Joint Tabling of Logic Program Abductions and Updates

Ari Saptawijaya  Luís Moniz Pereira

CENTRIA, Departamento de Informática
FCT, Universidade Nova de Lisboa

Motivation

- Applying abduction to decision making, subject to knowledge updates:
  
  Towards modeling morality computationally with logic programming.
  

- This work stems from our recent works on tabling:
  - Tabled abduction in logic programs
    
  
  - Incremental tabling for query-driven propagation of logic program updates
    
Idea

1. Input Program transforms to Processed Program: contextual abduction, holds-time, dual programs.
2. Pending Updates propagates fluents bottom-up to Actual Updates.
3a. Actual Updates activates Joint Table: fluents + abductive solutions.
3b. Joint Table look-up for query(Q,Sol).
Enabling Abducibles

An abducible $A$ can be assumed only if there is an expectation for it and if, moreover, there is no expectation to the contrary.

$$\text{consider}(A) \leftarrow \text{expect}(A), \ not \ \text{expect\_not}(A), \ A.$$ 

L. M. Pereira et al., *Inspecting and preferring abductive models.*

Rule Name Fluent

Program: \[ q \leftarrow a. \]

is first processed to: \[ q \leftarrow \text{consider}(a). \]

Next, uniquely assign rule name fluent \#r(Head,Body) to turn the rule on/off:

\[ q \leftarrow \#r(q,[a]), \text{consider}(a). \]
Finding Abductive Solutions

1. transforms

Input Program

Processed Program: contextual abduction, holds-time, dual programs

System Predicates

2. Pending Updates

processed propagates
fluent bottom-up

Actual Updates

Joint Table: fluents + abductive solutions

3a activates

? - query(Q,Sol)

3b look-up
Finding Abductive Solutions

Program:  
\[ q \leftarrow a. \]
\[ q \leftarrow \#r(q,[a]), \text{ consider}(a). \]

and next equipped with input and output abductive context args:
Maintaining holds-time

1. Input Program transforms to Processed Program: contextual abduction, holds-time, dual programs

2. Pending Updates propagates fluents bottom-up

3a. Activates

3b. Joint Table: fluents + abductive solutions look-up

?- query(Q,Sol)
Maintaining holds-time

Program:

\[ q \leftarrow a. \]
\[ q \leftarrow \#r(q,[a]), \text{consider}(a). \]

- holds-time \( H_q \) of \( q \): the time when \( q \) is true
- table holds-time \( H_q \) incrementally

\[ \text{fluent} \  \#r(Head,Body) \]
\[ \text{consider}(a) \]

input abductive context \( H_r \)

intermediate abductive context

output abductive context \( H_a \)

\( H_q \)
Tabling Abductive Solutions

\[ \text{consider}(A) \leftarrow \text{expect}(A), \not\text{expect}_\text{not}(A), A. \]

\[ \text{consider}_{ab}(A) \]

- Re-use solution:
  \[ \text{consider}(A,I,O,H) \leftarrow \text{consider}_{ab}(A,E,H), \text{produce}(O,I,E). \]
Dual Program Transformation

1. **Input Program** transforms to **Processed Program**: contextual abduction, holds-time, dual programs

2. Pending Updates propagates fluents bottom-up to **Actual Updates**

3. **Joint Table**: fluents + abductive solutions

4. ?- query(Q,Sol)

   - 3a activates
   - 3b look-up
Dual Program Transformation

- We’ve seen $not\ expect\_not(A)$ becomes ‘positive’:
  
  $not\_expect\_not(A)$

  - Program: $q \leftarrow a, b. \quad q \leftarrow c.$
  - Dual rule: $not\_q \leftarrow q^{*1}, q^{*2}.$
Conclusions

- Implementation techniques jointly combining:
  - tabled abduction: **TABDUAL**
  - incremental tabling of fluents: **EVOLP/R**

- Further work:
  - More experimental evaluation to validate and improve implementation.
  - Concrete application to hypothetical and counterfactual reasoning, namely to moral decision making.
Thank you!