Generating Random Logic Programs Using Constraint Programming

Paulius Dilkas

AIAI Seminar

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

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

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

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

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

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

k-Optimal: a novel approximate inference algorithm for ProbLog

Joris Renkens · Guy Van den Broeck · Siegfried Nijssen

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

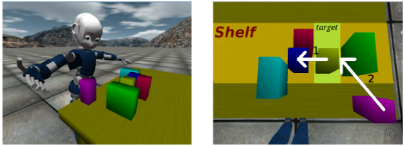
Outline

Probabilistic Logic 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

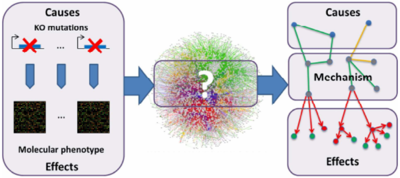
Q1: In a group of 10 people, 60 percent have brown eyes. Two people are to be selected at random from the group. What is the probability that neither person selected will have brown eyes?

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

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

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

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

Probabilistic Inference

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

$$\boxed{\begin{array}{l} 0.1::\texttt{cancer\_spont}(\mathsf{P}) :\text{-}\ \texttt{person}(\mathsf{P}). \\ 0.3::\texttt{cancer\_smoke}(\mathsf{P}) :\text{-}\ \texttt{person}(\mathsf{P}). \end{array}}$$

Probabilistic Inference

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

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

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

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

Probabilistic Inference

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

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

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

$$0.2 :: \texttt{stress}(\mathsf{P}) :\!-\ \texttt{person}(\mathsf{P}).$$
$$0.3 :: \texttt{influences}(\mathsf{P}_1, \mathsf{P}_2) :\!-\ \texttt{friend}(\mathsf{P}_1, \mathsf{P}_2).$$

Probabilistic Inference

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

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

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

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

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

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

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

✗✓ $C \wedge (A \vee \neg B)$

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

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

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

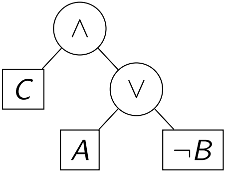


Figure: NNF

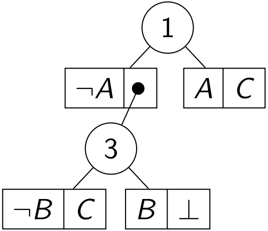


Figure: SDD

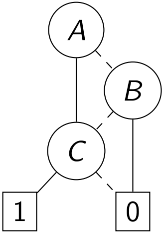


Figure: BDD

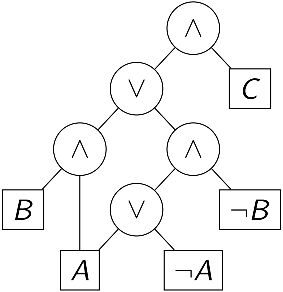


Figure: d-DNNF

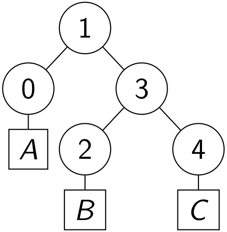


Figure: vtree

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?

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

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

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

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

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

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

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

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

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

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

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

- 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?

$$\left\{\begin{array}{l}
0.2 :: \texttt{stress}(P) \text{ :- } \texttt{person}(P). \\
0.3 :: \texttt{influences}(P_1, P_2) \text{ :- } \texttt{friend}(P_1, P_2). \\
0.1 :: \texttt{cancer_spont}(P) \text{ :- } \texttt{person}(P). \\
0.3 :: \texttt{cancer_smoke}(P) \text{ :- } \texttt{person}(P). \\
\quad \texttt{smokes}(X) \text{ :- } \texttt{stress}(X). \\
\quad \texttt{smokes}(X) \text{ :- } \texttt{smokes}(Y), \texttt{influences}(Y, X). \\
\quad \texttt{cancer}(P) \text{ :- } \texttt{cancer_spont}(P). \\
\quad \texttt{cancer}(P) \text{ :- } \underbrace{\texttt{smokes}(P), \texttt{cancer_smoke}(P)}. \\
\quad \texttt{person}(\mathit{michelle}). \\
\quad \texttt{person}(\mathit{timothy}). \\
\quad \texttt{friend}(\mathit{timothy}, \mathit{michelle}).
\end{array}\right.$$

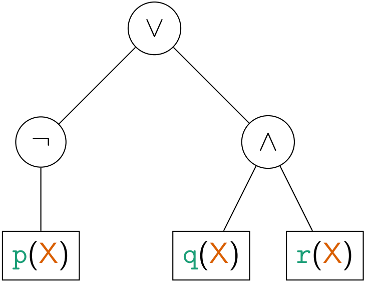
- predicates, arities
- variables
- constants
- probabilities
- length
- complexity

Clauses As Trees

$$\neg\mathtt{p}(X) \lor (\mathtt{q}(X) \land \mathtt{r}(X))$$

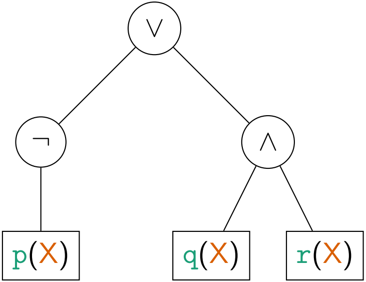
Clauses As Trees

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



Clauses As Trees

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



0	0	0	1	2	2	6
$\vee$	$\neg$	$\wedge$	$\mathtt{p}(X)$	$\mathtt{q}(X)$	$\mathtt{r}(X)$	$\top$

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

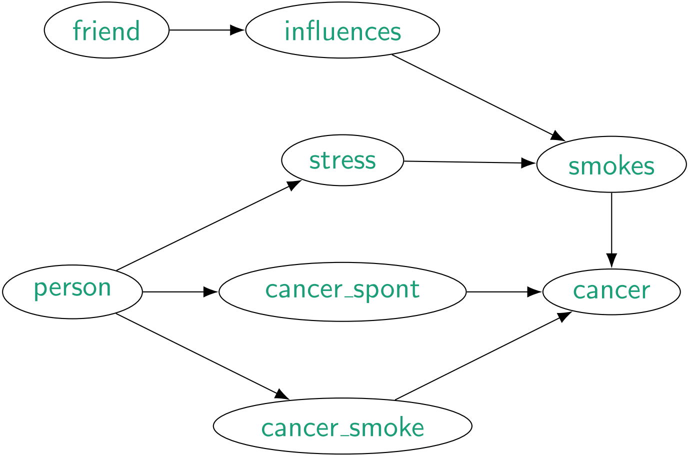
$W \mapsto 1$

$X \mapsto 2$

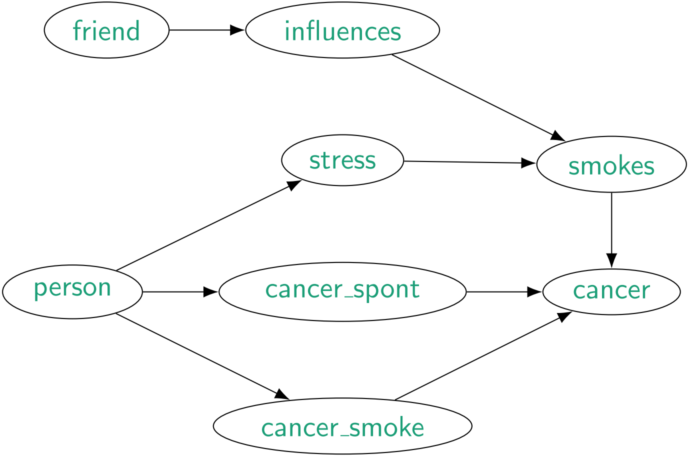
$Y \mapsto 0$

not sorted!

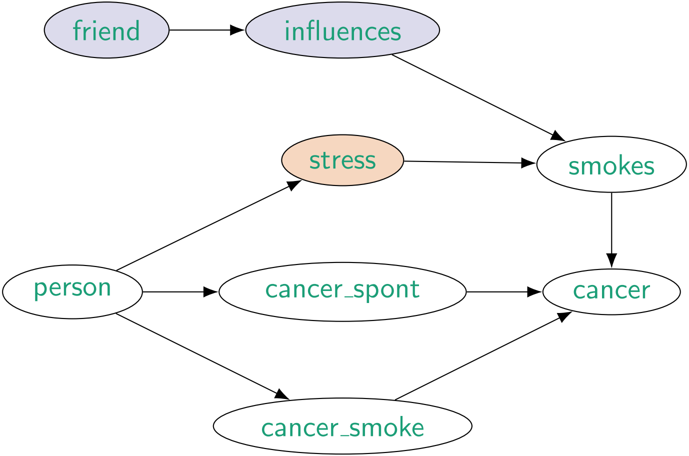
Predicate Dependency Graph



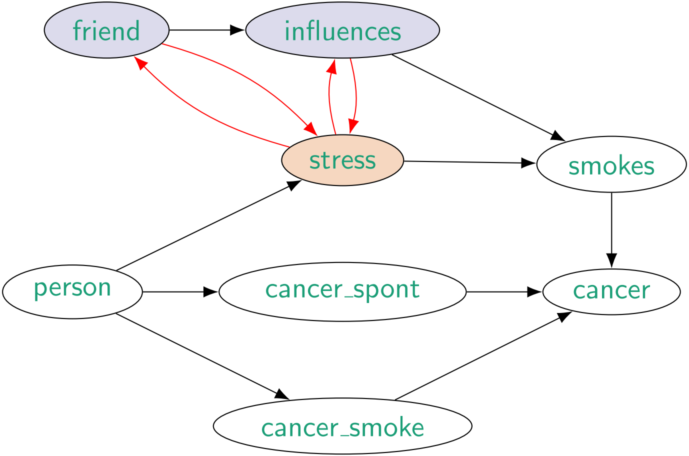
Independence: $\mathrm{influences} \perp\!\!\!\perp \mathrm{stress}$



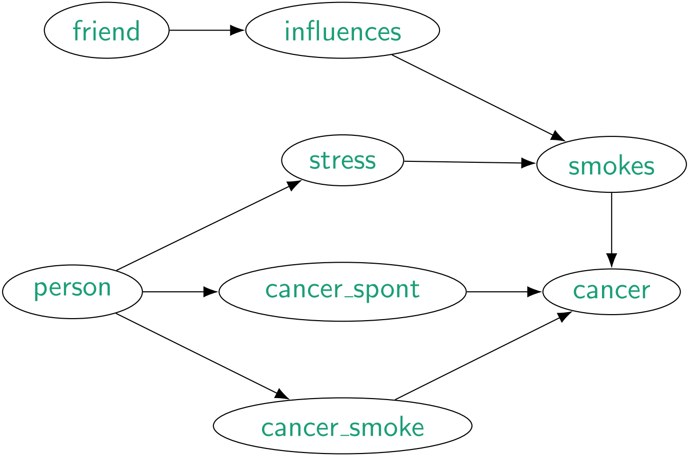
Independence: $\text{influences} \perp\!\!\!\perp \text{stress}$



Independence: influences $\perp\!\!\!\perp$ stress

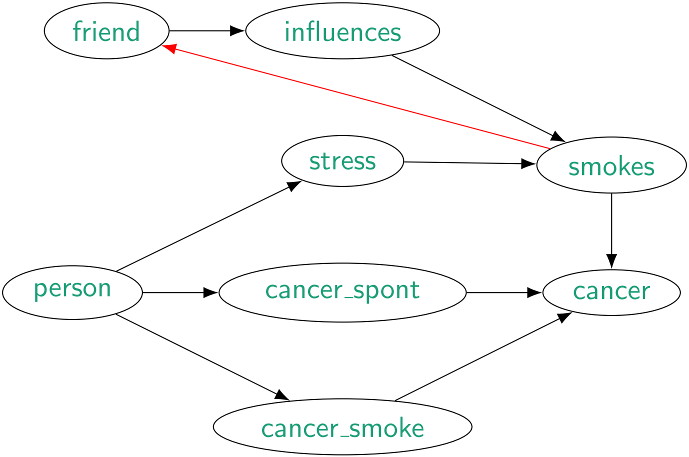


Stratification and Negative Cycles



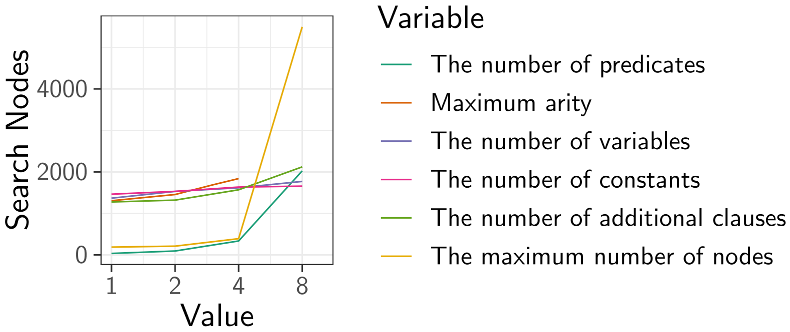
$0.1 :: \texttt{friend}(X, Y) \text{ :- } \backslash + \ \texttt{smokes}(Y).$

Stratification and Negative Cycles

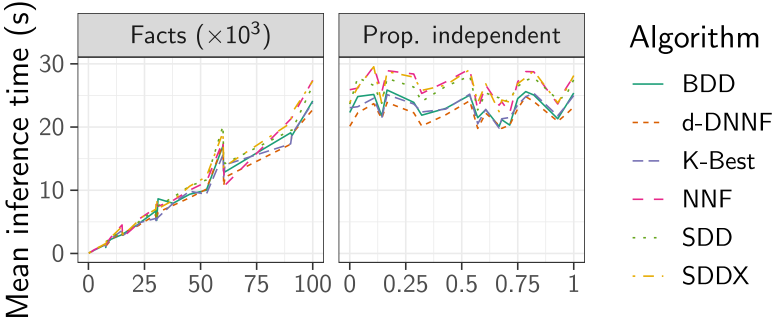


0.1 :: friend(X, Y) :- \+ smokes(Y).

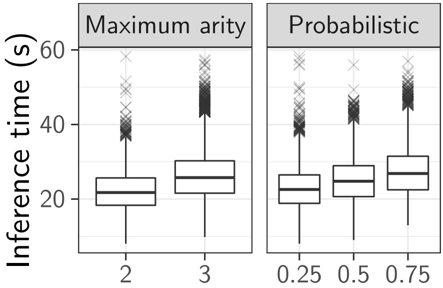
Scalability



Properties of Programs vs. Inference Algorithms



Properties of Programs vs. Inference Algorithms



Summary

- foo
- bar
- baz

The implementation of the model is available at

`https://github.com/dilkas/random-logic-programs`

Clause constraints

- `treeStructure` represents `numTrees` trees.
- $\texttt{treeStructure}[0] = 0$
- $\texttt{numTrees} + \texttt{numNodes} = \texttt{maxNumNodes} + 1$
- `treeStructure` is sorted
- For $i = 0, \ldots, \texttt{maxNumNodes} - 1$,
  - If $\texttt{numNodes} \leq i$,
  - then $\texttt{treeStructure}[i] = i$ and $\texttt{treeValues}[i] = \top$,
  - else $\texttt{treeStructure}[i] < \texttt{numNodes}$.
  - has 0 children $\iff$ $\texttt{treeValues}[i]$ is a predicate
  - has 1 child $\iff$ $\texttt{treeValues}[i] = \neg$
  - has > 1 child $\iff$ $\texttt{treeValues}[i] \in \{\wedge, \vee\}$
  - $\texttt{treeStructure}[i] \neq i \implies \texttt{treeValues}[i] \neq \top$
- If the clause should be disabled, $\texttt{numNodes} = 1$ and $\texttt{treeValues}[0] = \top$.

Adjacency matrix representation

$A[i][j] = 0 \iff \nexists k : \texttt{clauseAssignments}[k] = j$ and $i \in \texttt{clauses}[k].\texttt{treeValues}$

New constraints

- No (negative) cycles
  - No clever propagation, just entailment checking.
- Independence. Propagation:
  - Two types of dependencies: determined and one-undetermined-edge-away-from-being-determined.
  - Look up the dependencies of both predicates. For each pair of matching dependencies:
    - If both are determined, fail.
    - If one is determined, the selected edge of the other must not exist.