Generating Grammar Rules for Syntax-Guided Synthesis

ABSTRACT: Conditional Linear Integer Arithmetic (CLIA) Track synthesizes function in Linear Integer Arithmetic (LIA) combined with if-then-else construct (ite). Specification is stated in the form of logical constraints for the SyGuS problem. Even though CVC4 solves most of the benchmark problem but the synthesis time is slow for some of the simple problems and even timeouts in some cases. We propose to predict the grammar rules for given set of logical constraints to restrict the search space consequently improving the synthesis time.

INTRODUCTION:
Program Synthesis from a high level specification like natural language descriptions or input output examples has gained a lot of attention from both PL community and the DL community. Works like DeepCoder and RobustFill proves that deep learning can be helpful in synthesizing domain specific programs. In spite of such advancements, synthesizing correct by construction programs is a challenging task. Inferring programs from high level specification does not guarantee correctness for the inferred programs. Therefore, instead of high level spec, we propose to use logical constraints as specifications and instead of asking a Neural network to synthesize a program we generate a restricted grammar to be fed along with the constraints to a sygus solver (CVC4 in our case).

SyGuS competition accepts solvers that compete on a given set of benchmark problems. The benchmark problems consists of a background theory T, semantic specification given by logical formula and a syntactic set of grammar rules. SyGuS solvers are required to solve this problem and find an instance of program that satisfies the logical constraints in the given theory.

RELATED WORK:
Grammar Filtering For Syntax-Guided Synthesis by Kairo Morton falls in the domain of Programming By Examples (PBE) which filters out the unnecessary grammar productions from the syntax provided. As a result it reduces the runtime by 47.65% and outperforms the other tool in PBE Strings Track of SyGuS Competition 2019. This motivates us to try this for other tracks. CLIA track does not specify an explicit grammar but assumes a general grammar which leads to a large search space. Our idea is to generate grammar for sygus problems of CLIA track.

PROBLEM STATEMENT:
Given a set of logical constraints **C**, we would like to output a sequence of grammar rules **g** such that it restricts the search space and hence improves the synthesis time **T** for synthesizing a program **P** that satisfies **C**.

$$\mathbf{M(C) = g},$$

where M is the proposed model.

MOTIVATING EXAMPLE:
We found a couple of benchmark examples on which CVC4 either takes longer synthesis time or timeouts.

Example (i): To synthesize program for diff.sl[1] CLIA benchmark, CVC4 takes ~80 sec when there's no syntactic restriction[2]. When we provide explicit grammar to it, Same solution is reached in just ~14 sec i.e. around 83% faster.
Example (ii): On following constraints

```
(constraint (=> (= ( mod x 2) 1) ( = (f x) 0) ))
```

```
(constraint (=> (= ( mod x 2) 0) ( = ( + (f x) (f x)) x) ) )
```

The to be synthesized function should be **div1(x) = x/2**, but the cvc4 is unable to get to the solution without explicit grammar.
When following grammar rules are included

```
((I Int (x 2 1 0
            (div I 2)
            (mod I I)
            (ite B I I)))
 (B Bool ((= I I)))),
```

cvc4 gives a correct solution as

```
(define-fun f ((x Int)) Int (div (ite (= (mod x 2) 1) 0 x) 2))
```

Which means that if x is odd f(x) = 0 because of the background theory of LIA and if x is even it evaluates to f(x) = x/2

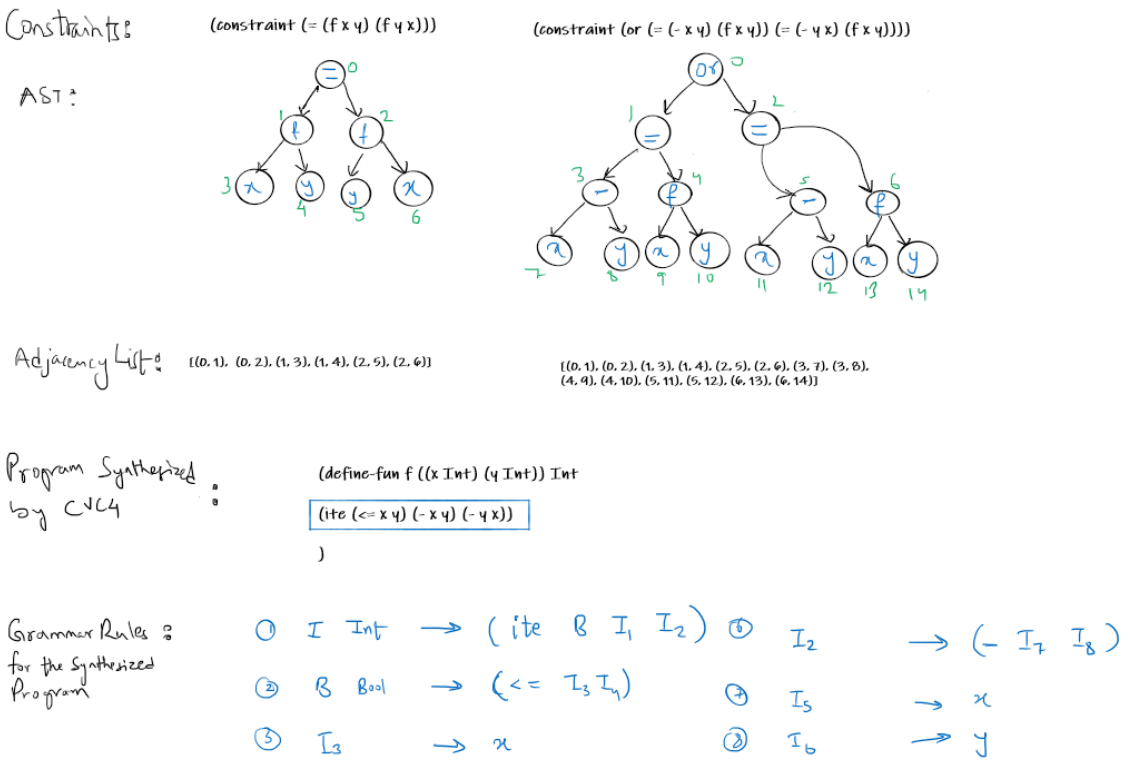
PROPOSED MODEL:
Our proposed model will have two parts - (i) Constraint Encoder and (ii) Syntax Decoder.

(i) Constraint Encoder: We propose to use a Gated Graph Neural Network (GGNN) as a constraint encoder which takes Abstract Syntax Trees (AST) of the constraints in a sygus problem (S) and learns a Graph Embedding (E_{AST}) for it. If there are more than one constraint then each embeddings will be combined to get one final embedding (E_s) to represent the sygus problem completely.

(ii) Syntax Decoder: For decoding the syntax (i.e. grammar rules) we use an RNN decoder that will accept the embedding from Constraint Encoder and produce the most likely grammar rules g. These grammar rules along with the constraints are then fed to existing sygus-solver to obtain correct by construction program and with faster synthesis time.

INPUT/OUTPUT:
Input to Model: Logical Constraints in the form of AST.
Output to Model: Sequence of Production Rules

An Example of Input/Output -



DATA COLLECTION STRATEGY:

As the existing benchmarks consists of only 88 examples, it is insufficient to learn anything significant from it. I plan to generate the dataset through random constraint generation strategy which I discuss below.

Random Constraint Generation:

i. Set precedence order for the operators (Arithmetic -> Comparison -> Logical)
ii. Select an operator randomly
iii. Expand its operands randomly from the operand list (Initially contains variables and uninterpreted function)
iv. Repeat these steps for certain number of times depending on how complex constraints one wants to generate.

The randomly generated constraints are then fed to the cvc4 in sygus format[3] to get a program. Finally the program is then given to a parser that will parse it and determine the production rules that was used to get the program.

The dataset will be of format (constraint, sequence of grammar rules).

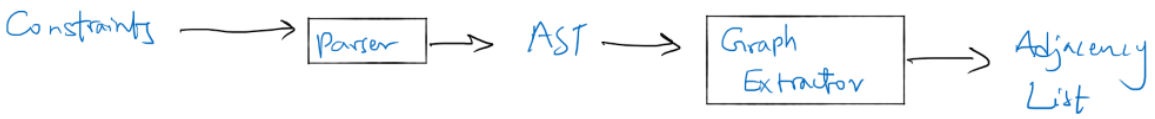
DATA PREPARATION STEP:

Data needs to be mapped to the real space to feed it to the neural network to learn from it. Following

For Constraints -

1. Take each constraint and get its Abstract Syntax Tree

2. Obtain the Adjacency List for each AST which can be directly fed to the Graph Neural Network directly.



For Grammar Rules – we just obtain the one-hot vector for each production in the grammar. Graph Extractor will obtain the adjacency list from a given AST.

Pre-processed dataset will look like (Adjacency List, Sequence of Grammar Rules in the form of OHE)

[1] Link to benchmark problem Diff.sl
[2] No syntactic restriction only means that it considers general CLIA grammar as its search space when no explicit grammar is provided to it.
[3] https://sygus.org/language/