Abstract

This document has examples for problems that can be formulated as instances of SyGuS from three application domains (1) invariant generation (2) optimizing compilers (3) BitVector manipulation. For each of them you can try (a) to formulate the problem as a SyGuS problem (b) try to solve it yourself (c) upload the instance to StarExec and run some of the solvers on it to see their solution.

Invariant Generation

Consider the following program:

```c
void cegar2(int N) {
    int x = 0;
    int m = 0;

    //printf("%d %d %d ;accept\\n", x, m, N);
    while (x < N) {
        if(rand() % 2)
            m = x;
        x = x + 1;
        printf("%d %d %d ;accept\\n", x, m, N);
    }
    if (N > 0)
        assert ((0 <= m) && (m < N));
}
```

Questions:

1. Formulate a SyGuS instance that synthesizes a loop invariant for the above program. **Hint:** Use the following template:
(set-logic LIA)

(synth-inv inv-f (... <Typed Parameters> ...))

(declare-primed-var x Int)
(declare-primed-var ...)
(declare-primed-var ...)

(define-fun pre-f (... <Typed Parameters> ...) Bool
   ... <constraint>)

(define-fun trans-f (... <Typed Parameters and their primed version> ...) Bool
   ... <constraint>)

(define-fun post-f (... <Typed Parameters> ...) Bool
   ... <constraint>)

(inv-constraint inv-f pre-f trans-f post-f)

(check-synth)

2. Write a possible solution for an invariant.

3. Upload your formulation of the SyGuS instance to StarExec (for “benchmark type” choose “SyGuS-INV”) and run the solvers ICE-DT, Alchemist-CS and CVC4 on this instance. Compare their solutions and yours.

Compiler Optimization

A quantum control computer uses expressions from the following grammar:

\[ g ::= c \mid g + g \mid g - g \mid g?g \]

where \( c \) is any integer constant, \( + \) is addition, \( - \) is subtraction, and \( a?b \) stands for “if \( a \geq 0 \) then \( a \) else \( b \)”. That is, it is equivalent to the expression \((a \geq 0)?a : b\) in the C programming language, and the expression \((\text{ite } (\geq a 0) \ a \ b)\) in SyGuS syntax.

1. Formulate a SyGuS instance that synthesize a function in the grammar \( g \) that implements a maximum of two numbers. **Hint:** complete the following template

\[
\begin{align*}
\text{(set-logic LIA)} \\
\text{(define-fun qm ((a Int) (b Int)) Int} \quad &\quad (\text{ite } (\geq a 0) \ a \ b)) \\
\text{(synth-fun max2 ((x Int) (y Int)) Int} \quad &\quad ((\text{Start Int} \ x y
\end{align*}
\]
(declare-var x Int)
(declare-var y Int)
(constraint (= (max2 x y) ...
(check-synth)

2. Write a possible solution

3. To implement a loop variable for the quantum control computer, formulate a SyGuS instance that synthesize a function in the grammar $g$ over two naturals $k$ and $i$ that decrements $i$ modulo $k + 1$.

4. Write a possible solution for this question; the smaller the better.

5. Formulate a SyGuS instance that synthesize a function in the grammar $g$ over two naturals $k$ and $i$ that increments $i$ modulo $k + 1$.

6. Write a possible solution for this question; the smaller the better.

7. To implement variable for two nested loops in the quantum control computer, where the inner loop is decremented as above, formulate a SyGuS instance that synthesize a function in the grammar $g$ over three naturals $k$, $i$ and $j$ that decrements $j$ in a cyclic manner modulo $k + 1$ only when $i$ reaches 0 (and remains the same otherwise).

8. Write a possible solution for this question; the smaller the better.

9. Upload your phrasing of the SyGuS instances in the problems above to StarExec (for “benchmark type” choose “SyGuS-General”) and run the solvers Enumerative, Stochastic, Sketch-based, CVC4 on these instances. Compare theirs solutions and yours.

BitVector Manipulations

Consider the function $seAvg$ that computes the average value of two bitvectors $x$ and $y$:

```
(define-fun seAvg ((x (BitVec 8)) (y (BitVec 8))) (BitVec 8)
  (bvashr (bvadd x y) #x01))
```

The function $seAvg$ doesn’t quite handle the cases where addition of $x$ and $y$ i.e. $(bvadd x y)$ might result in an overflow. Write a SyGuS formulation of a function $(Avg x y)$ that implements the full bit vector addition including the overflow cases. **Hint:** use concrete examples to specify the behavior of the unknown function for the overflow cases, and specify that for non-overflow cases it should have the same behavior as $seAvg$. 

3