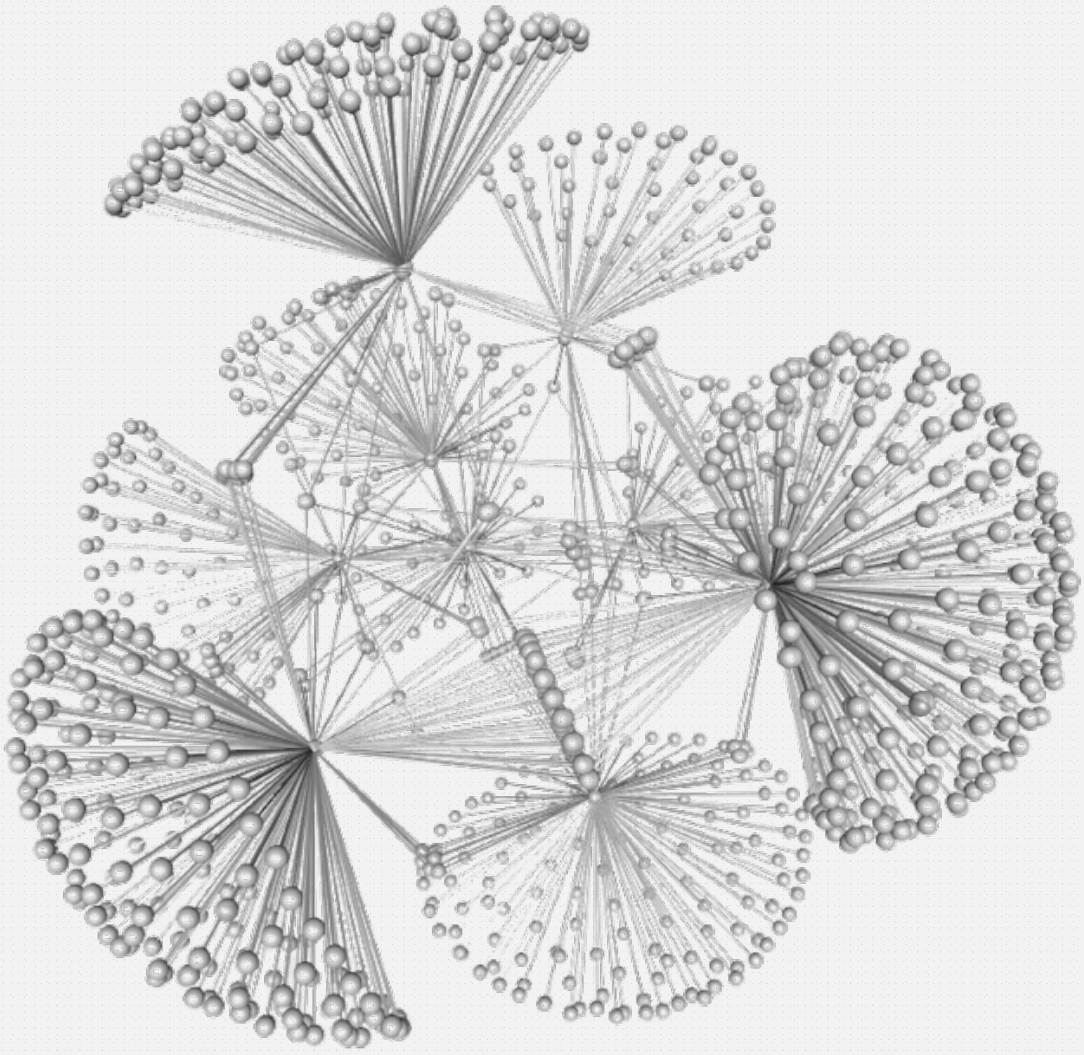
Probabilistic Programming

Michiel Janssen & Bruno Vandekerkhove



Academisch Jaar 2020

H05N0A: Capita Selecta : Artificial Intelligence

Contents

Probabilistic Inference Using Weighted Model Counting **1**
1.1 SRL to CNF . . . 1
1.2 SRL to PGM . . . 2
1.3 PGM to CNF . . . 2
1.4 Weighted Model Counting . . . 4

Lifted Inference **4**

Parameter Learning **4**

Below's our solution for the given challenges. The questions in each section of the original assignment are answered in a section having the same title.

```
person(a).
person(b).
person(c).
0.2::stress(X) :- person(X).
0.1::friends(X,Y) :- person(X), person(Y).
0.3::smokes(X) :- stress(X).
0.4::smokes(X) :- friends(X,Y), smokes(Y).
query(smokes(a)).
```

Code snippet 1: PROBLOG program used throughout the first two chapters of the report.

Probabilistic Inference Using Weighted Model Counting

SRL to CNF

First the program is grounded. This is a matter of collecting all atoms involved in all proofs of the query.

```
0.2::stress(a).
0.2::stress(b).
0.2::stress(c).

0.1::friends(a,a).
0.1::friends(a,b).
0.1::friends(a,c).

0.1::friends(b,a).
0.1::friends(b,b).
0.1::friends(b,c).

0.1::friends(c,a).
0.1::friends(c,b).
0.1::friends(c,c).

0.3::smokes(a) :- stress(a).
0.3::smokes(b) :- stress(b).
0.3::smokes(c) :- stress(c).

0.4::smokes(a) :- friends(a,a), smokes(a).
0.4::smokes(a) :- friends(a,b), smokes(b).
0.4::smokes(a) :- friends(a,c), smokes(c).
0.4::smokes(b) :- friends(b,a), smokes(a).
0.4::smokes(b) :- friends(b,b), smokes(b).
0.4::smokes(b) :- friends(b,c), smokes(c).

0.4::smokes(c) :- friends(c,a), smokes(a).
0.4::smokes(c) :- friends(c,b), smokes(b).
0.4::smokes(c) :- friends(c,c), smokes(c).
```

Code snippet 2: Relevant ground program.

The proofs of the query make for a trie as shown in figure 1, where colourings indicate the presence of cycles. Any proof involving an atom `friends(X,X)` or `friends(Y,a)` (with $Y \in \{b, c\}$) is non-minimal and doesn't affect the final probability. These atoms are disregarded. For the remaining cycles (involving `friends(b,c)` and `friends(c,b)`) auxiliary variables can be used to obtain a cycle-free program without intensional probabilistic facts :

```
0.2::stress(a).
0.2::stress(b).
0.2::stress(c).

```

```
0.1::friends(a,b).
0.1::friends(a,c).
0.1::friends(b,c).
0.1::friends(c,b).

0.3::p(a).
0.3::p(b).
0.3::p(c).

0.4::p(a,b).
0.4::p(a,c).
0.4::p(b,c).
0.4::p(c,b).

smokes(a) :- stress(a), p(a).
smokes(b) :- stress(b), p(b).
smokes(c) :- stress(c), p(c).

smokes(a) :-
    friends(a,b), smokes(b), p(a,b).
smokes(a) :-
    friends(a,c), smokes(c), p(a,c).
smokes(b) :-
    friends(b,c), stress(c), p(c), p(b,c).
smokes(c) :-
    friends(c,b), stress(b), p(b), p(c,b).

query(smokes(a)).
```

Code snippet 3: Relevant ground program without cycles.

The above logic program is equivalent to the following propositional formula :

$$
\begin{aligned}
&(smokes(a) \leftrightarrow \quad (stress(a) \wedge p(a)) \\
&\quad \vee (friends(a,b) \wedge smokes(b) \wedge p(a,b)) \\
&\quad \vee (friends(a,c) \wedge smokes(c) \wedge p(a,c))) \\
&\qquad\qquad \wedge \\
&(smokes(b) \leftrightarrow \quad (stress(b) \wedge p(b)) \\
&\vee (friends(b,c) \wedge stress(c) \wedge p(c) \wedge p(b,c))) \\
&\qquad\qquad \wedge \\
&(smokes(c) \leftrightarrow \quad (stress(c) \wedge p(c)) \\
&\vee (friends(c,b) \wedge stress(b) \wedge p(b) \wedge p(c,b)))
\end{aligned}
$$

Which corresponds to the following `CNF` :

$$
\begin{aligned}
&(\neg smokes(a) \vee stress(a) \vee friends(a,b) \vee friends(a,c)) \\
&\wedge (\neg smokes(a) \vee stress(a) \vee friends(a,b) \vee smokes(c)) \\
&\wedge (\neg smokes(a) \vee stress(a) \vee friends(a,b) \vee p(a,c)) \\
&\wedge (\neg smokes(a) \vee stress(a) \vee smokes(b) \vee friends(a,c)) \\
&\wedge (\neg smokes(a) \vee stress(a) \vee smokes(b) \vee smokes(c)) \\
&\wedge (\neg smokes(a) \vee stress(a) \vee smokes(b) \vee p(a,c)) \\
&\wedge (\neg smokes(a) \vee stress(a) \vee p(a,b) \vee friends(a,c)) \\
&\wedge (\neg smokes(a) \vee stress(a) \vee p(a,b) \vee smokes(c)) \\
&\wedge (\neg smokes(a) \vee stress(a) \vee p(a,b) \vee p(a,c)) \\
&\wedge (\neg smokes(a) \vee p(a) \vee friends(a,b) \vee friends(a,c)) \\
&\wedge (\neg smokes(a) \vee p(a) \vee friends(a,b) \vee smokes(c)) \\
&\wedge (\neg smokes(a) \vee p(a) \vee friends(a,b) \vee p(a,c)) \\
&\wedge (\neg smokes(a) \vee p(a) \vee smokes(b) \vee friends(a,c)) \\
&\wedge (\neg smokes(a) \vee p(a) \vee smokes(b) \vee smokes(c)) \\
&\wedge (\neg smokes(a) \vee p(a) \vee smokes(b) \vee p(a,c)) \\
&\wedge (\neg smokes(a) \vee p(a) \vee p(a,b) \vee friends(a,c)) \\
&\wedge (\neg smokes(a) \vee p(a) \vee p(a,b) \vee smokes(c)) \\
&\wedge (\neg smokes(a) \vee p(a) \vee p(a,b) \vee p(a,c)) \\
&\wedge (\neg stress(a) \vee \neg p(a) \vee smokes(a)) \\
&\wedge (\neg friends(a,b) \vee \neg smokes(b) \vee \neg p(a,b) \vee smokes(a)) \\
&\wedge (\neg friends(a,c) \vee \neg smokes(c) \vee \neg p(a,c) \vee smokes(a)) \\
&\wedge (\neg smokes(b) \vee stress(b) \vee friends(b,c)) \\
&\wedge (\neg smokes(b) \vee stress(b) \vee stress(c)) \\
&\wedge (\neg smokes(b) \vee stress(b) \vee p(c)) \\
&\wedge (\neg smokes(b) \vee stress(b) \vee p(b,c) \\
&\wedge (\neg smokes(b) \vee p(b) \vee friends(b,c)) \\
&\wedge (\neg smokes(b) \vee p(b) \vee stress(c)) \\
&\wedge (\neg smokes(b) \vee p(b) \vee p(c)) \\
&\wedge (\neg smokes(b) \vee p(b) \vee p(b,c) \\
&\wedge (\neg stress(b) \vee \neg p(b) \vee smokes(b)) \\
&\wedge (\neg friends(b,c) \vee \neg stress(c) \vee \neg p(c) \vee \neg p(b,c) \vee \\
&smokes(b)) \\
&\wedge (\neg smokes(c) \vee stress(c) \vee friends(c,b)) \\
&\wedge (\neg smokes(c) \vee stress(c) \vee stress(b)) \\
&\wedge (\neg smokes(c) \vee stress(c) \vee p(b)) \\
&\wedge (\neg smokes(c) \vee stress(c) \vee p(c,b)) \\
&\wedge (\neg smokes(c) \vee p(c) \vee friends(c,b)) \\
&\wedge (\neg smokes(c) \vee p(c) \vee stress(b)) \\
&\wedge (\neg smokes(c) \vee p(c) \vee p(b)) \\
&\wedge (\neg smokes(c) \vee p(c) \vee p(c,b)) \\
&\wedge (\neg stress(c) \vee \neg p(c) \vee smokes(c)) \\
&\wedge (\neg friends(c,b) \vee \neg stress(b) \vee \neg p(b) \vee \neg p(c,b) \vee \\
&smokes(c))
\end{aligned}
$$

The probabilistic literals in the `CNF` are assigned weights (derived literals get a weight of 1) :

Literal	Weight
stress(a)	0.2
¬stress(a)	0.8
stress(b)	0.2
¬stress(b)	0.8
stress(c)	0.2
¬stress(c)	0.8
friends(a,b)	0.1
¬friends(a,b)	0.9
friends(a,c)	0.1
¬friends(a,c)	0.9
friends(b,c)	0.1
¬friends(b,c)	0.9
friends(c,b)	0.1
¬friends(c,b)	0.9
p(a)	0.3
¬p(a)	0.7
p(b)	0.3
¬p(b)	0.7
p(c)	0.3
¬p(c)	0.7
p(a,b)	0.4
¬p(a,b)	0.6
p(a,c)	0.4
¬p(a,c)	0.6
p(b,c)	0.4
¬p(b,c)	0.6
p(c,b)	0.4
¬p(c,b)	0.6

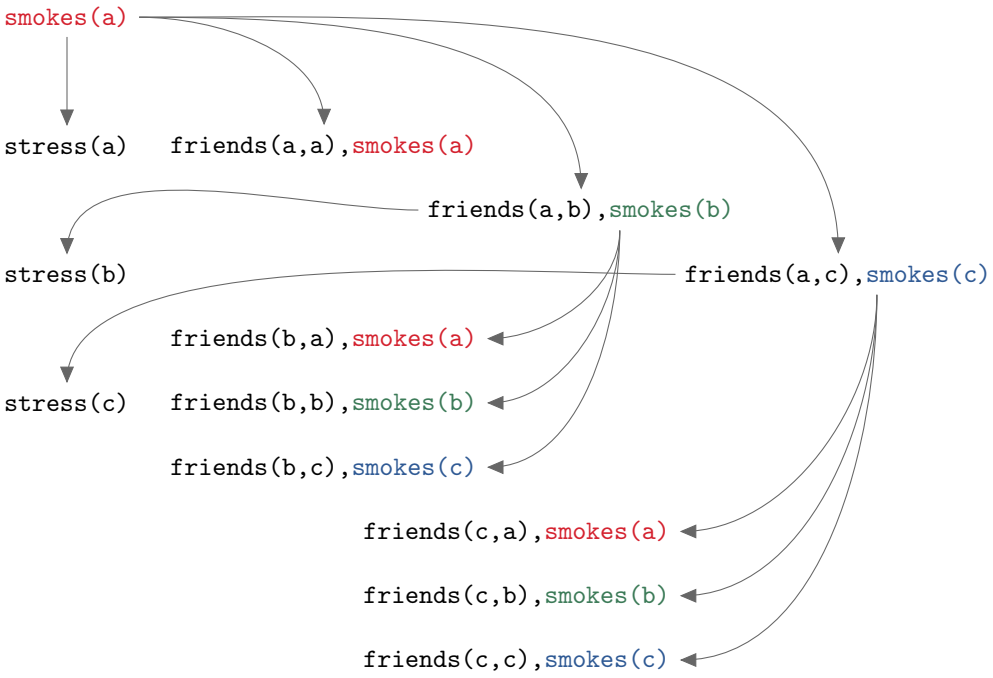


Figure 1: Trie representing proofs of the query. Coloured atoms indicate the presence of cycles.

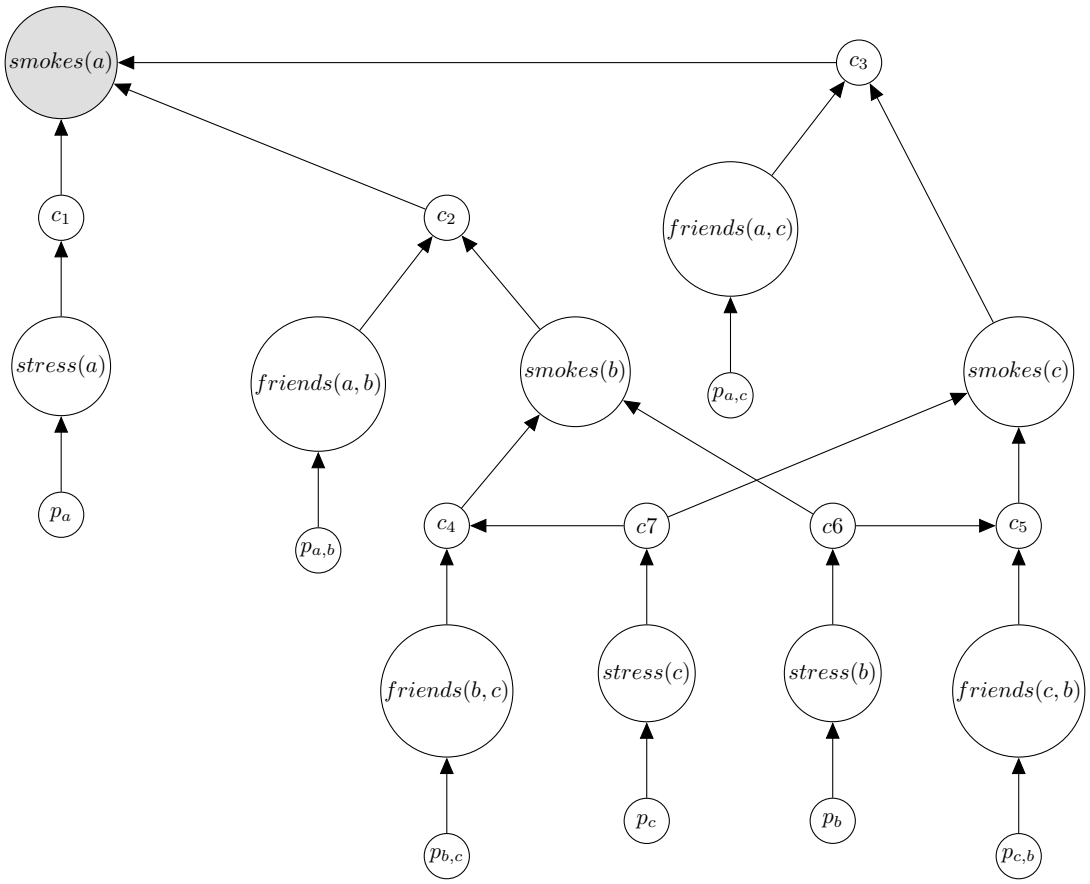


Figure 2: Bayesian network representing the ground acyclic program.

SRL to PGM

A Bayesian network is shown in figure 2. The conditional probability tables (CPTs) for the nodes are the following :

PGM to CNF

The above Bayesian network can subsequently be encoded by a logical formula. Since it consists of nothing but boolean variables, indicator variables representing each value of each variable (e.g. $\lambda_{stress(a)=true}$ and $\lambda_{stress(a)=false}$) are not used in any of the DIMACS files corresponding to encodings presented below. This to avoid unnecessary cluttering. Instead just one atom is introduced for each network variable. This also makes the indicator clauses redundant.

In `ENC1` each row in each CPT is encoded by a parameter clause.

In `ENC2` an order is assumed over each variable's values. Then, each row of each CPT is encoded by an equivalence.

Encoding the deterministic noisy-OR makes either of these more compact. For example, the CPT of `smokes(a)` can be represented by the following one-liner :

$$smokes(a) \Leftrightarrow aux_1 \vee aux_6 \vee aux_7$$

Weighted Model Counting

Lifted Inference

Parameter Learning