Modelling Probabilistic Causation in Decision Making

Luís Moniz Pereira Carroline Kencana Ramli

Artificial Intelligence Centre — CENTRIA
Universidade Nova de Lisboa
Portugal

Outline

Motivation

Motivation

- Humans reason based on cause and effect expressed as logical rules.
- Causal Bayes Nets help define imperfect regularities.
- We adopt declarative logic programming to model hypothetical causal scenarios, coupled with Bayes Nets to model their uncertainty and utility.

Prospective Logic Programming

Prospective Logic Programming – Background

- Prospective logic programming is an instance of an architecture for causal models.
- The program is capable of conjuring up hypothetical *what-if* scenaria and formulating abductive explanations for both external and internal observations.
- We need preference specifications, which can be:
  - **A priori** preferences
  - **A posteriori** preferences

Language

Definition (Language)

Let $\mathcal{L}$ be a *first order language*. A domain literal in $\mathcal{L}$ is a domain atom A or its default negation *not* A, the latter expressing that the atom is false by default.
A domain rule in $\mathcal{L}$ is of the form:

$$A \leftarrow L_1, \ldots, L_t \ \ (t \geq 0)$$

where A is a domain atom and $L_1, \ldots, L_t$ are domain literals.

Integrity Constraint

Definition (Integrity Constraint)

An *integrity constraint* in $\mathcal{L}$ is of the form:

$$\bot \leftarrow L_1, \ldots, L_t \quad (t > 0)$$

where $\bot$ is a domain atom denoting falsity, and $L_1, \ldots, L_t$ are domain literals.

Logic Program

Definition (Logic Program)

A *logic program* P over $\mathcal{L}$ is a set of domain rules and integrity constraints, standing for all their ground instances.

Abducibles

Definition (Abducibles)

With each P is associated a *set of abducibles* $\mathcal{A}_P \subseteq \mathcal{L}$.
An abducible a can be assumed, when needed by a query, only if it is a considered one, i.e. it is expected in the given situation, and moreover there is no expectation to the contrary.
Atom $consider(a)$ will be true if and only if abducible a is considered:

$$consider(A) \leftarrow expect(A), not\ expect\_not(A)$$

Rules for expectations and counter expectations are domain-specific knowledge that constrain the available hypotheses given some situation.

A Priori Preferring Abducibles

Definition (Relevance Rule)

Let a and b are abducibles. A *relevance atom* $a \triangleleft b$ means abducible a is more relevant or preferred than abducible b, i.e. we cannot have b without also having a.
A *relevance rule* is of the form:

$$a \triangleleft b \leftarrow L_1, \ldots, L_t \ \ (t \geq 0)$$

where $a \triangleleft b$ is a relevance atom and every L_i can be a domain or a relevance literal.

Preferring Abducibles

Example (Claire 1)

Consider Claire drinks either tea or coffee (but not both).
She prefers coffee over tea when she wants to keep awake, and doesn't drink coffee when her blood pressure is high.
If she wants to socialize, she prefers tea over coffee; and prefers socializing over staying wide awake if she's in the mood.
If Claire's sleepy she expects waking up; and expects to socialize if she's in the mood.
This situation is described by program P over $\mathcal{L}$ with abducibles $\mathcal{A}_P = \{tea,\ coffee,\ wake\_up,\ socialize\}$:

Preferring Abducibles

Example (Claire 1)

```
1. drink <- tea.
   drink <- coffee.
2. expect(tea).
   expect(coffee).
   expect(socialize)  <- in_the_mood.
   expect(wake_up)    <- sleepy.
3. expect_not(coffee) <- blood_pressure_high.
4. constrain(0, [tea, coffee], 1).
5. coffee    < tea     <- wake_up.
6. tea       < coffee  <- socialize.
7. socialize < wake_up <- in_the_mood.
```

Preferring Abducibles

Example (Claire 1)

- There are two abductive solutions answering the question "?- drink":
  1. $A_1 = \{coffee\}$
  2. $A_2 = \{tea\}$

Preferring Abducibles

Example (Claire 1)

- There are two abductive solutions answering the question "?- drink":
  1. $A_1 = \{coffee\}$
  2. $A_2 = \{tea\}$
- If we assert *sleepy* is true, the only explanation left is A_1.

Preferring Abducibles

Example (Claire 1)

- There are two abductive solutions answering the question "?- drink":
  1. $A_1 = \{coffee\}$
  2. $A_2 = \{tea\}$
- If we assert *sleepy* is true, the only explanation left is A_1.
- If we assert *blood_pressure_high* is true, the only remaining solution is A_2 because the abducible *coffee* becomes not expected.

$$expect\_not(coffee) \leftarrow blood\_pressure\_high$$

A Posteriori Preferences

- Abduction is a mechanism to conjecture possible states by an agent, each abductive solution representing a hypothetical reachable scenario of interest.
- Preferring over abducibles enacts preferences over the possible hypothetical states of the agent.
- In this context, it is unavoidable to deal with uncertainty, a problem decision theory addresses using probabilities coupled with utility.

A Posteriori Preferences

Example (Claire 3)

Claire is spending a day at the beach and needs to decide the means of transportation.
She knows it is usually faster and more comfortable to go by car but also knows — because it is hot — a traffic jam is likely.
She can take a train, but it will take longer, though it meets her wishes of being more environment friendly.
The situation can be modeled by the abductive logic program:

A Posteriori Preferences

Example (Claire 3)

```
1. hot.
2. go_to(beach) <- car.
   go_to(beach) <- train.
3. expect(car).
   expect(train).
4. constraint(1, [car, train], 1).
5. probability(traffic_jam, 0.7)     <- hot.
   probability(not traffic_jam, 0.3) <- hot.
6. utility(stuck_in_traffic, -8).
   utility(wasting_time, -4).
   utility(comfort, 10).
   utility(environment_friendly, 3).
```

A Posteriori Preferences

Example (Claire 3)

By assuming each of the abductive hypotheses, the general utility of going to the beach can be computed for each particular scenario:

Assume car
Probability of being stuck in traffic = 0.7
Probability of a comfortable ride = 0.3
Expected utility = 10 * 0.3 + 0.7 * -8 = -2.6

Assume train
Expected utility = -4 + 3 = -1

Probabilistic Logic Programming

Probabilistic Logic Programming – Background

- Probabilistic Logic (P-log) is a declarative language that combines logical and probabilistic reasoning, which uses Answer Set Programming (ASP) as its logical foundation and Causal Bayes Nets [Pea00] as its probabilistic foundation.
- First introduced by Chitta Baral et. al [BGR04, BGR09].
- Newer developments of P-log [ARD08] use the XASP package of XSB Prolog for interfacing with an ASP solver.

Attribute

Definition (Attribute)

Let $c_0, c_1, \ldots, c_n$ be sorts. An *attribute* a with the domain $c_1 \times ... \times c_n$ and the range c_0 is represented as follows:

$$a : c_1 \times ... \times c_n \rightarrow c_0$$

If attribute a has no domain parameter, we simply write $a : c_0$. The range of attribute a is denoted by $range(a)$. Attribute terms are expressions of the form $a(\bar{t})$, where a is an attribute and $\bar{t}$ is a vector of terms of the sorts required by a.

Random Selection Rule

Definition (Random Selection Rule)

A *random selection rule* has a form:

$$random(RandomName, a(\bar{t}),\ DynamicRange)\ \texttt{:-}\ Condition$$

Probabilistic Information

Definition (Probabilistic Information)

Information about probabilities of random attribute instances $a(\bar{t}, y)$ taking particular value y is given by probability atoms (or simply pa-atoms) which have the following form:

$$pa(RandomName, a(\bar{t}, y), d\_(A, B)) :- Condition$$

Example (Claire 4)

Suppose we know that the availability of tea for agent Claire is around 60%.

```
1. beginPr.
2.    beverage = {tea, coffee}.
3.    available : beverage.
4.    random(rd, available, full).
5.    pa(rd, available(tea), d_(60, 100)).
6. endPr.
```

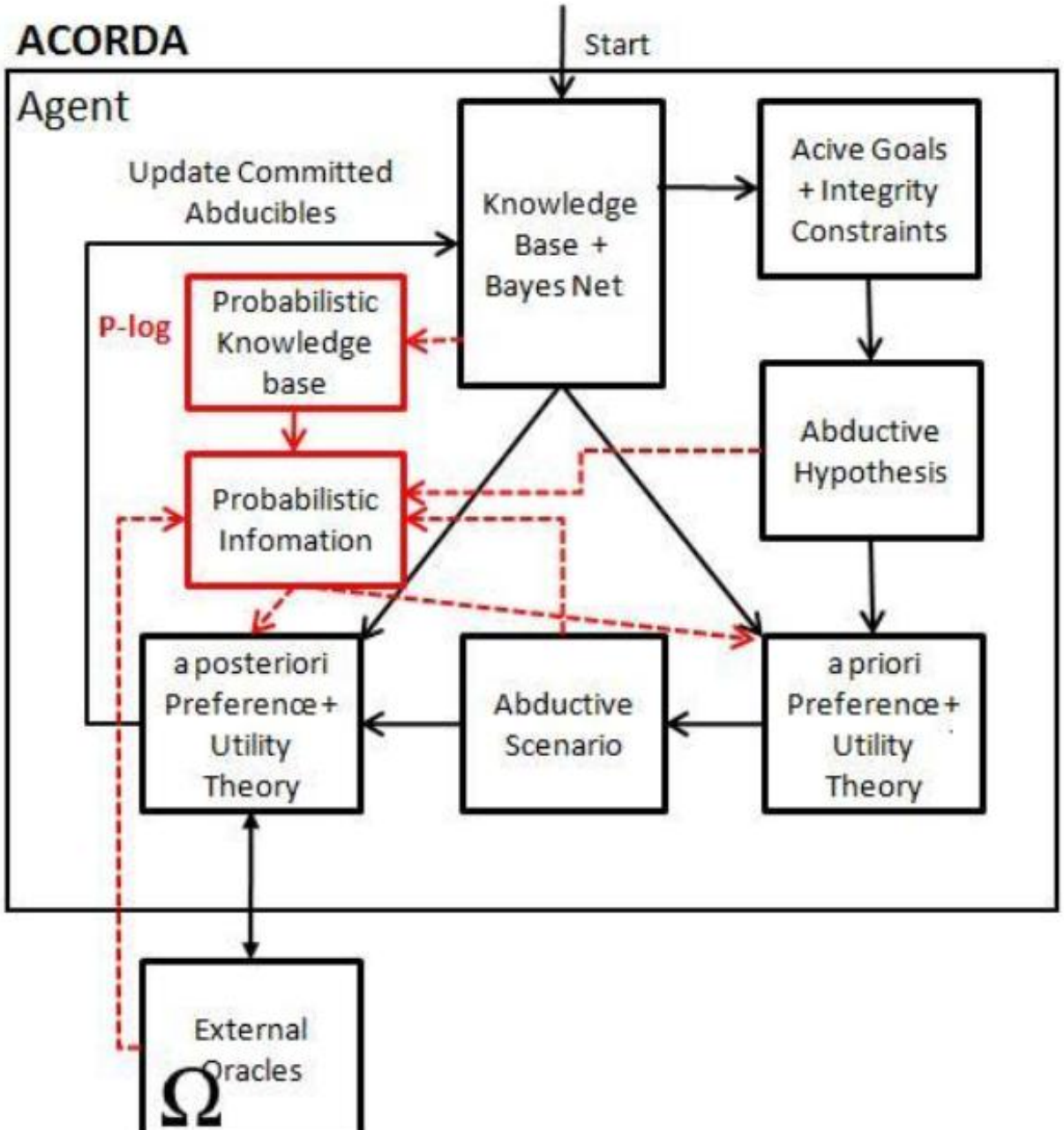
Observations and Actions

Definition (Observations and Actions)

Observations and *actions* are, respectively, statements of the forms $obs(l)$ and $do(l)$, where l is a literal. Observations are used to record the outcomes of random events, i.e. random attributes and attributes dependent on them. The statement $do(a(t, y))$ indicates that $a(t) = y$ is made true as the result of a deliberate (non-random) action.

The statement $obs(available(tea))$ indicates that we want to record the outcome of the availability of tea, on another hand the statement $do(available(tea))$ means tea was simply put on the table in the described action that tea is really available and we have tea already.

Integration Architecture



ACORDA – A Posteriori Preference

Example (Claire 5)

Consider now we introduce the utility rate for coffee and tea for agent Claire based on her preference. For the first time, the utility rate of tea is 80% and the utility rate of coffee is 70%. What do we suggest for agent Claire's beverage when the utility probability value is taken into account?

ACORDA – A Posteriori Preference

- First, ACORDA acquires information about Claire's situation. If there is no contrary expectation for any of the beverages, we will have two different abductive solutions:
  1. $M_1 = \{coffee\}$,
  2. $M_2 = \{tea\}$.
- Next, ACORDA performs a posteriori selection.
- After the computation we get the result:
  1. $M_1 = \{utilityModel(0.2800),\ coffee\}$,
  2. $M_2 = \{utilityModel(0.4800),\ tea\}$.
- Selection will retain highest utility model and the final result is $M_2 = \{utilityModel(0.4800),\ tea\}$ meaning that based on the a posteriori preference using our utility function, **agent Claire is encouraged to have tea**.

Example – Risk Analysis

Example (Mamma Mia)

The owner of Restaurant "Mamma Mia" considers offering a new menu.
Before offering it he performed some research.
And found 75% of teenagers prefer the new menu and 80% of adults prefer the original one.
Around 40% of his costumers are teenagers.
If he offers the new menu, each will return 5 Euros.
The cost he will incur for serving 100 new menus is 200 Euros.
What is your suggestion to the owner of Restaurant "Mamma Mia"?
The owner's utility function is represented by
$U(X) = 2 * X - 0.01X^2$, $(X \leq 100)$, X being his income.

Conclusion and Future Work

Conclusion and Future Work

- ACORDA + P-log = causal models + Bayes Nets
- For future work, we can extend P-log to handle stochastic processes.
- We can use PRISM [Sat95, SK08] ideas to expand the semantics of P-log to allow infinite possible worlds.

References I

H. T. Anh, C. D. P. K. Ramli, and C. V. Damásio.

An implementation of extended p-log using xasp.

In *Proceedings of 24th International Conference on Logic Programming*, LNCS 5366, pages 739–743. Springer, 2008.

C. Baral, M. Gelfond, and N. Rushton.

Probabilistic reasoning with answer sets.

In *LPNMR7*, LNAI 2923, pages 21–33, 2004.

C. Baral, M. Gelfond, and N. Rushton.

Probabilistic reasoning with answer sets.

Theory and Practice of Logic Programming, pages 57–144, January 2009.

J. Pearl.

Causality: Models, Reasoning, and Inference.

Cambridge University Press, 2000.

References II

T. Sato.

A statistical learning method for logic programs with distribution semantics.

In *Proceedings of the 12th International Conference on Logic Programming (ICLP95)*, pages 715–729. MIT Press, 1995.

T. Sato and Y. Kameya.

New advances in logic-based probabilistic modeling by prism.

In *Probabilistic Inductive Logic Programming, LNCS 4911*, pages 118–155. Springer, 2008.