

Elements of Weighted ASP

Francisco Coelho^{1,2}

Bruno Dinis³

Salvador Abreu¹

Dietmar Siepel⁴

¹NOVA LINCS & Departamento de Informática, Universidade de Évora

²High Performance Computing Chair

³CIMA & Departamento de Matemática, Universidade de Évora

⁴Universität Würzburg, Germany

SLATE'25

July 3, 2025

Why weighted answer-set programs?

To **combine** logic and statistics.

What WASPs do?

Propagate weights from atoms to events.

How is propagation done?

Relating atoms, stable models and events.

What are the **results**?

Illustrate & prototype weight propagation.

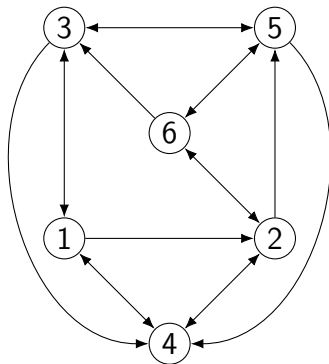
Combine logic and statistics

- **State of the art** of Probabilistic Logic Programming.
- Current **limitations**.
- **Proposed** framework.

We view **logic programs** as **representation of knowledge**.

- **Models** of such programs represent **possibilities**.
- Often our knowledge is **probabilistic**.

An Example of Logic Programming



```
node(1..6).
```

```
edge(1,2). edge(2,4). edge(3,1).
edge(4,1). edge(5,3). edge(6,2).
edge(1,3). edge(2,5). edge(3,4).
edge(4,2). edge(5,4). edge(6,3).
edge(1,4). edge(2,6). edge(3,5).
edge(5,6). edge(6,5).
```

```
col(r). col(b). col(g).
```

```
1 { color(X,C) : col(C) } 1 :- node(X).
:- edge(X,Y), color(X,C), color(Y,C).
```

```
#show color/2.
```

```
color(2,b) color(1,g) color(4,r) color(3,b) color(5,g) color(6,r)
color(1,r) color(2,b) color(4,g) color(3,b) color(5,r) color(6,g)
color(1,r) color(2,g) color(4,b) color(3,g) color(5,r) color(6,b)
color(1,b) color(2,g) color(4,r) color(3,g) color(5,b) color(6,r)
color(2,r) color(1,g) color(4,b) color(3,r) color(5,g) color(6,b)
color(2,r) color(1,b) color(4,g) color(3,r) color(5,b) color(6,g)
```

State of the art of PLP

Syntax extension

- Probabilistic fact: $a : p$.

A **total choice** is a subset of a_i from the probabilistic facts $\{a_1 : p_1, \dots, a_n : p_n\}$.

Distribution semantics

$$P(T) = \prod_{a_i \in T} p_i \times \prod_{a_i \notin T} (1 - p_i)$$

A PLP where the probabilistic facts are replaced by a total choice is a standard logic program.

Probabilistic logic systems

- LP^{MLN} is the most general framework for PLP.
- Problog, P-log are other relevant PLP systems.

Current limitations

Burning questions:

- How to **assign probability** to the models of a program?
- How to deal with *a-posteriori* data?

Currently:

- Probability results from the **syntax** of a program.
- **A-priori assumptions** must be made for the models of such programs.

Instead, we propose to:

- 1 Use **parameters** to avoid *a-priori* assumptions.
- 2 Use **observations** to estimate those parameters.

Proposed framework

	<i>Representation</i>	<i>Example</i>
System	WASP	$\left\{ \begin{array}{l} a : 0.3, \\ b \vee c \leftarrow a \end{array} \right.$
State	Stable model	ab
Observation	Boolean Variables	$\sim a, b, \neg c$

- **Systems** are represented by **weighted ASPs**.
- **States** of the system are the **stable models** of the ASP.
- **Observations (events)** result from **boolean variables**.
- **But** sensors may induce **stochastic perturbations**.

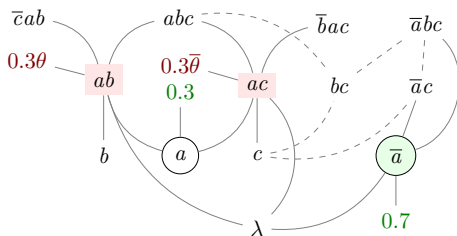
Propagate weights from atoms to events

- From **total choices** to **stable models**.
- From **stable models** to **events**.
- Probabilities.

From total choices to stable models

One total choice can entail **many stable models**.

$$\begin{aligned} & \begin{cases} a : \mathbf{0.3}, \\ b \vee c \leftarrow a \end{cases} \\ & \mathcal{T} = a, \bar{a} \\ & \Downarrow \text{derive} \\ & \begin{cases} a \vee \neg a, \\ b \vee c \leftarrow a \end{cases} \\ & \mathcal{S} = \bar{a}, ab, ac \end{aligned}$$



Parameters $\theta_{s,t}$
represent **lack of**
knowledge...

$$\omega_{\mathcal{S}}(s, t) := \begin{cases} \theta_{s,t} & \text{if } s \in \mathcal{S}(t) \\ 0 & \text{otherwise.} \end{cases}$$

s.t. $\sum_{s \in \mathcal{S}(t)} \theta_{s,t} = 1$

... to be **estimated**
from **data**.

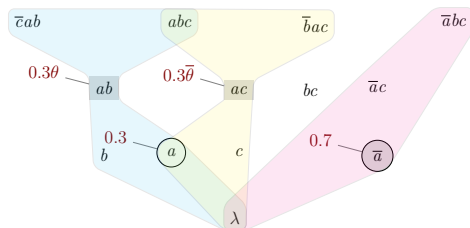
From stable models to events

Observations (events) reduce **uncertainty**.

$$\begin{cases} a : 0.3, \\ b \vee c \leftarrow a \end{cases}$$

$$\mathcal{T} = a, \bar{a}$$

$$\mathcal{S} = \bar{a}, ab, ac$$



Equivalence classes for events are based on \subseteq, \supseteq relations with the **stable models**.

$$[e] := \{ s \in \mathcal{S} \mid s \subseteq e \vee e \subseteq s \}$$

$$\omega_{\mathcal{E}}(e) := \frac{1}{\#[e]_{\sim}} \sum_{t \in \mathcal{T}} \sum_{s \in \mathcal{S}(t) \cap [e]} \omega_{\mathcal{T}}(t) \theta_{s,t}$$

Probabilities

Probabilities result from **normalization** of $\omega_{\mathcal{E}}$.

$$Z := \sum_{e \in \mathcal{E}} \omega_{\mathcal{E}}(e)$$

$\llbracket e \rrbracket$	\perp	\diamond	\bar{a}	ab	ac	\bar{a}, ab	\bar{a}, ac	ab, ac	Λ
$P_{\mathcal{E}}(e)$	0	0	$\frac{7}{207}$	$\frac{1}{23}\theta$	$\frac{1}{23}\bar{\theta}$	0	0	$\frac{3}{46}$	$\frac{10}{23}$

The θ can now be **estimated** from a dataset D of events by

$$\hat{\theta} := \arg_{\theta} \min \left\{ \sum_{e \in D} |P_{\mathcal{E}}(e \mid \theta) - P_D(e)| \right\}$$

Illustrate & prototype weight propagation

- A theoretical **example**.
- Prototype **results** and current state.
- Present & future **development**.

A theoretical example

$$P(B = b) = 0.001$$



$$P(E = e) = 0.002$$



$$P(A | B, E)$$



$$P(M | A)$$



$$P(J | A)$$

$P(M A)$	m	\bar{m}
a	0.9	0.1
\bar{a}	0.05	0.95

$P(J A)$	j	\bar{j}
a	0.7	0.3
\bar{a}	0.01	0.99

$P(A B, E)$		a	\bar{a}
b	e	0.95	0.05
b	\bar{e}	0.94	0.06
\bar{b}	e	0.29	0.71
\bar{b}	\bar{e}	0.001	0.999

$$\left\{ \begin{array}{ll} b : 0.001, & e : 0.002, \\ a : 0.95 \leftarrow b \wedge e, & a : 0.94 \leftarrow b \wedge \bar{e}, \\ a : 0.29 \leftarrow \bar{b} \wedge e, & a : 0.001 \leftarrow \bar{b} \wedge \bar{e} \\ m : 0.9 \leftarrow a, & m : 0.05 \leftarrow \bar{a}, \\ j : 0.7 \leftarrow a, & j : 0.01 \leftarrow \bar{a}, \end{array} \right.$$

Prototype results and current state

Pain points: Computation of $\#[e]_{\sim}$ and Z .

OBSERVATIONS

<-a>	p_D: 0.3230
<a, c>	p_D: 0.1630
<a, b>	p_D: 0.5140

OPTIMIZATION Terminated(SolverConverged)

epsilon: 0.0100; Domain penalty: 100.0000
theta: [0.6330, 0.3569, 1.0031]

COSTS Target: 0.2690; Domain: 0.0000; Total: 0.2696

MODELS

<a, c>	w_E: 0.3976	p_E: 0.0264	p_D: 0.1630
<-a>	w_E: 0.3010	p_E: 0.0200	p_D: 0.3230
<a, b>	w_E: 0.5264	p_E: 0.0349	p_D: 0.5140

*Output from a **generalization** (next slide).*

Present & future development

Generalize $\llbracket e \rrbracket$ to $f(e, s)$ so that

$$w_{\mathcal{E}}(e) = \sum_{t \in \mathcal{T}} \sum_{s \in \mathcal{S}(t)} f(e, s) \theta_{s,t} \omega_{\mathcal{T}}(t)$$

which resembles *Bayes' Rule*:

$$P(E = e) = \sum_{t \in \mathcal{T}} \sum_{s \in \mathcal{S}(t)} \underbrace{P(E = e \mid S = s)}_{\text{sensor model}} \underbrace{P(S = s \mid T = t) P(T = t)}_{\text{system model}}$$

- Compute the **most probable state** given an observation:

$$P(S = s \mid E = e).$$

- Include more elements of **full ASP**.

- Many steps are **parallelizable**.
- **Score** a WASP with

$$\sum_{e \in D} |P_{\mathcal{E}}(e \mid \theta) - P_D(e)|.$$

Thank You!

Questions?

Assumptions of Machine Learning

- Each instance is described in a **single row** by a **fixed set of features**

\mathbf{x}_1	\mathbf{x}_2	\dots	\mathbf{x}_n	\mathbf{y}
x_{11}	x_{21}	\dots	x_{n1}	y_1
		\vdots		
x_{1m}	x_{2m}	\dots	x_{nm}	y_m

- Instances are **independent** of one another, **given the model**

$$y = f_{\theta}(\vec{x}).$$

- Parameters **minimize estimation error** e.g.

$$\hat{\theta} = \arg \min_{\theta} \sum_i \|y_i - f_{\theta}(\vec{x}_i)\|.$$

Failing Assumptions

1/2

student	course	grade
s_1	c_1	a
s_2	c_1	c
s_1	c_2	b
s_2	c_3	b
s_3	c_2	b
s_4	c_3	b
s_3	c_4	X
s_4	c_4	Y

Infer $X \succ_g Y$.

"Proof."

$s_1 \succ_s s_2$ (rows 1 & 2)
so $c_2 \succ_c c_3$ (3 & 4).

Then $s_3 \succ_s s_4$ (5 & 6)
therefore $X \succ_g Y$ \square .

- Instances are **not independent**.
- Small dataset, so **no statistics**.

Failing Assumptions

2/2

Prompt: Given $f(\text{machine}) = e$, $f(\text{learning}) = g$ and $f(\text{algorithm}) = m$, what is the value of $f(\text{ksadbf})$?

input	output
<i>machine</i>	<i>e</i>
<i>learning</i>	<i>g</i>
<i>algorithm</i>	<i>m</i>
<i>ksadbf</i>	<i>?</i>

Gemini: I cannot determine the value of $f(\text{program})$ with the given information...

ChatGPT: The information provided seems to define a function f where certain inputs are associated with specific outputs. Based on the given information:

- $f(\text{machine}) = e$
- $f(\text{learning}) = g$
- $f(\text{algorithm}) = m$

However, there is no information given about $f(\text{ksadbf})$. Without additional information or a specific rule that determines the output for $f(\text{ksadbf})$, *it is not possible to determine its value*. If there is a pattern or rule for how the outputs are assigned based on the inputs, you would need to provide that information for me to help determine the value of $f(\text{ksadbf})$.

No help from state-of-the-art Natural Language Processing or LLMs.

Where Machine Learning Fails

- No *interpretability*: large models ($\dim \theta \sim$ millions) are **not understandable by humans**.
- Need for *large number of training* examples.
- Unfit for *multiple relations, intersample dependencies* and *variable features*: The *Students, Courses and Results* and *Last Letter* examples.
- Poor *generalization*: A model trained in the iris dataset is useless in any other domain.
- **No use of background knowledge**.

References

- Gary Marcus, *Deep Learning: A Critical Appraisal*, 2018.
- François Chollet, *On the Measure of Intelligence*, 2019.
- Bengio *et al.*, *A Meta-Transfer Objective for Learning to Disentangle Causal Mechanisms*, 2019.
- Cropper *et al.*, *Turning 30: New Ideas in Inductive Logic Programming*, 2020.
- Fabrizio Riguzzi, *Foundations of Probabilistic Logic Programming*, 2023.