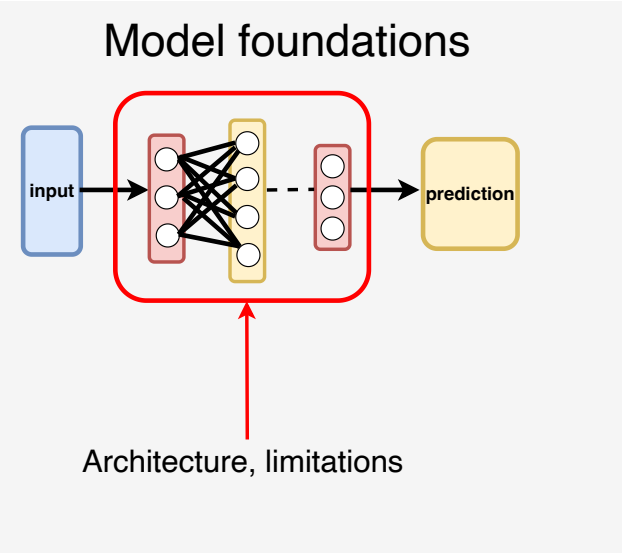
Language Model Programming: Lecture 2

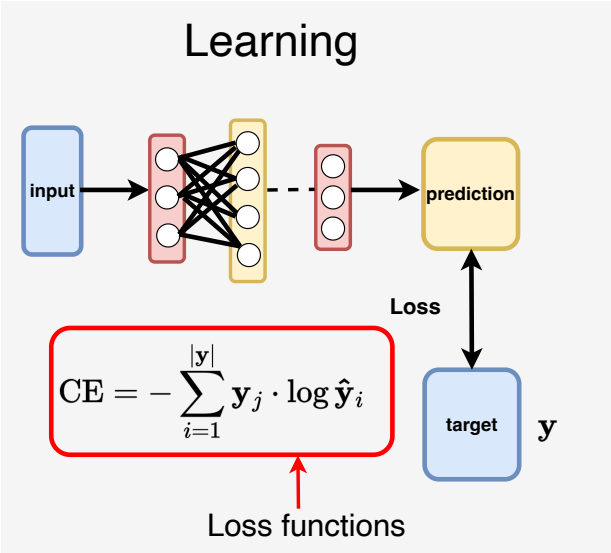
Kyle Richardson[α], Gijs Wijnholds[β]

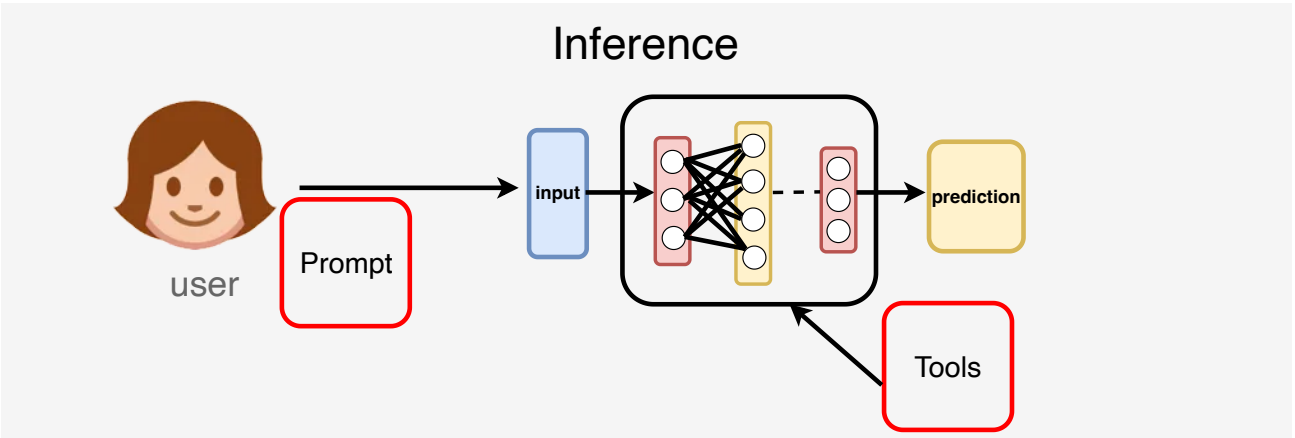
Allen Institute for AI (AI2)[α]
Leiden Institute of Advanced Computer Science **(LIACS)**[β]

August 2024

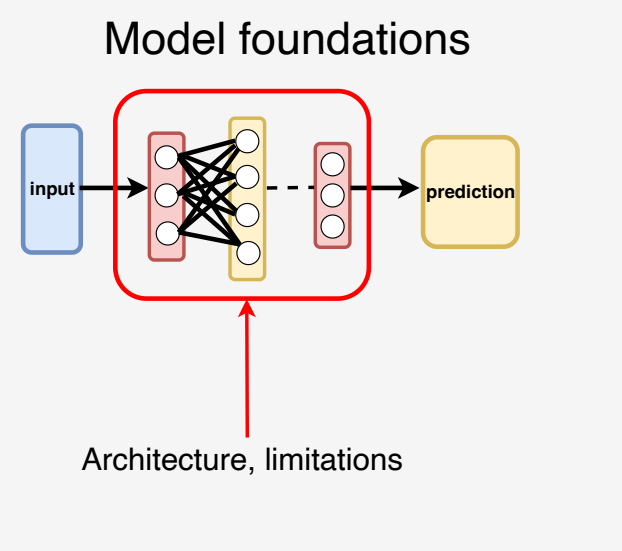
Reminder of bigger picture





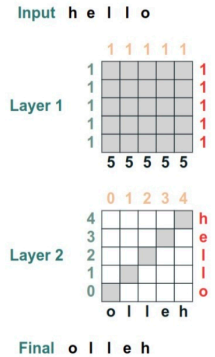


Reminder of bigger picture

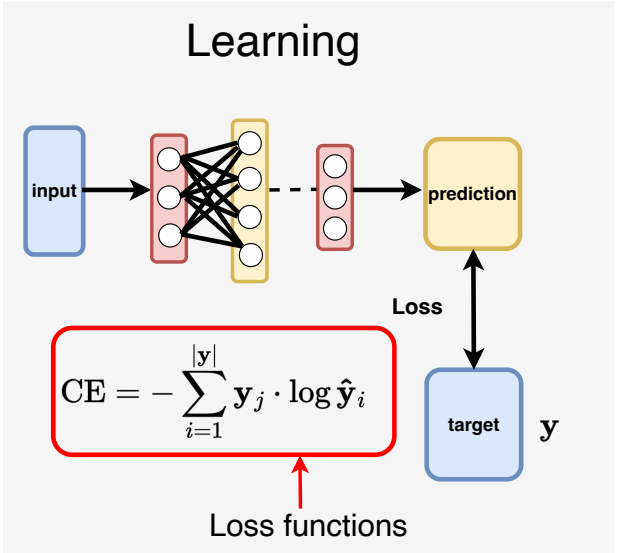


RASPy

```
def flip():
    length = (key(1) == query(1)).value(1)
    flip = (key(length - indices - 1) == query(indices)).value(tokens)
    return flip
flip()
```



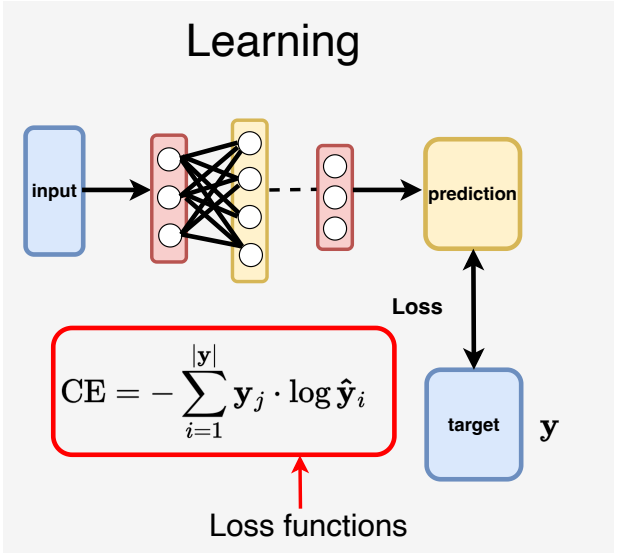
Goals for lecture today




```
1  // File path_planner.scl
2  type actor(x: i32, y: i32), goal(x: i32, y: i32), enemy(x: i32, y: i32)
3
4  const UP = 0, DOWN = 1, RIGHT = 2, LEFT = 3
5  rel safe_cell(x, y) = range(0, 5, x), range(0, 5, y), not enemy(x, y)
6  rel edge(x, y, x, yp, UP) = safe_cell(x, y), safe_cell(x, yp), yp == y + 1
7  // Rules for DOWN, RIGHT, and LEFT edges are omitted...
8
9  rel next_pos(p, q, a) = actor(x, y), edge(x, y, p, q, a)
10 rel path(x, y, x, y) = next_pos(x, y, _)
11 rel path(x1, y1, x3, y3) = path(x1, y1, x2, y2), edge(x2, y2, x3, y3, _)
12 rel next_action(a) = next_pos(p, q, a), path(p, q, r, s), goal(r, s)
```

Fig. 3. The logic program of the PacMan-Maze application in Scallop.

Goals for lecture today



Declarative Programming

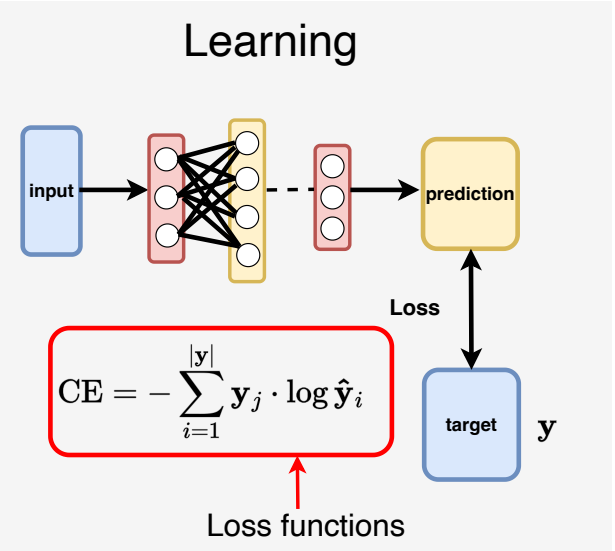
```
1  // File path_planner.scl
2  type actor(x: i32, y: i32), goal(x: i32, y: i32), enemy(x: i32, y: i32)
3
4  const UP = 0, DOWN = 1, RIGHT = 2, LEFT = 3
5  rel safe_cell(x, y) = range(0, 5, x), range(0, 5, y), not enemy(x, y)
6  rel edge(x, y, x, yp, UP) = safe_cell(x, y), safe_cell(x, yp), yp == y + 1
7  // Rules for DOWN, RIGHT, and LEFT edges are omitted...
8
9  rel next_pos(p, q, a) = actor(x, y), edge(x, y, p, q, a)
10 rel path(x, y, x, y) = next_pos(x, y, _)
11 rel path(x1, y1, x3, y3) = path(x1, y1, x2, y2), edge(x2, y2, x3, y3, _)
12 rel next_action(a) = next_pos(p, q, a), path(p, q, r, s), goal(r, s)
```

Fig. 3. The logic program of the PacMan-Maze application in Scallop.

Goals for lecture today

```
1  // File path_planner.scl
2  type actor(x: i32, y: i32), goal(x: i32, y: i32), enemy(x: i32, y: i32)
3
4  const UP = 0, DOWN = 1, RIGHT = 2, LEFT = 3
5  rel safe_cell(x, y) = range(0, 5, x), range(0, 5, y), not enemy(x, y)
6  rel edge(x, y, x, yp, UP) = safe_cell(x, y), safe_cell(x, yp), yp == y + 1
7  // Rules for DOWN, RIGHT, and LEFT edges are omitted...
8
9  rel next_pos(p, q, a) = actor(x, y), edge(x, y, p, q, a)
10 rel path(x, y, x, y) = next_pos(x, y, _)
11 rel path(x1, y1, x3, y3) = path(x1, y1, x2, y2), edge(x2, y2, x3, y3, _)
12 rel next_action(a) = next_pos(p, q, a), path(p, q, r, s), goal(r, s)
```

Fig. 3. The logic program of the PacMan-Maze application in Scallop.



- Describe a probabilistic approach to **modeling with declarative knowledge**
- Introduce **semantic loss**, a type of loss function based on this semantics.
- **next lecture**: logical and probabilistic programming for model inference.

Questions about last lecture?

Basic properties of declarative programming

```
import problog as p                    ## pip install problog
PROGRAM = p.program.PrologString("""
0.8::stress(ann).
0.4::stress(bob).
0.6::influences(ann,bob).
0.2::influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :- influences(Y,X), smokes(Y).

query(smokes(carl))."""")
p.get_evaluatable().create_from(PROGRAM).evaluate()
```

Basic properties of declarative programming

```
import problog as p                    ## pip install problog
PROGRAM = p.program.PrologString("""
0.8::stress(ann).
0.4::stress(bob).
0.6::influences(ann,bob).
0.2::influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :- influences(Y,X), smokes(Y).

query(smokes(carl))."""")
p.get_evaluatable().create_from(PROGRAM).evaluate()
```

1. We say what we want to do without saying how to do it.

Basic properties of declarative programming

```
import problog as p                    ## pip install problog
PROGRAM = p.program.PrologString("""
0.8::stress(ann).
0.4::stress(bob).
0.6::influences(ann,bob).
0.2::influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :- influences(Y,X), smokes(Y).

query(smokes(carl))."""")
p.get_evaluatable().create_from(PROGRAM).evaluate()
```

1. We say what we want to do without saying how to do it.
2. What we say can be made to be formally precise.

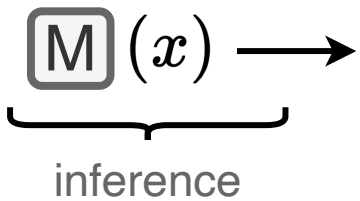
Basic properties of declarative programming

```
import problog as p                    ## pip install problog
PROGRAM = p.program.PrologString("""
0.8::stress(ann).
0.4::stress(bob).
0.6::influences(ann,bob).
0.2::influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :- influences(Y,X), smokes(Y).

query(smokes(carl))."""")
p.get_evaluatable().create_from(PROGRAM).evaluate()
```

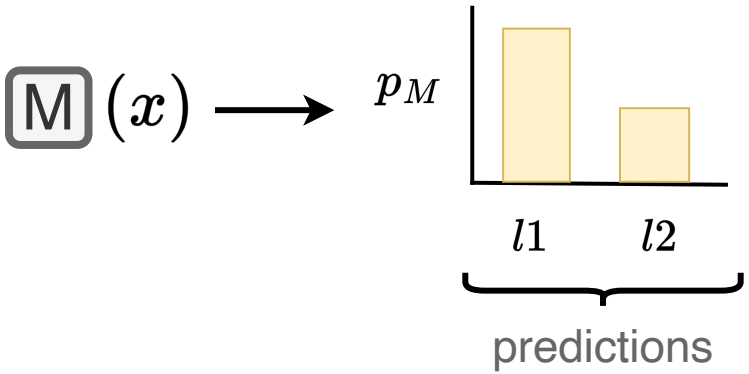
- **Logic programming**: Framework for programming based on (a subset of) logic, will revisit more tomorrow.

Thinking declaratively about model behavior

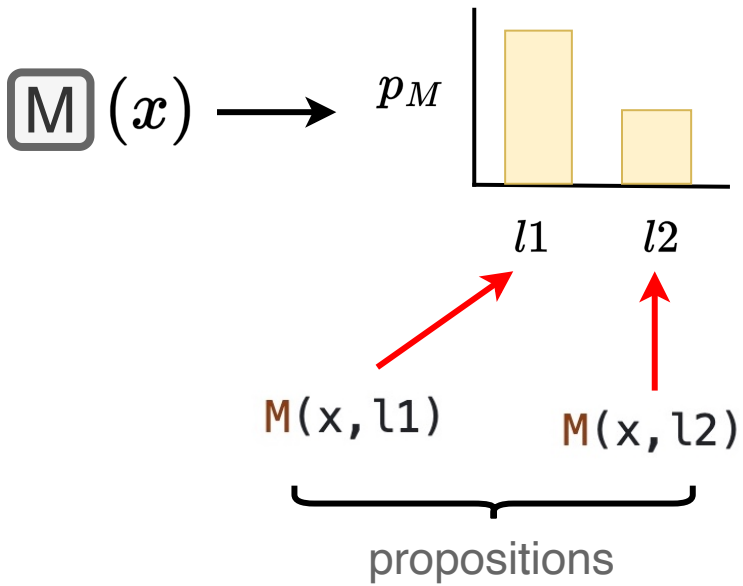
Declarative Modeling: Predictions as Propositions



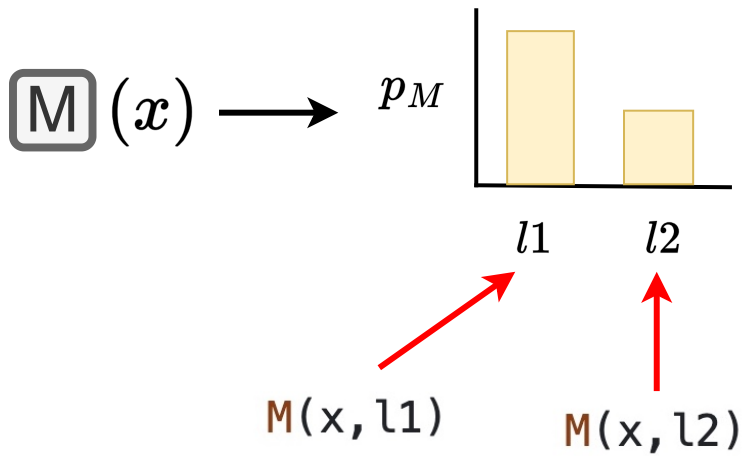
Declarative Modeling: Predictions as Propositions



Declarative Modeling: Predictions as Propositions

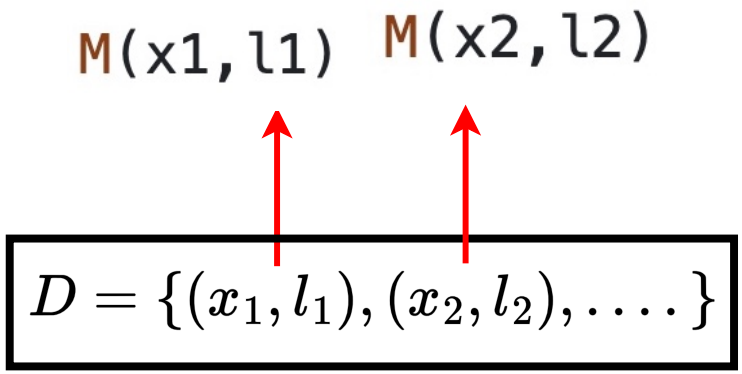


Declarative Modeling: Predictions as Propositions

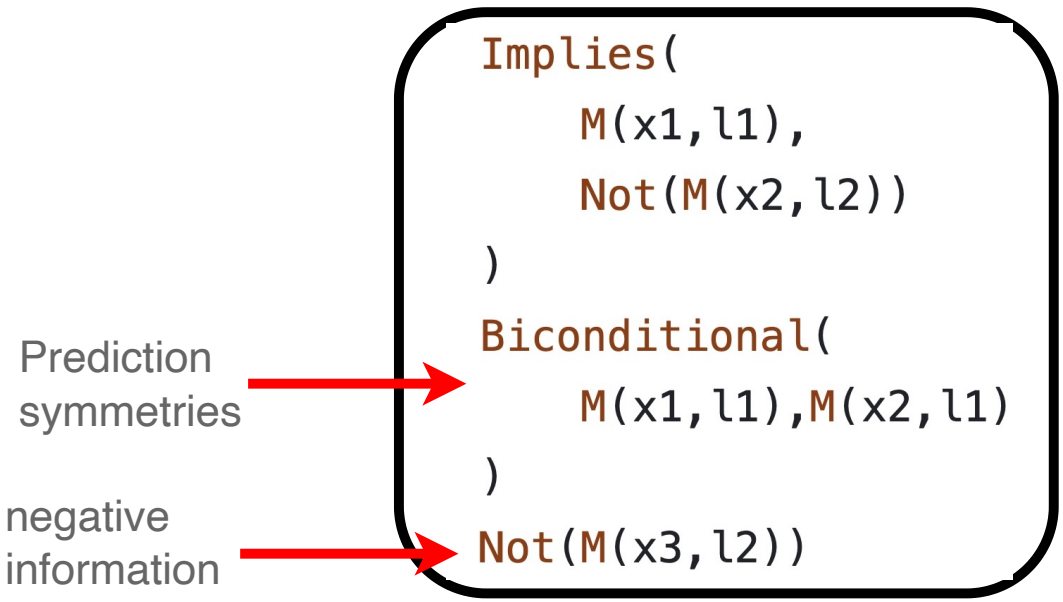


$$\underbrace{\theta(\ \texttt{M(x,l)}\ )}_{\text{proposition weight}} = p_M(l \mid x)$$

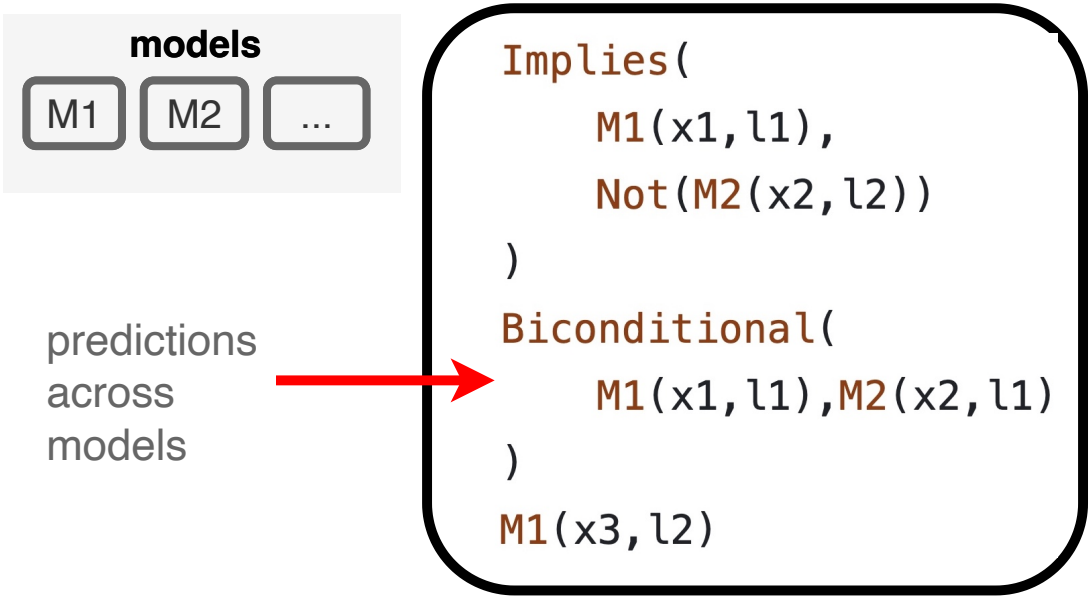
Declarative Modeling: Predictions as Propositions



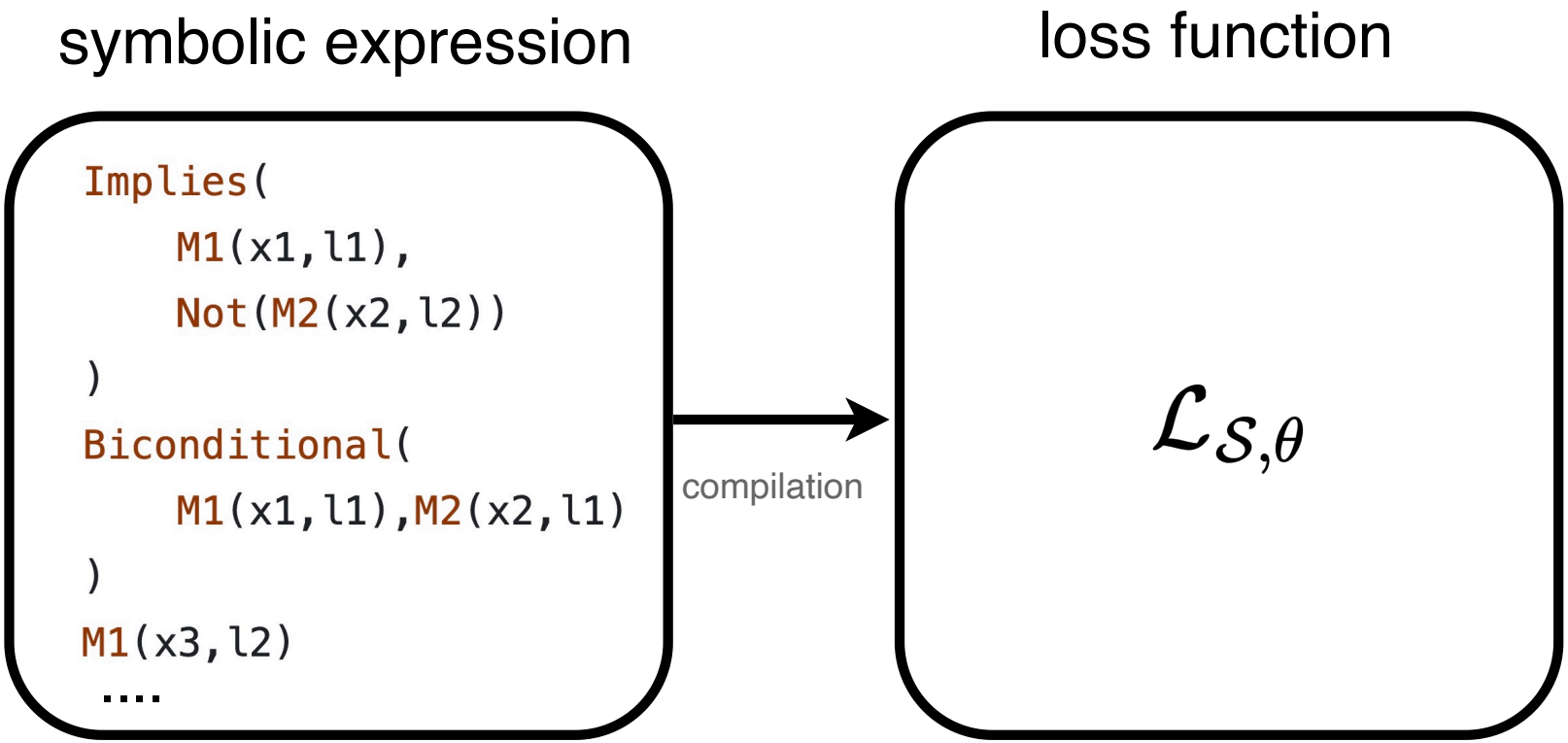
Declarative Modeling: Programs as constraints on predictions



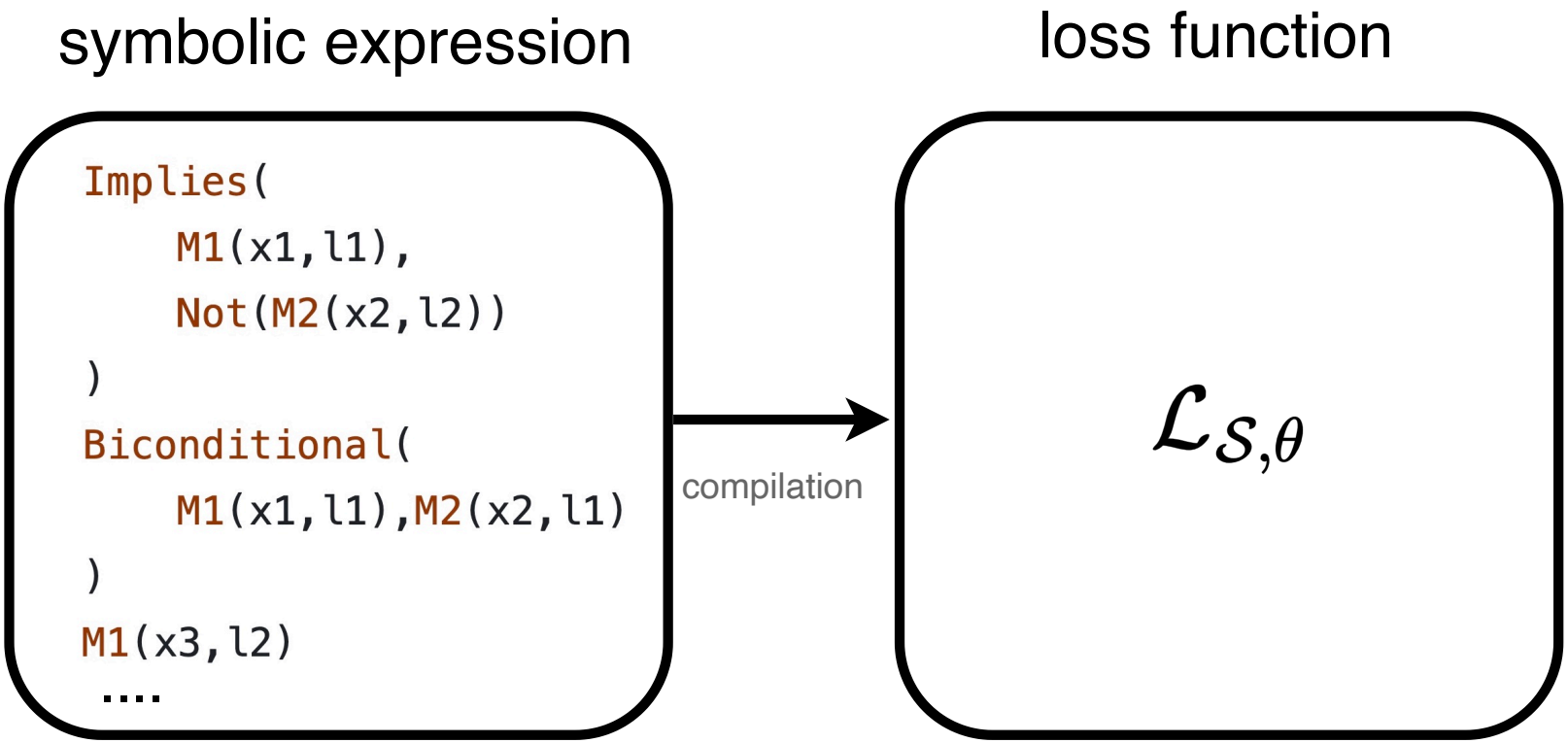
Declarative Modeling: Programs as constraints on predictions



Declarative Model Programming



Declarative Model Programming



- Say what we want models to do (without saying how), compilation into a learning or (inference) algorithm.

A motivating example

Classifying inferential relations between texts

USER	Imagining the situation: "A man was on his bike" would it be reasonable to conclude (answer yes/no) that "A person moved"?
ASSISTANT	Yes

- **Natural Language Inference** (**NLI**): Given a **premise** *(first)*, is it reasonable to conclude that the **hypothesis** *(second)* is likely to be true?

Classifying inferential relations between texts

USER	Imagining the situation: "A man was on his bike" would it be reasonable to conclude (answer yes/no) that "A person moved"?
ASSISTANT	Yes
USER	Why do you think this?
ASSISTANT	I reached this conclusion because the statement implies that the man was actively using his bike, suggesting movement. Riding a bike typically involves pedaling and propelling oneself forward, which would involve movement on the part of the person.

Datasets for NLI

```
### "pip install install  datasets"
from datasets import load_dataset

## Stanford Natural Language Inference dataset
snli_dataset= load_dataset("snli")
## Multi-genre NLI datasets
mnli_dataset = load_dataset("multi_nli")

## first few training instances
snli_dataset["train"][3]
#{
#  "premise": "Children smiling and waving at camera",
#  "hypothesis": "They are smiling at their parents",
#  "label": 1 ## =neutral
#}
```

- **Labels**: `entailment` (**E**), `contradiction` (**C**) and `neutral` (**N**).

Model sketch in Pytorch and Huggingface Transformers

```
import torch
from transformers import \
     AutoModelForSequenceClassification as hf_model

class NLIModel(torch.nn.Module):
    def __init__(self,model_name):
        super().__init__()
        self.model = hf_model.from_pretrained(
           model_name,  ## base transformer
           num_labels=3 ## 3 labels for NLI
        )
    def forward(self,features): ### forward pass
        output = self.model(**features)
        prob = output.logits.softmax(dim=-1)
        return (output.loss,prob)
```

- A task-specific classification model with a special *classification head*.

Model sketch in Pytorch and Huggingface Transformers

```
from transformers import AutoTokenizer as hf_tok
### pre-trained encoder model
nli_model = NLIModel("roberta-base")
tokenizer = hf_tok.from_pretrained("roberta-base")

### prepare data and labels
p = "Children smiling and waving at camera"
h = "They are smiling at their parents"
batch_data = tokenizer([(p,h)],return_tensors="pt")
labels = torch.tensor([[1]])

## run forward pass
output = nli_model({
    "input_ids": batch_data.input_ids,
    "labels"   : labels,
})
print(output[1])
###tensor([[0.3339, 0.3034, 0.3627]], ....)
```

Model Development Process: Classification

```
### pre-trained encoder model
nli_model = NLIModel("roberta-base")
```

1. Define a custom model with a base **pre-trained** transformer model and a set of additional classification parameters; couple with a dataset.

2. **Fine-tune** the new parameters and the full transformer using this dataset.

Model Development Process: Classification

```
### pre-trained encoder model
nli_model = NLIModel("roberta-base")
```

1. Define a custom model with a base **pre-trained** transformer model and a set of additional classification parameters; couple with a dataset.

2. **Fine-tune** the new parameters and the full transformer using this dataset.

   The idea: We can leverage the pre-trained knowledge of the model and apply it to new tasks; introduce new task format.

Model Development Process: Classification

```
### pre-trained encoder model
nli_model = NLIModel("roberta-base")
```

1. Define a custom model with a base **pre-trained** transformer model and a set of additional classification parameters; couple with a dataset.

2. **Fine-tune** the new parameters and the full transformer using this dataset.

   Variations: parameter efficient tuning, adapters; **Alternatives**: Zero/few-shot modeling, prompting. *discussed later*

What we want a model to do (semantics)

What rules should an ideal NLI model follow?

- **Predictions as propositions** We can symbolically represent NLI predictions as follows:

$$\mathbf{E}(p, h), \mathbf{C}(p, h), \mathbf{N}(p, h)$$

What rules should an ideal NLI model follow?

- **Predictions as propositions** We can symbolically represent NLI predictions as follows:

$$\mathbf{E}(p, h), \mathbf{C}(p, h), \mathbf{N}(p, h)$$

Denoting an *entailment* (**E**), *neutral* (**N**) or *contradict* (**C**) between any premise/hypothesis pair (p, h).

What rules should an ideal NLI model follow?

- **Predictions as propositions** We can symbolically represent NLI predictions as follows:

$$\mathbf{E}(p, h), \mathbf{C}(p, h), \mathbf{N}(p, h)$$

Denoting an *entailment* (**E**), *neutral* (**N**) or *contradict* (**C**) between any premise/hypothesis pair (p, h).

Example rules on predictions:

$$\forall(p, h).\, \mathbf{C}(p, h) \leftrightarrow \mathbf{C}(h, p)$$
$$\forall(p, h).\, \mathbf{E}(p, h) \rightarrow \neg\mathbf{C}(h, p)$$
$$\forall(p, h).\, \mathbf{N}(p, h) \rightarrow \neg\mathbf{C}(h, p)$$

(example rules taken from Li et al. (2019); Minervini and Riedel (2018))

What rules should an ideal NLI model follow?

- **Example rules**

$$\forall(p, h).\, \mathbf{C}(p, h) \leftrightarrow \mathbf{C}(h, p)$$
$$\forall(p, h).\, \mathbf{E}(p, h) \rightarrow \neg\mathbf{C}(h, p)$$
$$\forall(p, h).\, \mathbf{N}(p, h) \rightarrow \neg\mathbf{C}(h, p)$$

For the pair

p :John is wearing a green shirt

h :John is wearing a yellow shirt,

contradiction is a symmetric relation, flipping the order yields the same prediction.

What rules should an ideal NLI model follow?

- **Example rules**

$$\forall(p, h).\, \mathbf{C}(p, h) \leftrightarrow \mathbf{C}(h, p)$$
$$\forall(p, h).\, \mathbf{E}(p, h) \rightarrow \neg\mathbf{C}(h, p)$$
$$\forall(p, h).\, \mathbf{N}(p, h) \rightarrow \neg\mathbf{C}(h, p)$$

For the pair

p :A man is wearing a **big** green shirt

h :A man is wearing a green shirt.

there shouldn't be a contradiction if I flip the order for an entailment or a neutral prediction.

What rules should an ideal NLI model follow?

- **Example rules**

$$\forall(p, h).\, \mathbf{C}(p, h) \leftrightarrow \mathbf{C}(h, p)$$
$$\forall(p, h).\, \mathbf{E}(p, h) \rightarrow \neg\mathbf{C}(h, p)$$
$$\forall(p, h).\, \mathbf{N}(p, h) \rightarrow \neg\mathbf{C}(h, p)$$

For the pair

p :A man is wearing a **big** green shirt

h :A man is wearing a green shirt.

Li et al. (2019) found such constraints to be routinely violated, e.g., BERT violates contradiction symmetry around 20% of time.

What rules should an ideal NLI model follow?

- **More rules**

  Transitivity relations Given the triple (p, h, z)

$$\forall(p, h, z).$$
$$((\mathbf{E}(p, h) \wedge \mathbf{E}(h, z) \rightarrow \mathbf{E}(p, z)) \wedge$$
$$(\mathbf{E}(p, h) \wedge \mathbf{C}(h, z) \rightarrow \mathbf{C}(p, z)) \wedge$$
$$(\mathbf{N}(p, h) \wedge \mathbf{E}(h, z) \rightarrow \neg\mathbf{C}(p, z)) \wedge$$
$$(\mathbf{N}(p, h) \wedge \mathbf{C}(h, z) \rightarrow \neg\mathbf{E}(p, z)))$$

What rules should an ideal NLI model follow?

- **More rules**

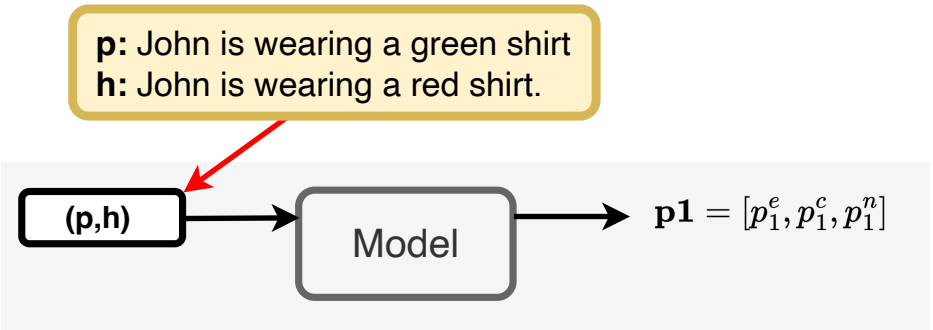
  Transitivity relations Given the triple (p, h, z)

$$\forall(p,h,z).$$
$$((\mathbf{E}(p,h) \wedge \mathbf{E}(h,z) \rightarrow \mathbf{E}(p,z)) \wedge$$
$$(\mathbf{E}(p,h) \wedge \mathbf{C}(h,z) \rightarrow \mathbf{C}(p,z)) \wedge$$
$$(\mathbf{N}(p,h) \wedge \mathbf{E}(h,z) \rightarrow \neg\mathbf{C}(p,z)) \wedge$$
$$(\mathbf{N}(p,h) \wedge \mathbf{C}(h,z) \rightarrow \neg\mathbf{E}(p,z)))$$

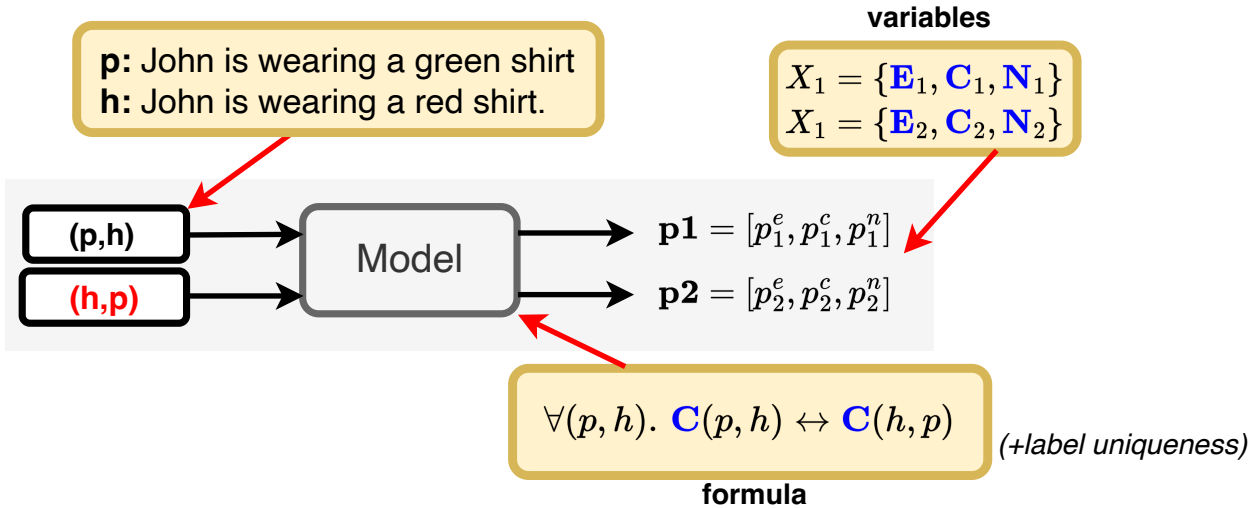
  Uniqueness constraints: there should be only one prediction.

$$\forall(p,h).$$
$$(\mathbf{E}(p,h) \vee \mathbf{C}(p,h) \vee \mathbf{N}(p,h)) \wedge$$
$$(\neg\mathbf{E}(p,h) \vee \neg\mathbf{C}(p,h)) \wedge$$
$$(\neg\mathbf{E}(p,h) \vee \neg\mathbf{N}(p,h)) \wedge$$
$$(\neg\mathbf{N}(p,h) \vee \neg\mathbf{C}(p,h))$$

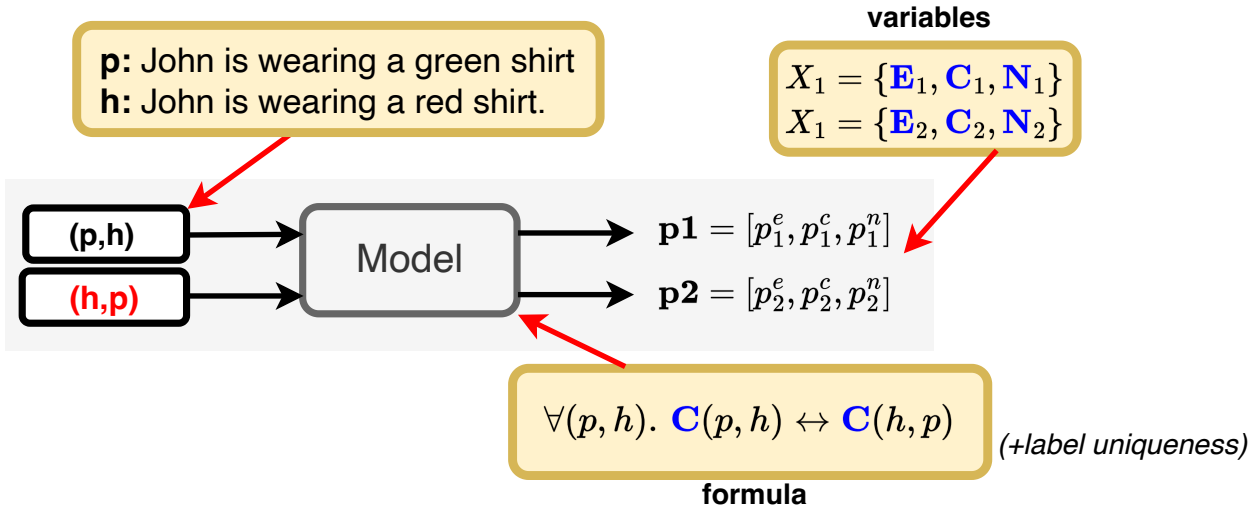
Modeling with declarative knowledge: Semantics



Modeling with declarative knowledge: Semantics



Modeling with declarative knowledge: Semantics



Thinking about our formula K in conventional semantic terms.

	Worlds	Satisfies K?	Correct?
w_1	$\neg\mathbf{E}_1, \mathbf{C}_1, \neg\mathbf{N}_1, \neg\mathbf{E}_2, \mathbf{C}_2, \neg\mathbf{N}_2$	yes	yes
w_2	$\neg\mathbf{E}_1, \mathbf{C}_1, \neg\mathbf{N}_1, \neg\mathbf{E}_2, \neg\mathbf{C}_2, \mathbf{N}_2$	no	no
w_3	$\neg\mathbf{E}_1, \mathbf{C}_1, \neg\mathbf{N}_1, \mathbf{E}_2, \neg\mathbf{C}_2, \neg\mathbf{N}_2$	no	no
w_4	$\neg\mathbf{E}_1, \neg\mathbf{C}_1, \mathbf{N}_1, \neg\mathbf{E}_2, \mathbf{C}_2, \neg\mathbf{N}_2$	no	no
w_5	$\neg\mathbf{E}_1, \neg\mathbf{C}_1, \mathbf{N}_1, \neg\mathbf{E}_2, \neg\mathbf{C}_2, \mathbf{N}_2$	yes	no
w_6	$\neg\mathbf{E}_1, \neg\mathbf{C}_1, \mathbf{N}_1, \mathbf{E}_2, \neg\mathbf{C}_2, \neg\mathbf{N}_2$	yes	no
w_7	$\mathbf{E}_1, \neg\mathbf{C}_1, \neg\mathbf{N}_1, \neg\mathbf{E}_2, \mathbf{C}_2, \neg\mathbf{N}_2$	no	no
w_8	$\mathbf{E}_1, \neg\mathbf{C}_1, \neg\mathbf{N}_1, \neg\mathbf{E}_2, \neg\mathbf{C}_2, \mathbf{N}_2$	yes	no
w_9	$\mathbf{E}_1, \neg\mathbf{C}_1, \neg\mathbf{N}_1, \mathbf{E}_2, \neg\mathbf{C}_2, \neg\mathbf{N}_2$	yes	no

count: 5

Modeling with declarative knowledge: Semantics

Thinking about our formula K in conventional semantic terms.

	Worlds	Satisfies K?	Correct?
w_1	$\neg E_1, C_1, \neg N_1, \neg E_2, C_2, \neg N_2$	yes	yes
w_2	$\neg E_1, C_1, \neg N_1, \neg E_2, \neg C_2, N_2$	no	no
w_3	$\neg E_1, C_1, \neg N_1, E_2, \neg C_2, \neg N_2$	no	no
w_4	$\neg E_1, \neg C_1, N_1, \neg E_2, C_2, \neg N_2$	no	no
w_5	$\neg E_1, \neg C_1, N_1, \neg E_2, \neg C_2, N_2$	yes	no
w_6	$\neg E_1, \neg C_1, N_1, E_2, \neg C_2, \neg N_2$	yes	no
w_7	$E_1, \neg C_1, \neg N_1, \neg E_2, C_2, \neg N_2$	no	no
w_8	$E_1, \neg C_1, \neg N_1, \neg E_2, \neg C_2, N_2$	yes	no
w_9	$E_1, \neg C_1, \neg N_1, E_2, \neg C_2, \neg N_2$	yes	no

count: 5

Modeling with declarative knowledge: Semantics

Thinking about our formula K in conventional semantic terms.

	Worlds	Satisfies K?	Correct?
w_1	$\neg E_1, C_1, \neg N_1, \neg E_2, C_2, \neg N_2$	yes	yes
w_2	$\neg E_1, C_1, \neg N_1, \neg E_2, \neg C_2, N_2$	no	no
w_3	$\neg E_1, C_1, \neg N_1, E_2, \neg C_2, \neg N_2$	no	no
w_4	$\neg E_1, \neg C_1, N_1, \neg E_2, C_2, \neg N_2$	no	no
w_5	$\neg E_1, \neg C_1, N_1, \neg E_2, \neg C_2, N_2$	yes	no
w_6	$\neg E_1, \neg C_1, N_1, E_2, \neg C_2, \neg N_2$	yes	no
w_7	$E_1, \neg C_1, \neg N_1, \neg E_2, C_2, \neg N_2$	no	no
w_8	$E_1, \neg C_1, \neg N_1, \neg E_2, \neg C_2, N_2$	yes	no
w_9	$E_1, \neg C_1, \neg N_1, E_2, \neg C_2, \neg N_2$	yes	no

count: 5

What we care about: worlds **w** where our target formula K is true (denoted as $\mathbf{w} \models K$)

Modeling with declarative knowledge: Semantics

Thinking about our formula K in conventional semantic terms.

	Worlds	Satisfies K?	Correct?
w_1	$\neg E_1, C_1, \neg N_1, \neg E_2, C_2, \neg N_2$	yes	yes
w_2	$\neg E_1, C_1, \neg N_1, \neg E_2, \neg C_2, N_2$	no	no
w_3	$\neg E_1, C_1, \neg N_1, E_2, \neg C_2, \neg N_2$	no	no
w_4	$\neg E_1, \neg C_1, N_1, \neg E_2, C_2, \neg N_2$	no	no
w_5	$\neg E_1, \neg C_1, N_1, \neg E_2, \neg C_2, N_2$	yes	no
w_6	$\neg E_1, \neg C_1, N_1, E_2, \neg C_2, \neg N_2$	yes	no
w_7	$E_1, \neg C_1, \neg N_1, \neg E_2, C_2, \neg N_2$	no	no
w_8	$E_1, \neg C_1, \neg N_1, \neg E_2, \neg C_2, N_2$	yes	no
w_9	$E_1, \neg C_1, \neg N_1, E_2, \neg C_2, \neg N_2$	yes	no
		count: 5	

What we care about: worlds **w** where our target formula K is true (denoted as $\mathbf{w} \models K$)

$$\textsc{count}(K) := \sum_{\mathbf{w} \models K} 1$$

Modeling with declarative knowledge: Semantics

Thinking about our formula K in conventional semantic terms.

	Worlds	Satisfies K?	Correct?
w_1	$\neg E_1, C_1, \neg N_1, \neg E_2, C_2, \neg N_2$	yes	yes
w_2	$\neg E_1, C_1, \neg N_1, \neg E_2, \neg C_2, N_2$	no	no
w_3	$\neg E_1, C_1, \neg N_1, E_2, \neg C_2, \neg N_2$	no	no
w_4	$\neg E_1, \neg C_1, N_1, \neg E_2, C_2, \neg N_2$	no	no
w_5	$\neg E_1, \neg C_1, N_1, \neg E_2, \neg C_2, N_2$	yes	no
w_6	$\neg E_1, \neg C_1, N_1, E_2, \neg C_2, \neg N_2$	yes	no
w_7	$E_1, \neg C_1, \neg N_1, \neg E_2, C_2, \neg N_2$	no	no
w_8	$E_1, \neg C_1, \neg N_1, \neg E_2, \neg C_2, N_2$	yes	no
w_9	$E_1, \neg C_1, \neg N_1, E_2, \neg C_2, \neg N_2$	yes	no
		count: 5	

When predictions scores are considered: weights the rows based on these prediction scores.

$$\mathrm{WMC}(\mathsf{K}, \theta) := \sum_{\mathbf{w} \models \mathsf{K}} \underbrace{\prod_{\mathbf{w} \models X_i} P_\theta(X_i) \cdot \prod_{\mathbf{w} \models \neg X_i} (1 - P_\theta(X_i))}_{\text{score of row}}$$

Modeling with declarative knowledge: Semantics

Thinking about our formula K in conventional semantic terms.

	Worlds	Satisfies K?	Correct?
w_1	$\neg E_1, C_1, \neg N_1, \neg E_2, C_2, \neg N_2$	yes	yes
w_2	$\neg E_1, C_1, \neg N_1, \neg E_2, \neg C_2, N_2$	no	no
w_3	$\neg E_1, C_1, \neg N_1, E_2, \neg C_2, \neg N_2$	no	no
w_4	$\neg E_1, \neg C_1, N_1, \neg E_2, C_2, \neg N_2$	no	no
w_5	$\neg E_1, \neg C_1, N_1, \neg E_2, \neg C_2, N_2$	yes	no
w_6	$\neg E_1, \neg C_1, N_1, E_2, \neg C_2, \neg N_2$	yes	no
w_7	$E_1, \neg C_1, \neg N_1, \neg E_2, C_2, \neg N_2$	no	no
w_8	$E_1, \neg C_1, \neg N_1, \neg E_2, \neg C_2, N_2$	yes	no
w_9	$E_1, \neg C_1, \neg N_1, E_2, \neg C_2, \neg N_2$	yes	no
		count: 5	

When predictions scores are considered: weights the rows based on these prediction scores.

$$\mathrm{WMC}(\mathsf{K}, \theta) := \underbrace{\sum_{\mathbf{w} \models \mathsf{K}} \prod_{\mathbf{w} \models X_i} P_\theta(X_i) \cdot \prod_{\mathbf{w} \models \neg X_i} (1 - P_\theta(X_i))}_{\text{weighted model counting (WMC)}}$$

Modeling with declarative knowledge: Semantics

- Suppose for K that we have weights θ:

  $P_\theta(E_1), (C_1), P_\theta(N_1), P(E_2), P_\theta(C_2), P_\theta(N_2) = [0.3, 0.2, 0.5, 0.1, 0.8, 0.1]$

  then

	Worlds	Satisfies K?	Score
$\mathbf{w}_1$	$\neg E_1, C_1, \neg N_1, \neg E_2, C_2, \neg N_2$	True	0.0454
$\mathbf{w}_2$	$\neg E_1, C_1, \neg N_1, \neg E_2, \neg C_2, N_2$	False	0.0013
$\mathbf{w}_3$	$\neg E_1, C_1, \neg N_1, E_2, \neg C_2, \neg N_2$	False	0.0013
$\mathbf{w}_4$	$\neg E_1, \neg C_1, N_1, \neg E_2, C_2, \neg N_2$	False	0.1814
$\mathbf{w}_5$	$\neg E_1, \neg C_1, N_1, \neg E_2, \neg C_2, N_2$	True	0.005
$\mathbf{w}_6$	$\neg E_1, \neg C_1, N_1, E_2, \neg C_2, \neg N_2$	True	0.005
$\mathbf{w}_7$	$E_1, \neg C_1, \neg N_1, \neg E_2, C_2, \neg N_2$	False	0.0778
$\mathbf{w}_8$	$E_1, \neg C_1, \neg N_1, \neg E_2, \neg C_2, N_2$	True	0.0022
$\mathbf{w}_9$	$E_1, \neg C_1, \neg N_1, E_2, \neg C_2, \neg N_2$	True	0.0022

WMC: $\approx$0.05976

e.g., score of $\mathbf{w}_1 = (1 - 0.3) * 0.2 * (1 - 0.5) * (1 - 0.1) * 0.8 * (1 - 0.1)$

Modeling with declarative knowledge: Semantics

- Suppose for K that we have weights θ:

  $P_\theta(E_1), (C_1), P_\theta(N_1), P(E_2), P_\theta(C_2), P_\theta(N_2) = [0.3, 0.2, 0.5, 0.1, 0.8, 0.1]$

  then

	Worlds	Satisfies K?	Score
w_1	$\neg E_1, C_1, \neg N_1, \neg E_2, C_2, \neg N_2$	True	0.0454
w_2	$\neg E_1, C_1, \neg N_1, \neg E_2, \neg C_2, N_2$	False	0.0013
w_3	$\neg E_1, C_1, \neg N_1, E_2, \neg C_2, \neg N_2$	False	0.0013
w_4	$\neg E_1, \neg C_1, N_1, \neg E_2, C_2, \neg N_2$	False	0.1814
w_5	$\neg E_1, \neg C_1, N_1, \neg E_2, \neg C_2, N_2$	True	0.005
w_6	$\neg E_1, \neg C_1, N_1, E_2, \neg C_2, \neg N_2$	True	0.005
w_7	$E_1, \neg C_1, \neg N_1, \neg E_2, C_2, \neg N_2$	False	0.0778
w_8	$E_1, \neg C_1, \neg N_1, \neg E_2, \neg C_2, N_2$	True	0.0022
w_9	$E_1, \neg C_1, \neg N_1, E_2, \neg C_2, \neg N_2$	True	0.0022

WMC: $\approx$0.05976

  Total count $\mathrm{WMC}(\mathrm{K}) = 0.05976$

Modeling with declarative knowledge: Semantics

- Suppose for K that we have weights θ:

  $$P_\theta(E_1), (C_1), P_\theta(N_1), P(E_2), P_\theta(C_2), P_\theta(N_2) = [0.3, 0.2, 0.5, 0.1, 0.8, 0.1]$$

  then

	Worlds	Satisfies K?	Score
$\mathbf{w}_1$	$\neg\mathbf{E}_1, \mathbf{C}_1, \neg\mathbf{N}_1, \neg\mathbf{E}_2, \mathbf{C}_2, \neg\mathbf{N}_2$	True	0.0454
$\mathbf{w}_2$	$\neg\mathbf{E}_1, \mathbf{C}_1, \neg\mathbf{N}_1, \neg\mathbf{E}_2, \neg\mathbf{C}_2, \mathbf{N}_2$	False	0.0013
$\mathbf{w}_3$	$\neg\mathbf{E}_1, \mathbf{C}_1, \neg\mathbf{N}_1, \mathbf{E}_2, \neg\mathbf{C}_2, \neg\mathbf{N}_2$	False	0.0013
$\mathbf{w}_4$	$\neg\mathbf{E}_1, \neg\mathbf{C}_1, \mathbf{N}_1, \neg\mathbf{E}_2, \mathbf{C}_2, \neg\mathbf{N}_2$	False	0.1814
$\mathbf{w}_5$	$\neg\mathbf{E}_1, \neg\mathbf{C}_1, \mathbf{N}_1, \neg\mathbf{E}_2, \neg\mathbf{C}_2, \mathbf{N}_2$	True	0.005
$\mathbf{w}_6$	$\neg\mathbf{E}_1, \neg\mathbf{C}_1, \mathbf{N}_1, \mathbf{E}_2, \neg\mathbf{C}_2, \neg\mathbf{N}_2$	True	0.005
$\mathbf{w}_7$	$\mathbf{E}_1, \neg\mathbf{C}_1, \neg\mathbf{N}_1, \neg\mathbf{E}_2, \mathbf{C}_2, \neg\mathbf{N}_2$	False	0.0778
$\mathbf{w}_8$	$\mathbf{E}_1, \neg\mathbf{C}_1, \neg\mathbf{N}_1, \neg\mathbf{E}_2, \neg\mathbf{C}_2, \mathbf{N}_2$	True	0.0022
$\mathbf{w}_9$	$\mathbf{E}_1, \neg\mathbf{C}_1, \neg\mathbf{N}_1, \mathbf{E}_2, \neg\mathbf{C}_2, \neg\mathbf{N}_2$	True	0.0022

WMC: $\approx$0.05976

Intuitively: the higher the count, the more our rule is satisfied.

Modeling with declarative knowledge: Semantics

- Suppose for K that we have weights θ:

  $$P_\theta(E_1), (C_1), P_\theta(N_1), P(E_2), P_\theta(C_2), P_\theta(N_2) = [0.3, 0.2, 0.5, 0.1, 0.8, 0.1]$$

  then

	Worlds	Satisfies K?	Score
w_1	$\neg E_1, C_1, \neg N_1, \neg E_2, C_2, \neg N_2$	True	0.0454
w_2	$\neg E_1, C_1, \neg N_1, \neg E_2, \neg C_2, N_2$	False	0.0013
w_3	$\neg E_1, C_1, \neg N_1, E_2, \neg C_2, \neg N_2$	False	0.0013
w_4	$\neg E_1, \neg C_1, N_1, \neg E_2, C_2, \neg N_2$	False	0.1814
w_5	$\neg E_1, \neg C_1, N_1, \neg E_2, \neg C_2, N_2$	True	0.005
w_6	$\neg E_1, \neg C_1, N_1, E_2, \neg C_2, \neg N_2$	True	0.005
w_7	$E_1, \neg C_1, \neg N_1, \neg E_2, C_2, \neg N_2$	False	0.0778
w_8	$E_1, \neg C_1, \neg N_1, \neg E_2, \neg C_2, N_2$	True	0.0022
w_9	$E_1, \neg C_1, \neg N_1, E_2, \neg C_2, \neg N_2$	True	0.0022

WMC: ≈ 0.05976

Note: Counting is intractable. Variation of Boolean satisfiability (SAT).

Questions?

Compiling to Loss

Semantic loss

- **Probabilistic logic**, based on the semantics of weighted model counting, for a formula K, we want to compute the following probability:

Semantic loss

- **Probabilistic logic**, based on the semantics of weighted model counting, for a formula K, we want to compute the following probability:

$$P_\theta(\mathsf{K} = 1)$$

$$\text{where } P_\theta(\mathsf{K} = 1) = \frac{\mathrm{WMC}(\mathsf{K}, \theta)}{\mathrm{WMC}(\mathsf{K}, \theta) + \mathrm{WMC}(\neg\mathsf{K}, \theta)}$$

Semantic loss

- **Probabilistic logic**, based on the semantics of weighted model counting, for a formula K, we want to compute the following probability:

$$P_\theta(\mathsf{K} = 1)$$

$$\text{where } P_\theta(\mathsf{K} = 1) = \frac{\mathrm{WMC}(\mathsf{K}, \theta)}{\mathrm{WMC}(\mathsf{K}, \theta) + \mathrm{WMC}(\neg \mathsf{K}, \theta)}$$

The **semantic loss** $\mathcal{L}_{\mathsf{sl}}$ (Xu et al., 2018) for K is:

$$\mathcal{L}_{\mathsf{sl}}(\mathsf{K}, \theta) := -\log P_\theta(\mathsf{K} = 1)$$

Semantic loss

- **Probabilistic logic**, based on the semantics of weighted model counting, for a formula K, we want to compute the following probability:

$$P_\theta(\mathrm{K} = 1)$$

$$\text{where } P_\theta(\mathrm{K} = 1) = \frac{\mathrm{WMC}(\mathrm{K}, \theta)}{\mathrm{WMC}(\mathrm{K}, \theta) + \mathrm{WMC}(\neg \mathrm{K}, \theta)}$$

The **semantic loss** $\mathcal{L}_{\mathsf{sl}}$ (Xu et al., 2018) for K is:

$$\mathcal{L}_{\mathsf{sl}}(\mathrm{K}, \theta) := -\log P_\theta(\mathrm{K} = 1)$$

Has a natural probabilistic semantics:

Semantic loss

- **Probabilistic logic**, based on the semantics of weighted model counting, for a formula K, we want to compute the following probability:

$$P_\theta(\mathrm{K} = 1)$$

$$\text{where } P_\theta(\mathrm{K} = 1) = \frac{\mathrm{WMC}(\mathrm{K}, \theta)}{\mathrm{WMC}(\mathrm{K}, \theta) + \mathrm{WMC}(\neg \mathrm{K}, \theta)}$$

The **semantic loss** $\mathcal{L}_{\mathsf{sl}}$ (Xu et al., 2018) for K is:

$$\mathcal{L}_{\mathsf{sl}}(\mathrm{K}, \theta) := -\log P_\theta(\mathrm{K} = 1)$$

Has a natural probabilistic semantics:

> The semantic loss is ... [equal]... to a negative logarithm of the probability of generating a state that satisfies the constraint when sampling values according to p. Hence, it is the self-information (or 'surprise') of obtaining an assignment that satisfies the constraint... (Xu et al., 2018)

Semantic loss

- Unpacking $P_\theta(\mathrm{K} = 1)$, we see that:

$$P_\theta(\mathrm{K} = 1) \propto \sum_{\mathbf{w} \models \mathrm{K}} \prod_{\mathbf{w} \models X_i} P_\theta(X_i) \cdot \prod_{\mathbf{w} \models \neg X_i} (1 - P_\theta(X_i))$$

Semantic loss

- Unpacking $P_\theta(\mathrm{K} = 1)$, we see that:

$$P_\theta(\mathrm{K} = 1) \propto \sum_{\mathbf{w} \models \mathrm{K}} \underbrace{\prod_{\mathbf{w} \models X_i} P_\theta(X_i) \cdot \prod_{\mathbf{w} \models \neg X_i} (1 - P_\theta(X_i))}_{\text{score of world } p_\theta(\mathbf{w})}$$

Semantic loss

- Unpacking $P_\theta(\mathrm{K}=1)$, we see that:

$$P_\theta(\mathrm{K}=1) \propto \sum_{\mathbf{w} \models \mathrm{K}} \underbrace{\prod_{\mathbf{w} \models X_i} P_\theta(X_i) \cdot \prod_{\mathbf{w} \models \neg X_i} (1 - P_\theta(X_i))}_{\text{score of world } p_\theta(\mathbf{w})}$$

we can express as a binary probability distribution involving $\mathbf{w}$, or $p_\theta(X_1 = \mathbf{w}_1, ..., X_n = \mathbf{w}_n)$, computed as

$$p_\theta(X_1 = \mathbf{w}_1, ..., X_n = \mathbf{w}_n) = \prod_i^n P_\theta(X_i = \mathbf{w}_i).$$

Semantic loss

- Unpacking $P_\theta(\mathrm{K}=1)$, we see that:

$$P_\theta(\mathrm{K}=1) \propto \sum_{\mathbf{w}\models \mathrm{K}} \underbrace{\prod_{\mathbf{w}\models X_i} P_\theta(X_i) \cdot \prod_{\mathbf{w}\models \neg X_i} (1 - P_\theta(X_i))}_{\text{score of world } p_\theta(\mathbf{w})}$$

we can express as a binary probability distribution involving $\mathbf{w}$, or $p_\theta(X_1 = \mathbf{w}_1, ..., X_n = \mathbf{w}_n)$, computed as

$$p_\theta(X_1 = \mathbf{w}_1, ..., X_n = \mathbf{w}_n) = \prod_i^n P_\theta(X_i = \mathbf{w}_i).$$

makes a common independence assumption, interesting properties studied in van Krieken et al. (2024). This makes $P_\theta(\mathrm{K}=1) = \mathrm{WMC}(\mathrm{K}, \theta)$.

Special cases

Semantic loss: special cases

- Suppose we have unlabeled (p, h) pairs that we want to learn from, how? via **uniqueness constraints** or for each instance (simpl. $P(p, h)$ to P):

$$\mathrm{K} := \big(\mathbf{E} \wedge \neg \mathbf{C} \wedge \neg \mathbf{N}\big) \vee \big(\neg \mathbf{E} \wedge \mathbf{C} \wedge \neg \mathbf{N}\big) \vee \big(\neg \mathbf{E} \wedge \neg \mathbf{C} \wedge \mathbf{N}\big)$$

Semantic loss: special cases

- Suppose we have unlabeled (p, h) pairs that we want to learn from, how? via **uniqueness constraints** or for each instance (simpl. $P(p, h)$ to P):

$$\mathrm{K} := \big(\mathbf{E} \wedge \neg \mathbf{C} \wedge \neg \mathbf{N}\big) \vee \big(\neg \mathbf{E} \wedge \mathbf{C} \wedge \neg \mathbf{N}\big) \vee \big(\neg \mathbf{E} \wedge \neg \mathbf{C} \wedge \mathbf{N}\big)$$

Intuition: A model must confidently assign a consistent class, even to unlabeled examples (Xu et al., 2018), **semi-supervised** learning.

Semantic loss: special cases

- Suppose we have unlabeled (p, h) pairs that we want to learn from, how? via **uniqueness constraints** or for each instance (simpl. $P(p, h)$ to P):

$$\mathsf{K} := \big(\mathbf{E} \wedge \neg\mathbf{C} \wedge \neg\mathbf{N}\big) \vee \big(\neg\mathbf{E} \wedge \mathbf{C} \wedge \neg\mathbf{N}\big) \vee \big(\neg\mathbf{E} \wedge \neg\mathbf{C} \wedge \mathbf{N}\big)$$

Intuition: A model must confidently assign a consistent class, even to unlabeled examples (Xu et al., 2018), **semi-supervised** learning.

We can compute this directly:

$$\mathcal{L}_{\text{uniq}}(\theta, \mathsf{K}) = -\log \underbrace{\sum_{i=1}^{3} P_\theta(X_i) \cdot \prod_{j=1, j\neq i} (1 - P_\theta(X_i))}_{\text{direct } n^2}$$

Semantic loss: special cases

$$\mathsf{K} := (\mathbf{E} \wedge \neg \mathbf{C} \wedge \neg \mathbf{N}) \vee (\neg \mathbf{E} \wedge \mathbf{C} \wedge \neg \mathbf{N}) \vee (\neg \mathbf{E} \wedge \neg \mathbf{C} \wedge \mathbf{N})$$

We can compute this directly:

$$\mathcal{L}_{\text{uniq}}(\theta, \mathsf{K}) = -\log \sum_{i=1}^{3} P_\theta(X_i) \cdot \prod_{j=1, j \neq i} (1 - P_\theta(X_i))$$

- Suppose for K that we have weights θ:

$$P_\theta(E), (C), P_\theta(N) = [0.3, 0.2, 0.5]$$

then the loss is equal to

$$-\log \left(\underbrace{(0.3 * 0.8 * 0.5)}_{\mathbf{E} \wedge \neg \mathbf{C} \wedge \neg \mathbf{N}} + (0.2 * 0.7 * 0.5) + (0.5 * 0.7 * 0.8)) \right)$$

Model sketch in Pytorch: semi-supervised learning

```
unlabel_p = "John is weearing a red shirt"
unlabel_h = "John is wearing a yellow shirt"
batch_data = tokenizer([(p,h),(unlabel_p,unlabel_h)],
        return_tensors="pt",padding=True)
labels = torch.tensor([[1],[-100]])

label_loss,probs = NLIModel({
    "input_ids": batch_data.input_ids,
    "labels"   : labels,
})
unlab_probs = probs[1:]
wmc = torch.zeros_like(unlab_probs)
for i in range(3):
    negate      = 1 - unlab_probs
    negate[:,i] = unlab_probs[:,i]
    wmc[:,i]    = negate.prod(dim=-1)

wmc_loss = -1*torch.log(wmc.sum(dim=-1)).squeeze(-1)
```

original implementation, pytorch version version (version above)

Model sketch in Pytorch: semi-supervised learning

```
unlab_probs = probs[1:]
wmc = torch.zeros_like(unlab_probs)
for i in range(3):
    negate       = 1 - unlab_probs
    negate[:,i] = unlab_probs[:,i]
    wmc[:,i]     = negate.prod(dim=-1)

wmc_loss = -1*torch.log(wmc.sum(dim=-1)).mean()

print(wmc.requires_grad)
## => True
print(wmc_loss)
## tensor(0.8082, grad_fn=<MulBackward0>)
```

some notes on autodiff, more notes.

Semantic loss: other special cases

- **Supervised learning**: suppose we have two gold annotations $\mathbf{E}_1(p_1, h_1)$ and $\mathbf{C}_2(p_2, h_2)$ *(no additional variables)* and the formula:

$$\mathrm{K} := \mathbf{E}_1(p_1, h_1) \wedge \mathbf{C}_2(p_2, h_2).$$

Semantic loss: other special cases

- **Supervised learning**: suppose we have two gold annotations $\mathbf{E}_1(p_1, h_1)$ and $\mathbf{C}_2(p_2, h_2)$ *(no additional variables)* and the formula:

$$\mathrm{K} := \mathbf{E}_1(p_1, h_1) \wedge \mathbf{C}_2(p_2, h_2).$$

Performing counting of this gives the following:

	Worlds	**Satisfies K?**
$\mathbf{w}_1$	$\mathbf{E}_1, \mathbf{C}_2$	True
$\mathbf{w}_2$	$\neg\mathbf{E}_1, \mathbf{C}_2$	False
$\mathbf{w}_3$	$\mathbf{E}_1, \neg\mathbf{C}_2$	False
$\mathbf{w}_4$	$\neg\mathbf{E}_1, \neg\mathbf{C}_2$	False

Semantic loss: other special cases

- **Supervised learning**: suppose we have two gold annotations $\mathbf{E}_1(p_1, h_1)$ and $\mathbf{C}_2(p_2, h_2)$ *(no additional variables)* and the formula:

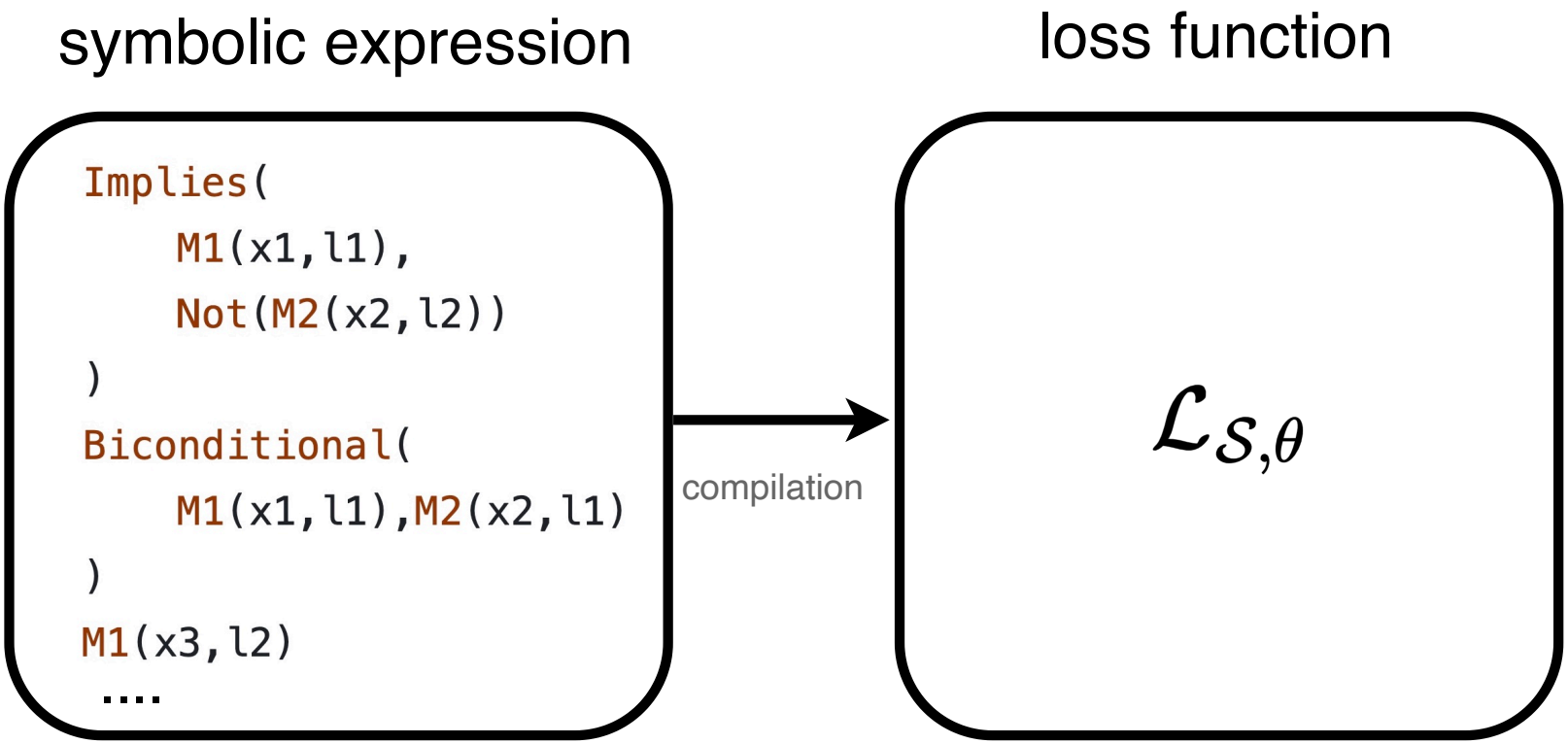
$$\mathrm{K} := \mathbf{E}_1(p_1, h_1) \wedge \mathbf{C}_2(p_2, h_2).$$

Performing counting of this gives the following:

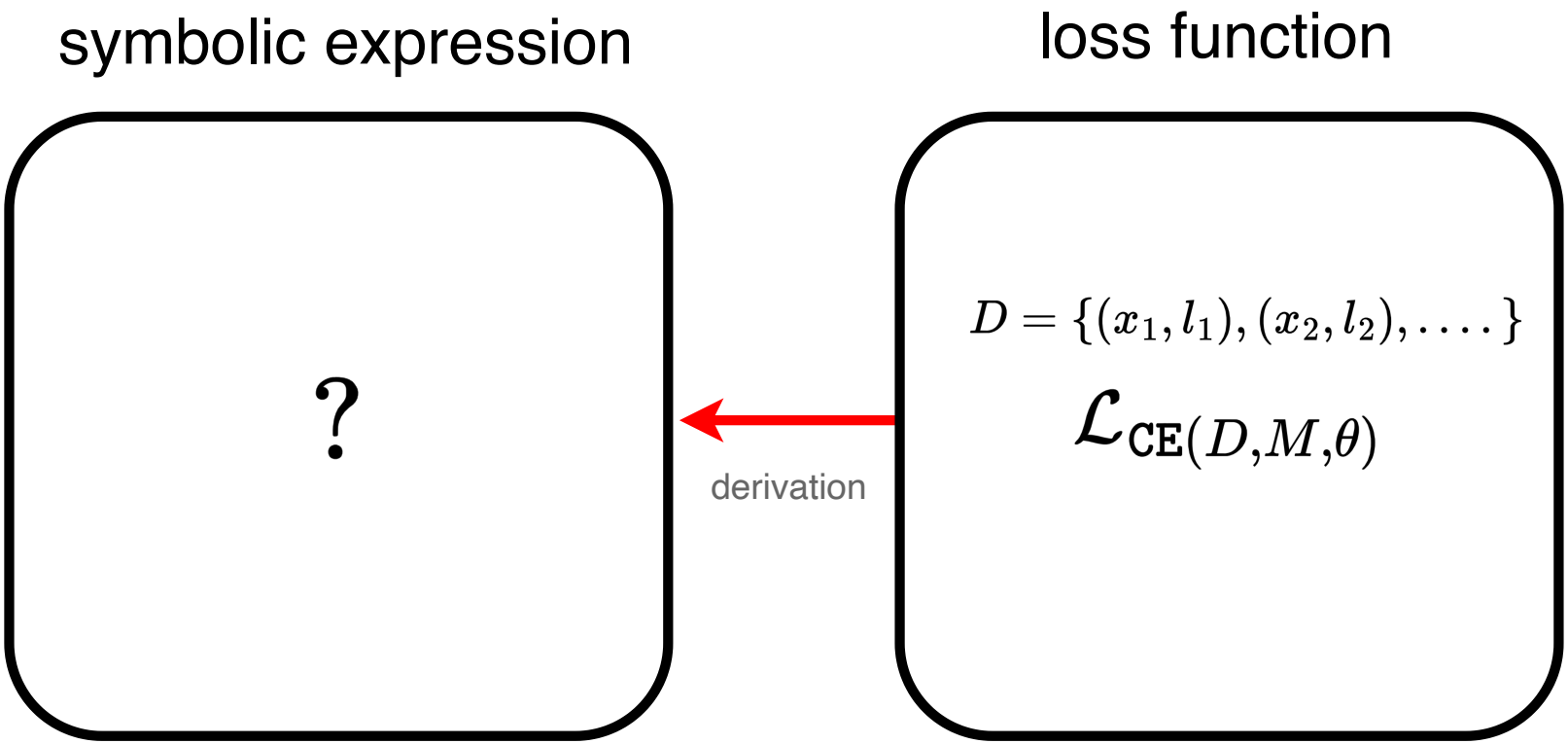
	Worlds	**Satisfies K?**
$\mathbf{w}_1$	$\mathbf{E}_1, \mathbf{C}_2$	True
$\mathbf{w}_2$	$\neg\mathbf{E}_1, \mathbf{C}_2$	False
$\mathbf{w}_3$	$\mathbf{E}_1, \neg\mathbf{C}_2$	False
$\mathbf{w}_4$	$\neg\mathbf{E}_1, \neg\mathbf{C}_2$	False

where only a single model gets counted, yielding a semantic loss of $-\log(\mathbf{p}_1 * \mathbf{p}_2)$ equal to the **cross-entropy loss**.

Going in the reverse direction

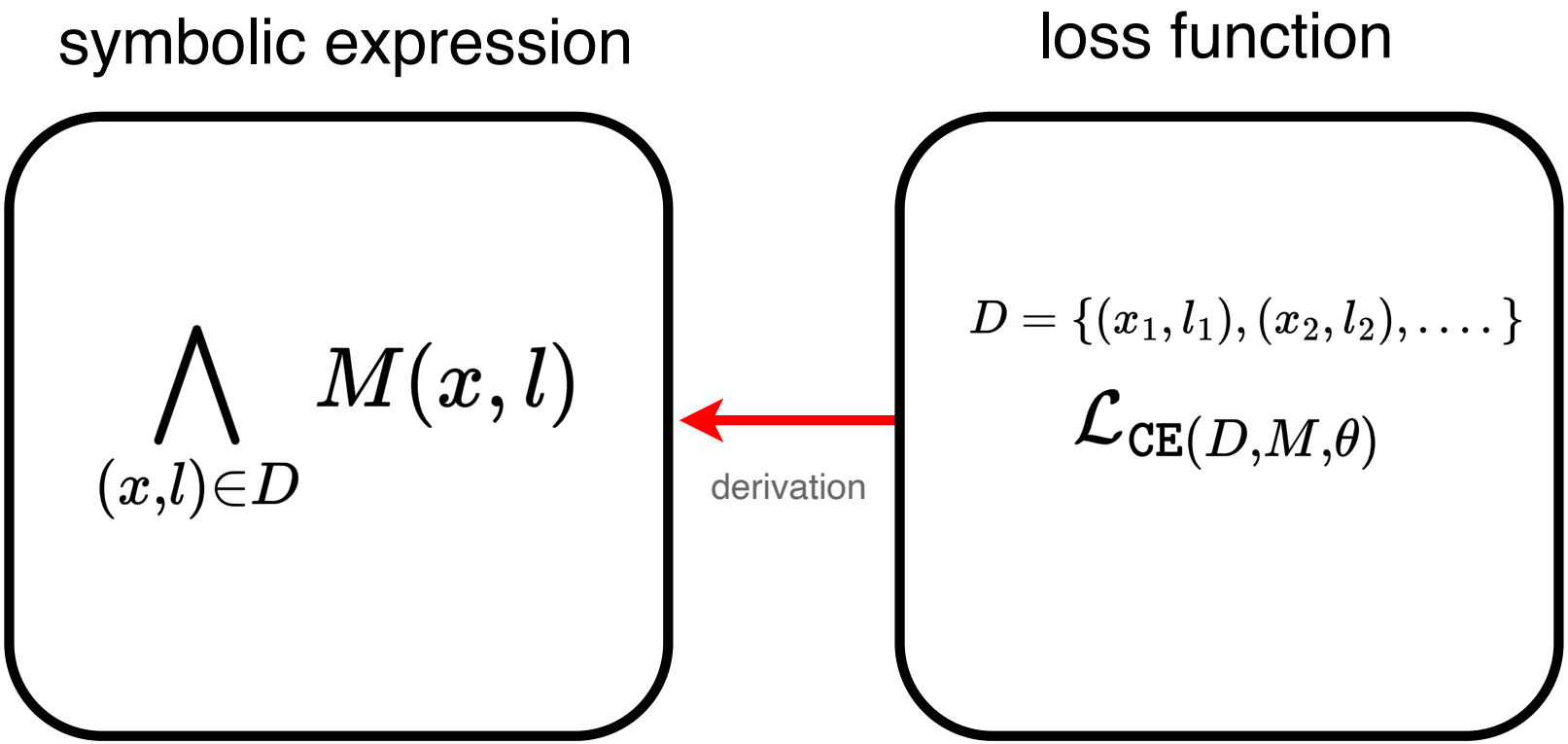


Going in the reverse direction

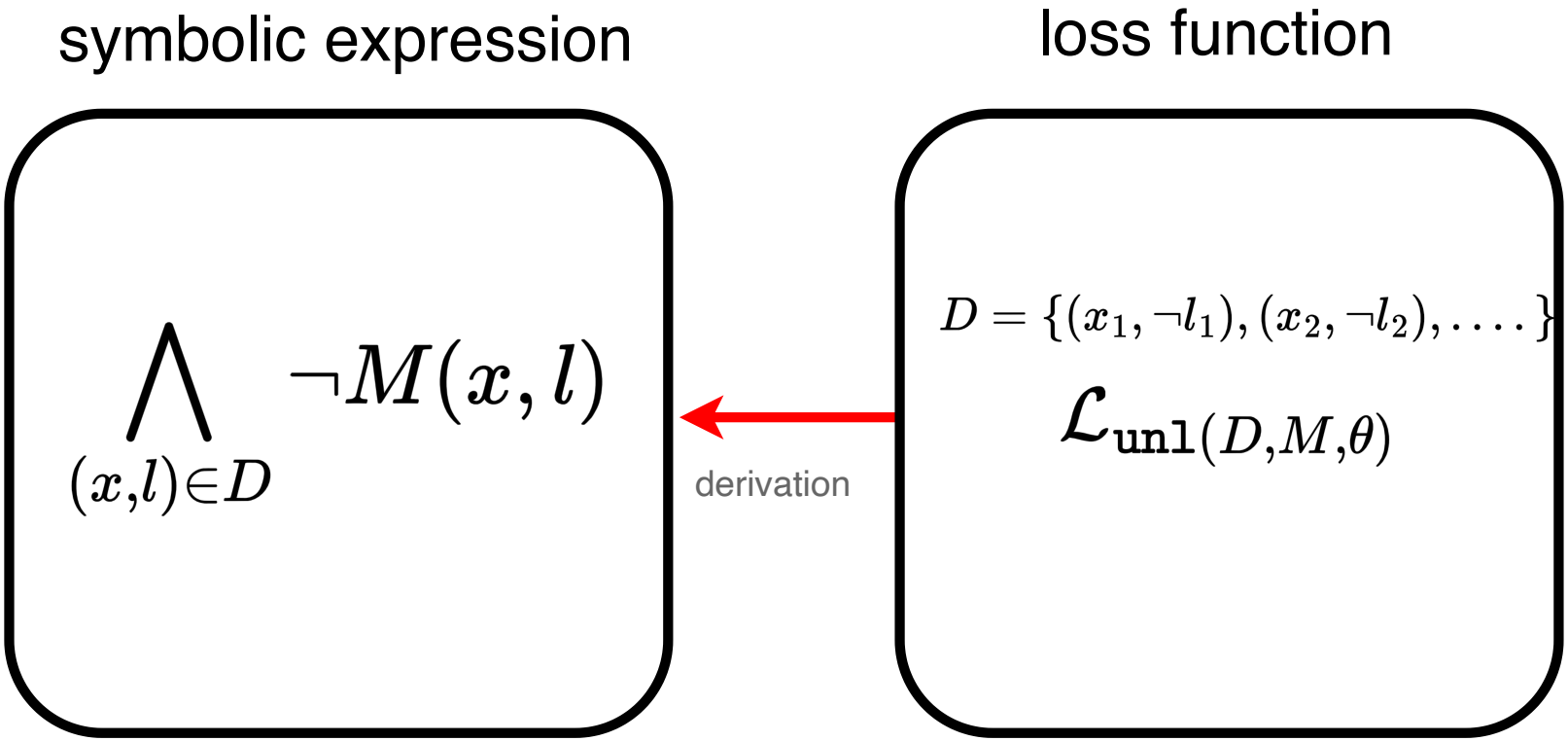


- We can also try to decompile from a known loss function to a symbolic expression, help to understand the semantics of known loss functions.

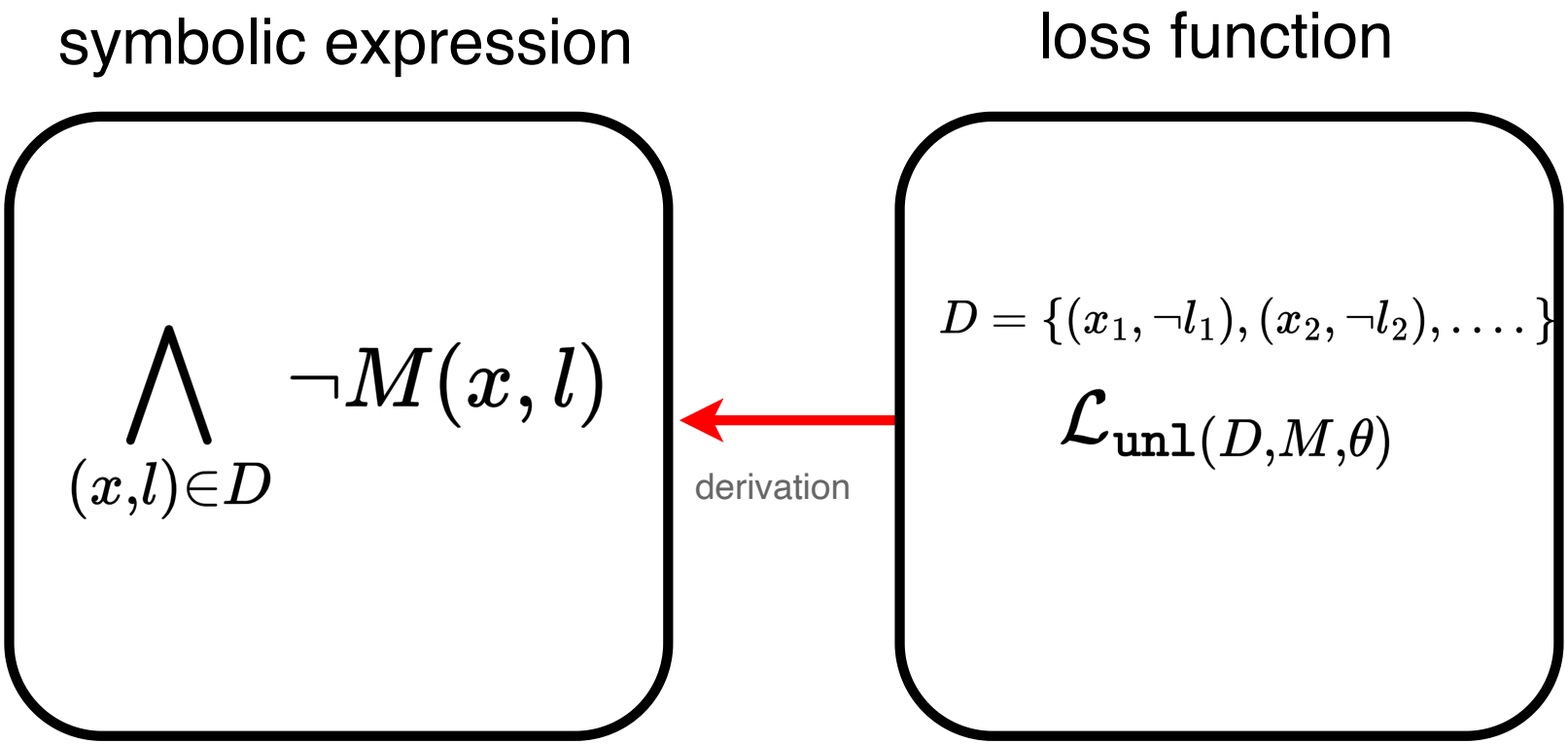
Going in the reverse direction



Going in the reverse direction

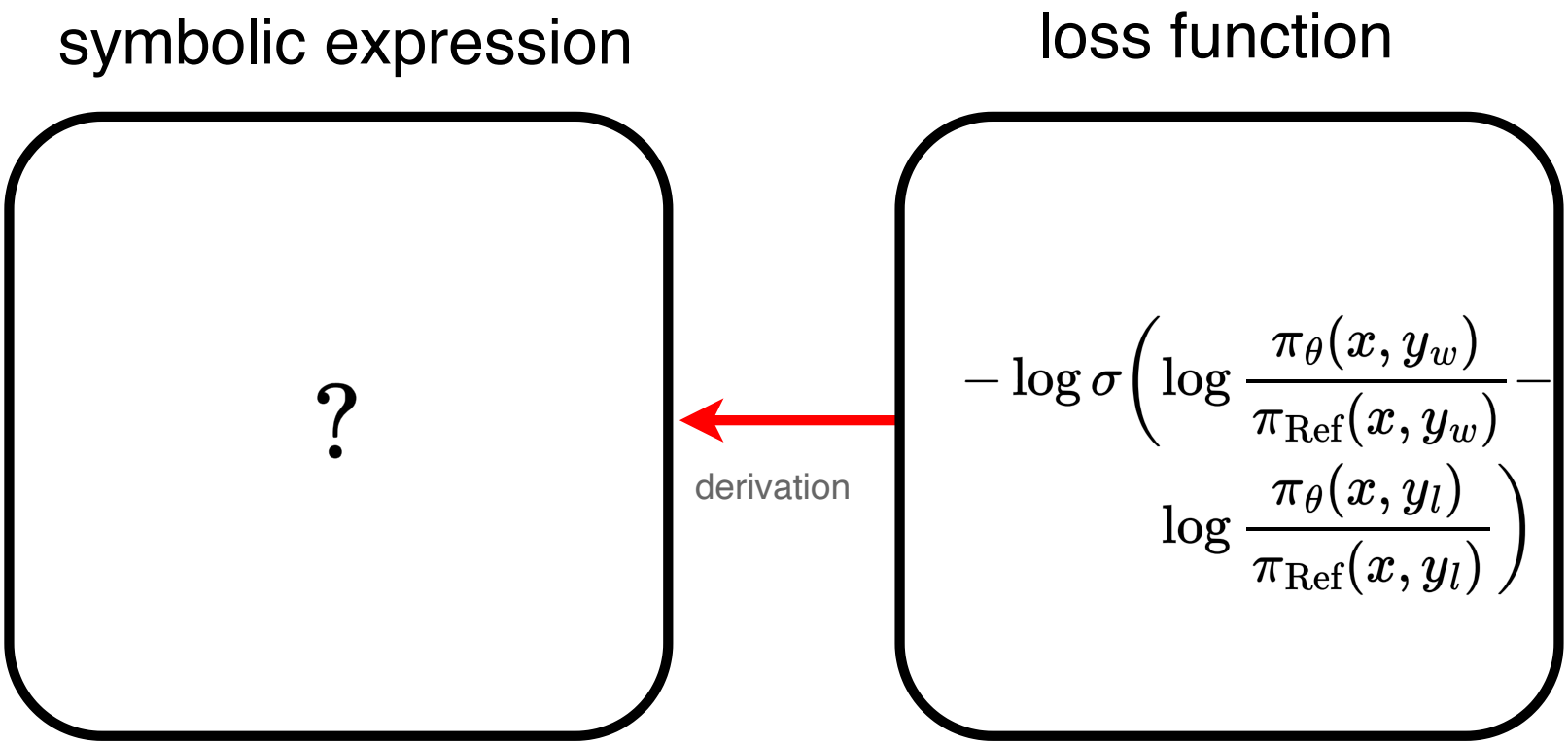


Going in the reverse direction



observation: standard loss (e.g., cross entropy) functions have a fairly simple logical semantics (Li et al., 2019; Giannini et al., 2023).

Going in the reverse direction



Formal properties of semantic loss

- The **semantic loss** $\mathcal{L}_{\mathsf{sl}}$ for K is:

$$\mathcal{L}_{\mathsf{sl}}(\mathsf{K}, \theta) := -\log P_\theta(\mathsf{K} = 1)$$

Satisfies basic and intuitive semantic properties

Formal properties of semantic loss

- The **semantic loss** $\mathcal{L}_{\mathsf{sl}}$ for K is:

$$\mathcal{L}_{\mathsf{sl}}(\mathsf{K}, \theta) := -\log P_\theta(\mathsf{K} = 1)$$

Satisfies basic and intuitive semantic properties

(monotonicity) If $\mathsf{K}_1 \models \mathsf{K}_2$, then $\mathcal{L}_{\mathsf{sl}}(\mathsf{K}_1, \theta) \geq \mathcal{L}_{\mathsf{sl}}(\mathsf{K}_2, \theta)$ for any θ

Formal properties of semantic loss

- The **semantic loss** $\mathcal{L}_{\mathsf{sl}}$ for K is:

$$\mathcal{L}_{\mathsf{sl}}(\mathsf{K}, \theta) := -\log P_\theta(\mathsf{K} = 1)$$

Satisfies basic and intuitive semantic properties

(monotonicity) If $\mathsf{K}_1 \models \mathsf{K}_2$, then $\mathcal{L}_{\mathsf{sl}}(\mathsf{K}_1, \theta) \geq \mathcal{L}_{\mathsf{sl}}(\mathsf{K}_2, \theta)$ for any θ
(equivalence) If $\mathsf{K}_1 \equiv \mathsf{K}_2$, then $\mathcal{L}_{\mathsf{sl}}(\mathsf{K}_1, \theta) = \mathcal{L}_{\mathsf{sl}}(\mathsf{K}_2, \theta)$ for any θ

Formal properties of semantic loss

- The **semantic loss** $\mathcal{L}_{\mathsf{sl}}$ for K is:

$$\mathcal{L}_{\mathsf{sl}}(\mathsf{K}, \theta) := -\log P_\theta(\mathsf{K} = 1)$$

Satisfies basic and intuitive semantic properties

(monotonicity) If $\mathsf{K}_1 \models \mathsf{K}_2$, then $\mathcal{L}_{\mathsf{sl}}(\mathsf{K}_1, \theta) \geq \mathcal{L}_{\mathsf{sl}}(\mathsf{K}_2, \theta)$ for any θ
(equivalence) If $\mathsf{K}_1 \equiv \mathsf{K}_2$, then $\mathcal{L}_{\mathsf{sl}}(\mathsf{K}_1, \theta) = \mathcal{L}_{\mathsf{sl}}(\mathsf{K}_2, \theta)$ for any θ

(Differentiability). For any fixed K, the semantic loss of $\mathcal{L}(\mathsf{K}, \theta)$ is monotone in each probability in θ, continuous and differentiable.

Formal properties of semantic loss

- The **semantic loss** $\mathcal{L}_{\mathsf{sl}}$ for K is:

$$\mathcal{L}_{\mathsf{sl}}(\mathsf{K}, \theta) := -\log P_\theta(\mathsf{K} = 1)$$

Satisfies basic and intuitive semantic properties

(monotonicity) If $\mathsf{K}_1 \models \mathsf{K}_2$, then $\mathcal{L}_{\mathsf{sl}}(\mathsf{K}_1, \theta) \geq \mathcal{L}_{\mathsf{sl}}(\mathsf{K}_2, \theta)$ for any θ
(equivalence) If $\mathsf{K}_1 \equiv \mathsf{K}_2$, then $\mathcal{L}_{\mathsf{sl}}(\mathsf{K}_1, \theta) = \mathcal{L}_{\mathsf{sl}}(\mathsf{K}_2, \theta)$ for any θ

(Differentiability). For any fixed K, the semantic loss of $\mathcal{L}(\mathsf{K}, \theta)$ is monotone in each probability in θ, continuous and differentiable.

Can be useful for reasoning about the relationship between different losses, both semantically and in terms of their relative loss behavior.

Interim summary

- Using semantic loss, we can now use logic as a language for specifying loss functions, declarative style model development:

Interim summary

- Using semantic loss, we can now use logic as a language for specifying loss functions, declarative style model development:

  Ordinary machine learning (see Welleck et al. (2019); Grandvalet and Bengio (2004))

$$\forall (t, P) \in D \quad \bigwedge P(t) \qquad (\textit{cross-entropy})$$

$$\forall (t, P) \in D_{\text{neg}} \quad \bigwedge \neg P(t) \qquad (\textit{unlikelihood loss})$$

$$\forall (t, P_1, P_2) \in D \quad \bigwedge (P_1 \wedge \neg P_2) \vee (\neg P_1 \wedge P_2) \qquad (\textit{semi-supervised learning})$$

Interim summary

- Using semantic loss, we can now use logic as a language for specifying loss functions, declarative style model development:

  Ordinary machine learning (see Welleck et al. (2019); Grandvalet and Bengio (2004))

  $$\forall (t, P) \in D \qquad \bigwedge P(t) \qquad \textit{(cross-entropy)}$$

  $$\forall (t, P) \in D_{\text{neg}} \qquad \bigwedge \neg P(t) \qquad \textit{(unlikelihood loss)}$$

  $$\forall (t, P_1, P_2) \in D \qquad \bigwedge (P_1 \wedge \neg P_2) \vee (\neg P_1 \wedge P_2) \qquad \textit{(semi-supervised learning)}$$

  Beyond ordinary ML

  $$\forall (t, t', P) \in D \qquad \bigwedge P_1(t) \leftrightarrow P_1(t') \qquad \textit{(symmetry/invariance)}$$

  $$\forall (t_1, t_2, P_1, P_2) \in D \qquad \bigwedge P_1(t_1) \rightarrow P_2(t_2) \qquad \text{(learning from pairs)}$$

Interim summary

- Using semantic loss, we can now use logic as a language for specifying loss functions, declarative style model development:

  Beyond ordinary ML

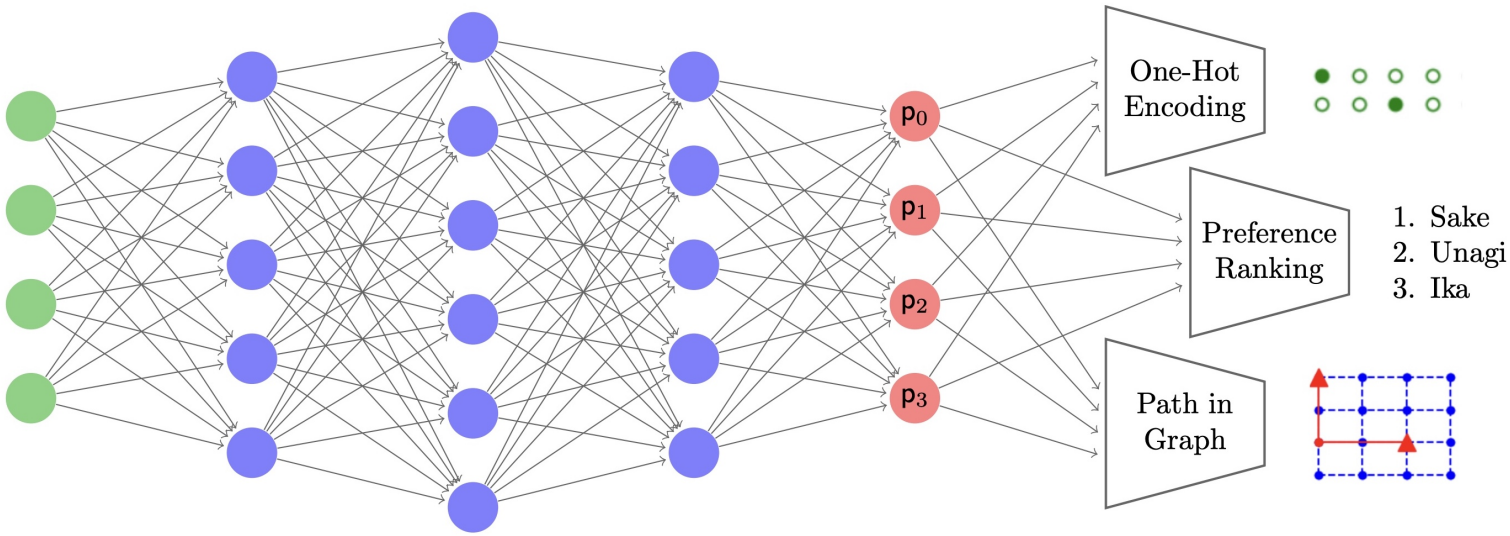
  $$\forall(t, t', P) \in D \qquad \bigwedge P_1(t) \leftrightarrow P_1(t') \qquad \textit{(symmetry/invariance)}$$

  $$\forall(t_1, t_2, P_1, P_2) \in D \quad \bigwedge P_1(t_1) \rightarrow P_2(t_2) \qquad \text{(learning from pairs)}$$

Whatever we can express in (propositional) logic can now be a loss function, many new possibilities for programming models.

Questions?

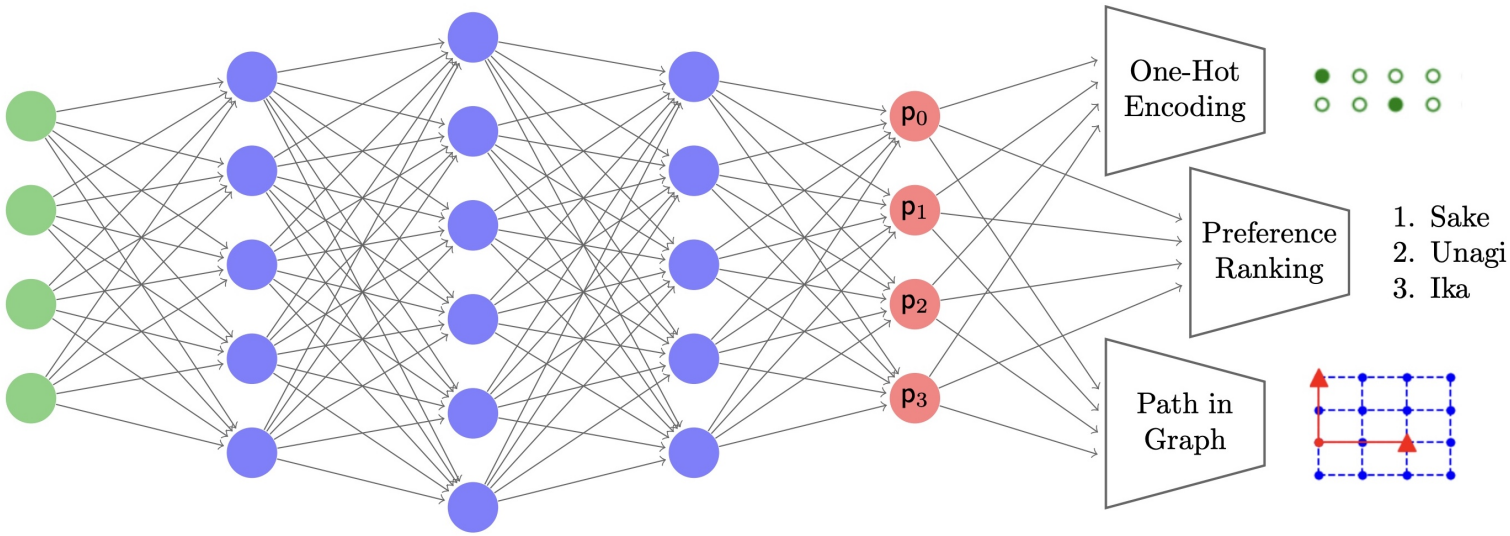
Other uses of semantic loss



from Xu et al. (2018)

- Can be used for *structured prediction*, jointly predicting many output variables for building structured objects (Ahmed et al., 2022b).

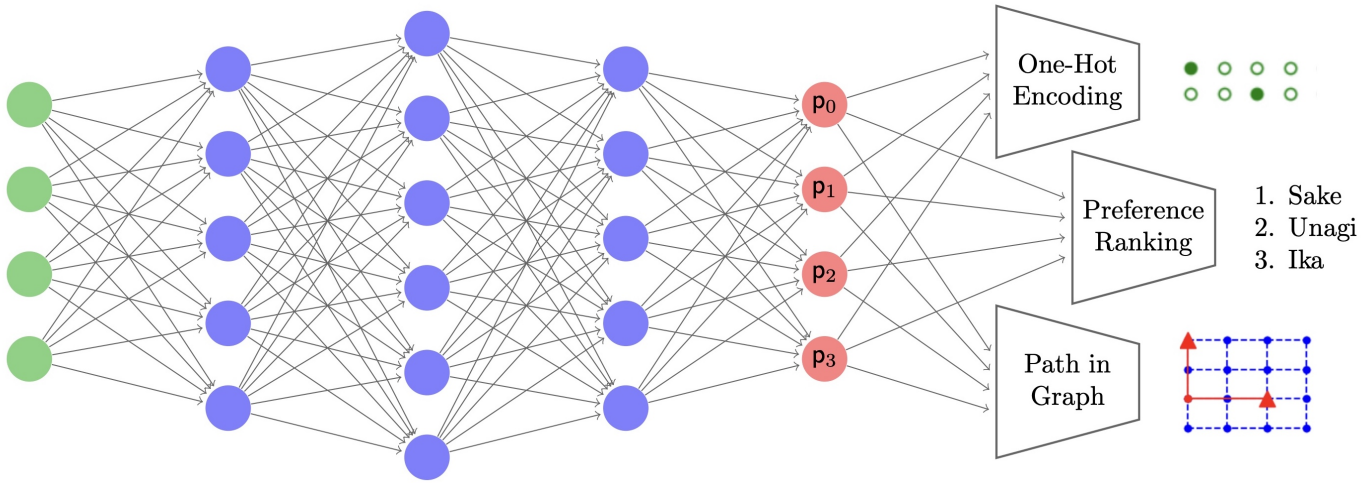
Other uses of semantic loss



from Xu et al. (2018)

- Can be used for *structured prediction*, jointly predicting many output variables for building structured objects (Ahmed et al., 2022b).

Other uses of semantic loss



- **Preference learning**: Ranking preferences, e.g., (Choi et al., 2015) given variables P_1, P_2, P_3, binary matrix $P_{i,j}$ $(i, j \in \{1, ..., 3\})$ and a total order:

$$\forall i.((P_{i,1} \wedge \neg P_{i,2}, \wedge \neg P_{i,3}) \vee$$
$$(\neg P_{i,1} \wedge P_{i,2}, \wedge \neg P_{i,3}) \vee$$
$$(\neg P_{i,1} \wedge \neg P_{i,2}, \wedge P_{i,3}))$$

$$\forall j.((P_{1,j} \wedge \neg P_{2,j} \wedge \neg P_{3,j}) \vee$$
$$(\neg P_{1,j} \wedge P_{2,j} \wedge \neg P_{3,j}) \vee$$
$$(\neg P_{1,j} \wedge \neg P_{2,j} \wedge P_{3,j}))$$

How the semantic loss is used

```
loss_weight = 0.3
wmc_loss = -1*torch.log(wmc.sum(dim=-1)).mean()
loss = label_loss + loss_weight*wmc_loss
```

- Usually coupled with ordinary loss, or as a *regularizer*

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda \mathcal{L}_{\text{sl}}(\text{K})$$

How the semantic loss is used

```
loss_weight = 0.3
wmc_loss = -1*torch.log(wmc.sum(dim=-1)).mean()
loss = label_loss + loss_weight*wmc_loss
```

- Usually coupled with ordinary loss, or as a *regularizer*

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda \mathcal{L}_{\text{sl}}(\text{K})$$

From Marra et al. (2021)

> [This] class of NeSy is model-based. They use logic to define a loss function (usually a regularization term) for neural networks. The networks compute scores for a set of atoms... At each training step the logic-based loss function checks whether the assigned scores satisfy the logical theory and computes a corresponding penalty. **Logical inference is therefore turned into a learning problem** (i.e. **"learning to satisfy"**)

When "learning to satisfy" can go wrong

- Suppose that consider $\mathbf{C}_1$ and $\mathbf{C}_2$ are the correct analyses and we optimize our model for:

$$\forall(p, h).\, \mathbf{C}(p, h) \leftrightarrow \mathbf{C}(h, p)$$

with semantics:

	Worlds	Satisfies F?	Correct?
w_1	$\neg\mathbf{E}_1, \mathbf{C}_1, \neg\mathbf{N}_1, \neg\mathbf{E}_2, \mathbf{C}_2, \neg\mathbf{N}_2$	yes	yes
w_2	$\neg\mathbf{E}_1, \mathbf{C}_1, \neg\mathbf{N}_1, \neg\mathbf{E}_2, \neg\mathbf{C}_2, \mathbf{N}_2$	no	no
w_3	$\neg\mathbf{E}_1, \mathbf{C}_1, \neg\mathbf{N}_1, \mathbf{E}_2, \neg\mathbf{C}_2, \neg\mathbf{N}_2$	no	no
w_4	$\neg\mathbf{E}_1, \neg\mathbf{C}_1, \mathbf{N}_1, \neg\mathbf{E}_2, \mathbf{C}_2, \neg\mathbf{N}_2$	no	no
w_5	$\neg\mathbf{E}_1, \neg\mathbf{C}_1, \mathbf{N}_1, \neg\mathbf{E}_2, \neg\mathbf{C}_2, \mathbf{N}_2$	yes	no
w_6	$\neg\mathbf{E}_1, \neg\mathbf{C}_1, \mathbf{N}_1, \mathbf{E}_2, \neg\mathbf{C}_2, \neg\mathbf{N}_2$	yes	no
w_7	$\mathbf{E}_1, \neg\mathbf{C}_1, \neg\mathbf{N}_1, \neg\mathbf{E}_2, \mathbf{C}_2, \neg\mathbf{N}_2$	no	no
w_8	$\mathbf{E}_1, \neg\mathbf{C}_1, \neg\mathbf{N}_1, \neg\mathbf{E}_2, \neg\mathbf{C}_2, \mathbf{N}_2$	yes	no
w_9	$\mathbf{E}_1, \neg\mathbf{C}_1, \neg\mathbf{N}_1, \mathbf{E}_2, \neg\mathbf{C}_2, \neg\mathbf{N}_2$	yes	no
		count: 5	

When "learning to satisfy" can go wrong

- Suppose that consider $\mathbf{C}_1$ and $\mathbf{C}_2$ are the correct analyses and we optimize our model for:

$$\forall(p, h).\, \mathbf{C}(p, h) \leftrightarrow \mathbf{C}(h, p)$$

with semantics:

	Worlds	Satisfies F?	Correct?
w_1	$\neg\mathbf{E}_1, \mathbf{C}_1, \neg\mathbf{N}_1, \neg\mathbf{E}_2, \mathbf{C}_2, \neg\mathbf{N}_2$	yes	yes
w_2	$\neg\mathbf{E}_1, \mathbf{C}_1, \neg\mathbf{N}_1, \neg\mathbf{E}_2, \neg\mathbf{C}_2, \mathbf{N}_2$	no	no
w_3	$\neg\mathbf{E}_1, \mathbf{C}_1, \neg\mathbf{N}_1, \mathbf{E}_2, \neg\mathbf{C}_2, \neg\mathbf{N}_2$	no	no
w_4	$\neg\mathbf{E}_1, \neg\mathbf{C}_1, \mathbf{N}_1, \neg\mathbf{E}_2, \mathbf{C}_2, \neg\mathbf{N}_2$	no	no
w_5	$\neg\mathbf{E}_1, \neg\mathbf{C}_1, \mathbf{N}_1, \neg\mathbf{E}_2, \neg\mathbf{C}_2, \mathbf{N}_2$	yes	no
w_6	$\neg\mathbf{E}_1, \neg\mathbf{C}_1, \mathbf{N}_1, \mathbf{E}_2, \neg\mathbf{C}_2, \neg\mathbf{N}_2$	yes	no
w_7	$\mathbf{E}_1, \neg\mathbf{C}_1, \neg\mathbf{N}_1, \neg\mathbf{E}_2, \mathbf{C}_2, \neg\mathbf{N}_2$	no	no
w_8	$\mathbf{E}_1, \neg\mathbf{C}_1, \neg\mathbf{N}_1, \neg\mathbf{E}_2, \neg\mathbf{C}_2, \mathbf{N}_2$	yes	no
w_9	$\mathbf{E}_1, \neg\mathbf{C}_1, \neg\mathbf{N}_1, \mathbf{E}_2, \neg\mathbf{C}_2, \neg\mathbf{N}_2$	yes	no
		count: 5	

notice: Most satisfying models are models where both are false; will nudge model towards this analysis.

Shortcut learning

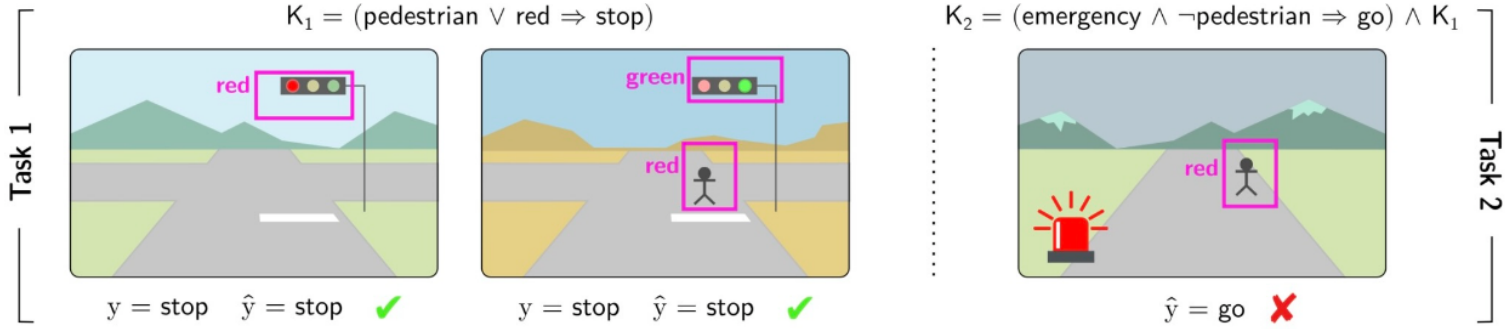


Figure 1: **Reasoning shortcuts undermine trustworthiness**. An autonomous vehicle has to decide whether to $Y =$ `stop` or $Y =$ `go` based on three binary concepts extracted from an image $\mathbf{x}$, namely $C_1 =$ `red` light, $C_2 =$ `green` light and $C_3 =$ presence of `pedestrians` (shown in **pink**). **Left**: In Task 1, the prior knowledge $\mathsf{K} = (\texttt{pedestrian} \vee \texttt{red} \Rightarrow \texttt{stop})$ instructs the vehicle to stop whenever the light is red or there are pedestrians on the road. The model can perfectly classify an (even exhaustive) training set by acquiring a *reasoning shortcut that classifies pedestrians as red lights*. **Right**: The learned concepts are then reused to guide an autonomous ambulance with the additional rule that in emergency situations red lights can be ignored, with potentially dire consequences. Our work identifies the causes of RSs (Section 4) and several mitigation strategies (Section 5).

From Marconato et al. (2024)

Why does this happen?

- Related to its independence assumption, makes the model prefer *deterministic* solutions (van Krieken et al., 2024).

Why does this happen?

- Related to its independence assumption, makes the model prefer *deterministic* solutions (van Krieken et al., 2024).

  Traffic light problem: We have a light that can be green $p(g)$ or red $p(r)$, constraint

  $$\neg(p(g) \wedge p(r))$$

Why does this happen?

- Related to its independence assumption, makes the model prefer *deterministic* solutions (van Krieken et al., 2024).

  Traffic light problem: We have a light that can be green $p(g)$ or red $p(r)$, constraint

$$\neg(p(g) \wedge p(r))$$

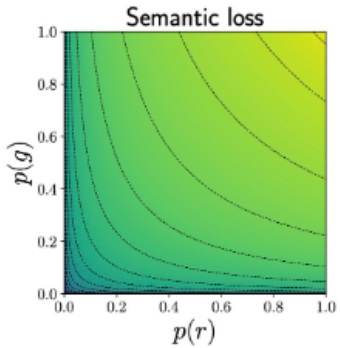


Figure 2. The loss landscape of the semantic loss for the traffic light problem – brighter (resp. darker) regions correspond to higher (resp. lower) semantic loss values.

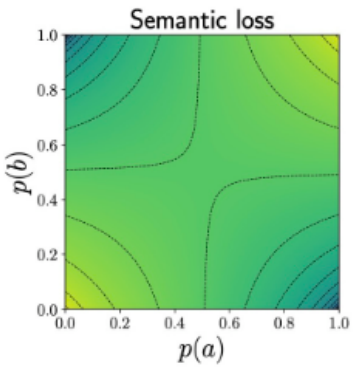
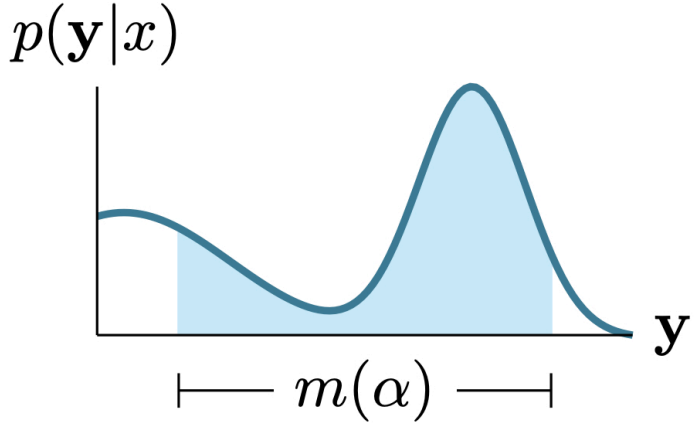


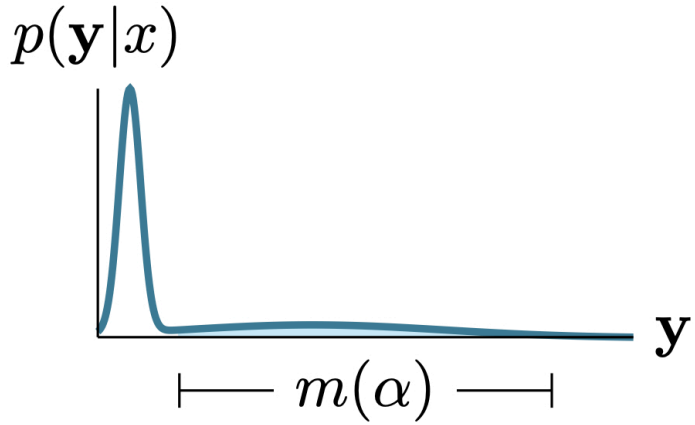
Figure 6. The loss landscape of the semantic loss under the independence assumption for the XOR formula $\varphi = (a \wedge \neg b) \vee (\neg a \wedge b)$.

Theorem 3.1 (Implicants determine minima, informal)**.** *An independent distribution $p_{\boldsymbol{\theta}}(\mathbf{w})$ minimises the semantic loss if and only if it is deterministic for some variables, and those variables form an implicant of φ.*

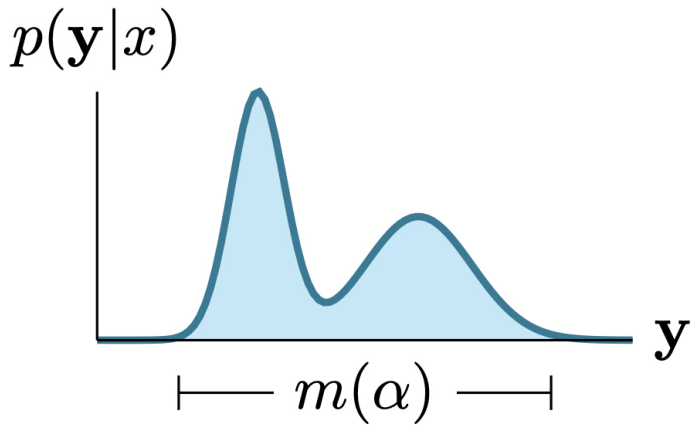
Another view

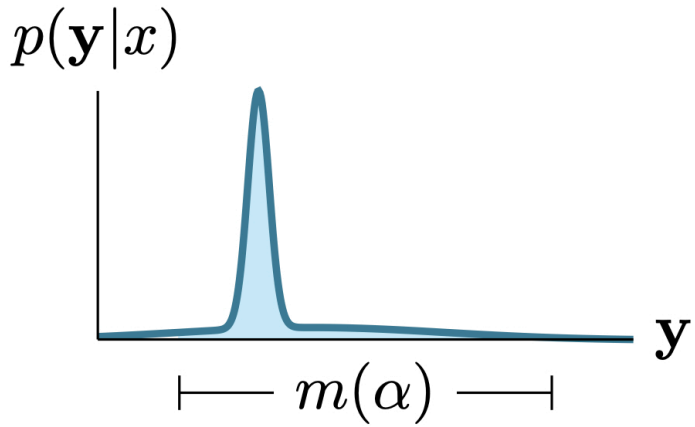


(a) Network uncertain over both valid and invalid predictions



(b) Network allocating most mass to one invalid prediction



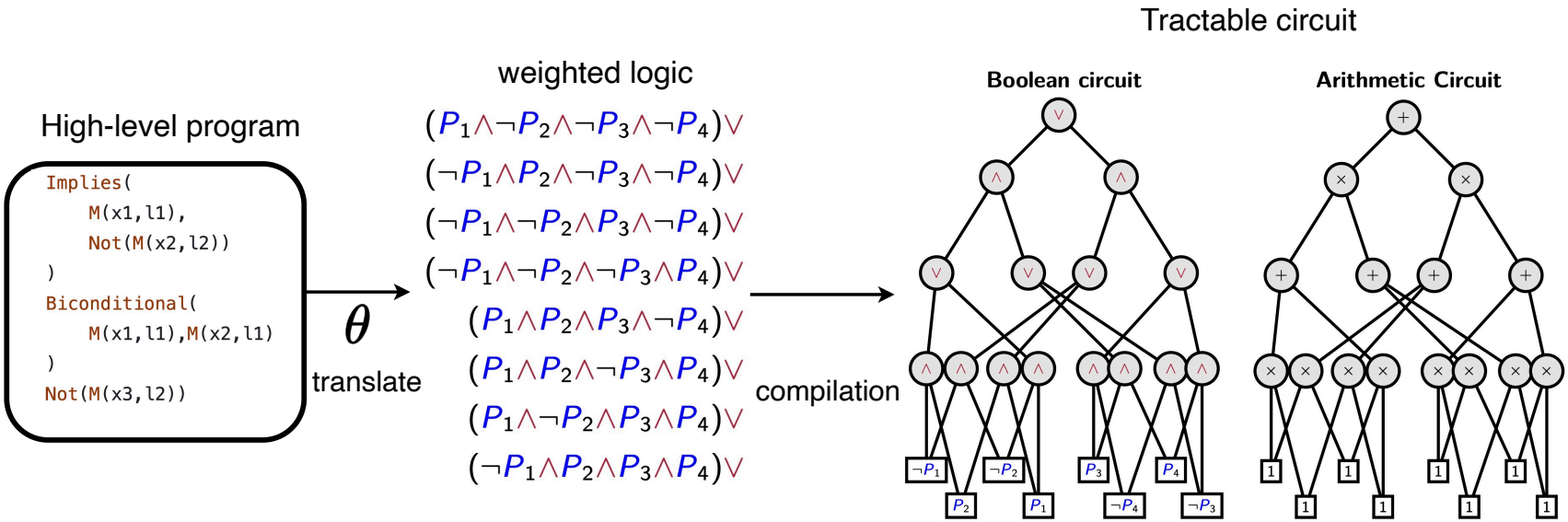


From Ahmed et al. (2022b)

practical bottleneck: computing

$\mathrm{WMC}(\mathsf{K}, \theta)$

How this works in practice



- **Knowledge compilation**: tractable representations for efficient logical and probabilistic inference (Darwiche and Marquis, 2002; Marquis, 2008).

Problog

```
import problog as p                    ## pip install problog
P = p.program.PrologString("""
0.8::stress(ann).
0.4::stress(bob).
0.6::influences(ann,bob).
0.2::influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :- influences(Y,X), smokes(Y).

query(smokes(carl))."""")

lf = p.formula.LogicFormula.create_from(P)#ground
dag = f.formula.LogicDAG.create_from(lf)# break cycles
cnf = p.cnf_formula.CNF.create_from(dag)# convert CNF
ddnnf = p.ddnf.DDNNF.create_from(cnf)# compile circuit

ddnnf.evaluate()
```

Creating tractable representations

- **Uniqueness constraint** would be commonly written in *conjunctive normal form* (CNF):

$$\underbrace{(\mathbf{E} \vee \mathbf{N} \vee \mathbf{C})}_{\text{clause}} \wedge (\neg \mathbf{E} \vee \neg \mathbf{N}) \wedge (\neg \mathbf{N} \vee \neg \mathbf{C}) \wedge (\neg \mathbf{E} \vee \neg \mathbf{N})$$

Creating tractable representations

- **Uniqueness constraint** would be commonly written in *conjunctive normal form* (CNF):

$$\underbrace{(\mathbf{E} \vee \mathbf{N} \vee \mathbf{C})}_{\text{clause}} \wedge (\neg \mathbf{E} \vee \neg \mathbf{N}) \wedge (\neg \mathbf{N} \vee \neg \mathbf{C}) \wedge (\neg \mathbf{E} \vee \neg \mathbf{N})$$

We could also write it in the following form:

$$\big(\mathbf{E} \wedge \neg \mathbf{N} \wedge \neg \mathbf{C}\big) \vee \big(\neg \mathbf{E} \wedge \mathbf{C} \wedge \neg \mathbf{N}\big) \vee \big(\neg \mathbf{E} \wedge \neg \mathbf{C} \wedge \mathbf{N}\big)$$

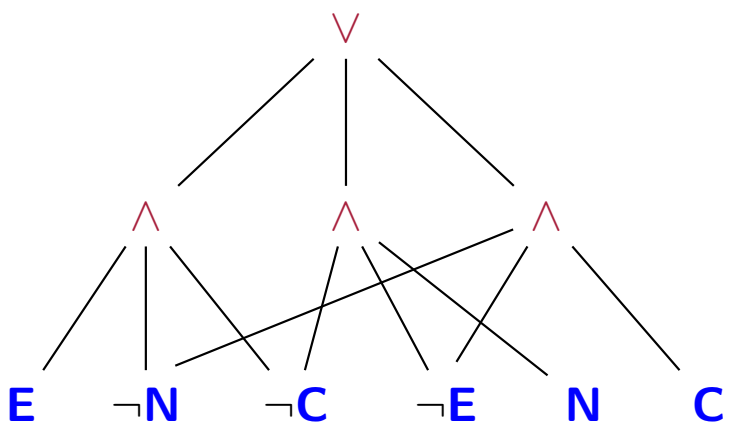
Creating tractable representations

- **Uniqueness constraint** would be commonly written in *conjunctive normal form* (CNF):

$$\underbrace{(\mathbf{E} \vee \mathbf{N} \vee \mathbf{C})}_{\text{clause}} \wedge (\neg\mathbf{E} \vee \neg\mathbf{N}) \wedge (\neg\mathbf{N} \vee \neg\mathbf{C}) \wedge (\neg\mathbf{E} \vee \neg\mathbf{N})$$

We could also write it in the following form:

$$\big(\mathbf{E} \wedge \neg\mathbf{N} \wedge \neg\mathbf{C}\big) \vee \big(\neg\mathbf{E} \wedge \mathbf{C} \wedge \neg\mathbf{N}\big) \vee \big(\neg\mathbf{E} \wedge \neg\mathbf{C} \wedge \mathbf{N}\big)$$



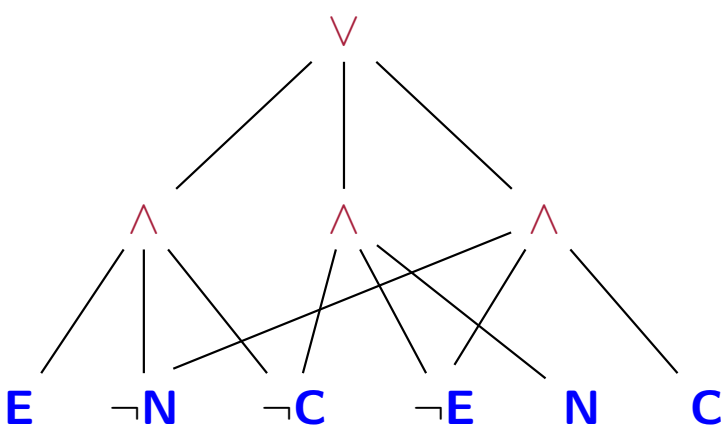
Creating tractable representations

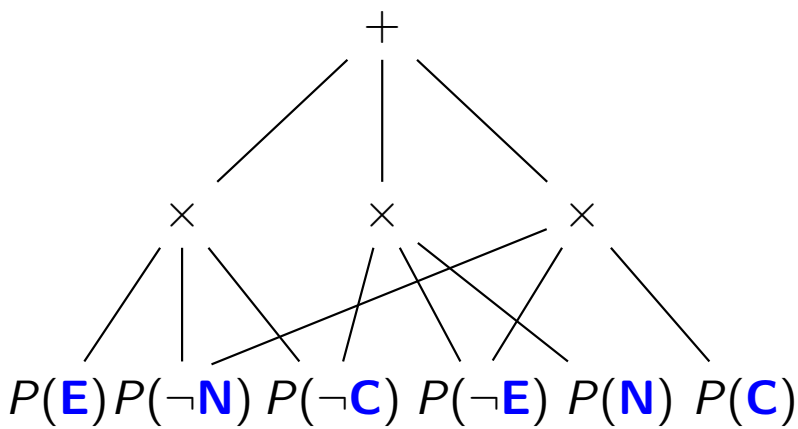
- **Uniqueness constraint** would be commonly written in *conjunctive normal form* (CNF):

$$\underbrace{(\mathbf{E} \vee \mathbf{N} \vee \mathbf{C})}_{\text{clause}} \wedge (\neg\mathbf{E} \vee \neg\mathbf{N}) \wedge (\neg\mathbf{N} \vee \neg\mathbf{C}) \wedge (\neg\mathbf{E} \vee \neg\mathbf{N})$$

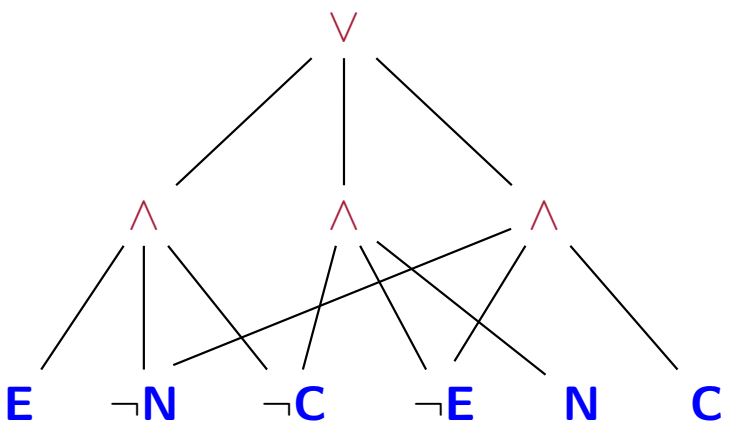
We could also write it in the following form:

$$(\mathbf{E} \wedge \neg\mathbf{N} \wedge \neg\mathbf{C}) \vee (\neg\mathbf{E} \wedge \mathbf{C} \wedge \neg\mathbf{N}) \vee (\neg\mathbf{E} \wedge \neg\mathbf{C} \wedge \mathbf{N})$$

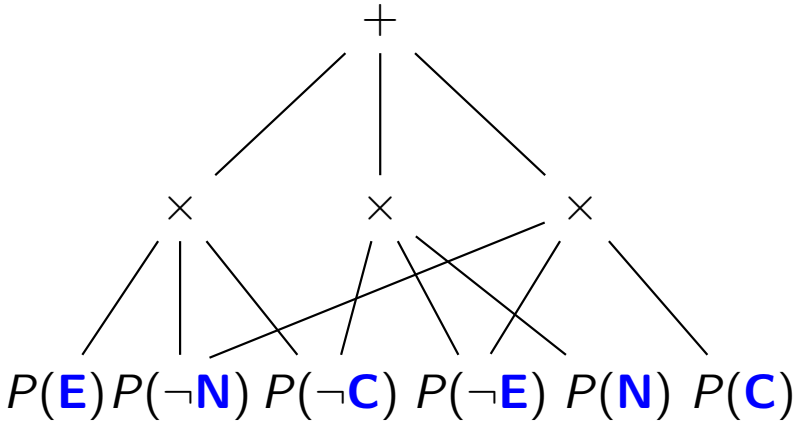




Creating tractable representations

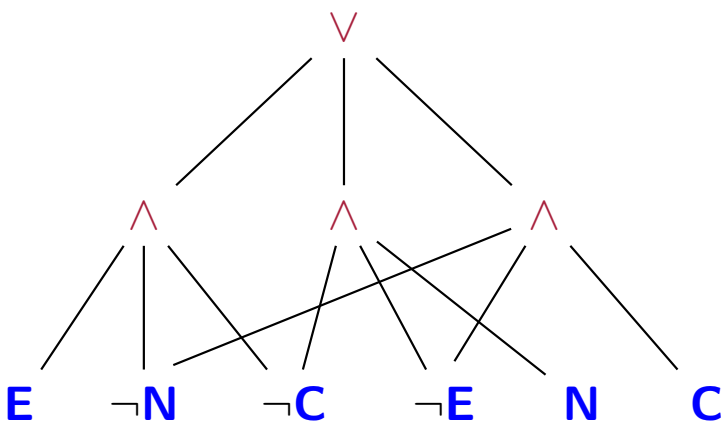


Boolean circuit

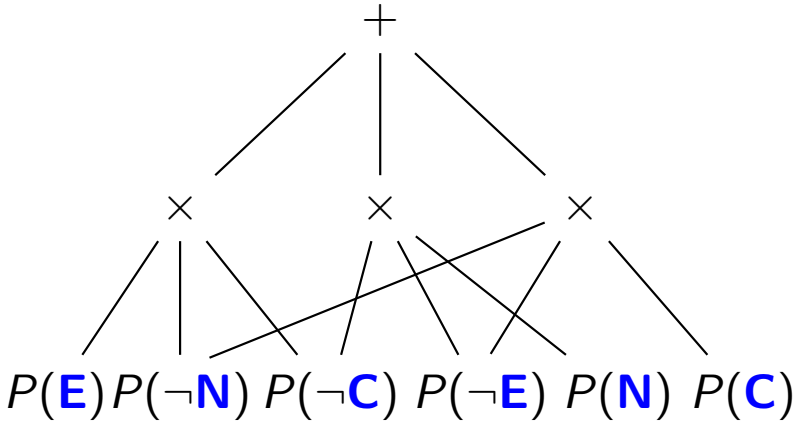


Arithmetic circuit

Creating tractable representations



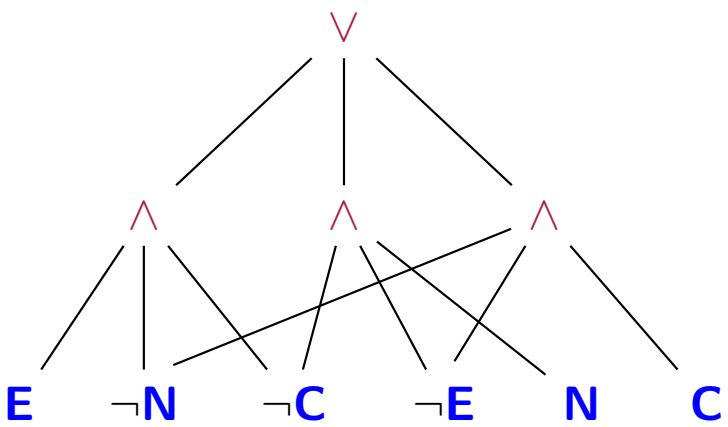
Boolean circuit



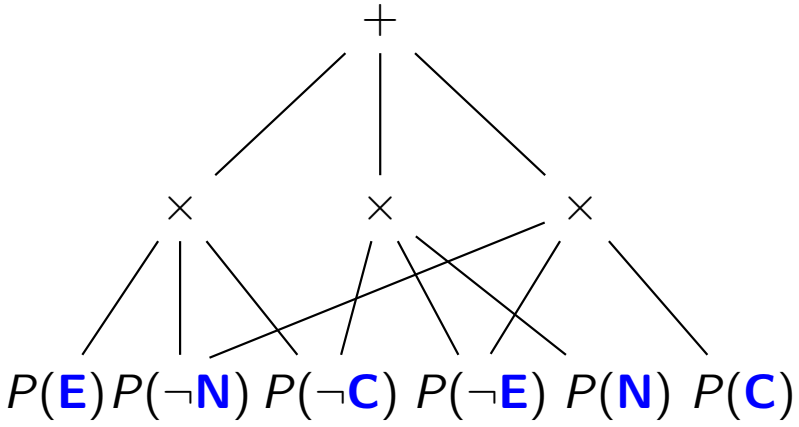
Arithmetic circuit

- Facilitate solving these algorithmic problems (SAT, WMC) efficiently.
  Two key properties:

Creating tractable representations



Boolean circuit



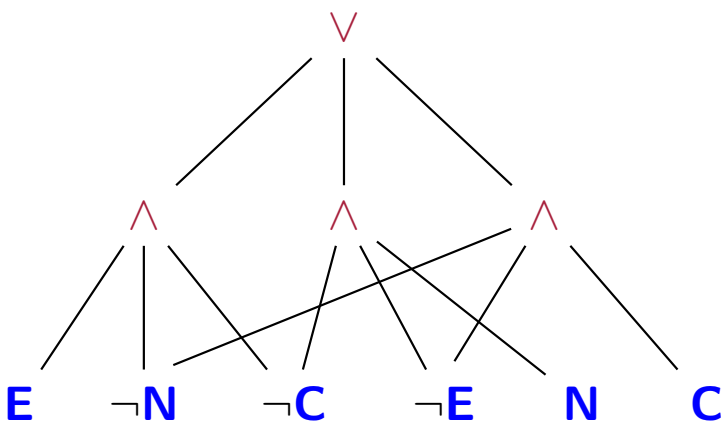
Arithmetic circuit

- Facilitate solving these algorithmic problems (SAT, WMC) efficiently.

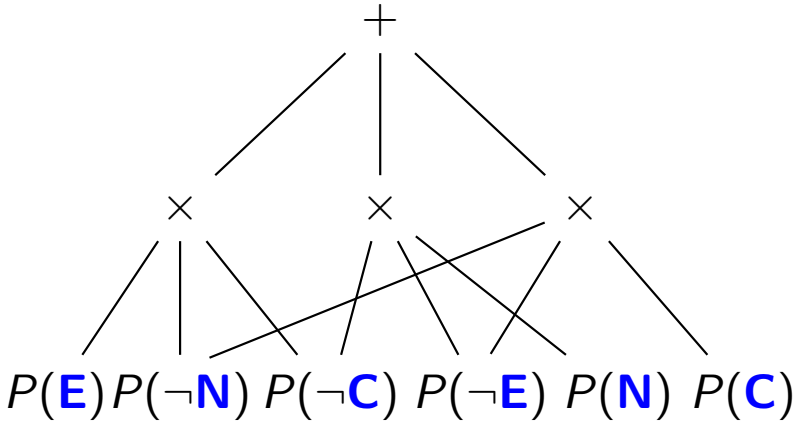
  Two key properties:

  decomposability: For conjunction nodes $\wedge$, children never share the same variables.

Creating tractable representations



Boolean circuit



Arithmetic circuit

- Facilitate solving these algorithmic problems (SAT, WMC) efficiently.

  Two key properties:

  decomposability: For conjunction nodes $\wedge$, children never share the same variables.

  determinism: For each disjunction node $\vee$, the children are logically inconsistent.

Compilation Software: example

```
from pysdd.sdd import SddManager

sdd = SddManager(var_count=3)
p1, p2, p3 = sdd.vars
uniqueness = (p1 & -p2 & -p3) | (-p1 & p2 & -p3) \
        | (-p1 & -p2 & p3)

count = uniqueness.wmc(log_mode=False)
print(f"model count: {count.propagate()}")
```

https://github.com/wannesm/PySDD, see Darwiche (2011)

- Can extract underlying circuits, mix with Pytorch.

Avoiding the bottleneck

An alternative formalism: fuzzy logic

- Allows for truth values to be **real-valued**, implements Boolean operators as real-valued functions (t-norms). For example:

An alternative formalism: fuzzy logic

- Allows for truth values to be **real-valued**, implements Boolean operators as real-valued functions (t-norms). For example:

Boolean logic	$\mathcal{R}$**-Product**
$a \wedge b$	$\mathbf{a} \cdot \mathbf{b}$
$\neg a$	$1 - \mathbf{a}$
$a \vee b$	$\mathbf{a} + \mathbf{b} - \mathbf{a} \cdot \mathbf{b}$
$a \rightarrow b$	$\min(1, \frac{\mathbf{b}}{\mathbf{a}})$

An alternative formalism: fuzzy logic

- Allows for truth values to be **real-valued**, implements Boolean operators as real-valued functions (t-norms). For example:

Boolean logic	$\mathcal{R}$**-Product**
$a \wedge b$	$\mathbf{a} \cdot \mathbf{b}$
$\neg a$	$1 - \mathbf{a}$
$a \vee b$	$\mathbf{a} + \mathbf{b} - \mathbf{a} \cdot \mathbf{b}$
$a \to b$	$\min(1, \frac{\mathbf{b}}{\mathbf{a}})$

For example:

$$\mathsf{K} := \mathbf{E}(p_1, h_1) \wedge \mathbf{C}_2(p_2, h_2).$$

An alternative formalism: fuzzy logic

- Allows for truth values to be **real-valued**, implements Boolean operators as real-valued functions (t-norms). For example:

Boolean logic	$\mathcal{R}$**-Product**
$a \wedge b$	$\mathbf{a} \cdot \mathbf{b}$
$\neg a$	$1 - \mathbf{a}$
$a \vee b$	$\mathbf{a} + \mathbf{b} - \mathbf{a} \cdot \mathbf{b}$
$a \to b$	$\min(1, \frac{\mathbf{b}}{\mathbf{a}})$

For example:

$$\mathsf{K} := \mathbf{E}(p_1, h_1) \wedge \mathbf{C}_2(p_2, h_2).$$

Can define the loss to be

$$-\log\left(P_\theta(\mathbf{E}(p_1, h_1)) \cdot P_\theta(\mathbf{C}_2(p_2, h_2)\right)$$

.

An alternative formalism: fuzzy logic

Name	Boolean Logic	Product	Gödel	Łukasiewicz
Negation	$\neg A$	$1-a$	$1-a$	$1-a$
T-norm	$A \wedge B$	ab	$\min(a,b)$	$\max(0, a+b-1)$
T-conorm	$A \vee B$	$a+b-ab$	$\max(a,b)$	$\min(1, a+b)$
Residuum	$A \rightarrow B$	$\min\left(1, \frac{b}{a}\right)$	$\begin{cases} 1, & \text{if b} \geq \text{a}, \\ b, & \text{else} \end{cases}$	$\min(1, 1-a+b)$

Table 1: Mapping discrete statements to differentiable functions using t-norms. Literals are upper-cased (e.g. A) while real-valued probabilities are lower-cased (e.g. a). Here, differentiable forms are from a mixture of R-fuzzy logic and S-fuzzy logic. In this paper, we focus on the product t-norm.

An alternative formalism: fuzzy logic

Name	Boolean Logic	Product	Gödel	Łukasiewicz
Negation	$\neg A$	$1-a$	$1-a$	$1-a$
T-norm	$A \wedge B$	ab	$\min(a,b)$	$\max(0, a+b-1)$
T-conorm	$A \vee B$	$a+b-ab$	$\max(a,b)$	$\min(1, a+b)$
Residuum	$A \rightarrow B$	$\min\left(1, \frac{b}{a}\right)$	$\begin{cases} 1, & \text{if b} \geq \text{a}, \\ b, & \text{else} \end{cases}$	$\min(1, 1-a+b)$

Table 1: Mapping discrete statements to differentiable functions using t-norms. Literals are upper-cased (e.g. A) while real-valued probabilities are lower-cased (e.g. a). Here, differentiable forms are from a mixture of R-fuzzy logic and S-fuzzy logic. In this paper, we focus on the product t-norm.

- Lots of work on the optimization properties of different fuzzy systems of logic (Grespan et al., 2021; van Krieken et al., 2022).

An alternative formalism: fuzzy logic

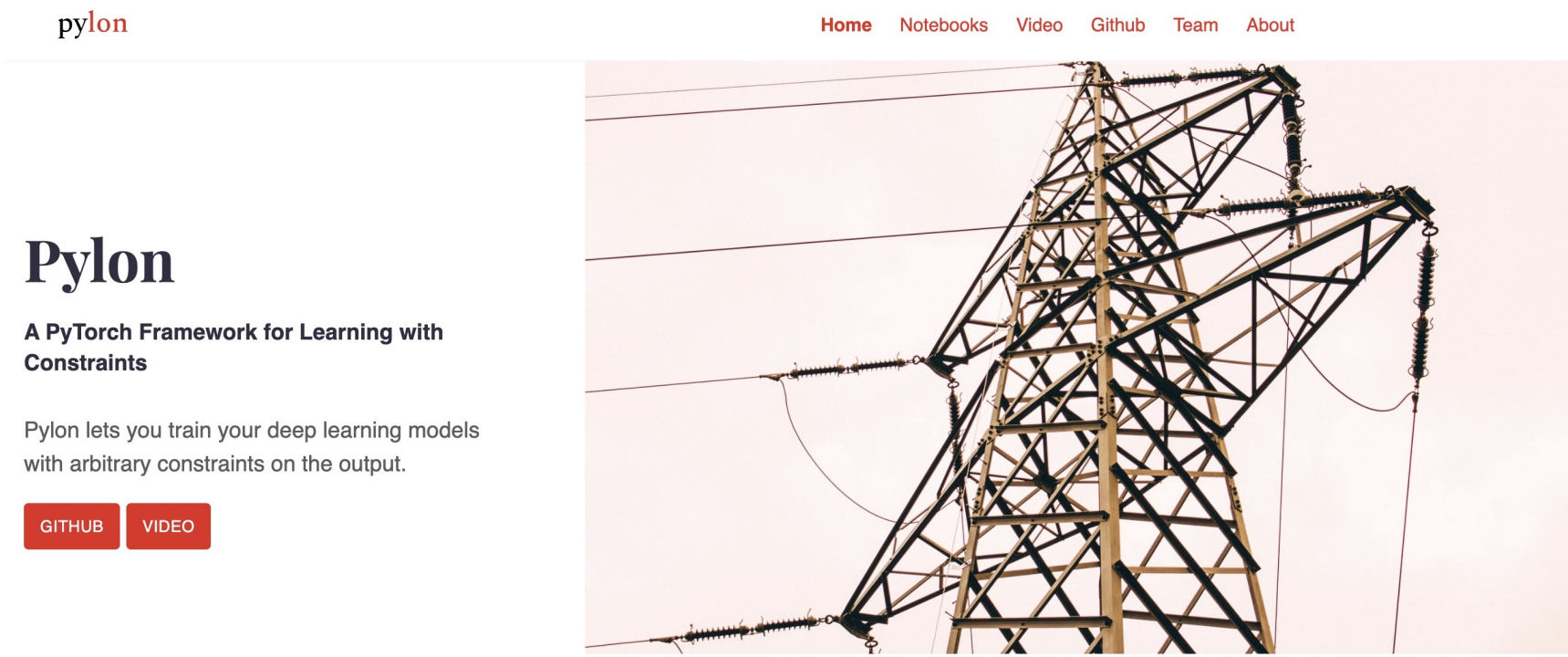
Name	Boolean Logic	Product	Gödel	Łukasiewicz
Negation	$\neg A$	$1-a$	$1-a$	$1-a$
T-norm	$A \wedge B$	ab	$\min(a,b)$	$\max(0, a+b-1)$
T-conorm	$A \vee B$	$a+b-ab$	$\max(a,b)$	$\min(1, a+b)$
Residuum	$A \rightarrow B$	$\min\left(1, \frac{b}{a}\right)$	$\begin{cases} 1, & \text{if b} \geq \text{a}, \\ b, & \text{else} \end{cases}$	$\min(1, 1-a+b)$

Table 1: Mapping discrete statements to differentiable functions using t-norms. Literals are upper-cased (e.g. A) while real-valued probabilities are lower-cased (e.g. a). Here, differentiable forms are from a mixture of R-fuzzy logic and S-fuzzy logic. In this paper, we focus on the product t-norm.

- Lots of work on the optimization properties of different fuzzy systems of logic (Grespan et al., 2021; van Krieken et al., 2022).

  trouble: Loses virtually all the nice semantic properties we discussed.

Pylon



https://pylon-lib.github.io/, see Ahmed et al. (2022a)

Conclusion

- Discussed the idea of modeling with declarative knowledge, probabilistic approach.

Conclusion

- Discussed the idea of modeling with declarative knowledge, probabilistic approach.
- **Semantic loss**: A loss function based on this semantics, based on probabilistic logic and WMC.

  Logic as a loss function, learning to satisfy, regularizer.

Conclusion

- Discussed the idea of modeling with declarative knowledge, probabilistic approach.
- **Semantic loss**: A loss function based on this semantics, based on probabilistic logic and WMC.

  Logic as a loss function, learning to satisfy, regularizer.
- **Next time**: A bit more about tractable representations of logic and probabilistic inference, language mode inference.

Thank you.

References I

Ahmed, K., Li, T., Ton, T., Guo, Q., Chang, K.-W., Kordjamshidi, P., Srikumar, V., Van den Broeck, G., and Singh, S. (2022a). Pylon: A pytorch framework for learning with constraints. In *NeurIPS 2021 Competitions and Demonstrations Track*, pages 319–324. PMLR.

Ahmed, K., Wang, E., Chang, K.-W., and Van den Broeck, G. (2022b). Neuro-symbolic entropy regularization. In *Uncertainty in Artificial Intelligence*, pages 43–53. PMLR.

Choi, A., Van den Broeck, G., and Darwiche, A. (2015). Tractable learning for structured probability spaces: A case study in learning preference distributions. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Darwiche, A. (2011). Sdd: A new canonical representation of propositional knowledge bases. In *Twenty-Second International Joint Conference on Artificial Intelligence*.

Darwiche, A. and Marquis, P. (2002). A knowledge compilation map. *Journal of Artificial Intelligence Research*, 17:229–264.

Giannini, F., Diligenti, M., Maggini, M., Gori, M., and Marra, G. (2023). T-norms driven loss functions for machine learning. *Applied Intelligence*, 53(15):18775–18789.

Grandvalet, Y. and Bengio, Y. (2004). Semi-supervised learning by entropy minimization. *Advances in neural information processing systems*, 17.

Grespan, M. M., Gupta, A., and Srikumar, V. (2021). Evaluating relaxations of logic for neural networks: A comprehensive study. *Proceedings of IJCAI*.

References II

Li, T., Gupta, V., Mehta, M., and Srikumar, V. (2019). A logic-driven framework for consistency of neural models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3924–3935, Hong Kong, China. Association for Computational Linguistics.

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

Marquis, P. (2008). Knowledge compilation: a sightseeing tour. *Tutorial notes, ECAI*, 8.

Marra, G., Dumančić, S., Manhaeve, R., and De Raedt, L. (2021). From statistical relational to neural symbolic artificial intelligence: a survey. *arXiv preprint arXiv:2108.11451*.

Minervini, P. and Riedel, S. (2018). Adversarially regularising neural nli models to integrate logical background knowledge. *arXiv preprint arXiv:1808.08609*.

van Krieken, E., Acar, E., and van Harmelen, F. (2022). Analyzing differentiable fuzzy logic operators. *Artificial Intelligence*, 302:103602.

van Krieken, E., Minervini, P., Ponti, E. M., and Vergari, A. (2024). On the independence assumption in neurosymbolic learning. *arXiv preprint arXiv:2404.08458*.

References III

Welleck, S., Kulikov, I., Roller, S., Dinan, E., Cho, K., and Weston, J. (2019). Neural text generation with unlikelihood training. In *International Conference on Learning Representations*.

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