries
- Output program is a sequence of tokens in Polish notation
- Parsed by us to a syntax tree and translatable to Python
- Example: $a(n) = n!$



Search-Verify-Train Feedback Loop



Analogous to our Prove/Learn feedback loops in learning-guided proving (since 2006 – MaLARea)

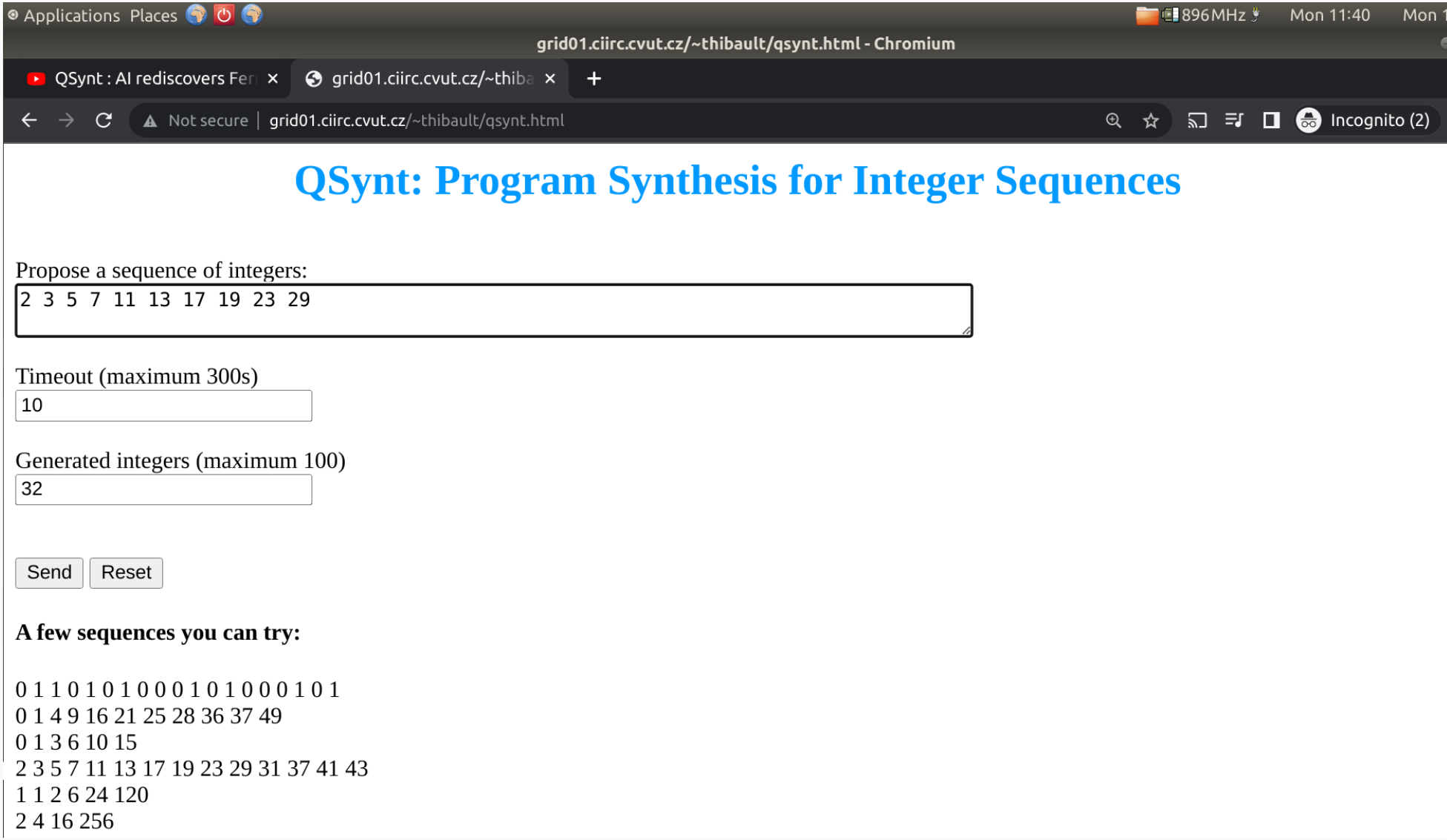
Search-Verify-Train Feedback Loop for OEIS

- **search phase**: LM synthesizes many programs for input sequences
- typically 240 candidate programs for each input using beam search
- 84M programs for OEIS in several hours on the GPU (depends on model)
- **checking phase**: the millions of programs efficiently evaluated
- resource limits used, fast indexing structures for OEIS sequences
- check if the program generates *any* OEIS sequence (hindsight replay)
- we keep the shortest (Occams's razor) and fastest program (efficiency)
- **learning phase**: LM trains to translate the "solved" OEIS sequences into the best program(s) generating them

Search-Verify-Train Feedback Loop

- The weights of the LM either trained from scratch or continuously updated
- This yields *new minds* vs *seasoned experts* (who have seen it all)
- We also train experts on varied selections of data, in varied ways
- Orthogonality: common in theorem proving - different experts help
- Each iteration of the self-learning loop discovers more solutions
- ... also improves/optimizes existing solutions
- The alien mathematician thus self-evolves
- Occam's razor and efficiency are used for its weak supervision
- Quite different from today's LLM approaches:
- LLMs do one-time training on everything human-invented
- Our alien instead starts from zero knowledge
- Evolves increasingly nontrivial skills, may diverge from humans
- Turing complete (unlike Go/Chess) – arbitrary complex algorithms

QSynt web interface for program invention



QSynt inventing Fermat pseudoprimes

Positive integers k such that $2^k \equiv 2 \mod k$. ($341 = 11 * 31$ is the first non-prime)

```
First 16 generated numbers (f(0),f(1),f(2),...):
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53
Generated sequence matches best with: A15919(1-75), A100726(0-59), A40(0-58)

Program found in 5.81 seconds
f(x) := 2 + compr(\x.loop(\(x,i).2*x + 2, x, 2) mod (x + 2), x)
Run the equivalent Python program here or in the window below:
```

BRYTHON

Tutorial Demo Documentation Console Editor Gallery Resources

English

Brython version: 3.10.6

run | Python | Javascript | Share code

```
def f2(X):
  x = 2
  for i in range (1,X + 1):
    x = 2*x + 2
  return x

def f1(X):
  x,i = 0,0
  while i <= X:
    if f2(x) % (x + 2) <= 0:
      i = i + 1
    x = x + 1
  return x - 1

def f0(X):
  return 2 + f1(X)

for x in range(32):
  print (f0(x))
```

```
2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
53
59
61
67
```

Lucas/Fibonacci characterization of (pseudo)primes

```
input sequence: 2,3,5,7,11,13,17,19,23,29

invented output program:
f(x) := compr(\(x,y).(loop2(\(x,y).x + y, \(x,y).x, x, 1, 2) - 1)
                mod (1 + x), x + 1) + 1

human conjecture: x is prime iff? x divides (Lucas(x) - 1)

PARI program:
? lucas(n) = fibonacci(n+1)+fibonacci(n-1)
? b(n) = (lucas(n) - 1) % n

Counterexamples (Bruckman-Lucas pseudoprimes):
? for(n=1,4000,if(b(n)==0,if(isprime(n),0,print(n))))
1
705
2465
2737
3745
```

QSynt inventing primes using Wilson's theorem

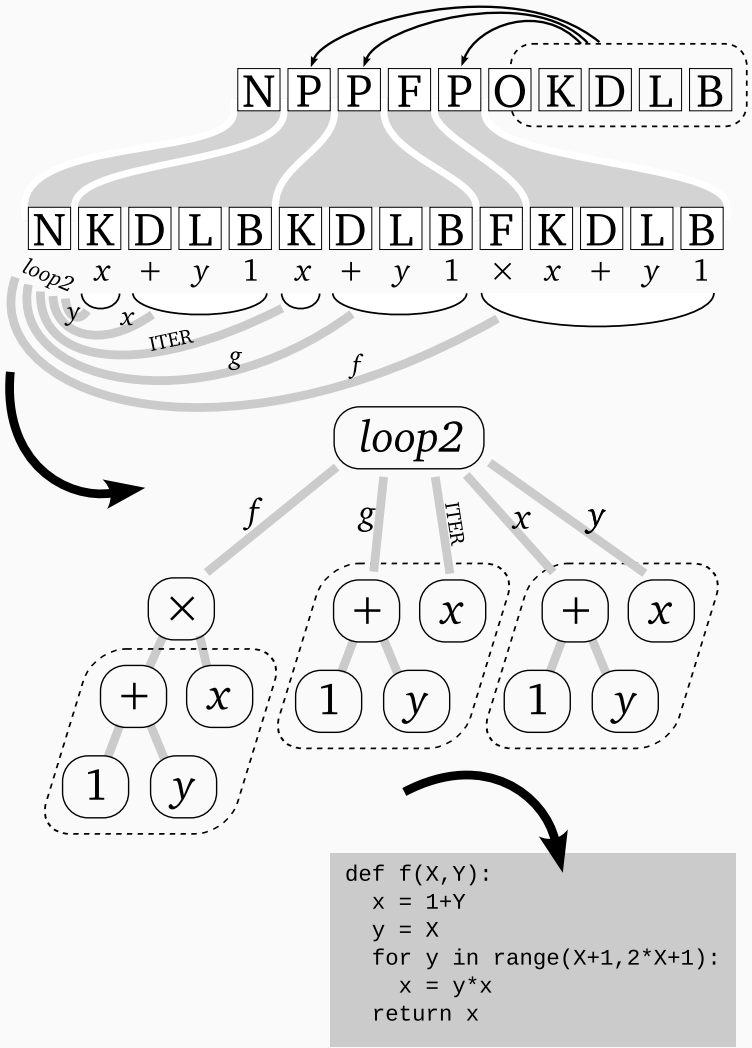
n is prime iff $(n-1)! + 1$ is divisible by n (i.e.: $(n-1)! \equiv -1 \mod n$)

```
First 32 generated numbers (f(0),f(1),f(2),...):
0 1 1 0 1 0 1 0 0 0 1 0 1 0 0 0 1 0 1 0 0 0 1 0 0 0 0 0 1 0 1 0
Generated sequence matches best with: A10051(0-100), A252233(0-29), A283991(0-24)

Program found in 5.17 seconds
f(x) := (loop(\(x,i).x * i, x, x) mod (x + 1)) mod 2
Run the equivalent Python program here or in the window below:
```

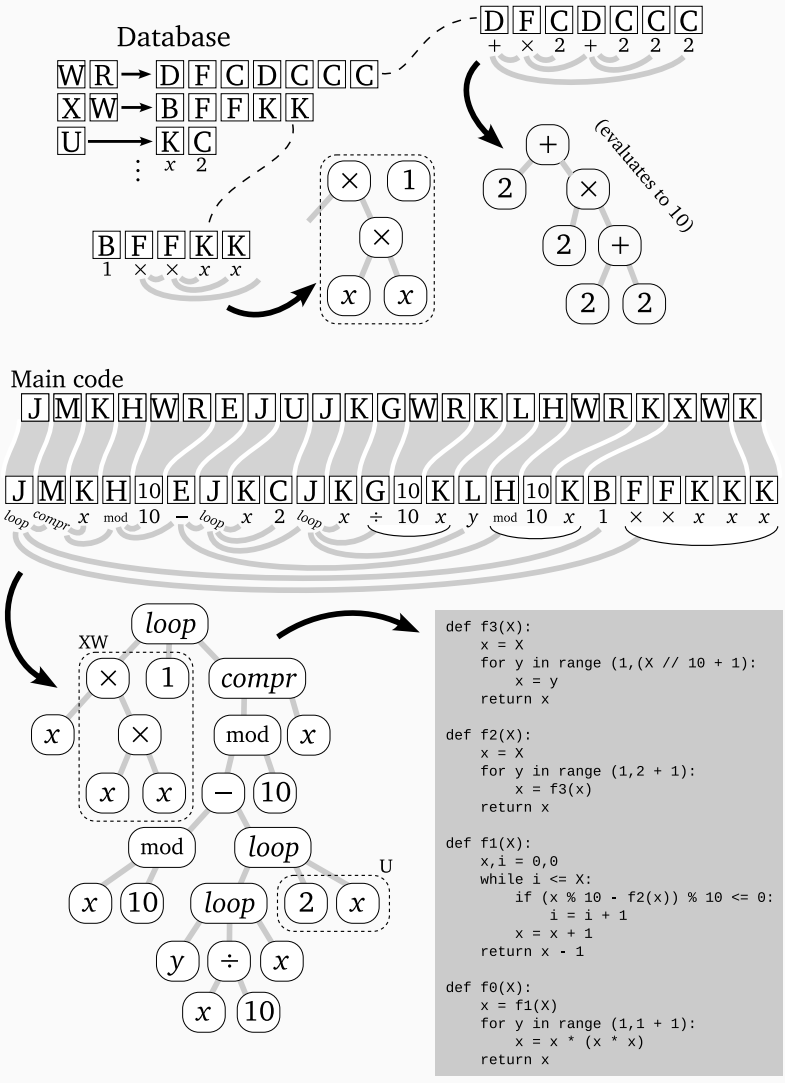
Introducing Local Macros/Definitions

A macro/expanded version of a program invented for A1813: $a(n) = (2n)!/n!$.
1, 2, 12, 120, 1680, 30240, 665280, 17297280, 518918400, 17643225600,



Introducing Global Macros/Definitions

A macro/expanded version of A14187 (cubes of palindromes).
0, 1, 8, 27, 64, 125, 216, 343, 512, 729, 1331, 10648, 35937, 85184,



Five Different Self-Learning Runs

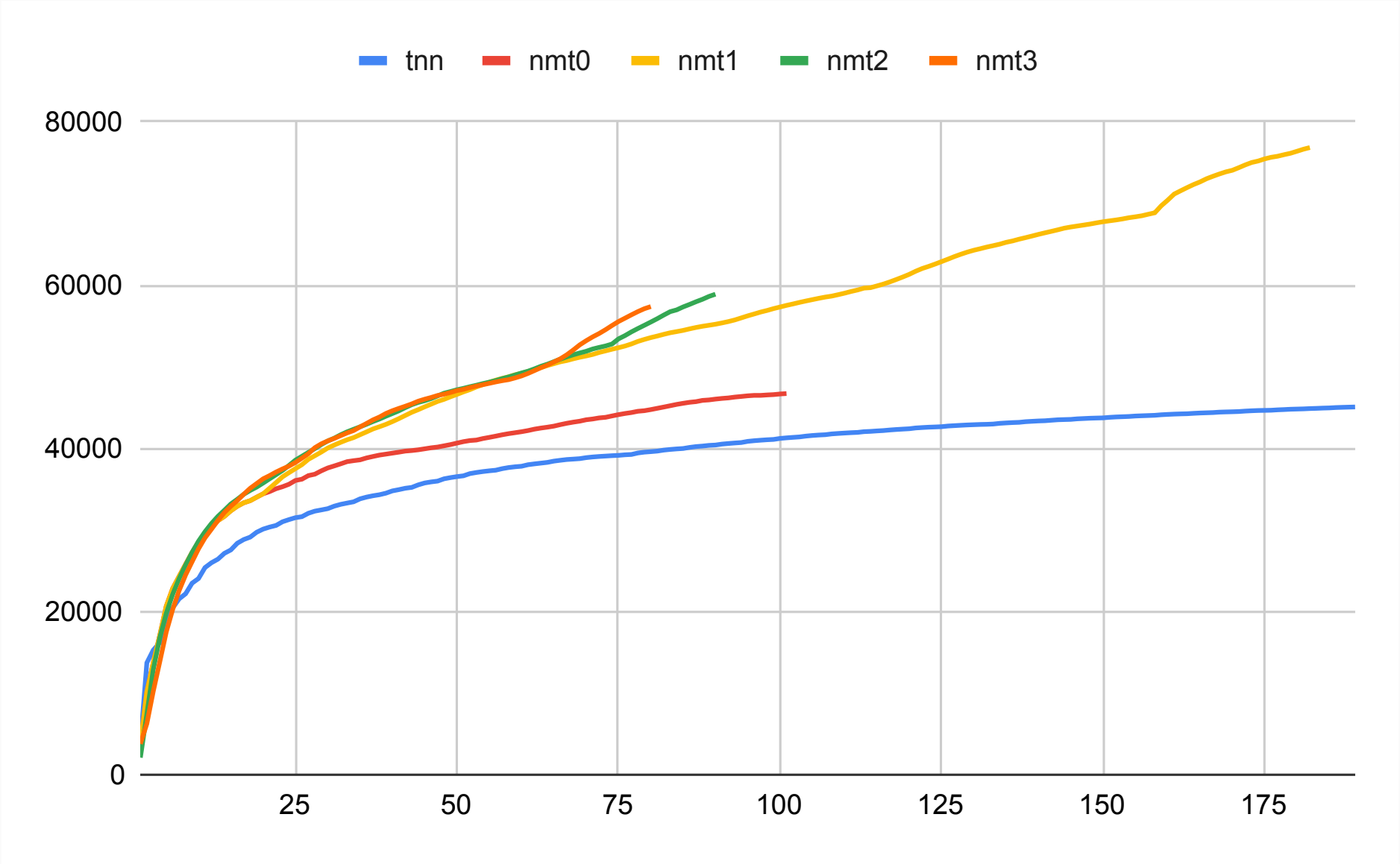


Figure: Cumulative counts of solutions.

Five Different Self-Learning Runs

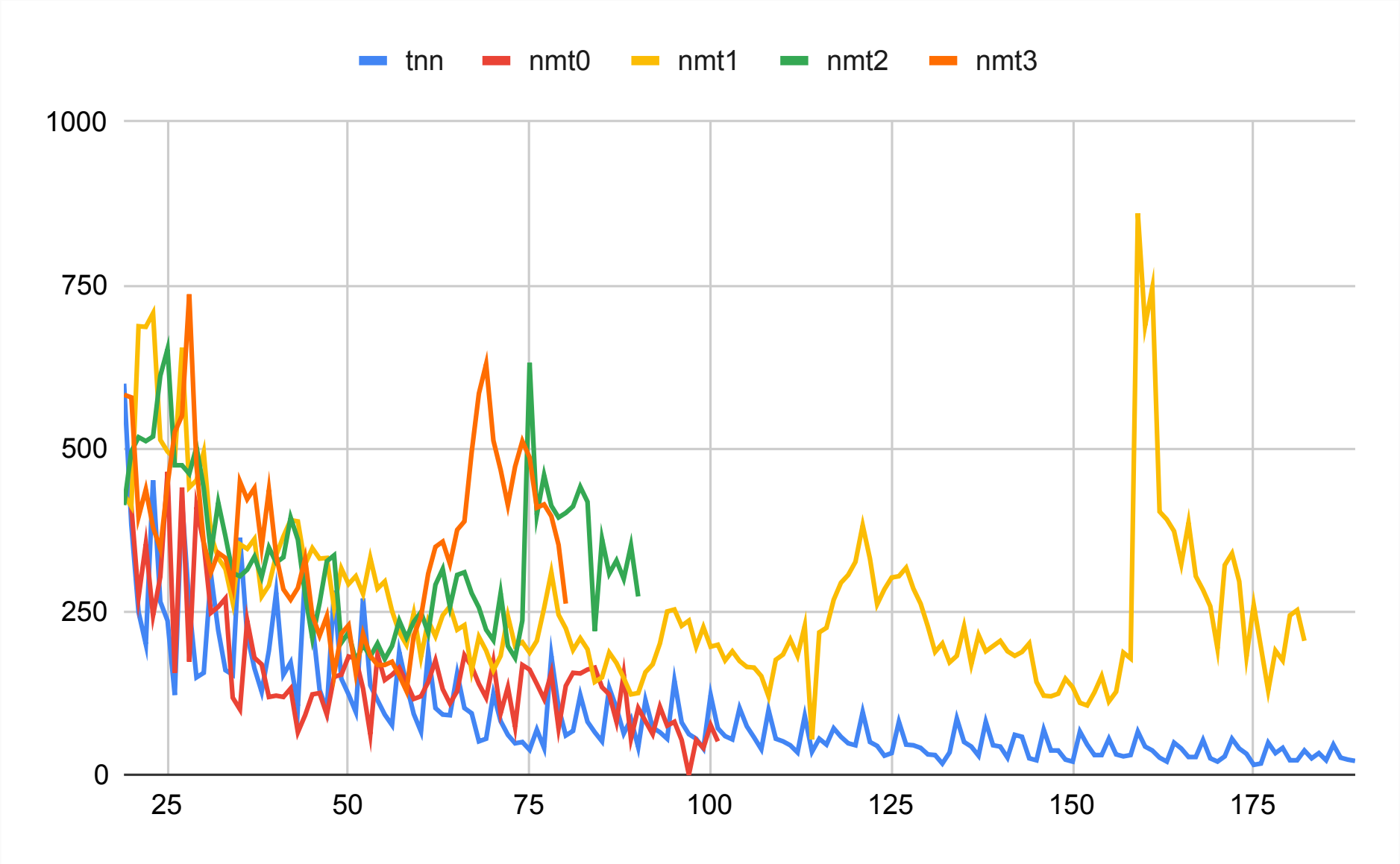


Figure: Increments of solutions.

Size Evolution

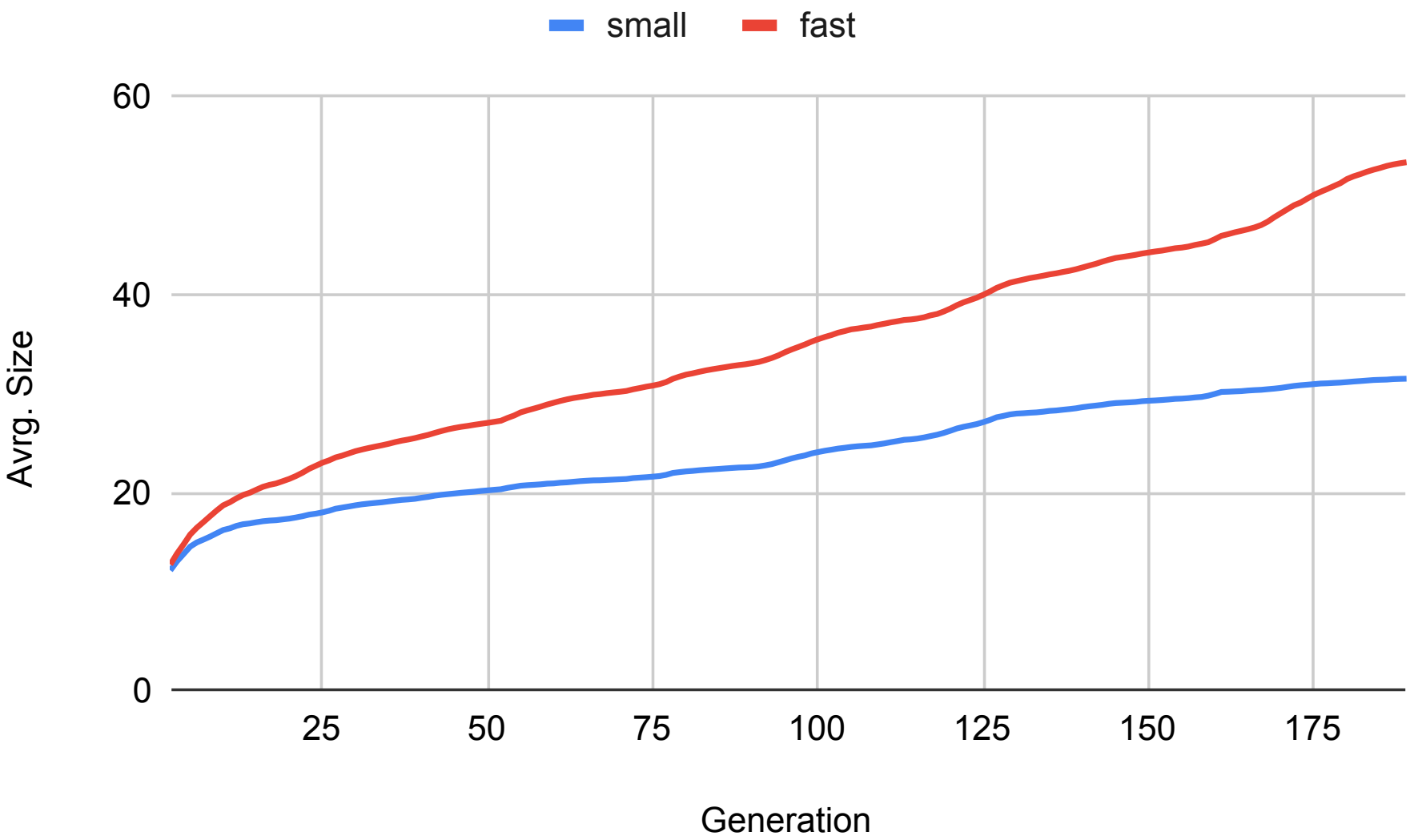


Figure: Avrg. size in iterations

Speed Evolution

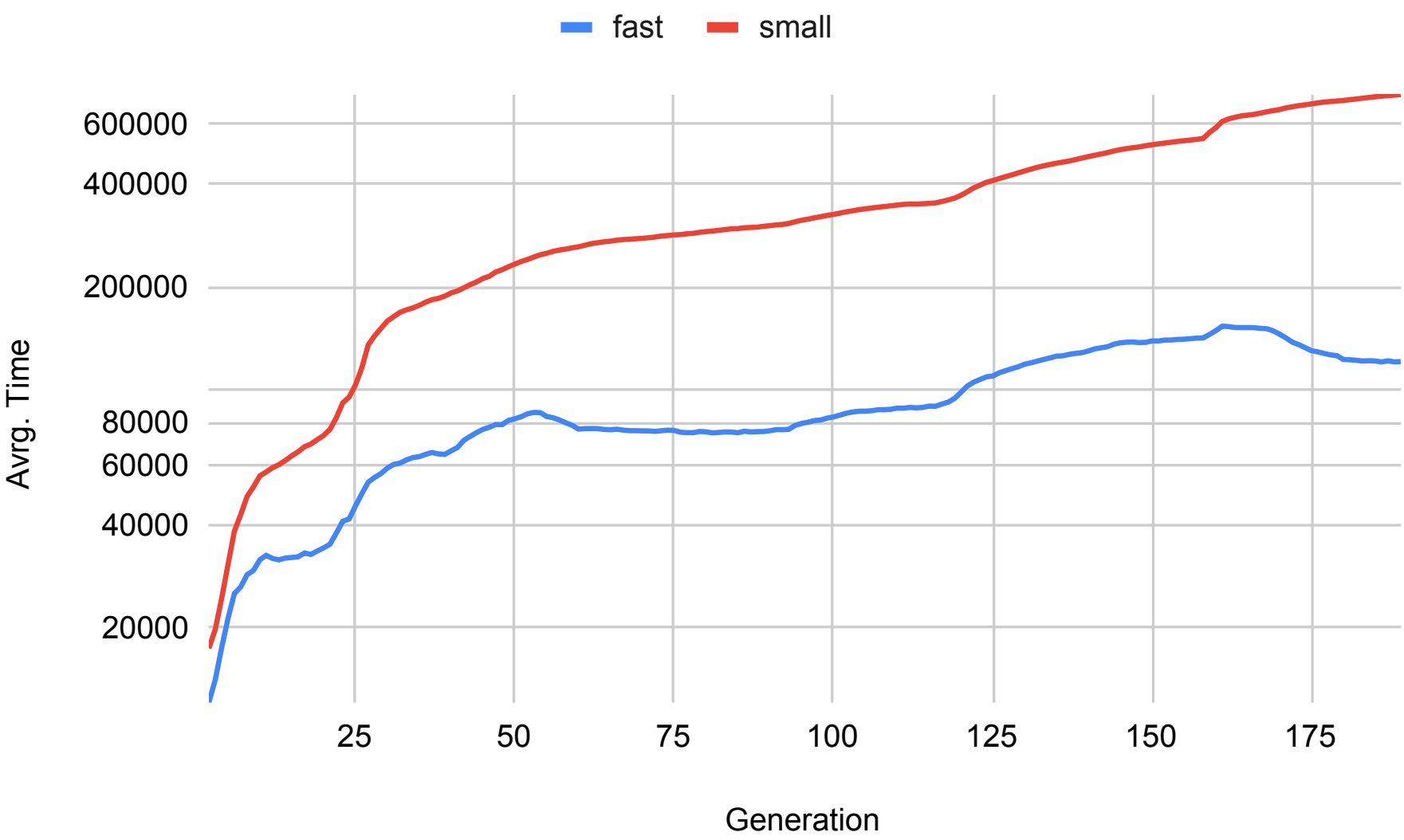


Figure: Avrg. time in iterations

Generalization of the Solutions to Larger Indices

- Are the programs correct?
- OEIS provides additional terms for some of the OEIS entries
- Among 78118 solutions, 40,577 of them have a b-file with 100 terms
- We evaluate both the small and the fast programs on them
- Here, 14,701 small and 11,056 fast programs time out.
- 90.57% of the remaining slow programs check
- 77.51% for the fast programs
- A common error is reliance on an approximation for a real number, such as π.

Are two QSynt programs equivalent?

- As with primes, we often find many programs for one OEIS sequence
- Currently we have almost 2M programs for the 100k sequences
- It may be quite hard to see that the programs are equivalent
- 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 $n > 0$
  - $g(n) = 2 * n$
  - conjecture: $\forall n \in \mathbb{N}.g(n) = f(n)$
- We can ask mathematicians, but we have thousands of such problems
- Or we can try to ask our ATPs (and thus create a large ATP benchmark)!
- Here is one SMT encoding by Mikolas Janota:

```
(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)
```

Inductive proof by Vampire of the $f = g$ equivalence

```
% SZS output start Proof for rec2
1. f(X0) = $ite($lesseq(X0,0), 0,$sum(2,f($difference(X0,1)))) [input]
2. ? [X0 : $int] : ($greater(X0,0) & ~f(X0) = $product(2,X0)) [input]
[...]
43. ~$less(0,X0) | iG0(X0) = $sum(2,iG0($sum(X0,-1))) [evaluation 40]
44. (! [X0 : $int] : (($product(2,X0) = iG0(X0) & ~$less(X0,0)) => $product(2,$sum(X0,1)) = iG0($sum(X0,1)))
    &  $product(2,0) = iG0(0))   => ! [X1 : $int] : ($less(0,X1) => $product(2,X1) = iG0(X1)) [induction hypo]
[...]
49. $product(2,0) != iG0(0) | $product(2,$sum(sK3,1)) != iG0($sum(sK3,1)) | ~$less(0,sK1) [resolution 48,41]
50. $product(2,0) != iG0(0) | $product(2,sK3) = iG0(sK3) | ~$less(0,sK1) [resolution 47,41]
51. $product(2,0) != iG0(0) | ~$less(sK3,0) | ~$less(0,sK1) [resolution 46,41]
52. 0 != iG0(0) | $product(2,$sum(sK3,1)) != iG0($sum(sK3,1)) | ~$less(0,sK1) [evaluation 49]
53. 0 != iG0(0) | $product(2,sK3) = iG0(sK3) | ~$less(0,sK1) [evaluation 50]
54. 0 != iG0(0) | ~$less(sK3,0) | ~$less(0,sK1) [evaluation 51]
55. 0 != iG0(0) | ~$less(sK3,0) [subsumption resolution 54,39]
57. 1 <=> $less(sK3,0) [avatar definition]
59. ~$less(sK3,0) <- (~1) [avatar component clause 57]
61. 2 <=> 0 = iG0(0) [avatar definition]
64. ~1 | ~2 [avatar split clause 55,61,57]
65. 0 != iG0(0) | $product(2,sK3) = iG0(sK3) [subsumption resolution 53,39]
67. 3 <=> $product(2,sK3) = iG0(sK3) [avatar definition]
69. $product(2,sK3) = iG0(sK3) <- (3) [avatar component clause 67]
70. 3 | ~2 [avatar split clause 65,61,67]
71. 0 != iG0(0) | $product(2,$sum(sK3,1)) != iG0($sum(sK3,1)) [subsumption resolution 52,39]
72. $product(2,$sum(1,sK3)) != iG0($sum(1,sK3)) | 0 != iG0(0) [forward demodulation 71,5]
74. 4 <=> $product(2,$sum(1,sK3)) = iG0($sum(1,sK3)) [avatar definition]
76. $product(2,$sum(1,sK3)) != iG0($sum(1,sK3)) <- (~4) [avatar component clause 74]
77. ~2 | ~4 [avatar split clause 72,74,61]
82. 0 = iG0(0) [resolution 36,10]
85. 2 [avatar split clause 82,61]
246. iG0($sum(X1,1)) = $sum(2,iG0($sum($sum(X1,1),-1))) | $less(X1,0) [resolution 43,14]
251. $less(X1,0) | iG0($sum(X1,1)) = $sum(2,iG0(X1)) [evaluation 246]
[...]
1176. $false <- (~1, 3, ~4) [subsumption resolution 1175,1052]
1177. 1 | ~3 | 4 [avatar contradiction clause 1176]
1178. $false [avatar sat refutation 64,70,77,85,1177]
% SZS output end Proof for rec2
% Time elapsed: 0.016 s
```

80 Programs That Have Most Evolved

120	https://oeis.org/A238952	101	https://oeis.org/A97012	98	https://oeis.org/A17666
117	https://oeis.org/A35218	101	https://oeis.org/A71190	98	https://oeis.org/A113184
116	https://oeis.org/A1001	101	https://oeis.org/A70824	97	https://oeis.org/A82
112	https://oeis.org/A35178	101	https://oeis.org/A64987	97	https://oeis.org/A6579
111	https://oeis.org/A88580	101	https://oeis.org/A57660	97	https://oeis.org/A56595
111	https://oeis.org/A62069	101	https://oeis.org/A54024	97	https://oeis.org/A293228
111	https://oeis.org/A163109	101	https://oeis.org/A53222	97	https://oeis.org/A27847
111	https://oeis.org/A1615	101	https://oeis.org/A50457	97	https://oeis.org/A23645
109	https://oeis.org/A66446	101	https://oeis.org/A23888	97	https://oeis.org/A10
108	https://oeis.org/A48250	101	https://oeis.org/A209295	96	https://oeis.org/A92403
108	https://oeis.org/A321516	101	https://oeis.org/A206787	96	https://oeis.org/A90395
108	https://oeis.org/A2654	100	https://oeis.org/A99184	96	https://oeis.org/A83919
107	https://oeis.org/A75653	100	https://oeis.org/A63659	96	https://oeis.org/A7862
107	https://oeis.org/A60278	100	https://oeis.org/A62968	96	https://oeis.org/A78306
107	https://oeis.org/A23890	100	https://oeis.org/A35154	96	https://oeis.org/A69930
106	https://oeis.org/A62011	100	https://oeis.org/A339965	96	https://oeis.org/A69192
106	https://oeis.org/A346613	100	https://oeis.org/A277791	96	https://oeis.org/A54519
106	https://oeis.org/A344465	100	https://oeis.org/A230593	96	https://oeis.org/A53158
105	https://oeis.org/A49820	100	https://oeis.org/A182627	96	https://oeis.org/A351267
104	https://oeis.org/A55155	99	https://oeis.org/A9191	96	https://oeis.org/A334136
104	https://oeis.org/A349215	99	https://oeis.org/A82051	96	https://oeis.org/A33272
104	https://oeis.org/A143348	99	https://oeis.org/A62354	96	https://oeis.org/A325939
103	https://oeis.org/A92517	99	https://oeis.org/A247146	96	https://oeis.org/A211779
103	https://oeis.org/A64840	99	https://oeis.org/A211261	96	https://oeis.org/A186099
102	https://oeis.org/A9194	99	https://oeis.org/A147588	96	https://oeis.org/A143152
102	https://oeis.org/A51953	98	https://oeis.org/A318446	96	https://oeis.org/A125168
102	https://oeis.org/A155085	98	https://oeis.org/A203		

Evolution and Proliferation of Primes and Others

`https://bit.ly/3XHZsjK`: triangle coding, sigma (sum of divisors), primes. `https://bit.ly/3iJ4oGd` (the first 24, now 50)

Nr	Program
P1	(if x <= 0 then 2 else 1) + (compr (((loop (x + x) (x mod 2) (loop (x * x) 1 (loop (x + x) (x div 2) 1))) + x) mod (1 + x)) x)
P2	1 + (compr ((((loop (x * x) 1 (loop (x + x) (x div 2) 1)) + x) * x) mod (1 + x)) (1 + x))
P3	1 + (compr (((loop (x * x) 1 (loop (x + x) (x div 2) 1)) + x) mod (1 + x)) (1 + x))
P4	2 + (compr ((loop2 (1 + (if (x mod (1 + y)) <= 0 then 0 else x)) (y - 1) x 1 x) mod (1 + x)) x)
P5	1 + (compr ((loop (if (x mod (1 + y)) <= 0 then (1 + y) else x) x (1 + x)) mod (1 + x)) (1 + x))
P6	1 + (compr ((loop (if (x mod (1 + y)) <= 0 then (1 + y) else x) (2 + (x div (2 + (2 + 2)))) (1 + x)) mod (1 + x)) (1 + x))
P7	compr ((1 + (loop (if (x mod (1 + y)) <= 0 then (1 + y) else x) x x)) mod (1 + x)) (2 + x)
P8	1 + (compr ((loop (if (x mod (1 + y)) <= 0 then (1 + y) else x) (1 + ((2 + x) div (2 + (2 + 2)))) (1 + x)) mod (1 + x)) (1 + x))
P9	compr (x - (loop (if (x mod (1 + y)) <= 0 then (1 + y) else x) x x)) (2 + x)
P10	compr (x - (loop (if (x mod (1 + y)) <= 0 then 2 else x) (x div 2) x)) (2 + x)
P11	1 + (compr ((loop (if (x mod (1 + y)) <= 0 then (1 + y) else x) (1 + (x div (2 + (2 + 2)))) (1 + x)) mod (1 + x)) (1 + x))
P12	compr ((x - (loop (if (x mod (1 + y)) <= 0 then y else x) x x)) - 2) (2 + x)
P13	1 + (compr ((loop (if (x mod (1 + y)) <= 0 then (1 + y) else x) (2 + (x div (2 * (2 + (2 + 2)))) (1 + x)) mod (1 + x)) (1 + x))
P14	compr ((x - (loop (if (x mod (1 + y)) <= 0 then y else x) x x)) - 1) (2 + x)
P15	1 + (compr (x - (loop (if (x mod (1 + y)) <= 0 then (1 + y) else x) (2 + (x div (2 * (2 + (2 + 2)))) (1 + x))) (1 + x))
P16	compr (2 - (loop (if (x mod (1 + y)) <= 0 then 0 else x) (x - 2) x)) x
P17	1 + (compr (x - (loop (if (x mod (1 + y)) <= 0 then 2 else x) (2 + (x div (2 * (2 + (2 + 2)))) (1 + x))) (1 + x))
P18	1 + (compr (x - (loop (if (x mod (1 + y)) <= 0 then 2 else x) (1 + (2 + (x div (2 * (2 * (2 + 2)))) (1 + x))) (1 + x))
P19	1 + (compr (x - (loop2 (loop (if (x mod (1 + y)) <= 0 then 2 else x) (2 + (y div (2 * (2 + (2 + 2)))) (1 + y)) 0 (1 - (x mod 2)) 1 x)) (1 + x))
P20	1 + (compr (x - (loop2 (loop (if (x mod (1 + y)) <= 0 then 2 else x) (1 + (2 + (y div (2 * (2 * (2 + 2)))) (1 + y)) 0 (1 - (x mod 2)) 1 x)) (1 + x))
P21	1 + (compr (x - (loop2 (loop (if (x mod (2 + y)) <= 0 then 2 else x) (2 + (y div (2 * ((2 + 2) + (2 + 2)))) (1 + y)) 0 (1 - (x mod 2)) 1 x)) (1 + x))
P22	1 + (compr (x - (loop2 (loop (if (x mod (2 + y)) <= 0 then 2 else x) (2 + (y div (2 * (2 * (2 + 2)))) (1 + y)) 0 (1 - (x mod 2)) 1 x)) (1 + x))
P23	2 + (compr (loop (x - (if (x mod (1 + y)) <= 0 then 0 else 1)) x x) x)
P24	loop (1 + x) (1 - x) (1 + (2 * (compr (x - (loop (if (x mod (2 + y)) <= 0 then 1 else x) (2 + (x div (2 * (2 + 2)))) (1 + (x + x)))) x)))

Evolution and Proliferation of Primes

Iter	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20	P21	P22	P23	P24
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	4	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32	6	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33	8	1	6	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
34	12	4	6	6	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
35	7	12	6	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
36	4	10	6	0	17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
37	3	4	6	0	18	6	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
38	2	3	1	0	12	18	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
39	2	3	1	0	9	56	31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
40	2	5	2	0	7	59	49	9	1	0	2	0	1	0	0	0	0	0	0	0	0	0	0	0
41	1	2	3	0	4	52	58	42	23	0	13	0	8	0	0	0	0	0	0	0	0	0	0	0
42	0	2	4	0	3	44	50	38	60	8	11	0	55	0	0	0	0	0	0	0	0	0	0	0
43	0	2	12	0	0	37	55	14	116	35	16	7	90	0	0	0	0	0	0	0	0	0	0	0
44	0	2	13	0	0	28	40	6	176	73	19	8	122	9	12	0	0	0	0	0	0	0	0	0
45	0	2	9	0	0	19	24	4	147	185	26	16	94	25	29	0	7	0	0	0	0	0	0	0
46	0	2	4	0	0	11	14	0	101	256	21	14	66	64	30	0	29	0	0	0	0	0	0	0
47	0	0	0	0	0	9	4	0	55	290	23	3	43	116	16	6	62	14	0	0	0	0	0	0
48	0	0	0	0	0	8	0	0	22	261	16	0	34	192	10	6	89	30	0	0	0	0	0	0
49	0	0	0	0	0	8	0	0	6	195	11	0	36	225	8	6	99	34	0	0	0	0	0	0
50	0	0	0	0	0	5	0	0	2	154	8	0	29	168	6	6	108	39	0	0	0	0	0	0
51	0	0	0	0	0	4	0	0	0	121	7	0	21	97	6	6	113	43	0	0	0	0	0	0
52	0	0	0	0	0	2	0	0	0	118	8	0	12	62	6	6	110	51	0	0	0	0	0	0
53	0	0	0	0	0	1	0	0	0	59	7	0	15	33	6	6	125	62	0	0	0	0	0	0
54	0	0	0	0	0	1	0	0	0	41	4	0	16	17	6	9	137	72	0	0	0	0	0	0
55	0	0	0	0	0	2	0	0	0	32	4	0	15	9	6	17	147	82	0	0	0	0	0	0
56	0	0	0	0	0	1	0	0	0	29	4	0	10	7	6	39	152	98	0	0	0	0	0	0

Evolution and Proliferation of Primes

Iter	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20	P21	P22	P23	P24
57	0	0	0	0	0	1	0	0	1	28	3	0	9	5	6	103	142	108	0	0	0	0	0	0
58	0	0	0	0	0	0	0	0	1	17	3	0	7	4	6	146	146	120	0	0	0	0	0	0
59	0	0	0	0	0	0	0	0	1	11	3	0	6	2	0	179	153	122	0	0	0	0	0	0
60	0	0	0	0	0	0	0	0	1	6	3	0	3	2	0	206	148	121	0	0	0	0	0	0
61	0	0	0	0	0	0	0	0	0	6	3	0	3	1	0	220	139	138	0	0	0	0	0	0
62	0	0	0	0	0	0	0	0	0	5	3	0	2	0	0	245	118	145	0	0	0	0	0	0
63	0	0	0	0	0	0	0	0	0	5	3	0	2	0	0	263	103	160	0	0	0	0	0	0
64	0	0	3	0	0	0	0	0	0	6	4	0	2	0	0	284	87	162	0	0	0	0	0	0
65	0	0	13	0	0	0	0	0	0	5	4	0	1	0	0	303	74	173	0	0	0	0	0	0
66	0	0	34	0	0	0	0	0	0	3	2	0	1	0	0	315	67	174	0	0	0	0	0	0
67	0	0	53	0	0	0	0	0	0	1	1	0	1	0	0	321	64	180	0	0	0	0	0	0
68	0	0	61	0	0	0	0	0	0	1	1	0	1	0	0	323	66	178	0	0	0	0	0	0
69	0	0	67	0	0	0	0	0	0	1	1	0	1	0	0	325	63	178	0	0	0	0	0	0
70	0	0	70	0	0	0	0	0	0	1	1	0	1	0	0	324	60	182	0	0	0	0	0	0
71	0	0	72	0	0	0	0	0	0	1	1	0	1	0	0	330	56	181	0	0	0	0	0	0
72	0	0	73	0	0	0	0	0	0	0	2	0	1	0	0	332	57	190	0	0	0	0	0	0
73	0	0	72	0	0	0	0	0	0	0	2	0	1	0	0	330	58	191	0	0	0	0	0	0
74	0	0	71	0	0	0	0	0	0	0	3	0	1	0	0	336	56	189	0	0	0	0	0	0
75	0	0	74	0	0	0	0	0	0	0	2	0	1	0	0	340	55	192	0	0	0	0	0	0
76	0	0	77	0	0	0	0	0	0	0	1	0	0	0	0	341	57	195	0	0	0	0	0	0
77	0	0	79	0	0	0	0	0	0	0	1	0	0	0	0	343	56	191	0	0	0	0	0	0
78	0	0	79	0	0	0	0	0	0	0	1	0	0	0	0	344	57	201	0	0	0	0	0	0
79	0	0	81	0	0	0	0	0	0	0	0	0	0	0	0	344	56	200	0	0	0	0	0	0
80	0	0	80	0	0	0	0	0	0	0	0	0	0	0	0	346	55	210	0	0	0	0	0	0
81	0	0	75	0	0	0	0	0	0	0	0	0	0	0	0	351	55	206	0	0	0	0	0	0
82	0	0	77	0	0	0	0	0	0	0	0	0	0	0	0	354	53	206	0	0	0	0	0	0
83	0	0	77	0	0	0	0	0	0	1	0	0	0	0	0	360	53	207	0	0	0	0	0	0
84	0	0	76	0	0	0	0	0	0	1	0	0	0	0	0	360	53	208	0	0	0	0	0	0
85	0	0	74	0	0	0	0	0	0	1	0	0	0	0	0	363	53	207	0	0	0	0	0	0
86	0	0	75	0	0	0	0	0	0	0	0	0	0	0	0	361	54	205	0	0	0	0	0	0
87	0	0	77	0	0	0	0	0	0	0	0	0	0	0	0	359	54	198	0	0	0	0	0	0
88	0	0	80	0	0	0	0	0	0	1	0	0	0	0	0	359	54	199	0	0	0	0	0	0
89	0	0	83	0	0	0	0	0	0	1	0	0	0	0	0	357	56	196	0	0	0	0	0	0

Evolution and Proliferation of Primes

Iter	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20	P21	P22	P23	P24
90	0	0	82	0	0	0	0	0	0	1	0	0	0	0	0	345	56	194	0	0	0	0	0	0
91	0	0	81	0	0	0	0	0	0	1	0	0	0	0	0	331	50	188	0	0	0	0	0	0
92	0	0	66	0	0	0	0	0	0	0	0	0	0	0	0	322	31	176	9	1	0	0	0	0
93	0	0	56	0	0	0	0	0	0	0	0	0	0	0	0	313	18	162	37	17	0	0	0	0
94	0	0	41	0	0	0	0	0	0	0	0	0	0	0	0	303	9	145	52	30	0	0	0	0
95	0	0	26	0	0	0	0	0	0	0	0	0	0	0	0	300	8	130	54	38	0	0	0	0
96	0	0	22	0	0	0	0	0	0	1	0	0	0	0	0	293	5	119	64	47	1	0	0	0
97	0	0	17	0	0	0	0	0	0	0	0	0	0	0	0	285	4	100	78	56	2	0	0	0
98	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	276	3	88	79	64	10	0	0	0
99	0	0	12	0	0	0	0	0	0	0	0	0	0	0	0	269	0	78	56	56	85	0	0	0
100	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	266	0	73	40	51	116	0	0	0
101	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	266	1	63	33	55	49	0	0	0
102	0	0	7	0	0	0	0	0	0	0	0	0	0	0	0	259	0	56	23	64	25	0	0	0
103	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	256	2	50	21	56	29	0	0	0
104	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	258	1	49	20	49	29	0	0	0
105	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	255	1	47	18	43	27	0	0	0
106	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	253	1	44	14	37	15	0	0	0
107	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	252	1	40	12	36	11	0	0	0
108	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	255	2	38	12	34	8	0	0	0
109	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	255	3	34	11	33	8	0	0	0
110	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	256	2	35	10	30	8	0	0	0
111	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	258	2	32	10	31	7	0	0	0
112	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	262	2	31	11	31	7	0	0	0
113	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	263	0	31	10	29	1	0	0	0
114	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	263	0	31	7	30	1	0	0	0
115	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	261	0	30	5	28	1	0	0	0
116	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	263	0	27	6	29	1	0	0	0
117	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	263	0	28	4	27	1	0	0	0
118	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	266	1	28	3	25	1	0	0	0
119	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	264	1	28	3	24	1	0	0	0
120	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	261	1	29	3	21	1	0	0	0
121	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	268	1	28	2	20	1	0	0	0
122	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	274	1	28	3	20	1	2	0	0

Evolution and Proliferation of Primes

Iter	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20	P21	P22	P23	P24
123	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	276	1	28	2	19	1	9	0	0
124	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	277	1	27	2	19	0	29	0	0
125	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	279	1	26	2	18	0	48	0	0
126	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	277	1	24	2	15	0	61	0	0
127	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	274	1	24	2	13	0	73	0	0
128	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	275	0	24	2	13	0	79	0	0
129	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	282	0	24	1	12	0	92	1	0
130	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	278	0	24	1	12	0	103	5	0
131	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	275	0	24	0	11	0	109	17	0
132	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	261	0	24	0	11	0	112	37	0
133	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	225	0	22	0	10	0	113	110	0
134	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	182	0	22	0	10	0	114	176	0
135	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	159	0	22	0	10	0	114	209	0
136	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	127	0	22	0	7	0	112	247	0
137	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	105	0	23	0	6	0	109	287	2
138	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	96	0	23	0	6	0	111	299	14
139	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	89	0	23	0	6	0	117	310	45
140	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	80	0	22	0	4	0	115	319	51
141	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	67	0	23	0	4	0	118	335	36
142	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	65	0	23	0	3	0	118	342	19
143	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	55	0	22	0	3	0	116	352	6
144	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	52	0	22	0	3	0	109	359	2
145	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	51	0	23	0	3	0	101	363	2
146	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	50	0	24	0	3	0	93	364	7
147	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	50	0	27	0	3	0	93	369	7
148	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	45	0	26	0	3	0	97	378	8

Selection of 123 Solved Sequences

https://github.com/Anon52MI4/oeis-alien

Table: Samples of the solved sequences.

https://oeis.org/A317485	Number of Hamiltonian paths in the n-Bruhat graph.
https://oeis.org/A349073	a(n) = U(2*n, n), where U(n, x) is the Chebyshev polynomial of the second kind.
https://oeis.org/A293339	Greatest integer k such that $k/2^n < 1/e$.
https://oeis.org/A1848	Crystal ball sequence for 6-dimensional cubic lattice.
https://oeis.org/A8628	Molien series for A_5.
https://oeis.org/A259445	Multiplicative with $a(n) = n$ if n is odd and $a(2^s) = 2$.
https://oeis.org/A314106	Coordination sequence Gal.6.199.4 where G.u.t.v denotes the coordination sequence for a vertex of type v in tiling number t in the Galebach list of u-uniform tilings
https://oeis.org/A311889	Coordination sequence Gal.6.129.2 where G.u.t.v denotes the coordination sequence for a vertex of type v in tiling number t in the Galebach list of u-uniform tilings.
https://oeis.org/A315334	Coordination sequence Gal.6.623.2 where G.u.t.v denotes the coordination sequence for a vertex of type v in tiling number t in the Galebach list of u-uniform tilings.
https://oeis.org/A315742	Coordination sequence Gal.5.302.5 where G.u.t.v denotes the coordination sequence for a vertex of type v in tiling number t in the Galebach list of u-uniform tilings.
https://oeis.org/A004165	OEIS writing backward
https://oeis.org/A83186	Sum of first n primes whose indices are primes.
https://oeis.org/A88176	Primes such that the previous two primes are a twin prime pair.
https://oeis.org/A96282	Sums of successive twin primes of order 2.
https://oeis.org/A53176	Primes p such that $2p + 1$ is composite.
https://oeis.org/A267262	Total number of OFF (white) cells after n iterations of the "Rule 111" elementary cellular automaton starting with a single ON (black) cell.

More topics if time remains

- Guided search vs direct synthesis - the example of infinitude of primes $(n! + 1)$
- Triangle coding and introducing a new virus into our population
- Memorization, targeted vs untargeted invention
- Finding programs by training useful transformations
- Orthogonality of models

Thanks and Advertisement

- Thanks for your attention!
- AITP – Artificial Intelligence and Theorem Proving
- September 1–6, 2024, Aussois, France, `aitp-conference.org`
- ATP/ITP/Math vs AI/Machine-Learning people, Computational linguists
- Discussion-oriented and experimental
- Grown to 80 people in 2019
- Invited talks by people who do AI/ML/TP for math, physics, ...

References to our work mentioned here

- Thibault Gauthier, Josef Urban: Learning Program Synthesis for Integer Sequences from Scratch. AAAI 2023: 7670-7677
- Thibault Gauthier, Miroslav Olsák, Josef Urban: Alien Coding. CoRR abs/2301.11479 (2023)
- Thibault Gauthier, Chad E. Brown, Mikolas Janota, Josef Urban: A Mathematical Benchmark for Inductive Theorem Provers. LPAR 2023: 224-237
- Thibault Gauthier, Cezary Kaliszyk, Josef Urban: TacticToe: Learning to Reason with HOL4 Tactics. LPAR 2017: 125-143
- Thibault Gauthier, Cezary Kaliszyk, Josef Urban, Ramana Kumar, Michael Norrish: Learning to Prove with Tactics. CoRR abs/1804.00596 (2018)
- Jan Jakubuv, Karel Chvalovský, Zarathustra Amadeus Goertzel, Cezary Kaliszyk, Mirek Olsák, Bartosz Piotrowski, Stephan Schulz, Martin Suda, Josef Urban: MizAR 60 for Mizar 50. ITP 2023: 19:1-19:22
- Thibault Gauthier: Deep Reinforcement Learning for Synthesizing Functions in Higher-Order Logic. LPAR 2020: 230-248
- Thibault Gauthier: Tree Neural Networks in HOL4. CICM 2020: 278-283
- Qingxiang Wang, Cezary Kaliszyk, Josef Urban: First Experiments with Neural Translation of Informal to Formal Mathematics. CICM 2018: 255-270
- Bartosz Piotrowski, Josef Urban: Stateful Premise Selection by Recurrent Neural Networks. LPAR 2020: 409-422
- Bartosz Piotrowski, Josef Urban: Guiding Inferences in Connection Tableau by Recurrent Neural Networks. CICM 2020: 309-314
- Josef Urban, Jan Jakubuv: First Neural Conjecturing Datasets and Experiments. CICM 2020: 315-323
- Bartosz Piotrowski, Josef Urban, Chad E. Brown, Cezary Kaliszyk: Can Neural Networks Learn Symbolic Rewriting? CoRR abs/1911.04873 (2019)
- J. Urban. ERC project AI4Reason final scientific report, 2021. `http://grid01.ciirc.cvut.cz/~mptp/ai4reason/PR_CORE_SCIENTIFIC_4.pdf`

Some More Conjecturing References

- Douglas Bruce Lenat. *AM: An Artificial Intelligence Approach to Discovery in Mathematics as Heuristic Search*. PhD thesis, Stanford, 1976.
- Siemion Fajtlowicz. On conjectures of Graffiti. *Annals of Discrete Mathematics*, 72(1–3):113–118, 1988.
- Simon Colton. *Automated Theory Formation in Pure Mathematics*. Distinguished Dissertations. Springer London, 2012.
- Moa Johansson, Dan Rosén, Nicholas Smallbone, and Koen Claessen. Hipster: Integrating theory exploration in a proof assistant. In *CICM 2014*, pages 108–122, 2014.
- Thibault Gauthier, Cezary Kaliszyk, and Josef Urban. Initial experiments with statistical conjecturing over large formal corpora. In *CICM'16 WiP Proceedings*, pages 219–228, 2016.
- Thibault Gauthier, Cezary Kaliszyk: Sharing HOL4 and HOL Light Proof Knowledge. LPAR 2015: 372-386
- Thibault Gauthier. Deep reinforcement learning in HOL4. *CoRR*, abs/1910.11797, 2019.
- Chad E. Brown and Thibault Gauthier. Self-learned formula synthesis in set theory. *CoRR*, abs/1912.01525, 2019.
- Zarathustra Goertzel and Josef Urban. Usefulness of Lemmas via Graph Neural Networks (Extende Abstract). AITP 2019.
- Karel Chvalovský, Thibault Gauthier and Josef Urban: First Experiments with Data Driven Conjecturing (Extended Abstract). AITP 2019.