Program Synthesis from Integer Sequences: Initial Self-Learning Run on the OEIS [*]

Thibault Gauthier

Czech Technical University in Prague, Prague, Czech Republic
email@thibaultgauthier.fr

Abstract

Through self-learning, our system discovers in three weeks programs that generate the first 16 numbers of more than 50000 OEIS sequences

1 Introduction

In this work, we propose to rely on a "self-learning" system (a system that learns from its own searches) to create programs generating sequences from the On-Line Encyclopedia of Integer Sequences (OEIS) [6] and beyond. Program synthesis for different domains (e.g. operations on lists) has been attempted by inductive logic programming systems (such as Popper [5]) and reinforcement learning systems (such as DeepCoder [1]). Within the theorem proving community, the development of methods for term synthesis has been explored in inductive theorem proving [4] and in counterexample generators [2, 3].

Here is how our self-learning approach creates programs for OEIS sequences. Its self-learning loop consists of two alternating phases: a search phase and a learning phase. Initially, our search discovers some solutions by randomly building programs and checking if they generate OEIS sequences. From those solutions, a tree neural network is trained to predict what the right building action is, given a target sequence and a partially built program. The next search is then guided by the statistical correlations learned by the network, usually producing even more solutions. One iteration of the self-learning loop is called a generation. Note, as it is usual for reinforcement learning systems, that our approach is completely unsupervised. That is to say, the system is never told the corresponding program for a particular sequence but has to discover it through guided search.

2 Programming Language

Our language contains the tokens $0, 1, 2, +, -, x, i, \times, div, mod, cond, \lambda, loop, compr$ which follow the semantics of Standard ML except for $cond, loop, compr$ defined by:

$$\begin{aligned}
cond(a,b,c) &:= \text{if } a \leq 0 \text{ then } b \text{ else } c \\
loop(f,a,b) &:= b && \text{if } a \leq 0 \\
&\quad\; f(loop(f,a-1,b),a) && \text{otherwise} \\
compr(f,a) &:= \text{failure} && \text{if } a < 0 \\
&\quad\; min\{y \mid y \geq 0 \wedge f(y,0) \leq 0\} && \text{if } a = 0 \\
&\quad\; min\{y \mid y > compr(f,a-1) \wedge f(y,0) \leq 0\} && \text{otherwise}
\end{aligned}$$

[*]This work was supported by the Czech Science Foundation project 20-06390Y.

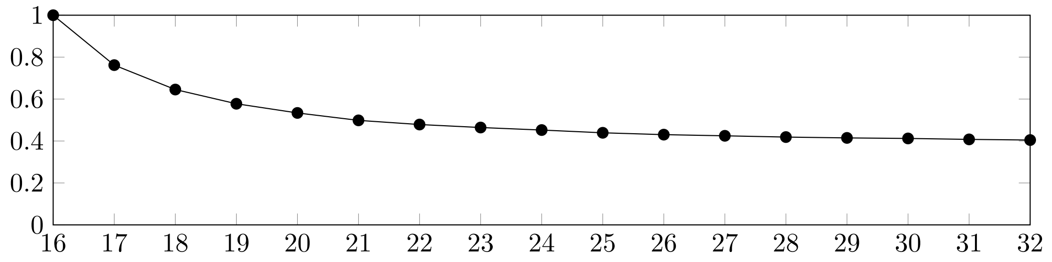

Figure 1: Percentage of sequences n-covered among 16-covered sequences with at least 32 elements

Table 1: Solutions for famous OEIS sequences

Sequence	Program
Catalan numbers	loop(f,x,1)
	where $f(x',i') = ((i' \times 2 - 1) \times x' \times 2)\ \mathit{div}\ (i'+1)$
Pseudo-prime numbers	$\mathit{compr}(\{x' \geq 0 \mid (2^{x'+2} - 2)\ \mathit{mod}\ (x'+2) = 0\},\ x) + 2$
	where $2^{x'+2} - 2 := \mathit{loop}(\lambda(x'', i'').\ (x''+1) \times 2,\ x',\ 2)$
Prime characteristic function	$(\mathit{loop}(\lambda(x,i).i \times x, x, x)\ \mathit{mod}\ (x+1))\ \mathit{mod}\ 2$

3 Results

The code for our project is available in this repository [7]. A user can also test our system using the web interface `http://grid01.ciirc.cvut.cz/~thibault/qsynt.html`.

We report on the number of solutions found during self-learning. At generation 0, the search finds 5247 16-solutions (covering the first 16 elements of an OEIS sequence) using a tree neural network initialized with random weights. After generation 11, we get 37400 16-solutions. After 100 generations, more than 50000 OEIS sequences had their first 16 elements generated by at least one program. Figure 1 measures our programs' ability to cover sequences for increasing value of n. Extrapolating the plot, we can conjecture that the percentage of solutions that generalize to arbitrary inputs converges towards 40%. In Table 1, solutions for famous sequences are presented. Each of these three solutions was manually proven to match the description given by OEIS editors for the corresponding sequence.

In the future, our priority will be to increase the number of tested inputs before declaring a program to be a solution. With this change, we should observe a decrease in the number of solutions but those solutions will be more likely to generalize to larger inputs.

References

[1] Matej Balog, Alexander L. Gaunt, Marc Brockschmidt, Sebastian Nowozin, and Daniel Tarlow. Deepcoder: Learning to write programs. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[2] Jasmin Christian Blanchette and Tobias Nipkow. Nitpick: A counterexample generator for higher-order logic based on a relational model finder. In *Interactive Theorem Proving, First International Conference, ITP 2010, Edinburgh, UK, July 11-14, 2010. Proceedings*, pages 131–146, 2010.

[3] Lukas Bulwahn. The new Quickcheck for Isabelle - random, exhaustive and symbolic testing under one roof. In *Certified Programs and Proofs - Second International Conference, CPP 2012, Kyoto, Japan, December 13-15, 2012. Proceedings*, pages 92–108, 2012.

[4] Koen Claessen, Moa Johansson, Dan Rosén, and Nicholas Smallbone. Automating inductive proofs using theory exploration. In Maria Paola Bonacina, editor, *Conference on Automated Deduction (CADE)*, volume 7898 of *LNCS*, pages 392–406. Springer, 2013.

[5] Bruce Nielson and Daniel C. Elton. Induction, popper, and machine learning. *CoRR*, abs/2110.00840, 2021.

[6] Neil J. A. Sloane. The on-line encyclopedia of integer sequences. In Manuel Kauers, Manfred Kerber, Robert Miner, and Wolfgang Windsteiger, editors, *Towards Mechanized Mathematical Assistants, 14th Symposium, Calculemus 2007, 6th International Conference, MKM 2007, Hagenberg, Austria, June 27-30, 2007, Proceedings*, volume 4573 of *Lecture Notes in Computer Science*, page 130. Springer, 2007.

[7] Gauthier Thibault. Software accompanying the paper "Program Synthesis for the OEIS". `https://github.com/barakeel/oeis-synthesis`, 2022.