Automatic Program Synthesis of Long Programs with a Learned Garbage Collector

Amit Zohar\textsuperscript{1}, Lior Wolf\textsuperscript{1,2}
\textsuperscript{1}Tel Aviv University \textsuperscript{2}Facebook AI Research

Problem:
- \textbullet \hspace{2mm} Input: A handful of input/output examples for a program
- \textbullet \hspace{2mm} Goal: Generate a program that corresponds to all of the examples

Method:
- \textbullet \hspace{2mm} Step-Wise Approach: Predict the program statement by statement
- \textbullet \hspace{2mm} Garbage Collector: Predict which variables can be discarded at each step
- \textbullet \hspace{2mm} Dynamic Input: Use the intermediate program states as input for the model
- \textbullet \hspace{2mm} Prediction Guided Search: Perform a tree search guided by the model’s predictions to find a correct program

Achievements:
- \textbullet \hspace{2mm} Synthesis of programs more than twice as long as state-of-the-art
- \textbullet \hspace{2mm} Near perfect success rate for existing lengths

Overview

Problem: A handful of input/output examples for a program

Goal: Generate a program that corresponds to all of the examples

Method:
- Step-Wise Approach: Predict the program statement by statement
- Garbage Collector: Predict which variables can be discarded at each step
- Dynamic Input: Use the intermediate program states as input for the model
- Prediction Guided Search: Perform a tree search guided by the model’s predictions to find a correct program

Achievements:
- Synthesis of programs more than twice as long as state-of-the-art
- Near perfect success rate for existing lengths

Program Environment

We wish to represent an intermediate state of a program during its execution:

Program State:
The sequence of all the variable values acquired thus far, starting with the program’s input, and concatenated with the desired output

Program Environment:
The concatenation of all the program’s states (one for each input/output example)

Inference:
- Search for a correct program:
  - Tree search – nodes are environments, edges are statements
  - Use Complete Anytime Beam Search (CAB) – perform beam search in a loop, increasing the beam size and width at each iteration
  - Maintain a program environment with the program predicted thus far
  - Query the network at each step and order edges accordingly
  - When the number of variables is exceeded – drop variables according to the network’s predictions

Training:
- Optimize three tasks concurrently:
  - Statement Prediction: The main task
  - Variable Dropping: Allows to generate longer programs
  - Function Prediction: Auxiliary task - provides hierarchical structure to statements

Comparison of our method (PCCoder) with DeepCoder (*reimplementation) for various program lengths

Ablation analysis of our method (program length 8)