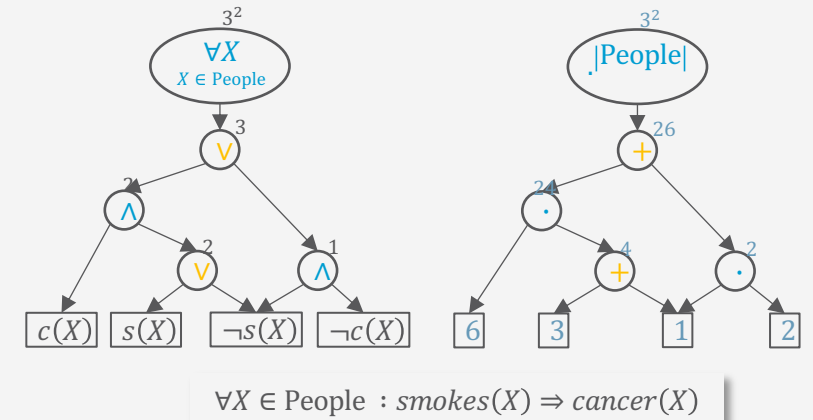


Exact Inference: (Weighted) Model Counting

Statistical Relational Artificial Intelligence
(StaRAI)



Contents

1. Introduction

- Notions of AI, agents, StaRAI

2. Modelling Formalisms

- a. *Logic*
- b. *Probabilistic Graphical Models (PGMs)*
- c. *Probabilistic Relational Models (PRMs)*
- Syntax, semantics, inference tasks

3. Inference

- a. *Exact Inference*
 - i. *(Weighted) Model Counting*
 - ii. *Variable Elimination*
 - iii. *Lifted Inference*

b. *Approximate Inference*

- *Inference algorithms*

4. Learning

- Relation learning in logic
- Parameter / structure learning for PGMs
- From PGMs to PRMs, PRM learning

5. Sequential Models and Inference

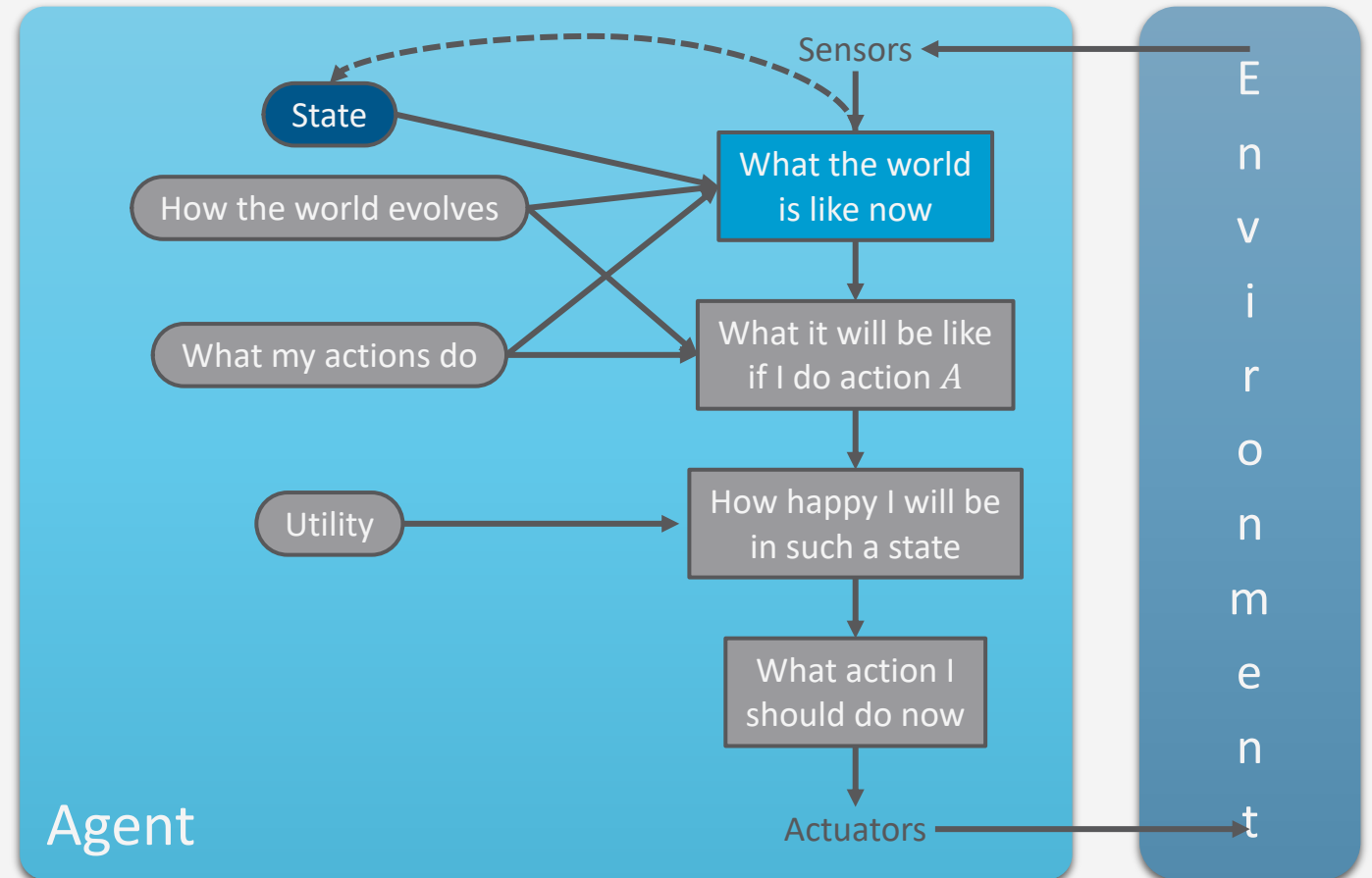
- Syntax, semantics, inference tasks, algorithms

6. Decision-theoretic Models and Inference

- Preferences, utility
- Syntax, semantics, inference task, algorithm

Contents in this Lecture Related to *Utility-based Agents*

- Further topics
 2. Modelling formalisms
 3. Inference (in episodic models)
 4. Learning (of episodic models)
 5. Sequential models and inference
 6. Decision-theoretic models and inference



Outline: 3. Inference

a. Exact Inference

i. (Weighted) Model Counting

- Model counting
- Weighted model counting, knowledge compilation (KC)

ii. Variable Elimination and the Like

- Variable elimination (VE)
- Junction tree algorithm (JT)

iii. Lifted Inference

- First-order KC, Lifted VE, lifted JT

b. Approximate Inference

- Direct sampling: Rejection sampling, (lifted) importance sampling
- (Lifted) Markov Chain Monte Carlo sampling

Model Counting

- Solve inference problem by solving a **model counting** problem
 - Model count (**MC**) given a sentence φ in propositional logic is defined by

$$MC(\varphi) = \sum_{\omega \in \Omega_{\varphi}} 1 | \omega \models \varphi$$

- where Ω_{φ} refers to the set of worlds of φ
- Example
 - Sentence: $sun \wedge rain \Rightarrow rainbow$

$$MC(\varphi) = \sum_{\omega \in \Omega_{\varphi}} \underbrace{1 | \omega \models \varphi}$$

<i>rain</i>	<i>sun</i>	<i>rainbow</i>	Count
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1
			7

Each line a world $\omega \in \Omega_{\varphi}$

MC Complexity

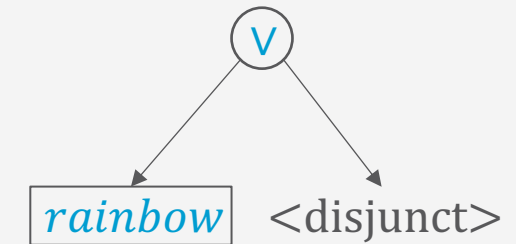
- Solving an MC problem for a sentence φ as introduced on previous slide is exponential in number of constants
- To be more efficient, build a helper structure $\rightarrow$ circuits
 - First, bring sentence into negation normal form (NNF)
 - NNF: Formulas contain only negations directly in front of variables, conjunctions, and disjunctions
 - E.g.,
 - $sun \wedge rain \Rightarrow rainbow$ (Apply $A \Rightarrow B \equiv \neg A \vee B$)
 $\equiv \neg(sun \wedge rain) \vee rainbow$ (Apply De Morgan's law)
 $\equiv \neg sun \vee \neg rain \vee rainbow$ (NNF)

Circuits

- Represent the NNF sentence as a directed, acyclic graph called **circuit** with leaves labelled with literals (l or $\neg l$) or *true*, *false* with inner nodes being
 - *Deterministic* disjunctions
 - Only one disjunct (child node) can be true at the same time
 - I.e., their conjunction is unsatisfiable
 - *Decomposable* conjunctions
 - Each pair of conjuncts (child nodes) must be independent
 - I.e., they cannot share any variables
- Circuit is then in **d-DNNF**
 - deterministic Decomposable NNF
 - See later why important

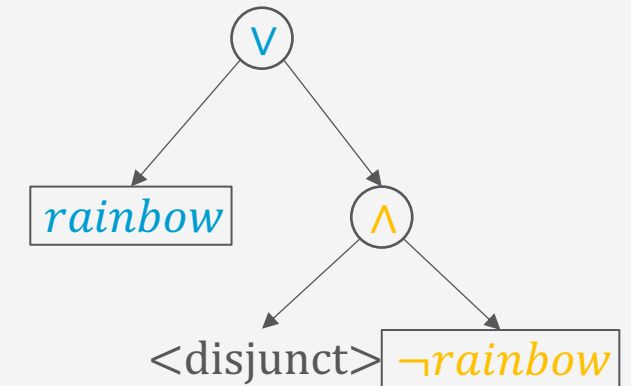
Circuits: Example

- *Deterministic* disjunctions
 - Only one disjunct (child node) can be true at the same time
 - I.e., their conjunction is unsatisfiable
- *Decomposable* conjunctions
 - Each pair of conjuncts (child nodes) must be independent
 - I.e., they cannot share any variables
- E.g., $\neg sun \vee \neg rain \vee rainbow$
- $\langle \text{disjunct} \rangle \vee rainbow$
 - Determinism: $\langle \text{disjunct} \rangle$ can only be true if $rainbow$ is not
 - Add $\neg rainbow$ to disjunct: $\neg rainbow \wedge \langle \text{disjunct} \rangle$



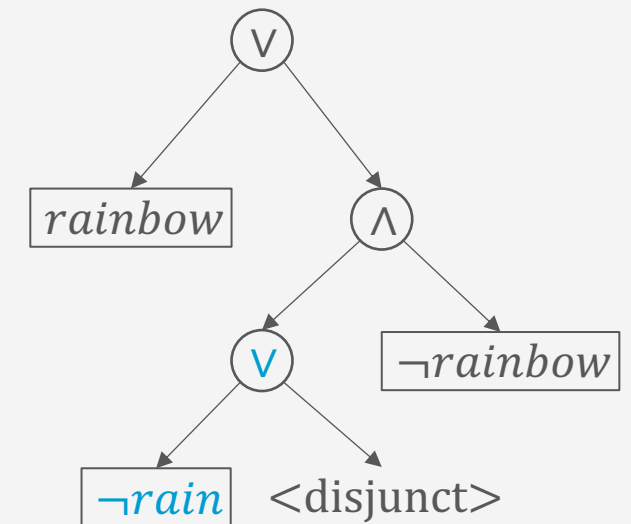
Circuits: Example

- *Deterministic* disjunctions
 - Only one disjunct (child node) can be true at the same time
 - I.e., their conjunction is unsatisfiable
- *Decomposable* conjunctions
 - Each pair of conjuncts (child nodes) must be independent
 - I.e., they cannot share any variables
- E.g., $\neg sun \vee \neg rain \vee rainbow$
- $\langle \text{disjunct} \rangle \vee rainbow$
 - Determinism: $\langle \text{disjunct} \rangle$ can only be true if $rainbow$ is not
 - Add $\neg rainbow$ to disjunct: $\neg rainbow \wedge \langle \text{disjunct} \rangle$
 - $\langle \text{disjunct} \rangle$ now part of a conjunction with $\neg rainbow$
 - Decomposability: May not contain $Rainbow$



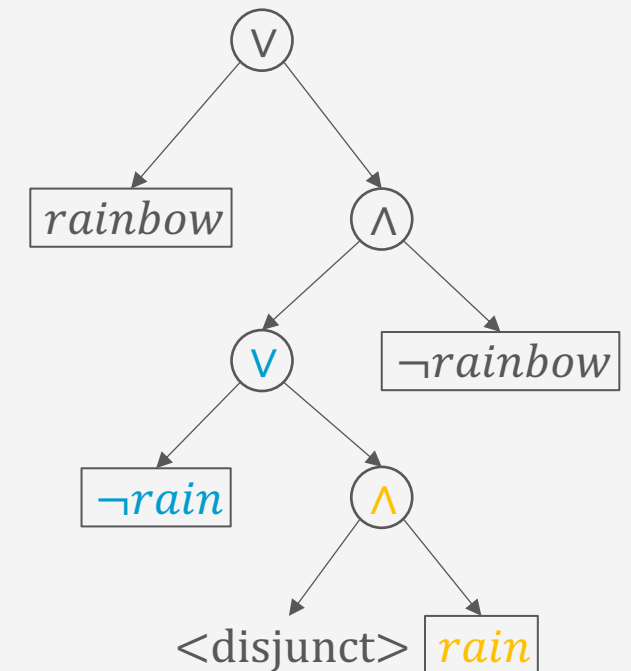
Circuits: Example

- *Deterministic* disjunctions
 - Only one disjunct (child node) can be true at the same time
 - I.e., their conjunction is unsatisfiable
- *Decomposable* conjunctions
 - Each pair of conjuncts (child nodes) must be independent
 - I.e., they cannot share any variables
- E.g., $\neg sun \vee \neg rain \vee rainbow$
- $\langle \text{disjunct} \rangle \vee \neg rain$
 - Determinism: $\langle \text{disjunct} \rangle$ can only be true if $\neg rain$ is not, i.e., if $rain$ is
 - Add $rain$ to disjunct: $rain \wedge \langle \text{disjunct} \rangle$



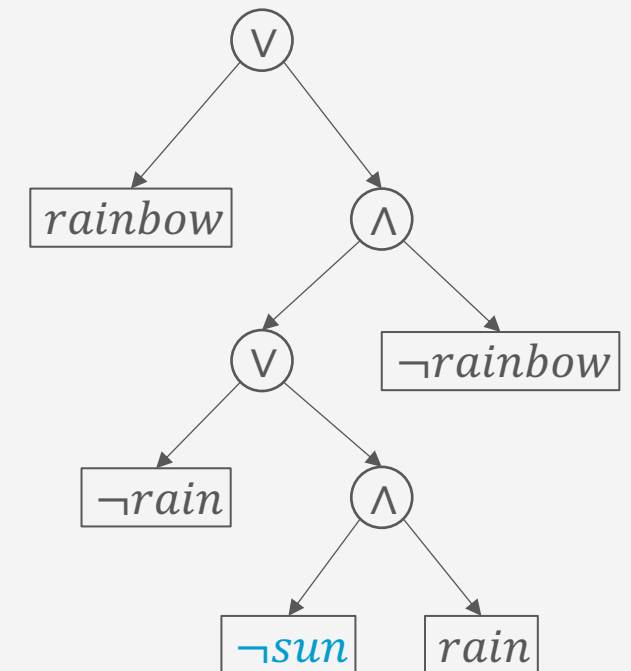
Circuits: Example

- *Deterministic* disjunctions
 - Only one disjunct (child node) can be true at the same time
 - I.e., their conjunction is unsatisfiable
- *Decomposable* conjunctions
 - Each pair of conjuncts (child nodes) must be independent
 - I.e., they cannot share any variables
- E.g., $\neg sun \vee \neg rain \vee rainbow$
- $\langle \text{disjunct} \rangle \vee \neg rain$
 - Determinism: $\langle \text{disjunct} \rangle$ can only be true if $\neg rain$ is not, i.e., if $rain$ is
 - Add $rain$ to disjunct: $rain \wedge \langle \text{disjunct} \rangle$
 - $\langle \text{disjunct} \rangle$ now part of a conjunction with $rain$
 - Decomposability: May not contain $Rain$



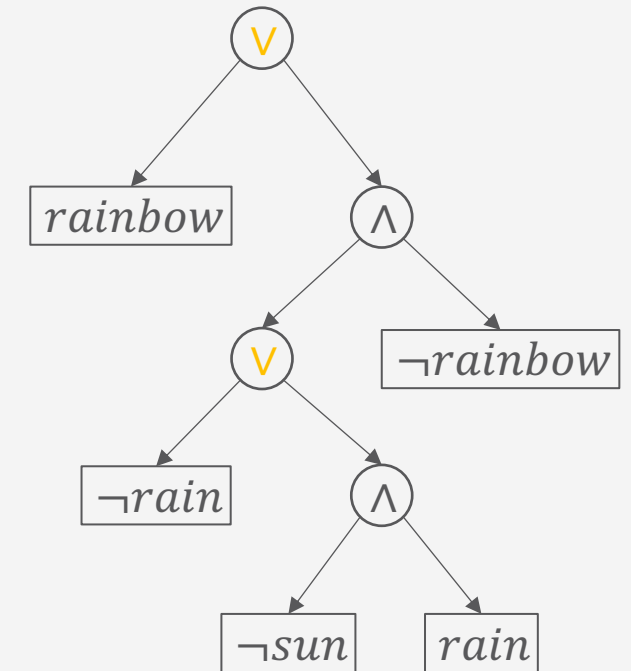
Circuits: Example

- *Deterministic* disjunctions
 - Only one disjunct (child node) can be true at the same time
 - I.e., their conjunction is unsatisfiable
- *Decomposable* conjunctions
 - Each pair of conjuncts (child nodes) must be independent
 - I.e., they cannot share any variables
- E.g., $\neg sun \vee \neg rain \vee rainbow$
 - Add $\neg sun$ as conjunct
 - Decomposability: Does not share variables with sibling node



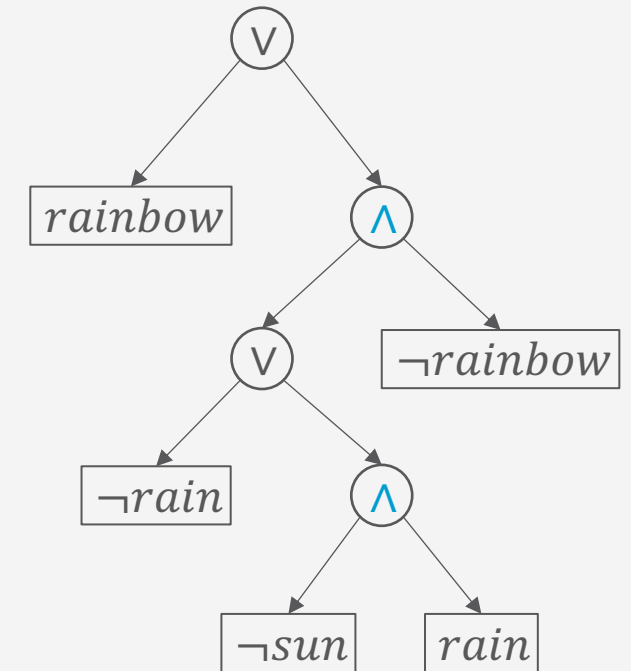
Effects of d-DNNF

- Effects of **d**-DNNF
 - **Deterministic** disjunctions
 - Only one disjunct (child node) can be true at the same time
 - I.e., their conjunction is unsatisfiable
 - Assume children c_i, c_j represent probabilities p_i, p_j
 - Node then represents probability of $P(c_i \vee c_j)$
 - $P(c_i \vee c_j) = P(c_i) + P(c_j) - P(c_i \wedge c_j)$
 - If only c_i or c_j can be true at a time, $P(c_i \wedge c_j) = 0$, i.e.,
 - $P(c_i \vee c_j) = P(c_i) + P(c_j)$
 - Can replace **$\vee$** with **$+$** for inference calculations



Effects of d-DNNF

- Effects of d-DNNF
 - **Decomposable** conjunctions
 - Each pair of conjuncts (child nodes) must be independent
 - I.e., they cannot share any variables
 - Assume children c_i, c_j represent probabilities p_i, p_j
 - Node then represents probability of $P(c_i \wedge c_j)$
 - If c_i and c_j independent (decomposable), then $P(c_i \wedge c_j) = P(c_i) \cdot P(c_j)$
 - Can replace $\wedge$ with $\cdot$ for inference calculations

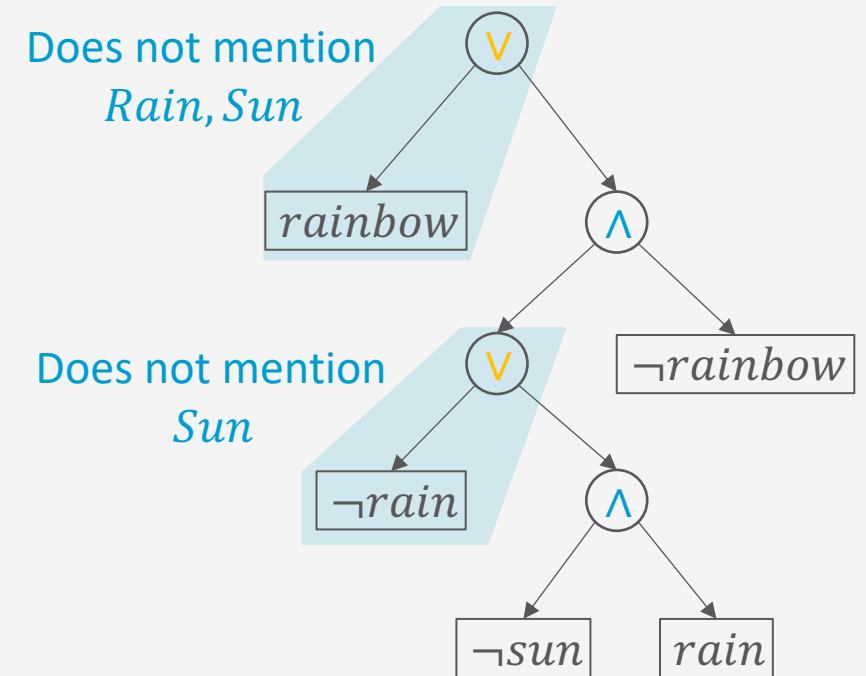


Smooth d-DNNF (sd-DNNF)

- Smooth circuits: constant runtime for certain queries
 - Any pair of disjuncts mentions the same set of variables
 - E.g., $\neg sun \vee \neg rain \vee rainbow$
 - Two disjunctions that do not fulfil the smoothness property
- Rules for conversion
 - For each negation of a positive literal l not appearing, replace l by

$$l \vee (\neg l \wedge false)$$
 - For each variable A not mentioned in a disjunct $\langle disjunct \rangle$, add $a \vee \neg a$ with a conjunction to $\langle disjunct \rangle$:

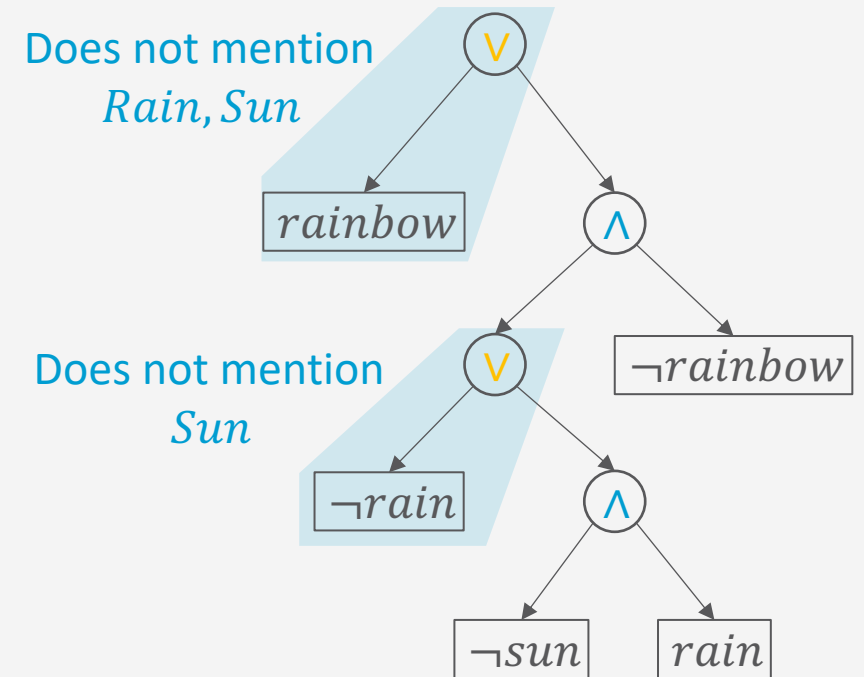
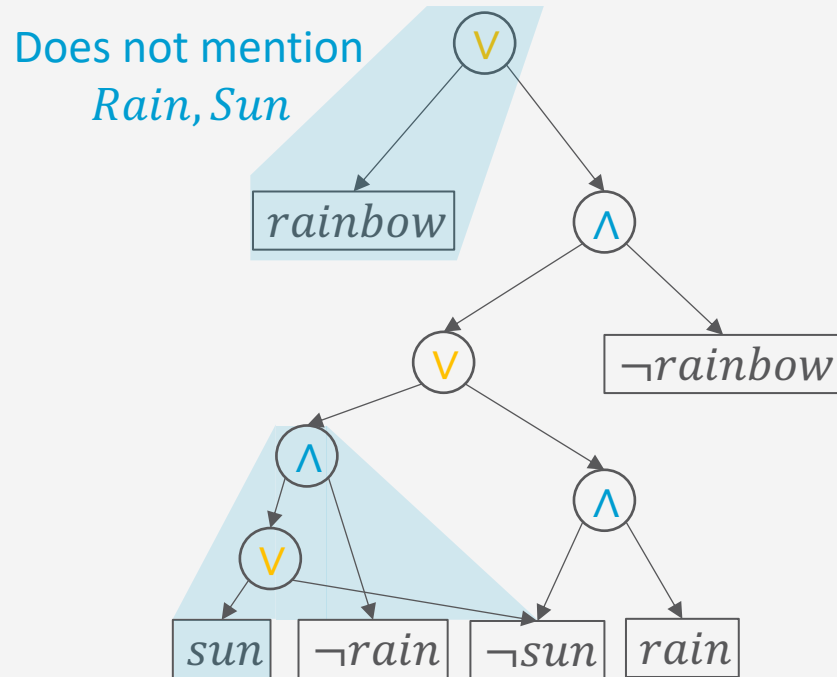
$$\langle disjunct \rangle \wedge (a \vee \neg a)$$



Smooth d-DNNF (sd-DNNF)

- Add $sun \vee \neg sun$ to $\neg rain$, replacing $\neg rain$ with

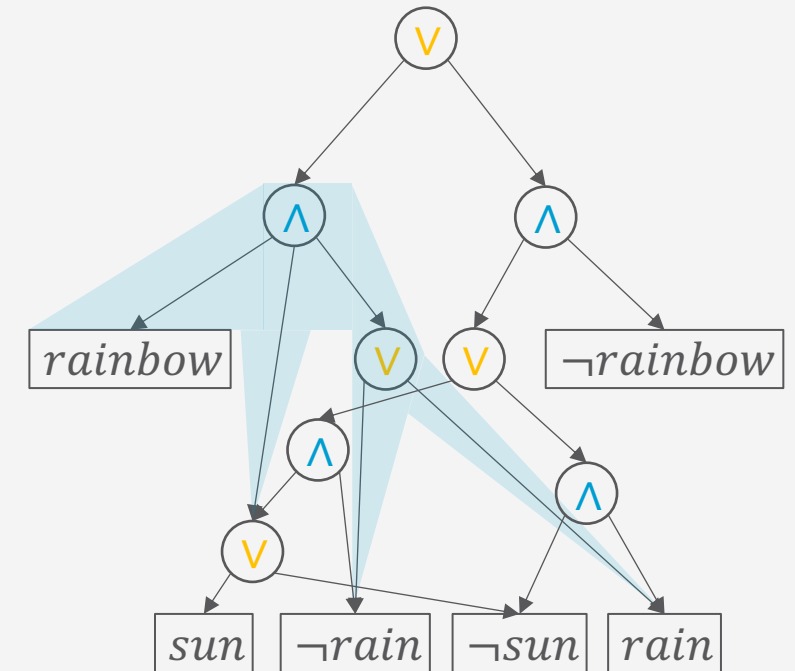
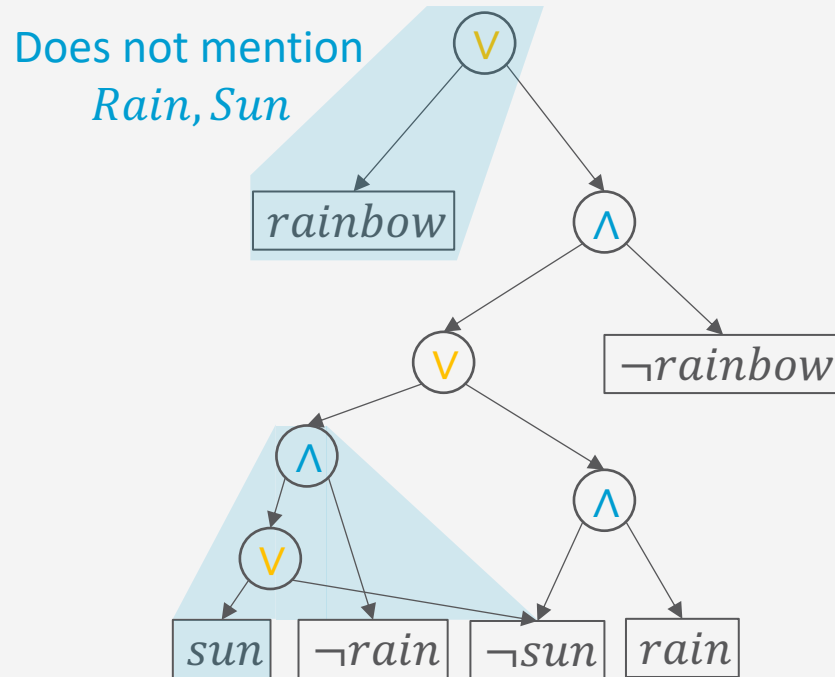
$$\neg rain \wedge (sun \vee \neg sun)$$



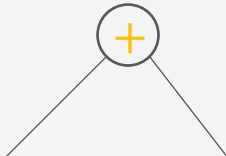
Smooth d-DNNF (sd-DNNF)

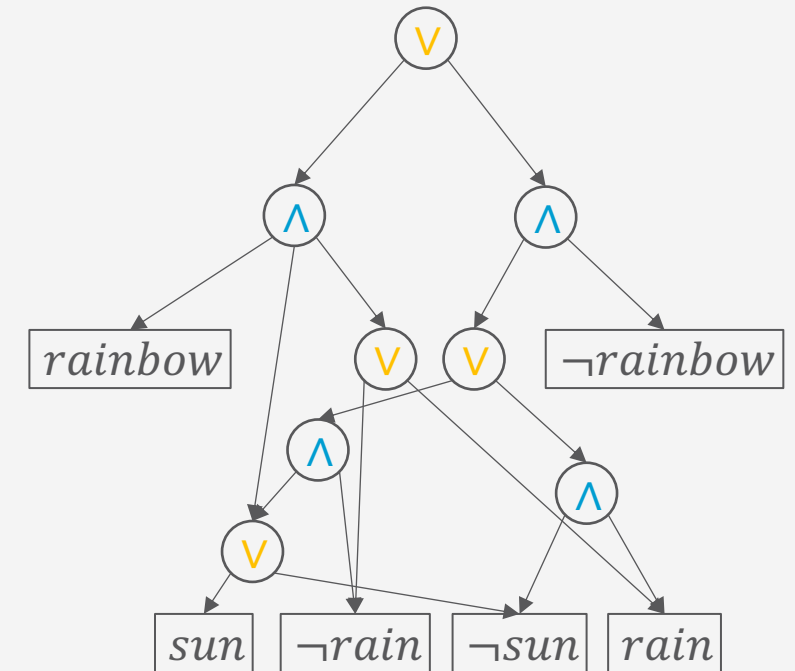
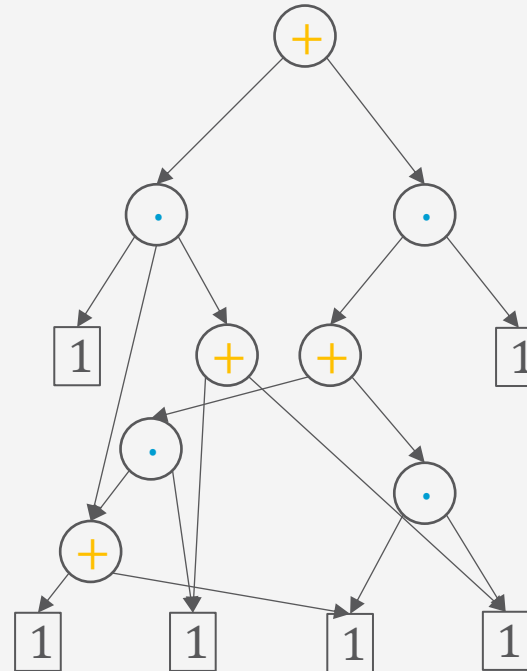
- Add $sun \vee \neg sun$ and $rain \vee \neg rain$, replacing *rainbow* with

$$rainbow \wedge (sun \vee \neg sun) \wedge (rain \vee \neg rain)$$



Circuit for Model Counting

- Model counting problem: Count how many models fulfil a sentence
 - Model counting arithmetic circuit
 - Replace $\wedge$ with $\cdot$
 - Replace $\vee$ with $+$
 - Replace leaves with 1's
- 

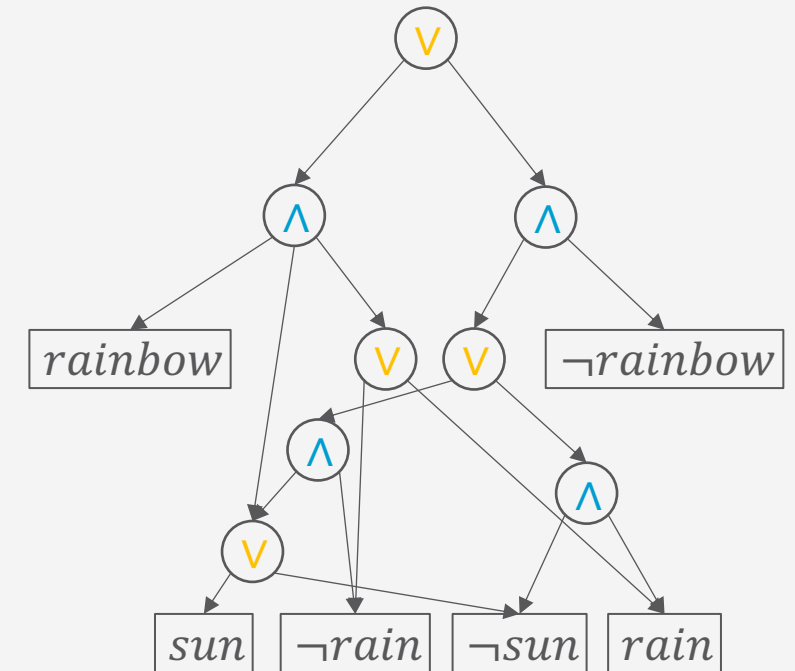
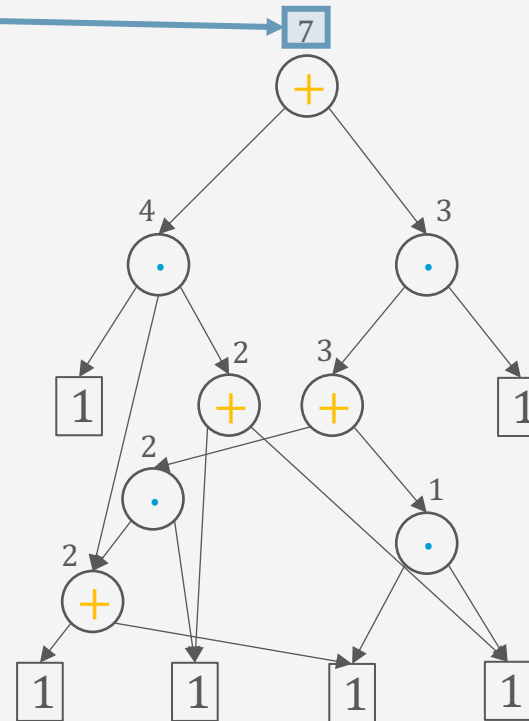


Circuit for Model Counting

- Propagate 1's upwards (from leaves to root), using arithmetic operations in inner nodes to combine incoming numbers
- Result at root: Model count

$sun \wedge rain \Rightarrow rainbow$

<i>rain</i>	<i>sun</i>	<i>rainbow</i>
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

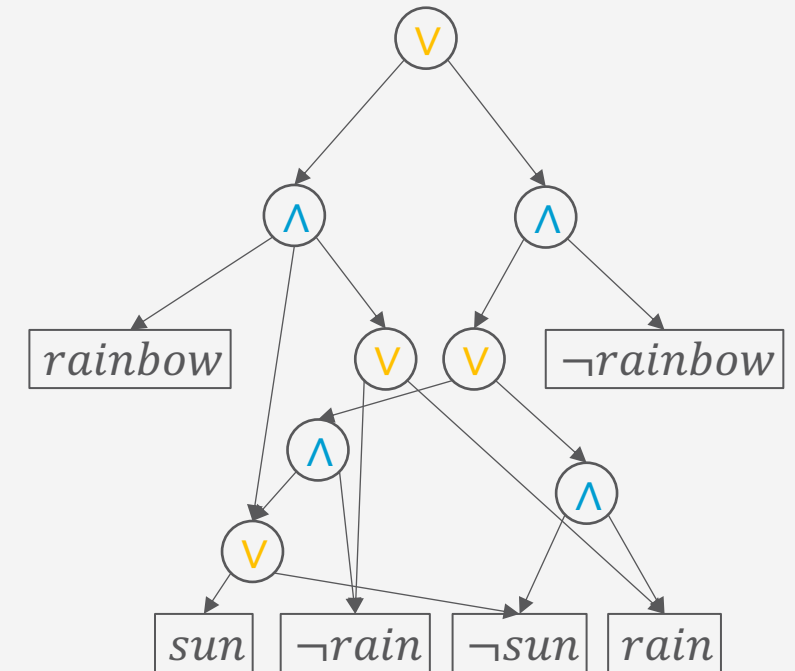
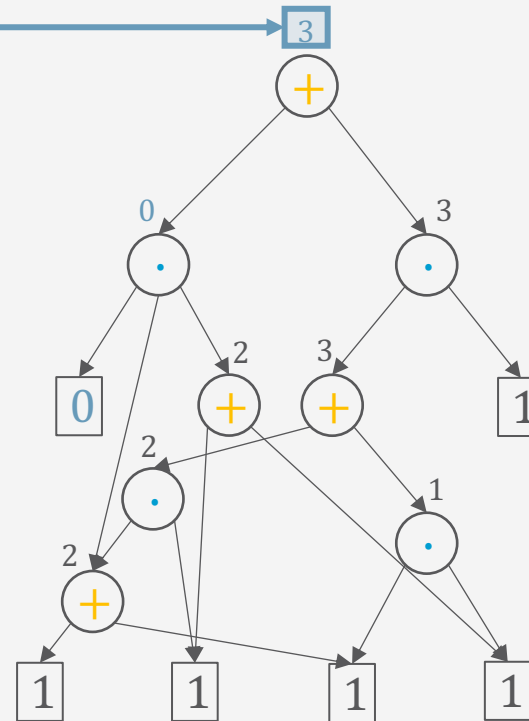


Conditioning

- To get model count of models fulfilling certain truth values
 - Replace 1's with zeros where literal contradicts truth values
 - E.g., condition on $\neg \text{rainbow}$ → 3

$\text{sun} \wedge \text{rain} \Rightarrow \text{rainbow}$

rain	sun	rainbow
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

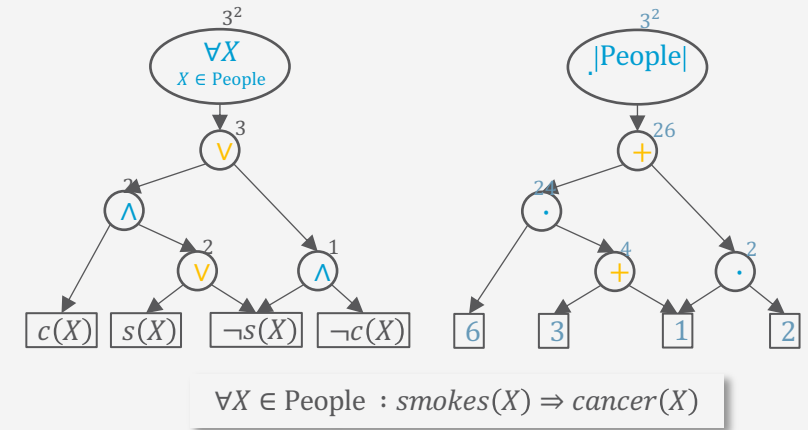


Interim Summary

- MC definition
- Circuits:
 - Rooted tree
 - Inner nodes: conjunctions / disjunctions → replaced by multiplication / addition
 - Leaves: literals, *true*, *false* → replaced by 1's and 0's
 - Properties: d-DNNF, smooth
 - Model counts by propagation from leaves to root
- Conditioning: replace literals with negated evidence with 0
- Can we use this idea to also answer queries in probabilistic models?

Knowledge Compilation

Weighted model counting problem, knowledge compilation algorithm



Local Symmetries and Structure

- Similar to MLNs, one can weight literals and specify a probability distribution in the same way
 - Allows for efficient encoding of distributions with symmetries
 - Example: $\phi(\text{Travel}(X), \text{Epid}, \text{Sick}(X))$ as given on the right
 - Only two weighted formulas (w, ψ) necessary
 - $(\ln 2, \neg \text{travel} \vee \neg \text{epid} \vee \neg \text{sick})$
 - $(\ln 7, \text{travel} \wedge \text{epid} \wedge \text{sick})$
 - If potential of 1 instead of 2, would reduce to
 - $(\ln 7, \text{travel} \wedge \text{epid} \wedge \text{sick})$
 - Assignments that do not make the formula true automatically get weight of 0 = $\ln 1$
- Extend MC definition to incorporate weights

<i>Travel</i>	<i>Epid</i>	<i>Sick</i>	ϕ
false	false	false	2
false	false	true	2
false	true	false	2
false	true	true	2
true	false	false	2
true	false	true	2
true	true	false	2
true	true	true	7

Weighted Model Counting

- Solve query answering problem by solving a [weighted model counting](#) problem
 - Weighted model count ([WMC](#)) given a sentence φ in propositional logic and a weight function $weight : L \rightarrow \mathbb{R}_{\geq 0}$ associating a non-negative weight to each literal in φ (set L) defined by

$$WMC(\varphi, weight) = \sum_{\omega \in \Omega_{\varphi}} \prod_{l \in \omega} weight(l)$$

- where Ω_{φ} refers to the set of worlds of φ
- Probability of a world ω of a sentence φ with weight function

$$P(\omega) = \frac{\prod_{l \in \omega} weight(l)}{WMC(\varphi, weight)} = \frac{WMC(\varphi \wedge \omega, weight)}{WMC(\varphi, weight)}$$

- A query for literal q given evidence e is solved by computing

$$P(q|e) = \frac{WMC(\varphi \wedge q \wedge e, weight)}{WMC(\varphi \wedge e, weight)}$$

Vgl. $P(Q|E) = \frac{P(Q,E)}{P(E)}$

Weighted Model Counting: Example

- Sentence
 - $sun \wedge rain \Rightarrow rainbow$
- Weight function:
 - $weight(sun) = 1$
 - $weight(\neg sun) = 5$
 - $weight(rain) = 2$
 - $weight(\neg rain) = 7$
 - $weight(rainbow) = 0.1$
 - $weight(\neg rainbow) = 10$

Each line a world $\omega \in \Omega_\varphi$

$$WMC(\varphi, weight) = \sum_{\omega \in \Omega_\varphi} \prod_{l \in \omega} weight(l)$$

<i>rain</i>	<i>sun</i>	<i>rainbow</i>	Weight	
0	0	0	$7 \cdot 5 \cdot 10$	350
0	0	1	$7 \cdot 5 \cdot 0.1$	3.5
0	1	0	$7 \cdot 1 \cdot 10$	70
0	1	1	$7 \cdot 1 \cdot 0.1$	0.7
1	0	0	$2 \cdot 5 \cdot 10$	100
1	0	1	$2 \cdot 5 \cdot 0.1$	1
~~1~~	~~1~~	~~0~~	~~$2 \cdot 1 \cdot 10$~~	~~20~~
1	1	1	$2 \cdot 1 \cdot 0.1$	0.2
			+	525.4

Weighted Model Counting: Example

- Sentence
 - $sun \wedge rain \Rightarrow rainbow$
- Weight function:
 - $weight(sun) = 1$
 - $weight(\neg sun) = 5$
 - $weight(rain) = 2$
 - $weight(\neg rain) = 7$
 - $weight(rainbow) = 0.1$
 - $weight(\neg rainbow) = 10$
- Probability of worlds:

$$P(sun, rain, rainbow) = \frac{0.2}{525.4} = 0.00038$$

$$P(\omega) = \frac{\prod_{l \in \omega} weight(l)}{WMC(\varphi, weight)} = \frac{WMC(\varphi \wedge \omega, weight)}{WMC(\varphi, weight)}$$

$$(sun \wedge rain \Rightarrow rainbow) \wedge sun \wedge rain \wedge rainbow$$

$$\omega = (sun, rain, rainbow) \in \Omega_{\varphi}$$

rain	sun	rainbow	Weight	
0	0	0	$7 \cdot 5 \cdot 10$	350
0	0	1	$7 \cdot 5 \cdot 0.1$	3.5
0	1	0	$7 \cdot 1 \cdot 10$	70
0	1	1	$7 \cdot 1 \cdot 0.1$	0.7
1	0	0	$2 \cdot 5 \cdot 10$	100
1	0	1	$2 \cdot 5 \cdot 0.1$	1
1	1	0	$2 \cdot 1 \cdot 10$	20
1	1	1	$2 \cdot 1 \cdot 0.1$	0.2
			+	525.4

Weighted Model Counting: Example

- Sentence
 - $sun \wedge rain \Rightarrow rainbow$
- Weight function:
 - $weight(sun) = 1$
 - $weight(\neg sun) = 5$
 - $weight(rain) = 2$
 - $weight(\neg rain) = 7$
 - $weight(rainbow) = 0.1$
 - $weight(\neg rainbow) = 10$
- Probability of worlds:
 - $P(rain) = \frac{100+1+0.2}{525.4} = 0.1926$

All $\omega \in \Omega_\varphi$ where $rain$ holds

$$P(q) = \frac{WMC(\varphi \wedge q, weight)}{WMC(\varphi, weight)}$$

$$(sun \wedge rain \Rightarrow rainbow) \wedge rain$$

<i>rain</i>	<i>sun</i>	<i>rainbow</i>	Weight	
0	0	0	$7 \cdot 5 \cdot 10$	350
0	0	1	$7 \cdot 5 \cdot 0.1$	3.5
0	1	0	$7 \cdot 1 \cdot 10$	70
0	1	1	$7 \cdot 1 \cdot 0.1$	0.7
1	0	0	$2 \cdot 5 \cdot 10$	100
1	0	1	$2 \cdot 5 \cdot 0.1$	1
1	1	0	$2 \cdot 1 \cdot 10$	20
1	1	1	$2 \cdot 1 \cdot 0.1$	0.2
			+	525.4

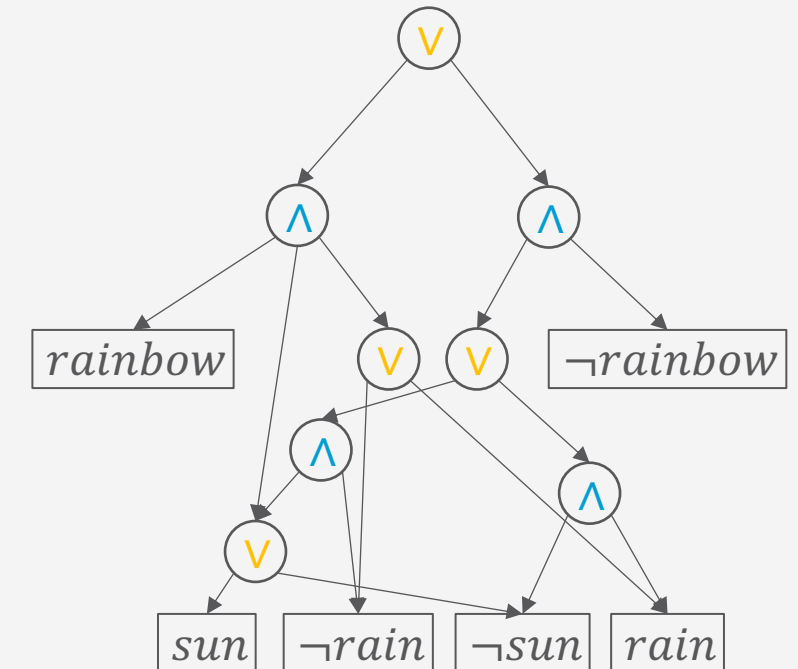
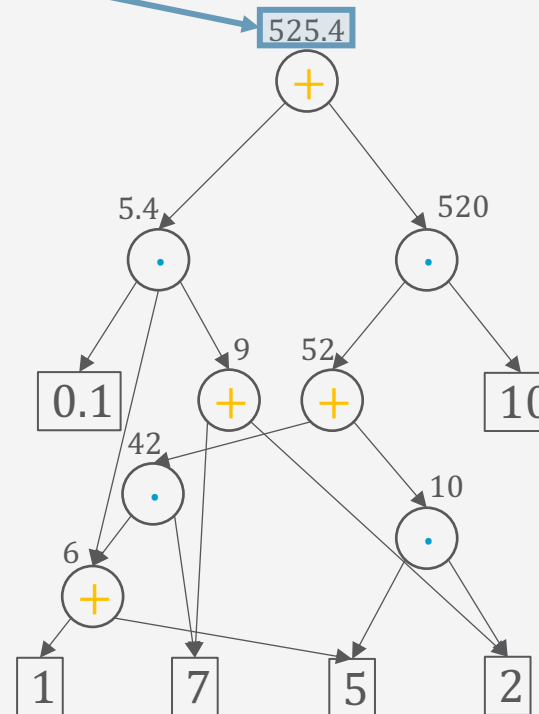
WMC and Inference

- As with model counting, solving a WMC problem for a sentence φ as introduced on previous slides is exponential in number of constants with probability > 0
- To be more efficient, build the same helper structure, circuits, but put weights into leaves
 - Again, bring sentence into negation normal form (NNF)
 - NNF: Formulas contain only negations directly in front of variables, conjunctions, and disjunctions
 - E.g.,
 - $sun \wedge rain \Rightarrow rainbow$ (Apply $A \Rightarrow B \equiv \neg A \vee B$)
 $\equiv \neg(sun \wedge rain) \vee rainbow$ (Apply De Morgan's law)
 $\equiv \neg sun \vee \neg rain \vee rainbow$ (NNF)

Circuit for Weighted Model Counting

- Replace literals with weights in leaves and propagate weights upwards
- Computes $WMC(\varphi, weight)$

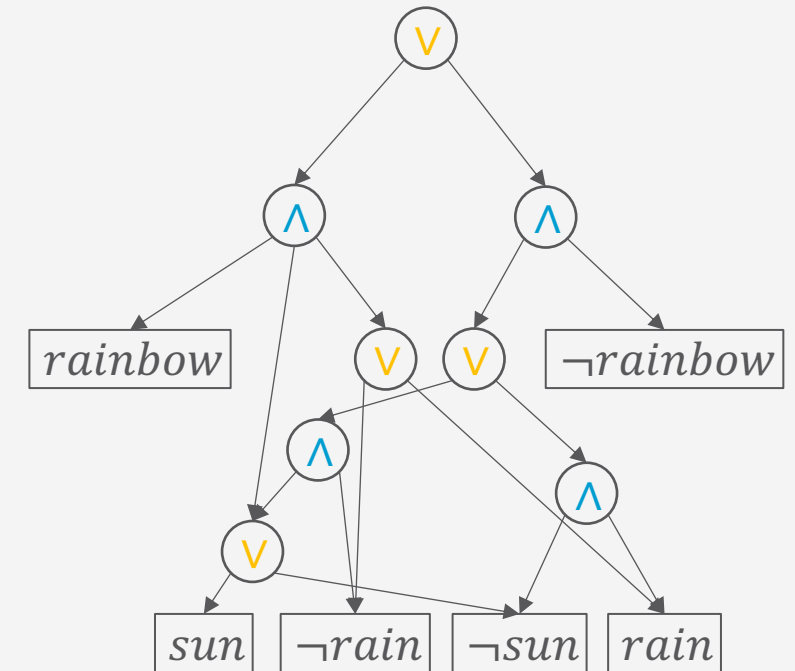
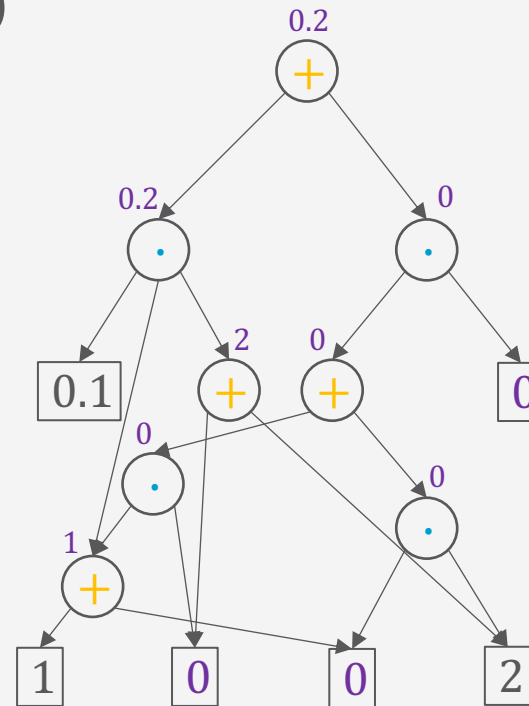
$weight(sun) = 1$
 $weight(\neg sun) = 5$
 $weight(rain) = 2$
 $weight(\neg rain) = 7$
 $weight(rainbow) = 0.1$
 $weight(\neg rainbow) = 10$



Circuit for Weighted Model Counting

- For probabilities of worlds or query terms ω , condition on truth values
 - Compute $WMC(\varphi, weight)$ ← Reuse for different queries
 - Compute $WMC(\varphi \wedge \omega, weight)$
 - Divide the two counts

$$\begin{aligned}
 &P(\omega = \{sun, rain, rainbow\}) \\
 &= \frac{WMC(\varphi \wedge \omega, weight)}{WMC(\varphi, weight)} \\
 &= \frac{0.2}{525.4} = 0.00038
 \end{aligned}$$



Knowledge Compilation

- Given a theory Δ and a set of queries $\{P(q_i|e)\}_{i=1}^m$
 - Build a circuit for theory Δ (a conjunction of sentences)
 - Make the circuit a WMC circuit
 - Replace inner nodes with arithmetic operations and leaves with weights
 - Condition on given evidence e
 - Replace weights with 0 where literals contradict e
 - Calculate $WMC(\Delta \wedge e, weight)$ in the circuit
 - By propagating the weights upwards
 - For each query $P(q_i|e)$ in the circuit
 - Compute $WMC(\Delta \wedge e \wedge q_i, weight)$
 - Return or store $P(q_i|e) = \frac{WMC(\Delta \wedge e \wedge q_i, weight)}{WMC(\Delta \wedge e, weight)}$

Knowledge Compilation

Interim Summary

- WMC definition
- Circuits:
 - Rooted tree (as before)
 - Inner nodes: conjunctions / disjunctions → replaced by multiplication / addition (as before)
 - Leaves: literals, *true*, *false* → replaced by weights, 1's, and 0's
 - Properties: d-DNNF, smooth (as before)
 - WMC by propagation from leaves to root
- Conditioning: replace literals with negated evidence with 0
- Knowledge compilation: Inference in circuits, i.e., query answering by weighted model counting in circuits
- Side note: there is a whole landscape on so called algebraic circuits by Darwiche
[Seminal paper: A Knowledge Compilation Map by Darwiche and Marquis. In *JAIR*, 2002]

Outline: 3. Inference

a. Exact Inference

i. (Weighted) Model Counting

- Model counting
- Weighted model counting, knowledge compilation (KC)

ii. Variable Elimination and the Like

- Variable elimination (VE)
- Junction tree algorithm (JT)

iii. Lifted Inference

- First-order KC, Lifted VE, lifted JT

b. Approximate Inference

- Direct sampling: Rejection sampling, (lifted) importance sampling
- (Lifted) Markov Chain Monte Carlo sampling