

Probabilistic Logic Languages

Fabrizio Riguzzi

ENDIF – University of Ferrara, Italy
fabrizio.riguzzi@unife.it



Outline

- 1 Probabilistic Logic Languages
- 2 Distribution Semantics
- 3 Expressive Power
- 4 Distribution Semantics with Function Symbols
- 5 Conversion to Bayesian Networks
- 6 Related Languages
- 7 Conclusion



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

$sneezing(X) \leftarrow flu(X), flu_sneezing(X).$
 $sneezing(X) \leftarrow hay_fever(X), hay_fever_sneezing(X).$
 $flu(bob).$
 $hay_fever(bob).$

$disjoint([flu_sneezing(X) : 0.7, null : 0.3]).$
 $disjoint([hay_fever_sneezing(X) : 0.8, null : 0.2]).$

- Distributions over facts by means of **disjoint** statements
- *null* does not appear in the body of any rule
- Worlds obtained by selecting one atom from every grounding of each disjoint statement



PRISM

$sneezing(X) \leftarrow flu(X), msw(flu_sneezing(X), 1).$
 $sneezing(X) \leftarrow hay_fever(X), msw(hay_fever_sneezing(X), 1).$
 $flu(bob).$
 $hay_fever(bob).$

$values(flu_sneezing_X, [1, 0]).$
 $values(hay_fever_sneezing_X, [1, 0]).$
 $: -set_sw(flu_sneezing_X, [0.7, 0.3]).$
 $: -set_sw(hay_fever_sneezing_X, [0.8, 0.2]).$

- Distributions over *msw* facts (random switches)
- Worlds obtained by selecting one value for every grounding of each *msw* statement



Logic Programs with Annotated Disjunctions

$sneezing(X) : 0.7 \vee null : 0.3 \leftarrow flu(X).$
 $sneezing(X) : 0.8 \vee null : 0.2 \leftarrow hay_fever(X).$
 $flu(bob).$
 $hay_fever(bob).$

- Distributions over the head of rules
- *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



ProbLog

$sneezing(X) \leftarrow flu(X), flu_sneezing(X).$

$sneezing(X) \leftarrow hay_fever(X), hay_fever_sneezing(X).$

$flu(bob).$

$hay_fever(bob).$

$0.7 :: flu_sneezing(X).$

$0.8 :: hay_fever_sneezing(X).$

- 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, θ, i)
 - a ProbLog fact $p :: F$ is interpreted as $F : p \vee \text{null} : 1 - p$.
- Example $C_1 = \text{disjoint}([flu_sneezing(X) : 0.7, \text{null} : 0.3])$,
 $(C_1, \{X/bob\}, 1)$
- **Composite choice** κ : consistent set of atomic choices
- $\kappa = \{(C_1, \{X/bob\}, 1), (C_1, \{X/bob\}, 2)\}$ not consistent
- The probability of composite choice κ is

$$P(\kappa) = \prod_{(C, \theta, i) \in \kappa} P_0(C, i)$$



Distribution Semantics

- **Selection** σ : 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 σ identifies a logic program w_σ called **world**
- The probability of w_σ is $P(w_\sigma) = P(\sigma) = \prod_{(C,\theta,i) \in \sigma} P_0(C, i)$
- Finite set of worlds: $W_T = \{w_1, \dots, w_m\}$
- $P(w)$ distribution over worlds: $\sum_{w \in W_T} P(w) = 1$



Distribution Semantics

- Herbrand base $H_T = \{A_1, \dots, A_n\}$
- Herbrand interpretation $I = \{a_1, \dots, a_n\}$
- $P(I|w) = 1$ 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 $P(A_1, \dots, A_n)$ over the atoms of H_T
- Query: $(A_j = \text{true}) = a_j$
- $P(a_j) = \sum_{a_i, i \neq j} P(a_1, \dots, a_m) = \sum_{I, a_j \in I} P(I)$
- $P(a_j) = \sum_{I, a_j \in I} \sum_{w \in W, I \text{ model of } w} P(w)$



Distribution Semantics

- Alternatively,
- $P(a_j|w) = 1$ if A_j is true in w and 0 otherwise
- $P(a_j) = \sum_w P(a_j, w) = \sum_w P(a_j|w)P(w) = \sum_{w \models A_j} P(w)$



Example Program (ICL)

- 4 worlds

$sneezing(X) \leftarrow flu(X), flu_sneezing(X).$

$sneezing(X) \leftarrow hay_fever(X), hay_fever_sneezing(X).$

$flu(bob).$

$hay_fever(bob).$

$flu_sneezing(bob).$

$null.$

$hay_fever_sneezing(bob).$

$hay_fever_sneezing(bob).$

$P(w_1) = 0.7 \times 0.8$

$P(w_2) = 0.3 \times 0.8$

$flu_sneezing(bob).$

$null.$

$null.$

$null.$

$P(w_3) = 0.7 \times 0.2$

$P(w_4) = 0.3 \times 0.2$

- $sneezing(bob)$ 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 (LPAD)

- 4 worlds

$sneezing(bob) \leftarrow flu(bob).$

$sneezing(bob) \leftarrow hay_fever(bob).$

$flu(bob).$

$hay_fever(bob).$

$P(w_1) = 0.7 \times 0.8$

$null \leftarrow flu(bob).$

$sneezing(bob) \leftarrow hay_fever(bob).$

$flu(bob).$

$hay_fever(bob).$

$P(w_2) = 0.3 \times 0.8$

$sneezing(bob) \leftarrow flu(bob).$

$null \leftarrow hay_fever(bob).$

$flu(bob).$

$hay_fever(bob).$

$P(w_3) = 0.7 \times 0.2$

$null \leftarrow flu(bob).$

$null \leftarrow hay_fever(bob).$

$flu(bob).$

$hay_fever(bob).$

$P(w_4) = 0.3 \times 0.2$

- $sneezing(bob)$ 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

$sneezing(X) \leftarrow flu(X), flu_sneezing(X).$

$sneezing(X) \leftarrow hay_fever(X), hay_fever_sneezing(X).$

$flu(bob).$

$hay_fever(bob).$

$flu_sneezing(bob).$

$hay_fever_sneezing(bob).$ $hay_fever_sneezing(bob).$

$P(w_1) = 0.7 \times 0.8$

$P(w_2) = 0.3 \times 0.8$

$flu_sneezing(bob).$

$P(w_3) = 0.7 \times 0.2$

$P(w_4) = 0.3 \times 0.2$

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

- $P(sneezing(bob)) = 0.7 \times 0.8 + 0.3 \times 0.8 + 0.7 \times 0.2 = 0.94$



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).
  
```

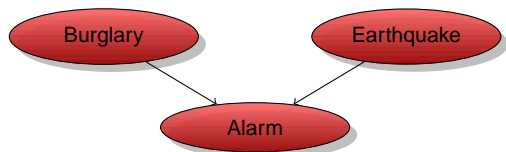
Probability of paths

```

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.
  
```



Encoding Bayesian Networks



burg	t	f
	0.1	0.9

earthq	t	f
	0.2	0.8

alarm	t	f
b=t,e=t	1.0	0.0
b=t,e=f	0.8	0.2
b=f,e=t	0.8	0.2
b=f,e=f	0.1	0.9

`burg(t):0.1 ; burg(f):0.9 .`

`earthq(t):0.2 ; earthqu(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



LPADs to ICL

- Clause C_i with variables \bar{X}

$$H_1 : p_1 \vee \dots \vee H_n : p_n \leftarrow B.$$

is translated into

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

$$\vdots$$

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

$$\text{disjoint}([\text{choice}_{i,1}(\bar{X}) : p_1, \dots, \text{choice}_{i,n}(\bar{X})]).$$



LPADs to ProbLog

- Clause C_i with variables \bar{X}

$$H_1 : p_1 \vee \dots \vee H_n : p_n \leftarrow B.$$

is translated into

$$H_1 \leftarrow B, f_{i,1}(\bar{X}).$$

$$H_2 \leftarrow B, \text{not}(f_{i,1}(\bar{X})), f_{i,2}(\bar{X}).$$

$$\vdots$$

$$H_n \leftarrow B, \text{not}(f_{i,1}(\bar{X})), \dots, \text{not}(f_{i,n-1}(\bar{X})).$$

$$\pi_1 :: f_{i,1}(\bar{X}).$$

$$\vdots$$

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

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

- In general $\pi_i = \frac{p_i}{\prod_{j=1}^{i-1} (1-\pi_j)}$



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 $P(A) = p_1 + p_2 - p_1p_2$
- Example

$sneezing(X) : 0.7 \vee null : 0.3 \leftarrow flu(X).$

$sneezing(X) : 0.8 \vee null : 0.2 \leftarrow hay_fever(X).$

$flu(bob).$

$hay_fever(bob).$

- $P(sneezing(bob)) = 0.7 + 0.8 - 0.7 \times 0.8 = 0.94$
- Particularly useful for modeling independent causes for the same effect



Negation

- 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

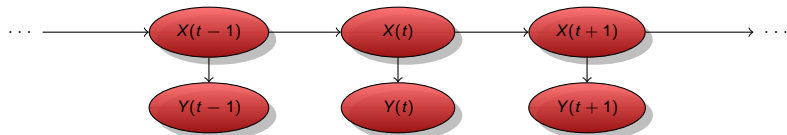
```

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 :-
  T1 is T-1, T1>=0, on(T1,F), \+ on(T1,3).

```



Hidden Markov Models



```
hmm(S,O):-hmm(q1,[],S,O).
```

```
hmm(end,S,S,[]).
```

```
hmm(Q,S0,S,[L|O]):-
```

```
    Q\= end,
```

```
    next_state(Q,Q1,S0),
```

```
    letter(Q,L,S0),
```

```
    hmm(Q1,[Q|S0],S,O).
```

```
next_state(q1,q1,_S):1/3;next_state(q1,q2,_S):1/3;
```

```
next_state(q1,end,_S):1/3.
```

```
next_state(q2,q1,_S):1/3;next_state(q2,q2,_S):1/3;
```

```
next_state(q2,end,_S):1/3.
```

```
letter(q1,a,_S):0.25;letter(q1,c,_S):0.25;
```

```
letter(q1,g,_S):0.25;letter(q1,t,_S):0.25.
```

```
letter(q2,a,_S):0.25;letter(q2,c,_S):0.25;
```

```
letter(q2,g,_S):0.25;letter(q2,t,_S):0.25.
```



Distribution Semantics with Function Symbols

- Semantics proposed for ICL and PRISM, applicable also to the other languages
- Definition of a probability measure μ over W_T
- μ assign a probability to every element of an algebra Ω of subsets of W_T , i.e. a set of subsets closed under union and complementation
- The algebra Ω is the set of sets of worlds identified by a finite set of finite composite choices



Composite Choices

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

$sneezing(X) \leftarrow flu(X), flu_sneezing(X).$

$sneezing(X) \leftarrow hay_fever(X), hay_fever_sneezing(X).$

$flu(bob).$

$hay_fever(bob).$

$C_1 = disjoint([flu_sneezing(X) : 0.7, null : 0.3]).$

$C_2 = disjoint([hay_fever_sneezing(X) : 0.8, null : 0.2]).$

- $\kappa = \{(C_1, \{X/bob\}, 1)\}$, $\omega_\kappa =$

$flu_sneezing(bob).$

$flu_sneezing(bob).$

$hay_fever_sneezing(bob).$

$null.$

$P(w_1) = 0.7 \times 0.8$

$P(w_2) = 0.7 \times 0.2$

- $P(\kappa) = 0.7 = P(w_1) + P(w_2)$



Sets of Composite Choices

- Set of composite choices K
- Set of worlds compatible with K : $\omega_K = \bigcup_{\kappa \in K} \omega_\kappa$
- $\Omega = \{\omega_K \mid K \text{ is finite set of finite composite choices}\}$
- Two composite choices κ_1 and κ_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 κ_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

- $K = \{\kappa_1, \dots, \kappa_n\}$
- $P(K) = P(\kappa_1 \vee \dots \vee \kappa_n)$
- $P(A \vee B) = P(A) + P(B) - P(AB)$
- $P(A \vee B \vee C) = P(A) + P(B) + P(C) - P(AB) - P(BC) + P(ABC)$
- ... (inclusion exclusion formula)
- $P(\kappa_1 \wedge \kappa_2)$ may be:
 - 0, if κ_1, κ_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 $P(\kappa_i \wedge \dots \wedge \kappa_j) = 0$
- $P(K) = P(\kappa_1) + \dots + P(\kappa_n)$



Set of Composite Choices

- Two set 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 $P(K) = P(K')$

- $\Omega = \{\omega_K \mid K \text{ is a finite set of finite composite choices}\}$
- Ω 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

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



Probability of a Query

- Given a query Q , a composite choice κ is an **explanation** for Q if

$$\forall w \in \omega_\kappa \quad w \models Q$$

- A set K of composite choices is **covering** wrt Q if every world in which Q is true belongs to ω_K

Definition

$$P(Q) = \mu(\{w \mid w \in W_T, w \models Q\})$$

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



Example Program (ICL)

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

- Goal $sneezing(bob)$
- $\kappa_1 = \{(C_1, \{X/bob\}, 1)\}$
- $\kappa_2 = \{(C_1, \{X/bob\}, 2), (C_2, \{X/bob\}, 1)\}$
- $K = \{\kappa_1, \kappa_2\}$ mutually exclusive finite set of finite explanations that are covering for $sneezing(bob)$
- $P(Q) = P(\kappa_1) + P(\kappa_2) = 0.7 + 0.3 \cdot 0.8 = 0.94$



Functions Symbols in ICL and PRISM

- 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

$$H_1 : p_1 \vee \dots \vee H_n : p_n \leftarrow B_1, \dots, B_m, \neg C_1, \dots, \neg C_l$$

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

- The CPT of CH_i is

	...	$B_1 = 1, \dots, B_m = 1, C_1 = 0, \dots, C_l = 0$...
$CH_i = H_1$	0.0	p_1	0.0
...			
$CH_n = H_n$	0.0	p_n	0.0
$CH_i = \text{null}$	1.0	$1 - \sum_{i=1}^n p_i$	1.0



Conversion to Bayesian Networks

- 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:

	at least one parent equal to A	remaining columns
$A = 1$	1.0	0.0
$A = 0$	0.0	1.0

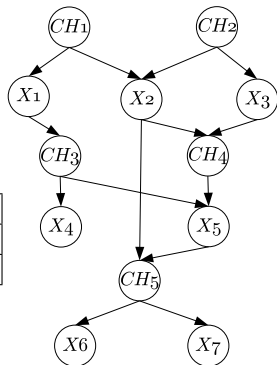


Conversion to Bayesian Networks

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

CH_1, CH_2	x_1, x_2	x_1, x_3	x_2, x_2	x_2, x_3
$x_2 = 1$	1.0	0.0	1.0	1.0
$x_2 = 0$	0.0	1.0	0.0	0.0

x_2, x_5	t,t	t,f	f,t	f,f
$CH_5 = x_6$	0.3	0.0	0.0	0.0
$CH_5 = x_7$	0.2	0.0	0.0	0.0
$CH_5 = null$	0.5	1.0	1.0	1.0



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

$$\begin{array}{lll}
 p : 0.5 \vee q : 0.5 \leftarrow r. & p \leftarrow r. & q \leftarrow r. \\
 r \leftarrow \neg p. & r \leftarrow \neg p. & r \leftarrow \neg p. \\
 r \leftarrow \neg q. & r \leftarrow \neg q. & r \leftarrow \neg q. \\
 & M = \{r, p\} & M = \{r, q\}
 \end{array}$$

- 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

```

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



Knowledge-Based Model Construction

- 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



CLP(BN)

- 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

```
{ Var = Function with p(Values, Dist) }  
{ Var = Function with p(Values, Dist, Parents) }
```



CLP(BN)

```
.....  
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).
```



CLP(BN)

```

.....
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 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)

```

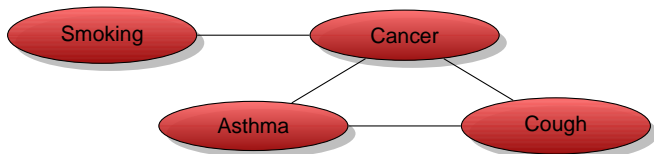
?- [school_32].
   ?- registration_grade(r0,G).
p(G=a)=0.4115,
p(G=b)=0.356,
p(G=c)=0.16575,
p(G=d)=0.06675 ?
?- registration_grade(r0,G),
   student_intelligence(s0,h).
p(G=a)=0.6125,
p(G=b)=0.305,
p(G=c)=0.0625,
p(G=d)=0.02 ?

```



Markov Networks

- Undirected graphical models



- Each clique in the graph is associated with a **potential** ϕ_i

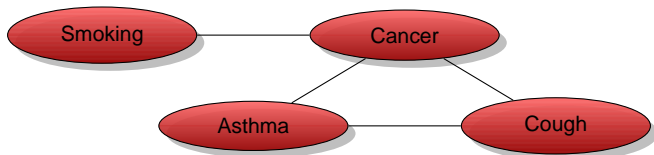
$$P(\mathbf{x}) = \frac{\prod_i \phi_i(\mathbf{x}_i)}{Z}$$

$$Z = \sum_{\mathbf{x}} \prod_i \phi_i(\mathbf{x}_i)$$

Smoking	Cancer	$\phi_i(V, T)$
false	false	4.5
false	true	4.5
true	false	2.7
true	true	4.5



Markov Networks



- If all the potentials are strictly positive, we can use a log-linear model

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

$$Z = \sum_{\mathbf{x}} \prod_i \phi_i(\mathbf{x}_i)$$

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

$$w_i = 1.5$$



Markov Logic

- A Markov Logic Network (MLN) is a set of pairs (F, w) where F is a formula in first-order logic 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

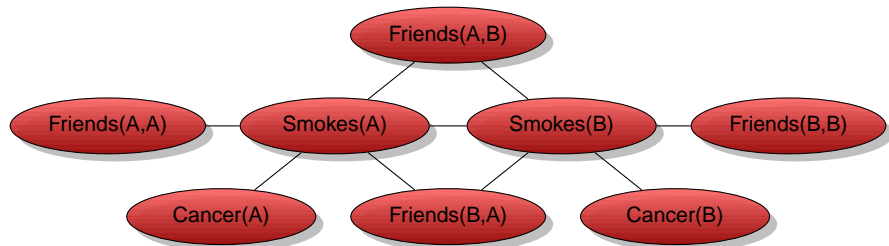


Markov Logic Example

1.5 $\forall x \text{ Smokes}(x) \rightarrow \text{Cancer}(x)$

1.1 $\forall x, y \text{ Friends}(x, y) \rightarrow (\text{Smokes}(x) \leftrightarrow \text{Smokes}(y))$

- Constants Anna (A) and Bob (B)



Markov Networks




- Probability of an interpretation \mathbf{x}

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

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






References I

-  Breese, J. S., Goldman, R. P., and Wellman, M. P. (1994). Introduction to the special section on knowledge-based construction of probabilistic and decision models. *IEEE Transactions On Systems, Man and Cybernetics*, 24(11):1577–1579.
-  C. Baral, Gelfond, M., and Rushton, N. (2009). Probabilistic reasoning with answer sets. *The. Pra. Log. Program.*, 9(1):57–144.
-  Dantsin, E. (1991). Probabilistic logic programs and their semantics. In *Russian Conference on Logic Programming*, volume 592 of *LNCS*, pages 152–164. Springer.



References II

-  De Raedt, L., Kimmig, A., and Toivonen, H. (2007).
Problog: A probabilistic prolog and its application in link discovery.
In International Joint Conference on Artificial Intelligence, pages 2462–2467.
-  Poole, D. (1993).
Logic programming, abduction and probability - a top-down
anytime algorithm for estimating prior and posterior probabilities.
New Gener. Comput., 11(3):377–400.
-  Poole, D. (1997).
The Independent Choice Logic for modelling multiple agents under
uncertainty.
Artif. Intell., 94(1–2):7–56.



References III



Poole, D. (2000).

Abducing through negation as failure: stable models within the independent choice logic.

J. Log. Program., 44(1-3):5–35.



Riguzzi, F. (2009).

Extended semantics and inference for the Independent Choice Logic.

Logic Journal of the IGPL.
to appear.





Sato, T. (1995).

A statistical learning method for logic programs with distribution semantics.

In *International Conference on Logic Programming*, pages 715–729.



References IV

-  Vennekens, J., Denecker, M., and Bruynooghe, M. (2009). Cp-logic: A language of causal probabilistic events and its relation to logic programming. *TPLP*, 9(3):245–308.
-  Vennekens, J., Verbaeten, S., and Bruynooghe, M. (2004). Logic programs with annotated disjunctions. In *International Conference on Logic Programming*, volume 3131 of *LNCS*, pages 195–209. Springer.

