The syntax-directed synthesis problem

Set $L$ of candidate programs
Formula $\varphi(x, y, z, ..., f, ...)$

Synthesizer
Program $e \in L$ so that $\varphi[f / e]$ is valid

Input format similar to SMT-LIB

```
(synth-fun max3 ((a Int) (b Int) (c Int)) Int
 ((Start Int (a b c 0 1 (+ Start Start))
   (ite BoolExpr Start Start))
 (BoolExpr Bool ((<= Start Start) (= Start Start)
   (>= Start Start))))
```

```
(declare-var x Int) (declare-var y Int) (declare-var z Int)
```

∀ $x, y, z \in \mathbb{Z}$, the following must hold:

```
(constraint (>= (max3 x y z) x))
...
(constraint (or (= x (max3 x y z))
 (or (= y (max3 x y z))
  (= z (max3 x y z))))
```

```
(check-synth) Do the magic
```

Motivation

- Common problem format so that different synthesis algorithms can be compared and understood.
- Allows front-end writers to target a single intermediate form and choose from a variety of off-the-shelf synthesizers.
- Basis for synthesis competition (next year)

Ongoing Work

- Complete work on the in-house backends.
- Collect benchmarks – from the AutoGrader, Transit, etc.
- Invite others to submit alternative backends for the synthesis competition.
- Choose evaluation metrics. Some options include: number of benchmarks solved, and average time per benchmark. Do we consider mean solution time for the randomized algorithms?

Choose program $e$
Verify $e$

Feedback: Process counterexample point

Symbolic CEGIS: Use an SMT solver to compute $e$ (Gulwani et al, PLDI 2011)
Enumerative CEGIS: Use efficient expression enumeration to find $e$ (Udupa et al, PLDI 2013)
Stochastic search: Use Monte Carlo iteration to find $e$ (Schufza et al, ASPLOS 2013)