Combining Probabilistic and Logical Reasoning

Michael Gelfond

Texas Tech University

Joint work with Chitta Baral and Nelson Rushton

The Goal

Find a knowledge representation language allowing natural, elaboration tolerant representation of commonsense knowledge involving logic and probability.

Success Criteria

The language should

- **allow elegant formalizations of non-trivial combinations of logical and probabilistic reasoning,**
- **help the language designers (and hopefully others) to better understand the meaning of probability and probabilistic reasoning,**
- **help to design and implement knowledge based software systems.**

Selecting the Logic

ASP - logic programs under the answer set semantics.

- **A non-monotonic logic with a high degree of elaboration tolerance**
- **Capable of representing and reasoning with defaults, causal relations, recursive definitions, various forms of incomplete information, etc.**
- **Has efficient inference engines.**

Modeling Probability

- **Probability is interpreted as a measure of the strength of beliefs of a rational reasoner.**
- **In purely probabilistic domains the corresponding probability distributions are given by Pearl's causal Bayesian networks.**

Monty Hall Problem in P-log

A player selects one of three closed doors, behind one of which there is a prize.

After selection is made, Monty is obligated to open one of the remaining doors which does not contain the prize.

The player can switch his selection to the other unopened door, or stay with his original choice.

Does it matter if he switches?

The Game's Rules in P-log

Declarations:

$select, prize, open : \{1, 2, 3\}.$

Regular Part:

$\neg can\_open(D) \leftarrow\ select = D.$

$\neg can\_open(D) \leftarrow\ prize = D.$

$can\_open(D) \leftarrow\ not\ \neg can\_open(D).$

Probabilistic Part:

$random(prize).$

$random(select).$

$random(open : \{X : can\_open(X)\}).$

The Semantics

The semantics of a P-log program Π is given by

- **Translating Π into an ASP program $\tau(\Pi)$.**
- **Viewing answer sets of $\tau(\Pi)$ as** *possible worlds* **of Π.**
- **Defining probability measure on possible worlds of Π.**

Translation

Rule

$$random(open : \{X : can\_open(X)\})$$

is translated into

$$open = 1 \lor open = 2 \lor open = 3 \leftarrow$$
$$not\ do(open = y)$$

and

$$\leftarrow open = D, \neg can\_open(D)$$

Possible Worlds and their Measures

Π has 12 possible worlds, e.g.

$$W_1 = \{\ prize = 1, select = 1, open = 2,$$

$$can\_open(2), can\_open(3)\quad\}$$

$$W_2 = \{prize = 2, select = 1, open = 3, can\_op(3)\}$$

Since all the choices were equally likely

$$\hat{\mu}(W_1) = 1/3 \times 1/3 \times 1/2 = 1/18$$

$$\hat{\mu}(W_2) = 1/3 \times 1/3 \times 1 = 1/9$$

Specifying Probabilistic Information

The above program has no explicit probabilistic information and so the possible results of each random selection are assumed to be equally likely.

If we learn that given a choice between opening doors 2 and 3, Monty opens door 2 four times out of five, we add:

$$pr(open = 2 \mid_c can\_open(2), can\_open(3)) = 4/5$$

These statements form Causal Bayesian Net and are used in the definition of measure.

Making the Decision

Consider a scenario:

$S = \{obs(select = 1),\ obs(open = 2),\ obs(prize \neq 2)\}$

$P_\Pi(prize = 1 \mid S) = P_{\Pi \cup S}(prize = 1)) = 1/3$

$P_\Pi(prize = 3 \mid S) = P_{\Pi \cup S}(prize = 3)) = 2/3$

It certainly makes sense to switch.

Distinctive Features of P-log

- P-log probabilities are defined with respect to an explicitly stated knowledge base.
- In addition to logical non-monotonicity P-log is "probabilistically non-monotonic" – addition of new information can add new possible worlds and substantially change the original probabilistic model.
- Possible updates include defaults, rules introducing new terms, observations, and deliberate actions in the sense of Pearl.

ASP knowledge base

P-log probabilities are defined with respect to an explicitly stated knowledge base written in ASP.

This KB can be viewed as a specification for the sets of beliefs which could be held by a rational reasoner associated with it. The reasoner believes, disbelieves, or remains undecided about f.

P-log expands ASP by allowing reasoning about degrees of such beliefs.

Non-monotonicity

Let Π consists of rules

$a : boolean$

$a \leftarrow \ not\ b$

$random(a) \leftarrow b$**.**

Clearly

$P_\Pi(a) = 1$

but

$P_\Pi(a|b) = P_{\Pi\cup\{b\}}(a) = 1/2$

New information creates new possible worlds and hence allows to reason about change of probabilistic models.

Observations and Interventions

New observation, $a = y$, is added to P-log KB as $obs(a = y)$, which is translated by τ into the rule

$\leftarrow obs(a = y), a \neq y$**.**

This removes possible worlds in which $a \neq y$.

Deliberate assignment of y to a is written as $do(a = y)$.

This statement stops a from being random.

Rat Example

Program, T, represents knowledge about whether a certain rat will eat arsenic today, and whether it will die today.

$arsenic, death : boolean.$

$random(arsenic).$

$random(death).$

$pr(arsenic) = 0.4.$

$pr(death \mid_c arsenic) = 0.8.$

$pr(death \mid_c \neg arsenic) = 0.01.$

Semantics of the *pr* atoms expresses that the rat's consumption of arsenic carries information about the cause of his death (as opposed to, say, the rat's death being informative about the causes of his eating arsenic).

A consequence of this reading is that seeing the rat die raises our suspicion that it has eaten arsenic, while killing the rat (say, with a pistol) does not affect our degree of belief that arsenic has been consumed.

Formally,

$P_T(arsenic) = 0.4$

$P_T(arsenic|obs(death)) = 0.8421$

$P_T(arsenic|do(death)) = 0.4$

Propositions relevant to a cause give equal evidence for the attendant effects whether they are forced to happen or passively observed.

$P_T(death) = 0.38$

$P_T(death|do(arsenic)) = 0.8$

$P_T(death|obs(asenic)) = 0.8$

Conclusions

- **P-log was used for formalizing a substantial number of non-trivial reasoning examples. The results so far are satisfying.**

- **We started using P-log for design and implementation of software systems (e.g. we can find most probable diagnoses for the reactive control system of the space shuttle).**

Need to combine probabilistic and ASP reasoning algorithms!