Introduction to Neurosymbolic AI
Day 5: Probabilistic Reasoning

Emile van Krieken
Vrije Universiteit Amsterdam

ESSLLI 2025
8th of August

VU VRIJE UNIVERSITEIT AMSTERDAM

This course

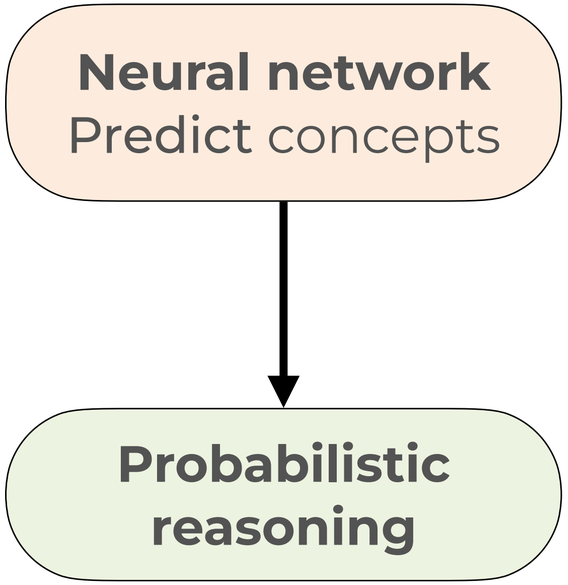
Day 1: Introduction to NeSy

Day 2: Learnability

Day 3: Learning Imbalances in NeSy

Day 4: Reasoning Shortcuts

Day 5: Probabilistic Reasoning



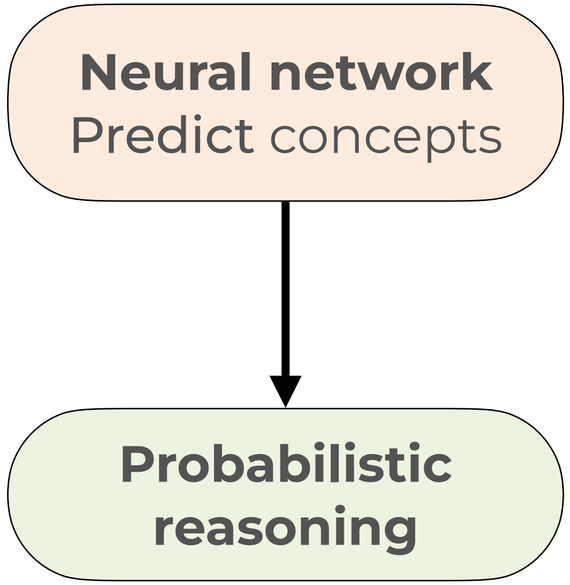
Content

Probabilistic Neurosymbolic methods

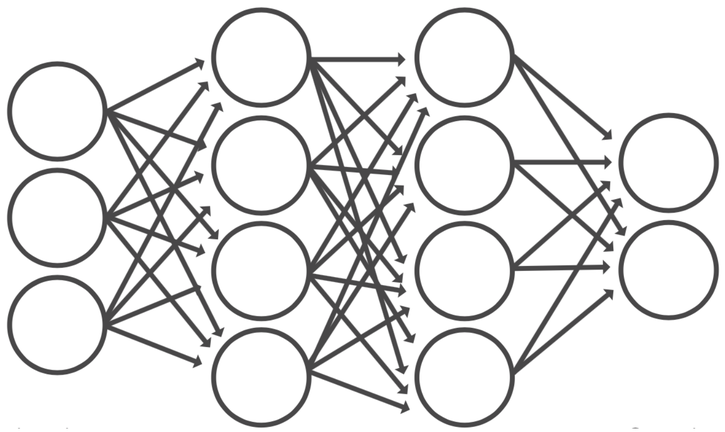
- Semantic loss
- Semantic probabilistic layers
- Neurosymbolic predictors
- Why probabilistic reasoning?

Inference

- Circuits & exact inference
- Approximate inference



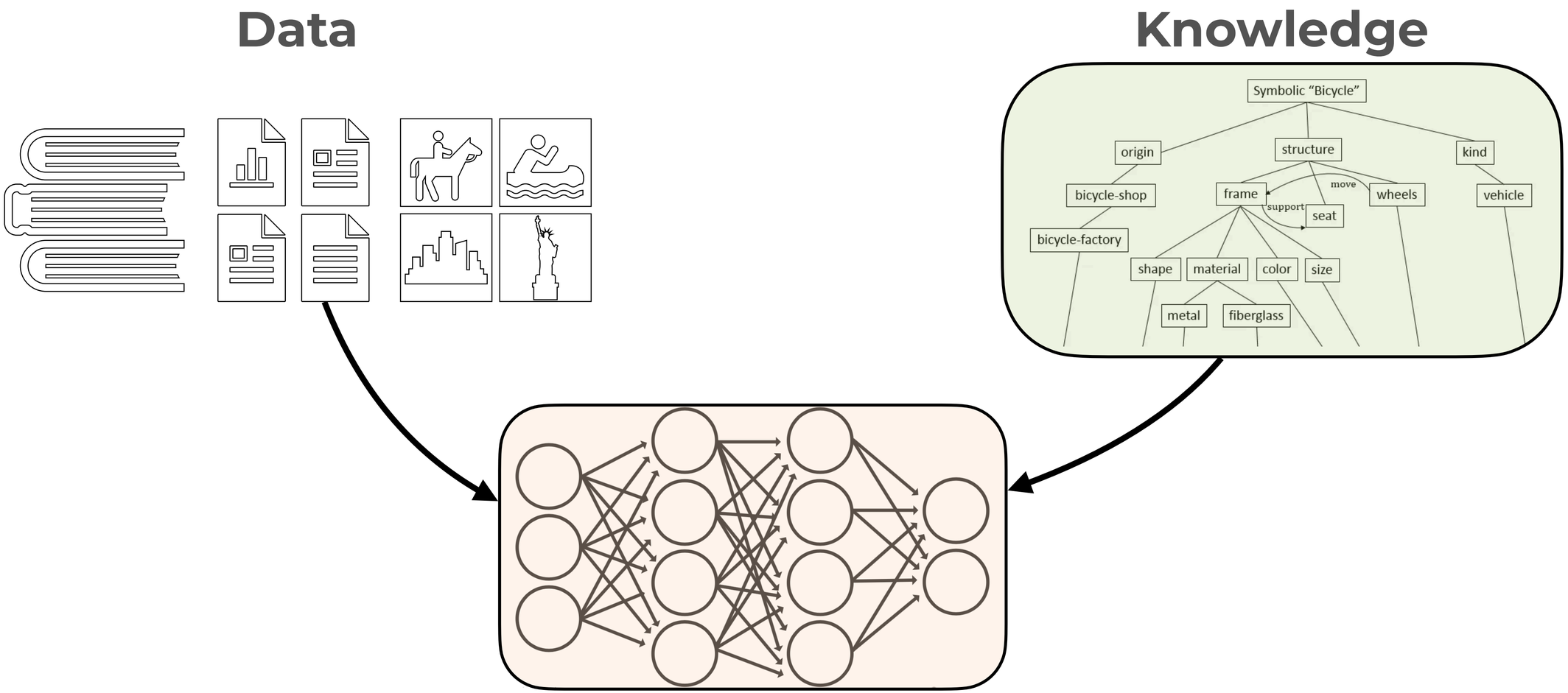
Illegal prediction!



Neural network:

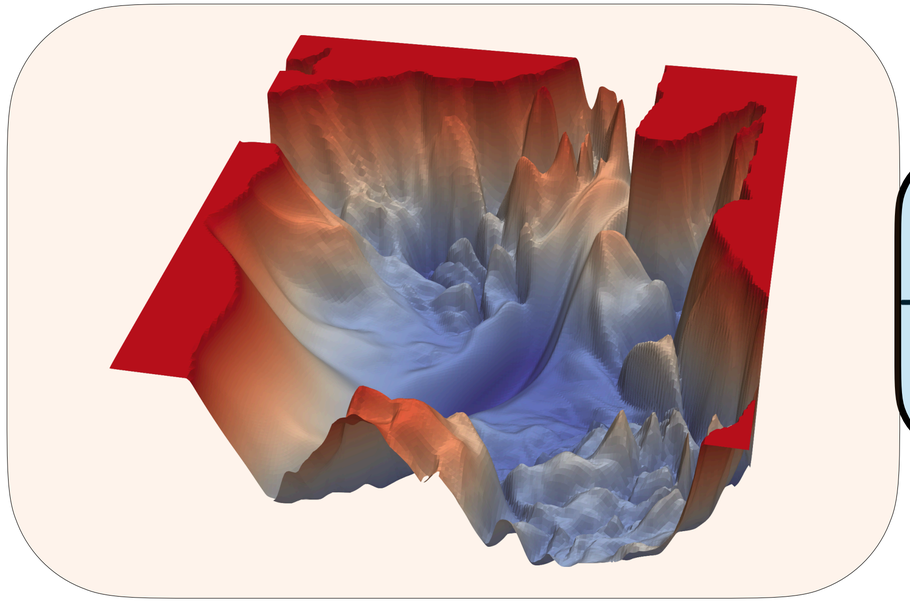
A traffic light that is red and green at the same time!

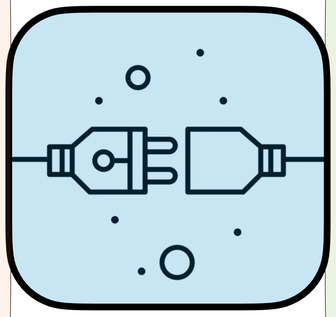
Giunchiglia, E., Stoian, M. C., Khan, S., Cuzzolin, F., & Lukasiewicz, T. (2022). Road-r: The autonomous driving dataset with logical requirements. arXiv preprint arXiv:2210.01597.

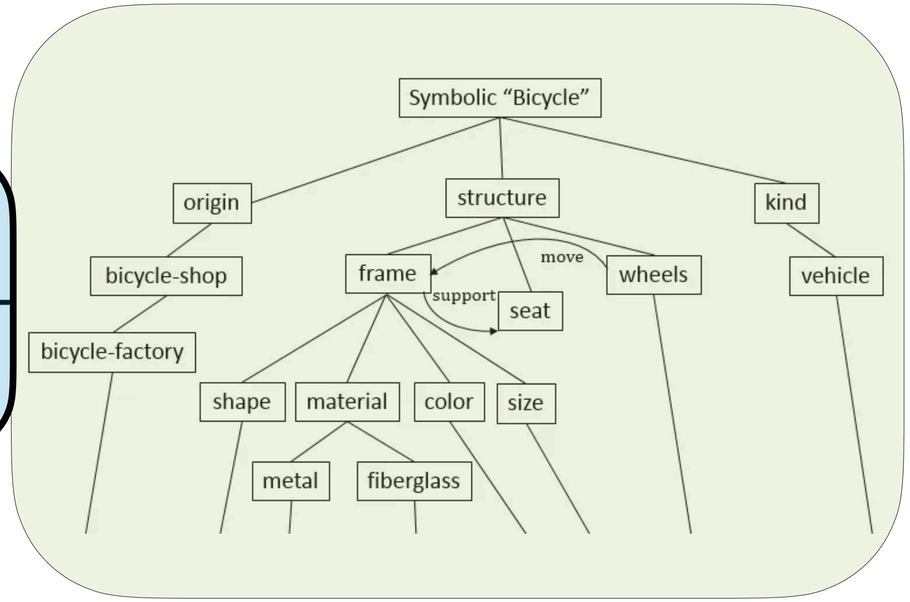

Data
Knowledge
Symbolic “Bicycle”
origin
structure
kind
bicycle-shop
frame
move
wheels
vehicle
support
seat
bicycle-factory
shape
material
color
size
metal
fiberglass

Neurosymbolic AI

How to build reliable AI methods that integrate knowledge and constraints?







Neural networks:
Continuous

“The plug”

Symbolic AI:
Discrete

It’s probability :-)

High-level symbolic **concepts**

$$\{r, g\}$$

r : Red traffic light on
g : Green traffic light on

Program φ is **Boolean formula** on concepts

$$\varphi = \neg r \vee \neg g$$

$$\{(\neg r, g), (r, \neg g), (\neg r, \neg g)\}$$

green on red on Both off

High-level symbolic **concepts**

$$\{r, g\}$$

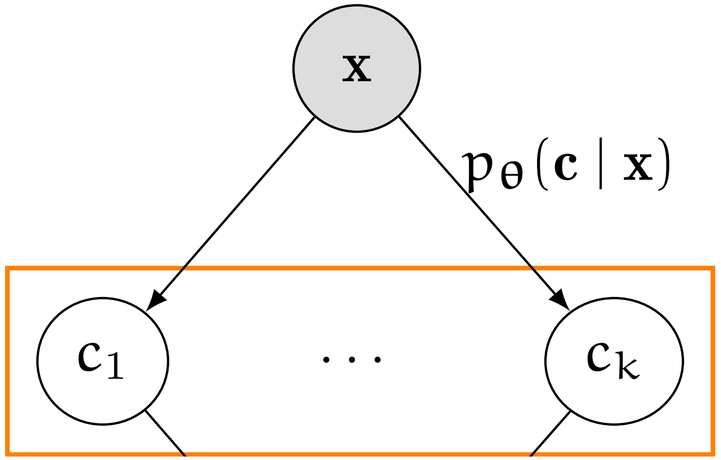
Program φ is **Boolean formula** on concepts

$$\varphi = \neg r \vee \neg g$$

How to prevent neural network from violating constraint?

Probabilistic Reasoning setting 1:
Semantic loss

Boring notation slide



$\mathbf{x} \in \mathcal{X}$ High-dim input, eg image

$\mathbf{c} \in \mathcal{C}$ Concepts predicted by neural network

$\varphi : \mathcal{C} \to \{0, 1\}$ Known constraint

$p_\theta(\mathbf{c} \mid \mathbf{x})$ Probability neural network assigns to $\mathbf{c}$ given input $\mathbf{x}$

θ Neural network parameters

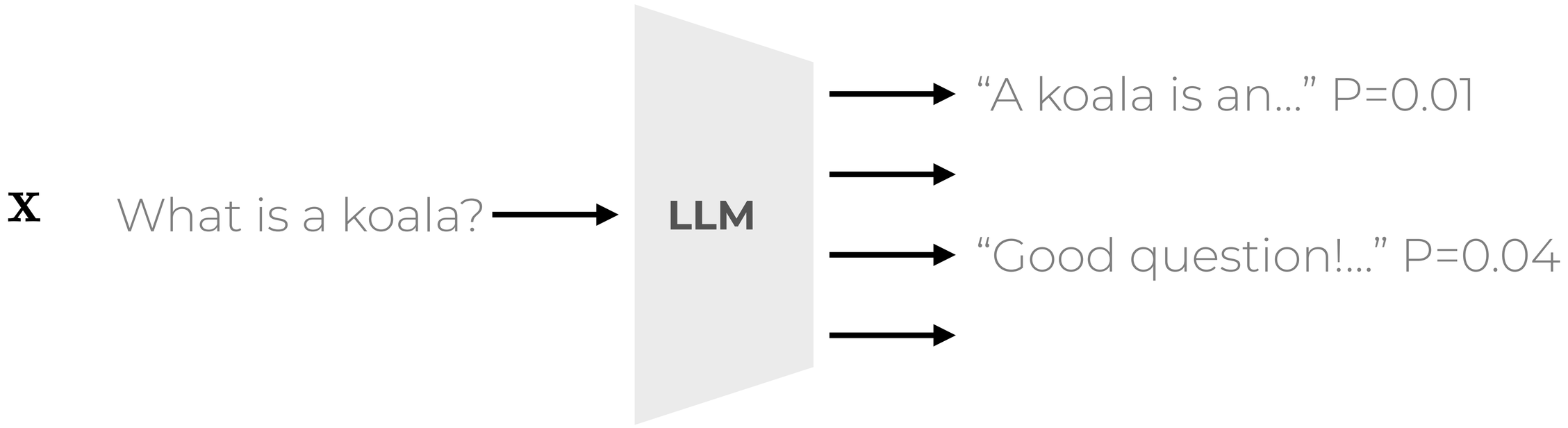
$\mathbf{x}$

Neural network

$$p_\theta(\neg r, \neg g \mid \mathbf{x})$$
$$p_\theta(r, \neg g \mid \mathbf{x})$$
$$p_\theta(\neg r, g \mid \mathbf{x})$$
$$p_\theta(r, g \mid \mathbf{x})$$

Neural networks produce a probability for **each possible concept vector** $\mathbf{c}$. These probabilities **sum to 1**.

Each probability can be seen as a **strength of belief** into different answer responses



Generative neural networks like **Large Language Models (LLMs)** are also probabilistic models!

The methods explained today apply directly to LLMs

Semantic Loss

Neural network
Extract concepts

$$p_\theta(\mathbf{c} \mid \mathbf{x})$$

$$\varphi = \neg r \vee \neg g$$

Semantic loss
Use probabilistic reasoning

Compute loss that
encourages formula

$$-\log \mathbb{P}_\theta[\varphi \text{ is true} \mid \mathbf{x}]$$

Xu, J., Zhang, Z., Friedman, T., Liang, Y., & Broeck, G. (2018, July). A semantic loss function for deep learning with symbolic knowledge. In International conference on machine learning (pp. 5502-5511). PMLR.

$$\mathbb{P}_\theta[\varphi \text{ is true} \mid \mathbf{x}]$$

Probabilistic reasoning

$$\varphi(r, g) = \neg r \vee \neg g$$

$$\mathbb{P}_\theta[\neg r \vee \neg g \mid \mathbf{x}] = p_\theta(r, \neg g \mid \mathbf{x}) + p_\theta(\neg r, g \mid \mathbf{x}) + p_\theta(\neg r, \neg g \mid \mathbf{x})$$

red on green on Both off

Weighted Model Count (WMC)

$$\mathbb{P}_\theta[\varphi \text{ is true} \mid \mathbf{x}] = \sum_{\mathbf{c} \in \mathcal{C}} p_\theta(\mathbf{c} \mid \mathbf{x}) \varphi(\mathbf{c})$$

Neural preference

Constraint

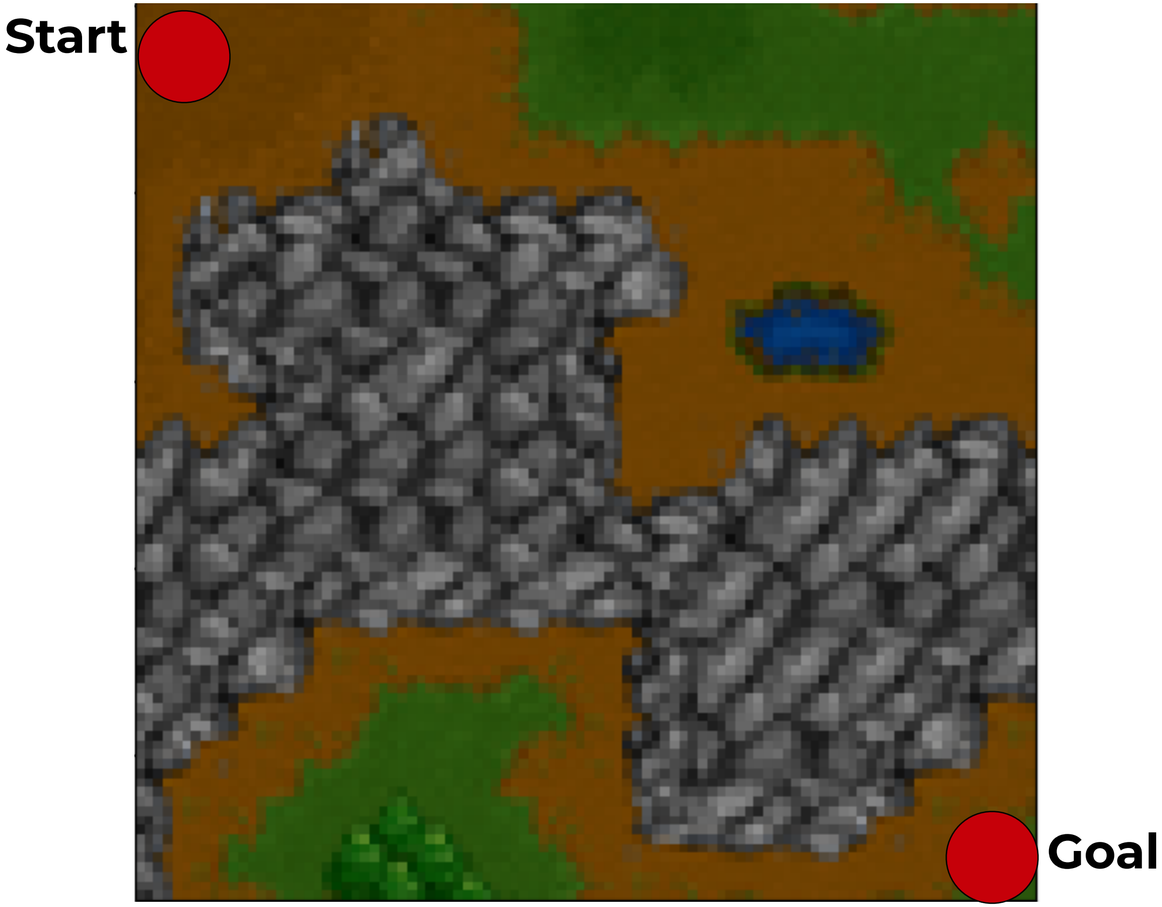
WMC is **differentiable**!

Train neural nets with WMC of constraint via **Semantic Loss**

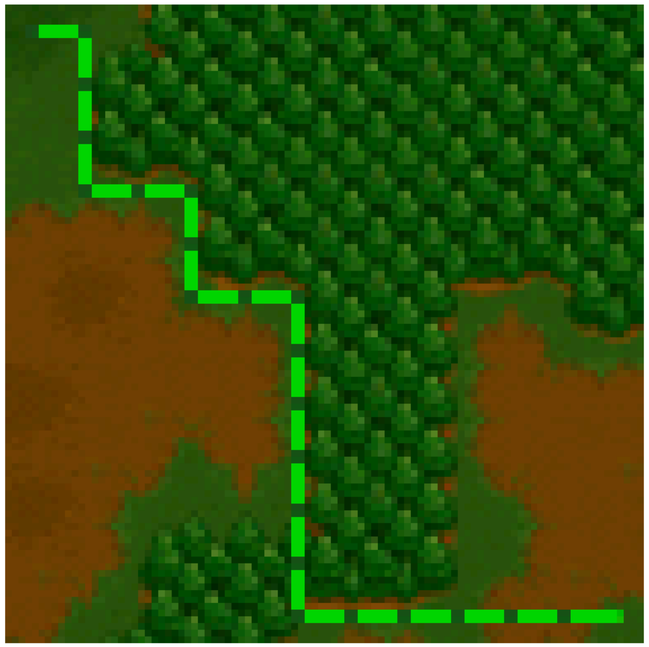
$$\mathcal{L}_{\text{SL}}(\mathbf{x}) = -\log \mathbb{P}_\theta[\varphi \text{ is true} \mid \mathbf{x}]$$

Often used in **semi-supervised settings** where only some inputs are labelled: Data efficient!

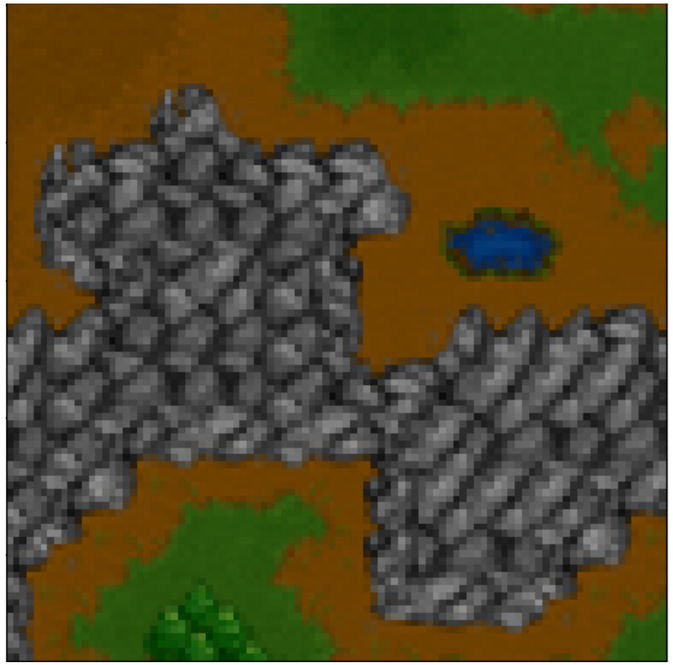
Visual path planning



Semi-supervised learning



Labelled example
With shortest path



Unlabelled example
Knowledge: **the output should be a valid path**!

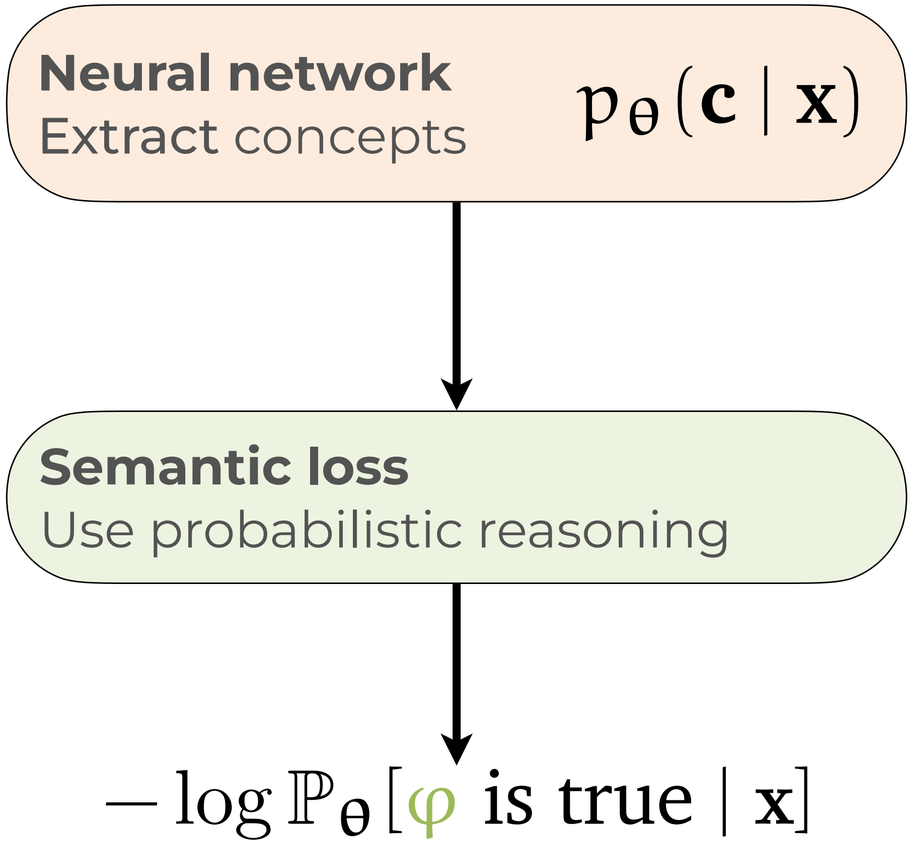
Semi-supervised learning algorithm

Given: Labelled examples,
unlabelled examples

Supervised learning
Compute NN outputs
Compute **supervised loss** with given label

Unsupervised learning
Compute NN outputs
Compute **semantic loss**

Compute gradient of both losses
Backpropagate



Semantic loss - Recap

Train neural nets with WMC of formula via the **semantic loss**

Use on **unlabelled** data for semi-supervised learning

Semantic loss **only encourages** satisfying the formula...
but does not guarantee this!

Neural network
Extract concepts

$$p_\theta(\mathbf{c} \mid \mathbf{x})$$

↓

Semantic loss
Use probabilistic reasoning

↓

$$-\log \mathbb{P}_\theta[\varphi \text{ is true} \mid \mathbf{x}]$$

Probabilistic Reasoning setting 2:
Semantic probabilistic layers

Semantic Probabilistic Layers (SPL)

Neural network
Extract concepts

$$p_\theta(\mathbf{c} \mid \mathbf{x})$$

$$\varphi(\mathbf{c}) = \neg r \vee \neg g$$

Semantic
probabilistic layer

$$p_{\theta,\varphi}(\mathbf{c} \mid \mathbf{x})$$

Guarantees constraint is satisfied!

$$\arg\max_{\mathbf{c}} p_{\theta,\varphi}(\mathbf{c} \mid \mathbf{x})$$

Ahmed, K., Teso, S., Chang, K. W., Van den Broeck, G., & Vergari, A. (2022). Semantic probabilistic layers for neuro-symbolic learning. Advances in Neural Information Processing Systems, 35, 29944-29959.

Semantic Probabilistic Layers:

$$p_{\theta,\varphi}(\mathbf{c} \mid \mathbf{x}) := \frac{p_\theta(\mathbf{c} \mid \mathbf{x}) \cdot \varphi(\mathbf{c})}{Z_\theta(\mathbf{x})}$$

Set all concepts that do not satisfy constraint to 0

Then, **renormalise**.

The **normaliser** is exactly the WMC!

$$Z_\theta(\mathbf{x}) = \mathbb{P}_\theta[\varphi \text{ is true} \mid \mathbf{x}]$$

Semantic Probabilistic Layers (SPLs):

$$p_{\theta,\varphi}(\mathbf{c} \mid \mathbf{x}) := \frac{p_\theta(\mathbf{c} \mid \mathbf{x}) \cdot \varphi(\mathbf{c})}{Z_\theta(\mathbf{x})}$$

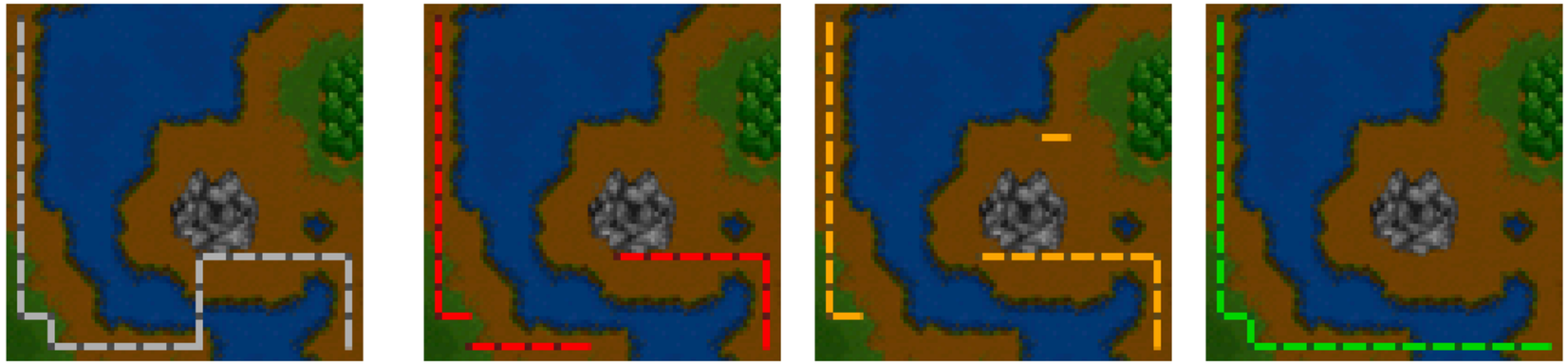
Set all concepts that do not satisfy constraint to 0 probability!
Then, **renormalise**.

SPLs is trained in **supervised** setting given pairs $(\mathbf{x}, \mathbf{c})$:

$$\mathcal{L}(\mathbf{x}, \mathbf{c}) = -\log p_{\theta,\varphi}(\mathbf{c} \mid \mathbf{x})$$

Predictions are found with

$$\arg\max_{\mathbf{c} \in \mathcal{C}} p_{\theta,\varphi}(\mathbf{c} \mid \mathbf{x})$$



GROUND TRUTH FIL $\mathcal{L}_{SL}$ SPL

Knowledge: the output should be a valid path

ARCHITECTURE	EXACT	HAMMING	CONSISTENT
RESNET-18+FIL	55.0	**97.7**	56.9
RESNET-18+$\mathcal{L}_{SL}$	59.4	**97.7**	61.2
RESNET-18+SPL	**78.2**	96.3	**100.0**

Ahmed, K., Teso, S., Chang, K. W., Van den Broeck, G., & Vergari, A. (2022). Semantic probabilistic layers for neuro-symbolic learning. Advances in Neural Information Processing Systems, 35, 29944-29959.

Semantic Probabilistic Layer

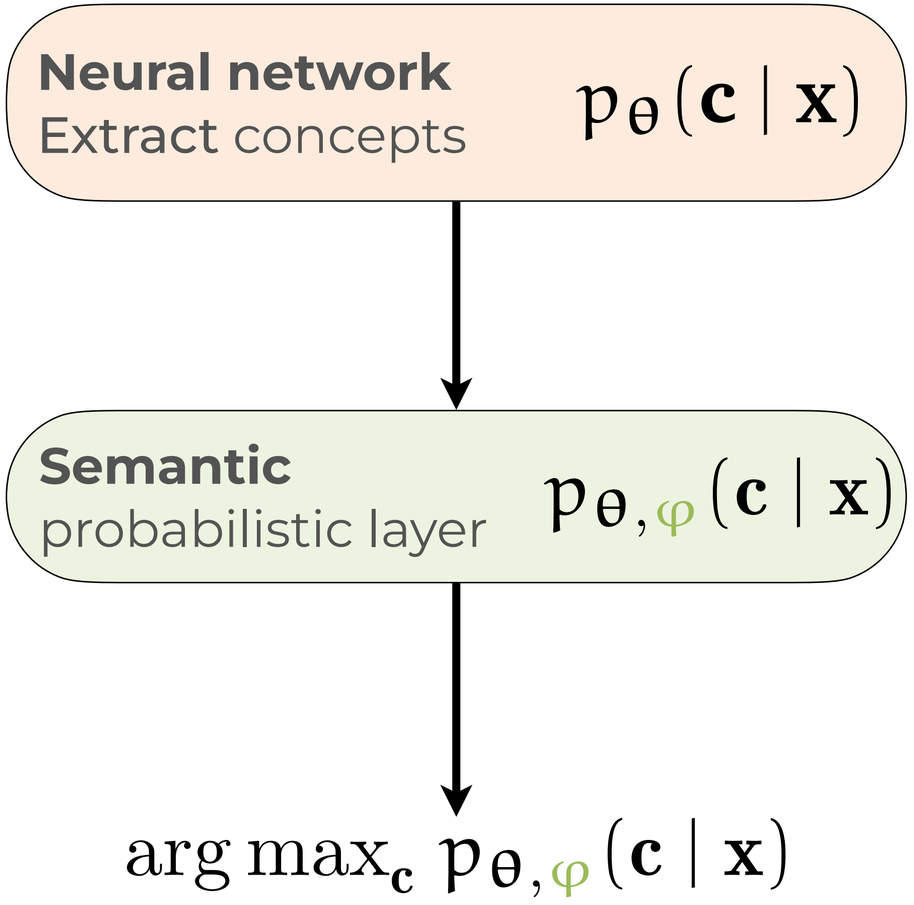
SPL requires WMC in normaliser

$$\mathbb{P}_\theta[\varphi \text{ is true} \mid \mathbf{x}]$$

And predictions require computing

$$\arg\max_{\mathbf{c} \in \mathcal{C}} p_{\theta,\varphi}(\mathbf{c} \mid \mathbf{x})$$

Both are **hard to compute**!



Enforcing Boolean constraints

Semantic Loss

- **No guarantees** constraint is satisfied at test-time
- Can be used in **semi-supervised** settings
- **No overhead** at test-time

Semantic probabilistic layers

- **Guarantees** constraint satisfied at test-time
- Can only be trained in supervised settings
- Requires **computing most likely concept vector** at test-time

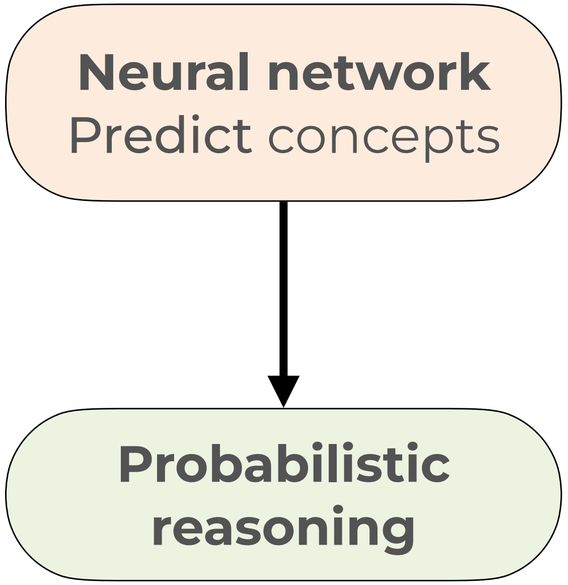
Content

Probabilistic Neurosymbolic methods

- Semantic loss
- Semantic probabilistic layers
- Neurosymbolic predictors
- Why probabilistic reasoning?

Inference

- Circuits & exact inference
- Approximate inference

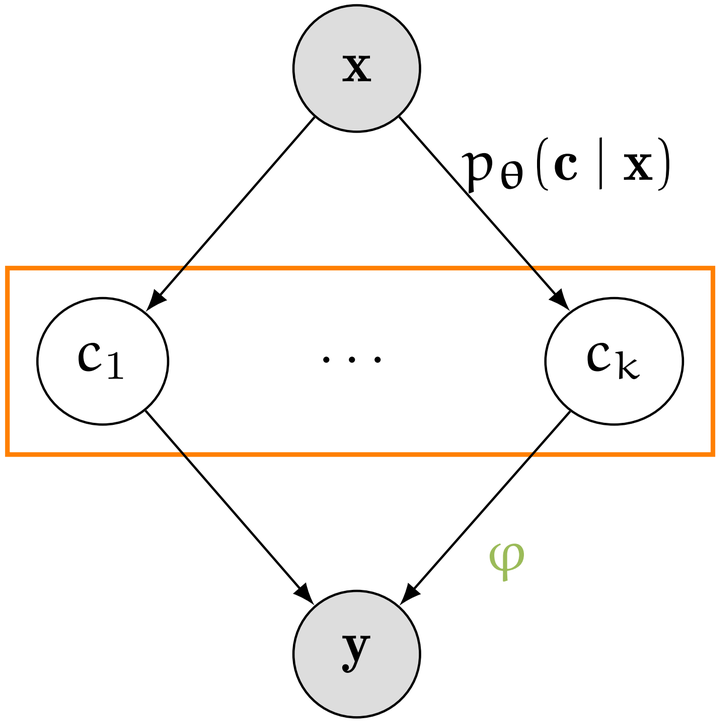


Neurosymbolic AI:
guarantee formula satisfaction ✅

What else?

Probabilistic Reasoning setting 3: Neurosymbolic Predictors

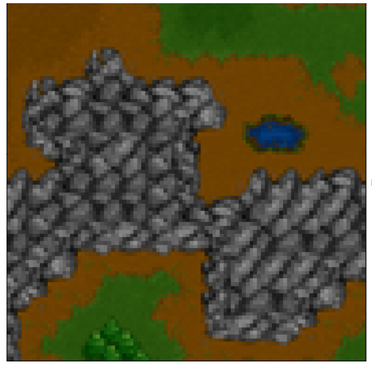
Boring notation slide



$\mathbf{x} \in \mathcal{X}$	High-dim input, eg image
$\mathbf{c} \in \mathcal{C}$	Concepts predicted by neural network
$\mathbf{c}^* \in \mathcal{C}$	Ground-truth concepts
$\mathbf{y} \in \mathcal{Y}$	Output labels
$\varphi : \mathcal{C} \to \mathcal{Y}$	Known program
$p_\theta(\mathbf{c} \mid \mathbf{x})$	Probability neural network assigns to $\mathbf{c}$ given input $\mathbf{x}$
θ	Neural network parameters
$p_\theta(\mathbf{y} \mid \mathbf{x})$	Predict label $\mathbf{y}$ given input $\mathbf{x}$

$$p_\theta(\mathbf{y} \mid \mathbf{x}) := \mathbb{P}_\theta[\varphi(\mathbf{c}) = \mathbf{y} \mid \mathbf{x}] = \sum_{\mathbf{c} \in \mathcal{C}} p_\theta(\mathbf{c} \mid \mathbf{x}) \cdot \mathbb{1}[\varphi(\mathbf{c}) = \mathbf{y}]$$

Neurosymbolic Predictors



Neural network
Extract concepts

$p_\theta(\mathbf{c} \mid \mathbf{x})$

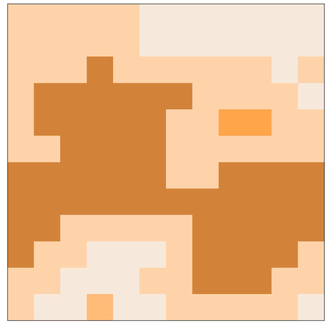
Unknown **costs**

grass sand rock

Program
(Shortest path search)

Probabilistic reasoning using program

Path



Neurosymbolic Predictors

Consider the setting discussed in days 2-4

Problem: Given

- Pairs of inputs and labels $(\mathbf{x}, \mathbf{y})$
- A program $\varphi : \mathcal{C} \to \mathcal{Y}$
- A neural network over concepts $p_\theta(\mathbf{c} \mid \mathbf{x})$
- No direct supervision over concept vectors $\mathbf{c}$

Learn input-output mapping!

Neurosymbolic Predictors

$$p_\theta(\mathbf{y} \mid \mathbf{x}) := \mathbb{P}_\theta[\varphi(\mathbf{c}) = \mathbf{y} \mid \mathbf{x}]$$

Belief in the label given input

Probability neural net gives to concept vecs *where* ***program*** *returns* $\mathbf{y}$

Probability neural net gives to concept vec

Constraint: Do we return the correct **program** value?

This is again **weighted model counting**!

Used in many systems:

DeepProbLog, Scallop, A-NeSI, NeurASP, TensorLog...

(Citations at the end)

MNIST Addition

Predict **sum** $\mathbf{y}$ of two MNIST digits $\mathbf{x}$.

Labels of digits $\mathbf{c}$ **not given**!

Training Neurosymbolic Predictors

Digit classifier Constraint

$$\mathrm{p}_\theta(\mathbf{y} = 13 \mid \mathbf{x}) = \sum_{\mathbf{c} \in \{0,\ldots 9\}^2} \mathrm{p}_\theta(\mathbf{c} \mid \mathbf{x}) \cdot \mathbb{1}[\mathrm{c}_1 + \mathrm{c}_2 = 13]$$

Then minimise

$$\mathcal{L}(\mathbf{x}, \mathbf{y}) = -\log \mathrm{p}_\theta(\mathbf{y} \mid \mathbf{x})$$

Marconato, E., Teso, S., Vergari, A., & Passerini, A. (2024). Not all neuro-symbolic concepts are created equal: Analysis and mitigation of reasoning shortcuts. Advances in Neural Information Processing Systems, 36.

$$\mathrm{p}_\theta(\mathbf{y} = 13 \mid \mathbf{x}) = \sum_{\mathbf{c} \in \{0,9\}^2} \mathrm{p}_\theta(\mathbf{c} \mid \mathbf{x}) \mathbb{1}[\mathrm{c}_1 + \mathrm{c}_2 = 13]$$

$$= \mathrm{p}_\theta(\mathbf{c} = (4,9) \mid \mathbf{x}) + \mathrm{p}_\theta(\mathbf{c} = (5,8) \mid \mathbf{x}) + \mathrm{p}_\theta(\mathbf{c} = (6,8) \mid \mathbf{x}) +$$

$$\mathrm{p}_\theta(\mathbf{c} = (7,6) \mid \mathbf{x}) + \mathrm{p}_\theta(\mathbf{c} = (8,5) \mid \mathbf{x}) + \mathrm{p}_\theta(\mathbf{c} = (9,4) \mid \mathbf{x})$$

Making predictions

Method 1:

- Find most likely concepts $\mathbf{c}$
- Run through program: $\mathbf{y}' = \varphi(\mathbf{c})$

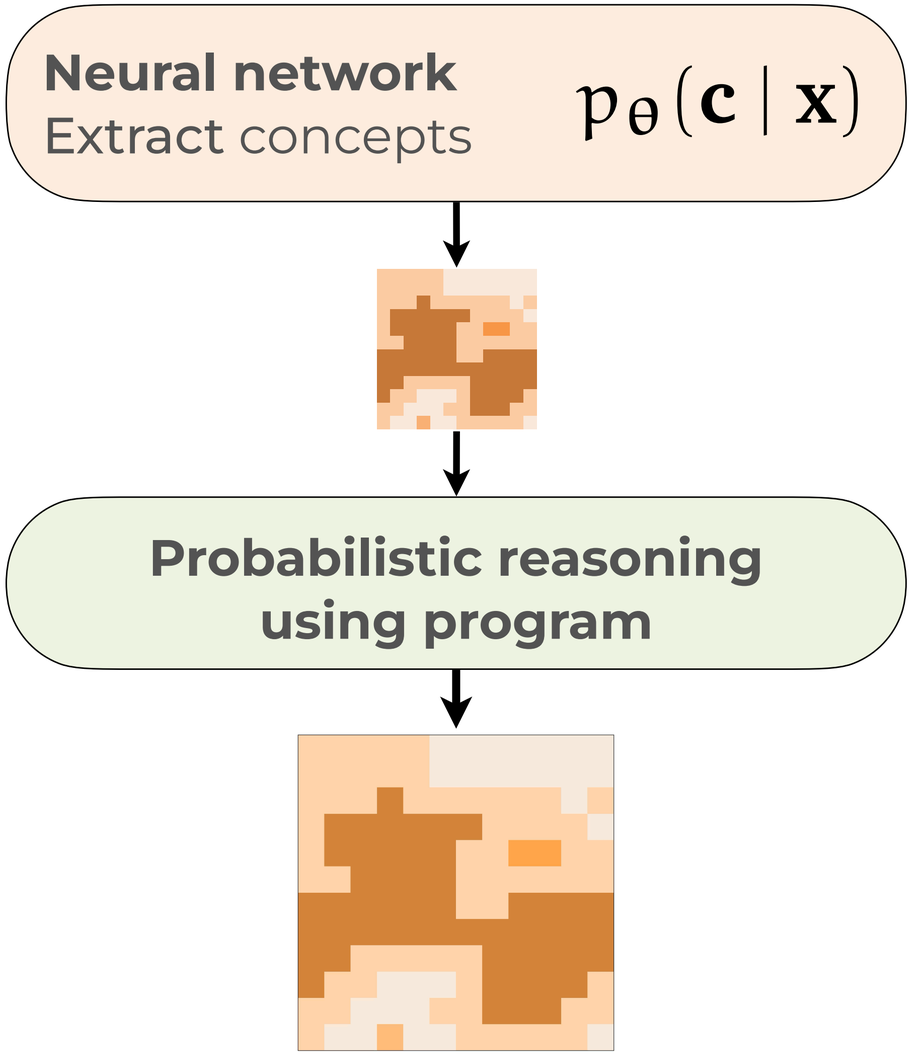
Method 2 (MAP):

Compute the most likely label using WMC

$$\mathbf{y}' = \arg\max_{\mathbf{y} \in \mathcal{Y}} p_\theta(\mathbf{y} \mid \mathbf{x})$$

This is much harder usually,

but the 'proper' thing to do



6 options

$$p_\theta(\mathbf{y} \mid \mathbf{x}) = \sum_{\mathbf{c} \in \mathcal{C}} p_\theta(\mathbf{c} \mid \mathbf{x}) \cdot \mathbb{1}[\varphi(\mathbf{c}) = \mathbf{y}]$$

Enumerate **all** options

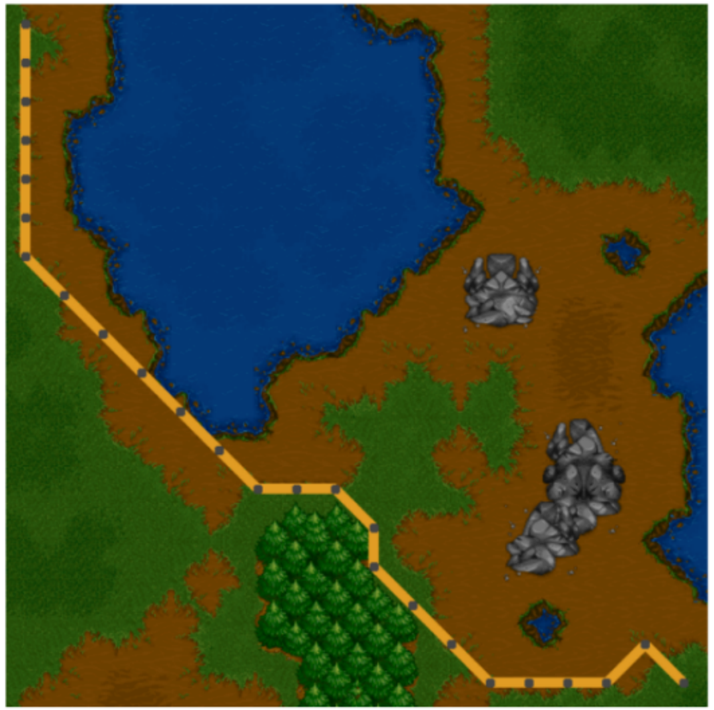
	N=1	N=2	N=4	N=15
		Symbolic prediction		
LTN	80.54 ± 23.33	77.54 ± 35.55	T/O	T/O
DeepProbLog	97.20 ± 0.50	95.20 ± 1.70	T/O	T/O
DPLA*	88.90 ± 14.80	83.60 ± 23.70	T/O	T/O
DeepStochLog	**97.90 ± 0.10**	**96.40 ± 0.10**	**92.70 ± 0.60**	T/O
Embed2Sym	97.62 ± 0.29	93.81 ± 1.37	91.65 ± 0.57	60.46 ± 20.36
A-NESI (predict)	97.66 ± 0.21	95.96 ± 0.38	92.56 ± 0.79	75.90 ± 2.21

van Krieken, Emile, et al. "A-nesi: A scalable approximate method for probabilistic neurosymbolic inference." Advances in Neural Information Processing Systems 36 (2023): 24586-24609.

$$|\mathcal{C}| = 10^{100}$$

Method	MNIST Addition			
	$N = 2$	$N = 4$	$N = 15$	$N = 50$
	Accuracy			
DPL	$94.20^{+2.04}_{-0.16}$	T/O	T/O	T/O
Scallop	$95.40^{+0.40}_{-0.08}$	$90.88^{+0.48}_{-0.48}$	T/O	T/O
A-NeSI	$95.40^{+0.60}_{-0.24}$	$92.40^{+0.64}_{-0.56}$	$74.71^{+2.94}_{-1.27}$	T/O
PLIA_t	$95.88^{+0.28}_{-0.16}$	$91.60^{+0.72}_{-0.16}$	$79.09^{+1.21}_{-2.12}$	$43.00^{+5.00}_{-3.00}$
	Timings (minutes)			
DPL	$88.21^{+15.58}_{-1.89}$	T/O	T/O	T/O
Scallop	$22.62^{+0.64}_{-0.06}$	$50.41^{+6.46}_{-0.17}$	T/O	T/O
A-NeSI	$41.02^{+8.54}_{-2.79}$	$53.62^{+6.40}_{-1.76}$	$714.55^{+27.66}_{-8.17}$	T/O
PLIA_t	$1.97^{+0.02}_{-0.01}$	$2.44^{+0.04}_{-0.04}$	$7.85^{+0.64}_{-0.19}$	$11.98^{+0.68}_{-0.04}$

De Smet, L., & Zuidberg Dos Martires, P. (2024). A fast convoluted story: Scaling probabilistic inference for integer arithmetics. Advances in Neural Information Processing Systems, 37, 102456-102478.



	12×12	30×30
I-MLE [41]	97.2 ± 0.5	$\mathbf{93.7} \pm \mathbf{0.6}$
RLOO [29]	43.75 ± 12.35	12.59 ± 16.38
A-NeSI	94.57 ± 2.27	17.13 ± 16.32
A-NeSI + RLOO	$\mathbf{98.96} \pm \mathbf{1.33}$	67.57 ± 36.76
$\lvert\mathcal{C}\rvert$	$5^{12\times12}$	$5^{30\times30}$

van Krieken, Emile, et al. "A-nesi: A scalable approximate method for probabilistic neurosymbolic inference." Advances in Neural Information Processing Systems 36 (2023): 24586-24609.

Probabilistic reasoning methods

Semantic Loss

$$-\log \mathbb{P}_\theta[\varphi \text{ is true} \mid \mathbf{x}]$$

Semantic Probabilistic Layers

$$p_{\theta,\varphi}(\mathbf{c} \mid \mathbf{x}) := \frac{p_\theta(\mathbf{c} \mid \mathbf{x}) \cdot \varphi(\mathbf{c})}{\mathbb{P}_\theta[\varphi \text{ is true} \mid \mathbf{x}]}$$

All weighted model counts!

NeSy predictors

$$p_\theta(\mathbf{y} \mid \mathbf{x}) := \mathbb{P}_\theta[\varphi(\mathbf{c}) = \mathbf{y} \mid \mathbf{x}]$$

These methods all use probabilistic reasoning!

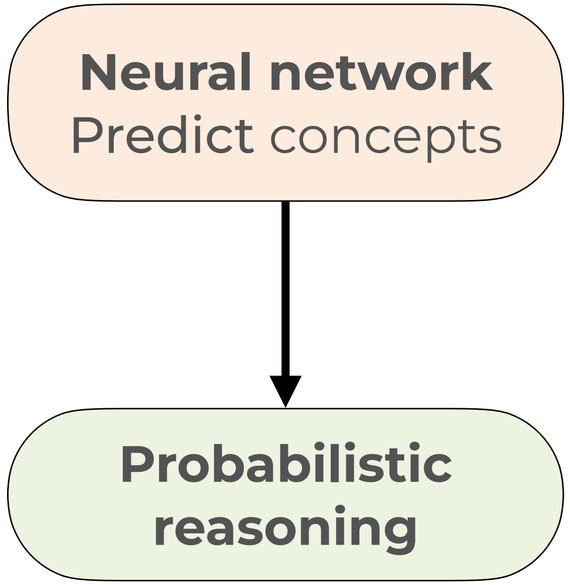
Content

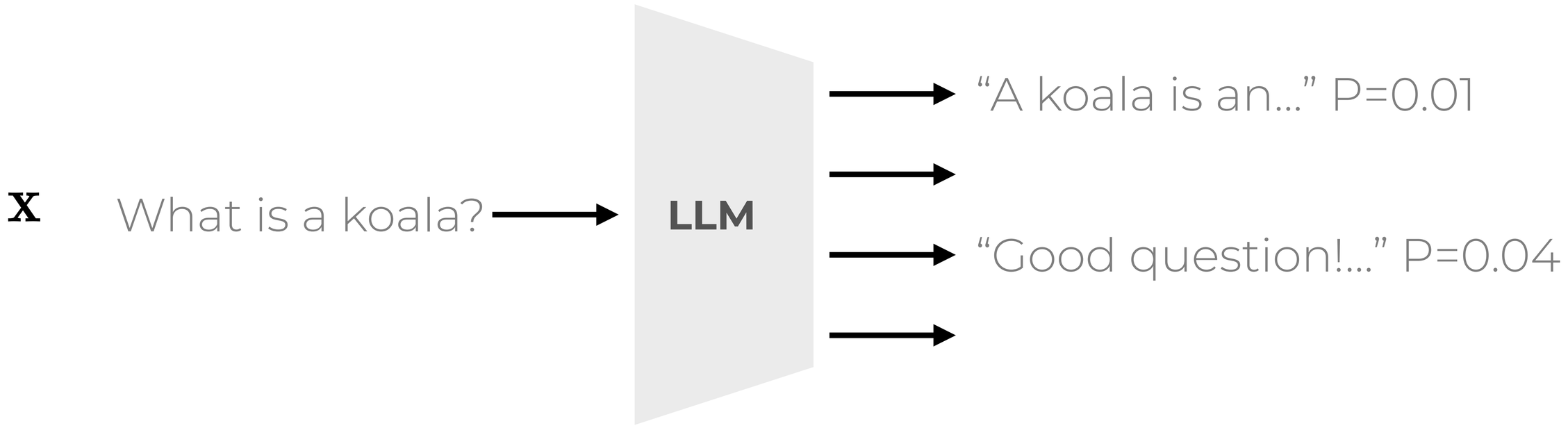
Probabilistic Neurosymbolic methods

- Semantic loss
- Semantic probabilistic layers
- Neurosymbolic predictors
- Why probabilistic reasoning?

Inference

- Circuits & exact inference
- Approximate inference





Most current-day machine learning models are actually probabilistic models!

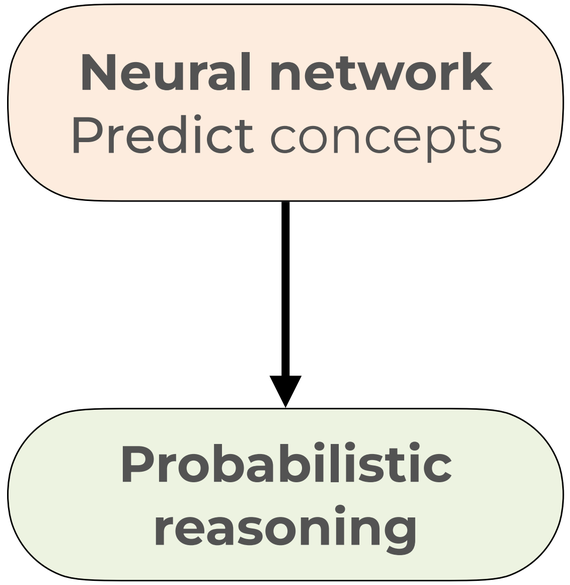
Both **generative models** and **classifiers**

Why probabilistic reasoning?

Uniform framework for creating methods:

- Reuse existing algorithms
- Combine with all kinds of models
  - Generative
  - Classifiers

And it has some **nice properties**



What are the properties of probabilistic reasoning?

Properties of probabilistic reasoning

1. If φ is a **tautology**, then $\mathbb{P}_\theta[\varphi \text{ is true}] = 1$

Why? For all concept vectors, $\varphi(\mathbf{c}) = 1$

$$\mathbb{P}_\theta[\varphi \text{ is true}] = \sum_{\mathbf{c} \in \mathcal{C}} p_\theta(\mathbf{c}) \cdot 1 = 1$$

Since distributions sum to 1

Properties of probabilistic reasoning

2. If φ is a **contradiction**, then $\mathbb{P}_\theta[\varphi \text{ is true}] = 0$

Why? For all concept vectors, $\varphi(\mathbf{c}) = 0$, so

$$\mathbb{P}_\theta[\varphi \text{ is true}] = \sum_{\mathbf{c} \in \mathcal{C}} p_\theta(\mathbf{c}) \cdot 0 = 0$$

Comparison: Fuzzy logic (Day 1)

These properties do **not** hold for fuzzy logics!

Tautology:

$$c \vee \neg c = \max(c, 1 - c)$$

Contradiction:

$$c \wedge \neg c = \min(c, 1 - c)$$

Why probabilistic reasoning?

3. Monotonicity: If $\varphi \models \psi$, then $\mathbb{P}_\theta[\varphi \text{ is true}] \leqslant \mathbb{P}_\theta[\varphi \text{ is true}]$

If $\varphi(\mathbf{c})$ is true, then also $\psi(\mathbf{c})$ is true.
So: We *at least* sum over the models of φ

$$\mathbb{P}_\theta[\psi \text{ is true}] = \mathbb{P}_\theta[\varphi \text{ is true}] + \mathbb{P}_\theta[\varphi \text{ is false}, \psi \text{ is true}]$$

Why probabilistic reasoning?

1. If φ is a **tautology**, then $\mathbb{P}_\theta[\varphi \text{ is true}] = 1$
2. If φ is a **contradiction**, then $\mathbb{P}_\theta[\varphi \text{ is true}] = 0$
3. **Monotonicity**: If $\varphi \models \psi$, then $\mathbb{P}_\theta[\varphi \text{ is true}] \leqslant \mathbb{P}_\theta[\varphi \text{ is true}]$
4. **Equivalence**: If $\varphi \equiv \psi$, then $\mathbb{P}_\theta[\varphi \text{ is true}] = \mathbb{P}_\theta[\varphi \text{ is true}]$

These are all natural logical properties!

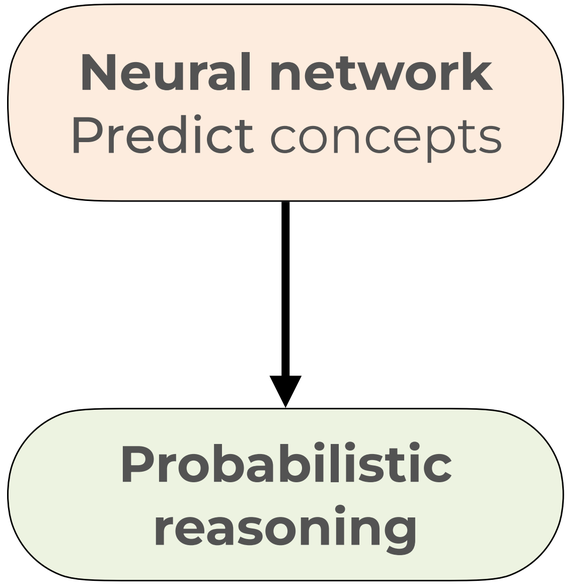
Content

Probabilistic Neurosymbolic methods

- Semantic loss
- Semantic probabilistic layers
- Neurosymbolic predictors
- Why probabilistic reasoning?

Inference

- Circuits & exact inference
- Approximate inference



Exact inference:
How to compute the weighted model count efficiently & exactly?

$$p_\theta(\mathbf{y} \mid \mathbf{x}) = \sum_{\mathbf{c} \in \mathcal{C}} p_\theta(\mathbf{c} \mid \mathbf{x}) \cdot \mathbb{1}[\varphi(\mathbf{c}) = \mathbf{y}]$$

Enumerate **all** options

$= 13$

6 options

$= 143$

56 options

$= 1443$

556 options

15 digits?

$5.6 \cdot 10^{14}$ ***options***

Enumerating all options is not feasible!

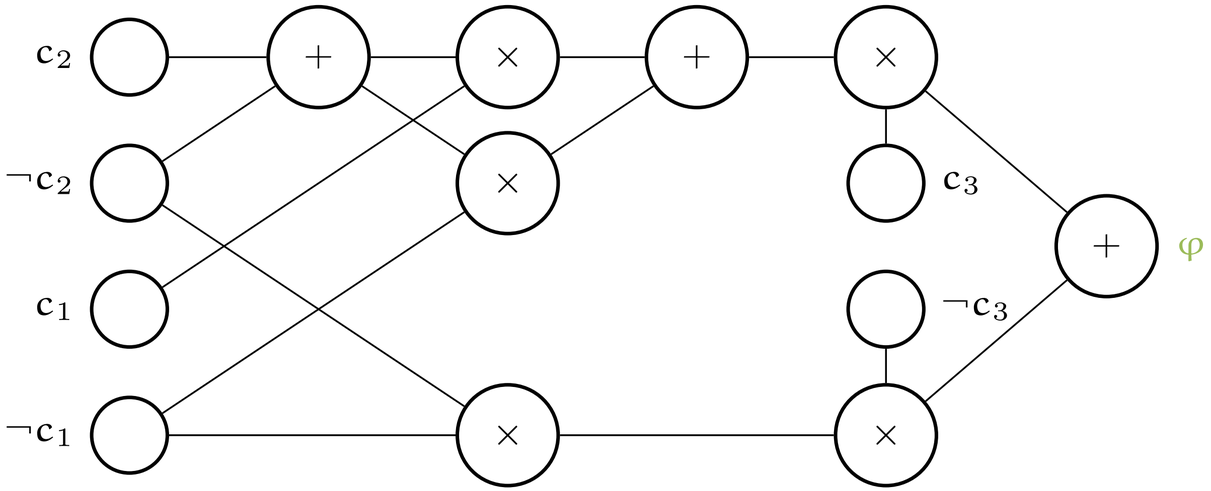
Scaling exact inference

Enumerating all concept vectors in $\mathcal{C}$ is infeasible

We use two ideas

1. **Simple models**

2. **Compile** Boolean formula into **Circuit**

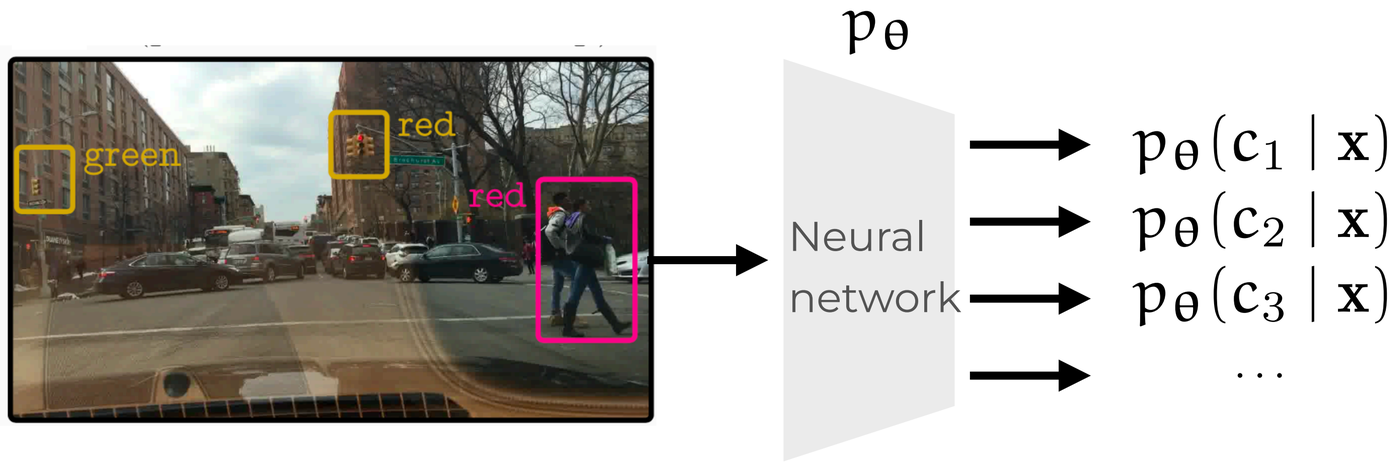


Idea 1: Simple models

For now, let's assume conditional **independence***:

$$p_\theta(\mathbf{c} \mid \mathbf{x}) = \prod_{i=1}^{k} p_\theta(c_i \mid \mathbf{x})$$

Common in practical NeSy systems, but not in eg LLMs :(



*Independence is **sufficient but not necessary** for tractability

Independence: Example

r	g	$\neg r \vee \neg g$	$p(r, g)$
0	0	true	$(1 - p(r))(1 - p(g))$
0	1	true	$(1 - p(r))p(g)$
1	0	true	$p(r)(1 - p(g))$
1	1	false	$p(r)p(g)$

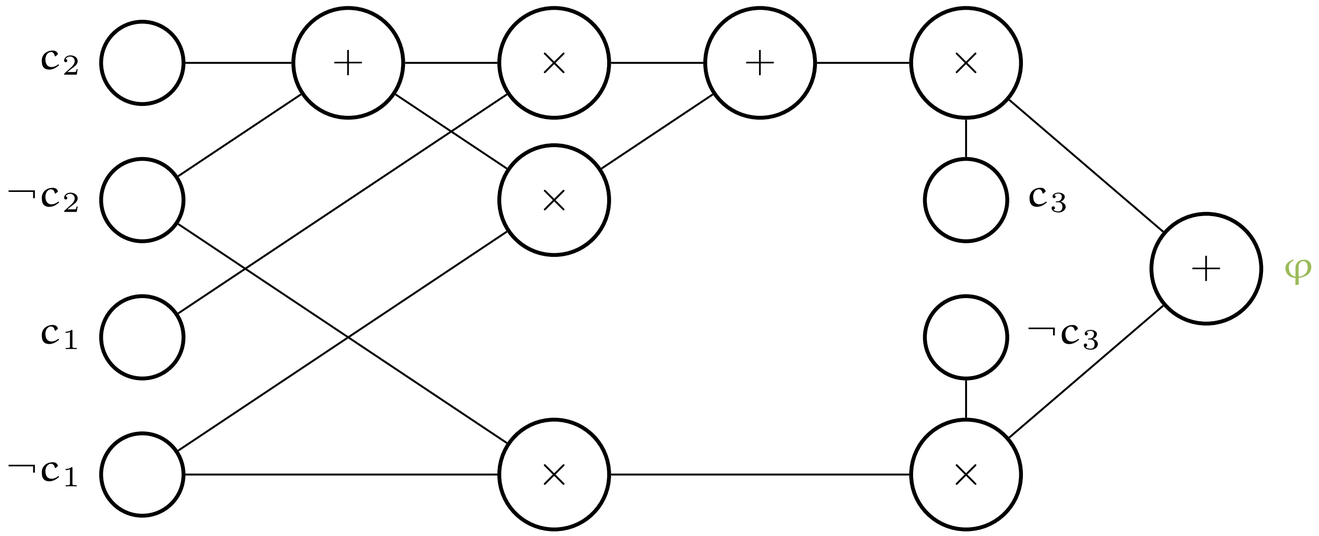
$$\mathbb{P}[\neg r \vee \neg g] = p(r, \neg g) + p(\neg r, g) + p(\neg r, \neg g)$$

Idea 2: Circuits

Circuits are DAGs defined by:

- Simple **inputs** (leaves, represent concepts)
- **Sum** (+) nodes (eg: disjunction)
- **Product** (x) nodes (eg: conjunction)

These circuits represent Boolean formulas!

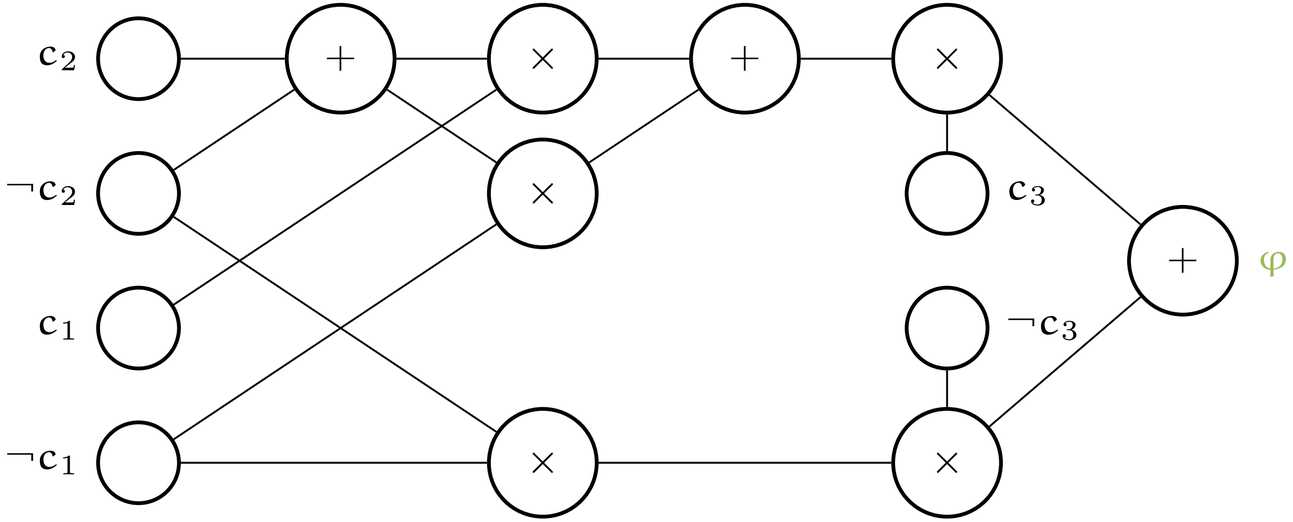


Slide adapted from Antonio Vergari. See https://april-tools.github.io/ for work on Probabilistic Circuits

Knowledge compilation

The process of turning formulas into circuits so we can answer certain queries **tractably**!

$$\varphi = (c_1 \rightarrow c_3) \wedge (c_2 \rightarrow c_3)$$



This is a very rich field - I will only provide a taste here.

Slides adapted from Antonio Vergari's tutorial . See https://april-tools.github.io/ for work on Probabilistic Circuits
Vergari, A., Choi, Y., Liu, A., Teso, S., & Van den Broeck, G. (2021). A compositional atlas of tractable circuit operations for probabilistic inference. Advances in Neural Information Processing Systems, 34, 13189-13201.

Tractability

"A set of queries is **tractable** on a **set of circuits** if

- for any specific query $\mathfrak{q}$ and circuit $\mathcal{M}$,
  - **exactly** *computing* $\mathfrak{q}$ on $\mathcal{M}$
  - runs in **polynomial time** wrt the *size* of $\mathcal{M}$"

Example queries:

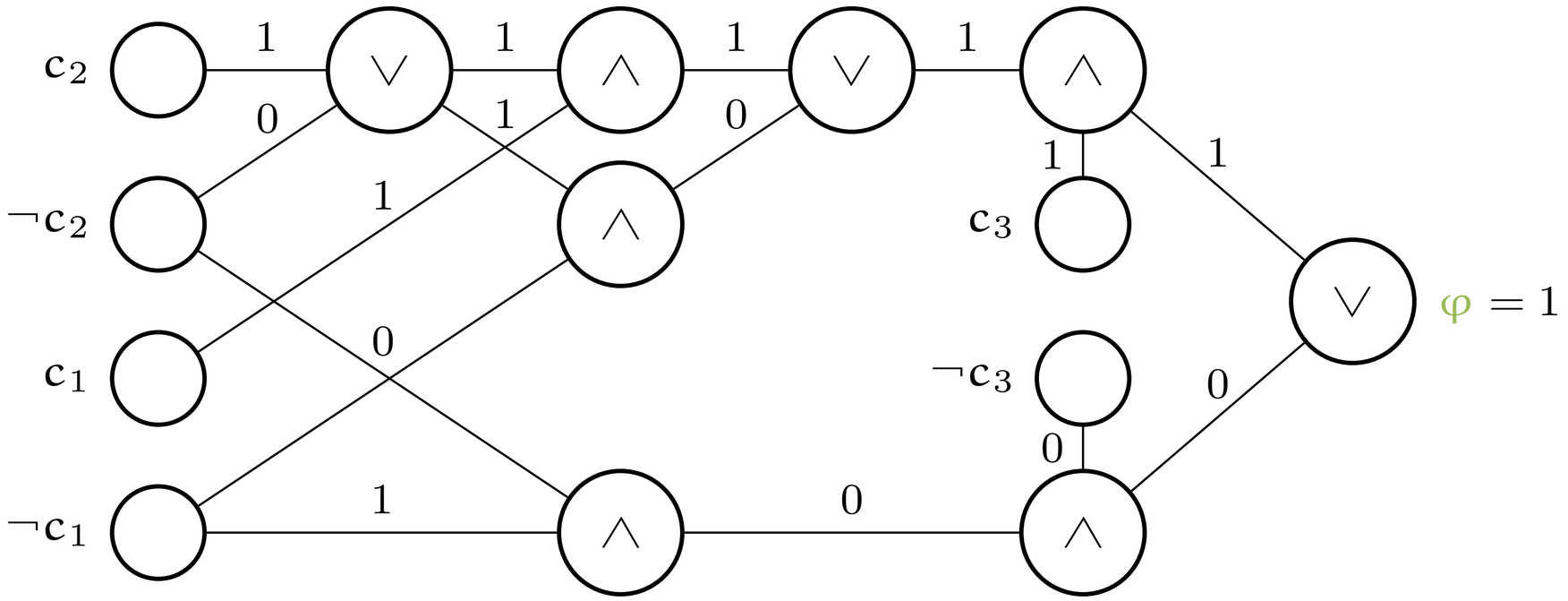
- WMC (usually #P-hard)
- Most likely output (~MAX-SAT)

Adapted from Antonio Vergari from *http://web.cs.ucla.edu/~guyvdb/slides/ECAI20.pdf***.**
See https://april-tools.github.io/ for work on Probabilistic Circuits

Query: Computing truth values

Replace sum nodes with $\vee$ and product nodes with $\wedge$

Then set c_i to the values: $c_1 = c_2 = c_3 = 1$

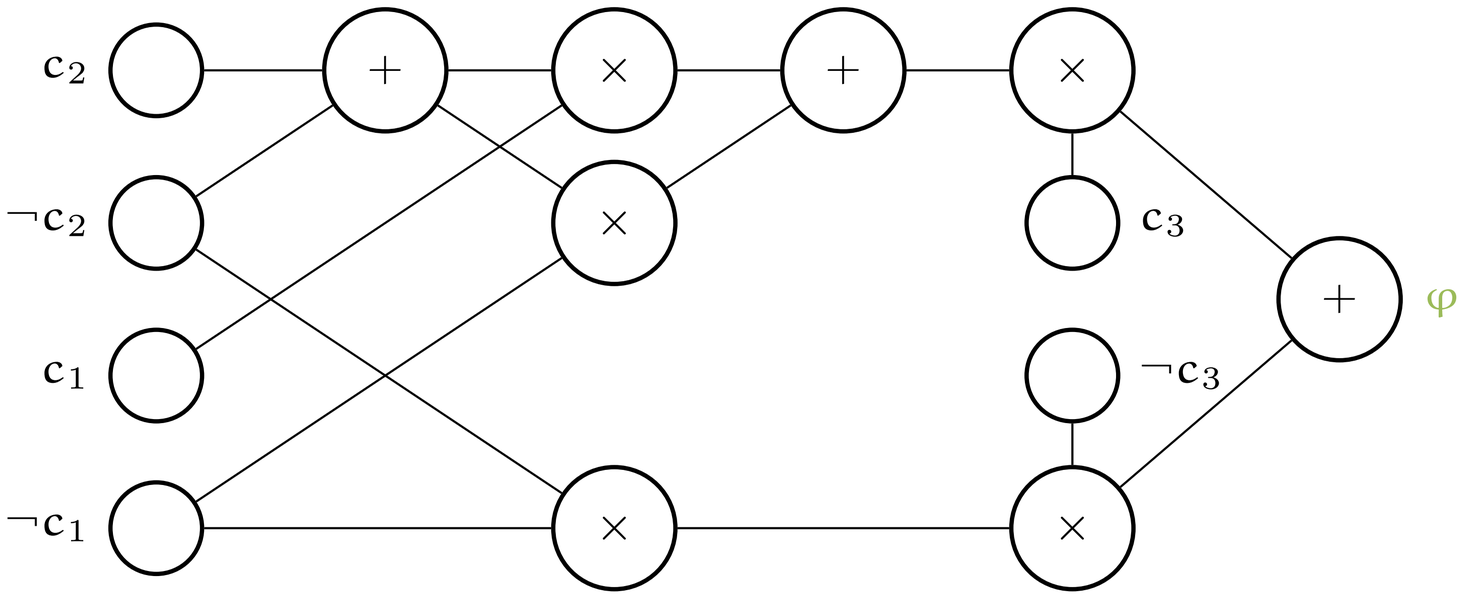


$$\varphi = (c_1 \rightarrow c_3) \wedge (c_2 \rightarrow c_3)$$

Slide adapted from Antonio Vergari. See https://april-tools.github.io/ for work on Probabilistic Circuits

Circuits

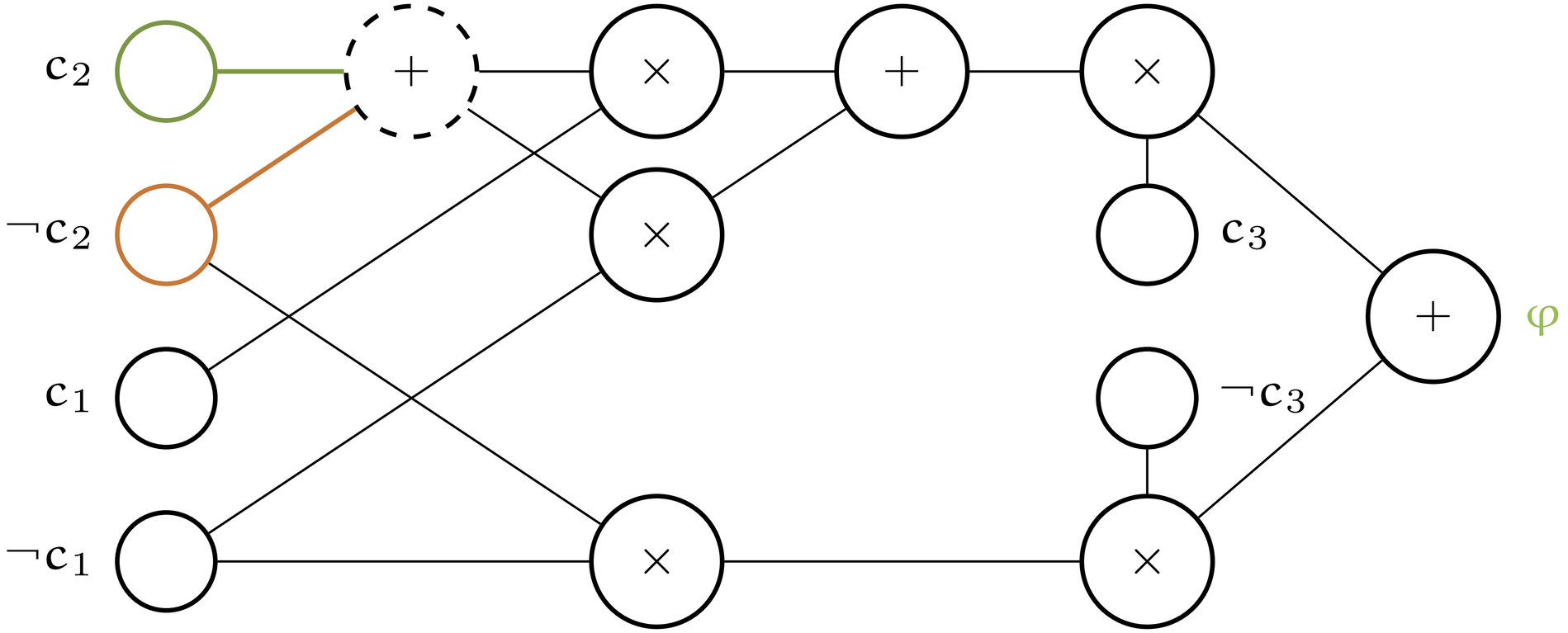
The circuit **structure** determines what queries are **tractable.**



$$\varphi = (c_1 \rightarrow c_3) \wedge (c_2 \rightarrow c_3)$$

Smooth circuits

All children of **sum** nodes depend on **same** set of **concepts**



$$\varphi = (c_1 \rightarrow c_3) \wedge (c_2 \rightarrow c_3)$$

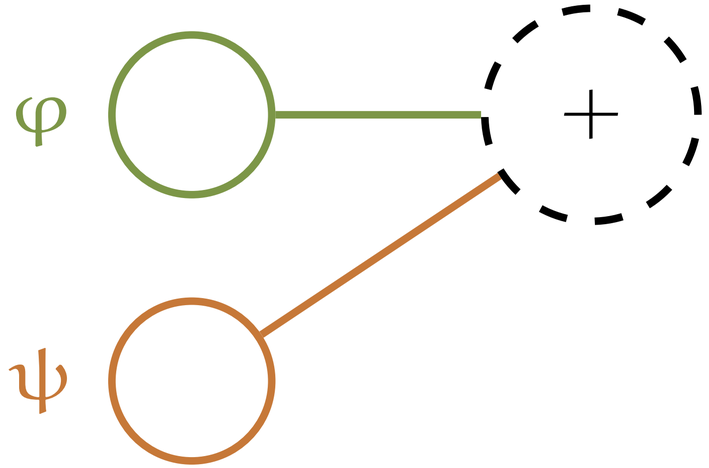
Smooth circuits

All children of **sum** nodes depend on **same concepts**

Then:

Probability of sum node = sum of probabilities of its children

$$\mathbb{P}_\theta[\varphi \text{ is true or } \psi \text{ is true}] = \mathbb{P}_\theta[\varphi \text{ is true}] + \mathbb{P}_\theta[\psi \text{ is true}]$$

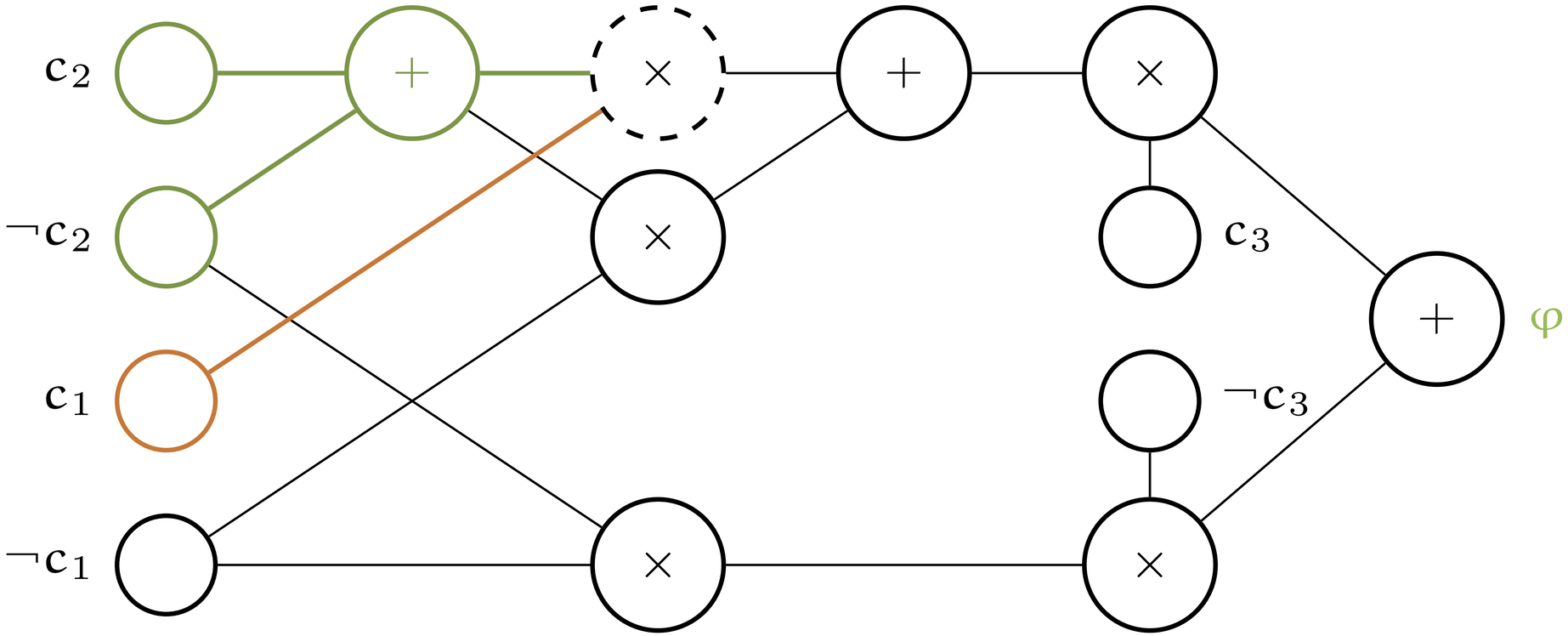


Moves probability computation 'inwards'!

Example later

Decomposable circuits

All children of **product** nodes depend on **different concepts**



$$\varphi = (c_1 \rightarrow c_3) \wedge (c_2 \rightarrow c_3)$$

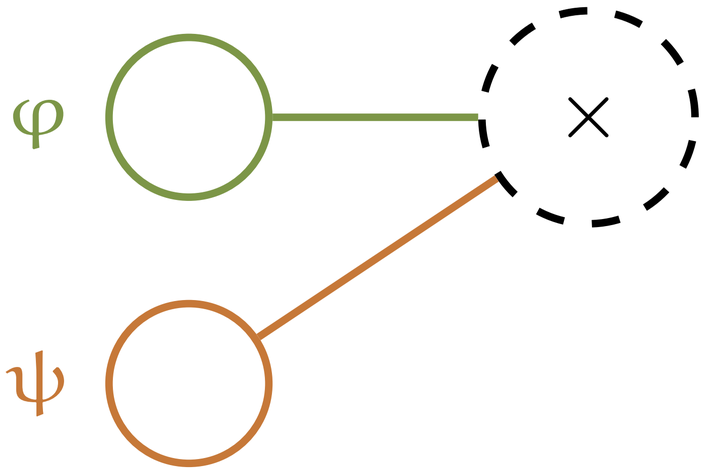
Decomposable circuits

All children of **product** nodes depend on **different concepts**

Then:

Probability of node = product of probabilities of its children

$$\mathbb{P}_\theta[\varphi \text{ is true and } \psi \text{ is true}] = \mathbb{P}_\theta[\varphi \text{ is true}] \cdot \mathbb{P}_\theta[\psi \text{ is true}]$$



Moves probability computation 'inwards'!

Example later

Query: Model counting

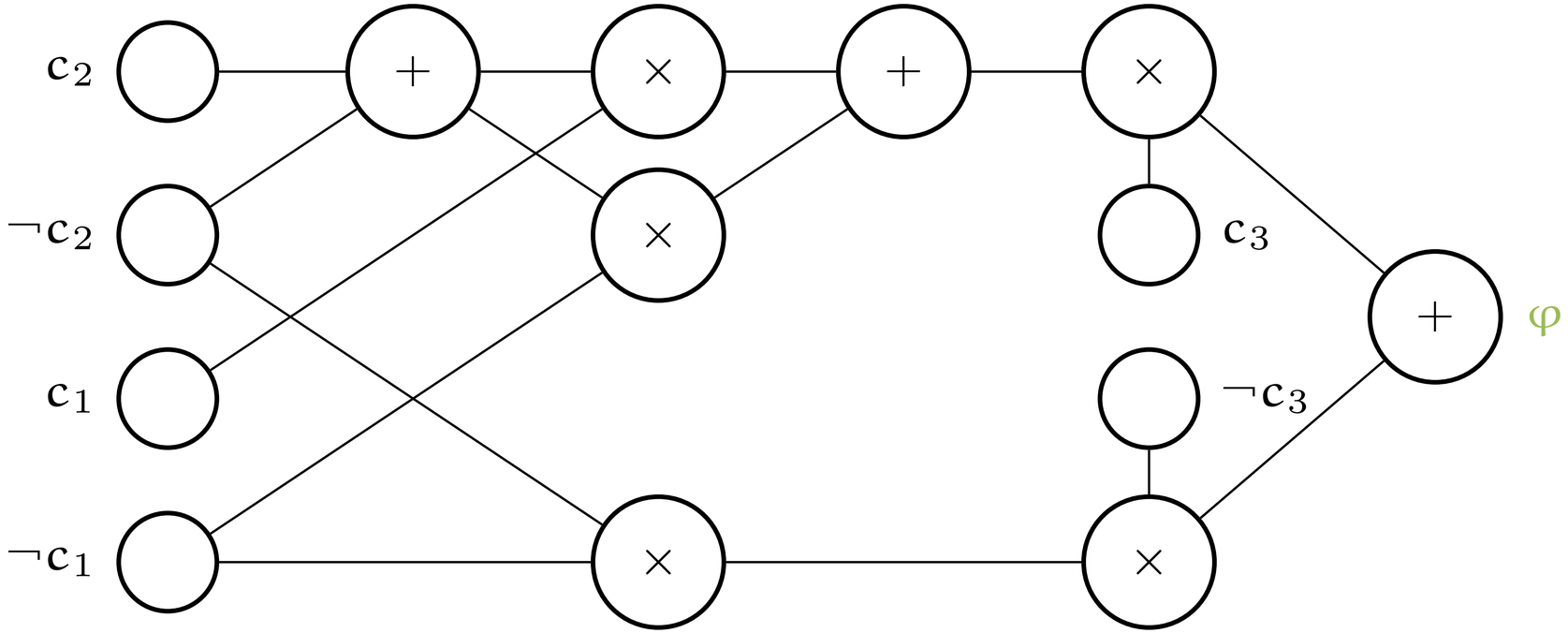
What is the number of models of the formula?

This is #P-complete

c_1	c_2	c_3	$\varphi = (c_2 \rightarrow c_3) \wedge (c_2 \rightarrow c_3)$
0	0	0	true
0	0	1	true
0	1	0	false
0	1	1	true
1	0	0	false
1	0	1	true
1	1	0	false
1	1	1	true

Smooth & decomposable circuits

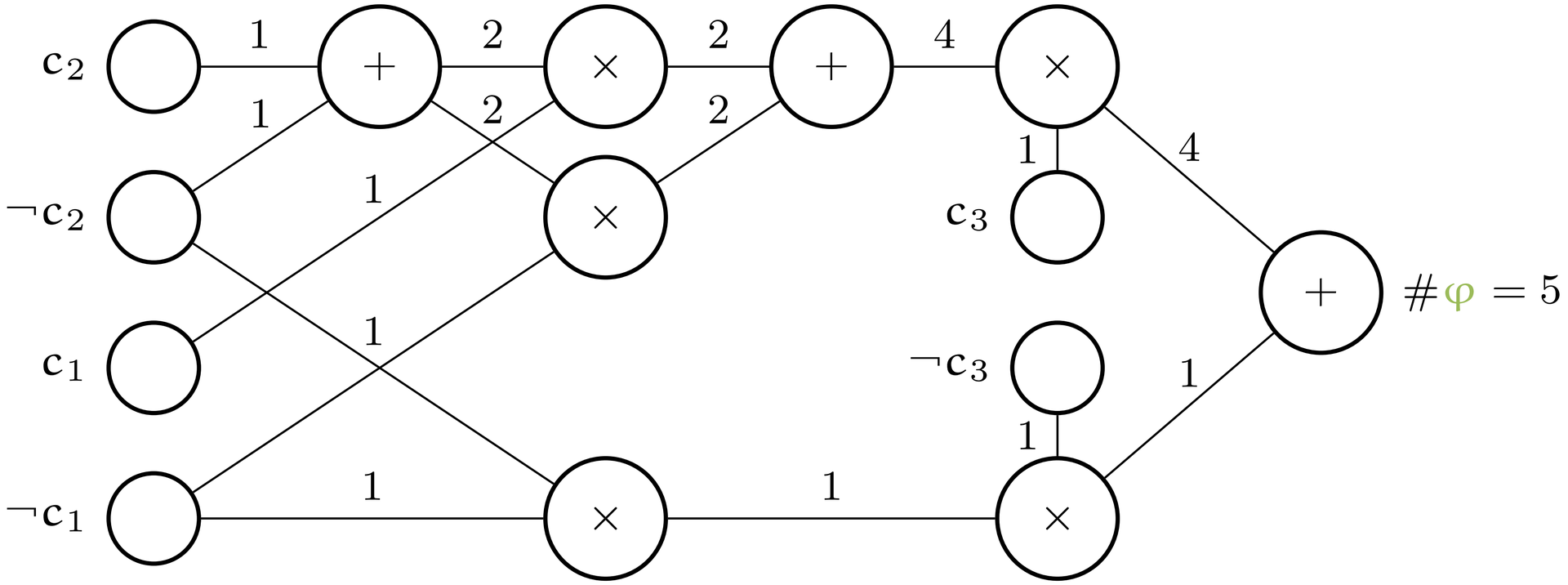
Result: Compute model count in **linear** time wrt circuit size!



$$\varphi = (c_1 \rightarrow c_3) \wedge (c_2 \rightarrow c_3)$$

Query: Model Counting

Set **all** inputs to the values 1 and compute with + and x



$$\varphi = (c_1 \rightarrow c_3) \wedge (c_2 \rightarrow c_3)$$

Slide adapted from Antonio Vergari. See https://april-tools.github.io/ for work on Probabilistic Circuits

Query: Weighted Model counting

$$\varphi = (c_1 \rightarrow c_3) \wedge (c_2 \rightarrow c_3)$$

$$p(c_1) = 0.4,$$
$$p(c_2) = 0.2,$$
$$p(c_3) = 0.5$$

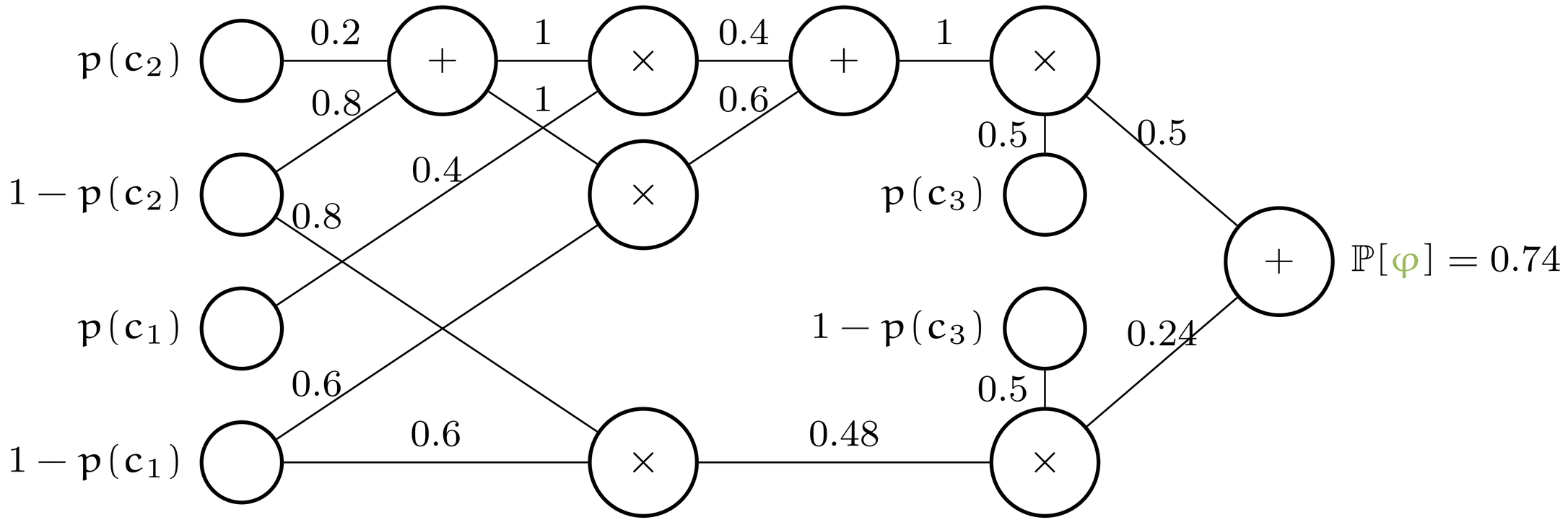
c_1	c_2	c_3	$p_\theta(\mathbf{c})$
0	0	0	0.24
0	0	1	0.24
0	1	1	0.06
1	0	1	0.16
1	1	1	0.04

How to compute this on the circuit?

$$\mathbb{P}[\varphi \text{ is true}] = 0.74$$

Combining with probabilities

Compute **weighted** model count in linear time wrt circuit size!

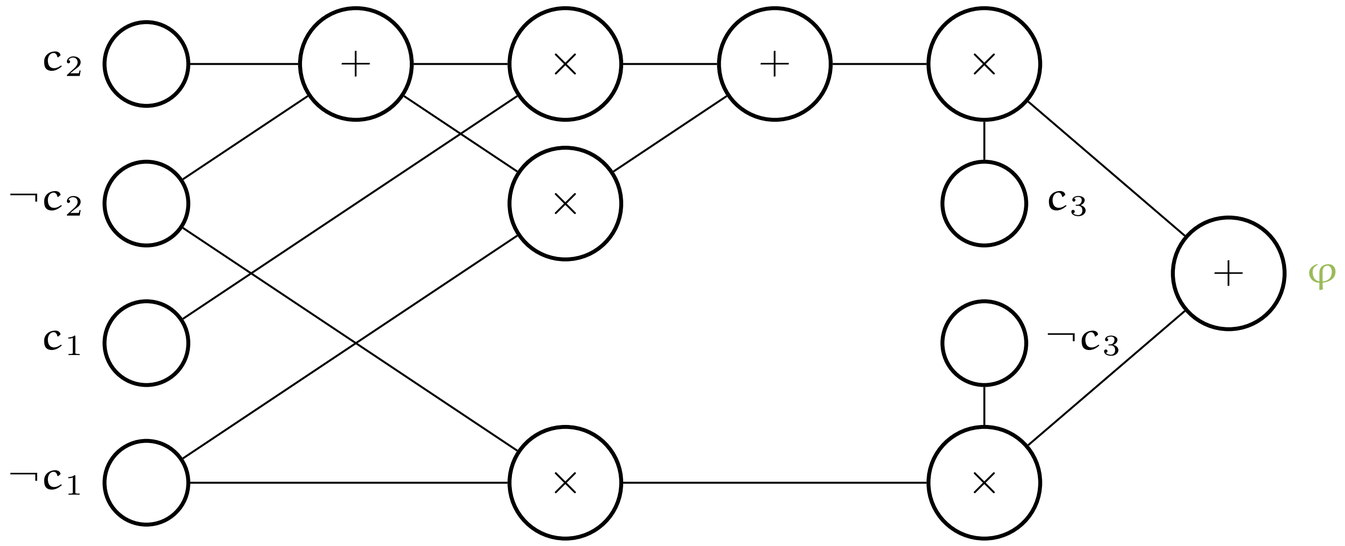


$$\varphi = (c_1 \rightarrow c_3) \wedge (c_2 \rightarrow c_3)$$

Smooth & decomposable circuits

With a **smooth & decomposable** circuit, we can do WMC!

Recall: We needed WMC for all three methods

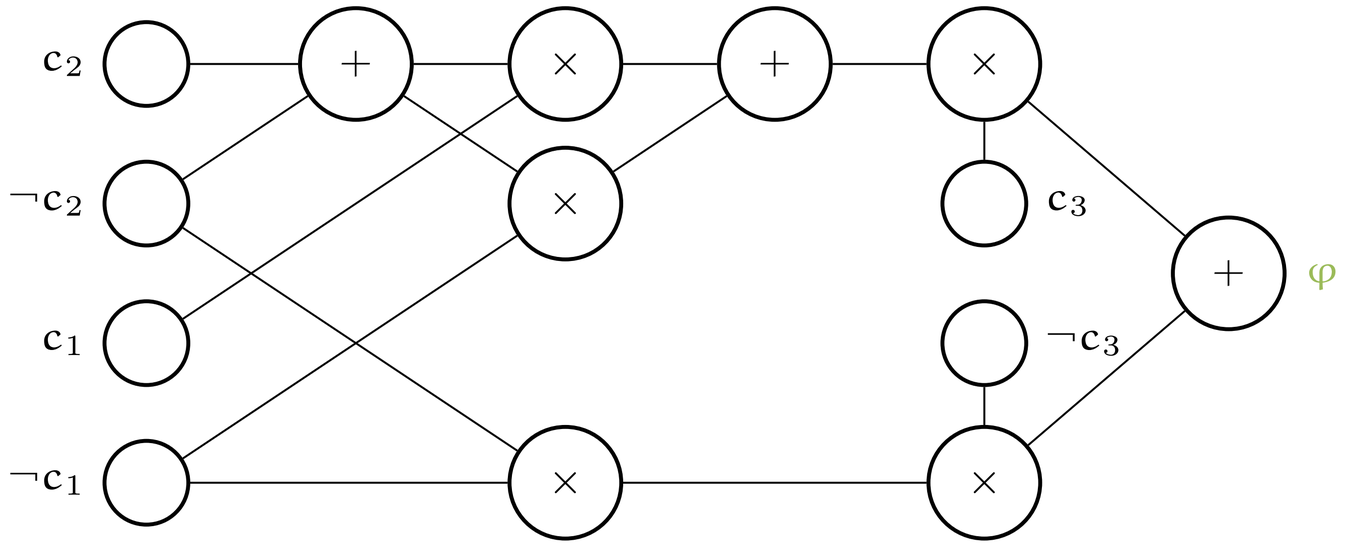


Semantic Probabilistic Layers

In addition to using WMC, SPL predicts with the **most likely assignment**

$$\arg\max_{\mathbf{c} \in \mathcal{C}} p_{\theta, \varphi}(\mathbf{c} \mid \mathbf{x})$$

But **smooth & decomposable** is **not sufficient** for computing this query tractably!



Query: Most likely assignment

What is the most likely assignment of the formula?

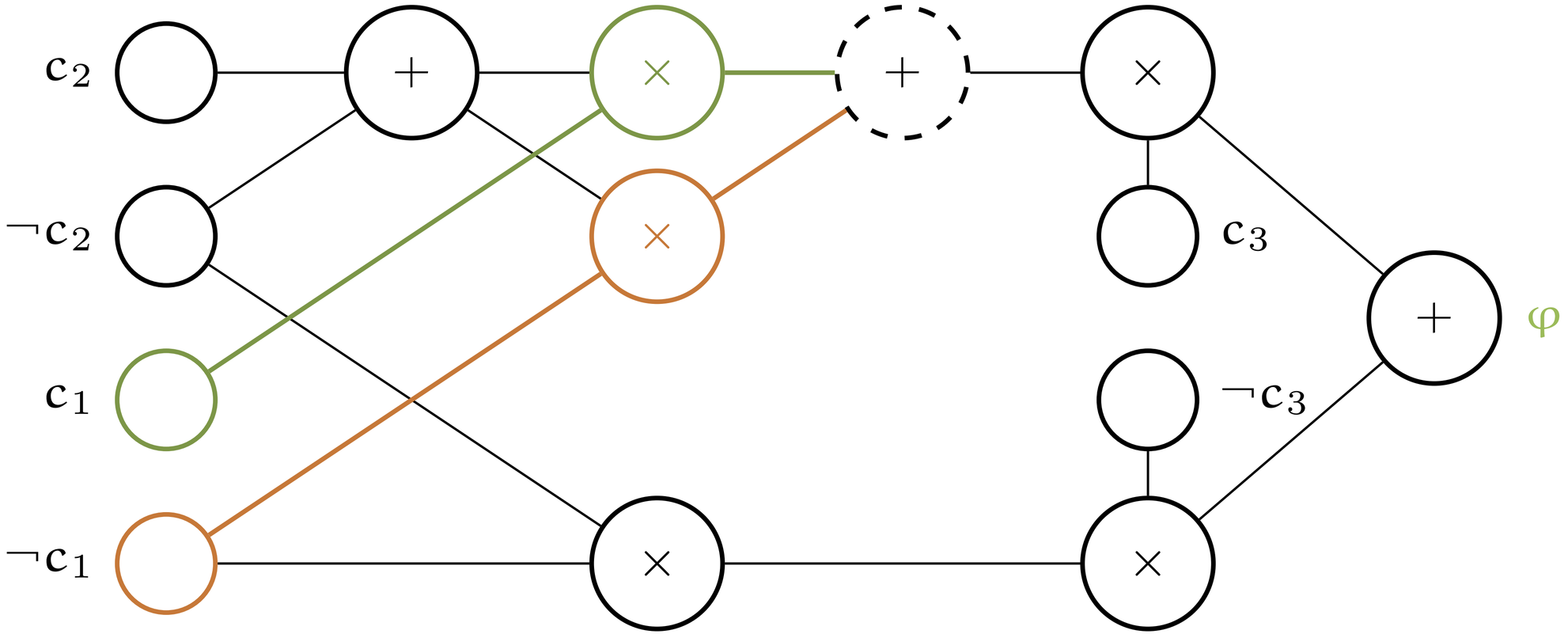
$$\varphi = (\mathbf{c}_1 \rightarrow \mathbf{c}_3) \wedge (\mathbf{c}_2 \rightarrow \mathbf{c}_3)$$

$$p_\theta(\mathbf{c}_1 \mid \mathbf{x}) = 0.4,$$
$$p_\theta(\mathbf{c}_2 \mid \mathbf{x}) = 0.2,$$
$$p_\theta(\mathbf{c}_3 \mid \mathbf{x}) = 0.6$$

$\mathbf{c}_1$	$\mathbf{c}_2$	$\mathbf{c}_3$	$p_\theta(\mathbf{c} \mid \mathbf{x})$
0	0	0	0.192
0	0	1	**0.288**
0	1	1	0.072
1	0	1	0.192
1	1	1	0.048

Deterministic circuits

All children of **sum** nodes have **disjoint support**

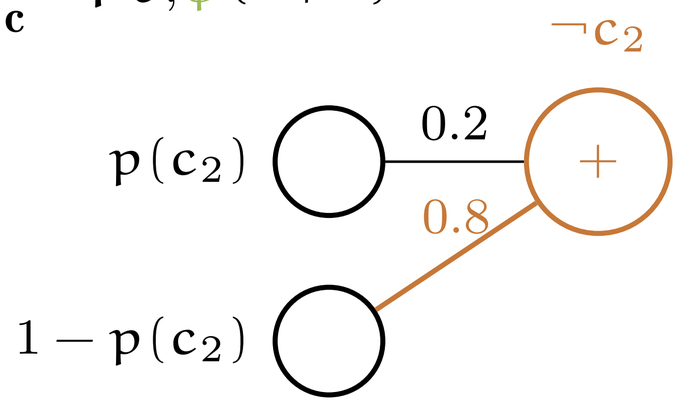


$$\varphi = (c_1 \rightarrow c_3) \wedge (c_2 \rightarrow c_3)$$

Smooth, decomposable & deterministic circuits

Compute **most likely assignment** in linear time! (Eg: SPL)

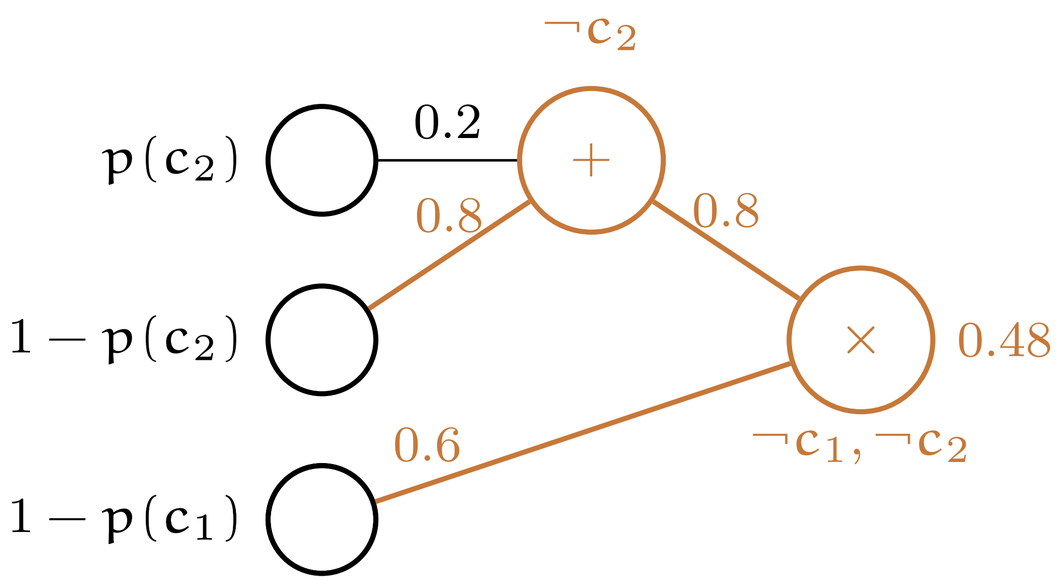
$$\max_{\mathbf{c}} p_{\theta,\varphi}(\mathbf{c} \mid \mathbf{x})$$



+ nodes: **Choose most likely input**

Smooth, decomposable & deterministic circuits

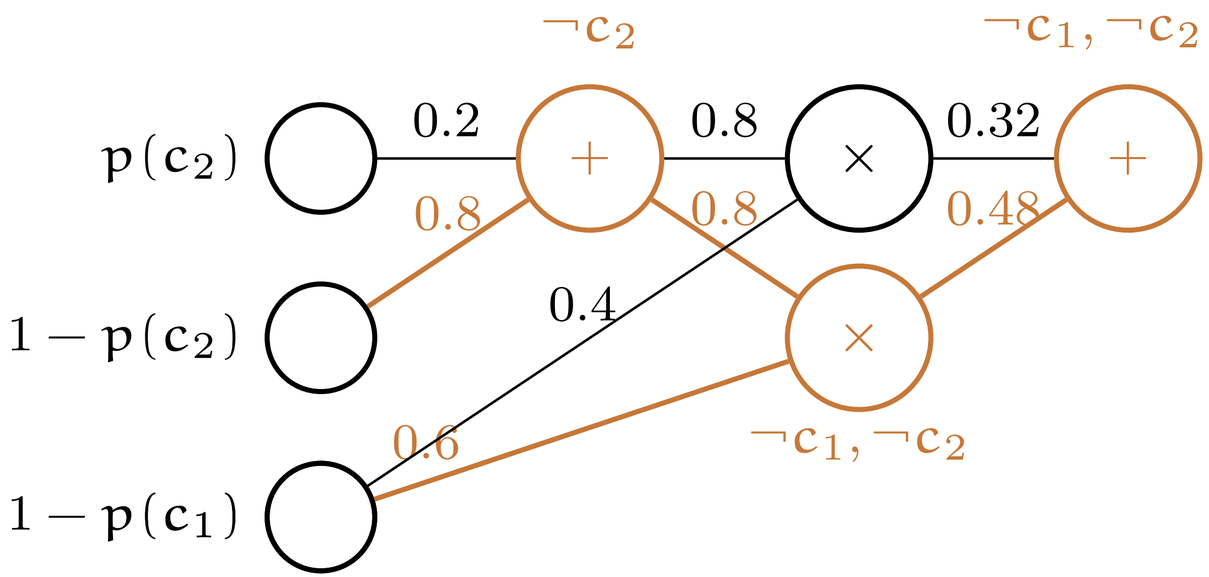
Compute **most likely assignment** in linear time!



x nodes: **Multiply probabilities and join inputs**

Smooth, decomposable & deterministic circuits

Compute **most likely assignment** in linear time!

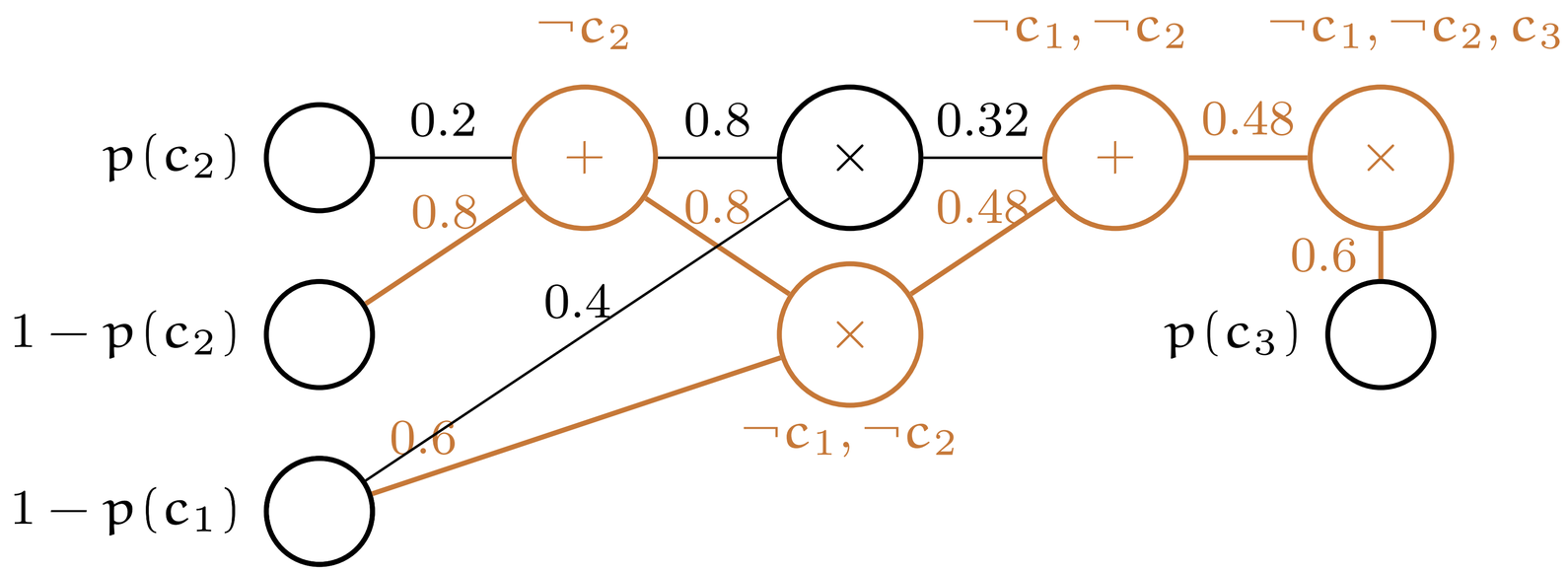


+ nodes: Choose most likely input

x nodes: Multiply probabilities and join inputs

Smooth, decomposable & deterministic circuits

Compute **most likely assignment** in linear time!

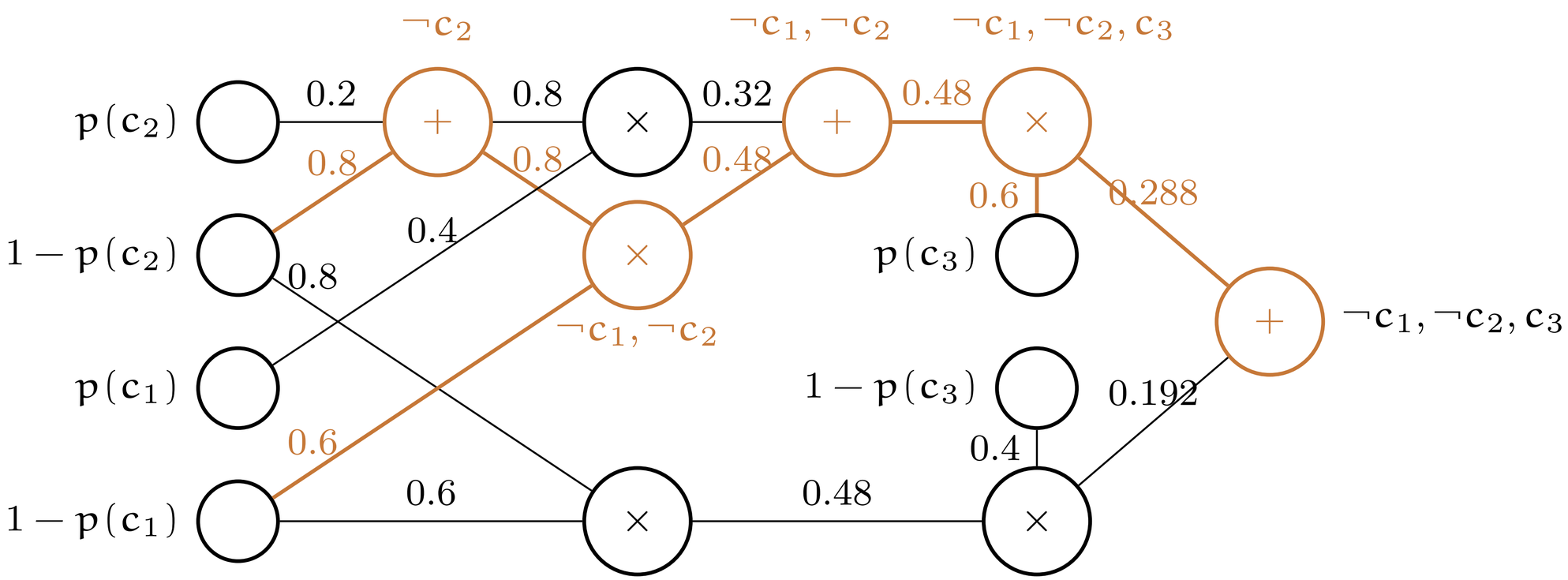


+ nodes: Choose most likely input

x nodes: Multiply probabilities and join inputs

Smooth, decomposable & deterministic circuits

Compute **most likely assignment** in linear time!



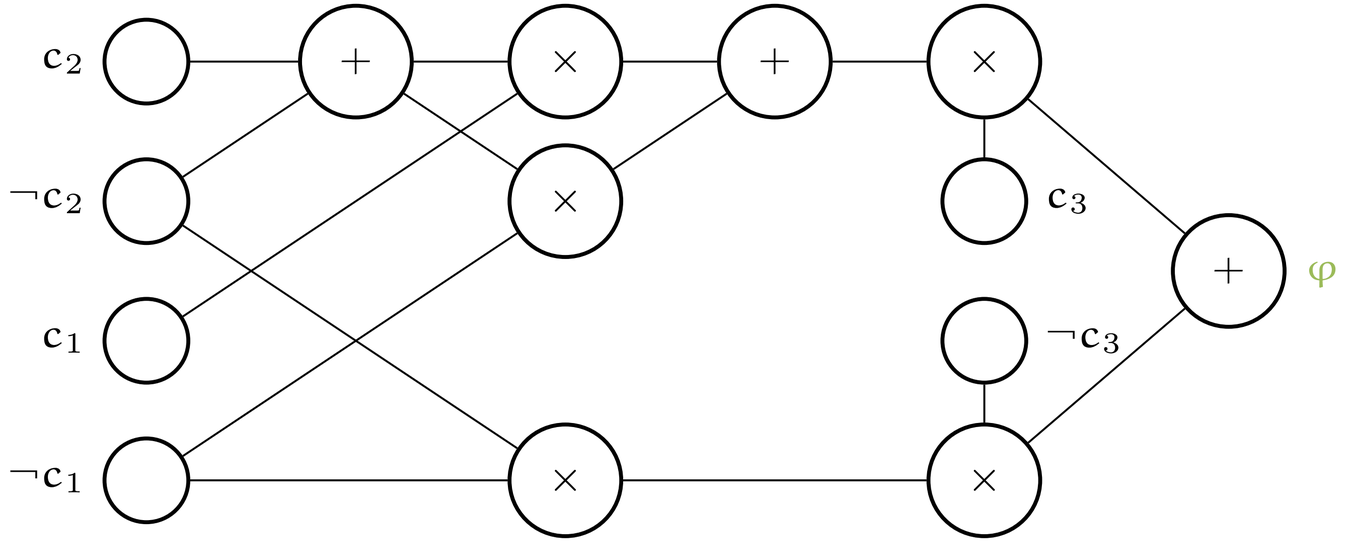
+ nodes: Choose most likely input

x nodes: Multiply probabilities and join inputs

Smooth, decomposable & deterministic circuits

With a **smooth & decomposable** circuit, we can do WMC!
Needed for Semantic Loss, SPL and NeSy Predictors

If we also have **determinism**, we can also find the most likely assignment: Needed for SPL



Circuits

Okay...

Beautiful...

Interesting...

But how do we get these circuits to begin with?

How long does that process take?

How big are these resulting circuits?

Knowledge compilation

Many questions!

But … it’s honestly an ESSLLI course of its own :-)

In practice, you can use the **SDD compiler**, which transforms CNF into smooth, decomposable and deterministic circuits

https://github.com/ML-KULeuven/PySDD

Darwiche, A., & Marquis, P. (2002). A knowledge compilation map. Journal of Artificial Intelligence Research, 17, 229-264.
Kisa, D., Van den Broeck, G., Choi, A., & Darwiche, A. (2014, July). Probabilistic Sentential Decision Diagrams. In KR.
Oztok, U., & Darwiche, A. (2015, July). A Top-Down Compiler for Sentential Decision Diagrams. In IJCAI (Vol. 15, pp. 3141-3148).

Summary: Circuits

Smooth + decomposable circuits: **Linear-time WMC**

+ determinism: **Linear-time most likely assignment**

Obtain circuits using **knowledge compilers (eg SDD)**

Nice circuit libraries:

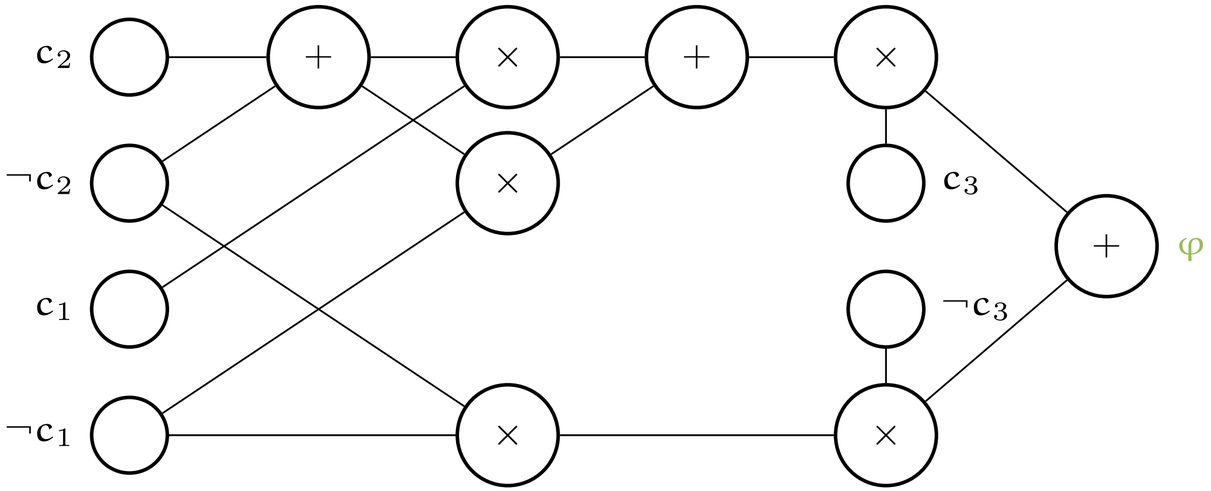
- Cirkit: https://github.com/april-tools/cirkit
- KLay: https://github.com/ML-KULeuven/klay

Summary: Exact inference

Efficient exact inference using:

1. **Simple models, eg conditionally independent**

2. **Knowledge compilation of** Boolean formula into **circuit** on which we answer queries **efficiently**



Nuance: **Beyond independence**

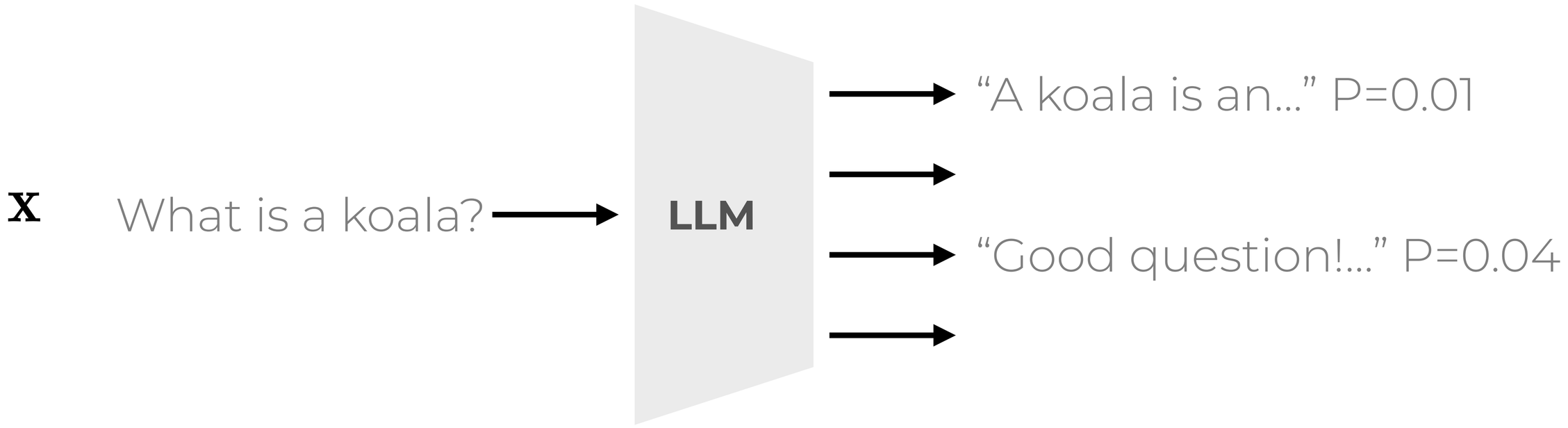
Do we require conditionally independent NN architectures?

No!

Other architectures (like mixtures/ensembles) are also tractable by using *compatible probabilistic circuits*.

But that's out of scope.

Ahmed, K., Teso, S., Chang, K. W., Van den Broeck, G., & Vergari, A. (2022). Semantic probabilistic layers for neuro-symbolic learning. Advances in Neural Information Processing Systems, 35, 29944-29959.



Even with this nuance, exact inference does not apply to LLMs...

But very nice tricks exist!

Zhang, H., Dang, M., Peng, N., & Van den Broeck, G. (2023, July). Tractable control for autoregressive language generation. In International Conference on Machine Learning (pp. 40932-40945). PMLR.
Zhang, H., Kung, P. N., Yoshida, M., Van den Broeck, G., & Peng, N. (2024). Adaptable logical control for large language models. Advances in Neural Information Processing Systems, 37, 115563-115587.
Ahmed, K., Chang, K. W., & Van den Broeck, G. (2023). A pseudo-semantic loss for autoregressive models with logical constraints. Advances in Neural Information Processing Systems, 36, 18325-18340.

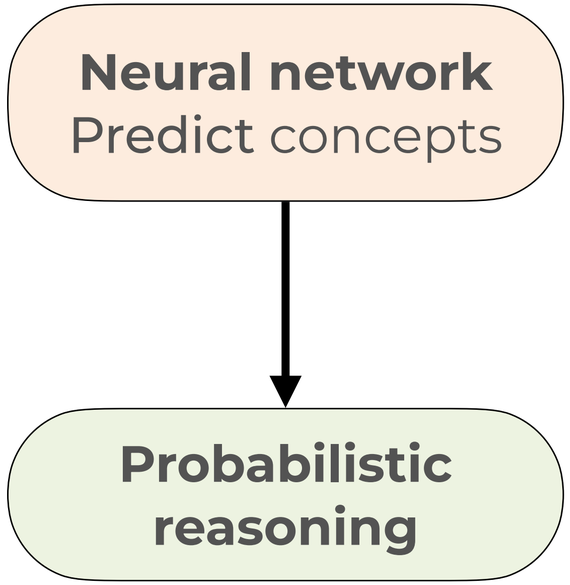
Content

Probabilistic Neurosymbolic methods

- Semantic loss
- Semantic probabilistic layers
- Neurosymbolic predictors
- Why probabilistic reasoning?

Inference

- Circuits & exact inference
- Approximate inference



Approximate inference:
How to *approximate* the weighted model count efficiently & effectively?

Top-k inference

Idea: Instead of taking **all** models for the WMC, only take the **3 best** models and sum those.

c_1	c_2	c_3	$p_\theta(\mathbf{c} \mid \mathbf{x})$
0	0	0	**0.192**
0	0	1	**0.288**
0	1	1	0.072
1	0	1	**0.192**
1	1	1	0.048

$$\varphi = (c_1 \rightarrow c_3) \wedge (c_2 \rightarrow c_3)$$

Top-K inference

Take the **k best** models and sum their probabilities.

Examples:

- **DPLA***: Heuristic search to get k ***most likely*** ‘proofs’ (~models)
- **ABL**: Finds ‘*best*’ concept vector according to heuristic
- **Scallop**

Issue: Gives a **biased** gradient, can lead to poor local optima

And top-k is still NP-hard!

Manhaeve, R., Marra, G., & De Raedt, L. (2021, November). Approximate inference for neural probabilistic logic programming. In Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning (pp. 475-486). IJCAI Organization.

Sampling based inference

Idea: Choose concept vectors randomly based on their probability, and favour those satisfying constraint

c_1	c_2	c_3	$p_\theta(\mathbf{c} \mid \mathbf{x})$
0	0	0	**0.192**
0	0	1	**0.288**
0	1	1	0.072
1	0	1	**0.192**
1	1	1	0.048

$$\varphi = (c_1 \rightarrow c_3) \wedge (c_2 \rightarrow c_3)$$

Sampling-based inference

Sample S concept vectors, then **increase probability** of samples where **constraint is true**.

This is **unbiased**!
And *can* converge smoothly... But...

Issue 1: Often high **variance:** Cannot distinguish noise from signal
Issue 2: Gives a gradient of **0** if $\mathbb{P}_\theta[\varphi \text{ is true} \mid \mathbf{x}]$ is very low:
We never sample concept vecs making constraint true...
No learning!

Sampling-based inference

Issue 1: High **variance**

Advanced methods overcome this (RLOO, CatLog)

Issue 2: Gives a gradient of **0** if $\mathbb{P}_\theta[\varphi \text{ is true} \mid \mathbf{x}]$ is very low

1. Exploit structure in problem
2. Exploit pre-trained models

Kool, W., van Hoof, H., & Welling, M. (2019). Buy 4 reinforce samples, get a baseline for free!.
De Smet, L., Sansone, E., & Zuidberg Dos Martires, P. (2023). Differentiable sampling of categorical distributions using the catlog-derivative trick. Advances in Neural Information Processing Systems, 36, 30416-30428.

Neural network-based inference

Perception neural network

$$p_\theta$$

$$\boldsymbol{\mu}$$

Exact WMC

$$p(\mathbf{y} = 13|\boldsymbol{\mu})$$

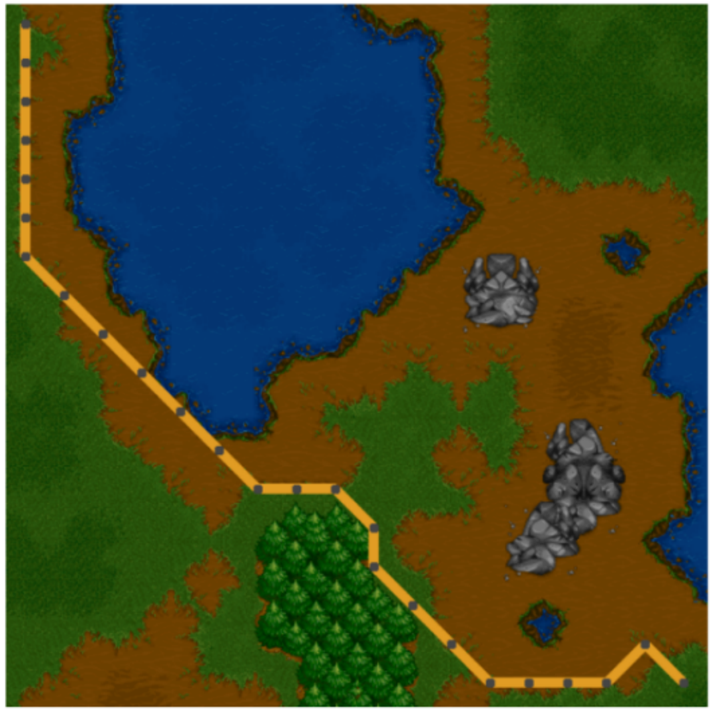
Prediction model

$$q_{\boldsymbol{\Phi}}(\mathbf{y} = 13|\boldsymbol{\mu})$$

A-NeSI

Train another neural net to **approximate** WMC

van Krieken, E., Thanapalasingam, T., Tomczak, J., Van Harmelen, F., & Ten Teije, A. (2023). A-nesi: A scalable approximate method for probabilistic neurosymbolic inference. Advances in Neural Information Processing Systems, 36, 24586-24609.



	12×12	30×30
I-MLE [41]	97.2 ± 0.5	$\mathbf{93.7} \pm \mathbf{0.6}$
RLOO [29]	43.75 ± 12.35	12.59 ± 16.38
A-NeSI	94.57 ± 2.27	17.13 ± 16.32
A-NeSI + RLOO	$\mathbf{98.96} \pm \mathbf{1.33}$	67.57 ± 36.76
$\lvert\mathcal{C}\rvert$	$5^{12\times12}$	$5^{30\times30}$

van Krieken, Emile, et al. "A-nesi: A scalable approximate method for probabilistic neurosymbolic inference." Advances in Neural Information Processing Systems 36 (2023): 24586-24609.

Approximate inference

- **Top-k inference:**
  - Biased, can converge to bad optima, NP-hard
- **Sampling-based:**
  - Unbiased, but bad when probability is low, fast!
- **Neural-network based:**
  - Biased, but fast if neural net can learn problem well

There are more variants and techniques, but that's again another whole course, really :-)

Relevant projects

- **A-NeSI:** Approximate inference using neural nets. NeurIPS 2023

- **BEARS**: Train ensemble for reasoning shortcut awareness. UAI 2024
- **Neurosymbolic diffusion models:** *Scalable* reasoning shortcut awareness using diffusion neural nets. Under review
- More about independence assumptions and relation to reasoning shortcuts. ICML 2024, NeSy 2025
- **RSBench**: Benchmark of problems with shortcuts: NeurIPS 2024

- **ULLER**: Accessible language for practical NeSy research. NeSy 2025
- **Fuzzy operators**: Analysis of fuzzy logic in context of NeSy. AI Journal 2022

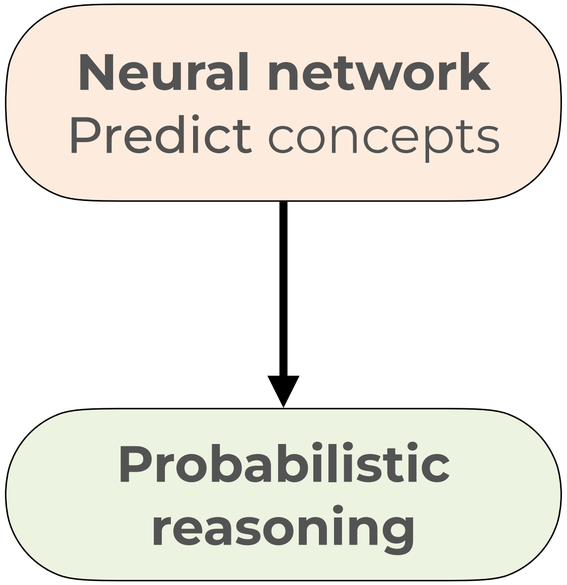
Interested? Do reach out!

Probabilistic Neurosymbolic methods

- Many NeSy methods use WMC
- Nice logical properties!

Inference

- Large literature on exact WMC via circuits
- Approximation: New, many questions open



Contact: e.van.krieken@vu.nl *Bluesky:* @EmilevanKrieken.com

Citations

DeepProbLog: *Manhaeve, R., Dumancic, S., Kimmig, A., Demeester, T., & De Raedt, L. (2018). Deepproblog: Neural probabilistic logic programming. Advances in neural information processing systems, 31.*

Scallop: *Li, Z., Huang, J., & Naik, M. (2023). Scallop: A language for neurosymbolic programming. Proceedings of the ACM on Programming Languages, 7(PLDI), 1463-1487.*

A-NeSI: *van Krieken, E., Thanapalasingam, T., Tomczak, J., Van Harmelen, F., & Ten Teije, A. (2023). A-nesi: A scalable approximate method for probabilistic neurosymbolic inference. Advances in Neural Information Processing Systems*

NeurASP: *Yang, Z., Ishay, A., & Lee, J. (2023). Neurasp: Embracing neural networks into answer set programming*

TensorLog: *Cohen, W. W. (2016). Tensorlog: A differentiable deductive database.*