Synthesis Project Proposal

Will Thomas, Logan Schmalz, Sarah Johnson

In this project we will use a dynamic programming (divide and conquer) style approach to improve the performance of Semantic Guided Synthesis (SemGuS). We will attempt to do this by determining a smaller version of a top level grammar (referred to as a sub-grammar) that is realizable and can then be sent off to a more optimized solver. We anticipate that the performance benefits will be achieved by reducing the size of the grammar that will be used during synthesis, as this smaller grammar can be more quickly enumerated.

We will follow the approach of Kim et al.[1], modifying the algorithm being run by the MESSY synthesis tool. We will find the smallest possible subset of grammar rules for which given IO examples will have their sem_{start} unrealizable on a certain subset of the input output examples. If we have two grammar subsets G_1 and G_2 such that the IO examples on which they are realizable combined are all the IO examples, then a unioned grammar $G_f = G_1 \cup G_2$ will be able to synthesize the entire program and realizable (so far as the given input output examples work). At a high level this would be an optimal approach, but the checking of realizability for the sub-grammars may be prohibitively long. One approach to solving this would be to develop a heuristic to terminate the checking of realizability on sub-grammars in order to maintain a timely approach to synthesis. Additionally, we could just use the easily proven unrealizability facts of sub-grammars and combine grammars until proving unrealizability is impossible (in which case it must be realizable) or takes prohibitively long (in which case it is likely realizable).

Given a set of input output examples IO and a grammar G, we will return a sub-grammar G_s (a grammar with less productions than grammar G) if one exists such that a program can be realized satisfying IO with G_s. This sub-grammar will then be handed off to a state-of-the-art solver such as Z_3 to be synthesized using the sub-grammar. It is hypothesized that the benefit of checking for a smaller (if not possibly minimal) sub-grammar will allow for faster synthesis of realizable problems. Our benchmarks for this problem will be the SyGuS benchmarks used in profiling the MESSY synthesis tools.

Example: Given a Grammar G:

```
Start   ::= while B do S
B       ::= E < E
S       ::= S; S | x := E | y := E
E       ::= x | y | (E & E) | (E | E) | (E ^ E)
```

and IO examples `<[<6,9>; <44,247>, <14,15>], [15; 219; 1]>` we would return a classification of **realizable** for a sub-grammar of G, G_s where E_s `:= x | y | E ^ E`, the remaining constructions are the same. This allows the searching of a solving program by a solver such as Z_3 to search a smaller space (removing the & and | productions). A possible solution that could be found by Z_3 would be `while (x < y) do (x := x ^ y; y := x)`.

[1] "Semantics-Guided Synthesis": https://dl.acm.org/doi/10.1145/3434311