

Generating Random Logic Programs Using Constraint Programming

Paulius Dilkas

AI/ML Seminar



THE UNIVERSITY OF EDINBURGH

informatics



EDINBURGH CENTRE FOR
ROBOTICS



Engineering and
Physical Sciences
Research Council

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, 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)

How Many Programs Are Used to Test Algorithms?

Anytime Inference in Probabilistic Logic Programs with T_p -Compilation

4
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

3
DAAN FIERENS, GUY VAN DEN BROECK, JORIS RENKENS,
DIMITAR SHTERIONOV, BENJAMIN 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

1
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, BENJAMIN 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, Vitor Santos Costa², Ricardo Rocha², Bart Demoen¹, and
Luc De Raedt¹

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, 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

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 1021/1055, 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 1021/1055, 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 Theofrastos Mantonakis and Angelika Kimmig and Bernd Gutmann
and Joost Vennekens and Gerda Janssens and Luc De Raedt¹

Outline

Probabilistic Logic Programming

The Constraint Model

Example Programs

Experimental Results

Conclusions

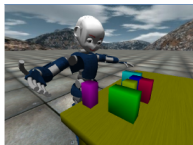
Probabilistic Logic Programs (PROBLOG)

“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

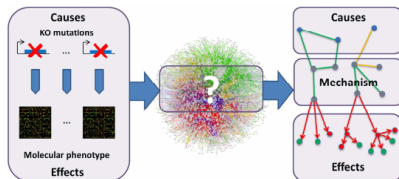
Q1: In a group of 10 people, 60 percent have brown eyes. Two people are to be selected at random from the group. What is the probability that neither person selected will have brown eyes?

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).
```

Côrte-Real, Dutra, and Rocha 2017



De Maeyer et al. 2013

Probabilistic Inference: Reasoning by Hand

Let $a \oplus b := a + b - ab$. Then

$$\begin{aligned} \Pr[\text{cancer}(\text{michelle})] &= \Pr[\text{cancer_spont}(\text{michelle})] \\ &\quad \oplus \Pr[\text{smokes}(\text{michelle})] \\ &\quad \times \Pr[\text{cancer_smoke}(\text{michelle})] \end{aligned}$$

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

Probabilistic Inference: Reasoning by Hand

Let $a \oplus b := a + b - ab$. Then

$$\Pr[\text{cancer}(\text{michelle})] = 0.1 \oplus 0.3 \times \Pr[\text{smokes}(\text{michelle})]$$

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

Probabilistic Inference: Reasoning by Hand

Let $a \oplus b := a + b - ab$. Then

$$\Pr[\text{cancer}(\text{michelle})] = 0.1 \oplus 0.3 \times \Pr[\text{smokes}(\text{michelle})]$$

$$\begin{aligned} \Pr[\text{smokes}(\text{michelle})] &= \Pr[\text{stress}(\text{michelle})] \\ &\quad \oplus \Pr[\text{smokes}(\text{timothy})] \\ &\quad \times \Pr[\text{influences}(\text{timothy}, \text{michelle})] \end{aligned}$$

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

Probabilistic Inference: Reasoning by Hand

Let $a \oplus b := a + b - ab$. Then

$$\Pr[\text{cancer}(\text{michelle})] = 0.1 \oplus 0.3 \times \Pr[\text{smokes}(\text{michelle})]$$

$$\Pr[\text{smokes}(\text{michelle})] = 0.2 \oplus 0.3 \times \Pr[\text{smokes}(\text{timothy})]$$

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

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

Probabilistic Inference: Reasoning by Hand

Let $a \oplus b := a + b - ab$. Then

$$\Pr[\text{cancer}(\text{michelle})] = 0.1 \oplus 0.3 \times \Pr[\text{smokes}(\text{michelle})]$$

$$\Pr[\text{smokes}(\text{michelle})] = 0.2 \oplus 0.3 \times \Pr[\text{smokes}(\text{timothy})]$$

$$\Pr[\text{smokes}(\text{timothy})] = \Pr[\text{stress}(\text{timothy})] = 0.2$$

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


Probabilistic Inference: Probabilities of Worlds

```

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).
    
```

Probabilistic Inference: Probabilities of Worlds

`cancer`(*michelle*) = \top

$\Pr(\text{world}) = 0.2 \times (1 - 0.3) \times 0.1 \times 0.3$

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

~~0.3~~ :: `influences`(*P*₁, *P*₂) :- ~~`friend`(*P*₁, *P*₂).~~

~~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*).

Probabilistic Inference: Probabilities of Worlds

`cancer`(*michelle*) = \top

$\Pr(\text{world}) = 0.2 \times 0.3 \times 0.1 \times (1 - 0.3)$

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

~~0.3~~ :: `influences`(*P*₁, *P*₂) :- `friend`(*P*₁, *P*₂).

~~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*).

Probabilistic Inference: Probabilities of Worlds

`cancer`(*michelle*) = \top

$\Pr(\text{world}) = (1 - 0.2) \times (1 - 0.3) \times 0.1 \times 0.3$

~~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*).

Probabilistic Inference: Probabilities of Worlds

`cancer(michelle)` = \perp

$$\Pr(world) = (1 - 0.2) \times (1 - 0.3) \times (1 - 0.1) \times (1 - 0.3)$$

~~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).`

Inference Algorithms and Knowledge Compilation Maps

NNF negation normal form

BDD binary decision diagrams

SDD sentential decision diagrams

***k*-Best** only use the *k* most probable proofs

d-DNNF deterministic decomposable negation normal form

Inference Algorithms and Knowledge Compilation Maps

NNF negation normal form

BDD binary decision diagrams

SDD sentential decision diagrams

k -Best only use the k most probable proofs

d-DNNF deterministic decomposable negation normal form

- for every pair $\alpha \vee \beta$, we have $\alpha \wedge \beta = \perp$
- for every pair $\alpha \wedge \beta$, no atoms are shared between α and β

Inference Algorithms and Knowledge Compilation Maps

NNF negation normal form

BDD binary decision diagrams

SDD sentential decision diagrams

k -Best only use the k most probable proofs

d-DNNF deterministic decomposable negation normal form

- for every pair $\alpha \vee \beta$, we have $\alpha \wedge \beta = \perp$
- for every pair $\alpha \wedge \beta$, no atoms are shared between α and β
- examples:

$$(A \vee C) \wedge (A \vee \neg B)$$

Inference Algorithms and Knowledge Compilation Maps

NNF negation normal form

BDD binary decision diagrams

SDD sentential decision diagrams

k -Best only use the k most probable proofs

d-DNNF deterministic decomposable negation normal form

- for every pair $\alpha \vee \beta$, we have $\alpha \wedge \beta = \perp$
- for every pair $\alpha \wedge \beta$, no atoms are shared between α and β
- examples:
 $\times (A \vee C) \wedge (A \vee \neg B)$

Inference Algorithms and Knowledge Compilation Maps

NNF negation normal form

BDD binary decision diagrams

SDD sentential decision diagrams

k -Best only use the k most probable proofs

d-DNNF deterministic decomposable negation normal form

- for every pair $\alpha \vee \beta$, we have $\alpha \wedge \beta = \perp$
- for every pair $\alpha \wedge \beta$, no atoms are shared between α and β
- examples:

$$\text{xx } (A \vee C) \wedge (A \vee \neg B)$$

Inference Algorithms and Knowledge Compilation Maps

NNF negation normal form

BDD binary decision diagrams

SDD sentential decision diagrams

k -Best only use the k most probable proofs

d-DNNF deterministic decomposable negation normal form

- for every pair $\alpha \vee \beta$, we have $\alpha \wedge \beta = \perp$
- for every pair $\alpha \wedge \beta$, no atoms are shared between α and β
- examples:

$$\text{xx } (A \vee C) \wedge (A \vee \neg B) \\ C \wedge (A \vee \neg B)$$

Inference Algorithms and Knowledge Compilation Maps

NNF negation normal form

BDD binary decision diagrams

SDD sentential decision diagrams

k -Best only use the k most probable proofs

d-DNNF deterministic decomposable negation normal form

- for every pair $\alpha \vee \beta$, we have $\alpha \wedge \beta = \perp$
- for every pair $\alpha \wedge \beta$, no atoms are shared between α and β
- examples:

$$\text{XX} \quad (A \vee C) \wedge (A \vee \neg B)$$

$$\text{X} \quad C \wedge (A \vee \neg B)$$

Inference Algorithms and Knowledge Compilation Maps

NNF negation normal form

BDD binary decision diagrams

SDD sentential decision diagrams

k -Best only use the k most probable proofs

d-DNNF deterministic decomposable negation normal form

- for every pair $\alpha \vee \beta$, we have $\alpha \wedge \beta = \perp$
- for every pair $\alpha \wedge \beta$, no atoms are shared between α and β
- examples:

~~XX~~ $(A \vee C) \wedge (A \vee \neg B)$

X✓ $C \wedge (A \vee \neg B)$

Inference Algorithms and Knowledge Compilation Maps

NNF negation normal form

BDD binary decision diagrams

SDD sentential decision diagrams

k -Best only use the k most probable proofs

d-DNNF deterministic decomposable negation normal form

- for every pair $\alpha \vee \beta$, we have $\alpha \wedge \beta = \perp$
- for every pair $\alpha \wedge \beta$, no atoms are shared between α and β
- examples:

$$\text{XX} \quad (A \vee C) \wedge (A \vee \neg B)$$

$$\text{X}\checkmark \quad C \wedge (A \vee \neg B)$$

$$B \wedge C \wedge [(B \wedge A) \vee \neg B]$$

Inference Algorithms and Knowledge Compilation Maps

NNF negation normal form

BDD binary decision diagrams

SDD sentential decision diagrams

k -Best only use the k most probable proofs

d-DNNF deterministic decomposable negation normal form

- for every pair $\alpha \vee \beta$, we have $\alpha \wedge \beta = \perp$
- for every pair $\alpha \wedge \beta$, no atoms are shared between α and β
- examples:

~~xx~~ $(A \vee C) \wedge (A \vee \neg B)$

x✓ $C \wedge (A \vee \neg B)$

✓ $B \wedge C \wedge [(B \wedge A) \vee \neg B]$

Inference Algorithms and Knowledge Compilation Maps

NNF negation normal form

BDD binary decision diagrams

SDD sentential decision diagrams

k -Best only use the k most probable proofs

d-DNNF deterministic decomposable negation normal form

- for every pair $\alpha \vee \beta$, we have $\alpha \wedge \beta = \perp$
- for every pair $\alpha \wedge \beta$, no atoms are shared between α and β
- examples:

~~XX~~ $(A \vee C) \wedge (A \vee \neg B)$

X✓ $C \wedge (A \vee \neg B)$

✓X $B \wedge C \wedge [(B \wedge A) \vee \neg B]$

Inference Algorithms and Knowledge Compilation Maps

NNF negation normal form

BDD binary decision diagrams

SDD sentential decision diagrams

k -Best only use the k most probable proofs

d-DNNF deterministic decomposable negation normal form

- for every pair $\alpha \vee \beta$, we have $\alpha \wedge \beta = \perp$
- for every pair $\alpha \wedge \beta$, no atoms are shared between α and β
- examples:

$\text{XX} (A \vee C) \wedge (A \vee \neg B)$

$\text{X}\checkmark C \wedge (A \vee \neg B)$

$\checkmark\text{X} B \wedge C \wedge [(B \wedge A) \vee \neg B]$
 $C \wedge [(B \wedge A) \vee \neg B]$

Inference Algorithms and Knowledge Compilation Maps

NNF negation normal form

BDD binary decision diagrams

SDD sentential decision diagrams

k -Best only use the k most probable proofs

d-DNNF deterministic decomposable negation normal form

- for every pair $\alpha \vee \beta$, we have $\alpha \wedge \beta = \perp$
- for every pair $\alpha \wedge \beta$, no atoms are shared between α and β
- examples:

$\times\times$ $(A \vee C) \wedge (A \vee \neg B)$

$\times\checkmark$ $C \wedge (A \vee \neg B)$

$\checkmark\times$ $B \wedge C \wedge [(B \wedge A) \vee \neg B]$

\checkmark $C \wedge [(B \wedge A) \vee \neg B]$

Inference Algorithms and Knowledge Compilation Maps

NNF negation normal form

BDD binary decision diagrams

SDD sentential decision diagrams

k -Best only use the k most probable proofs

d-DNNF deterministic decomposable negation normal form

- for every pair $\alpha \vee \beta$, we have $\alpha \wedge \beta = \perp$
- for every pair $\alpha \wedge \beta$, no atoms are shared between α and β
- examples:

$\times\times (A \vee C) \wedge (A \vee \neg B)$

$\times\checkmark C \wedge (A \vee \neg B)$

$\checkmark\times B \wedge C \wedge [(B \wedge A) \vee \neg B]$

$\checkmark\checkmark C \wedge [(B \wedge A) \vee \neg B]$

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

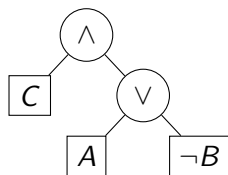


Figure: NNF

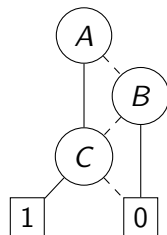


Figure: BDD

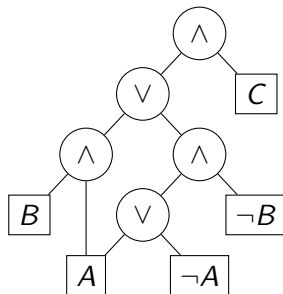


Figure: d-DNNF

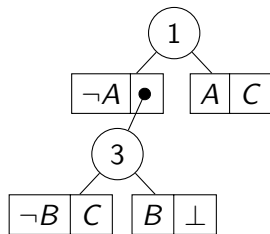


Figure: SDD

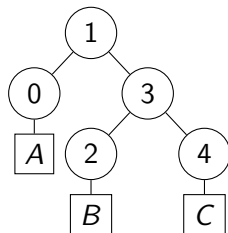


Figure: vtree

What Characterises 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 Characterises 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 Characterises 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 Characterises 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 Characterises 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 Characterises a (Probabilistic) Logic Program?

$0.2 :: \text{stress}(P) :- \text{person}(P).$
 $0.3 :: \text{influences}(P_1, P_2) :- \text{friend}(P_1, P_2).$
 $0.1 :: \text{cancer_spont}(P) :- \text{person}(P).$
 $0.3 :: \text{cancer_smoke}(P) :- \text{person}(P).$
 $\text{smokes}(X) :- \text{stress}(X).$
 $\text{smokes}(X) :- \text{smokes}(Y), \text{influences}(Y, X).$
 $\text{cancer}(P) :- \text{cancer_spont}(P).$
 $\text{cancer}(P) :- \text{smokes}(P), \text{cancer_smoke}(P).$
 $\text{person}(\text{michelle}).$
 $\text{person}(\text{timothy}).$
 $\text{friend}(\text{timothy}, \text{michelle}).$

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

What Characterises 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).
    
```

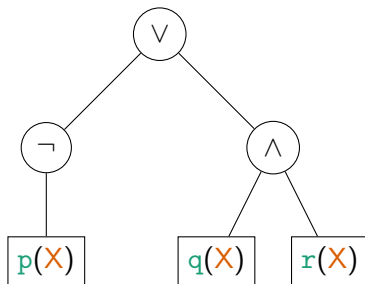
- predicates, arities
- variables
- constants
- probabilities
- length
- complexity

Formulas As Trees

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

Formulas As Trees

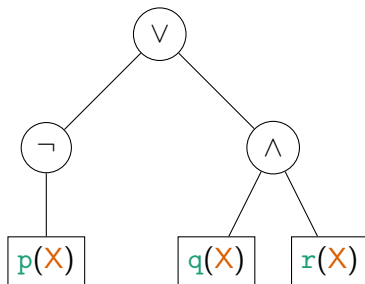
$$\neg p(X) \vee (q(X) \wedge r(X))$$



Formulas As Trees

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

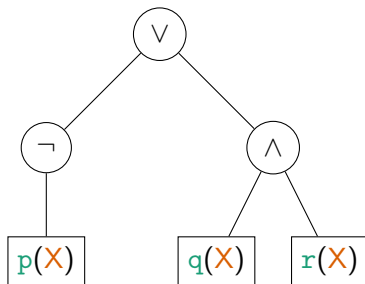
s:	0	0	0	1	2	2	6
v:	\vee	\neg	\wedge	$p(X)$	$q(X)$	$r(X)$	\top



Formulas As Trees

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

s:	0	0	0	1	2	2	6
v:	\vee	\neg	\wedge	$p(X)$	$q(X)$	$r(X)$	\top

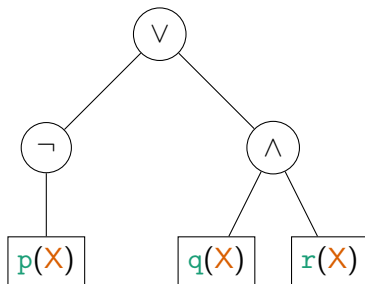


- **s** is a forest with $T = 2$ trees

Formulas As Trees

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

s:	0	0	0	1	2	2	6
v:	\vee	\neg	\wedge	$p(X)$	$q(X)$	$r(X)$	\top

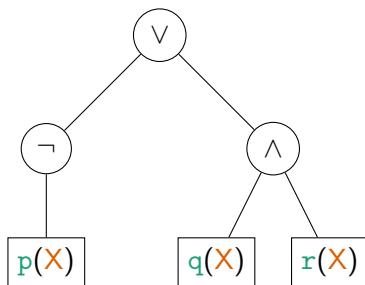


- **s** is a forest with $T = 2$ trees

Formulas As Trees

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

s :	0	0	0	1	2	2	6
v :	\vee	\neg	\wedge	$p(X)$	$q(X)$	$r(X)$	\top



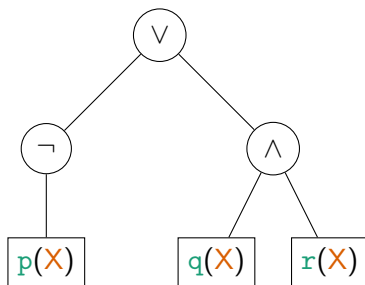
- s is a forest with $T = 2$ trees

- s is sorted
- $s_i \neq i \implies v_i \neq \top$

Formulas As Trees

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

$s:$	0	0	0	1	2	2	6
$v:$	\vee	\neg	\wedge	$p(X)$	$q(X)$	$r(X)$	\top

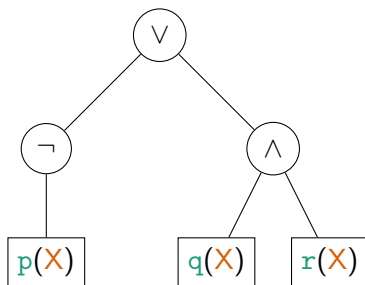


- s is a forest with $T = 2$ trees
- length $L = 7$
- number of nodes $N := L - T + 1 = 6$
- for $i = 1, \dots, L - 1$,
 - if $i < N$, then $s_i < i$
 - else $s_i = i$ and $v_i = \top$

- s is sorted
- $s_i \neq i \implies v_i \neq \top$

Formulas As Trees

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



- s is sorted
- $s_i \neq i \implies v_i \neq \top$

s :	0	0	0	1	2	2	6
v :	\vee	\neg	\wedge	$p(X)$	$q(X)$	$r(X)$	\top
c :	2	1	2	0	0	0	0

- s is a forest with $T = 2$ trees
- length $L = 7$
- number of nodes $N := L - T + 1 = 6$
- for $i = 1, \dots, L - 1$,
 - if $i < N$, then $s_i < i$
 - else $s_i = i$ and $v_i = \top$
- $c_i = 0 \iff v_i = \top$ or is a predicate
- $c_i = 1 \iff v_i = \neg$
- $c_i > 1 \iff v_i \in \{\wedge, \vee\}$

Variable Symmetry Breaking

The Problem

Let $\{W, X, Y\}$ be the set of variables. Then

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

is equivalent to

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

and to

`smokes(W) :- smokes(X), influences(X, W)`

Variable Symmetry Breaking

X	Y	Y	X
---	---	---	---

Occurrences
(channeling)

$W \mapsto \emptyset$

$X \mapsto \{0, 3\}$

$Y \mapsto \{1, 2\}$

Introductions

$1 + \min \text{occurrences}(v) \text{ or } 0$

$W \mapsto 0$

$X \mapsto 1$

$Y \mapsto 2$

sorted!

Variable Symmetry Breaking

Y	X	X	Y
---	---	---	---

Occurrences
(channeling)

$W \mapsto \emptyset$

$X \mapsto \{1, 2\}$

$Y \mapsto \{0, 3\}$

Introductions

$1 + \min \text{occurrences}(v) \text{ or } 0$

$W \mapsto 0$

$X \mapsto 2$

$Y \mapsto 1$

not sorted!

Variable Symmetry Breaking

W	X	X	W
---	---	---	---

Occurrences
(channeling)

W \mapsto {0, 3}

X \mapsto {1, 2}

Y \mapsto \emptyset

Introductions

1 + min occurrences(v) or 0

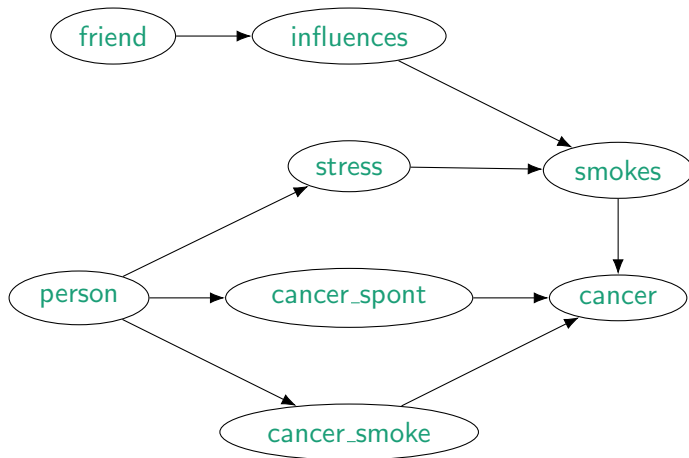
W \mapsto 1

X \mapsto 2

Y \mapsto 0

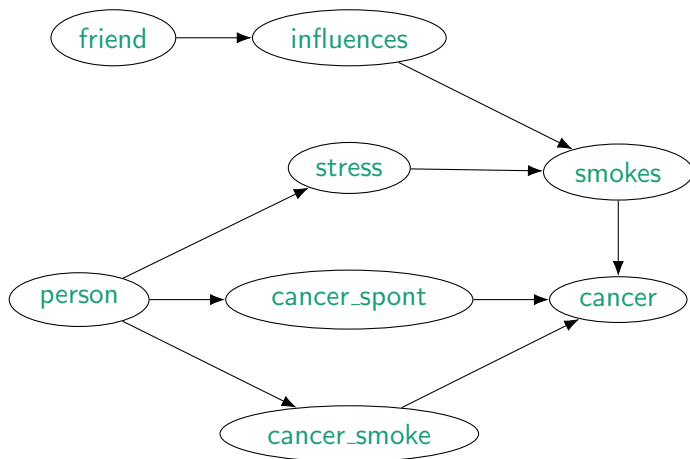
not sorted!

Predicate Dependency Graph



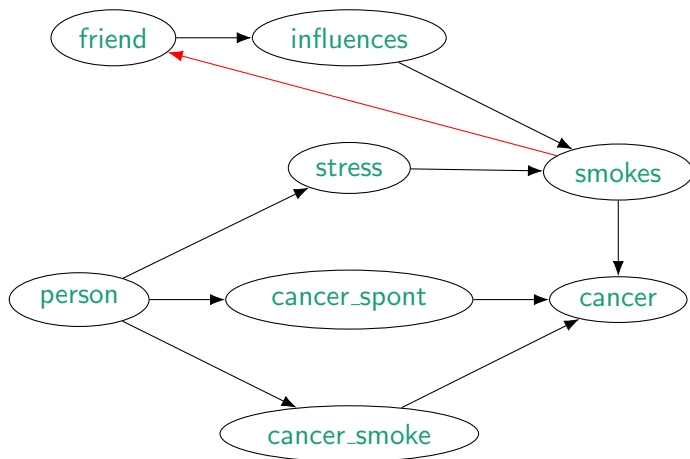
Stratification and Negative Cycles

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



Stratification and Negative Cycles

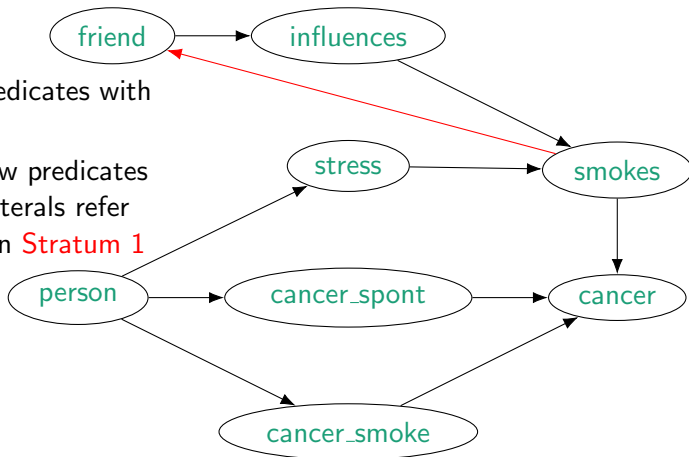
0.1::friend(X,Y):- \+ smokes(Y).



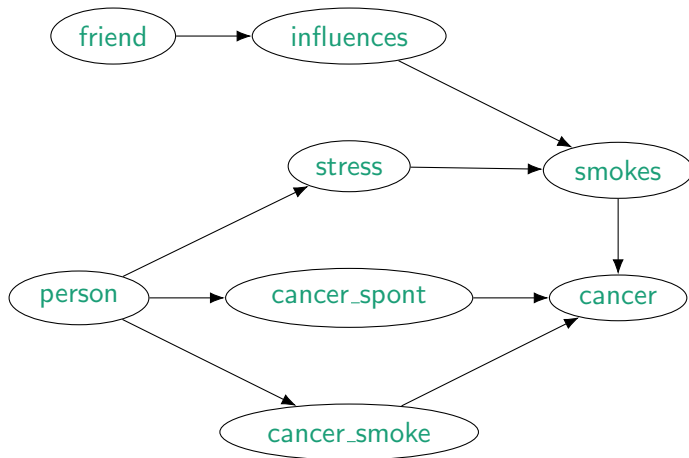
Stratification and Negative Cycles

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

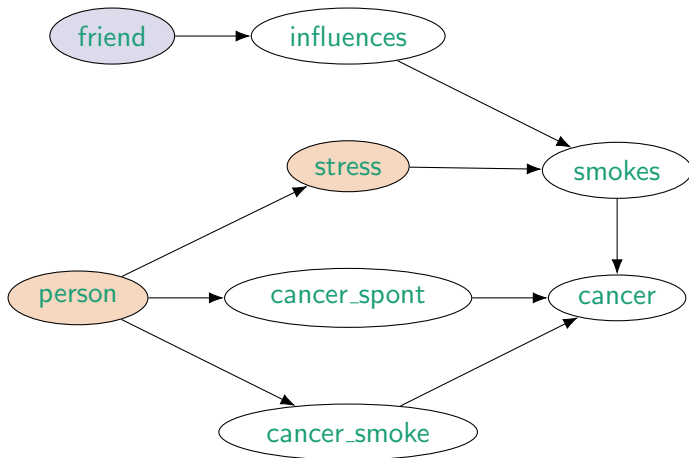
- **Stratum 1**: predicates with no negation
- **Stratum 2**: new predicates s.t. negative literals refer to predicates in **Stratum 1**



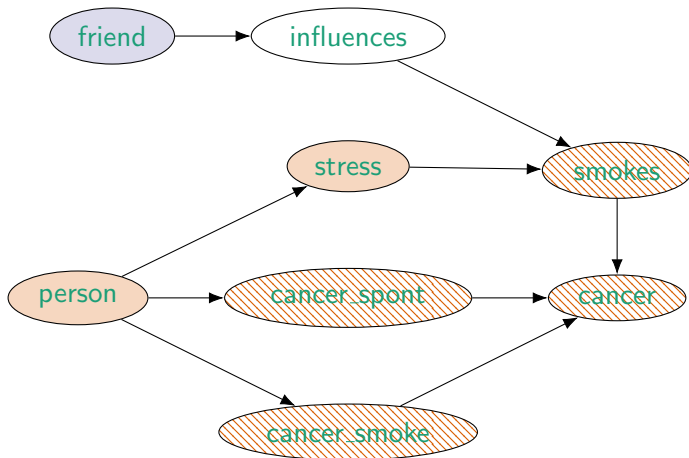
Predicate Independence: friend \perp stress



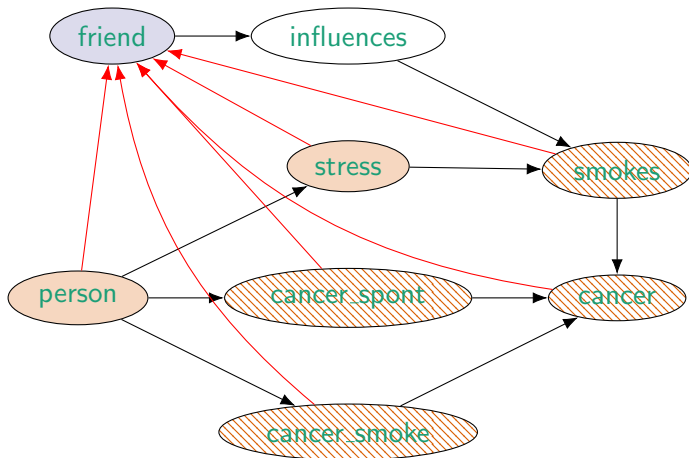
Predicate Independence: friend \perp stress



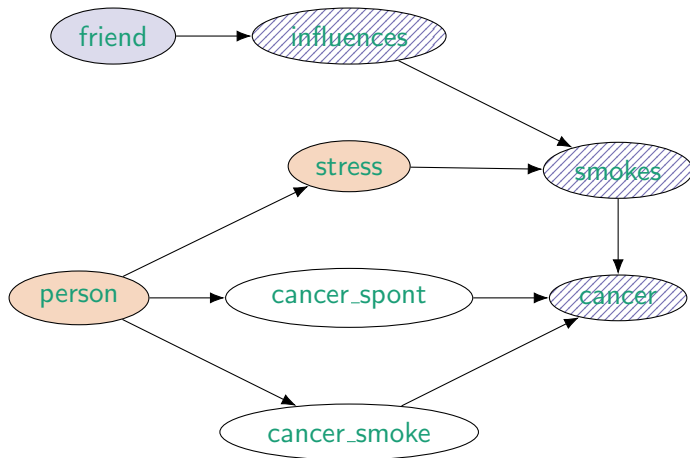
Predicate Independence: friend \perp stress



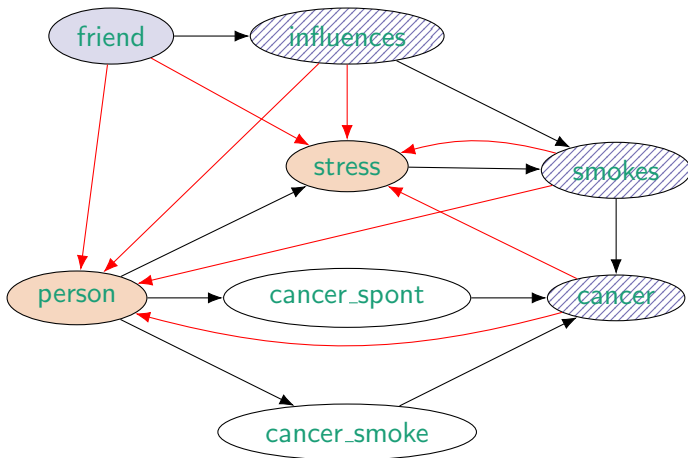
Predicate Independence: $\text{friend} \perp \text{stress}$



Predicate Independence: friend \perp stress



Predicate Independence: $\text{friend} \perp \text{stress}$



One-Liners

Setup

- predicate $p/1$
- variable X
- no constants
- 1 clause
- 4 nodes
- no negative cycles

(All) Programs

- $p(X).$
- $0.7 :: p(X) :- p(X).$
- $0.8 :: p(X) :- p(X); p(X).$
- $0.7 :: p(X) :- p(X), p(X).$
- $0.1 :: p(X) :- p(X); p(X); p(X).$
- $0.8 :: p(X) :- p(X), p(X), p(X).$

Symmetry Breaking in Action

Setup

- predicate $p/3$
- variables: X, Y, Z
- no constants
- 1 clause
- 1 node
- no cycles at all

(All) Programs

- $0.8 :: p(Z, Z, Z).$
- $p(Y, Y, Z).$
- $p(Y, Z, Y).$
- $p(Y, Z, Z).$
- $0.1 :: p(X, Y, Z).$

A Larger Example

Setup

- predicates: $p/1$, $q/2$, $r/3$
- variables: X , Y , Z
- constants: a , b , c
- 5 clauses
- 5 nodes
- no negative cycles

A Random Program

```

p(b) :- \+(q(a, b), q(X, Y), q(Z, X)).
0.4 :: q(X, X) :- \+ r(Y, Z, a).
q(X, a) :- r(Y, Y, Z).
q(X, a) :- r(Y, b, Z).
r(Y, b, Z).
    
```

Examples of Predicate Independence

A Few Random Programs

Setup

- predicates: $p/1$, $q/1$, $r/1$
- no variables
- constant a
- 3 clauses
- 3 nodes
- no negative cycles
- $p \perp\!\!\!\perp q$

- $0.5 :: p(a) :- \neg p(a); p(a).$

- $0.2 :: q(a) :- \neg q(a), q(a).$

- $0.4 :: r(a) :- \neg q(a).$

- $p(a) :- \neg p(a).$

- $0.5 :: q(a) :- \neg r(a); q(a).$

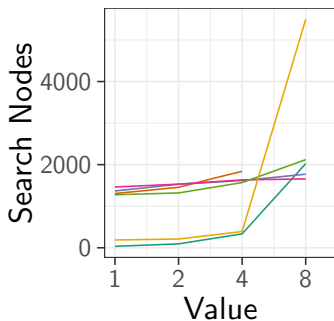
- $r(a) :- \neg r(a); r(a).$

- $p(a) :- \neg p(a); p(a).$

- $0.6 :: q(a) :- \neg q(a).$

- $0.7 :: r(a) :- \neg q(a).$

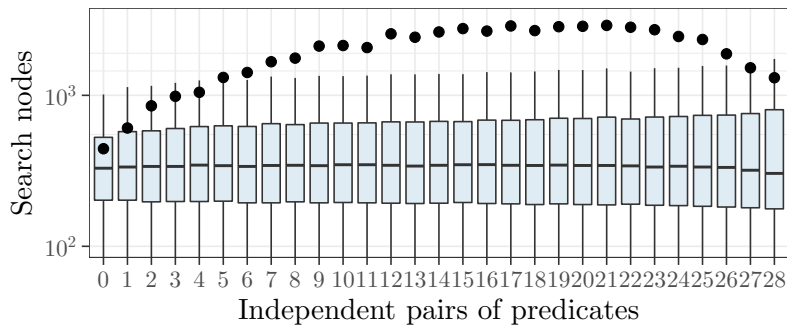
Scalability



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

How Predicate Independence Affects Search Complexity



What Programs Should We Generate?

- each program is divided into:
 - rules
 - e.g., `0.2 :: stress(P) :- person(P).`
 - facts
 - e.g., `friend(timothy, michele).`
- predicates, variables, nodes: 2, 4, 8
- maximum arity: 1, 2, 3
- all possible numbers of pairs of independent predicates
- 10 programs per configuration
 - fully restarting the constraint solver
- probabilities sampled from $\{0.1, 0.2, \dots, 0.9\}$
- query: random unlisted fact

Rules

$0.2 :: \text{stress}(P) :- \text{person}(P).$

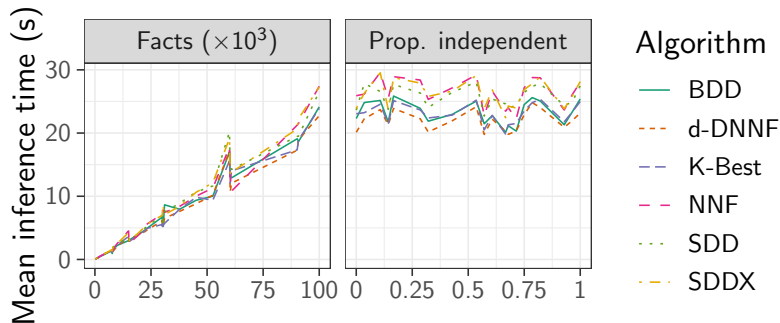
- no constants, no empty bodies
- one rule per predicate
- all rules are probabilistic

Facts

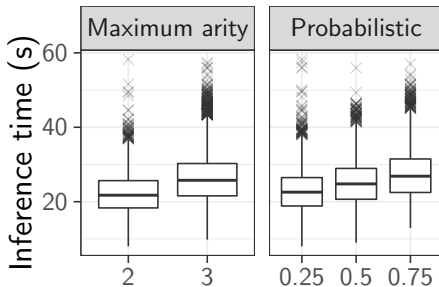
`friend`(*timothy*, *michelle*).

- proportion probabilistic: 25%, 50%, 75%
- constants: 100, 200, 400
- number of facts: 10^3 , 10^4 , 10^5
 - but only up to 75% of all possible facts

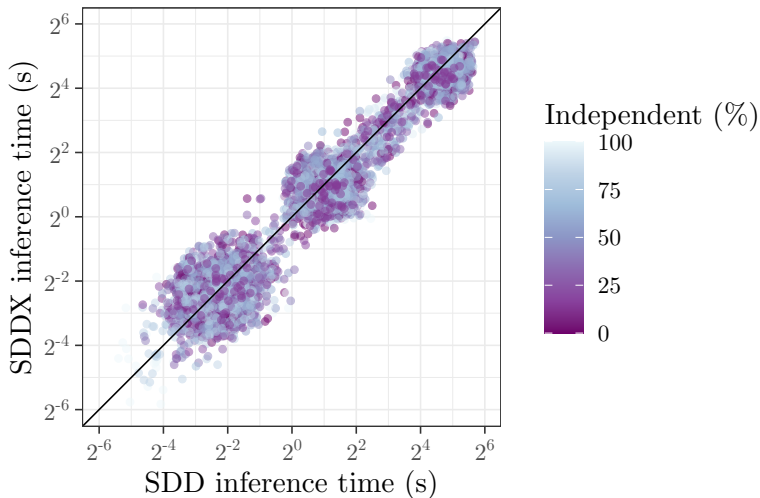
Properties of Programs vs. Inference Algorithms



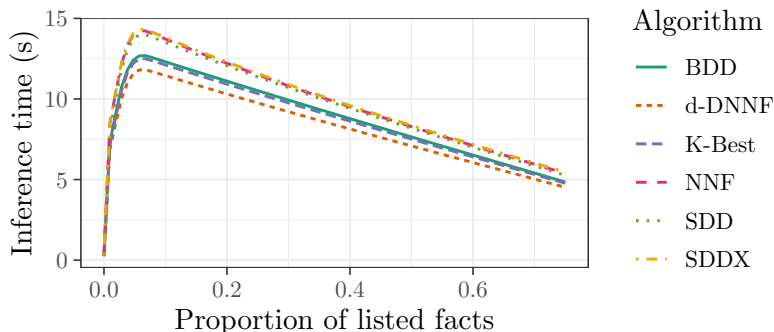
Properties of Programs vs. Inference Algorithms



How Encodings Compare Across Instances



The Ratio of Listed Facts to Possible Facts



Conclusions and Future Work

- The model can generate (approximately) realistic instances of reasonable size
- The main performance bottleneck can be addressed by generating programs with a simpler structure
- Open questions and future work
 - Can the model be used to ensure uniform sampling?
 - What is the reason behind all algorithms behaving similarly?
 - Why does independence have no effect on inference time?
 - Can random program generation be useful in learning?

The implementation of the model is available at

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