Synthesis of Recursive Programs

Philip Chen, philipchen247@gmail.com
Walnut High School, Class of 2024
USC Viterbi Department of Computer Science, SHINE 2022

Introduction

Program synthesis is the task of automatically finding a program in the underlying programming language that satisfies the user intent expressed in the form of some specification.

Recursion is a method of solving a computational problem where the solution depends on solutions to smaller instances of the same problem. Recursion solves such recursive problems by using functions that call themselves from within their own code.

Recently, there has been growing interest in recursive program synthesis, so our goal is to synthesize recursive programs implementing intended behaviour.

Objective & Impact of Professor's Research

Professor Ragohotaman's research is about program synthesis, formal verification, and static analysis through the use of machine learning and formal methods.

- Program synthesis is the task of automatically finding a program that satisfies the user intent expressed in the form of a specification.
- Formal verification is the act of proving or disproving the correctness of intended algorithms underlying a system with respect to a certain formal specification or property, using formal methods of mathematics.
- Static program analysis is the analysis of computer programs performed without executing them,
- Machine learning is a field of inquiry devoted to understanding and building methods that 'learn' that is, methods that leverage data to improve performance on some set of tasks.

Professor Raghothaman's research aims to combine these concepts to make software development easier for humans.

Methods

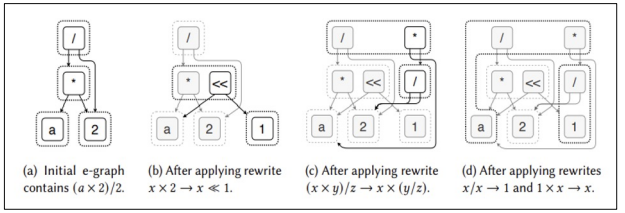
SyGuS and CVC5

The syntax-guided synthesis problem (SyGuS) is a logical framework for the program synthesis problem. The input to SyGuS consists of a background theory, a semantic correctness specification for the desired program given by a logical formula, and a syntactic set of candidate implementations given by a grammar. The computational problem then is to find an implementation from the set of candidate expressions so that it satisfies the specification in the given theory.

CVC5 is an efficient open-source automatic theorem prover for Satisfiability Modulo Theories (SMT) problems. It can be used to prove the satisfiability of first-order formulas with respect to (combinations of) a variety of useful background theories. It further provides a Syntax-Guided Synthesis (SyGuS) engine to synthesize functions with respect to background theories and their combinations.

Egg and E-Graphs

An e-graph efficiently represents a congruence relation over many expressions. The egg library uses e-graphs to provide a new way to build program optimizers and synthesizers and provides high-performance, flexible e-graphs implemented in Rust. It also allows syntactic rewrites in order to modify expressions. We want to use these concepts to rewrite expressions in order to find patterns.



The expression (a *2) / 2 can be rewritten a number of ways and some possibilities are shown in the e-graph.

Our Approach

Given a specification, CVC5 produces a non-recursive program. Unfortunately, non-recursive programs only work for limited inputs, not for any inputs. Our goal is to synthesize a recursive program that works for all inputs. We designed an algorithm with four steps to synthesize recursive programs. Using the non-recursive implementation from CVC5, we generate all equal expressions using Egg. This involves rewriting rules that enable us to expand an e-graph with all possible combinations of expressions. From there, we detect a pattern that generalizes the implementation. However, not all recursive programs have a pattern. Therefore, we locate the recursive expressions that do have patterns and extract them. Finally, we synthesize a a recursive program that works for any inputs of the function based on the pattern.

The steps for our algorithm to synthesize recursive programs.

Acknowledgements

Firstly, I'd like to thank my parents for making this possible. I'd also like to formally thank Professor Mukund Ragohotaman and Amirmohammad Nazari for mentoring me. Amir has been such an amazing person to work with and he has really opened my eyes. Lastly I would like to thank the SHINE staff for being such great accommodators!