

Generating Random Logic Programs Using Constraint Programming

Paulius Dilkas¹ Vaishak Belle^{1,2}

¹University of Edinburgh, Edinburgh, UK

²Alan Turing Institute, London, UK

CP 2020



THE UNIVERSITY OF EDINBURGH

informatics



EDINBURGH CENTRE FOR

ROBOTICS



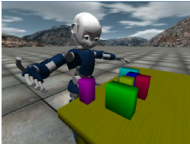
Engineering and
Physical Sciences
Research Council

Probabilistic Logic Programs (PROBLOG)

“Smokers” (Domingos et al. 2008; Fierens et al. 2015)

```
0.2::stress(P):-person(P).
0.3::influences(P1,P2):-friend(P1,P2).
0.1::cancer_spont(P):-person(P).
0.3::cancer_smoke(P):-person(P).
    smokes(X):-stress(X).
    smokes(X):-smokes(Y),influences(Y,X).
    cancer(P):-cancer_spont(P).
    cancer(P):-smokes(P),cancer_smoke(P).
    person(michelle).
    person(timothy).
    friend(timothy,michelle).
```

Applications



Moldovan et al. 2012

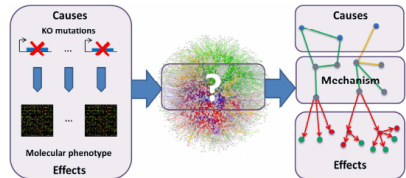
Q1: In a group of 10 people, 60 percent have brown eyes. Two people are to be selected at random from the group. What is the probability that neither person selected will have brown eyes?

Q2: Mike has a bag of marbles with 4 white, 8 blue, and 6 red marbles. He pulls out one marble from the bag and it is red. What is the probability that the second marble he pulls out of the bag is white?

Dries et al. 2017

```
is_malignant(Case):-
    biopsyProcedure(Case,usCore),
    changes_Sizeinc(Case,missing),
    feature_shape(Case).
is_malignant(Case):-
    assoFinding(Case,asymmetry),
    breastDensity(Case,scatteredFDensities),
    vacuumAssisted(Case,yes).
is_malignant(Case):-
    needleGauge(Case,9),
    offset(Case,14),
    vacuumAssisted(Case,yes).
```

Côte-Real, Dutra, and Rocha 2017



De Maeyer et al. 2013

Inference Algorithms and Knowledge Compilation Maps

- NNF negation normal form
- d-DNNF deterministic decomposable negation normal form
- BDD binary decision diagrams
- SDD sentential decision diagrams
- k*-Best only use the *k* most probable proofs

Example Diagrams for $C \wedge (A \vee \neg B)$

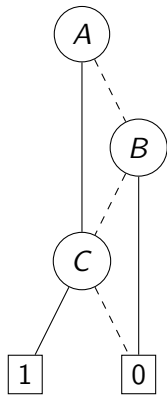


Figure: BDD

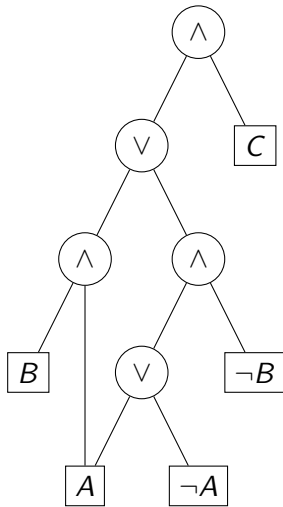


Figure: d-DNNF

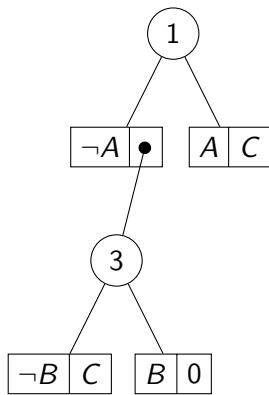


Figure: SDD

How Many Programs Are Used to Test Algorithms?

How Many Programs Are Used to Test Algorithms?

Anytime Inference in Probabilistic Logic Programs with T_p -Compilation

Jonas Vlasselaer, Guy Van den Broeck, Angelika Kimmig, Wannes Meert, Luc De Raedt

Department of Computer Science

KU Leuven, Belgium

firstname.lastname@cs.kuleuven.be

How Many Programs Are Used to Test Algorithms?

Anytime Inference in Probabilistic Logic Programs with T_p -Compilation

Jonas Vlasselaer, Guy Van den Broeck, Angelika Kimmig, Wannes Meert, Luc De Raedt
Department of Computer Science
KU Leuven, Belgium
firstname.lastname@cs.kuleuven.be

Inference and learning in probabilistic logic programs using weighted Boolean formulas

DAAN FIERENS, GUY VAN DEN BROECK, JORIS RENKENS,
DIMITAR SHTERIONOV, BERND GUTMANN, INGO THON,
GERDA JANSSENS and LUC DE RAEDT
Department of Computer Science, KU Leuven, Celestijnenlaan 200A, 3001 Heverlee, Belgium
(e-mail: `FirstName.LastName@cs.kuleuven.be`)

How Many Programs Are Used to Test Algorithms?

Anytime Inference in Probabilistic Logic Programs with T_p -Compilation

Jonas Vlasselaer, Guy Van den Broeck, Angelika Kimmig, Wannes Meert, Luc De Raedt
Department of Computer Science
KU Leuven, Belgium
firstname.lastname@cs.kuleuven.be

Inference and learning in probabilistic logic programs using weighted Boolean formulas

DAAN FIERENS, GUY VAN DEN BROECK, JORIS RENKENS,
DIMITAR SHTERIONOV, BERND GUTMANN, INGO THON,
GERDA JANSSENS and LUC DE RAEDT
Department of Computer Science, KU Leuven, Celestijnenlaan 200A, 3001 Heverlee, Belgium
(e-mail: FirstName.LastName@cs.kuleuven.be)

k-Optimal: a novel approximate inference algorithm for ProbLog

Joris Renkens · Guy Van den Broeck · Siegfried Nijssen

How Many Programs Are Used to Test Algorithms?

Anytime Inference in Probabilistic Logic Programs with T_p -Compilation

Jonas Vlasselaer, Guy Van den Broeck, Angelika Kimmig, Wannes Meert, Luc De Raedt
Department of Computer Science
KU Leuven, Belgium
firstname.lastname@cs.kuleuven.be

On the Efficient Execution of ProbLog Programs

Angelika Kimmig¹, Vitor Santos Costa², Ricardo Rocha², Bart Demeo¹, and
Luc De Raedt¹

Inference and learning in probabilistic logic programs using weighted Boolean formulas

DAAN FIERENS, GUY VAN DEN BROECK, JORIS RENKENS,
DIMITAR SHTERIONOV, BENNO GUTMANN, INGO THON,
GERDA JANSSENS and LUC DE RAEDT

Department of Computer Science, KU Leuven, Celestijnenlaan 200A, 3001 Heverlee, Belgium
(e-mail: Firstname.Lastname@cs.kuleuven.be)

k-Optimal: a novel approximate inference algorithm for ProbLog

Joris Renkens · Guy Van den Broeck · Siegfried Nijssen

How Many Programs Are Used to Test Algorithms?

Anytime Inference in Probabilistic Logic Programs with T_p -Compilation

Jonas Vlasselaer, Guy Van den Broeck, Angelika Kimmig, Wannes Meert, Luc De Raedt
Department of Computer Science
KU Leuven, Belgium
firstname.lastname@cs.kuleuven.be

Inference and learning in probabilistic logic programs using weighted Boolean formulas

DAAN FIERENS, GUY VAN DEN BROECK, JORIS RENKENS,
DIMITAR SHTERIONOV, BERND GUTMANN, INGO THON,
GERDA JANSSENS and LUC DE RAEDT

Department of Computer Science, KU Leuven, Celestijnenlaan 200A, 3001 Heverlee, Belgium
(e-mail: FirstName.LastName@cs.kuleuven.be)

***k*-Optimal: a novel approximate inference algorithm for ProbLog**

Joris Renkens · Guy Van den Broeck · Siegfried Nijssen

On the Efficient Execution of ProbLog Programs

Angelika Kimmig¹, Vítor Santos Costa², Ricardo Rocha², Bart Demoen¹, and
Luc De Raedt¹

On the Implementation of the Probabilistic Logic Programming Language ProbLog

Angelika Kimmig, Bart Demoen and Luc De Raedt
Department Computerwetenschappen, K.U. Leuven
Celestijnenlaan 200A - bus 2402, B-3001 Heverlee, Belgium
(e-mail: {Angelika.Kimmig,Bart.Demoen,Luc.DeRaedt}@cs.kuleuven.be)

Vítor Santos Costa and Ricardo Rocha
CRACS & INESC-Porto LA, Faculty of Sciences, University of Porto
R. do Campo Alegre 1091/1055, 4169-007 Porto, Portugal
(e-mail: {vsc,ricroc}@dcc.fc.up.pt)

How Many Programs Are Used to Test Algorithms?

Anytime Inference in Probabilistic Logic Programs with T_p -Compilation

Jonas Vlasselaer, Guy Van den Broeck, Angelika Kimmig, Wannes Meert, Luc De Raedt
Department of Computer Science
KU Leuven, Belgium
firstname.lastname@cs.kuleuven.be

Inference and learning in probabilistic logic programs using weighted Boolean formulas

DAAN FIERENS, GUY VAN DEN BROECK, JORIS RENKENS,
DIMITAR SHTERIONOV, BERND GUTMANN, INGO THON,
GERDA JANSSENS and LUC DE RAEDT

Department of Computer Science, KU Leuven, Celestijnenlaan 200A, 3001 Heverlee, Belgium
(e-mail: FirstName.LastName@cs.kuleuven.be)

***k*-Optimal: a novel approximate inference algorithm for ProbLog**

Joris Renkens · Guy Van den Broeck · Siegfried Nijssen

On the Efficient Execution of ProbLog Programs

Angelika Kimmig¹, Vítor Santos Costa², Ricardo Rocha², Bart Demoen¹, and
Luc De Raedt¹

On the Implementation of the Probabilistic Logic Programming Language ProbLog

Angelika Kimmig, Bart Demoen and Luc De Raedt
Department Computerwetenschappen, K.U. Leuven
Celestijnenlaan 200A - bus 2402, B-3001 Heverlee, Belgium
(e-mail: {Angelika.Kimmig,Bart.Demoen,Luc.DeRaedt}@cs.kuleuven.be)

Vítor Santos Costa and Ricardo Rocha
CRACS & INESC-Porto LA, Faculty of Sciences, University of Porto
R. do Campo Alegre 1091/1055, 4169-007 Porto, Portugal
(e-mail: {vsc,ricroc}@dcc.fc.up.pt)

ProbLog Technology for Inference in a Probabilistic First Order Logic

Maurice Bruynooghe and Theodoros Mantadelos and Angelika Kimmig and Bernd Gutmann
and Joost Vennekens and Gerda Janssens and Luc De Raedt¹

What Characterizes a (Probabilistic) Logic Program?

```
0.2::stress(P):-person(P).
0.3::influences(P1,P2):-friend(P1,P2).
0.1::cancer_spont(P):-person(P).
0.3::cancer_smoke(P):-person(P).
    smokes(X):-stress(X).
    smokes(X):-smokes(Y),influences(Y,X).
    cancer(P):-cancer_spont(P).
    cancer(P):-smokes(P),cancer_smoke(P).
    person(michelle).
    person(timothy).
    friend(timothy,michelle).
```

What Characterizes a (Probabilistic) Logic Program?

0.2::`stress`(P):-`person`(P).

0.3::`influences`(P₁, P₂):-`friend`(P₁, P₂).

0.1::`cancer_spont`(P):-`person`(P).

0.3::`cancer_smoke`(P):-`person`(P).

`smokes`(X):-`stress`(X).

`smokes`(X):-`smokes`(Y),`influences`(Y,X).

`cancer`(P):-`cancer_spont`(P).

`cancer`(P):-`smokes`(P),`cancer_smoke`(P).

`person`(*michelle*).

`person`(*timothy*).

`friend`(*timothy*, *michelle*).

- predicates, arities

What Characterizes a (Probabilistic) Logic Program?

```
0.2::stress(P):-person(P).
0.3::influences(P1,P2):-friend(P1,P2).
0.1::cancer_spont(P):-person(P).
0.3::cancer_smoke(P):-person(P).
    smokes(X):-stress(X).
    smokes(X):-smokes(Y),influences(Y,X).
    cancer(P):-cancer_spont(P).
    cancer(P):-smokes(P),cancer_smoke(P).
    person(michelle).
    person(timothy).
    friend(timothy,michelle).
```

- predicates, arities
- variables

What Characterizes a (Probabilistic) Logic Program?

```
0.2::stress(P):-person(P).
0.3::influences(P1,P2):-friend(P1,P2).
0.1::cancer_spont(P):-person(P).
0.3::cancer_smoke(P):-person(P).
    smokes(X):-stress(X).
    smokes(X):-smokes(Y),influences(Y,X).
    cancer(P):-cancer_spont(P).
    cancer(P):-smokes(P),cancer_smoke(P).
    person(michelle).
    person(timothy).
    friend(timothy,michelle).
```

- predicates, arities
- variables
- constants

What Characterizes a (Probabilistic) Logic Program?

0.2::stress(P):-person(P).

0.3::influences(P₁,P₂):-friend(P₁,P₂).

0.1::cancer_spont(P):-person(P).

0.3::cancer_smoke(P):-person(P).

smokes(X):-stress(X).

smokes(X):-smokes(Y),influences(Y,X).

cancer(P):-cancer_spont(P).

cancer(P):-smokes(P),cancer_smoke(P).

person(*michelle*).

person(*timothy*).

friend(*timothy*,*michelle*).

- predicates, arities
- variables
- constants
- probabilities

What Characterizes a (Probabilistic) Logic Program?

0.2::stress(P):-person(P).
0.3::influences(P₁,P₂):-friend(P₁,P₂).
0.1::cancer_spont(P):-person(P).
0.3::cancer_smoke(P):-person(P).
 smokes(X):-stress(X).
 smokes(X):-smokes(Y),influences(Y,X).
 cancer(P):-cancer_spont(P).
 cancer(P):-smokes(P),cancer_smoke(P).
 person(*michelle*).
 person(*timothy*).
 friend(*timothy*,*michelle*).

- predicates, arities
- variables
- constants
- probabilities
- length

What Characterizes a (Probabilistic) Logic Program?

0.2::stress(P):-person(P).
0.3::influences(P₁,P₂):-friend(P₁,P₂).
0.1::cancer_spont(P):-person(P).
0.3::cancer_smoke(P):-person(P).
 smokes(X):-stress(X).
 smokes(X):-smokes(Y),influences(Y,X).
 cancer(P):-cancer_spont(P).
 cancer(P):-smokes(P),cancer_smoke(P).
 person(*michelle*).
 person(*timothy*).
 friend(*timothy*,*michelle*).

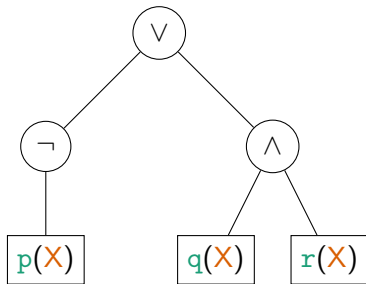
- predicates, arities
- variables
- constants
- probabilities
- length
- complexity

Clauses As Trees

$$\neg p(x) \vee (q(x) \wedge r(x))$$

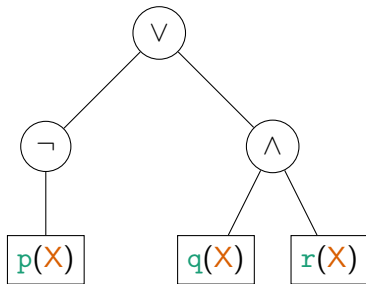
Clauses As Trees

$$\neg p(X) \vee (q(X) \wedge r(X))$$



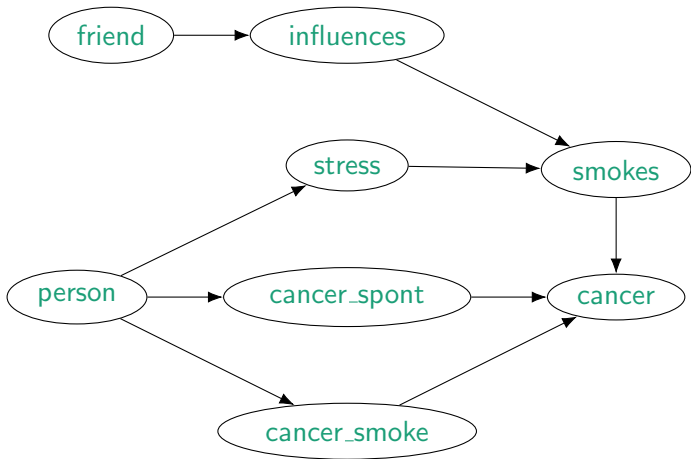
Clauses As Trees

$$\neg p(X) \vee (q(X) \wedge r(X))$$

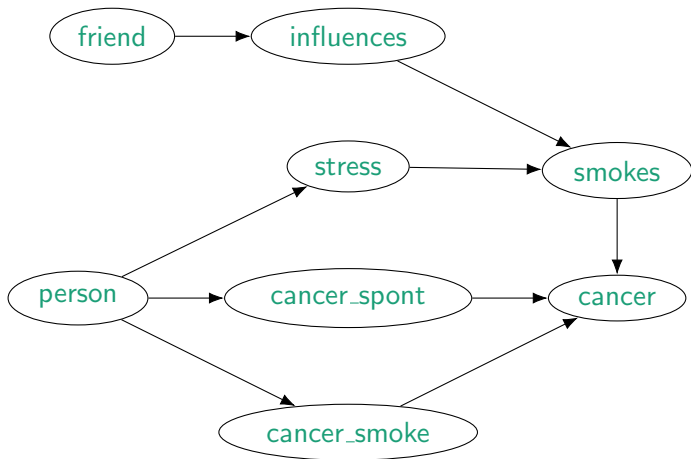


0	0	0	1	2	2
\vee	\neg	\wedge	$p(X)$	$q(X)$	$r(X)$

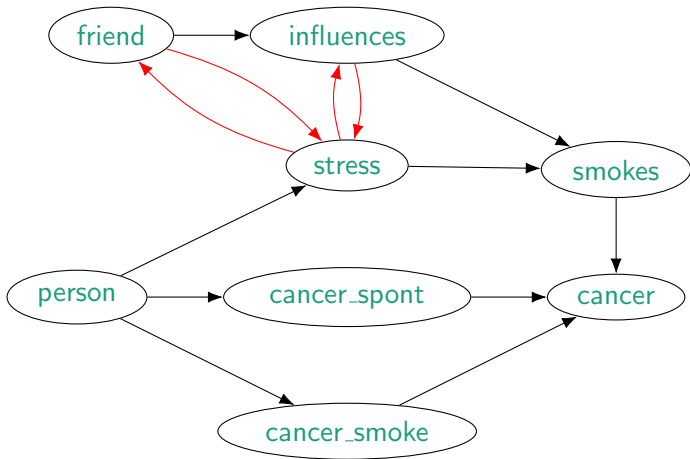
Predicate Dependency Graph



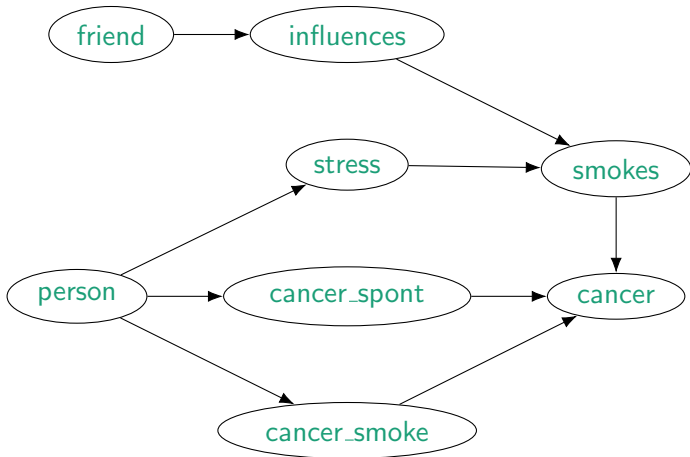
Independence: influences \perp stress



Independence: influences \perp stress

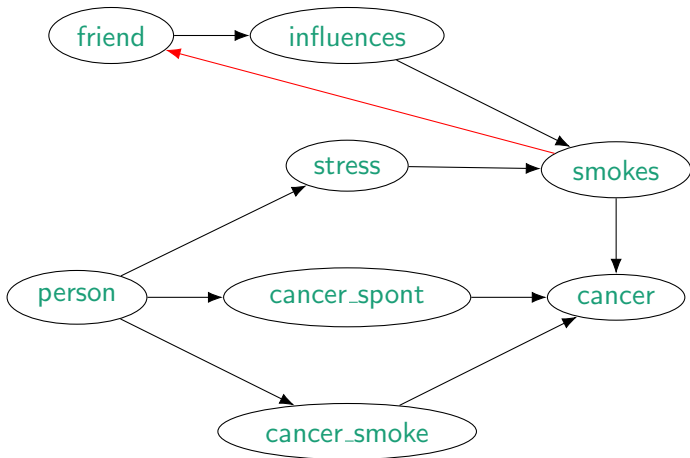


Stratification and Negative Cycles



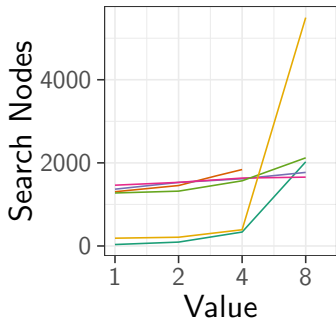
```
0.1::friend(X,Y):- \+ smokes(Y).
```

Stratification and Negative Cycles



```
0.1::friend(X,Y):- \+ smokes(Y).
```

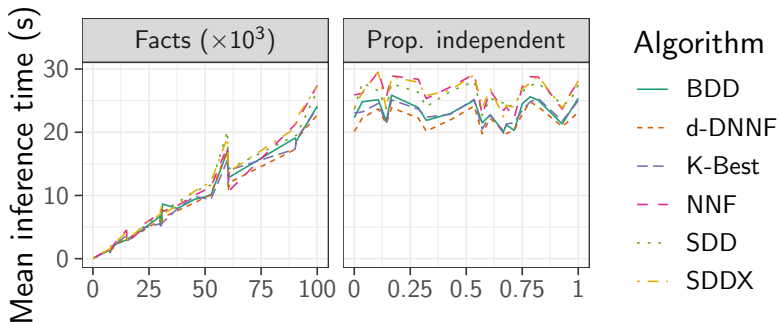
Scalability



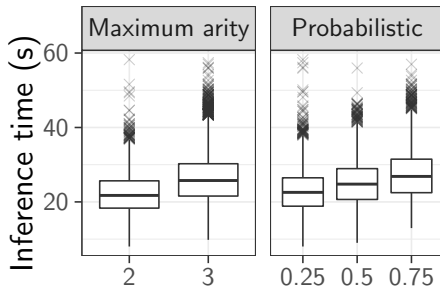
Variable

- The number of predicates
- Maximum arity
- The number of variables
- The number of constants
- The number of additional clauses
- The maximum number of nodes

Properties of Programs vs. Inference Algorithms



Properties of Programs vs. Inference Algorithms



Summary

- ...
- ...
- ...

The implementation of the model is available at

<https://github.com/dilkas/random-logic-programs>