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:
    - $shoes\_wet \leftarrow grass\_wet$.
    - $grass\_wet \leftarrow sprinkler\_running$.
    - $grass\_wet \leftarrow rained$.
    - $clothes\_wet \leftarrow rained$.
    - $clothes\_dry$.
    - IC: $false \leftarrow clothes\_wet, clothes\_dry$.
  - 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. `?-` 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.

Goal	Solutions
q	$[a]$

Tabled Abduction: Motivation & Main Idea

$$P_1 : \qquad q \leftarrow a. \qquad r \leftarrow b, q. \qquad p \leftarrow r, q.$$

- 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

Goal	Solutions
q	$[a]$
r	$[a, b]$

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
  - `?-`p: reuse solution q with r's solution ($[a, b]$) as its context.

Goal	Solutions
q	$[a]$
r	$[a, b]$
p	$[a, b]$

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 $p^{ab}/1$

$$q^{ab}([a]).$$
$$r^{ab}(E) \leftarrow q([b], E).$$
$$p^{ab}(E) \leftarrow r([\,], T), q(T, E).$$

  - Re-uptake context-independent solution E from "*ab*" tables into different input contexts I

$$q(I, O) \leftarrow q^{ab}(E), prod(O, I, E).$$
$$r(I, O) \leftarrow r^{ab}(E), prod(O, I, E).$$
$$p(I, O) \leftarrow p^{ab}(E), prod(O, I, E).$$

  - $prod/3$: produces consistent abduction result in O

Program Transformation: Dealing with "*not*"

- P_2 : $p \leftarrow a, not\ q.$ $q \leftarrow a, b.$ $q \leftarrow c.$
- Abductive solutions of *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 *not q* by *not_q*:

    $$p^{ab}(E) \leftarrow not\_q([a], E).$$
  - Provide dual rules, e.g., for *not_q*

$$not\_q(I, O) \leftarrow not\_q_1(I, T), not\_q_2(T, O).$$
$$not\_q_1(I, O) \leftarrow not\_a(I, O).$$
$$not\_q_1(I, O) \leftarrow not\_b(I, O).$$
$$not\_q_2(I, O) \leftarrow not\_c(I, O).$$

---

[1] J. J. Alferes, L. M. Pereira, and T. Swift. Abduction in well-founded semantics and generalized stable models via tabled dual programs. *Theory and Practice of Logic Programming*, 4(4):383428, 2004.

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., $not\_q \leftarrow not\_q$) 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!