GenSynth: Synthesizing Datalog Programs without Language Bias

Jonathan Mendelson*, Aaditya Naik*, Mukund Raghothaman, Mayur Naik

Background

Relational Input-Output Data

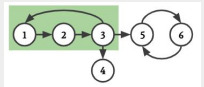
Edge

1	2
2	3
3	1

SCC

1	3
3	1
2	3

Represents nodes in the same strongly connected component



Given input-output data (above), can we *synthesize* a Datalog program (below)?

A Datalog program for the SCC task above:

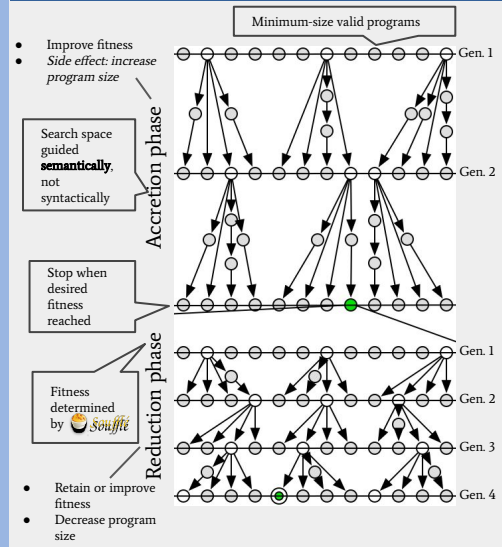
r1: path(x, y) :- edge(x, y).
r2: path(x, y) :- edge(x, z), path(z, y).
r3: scc(x, y) :- path(x, y), path(y, x).

Collection of rules applied until fixpoint

Why is this challenging?

- Invented predicates, recursion
- **Previous approaches rely on syntactic bias / program grammar**
  - Transforms the challenging problem of program synthesis to the challenging problem of defining an appropriate search space
- GenSynth does both at once using a genetic algorithm

Approach



Mutations

- Slight modifications to existing programs
- For example, the **Swap** mutation exchanges the position of two arguments:

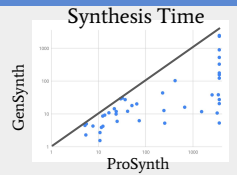
```
Original: SCC(x0, x1) :- Edge(x2, x0), Edge(x1, x3).
Mutated: SCC(x0, x1) :- Edge( x1, x0), Edge( x2, x3).
```

Results

- 42 benchmark programs from knowledge discovery, program analysis, SQL query families

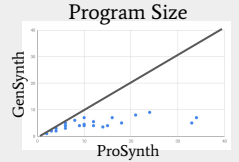
GenSynth is faster than ProSynth

- GenSynth is faster than ProSynth on all 42 benchmarks
- GenSynth never times out, while ProSynth times out on 11/42 benchmarks



GenSynth produces better programs

- GenSynth produces smaller programs than ProSynth on all 42 benchmarks
- GenSynth produces programs with <10 literals on all benchmarks



Take-aways

- New template-free Datalog synthesis approach
- High quality programs thanks to reduction phase
- Automatic predicate invention: schema determined dynamically
- Ability to handle noise

University of Pennsylvania & University of Southern California

Contact: {jonom, asnaik, mhnaik}@seas.upenn.edu; raghotha@usc.edu