Modeling Markov Decision Processes with Imprecise Probabilities Using Probabilistic Logic Programming

Thiago P. Bueno **Denis D. Mauá** **Leliane N. de Barros**

Fabio G. Cozman

Universidade de São Paulo, Brazil

ISIPTA 2017

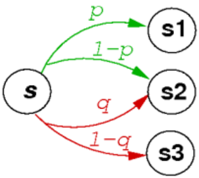
Goals

- To introduce a modeling language that can capture Markov Decision Processes with Imprecise Probabilities (MDPIPs),
- by employing Probabilistic Logic Programming (PLP).

Markov Decision Processes

A Markov Decision Process (MDP) consists of:

- a set of **states** $\mathcal{S}$;
- a set of **actions** $\mathcal{A}(s)$ for each state s;
- a **transition model** $\mathbb{P}(s'|s,a)$ specifying the probability of next state s' after executing action a in state s;
- a **reward model** $R(s,a,s')$ specifying the reward (or cost) of executing action a in state s and transitioning to state s';
- a set of decision stages $D = 1, ..., H$.



Optimal policy, optimal value function

- The **solution** of an MDP with infinite horizon (i.e., $H \to \infty$) is a stationary, deterministic **optimal policy** $\pi^* : \mathcal{S} \to \mathcal{A}(s)$ that maximizes
$$\sum_{t=0}^{\infty} \gamma^t R(s_t, a, s_{t+1}).$$

- The optimal policy produces the **optimal value function** $V^* : \mathcal{S} \to \mathbb{R}$ satisfying the Bellman equation
$$V^*(s) = \max_{a \in \mathcal{A}(s)} \left\{ \sum_{s' \in \mathcal{S}} \mathbb{P}(s'|s, a)(\mathcal{R}(s, a, s') + \gamma \; V^*(s')) \right\},$$

Markov Decision Processes with Imprecise Probabilities

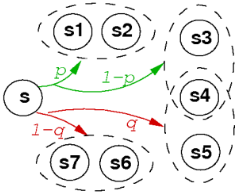
- Suppose there is a **set of probabilities** modeling each state transition.
- These sets are referred to as **transition credal sets** $\mathcal{K}(\cdot|s,a)$.
- The **Γ-maximin criterion** selects a policy such that

$$V^*(s) = \max_{a\in\mathcal{A}(s)} \left\{ \min_{\mathbb{P}(\cdot|s,a)\in\mathcal{K}(\cdot|s,a)} \sum_{s'\in\mathcal{S}} \mathbb{P}(s'|s,a)(\mathcal{R}(s,a,s')+\gamma\, V^*(s')) \right\}.$$

MDPSTs

- A **Markov Decision Process with Set-valued Transition** (MDPST) is a special MDPIP.

- After applying action a to state s, the probability that the next state s' is in the **reachable set** $k \in F(s, a)$ is given by $m(k|s, a)$.



F(s,a1) = {{s1, s2},{s3, s4}}

- Policy is obtained by simplified equation:

$$V^*(s) = \max_{a \in \mathcal{A}(s)} \left\{ \sum_{k \in F(s,a)} m(k|s,a) \min_{s' \in k} \left(R(s,a,s') + \gamma \; V^*(s') \right) \right\} .$$

Languages

- There are languages to specify MDPs; several combine logical expressions with probabilities.
  - The PPDDL language can even encode MDPSTs.
  - But not intuitive at all.

Probabilistic Logic Programming

- A **probabilistic logic program** is a pair $L_p = \langle \mathbf{BK}, \mathbf{PF} \rangle$ where:
  - **BK** is a set of logical rules, and
  - **PF** is a set of *independent* probabilistic facts.

Probabilistic Logic Programming

- A **probabilistic logic program** is a pair $L_p = \langle \mathbf{BK}, \mathbf{PF} \rangle$ where:
  - **BK** is a set of logical rules, and
  - **PF** is a set of *independent* probabilistic facts.
- A logical rule is of the form

$$\mathsf{h_1; ...; h_l :- b_1, ..., b_m, \mathbf{not}\ b_{m+1}, ..., \mathbf{not}\ b_n.}$$

- A probabilistic fact is denoted $\alpha :: \mathsf{f}$. where f is an atom annotated with probability $\alpha \in [0, 1]$.

Example: Viral Marketing

$$0.2 :: \text{buy\_from\_marketing}(\mathit{Person}).$$
$$0.3 :: \text{buy\_from\_trust}(\mathit{Person}).$$

$$\begin{aligned}
\text{buys}(\mathit{Person}) :- \ & \text{buy_from_marketing}(\mathit{Person}). \\
\text{buys}(\mathit{Person}) :- \ & \text{buy_from_trust}(\mathit{Person}), \\
& \text{trusts}(\mathit{Person}, \mathit{Person}2),\ \text{buys}(\mathit{Person}2).
\end{aligned}$$

$$\text{trusts}(\mathit{alice}, \mathit{eve}).\ \text{trusts}(\mathit{eve}, \mathit{bob}).$$

- What is the probability of Alice buying the product?

$$\mathbb{P}(\text{buys}(\mathit{alice})) = ?$$

Example: Viral Marketing (continued)

$$\begin{aligned}
&\ldots \\
0.15 &:: \text{invited_party}(\mathit{Person}). \\
\\
\text{buys}(\mathit{Husband}) &:- \text{invited_party}(\mathit{Husband}), \\
&\quad\ \text{married}(\mathit{Husband}, \mathit{Wife}),\ \textbf{not}\ \text{buys}(\mathit{Wife}). \\
\text{buys}(\mathit{Wife}) &:- \text{invited_party}(\mathit{Wife}), \\
&\quad\ \text{married}(\mathit{Husband}, \mathit{Wife}),\ \textbf{not}\ \text{buys}(\mathit{Husband}). \\
\\
&\quad\ \text{married}(\mathit{alice}, \mathit{bob}).
\end{aligned}$$

- How to compute the probability of $\mathbb{P}(\text{buys}(\mathit{alice}))$ now?
  In some situations, there is more than a (stable) model...

Credal Semantics

- Propositional probabilistic facts $\alpha_i :: f_1$, $\alpha_2 :: f_2$, etc.
- Each total choice of probabilistic facts has probability

$$\prod_{f_i \in \theta} \alpha_i \prod_{f_i \notin \theta} (1 - \alpha_i) \; .$$

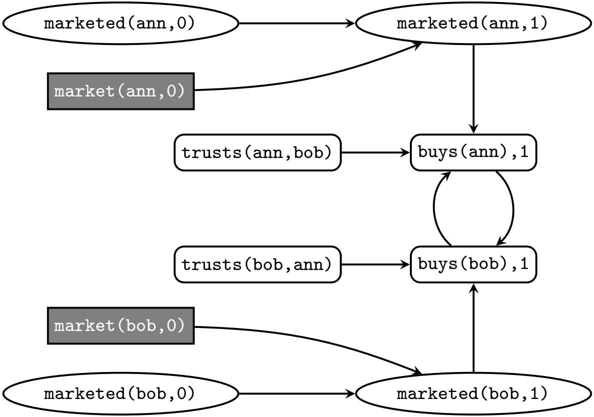
  - But some total choices may produce more than one stable model!
- Credal semantics of a program is the set of all joint distributions that can be produced this way.
  - Important: this set is the dominating set of an infinitely monotone Choquet capacity (!).

A PLP-based Language to Specify MDPIPs

- We need to extend the ProbLog language to define:
  - special-purpose predicates for state variables and actions;
  - syntax and semantics for specifying the transition function; and
  - the dependencies of reward function and its utility attributes.

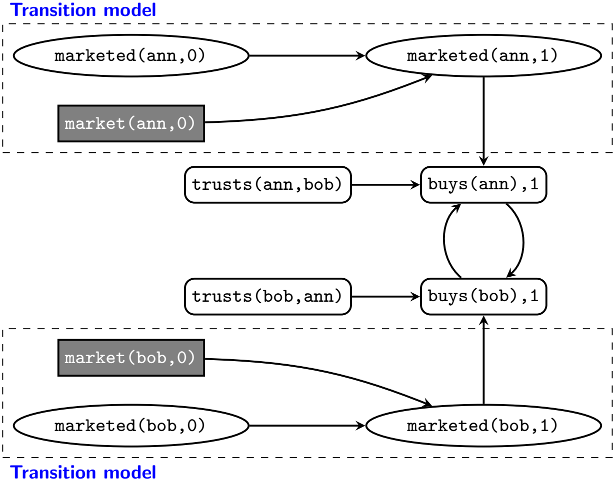
- An **MDP-ProbLog program** consists of three parts:
  (i) a program $L_{\mathrm{MDP}}^{\mathrm{SPACE}}$ declaring state variables and actions;
  (ii) a program $L_{\mathrm{MDP}}^{\mathrm{TRANSITION}}$ encoding a transition model; and
  (iii) a program $L_{\mathrm{MDP}}^{\mathrm{REWARD}}$ encoding the reward model

Viral Marketing (revisited)



$$\text{state\_fluent}(\text{marketed}(P)) \;{:}{-}\; \text{person}(P).$$
$$\text{state\_fluent}(\text{buys}(P)) \;{:}{-}\; \text{person}(P).$$
$$\text{action\_fluent}(\text{market}(P)) \;{:}{-}\; \text{person}(P).$$

Viral Marketing (revisited)

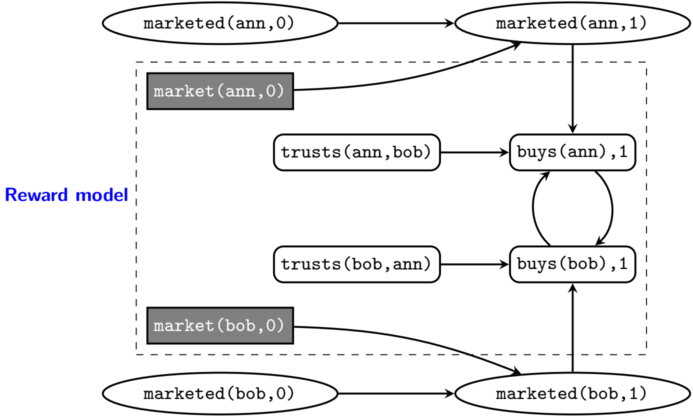


$$0.5 :: \text{forget}(Person).$$

$$\text{marketed}(Person,\ 1) :- \text{market}(Person).$$

$$\text{marketed}(Person,\ 1) :- \textbf{not}\ \text{market}(Person),\ \text{marketed}(Person,\ 0),\ \text{forget}(Person).$$

Viral Marketing (revisited)



$$0.2 :: \text{buy\_from\_marketing}(Person). \quad 0.3 :: \text{buy\_from\_trust}(Person).$$

$$\text{buys}(Person,\ 1) :- \text{marketed}(Person,\ 1),\ \text{buy\_from\_marketing}(Person).$$

$$\text{buys}(Person,\ 1) :- \text{trusts}(Person,\ Person2),\ \text{buys}(Person2,\ 1),\ \text{buy\_from\_trust}(Person).$$

$$\text{utility}(\text{buys}(Person, 1),\ 5).\ \text{utility}(\text{market}(Person),\ -1).$$

A result, and an extension

Theorem
An MDP-ProbLog program specifies an MDPST.

- Now suppose there is indeterminacy on probability values.
- For instance,

$$[0.1, 0.3] :: \text{buy\_from\_marketing}(\mathit{Person}).$$

Complexity of One-Step Inference

- If we have the state at time t, then what is the computational cost of computing the upper probability of $\{X_{t+1} = x\}$?
- More precisely: what is the cost of deciding whether $\overline{\mathbb{P}}(\mathbf{Q}|\mathbf{E}) > \gamma$? (Note: reject if $\overline{P}(\mathbf{E}) = 0$...)
- As input, a program with a bound on predicate arity, and the elements of the query.

Complexity of One-Step Inference

Theorem

Deciding one-step inference is an $\mathsf{NP}^{\mathsf{PP}}$*-complete problem.*

Theorem

Deciding one-step inference when all probabilities are point-valued is $\mathsf{PP}^{\Sigma_3^P}$*-complete problem.*

Conclusion

- Main goal: to introduce a language that can specify MDPIPs and MDPSTs by combining probabilities with logic programming.
- Besides the language, main contribution is complexity analysis for one-step inference.
- In the paper, a discussion of dynamic programming algorithm to build Γ-maximin policies.

- Thanks to CNPq and FAPESP for support.