PROJECTOR: an automatic logic program rewriting tool for better performance

Nick Hippen & Yuliya Lierler
What is Answer Set Programming (ASP)?

• Constraint programming paradigm geared towards solving difficult combinatorial search problems
• Prolog-like syntax

<table>
<thead>
<tr>
<th>Logic Rule</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>child(X,Y) ← parent(Y,X).</code></td>
<td>X is a child of Y if Y is a parent of X.</td>
</tr>
<tr>
<td><code>innocent(X) ← not guilty(X).</code></td>
<td>X is innocent if I have no reason to believe that X is guilty</td>
</tr>
</tbody>
</table>

Nick Hippen
University of Nebraska at Omaha
ASP Solver Architecture

logic program → Grounder → grounded program → ASP Solver → answer sets

logic program → PROJECTOR → rewritten logic program → Grounder → grounded program → ASP Solver → answer sets
# Grounding Logic Programs

<table>
<thead>
<tr>
<th>Logic Program</th>
<th>Grounded Program</th>
<th>Intelligently Grounded Program</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>parent(bob, ally).</code></td>
<td><code>parent(bob, ally).</code></td>
<td><code>parent(bob, ally).</code></td>
</tr>
<tr>
<td><code>parent(marry, john).</code></td>
<td><code>parent(marry, john).</code></td>
<td><code>parent(marry, john).</code></td>
</tr>
<tr>
<td><code>cousin(X, Y) ← parent(P1, X), parent(P2, Y), sibling(P1, P2), X ≠ Y.</code></td>
<td><code>cousin(john, marry) ← parent(ally, john), parent(bob, marry), sibling(ally, bob), john ≠ marry.</code></td>
<td><code>cousin(ally, john) ← parent(bob, ally), parent(marry, john), sibling(bob, marry), ally ≠ john.</code></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td><code>cousin(bob, bob) ← parent(bob, bob), parent(bob, bob), sibling(bob, bob), bob ≠ bob.</code></td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Improving Performance

Smaller grounding sizes often translate into faster solve times

Idea: Split a logic rule into multiple rules so that the number of variables present in each new rule is smaller than that of the original.

Projection
Two types: $\alpha$ and $\beta$
**PROJECTOR Result: \( \alpha \)**

<table>
<thead>
<tr>
<th>Logic Program</th>
<th>PROJECTOR: ( \alpha )-projection</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\text{cousin}(X,Y) \leftarrow \text{parent}(P1,X), \text{parent}(P2,Y), \text{Sibling}(P1,P2), \ X \neq Y.)</td>
<td>(p0(Y,P1) \leftarrow \text{Sibling}(P1,P2), \text{parent}(P2,Y).)  (p1(Y,X) \leftarrow \text{parent}(P1,X), p0(Y,P1).) (\text{cousin}(X,Y) \leftarrow X \neq Y, p1(Y,X).)</td>
</tr>
</tbody>
</table>
# Nondeterministic behavior

<table>
<thead>
<tr>
<th>Logic Program</th>
<th>PROJECTOR: $\alpha$-projection Scenario #1</th>
<th>PROJECTOR: $\alpha$-projection Scenario #2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$cousin(X,Y) \leftarrow parent(P1,X)$, $parent(P2,Y)$, $sibling(P1,P2)$, $X \neq Y$.</td>
<td>$p0(Y,P1) \leftarrow sibling(P1,P2)$, $parent(P2,Y)$. $p1(Y,X) \leftarrow parent(P1,X)$, $p0(Y,P1)$. $cousin(X,Y) \leftarrow X \neq Y$, $p1(Y,X)$.</td>
<td>$p0(P2,X) \leftarrow sibling(P1,P2)$, $parent(P1,X)$. $p1(Y,X) \leftarrow parent(P2,Y)$, $p0(P2,X)$. $cousin(X,Y) \leftarrow X \neq Y$, $p1(Y,X)$.</td>
</tr>
<tr>
<td>Logic Program</td>
<td>PROJECTOR: $\alpha$-projection</td>
<td>PROJECTOR: $\beta$-projection</td>
</tr>
<tr>
<td>---------------------------------------------------</td>
<td>---------------------------------</td>
<td>--------------------------------</td>
</tr>
</tbody>
</table>
| $\text{male}_1\text{cousin}(X,Y) \leftarrow \text{parent}(P1,X)$, \text{parent}(P2,Y), \text{sibling}(P1,P2), $X \neq Y$ \text{male}(X). | $p0(Y,P1) \leftarrow \text{sibling}(P1,P2), \text{parent}(P2,Y)$. $p1(Y,X) \leftarrow \text{parent}(P1,X), p0(Y,P1)$.

$\text{male}_1\text{cousin}(X,Y) \leftarrow X \neq Y, \text{male}(X), p1(Y,X)$. | $p0(Y,P1) \leftarrow \text{sibling}(P1,P2), \text{parent}(P2,Y)$. $p1(Y,X) \leftarrow \text{parent}(P1,X), p0(Y,P1), \text{male}(X)$. $\text{male}_1\text{cousin}(X,Y) \leftarrow X \neq Y, \text{male}(X), p1(Y,X)$. |
Experimental Analysis

• **ASPCCG**: ASP based natural language parser
  • 3 encodings of increasing levels of human optimization
    • Created by Matthew Buddenhagen, Yuliya Lierler & Peter Schuller
  • Enc1: No human optimization
  • Enc7: Moderate human optimization
  • Enc19: Notable human optimization
ASPCCG: Encoding 1
ASPCCG: Encoding 7

Solve Time

Grounding Size
ASPCCG: Encoding 19
ASPCCG: Overall
Related, Current & Future Work

- Related work: Ipopt (Bichler, Morak, Woltran, 2016)
- Paper will be submitted to Practical Aspects of Declarative Languages (PADL) 2019 this weekend
- System PROJECTOR available on the UNO NLPKR Lab website

Future Work
- Gather more benchmarks
- Grounding size prediction
- Improve language support
Acknowledgements

• Michael Dingess
• Brian Hodges
• Daniel Houston
• Roland Kaminski
• Liu Liu
• Dr. Mirek Truszczynski
• Stefan Woltran
Questions?