GenSynth: Synthesizing Datalog Programs without Language Bias
Jonathan Mendelson*, Aaditya Naik*, Mukund Raghothaman, Mayur Naik

Background

Relational Input-Output Data

<table>
<thead>
<tr>
<th>Edge</th>
<th>SCC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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).

Why is this challenging?
- Invented predicates, recursion
- Previous approaches rely on syntactic bias / program grammar
- GenSynth does both at once using a genetic algorithm

Approach

- Improve fitness
- Side effect: increase program size
- Search space guided semantically, not syntactically
- Stop when desired fitness reached
- Retain or improve fitness
- Decrease program size
- Stop when desired fitness reached

Minimum-size valid programs

GenSynth is faster than ProSynth

- GenSynth never times out, while ProSynth times out on 11/42 benchmarks
- GenSynth produces smaller programs than ProSynth on all 42 benchmarks
- GenSynth produces programs with <10 literals on all benchmarks

Results

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

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

Contact: [jonom, asnaik, mhnaik]@seas.upenn.edu; raghotha@usc.edu

Additional Diagrams:

- A collection of rules applied until fixpoint
- Collection of rules applied until fixpoint

- Minimum-size valid programs
- Minimum-size valid programs

- Search space guided semantically, not syntactically
- Search space guided semantically, not syntactically

- Slight modifications to existing programs
- 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).