ProbLog: A Probabilistic Prolog and its Application in Link Discovery

Max Le Blansch, Bogdan Simion

TU Delft

Paper context

- At the time when the paper was released, there were no programs for modelling the exact inference for discrete variables
- Discrete variables require separate rules than the continuous variables

Intro to Prolog

- Part of the logical programming languages family
- Program consists of a set of definite clauses
- Programs can contain the following: rules, facts and variables
- Clauses can be only True or False

Prolog example

```
burglary.
hears_alarm(mary).

alarm :- burglary.
alarm :- earthquake.

calls(X) :- alarm, hears_alarm(X).
call :- calls(X).
```

```
calls(mary).
true
calls(john).
false
```

- Alarm and calls are called rules
- hears_alarm, burglary are called facts
- Mary is a variable

Why extending Prolog to Probabilistic Programming?

- Adding probabilities to clauses is closer to real-world problems
- Probabilistic Database is slow -> 10 or more conjuncts are infeasible to compute
- Many practical applications (i.e. life sciences) require computing probabilities in network relations

Intro to ProbLog

- Built on top of Prolog, both being very similar
- Only major difference: Problog has probabilities of success attached to the clauses
- It has equivalent functions for sample and observe (can you spot them in the next slide?)

ProbLog example

```
1.0:: likes(X,Y):- friendof(X,Y).
0.8:: likes(X,Y):- friendof(X,Z), likes(Z,Y).
0.5:: friendof(john,mary).
0.5:: friendof(mary,pedro).
0.5:: friendof(pedro,tom).

evidence(likes(john, pedro), false).

query(likes(mary, tom)).
```

What are sample and observe here?

Query ▼	Location	Probability
likes(mary,tom)	11:7	0.15

Screenshots taken from: https://dtai.cs.kuleuven.be/problog/tutorial/basic/02_bayes.html (more examples there as well)

TU Delft

Computing queries

Two steps:

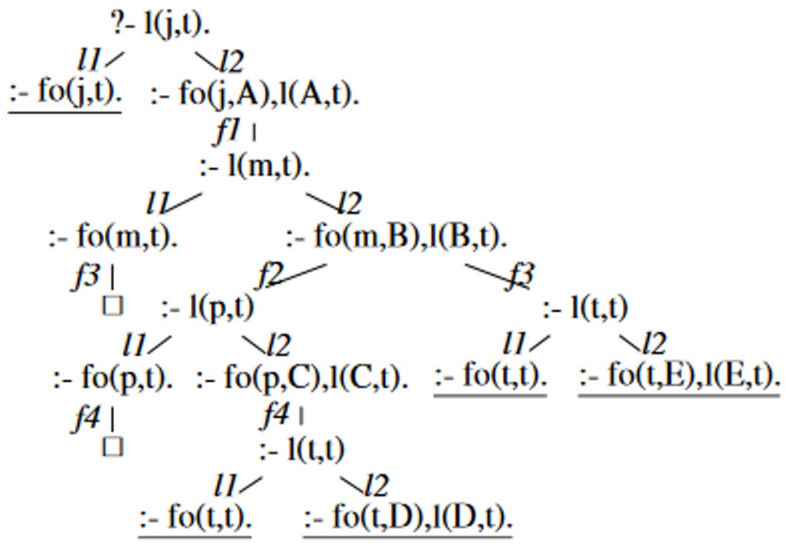
1. Build monotone DNF formula representing all solutions
2. Compute the probability of this DNF formula

Computing queries

Two steps:

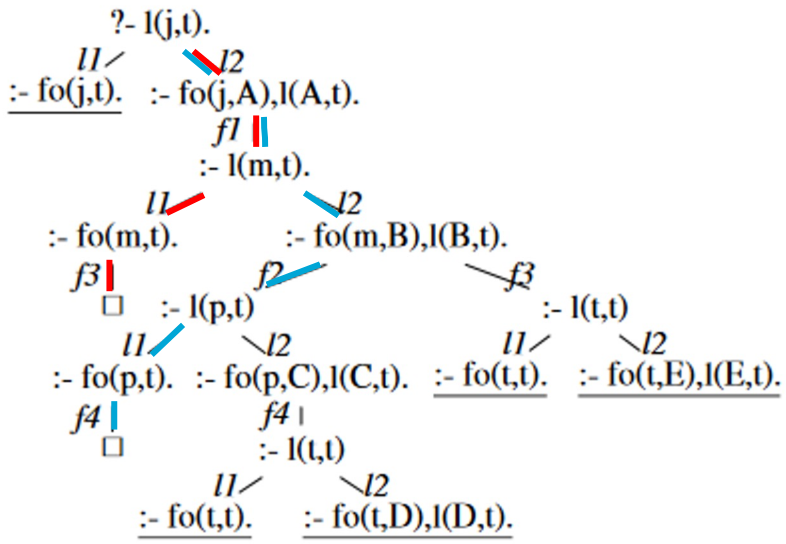
1. Build monotone DNF formula representing all solutions
   - SLD-resolution to transform query into equivalent tree
     - Root is query to be proven
     - Recursively generate subgoals
   - Use the disjunction of proof paths in tree as DNF
2. Compute the probability of this DNF formula

Computing queries | SLD-resolution example



1.0: likes(X,Y):- friendof(X,Y).
0.8: likes(X,Y):- friendof(X,Z), likes(Z,Y).
0.5: friendof(john,mary).
0.5: friendof(mary,pedro).
0.5: friendof(mary,tom).
0.5: friendof(pedro,tom).

Computing queries | SLD-resolution example



1.0: likes(X,Y):- friendof(X,Y).
0.8: likes(X,Y):- friendof(X,Z), likes(Z,Y).
0.5: friendof(john,mary).
0.5: friendof(mary,pedro).
0.5: friendof(mary,tom).
0.5: friendof(pedro,tom).

$$P(\underline{(l_1 \wedge l_2 \wedge f_1 \wedge f_2 \wedge f_4)} \vee \underline{(l_1 \wedge l_2 \wedge f_1 \wedge f_3)}).$$

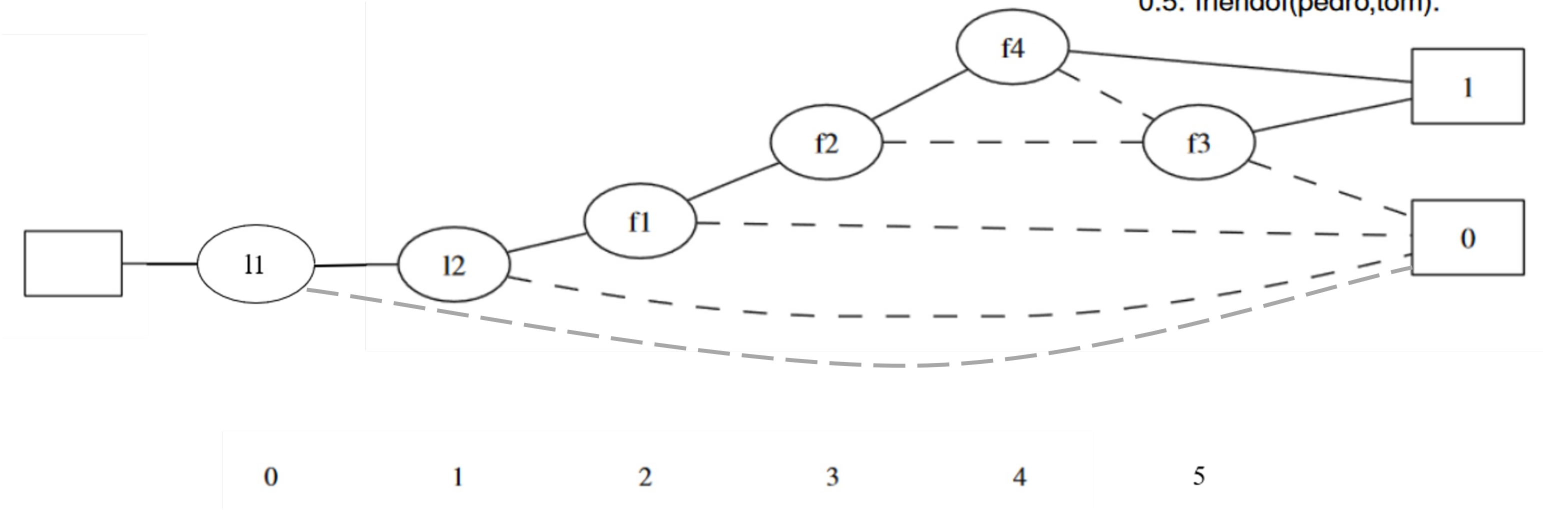
Computing queries

Two steps:

1. Build monotone DNF formula representing all solutions
2. Compute the probability of this DNF formula
   - Using Binary Decision Diagram (BDD) representation
     - Start from full binary tree, merging isomorphic subgraphs and deleting redundant nodes

Computing queries | BDD calculation example

1.0: likes(X,Y):- friendof(X,Y).
0.8: likes(X,Y):- friendof(X,Z), likes(Z,Y).
0.5: friendof(john,mary).
0.5: friendof(mary,pedro).
0.5: friendof(mary,tom).
0.5: friendof(pedro,tom).

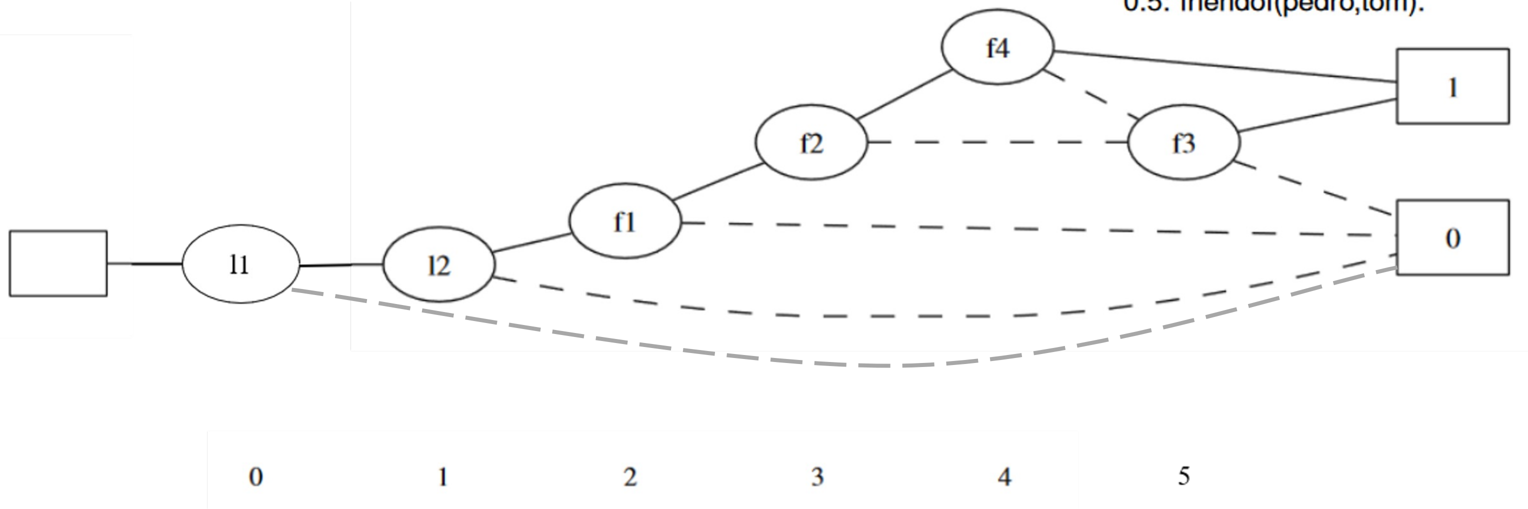


$$P((l_1 \wedge l_2 \wedge f_1 \wedge f_2 \wedge f_4) \vee (l_1 \wedge l_2 \wedge f_1 \wedge f_3)).$$

Computing queries | BDD calculation example

What node is redundant in this tree?

```
1.0: likes(X,Y):- friendof(X,Y).
0.8: likes(X,Y):- friendof(X,Z), likes(Z,Y).
0.5: friendof(john,mary).
0.5: friendof(mary,pedro).
0.5: friendof(mary,tom).
0.5: friendof(pedro,tom).
```



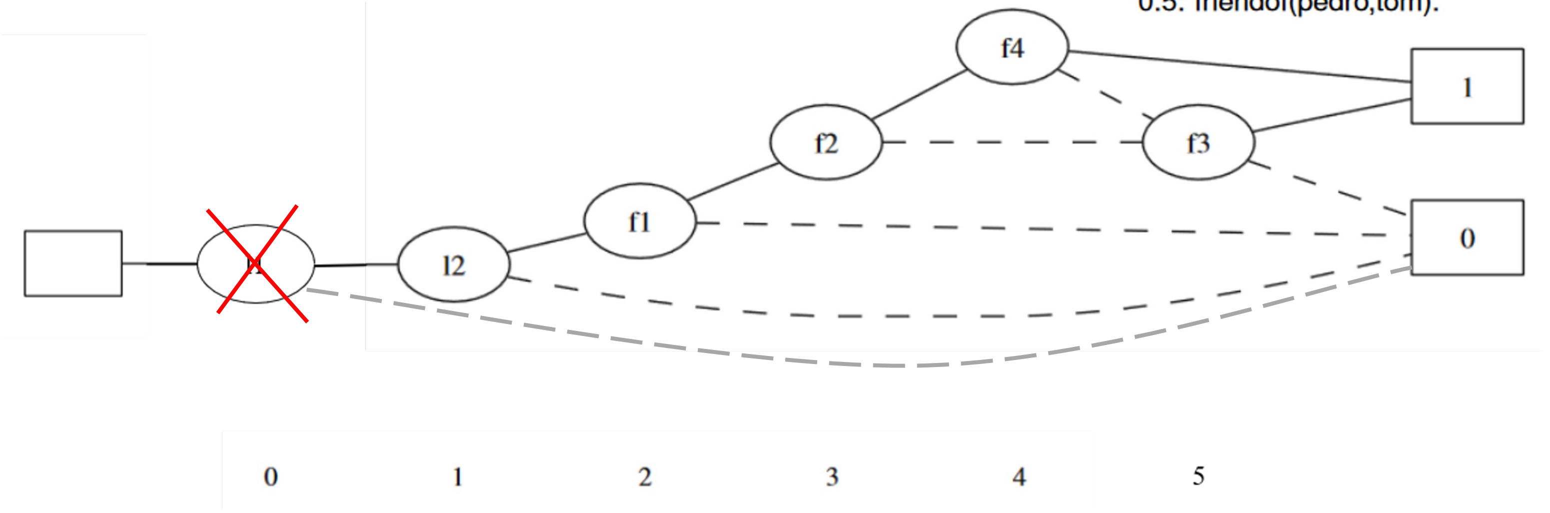
$$P((l_1 \wedge l_2 \wedge f_1 \wedge f_2 \wedge f_4) \vee (l_1 \wedge l_2 \wedge f_1 \wedge f_3)).$$

TU Delft

Computing queries | BDD calculation example

What node is redundant in this tree?

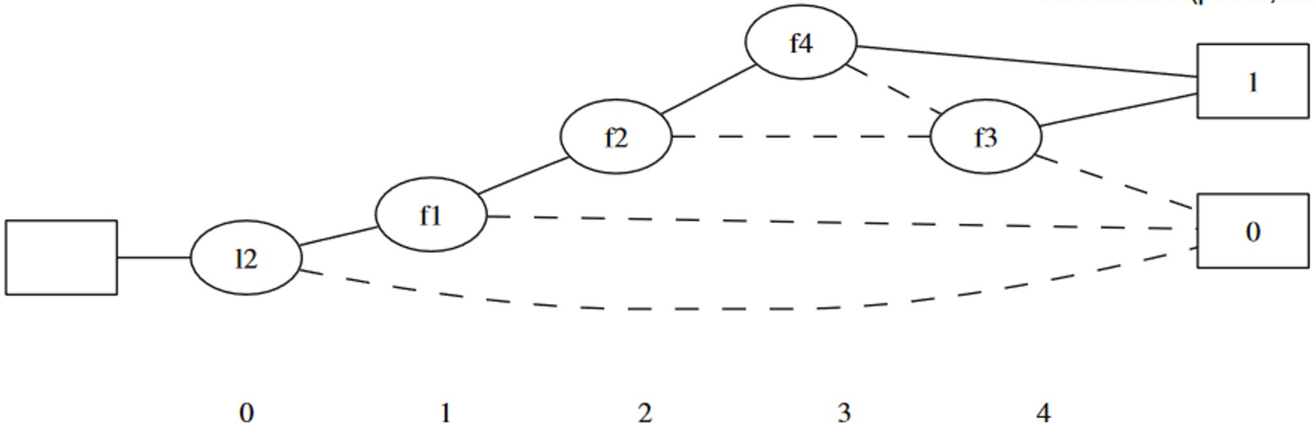
1.0: likes(X,Y):- friendof(X,Y).
0.8: likes(X,Y):- friendof(X,Z), likes(Z,Y).
0.5: friendof(john,mary).
0.5: friendof(mary,pedro).
0.5: friendof(mary,tom).
0.5: friendof(pedro,tom).



$$P((l_1 \wedge l_2 \wedge f_1 \wedge f_2 \wedge f_4) \vee (l_1 \wedge l_2 \wedge f_1 \wedge f_3)).$$

Computing queries | BDD calculation example

1.0: likes(X,Y):- friendof(X,Y).
0.8: likes(X,Y):- friendof(X,Z), likes(Z,Y).
0.5: friendof(john,mary).
0.5: friendof(mary,pedro).
0.5: friendof(mary,tom).
0.5: friendof(pedro,tom).



$$P((l_2 \wedge f_1 \wedge f_2 \wedge f_4) \vee (l_2 \wedge f_1 \wedge f_3))$$

Computing queries

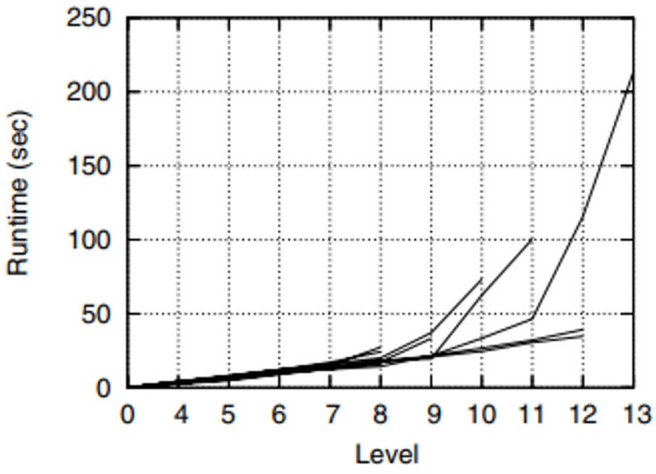
Two steps:

1. Build monotone DNF formula representing all solutions
2. Compute the probability of this DNF formula
   - Using Binary Decision Diagram (BDD) representation
     - Start from full binary tree, merging isomorphic subgraphs and deleting redundant nodes
   - Heuristically determine variable order in SOTA BDD algorithms
   - Reusable BDD for different queries

Approximating the success probability

- Why approximate?
- Iterative deepening to compute SLD-tree
- Use incomplete SLD-tree to derive upper and lower bound
  - Lower bound encodes successful proofs found so far
  - Upper bound encodes all proofs all proofs found so far
  - Keep growing tree until upper and lower bound are sufficiently close

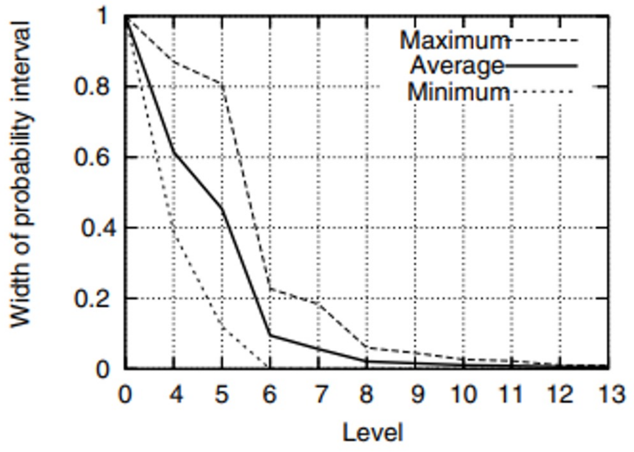
Results



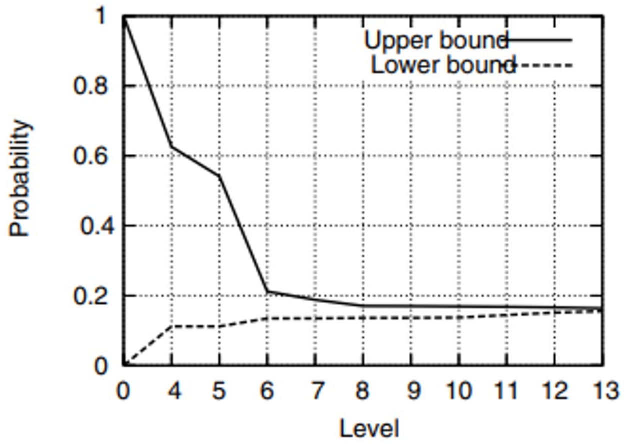
Running times for 10 test graphs with 1400 edges.

- Good runtime in terms of level depthness
- Can deal with many conjuncts, up to 100k.
- Probability is converging to the true one after the 6th depth level
- Bounds are converging to ~0.2 after the 6th depth level.

Results



Convergence of the probability interval for 10 test graphs with 1400 edges.



Convergence of bounds for one graph with 1800 edges, as a function of the search level.

TU Delft

Questions

- What is the addition of ProbLog to Prolog?

Questions

- What is the addition of ProbLog to Prolog?
- What other probabilistic programming languages also have inherently included upper and lower bounds?

Thank you for your attention!

Max Le Blansch, Bogdan Simion