Probabilistic Logic Programming under the L-Stable Semantics

Denis D. Mauá[1] Fabio G. Cozman[2] Alexandro Garces[3]

[1]Institute of Mathematics and Statistics
University of São Paulo

[2]Escola Politécnica
University of São Paulo

[3]MIT

22nd International Workshop on Nonmonotonic Reasoning – Nov 2nd, 2024

Roadmap

1. Motivation
2. L-Stable Semantics
3. Probabilistic Answer Set Programming
4. New Complexity Results
5. Inference

Motivation: Inconsistencies in Probabilistic Logic Program

- Probabilistic Answer Set Program (PASP) eases the specification of intricate discrete statistical models involving relations, logical constraints, context-specific indepedence
- Interesting approach for Neurosymbolic Reasoning (probabilities are output of neural concept learners)
- Most semantics require consistency (no world is inconsistent)
- Knowledge base construction often produces inconsistencies (multiple experts, learned rules, unwanted worlds, etc)
- L-Stable Semantics gracefully handles inconsistencies while preserving essence of Distribution Semantics (Independent Probabilistic Facts + Logical Rules)
- This Work: Complexity and Inference for PASP under L-Stable semantics

Answer Set Programming

An answer set program is a finite set of extended disjunctive rules:

head1; head2; ...; headM :- pbody1, ..., pbodyN, not nbody1, ..., not nbodyO.

- rule is a fact if body is empty
- program is normal if very rule has one atom in head
- we disallow integrity constraints (i.e., empty head rules)

Example: 3-coloring

```
% – FACTS
node(1). node(2). node(3). node(4). edge(1,2). edge(2,3). edge(3,4). edge(1,4). edge(1,3).
% – NORMAL RULE
conflict(X,Y) :- not conflict(X,Y), edge(X,Y), color(X,C), color(Y,C).
% – DISJUNCTIVE RULE
color(X,red); color(X,blue); color(X,green) :- node(X).
```

L-Stable Semantics For Propositional Programs

- Interpretation assigns true/false/undefined to each atom (total if no undefined atom)
- Rule is satisfied if body is false, if body and head are true, or if body and head are undefined
- A model satisfies all rules
- Partial stable model if minimal model of program reduct (replace false/true/undefined literals in bodies with false/true/undefined)
- Partial stable model is least undefined (L-Stable) if there is no partial stable model defining more atoms

L-Stable Semantics

```
a; b.
a :- not a.
b :- not b.
```

id	$I(a), I(b)$	$P/I - \{a; b\}$	MinModels(P/I)
1	(false, false)	$a \leftarrow$ true. $b \leftarrow$ true.	(true, true)
2	(false, undef)	$a \leftarrow$ true. $b \leftarrow$ undef.	(true, undef)
3	(false, true)	$a \leftarrow$ true. $b \leftarrow$ false.	(true, false)
4	(undef, false)	$a \leftarrow$ undef. $b \leftarrow$ true.	(undef, true)
5	(undef, undef)	$a \leftarrow$ undef. $b \leftarrow$ undef.	(true, undef), (undef, true)
6	(undef, true)	$a \leftarrow$ undef. $b \leftarrow$ false.	(true, false), (undef, true)
7	(true, false)	$a \leftarrow$ false. $b \leftarrow$ true.	(false, true)
8	(true, undef)	$a \leftarrow$ false. $b \leftarrow$ undef.	(true, undef), (false, true)
9	(true, true)	$a \leftarrow$ false. $b \leftarrow$ false.	(true, false), (false, true)

Probabilistic Answer Set Programming

Distribution Semantics [Dantsin 1990, Poole 1993, Fur 1995, Sato 1995]

- Probabilistic Choices: Collection of fully independent categorical RV's
- Each realization generates an ASP program

Probabilistic Answer Set Programming

Distribution Semantics [Dantsin 1990, Poole 1993, Fur 1995, Sato 1995]

- Probabilistic Choices: Collection of fully independent categorical RV's
- Each realization 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 $0.5 \times 0.5 = 0.25$ 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).
```

Probabilistic Answer Set Programming Under L-Stable Semantics

Example: 2-colorability of random graph

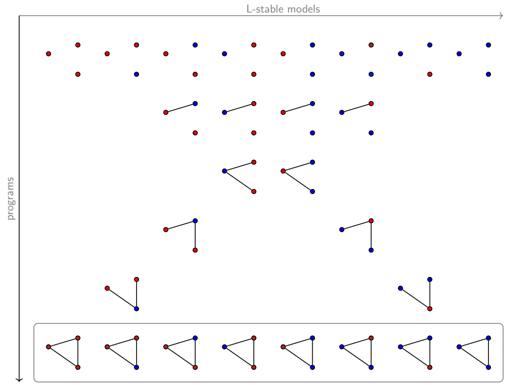
```
% graph has 3 nodes...
node(1). node(2). node(3).

% and random edges.
0.5::edge(X,Y) :- node(X), node(Y), X < Y.

% color each node either red or blue
color(X,red); color(X,blue) :- node(X).

% such that no two endpoints have same color
conflict(X,Y) :- edge(X,Y), color(X,C), color(Y,C).

% graph is colorable iff no conflict
conflict :- not conflict, conflict(X,Y).
colorable :- not conflict.
```



$$\Pr(\text{colorable}=1) = \sum_{\text{program}} 0.5^3 \times \Pr(\text{colorable}=1|\text{program}) = 1-0.5^3$$

Probabilistic Answer Set Programming: Probabilistic Semantics

$$\boxed{\Pr(\text{atom}) = \sum_{\text{program}} \underbrace{\Pr(\text{program})}_{\substack{\text{Distribution}\\\text{Semantics}}} \sum_{\text{model} \models \text{atom}} \Pr(\text{model} \mid \text{program})}$$

Probabilistic Answer Set Programming: Probabilistic Semantics

$$\boxed{\Pr(\text{atom}) = \sum_{\text{program}} \underbrace{\Pr(\text{program})}_{\text{Distribution Semantics}} \sum_{\text{model} \models \text{atom}} \Pr(\text{model} \mid \text{program})}$$

- Stratified: each induced program has exactly one model
  - $\Pr(\text{model} \mid \text{program}) = 1$
- Consistent: induced programs have 1 or more models
  - Credal Semantics: consider the *credal set* of all distributions $\Pr(\text{model} \mid \text{program})$
  - MaxEnt: consider uniform $\Pr(\text{model} \mid \text{program}) = 1/\#\text{models}(\text{program})$
- Plingo/LP$^{\text{MLN}}$: Renormalize over consistent programs

Credal Semantics: Properties

- P1:

$$\min \Pr(\text{atom}) = \sum_{\substack{\text{program:} \\ \forall \text{model} \models \text{atom}}} \Pr(\text{program})$$

- P2:

$$\max \Pr(\text{atom}) = 1 - \min \Pr(\neg \text{atom})$$

- P3:

$$\min \Pr(a|b) = \frac{\min \Pr(a, b)}{\min \Pr(a, b) + \max \Pr(\neg a, b)}$$

Example: Credal and MaxEnt Semantics

0.1::a. 0.3::b.

c :- a. d :- b. c ; d. c :- not c. d :- not d.

	C	Pr(program(C))	LSModels(program(C))
1	$\emptyset$	0.63	(false, false, undef, true), (false, false, true, undef)
2	a	0.07	(true, false, true, undef)
3	b	0.27	(false, true, undef, true)
4	a, b	0.03	(true, true, true, true)

- Credal Semantics yields $\Pr(c) \in [0.1, 0.73]$
- MaxEnt Semantics yields $\Pr(c) = 0.415$
- Both semantics assign $\Pr(\text{not } c) = 0$

Inference Complexity under Stable Semantics

- Credal Semantics: $\min \Pr(\text{atom}|\text{evidence})$
  - $\mathsf{PP}^{\Sigma_2^p}$-hard for propositional programs with disjunction and negation/aggregates
  - $\mathsf{PP}^{\mathsf{NP}}$-hard for disjunction-free, aggregate-free propositional programs
  - PP-hard for stratified propositional programs
  - One step higher in Counting Hierarchy for relational programas with bounded-arity predicates (e.g., $\mathsf{PP}^{\Sigma_3^p}$-hard for disjunctive programs)
  - EXPTIME if predicate arity is unbounded
- MaxEnt Semantics: $\Pr(\text{atom}|\text{evidence})$
  - PP-hard, from stratified programs

Inference Complexity under Credal L-Stable Semantics

Theorem (Mauá & Cozman, 2020)

If consistency in some logic programming language belongs to complexity class C, then probabilistic inference under the credal semantics in the corresponding probabilistic logic programming language belongs to PP^C.

- Consistency in propositional normal programs is Σ_2^p-complete [Eiter, Leone & Saccà 1998]
- Consistency in propositional disjunctive programs is Σ_3^p-complete [ELS 98]

Contributions: Complexity under Credal L-Stable Semantics

Theorem

Deciding if there is an L-stable model satisfying a given atom for a normal program with bounded-arity predicates is Σ_3^p-hard.

Proof. Reduction from 3-QBF with least undefinedness encoding boolean quantifier.
$\exists X_1, \ldots X_m \forall X_{m+1}, \ldots, X_n \exists X_{n+1}, \ldots, X_p \, \phi(X_1, \ldots, X_p)$,

Theorem

Deciding if there is an L-stable model satisfying a given atom for a disjunctive program with bounded-arity predicates is Σ_4^p-hard.

Proof. Reduction from 4-QBF using saturation and least undefinedness.

Contributions: Complexity under Credal L-Stable Semantics

Corollary

Probabilistic inference in propositional disjunctive programs under the credal L-stable semantics predicates is $PP^{\Sigma_3^p}$-complete.

Corollary

Probabilistic inference in normal probabilistic programs with bounded-arity predicates under the credal L-stable semantics is $PP^{\Sigma_3^p}$-complete.

Corollary

Probabilistic inference in disjunctive probabilistic programs with bounded-arity predicates under the credal L-stable semantics is $PP^{\Sigma_4^p}$-complete.

Contributions: Complexity under MaxEnt L-Stable Semantics

- Theorem of Mauá & Cozman 2021 doesn't apply for MaxEnt semantics (AFIK)
- Hence need of more direct proof
- Lack of inner decision problem makes result challenging

Contributions: Complexity under MaxEnt L-Stable Semantics

- Theorem of Mauá & Cozman 2021 doesn't apply for MaxEnt semantics (AFIK)
- Hence need of more direct proof
- Lack of inner decision problem makes result challenging

Theorem

Deciding whether the probability of an atom exceeds a given threshold under the MaxEnt semantics for propositional disjunctive programs is PP-complete when the L-stable models of any induced logic program are efficiently enumerated.

Proof. Hardness follows from stratified programs. Membership: Use no. of models to build Turing machine.

Contributions: Complexity under MaxEnt L-Stable Semantics

Theorem

Deciding if the probability of an atom exceeds a given threshold under the MaxEnt semantics for disjunctive programs with bounded-arity predicates is in PP^{PP}.

Proof. Build Turing machine and count with precision proportional to input size. Use integer gap to decide.

Theorem

Deciding if the probability of an atom exceeds a given threshold under the MaxEnt semantics for propositional normal programs is PP^{NP}-hard, even if all atoms are defined.

Proof. Reduction from MAJ-E-SAT using integer gap in count to decide E-SAT part.

Probabilistic Inference under the L-Stable Semantics

Janhunen et al. 2006's Translation from L-stable to Stable

```
a ; b.
a :- not a.
b :- not b.
```

```
a ; b.          _a ; _b.
a :- not _a.   _a :- not a.   _a :- a.
b :- not _b.   _b :- not b.   _b :- b.
```

id	$I(a), I(\underline{a}), I(b), I(\underline{b})$	$P/I - P'$	MinModels(P/I)
3	(false, false, true, true)	$a \leftarrow$ true. $\underline{a} \leftarrow$ true. $b \leftarrow$ false. $\underline{b} \leftarrow$ false.	(true, true, false, false)
6	(false, true, true, true)	$a \leftarrow$ false. $\underline{a} \leftarrow$ true. $b \leftarrow$ false. $\underline{b} \leftarrow$ false.	(true, true, false, false), (false, true, true, true)
7	(true, true, false, false)	$a \leftarrow$ false. $\underline{a} \leftarrow$ false. $b \leftarrow$ true. $\underline{b} \leftarrow$ true.	(false, false, true, true)
8	(true, true, false, true)	$a \leftarrow$ false. $\underline{a} \leftarrow$ false. $b \leftarrow$ false. $\underline{b} \leftarrow$ true.	(true, true, false, true), (false, false, true, true)
9	(true, true, true, true)	$a \leftarrow$ false. $\underline{a} \leftarrow$ false. $b \leftarrow$ false. $\underline{b} \leftarrow$ false.	(true, true, false, false), (false, false, true, true)

Note: atom is undefined iff x and $\underline{x}$ disagree

Experiments

We have implemented the inference algorithm in our comprehensive neuro-probabilistic-logic framework called dPASP:

`http://github.com/kamel-usp/dpasp`

Inconsistent example program:

```
person(1..4).
0.1::asthma(X).          0.3::stress(X).
0.3::influences(1,2).    0.6::influences(2,1).
0.2::influences(2,3).    0.7::influences(3,4).
0.9::influences(4,1).
0.6::inh_stress(X).      0.6::inh_smokes(X).
smokes_pos(X) :- stress(X), not inh_stress(X).
asthma(X) :- smokes(X), not inh_smokes(X).
smokes_pos(X) :- influences(Y,X), smokes(Y).
smokes_neg(X) :- asthma(X).
smokes(X) :- smokes_pos(X), not smokes_neg(X).
```

Semantics	Pr(smokes(X) = undef)			
	1	2	3	4
smProbLog	0.2223	0.2223	0.2223	0.2223
L-stable	0.1548	0.0828	0.0599	0.0909

- L-Stable semantics is less undefined
- Preserves independence of probabilistic choices

Probabilistic Logic Programming under the L-Stable Semantics

Denis D. Mauá, Fabio G. Cozman and Alexandro Garces

Email: ddm@ime.usp.br