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. Reasoning Tasks
6. Inference for PLP under DS
7. Inference with Tabling
8. Conclusions

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 $F : p \vee null : 1 - p$.
- Example $C_1 = disjoint([flu\_sneezing(X) : 0.7, null : 0.3])$, $(C_1, \{X/bob\}, 1)$
- 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)
- $\sigma = \{(C_1, \{X/bob\}, 1), (C_2, \{bob\}, 1)\}$

$$C_1 = disjoint([flu\_sneezing(X) : 0.7, null : 0.3]).$$
$$C_2 = disjoint([hay\_fever\_sneezing(X) : 0.8, null : 0.2]).$$

- 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 wrolds: $W_T = \{w_1, \ldots, w_m\}$
- $P(w)$ distribution over worlds: $\sum_{w\in W_T} P(w) = 1$

Distribution Semantics

- Herbrand base $H_T = \{A_1, \ldots, A_n\}$
- Query Q: $P(Q|w) = 1$ if $w \models Q$ and 0 otherwise
- $P(Q) = \sum_w P(Q, w) = \sum_w P(Q|w)P(w) = \sum_{w \models Q} P(w)$

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

$sneezing(X) \leftarrow flu(X), flu\_sneezing(X).$
$sneezing(X) \leftarrow hay\_fever(X), hay\_fever\_sneezing(X).$
$flu(bob).$
$hay\_fever(bob).$

$flu\_sneezing(bob).$
$hay\_fever\_sneezing(bob).$ $P(w_1) = 0.7 \times 0.8$

$hay\_fever\_sneezing(bob).$ $P(w_2) = 0.3 \times 0.8$

$flu\_sneezing(bob).$

$P(w_3) = 0.7 \times 0.2$

$P(w_4) = 0.3 \times 0.2$

- $sneezing(bob)$ 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

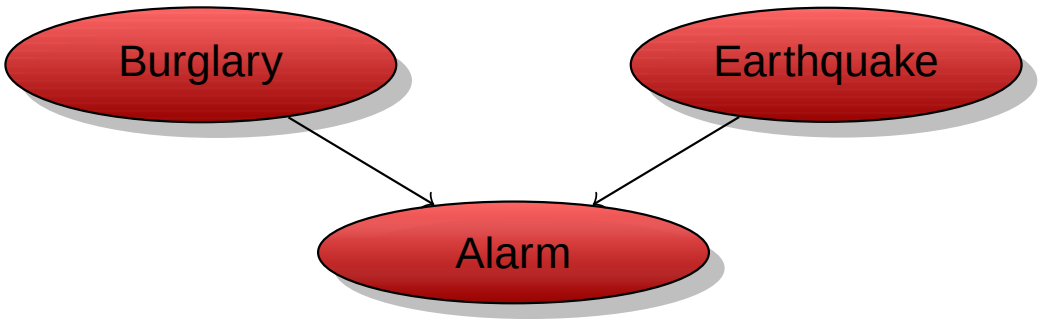
Mendel's inheritance rules for pea plants

```
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 ; 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

LPADs to ICL

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

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

is translated into

$$H_1 \leftarrow B, choice_{i,1}(\overline{X}).$$
$$\vdots$$
$$H_n \leftarrow B, choice_{i,1}(\overline{X}).$$

$$disjoint([choice_{i,1}(\overline{X}) : p_1, \ldots, choice_{i,n}(\overline{X}) : p_n]).$$

LPADs to ProbLog

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

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

is translated into

$$H_1 \leftarrow B, f_{i,1}(\overline{X}).$$
$$H_2 \leftarrow B, not(f_{i,1}(\overline{X})), f_{i,2}(\overline{X}).$$
$$\vdots$$
$$H_n \leftarrow B, not(f_{i,1}(\overline{X})), \ldots, not(f_{i,n-1}(\overline{X})).$$

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

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

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)

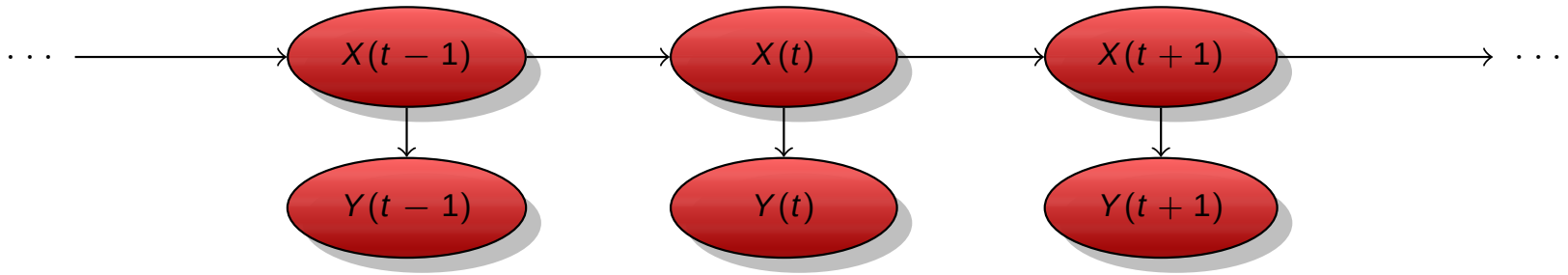
Function Symbols

- 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: $\omega_\kappa = \{w_\sigma \in W_T | \kappa \subseteq \sigma\}$
- 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: $\omega_K = \bigcup_{\kappa \in K} \omega_\kappa$
- $\Omega = \{\omega_K | K \text{ is finite set of finite composite choices}\}$
- 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.
- If K is mutually exclusive, define $P(K) = \sum_{\kappa \in K} P(\kappa)$

Lemma ([Poole, 2000])

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

Probability Measure

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 $\omega_K = \omega_{K'}$.

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

Definition

$\mu : \Omega \to [0, 1]$ 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

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

- 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 $sneezing(bob)$
- $P(Q) = P(\kappa_1) + P(\kappa_2) = 0.7 + 0.3 \cdot 0.8 = 0.94$

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

Inference for PLP under DS

- Computing the probability of a query (no evidence)
- Explanation based:
  - find explanations for queries
  - make the explanations mutually exclusive
    - by means of an iterative splitting algorithm (Ailog2 [Poole, 2000])
    - by means of Binary Decision Diagrams (ProbLog [De Raedt et al., 2007], `cplint` [Riguzzi, 2007, Riguzzi, 2009] PITA [Riguzzi and Swift, 2010])
- Bayesian Network based:
  - Convert to BN
  - Use BN inference algorithms (CVE [Meert et al., 2009])
  - Lifted inference

ProbLog

$$sneezing(X) \leftarrow flu(X), flu\_sneezing(X).$$
$$sneezing(X) \leftarrow hay\_fever(X), hay\_fever\_sneezing(X).$$
$$flu(david).$$
$$hay\_fever(david).$$
$$C_1 = 0.7 :: flu\_sneezing(X).$$
$$C_2 = 0.8 :: hay\_fever\_sneezing(X).$$

- Distributions over facts

Finding Explanations

- All explanations for the query are collected
- ProbLog: source to source transformation for facts, use of dynamic database
- `cplint`: meta-interpretation
- PITA: source to source transformation, addition of an argument to predicates

Explanation Based Inference Algorithm

- $K =$ set of explanations found for Q,
- They are not necessarily mutually exclusive
- The probability of Q is given by the probability of the formula

$$f_K(\mathbf{Y}) = \bigvee_{\kappa \in K} \bigwedge_{(C,\theta,i) \in \kappa} (Y_{C\theta} = i)$$

where $Y_{C\theta}$ is a random variable whose domain is $1, 2$ and $P(Y_{C\theta} = i) = P_0(C, i)$

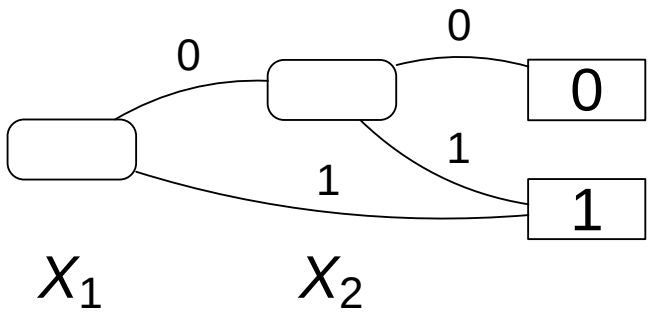
- Binary domain: we use a Boolean variable $X_{C\theta}$ to represent $(Y_{C\theta} = 1)$
- $\neg X_{C\theta}$ represents $(Y_{C\theta} = 2)$

Example

A set of covering explanations for $sneezing(david)$ is $K = \{\kappa_1, \kappa_2\}$
$\kappa_1 = \{(C_1, \{X/david\}, 1)\}$
$\kappa_2 = \{(C_2, \{X/david\}, 1)\}$
$K = \{\kappa_1, \kappa_2\}$
$f_K(\mathbf{Y}) = (Y_{C_1\{X/david\}} = 1) \vee (Y_{C_1\{X/david\}} = 1).$
$X_1 = (Y_{C_1\{X/david\}} = 1)$
$X_2 = (Y_{C_2\{X/david\}} = 1)$
$f_K(\mathbf{X}) = X_1 \vee X_2.$
$P(f_K(\mathbf{X})) = P(X_1 \vee X_2)$
$P(f_K(\mathbf{X})) = P(X_1) + P(X_2) - P(X_1)P(X_2)$

- In order to compute the probability, we must make the explanations mutually exclusive
- [De Raedt et al., 2007]: Binary Decision Diagram (BDD)

Binary Decision Diagrams



$$f_K(\mathbf{X}) = X_1 \times f_K^{X_1}(\mathbf{X}) + \neg X_1 \times f_K^{\neg X_1}(\mathbf{X})$$

$$P(f_K(\mathbf{X})) = P(X_1)P(f_K^{X_1}(\mathbf{X})) + (1 - P(X_1))P(f_K^{\neg X_1}(\mathbf{X}))$$

$$P(f_K(\mathbf{X})) = 0.7 \cdot P(f_K^{X_1}(\mathbf{X})) + 0.3 \cdot P(f_K^{\neg X_1}(\mathbf{X}))$$

Probability from a BDD

- Dynamic programming algorithm [De Raedt et al., 2007]

1: **function** PROB(n)
2: **if** n is a terminal note **then**
3: return $value(n)$
4: **else**
5: return
$\text{PROB}(child_0(n)) \times p(v(n)) + \text{PROB}(child_1(n)) \times (1 - p(v(node)))$
6: **end if**
7: **end function**

Logic Programs with Annotated Disjunctions

$$
\begin{array}{lll}
C_1 = & strong\_sneezing(X) : 0.3 \vee moderate\_sneezing(X) : 0.5 & \leftarrow \quad flu(X). \\
C_2 = & strong\_sneezing(X) : 0.2 \vee moderate\_sneezing(X) : 0.6 & \leftarrow \quad hay\_fever(X). \\
C_3 = & flu(david). & \\
C_4 = & hay\_fever(david). &
\end{array}
$$

- More than two head atoms

Example

A set of covering explanations for $strong\_sneezing(david)$ is
$K = \{\kappa_1, \kappa_2\}$
$\kappa_1 = \{(C_1, \{X/david\}, 1)\}$
$\kappa_2 = \{(C_2, \{X/david\}, 1)\}$
$K = \{\kappa_1, \kappa_2\}$
$X_1 = X_{C_1\{X/david\}}$
$X_2 = X_{C_2\{X/david\}}$
$f_K(\mathbf{X}) = (X_1 = 1) \vee (X_2 = 1)$.
$P(f_X) = P(X_1 = 1) + P(X_2 = 1) - P(X_1 = 1)P(X_2 = 1)$

- To make the explanations mutually exclusive: Multivalued Decision Diagram (MDD)
- Converted to BDD using a transformation similar to LPAD to ProbLog

Tabling

- PITA (Probabilistic Inference with Tabling and Answer subsumption) [Riguzzi and Swift, 2010, Riguzzi and Swift, 2011] (a package of XSB)
- All the explanations for a goal have to be found
- It makes sense to store the explanations for subgoals with tabling
- Associate to each answer (ground atom) a BDD representing its explanations
- Combine BDDs by using the Boolean operators offered by BDD manipulating packages
- Library for manipulating BDD directly in Prolog (interface to CUDD)
- A BDD is represented in Prolog by an integer indicating the address of its root node
- Casting for integer-pointer conversion

Library Predicates

- `init, end`: for allocation and deallocation of a BDD manager
- `zero(-BDD), one(-BDD), and(+BDD1,+BDD2,-BDDO), or(+BDD1,+BDD2,-BDDO), not(+BDDI,-BDDO)`: BDD operations
- `get_var_n(+R,+S,+Probs,-Var)`: returns a ground rule multi-valued random variable
- `equality(+Var,+Value,-BDD)`: BDD represents `Var=Value`
- `ret_prob(+BDD,-P)`: returns the probability of the formula encoded by BDD

Tabling

- Add an extra argument to each atom for storing a BDD
- When an answer $p(\mathbf{x}, bdd)$ is found, bdd represents the explanations for $p(\mathbf{x})$
- If the program is range restricted, $p(\mathbf{x})$ is ground
- Use program transformation to obtain a Prolog program from an LPAD

Answer Subsumption

- Use a lattice on terms to combine different answers for the same goal
- The bottom element and the join operator of the lattice have to be specified in the tabling directives
- E.g `:-table path(X,Y,or/3-zero/1)` means that, if two answers `path(a,b,bdd0)` and `path(a,b,bdd1)` are found, the single answer `path(a,b,bdd)` will be stored in the table where `or(bdd0,bdd1,bdd)`

Program Transformation

- $PITA(p(a,b,c)) = p(a,b,c,D)$
- The disjunctive clause
  $C_r = H_1 : \alpha_1 \vee \ldots \vee H_n : \alpha_n \leftarrow L_1, \ldots, L_m.$
  is transformed into the set of clauses $PITA(C_r)$

$$
\begin{aligned}
PITA(C_r, i) = PITA(H_1) \leftarrow \quad & one(BB_0), \\
& PITA(L_1), and(BB_0, B_1, BB_1), \\
& \ldots, \\
& PITA(L_m), and(BB_{m-1}, B_m, BB_m), \\
& get\_var\_n(r, VC, [\alpha_1, \ldots, \alpha_n], Var), \\
& equality(Var, i, BB), \\
& and(BB_m, BB, BDD).
\end{aligned}
$$

Example

Clause

$strong\_sneezing(X) : 0.3 \vee moderate\_sneezing(X) : 0.5 \leftarrow flu(X).$

is translated into

$$
\begin{aligned}
strong\_sneezing(X, BDD) \leftarrow\ & one(BB_0),\\
& flu(X, B_1), and(BB_0, B_1, BB_1),\\
& get\_var\_n(1, [X], [0.3, 0.5, 0.2], Var),\\
& equality(Var, 1, BB),\\
& and(BB_1, BB, BDD).\\
moderate\_sneezing(X, BDD) \leftarrow\ & one(BB_0),\\
& flu(X, B_1), and(BB_0, B_1, BB_1),\\
& get\_var\_n(1, [X], [0.3, 0.5, 0.2], Var),\\
& equality(Var, 2, BB),\\
& and(BB_1, BB, BDD).
\end{aligned}
$$

Query

- Query: $sneezing(bob)$

$$\leftarrow \quad init, \\ sneezing(bob, BDD), \\ ret\_prob(BDD, P), \\ end.$$

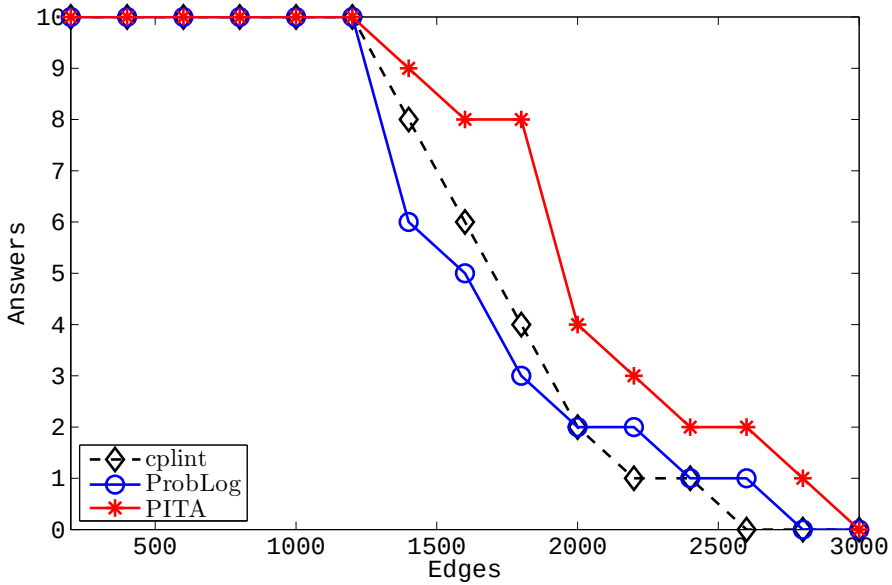
Experiments

- Biomine network: network of biological concepts
- Each edge has a probability
- Dataset from [De Raedt et al., 2007]: 50 sampled subnetworks of size 200, 400, . . ., 10000 edges
- Sampling repeated 10 times
- Linux PCs with Intel Core 2 Duo E6550 (2,333 MHz) and 4 GB of RAM
- Execution stopped after 24 hours

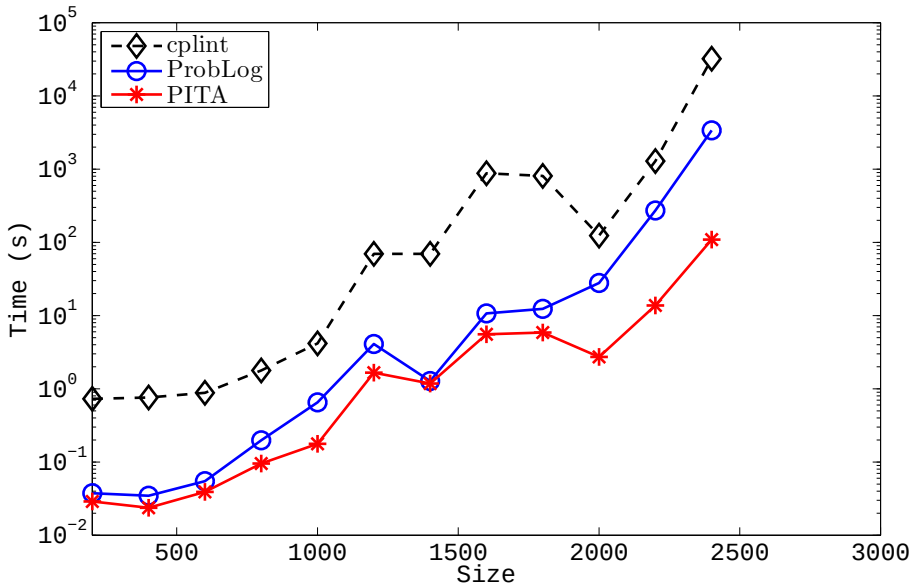
```
path(X,Y) :- path(X,Y,[X],Z).
path(X,Y,V,[Y|V]) :- arc(X,Y).
path(X,Y,V0,V1) :- arc(X,Z),append(V0,_S,V1),
\+ member(Z,V0),path(Z,Y,[Z|V0],V1).
arc(X,Y):-edge(X,Y).
arc(X,Y):-edge(Y,X).
edge('EntrezProtein_33339674','HGNC_620'):0.515062.
```

Dataset from [De Raedt et al., 2007]

Number of solved subgraphs

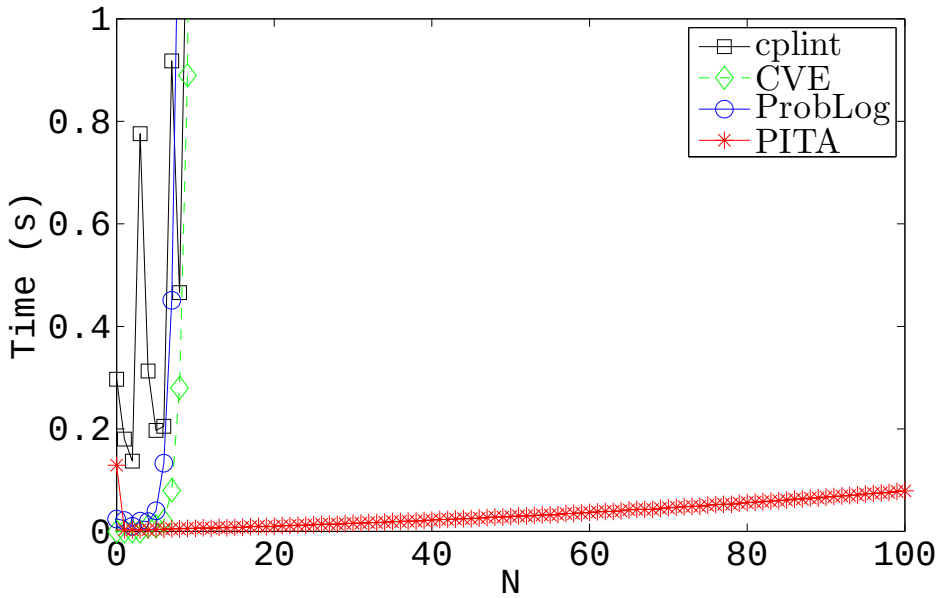


Average time



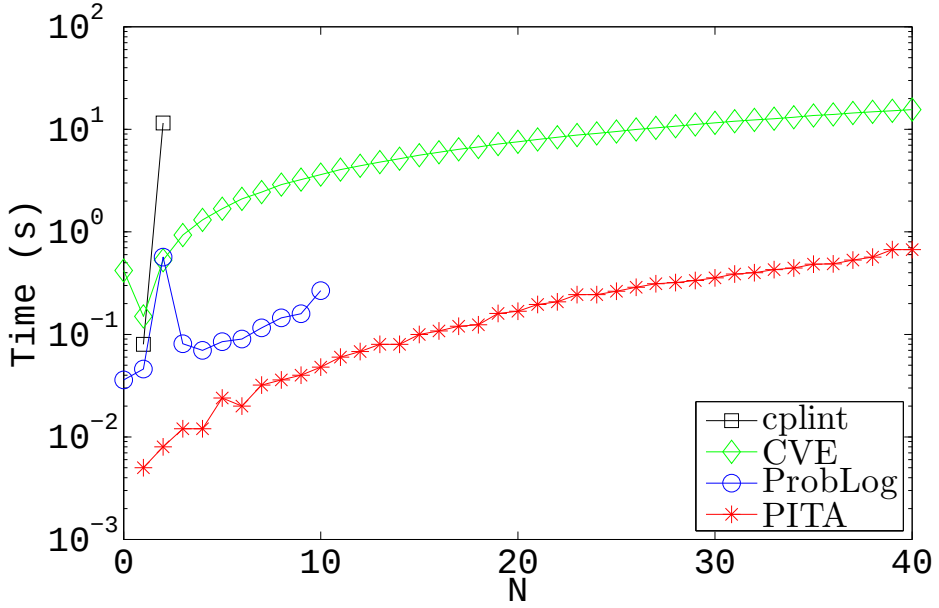
Game of dice

```
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).
```



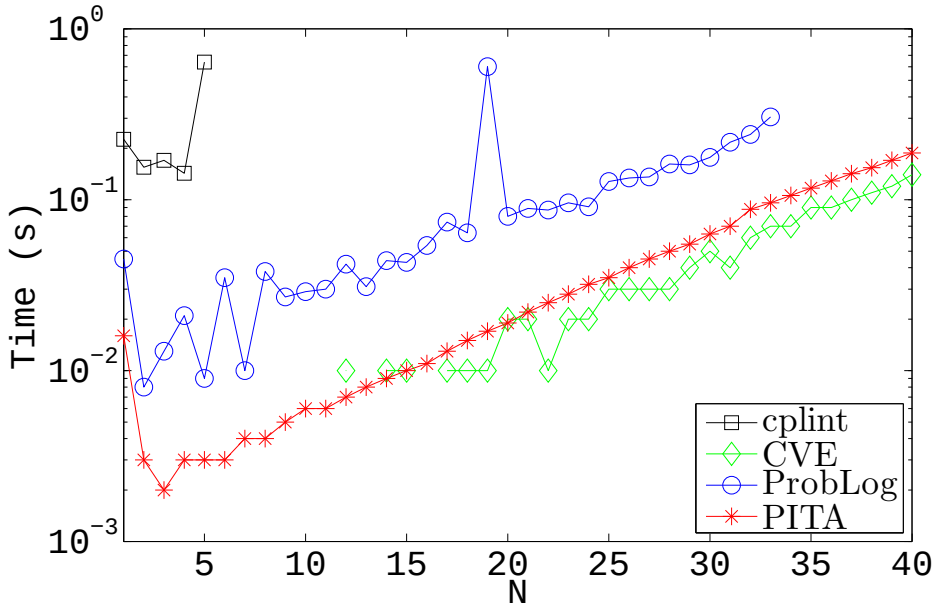
Blood Type [Meert et al., 2009]

```
mchrom(Person,a):0.90 ; mchrom(Person,b):0.05 ; mchrom(Person,null):0.05 :-
  mother(Mother,Person), pchrom(Mother,a   ), mchrom(Mother,a   ).
mchrom(Person,a):0.49 ; mchrom(Person,b):0.49 ; mchrom(Person,null):0.02 :-
  mother(Mother,Person), pchrom(Mother,b   ), mchrom(Mother,a   ).
.....
pchrom(Person,a):0.90 ; pchrom(Person,b):0.05 ; pchrom(Person,null):0.05 :-
  father(Father,Person), pchrom(Father,a   ), mchrom(Father,a   ).
......
bloodtype(Person,a):0.90 ; bloodtype(Person,b):0.03 ; bloodtype(Person,ab):0.03 ;
  bloodtype(Person,null):0.04 :- pchrom(Person,a   ),mchrom(Person,a   ).
bloodtype(Person,a):0.03 ; bloodtype(Person,b):0.03 ; bloodtype(Person,ab):0.90 ;
  bloodtype(Person,null):0.04 :- pchrom(Person,b   ),mchrom(Person,a   ).
```



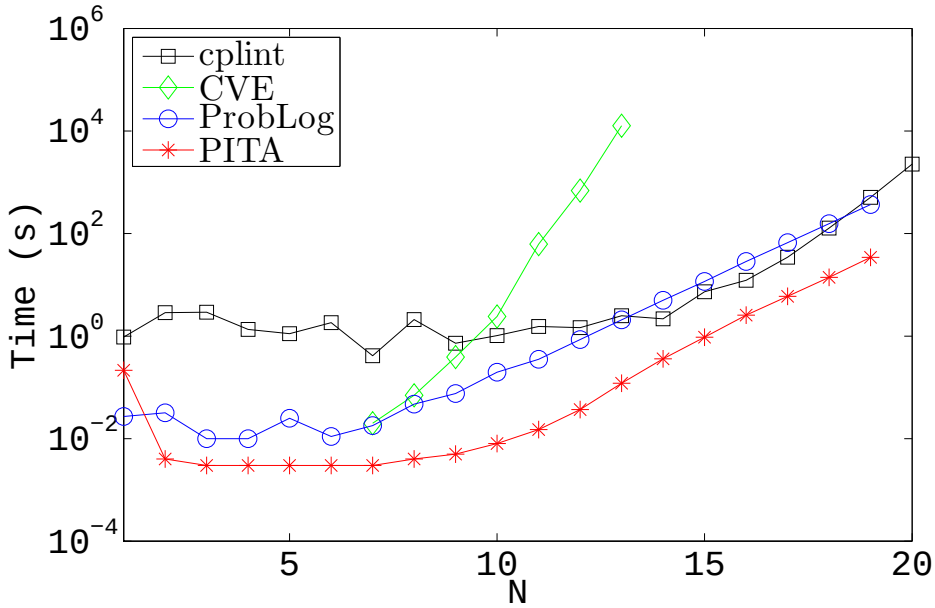
Growing negated body [Meert et al., 2009]

```
a0:0.5 :-  a1.
a0:0.5 :- \+ a1,  a2.
a0:0.5 :- \+ a1, \+ a2,  a3.
a1:0.5 :-  a2.
a1:0.5 :- \+ a2,  a3.
a2:0.5 :-  a3.
a3:0.5.
```



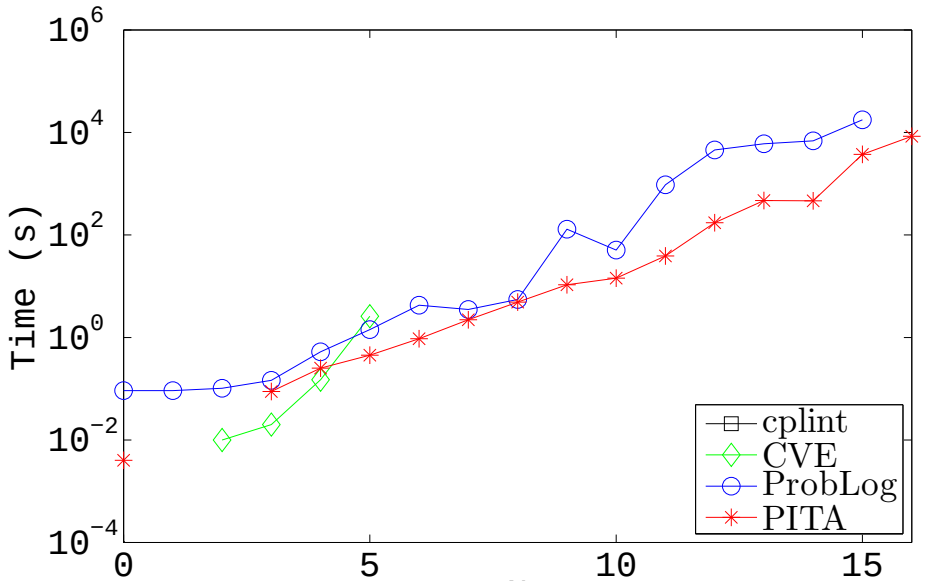
Growing head [Meert et al., 2009]

```
a0 :- a1.
a1:0.5.
a0:0.5; a1:0.5 :- a2.
a2:0.5.
a0:0.333333; a1:0.333333; a2:0.333333 :- a3.
a3:0.5.
```



UWCSE [Meert et al., 2009]

```
course(c1).
professor(p1).
student(s1).
advised_by(A,B):0.10708782742681 :- student(A),professor(B),
  position(B,faculty).
advised_by(A,B):0.027842273781903 :-student(A),professor(B),
  \+position(B,faculty).
course_level(A,level_300):0.066666666666667;
course_level(A,level_400):0.318518518518519;
course_level(A,level_500):0.614814814814815 :-
    course(A).
```



Approximate Inference

- Inference problem is #P hard
- For large models inference is intractable
- Approximate inference
  - Monte Carlo: draw samples of the truth value of the query
  - Iterative deepening: gives a lower and an upper bound
  - Compute only the best k explanations: branch and bound, gives a lower bound

Conclusions

- Probabilistic Logic Programming: Distribution semantics
- ICL, PRISM, LPADs, ProbLog
- Expressive power
- Reasoning tasks

Thank you!
Questions?

References I

Dantsin, E. (1991).
Probabilistic logic programs and their semantics.
In *Russian Conference on Logic Programming*, volume 592 of *LNCS*, pages 152–164. Springer.

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.

Meert, W., Struyf, J., and Blockeel, H. (2009).
CP-Logic theory inference with contextual variable elimination and comparison to bdd based inference methods.
In *ILP 2009*.

References II

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.

Poole, D. (2000).
Abducing through negation as failure: stable models within the independent choice logic.
J. Log. Program., 44(1-3):5–35.

References III

Riguzzi, F. (2007).
A top down interpreter for LPAD and CP-logic.
In *Congress of the Italian Association for Artificial Intelligence*, number 4733 in LNAI, pages 109–120. Springer.

Riguzzi, F. (2009).
Extended semantics and inference for the Independent Choice Logic.
Logic Journal of the IGPL.
to appear.

References IV

Riguzzi, F. and Swift, T. (2010).
Tabling and Answer Subsumption for Reasoning on Logic Programs with Annotated Disjunctions.
In Hermenegildo, M. and Schaub, T., editors, *Technical Communications of the 26th Int'l. Conference on Logic Programming (ICLP'10)*, volume 7 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 162–171, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

Riguzzi, F. and Swift, T. (2011).
The PITA system: Tabling and answer subsumption for reasoning under uncertainty.
Theory and Practice of Logic Programming, 27th International Conference on Logic Programming (ICLP'11) Special Issue, 11(4–5):433–449.

References V

Sato, T. (1995).
A statistical learning method for logic programs with distribution semantics.
In *International Conference on Logic Programming*, pages 715–729.

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.