Examining the Plausible Side-Effects of Abduction

Luís Moniz Pereira, UNL
Alexandre Miguel Pinto, UC
Outline

- Abduction in Computational Logic / Logic Programming
- Examples of Abductive Logic Programs (ALPs)
- Side-Effects as consequences of Abduction:
  - Computational Problem: Efficiency and Relevancy
  - Solution: Inspection Points (IPs)
  - Examples
  - IPs for enacting preferences
Abduction - 1

- Abduction is well studied in computational logic / logic programming (LP – e.g. Prolog)

- Abduction in LP ➔ declarative formalism to express and solve a variety of problems; e.g. decision-making, diagnosis, planning, belief revision, hypothetical reasoning,…

- Prolog systems are mature and practical ➔ Facilitate use of abduction in Prolog systems
Abduction - 2

- Abduction: choosing hypotheses that explain given observations, or that justify hypothetical conclusions.

- In LPs abductive hypotheses (or *abducibles*) are:
  - literals with no rules
  - truth value not initially assumed
  - may have arguments — but must be ground when abduced

- *Abductive (normal) logic program* = LPs allowing abducibles or their negations in the bodies of rules.

  - The negation ‘*not* a’ of abducible ‘a’ is not its default negation but rather the hypothetical, explicitly assumed, negation of *a*. 
Abduction – example 1

- **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.

- **Two abducibles:**
  - “The sprinkler was running”, “It rained”

- **Minimal explanations:**
  - “The grass is wet”, or % not an abducible
  - “The sprinkler was running”, or
  - “It rained”.
Abduction – example 2

- Solutions are consistent, but not necessarily minimal.
- Prior beliefs of example:
  - 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.
- New added beliefs:
  - The clothes outside are wet if it rained.
  - IC: No clothes are both dry and wet.
- Same abducibles: “The sprinkler was running”, “It rained”
- Satisfy IC + Observation: The shoes are wet.
- Single explanation: “The sprinkler was running”.


Abduction – example 3

- **ALP rules:**
  
  \[
  \begin{align*}
  \text{shoes}_\text{wet} & \leftarrow \text{grass}_\text{wet}. \\
  \text{grass}_\text{wet} & \leftarrow \text{sprinkler}_\text{running}.
  \end{align*}
  \]
  
  \[
  \begin{align*}
  \text{grass}_\text{wet} & \leftarrow rained. \\
  \text{clothes}_\text{wet} & \leftarrow rained.
  \end{align*}
  \]

- **IC:**
  
  \[
  \begin{align*}
  \text{false} & \leftarrow \text{clothes}_\text{wet}, \text{clothes}_\text{dry}. \\
  \text{clothes}_\text{dry}.
  \end{align*}
  \]

- **Abducibles:** \text{sprinkler}_\text{running}, \text{rained}

- **Query:**
  
  \[
  \begin{align*}
  ?- \text{shoes}_\text{wet}, \text{not false}.
  \end{align*}
  \]

- **Abductive solutions:** \text{sprinkler}_\text{running}
Abduction - 3

- Abucibles may be separately assumed true or false, via either their positive or their negated form.

- Abductive solutions to queries are consistent sets of assumed hypotheses – the actually abduced abducibles.

- When these are replaced by their truth values in the program, the solution yields a model which satisfies the Query + ICs – an abductive model.
Besides needing to abductively discover which hypotheses to assume in order to satisfy some condition, we might also want to know about some of the side-effects of the assumptions made.

Usually, we do not wish to know all possible side-effects of our assumptions — some may be irrelevant.

Side-effects inherent in abductive explanations might not all be of equal interest.
Extending abduction -1

- When discovering abductive solutions for given primary observations, check if given secondary observations are also true as deductive consequence of abductive explanations found for primary observations alone:
  - i.e. “are the secondary observations plausible in the abductive context of the primary ones?”

- If wanting to know every possible side-effect, compute whole abductive models for the query.

- Or focus instead just on side-effects specifically indicated by the user — “side-effect checking”.
Relevant side-effects example - 1

Satisfy IC:  ← thirsty, not drink. Abducibles: \textit{drink\_water}, \textit{drink\_beer}

\begin{align*}
\text{wet\_glass} & \leftarrow \text{use\_glass}. \\
\text{drink} & \leftarrow \text{drink\_water}. \\
\text{unsafe\_drive} & \leftarrow \text{drunk}. \\
\text{thirsty}. \\
\end{align*}

Check if get drunk or not (who cares about glass becoming wet?)

Decide possible action (e.g. to drive or not) only \textbf{after} we know which side-effects are true.

No extra IC allowed like:  ← \textit{unsafe\_drive} as it would always impose abducing \textit{not drink\_beer}.

Need for inspection mechanism checking truth value of given side-effect literals (like \textit{drunk}), without further abducing whilst checking.
Contextual Abductive Explanations

Given abductive solution context H, explaining O from background theory B, we define:

E1 is explained by H1 given H and B iff
(H1 is an abductive explanation for E1, given B)
B U H1 |= E1 where B U H1 is consistent,
B |=/= E1,
H1 is contained in H.

E2 is a strict contextual relevant consequence of explained E1 given B iff
(E2 is a side-effect of abductive explanation H1 for E1)
For every H2 such that E2 is explained by H2 given B, there exists H1 such that E1 is explained by H1 given B, H1 contains H2 consistently.

unsafe_drive is a ‘strict contextual relevant consequence’ of drink.
Relevant side-effects example - 2

Abducibles: tear_gas, fire, water_cannon (actions possible)

- police, riot, not contain. police. riot.
- contain ← tear_gas. smoke ← fire.
- contain ← water_cannon. smoke ← inspect(tear_gas).

- Explanations for ‘smoke’: fire or tear_gas thrown by police to contain riot.

- Riot situation much more unlikely than fire: allow tear_gas to be a plausible explanation for ‘smoke’ only if it necessary to assume tear_gas for other reasons. Represent this with ‘inspect(tear_gas)’.

- ‘inspect’ forbids producing further abducibles; only consuming them.

- IC + facts force ‘contain’ true: tear_gas or water_cannon or both must be abduced.

- ‘smoke’ is explained only if tear_gas is abduced to contain riot.
Abduction in LPs is accomplished by a top-down query-oriented procedure. Finds abductive solutions to queries:

- By need, i.e. as abducibles are encountered.
- Solution abducibles are leaves of the procedural query-rooted call-graph: i.e. the graph recursively engendered by procedure calls from literals in bodies of rules to heads of rules.

Pure top-down computation ("backward chaining") is possible only when the underlying semantics is relevant:

- It avoids computing whole models to find answers to queries.
- It suffices to use only rules relevant to a query to find its truth value – those in its procedural call-graph.
Extending Abduction - 3

- Concomitant side-effect checking requires an abductive reasoning mechanism not previously available.

- Inspection Point checking in Abductive Logic Programs:
  - Showed how to employ it to check side-effects of interest (the Inspection Points). Namely to help choose among abductive solutions.
  - Showed how to implement IPs on top of already existing abduction solving systems (ABDUAL and XSB-XASP).
  - Adoptable by other systems.
Agent: how to decide among several alternative courses of action?

*A priori* preferences reduce number of possible actions, but still several (possibly exclusive) may remain.

Make best possible informed decision ➔

- Foresee consequences of actions.
- Choose actions on the basis of consequences: with *a posteriori* preferences.
Most preferred set of consequences ⇔ Course of action to take.

Only consequences relevant to a posteriori preferences need to be calculated: there are virtually infinitely many consequences of an action, most irrelevant to the preference-based decision making.

Consequences may be simply plausible predictions about the state of the world, and observing whether they are verified can eliminate hypothetical scenarios where some decisions would appear only a priori to make sense.
Inspection Points - 3

- Not all consequences are experimentally observable → Inspection Points (IPs) guide experimentation required to decide among competing hypotheses.

- IPs → sift through competing explanations.

- In science, decisive consequences are known as "crucial" side-effects → they can exclude untoward and implausible hypotheses. IPs avoid mandatory generation of complete abductive models to find relevant consequences.
Thank You
Cautious and Brave Reasoning

- Deduction / Abduction
  - “brave reasoning”
  - “cautious reasoning”

- “Brave reasoning” = is there at least one consistent model of the program which entails the query?

- “Cautious reasoning” = do all models of the program entail the query?