sMProbLog: Stable Model Semantics in ProbLog and its Applications in Argumentation

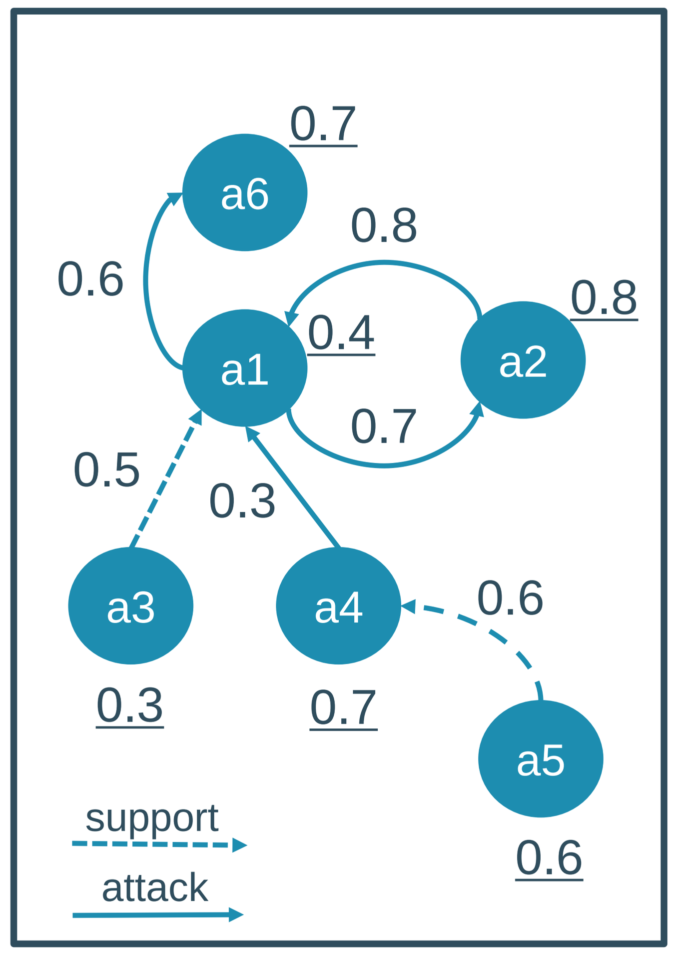
Pietro Totis, Angelika Kimmig and Luc De Raedt *name.surname@kuleuven.be*

SMProbLog: a probabilistic logic programming (PLP) system for reasoning and learning over beliefs and non-deterministic logical choices.

- A semantics for functor-free probabilistic normal logic programs
- A PLP system for inference and learning under the new semantics
- A novel approach to epistemic argumentation based on PLP

Probabilistic argumentation graphs: modelling beliefs and influences

Argumentative microtext[1]:
"Yes, it's annoying and cumbersome to separate your rubbish properly all the time (a1), but small gestures become natural by daily repetition (a2). Three different bin bags stink away in the kitchen and have to be sorted into different wheelie bins (a3). But still Germany produces way too much rubbish (a4) and too many resources are lost when what actually should be separated and recycled is burnt (a5). We Berliners should take the chance and become pioneers in waste separation! (a6)"

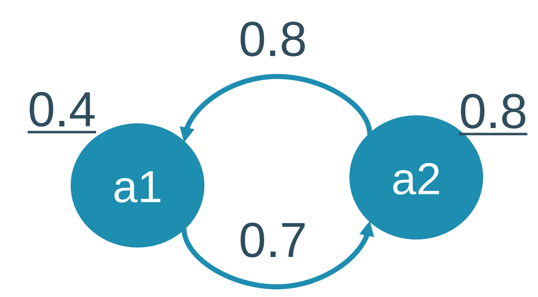


1: Stede et. Al. 2016. Parallel discourse annotations on a corpus of short texts.

Reasoning over normal logic programs

Problem: traditional PLP frameworks cannot reason over programs defining dependencies between atoms that are cyclic and involve negation.

```
0.4::bias(a1). 0.8::bias(a2).
arg(a1) ← arg_pos(a1), ~arg_neg(a1).
arg(a2) ← arg_pos(a2), ~arg_neg(a2).
0.8:: f. 0.7:: g.
arg_neg(a1) ← f, arg(a2).
arg_neg(a2) ← g, arg(a1).
```



The probability of an atom is the probability of success of the query in each possible world, that is, the logic rules plus a total choice of facts p::f. where each is either included (with probability p) or excluded (1-p).

ProbLog semantics is well-defined for programs where each total choice corresponds to exactly one two-valued well-founded model

Inference and Learning in sMProbLog

- SMProbLog reduces probabilistic inference to weighted model counting (WMC):
  1. Bottom-up grounding.
  2. Knowledge compilation (compile).
  3. Logic circuit to arithmetic circuit.
  4. Model counting for normalization (enumerate).
  5. WMC by evaluating the arithmetic circuit (evaluate).

$$\hat{WMC}_{\mathcal{L}}(\varphi) = \sum_{M \in MOD(\mathcal{L}), M \models \varphi} \hat{w}(M) \cdot \prod_{l \in M} w(l)$$

- Expectation-Maximization parameter learning on consistent programs

Probabilistic logic programming: modelling beliefs and influences in argumentation

Inhibition effect and negative heads[2] for modelling negative causal effects over beliefs in arguments by means of PLP.

```
0.4:: bias(a1). 0.8:: bias(a2).
0.3:: bias(a3). 0.7:: bias(a4).
0.6:: bias(a5). 0.7:: bias(a6).
arg(A) ← bias(A).
0.6:: ¬arg(a6) ← arg(a1).
0.6:: arg(a4)  ← arg(a5).
0.3:: ¬arg(a1) ← arg(a4).
0.5:: arg(a1)  ← arg(a3).
0.8:: ¬arg(a1) ← arg(a2).
0.7:: ¬arg(a2) ← arg(a1).
```

Arguments have an independent bias

Attacks *inhibit* the belief in an argument

Supports *increase* the belief in an argument

```
                        arg(a) ← arg_pos(a), ~arg_neg(a).
p::¬arg(a) ← arg(b). ≡  p::f.
                        arg_neg(a) ← f, arg(b).
```

2: Meert, W. And Vennekens, J. 2014. Inhibited effects in cp-logic.

Stable Model Semantics over Total Choices

In sMProblog each total choice corresponds to zero, one or many stable models.

Probability of total choices ω: $P(\omega) = \prod_{(p::f), f \in \omega} p \cdot \prod_{(p::f), f \notin \omega} (1-p)$

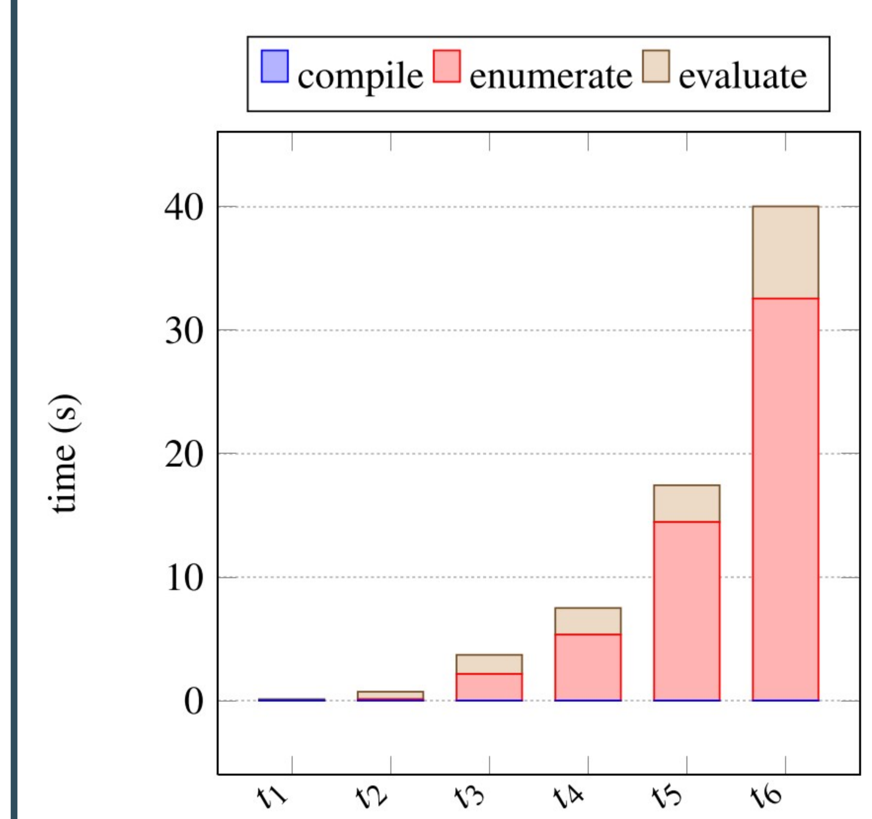
If a total choice has:

- No stable model: assign its probability mass to an "inconsistent" state.
- Exactly one model: assign its probability mass to the model.
- Many stable models: distribute its probability mass uniformly across the stable models induced by the total choice.
  - Compute a "normalization constant" $\hat{w}$(M) for each model M.

`Example: 0.4::a. 0.5::b. c :-a. d:-b. c:-~d. d:-~c.`

Total choice	2-WFM	3-WFM (T,F,U)	Stable Model	$\hat{w}$(M)
{a,b}	{a,b,c,d}	T={a,b,c,d}, F={}, U={}	{a,b,c,d}	1
{a}	{a,c}	T={a,c}, F={}, U={}	{a,c}	1
{b}	{b,d}	T={b,d}, F={}, U={}	{b,d}	1
{ }	/	T={}, F={a,b}, U={c,d}	{c}, {d}	1/2

Experiments




```
0.1 :: asthma(X) ← person(X).
0.3 :: stress(X) ← person(X).
0.4 :: smokes(X) ← stress(X).
smokes(X) ← influences(Y,X),smokes(Y).
0.4 :: asthma(X) ← smokes(X).
¬smokes(X) ← asthma(X).
person(1).
person(2)
0.3::influences(1,2).
0.6::influences(2,1).          % t1 ↑
person(3).                     % t2 ↑
person(4).                     % t3 ↑
0.2::influences(2,3).          % t4 ↑
0.7::influences(3,4).          % t5 ↑
0.9::influences(4,1).          % t6 ↑
```

Benchmark	t_1	t_2	t_3	t_4	t_5	t_6
# prob. facts	10	14	18	19	20	21
# nodes circuit	156	192	228	462	750	1465

Retrieving the normalization constants is expensive: compiling a circuit that allows grouping models by total choices is an interesting research direction for the future.