Generating Random Logic Programs Using Constraint Programming

Paulius Dilkas

AIAI 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 $T_{\mathcal{P}}$-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

4

How Many Programs Are Used to Test Algorithms?

Anytime Inference in Probabilistic Logic Programs with $T_{\mathcal{P}}$-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

4

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)

3

How Many Programs Are Used to Test Algorithms?

Anytime Inference in Probabilistic Logic Programs with $T_\mathcal{P}$-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

4

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)

3

k-Optimal: a novel approximate inference algorithm for ProbLog

Joris Renkens · Guy Van den Broeck · Siegfried Nijssen

1

How Many Programs Are Used to Test Algorithms?

Anytime Inference in Probabilistic Logic Programs with $T_\mathcal{P}$-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

4

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)

3

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

Joris Renkens · Guy Van den Broeck · Siegfried Nijssen

1

On the Efficient Execution of ProbLog Programs

Angelika Kimmig[1], Vítor Santos Costa[2], Ricardo Rocha[2], Bart Demoen[1], and Luc De Raedt[1]

1

How Many Programs Are Used to Test Algorithms?

Anytime Inference in Probabilistic Logic Programs with $T_{\mathcal{P}}$-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

4

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)

3

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

Joris Renkens · Guy Van den Broeck · Siegfried Nijssen

1

On the Efficient Execution of ProbLog Programs

Angelika Kimmig[1], Vítor Santos Costa[2], Ricardo Rocha[2], Bart Demoen[1], and Luc De Raedt[1]

1

On the Implementation of the Probabilistic Logic Programming Language ProbLog

Angelika Kimmig, Bart Demoen and Luc De Raedt
Departement 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)

1

How Many Programs Are Used to Test Algorithms?

Anytime Inference in Probabilistic Logic Programs with $T_{\mathcal{P}}$-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

4

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)

3

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

Joris Renkens · Guy Van den Broeck · Siegfried Nijssen

1

On the Efficient Execution of ProbLog Programs

Angelika Kimmig[1], Vítor Santos Costa[2], Ricardo Rocha[2], Bart Demoen[1], and Luc De Raedt[1]

1

On the Implementation of the Probabilistic Logic Programming Language ProbLog

Angelika Kimmig, Bart Demoen and Luc De Raedt
Departement 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)

1

ProbLog Technology for Inference in a Probabilistic First Order Logic

Maurice Bruynooghe and **Theofrastos Mantadelis** and **Angelika Kimmig** and **Bernd Gutmann** and **Joost Vennekens** and **Gerda Janssens** and **Luc De Raedt**[1]

2

Outline

Probabilistic Logic Programming

The Constraint Model

Example Programs

Experimental Results

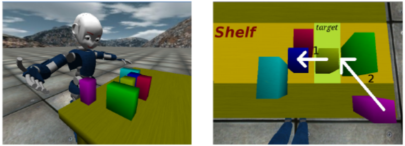
Summary

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

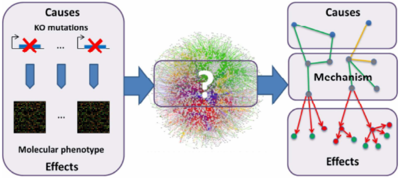
Q1: In a group of 10 people, 60 percent have brown eyes. Two people are to be selected at random from the group. What is the probability that neither person selected will have brown eyes?

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

$$
\begin{aligned}
\Pr[\texttt{cancer}(\mathit{michelle})] = {} & \Pr[\texttt{cancer\_spont}(\mathit{michelle})] \\
& \oplus \Pr[\texttt{smokes}(\mathit{michelle})] \\
& \times \Pr[\texttt{cancer\_smoke}(\mathit{michelle})]
\end{aligned}
$$

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

Probabilistic Inference: Reasoning by Hand

Let $a \oplus b \coloneqq a + b - ab$. Then
$\Pr[\mathtt{cancer}(\mathit{michelle})] = 0.1 \oplus 0.3 \times \Pr[\mathtt{smokes}(\mathit{michelle})]$

```
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

$$\begin{aligned}
\Pr[\texttt{cancer}(\textit{michelle})] &= 0.1 \oplus 0.3 \times \Pr[\texttt{smokes}(\textit{michelle})] \\
\Pr[\texttt{smokes}(\textit{michelle})] &= \Pr[\texttt{stress}(\textit{michelle})] \\
&\oplus \Pr[\texttt{smokes}(\textit{timothy})] \\
&\times \Pr[\texttt{influences}(\textit{timothy}, \textit{michelle})]
\end{aligned}$$

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

Probabilistic Inference: Reasoning by Hand

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

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

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

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

$0.3 :: \texttt{influences}(P_1, P_2) \text{ :- } \texttt{friend}(P_1, P_2).$

Probabilistic Inference: Reasoning by Hand

Let $a \oplus b := a + b - ab$. Then
$\Pr[\texttt{cancer}(\mathit{michelle})] = 0.1 \oplus 0.3 \times \Pr[\texttt{smokes}(\mathit{michelle})]$
$\Pr[\texttt{smokes}(\mathit{michelle})] = 0.2 \oplus 0.3 \times \Pr[\texttt{smokes}(\mathit{timothy})]$
$\Pr[\texttt{smokes}(\mathit{timothy})] = \Pr[\texttt{stress}(\mathit{timothy})] = 0.2$

```
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

$$\mathtt{cancer}(\mathit{michelle}) = \top$$

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

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

$\cancel{0.3 :: \mathtt{influences}(P_1, P_2) :- \mathtt{friend}(P_1, P_2).}$

$\cancel{0.1 ::}\mathtt{cancer\_spont}(P) :- \mathtt{person}(P).$

$\cancel{0.3 ::}\mathtt{cancer\_smoke}(P) :- \mathtt{person}(P).$

$\mathtt{smokes}(X) :- \mathtt{stress}(X).$

$\cancel{\mathtt{smokes}(X) :- \mathtt{smokes}(Y), \mathtt{influences}(Y, X).}$

$\mathtt{cancer}(P) :- \mathtt{cancer\_spont}(P).$

$\mathtt{cancer}(P) :- \mathtt{smokes}(P), \mathtt{cancer\_smoke}(P).$

$\mathtt{person}(\mathit{michelle}).$

$\mathtt{person}(\mathit{timothy}).$

$\mathtt{friend}(\mathit{timothy}, \mathit{michelle}).$

Probabilistic Inference: Probabilities of Worlds

$$\texttt{cancer}(\textit{michelle}) = \top$$

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

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

$\cancel{0.3 ::}$ $\texttt{influences}(P_1, P_2) :- \texttt{friend}(P_1, P_2).$

$\cancel{0.1 ::}$ $\texttt{cancer\_spont}(P) :- \texttt{person}(P).$

$0.3 :: \cancel{\texttt{cancer\_smoke}(P) :- \texttt{person}(P).}$

$\texttt{smokes}(X) :- \texttt{stress}(X).$

$\texttt{smokes}(X) :- \texttt{smokes}(Y), \texttt{influences}(Y, X).$

$\texttt{cancer}(P) :- \texttt{cancer\_spont}(P).$

$\cancel{\texttt{cancer}(P) :- \texttt{smokes}(P), \texttt{cancer\_smoke}(P).}$

$\texttt{person}(\textit{michelle}).$

$\texttt{person}(\textit{timothy}).$

$\texttt{friend}(\textit{timothy}, \textit{michelle}).$

Probabilistic Inference: Probabilities of Worlds

$$\mathtt{cancer}(\mathit{michelle}) = \top$$

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

~~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

$$\mathtt{cancer}(\mathit{michelle}) = \bot$$

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

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

$\cancel{0.3 :: \mathtt{influences}(P_1, P_2) :- \mathtt{friend}(P_1, P_2).}$

$\cancel{0.1 :: \mathtt{cancer\_spont}(P) :- \mathtt{person}(P).}$

$\cancel{0.3 :: \mathtt{cancer\_smoke}(P) :- \mathtt{person}(P).}$

$\cancel{\mathtt{smokes}(X) :- \mathtt{stress}(X).}$

$\cancel{\mathtt{smokes}(X) :- \mathtt{smokes}(Y), \mathtt{influences}(Y, X).}$

$\cancel{\mathtt{cancer}(P) :- \mathtt{cancer\_spont}(P).}$

$\cancel{\mathtt{cancer}(P) :- \mathtt{smokes}(P), \mathtt{cancer\_smoke}(P).}$

$\mathtt{person}(\mathit{michelle}).$

$\mathtt{person}(\mathit{timothy}).$

$\mathtt{friend}(\mathit{timothy}, \mathit{michelle}).$

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

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:

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

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:

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

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

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 $\alpha \lor \beta$, we have $\alpha \land \beta = \bot$
  - for every pair $\alpha \land \beta$, no atoms are shared between α and β
  - examples:

    ✗✗ $(A \lor C) \land (A \lor \neg B)$

    ✗ $C \land (A \lor \neg B)$

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:
  - ✗✗ $(A \vee C) \wedge (A \vee \neg B)$
  - ✗✓ $C \wedge (A \vee \neg B)$

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 $\alpha \wedge \beta = \bot$
- for every pair $\alpha \wedge \beta$, no atoms are shared between α and β
- examples:
  ✗✗ $(A \vee C) \wedge (A \vee \neg B)$
  ✗✓ $C \wedge (A \vee \neg B)$
  $B \wedge C \wedge [(B \wedge A) \vee \neg B]$

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 $\alpha \lor \beta$, we have $\alpha \land \beta = \bot$
  - for every pair $\alpha \land \beta$, no atoms are shared between α and β
  - examples:
    - ✗✗ $(A \lor C) \land (A \lor \neg B)$
    - ✗✓ $C \land (A \lor \neg B)$
    - ✓ $B \land C \land [(B \land A) \lor \neg B]$

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 $\alpha \lor \beta$, we have $\alpha \land \beta = \bot$
- for every pair $\alpha \land \beta$, no atoms are shared between α and β
- examples:
  - ✗✗ $(A \lor C) \land (A \lor \neg B)$
  - ✗✓ $C \land (A \lor \neg B)$
  - ✓✗ $B \land C \land [(B \land A) \lor \neg B]$

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

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

    ✗✓ $C \wedge (A \vee \neg B)$

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

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

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 $\alpha \lor \beta$, we have $\alpha \land \beta = \bot$
  - for every pair $\alpha \land \beta$, no atoms are shared between α and β
  - examples:
    - ✗✗ $(A \lor C) \land (A \lor \neg B)$
    - ✗✓ $C \land (A \lor \neg B)$
    - ✓✗ $B \land C \land [(B \land A) \lor \neg B]$
    - ✓ $C \land [(B \land A) \lor \neg B]$

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 $\alpha \lor \beta$, we have $\alpha \land \beta = \bot$
- for every pair $\alpha \land \beta$, no atoms are shared between α and β
- examples:
  - ✗✗ $(A \lor C) \land (A \lor \neg B)$
  - ✗✓ $C \land (A \lor \neg B)$
  - ✓✗ $B \land C \land [(B \land A) \lor \neg B]$
  - ✓✓ $C \land [(B \land A) \lor \neg B]$

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

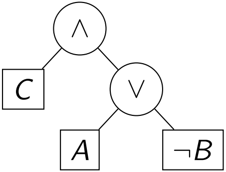


Figure: NNF

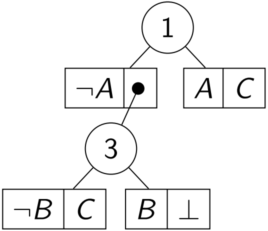


Figure: SDD

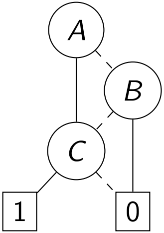


Figure: BDD

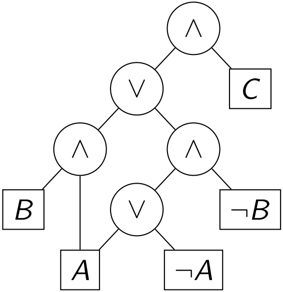


Figure: d-DNNF

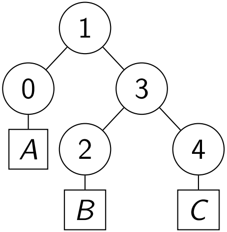


Figure: vtree

What Characterises a (Probabilistic) Logic Program?

$0.2::\texttt{stress}(\mathsf{P}) :\!- \texttt{person}(\mathsf{P}).$

$0.3::\texttt{influences}(\mathsf{P}_1, \mathsf{P}_2) :\!- \texttt{friend}(\mathsf{P}_1, \mathsf{P}_2).$

$0.1::\texttt{cancer\_spont}(\mathsf{P}) :\!- \texttt{person}(\mathsf{P}).$

$0.3::\texttt{cancer\_smoke}(\mathsf{P}) :\!- \texttt{person}(\mathsf{P}).$

$\texttt{smokes}(\mathsf{X}) :\!- \texttt{stress}(\mathsf{X}).$

$\texttt{smokes}(\mathsf{X}) :\!- \texttt{smokes}(\mathsf{Y}), \texttt{influences}(\mathsf{Y}, \mathsf{X}).$

$\texttt{cancer}(\mathsf{P}) :\!- \texttt{cancer\_spont}(\mathsf{P}).$

$\texttt{cancer}(\mathsf{P}) :\!- \texttt{smokes}(\mathsf{P}), \texttt{cancer\_smoke}(\mathsf{P}).$

$\texttt{person}(\mathit{michelle}).$

$\texttt{person}(\mathit{timothy}).$

$\texttt{friend}(\mathit{timothy}, \mathit{michelle}).$

What Characterises a (Probabilistic) Logic Program?

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

$0.3 :: \texttt{influences}(P_1, P_2) \text{ :- } \texttt{friend}(P_1, P_2).$

$0.1 :: \texttt{cancer\_spont}(P) \text{ :- } \texttt{person}(P).$

$0.3 :: \texttt{cancer\_smoke}(P) \text{ :- } \texttt{person}(P).$

$\texttt{smokes}(X) \text{ :- } \texttt{stress}(X).$

$\texttt{smokes}(X) \text{ :- } \texttt{smokes}(Y), \texttt{influences}(Y, X).$

$\texttt{cancer}(P) \text{ :- } \texttt{cancer\_spont}(P).$

$\texttt{cancer}(P) \text{ :- } \texttt{smokes}(P), \texttt{cancer\_smoke}(P).$

$\texttt{person}(\mathit{michelle}).$

$\texttt{person}(\mathit{timothy}).$

$\texttt{friend}(\mathit{timothy}, \mathit{michelle}).$

- 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 :: 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 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).
```

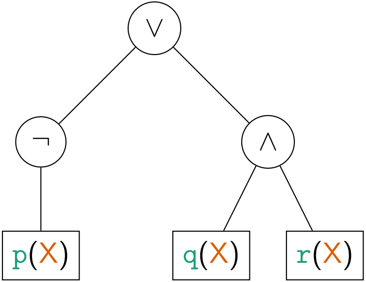
- predicates, arities
- variables
- constants
- probabilities
- length
- complexity

Formulas As Trees

$\neg \mathtt{p}(X) \vee (\mathtt{q}(X) \wedge \mathtt{r}(X))$

Formulas As Trees

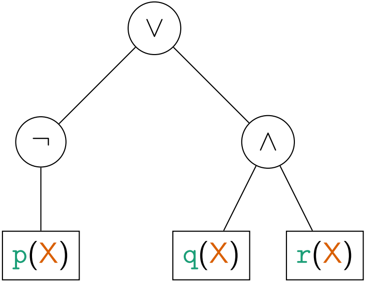
$\neg \mathtt{p}(X) \vee (\mathtt{q}(X) \wedge \mathtt{r}(X))$



Formulas As Trees

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

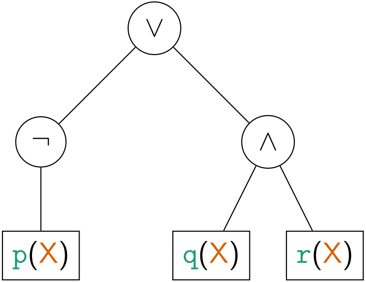
s:	0	0	0	1	2	2	6
v:	$\vee$	$\neg$	$\wedge$	p(X)	q(X)	r(X)	$\top$



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)	$\top$

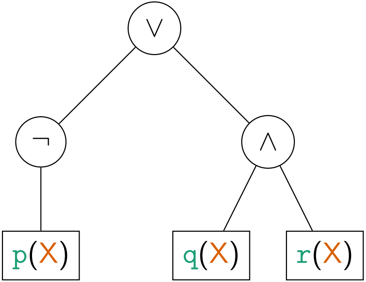


- s is a forest with $T = 2$ trees

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)	$\top$

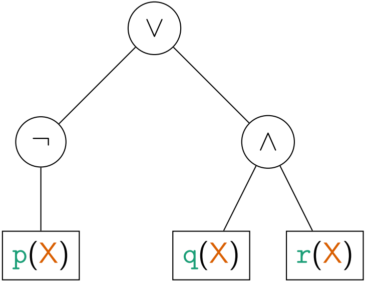


- s is a forest with $T = 2$ trees

Formulas As Trees

$\neg p(X) \lor (q(X) \land r(X))$

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

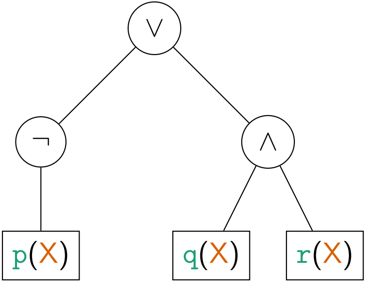


- s is a forest with $T = 2$ trees
- s is sorted
- $s_i \neq i \implies v_i \neq \top$

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)	$\top$



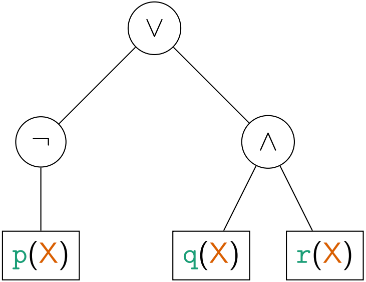
- s is a forest with $T = 2$ trees
- length $L = 7$
- number of nodes $N := L - T + 1 = 6$
- for $i = 1, \ldots, L - 1$,
  - if $i < N$, then $s_i < i$
  - else $s_i = i$ and $v_i = \top$

- 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:	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 sorted
- $s_i \neq i \implies v_i \neq \top$

- s is a forest with $T = 2$ trees
- length $L = 7$
- number of nodes $N := L - T + 1 = 6$
- for $i = 1, \ldots, L - 1$,
  - 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

$$\texttt{smokes}(X) :\!- \texttt{smokes}(Y), \texttt{influences}(Y, X)$$

is equivalent to

$$\texttt{smokes}(Y) :\!- \texttt{smokes}(X), \texttt{influences}(X, Y)$$

and to

$$\texttt{smokes}(W) :\!- \texttt{smokes}(X), \texttt{influences}(X, W)$$

Variable Symmetry Breaking

X	Y	Y	X

Occurrences

(channeling)

$W \mapsto \emptyset$

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

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

Introductions

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

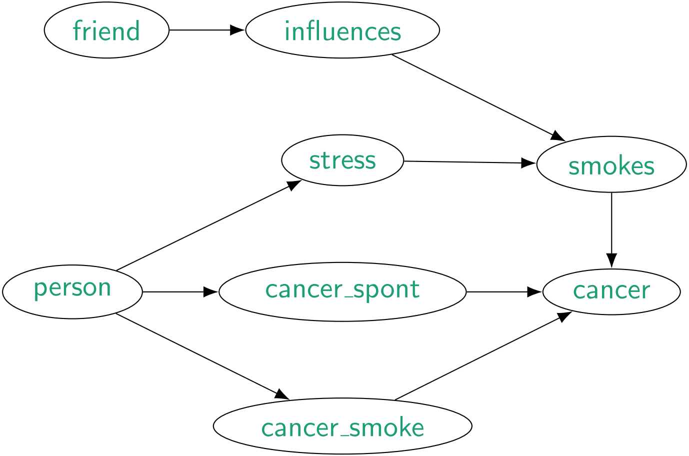
$W \mapsto 1$

$X \mapsto 2$

$Y \mapsto 0$

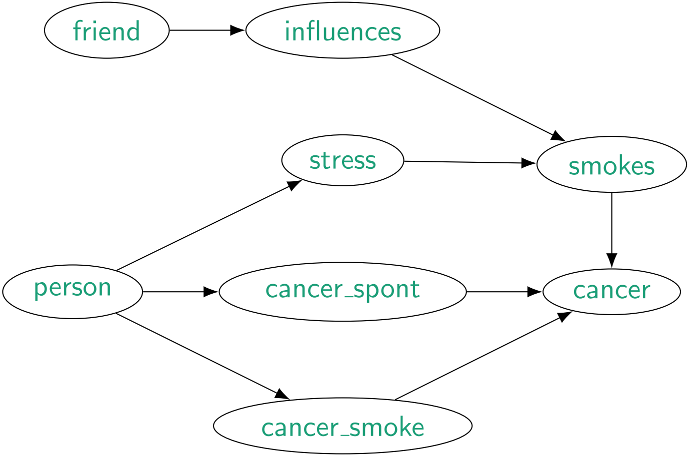
not sorted!

Predicate Dependency Graph



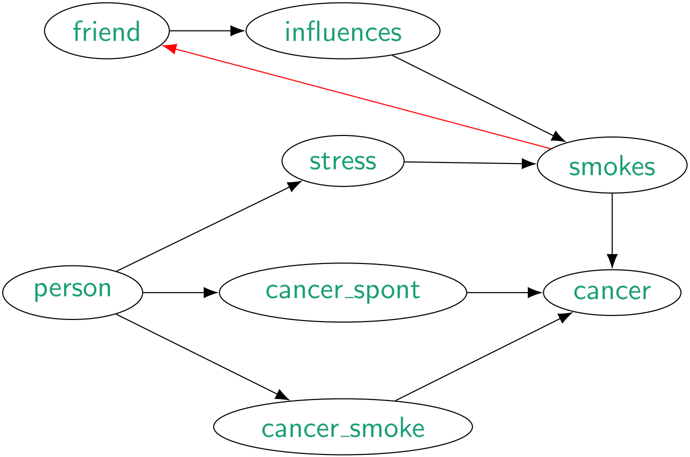
Stratification and Negative Cycles

$$0.1 :: \texttt{friend}(X, Y) \text{ :- } \backslash + \texttt{smokes}(Y).$$



Stratification and Negative Cycles

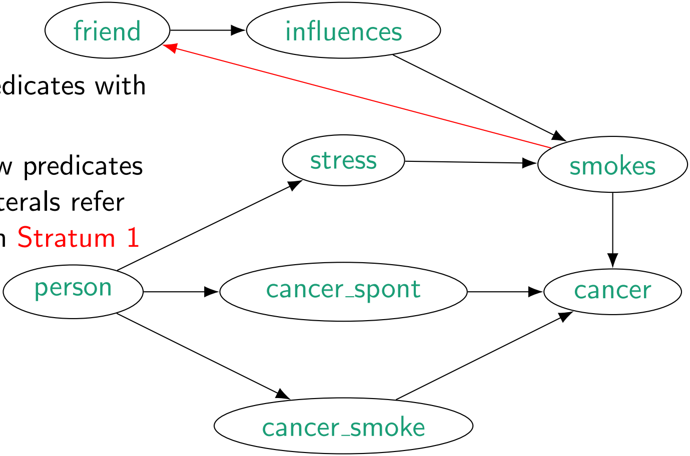
```
0.1::friend(X,Y):-\+smokes(Y).
```



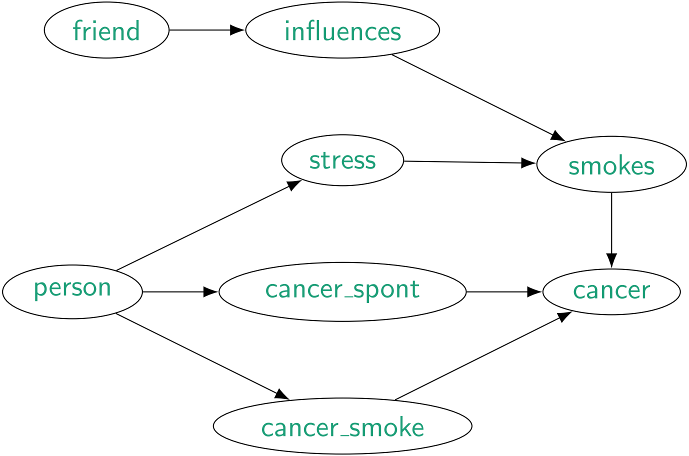
Stratification and Negative Cycles

$$0.1 :: \texttt{friend}(X, Y) :\text{-} \backslash + \texttt{smokes}(Y).$$

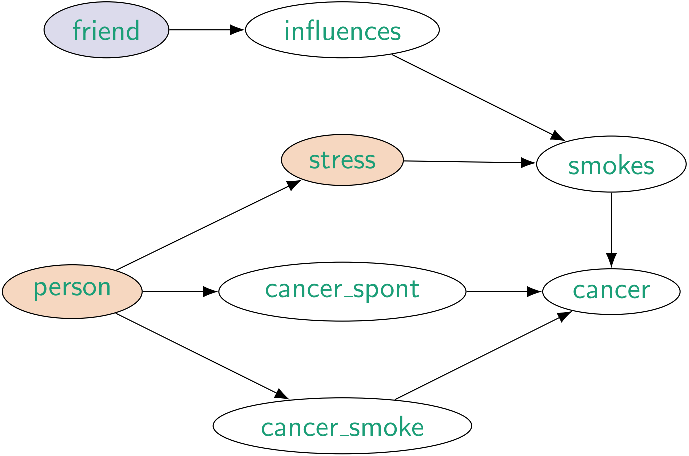
- Stratum 1: predicates with no negation
- Stratum 2: new predicates s.t. negative literals refer to predicates in Stratum 1



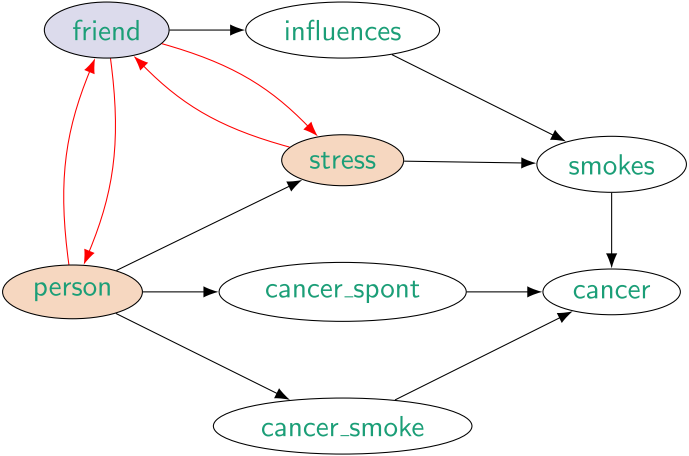
Independence: $\text{friend} \perp\!\!\!\perp \text{stress}$



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



Independence: $\mathrm{friend} \perp\!\!\!\perp \mathrm{stress}$



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

A Random Program

$$\texttt{p}(b) :- \backslash+(\texttt{q}(a, b), \texttt{q}(X, Y), \texttt{q}(Z, X)).$$

$$0.4 :: \texttt{q}(X, X) :- \backslash+ \texttt{r}(Y, Z, a).$$

$$\texttt{q}(X, a) :- \texttt{r}(Y, Y, Z).$$

$$\texttt{q}(X, a) :- \texttt{r}(Y, b, Z).$$

$$\texttt{r}(Y, b, Z).$$

Examples of Predicate Independence

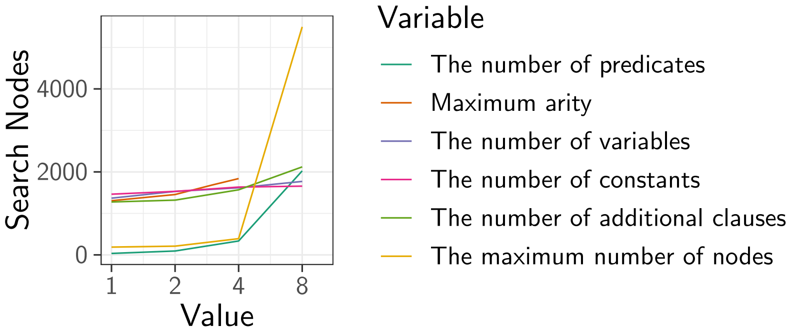
Setup

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

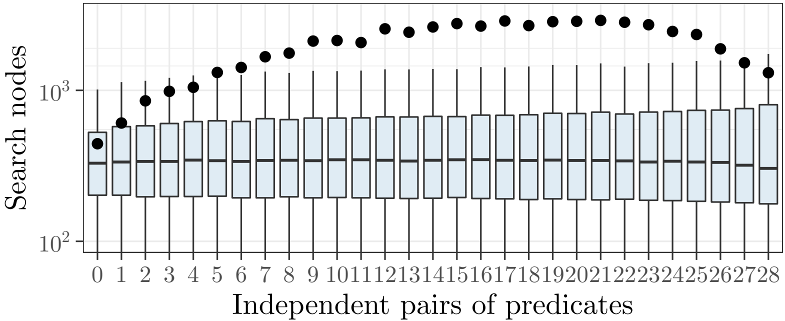
A Few Random Programs

- $0.5 :: \texttt{p}(a) \mathrel{:-} \texttt{p}(a); \texttt{p}(a).$
  $0.2 :: \texttt{q}(a) \mathrel{:-} \texttt{q}(a), \texttt{q}(a).$
  $0.4 :: \texttt{r}(a) \mathrel{:-} \backslash + \texttt{q}(a).$
- $\texttt{p}(a) \mathrel{:-} \texttt{p}(a).$
  $0.5 :: \texttt{q}(a) \mathrel{:-} \texttt{r}(a); \texttt{q}(a).$
  $\texttt{r}(a) \mathrel{:-} \texttt{r}(a); \texttt{r}(a).$
- $\texttt{p}(a) \mathrel{:-} \texttt{p}(a); \texttt{p}(a).$
  $0.6 :: \texttt{q}(a) \mathrel{:-} \texttt{q}(a).$
  $0.7 :: \texttt{r}(a) \mathrel{:-} \backslash + \texttt{q}(a).$

Scalability



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*, *michelle*`).`
- 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 $\{0.1, 0.2, \ldots, 0.9\}$
- query: random unlisted fact

Rules

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

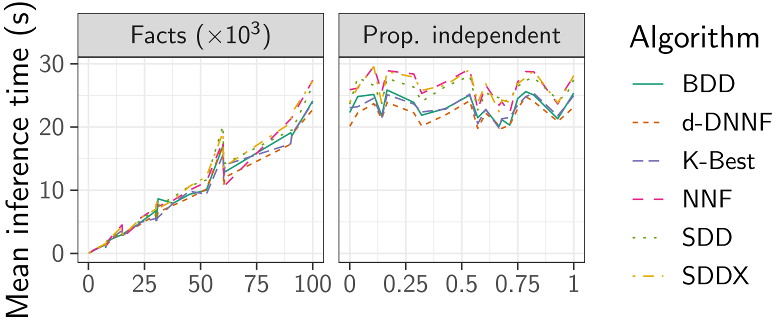
- no constants, no empty bodies
- one rule per predicate
- all rules are probabilistic

Facts

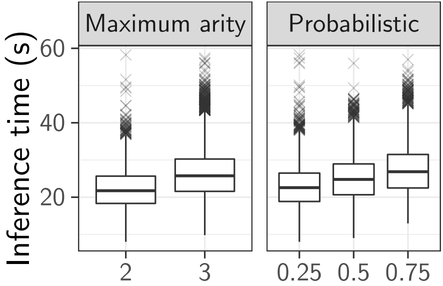
`friend`(*timothy*, *michelle*).

- 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

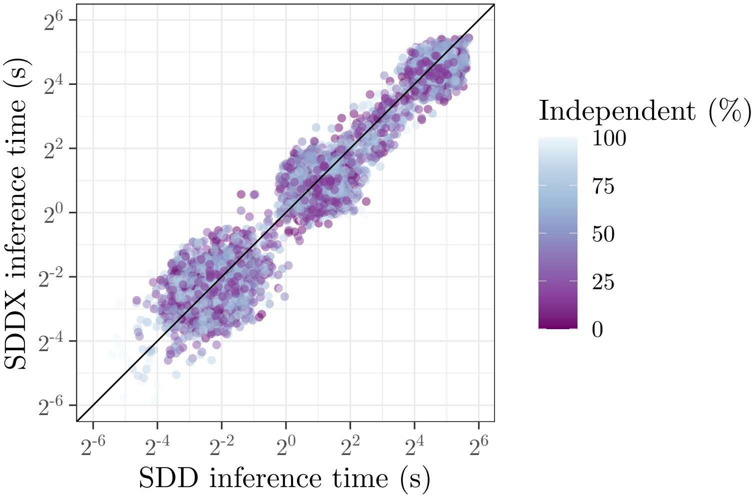
Properties of Programs vs. Inference Algorithms



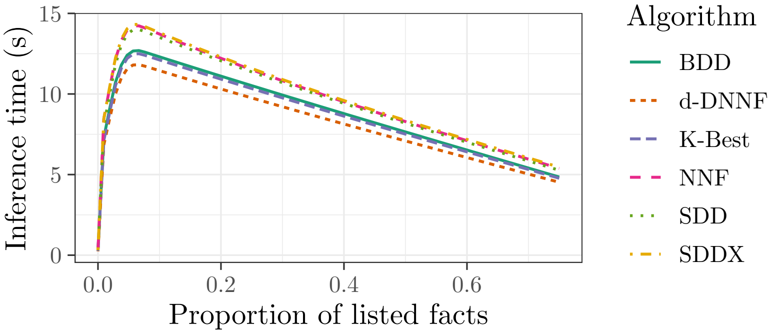
Properties of Programs vs. Inference Algorithms



How Encodings Compare Across Instances



The Ratio of Listed Facts to Possible Facts



Summary

- 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?

Summary

- 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`