Inference for SRL Report

Capita Selecta AI (Probabilistic Programming) 2016-2017

Alexander Tang s0189509
Daniel Andrés Pérez Pérez r0605947

I. Probabilistic Inference Using Weighted Model Counting

A. PGM to CNF

Table I shows the semantics of the domain variables used for those tasks.
Tables II and III show the logical variables used for encoding the Bayesian Network.
Table IV represents the encoded Bayesian Network using ENC1 and table V contains the corresponding weights. Table VI shows the fully expanded CNF from table IV.
Likewise, table VII represents the encoded Bayesian Network using ENC2 and table VIII contains the corresponding weights. Table IX shows the fully expanded CNF from table VII.
The variables that are not listed in table V and table VIII, have a weight equal to 1.

Table I. Variables and domain semantics

Variable	Domain
B = Burglary	b1 = theres is a burglary b2 = theres is no burglary
E = Earthquake	e1 = there is a heavy earthquake e2 = there is a mild earthquake e3 = there is no earthquake
A = Alarm	a1 = alarm rings a2 = alarm does not ring
J = John	j1 = John calls j2 = John does not call
M = Mary	m1 = Mary calls m2 = Mary does not call

Table II. Logical variables using ENC1

Network variables	Indicator Variable	CTP
B	λ_{b1}, λ_{b2}	θ_{b1}, θ_{b2}
E	λ_{e1}, λ_{e2}, λ_3	θ_{e1}, θ_{e2}, θ_3
A	λ_{a1}, λ_{a2}	$\theta_{a1\|b1,e1}$, $\theta_{a1\|b1,e2}$,$\theta_{a1\|b1,e3}$, $\theta_{a1\|b2,e1}$, $\theta_{a1\|b2,e2}$, $\theta_{a1\|b2,e3}$, $\theta_{a2\|b1,e1}$, $\theta_{a2\|b1,e2}$,$\theta_{a2\|b1,e3}$, $\theta_{a2\|b2,e1}$, $\theta_{a2\|b2,e2}$, $\theta_{a2\|b2,e3}$
J	λ_{j1}, λ_{j2}	$\theta_{j1\|a1}$, $\theta_{j2\|a1}$, $\theta_{j1\|a2}$, $\theta_{j2\|a2}$
M	λ_{m1}, λ_{m2}	$\theta_{m1\|a1}$, $\theta_{m2\|a1}$,$\theta_{m1\|a2}$, $\theta_{m2\|a2}$

Table III. Logical variables using ENC2

Variables	Indicator Variable	CTP
B	λ_{b1}, λ_{b2}	ρ_{b1}
E	λ_{e1}, λ_{e2}, λ_3	ρ_{e1}, ρ_{e2}
A	λ_{a1}, λ_{a2}	$\rho_{a1\|b1,e1}$, $\rho_{a1\|b1,e2}$, $\rho_{a1\|b1,e3}$, $\rho_{a1\|b2,e1}$, $\rho_{a1\|b2,c2}$, $\rho_{a1\|b2,e3}$
J	λ_{j1}, λ_{j2}	$\rho_{j1\|a1}$, $\rho_{j1\|a2}$
M	λ_{m1}, λ_{m2}	$\rho_{m1\|a1}$, $\rho_{m1\|a2}$

Table IV. ENC1 REPRESENTATION OF BAYESIAN NETWORK

Variables	CNF	
B	$\lambda_{b1} \vee \lambda_{b2}$	$\lambda_{b1} \Leftrightarrow \theta_{b1}$
	$\neg\lambda_{b1} \vee \neg\lambda_{b2}$	$\lambda_{b2} \Leftrightarrow \theta_{b2}$
E	$\lambda_{e1} \vee \lambda_{e2} \vee \lambda_{e3}$	$\lambda_{e1} \Leftrightarrow \theta_{e1}$
	$\neg\lambda_{e1} \vee \neg\lambda_{e2}$	$\lambda_{e2} \Leftrightarrow \theta_{e2}$
	$\neg\lambda_{e1} \vee \neg\lambda_{e3}$	$\lambda_{e3} \Leftrightarrow \theta_{e3}$
	$\neg\lambda_{e2} \vee \neg\lambda_{e3}$	
A	$\lambda_{a1} \vee \lambda_{a2}$	$\lambda_{a1} \wedge \lambda_{b1} \wedge \lambda_{e1} \Leftrightarrow \theta_{a1\|b1,e1}$
	$\neg\lambda_{a1} \vee \neg\lambda_{a2}$	$\lambda_{a1} \wedge \lambda_{b1} \wedge \lambda_{e2} \Leftrightarrow \theta_{a1\|b1,e2}$
		$\lambda_{a1} \wedge \lambda_{b1} \wedge \lambda_{e3} \Leftrightarrow \theta_{a1\|b1,e3}$
		$\lambda_{a1} \wedge \lambda_{b2} \wedge \lambda_{e1} \Leftrightarrow \theta_{a1\|b2,e1}$
		$\lambda_{a1} \wedge \lambda_{b2} \wedge \lambda_{e2} \Leftrightarrow \theta_{a1\|b2,e2}$
		$\lambda_{a1} \wedge \lambda_{b2} \wedge \lambda_{e3} \Leftrightarrow \theta_{a1\|b2,e3}$
		$\lambda_{a2} \wedge \lambda_{b1} \wedge \lambda_{e1} \Leftrightarrow \theta_{a2\|b1,e1}$
		$\lambda_{a2} \wedge \lambda_{b1} \wedge \lambda_{e2} \Leftrightarrow \theta_{a2\|b1,e2}$
		$\lambda_{a2} \wedge \lambda_{b1} \wedge \lambda_{e3} \Leftrightarrow \theta_{a2\|b1,e3}$
		$\lambda_{a2} \wedge \lambda_{b2} \wedge \lambda_{e1} \Leftrightarrow \theta_{a2\|b2,e1}$
		$\lambda_{a2} \wedge \lambda_{b2} \wedge \lambda_{e2} \Leftrightarrow \theta_{a2\|b2,e2}$
		$\lambda_{a2} \wedge \lambda_{b2} \wedge \lambda_{e3} \Leftrightarrow \theta_{a2\|b2,e3}$
J	$\lambda_{j1} \vee \lambda_{j2}$	$\lambda_{j1} \wedge \lambda_{a1} \Leftrightarrow \theta_{j1\|a1}$
	$\neg\lambda_{j1} \vee \neg\lambda_{j2}$	$\lambda_{j1} \wedge \lambda_{a2} \Leftrightarrow \theta_{j1\|a2}$
		$\lambda_{j2} \wedge \lambda_{a1} \Leftrightarrow \theta_{j2\|a1}$
		$\lambda_{j2} \wedge \lambda_{a2} \Leftrightarrow \theta_{j2\|a2}$
M	$\lambda_{m1} \vee \lambda_{m2}$	$\lambda_{m1} \wedge \lambda_{a1} \Leftrightarrow \theta_{m1\|a1}$
	$\neg\lambda_{m1} \vee \neg\lambda_{m2}$	$\lambda_{m1} \wedge \lambda_{a2} \Leftrightarrow \theta_{m1\|a2}$
		$\lambda_{m2} \wedge \lambda_{a1} \Leftrightarrow \theta_{m2\|a1}$
		$\lambda_{m2} \wedge \lambda_{a2} \Leftrightarrow \theta_{m2\|a2}$

Table V. WEIGHTS ASSOCIATION USING ENC1

Weights	Value
W(θ_{b1})	0.7
W(θ_{b2})	0.3
W(θ_{e1})	0.01
W(θ_{e2})	0.19
W(θ_{e3})	0.80
W($\theta_{a1\|b1,e1}$)	0.90
W($\theta_{a1\|b1,e2}$)	0.85
W($\theta_{a1\|b1,e3}$)	0.80
W($\theta_{a1\|b2,e1}$)	0.30
W($\theta_{a1\|b2,e2}$)	0.10
W($\theta_{a1\|b2,e3}$)	0.00
W($\theta_{a2\|b1,e1}$)	0.10
W($\theta_{a2\|b1,e2}$)	0.15
W($\theta_{a2\|b1,e3}$)	0.20
W($\theta_{a2\|b2,e1}$)	0.70
W($\theta_{a2\|b2,e2}$)	0.90
W($\theta_{a2\|b2,e3}$)	1.00
W($\theta_{j1\|a1}$)	0.80
W($\theta_{j1\|a2}$)	0.10
W($\theta_{j2\|a1}$)	0.20
W($\theta_{j2\|a2}$)	0.90
W($\theta_{m1\|a1}$)	0.80
W($\theta_{m1\|a2}$)	0.10
W($\theta_{m2\|a1}$)	0.20
W($\theta_{m2\|a2}$)	0.90

Table VI. FULL CNF REPRESENTATION OF BAYESIAN NETWORK USING ENC1

Variables	CNF	
B		$\neg\lambda_{b1} \vee \theta_{b1}$
	$\lambda_{b1} \vee \lambda_{b2}$	$\lambda_{b1} \vee \neg\theta_{b1}$
	$\neg\lambda_{b1} \vee \neg\lambda_{b2}$	$\neg\lambda_{b2} \vee \theta_{b2}$
		$\lambda_{b2} \vee \neg\theta_{b2}$
E		$\neg\lambda_{e1} \vee \theta_{e1}$
	$\lambda_{e1} \vee \lambda_{e2} \vee \lambda_{e3}$	$\lambda_{e1} \vee \neg\theta_{e1}$
	$\neg\lambda_{e1} \vee \neg\lambda_{e2}$	$\neg\lambda_{e2} \vee \theta_{e2}$
	$\neg\lambda_{e1} \vee \neg\lambda_{e3}$	$\lambda_{e2} \vee \neg\theta_{e2}$
	$\neg\lambda_{e2} \vee \neg\lambda_{e3}$	$\neg\lambda_{e3} \vee \theta_{e3}$
		$\lambda_{e3} \vee \neg\theta_{e3}$
A		$\neg\lambda_{a1} \vee \neg\lambda_{b1} \vee \neg\lambda_{e1} \vee \theta_{a1\|b1,e1}$
		$(\lambda_{a1} \vee \neg\theta_{a1\|b1,e1}) \wedge (\lambda_{b1} \vee \neg\theta_{a1\|b1,e1}) \wedge (\lambda_{e1} \vee \neg\theta_{a1\|b1,e1})$
		$\neg\lambda_{a1} \vee \neg\lambda_{b1} \vee \neg\lambda_{e2} \vee \theta_{a1\|b1,e2}$
		$(\lambda_{a1} \vee \neg\theta_{a1\|b1,e2}) \wedge (\lambda_{b1} \vee \neg\theta_{a1\|b1,e2}) \wedge (\lambda_{e2} \vee \neg\theta_{a1\|b1,e2})$
		$\neg\lambda_{a1} \vee \neg\lambda_{b1} \vee \neg\lambda_{e3} \vee \theta_{a1\|b1,e3}$
		$(\lambda_{a1} \vee \neg\theta_{a1\|b1,e3}) \wedge (\lambda_{b1} \vee \neg\theta_{a1\|b1,e3}) \wedge (\lambda_{e3} \vee \neg\theta_{a1\|b1,e3})$
		$\neg\lambda_{a1} \vee \neg\lambda_{b2} \vee \neg\lambda_{e1} \vee \theta_{a1\|b2,e1}$
		$(\lambda_{a1} \vee \neg\theta_{a1\|b2,e1}) \wedge (\lambda_{b2} \vee \neg\theta_{a1\|b2,e1}) \wedge (\lambda_{e1} \vee \neg\theta_{a1\|b2,e1})$
		$\neg\lambda_{a1} \vee \neg\lambda_{b2} \vee \neg\lambda_{e2} \vee \theta_{a1\|b2,e2}$
		$(\lambda_{a1} \vee \neg\theta_{a1\|b2,e2}) \wedge (\lambda_{b2} \vee \neg\theta_{a1\|b2,e2}) \wedge (\lambda_{e2} \vee \neg\theta_{a1\|b2,e2})$
		$\neg\lambda_{a1} \vee \neg\lambda_{b2} \vee \neg\lambda_{e3} \vee \theta_{a1\|b2,e3}$
	$\lambda_{a1} \vee \lambda_{a2}$	$(\lambda_{a1} \vee \neg\theta_{a1\|b2,e3}) \wedge (\lambda_{b2} \vee \neg\theta_{a1\|b2,e3}) \wedge (\lambda_{e3} \vee \neg\theta_{a1\|b2,e3})$
	$\neg\lambda_{a1} \vee \neg\lambda_{a2}$	$\neg\lambda_{a2} \vee \neg\lambda_{b1} \vee \neg\lambda_{e1} \vee \theta_{a2\|b1,e1}$
		$(\lambda_{a2} \vee \neg\theta_{a2\|b1,e1}) \wedge (\lambda_{b1} \vee \neg\theta_{a2\|b1,e1}) \wedge (\lambda_{e1} \vee \neg\theta_{a2\|b1,e1})$
		$\neg\lambda_{a2} \vee \neg\lambda_{b1} \vee \neg\lambda_{e2} \vee \theta_{a2\|b1,e2}$
		$(\lambda_{a2} \vee \neg\theta_{a2\|b1,e2}) \wedge (\lambda_{b1} \vee \neg\theta_{a2\|b1,e2}) \wedge (\lambda_{e2} \vee \neg\theta_{a2\|b1,e2})$
		$\neg\lambda_{a2} \vee \neg\lambda_{b1} \vee \neg\lambda_{e3} \vee \theta_{a2\|b1,e3}$
		$(\lambda_{a2} \vee \neg\theta_{a2\|b1,e3}) \wedge (\lambda_{b1} \vee \neg\theta_{a2\|b1,e3}) \wedge (\lambda_{e3} \vee \neg\theta_{a2\|b1,e3})$
		$\neg\lambda_{a2} \vee \neg\lambda_{b2} \vee \neg\lambda_{e1} \vee \theta_{a2\|b2,e1}$
		$(\lambda_{a2} \vee \neg\theta_{a2\|b2,e1}) \wedge (\lambda_{b2} \vee \neg\theta_{a2\|b2,e1}) \wedge (\lambda_{e1} \vee \neg\theta_{a2\|b2,e1})$
		$\neg\lambda_{a2} \vee \neg\lambda_{b2} \vee \neg\lambda_{e2} \vee \theta_{a2\|b2,e2}$
		$(\lambda_{a2} \vee \neg\theta_{a2\|b2,e2}) \wedge (\lambda_{b2} \vee \neg\theta_{a2\|b2,e2}) \wedge (\lambda_{e2} \vee \neg\theta_{a2\|b2,e2})$
		$\neg\lambda_{a2} \vee \neg\lambda_{b2} \vee \neg\lambda_{e3} \vee \theta_{a2\|b2,e3}$
		$(\lambda_{a2} \vee \neg\theta_{a2\|b2,e3}) \wedge (\lambda_{b2} \vee \neg\theta_{a2\|b2,e3}) \wedge (\lambda_{e3} \vee \neg\theta_{a2\|b2,e3})$
J		$\neg\lambda_{j1} \vee \neg\lambda_{a1} \vee \theta_{j1\|a1}$
		$(\lambda_{j1} \vee \neg\theta_{j1\|a1}) \wedge (\lambda_{a1} \vee \neg\theta_{j1\|a1})$
		$\neg\lambda_{j1} \vee \neg\lambda_{a2} \vee \theta_{j1\|a2}$
	$\lambda_{j1} \vee \lambda_{j2}$	$(\lambda_{j1} \vee \neg\theta_{j1\|a2}) \wedge (\lambda_{a2} \vee \neg\theta_{j1\|a2})$
	$\neg\lambda_{j1} \vee \neg\lambda_{j2}$	$\neg\lambda_{j2} \vee \neg\lambda_{a1} \vee \theta_{j2\|a1}$
		$(\lambda_{j2} \vee \neg\theta_{j2\|a1}) \wedge (\lambda_{a1} \vee \neg\theta_{j2\|a1})$
		$\neg\lambda_{j2} \vee \neg\lambda_{a2} \vee \theta_{j2\|a2}$
		$(\lambda_{j2} \vee \neg\theta_{j2\|a2}) \wedge (\lambda_{a2} \vee \neg\theta_{j2\|a2})$
M		$\neg\lambda_{m1} \vee \neg\lambda_{a1} \vee \theta_{m1\|a1}$
		$(\lambda_{m1} \vee \neg\theta_{m1\|a1}) \wedge (\lambda_{a1} \vee \neg\theta_{m1\|a1})$
		$\neg\lambda_{m1} \vee \neg\lambda_{a2} \vee \theta_{m1\|a2}$
	$\lambda_{m1} \vee \lambda_{m2}$	$(\lambda_{m1} \vee \neg\theta_{m1\|a2}) \wedge (\lambda_{a2} \vee \neg\theta_{m1\|a2})$
	$\neg\lambda_{m1} \vee \neg\lambda_{m2}$	$\neg\lambda_{m2} \vee \neg\lambda_{a1} \vee \theta_{m2\|a1}$
		$(\lambda_{m2} \vee \neg\theta_{m2\|a1}) \wedge (\lambda_{a1} \vee \neg\theta_{m2\|a1})$
		$\neg\lambda_{m2} \vee \neg\lambda_{a2} \vee \theta_{m2\|a2}$
		$(\lambda_{m2} \vee \neg\theta_{m2\|a2}) \wedge (\lambda_{a2} \vee \neg\theta_{m2\|a2})$

Table VII. ENC2 REPRESENTATION OF BAYESIAN NETWORK

Variables	CNF	
B	$\lambda_{b1} \vee \lambda_{b2}$	$\rho_{b1} \Rightarrow \lambda_{b1}$
	$\neg\lambda_{b1} \vee \neg\lambda_{b2}$	$\neg\rho_{b1} \Rightarrow \lambda_{b2}$
E	$\lambda_{e1} \vee \lambda_{e2} \vee \lambda_{e3}$	$\rho_{e1} \Rightarrow \lambda_{e1}$
	$\neg\lambda_{e1} \vee \neg\lambda_{e2}$	$\neg\rho_{e1} \wedge \rho_{e2} \Rightarrow \lambda_{e2}$
	$\neg\lambda_{e1} \vee \neg\lambda_{e3}$	$\neg\rho_{e1} \wedge \neg\rho_{e2} \Rightarrow \lambda_{e3}$
	$\neg\lambda_{e2} \vee \neg\lambda_{e3}$	
A	$\lambda_{a1} \vee \lambda_{a2}$	$\lambda_{b1} \wedge \lambda_{e1} \wedge \rho_{a1\|b1,e1} \Rightarrow \lambda_{a1}$
		$\lambda_{b1} \wedge \lambda_{e2} \wedge \rho_{a1\|b1,e2} \Rightarrow \lambda_{a1}$
		$\lambda_{b1} \wedge \lambda_{e3} \wedge \rho_{a1\|b1,e3} \Rightarrow \lambda_{a1}$
		$\lambda_{b2} \wedge \lambda_{e1} \wedge \rho_{a1\|b2,e1} \Rightarrow \lambda_{a1}$
		$\lambda_{b2} \wedge \lambda_{e2} \wedge \rho_{a1\|b2,e2} \Rightarrow \lambda_{a1}$
		$\lambda_{b2} \wedge \lambda_{e3} \wedge \rho_{a1\|b2,e3} \Rightarrow \lambda_{a1}$
	$\neg\lambda_{a1} \vee \neg\lambda_{a2}$	$\lambda_{b1} \wedge \lambda_{e1} \wedge \neg\rho_{a1\|b1,e1} \Rightarrow \lambda_{a2}$
		$\lambda_{b1} \wedge \lambda_{e2} \wedge \neg\rho_{a1\|b1,e2} \Rightarrow \lambda_{a2}$
		$\lambda_{b1} \wedge \lambda_{e3} \wedge \neg\rho_{a1\|b1,e3} \Rightarrow \lambda_{a2}$
		$\lambda_{b2} \wedge \lambda_{e1} \wedge \neg\rho_{a1\|b2,e1} \Rightarrow \lambda_{a2}$
		$\lambda_{b2} \wedge \lambda_{e2} \wedge \neg\rho_{a1\|b2,e2} \Rightarrow \lambda_{a2}$
		$\lambda_{b2} \wedge \lambda_{e3} \wedge \neg\rho_{a1\|b2,e3} \Rightarrow \lambda_{a2}$
J	$\lambda_{j1} \vee \lambda_{j2}$	$\lambda_{a1} \wedge \rho_{j1\|a1} \Rightarrow \lambda_{j1}$
		$\lambda_{a2} \wedge \rho_{j1\|a2} \Rightarrow \lambda_{j1}$
	$\neg\lambda_{j1} \vee \neg\lambda_{j2}$	$\lambda_{a1} \wedge \neg\rho_{j1\|a1} \Rightarrow \lambda_{j2}$
		$\lambda_{a2} \wedge \neg\rho_{j1\|a2} \Rightarrow \lambda_{j2}$
M	$\lambda_{m1} \vee \lambda_{m2}$	$\lambda_{a1} \wedge \rho_{m1\|a1} \Rightarrow \lambda_{m1}$
		$\lambda_{a2} \wedge \rho_{m1\|a2} \Rightarrow \lambda_{m1}$
	$\neg\lambda_{m1} \vee \neg\lambda_{m2}$	$\lambda_{a1} \wedge \neg\rho_{m1\|a1} \Rightarrow \lambda_{m2}$
		$\lambda_{a2} \wedge \neg\rho_{m1\|a2} \Rightarrow \lambda_{m2}$

Table VIII. WEIGHTS ASSOCIATION USING ENC2

Weights	Value
W(ρ_{b1})	0.7
W($\neg\rho_{b1}$)	0.3
W(ρ_{e1})	0.01
W(ρ_{e2})	0.19/(1-0.01) $\approx$ 0.19
W($\neg\rho_{e1}$)	1-0.01 = 0.99
W($\neg\rho_{e2}$)	1-0.19 = 0.81
W($\rho_{a1\|b1,e1}$)	0.90
W($\neg\rho_{a1\|b1,e1}$)	1-0.90=0.10
W($\rho_{a1\|b1,e2}$)	0.85
W($\neg\rho_{a1\|b1,e2}$)	1-0.85=0.15
W($\rho_{a1\|b1,e3}$)	0.80
W($\neg\rho_{a1\|b1,e3}$)	1-0.80=0.20
W($\rho_{a1\|b2,e1}$)	0.30
W($\neg\rho_{a1\|b2,e1}$)	1-0.30=0.70
W($\rho_{a1\|b2,e2}$)	0.10
W($\neg\rho_{a1\|b2,e2}$)	1-0-10=0.90
W($\rho_{a1\|b2,e3}$)	0
W($\neg\rho_{a1\|b2,e3}$)	1-0=1
W($\rho_{j1\|a1}$)	0.80
W($\neg\rho_{j1\|a1}$)	1-0.80=0.20
W($\rho_{j1\|a2}$)	0.10
W($\neg\rho_{j1\|a2}$)	1-0.10=0.90
W($\rho_{m1\|a1}$)	0.80
W($\neg\rho_{m1\|a1}$)	1-0.80=0.20
W($\rho_{m1\|a2}$)	0.10
W($\neg\rho_{m1\|a2}$)	1-0.10=0.90

Table IX. FULL CNF REPRESENTATION OF BAYESIAN NETWORK USING ENC2

Variables	CNF	
B	$\lambda_{b1} \vee \lambda_{b2}$ $\neg\lambda_{b1} \vee \neg\lambda_{b2}$	$\neg\rho_{b1} \vee \lambda_{b1}$ $\rho_{b1} \vee \lambda_{b2}$
E	$\lambda_{e1} \vee \lambda_{e2} \vee \lambda_{e3}$ $\neg\lambda_{e1} \vee \neg\lambda_{e2}$ $\neg\lambda_{e1} \vee \neg\lambda_{e3}$ $\neg\lambda_{e2} \vee \neg\lambda_{e3}$	$\neg\rho_{e1} \vee \lambda_{e1}$ $\rho_{e1} \vee \neg\rho_{e2} \vee \lambda_{e2}$ $\rho_{e1} \vee \rho_{e2} \vee \lambda_{e3}$
A	$\lambda_{a1} \vee \lambda_{a2}$ $\neg\lambda_{a1} \vee \neg\lambda_{a2}$	$\neg\lambda_{b1} \vee \neg\lambda_{e1} \vee \neg\rho_{a1\|b1,e1} \vee \lambda_{a1}$ $\neg\lambda_{b1} \vee \neg\lambda_{e2} \vee \neg\rho_{a1\|b1,e2} \vee \lambda_{a1}$ $\neg\lambda_{b1} \vee \neg\lambda_{e3} \vee \neg\rho_{a1\|b1,e3} \vee \lambda_{a1}$ $\neg\lambda_{b2} \vee \neg\lambda_{e1} \vee \neg\rho_{a1\|b2,e1} \vee \lambda_{a1}$ $\neg\lambda_{b2} \vee \neg\lambda_{e2} \vee \neg\rho_{a1\|b2,e2} \vee \lambda_{a1}$ $\neg\lambda_{b2} \vee \neg\lambda_{e3} \vee \neg\rho_{a1\|b2,e3} \vee \lambda_{a1}$ $\neg\lambda_{b1} \vee \neg\lambda_{e1} \vee \rho_{a1\|b1,e1} \vee \lambda_{a2}$ $\neg\lambda_{b1} \vee \neg\lambda_{e2} \vee \rho_{a1\|b1,e2} \vee \lambda_{a2}$ $\neg\lambda_{b1} \vee \neg\lambda_{e3} \vee \rho_{a1\|b1,e3} \vee \lambda_{a2}$ $\neg\lambda_{b2} \vee \neg\lambda_{e1} \vee \rho_{a1\|b2,e1} \vee \lambda_{a2}$ $\neg\lambda_{b2} \vee \neg\lambda_{e2} \vee \rho_{a1\|b2,e2} \vee \lambda_{a2}$ $\neg\lambda_{b2} \vee \neg\lambda_{e3} \vee \rho_{a1\|b2,e3} \vee \lambda_{a2}$
J	$\lambda_{j1} \vee \lambda_{j2}$ $\neg\lambda_{j1} \vee \neg\lambda_{j2}$	$\neg\lambda_{a1} \vee \neg\rho_{j1\|a1} \vee \lambda_{j1}$ $\neg\lambda_{a2} \vee \neg\rho_{j1\|a2} \vee \lambda_{j1}$ $\neg\lambda_{a1} \vee \rho_{j1\|a1} \vee \lambda_{j2}$ $\neg\lambda_{a2} \vee \rho_{j1\|a2} \vee \lambda_{j2}$
M	$\lambda_{m1} \vee \lambda_{m2}$ $\neg\lambda_{m1} \vee \neg\lambda_{m2}$	$\neg\lambda_{a1} \vee \neg\rho_{m1\|a1} \vee \lambda_{m1}$ $\neg\lambda_{a2} \vee \neg\rho_{m1\|a2} \vee \lambda_{m1}$ $\neg\lambda_{a1} \vee \rho_{m1\|a1} \vee \lambda_{m2}$ $\neg\lambda_{a2} \vee \rho_{m1\|a2} \vee \lambda_{m2}$

B. SRL to CNF

First the program must be grounded, while taking into account **Q** and **E**. In this case the evidence set **E** is empty (there is no evidence available). The grounding process of the queries will be described step-by-step in listings 1 and 2. If only **query(path(1,5))** was considered, then **edge(5,6)** and **edge(2,6)** would have been irrelevant. With the inclusion of **query(path(1,6))** all edges become relevant.

```
% grounding path(1,5) becomes:
path(1,5) :- edge(1,3), 5 \== 3, path(3,5).
path(1,5) :- edge(1,2), 5 \== 2, path(2,5).
% grounding path(3,5)
path(3,5) :- edge(3,4), 5 \== 4, path(4,5).
% grounding path(4,5).
path(4,5) :- edge(4,5).
% grounding path(2,5)
path(2,5) :- edge(2,5).
% putting the results together (and resolving the inequalities) gives:
path(1,5) :- edge(1,3), edge(3,4), edge(4,5).
path(1,5) :- edge(1,2), edge(2,5).
```

Listing 1: Grounding of **path(1,5)**

```
% grounding path(1,6) becomes:
path(1,6) :- edge(1,3), 6 \== 3, path(3,6).
path(1,6) :- edge(1,2), 6 \== 2, path(2,6).
% grounding path(3,6)
path(3,6) :- edge(3,4), 6 \== 4, path(4,6).
% grounding path(4,6).
path(4,6) :- edge(4,5), 6 \== 5, path(5,6).
% grounding path(5,6)
path(5,6) :- edge(5,6).
% grounding path(2,6)
path(2,6) :- edge(2,6).
path(2,6) :- edge(5,6), 6 \== 5, path(5,6).
% path(5,6) has already been grounded
% putting the results together (and resolving the inequalities) gives:
path(1,6) :- edge(1,3), edge(3,4), edge(4,5), edge(5,6).
path(1,6) :- edge(1,2), edge(2,5), edge(5,6).
path(1,6) :- edge(1,2), edge(2,6).
```

Listing 2: Grounding of **path(1,6)**

The second step is to find an equivalent CNF of the ground program. Given the grounded rules **w :- r** and **w :- s**, the equivalent CNF contains the following three clauses: $\neg r \vee w$, $\neg s \vee w$ and $\neg w \vee s \vee r$. In our case r and s both are conjunctions, so De Morgans law is used to write the first two clauses. For the last clause, all permutations of the combinations of the elements $\neg w$, r and s are considered. For **path(1,5)** this yields $2 * 3 = 6$ combinations. For **path(1,6)** there are $2 * 3 * 4 = 24$ combinations. The CNF is shown in table X. Note that on the last big block of $path_{16}$ that the *and* operators can be removed, and the separate clauses can be listed underneath each other. We chose to use the current format because the resulting table would become too large (vertically) otherwise. It also clearly shows which clauses corresponds to the 24 combinations.

Table X. CNF REPRESENTATION OF THE GROUND RULES

Variables	CNF
$path_{15}$	$path_{15} \vee \neg\, edge_{13} \vee \neg\, edge_{34} \vee \neg\, edge_{45}$ $path_{15} \vee \neg\, edge_{12} \vee \neg\, edge_{25}$ $(\neg path_{15} \vee edge_{12} \vee edge_{13}) \wedge (\neg path_{15} \vee edge_{12} \vee edge_{34}) \wedge (\neg path_{15} \vee edge_{12} \vee edge_{45}) \wedge$ $(\neg path_{15} \vee edge_{25} \vee edge_{13}) \wedge (\neg path_{15} \vee edge_{25} \vee edge_{34}) \wedge (\neg path_{15} \vee edge_{25} \vee edge_{45})$
$path_{16}$	$path_{16} \vee \neg\, edge_{13} \vee \neg\, edge_{34} \vee \neg\, edge_{45} \vee \neg\, edge_{56}$ $path_{16} \vee \neg\, edge_{12} \vee \neg\, edge_{25} \vee \neg\, edge_{56}$ $path_{16} \vee \neg\, edge_{12} \vee \neg\, edge_{26}$ $(\neg path_{16} \vee edge_{13} \vee edge_{12}) \wedge (\neg path_{16} \vee edge_{13} \vee edge_{12} \vee edge_{26}) \wedge$ $(\neg path_{16} \vee edge_{13} \vee edge_{25} \vee edge_{12}) \wedge (\neg path_{16} \vee edge_{13} \vee edge_{25} \vee edge_{26}) \wedge$ $(\neg path_{16} \vee edge_{13} \vee edge_{56} \vee edge_{12}) \wedge (\neg path_{16} \vee edge_{13} \vee edge_{56} \vee edge_{26}) \wedge$ $(\neg path_{16} \vee edge_{34} \vee edge_{12}) \wedge (\neg path_{16} \vee edge_{34} \vee edge_{12} \vee edge_{26}) \wedge$ $(\neg path_{16} \vee edge_{34} \vee edge_{25} \vee edge_{12}) \wedge (\neg path_{16} \vee edge_{34} \vee edge_{25} \vee edge_{26}) \wedge$ $(\neg path_{16} \vee edge_{34} \vee edge_{56} \vee edge_{12}) \wedge (\neg path_{16} \vee edge_{34} \vee edge_{56} \vee edge_{26}) \wedge$ $(\neg path_{16} \vee edge_{45} \vee edge_{12}) \wedge (\neg path_{16} \vee edge_{45} \vee edge_{12} \vee edge_{26}) \wedge$ $(\neg path_{16} \vee edge_{45} \vee edge_{25} \vee edge_{12}) \wedge (\neg path_{16} \vee edge_{45} \vee edge_{25} \vee edge_{26}) \wedge$ $(\neg path_{16} \vee edge_{45} \vee edge_{56} \vee edge_{12}) \wedge (\neg path_{16} \vee edge_{45} \vee edge_{56} \vee edge_{26}) \wedge$ $(\neg path_{16} \vee edge_{56} \vee edge_{12}) \wedge (\neg path_{16} \vee edge_{56} \vee edge_{12} \vee edge_{26}) \wedge$ $(\neg path_{16} \vee edge_{56} \vee edge_{25} \vee edge_{12}) \wedge (\neg path_{16} \vee edge_{56} \vee edge_{25} \vee edge_{26}) \wedge$ $(\neg path_{16} \vee edge_{56} \vee edge_{56} \vee edge_{12}) \wedge (\neg path_{16} \vee edge_{56} \vee edge_{56} \vee edge_{26})$

The final step is to obtain a weighted CNF. Since there's no evidence in our example, the CNF remains the same as shown in table X. Table XI displays the weighted literals. The weights for $path_{15}$, $path_{16}$, $\neg path_{15}$ and $\neg path_{16}$ equal 1 because they're defined in clauses. The weight of any world ω can be calculated as the product of the weight of all literals in ω. For example, the world $path_{15}, edge_{12}, edge_{25}, edge_{13}, edge_{34}, \neg edge_{45}$ has the weight $0.6 * 0.4 * 0.1 * 0.3 * 0.2 = 0.00144$.

Table XI. WEIGHTED LITERALS

Variables	Weight	Variables	Weight
$edge_{12}$	0.6	$\neg edge_{12}$	0.4
$edge_{13}$	0.1	$\neg edge_{13}$	0.9
$edge_{25}$	0.4	$\neg edge_{25}$	0.6
$edge_{26}$	0.3	$\neg edge_{26}$	0.7
$edge_{34}$	0.3	$\neg edge_{34}$	0.7
$edge_{45}$	0.8	$\neg edge_{45}$	0.2
$edge_{56}$	0.2	$\neg edge_{56}$	0.8
$path_{15}$	1	$\neg path_{15}$	1
$path_{16}$	1	$\neg path_{16}$	1

C. Weighted Model Counting

We used the following exact model counters: **MiniC2D**, **SDD** and **sharpSAT**.

MiniC2D uses exhaustive DPLL, a backtracking based search algorithm to solve the *boolean satisfiability problem*. It compiles CNFs into Decision-SDDs for the knowledge compilation, using a top-down approach. The top-down approach is considered to be faster by *Umut Oztok and Adnan Darwiche*[1]. The Decision-SDDs are a subset of SDDs which facilitate the top-down compilation of SDDs.

The SDD program is similar to MiniC2D in the sense that it compiles CNFs into SDD datastructures. The shape of the SDD can be manipulated to improve efficiency, as will be explained in section I-D.

sharpSAT also uses DPLL, but combines this with *look ahead* technique that is based on *boolean constraint propagation* [2]. The look ahead technique eliminates more variables that can not be part of a solution, and thus reduces the search space to backtrack over.

Table XII to table XIV show the computational requirements of each exact model counter on the CNF from task 1 with encoding 1, task 1 with encoding 2 and task 2 respectively. There are 36 variables 94 clauses. *The CNF encodings are included with our program.*

Table XII. COMPUTATIONAL REQUIREMENTS TASK 1 ENC1.

	miniC2D	**SDD**	**sharpSAT**
total runtime	0.019s	0.044s	0.008s
memory (cache size)	0.011MB	0.3MB	7MB
cache hit rate	66.7%	85.2%	100%

Table XIII. COMPUTATIONAL REQUIREMENTS TASK 1 ENC2.

	miniC2D	**SDD**	**sharpSAT**
total runtime	0.017s	0.028s	0.004s
memory (cache size)	0.007MB	0.2MB	7MB
cache hit rate	43.4%	84.2%	100%

Table XIV. COMPUTATIONAL REQUIREMENTS TASK 2.

	miniC2D	**SDD**	**sharpSAT**
total runtime	0.020s	0.001s	0.005s
memory (cache size)	0.002MB	0.0MB	7MB
cache hit rate	40.0%	84.4%	100%

D. Knowledge Compilation

We used **MiniC2D** knowledge compiler as tool for this section.

We tested different scenarios for the CNFs of tasks 1.1 and 1.2 (with evidence, no evidence and some

queries). Likewise, we tested the CNFs constructed in tasks 1.3 separately, considering the grounded queries for path(1,5) and path(1,6).

Table XV to table XXIII show the results of the different experiments. According with the results, the best heuristic that gave us the most compact circuit was the *natural elimination order* method with *incidence graph* type, even though *natural elimination order* method with *primal graph* type showed the best performs in terms of computational time.

In contrast, the CNFs of tasks 1.3 (tables XXII and XXIII) presented different results, *natural elimination order* was the worst.

Furthermore, we compared the different vtree parameters on the grounded CNF given by the pipeline described in section II; results are shown in table XXIV. We obtained similar result than CNFs of tasks 1.3, *natural elimination order* showed the widest circuits. *Natural elimination order* was even worst when it was used on combination with *incident graph*. We could not fetch a result after 5 minutes of waiting.

We concluded that the grounding process is an important factor to choose wisely the proper configuration for the creation of the vtrees. Additionally, we concluded that *incident graph* performs better than *primal graph* when it is used in combination with *hypergraph with fixed balance factor* method. This assumption highly relies on the results of table XXIV since the other CNFs show a slightly variation on the vtree circuit. But care must be taken when using the incidence graph. The primal graph shows a uniform distribution in the vtree, while the incidence graph has a larger variation in the depth of leaf nodes. This can cause some computations to go a little faster than using a primal graph, but it can also cause the computation to take much longer. Such an unlucky case is shown in table XXIV.

Table XV. VTREE STATISTICS OF CNF ENC1 WITH NO EVIDENCE

	hypergraph with random balance factor (default)	**hypergraph with fixed balance factor**	**natural elimination order**	**reverse elimination order**	**minfill elimination order**
primal graph	Vtree widths: con<=6, c_con=65 v_con=6 Vtree Time: 0.002s	Vtree widths: con<=6, c_con=65 v_con=6 Vtree Time: 0.003s	Vtree widths: con<=20, c_con=48 v_con=20 Vtree Time: 0.002s	Vtree widths: con<=7, c_con=52 v_con=7 Vtree Time:0.002s	Vtree widths: con<=7, c_con=52 v_con=7 Vtree Time: 0.003s
incidence graph	Vtree widths: con<=7, c_con=55 v_con=7 Vtree Time: 0.003s	Vtree widths: con<=6, c_con=59 v_con=6 Vtree Time: 0.004s	Vtree widths: con<=7, c_con=42 v_con=7 Vtree Time: 0.003s	Vtree widths: con<=7, c_con=52 v_con=7 Vtree Time: 0.002s	Vtree widths: con<=7, c_con=52 v_con=7 Vtree Time: 0.004s

Table XVI. VTREE STATISTICS OF CNF ENC1 WITH QUERY(BURGLARY)

	hypergraph with random balance factor (default)	**hypergraph with fixed balance factor**	**natural elimination order**	**reverse elimination order**	**minfill elimination order**
primal graph	Vtree widths: con<=6, c_con=65 v_con=6 Vtree Time: 0.002s	Vtree widths: con<=6, c_con=65 v_con=6 Vtree Time: 0.002s	Vtree widths: con<=20, c_con=48 v_con=20 Vtree Time: 0.002s	Vtree widths: con<=7, c_con=52 v_con=7 Vtree Time: 0.002s	Vtree widths: con<=7, c_con=52 v_con=7 Vtree Time: 0.002s
incidence graph	Vtree widths: con<=7, c_con=53 v_con=7 Vtree Time: 0.003s	Vtree widths: con<=6, c_con=59 v_con=6 Vtree Time: 0.004s	Vtree widths: con<=7, c_con=42 v_con=7 Vtree Time: 0.003s	Vtree widths: con<=7, c_con=52 v_con=7 Vtree Time: 0.003s	Vtree widths: con<=7, c_con=52 v_con=7 Vtree Time: 0.004s

Table XVII. VTREE STATISTICS OF CNF ENC1 WITH QUERY(EARTHQUAKE=HEAVY)

	hypergraph with random balance factor (default)	**hypergraph with fixed balance factor**	**natural elimination order**	**reverse elimination order**	**minfill elimination order**
primal graph	Vtree widths: con<=6, c_con=68 v_con=6 Vtree Time: 0.002s	Vtree widths: con<=6, c_con=65 v_con=6 Vtree Time: 0.002s	Vtree widths: con<=20, c_con=48 v_con=20 Vtree Time: 0.002s	Vtree widths: con<=7, c_con=52 v_con=7 Vtree Time: 0.002s	Vtree widths: con<=7, c_con=52 v_con=7 Vtree Time: 0.002s
incidence graph	Vtree widths: con<=6, c_con=49 v_con=6 Vtree Time: 0.003s	Vtree widths: con<=6, c_con=59 v_con=6 Vtree Time: 0.003s	Vtree widths: con<=7, c_con=42 v_con=7 Vtree Time: 0.003s	Vtree widths: con<=7, c_con=52 v_con=7 Vtree Time: 0.003s	Vtree widths: con<=7, c_con=52 v_con=7 Vtree Time: 0.003s

Table XVIII. VTREE STATISTICS OF CNF ENC1 WITH EVIDENCE

	hypergraph with random balance factor (default)	**hypergraph with fixed balance factor**	**natural elimination order**	**reverse elimination order**	**minfill elimination order**
primal graph	Vtree widths: con<=6, c_con=65 v_con=6 Vtree Time: 0.003s	Vtree widths: con<=6, c_con=59 v_con=6 Vtree Time: 0.003s	Vtree widths: con<=20, c_con=48 v_con=20 Vtree Time: 0.002s	Vtree widths: con<=7, c_con=52 v_con=7 Vtree Time: 0.003s	Vtree widths: con<=7, c_con=52 v_con=7 Vtree Time: 0.002s
incidence graph	Vtree widths: con<=6, c_con=59 v_con=6 Vtree Time: 0.003s	Vtree widths: con<=6, c_con=49 v_con=6 Vtree Time: 0.003s	Vtree widths: con<=7, c_con=42 v_con=7 Vtree Time: 0.003s	Vtree widths: con<=7, c_con=52 v_con=7 Vtree Time: 0.003s	Vtree widths: con<=7, c_con=52 v_con=7 Vtree Time: 0.004s

II. BUILD AN INFERENCE ENGINE: PIPELINE

The pipeline has been implemented in Python3. The README is in the same directory as the code and will explain how the code should be run. The WMC computation has been carried out in miniC2D whereas

Table XIX. VTREE STATISTICS OF CNF ENC2 WITH NO EVIDENCE

	hypergraph with random balance factor (default)	hypergraph with fixed balance factor	natural elimination order	reverse elimination order	minfill elimination order
primal graph	Vtree widths: con<=7, c_con=20 v_con=7 Vtree Time=0.001s	Vtree widths: con<=7, c_con=20 v_con=7 Vtree Time=0.002s	Vtree widths: con<=9, c_con=17 v_con=9 Vtree Time=0.001s	Vtree widths: con<=7, c_con=20 v_con=7 Vtree Time=0.001s	Vtree widths: con<=7, c_con=20 v_con=7 Vtree Time=0.001s
incidence graph	Vtree widths: con<=7, c_con=27 v_con=7 Vtree Time=0.002s	Vtree widths: con<=7, c_con=20 v_con=7 Vtree Time=0.002s	Vtree widths: con<=7, c_con=18 v_con=7 Vtree Time=0.001s	Vtree widths: con<=7, c_con=20 v_con=7 Vtree Time=0.001s	Vtree widths: con<=7, c_con=23 v_con=7 Vtree Time=0.001s

Table XX. VTREE STATISTICS OF CNF ENC2 WITH QUERY(BURGLARY)

	hypergraph with random balance factor (default)	hypergraph with fixed balance factor	natural elimination order	reverse elimination order	minfill elimination order
primal graph	Vtree widths: con<=7, c_con=20 v_con=7 Vtree Time=0.002s	Vtree widths: con<=7, c_con=20 v_con=7 Vtree Time=0.001s	Vtree widths: con<=9, c_con=17 v_con=9 Vtree Time=0.001s	Vtree widths: con<=7, c_con=20 v_con=7 Vtree Time=0.001s	Vtree widths: con<=7, c_con=20 v_con=7 Vtree Time=0.001s
incidence graph	Vtree widths: con<=7, c_con=20 v_con=7 Vtree Time=0.002s	Vtree widths: con<=7, c_con=20 v_con=7 Vtree Time=0.002s	Vtree widths: con<=7, c_con=18 v_con=7 Vtree Time=0.001s	Vtree widths: con<=7, c_con=20 v_con=7 Vtree Time=0.001s	Vtree widths: con<=7, c_con=23 v_con=7 Vtree Time=0.002s

gvpr (graph pattern scanning and processing language) has been used to obtain the vtree statistics using de dot file created by miniC2D. The results are shown in tables XXV to XXVII. We use the default flag settings for the vtrees because they perform quite well generally. Sometimes there are other vtree flags that perform better, as mentioned in Section I-D, but the difference is negligible. Any of the vtrees that are compact, which corresponds to a low width, is a good candidate.

We used ENC2, as specified in the first task, as our encoding. We chose this because it minimizes the amount of variables needed to get a full CNF. The encoding, as we implemented it, is not always very efficient. In the case of task 1.2 we have more variables than actually needed. Table XXVI shows that we need 16 variables and 28 clauses to represent the CNF for `query(path15)`. In the CNF folder where, where we manually created the CNFs, we only need 6 variables and 8 clauses. The difference can be explained as we created two lambda variables and a rho variable for every atom in ENC2. However, edges are never used in their negation in clauses. This means that there are no rules that express `\+edge(1,3)` for example. The lambda that expresses *not* an edge and the corresponding ρ are actually useless. A possible way to fix this is to scan over all the rules and ensure that the negation never occurs (or that only the negation occurs).

For tasks with no evidence, we only need to compute the model with the query. The number of counted models equals the probability of the query. If there is evidence present, then we need to perform two computations: one with the query and one without the query. Dividing the joint probability (query and evidence $P(Q, E)$) by the joint probability of the conditionals $P(E)$ equals the conditional probability we are looking for.

The results of the big Bayesian network are shown in table XXVII. We have definitely succeeded in topping problog in terms of computational speed. One of the leaf nodes at the bottom of the graph is `hREKG`. This node needed about 10 to 15 minutes to compute in problog, but it was computed by miniC2D in 100 milliseconds on average. Our encoding does not suffer from the overload in variables as in task 1.2 because each atom occurs both in its normal form and its negation in the rule bodies. However, the computed probabilities seem to fluctuate around the result that problog achieves. The computed answer of miniC2D differs every time we run it, but it approximates the correct answer quite closely. We do not know the exact reason for this, but we have some speculations:

- MiniC2D uses rounding frequently for computed intermediate results. The errors cascade and become larger when the amount of variables increase enough.
- We use dictionaries and sets in our program to map logic and remove duplicate elements. This may cause the ordering in which clauses and variables get selected to become random. In conjunction with the rounding problem above, this means that the errors do vary indeed.
- We made an implementation mistake somewhere that only occurs for the Bayesian network somehow.

We have been going on a wrong trail for a very long time, which caused us to lose precious time. For $\sim$60

Table XXI. VTREE STATISTICS OF CNF ENC2 WITH EVIDENCE

	hypergraph with random balance factor (default)	hypergraph with fixed balance factor	natural elimination order	reverse elimination order	minfill elimination order
primal graph	Vtree widths: con<=7, c_con=20 v_con=7 Vtree Time=0.002s	Vtree widths: con<=7, c_con=20 v_con=7 Vtree Time=0.002s	Vtree widths: con<=9, c_con=17 v_con=9 Vtree Time=0.001s	Vtree widths: con<=7, c_con=20 v_con=7 Vtree Time=0.001s	Vtree widths: con<=7, c_con=20 v_con=7 Vtree Time=0.001s
incidence graph	Vtree widths: con<=7, c_con=20 v_con=7 Vtree Time=0.003s	Vtree widths: con<=7, c_con=20 v_con=7 Vtree Time=0.002s	Vtree widths: con<=7, c_con=18 v_con=7 Vtree Time=0.001s	Vtree widths: con<=7, c_con=20 v_con=7 Vtree Time=0.001s	Vtree widths: con<=7, c_con=23 v_con=7 Vtree Time=0.001s

Table XXII. VTREE STATISTICS OF CNF WITH GROUNDED QUERY(PATH15)

	hypergraph with random balance factor (default)	hypergraph with fixed balance factor	natural elimination order	reverse elimination order	minfill elimination order
primal graph	Vtree widths: con<=4, c_con=7 v_con=4 Vtree Time=0.001s	Vtree widths: con<=4, c_con=8 v_con=4 Vtree Time=0.001s	Vtree widths: con<=4, c_con=8 v_con=4 Vtree Time=0.000s	Vtree widths: con<=4, c_con=7 v_con=4 Vtree Time=0.000s	Vtree widths: con<=4, c_con=7 v_con=4 Vtree Time=0.000s
incidence graph	Vtree widths: con<=4, c_con=7 v_con=4 Vtree Time=0.001s	Vtree widths: con<=4, c_con=7 v_con=4 Vtree Time=0.001s	Vtree widths: con<=4, c_con=7 v_con=4 Vtree Time=0.000s	Vtree widths: con<=4, c_con=7 v_con=4 Vtree Time=0.000s	Vtree widths: con<=4, c_con=7 v_con=4 Vtree Time=0.000s

hours we have been dabbling in the problog source code to figure out a way to generate a CNF that fits our encoding. We got on the right track after the discussion the week before the deadline. Due to this, we had to implement some constraints as a shortage of time did not allow us to streamline everything:

- All clauses must be supplied (even the 0 probability ones). In task11, there is a missing clause for *not burglary* ad *no earthquake*. It is possible to sort variables into categories and for each rule to loop over all the combinations of the body. From there we can verify which ones are missing and assign a probability of 0 to them. This required quite some time which we did not have, so we kept it as a constraint.
- We assume in the case of multiple headers, that they are on the same line (and not spread over multiple lines). Looking for all rules that have the exact same body and merging the headers together (separated by a semicolon) will fix this.
- We assume the evidence is always true. If this is not the case, the evidence list must not just accept the names of the evidence, but a tuple of `(<name>,<boolean)`. The getOrderedWeights method must be adjusted accordingly.
- **Task23 does not work.** Queries from task23 must be grounded before we can loop over them, but we can only verify one query at a time. First ground the program, extract all the queries and store them in a list. Then we replace the query line with one of the stored queries at a time. This will means that the queries need to be manually stored every time the program changes, but it would be a possible solution.
- We keep the number of variables to a minimum by replacing occurrences of rules with headers with probability 1 into the body of other rules. This approach works fine for task12, but not for task23 where we may still need those rules. The solution for this would be to see if the rules are still needed (for query or evidence purposes).

The constraints and the solutions for it are explained in more detail in `cnf_converter.py` in the source code.

III. COMPARISON WITH APPROXIMATE INFERENCE

A. SampleCount

Variables with binary values can be set to true or false through logical assertion. If A and B cannot be both true and we know that A is true, then B must be false. The most complete form of propagation is done by generating all possible models and to assert that: (i) a variable I is certainly true if I is true in every model, and (ii) a variable J is certainly false if J is false in every model. Every other variable is in an *unknown* state, which can be either true or false.

The `approxcount` function in **SampleCount** allows us to set a threshold for which a variable is unknown. For example, the `-dont_care_fract N` option considers variables that are positive in (50-N)% to (50+N)% samples to be dont_cares. If N equals 25, then any variable that has a true value between 25% and 75% is unknown. A variable that is positive more than 75% of the samples is certainly true, and certainly false if it is positive less than 25% of the samples.

Table XXIII. VTREE STATISTICS OF CNF WITH GROUNDED QUERY(PATH16)

	hypergraph with random balance factor (default)	hypergraph with fixed balance factor	natural elimination order	reverse elimination order	minfill elimination order
primal graph	Vtree widths: con<=6, c_con=27 v_con=6 Vtree Time=0.001s	Vtree widths: con<=6, c_con=27 v_con=6 Vtree Time=0.001s	Vtree widths: con<=6, c_con=27 v_con=6 Vtree Time=0.000s	Vtree widths: con<=6, c_con=26 v_con=6 Vtree Time=0.000s	Vtree widths: con<=6, c_con=26 v_con=6 Vtree Time=0.000s
incidence graph	Vtree widths: con<=6, c_con=26 v_con=6 Vtree Time=0.002s	Vtree widths: con<=6, c_con=26 v_con=6 Vtree Time=0.002s	Vtree widths: con<=6, c_con=26 v_con=6 Vtree Time=0.001s	Vtree widths: con<=6, c_con=26 v_con=6 Vtree Time=0.001s	Vtree widths: con<=6, c_con=26 v_con=6 Vtree Time=0.001s

Table XXIV. VTREE STATISTICS OF CNF BAYESIAN NETWORK ALARM WITH QUERY (HREKG("LOW"))

	hypergraph with random balance factor (default)	hypergraph with fixed balance factor	natural elimination orde	reverse elimination order	minfill elimination order
primal graph	Vtree widths: con<=17, c_con=273 v_con=17, Vtree Time,0.044s	Vtree widths: con<=17, c_con=273 v_con=17, Vtree Time,0.026s	Vtree widths: con<=65, c_con=364 v_con=69, Vtree Time,0.165s	Vtree widths: con<=22, c_con=302 v_con=22, Vtree Time,0.068s	Vtree widths: con<=13, c_con=204 v_con=13, Vtree Time,0.044s
incidence graph	Vtree widths: con<=17, c_con=195 v_con=17, Vtree Time,0.041s	Vtree widths: con<=17, c_con=195 v_con=17, Vtree Time,0.047s	——	Vtree widths: con<=22, c_con=302 v_con=22, Vtree Time,0.059s	Vtree widths: con<=13, c_con=204 v_con=13, Vtree Time,0.062s

Having this threshold in place means that variables are set to certainly true, even if there are (relatively) few samples that are actually false. This allows a higher degree of propagation and will improve computational speed. The drawback is the loss of accuracy of the results. A model that may not have been considered a valid solution because of a variable that was supposed to be false, may be suddenly true because it is true in the majority of models. This model will then be added to the count of the query while it shouldn't have. This reasoning also holds in the other direction: a model that should have been counted as a solution may be considered invalid.

Inserting this functionality in the pipeline is not very difficult. The execution path of miniC2D in `run.py` should be replaced by the path to `./approxcount` and should accept the generated cnf file as an argument. The appropiate flags can be passed along in the run method in case this inference engine is selected. Note that if no inference engine is entered, then the results will be computed by problog. The result of approxcount are sampled solutions, but these do not contain any weights. To compute the weighted models, the weights must still be added to each literal. MiniC2D then computes the conditional probability for a given query.

B. Search Tree Sampler

The **SearchTreeSampler (STS)** is a sampling technique. The search tree is explored uniformly in a breadth-first way. At each level, a subset of the representative nodes are subsampled. The samples can be used to estimate the model count. So instead of counting all models, it takes a subset of each subtree to generalize over. As with the SampleCount, the accuracy of the result is sacrificed for computational speed, albeit using a different method.

Running this is rather straightforward: only the cnf file is required to return all the samples and corresponding values for the variables. Every level corresponds with a variable, so if there are 319 variables then there are 319 levels. This is not supported in our pipeline because the output is in a completely different format and would require too much time for our implementation to add.

REFERENCES

[1] Umut Oztok and Adnan Darwiche. A top-down compiler for sentential decision diagrams. In *IJCAI*, pages 3141–3148, 2015.

[2] Marc Thurley. sharpsat–counting models with advanced component caching and implicit bcp. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 424–429. Springer, 2006.

Table XXV. RESULTS TASK11

query	probability	count time (s)	vtree time (s)	CNF time (s)	#vars CNF	#clauses CNF	depth vtree	branching factor vtree	#edges circuit	#nodes circuit	total runtime (s)
burglary	$\frac{0.364}{0.372} = 0.98$	0.006	0.001	0.000	24	37	13	2	28	29	0.021
e(none)	$\frac{0.290}{0.371} = 0.78$	0.005	0.000	0.000	24	37	13	2	28	29	0.020

Table XXVI. RESULTS TASK12

query	probability	count time (s)	vtree time (s)	CNF time (s)	#vars CNF	#clauses CNF	depth vtree	branching factor vtree	#edges circuit	#nodes circuit	total runtime (s)
path15	0.258	0.000	0.000	0.000	16	28	10	2	19	20	0.015
path16	0.217	0.002	0.001	0.000	22	55	13	2	27	28	0.017

Table XXVII. RESULTS TASK BAYESIAN NETWORK

query	probability	count time (s)	vtree time (s)	CNF time (s)	#vars CNF	#clauses CNF	depth vtree	branching factor vtree	#edges circuit	#nodes circuit	total runtime (s)
hREKG(LOW)	0.178	0.039	0.015	0.001	319	482	32	2	420	421	0.070
pRESS(HIGH)	0.484	0.008	0.003	0.000	83	130	22	2	104	105	0.026
aRTCO2(LOW)	0.193	0.023	0.006	0.000	179	248	28	2	228	229	0.045
sAO2(LOW)	0.770	0.027	0.007	0.000	214	302	30	2	274	275	0.050