Specifying Credal Sets With Probabilistic Answer Set Programming

Denis Mauá and **Fabio Cozman**

11/07/2023

Center for Artificial Intelligence

USP

Authors

Denis D. Mauá

Associate Professor - Department of Computer Science

Institute of Mathematics and Statistics - USP

ddm@ime.usp.br

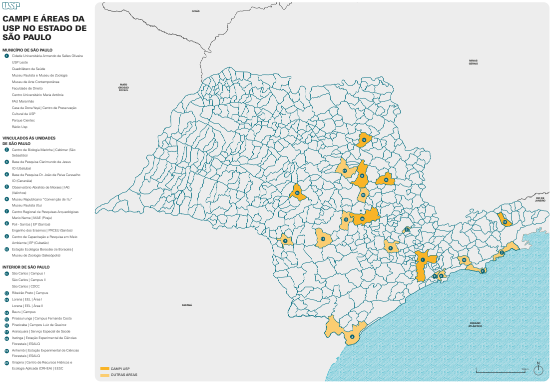
Fabio G. Cozman

Full Professor - Director of C4AI

Escola Politécnica - USP

fgcozman@usp.br

The University of São Paulo



- Largest Public State University of Brazil’s largest state (about 40 million people)
- 11 campi distributed in 7 cities
- ∼60k undergrad students
- ∼30k grad students
- ∼5k faculty

Probabilistic Answer Set Programming

specify **probabilistic knowledge** with **recursive definitions**, **constraints** and **relations**

Logic Programming + Constraint Satisfaction + Uncertain Reasoning
= **Probabilistic Answer Set Programming**

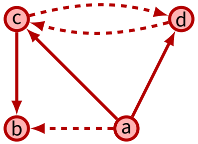
- Generic **inference and learning** routines readily available
- extendable to Neural-Symbolic Reasoners (see dPASP system)
- Key features: Declarative syntax and clear semantics
- Interesting application domain: Argumentation under uncertainty (e.g. driving public discourse on climate change).

Probabilistic Answer Set Programming

Semantics

- Nondisjunctive acyclic programs ⇒ **Bayesian networks**
- Nondisjunctive stratified programs ⇒ **cyclic graphical models**
- Nonstratified or disjunctive programs ⇒ **belief functions**

```
0.3::a.
c :- not d.   d :- not c.
c :- a.       d :- a.
b :- a.       b :- not a, c.
```

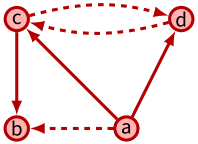


Probabilistic Answer Set Programming

Semantics

- Nondisjunctive acyclic programs ⇒ **Bayesian networks**
- Nondisjunctive stratified programs ⇒ **cyclic graphical models**
- Nonstratified or disjunctive programs ⇒ **belief functions**

```
0.3::a.
c :- not d.   d :- not c.
c :- a.       d :- a.
b :- a.       b :- not a, c.
```



This work: How to extend to more general imprecise probability models?

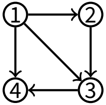
Answer Set Programming

is a powerful declarative language to describe **NP-hard combinatorial problems**

Example: 3-coloring a 4-node graph

Answer Set Programming

is a powerful declarative language to describe **NP-hard combinatorial problems**



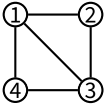
Example: 3-coloring a 4-node graph

```
% --- FACTS ---
% graph has 4 nodes
node(1). node(2). node(3). node(4).
% and the following edges
edge(1,2). edge(2,3). edge(3,4). edge(1,4). edge(1,3).
```

stable models

Answer Set Programming

is a powerful declarative language to describe **NP-hard combinatorial problems**



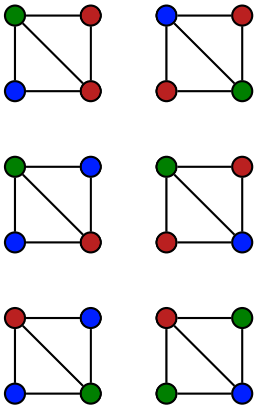
Example: 3-coloring a 4-node graph

```
% --- FACTS ---
% graph has 4 nodes
node(1). node(2). node(3). node(4).
% and the following edges
edge(1,2). edge(2,3). edge(3,4). edge(1,4). edge(1,3).
% --- CONSTRAINTS ---
% graph is undirected
edge(X,Y) :- edge(Y,X).
% adjacent nodes must be colored differently
conflict(X,Y) :- not conflict(X,Y), edge(X,Y), color(X,C), color(Y,C).
```

stable models

Answer Set Programming

is a powerful declarative language to describe **NP-hard combinatorial problems**



stable models

Example: 3-coloring a 4-node graph

```
% --- FACTS ---
% graph has 4 nodes
node(1). node(2). node(3). node(4).
% and the following edges
edge(1,2). edge(2,3). edge(3,4). edge(1,4). edge(1,3).
% --- CONSTRAINTS ---
% graph is undirected
edge(X,Y) :- edge(Y,X).
% adjacent nodes must be colored differently
conflict(X,Y) :- not conflict(X,Y), edge(X,Y), color(X,C), color(Y,C).
% --- CHOICES ---
% a node must have at least 1 of 3 colors
color(X,red); color(X,blue); color(X,green) :- node(X).
```

Probabilistic Answer Set Programming

extends ASP with independent **probabilistic choices**, to encode uncertain knowledge

Sato's Distribution Semantics

Each **probabilistic choice** is associated with a Categorical **random variable**

A realization of the probabilistic choices generates an ASP program

Probabilistic Answer Set Programming

extends ASP with independent **probabilistic choices**, to encode uncertain knowledge

Sato's Distribution Semantics

Each **probabilistic choice** is associated with a Categorical **random variable**

A realization of the probabilistic choices generates an ASP program

Random graph example: probabilistic program

node(1). node(2). node(3). 0.5::edge(1,2). 0.5::edge(2,3).

generates four programs, with probability $1/4$ each:

node(1). node(2). node(3).	node(1). node(2). node(3). edge(1,2).
node(1). node(2). node(3). edge(2,3).	node(1). node(2). node(3). edge(1,2). edge(2,3).

Neural Answer Set Programming

The dPASP Framework[1]

Probabilistic choices can arise from outputs of neural network classifiers

Example: Parsing arithmetic expressions

```
digit(1). digit(2). ... digit(10).
% neural atom: takes image I and produces probability of class 1 to 10
?::digit(I,[1,...,10]) :- image(I).
% arithmetic operations
add(I1,X) :- digit(I2,Y), digit(I2,Z), X = Y + Z.
subtract(I1,X) :- digit(I2,Y), digit(I2,Z), X = Y − Z.
multiply(I1,X) :- digit(I2,Y), digit(I2,Z), X = Y * Z.
```

[1] https://kamel.ime.usp.br/dpasp

Probabilistic Answer Set Programming

Semantics

$$\Pr(\text{atom}) = \sum_{\text{program}} \Pr(\text{program}) \sum_{\substack{\text{model:}\\ \text{atom} \in \text{model}}} \Pr(\text{model} \mid \text{program})$$

Probabilistic Answer Set Programming

Semantics

$$\Pr(\text{atom}) = \sum_{\text{program}} \Pr(\text{program}) \sum_{\substack{\text{model:}\\ \text{atom} \in \text{model}}} \Pr(\text{model} \mid \text{program})$$

- **Acyclic/Stratified:** one model for induced (det.) program
  - $\Pr(\text{model} \mid \text{program}) = 1$

Probabilistic Answer Set Programming

Semantics

$$\Pr(\text{atom}) = \sum_{\text{program}} \Pr(\text{program}) \sum_{\substack{\text{model:} \\ \text{atom} \in \text{model}}} \Pr(\text{model} \mid \text{program})$$

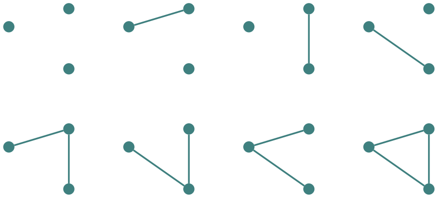
- **Acyclic/Stratified:** one model for induced (det.) program
  - $\Pr(\text{model} \mid \text{program}) = 1$
- **Disjunctive/Nonstratified:** credal set of all distributions $\Pr(\text{model} \mid \text{program})$
  - Defines **belief function** $\underline{\Pr}(\cdot)$ over interpretations through probability distribution $\Pr(\text{program})$ and the multivalued mapping program $\mapsto$ models

Probabilistic Answer Set Programming

Semantics

Example: reachability in 3-node random graph

```
node(1). node(2). node(3).
0.5::edge(X,Y) :- node(X), node(Y), X < Y.
edge(X,Y) :- edge(Y,X).
% Definition of reachable
reachable(X,Y) :- edge(X,Y).
reachable(X,Y) :- reachable(X,Z), edge(Z,Y).
```



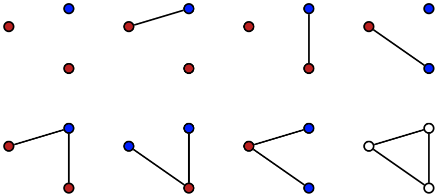
$$\Pr(\text{rechable}(1,3)) = \sum_{\text{program}} \frac{1}{2^3} \times 1 \times [\![\text{reachable(1,3)?}]\!] = \frac{5}{8}$$

Probabilistic Answer Set Programming

Semantics

Example: 2-colorability of random graph

```
node(1). node(2). node(3).
0.5::edge(X,Y) :- node(X), node(Y), X < Y.
edge(X,Y) :- edge(Y,X).
conflict :- edge(X,Y), color(X,C), color(Y,C).
color(X,red); color(X,blue).
colorable :- not conflict.
```



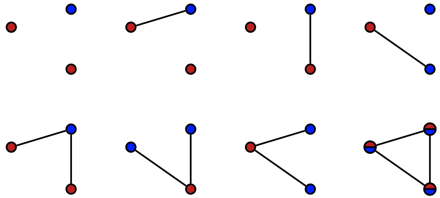
$$\overline{\Pr}(\text{colorable}) = \sum_{\text{program}} \frac{1}{2^3} \times \overline{\Pr}(\text{colorable}|\text{program}) = \frac{7}{8}$$

Probabilistic Answer Set Programming

Semantics

Example: 2-colorability of random graph (with saturation)

```
node(1). node(2). node(3).
0.5::edge(X,Y) :- node(X), node(Y), X < Y.
edge(X,Y) :- edge(Y,X).
conflict :- edge(X,Y), color(X,C), color(Y,C).
color(X,red) :- conflict, node(X).
color(X,blue) :- conflict, node(X).
color(X,red); color(X,blue).
```



$$\Pr(\text{not conflict}) = \sum_{\text{program}} \frac{1}{2^3} \times \Pr(\text{not conflict} \mid \text{program}) = \frac{7}{8}$$

This Work: *Extended* Probabilistic Answer Set Programming

Objective: Specify set of PASP programs, by varying probability annotations of choices

Interval-valued PASP

```
[0.1,0.3]::red(X); [0.2,0.4]::green(X); [0.4,0.6]::blue(X) :- node(X).
```

Parametrized PASP

```
W::win(X); D::draw(X); L::loose(X) :- match(X), W > D, W > L, L <= 0.3.
```

Extended Answer Set Programming

Semantics

$$\Pr(\text{atom}) = \sum_{\text{program}} \Pr(\text{program}) \sum_{\substack{\text{model:}\\ \text{atom}\in\text{model}}} \Pr(\text{model} \mid \text{program})$$

Extended Answer Set Programming

Semantics

$$\Pr(\text{atom}) = \sum_{\text{program}} \Pr(\text{program}) \sum_{\substack{\text{model:}\\ \text{atom}\in\text{model}}} \Pr(\text{model} \mid \text{program})$$

- **Acyclic/Stratified:** credal set of all distributions $\Pr(\text{program} \mid \text{program})$, still one model for each induced (det.) program
  - $\Pr(\text{atom})$ is imprecise

Extended Answer Set Programming

Semantics

$$\Pr(\text{atom}) = \sum_{\text{program}} \Pr(\text{program}) \sum_{\substack{\text{model:}\\ \text{atom} \in \text{model}}} \Pr(\text{model} \mid \text{program})$$

- **Acyclic/Stratified:** credal set of all distributions $\Pr(\text{program} \mid \text{program})$, still one model for each induced (det.) program
  - $\Pr(\text{atom})$ is imprecise
- **Disjunctive/Nonstratified:** credal set of all distributions $\Pr(\text{program} \mid \text{program})$, for each we have a credal set of all distributions $\Pr(\text{model} \mid \text{program})$

Expressivity I

Theorem (Standard PASP capture belief functions)

*Every **infinitely monotone lower probability over a finite domain** can be specified by a probabilistic answer set program with precise probabilities in size proportional to the number of focal sets of its m-function characterization.*

Theorem (Interval-Valued PASP capture finite credal sets)

*Every **finitely-generated credal set over a finite domain** can be represented by an acyclic and positive probabilistic logic program with a single vacuous interval-valued annotated disjunction and a set of precise annotated disjunction.*

Expressivity II

Theorem (Interval-Valued PASP)

Any interval-valued probabilistic answer set program with interval-valued annotated disjunctions can be converted into an equivalent program containing only interval-valued probabilistic facts (Bernoulli vars) and non-probabilistic rules. If the original program is acyclic (resp., nondisjunctive), the resulting program is also acyclic (resp., nondisjunctive).

Theorem

The semantics of an ***acyclic parametrized probabilistic answer set program*** *is given by a credal network; if only probabilistic facts and nonprobabilistic rules appear, the network structure is the dependency graph of the program.*

Complexity

Inference

Compute $\underline{\Pr}(a|b)$ by GBR (solve for μ using binary search):

$$\min_{\Pr} \Pr(a, b) - \mu \Pr(b) = 0 \Leftrightarrow \min_{\Pr} \Pr(a, b) + \mu \Pr(\neg b) = \mu$$

augment program with

query :- a,b. μ::query :- not a, b.

Theorem

*Deciding whether $\underline{\Pr}(atom) \geq \gamma$ is NP^{PP}-**complete** in both interval-valued and parametrized probabilistic answer set programs.*

Extended Probabilistic Answer Set Programming

Conclusions

- Probabilistic Answer Set Programming captures belief functions
- **This work:** Extend language to capture any finite credal set
- Interval-valued PASP implemented in dPASP
- **Challenge:** Probabilistic inference is *too* costly
  - Needs approximate inference algorithms
- **Application:** Connection with imprecise Neural Networks

IME

UNIVERSIDADE DE SÃO PAULO
ESCOLA POLITÉCNICA

Center for
Artificial
Intelligence

USP