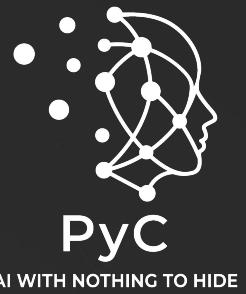
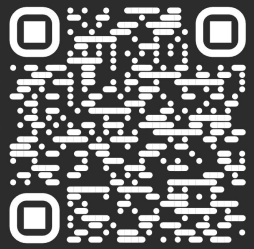
Foundations of Concept-Based Interpretable Deep Learning



Giuseppe Marra & Pietro Barbiero
giuseppe.marra@kuleuven.be pietro.barbiero@ibm.com

In collaboration with: Giovanni de Felice and Mateo Espinosa Zarlenga

ESSAi26



KU LEUVEN IBM Research

fwo Swiss National Science Foundation HASLERSTIFTUNG

Course Outline

I. Interpretability theory

II. Blueprint for interpretable models

+ interpretability in PyTorch

III. Concept representations

IV. Concept-based predictors

Mon 2:00 pm – 3:30 pm

Tue 2:00 pm – 3:30 pm

Course Outline

I. Interpretability theory

II. Blueprint for interpretable models

+ interpretability in PyTorch

III. Concept representations

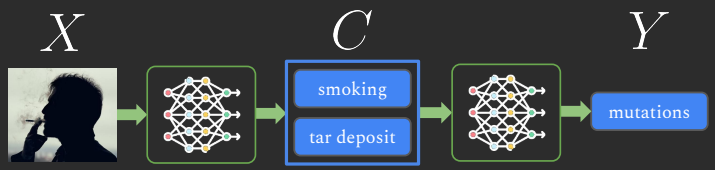
IV. Concept-based predictors

Wed 2:00 pm – 3:30 pm

Fri 2:00 pm – 3:30 pm

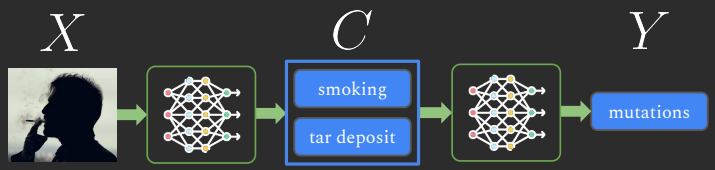
Part 4 of Tutorial:
Task Prediction

Recap: The concept bottleneck model



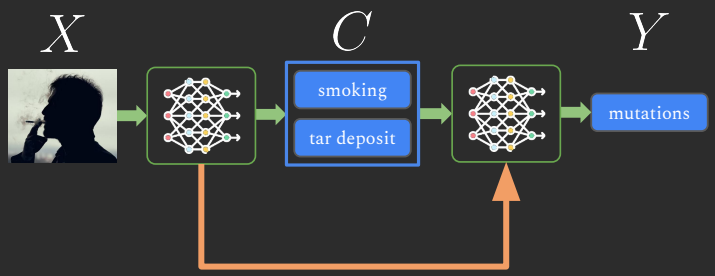
Recap: The concept bottleneck model

Ideal model

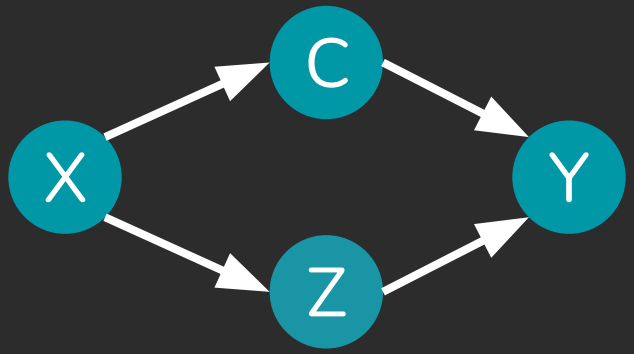


concepts = human-aligned intermediate representation

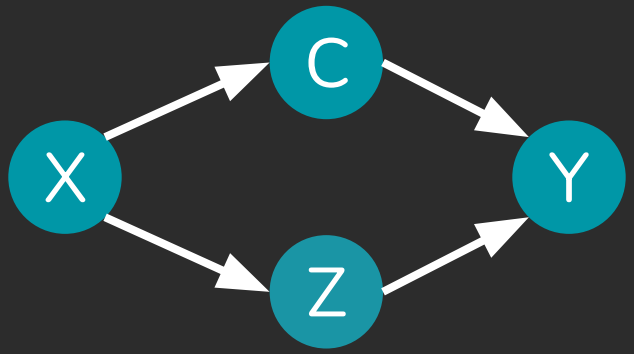
Recap: The concept bottleneck model



residual path = extra non-aligned information



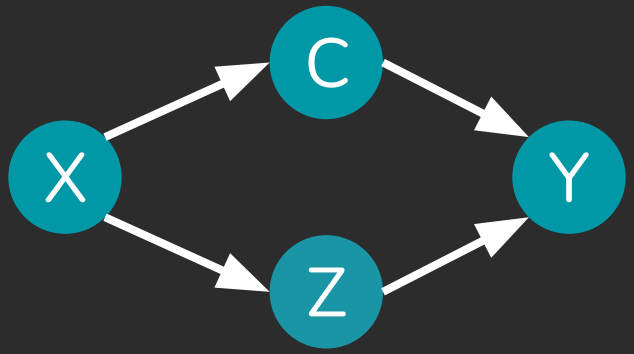
How should we predict task labels P(Y|C, Z)?



How should we predict task labels P(Y|C, Z)?

The diamond diagram

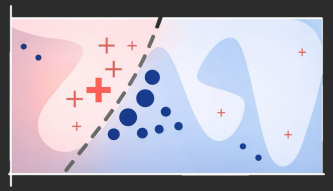
Instantiating the task predictor



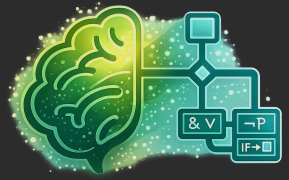
What is P(Z | X)?

What is P(Y | C,Z)?

Task predictors



Locally-interpretable Predictors



Neuro-Symbolic Predictors

Causal Predictors

The concept-bottleneck model

So far, we have assumed that the task predictor P(Y|C) is a logistic regressor

$$p(C = \mathbf{1}|X = \mathbf{x}) = f_\phi(\mathbf{x}) \qquad p(Y = 1|C = \mathbf{c}; \boldsymbol{\theta}, b) = \sigma\left(\boldsymbol{\theta}^\top \mathbf{c} + b\right)$$

The concept-bottleneck model

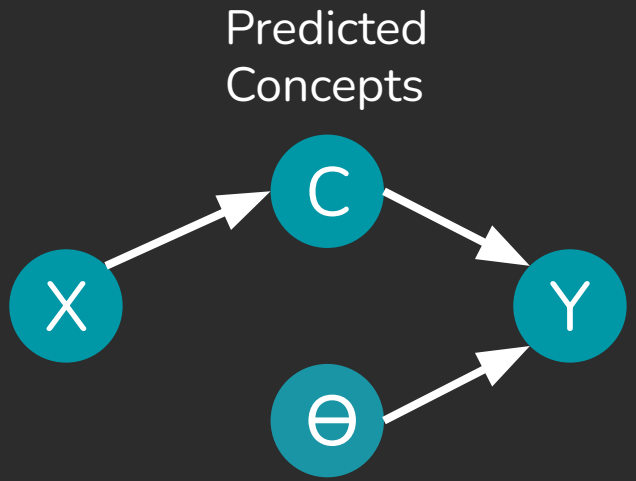
So far, we have assumed that the task predictor P(Y|C) is a logistic regressor

$$p(C = \mathbf{1}|X = \mathbf{x}) = f_\phi(\mathbf{x}) \qquad p(Y = 1|C = \mathbf{c}; \boldsymbol{\theta}, b) = \sigma\left(\boldsymbol{\theta}^\top \mathbf{c} + b\right)$$

In this representation, the parameters Θ and b are implicit in P(Y|C).

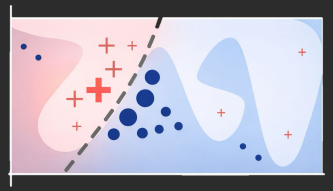
Reify the logistic regressor

$$p(Y = 1|C = \mathbf{c}; \boldsymbol{\theta}, b) = \sigma\left(\boldsymbol{\theta}^\top \mathbf{c} + b\right)$$



Inner Product
+
Activation

Task predictors



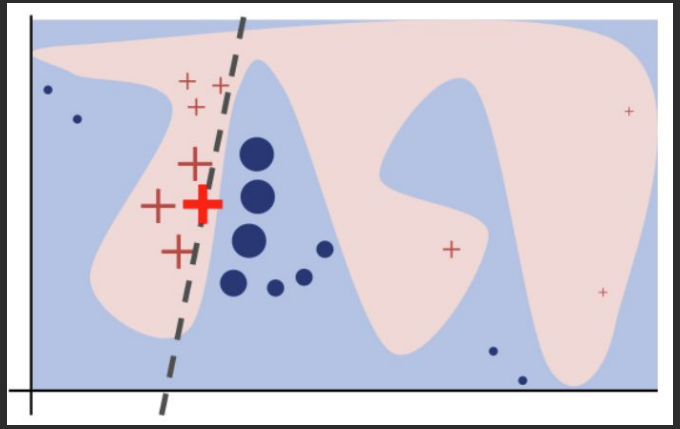
Locally-interpretable Predictors

Neuro-Symbolic Predictors

Causal Predictors

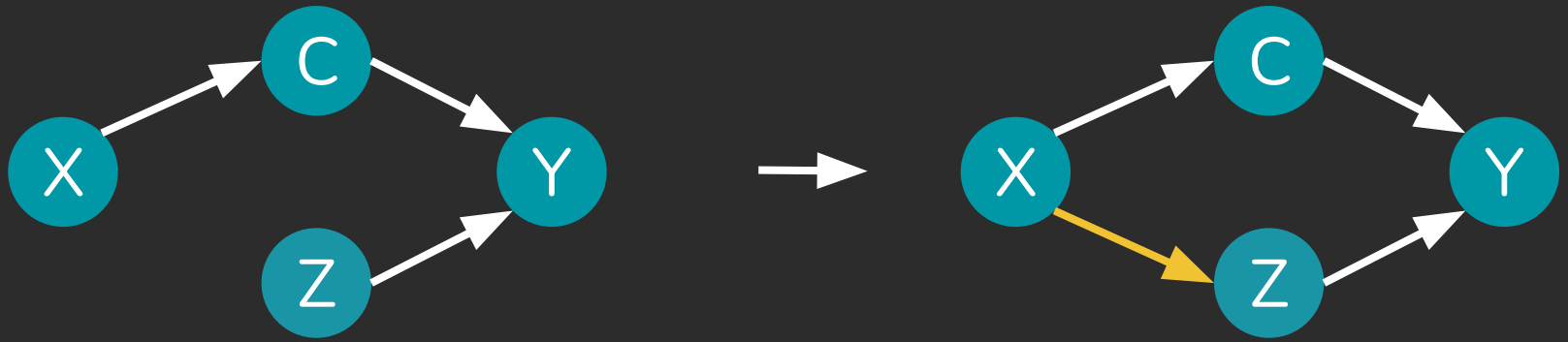
Locally-interpretable task predictors

As a first step, we can construct a nonlinear task predictor $f(\hat{c}) = P(Y \mid \hat{c})$ that behaves "linear-like" in the local neighborhood of $\hat{c}$



[1] Alvarez Melis et al. "Towards robust interpretability with self-explaining neural networks." NeurIPS (2018).
[2] Image adapted from Ribeiro et al. ""Why should Itrust you?" Explaining the predictions of any classifier." KDD (2016).

Locally-interpretable Task predictors



We keep P(Y | C,Z) a linear projection + activation

But we make weights Z a function of the input

Locally-interpretable task predictors

[Local Linearity] The model should

- **behave as a linear classifier**, at least in the neighborhood of a sample's concept space,
- **be locally stable,** small changes in Z should not change the prediction drastically,

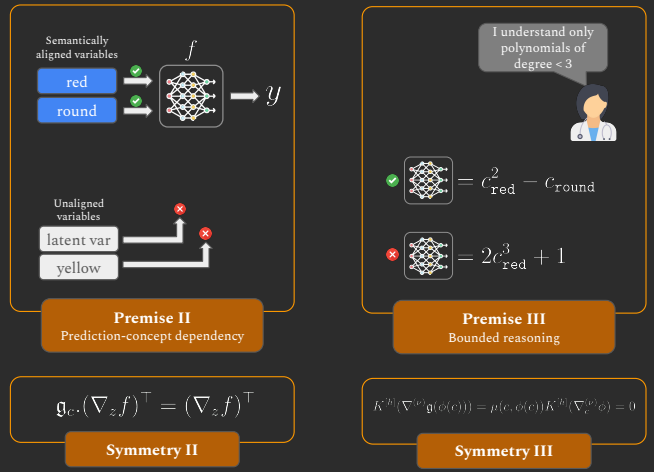
This can be enforced by minimizing a loss that implicitly leads to this condition holding:

$$L(x, y) = \|\nabla_x logit(P(Y = y \mid C(x), Z(x))) - Z(x)^\top J_x C(x)\|_2^2$$

[1] Alvarez Melis et al. "Towards robust interpretability with self-explaining neural networks." NeurIPS (2018).

Locally-interpretable task predictors

$$L(x, y) = \|\nabla_x logit(P(Y = y \mid C(x), Z(x))) - Z(x)^\top J_x C(x)\|_2^2$$



This is akin to a local version of the
Premise II + Premise III

(locally the prediction should only depend on concepts and only in a linear way)

[1] Alvarez Melis et al. "Towards robust interpretability with self-explaining neural networks." NeurIPS (2018).

Problems with locally-linear predictors

Unfortunately, locally-linear predictors cannot:

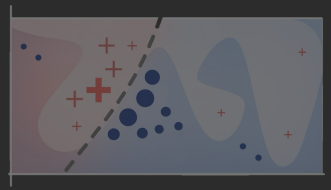
1. Correctly model features with strong (even local) higher order relationships

Problems with locally-linear predictors

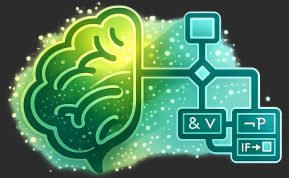
Unfortunately, locally-linear predictors cannot:

1. Correctly model features with strong (even local) higher order relationships
2. Provide insights about global behavior of the predictor

Task Predictors



Locally-interpretable Predictors



Neuro-Symbolic Predictors

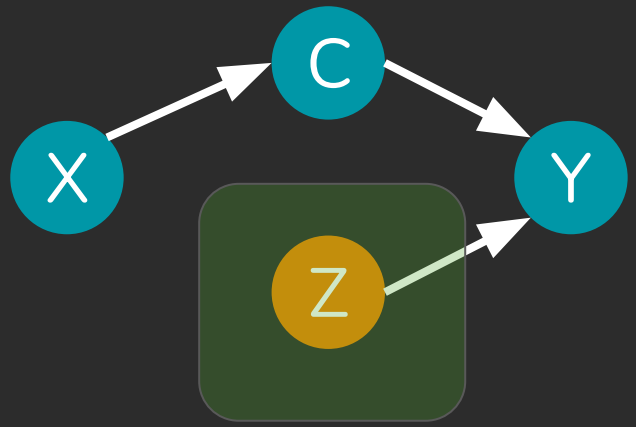
Causal Predictors

What can we use other than linear layers?

Can we use logical formulas over the concepts to design more expressive but still interpretable task predictors?

What can we use other than linear layers?

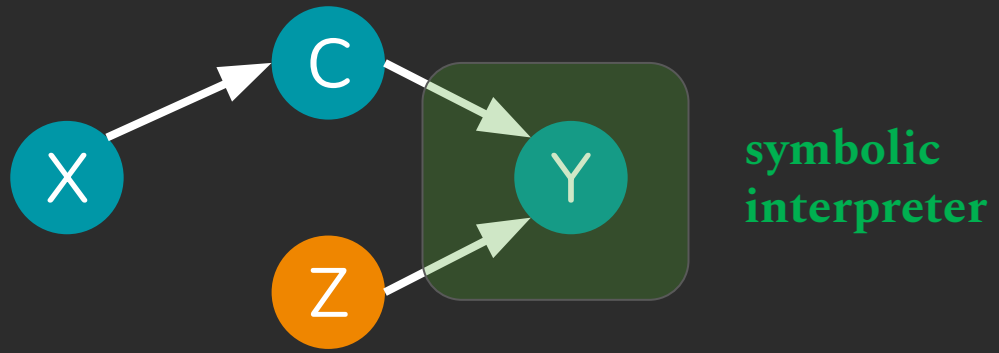
Can we use logical formulas over the concepts to design more expressive but still interpretable task predictors?



(probabilistic) logic rules

What can we use other than linear layers?

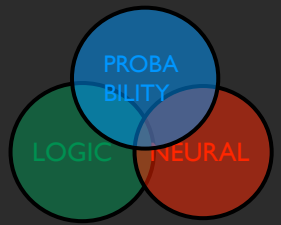
Can we use logical formulas over the concepts to design more expressive but still interpretable task predictors?



What can we use other than linear layers?

Can we use logical formulas over the concepts to design more expressive but still interpretable task predictors?

Neurosymbolic AI



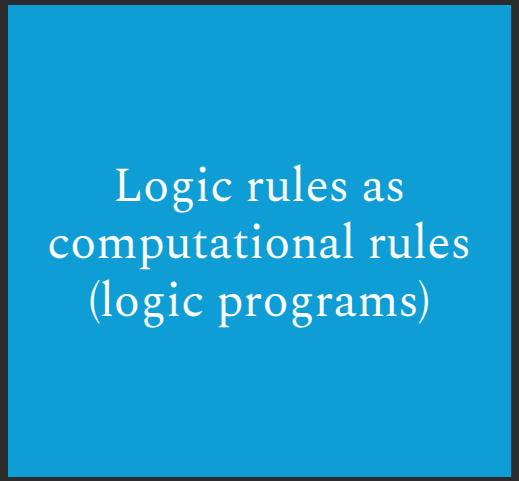
The use of logic: Proof vs Model
Logic

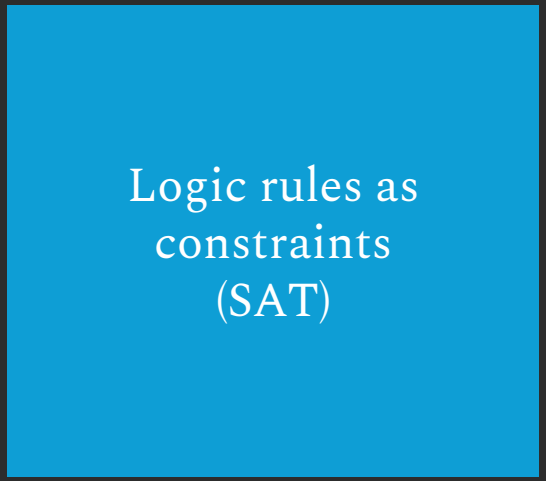
Lion AND **G**ate -> **Z**oo.
Lion AND **W**all -> **Z**oo.

How do we interpret these logic formulas?

The use of logic: Proof vs Model
Logic

Lion AND **Gate** -> **Zoo**.
Lion AND **Wall** -> **Zoo**.





proof-based

model-based

The use of logic: Proof vs Model Logic

R1: Lion AND Gate -> Zoo.
R2: Lion AND Wall -> Zoo.
R3: Lion. **R4:** Gate. **R5:** Wall.

Logic rules as computational rules (logic programs)

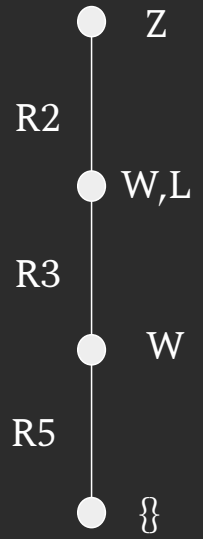
proof-based

The use of logic: Proof vs Model Logic

R1: Lion AND **G**ate -> **Z**oo.
R2: Lion AND **W**all -> **Z**oo.
R3: Lion. **R4:** Gate. **R5:** Wall.

Logic rules as computational rules (logic programs)

proof-based

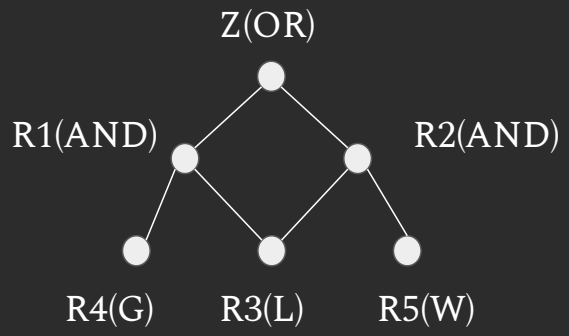


The use of logic: Proof vs Model Logic

R1: Lion AND **G**ate -> **Z**oo.
R2: Lion AND **W**all -> **Z**oo.
R3: Lion. **R4:** Gate. **R5:** Wall.

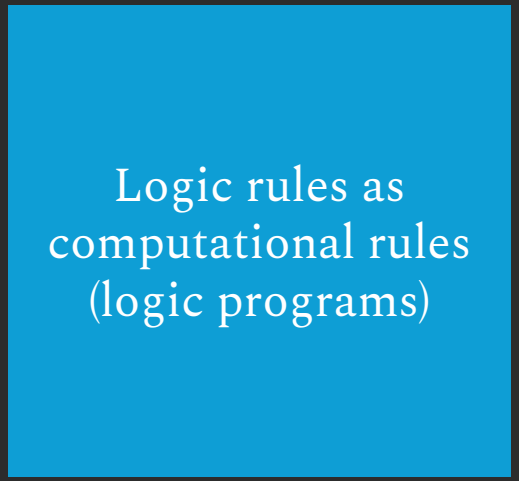
Logic rules as computational rules (logic programs)

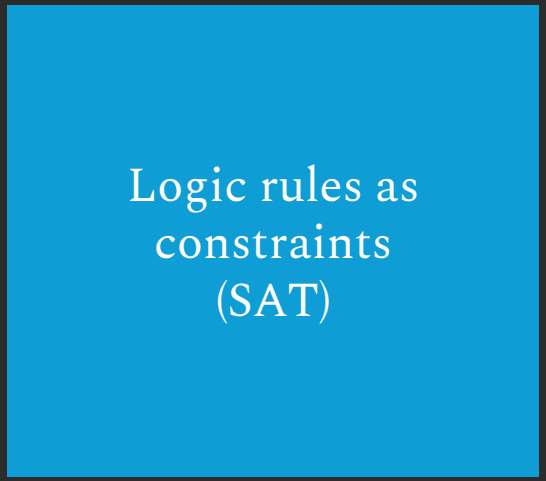
proof-based



The use of logic: Proof vs Model
Logic

Lion AND **Gate** -> **Zoo**.
Lion AND **Wall** -> **Zoo**.



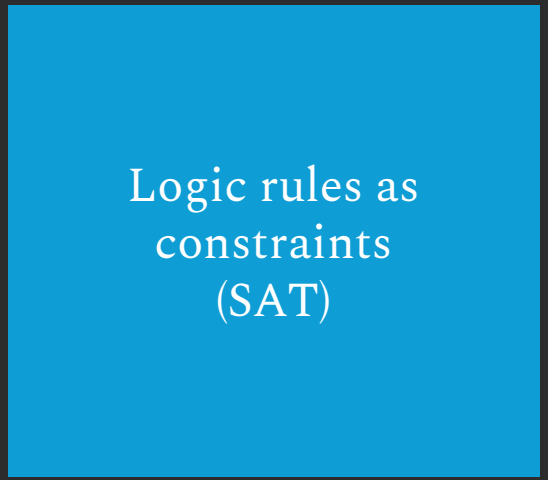


proof-based

model-based

The use of logic: Proof vs Model
Logic

Lion AND **G**ate -> **Z**oo.



model-based

The use of logic: Proof vs Model Logic

Lion AND **G**ate -> **Z**oo.

Logic rules as constraints (SAT)

model-based

L	G	Z	M
F	F	F	T
F	F	T	T
F	T	F	T
F	T	T	T
T	F	F	T
T	F	T	T
T	T	F	F
T	T	T	T

The use of logic: Proof vs Model Logic

Lion AND **G**ate -> **Z**oo.

Logic rules as constraints (SAT)

model-based

L	G	Z	M
F	F	F	T
F	F	T	T
F	T	F	T
F	T	T	T
T	F	F	T
T	F	T	T
T	T	F	F
T	T	T	T

The use of logic: Proof vs Model Logic

Lion AND **G**ate -> **Z**oo.

Logic rules as constraints (SAT)

model-based

L	G	Z	M
F	F	F	T
F	F	T	T
F	T	F	T
F	T	T	T
T	F	F	T
T	F	T	T
T	T	F	F
T	T	T	T

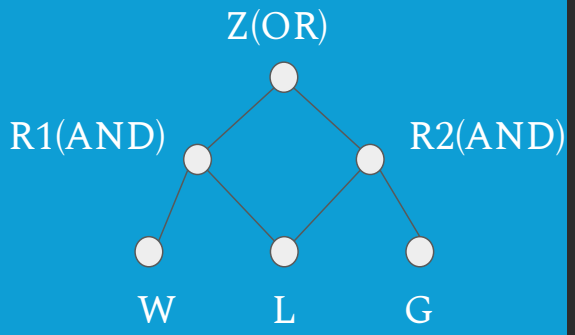
SAT: Does a model exist?

Model Counting: How many of such models?

The use of logic: Proof vs Model Logic

Lion AND **G**ate -> **Z**oo.
Lion AND **W**all -> **Z**oo.

Logic rules as computational rules
(logic programs)



proof-based

Logic rules as constraints
(SAT)

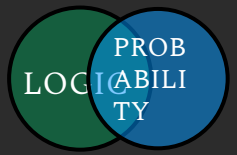
L	G	Z	M
F	F	F	T
F	F	T	T
…	…	…	…

model-based

The use of logic: Proof vs Model
Logic

0.7:: **L**ion AND **G**ate -> **Z**oo.
0.3:: **L**ion AND **W**all -> **Z**oo.

Can we use the same perspective when we deal with uncertainty?

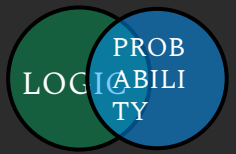


The use of logic: Proof vs Model
StarAI = Logic + Probabilities

0.7:: **L**ion AND **G**ate -> **Z**oo.
0.3:: **L**ion AND **W**all -> **Z**oo.

Stochastic Logic Programs

Markov Logic



proof-based

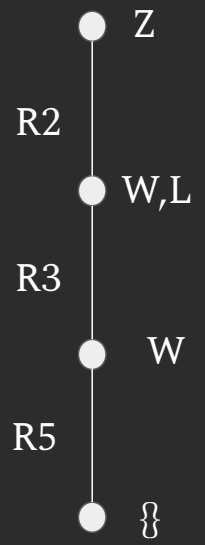
model-based

The use of logic: Proof vs Model
StarAI = Logic + Probabilities

0.7:: **L**ion AND **G**ate -> **Z**oo.
0.3:: **L**ion AND **W**all -> **Z**oo.
1**: L**ion. 1: **G**ate. 1: **W**all.

Stochastic Logic Programs

proof-based



Proof1

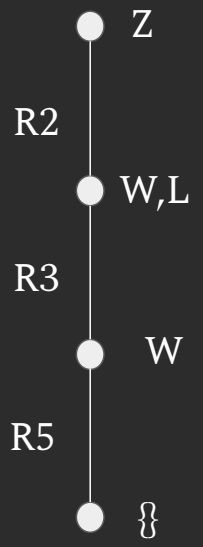
P(proof1) = p(R2) * p(R3) * p (R5)

The use of logic: Proof vs Model
StarAI = Logic + Probabilities

0.7:: **L**ion AND **G**ate -> **Z**oo.
0.3:: **L**ion AND **W**all -> **Z**oo.
1**: L**ion. 1: **G**ate. 1: **W**all.

Stochastic Logic Programs

proof-based



Proof1

P(proof1) = p(R2) * p(R3) * p (R5)

Semantics:

Probability distribution over proofs

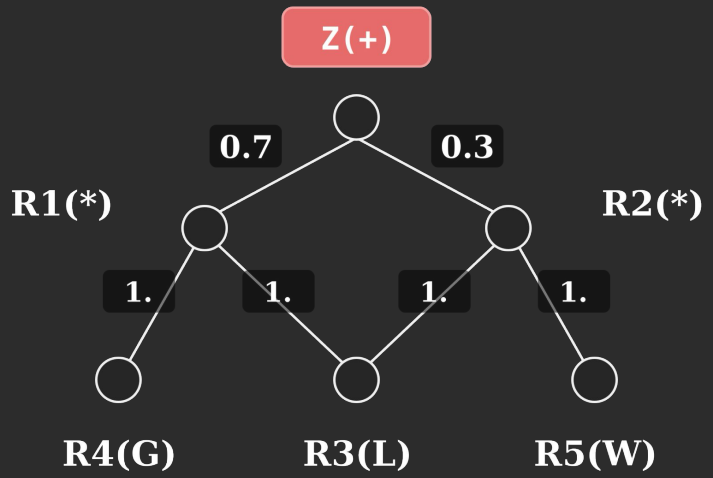
(akin to probabilistic grammars)

The use of logic: Proof vs Model
StarAI = Logic + Probabilities

0.7:: Lion AND Gate -> Zoo.
0.3:: Lion AND Wall -> Zoo.
1: Lion. 1: Gate. 1: Wall.

Stochastic Logic Programs

proof-based



The use of logic: Proof vs Model
StarAI = Logic + Probabilities

0.7:: Lion and Gate and Cue(X) -> ZooGivenCue(X).

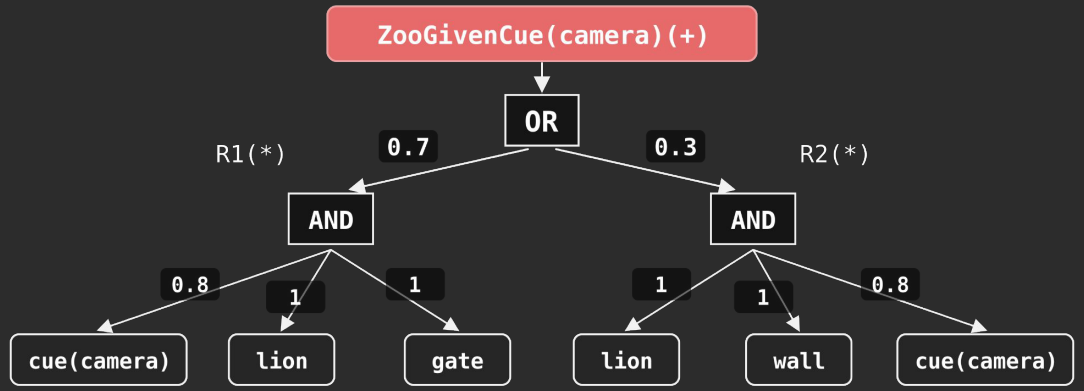
0.3:: Lion and Wall and Cue(X) -> ZooGivenCue(X).

1.0:: Lion. 1.0:: Gate. 1.0:: Wall.

0.8:: Cue(camera). 0.2:: Cue(rock).

Stochastic Logic Programs

proof-based



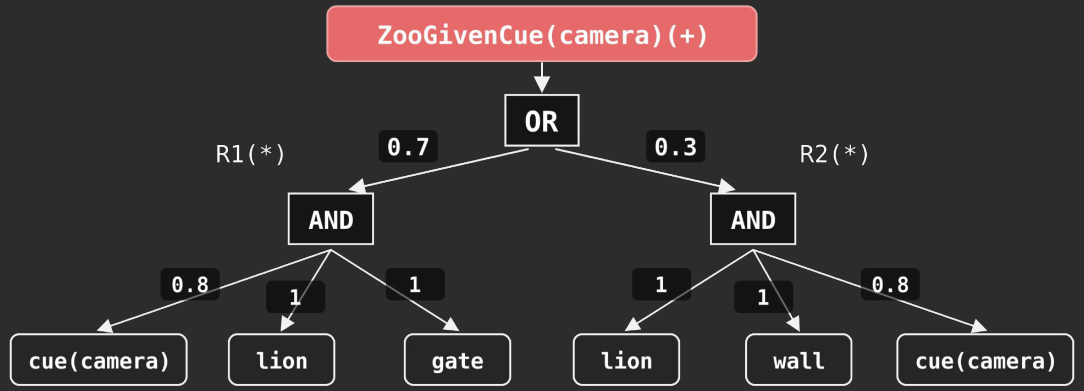
The use of logic: Proof vs Model
StarAI = Logic + Probabilities

0.7:: Lion and Gate and Cue(X) -> ZooGivenCue(X).
0.3:: Lion and Wall and Cue(X) -> ZooGivenCue(X).
1.0:: Lion. 1.0:: Gate. 1.0:: Wall.

0.8:: Cue(camera). 0.2:: Cue(rock).

Stochastic Logic Programs

proof-based



AND -> x, OR -> +
inference = post-order traversal

The use of logic: Proof vs Model
StarAI = Logic + Probabilities

0.7:: Lion and Gate and Cue(X) -> ZooGivenCue(X).

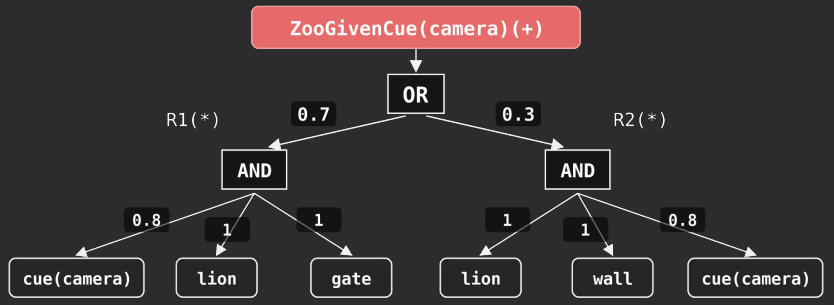
0.3:: Lion and Wall and Cue(X) -> ZooGivenCue(X).

1.0:: Lion. 1.0:: Gate. 1.0:: Wall.

0.8:: Cue(camera). 0.2:: Cue(rock).

Stochastic Logic Programs

proof-based



non terminal rules

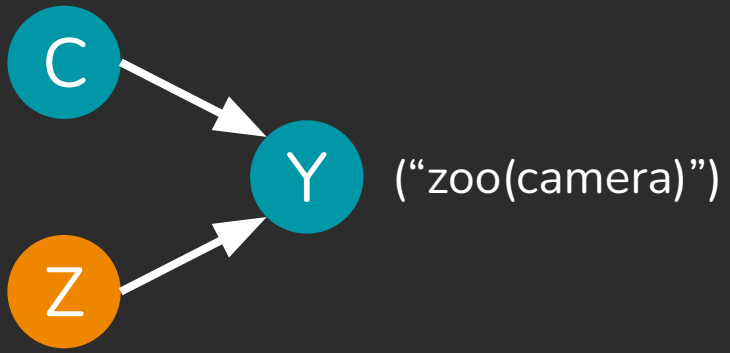
terminal rules

The use of logic: Proof vs Model
StarAI = Logic + Probabilities

Stochastic Logic Programs

proof-based

terminal rules = concepts



non terminal rules

The use of logic: Proof vs Model
StarAI = Logic + Probabilities

Stochastic Logic Programs

proof-based

terminal rules = concepts

non terminal rules

symbolic evaluation of the parse tree

The use of logic: Proof vs Model
StarAI = Logic + Probabilities

w1:: **L**ion AND **G**ate -> **Z**oo.
w2:: **L**ion AND **W**all -> **Z**oo.

Markov Logic

model-based

L	W	G	Z	W
F	F	F	F	w1 + w2
F	F	F	T	...
F	F	T	F	...
F	F	T	T	...
...	...	...		...
T	T	T	F	0 + 0

The use of logic: Proof vs Model
StarAI = Logic + Probabilities

w1:: **L**ion AND **G**ate -> **Z**oo.
w2:: **L**ion AND **W**all -> **Z**oo.

Markov Logic

model-based

L	W	G	Z	W
F	F	F	F	w1 + w2
F	F	F	T	...
F	F	T	F	...
F	F	T	T	...
...	...	...		...
T	T	T	F	0 + 0

Weight of a model =
sum of the weights
of the formulas it
makes True

$$p(m) = \frac{e^W}{Z}$$

The use of logic: Proof vs Model
StarAI = Logic + Probabilities

w1:: **L**ion AND **G**ate -> **Z**oo.
w2:: **L**ion AND **W**all -> **Z**oo.

Markov Logic

model-based

L	W	G	Z	W
F	F	F	F	w1 + w2
F	F	F	T	...
F	F	T	F	...
F	F	T	T	...
...	...	...		...
T	T	T	F	0 + 0

Weight of a model = sum of the weights of the formulas it makes True

$$p(m) = \frac{e^W}{Z}$$

Semantics:

Probability distribution over interpretations

The use of logic: Proof vs Model
StarAI = Logic + Probabilities

w1:: **L**ion AND **G**ate -> **Z**oo.
w2:: **L**ion AND **W**all -> **Z**oo.

Markov Logic

model-based

L	W	G	Z	W
F	F	F	F	p(m1)
F	F	F	T	p(m2)
F	F	T	F	p(m3)
F	F	T	T	p(m4)
...	...	...		...
T	T	T	F	0 + 0

q = G and not W

p(q) = weighted model count

$$\sum_m p(m)q(m)$$

The use of logic: Proof vs Model
StarAI = Logic + Probabilities

w1:: **L**ion AND **G**ate -> **Z**oo.
w2:: **L**ion AND **W**all -> **Z**oo.

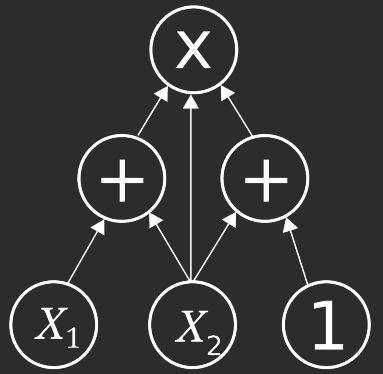
Markov Logic

model-based

$$\sum_m p(m)q(m)$$

Weighted Model Counting

Knowledge Compilation



Arithmetic Circuits

The use of logic: Model ProbLog

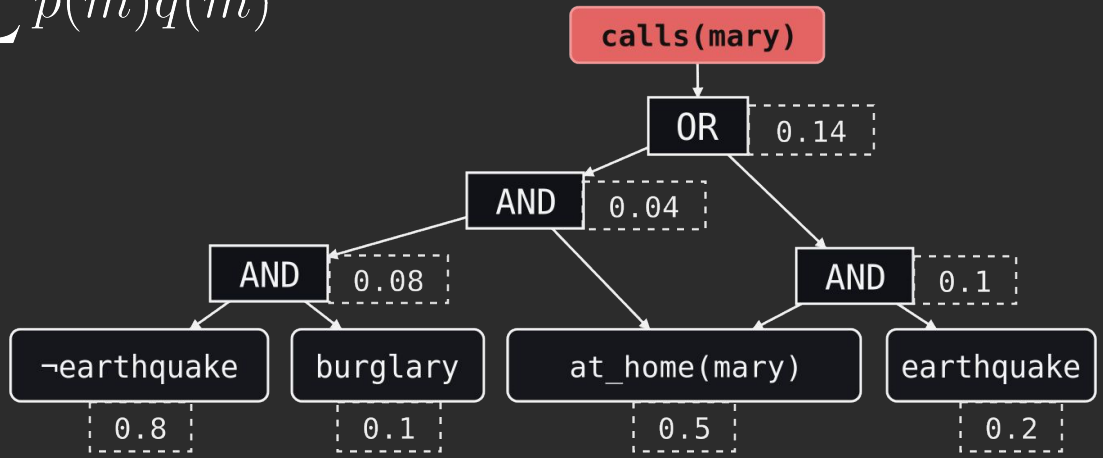
##Query:
query(calls(mary)).

##Rules
alarm :- earthquake.
alarm :- burglary.
calls(X):-alarm,at_home(X).

##Weights for models
0.2::earthquake.
0.1::burglary.
0.5::at_home(mary).
0.4::at_home(john).

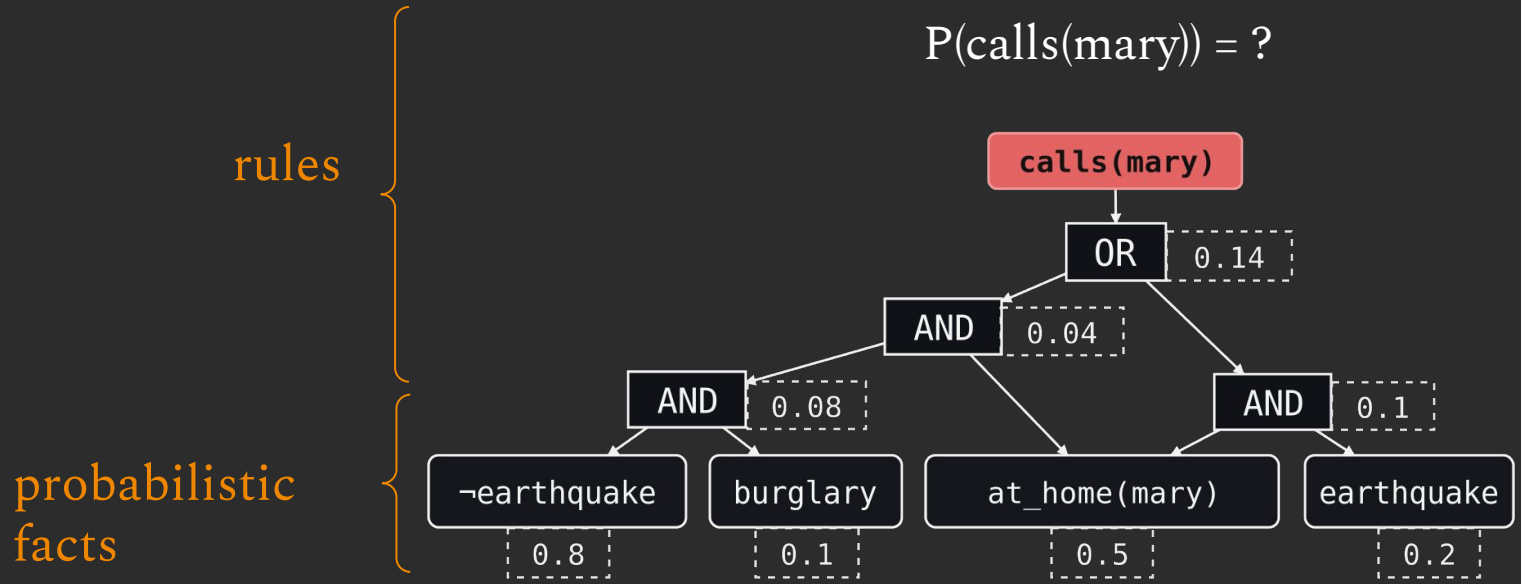
$$\sum_m p(m)q(m)$$

P(calls(mary)) = ?



[1] Manhaeve, Robin, et al. "Deepproblog: Neural probabilistic logic programming." NeurIPS (2018).

The use of logic: Model ProbLog



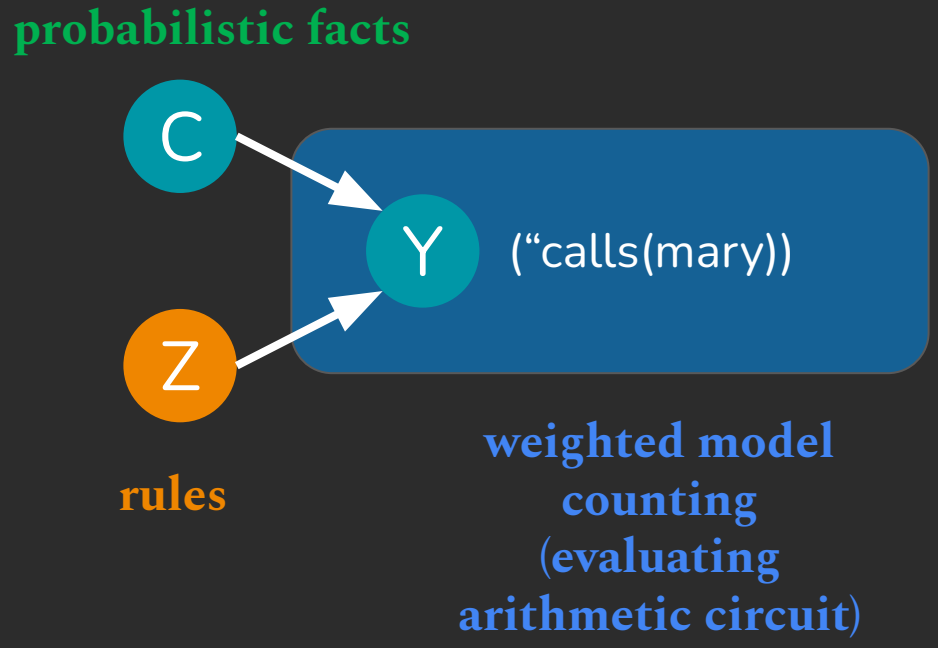
[1] Manhaeve, Robin, et al. "Deepproblog: Neural probabilistic logic programming." NeurIPS (2018).

The use of logic: Proof vs Model
StarAI = Logic + Probabilities

##Query:
query(calls(mary)).

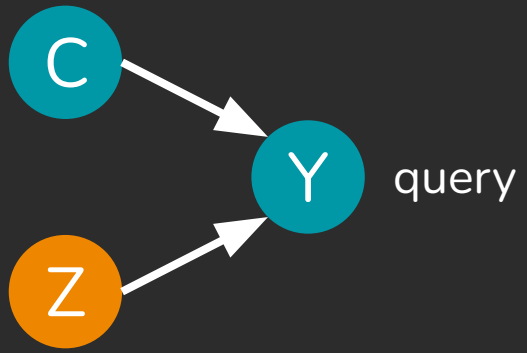
##Rules
alarm :- earthquake.
alarm :- burglary.
calls(X):-alarm,at_home(X).

##Weights for models
0.2::earthquake.
0.1::burglary.
0.5::at_home(mary).
0.4::at_home(john).



The use of logic: Proof vs Model
StarAI = Logic + Probabilities

terminal rules / facts



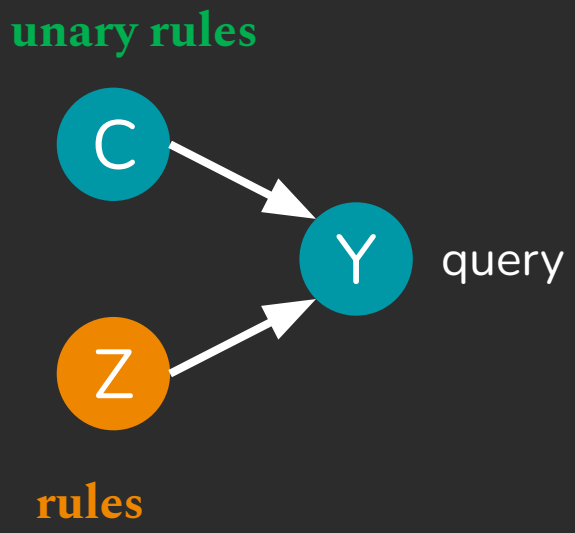
(non-terminal) rules

The distinction unary/n-ary is only superficial...

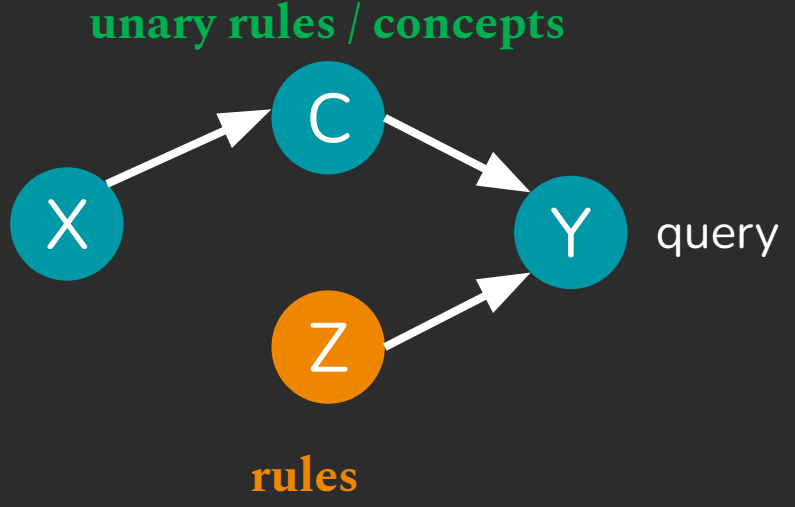
...but useful for the parallel with NeSy and CBMs

NeSy = neural reparameterization of StarAI

StarAI



NeSy (CBMs)

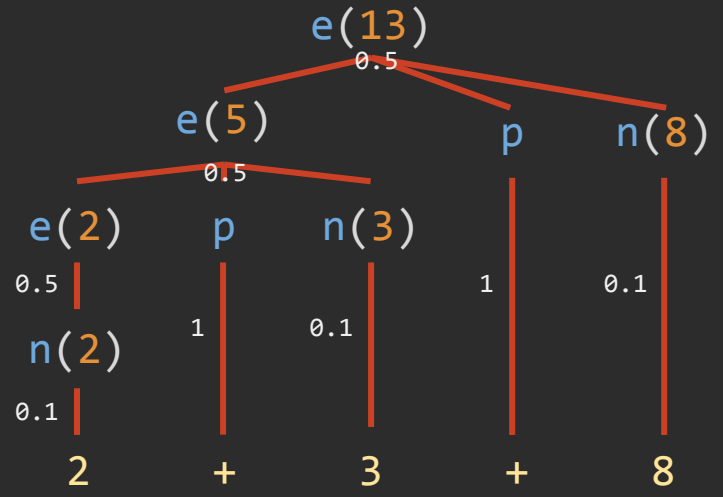


From StarAI …
Stochastic Definite Clause Grammars

Parse Sequences: [“2”, “+”, “3”, “+”, “8”].

```
0.5 :: e(N) --> n(N).
0.5 :: e(N) --> e(N1), p, n(N2),
                {N is N1 + N2}.
1.0 :: p    --> [“+”].

0.1 :: n(0) --> [“0”].
0.1 :: n(1) --> [“1”].
       …
0.1 :: n(9) --> [“9”].
```



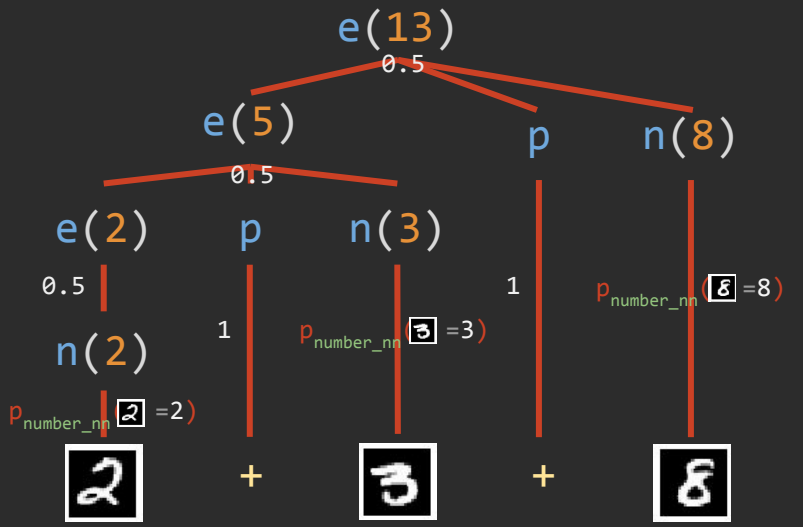
From StarAI to NeSy/CBM

DeepStochLog

Parse Sequences: [ , “+”,  , “+”,  .

```
0.5 :: e(N) --> n(N).
0.5 :: e(N) --> e(N1), p, n(N2),
                {N is N1 + N2}.
1:::: p --> [“=”].

nn(  ,0):: n(0) --> [  ].
nn(  ,1):: n(1) --> [  ].
        ...
nn(  ,9):: n(9) --> [  ].
```



[1] Winters et al, DeepStochLog: Neural Stochastic Logic Programming, AAAI 2022

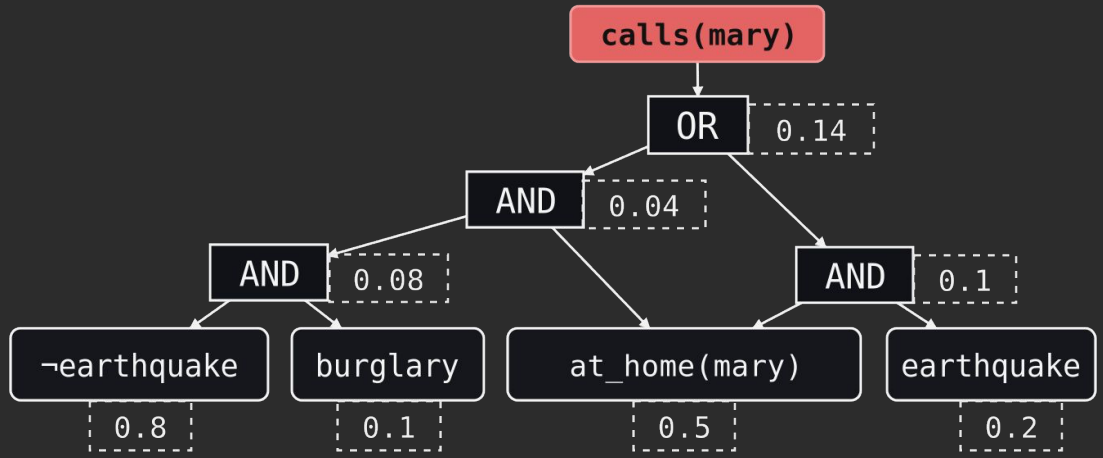
From StarAI ….
ProbLog

##Query:
query(calls(mary)).

##Rules
alarm :- earthquake.
alarm :- burglary.
calls(X):-alarm,at_home(X).

##Weights for models
0.2::earthquake.
0.1::burglary.
0.5::at_home(mary).
0.4::at_home(john).

P(calls(mary)) = ?



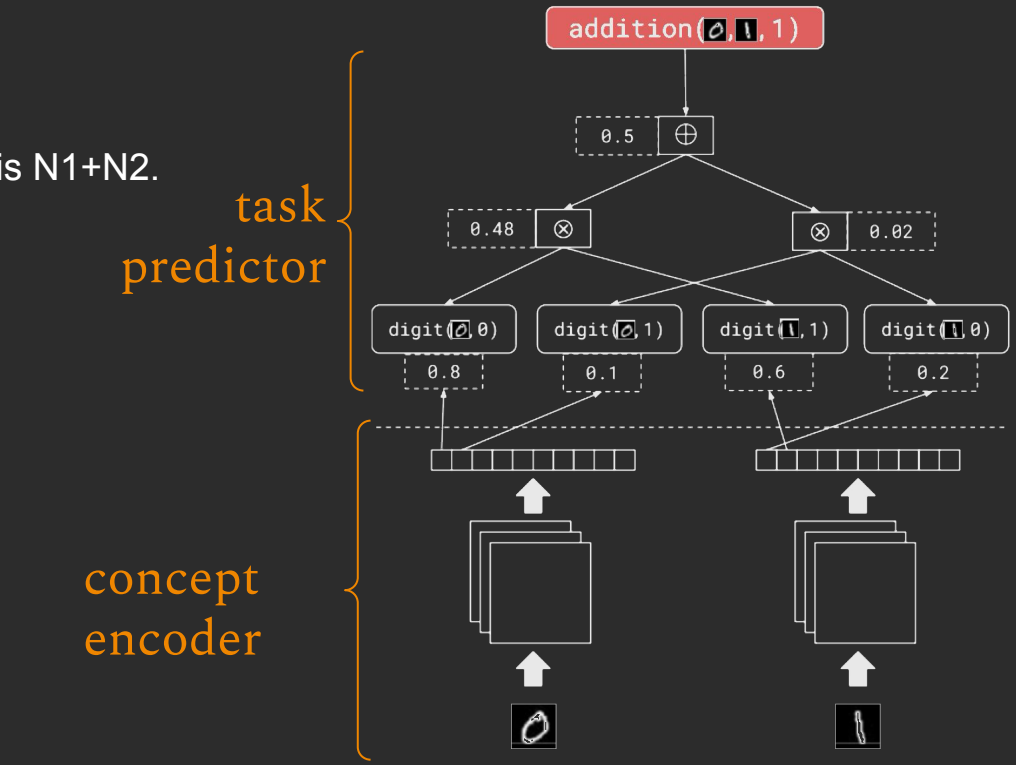
$$\sum_m p(m)q(m)$$

[1] Manhaeve, Robin, et al. "Deepproblog: Neural probabilistic logic programming." NeurIPS (2018).

From StarAI to NeSy / CBM
DeepProbLog

addition(X,Y,Z) :- digit(X,N1), digit(Y,N2), Z is N1+N2.

nn(m_digit, [X], Y, [0...9]) :: digit(X,Y).



[1] Manhaeve, Robin, et al. "Deepproblog: Neural probabilistic logic programming." NeurIPS (2018).

From StarAI to NeSy CBMs

w1:: Lion AND Gate -> Zoo.
w2:: Lion AND Wall -> Zoo.
w3()::Lion.
w4()::Wall.
w5()::Zoo.

```
humid(Data) ~ bernoulli(hum_det(Data)).
temp(Data, T) ~ normal(temp_pred(Data)).
good_weather(Data):- humid(Data) =:= 1,
                     temp(Data) < 0.
good_weather(Data):- humid(Data) =:= 0,
                     temp(Data) > 15.
```

```
model = Model()
n = given.shape[0]

for i in range(n):
    model += AllDifferent([puzzle[i,j] for j in range(n)])
    model += AllDifferent([puzzle[j,i] for j in range(n)])
```

Relational Neural Machines

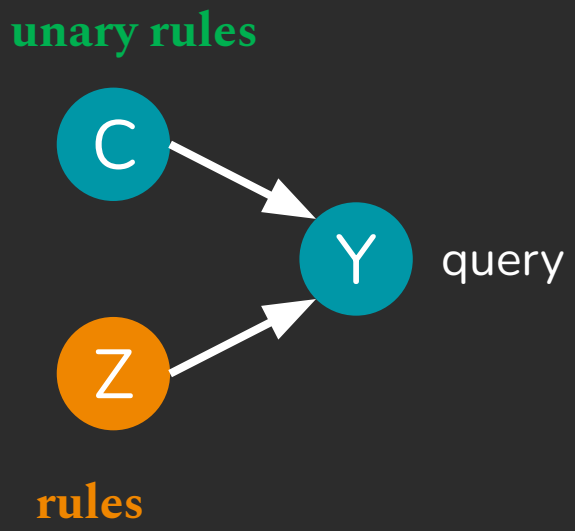
DeepSeaProbLog

Predict-and-Optimize

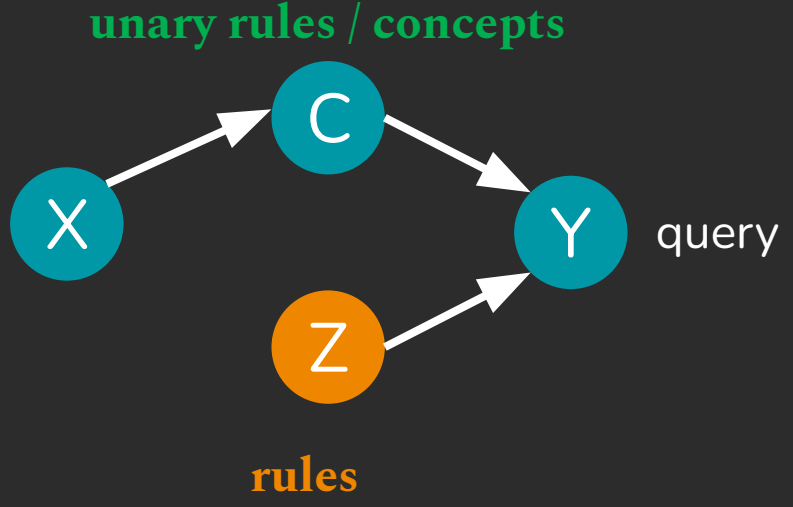
[1] Marra et al, Relational Neural Machines, ECAI 2020
[2] De Smet et al, Neural Probabilistic Logic Programming in Discrete-Continuous Domains, UAI 2023
[3] Mandi et al, Decision-focused learning: Foundations, state of the art, benchmark and future opportunities. JAIR 2024

NeSy = neural reparameterization of StarAI

StarAI

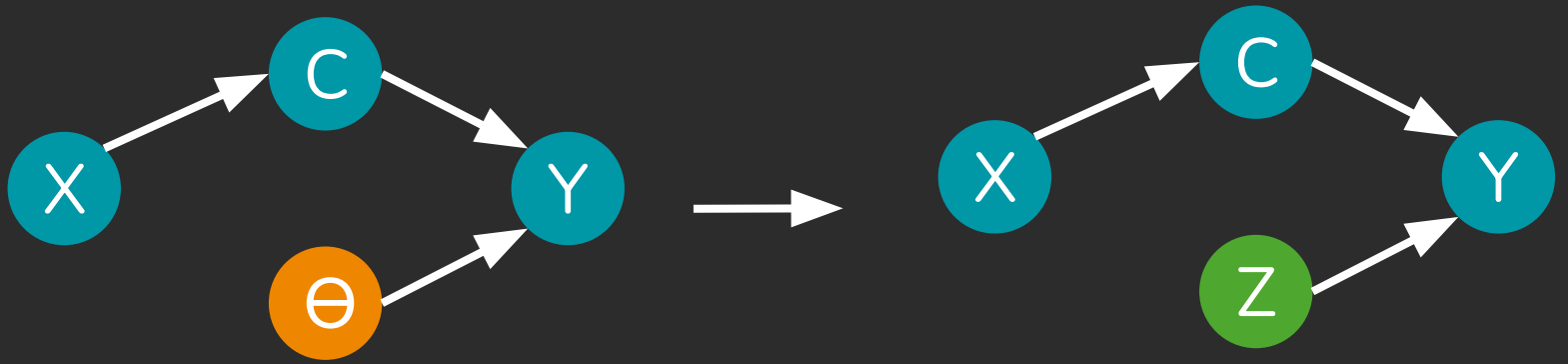


NeSy / CBMs



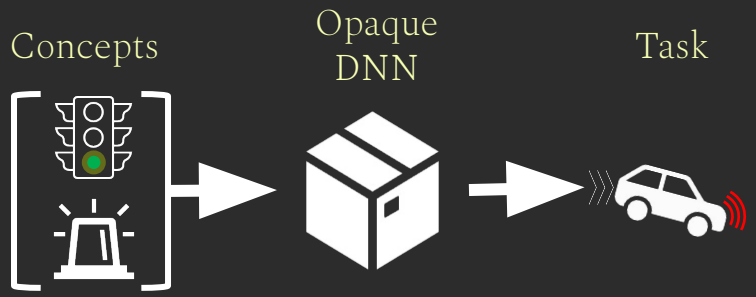
What if we do not know the logical formulas?

Start with a (deep) CBM, then extract rules post-hoc

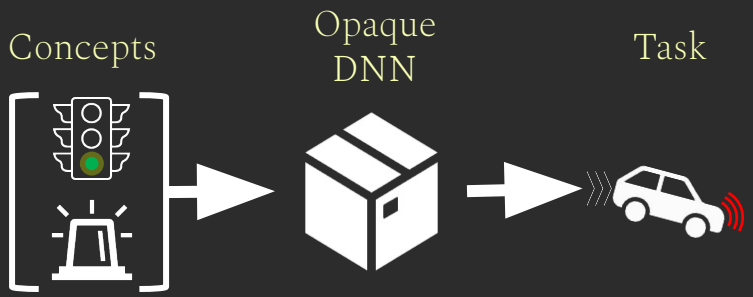


Start with a CBM, then extract rules post-hoc

Imagine we have all the time in the world to intervene on the inputs of DNN task predictor



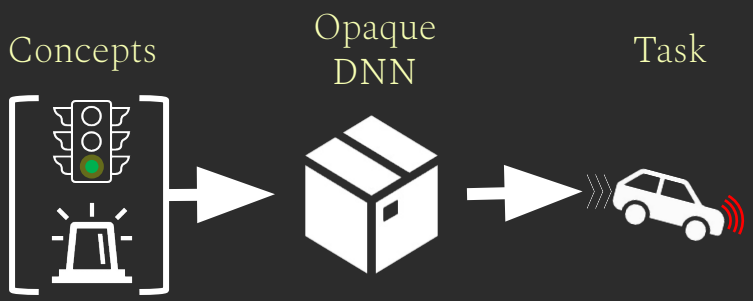
Going beyond linear predictors

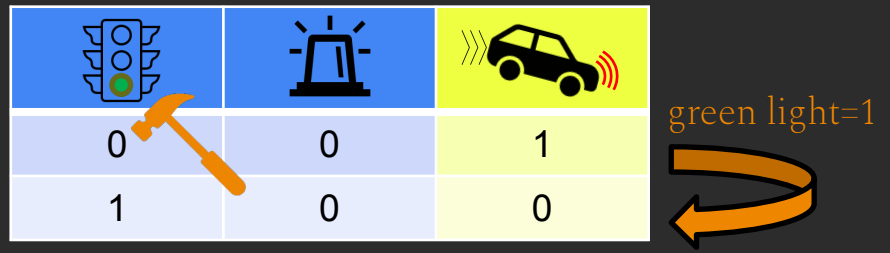


0	0	1

Going beyond linear predictors

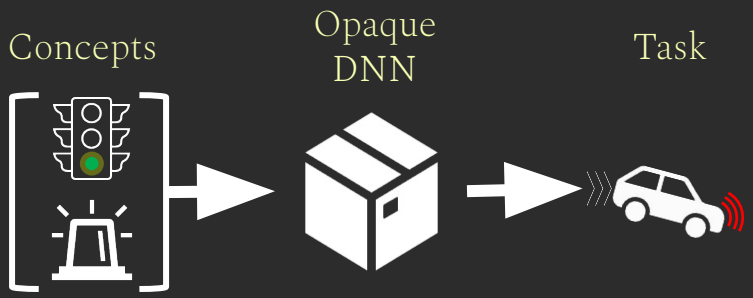
Imagine we have all the time in the world to intervene on the inputs of a DNN predicting task labels from the concepts:





Going beyond linear predictors

Imagine we have all the time in the world to intervene on the inputs of a DNN predicting task labels from the concepts:

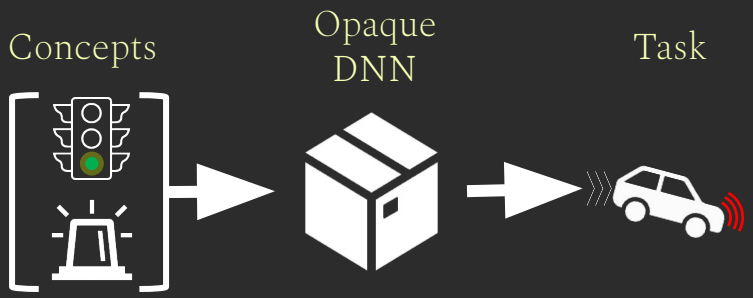


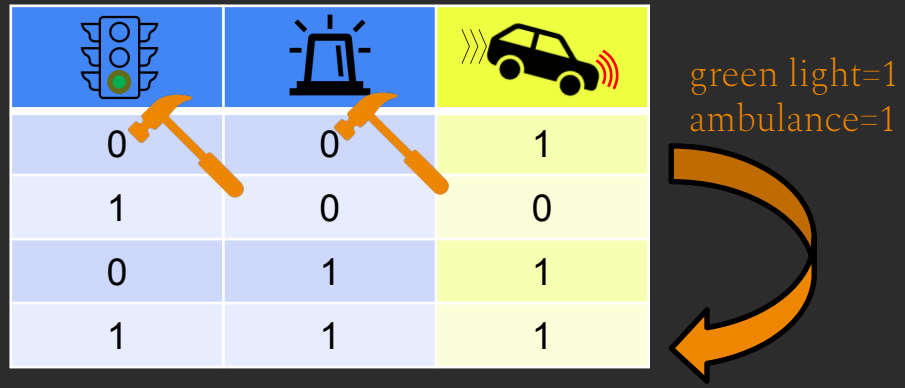
0	0	1
1	0	0
0	1	1

ambulance=1

Going beyond linear predictors

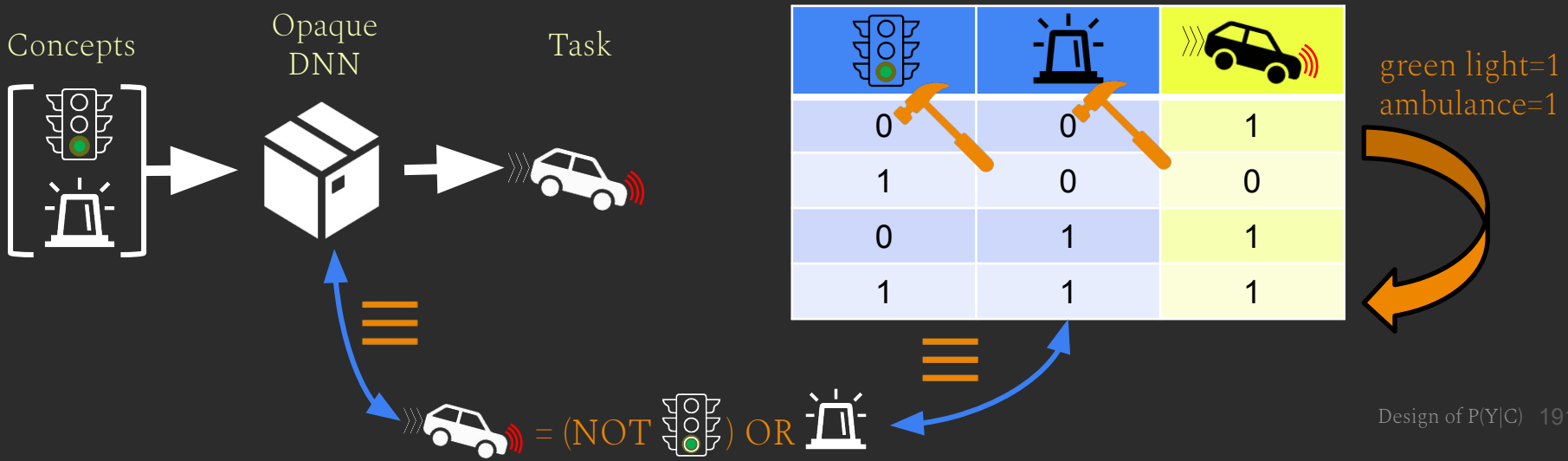
Imagine we have all the time in the world to intervene on the inputs of a DNN predicting task labels from the concepts:





Going beyond linear predictors

Imagine we have all the time in the world to intervene on the inputs of a DNN predicting task labels from the concepts:



Going beyond linear predictors

Idea: Can we use this approach to construct a truth table/logic rule mapping concepts to downstream tasks?

This would allow us to capture non-linear relationships between concepts using logic rules we can understand!

Going beyond linear predictors

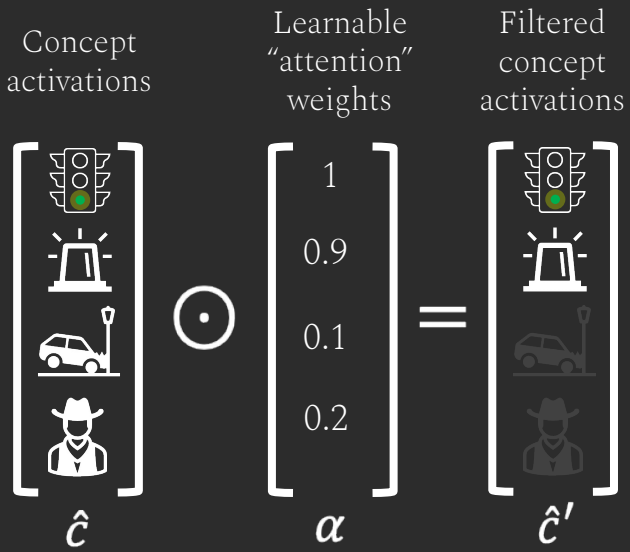
Idea: Can we use this approach to construct a truth table/logic rule mapping concepts to downstream tasks?

Problem: # of interventions required for this is exponential on the # of concepts!

Efficiently constructing logic-based predictors

One approach for efficiently construct a logic-based predictor is:

Step 1: Filter concept activations using learnable *attention weights* α

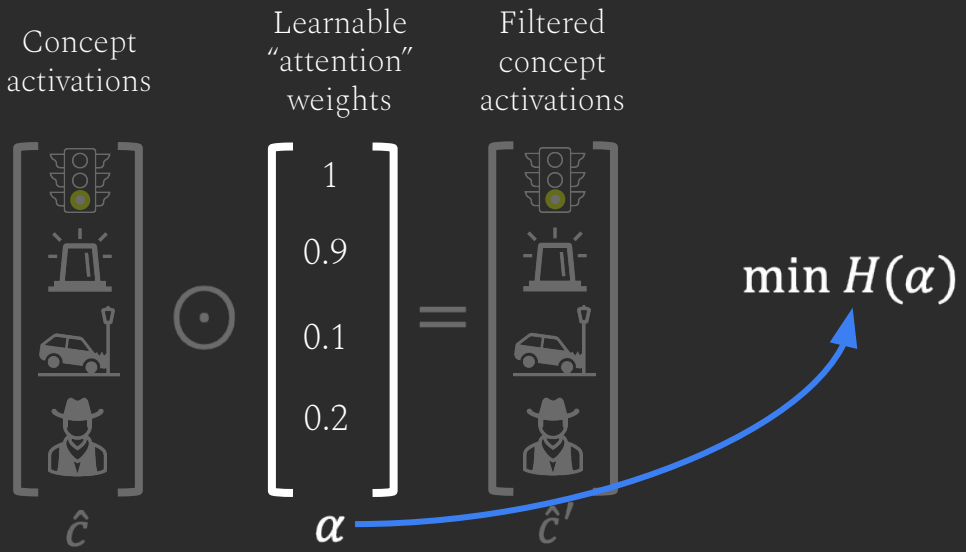


[1] Barbiero et al. "Entropy-based logic explanations of neural networks." AAAI (2022).

Efficiently constructing logic-based predictors

One approach for efficiently construct a logic-based predictor is:

Step 2: Select a small number of concepts by minimizing the entropy of the attention weights α.

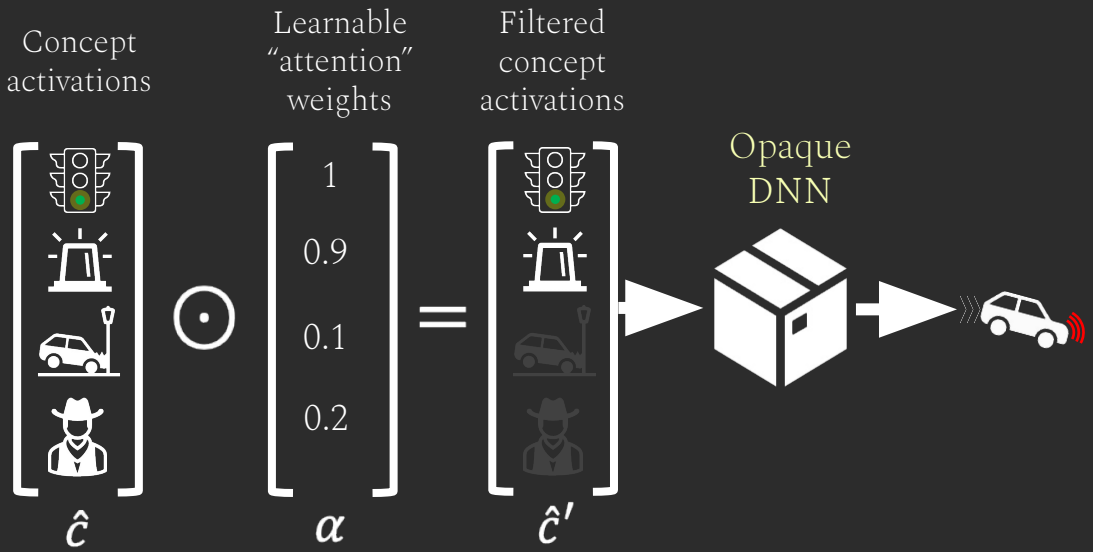


[1] Barbiero et al. "Entropy-based logic explanations of neural networks." AAAI (2022).

Efficiently constructing logic-based predictors

One approach for efficiently construct a logic-based predictor is:

Step 3: Solve task with the (small) set of selected concepts

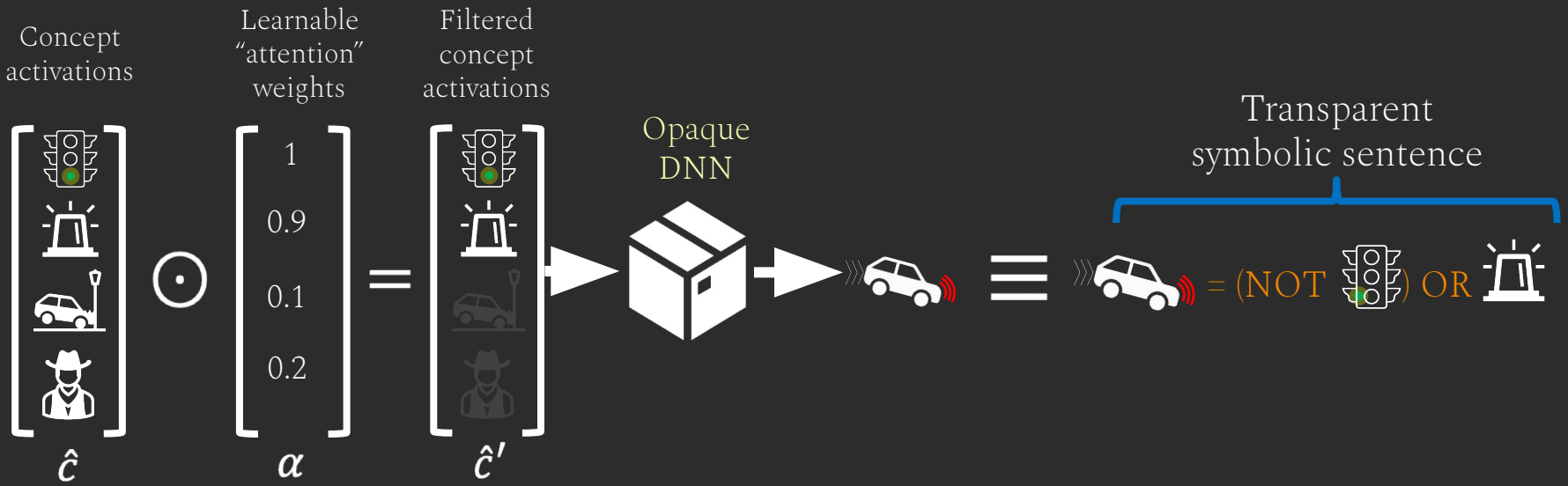


[1] Barbiero et al. "Entropy-based logic explanations of neural networks." AAAI (2022).

Efficiently constructing logic-based predictors

One approach for efficiently construct a logic-based predictor is:

Step 4: derive an explanation in DNF from the (empirical and much smaller) truth table

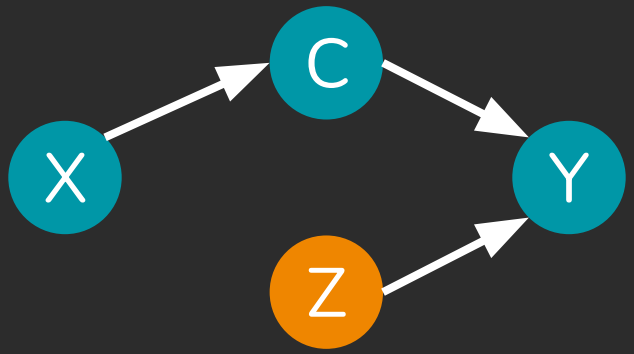


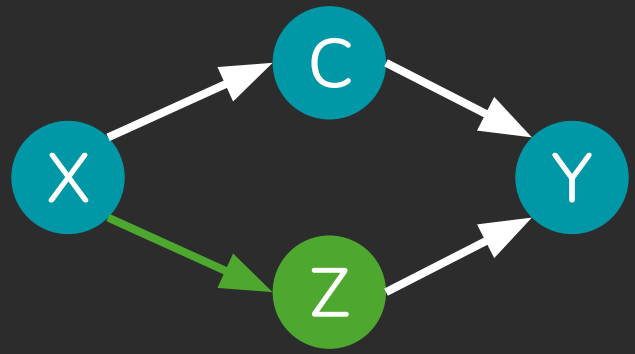
[1] Barbiero et al. "Entropy-based logic explanations of neural networks." AAAI (2022).

Deep Concept Reasoners

So far, the logic rules were a prior: they are the same for any input X

We can also make the rules a posterior, i.e., depend on the input X



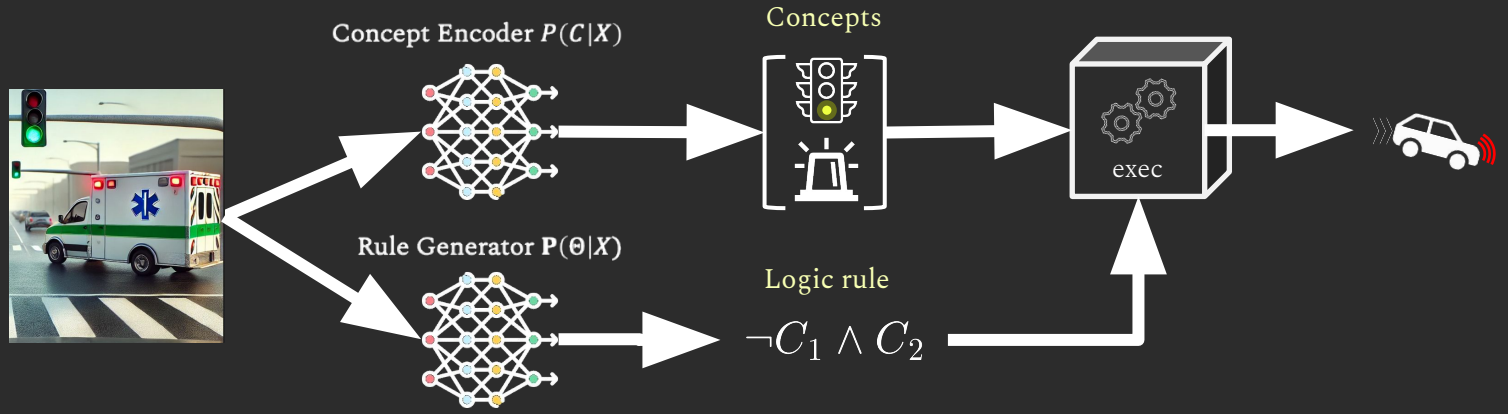


Neuro-symbolic task predictors

ICML 23 NeurIPS 24

Generally, such NeSy predictor are constructed as follows:

Step 1: A DNN generates both concept activations & rule parameters (neural generation)
Step 2: A symbolic engine executes rules using predicted concepts (interpretable execution)



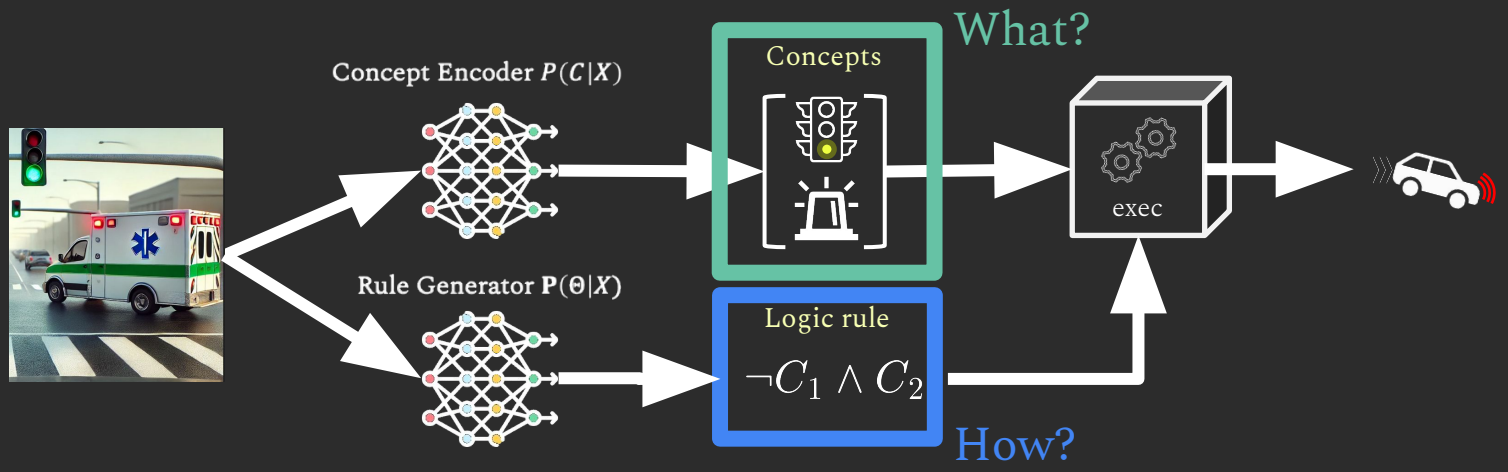
[1] Barbiero et al. "Interpretable neural-symbolic concept reasoning." International Conference on Machine Learning. PMLR, 2023.
[2] Debot et al. "Interpretable concept-based memory reasoning." NeurIPS 2024.

Neuro-symbolic task predictors

ICML 23 NeurIPS 24

Generally, such NeSy predictor are constructed as follows:

Step 1: A DNN generates both concept activations & rule parameters (neural generation)
Step 2: A symbolic engine executes rules using predicted concepts (interpretable execution)

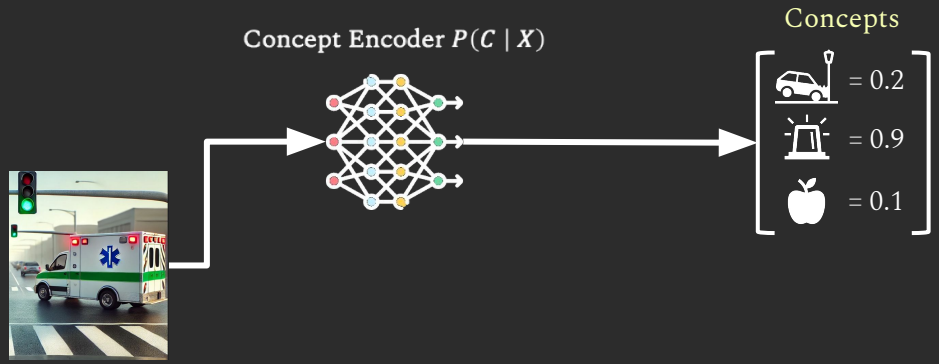


[1] Barbiero et al. "Interpretable neural-symbolic concept reasoning." International Conference on Machine Learning. PMLR, 2023.
[2] Debot et al. "Interpretable concept-based memory reasoning." NeurIPS 2024.

Memory-based NeSy predictors

We can build a concept-based memory reasoner (CMR) by:

Step 1: Predict a set of bounded scalar concept representations $\mathbf{P(C \mid X)}$ with a DNN

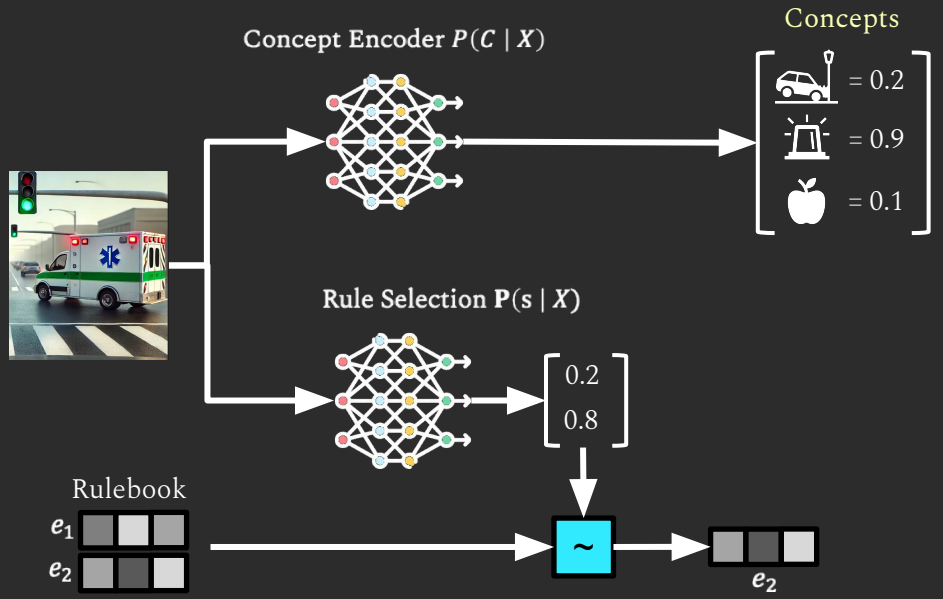


[1] Debot et al. "Interpretable concept-based memory reasoning." NeurIPS 2024.

Memory-based NeSy predictors

For example, we can build a concept-based memory reasoner (CMR) by:

Step 2: Select a rule "embedding" from a learnable "rulebook" using a DNN

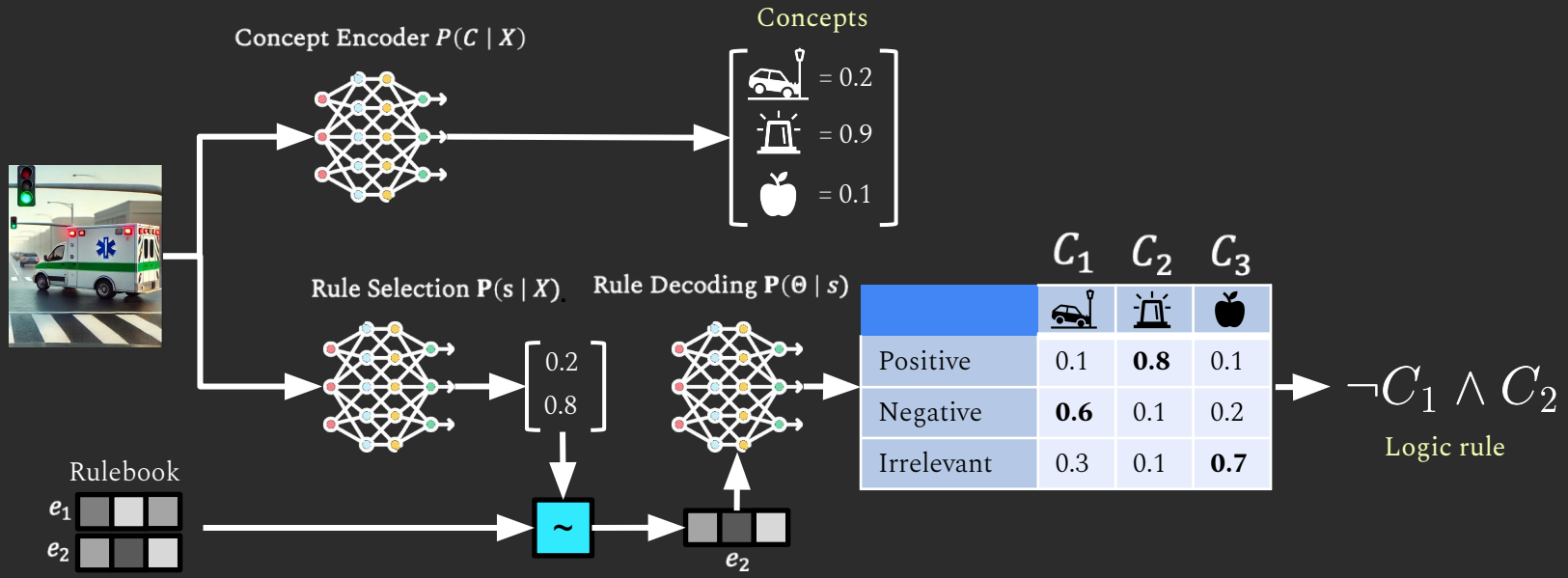


[1] Debot et al. "Interpretable concept-based memory reasoning." NeurIPS 2024.

Memory-based NeSy predictors

For example, we can build a concept-based memory reasoner (CMR) by:

Step 3: Use a learnable model to decode the rule embedding into three states per concept

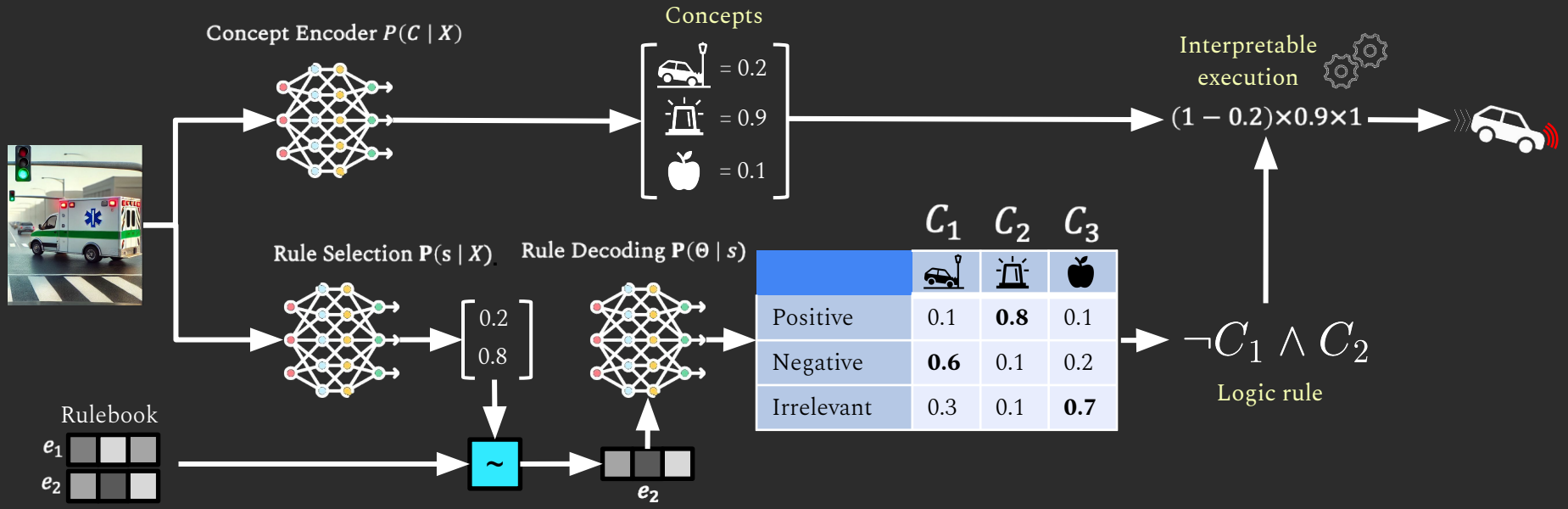


[1] Debot et al. "Interpretable concept-based memory reasoning." NeurIPS 2024.

Memory-based NeSy predictors

For example, we can build a concept-based memory reasoner (CMR) by:

Step 4: Execute the rule combining concept states and concept activations



[1] Debot et al. "Interpretable concept-based memory reasoning." NeurIPS 2024.

Properties of CMR

These steps enable CMR to:

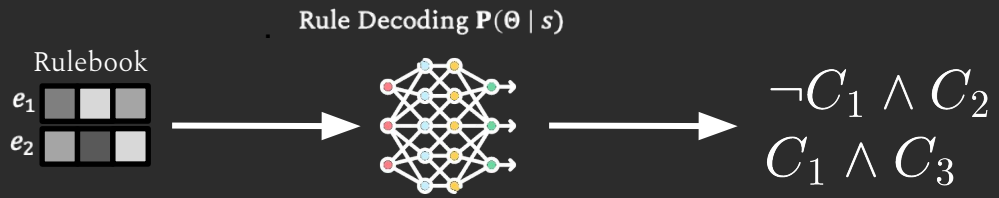
1. Become a universal approximator akin to opaque DNNs

[1] Debot et al. "Interpretable concept-based memory reasoning." NeurIPS 2024.

Properties of CMR

These steps enable CMR to:

1. Become a universal approximator akin to opaque DNNs
2. Provide both local and global interpretability



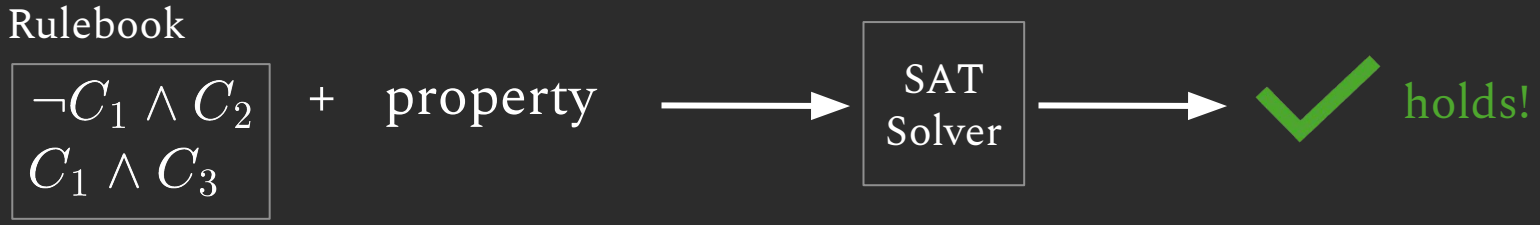
Inference mechanisms can only be selected from a finite set of transparent rules!

[1] Debot et al. "Interpretable concept-based memory reasoning." NeurIPS 2024.

Properties of CMR

These steps enable CMR to:

1. Become a universal approximator akin to opaque DNNs
2. Provide both local and global interpretability
3. Allow formal verification of desirable properties



"Does a property hold no matter which rule is selected?"

[1] Debot et al. "Interpretable concept-based memory reasoning." NeurIPS 2024.

Task Predictors

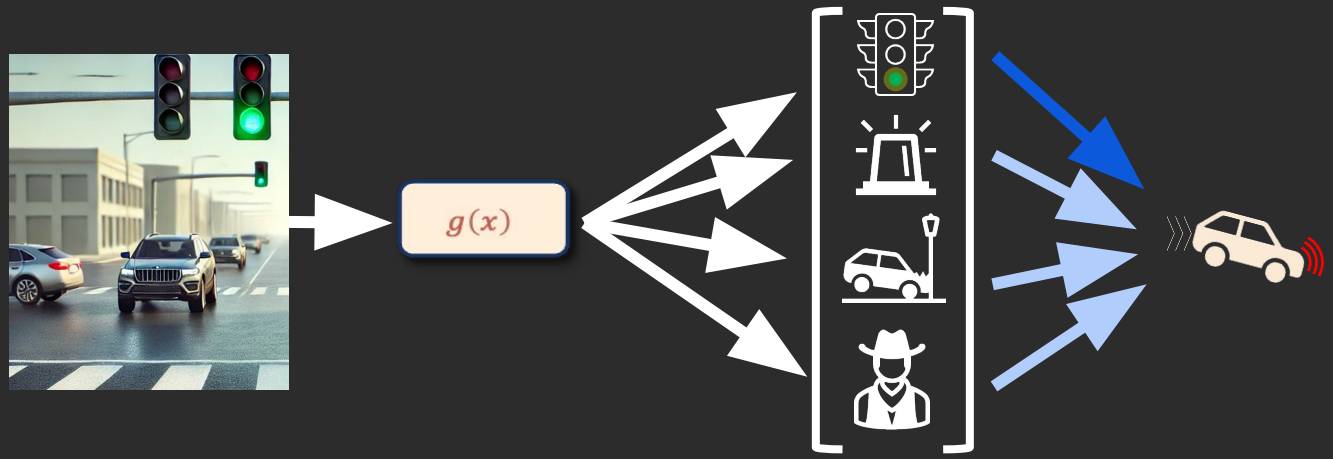
Locally-interpretable Predictors

Neuro-Symbolic Predictors

Causal Predictors

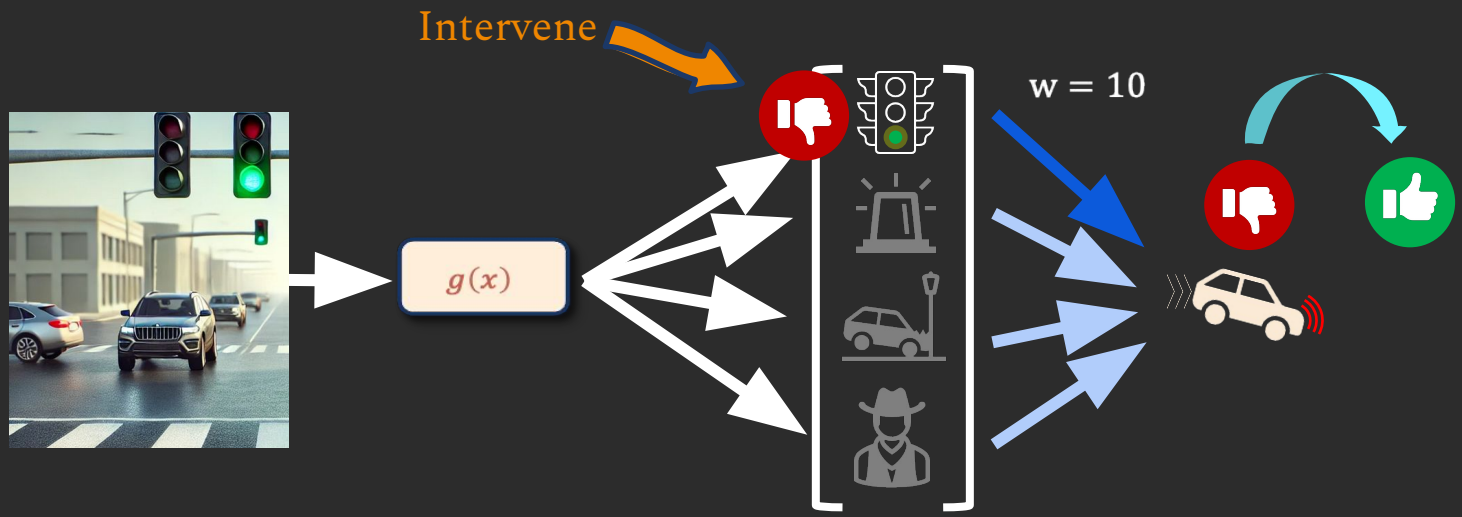
Causal task predictors

Let's consider interventions on a CBM-like model:



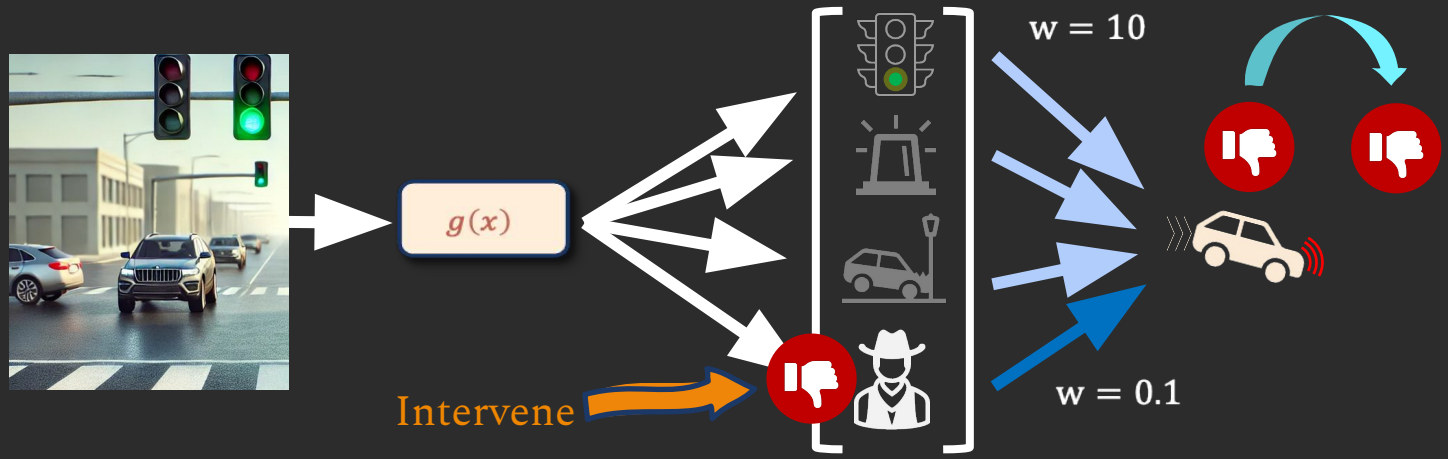
Causal task predictors

Sometimes intervening on a mispredicted helps...



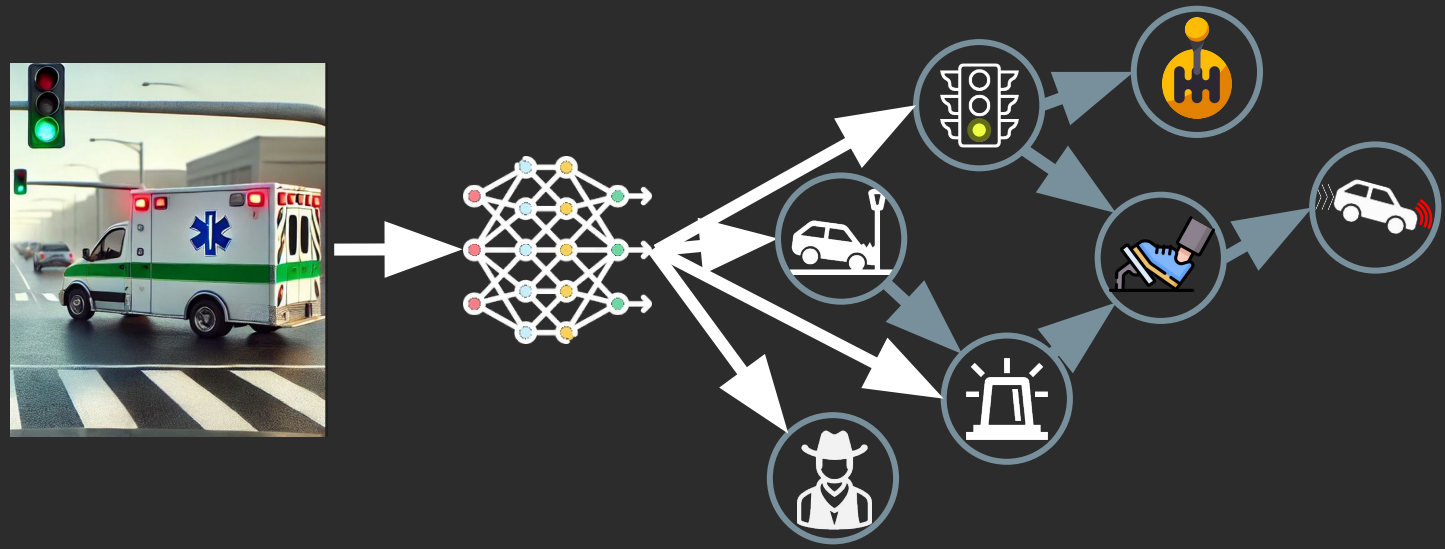
Causal task predictors

Sometimes intervening on a mispredicted concepts helps… but sometimes it doesn't!



Causal task predictors

Causal predictors explain how concepts affect other concepts and task!



Causal opacity

However, for this it is important to understand make a distinction between:

1. Causal reliability
2. Causal opacity

Causal opacity

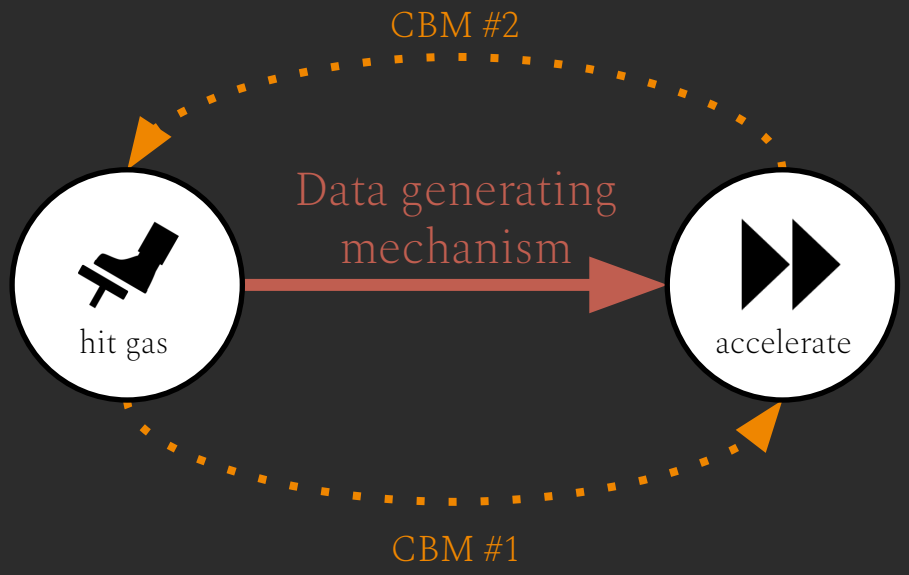
However, for this it is important to understand make a distinction between:

1. Causal reliability: discover causal mechanism of the data generating process

Causal opacity

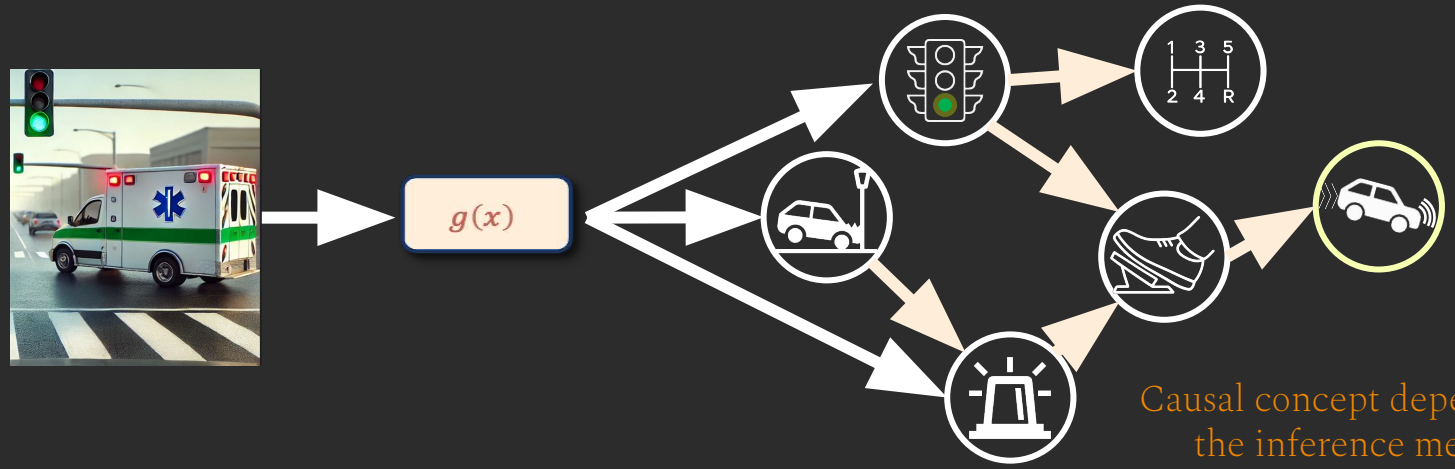
However, for this it is important to understand make a distinction between:

1. Causal reliability: discover causal mechanism of the data generating process
2. Causal opacity: discover causal mechanism of a model's inference process



Concept graph models

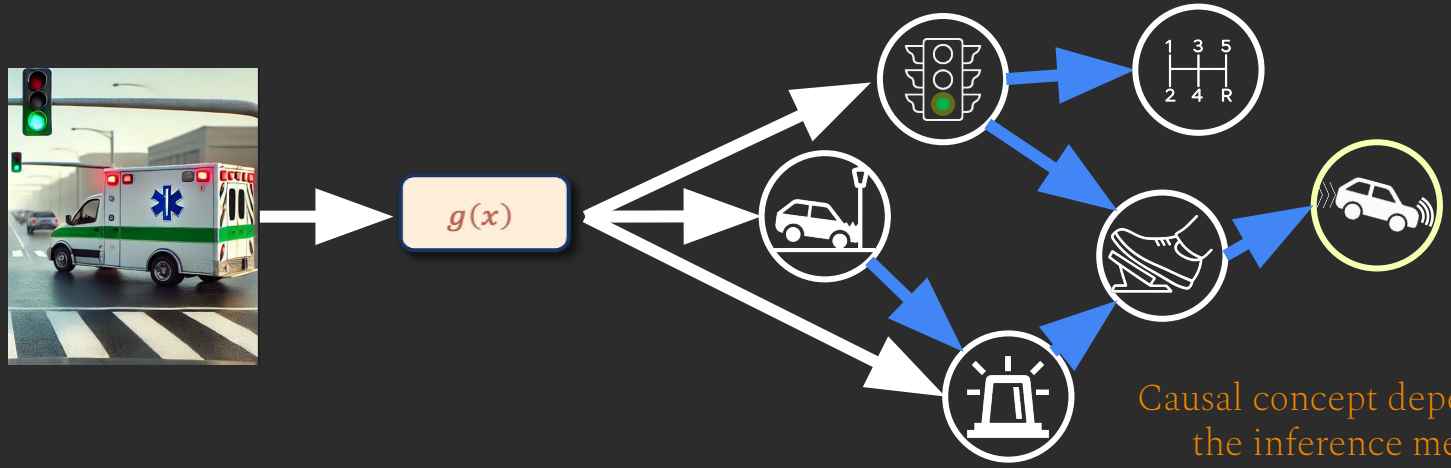
We can build a causal label predictor that explicitly uses a concept graph:



[1] Dominici et al. "Causal Concept Graph Models: Beyond Causal Opacity in Deep Learning." ICLR 2025.
[2] Moreira et al. "Diconstruct: Causal concept-based explanations through black-box distillation." CLeaR 2024.
[3] Debot et al. "Interpretable Hierarchical Concept Reasoning through Attention-Guided Graph Learning." Arxiv 2025.

Concept graph models

Different approaches use different functions: e.g. neural networks

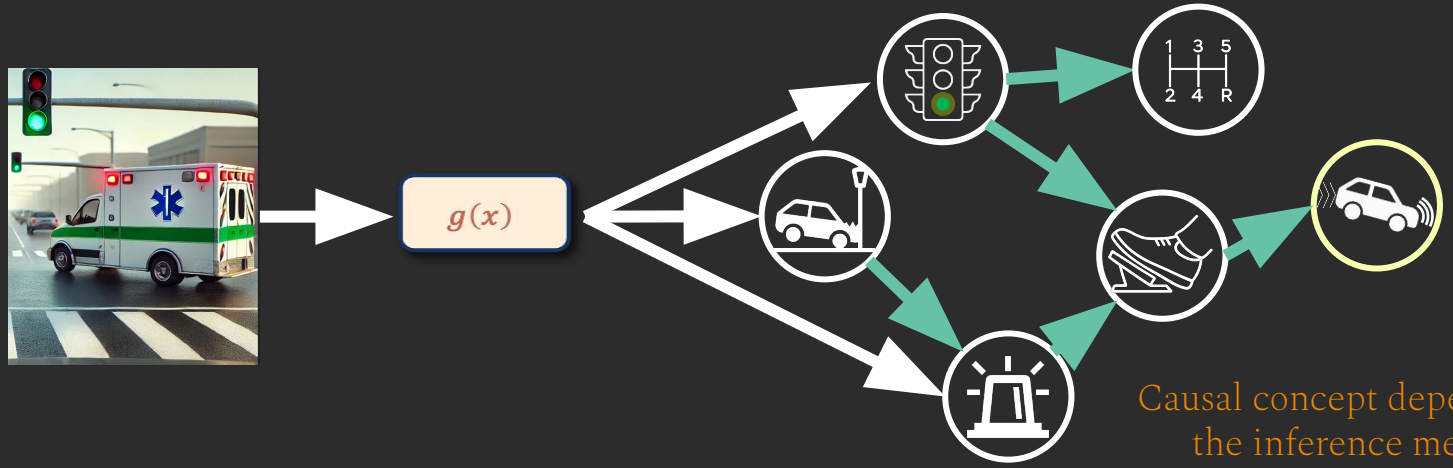


Causal concept dependencies of the inference mechanism

[1] Dominici et al. "Causal Concept Graph Models: Beyond Causal Opacity in Deep Learning." ICLR 2025.
[2] Moreira et al. "Diconstruct: Causal concept-based explanations through black-box distillation." CLeaR 2024.
[3] Debot et al. "Interpretable Hierarchical Concept Reasoning through Attention-Guided Graph Learning." Arxiv 2025.

Concept graph models

Different approaches use different functions: e.g. neural networks + logic rules

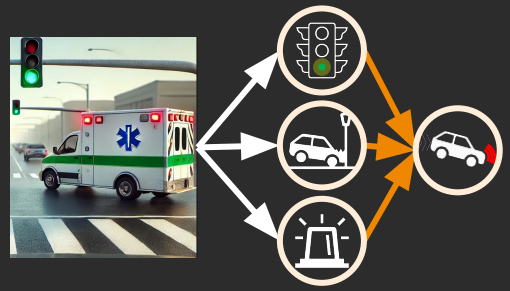


[1] Dominici et al. "Causal Concept Graph Models: Beyond Causal Opacity in Deep Learning." ICLR 2025.
[2] Moreira et al. "Diconstruct: Causal concept-based explanations through black-box distillation." CLeaR 2024.
[3] Debot et al. "Interpretable Hierarchical Concept Reasoning through Attention-Guided Graph Learning." Arxiv 2025.

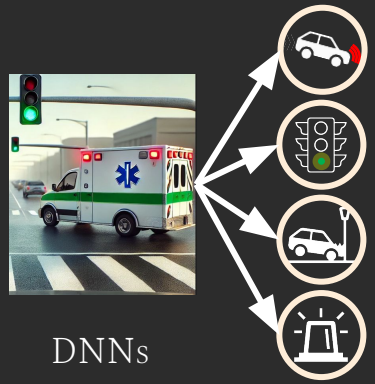
Concept graph models

The concept graph itself can be:

1. Given as a *prior*



CBMs



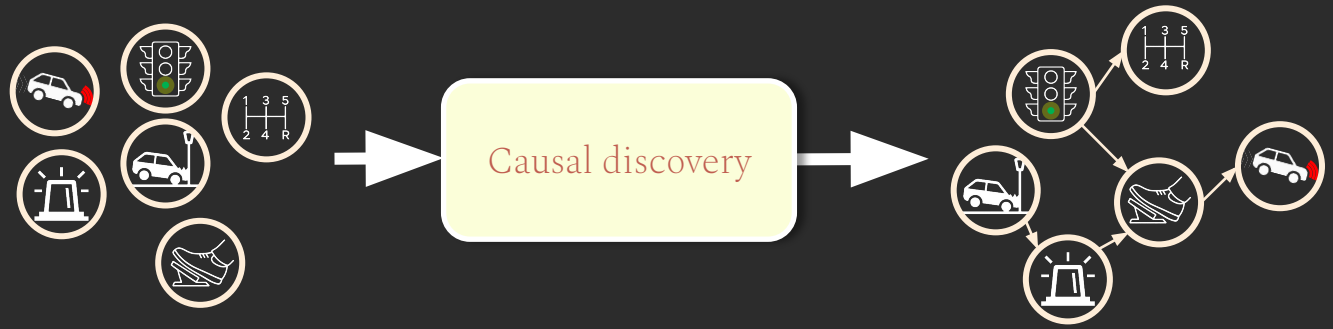
DNNs

[1] Dominici et al. "Causal Concept Graph Models: Beyond Causal Opacity in Deep Learning." ICLR 2025.
[2] Moreira et al. "Diconstruct: Causal concept-based explanations through black-box distillation." CLeaR 2024.
[3] Debot et al. "Interpretable Hierarchical Concept Reasoning through Attention-Guided Graph Learning." Arxiv 2025.

Concept graph models

The concept graph itself can be:

1. Given as a *prior*
2. Learnt from data via *causal discovery methods*

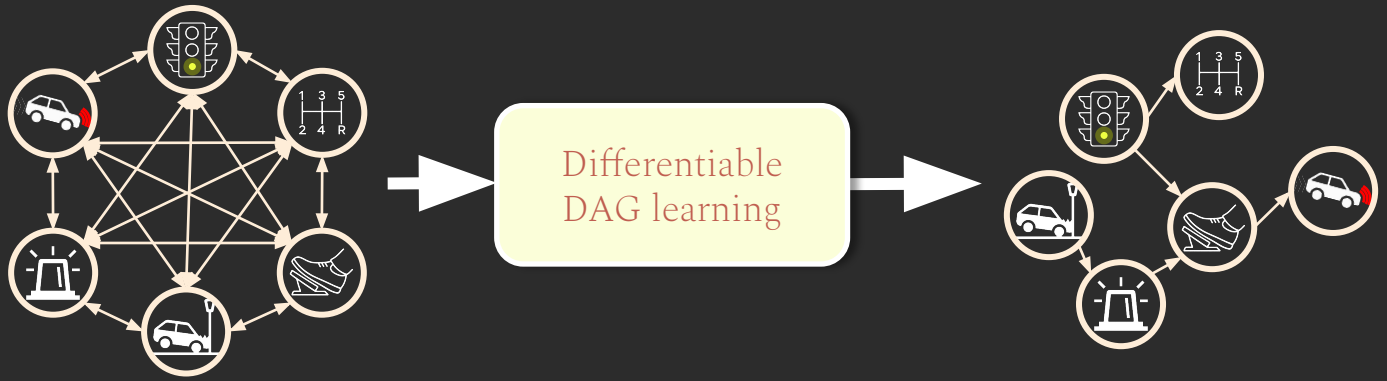


[1] Dominici et al. "Causal Concept Graph Models: Beyond Causal Opacity in Deep Learning." ICLR 2025.
[2] Moreira et al. "Diconstruct: Causal concept-based explanations through black-box distillation." CLeaR 2024.
[3] Debot et al. "Interpretable Hierarchical Concept Reasoning through Attention-Guided Graph Learning." Arxiv 2025.

Concept graph models

The concept graph itself can be:

1. Given as a *prior*
2. Learnt from data via *causal discovery methods*
3. Learnt end-to-end via *(differentiable) DAG learning*

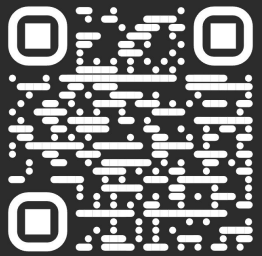


[1] Dominici et al. "Causal Concept Graph Models: Beyond Causal Opacity in Deep Learning." ICLR 2025.
[2] Moreira et al. "Diconstruct: Causal concept-based explanations through black-box distillation." CLeaR 2024.
[3] Debot et al. "Interpretable Hierarchical Concept Reasoning through Attention-Guided Graph Learning." Arxiv 2025.

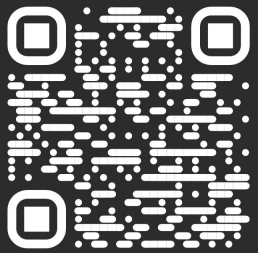
Further causal task predictors

Recent causal task predictors include:

DiConstruct

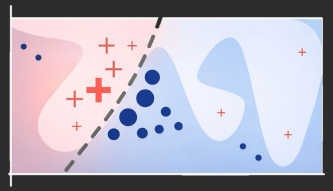


Causally Interpretable Models

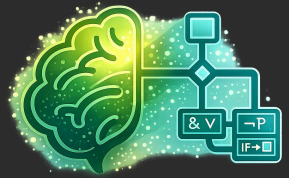


[1] Moreira et al. "Diconstruct: Causal concept-based explanations through black-box distillation." CLeaR (2024).
[2] Hwang et al. "From black-box to causal-box: Towards building more interpretable models." NeurIPS (2025).

Thank you



Locally-interpretable Predictors



Neuro-Symbolic Predictors

Causal Predictors