Reasoning with Probabilistic Logic Languages

Fabrizio Riguzzi

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

1. Reasoning Tasks
2. Inference for PLP under DS
3. Explanation Based Inference Algorithm
4. Inference with Tabling
5. Inference in Simpler Settings
6. Approximate Inference
7. Inference by Conversion to Bayesian Networks
8. Learning Parameters
9. Directions for Future Work
10. Conclusions
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])
  - Lifter inference
sneezing(X) ← flu(X), flu_sneezing(X).
sneezing(X) ← 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
Definitions

- **Composite choice** $\kappa$: consistent set of atomic choices $(C, \theta, i)$ with $i \in \{0, 1\}$
- **Explanation** $\kappa$ for a query $Q$: $Q$ is true in every world compatible with $\kappa$ (every world of $\omega_\kappa$)
- A set of composite choices $K$ **covering** with respect to $Q$: every world $w$ in which $Q$ is true is such that $w \in \omega_K$.
- **Example:**

$$K_1 = \{\{(C_1, \{X/david\}, 1)\}, \{(C_2, \{X/david\}, 1)\}\} \quad (1)$$

is covering for $\text{sneezing}(david)$. 
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

- $K = \text{set of explanations found for } Q$, the probability of $Q$ is given by the probability of the formula

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

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 \textit{sneezing}(david) is \( K = \{ \kappa_1, \kappa_2 \} \)

\[ \kappa_1 = \{(C_1, \{X/david\}, 1)\} \]

\[ \kappa_2 = \{(C_2, \{X/david\}, 1)\} \]

\[ K = \{ \kappa_1, \kappa_2 \} \]

\[ f_K(Y) = (Y_{C_1}\{X/david\} = 1) \lor (Y_{C_1}\{X/david\} = 1). \]

\[ X_1 = (Y_{C_1}\{X/david\} = 1) \]

\[ X_2 = (Y_{C_2}\{X/david\} = 1) \]

\[ f_K(X) = X_1 \lor X_2. \]

\[ P(f_K(X)) = P(X_1 \lor X_2) \]

\[ P(f_K(X)) = P(X_1) + P(X_2) - P(X_1)P(X_2) \]

- 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

\[ f_K(\mathbf{X}) = X_1 \times f_{K}^{X_1}(\mathbf{X}) + \neg X_1 \times f_{K}^{\neg X_1}(\mathbf{X}) \]

\[ P(f_K(\mathbf{X})) = P(X_1)P(f_{K}^{X_1}(\mathbf{X})) + (1 - P(X_1))P(f_{K}^{\neg X_1}(\mathbf{X})) \]

\[ P(f_K(\mathbf{X})) = 0.7 \cdot P(f_{K}^{X_1}(\mathbf{X})) + 0.3 \cdot P(f_{K}^{\neg X_1}(\mathbf{X})) \]
Dynamic programming algorithm [De Raedt et al., 2007]

1: function \textsc{Prob}(n)
2: \textbf{if} \ n \ \textbf{is a terminal note} \ \textbf{then}
3: \hspace{1em} return \ \textit{value}(n)
4: \textbf{else}
5: \hspace{1em} return \ \textsc{Prob}(\textit{child}_0(n)) \times \textit{p}((\textit{v}(n)) + \textsc{Prob}(\textit{child}_1(n)) \times (1 - \textit{p}(\textit{v}(\textit{node})))
6: \textbf{end if}
7: \textbf{end function}
Logic Programs with Annotated Disjunctions

- Distributions over the head of rules
- More than two head atoms

\[ C_1 = \text{strong} \_\text{sneezing}(X) : 0.3 \lor \text{moderate} \_\text{sneezing}(X) : 0.5 \leftarrow \text{flu}(X). \]
\[ C_2 = \text{strong} \_\text{sneezing}(X) : 0.2 \lor \text{moderate} \_\text{sneezing}(X) : 0.6 \leftarrow \text{hay} \_\text{fever}(X). \]
\[ C_3 = \text{flu}(\text{david}). \]
\[ C_4 = \text{hay} \_\text{fever}(\text{david}). \]
A set of covering explanations for \textit{strong_sneezing(david)} is

\[ K = \{ \kappa_1, \kappa_2 \} \]

\[ \kappa_1 = \{(C_1, \{X/david\}, 1)\} \]

\[ \kappa_2 = \{(C_2, \{X/david\}, 1)\} \]

\[ K = \{ \kappa_1, \kappa_2 \} \]

\[ X_1 = X_{C_1}\{X/david\} \]

\[ X_2 = X_{C_2}\{X/david\} \]

\[ f_K(X) = (X_1 = 1) \lor (X_2 = 1). \]

\[ P(f_X) = P(X_1 = 1) + P(X_2 = 1) - P(X_1 = 1)P(X_2 = 1) \]

- To make the explanations mutually exclusive: Multivalued Decision Diagram (MDD)
Multivalued Decision Diagrams

\[ f_K(X) = \bigvee_{i \in |X_1|} (X_1 = i) \land f_K^{X_1=i}(X) \]

\[ P(f_K(X)) = \sum_{i \in |X_1|} P(X_1 = i)P(f_K^{X_1=i}(X)) \]

\[ f_K(X) = (X_1 = 1) \land f_K^{X_1=1}(X) + (X_1 = 2) \land f_K^{X_1=2}(X) + (X_3 = 3) \land f_K^{X_3=1}(X) \]

\[ f_K(X) = 0.3 \cdot P(f_K^{X_1=1}(X)) + 0.5 \cdot P(f_K^{X_1=2}(X)) + 0.2 \cdot P(f_K^{X_3=1}(X)) \]
Manipulating Multivalued Decision Diagrams

- Use an MDD package
- Convert to BDD, use a BDD package: BDD packages more developed, more efficient
- Conversion to BDD
  - Log encoding
  - Binary splits: more efficient
For a variable $X_1$ having $n$ values, we use $n - 1$ Boolean variables $X_{11}, \ldots, X_{1n-1}$.

- $X_1 = i$ for $i = 1, \ldots n - 1$: $\overline{X_{11}} \land \overline{X_{12}} \land \ldots \land \overline{X_{1i-1}} \land X_{1i}$,
- $X_1 = n$: $\overline{X_{11}} \land \overline{X_{12}} \land \ldots \land \overline{X_{1n-1}}$.

**Parameters:** $P(X_{11}) = P(X_1 = 1) \ldots P(X_{1i}) = \frac{P(X_{1}=i)}{\prod_{j=1}^{i-1}(1 - P(X_{1j-1}))}$.
Tabling

- PITA (Probabilistic Inference with Tabling and Answer subsumption) [Riguzzi and Swift, 2010] (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,−BDD0), **or**(+BDD1,+BDD2,−BDD0), **not**(+BDD1,−BDD0): BDD operations
- **add_var**(+N_Val,+Probs,−Var): addition of a new multi-valued variable with N_Val values and parameters Probs
- **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 $p(x, bdd)$ is found, $bdd$ represents the explanations for $p(x)$
- If the program is range restricted, $p(x)$ 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.: \texttt{-table path(X,Y,or/3-zero/1)} means that, if two answers \texttt{path(a,b,bdd0)} and \texttt{path(a,b,bdd1)} are found, the single answer \texttt{path(a,b,bdd)} will be stored in the table where \texttt{or(bdd0,bdd1,bdd)}. 
Program Transformation

- Atom $A = p(t)$: $PITA(A) = p(t, BDD)$
- Literal $\neg A$: $PITA(\neg A) = (PITA(A) \rightarrow one(BDD); not(BDD, BDD'))$
- $get\_var\_n(+R,+S,+Probs,-Var)$ wraps $add\_var/3$

\[
get\_var\_n(R, S, Probs, Var) \leftarrow \\
(var(R, S, Var) \rightarrow true \\
; \\
length(Probs, L), \\
add\_var(L, Probs, Var), \\
assert(var(R, S, Var)) \\
).
\]
Program Transformation

The disjunctive clause

\[ C_r = H_1 : \alpha_1 \lor \ldots \lor H_n : \alpha_n \leftarrow L_1, \ldots, L_m. \]

is transformed into the set of clauses \( \text{PITA}(C_r) \)

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

\[ \ldots \]

\[ \text{PITA}(C_r, n) = \text{PITA}(H_n) \leftarrow \text{one}(BB_0), \]
\[ PIMA(L_1), \text{and}(BB_0, B_1, BB_1), \]
\[ \ldots, \]
\[ PIMA(L_m), \text{and}(BB_{m-1}, B_m, BB_m), \]
\[ \text{get\_var\_n}(r, VC, [\alpha_1, \ldots, \alpha_n], \text{Var}), \]
\[ \text{equality}(\text{Var}, n, BB), \text{and}(BB_m, BB, BDD). \]
Example

Clause
\[
\text{strong\_sneezing}(X) : 0.3 \lor \text{moderate\_sneezing}(X) : 0.5 \leftarrow \text{flu}(X).
\]
is translated into

\[
\text{strong\_sneezing}(X, BDD) \leftarrow \begin{align*}
on(\text{BB}_0), \\
\text{flu}(X, B_1), \text{and}(\text{BB}_0, B_1, \text{BB}_1), \\
\text{get\_var\_n}(1, [X], [0.3, 0.5, 0.2], \text{Var}), \\
\text{equality}(\text{Var}, 1, \text{BB}), \\
\text{and}(\text{BB}_1, \text{BB}, \text{BDD}).
\end{align*}
\]

\[
\text{moderate\_sneezing}(X, BDD) \leftarrow \begin{align*}
on(\text{BB}_0), \\
\text{flu}(X, B_1), \text{and}(\text{BB}_0, B_1, \text{BB}_1), \\
\text{get\_var\_n}(1, [X], [0.3, 0.5, 0.2], \text{Var}), \\
\text{equality}(\text{Var}, 2, \text{BB}), \\
\text{and}(\text{BB}_1, \text{BB}, \text{BDD}).
\end{align*}
\]
Example

\[
\text{path}(X,X).
\]
\[
\text{path}(X,Y) :\neg \text{path}(X,Z), \text{edge}(Z,Y).
\]
\[
\text{edge}(a,b):0.3.
\]

\[
:\neg \text{table \ path}(X,Y,\text{or}/3-\text{zero}/1), \text{edge}(X,Y,\text{or}/3-\text{zero}/1).
\]
\[
\text{path}(X,X,\text{One}):\neg \text{one}(\text{One}).
\]
\[
\text{path}(X,Y,\text{BDD}) :\neg \text{path}(X,Z,\text{BDD0}), \text{edge}(Z,Y,\text{BDD1}),\text{and}(\text{BDD0},\text{BDD1},\text{BDD}).
\]
\[
\text{edge}(a,b,\text{BDD}):\neg.
\]
\[
\text{get\_var}(3,[],[0.3,0.7],\text{Var}),
\]
\[
\text{equality}(\text{Var},0,\text{BDD}).
\]

\[
\text{....}
\]
Query: path(a, b)

:-init,
    path('HGNC_620','HGNC_983', 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_3339674','HGNC_620'):0.515062.
```
Inference with Tabling

Dataset from [De Raedt et al., 2007]

Number of solved subgraphs

Average time

500 1000 1500 2000 2500 3000

Edges

0 1 2 3 4 5 6 7 8 9

Answers

500 1000 1500 2000 2500 3000

Size

0 1 2 3 4 5 6 7 8 9

Time (s)

Fabrizio Riguzzi (University of Ferrara)
Game of dice

\[
\begin{align*}
on(0,1) & : 1/3 ; \ on(0,2) : 1/3 ; \ on(0,3) : 1/3. \\
on(T,1) & : 1/3 ; \ on(T,2) : 1/3 ; \ on(T,3) : 1/3 : \- \\
T1 & \text{ is } T-1, \ T1 \geq 0, \ on(T1,F), \ \ or \ on(T1,3).
\end{align*}
\]
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]

\[
\begin{align*}
  a_0 & : 0.5 : - a_1. \\
  a_0 & : 0.5 : - \neg a_1, a_2. \\
  a_0 & : 0.5 : - \neg a_1, \neg a_2, a_3. \\
  a_1 & : 0.5 : - a_2. \\
  a_1 & : 0.5 : - \neg a_2, a_3. \\
  a_2 & : 0.5 : - a_3. \\
  a_3 & : 0.5. 
\end{align*}
\]
Growing head [Meert et al., 2009]

\[ \begin{align*}
a_0 & :- a_1. \\
a_1 & : 0.5. \\
a_0 & : 0.5; a_1 & : 0.5 :- a_2. \\
a_2 & : 0.5. \\
a_0 & : 0.333333; a_1 & : 0.333333; a_2 & : 0.333333 :- a_3. \\
a_3 & : 0.5. 
\end{align*} \]
Inference with Tabling

UWCSE [Meert et al., 2009]

course(c1).
professor(p1).
student(s1).

\texttt{advised\_by(A,B):0.10708782742681 :- student(A),professor(B), position(B,faculty).}

\texttt{advised\_by(A,B):0.0278422273781903 :-student(A),professor(B), +position(B,faculty).}

course\_level(A,\text{level\_300}):0.0666666666666667;
course\_level(A,\text{level\_400}):0.318518518518519;
course\_level(A,\text{level\_500}):0.614814814814815 :- course(A).}
Simpler setting: PRISM

- The PRISM system considers a simpler setting:
  - The probability of a conjunction \((A,B)\) is computed as the product of the probabilities of \(A\) and \(B\) (independence assumption).
  - The probability of a disjunction \((A;B)\) is computed as the sum of the probabilities of \(A\) and \(B\) (exclusiveness assumption).

- The program has to be written so that these requirements are met.
- Not always possible.
Simpler setting: PRISM

- Not all programs satisfy the two conditions
- Coin, Pea plants, Blood type, Growing negated body satisfy both
- Russian roulette satisfies and
- Dice satisfies or
- Path, Growing head, UWCSE does not satisfy any

\[
p :- a, b. \\
q :- a, b. \\
a : 0.3 ; b : 0.4. \\
a :- c. \\
b :- c. \\
c : 0.2. \\
\]

- do not satisfy and: \( P(p) = 0, P_{PRISM}(p) = 0.12, P(q) = 0.2, P_{PRISM}(p) = 0.04 \)
PRISM simpler setting

- PITA can be optimized for PRISM simpler setting
- The disjunctive clause
  \[ C_r = H_1 : \alpha_1 \lor \ldots \lor H_n : \alpha_n \leftarrow L_1, \ldots, L_m. \]
  is transformed into the set of clauses \( PITA'(C_r) \)

\[
PITA'(C_r, 1) = PITA(H_1) \leftarrow one(BB_0),
\]
\[
PITA(L_1), \text{and}(BB_0, B_1, BB_1), \ldots,
\]
\[
PITA(L_m), \text{and}(BB_{m-1}, B_m, BB_m),
\]
\[
equality([\alpha_1, \ldots, \alpha_n], 1, BB),
\]
\[
\text{and}(BB_m, BB, B).
\]

\[
\ldots
\]

\[
PITA'(C_r, n) = PITA(H_n) \leftarrow one(BB_0),
\]
\[
PITA(L_1), \text{and}(BB_0, B_1, BB_1), \ldots,
\]
\[
PITA(L_m), \text{and}(BB_{m-1}, B_m, BB_m),
\]
\[
equality([\alpha_1, \ldots, \alpha_n], n, BB),
\]
\[
\text{and}(BB_m, BB, B).
\]
Inference in Simpler Settings

PRISM simpler setting

equality(Probs,N,P):- nth0(N,Probs,P).
or(A,B,C):- C is A+B.
and(A,B,C):- C is A*B.
zero(0.0).
one(1.0).
not(P,P1):- P1 is 1-P.
ret_prob(P,P).
Hidden Markov Models

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

\[
\text{hmm}(O) :- \text{hmm1}(_{-}, O).
\]
\[
\text{hmm1}(S, O) :- \text{hmm}(q1, [], S, O).
\]
\[
\text{hmm}(\text{end}, S, S, []).
\]
\[
\text{hmm}(Q, S0, S, [L | O]) :-
\]
\[
\quad Q \neq \text{end},
\quad \text{next_state}(Q, Q1, S0),
\quad \text{letter}(Q, L, S0),
\quad \text{hmm}(Q1, [Q | S0], S, O).
\]
\[
\text{next_state}(q1, q1, _S): 1/3; \text{next_state}(q1, q2, _S): 1/3;
\]
\[
\quad \text{next_state}(q1, \text{end}, _S): 1/3.
\]
\[
\text{next_state}(q2, q1, _S): 1/3; \text{next_state}(q2, q2, _S): 1/3;
\]
\[
\quad \text{next_state}(q2, \text{end}, _S): 1/3.
\]
\[
\text{letter}(q1, a, _S): 0.25; \text{letter}(q1, c, _S): 0.25;
\]
\[
\quad \text{letter}(q1, g, _S): 0.25; \text{letter}(q1, t, _S): 0.25.
\]
\[
\text{letter}(q2, a, _S): 0.25; \text{letter}(q2, c, _S): 0.25;
\]
\[
\quad \text{letter}(q2, g, _S): 0.25; \text{letter}(q2, t, _S): 0.25.
\]
HMM

Time for computing $P(hmm([a, \ldots, a])$ as a function of sequence length

Exponential cost
The optimized PITA can be used to count explanations when explanations for different goals can not be incompatible.

We have to modify \texttt{equality} as \texttt{equality(\_Probs,\_N,1)}.

In the Biomine network, series 1, the number of paths is:

<table>
<thead>
<tr>
<th>Edges</th>
<th>200</th>
<th>400</th>
<th>600</th>
<th>800</th>
<th>1000</th>
<th>1200</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explanations</td>
<td>10</td>
<td>42</td>
<td>380</td>
<td>1280</td>
<td>3,480</td>
<td>61,2140</td>
<td>...</td>
</tr>
</tbody>
</table>

The definition of \texttt{path} implies that these are also the counts of the number of distinct paths from source to target that do not contain loops.
Further Optimization

- [Christiansen and Gallagher, 2009] proposed to remove non-discriminating arguments, resulting in a program whose computation trees are isomorphic to those of the original program.
- The results of the original program can be reconstructed from trace of the transformed program.
- Useful with tabling: calls of a tabled predicate differing only in the non-discriminating arguments will merge into a single call.
- Much smaller table and larger chance that the current call has a match in the table.

```prolog
hmm(O) :- hmm(q1,O).
hmm(end,[]).
hmm(Q,[L|O]) :-
    Q\= end,
    next_state(Q,Q1,S0),
    letter(Q,L,S0),
hmm(Q1,O).
```
Inference in Simpler Settings

HMM

Time for computing $P(hmm([a, \ldots, a])$ as a function of sequence length
It should increase linearly
Computing the Viterbi Path

- Viterbi path: most probable explanation, its probability is the Viterbi probability

```prolog
equality(R,S,Probs,N,e([(R,S,N)],P)):-
    nth0(N,Probs,P).
or(e(E1,P1),e(_E2,P2),e(E1,P1)):- P1 >=P2, !.
or(e(_E1,_P1),e(E2,P2),e(E2,P2)).
and(e(E1,P1),e(E2,P2),e(E3,P3)):-
    P3 is P1*P2,
    append(E1,E2,E3).
zero(e(null,0)).
zero(e([],1)).
ret_prob(B,B).
```
HMM Viterbi Path

Time for computing the Viterbi path and probability of $hmm([a, \ldots, a])$ as a function of sequence length. It should depend linearly from the sequence length.
Possibilistic Logic

- $\Pi(\phi)$: **possibility** of a logical formula $\phi$, the degree of compatibility of $\phi$ with the available knowledge
- $N(\phi)$: **necessity** of a logical formula $\phi$, the degree of certainty of $\phi$ given the available knowledge
- **Possibilistic Logic Program**: set of formulas for the form $(\phi, \alpha)$ where $\phi$ is a program clause
  
  $H \leftarrow L_1, \ldots, L_n.$

- **Meaning of $(\phi, \alpha)$**: $N(\phi) \geq \alpha$
- **Inference**: compute the maximum value of $\alpha$ such that $N(Q) \geq \alpha$ holds for a query $Q$. 
Inference rules:
1. \((\phi, \alpha), (\psi, \beta) \vdash (R(\phi, \phi), \min(\alpha, \beta))\) where \(R(\phi, \phi)\) is the resolvent of \(\phi\) and \(\psi\)
2. \((\phi, \alpha), (\phi, \beta) \vdash (\phi, \max(\alpha, \beta))\)

In PITA, interpret the formula \(H : \alpha \leftarrow B_1, \ldots, B_n\) as
\((H \leftarrow B_1, \ldots, B_n, \alpha)\)

- `equality([P|T],_N,P).`
- `or(A,B,C):- C is max(A,B).`
- `and(A,B,C):- C is min(A,B).`
- `zero(0.0).`
- `one(1.0).`
- `ret_prob(P,P).`
PITA for Possibilistic Logic

The possibilistic program

\[
\text{path}(X, X).
\]
\[
\text{path}(X, Y) :- \text{path}(X, Z), \text{edge}(Z, Y).
\]
\[
\text{edge}(a, b): 0.3.
\]

computes the least unsure path in a graph, i.e., the path with maximal weight, the weight of a path being the weight of its weakest link.
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
The disjunctive clause
\[ C_r = H_1 : \alpha_1 \lor \ldots \lor H_n : \alpha_n \leftarrow L_1, \ldots, L_m. \]
is transformed into the set of clauses \( MC(C_r) \)
\[ MC(C_r, 1) = H_1 \leftarrow L_1, \ldots, L_m, \text{sample\_head}(n, r, VC, NH), NH = 1. \]
\[ \ldots \]
\[ MC(C_r, n) = H_1 \leftarrow L_1, \ldots, L_m, \text{sample\_head}(n, r, VC, NH), NH = n. \]

**Definition of sample\_head:**
\[
\text{sample\_head}(\text{NHead}, R, VC, NH) :-
\text{exp(Exp)}, \text{member}((R, VC, NH), Exp), !.
\]
\[
\text{sample\_head}(\text{NHead}, R, VC, NH) :- \text{sample(}\text{NHead}, \text{NH}),
\text{retract}(\text{exp(Exp)}), \text{assert(}\text{exp([[(R, VC, NH) | Exp]])}).
\]

**Sample truth value of query \( Q \):**
\[
\ldots
\]
\[
\text{assert(}\text{exp([])}) ,
\text{(call(Q)} \rightarrow \text{NT1 is NT+1 ; NT1 =NT),}
\text{retract(}\text{exp(_E)}) ,
\ldots
\]
Monte Carlo

- The proportion of successes in a Bernoulli trial process is in the binomial proportion confidence interval

\[ \hat{p} \pm z_{1-\alpha/2} \sqrt{\frac{\hat{p}(1-\hat{p})}{n}} \]

- Algorithm:
  
  - \( n := 0, \ nt := 0 \)
  
  - Repeat
    
      - Test query \( n' \) times, \( nt' \) successes
      
      - \( n := n + n' \), \( nt := nt + nt' \), \( \hat{p} = nt/n \)
      
      - Compute interval size \( s \)
    
  - until \( s < \delta \)

  - return \( \hat{p}, s \)
Inference by Conversion to Bayesian Networks

- Convert the program to a BN, perform inference on the BN with belief propagation, variable elimination, etc.
- Problem: grounding the program
- With function symbols, infinite grounding
- Even without function symbols, the grounding can be huge (exponential size)
- Most of the network is irrelevant to the query
Grounding

- Use a lifted inference algorithm
- Build only the relevant network and apply an inference algorithm
- Combination of the two approaches
Lifted Belief Propagation

- Belief propagation: nodes exchange messages, at convergence the marginal probability of each node can be extracted
- Correct for polytrees, approximate for general DAGs
- Lifted Belief Propagation: exploit the symmetries in the network to group nodes that exchange equal or similar messages into super nodes
- Perform belief propagation between super nodes taking into account the cardinalities of the messages
Bayes Ball [Shachter, 1998]: algorithm for identifying the portion of a network that is relevant to query and evidence

First-Order Bayes Ball [Meert et al., 2010]: lifted version of Bayes Ball

Then apply a (lifted) inference algorithm
Learning Parameters

- Problem: given a set of interpretations, a program, find the parameters maximizing the likelihood of the interpretations (or of instances of a target predicate)
- Exploit the equivalence with BN to use BN learning algorithms
- The interpretations record the truth value of ground atoms, not of the choice variables
- Unseen data: relative frequency can’t be used
- An Expectation-Maximization algorithm must be used:
  - Expectation step: the distribution of the unseen variables in each instance is computed given the observed data
  - Maximization step: new parameters are computed from the distributions using relative frequency
  - End when likelihood does not improve anymore
[Thon et al., 2008] proposed an adaptation of EM for CPT-L, a simplified version of LPADs.

The algorithm computes the counts efficiently by repeatedly traversing the BDDs representing the explanations.

[Ishihata et al., 2008] independently proposed a similar algorithm.

COPREM [Gutmann et al., 2010] is the adaptation of EM to ProbLog.
Learning Parameters

- EM can get trapped into local maxima
- Information Bottleneck: uses an evaluation function with a parameter
- When the parameter is 0, the maximum is easy to find
- When the parameter is 1, the function is the EM evaluation function, difficult to optimize
- Optimize the function with a deterministic annealing strategy: start with the parameter = 0 and then gradually increase it to 1, in the hope of finding an optimum better than EM
- Application to LPADs: Relational Information Bottleneck [Riguzzi and Mauro, 2010]
Directions for Future Works

- Approximate inference: iterative deepening, best-$K$
- Lifted inference for PLP: lifted variable elimination, lifted (loopy) belief propagation, first-order Bayes ball
- PLP structure learning
Approximate Inference

- Iterative deepening: build the SLG forest only up to a certain depth,
- Completed derivation give a lower bound, completed plus incomplete derivations an upper bound
- How to do it efficiently?
- Best-$k$ explanations: with SLD, each time an explanation is found, update the set of explanations
- Cut a derivation if its probability falls below that of the $k$-th best explanation
- With PITA, you can’t simply record the best-$k$ explanation for each goal
- The best-$k$ explanations for a supergoal may include other subgoals explanations.
- Sub case: best-1 explanation: Viterbi explanation
Conclusions

- Tabling and answer subsumption very useful for exact probabilistic inference
- More investigation for approximate inference
- Lifted inference
- Learning algorithms
References I


References II


References III


References IV

