Weighted Answer Set Programs

Francisco, Bruno, Salvador, Dietmar

November 5, 2024

Abstract

Drawing inspiration from HMMs; State of the art of PLP and Probabilistic ASP; Leveraging current Prob ASP systems;

Using a logic program to model and reason over a real world situation is often complicated because of uncertainty underlying the problem being worked on. Classic logic programs represent knowledge in precise terms, which turn out to be problematic when the application data is affected by stochastic factors, which is frequently the case.

1 Uncertainty and ASPs

Hidden Markov models (HMMs) setup a well known, widely used, framework to describe and study partially observable stochastic processes[1]. Briefly, a HMM process is described by a sequence of states, each defined as a set of random variables that are partially observed. This means that (1) observation of states include only a fixed subset of those variables, and also that (2) the observed values are subject to random perturbations.

[1] HMMs stress the **process** and we are not looking into that.

We try to capture both the partial observability and the stochastic nature of a system in a logic setting. To this end we think of a system, S, as a formal model, P, together with a set of observations, D:

$$S = (P, D) \tag{1}$$

The formal model is an answer set program extended with *weights*, that we describe later in this paper, and such that the stable models of the program are the system states. On the other hand, any given system state can be (or has been) observed. The outcome of each single observation is a set of literals of that program.

Example 1 (Coins, Dices, and Cards) *Consider a scenario where a coin is tossed and, if the result is* heads, a*, then a dice is thrown,* b*, or a card is drawn,* c*. If the coin lands in* tails, $\neg a$*, no information is given. A formal model of this system is the program*

$$P_{\text{sbf}} = \begin{cases} a \vee \neg a, \\ \quad b \vee c \leftarrow a \end{cases} \tag{2}$$

that has atoms $\{a, b, c\}$*, literals* $\{a, \neg a, b, \neg b, c, \neg c\}$ *and stable models*

$$\{a, b\}, \{a, c\}, \{\neg a\}. \tag{3}$$

Possible observations include:

- $\{a, c\}$*: a stable model (SM), therefore a system state, observed* heads *and* card.
- $\{a\}$*: not a SM but contained in the two SMs* $\{a, b\}$ *and* $\{a, c\}$*, observed* heads *but we have no information about the* cards *or the* dice.
- $\{\neg a, \neg b\}$*: not a SM but contains the SM* $\{\neg a\}$*, the coin was observed as* not heads *(*tails*?); also we are certain that no card was drawn but have no information about the dice.*
- $\{b, c\}$*: not related with any SM.*
- $\{a, \neg a\}$*: an inconsistent observation.*

We use sets of literals, instead of atoms, to represent observations because they make possible the distintion between a *explicit negative observation* $\neg\alpha$ (*e.g.*"I see tails.") and *not observed* $\sim\alpha$ (*e.g.*"The coin in hidden."). This corresponds to assuming a "boolean sensor" for each α and $\neg\alpha$.

So, the program 2 defines the "sensors"

$$a, \neg a, b, \neg b, c, \neg c$$

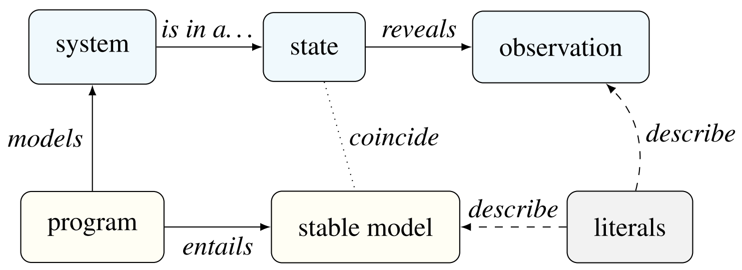
system
is in a...
state
reveals
observation
models
coincide
describe
program
entails
stable model
describe
literals

Figure 1: