Evolution Prospection

Luís Moniz Pereira
Han The Anh

AI Centre – UNL, Portugal
Evolution Prospecting Agents - EPAs

• A self-updating evolving program can look ahead prospectively to its possible future states, and prefer among them, to better satisfy its goals.

• EPAs can look at their possible future evolutions, and prefer on the basis of evolution history too.

• EPAs are implemented in Abdual – an XSB-Prolog system that performs abduction over the Well-Founded Semantics.
Constructs of the EPA system

- Abducibles
- *a priori* or *pre-*preferences
- *a posteriori* or *post-*preferences
- Active goals
- Contextual integrity constraints
- Levels of commitment
Abducibles

• Each program contains a set of available abducibles.

• Abducibles can be assumed, as needed, to provide solution support for queries.

• A considered abducible A may be assumed only if it is expected in the given situation, and if there is no expectation to the contrary:

\[
\text{consider}(A) \leftarrow \text{expect}(A), \quad \text{not} \quad \text{expect\_not}(A), \quad A
\]

where not stands for default negation. Expectations are defined by logic program rules.
**a priori preferences**

- Preferences over abducibles, have the form:
  
  $$a <| b <- L_1, ..., L_n$$

  where:
  
  - $a, b$ are abducibles
  - $L_i$ are domain literals (including preference literals)

- Meaning:
  
  if $b$ is assumed, then $a <| b$

  forces $a$ to be assumed too
**a posteriori** preferences

- Preference over abductive solutions:
  \[ \text{Ai} \ll \text{Aj} \ll \text{holds}_{\text{given}}(\text{Li}, \text{Ai}), \]
  \[ \quad \text{holds}_{\text{given}}(\text{Lj}, \text{Aj}) \]
  where \( \text{Ai}, \text{Aj} \) abductive solutions

- Meaning \( \text{Ai} \) is preferred to \( \text{Aj} \) *a posteriori*,
  where \( \text{Li} \) and \( \text{Lj} \) are *side-effects* of \( \text{Ai} \) and \( \text{Aj} \),
  i.e. with no further abductions.
Active goals

• At each cycle, the agent has a set of active goals it wants to satisfy. An active goal AG is coded by a rule:

\[
on\_observe(\text{AG}) \leftarrow L_1, ..., L_n
\]

meaning “on observing \(L_1, ..., L_n\) trigger goal AG”

• An active goal may be triggered by events, previous commitments, or history-related information.
Contextual integrity constraint - IC

- All integrity constraints coded in the KB must be satisfied. When considering evolving agents, ICs are employed which depend on context (time points, external environment).

- An IC depending on context C is coded by an active goal:
  
  \[
  \text{onobserve}(\text{not } \text{icName}) \leftarrow C \\
  \text{icName} \leftarrow \text{icBody}
  \]

  When C is true, active goal \text{not icName} is launched, forcing the intended IC
  
  \[
  \text{false} \leftarrow \text{icBody}
  \]

  to be satisfied.
Levels of commitment

• Each prospective cycle is completed by committing to preferred remaining ongoing abductive solutions.

• History of evolution is kept by setting time stamps for abducibles committed to.

• Depending on their influence on the future, commitments are of 3 types:
  
  Hard: indefeasible; permanently affect the future

  Ongoing: defeasible; keep on affecting the future till defeated

  Temporary: momentary; affect only the current state
Prospective Agents - PA

- Depending on capabilities and current need, a PA may wish or has to satisfy:

  - only active goals at hand – **single-step PA**

  - also long-term active goals – **multiple-step PA** whose preferences may depend on history
Time-sensitive preferences

• Because the agent is evolving, its preferences may change.

• How they change depends on the evolution point the agent is situated at, or even on the whole evolution history.

• Example: John has lunch everyday, fast food or fruit. His favourite is fast food, to save time for work. He wants not to be fat, and avoids fast food 3 days in a row. After 2 days he prefers fruit instead.
Inevitable choices

• Abducibles belonging to all abductive solutions are said *inevitable*. They are committed to, whatever the final abductive solution may be.

• Committing to abducibles changes the KB and may trigger new preferences. This can make some alternative abductive solutions irrelevant.

• Also, committing to inevitable abducibles may render other abducibles inevitable too – propagation of the inevitable.
Multiple-step prospective agents

• When looking ahead a number of steps into the future, simple *(a posteriori)* local preferences aren’t appropriate enough.

• The agent must be able to prefer amongst possible evolutions by:
  – evaluating their consequences – *a posteriori evolution preferences*,
  – or evaluating their historical information – *evolution history preferences*. 
**a posteriori** Evolution Preferences

• Preferences over evolutions, for quantitatively or qualitatively evaluating their consequences:

\[
E_i \lll E_j \iff \text{holds\_in\_evol}(L_i, E_i), \quad \text{holds\_in\_evol}(L_j, E_j)
\]

where \(E_i, E_j\) – evolutions, \(L_i, L_j\) – domain literals

• Meaning \(E_i\) is preferred to \(E_j\)

  when \(L_i\) and \(L_j\) are *side-effects* of evolutions \(E_i\) and \(E_i\);
  
  i.e. with no any further abductions.
During war time David, a good general, needs to decide to save one of his cities, a or b, from an attack. He does not have enough military resources to save both. If a city is saved, its citizens are saved. A bad general, who just sees the situation at hand, would prefer to save the city with most population, but a good one looks ahead a number of steps into the future, in order to choose the best strategy for the war as a whole.
example (cont.)

Having been scheduled, for the next day, to make a counter-attack on one of two cities of the enemy, either a small or a big city, the prior action of saving a city should take this foreseen future into account.

A successful attack on a small city is always expected, but a successful attack on a big city leads to a better probability of further wins in the war.

It is expected to successfully attack the big city only if the person who knows secret enemy information is alive in the city to be saved.
**a posteriori** vs. evolution **a posteriori**

- simple **a posteriori** preference – single-step PA (bad general)
  \[
  A_i \ll A_j \rightarrow \text{holds\_given}(\text{save\_men}(N_i), A_i), \text{holds\_given}(\text{save\_men}(N_j), A_j), N_i > N_j
  \]

- evolution **a posteriori** preference – multiple-step PA (good general)
  \[
  E_i \ll\ll E_j \rightarrow \text{holds\_in\_evol}(\text{pr\(\text{win}\), \(P_i\)}, E_i), \text{holds\_in\_evol}(\text{pr\(\text{win}\), \(P_j\)}, E_j), P_i > P_j
  \]

- With the 1\text{st} preference a general can attack only the small city, while with the 2\text{nd} he can attack the big city.

- Agents can make more reasonable decisions with evolution **a posteriori** preferences.
Evolution history preferences - 1

- Takes into account commitment history information from the evolutions.
- It can be quantitative (e.g. having maximal or minimal number of some type of commitment) or qualitative (e.g. time order of commitments along an evolution).
- Can be used *a priori* (upon finding possible evolutions) or *a posteriori* (upon interaction with users, if there is more than one evolution left).
Evolution history preferences - 2

- **max(C):** evolutions with maximal # of commitments to C.
- **min(C):** evolutions with minimal # of commitments to C.
- **greater(C, N):** evolutions with # of commitments to C greater than N.
- **times(C, N):** evolutions with exactly N commitments to C.
- **smaller(C, N):** evolutions with # commitments to C smaller than N.
- **next(C1,C2):** evolutions with commitment C1 succeeding C2 in time.
- **prec(C1,C2):** evolutions with commitment C1 preceding C2 in time.
Evolution history preferences - 3

• Having applied all preferences, if there is still more than one possible evolution, interaction mode is entered to ask for the user’s additional preferences, in the form of a list.

• A preference standing before another in the list takes priority.

• If a preference is not satisfied by any evolution, the system skips it and applies the next one.
Future Work - 1

• Agent relaxation by setting a scale of priorities for active goals. Focus on most important ones if not all satisfiable.
  – Implemented by preferring among the literals `on_observe/1` used for representing active goals.

• Agent relaxation by setting a scale of priorities for ICs.
  – Similarly implemented.
Future Work - 2

• Embedding heuristic search algorithms for searching the evolution tree, e.g. best-first search.

• Using multi-threads of XSB
  – Independent threads can evolve on their own, and communicate with one another to decide whether some thread should be canceled or kept evolving, according to search algorithm.