Towards Practical Tabled Abduction Usable in Decision Making

Ari Saptawijaya, Luís Moniz Pereira

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

KES IDT 2013
Sesimbra, 28 June 2013
Abduction (1)

- From observed evidence to its best explanation
- Example
  - Beliefs:
    - The shoes are wet if the grass is wet.
    - The grass is wet if the sprinkler was running.
    - The grass is wet if it rained.
  - Observation
    - The shoes are wet.
  - Abducibles:
    - “The sprinkler was running”,
    - “It rained”.
  - Minimal explanations:
    - “The grass is wet”, or “The grass is wet”, or
    - “The sprinkler was running”, or
    - “It rained”.
Abduction (2)

- Consistent explanations, not necessarily minimal.
- Example
  - Previous beliefs:
    - The shoes are wet if the grass is wet.
    - The grass is wet if the sprinkler was running.
    - The grass is wet if it rained.
  - Plus, new beliefs:
    - The clothes outside are wet if it rained.
    - The clothes are dry.
    - **Integrity Constraint**: Clothes are not both dry and wet.
  - Same abducibles: “The sprinkler was running”, “It rained”
  - Satisfying IC + Observation “The shoes are wet”
  - Single Explanation: The sprinkler was running.
Abductive Logic Programming

- Abduction in Logic Programs
- Example (cont’d)
  - 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 with TABDUAL
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: \texttt{?- q.  ?- r.  ?- p.}
  - Explaining \( q \): [a].
  - Explaining \( r \): recompute \( q \)?
  - Explaining \( p \): recompute \( r \) and \( q \)?
- Adopt \underline{tabling} in LP, for abductive solution reuse
  - Solutions reuse in distinct context!
- Example
  - \texttt{?-q}: table [a] as solution to \( q \).
  - \texttt{?-r}: reuse solution \( q \) with context [b], but
  - \texttt{?-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
  - $P_1 : q ← a. \ r ← b, q. \ p ← r, q.$
  - 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 $I$

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

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

- $P_2 : \quad p \leftarrow a, \text{not } q. \quad q \leftarrow a, b. \quad q \leftarrow c.$

- Abductive solutions of $\text{not } q$
  - Needs to compute all abductive solutions for $q$, before negating them,

- Dual rules for negation, via dual transformation
  - 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 $\text{not}_q$
    \[
    \text{not}_q(I, O) \leftarrow \text{not}_q_1(I, T), \text{not}_q_2(T, O).
    \]
    \[
    \text{not}_q_1(I, O) \leftarrow \text{not}_a(I, O).
    \]
    \[
    \text{not}_q_1(I, O) \leftarrow \text{not}_b(I, O).
    \]
    \[
    \text{not}_q_2(I, O) \leftarrow \text{not}_c(I, O).
    \]
TABDUAL Extensions and Applications in Decision Making
Picking up Abduction-based Actions

- Decision making under hypothetical reasoning
- Given an observation:
  - Several scenarios exist, each characterized by abducibles
  - Decisions are based on explanatory abducibles
- Example:

```
smoke ← fire. smoke ← tear_gas.
beginProlog.
decide(call_firefighters, Abds) ← member(fire, Abds).
decide(police_protection, Abds) ← member(tear_gas, Abds).
endProlog.
```

- Top-goal queries: do(Action, Abducibles, Observation)
  - `?- do(Action, Abducibles, smoke).` 
    - `Action = call_firefighters, Abducibles = [fire];` 
    - `Action = police_protection, Abducibles = [tear_gas]`
Declarative Debugging: Incorrect Solutions

A buggy program $P$:

\[
\begin{align*}
a(1). & \quad a(X) \leftarrow b(X), c(Y, Y). \\
b(2). & \quad b(3). \quad c(1, X). \quad c(2, 2). \\
\end{align*}
\]

- Transformation (inc/2 abducible):

\[
\begin{align*}
a(1) & \leftarrow \text{not } inc(1, [1]). \\
a(X) & \leftarrow b(X), c(Y, Y), \text{not } inc(2, [X]). \\
b(2) & \leftarrow \text{not } inc(3, [2]). \\
b(3) & \leftarrow \text{not } inc(4, [3]). \\
c(1, X) & \leftarrow \text{not } inc(5, [1, X]). \\
c(2, 2) & \leftarrow \text{not } inc(6, [2, 2]). \\
\end{align*}
\]

- IC: $\text{false } \leftarrow a(3)$.

- Solutions:

\[
\begin{align*}
[\text{inc}(2, [3])], [\text{inc}(4, [3])], [\text{inc}(5, [1, 1]), \text{inc}(6, [2, 2])].
\end{align*}
\]
Declarative Debugging: Missing Solutions

The same buggy program $P$:

\begin{align*}
a(1). & \quad a(X) \leftarrow b(X), c(Y, Y). \\
b(2). & \quad b(3). \quad c(1, X). \quad c(2, 2).
\end{align*}

- Transformation (miss/1 abducible):
  Program $P$ plus

\begin{align*}
a(X) & \leftarrow \text{miss}(a(X)). \\
b(X) & \leftarrow \text{miss}(b(X)). \\
c(X, Y) & \leftarrow \text{miss}(c(X, Y)).
\end{align*}

- IC: false $\leftarrow$ not $a(5)$.

- Solutions:
  \[[\text{miss}(a(5))], [\text{miss}(b(5))], [\text{miss}(b(5)), \text{miss}(c(X, X))].\]
A Medical Dental Case

\[
\begin{align*}
\text{percussion\_pain} & \leftarrow \text{periapical\_lesion}. \\
\text{percussion\_pain} & \leftarrow \text{fracture}. \\
\text{radiolucency} & \leftarrow \text{periapical\_lesion}. \\
\text{fracture} & \leftarrow \text{horizontal\_fracture}. \\
\text{elliptic\_fracture\_trace} & \leftarrow \text{horizontal\_fracture}. \\
\text{tooth\_mobility} & \leftarrow \text{horizontal\_fracture}. \\
\text{fracture} & \leftarrow \text{vertical\_fracture}. \\
\text{decompression\_pain} & \leftarrow \text{vertical\_fracture}. \\
\text{false} & \leftarrow \text{not percussion\_pain}. \\
\text{false} & \leftarrow \text{tooth\_mobility}. \\
\end{align*}
\]

\[?
\begin{align*}
\text{fracture}([ \ ], T), \text{not\_false}(T, O). \\
\end{align*}
\]

\[
\begin{align*}
\Rightarrow & \quad O = [\text{horizontal\_fracture}] \times \\
\Rightarrow & \quad O = [\text{vertical\_fracture}] \checkmark \\
\Rightarrow & \quad O = [\text{periapical\_lesion, vertical\_fracture}] \checkmark \\
\end{align*}
\]
Conclusions and Future Work

- Addressed the issue of tabling abductive solutions: TABDUAL
- Improved and extended TABDUAL towards more practical use
  - System predicate for abducible-based actions
  - System predicate for accessing ongoing abductive solutions
  - Other improvements: simpler facts transformation, dual by-need, ...
- Showed declarative debugging as abduction
- Applied TABDUAL to medical diagnosis
- Future work and application:
  - Perfecting implementation
  - Integrating TABDUAL with program updates and other logic programming features
  - Application to abductive moral decision making
Thank you!