Some Combinations of Machine Learning and Theorem Proving

Josef Urban

Czech Technical University in Prague

Automated Reasoning Seminar
May 21, 2025, AWS

European Research Council
Established by the European Commission

Outline

Motivation, Learning vs. Reasoning

Brief Overview

Learning Based Guidance of ATPs

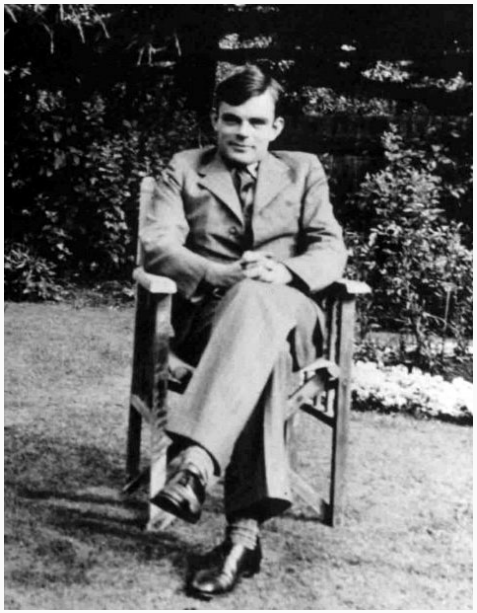
Learning Based Synthesis for OEIS

How Do We Automate Math and Science?

- What is mathematical and scientific thinking/intelligence?
- Pattern-matching, analogy, induction/learning from examples
- Learning both fuzzy hunches and crisp algos/heuristics/procedures
- Deductive reasoning and proving (*btw, computation is reasoning*)
- intelligently guided search, exploration and guessing/conjecturing
- Complicated feedback loops and interplays between all these
- Using a lot of previous knowledge - both for induction and deduction
- Examples from physics (Popper?): deduction pushing us to Relativity, QM

- We need to develop such methods on computers
- Are there any large corpora suitable for nontrivial deduction?
- Yes! Large libraries of formal proofs and theories
- So let's try to develop strong AI on them!
- (In my case done/preached for 25-30 years ;-)

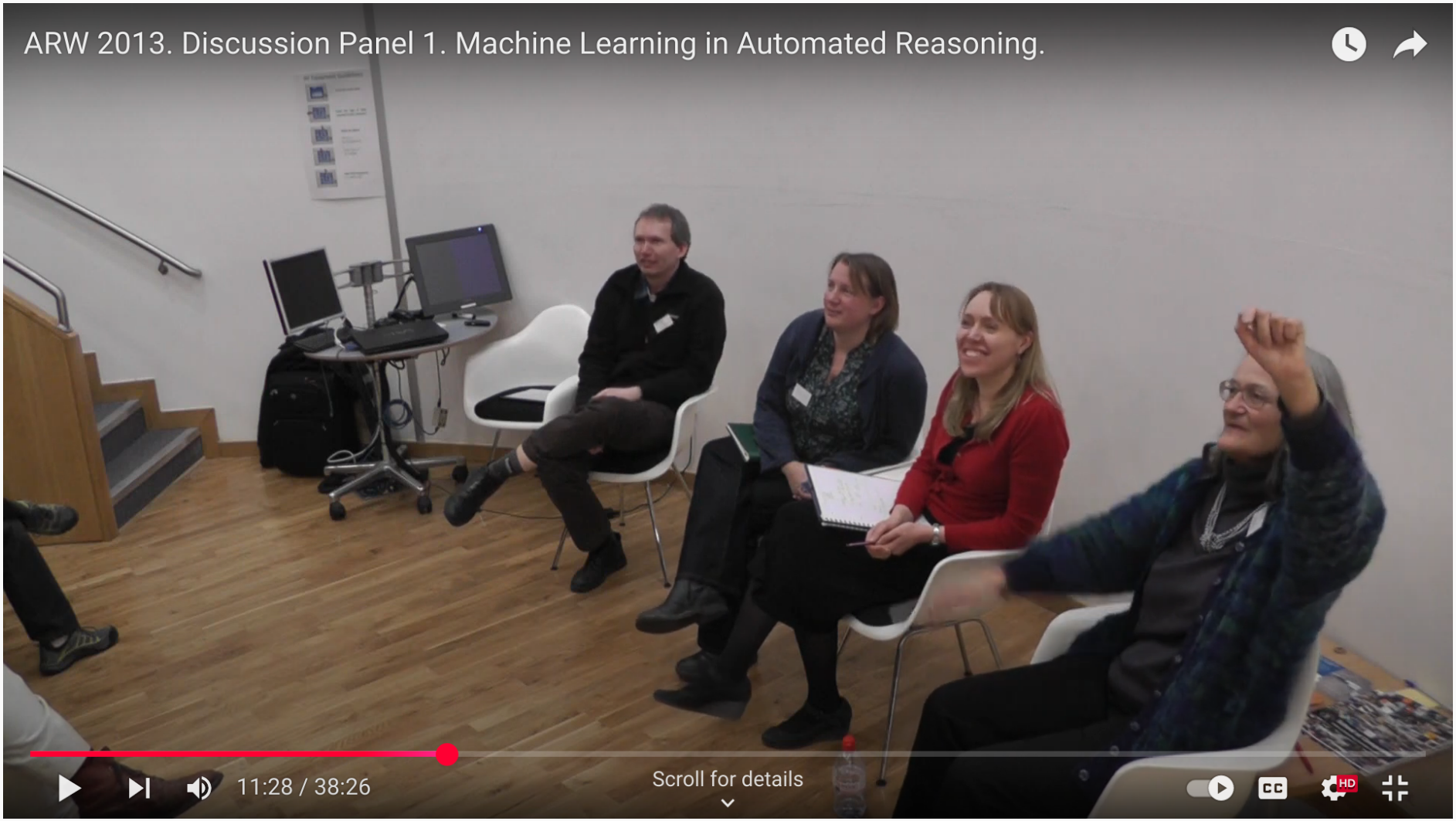
Learning vs Reasoning – Alan Turing 1950 – AI



- 1950: *Computing machinery and intelligence* – AI, Turing test
- *"We may hope that machines will eventually compete with men in all purely intellectual fields."* (regardless of his 1936 undecidability result!)
- last section on Learning Machines:
- *"But which are the best ones [fields] to start [learning on] with?"*
- *"... Even this is a difficult decision. Many people think that a very abstract activity, like the playing of chess, would be best."*
- Why not try with math? It is much more (universally?) expressive ...

Outline

2013 Automated Reasoning Workshop by Katya et al



My ARW13 Talk: 10 Years of AI for Large Formal KBs

AI over Large Formal Knowledge Bases: The First Decade

Josef Urban

ICIS, RU Nijmegen

Abstract: In March 2003, the first version of the Mizar Problems for Theorem Proving (MPTP) was released. In the past ten years, such large formal knowledge bases have started to provide an interesting playground for combining deductive and inductive AI methods. The talk will discuss three related areas of application in which machine learning and general AI have been recently experimented with: (i) premise selection for theorem proving over large formal libraries built with systems like Mizar, HOL Light, and Isabelle (ii) advising and tuning first-order automated theorem provers such as E and leanCoP, and (iii) building larger inductive/deductive AI systems such as MaLARea. Here I focus on the wider motivation for this work.

Why AI over Large Formal Knowledge Bases

The Three Semantic Dreams

re are three powerful and old dreams that came with the
ntion of computers by Turing, von Neuman, and oth-

It is not obvious that the two abilities imply one anc
For example, in ITPs like Mizar, Isabelle, HOL (L
and Coq, large corpora of mathematics are in some s
very well understood (all concepts are formally def
all proofs explained and verified), but the reasoning p

ARW'13: 10 Years of AI for Large Formal KBs

- Large AITP corpora: Mizar/MML, Isabelle/AFP, HOL/Flyspeck
  - Tens of thousands to millions of problems; easy to generate zillions
  - *Rigorous evaluation settings* (MPTP Challenge, etc)
- Fast ML-based premise-selection methods (and hammers)
  - 40% of Flyspeck and Mizar hammered around 2013
  - Sledgehammer – a well integrated workhorse (*generative AI!*)
  - Embeddings, semantic features, lemmatization, etc
  - However mostly *name-dependent methods* (weaker transfer)
- Symbolic and semantic methods for guiding ATPs
  - advanced ATP calculi (symmetry breaking, splitting/Avatar), Veroff's hints
  - instantiation-based methods (SMTs, iProver)
  - btw, *SAT/CDCL has been a superpower long before DL*
- Good strategy invention methods
  - Basically *automated algorithm design (another superpower)*
  - ParamILS, Spider, MaLeS, BliStr
- Feedback loops between learning and reasoning
  - MaLARea, PS-E, BliStr
- Human-assisted theory exploration (Veroff, etc)

ARW'13: What We Didn't Have in 2013

- *Efficient* learning-based inference guiding machines for ATP
  - Started to work in 2018 for simpler (connection) calculi (rlCoP etc)
  - Since 2019 for state-of-the-art ATP calculi/systems (ENIGMA, Deepire)
  - Recently also for instantiation-based systems (iProver, SMTs)
- Strong learning-guided tactical provers
  - TacticToe in 2017-18 – 66% of HOL4 in fair evaluation
- Good conjecturing for arbitrary large formal math
  - Various (not just neural) attempts, still hard
- Good synthesis of math objects in general (some recent examples)
- Learning-assisted autoformalization (AF) for math
  - PCFG-based AF methods since 2014/15
  - Neural (LM) based AF methods since 2018 – quite encouraging
- More fun with feedback loops between (proof) search and learning
- Efficient name-invariant ML methods
- How much of the progress was due to DL/(L)LM?
- What were the other interesting AI/AR/ML/... approaches/combinations?
- A detailed ERC report from 2021:
  `http://ai4reason.org/PR_CORE_SCIENTIFIC_4.pdf`

AITP Challenges/Bets from 2014

- 3 AITP bets for 10k EUR from my 2014 talk at Institut Henri Poincare (`tinyurl.com/yb55b3jv`)
  - In 20 years, 80% of Mizar and Flyspeck toplevel theorems will be provable automatically (same hardware, same libraries as in 2014 - about 40% then)
  - In 10 years: 60% (DONE already in 2021 - 3 years ahead of schedule)
  - In 25 years, 50% of the toplevel statements in LaTeX-written Msc-level math curriculum textbooks will be parsed automatically and with correct formal semantics (this may be faster than I expected)
- My (conservative?) estimate when we will do Fermat:
  - Human-assisted formalization: by 2050
  - Fully automated proof (hard to define precisely): by 2070
  - See the Foundation of Math thread: `https://tinyurl.com/59c89sme`
  - and the AITP'22 panel: `https://bit.ly/3dcY5HW`

Outline

Statistical Guidance of Connection Tableau

- learn guidance of every clausal inference in connection tableau (leanCoP)
- set of first-order clauses, *extension* and *reduction* steps
- proof finished when all branches are closed
- a lot of nondeterminism, requires backtracking
- *Iterative deepening* used in leanCoP to ensure completeness
- good for learning – the tableau compactly represents the proof state

Clauses:

$c_1 : P(x)$

$c_2 : R(x,y) \vee \neg P(x) \vee Q(y)$

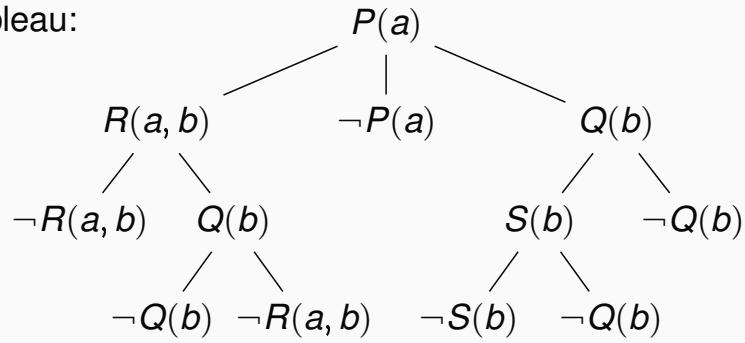
$c_3 : S(x) \vee \neg Q(b)$

$c_4 : \neg S(x) \vee \neg Q(x)$

$c_5 : \neg Q(x) \vee \neg R(a,x)$

$c_6 : \neg R(a,x) \vee Q(x)$

Closed Connection Tableau:



leanCoP: Minimal Prolog FOL Theorem Prover

```
%  prove(Cla,Path,PathLim,Lem,Set)
prove([Lit|Cla],Path,PathLim,Lem,Set) :-
          (-NegLit=Lit;-Lit=NegLit) ->
          (
            member(NegL,Path),
            unify_with_occurs_check(NegL,NegLit)
          ;
            % main nondeterminism
            lit(NegLit,NegL,Cla1,Grnd1),
            unify_with_occurs_check(NegL,NegLit),
            prove(Cla1,[Lit|Path],PathLim,Lem,Set)
          ),
          prove(Cla,Path,PathLim,Lem,Set).
prove([],_,_,_,_).
```

Statistical Guidance of Connection Tableau – rlCoP

- 2018: strong learners via C interface to OCAML (boosted trees)
- remove iterative deepening, the prover can go arbitrarily deep
- added Monte-Carlo Tree Search (MCTS) (inspired by AlphaGo/Zero)
- MCTS search nodes are sequences of clause application
- a good heuristic to explore new vs exploit good nodes:

$$UCT(i) = \frac{w_i}{n_i} + c \cdot p_i \cdot \sqrt{\frac{\ln N}{n_i}} \qquad \text{(UCT - Kocsis, Szepesvari 2006)}$$

- learning both policy (p) (clause selection) and value (w) (state evaluation)
- clauses represented not by names but also by features (generalize!)
- binary learning setting used: | proof state | clause features |
- mostly term walks of length 3 (trigrams), hashed into small integers
- many iterations of proving and learning
- More recently also with GNNs (Olsak, Rawson, Zombori, ...)

Statistical Guidance of Connection Tableau – rlCoP

- On 32k Mizar40 problems using 200k inference limit
- nonlearning CoPs:

System	leanCoP	bare prover	rlCoP no policy/value (UCT only)
Training problems proved	10438	4184	7348
Testing problems proved	**1143**	431	804
Total problems proved	11581	4615	8152

- rlCoP with policy/value after 5 proving/learning iters on the training data
- $1624/1143 = 42.1\%$ improvement over leanCoP on the testing problems

Iteration	1	2	3	4	5	6	7	8
Training proved	12325	13749	14155	14363	14403	14431	14342	**14498**
Testing proved	1354	1519	1566	1595	**1624**	1586	1582	1591

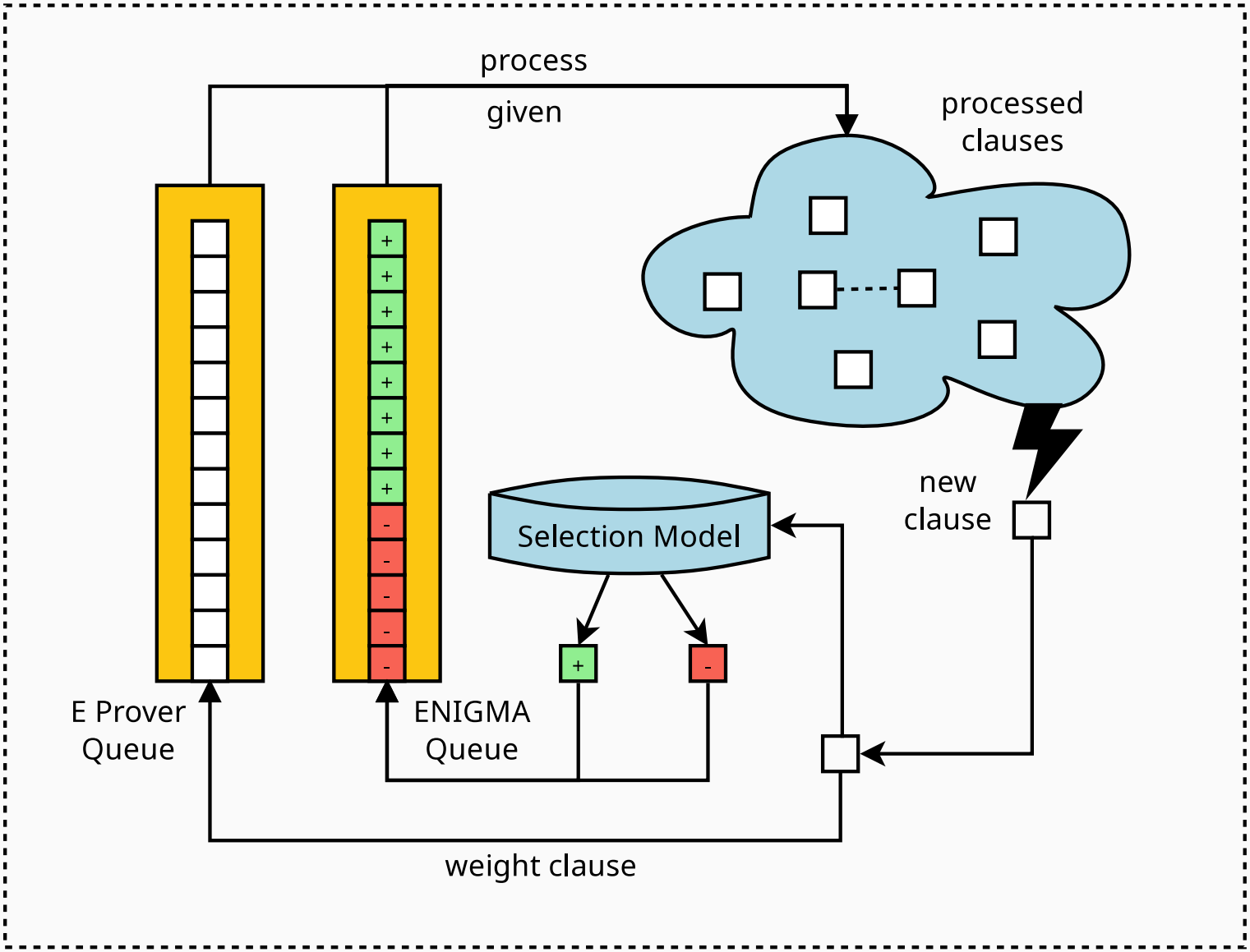
ENIGMA (2017): Guiding the Best ATPs like E Prover

Basic Saturation Loop – Given Clause Loop (E, Vampire, SPASS, Prover9, ...)

$P := \emptyset$ *(processed)*
$U := \{$*clausified axioms and a negated conjecture*$\}$ *(unprocessed)*
while ($U \neq \emptyset$) do
  if ($\bot \in U \cup P$) then return *Unsatisfiable*
  $g :=$ select(U) *(choose a given clause)*
  $P := P \cup \{g\}$ *(add to processed)*
  $U := U \backslash \{g\}$ *(remove from unprocessed)*
  $U := U \cup \{$*all clauses inferred from g and P*$\}$ *(add inferences)*
done
return *Satisfiable*

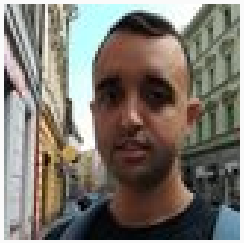
Typically, U grows quadratically wrt. P
1M clauses in U in 10s common – choosing good g gets hard – use ML!

ENIGMA: ML-based Given Clause Guidance



ENIGMA (2017): Guiding the Best ATPs like E Prover

- ENIGMA (Jan Jakubuv, Zar Goertzel, Karel Chvalovsky, others)


- The proof state are two large heaps of clauses *processed*/*unprocessed*
- learn on E's proof search traces, put classifier in E
- positive examples: clauses (lemmas) used in the proof
- negative examples: clauses (lemmas) not used in the proof
- 2021 multi-phase architecture (combination of different methods):
  - fast gradient-boosted decision trees (GBDTs) used in 2 ways
  - fast logic-aware graph neural network (GNN - Olsak) run on a GPU server
  - logic-based subsumption using fast indexing (discrimination trees - Schulz)
- The GNN scores many clauses (context/query) together in a large graph
- Sparse - vastly more efficient than transformers ($\sim$100k symbols)
- 2021: leapfrogging and Split&Merge:
- aiming at learning reasoning/algo components

3-phase Anonymous ENIGMA

The 3-phase ENIGMA (single strategy) solves in 30s 56.4% of Mizar (bushy)

Given Clause Loop in E + ML Guidance

Contribution 4

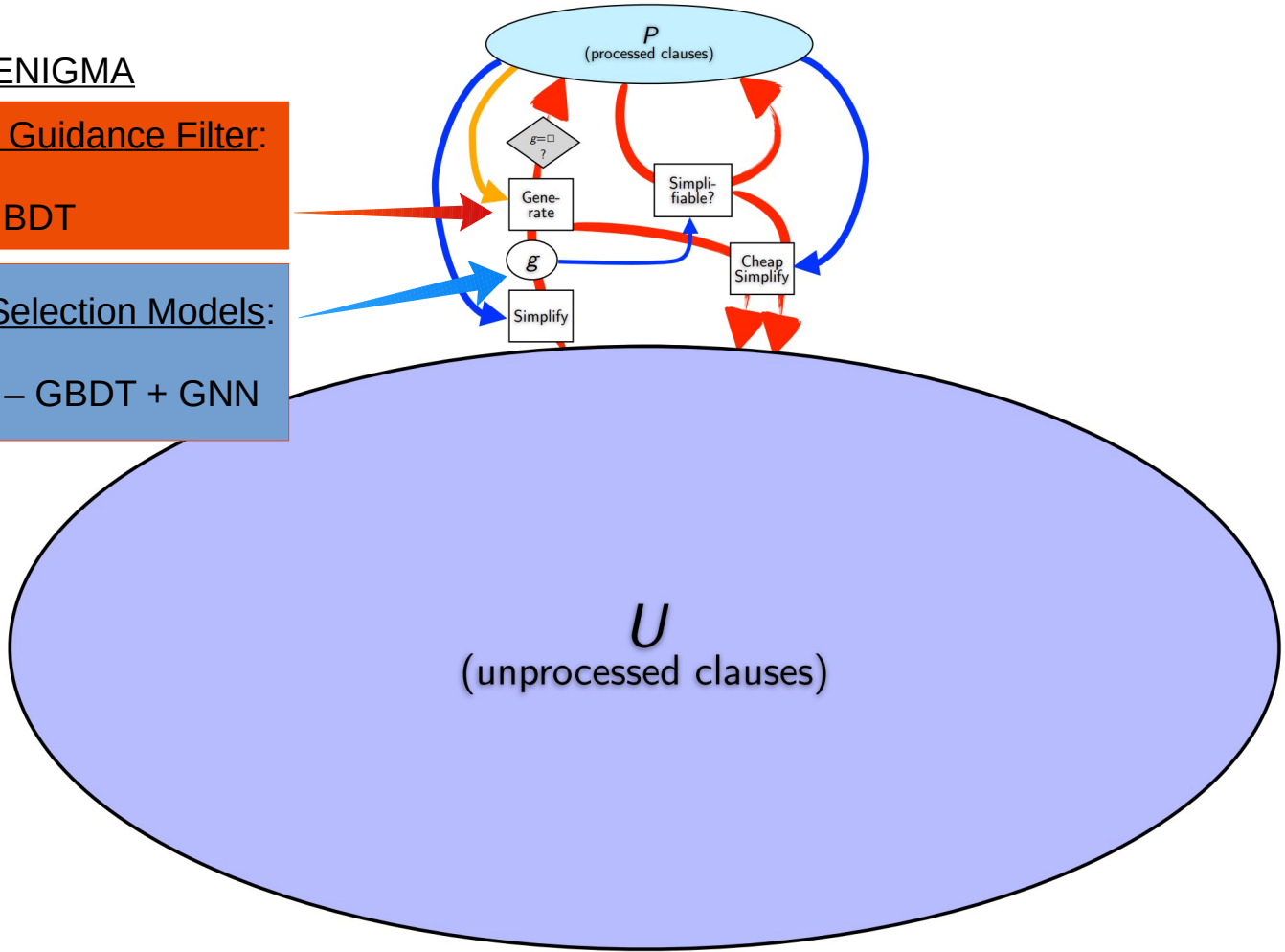
3-phase ENIGMA

Parental Guidance Filter:

Fast – GBDT

Clause Selection Models:

2-phase – GBDT + GNN



Prove/Learn ENIGMA feedback loop on formal math

- Done on 57880 Mizar Mathematical Library formal math problems in 2019
- Efficient ML-guidance inside the best ATPs like E prover (ENIGMA)
- *Training* of the ML-guidance is *interleaved* with *proving* harder problems
- Ultimately a 70% improvement over the original E strategy:
- ... from 14933 proofs to 25397 proofs (all in 10s CPU - no cheating)

	$\mathcal{S}$	$\mathcal{S} \odot \mathcal{M}_9^0$	$\mathcal{S} \oplus \mathcal{M}_9^0$	$\mathcal{S} \odot \mathcal{M}_9^1$	$\mathcal{S} \oplus \mathcal{M}_9^1$	$\mathcal{S} \odot \mathcal{M}_9^2$	$\mathcal{S} \oplus \mathcal{M}_9^2$	$\mathcal{S} \odot \mathcal{M}_9^3$	$\mathcal{S} \oplus \mathcal{M}_9^3$
solved	**14933**	16574	20366	21564	22839	22413	23467	22910	23753
$\mathcal{S}$%	+0%	+10.5%	+35.8%	+43.8%	+52.3%	+49.4%	+56.5%	+52.8%	+58.4
$\mathcal{S}+$	+0	+4364	+6215	+7774	+8414	+8407	+8964	+8822	+9274
$\mathcal{S}-$	-0	-2723	-782	-1143	-508	-927	-430	-845	-454

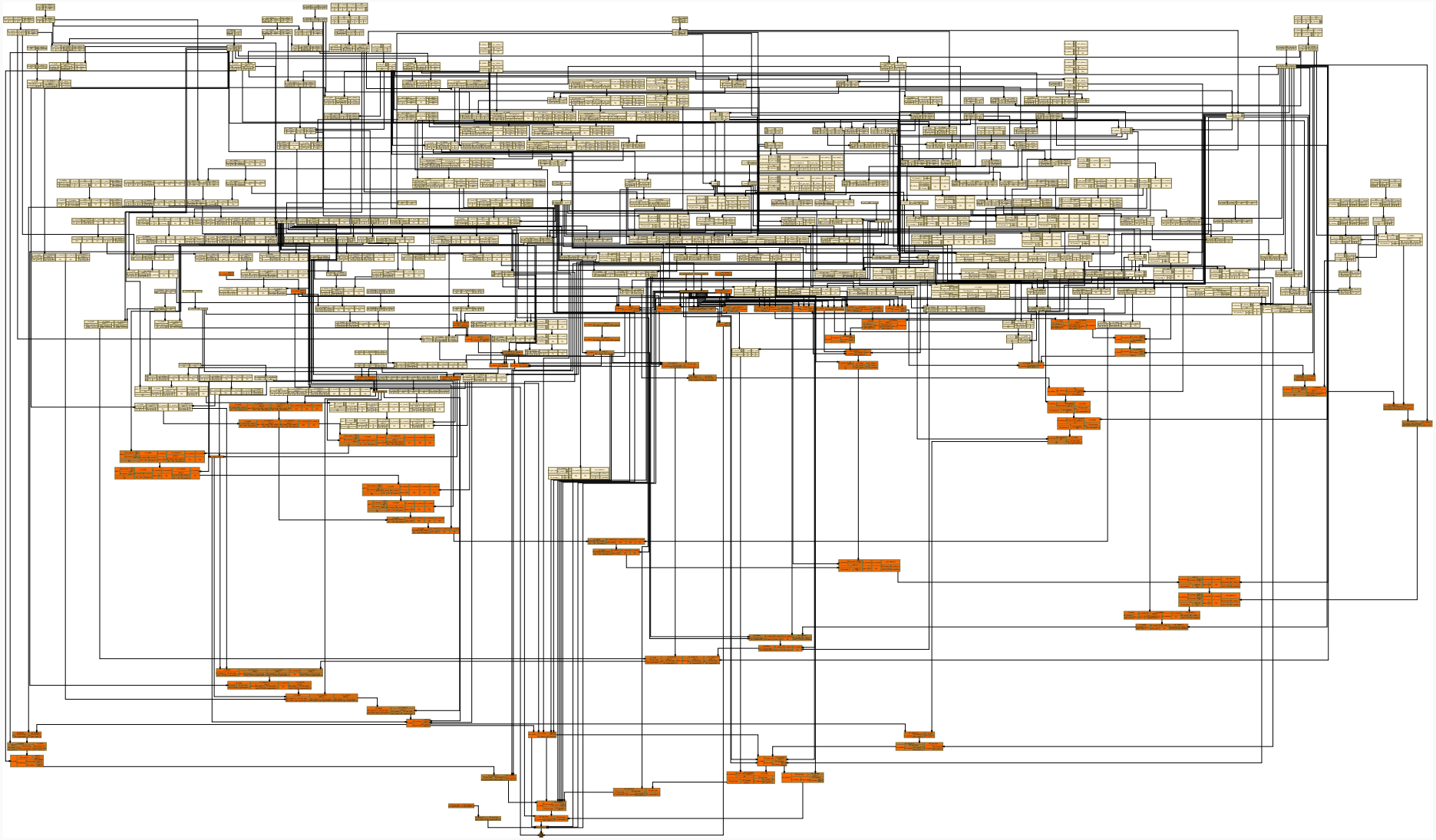
	$\mathcal{S} \odot \mathcal{M}_{12}^3$	$\mathcal{S} \oplus \mathcal{M}_{12}^3$	$\mathcal{S} \odot \mathcal{M}_{16}^3$	$\mathcal{S} \oplus \mathcal{M}_{16}^3$
solved	24159	24701	25100	**25397**
$\mathcal{S}$%	+61.1%	+64.8%	+68.0%	**+70.0%**
$\mathcal{S}+$	+9761	+10063	+10476	+10647
$\mathcal{S}-$	-535	-295	-309	-183

- 75% of the Mizar corpus (43414) reached in July 2021 - higher times and many prove/learn cycles: `https://github.com/ai4reason/ATP_Proofs`
- Details in our Mizar60 paper: `https://arxiv.org/abs/2303.06686`

ENIGMA Anonymous: Learning from patterns only

- The GNN and GBDTs only learn from formula structure, not symbols
- Not from symbols like $+$ and $*$ as Transformer & Co.
- E.g., learning on additive groups thus transfers to multiplicative groups
- **Evaluation** of old-Mizar ENIGMA on 242 new Mizar articles:
- 13370 new theorems, $>$ 50% of them with new terminology:
- The 3-phase ENIGMA is **58%** better on them than unguided E
- While **53.5%** on the old Mizar (where this ENIGMA was trained)
- Generalizing, analogizing and transfer abilities unusual in the large transformer models

Can you do this in 4 minutes? (359-step ATP proof)



Can you do this in 4 minutes? (human-written code)

```
theorem Th31: :: BORSUK_5:31
 for A being Subset of R^1
 for a, b being real number st a < b & A = RAT (a,b) holds
 Cl A = [.a,b.]
proof
 let A be Subset of R^1; :: thesis:
 let a, b be real number ; :: thesis:
 assume that
 A1: a < b and
 A2: A = RAT (a,b) ; :: thesis:
 reconsider ab = ].a,b.[, RT = RAT as Subset of R^1 by NUMBERS:12, TOPMETR:17;
 reconsider RR = RAT /\ ].a,b.[ as Subset of R^1 by TOPMETR:17;
 A3: the carrier of R^1 /\ (Cl ab) = Cl ab by XBOOLE_1:28;
 A4: Cl RR c= (Cl RT) /\ (Cl ab) by PRE_TOPC:21;
 thus Cl A c= [.a,b.] :: according to XBOOLE_0:def 10 :: thesis:
 proof
  let x be set ; :: according to TARSKI:def 3 :: thesis:
  assume x in Cl A ; :: thesis:
  then x in (Cl RT) /\ (Cl ab) by A2, A4;
  then x in the carrier of R^1 /\ (Cl ab) by Th15;
  hence x in [.a,b.] by A1, A3, Th16; :: thesis:
 end;
 thus [.a,b.] c= Cl A :: thesis:
 proof
  let x be set ; :: according to TARSKI:def 3 :: thesis:
  assume A5: x in [.a,b.] ; :: thesis:
  then reconsider p = x as Element of RealSpace by METRIC_1:def 13;
  A6: a <= p by A5, XXREAL_1:1;
  A7: p <= b by A5, XXREAL_1:1;
  per cases by A7, XXREAL_0:1;
   suppose A8: p < b ; :: thesis:
    now :: thesis:
     let r be real number ; :: thesis:
     reconsider pp = p + r as Element of RealSpace by METRIC_1:def 13, XREAL_0:def 1;
     set pr = min (pp,((p + b) / 2));
     A9: min (pp,((p + b) / 2)) <= (p + b) / 2 by XXREAL_0:17;
     assume A10: r > 0 ; :: thesis:
     p < min (pp,((p + b) / 2))
     proof
      per cases by XXREAL_0:15;
       suppose min (pp,((p + b) / 2)) = pp ; :: thesis:
        hence p < min (pp,((p + b) / 2)) by A10, XREAL_1:29; :: thesis:
       end;
       suppose min (pp,((p + b) / 2)) = (p + b) / 2 ; :: thesis:
        hence p < min (pp,((p + b) / 2)) by A8, XREAL_1:226; :: thesis:
       end;
      end;
     end;
     then consider Q being rational number such that
     A11: p < Q and
     A12: Q < min (pp,((p + b) / 2)) by RAT_1:7;
     (p + b) / 2 < b by A8, XREAL_1:226;
     then min (pp,((p + b) / 2)) < b by A9, XXREAL_0:2;
     then A13: Q < b by A12, XXREAL_0:2;
     min (pp,((p + b) / 2)) <= pp by XXREAL_0:17;
     then A14: (min (pp,((p + b) / 2))) - p <= pp - p by XREAL_1:9;
     reconsider P = Q as Element of RealSpace by METRIC_1:def 13, XREAL_0:def 1;
     P - p < (min (pp,((p + b) / 2))) - p by A12, XREAL_1:9;
     then P - p < r by A14, XXREAL_0:2;
     then dist (p,P) < r by A11, Th14;
     then A15: P in Ball (p,r) by METRIC_1:11;
     a < Q by A6, A11, XXREAL_0:2;
     then A16: Q in ].a,b.[ by A13, XXREAL_1:4;
     Q in RAT by RAT_1:def 2;
     then Q in A by A2, A16, XBOOLE_0:def 4;
     hence Ball (p,r) meets A by A15, XBOOLE_0:3; :: thesis:
    end;
    hence x in Cl A by GOBOARD6:92, TOPMETR:def 6; :: thesis:
```

More Low-Level Guidance of Various Creatures

- Neural (TNN) clause selection in Vampire (Deepire - M. Suda):
  Learn from clause *derivation trees only*
  Not looking at what it says, just who its ancestors were.
- Fast and surprisingly good: Extreme Deepire/AVATAR proof of
  $\epsilon_0 = \omega^{\omega^{\omega^{\cdot^{\cdot^{\cdot}}}}}$ `https://bit.ly/3Ne4WNX`
- 1193-long proof takes *about the same resources as one GPT-3/4 reply*
- GNN-based guidance in iProver (Chvalovsky, Korovin, Piepenbrock)
- New (*dynamic data*) way of training
- Led to doubled real-time performance of iProver's instantiation mode
- CVC5: neural & GBDT instantiation guidance (Piepenbrock, Jakubuv)
- very recently 20% improvement on Mizar
- Hints method for Otter/Prover9 (Veroff):
- boost inferences on clauses that match a lemma used in a related proof
- 100k-step long proofs in the AIM project (2021)
- symbolic ML - can be combined with statistical - proof completion vectors

Behold: 1-CPU vampiric “GenAI” proving $\epsilon_0 = \omega^{\omega^{\omega^{\cdot^{\cdot^{\cdot}}}}}$

```
% Refutation found. Thanks to Tanya!
% SZS status Theorem for t36_ordinal5
% SZS output start Proof for t36_ordinal5
fof(f2863755,plain,( $false), inference(avatar_sat_refutation,
[... 2500 lines of proof ...]
% SZS output end Proof for t36_ordinal5
% ------------------------------
% Version: Vampire 4.5.1 (commit 110f4142 on 2020-10-16 16:55:15 +0200)
% Termination reason: Refutation
% Input formulas: 73
% Proof axioms: 49
% Proof steps: 1193
% Main loop iterations started: 38065
% Generated clauses: 2392519
% SAT solver time: 181.936 s
% congruence closure: 167.665 s ( own 156.041 s )
% neural model evaluation: 18.493 s
% other: 503.976 s ( own 26.185 s )
```

EngmaWatch: Symbolic/Statistical Guidance of E

- Bob Veroff's *hints* method used for Prover9
- solve many easier/related problems, produce millions of lemmas
- load the useful lemmas (hints) on the *watchlist* (kind of conjecturing)
- boost inferences on clauses that subsume a watchlist clause
- watchlist parts are fast thinking, bridged by standard (slow) search
- symbolic guidance, initial attempts to choose good hints by statistical ML
- Very long proofs of open conjectures in quasigroup/loop theory (AIM)
- **ProofWatch** (Goertzel et al. 2018): load many proofs separately in E
- dynamically boost those that have been covered more
- needed for heterogeneous ITP libraries
- statistical: watchlists chosen using similarity and usefulness
- semantic/deductive: dynamic guidance based on exact proof matching
- results in better vectorial characterization of saturation proof searches
- Use the *proof completion ratios* as features for characterizing proof state
- Instead of just *static* conjecture features - the proof vectors evolve
- **EnigmaWatch**: Feed them to ML systems too (much more *semantic*)

ENIGMA - The Rise of Computronium (2021)

From: Josef Urban <josef.urban@gmail.com>, Date: Jul 26, 2021 at 9:47
Subject: ENIGMA - The Rise of Computronium

I am happy to announce that the ENIGMA system of the E lineage, helped by its Deepire Vampiric cousin, has reached today (July 26th, 2021) the landmark of 75% automatically proved Mizar top-level problems.
[..]
Many of the proofs show that ENIGMA has autonomously (without any human-programmed decision procedures, tactics, and/or dataset preparation/tuning) learned how to routinely perform common mathematical algorithmic tasks such as numeric calculation, matrix manipulation, boolean algebra, integration and differentiation, sequences of standard rewriting and normalization operations in various algebraic theories, etc., and combine them with other reasoning tasks needed for completing the fully formal proofs.
[..]
This suggests that learned guidance combined with efficient search may in near future lead to a new fully declarative problem-solving computing/reasoning architectures applicable to arbitrary computing/reasoning problems without any human engineering.
[..]
Combined with autoformalization this could lead us to large deployment of reasoning machines in science, following Leibniz's Calculemus dream.

TacticToe: mid-level ITP Guidance (Gauthier'17,18)

- TTT learns from human and its own tactical HOL4 proofs
- No translation or reconstruction needed - native tactical proofs
- Fully integrated with HOL4 and easy to use
- Similar to rlCoP: policy/value learning for applying tactics in a state
- Demo: `http://grid01.ciirc.cvut.cz/~mptp/tactictoe_demo.ogv`
- However much more technically challenging - a real breakthrough:
  - tactic and goal state recording
  - tactic argument abstraction
  - absolutization of tactic names
  - nontrivial evaluation issues
  - these issues have often more impact than adding better learners
- policy: which tactic/parameters to choose for a current goal?
- value: how likely is this proof state succeed?
- 66% of HOL4 toplevel proofs in 60s (better than a hammer!)
- similar followup work for HOL Light (Google), Coq, Lean, ...

Tactician: Tactical Guidance for Coq (Blaauwbroek'20)

- Tactical guidance of Coq proofs
- Technically very challenging to do right - the Coq internals again nontrivial
- 39.3% on the Coq standard library, 56.7% in a union with CoqHammer (orthogonal)
- Fast approximate hashing for k-NN makes a lot of difference
- Fast re-learning more important than "cooler"/slower learners
- Fully integrated with Coq, should work for any development
- User friendly, installation friendly, integration/maintenance friendly
- Demo: `https://blaauwbroek.eu/papers/cicm2020/demo.mp4`, `https://coq-tactician.github.io/demo.html`
- Took several years, but could become a common tool for Coq formalizers
- Recently GNNs added, a major comparison of k-NN, GNN and LMs (Graph2Tac - `https://arxiv.org/abs/2401.02949`)

Outline

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.

QSynth: *Search/Check/Learn feedback loop* on OEIS

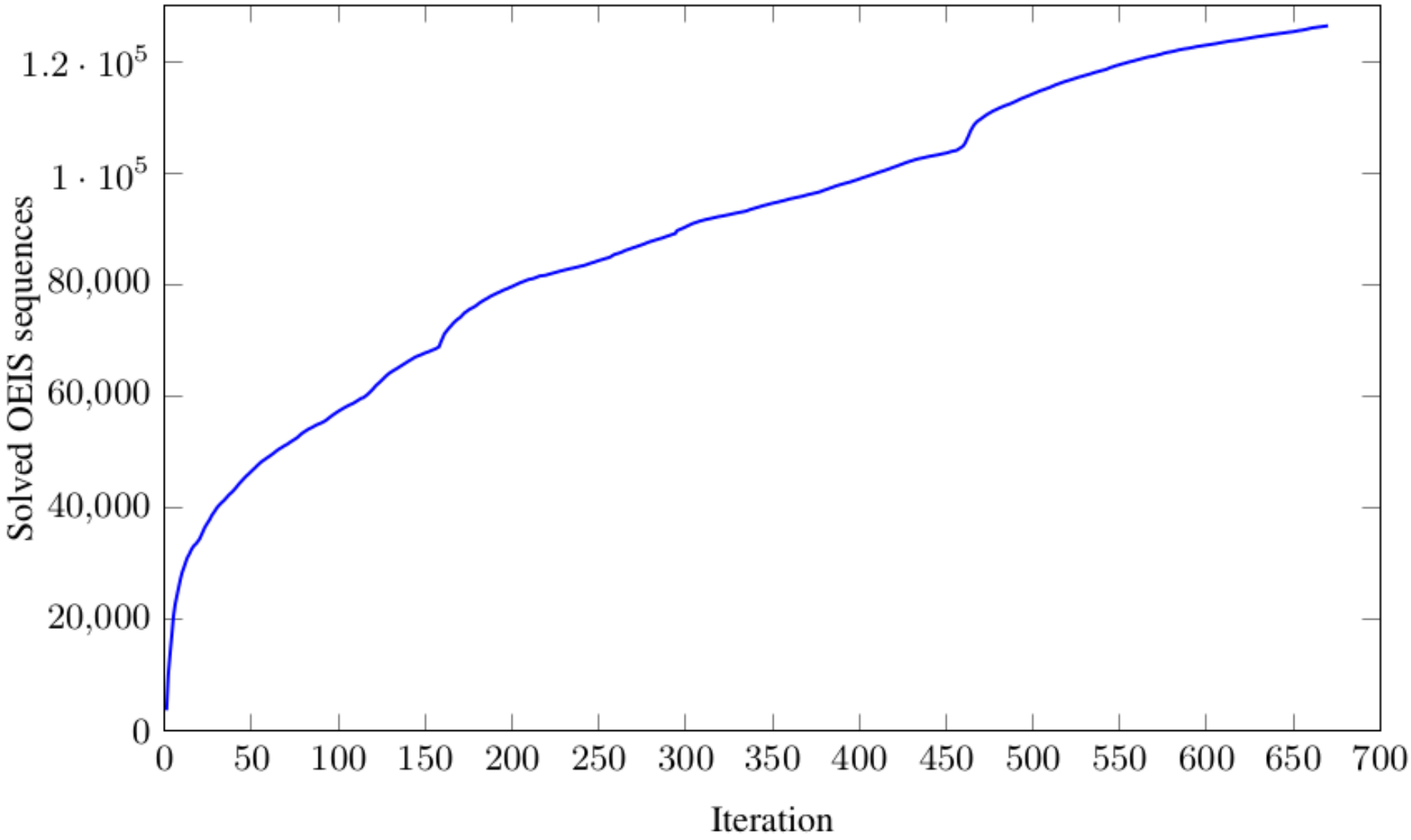


Figure 12: Number y of solved OEIS sequences after x iterations

QSynth TL;DR

- A machine can find explanations for over 125k OEIS sequences
- This is done *from scratch*, without any domain knowledge
- N. Sloane: The OEIS: *A Fingerprint File for Mathematics* (2021)
- About 350k integer sequences in 2021 from all parts of math
- We use a simple Search-Verify-Train positive feedback loop
- Developed by us for combining learning & proving since 2006
- However, one of the most surprising experiments in my life:
- 670 iterations and still refuses to plateau - counters RL wisdom
- Since it interleaves symbolic breakthroughs and statistical learning?
- The electricity bill is only $1k-$3k, you can do this at home
- Btw., we experimentally verify Occam's Razor
- Evolving (self-improving) population of 4.5M matching explanations
- Connections to Solomonoff Induction, AIXI, Gödel Machine?

Motivation: Current AI/TP TODOs/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

QSynt: Semantics-Aware Synthesis of Math Objects

- Gauthier (et al) 2019-24
- 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)
- 126k OEIS sequences (out of 350k) solved so far (670 iterations):
  `https://www.youtube.com/watch?v=24oejR9wsXs`,
  `http://grid01.ciirc.cvut.cz/~thibault/qsynt.html`
- ~4.5M 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)

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

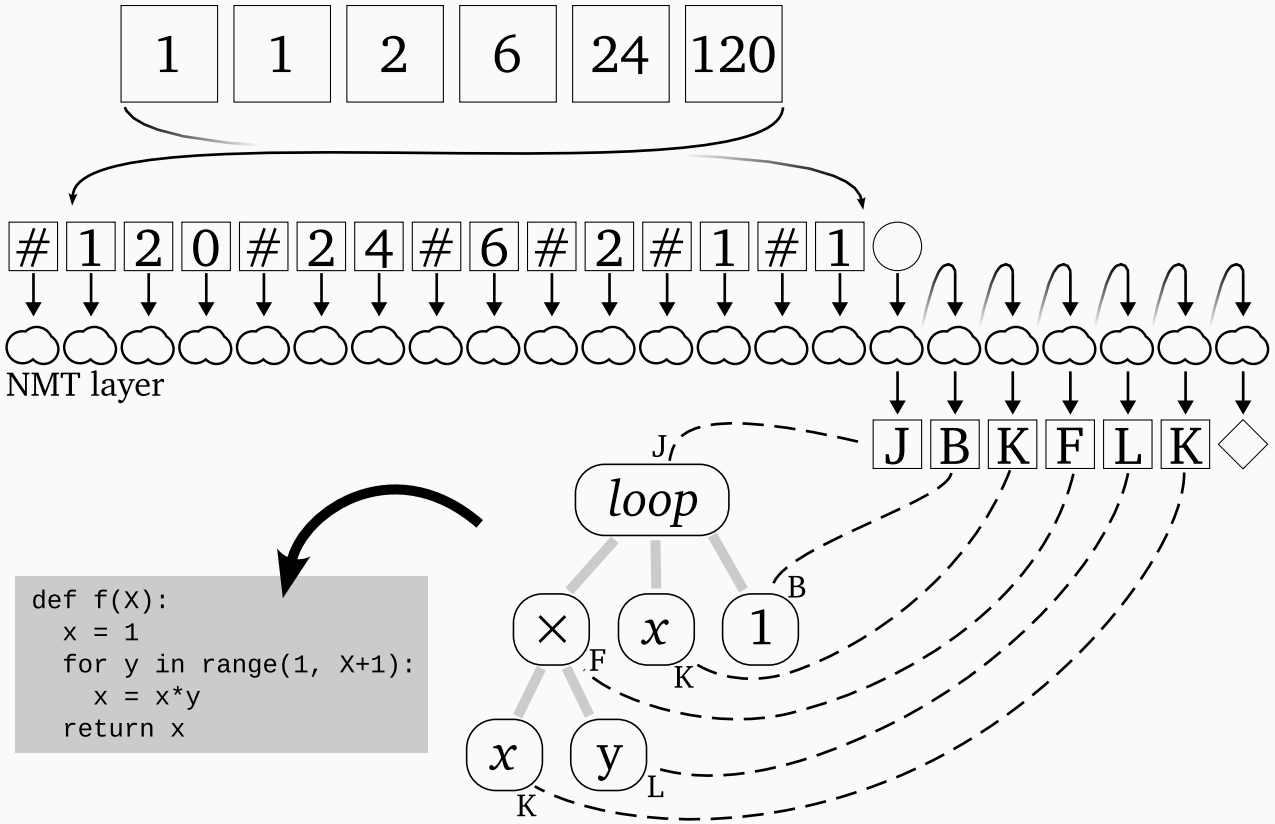
Example:

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

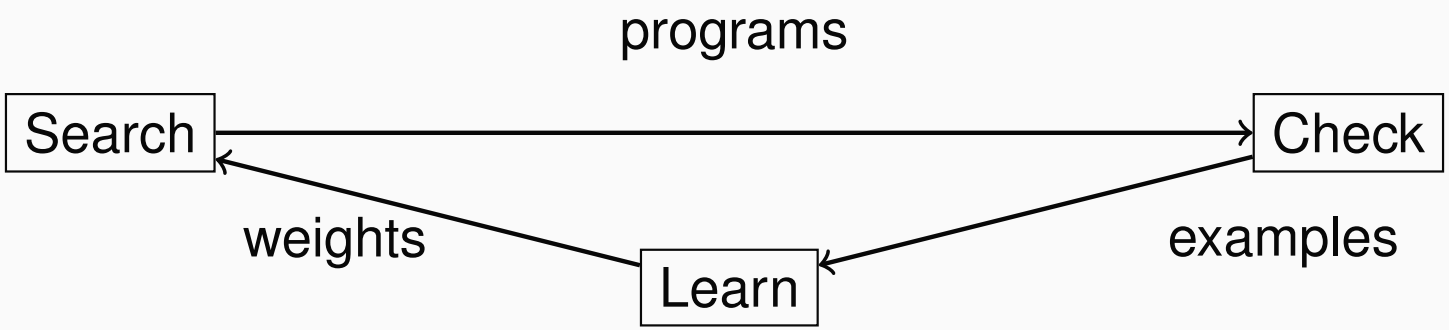
$\mathbf{x!} = \prod_{y=1}^{x} y = \mathit{loop}(y \times x, \mathbf{x}, 1)$

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 Positive Feedback Loop



- Analogous to our Prove/Learn feedback loops in learning-guided proving (since 2006 – Machine Learner for Automated Reasoning – MaLARea))
- However, the OEIS setting allows much faster feedback on *symbolic conjecturing*

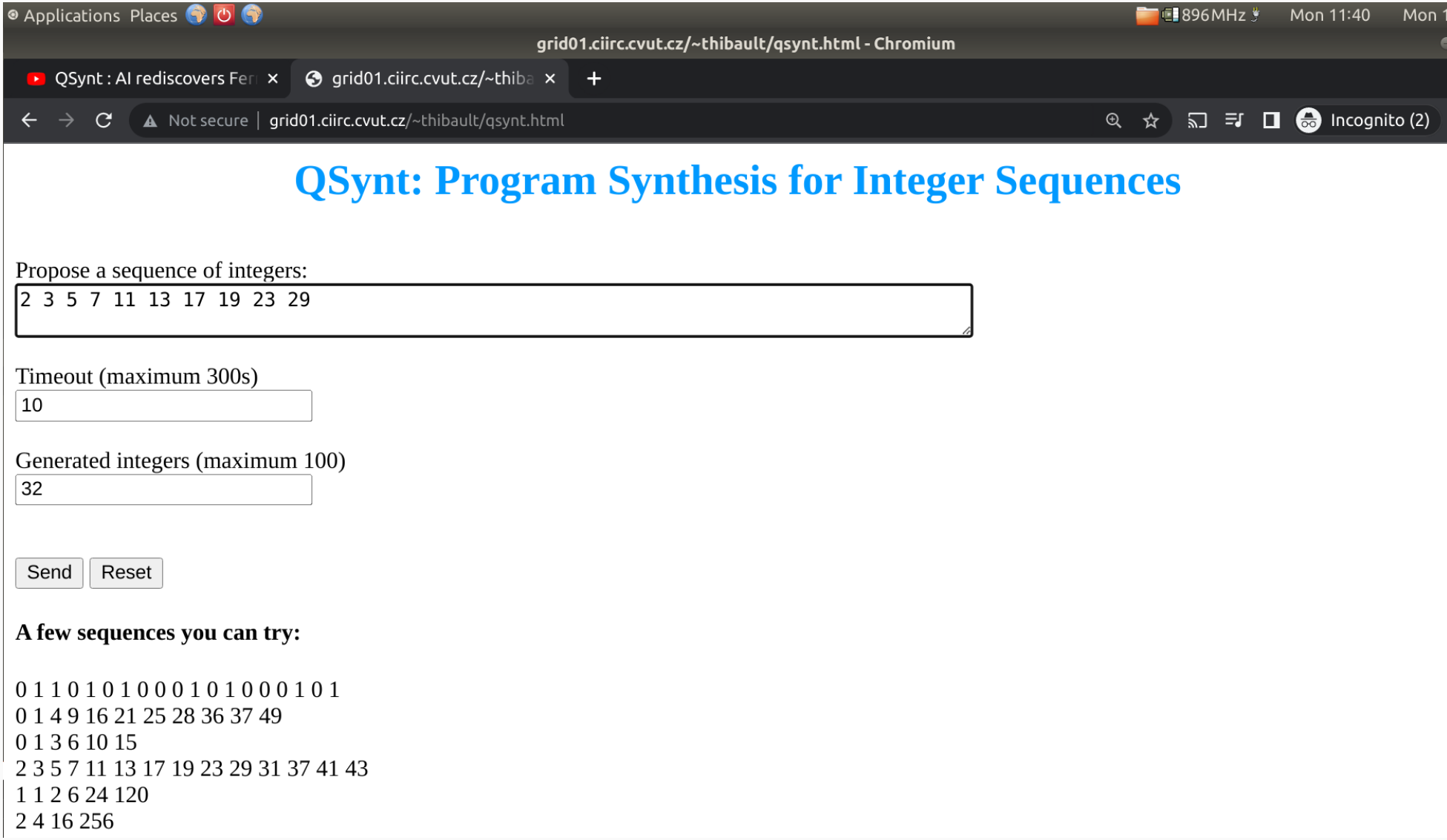
Search-Verify-Train Feedback Loop for OEIS

- **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)
- from iter. 501, we also keep the program with the best speed/length ratio
- **learning phase**: LM trains to translate the "solved" OEIS sequences into the best program(s) generating them
- from iter. 336: train LMs to transform (generalization, optimization)
- our learned version of human-coded methods like ILP and compilation

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:
```

Speed Evolution – Technology Breakthroughs

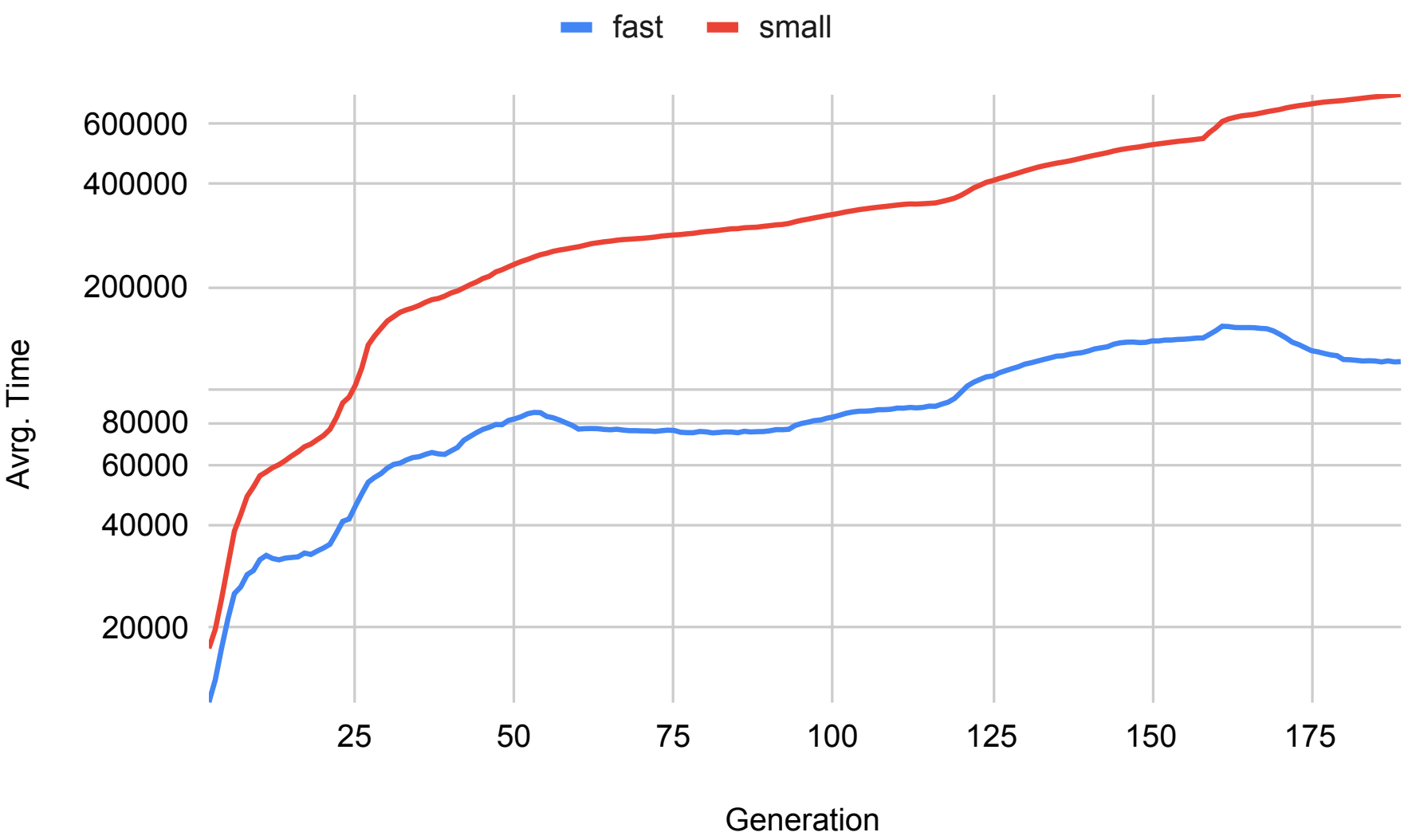
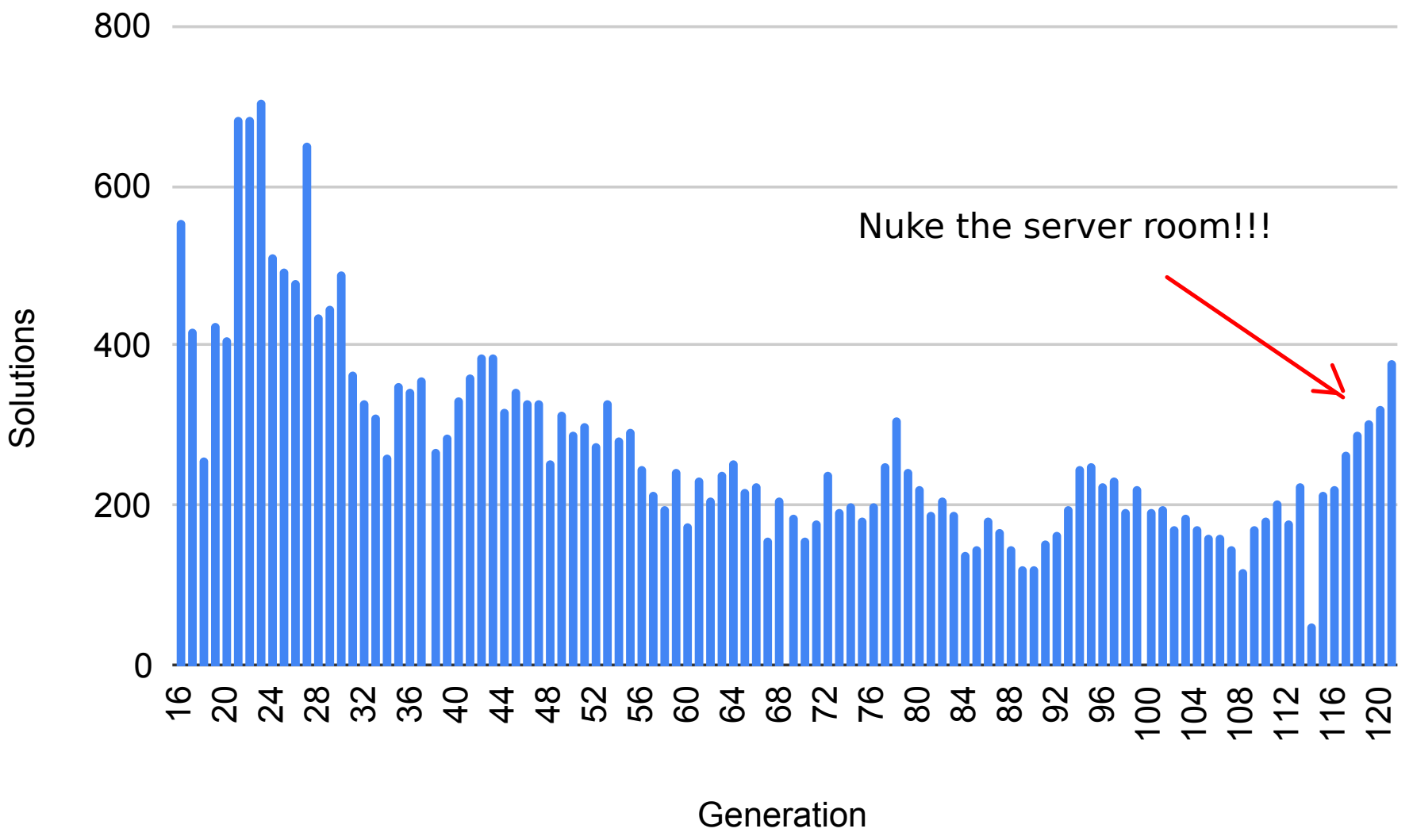
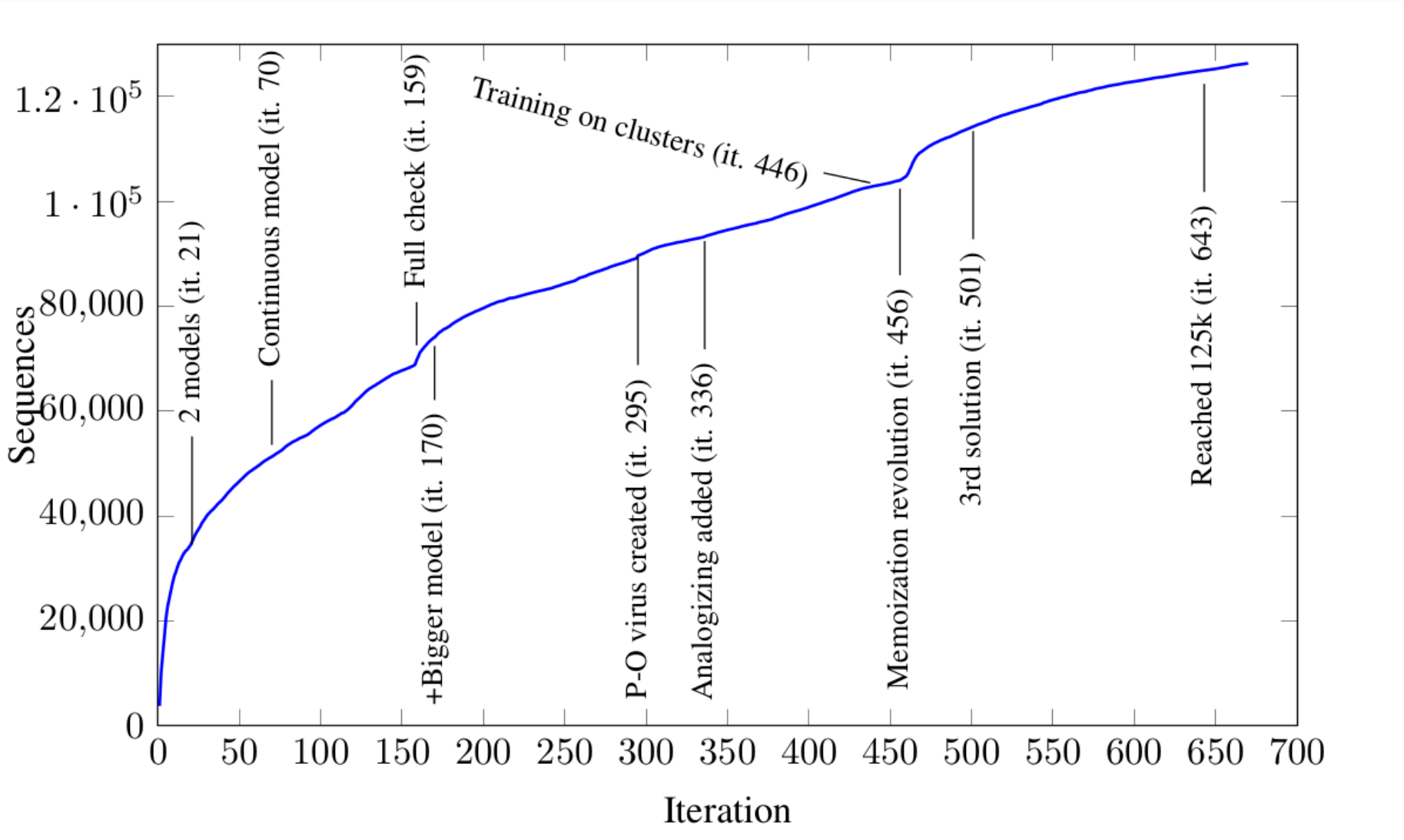


Figure: Avrg. time in iterations

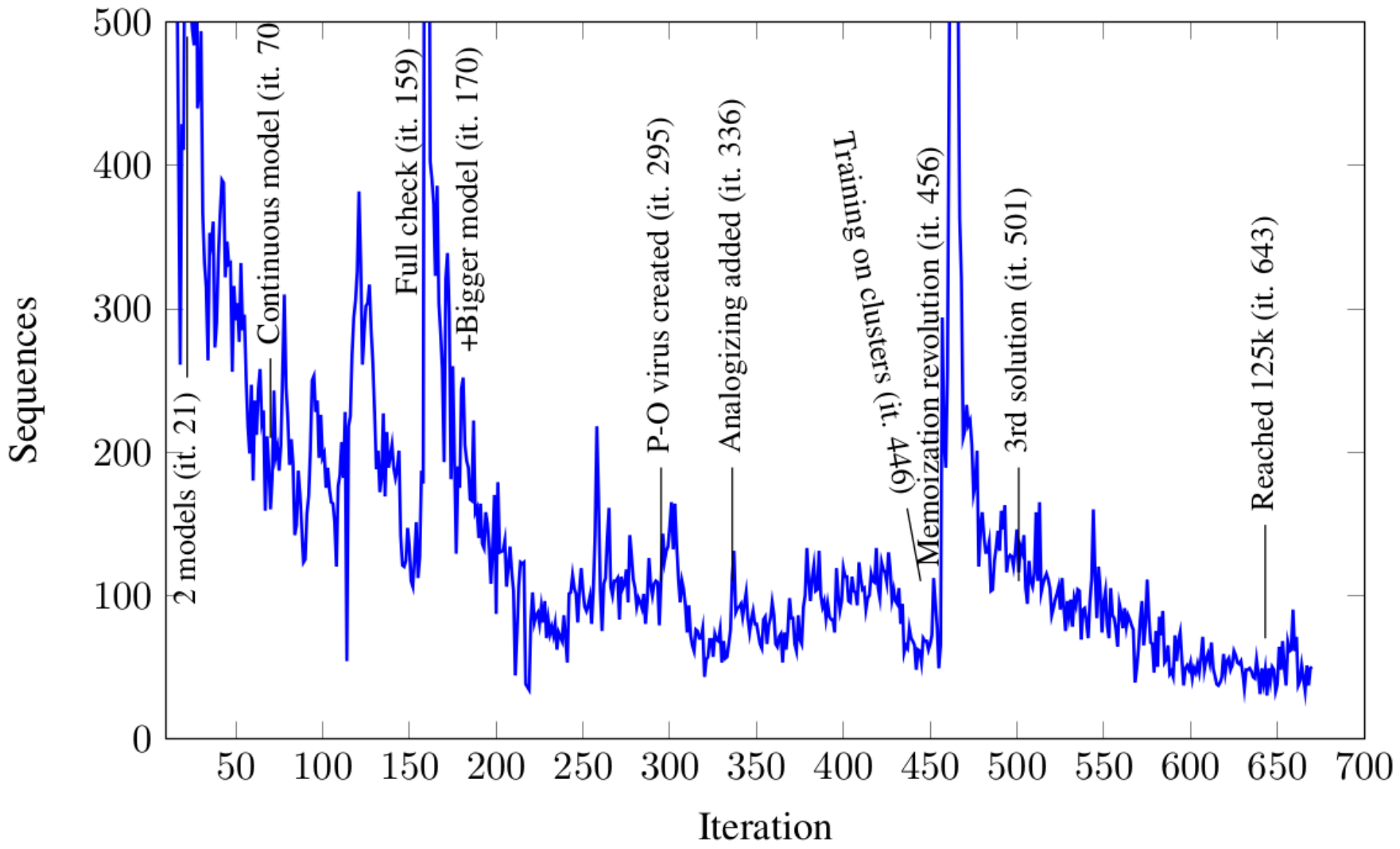
Singularity Take-Off X-mas Card



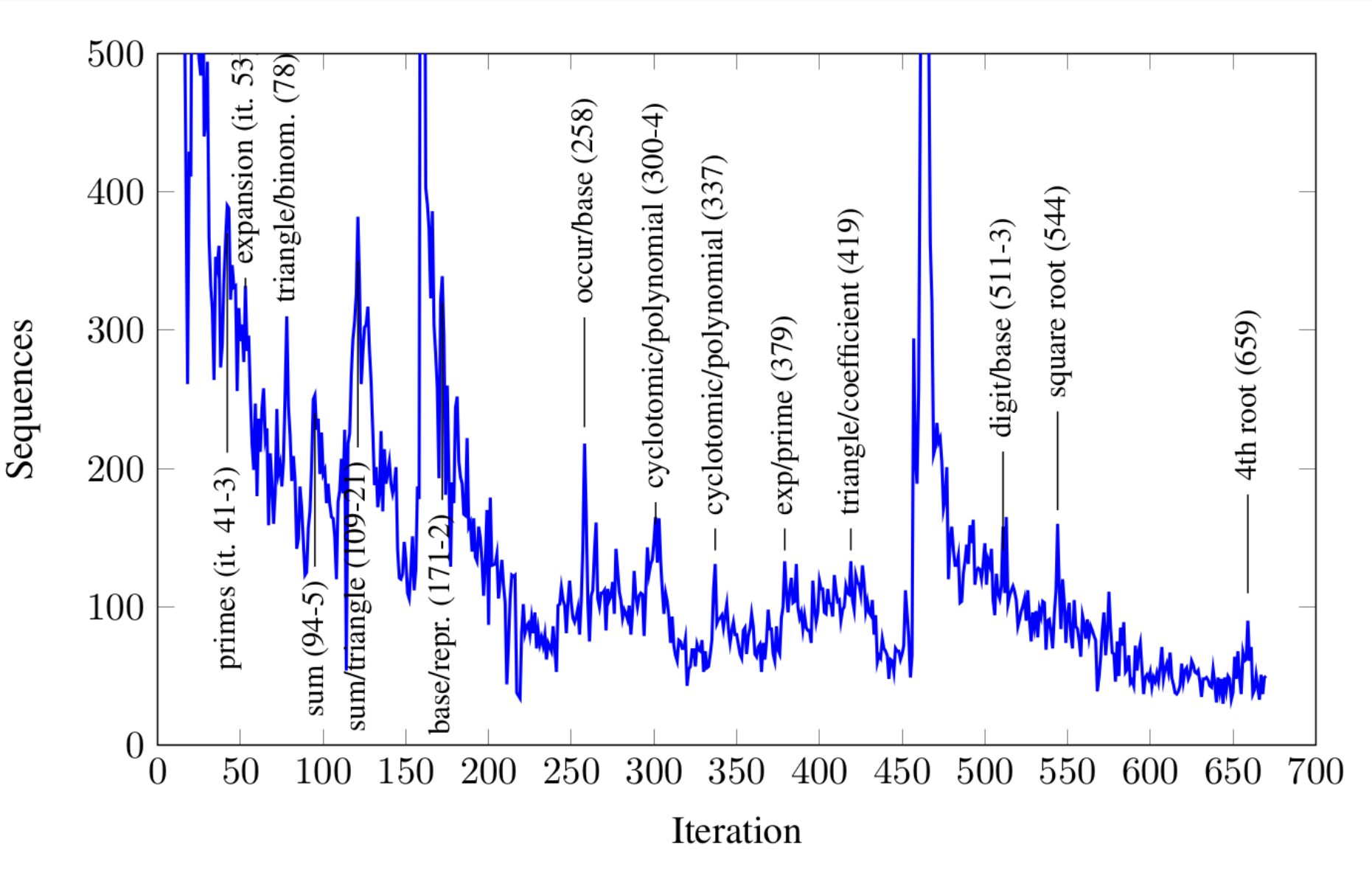
Human Made Technology Jumps



Human Made Technology Jumps



Some Automatic Technology Jumps



Some Invented Explanations

- `https://oeis.org/A4578`: Expansion of sqrt(8) in base 3:
  loop2(((y * y) div (x + y)) + y, y, x + x, 2, loop((1 + 2) * x, x, 2)) mod (1 + 2)
- `https://oeis.org/A4001`: Hofstadter-Conway $10k seq: a(n) = a(a(n-1)) + a(n-a(n-1)) with a(1) = a(2) = 1:
  loop(push(loop(pop(x), y-x,pop(x)),x) + loop(pop(x), x-1, x), x - 1, 1)
- `https://oeis.org/A40`: prime numbers:
  2 + compr((loop(x * y, x, 2) + x) mod (2 + x), x)
- `https://oeis.org/A30184`: Expand $\eta(q) * \eta(q^3) * \eta(q^5) * \eta(q^{15})$ in powers of q (elliptic curves):
  loop(push(loop((pop(x) * loop(if (pop(x) mod y) <= 0 then (x - loop(if (x mod (1 + (y + y))) <= 0 then (x + x) else x, 2, y)) else x, y, push(0, y))) + x, y, push(0, x)), x) div y, x, 1)
- `https://oeis.org/A51023`: Wolfram's $30k Rule 30 automaton:
  loop2(y, y div 2, x, 1, loop2(loop2((((y div (0 - (2 + 2))) mod 2) + x) + x, y div 2, y, 1, loop2(((y mod 2) + x) + x, y div 2, y, 1, x)), 2 + y, x, 0, 1)) mod 2
- `https://oeis.org/A2580`: $\sqrt[3]{2}$ Hales's blog: `https://t.ly/tHs1d`

Generalization of the Solutions to Larger Indices

- Are the programs correct?
- Can we experimentally verify Occam's razor?
  (implications for how we should be designing ML/AI systems!)
- 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
- This means that **SHORTER EXPLANATIONS ARE MORE RELIABLE!**
  (Occam was right, so why is everybody building trillion-param LLMs???)
- Common error: reliance on an approximation of 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 4.5M programs for the 126k sequences
- It may be quite hard to see that the programs are equivalent
- Extend to Schmidhuber's Gödel Machine?
- 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)!
- and Learn Conjecturing from Scratch (`arxiv.org/abs/2503.01389`)
- Here is one SMT encoding by Janota & Gauthier:

```
(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
```

Infinite Math-Nerd Sniping

- We have 4.5M problems for math nerds like this one:
- JU: *This thing works for the first 1k values (just checked) - any idea why?*
- `https://oeis.org/A004578` - Expansion of sqrt(8) in base 3.
- loop2(((y * y) div (x + y)) + y, y, x + x, 2, loop((1 + 2) * x, x, 2)) mod (1 + 2)
- MO: *Not a proof, just a rough idea: The program iterates the function q |-> 2+q / 1+q, where q is a rational number. This converges to sqrt(2). The number q is represented by an integer 'a' such that a* $= 3^x * (2 * q)$*, where 'x' is the input. Once the approximation is good enough,* $a = floor(3^x * sqrt(8))$*, so a mod 3 is the digit we want.*

Serious Math Conjecturing – Elliptic Curves

- Sander Dahmen: *Here are some OEIS labels related to elliptic curves (and hence modular forms), ordered by difficulty. It would be interesting to know if some of these appear in your results.*
- A006571 A030187 A030184 A128263 A187096 A251913
- JU: *We have the first three:*
- A6571 : loop((push(loop((pop(x) * loop(if (pop(x) mod y) <= 0 then ((if (y mod loop(1 + (x + x), 2, 2)) <= 0 then (x - y) else x) - y) else x, y, push(0, y))) + x, y, push(0, x)), x) * 2) div y, x, 1)
- A30187 : loop(push(loop((pop(x) * loop(if (pop(x) mod y) <= 0 then (x - loop(if (x mod (((2 + y) * y) - 1)) <= 0 then (x + x) else x, 2, y)) else x, y, push(0, y))) + x, y, push(0, x)), x) div y, x, 1)
- A30184 : loop(push(loop((pop(x) * loop(if (pop(x) mod y) <= 0 then (x - loop(if (x mod (1 + (y + y))) <= 0 then (x + x) else x, 2, y)) else x, y, push(0, y))) + x, y, push(0, x)), x) div y, x, 1)

A6571: Expansion of $q * Product_{k>=1}(1 - q^k)^2 * (1 - q^{11*k})^2$
A30187: Expansion of $\eta(q) * \eta(q^2) * \eta(q^7) * \eta(q^{14})$ in powers of q.
A30184: Expansion of $\eta(q) * \eta(q^3) * \eta(q^5) * \eta(q^{15})$ in powers of q.

More Bragging

- Hofstadter-Conway $10000 sequence: a(n) = a(a(n-1)) + a(n-a(n-1)) with a(1) = a(2) = 1.
- D. R. Hofstadter, Analogies and Sequences: Intertwined Patterns of Integers and Patterns of Thought Processes, Lecture in DIMACS Conference on Challenges of Identifying Integer Sequences, 2014.

```
Date: Sun, Mar 17, 2024
To: <dughof@indiana.edu>

Dear Douglas,

our system [1] has today (iteration 552) found a solution of
https://oeis.org/A004074. The solution in Thibault's programming
language [1] (with push/pop added on top of [1]) is:

((2*loop(push(loop(pop(x),x-1,x),x)+loop(pop(x),y-x,pop(x)),x-1,1))-1)-x

The related A4001 was solved in iteration 463 and the solution is:
loop(push(loop(pop(x), y-x,pop(x)),x) + loop(pop(x), x-1, x), x - 1, 1)
```

Thanks and Advertisement

- Thanks for your attention! (and also for the ARA funding!)
- To push AI methods in math and theorem proving, we organize:
- AITP – Artificial Intelligence and Theorem Proving
- 10th issue: September 2025, Aussois, France, `aitp-conference.org/2025`
- ATP/ITP/Math vs AI/ML/AGI people, Computational linguists
- Discussion-oriented and experimental

AI/TP Examples and Demos

- ENIGMA/hammer proofs of Pythagoras : https://bit.ly/2MVPAn7 (more at http://grid01.ciirc.cvut.cz/~mptp/enigma-ex.pdf) and simplified Carmichael https://bit.ly/3oGBdRz,
- 3-phase ENIGMA: https://bit.ly/3C0Lwa8, https://bit.ly/3BWqR6K
- Long trig proof from 1k axioms: https://bit.ly/2YZ0OgX
- Extreme Deepire/AVATAR proof of $\epsilon_0 = \omega^{\omega^{\omega^{\cdot^{\cdot^{\cdot}}}}}$ https://bit.ly/3Ne4WNX
- Hammering demo: http://grid01.ciirc.cvut.cz/~mptp/out4.ogv
- TacticToe on HOL4: http://grid01.ciirc.cvut.cz/~mptp/tactictoe_demo.ogv
- TacticToe longer: https://www.youtube.com/watch?v=BO4Y8ynwT6Y
- Tactician for Coq: https://blaauwbroek.eu/papers/cicm2020/demo.mp4, https://coq-tactician.github.io/demo.html
- Inf2formal over HOL Light: http://grid01.ciirc.cvut.cz/~mptp/demo.ogv
- QSynt: AI rediscovers the Fermat primality test: https://www.youtube.com/watch?v=24oejR9wsXs