Tabled Abduction in Logic Programs

Ari Saptawijaya, Luís Moniz Pereira

Centro de Inteligência Artificial (CENTRIA)
Departamento de Informática
Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa

ICLP 2013
Istanbul, 27 August 2013
Abductive Logic Programming

- Abduction: from observed evidence to its best explanation
- Abduction in Logic Programs
  - Rules:
    - shoes_wet ← grass_wet.
    - grass_wet ← sprinkler_running.
    - grass_wet ← rained.
    - clothes_wet ← rained.
    - clothes_dry.
    - IC: false ← clothes_wet, clothes_dry.
  - Abducibles: sprinkler_running, rained.
  - Query: ?- shoes_wet, not false.
  - Abductive solutions: sprinkler_running
- Applications: diagnosis, decision making, reasoning of rational agents, . . .
Tabled Abduction: Motivation & Main Idea

\[ P_1 : \quad q \leftarrow a. \quad r \leftarrow b, q. \quad p \leftarrow r, q. \]

- Abducibles: \{a, b\}
- Query: \(?- q. \quad ?- r. \quad ?- p.\)
  - Explaining q: \[a\].
  - Explaining r: recompute q?
  - Explaining p: recompute r and q?
- Adopt tabling in LP, for abductive solution reuse
  - Solutions reuse in distinct context!
- Example
  - \(?-q: \text{table } [a] \text{ as solution to } q.\)

<table>
<thead>
<tr>
<th>Goal</th>
<th>Solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>q</td>
<td>[a]</td>
</tr>
</tbody>
</table>
Tabled Abduction: Motivation & Main Idea

\[ P_1 : \quad q \leftarrow a. \quad r \leftarrow b, q. \quad p \leftarrow r, q. \]

- **Abducibles**: \{a, b\}
- **Query**: ?- q. ?- r. ?- p.
  - Explaining q: [a].
  - Explaining r: recompute q?
  - Explaining p: recompute r and q?
- **Adopt tabling** in LP, for abductive solution reuse
  - Solutions reuse in distinct context!
- **Example**
  - ?-q: table [a] as solution to q.
  - ?-r: reuse solution q with context [b], but

<table>
<thead>
<tr>
<th>Goal</th>
<th>Solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>q</td>
<td>[a]</td>
</tr>
<tr>
<td>r</td>
<td>[a, b]</td>
</tr>
</tbody>
</table>
Tabled Abduction: Motivation & Main Idea

\[ P_1 : \quad q \leftarrow a. \quad r \leftarrow b, q. \quad p \leftarrow r, q. \]

- Abducibles: \{a, b\}
- Query: \(?- q. \quad ?- r. \quad ?- p.\)
  - Explaining \(q\): \([a]\).
  - Explaining \(r\): recompute \(q\)?
  - Explaining \(p\): recompute \(r\) and \(q\)?
- Adopt tabling in LP, for abductive solution reuse
  - Solutions reuse in distinct context!
- Example
  - \(?-q\): table \([a]\) as solution to \(q\).
  - \(?-r\): reuse solution \(q\) with context \([b]\), but
  - \(?-p\): reuse solution \(q\) with \(r\)’s solution \(([a, b])\) as its context.

<table>
<thead>
<tr>
<th>Goal</th>
<th>Solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>(q)</td>
<td>([a])</td>
</tr>
<tr>
<td>(r)</td>
<td>([a, b])</td>
</tr>
<tr>
<td>(p)</td>
<td>([a, b])</td>
</tr>
</tbody>
</table>
Program Transformation: Tabling Solutions

- Table abductive solution entry
  - XSB-Prolog tabling
  - $P_1: \quad q \leftarrow a. \quad r \leftarrow b, q. \quad p \leftarrow r, q.$
  - Table $q^{ab}/1, r^{ab}/1,$ and $p^{ab}/1$

  \[
  \begin{align*}
  q^{ab}([a]). \\
  r^{ab}(E) & \leftarrow q([b], E). \\
  p^{ab}(E) & \leftarrow r([\ ], T), q(T, E). 
  \end{align*}
  \]

- Re-uptake context-independent solution $E$ from “ab” tables into different input contexts $I$

  \[
  \begin{align*}
  q(I, O) & \leftarrow q^{ab}(E), \text{prod}(O, I, E). \\
  r(I, O) & \leftarrow r^{ab}(E), \text{prod}(O, I, E). \\
  p(I, O) & \leftarrow p^{ab}(E), \text{prod}(O, I, E). 
  \end{align*}
  \]

- prod/3: produces consistent abduction result in $O$
Program Transformation: Dealing with “not”

- $P_2 : p \leftarrow a, \text{not } q. \quad q \leftarrow a, b. \quad q \leftarrow c.$
- Abductive solutions of $\text{not } q$
  - Needs to compute all abductive solutions for $q$, before negating them.
- Dual rules for negation, via dual transformation\(^1\)
  - Produce negation rules from the positive ones.
  - Find solutions incrementally.
  - Replace default literal $\text{not } q$ by $\text{not} \_ q$:
    $$p^{ab}(E) \leftarrow \text{not} \_ q([a], E).$$
  - Provide dual rules, e.g., for $\text{not} \_ q$
    $$\text{not} \_ q(I, O) \leftarrow \text{not} \_ q_1(I, T), \text{not} \_ q_2(T, O).$$
    $$\text{not} \_ q_1(I, O) \leftarrow \text{not} \_ a(I, O).$$
    $$\text{not} \_ q_1(I, O) \leftarrow \text{not} \_ b(I, O).$$
    $$\text{not} \_ q_2(I, O) \leftarrow \text{not} \_ c(I, O).$$

Other Aspects

- Dual transformation for programs with variables.
- Dealing with loops (i.e., positive loops and loops over negation) in the presence of abduction.
  - Employs XSB tabling as much as possible, e.g., for positive loops.
  - Pragmatic approaches to deal with positive loops in dualized negation (e.g., $\text{not} \ q \leftarrow \text{not} \ q$) and negative loops over negation.
- Implementation:
  - Transformation of predicates comprised just of facts.
  - Lazy by-need dual transformation by storing dual rules in a trie (available in XSB), in lieu of batched table scheduling.
- Evaluation: benefit of tabling abductive solutions, variants of dual transformation, tabling nogoods of subproblems, programs with loops.
Conclusions and Future Work

- Addressed the issue of tabling abductive solutions.
  - Abductive solution reuse from one context to others.
  - Abduction under negative goals with dual transformation.
  - Various pragmatics to deal with loops and dual transformation.
- Future work:
  - Answer subsumption on abductive solutions.
  - Integrating TABDUAL with program updates (EVOLP/R) and other logic programming features.
  - Application to abductive moral decision making and others.
Thank you!