Probabilistic Logic Languages

Fabrizio Riguzzi

ENDIF – University of Ferrara, Italy
fabrizio.riguzzi@unife.it
Combining Logic and Probability

- Useful to model domains with complex and uncertain relationships among entities
- Many approaches proposed in the areas of Logic Programming, Uncertainty in AI, Machine Learning, Databases
- Logic Programming: Distribution Semantics [Sato, 1995]
- A probabilistic logic program defines a probability distribution over normal logic programs (called instances or possible worlds or simply worlds)
- The distribution is extended to a joint distribution over worlds and interpretations (or queries)
- The probability of a query is obtained from this distribution
Probabilistic Logic Programming (PLP) Languages under the Distribution Semantics

- Probabilistic Logic Programs [Dantsin, 1991]
- Probabilistic Horn Abduction [Poole, 1993], Independent Choice Logic (ICL) [Poole, 1997]
- PRISM [Sato, 1995]
- Logic Programs with Annotated Disjunctions (LPADs) [Vennekens et al., 2004]
- ProbLog [De Raedt et al., 2007]

They differ in the way they define the distribution over logic programs
Independent Choice Logic

\[
\text{sneezing}(X) \leftarrow \text{flu}(X), \text{flu\_sneezing}(X).
\]

\[
\text{sneezing}(X) \leftarrow \text{hay\_fever}(X), \text{hay\_fever\_sneezing}(X).
\]

\[
\text{flu}(bob).
\]

\[
\text{hay\_fever}(bob).
\]

\[
\text{disjoint}([\text{flu\_sneezing}(X) : 0.7, \text{null} : 0.3]).
\]

\[
\text{disjoint}([\text{hay\_fever\_sneezing}(X) : 0.8, \text{null} : 0.2]).
\]

- Distributions over facts by means of disjoint statements
- \text{null} does not appear in the body of any rule
- Worlds obtained by selecting one atom from every grounding of each disjoint statement
sneezing(X) ← flu(X), msw(flu_sneezing(X), 1).
sneezing(X) ← hay_fever(X), msw(hay_fever_sneezing(X), 1).
flu(bob).
hay_fever(bob).

values(flu_sneezing(_X), [1, 0]).
values(hay_fever_sneezing(_X), [1, 0]).
: − set_sw(flu_sneezing(_X), [0.7, 0.3]).
: − set_sw(hay_fever_sneezing(_X), [0.8, 0.2]).

- Distributions over msw facts (random switches)
- Worlds obtained by selecting one value for every grounding of each msw statement
Logic Programs with Annotated Disjunctions

\[
\begin{align*}
\text{sneezing}(X) & : 0.7 \lor \text{null} : 0.3 \leftarrow \text{flu}(X). \\
\text{sneezing}(X) & : 0.8 \lor \text{null} : 0.2 \leftarrow \text{hay\_fever}(X). \\
\text{flu}(\text{bob}). \\
\text{hay\_fever}(\text{bob}).
\end{align*}
\]

- Distributions over the head of rules
- \texttt{null} does not appear in the body of any rule
- Worlds obtained by selecting one atom from the head of every grounding of each clause
\[ \text{sneezing}(X) \leftarrow \text{flu}(X), \text{flu}\_\text{sneezing}(X). \]
\[ \text{sneezing}(X) \leftarrow \text{hay}\_\text{fever}(X), \text{hay}\_\text{fever}\_\text{sneezing}(X). \]
\[ \text{flu}(\text{bob}). \]
\[ \text{hay}\_\text{fever}(\text{bob}). \]
\[ 0.7 :: \text{flu}\_\text{sneezing}(X). \]
\[ 0.8 :: \text{hay}\_\text{fever}\_\text{sneezing}(X). \]

- Distributions over facts
- Worlds obtained by selecting or not every grounding of each probabilistic fact
Distribution Semantics

- Case of no function symbols: finite Herbrand universe, finite set of groundings of each disjoint statement/switch/clause
- **Atomic choice**: selection of the $i$-th atom for grounding $C\theta$ of disjoint statement/switch/clause $C$
  - represented with the triple $(C, \theta, i)$
  - a ProbLog fact $p :: F$ is interpreted as $F : p \lor \text{null} : 1 - p$.
- Example $C_1 = \text{disjoint}([\text{flu_sneezing}(X) : 0.7, \text{null} : 0.3])$, $(C_1, \{X/bob\}, 1)$
- **Composite choice** $\kappa$: consistent set of atomic choices
  - $\kappa = \{(C_1, \{X/bob\}, 1), (C_1, \{X/bob\}, 2)\}$ not consistent
  - The probability of composite choice $\kappa$ is

$$P(\kappa) = \prod_{(C,\theta,i) \in \kappa} P_0(C, i)$$
Selection $\sigma$: a total composite choice (one atomic choice for every grounding of each disjoint statement/clause)

$\sigma = \{(C_1, \{X/bob\}, 1), (C_2, \{bob\}, 1)\}$

$C_1 = \text{disjoint}([\text{flu\_sneezing}(X) : 0.7, \text{null} : 0.3])$.

$C_2 = \text{disjoint}([\text{hay\_fever\_sneezing}(X) : 0.8, \text{null} : 0.2])$.

A selection $\sigma$ identifies a logic program $w_\sigma$ called world.

The probability of $w_\sigma$ is $P(w_\sigma) = P(\sigma) = \prod_{(C, \theta, i) \in \sigma} P_0(C, i)$

Finite set of worlds: $W_T = \{w_1, \ldots, w_m\}$

$P(w)$ distribution over worlds: $\sum_{w \in W_T} P(w) = 1$
**Distribution Semantics**

- Herbrand base $H_T = \{ A_1, \ldots, A_n \}$
- Herbrand interpretation $I = \{ a_1, \ldots, a_n \}$
- $P(I|w) = 1$ if $I$ is a model of $w$ and $0$ otherwise
- $P(I) = \sum_w P(I, w) = \sum_w P(I|w)P(w) = \sum_{w,I \text{ model of } w} P(w)$
- The distribution over interpretations can be seen as a joint distribution $P(A_1, \ldots, A_n)$ over the atoms of $H_T$
- Query: $(A_j = \text{true}) = a_j$
- $P(a_j) = \sum_{a_i, i \neq j} P(a_1, \ldots, a_m) = \sum_{I,a_j \in I} P(I)$
- $P(a_j) = \sum_{I,a_j \in I} \sum_{w \in W} P(w)P(w)$
Alternatively,

\[ P(a_j | w) = 1 \text{ if } A_j \text{ is true in } w \text{ and } 0 \text{ otherwise} \]

\[ P(a_j) = \sum_w P(a_j, w) = \sum_w P(a_j | w)P(w) = \sum_{w | = A_j} P(w) \]
Example Program (ICL)

- 4 worlds

sneezing(X) ← flu(X), flu_sneezing(X).
sneezing(X) ← hay_fever(X), hay_fever_sneezing(X).
flu(bob).
hay_fever(bob).
flu_sneezing(bob).

null.

P(w_1) = 0.7 \times 0.8
P(w_2) = 0.3 \times 0.8
P(w_3) = 0.7 \times 0.2
P(w_4) = 0.3 \times 0.2

- sneezing(bob) is true in 3 worlds
- \[ P(\text{sneezing}(\text{bob})) = 0.7 \times 0.8 + 0.3 \times 0.8 + 0.7 \times 0.2 = 0.94 \]
Example Program (LPAD)

- **4 worlds**

  ```
  sneezing(bob) ← flu(bob).
  sneezing(bob) ← hay_fever(bob).
  flu(bob).
  hay_fever(bob).
  P(w_1) = 0.7 \times 0.8
  
  sneezing(bob) ← flu(bob).
  null ← hay_fever(bob).
  flu(bob).
  hay_fever(bob).
  P(w_2) = 0.3 \times 0.8
  
  sneezing(bob) ← flu(bob).
  null ← hay_fever(bob).
  flu(bob).
  hay_fever(bob).
  P(w_3) = 0.7 \times 0.2
  
  sneezing(bob) ← flu(bob).
  null ← hay_fever(bob).
  flu(bob).
  hay_fever(bob).
  P(w_4) = 0.3 \times 0.2
  ```

- **sneezing(bob) is true in 3 worlds**

  - **P(sneezing(bob)) = 0.7 \times 0.8 + 0.3 \times 0.8 + 0.7 \times 0.2 = 0.94**
Example Program (ProbLog)

- 4 worlds

\[
\begin{align*}
\text{sneezing}(X) & \leftarrow \text{flu}(X), \text{flu\_sneezing}(X). \\
\text{sneezing}(X) & \leftarrow \text{hay\_fever}(X), \text{hay\_fever\_sneezing}(X). \\
\text{flu}(bob). \\
\text{hay\_fever}(bob). \\
\text{flu\_sneezing}(bob). \\
\text{hay\_fever\_sneezing}(bob). \\
\text{hay\_fever\_sneezing}(bob). \\
P(w_1) &= 0.7 \times 0.8 \\
P(w_2) &= 0.3 \times 0.8 \\
P(w_3) &= 0.7 \times 0.2 \\
P(w_4) &= 0.3 \times 0.2
\end{align*}
\]

- \text{sneezing}(bob) is true in 3 worlds

\[
P(\text{sneezing}(bob)) = 0.7 \times 0.8 + 0.3 \times 0.8 + 0.7 \times 0.2 = 0.94
\]
Examples

Throwing coins

heads(Coin):1/2 ; tails(Coin):1/2 :-
   toss(Coin),\+biased(Coin).
heads(Coin):0.6 ; tails(Coin):0.4 :-
   toss(Coin),biased(Coin).
fair(Coin):0.9 ; biased(Coin):0.1.
toss(coin).

Russian roulette with two guns

death:1/6 :- pull_trigger(left_gun).
death:1/6 :- pull_trigger(right_gun).
pull_trigger(left_gun).
pull_trigger(right_gun).
Examples

Mendel’s inheritance rules for pea plants

color(X, white) :- cg(X, 1, w), cg(X, 2, w).
color(X, purple) :- cg(X, _A, p).
cg(X, 1, A) : 0.5 ; cg(X, 1, B) : 0.5 :-
    mother(Y, X), cg(Y, 1, A), cg(Y, 2, B).
cg(X, 2, A) : 0.5 ; cg(X, 2, B) : 0.5 :-
    father(Y, X), cg(Y, 1, A), cg(Y, 2, B).

Probability of paths

path(X, X).
path(X, Y) :- path(X, Z), edge(Z, Y).
edge(a, b) : 0.3.
edge(b, c) : 0.2.
edge(a, c) : 0.6.
Distribution Semantics

Encoding Bayesian Networks

<table>
<thead>
<tr>
<th></th>
<th>t</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>burg</td>
<td>0.1</td>
<td>0.9</td>
</tr>
<tr>
<td>earthq</td>
<td>0.2</td>
<td>0.8</td>
</tr>
<tr>
<td>alarm</td>
<td>0.8</td>
<td>0.2</td>
</tr>
<tr>
<td>alarm(t)</td>
<td>1.0</td>
<td>0.0</td>
</tr>
<tr>
<td>alarm(f)</td>
<td>0.8</td>
<td>0.2</td>
</tr>
<tr>
<td>alarm(t)</td>
<td>0.8</td>
<td>0.2</td>
</tr>
<tr>
<td>alarm(f)</td>
<td>0.1</td>
<td>0.9</td>
</tr>
<tr>
<td>alarm(t)</td>
<td>0.8</td>
<td>0.2</td>
</tr>
<tr>
<td>alarm(f)</td>
<td>0.8</td>
<td>0.2</td>
</tr>
</tbody>
</table>

burg(t): 0.1 ; burg(f): 0.9.
earthq(t): 0.2 ; earthq(f): 0.8.
alarm(t): ~burg(t), earthq(t).
alarm(t): 0.8 ; alarm(f): 0.2: ~burg(t), earthq(f).
alarm(t): 0.8 ; alarm(f): 0.2: ~burg(f), earthq(t).
alarm(t): 0.1 ; alarm(f): 0.9: ~burg(f), earthq(f).
Expressive Power

- All these languages have the same expressive power
- LPADs have the most general syntax
- There are transformations that can convert each one into the others
- ICL, PRISM: direct mapping
- ICL, PRISM to LPAD: direct mapping
Clause $C_i$ with variables $\overline{X}$

$$H_1 : p_1 \lor \ldots \lor H_n : p_n \leftarrow B.$$ 

is translated into

$$H_1 \leftarrow B, \text{choice}_{i,1}(\overline{X}).$$

$$\vdash$$

$$H_n \leftarrow B, \text{choice}_{i,n}(\overline{X}).$$

$$\text{disjoint}([\text{choice}_{i,1}(\overline{X}) : p_1, \ldots, \text{choice}_{i,n}(\overline{X})]).$$
LPADs to ProbLog

- Clause $C_i$ with variables $\bar{X}$

$$H_1 : p_1 \lor \ldots \lor H_n : p_n \leftarrow B.$$ is translated into

$$H_1 \leftarrow B, f_{i,1}(\bar{X}).$$
$$H_2 \leftarrow B, \neg(f_{i,1}(\bar{X})), f_{i,2}(\bar{X}).$$
$$\vdots$$
$$H_n \leftarrow B, \neg(f_{i,1}(\bar{X})), \ldots, \neg(f_{i,n-1}(\bar{X})).$$

$$\pi_1 :: f_{i,1}(\bar{X}).$$
$$\vdots$$
$$\pi_{n-1} :: f_{i,n-1}(\bar{X}).$$

where $\pi_1 = p_1$, $\pi_2 = \frac{p_2}{1-\pi_1}$, $\pi_3 = \frac{p_3}{(1-\pi_1)(1-\pi_2)}$, $\ldots$

- In general $\pi_j = \frac{p_j}{\prod_{j=1}^{j-1}(1-\pi_j)}$
Combining Rule

These languages combine independent evidence for a ground atom coming from different clauses with a noisy-or combining rule.

If atom $A$ can be derived with probability $p_1$ from a rule and with probability $p_2$ from a different rule and the two derivations are independent, then $P(A) = p_1 + p_2 - p_1 p_2$.

Example

\[
\begin{align*}
\text{sneezing}(X) : & 0.7 \lor \text{null} : 0.3 \leftarrow \text{flu}(X) . \\
\text{sneezing}(X) : & 0.8 \lor \text{null} : 0.2 \leftarrow \text{hay\_fever}(X) . \\
\text{flu}(\text{bob}) . \\
\text{hay\_fever}(\text{bob}) .
\end{align*}
\]

$P(\text{sneezing}(\text{bob})) = 0.7 + 0.8 - 0.7 \times 0.8 = 0.94$

Particularly useful for modeling independent causes for the same effect.
How to deal with negation?
Each world should have a single total model because we consider two-valued interpretations
We want to model uncertainty only by means of random choices
This can be required explicitly: each world should have a total well founded model/single stable model (sound programs)
Function Symbols

- What if function symbols are present?
- Infinite, countable Herbrand universe
- Infinite, countable Herbrand base
- Infinite, countable grounding of the program $T$
- Uncountable $W_T$
- Each world infinite, countable
- $P(w) = 0$
- Semantics not well-defined
Game of dice

\[
on(0,1): 1/3 \ ; \ on(0,2): 1/3 \ ; \ on(0,3): 1/3 . \\
on(T,1): 1/3 \ ; \ on(T,2): 1/3 \ ; \ on(T,3): 1/3 : - \\
\quad T1 \ is \ T-1, \ T1>=0, \ on(T1,F), \ \backslash+ \ on(T1,3).
\]
Hidden Markov Models

\[ X(t-1) \rightarrow Y(t-1) \rightarrow X(t) \rightarrow Y(t) \rightarrow X(t+1) \rightarrow Y(t+1) \rightarrow \ldots \]

\[
\text{hmm}(S,O) : \neg \text{hmm}(q1, [], S, O).
\]
\[
\text{hmm}(\text{end}, S, S, []).
\]
\[
\text{hmm}(Q, S0, S, [L|O]) :-
\]
\[
Q \neq \text{end},
\]
\[
\text{next_state}(Q, Q1, S0),
\]
\[
\text{letter}(Q, L, S0),
\]
\[
\text{hmm}(Q1, [Q|S0], S, O).
\]
\[
\text{next_state}(q1, q1, _S): 1/3; \text{next_state}(q1, q2, _S): 1/3;
\]
\[
\text{next_state}(q1, \text{end}, _S): 1/3.
\]
\[
\text{next_state}(q2, q1, _S): 1/3; \text{next_state}(q2, q2, _S): 1/3;
\]
\[
\text{next_state}(q2, \text{end}, _S): 1/3.
\]
\[
\text{letter}(q1, a, _S): 0.25; \text{letter}(q1, c, _S): 0.25;
\]
\[
\text{letter}(q1, g, _S): 0.25; \text{letter}(q1, t, _S): 0.25.
\]
\[
\text{letter}(q2, a, _S): 0.25; \text{letter}(q2, c, _S): 0.25;
\]
\[
\text{letter}(q2, g, _S): 0.25; \text{letter}(q2, t, _S): 0.25.
\]
Semantics proposed for ICL and PRISM, applicable also to the other languages

Definition of a probability measure $\mu$ over $W_T$

$\mu$ assign a probability to every element of an algebra $\Omega$ of subsets of $W_T$, i.e. a set of subsets closed under union and complementation

The algebra $\Omega$ is the set of sets of worlds identified by a finite set of finite composite choices
Composite Choices

- Set of worlds compatible with $\kappa$: $\omega_\kappa = \{ w_\sigma \in W_T | \kappa \subseteq \sigma \}$
- For programs without function symbols $P(\kappa) = \sum_{w \in \omega_\kappa} P(w)$

```
sneezing(X) ← flu(X), flu_sneezing(X).
sneezing(X) ← hay_fever(X), hay_fever_sneezing(X).
flu(bob).
hay_fever(bob).
C_1 = disjoint([[flu_sneezing(X) : 0.7, null : 0.3]]).
C_2 = disjoint([[hay_fever_sneezing(X) : 0.8, null : 0.2]].
```

- $\kappa = \{(C_1, \{X/bob\}, 1)\}$, $\omega_\kappa =$
  
  $\text{flu_sneezing}(bob)$.
  $\text{flu_sneezing}(bob)$.
  $\text{hay_fever_sneezing}(bob)$.
  null.
  $P(w_1) = 0.7 \times 0.8$
  $P(w_2) = 0.7 \times 0.2$

- $P(\kappa) = 0.7 = P(w_1) + P(w_2)$
Sets of Composite Choices

- Set of composite choices $K$
- Set of worlds compatible with $K$: $\omega_K = \bigcup_{\kappa \in K} \omega_\kappa$
- $\Omega = \{\omega_K | K$ is finite set of finite composite choices\}$
- Two composite choices $\kappa_1$ and $\kappa_2$ are exclusive if their union is inconsistent

$$\kappa_1 = \{(C_1, \{X/bob\}, 1)\},$$
$$\kappa_2 = \{(C_1, \{X/bob\}, 2), (C_2, \{X/bob\}, 1)\}$$

- $\kappa_1 \cup \kappa_2$ inconsistent
- A set $K$ of composite choices is mutually exclusive if for all $\kappa_1 \in K, \kappa_2 \in K, \kappa_1 \neq \kappa_2 \Rightarrow \kappa_1$ and $\kappa_2$ are exclusive.
Sets of Composite Choices

- Case of no functions symbols
  \[ \sum_{\kappa \in K} P(\kappa) \neq \sum_{w \in \omega_K} P(w) \]
- \( \kappa_1 = \{(C_1, \{X/bob\}, 1)\} \), \( \kappa_2 = \{(C_2, \{X/bob\}, 1)\} \), \( K = \{\kappa_1, \kappa_2\} \)
- \( P(\kappa_1) = 0.7 \), \( P(\kappa_2) = 0.8 \), \( \sum_{w \in \omega_K} P(w) = 0.94 \)
- If \( K \) is mutually incompatible, \( \sum_{\kappa \in K} P(\kappa) = \sum_{w \in \omega_K} P(w) \)
- \( \kappa'_2 = \{(C_1, \{X/bob\}, 2), (C_2, \{X/bob\}, 1)\} \), \( K' = \{\kappa_1, \kappa'_2\} \)
- \( P(\kappa'_2) = 0.3 \cdot 0.8 = 0.24 \)
- Probability of mutually exclusive set \( K \) of composite choices:
  \[ P(K) = \sum_{\kappa \in K} P(\kappa) \]
Sets of Composite Choices

- \( K = \{ \kappa_1, \ldots, \kappa_n \} \)
- \( P(K) = P(\kappa_1 \lor \ldots \lor \kappa_n) \)
- \( P(A \lor B) = P(A) + P(B) - P(AB) \)
- \( P(A \lor B \lor C) = P(A) + P(B) + P(C) - P(AB) - P(BC) + P(ABC) \)
- ... (inclusion exclusion formula)
- \( P(\kappa_1 \land \kappa_2) \) may be:
  - 0, if \( \kappa_1, \kappa_2 \) are inconsistent
  - \( P(\kappa_1)P(\kappa_2) \) if they are independent (no common grounding \( C\theta \))
  - In general, we have to count only once repeated atomic choices

If \( K \) is mutually incompatible \( P(\kappa_i \land \ldots \land \kappa_j) = 0 \)
- \( P(K) = P(\kappa_1) + \ldots + P(\kappa_n) \)
Set of Composite Choices

Two sets $K_1$ and $K_2$ of finite composite choices may correspond to the same set of worlds: $\omega_{K_1} = \omega_{K_2}$

Lemma ([Poole, 2000])

Given a finite set $K$ of finite composite choices, there exists a finite set $K'$ of finite composite choices that is mutually exclusive and such that $\omega_K = \omega_{K'}$. 
Probability Measure

Lemma ([Poole, 2000])

If $K$ and $K'$ are both mutually exclusive sets of composite choices such that $\omega_K = \omega_{K'}$, then $P(K) = P(K')$

- $\Omega = \{\omega_K | K$ is a finite set of finite composite choices\}
- $\Omega$ is an algebra

Definition

$\mu : \Omega \rightarrow [0, 1]$ is

$$\mu(\omega) = P(K)$$

for $\omega \in \Omega$ where $K$ is a mutually exclusive finite set of finite composite choices such that $\omega_K = \omega$. 
**Probability Measure**

- $\mu$ satisfies **the finite additivity version** of Kolmogorov probability axioms
  1. $\mu(\omega) \geq 0$ for all $\omega \in \Omega$
  2. $\mu(\emptyset) = 1$
  3. $\omega_1 \cap \omega_2 = \emptyset \rightarrow \mu(\omega_1 \cup \omega_2) = \mu(\omega_1) + \mu(\omega_2)$ for all $\omega_1, \omega_2 \in \Omega$

- So $\mu$ is a probability measure
Probability of a Query

- Given a query $Q$, a composite choice $\kappa$ is an explanation for $Q$ if
  \[ \forall w \in \omega_\kappa \ w \models Q \]
- A set $K$ of composite choices is covering wrt $Q$ if every world in which $Q$ is true belongs to $\omega_K$

Definition

\[ P(Q) = \mu(\{w | w \in W_T, w \models Q\}) \]

- If $Q$ has a finite set of finite explanations that is covering, $P(Q)$ is well-defined
Example Program (ICL)

\[
\text{sneezing}(X) \leftarrow \text{flu}(X), \text{flu\_sneezing}(X).
\]
\[
\text{sneezing}(X) \leftarrow \text{hay\_fever}(X), \text{hay\_fever\_sneezing}(X).
\]
\[
\text{flu}(\text{bob}).
\]
\[
\text{hay\_fever}(\text{bob}).
\]
\[
C_1 = \text{disjoint}([\text{flu\_sneezing}(X) : 0.7, \text{null} : 0.3]).
\]
\[
C_2 = \text{disjoint}([\text{hay\_fever\_sneezing}(X) : 0.8, \text{null} : 0.2]).
\]

- **Goal** \(\text{sneezing}(\text{bob})\)
- \(\kappa_1 = \{(C_1, \{X/\text{bob}\}, 1)\}\)
- \(\kappa_2 = \{(C_1, \{X/\text{bob}\}, 2), (C_2, \{X/\text{bob}\}, 1)\}\)
- \(K = \{\kappa_1, \kappa_2\}\) mutually exclusive finite set of finite explanations that are covering for \(\text{sneezing}(\text{bob})\)
- \(P(Q) = P(\kappa_1) + P(\kappa_2) = 0.7 + 0.3 \cdot 0.8 = 0.94\)
The probability is well defined provided that the query has a finite set of finite explanations that are covering.

In PRISM this is explicitly required.

In ICL the program is required to be acyclic.

What conditions can we impose on the program so that these requirements are met?
Conditions

- Acyclic programs
- Modularly acyclic program
- Extended to PLP by requiring that each world is acyclic, modularly acyclic [Riguzzi, 2009].
- New conditions: dynamic stratification, bounded term size,... ?
Conversion to Bayesian Networks

- PLP can be converted to Bayesian networks
- Conversion for an LPAD $T$
- For each atom $A$ in $H_T$ a binary variable $A$
- For each clause $C_i$ in the grounding of $T$

\[ H_1 : p_1 \lor \ldots \lor H_n : \! p_n \leftarrow B_1, \ldots B_m, \neg C_1, \ldots, \neg C_l \]

a variable $CH_i$ with $B_1, \ldots, B_m, C_1, \ldots, C_l$ as parents and $H_1, \ldots, H_n$ and null as values

The CPT of $CH_i$ is

<table>
<thead>
<tr>
<th>$CH_i = H_1$</th>
<th>$\ldots$</th>
<th>$B_1 = 1, \ldots, B_m = 1, C_1 = 0, \ldots, C_l = 0$</th>
<th>$\ldots$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td></td>
<td>$p_1$</td>
<td>0.0</td>
</tr>
<tr>
<td>$\ldots$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$CH_n = H_n$</td>
<td>0.0</td>
<td>$p_n$</td>
<td>0.0</td>
</tr>
<tr>
<td>$CH_i = \text{null}$</td>
<td>1.0</td>
<td>$1 - \sum_{i=1}^n p_i$</td>
<td>1.0</td>
</tr>
</tbody>
</table>
Each variable $A$ corresponding to atom $A$ has as parents all the variables $CH_i$ of clauses $C_i$ that have $A$ in the head.

The CPT for $A$ is:

<table>
<thead>
<tr>
<th></th>
<th>at least one parent equal to $A$</th>
<th>remaining columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A = 1$</td>
<td>1.0</td>
<td>0.0</td>
</tr>
<tr>
<td>$A = 0$</td>
<td>0.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>
Conversion to Bayesian Networks

\[ C_1 = x_1 : 0.4 \lor x_2 : 0.6. \]
\[ C_2 = x_2 : 0.1 \lor x_3 : 0.9. \]
\[ C_3 = x_4 : 0.6 \lor x_5 : 0.4 \leftarrow x_1. \]
\[ C_4 = x_5 : 0.4 \leftarrow x_2, x_3. \]
\[ C_5 = x_6 : 0.3 \lor x_7 : 0.2 \leftarrow x_2, x_5. \]

<table>
<thead>
<tr>
<th>( CH_1, CH_2 )</th>
<th>( x_1, x_2 )</th>
<th>( x_1, x_3 )</th>
<th>( x_2, x_2 )</th>
<th>( x_2, x_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_2 = 1 )</td>
<td>1.0</td>
<td>0.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>( x_2 = 0 )</td>
<td>0.0</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( x_2, x_5 )</th>
<th>t,t</th>
<th>t,f</th>
<th>f,t</th>
<th>f,f</th>
</tr>
</thead>
<tbody>
<tr>
<td>( CH_5 = x_6 )</td>
<td>0.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>( CH_5 = x_7 )</td>
<td>0.2</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>( CH_5 = \text{null} )</td>
<td>0.5</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>
Related Languages

- CP-logic [Vennekens et al., 2009]
- P-log [C.Baral et al., 2009]
CP-logic

- Syntactically equal to LPADs
- Aim: modeling causation
- Semantics defined in term of a tree representing a probabilistic process
- Each valid CP-theory is a valid LPAD with the same meaning
- There are LPADs that are not valid CP-theories

\[
\begin{align*}
p : 0.5 \lor q : 0.5 & \leftarrow r. \\
r & \leftarrow \neg p. \\
r & \leftarrow \neg q.
\end{align*}
\]

\[
\begin{align*}
p & \leftarrow r. \\
r & \leftarrow \neg p. \\
r & \leftarrow \neg q.
\end{align*}
\]

\[
M = \{r, p\} \quad M = \{r, q\}
\]

- No process satisfying **temporal precedence**: a rule cannot fire until the part of the process that determines whether its precondition holds is fully finished.
P-log

- Based on Answer Set Programming (ASP).
- A P-log program $T$ defines a distribution over the stable models of a related Answer Set program $\pi(T)$.
- The probability of a query is then obtained by marginalization

```prolog
bool={t,f}.
node={a,b,c,...}.
edge: node,node -> bool.
#domain node(X),node(Y),node(Z).
path(X,Y):- edge(X,Y,t).
path(X,Y):- edge(X,Z,t), path(Z,Y).
[r(a,b)] random(edge(a,b)).
[r(a,b)] pr(edge(a,b,t))=4/10.
```

- Disjunctions and unstratified negations allowed: some programs have no model
- The distribution obtained by multiplication is not normalized.
- The probability of each stable model must be normalized.
Conclusions

- PLP: lively research field
- Distribution semantics
- Reasoning: come to tomorrow seminar!
Reasoning Tasks

- **Inference**: we want to compute the probability or an explanation of a query given the model and, possibly, some evidence.
- **Weight learning**: we know the structural part of the model (the logic formulas) but not the numeric part (the weights) and we want to infer the weights from data.
- **Structure learning**: we want to infer both the structure and the weights of the model from data.
Inference Tasks

- Computing the (conditional) probability of a ground query given the model and, possibly, some evidence.
- Finding the most likely state of a set of query atoms given the evidence (Maximum A Posteriori/Most Probable Explanation inference).
  - In Hidden Markov Models, the most likely state of the state variables given the observations is the Viterbi path, its probability the Viterbi probability.
- Finding the \((k)\) most probable explanation(s).
- Finding the distribution of variable substitutions for a non-ground query.
- Finding the most probable variable substitution for a non-ground query.
Weight Learning

- Given
  - model: a probabilistic logic model with unknown parameters
  - data: a set of interpretations

- Find the values of the parameters that maximize the probability of the data given the model

- Discriminative learning: maximize the conditional probability of a set of outputs (e.g. ground instances for a predicate) given a set of inputs

- Alternatively, the data are queries for which we know the probability: minimize the error in the probability of the queries that is returned by the model
Structure Learning

- **Given**
  - language bias: a specification of the search space
  - data: a set of interpretations

- Find the formulas and the parameters that maximize the likelihood of the data given the model

- Discriminative learning: again maximize the conditional likelihood of a set of outputs given a set of inputs
The probabilistic logic theory is used directly as a template for generating an underlying complex graphical model [Breese et al., 1994].

Languages: CLP(BN), Markov Logic
Variables in a CLP(BN) program can be random
- Their values, parents and CPTs are defined with the program
- To answer a query with uninstantiated random variables, CLP(BN) builds a BN and performs inference
- The answer will be a probability distribution for the variables
- Probabilistic dependencies expressed by means of CLP constraints

\{ \text{Var} = \text{Function with } p(\text{Values, Dist}) \} \\
\{ \text{Var} = \text{Function with } p(\text{Values, Dist, Parents}) \}
course_difficulty(Key, Dif) :-
  { Dif = difficulty(Key) with p([h,m,l], [0.25, 0.50, 0.25]) }.

student_intelligence(Key, Int) :-
  { Int = intelligence(Key) with p([h, m, l], [0.5,0.4,0.1]) }.

registration(r0,c16,s0).
registration(r1,c10,s0).
registration(r2,c57,s0).
registration(r3,c22,s1).
registration_grade(Key, Grade):-
registration(Key, CKey, SKey),
course_difficulty(CKey, Dif),
student_intelligence(SKey, Int),
{ Grade = grade(Key) with
  p([a,b,c,d],
  %h h h m h l m h m m l l h l m l l
  [0.20,0.70,0.85,0.10,0.20,0.50,0.01,0.05,0.10,
    0.60,0.25,0.12,0.30,0.60,0.35,0.04,0.15,0.40,
    0.15,0.04,0.02,0.40,0.15,0.12,0.50,0.60,0.40,
    0.05,0.01,0.01,0.20,0.05,0.03,0.45,0.20,0.10 ],
  [Int,Dif]))
}.

.....
CLP(BN)

?- [school_32].
  ?- registration_grade(r0,G).
  p(G=a)=0.4115,
  p(G=b)=0.356,
  p(G=c)=0.16575,
  p(G=d)=0.06675 ?
?- registration_grade(r0,G),
    student_intelligence(s0,h).
  p(G=a)=0.6125,
  p(G=b)=0.305,
  p(G=c)=0.0625,
  p(G=d)=0.02 ?
Markov Networks

- Undirected graphical models

Each clique in the graph is associated with a potential $\phi_i$

\[ P(x) = \frac{\prod_i \phi_i(x_i)}{Z} \]

\[ Z = \sum_x \prod_i \phi_i(x_i) \]

<table>
<thead>
<tr>
<th>Smoking</th>
<th>Cancer</th>
<th>$\phi_i(V, T)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>4.5</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>4.5</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>2.7</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>4.5</td>
</tr>
</tbody>
</table>
If all the potential are strictly positive, we can use a log-linear model

\[
P(x) = \frac{\exp\left(\sum_i w_i f_i(x_i)\right)}{Z}
\]

\[
Z = \sum_x \prod_i \phi_i(x_i)
\]

\[
f_i(\text{Smoking}, \text{Cancer}) = \begin{cases} 
1 & \text{if } \neg\text{Smoking} \lor \text{Cancer} \\
0 & \text{otherwise}
\end{cases}
\]

\[
w_i = 1.5
\]
A Markov Logic Network (MLN) is a set of pairs \((F, w)\) where \(F\) is a formula in first-order logic and \(w\) is a real number.

Together with a set of constants, it defines a Markov network with:
- One node for each grounding of each predicate in the MLN.
- One feature for each grounding of each formula \(F\) in the MLN, with the corresponding weight \(w\).
Markov Logic Example

1.5 \( \forall x \ Smokes(x) \rightarrow Cancer(x) \)
1.1 \( \forall x, y \ Friends(x, y) \rightarrow (Smokes(x) \leftrightarrow Smokes(y)) \)

- Constants Anna (A) and Bob (B)
Markov Networks

- Probability of an interpretation $\mathbf{x}$

$$P(\mathbf{x}) = \frac{\exp(\sum_i w_in_i(\mathbf{x}_i))}{Z}$$

- $n_i(\mathbf{x}_i) =$ number of true groundings of formula $F_i$ in $\mathbf{x}$
- Typed variables and constants greatly reduce size of ground Markov net
References I


References II


References III


References IV
