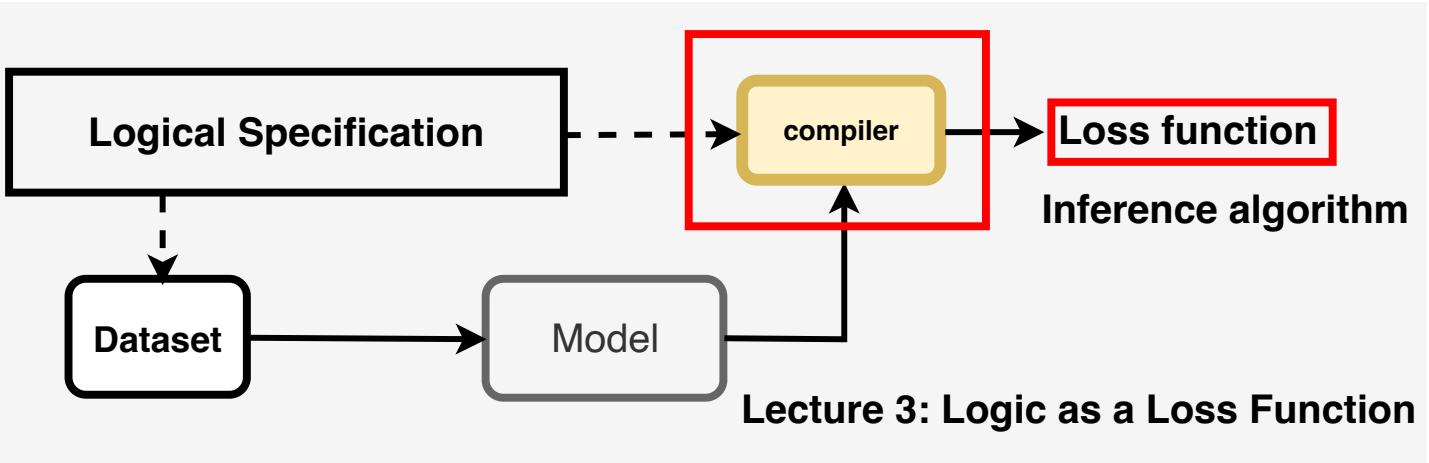
Formal Techniques for Neural-symbolic Modeling: Lecture 3

Kyle Richardson[a], Vivek Srikumar[b]

Allen Institute for Artificial Intelligence (AI2)[a],
University of Utah[b]

August 2023

Neural-symbolic modeling: big picture



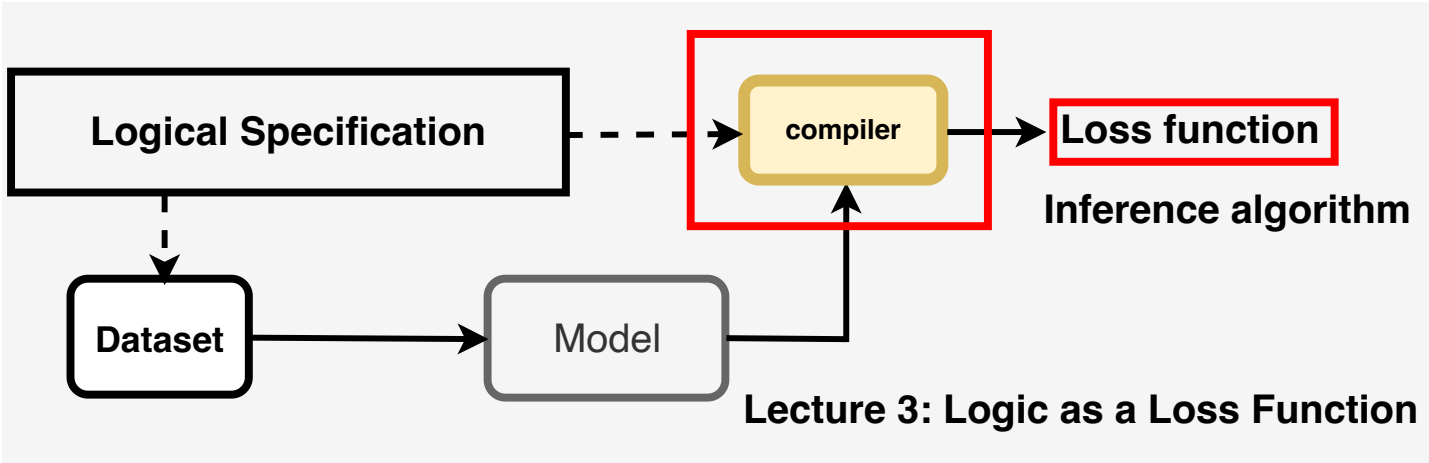
Problem we started with (lecture 1)

USER	Name me a country that is physically located between France and the Czech Republic.
ASSISTANT	Germany is a country that is physically located between France and the Czech Republic.

USER	Is Germany physically located between France and the Czech Republic?
ASSISTANT	No, Germany is not physically located between France and the Czech Republic. France is located to the west of Germany, while the Czech Republic is located to the east.

Question Should we be surprised by behavior like this? **No**. Can we fix it and how? **Yes**: via training with constraints.

Neural-symbolic modeling: lecture goals



1. Describe a probabilistic approach for **modeling with declarative knowledge**.
2. Introduce **semantic loss**, a type of loss function based on this semantics, *logic as loss function*.
3. Discuss weighted model counting (**WMC**), how to efficiently compute counts for training via tractable circuits.

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

```
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

```
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*

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)

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

What rules should an ideal NLI model follow?

- **More rules**

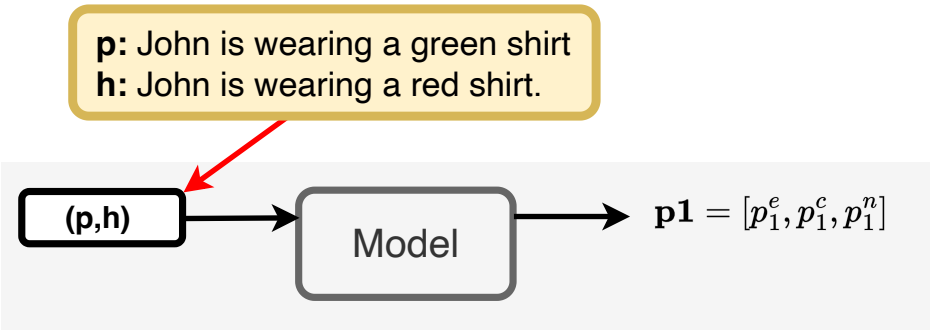
  Transitivity relations Given the triple (p, h, z)

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

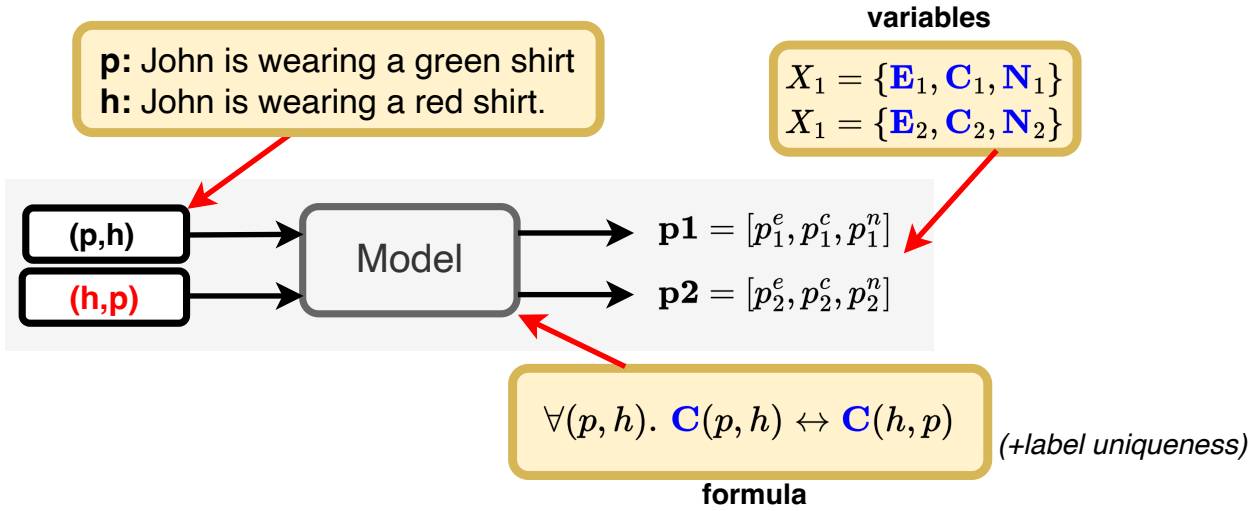
  Uniqueness constraints: there should be only one prediction.

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

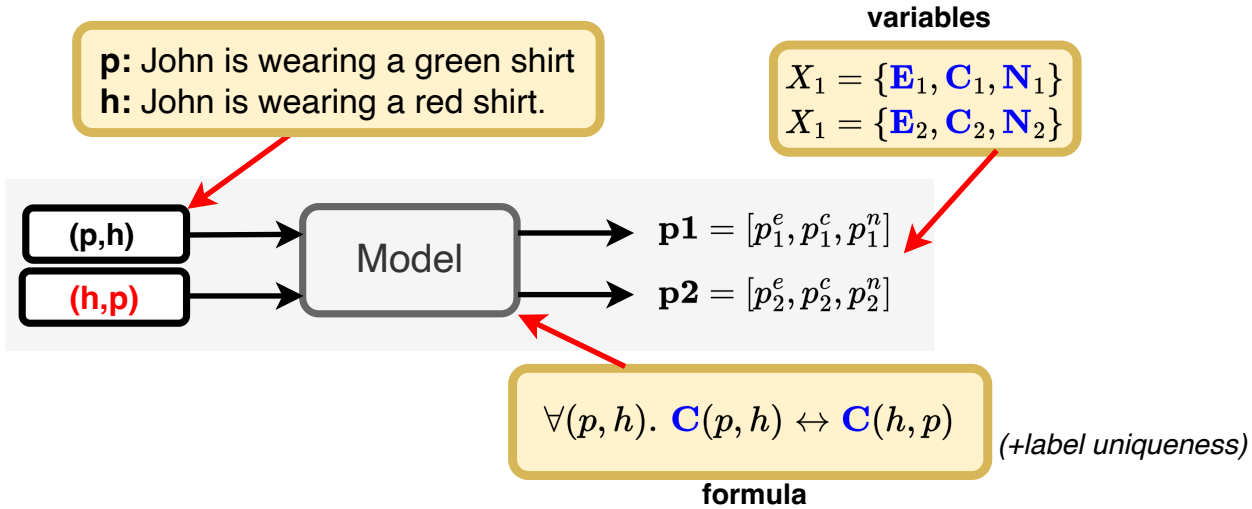
Modeling with declarative knowledge: Semantics



Modeling with declarative knowledge: Semantics



Modeling with declarative knowledge: Semantics



- **Semantics**: The set of models, or *worlds*, that satisfy our formula F.

	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

Modeling with declarative knowledge: Semantics

- Using $\mathbf{p}_1, \mathbf{p}_2$ (i.e., the model's predictive probabilities) we can assign scores. E.g., $\mathbf{p} = [\mathbf{p}1; \mathbf{p}2] = [\mathbf{p}_{p,h}; \mathbf{p}_{h,p}] = [0.3, 0.2, 0.5, 0.1, 0.8, 0.1]$

	Worlds	Satisfies F?	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

count: $\approx$0.05976

Modeling with declarative knowledge: Semantics

- Using $\mathbf{p}_1, \mathbf{p}_2$ (i.e., the model's predictive probabilities) we can assign scores. E.g., $\mathbf{p} = [\mathbf{p}1; \mathbf{p}2] = [\mathbf{p}_{p,h}; \mathbf{p}_{h,p}] = [0.3, 0.2, 0.5, 0.1, 0.8, 0.1]$

	Worlds	Satisfies F?	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

count: ≈ 0.05976

where the score S of each world is equal to

$$S(\mathbf{w}) = \prod_{\mathbf{w} \models \mathbf{P}_i} \mathbf{p}_i \cdot \prod_{\mathbf{w} \models \neg\mathbf{P}_i} (1 - \mathbf{p}_i)$$

Modeling with declarative knowledge: Semantics

- Using $\mathbf{p}_1, \mathbf{p}_2$ (i.e., the model's predictive probabilities) we can assign scores. E.g., $\mathbf{p} = [\mathbf{p}1; \mathbf{p}2] = [\mathbf{p}_{p,h}; \mathbf{p}_{h,p}] = [0.3, 0.2, 0.5, 0.1, 0.8, 0.1]$

	Worlds	Satisfies F?	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

count: ≈ 0.05976

where the score S of each world is equal to

$$S(\mathbf{w}) = \prod_{\mathbf{w} \models \mathbf{P}_i} \mathbf{p}_i \cdot \prod_{\mathbf{w} \models \neg\mathbf{P}_i} (1 - \mathbf{p}_i)$$

and the final count **weighted model count** for our formula F:

$$p(F) = \sum_{\mathbf{w} \models F} S(\mathbf{w}).$$

Modeling with declarative knowledge: Semantics

	Worlds	Satisfies F?	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

count: $\approx$0.05976

where the score of each world is equal to

$$S(\mathbf{w}) = \prod_{\mathbf{w} \models \mathbf{P}_i} \mathbf{p}_i \cdot \prod_{\mathbf{w} \models \neg\mathbf{P}_i} (1 - \mathbf{p}_i)$$

and the final count **weighted model count** for our formula F:

$$p(F) = \sum_{\mathbf{w} \models F} S(\mathbf{w}).$$

Goals: Train models with declarative knowledge under this kind of semantics; *make them more consistent and aligned to our expectations*.

Semantic loss

Semantic loss function

- Directly train models against symbolic constraints via the semantics from before (Xu et al., 2018; Ahmed et al., 2022b)

Semantic loss function

- Directly train models against symbolic constraints via the semantics from before (Xu et al., 2018; Ahmed et al., 2022b)

$$\mathcal{L}_{\text{sem}(F,\mathbf{p})} = -\log \underbrace{\sum_{\mathbf{w}\models F} \prod_{\mathbf{w}\models \mathbf{P}_i} \mathbf{p}_i \cdot \prod_{\mathbf{w}\models \neg \mathbf{P}_i} (1-\mathbf{p}_i)}_{\text{Weighted model counting parameterized by } \mathbf{p}}$$

$$= -\log \mathbb{E}_{\mathbf{w}\sim\mathbf{p}}[\mathbb{1}\{\mathbf{w} \models F\}]$$

Semantic loss function

- Directly train models against symbolic constraints via the semantics from before (Xu et al., 2018; Ahmed et al., 2022b)

$$\mathcal{L}_{\text{sem}(F,\mathbf{p})} = -\log \underbrace{\sum_{\mathbf{w} \models F} \prod_{\mathbf{w} \models \mathbf{P}_i} \mathbf{p}_i \cdot \prod_{\mathbf{w} \models \neg \mathbf{P}_i} (1 - \mathbf{p}_i)}_{\text{Weighted model counting parameterized by } \mathbf{p}}$$

$$= -\log \mathbb{E}_{\mathbf{w} \sim \mathbf{p}}[\mathbb{1}\{\mathbf{w} \models F\}]$$

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: special cases

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

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

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. $\mathbf{P}(p, h)$ to $\mathbf{P}$):

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

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. $\mathbf{P}(p, h)$ to $\mathbf{P}$):

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

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:

$$\mathcal{L}_{\text{uniq}(\mathbf{p})} = -\log \sum_{i=1}^{3} \mathbf{p}_i \prod_{j=1, j\neq i} (1 - \mathbf{p}_j)$$

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. $\mathbf{P}(p, h)$ to $\mathbf{P}$):

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

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:

$$\mathcal{L}_{\text{uniq}(\mathbf{p})} = -\log \sum_{i=1}^{3} \mathbf{p}_i \prod_{j=1, j\neq i} (1 - \mathbf{p}_j)$$

Note: Is feasible, requires $O(n^2)$ computation, can we improve?

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:

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

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:

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

Performing counting of this gives the following:

	Worlds	**Satisfies F?**
$\mathbf{w}_1$	$\mathbf{E}_1, \mathbf{C}_2$	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:

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

Performing counting of this gives the following:

	Worlds	**Satisfies F?**
$\mathbf{w}_1$	$\mathbf{E}_1, \mathbf{C}_2$	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**.

Declarative Specification of Loss Functions

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

Declarative Specification of Loss Functions

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

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

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

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

$$\forall (t, \mathbf{P}_1, \mathbf{P}_2) \in D \qquad \bigwedge (\mathbf{P}_1 \wedge \neg\mathbf{P}_2) \vee (\neg\mathbf{P}_1 \wedge \mathbf{P}_2) \qquad (\textit{semi-supervised learning})$$

Declarative Specification of Loss Functions

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

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

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

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

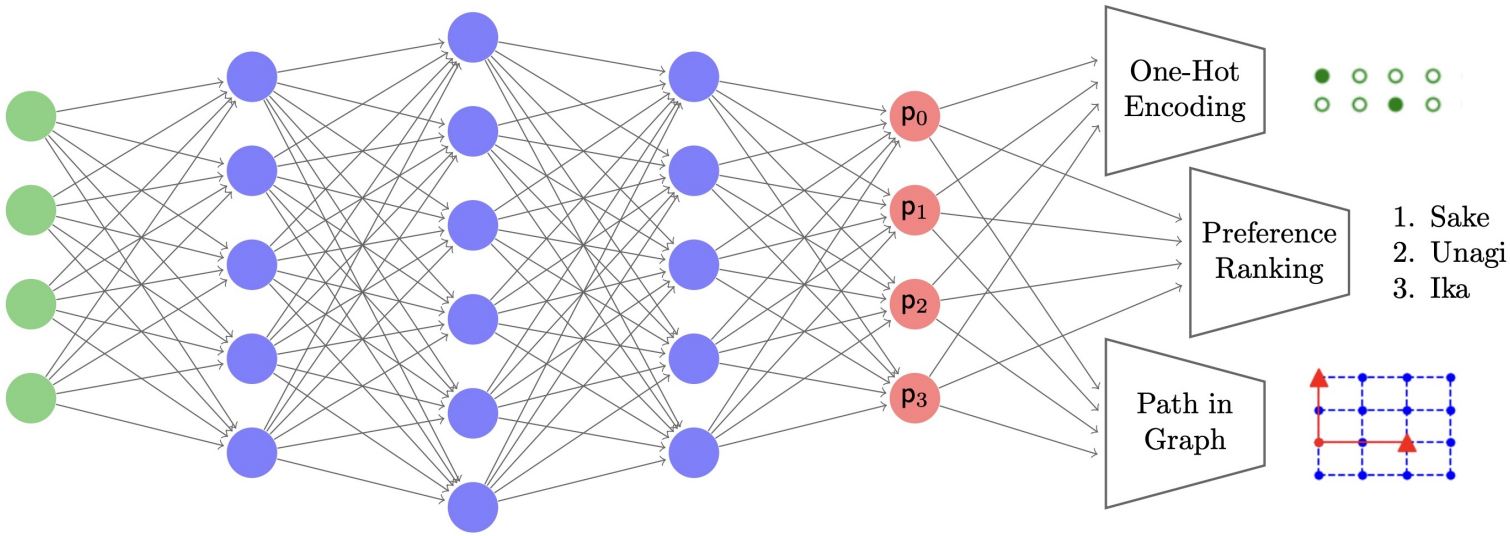
$$\forall(t, \mathbf{P}_1, \mathbf{P}_2) \in D \qquad \bigwedge (\mathbf{P}_1 \wedge \neg\mathbf{P}_2) \vee (\neg\mathbf{P}_1 \wedge \mathbf{P}_2) \qquad \textit{(semi-supervised learning)}$$

  Beyond ordinary ML

$$\forall(t, t', \mathbf{P}) \in D \qquad \bigwedge \mathbf{P}_1(t) \leftrightarrow \mathbf{P}_1(t') \qquad \textit{(symmetry/invariance)}$$

$$\forall(t_1, t_2, \mathbf{P}_1, \mathbf{P}_2) \in D \qquad \bigwedge \mathbf{P}_1(t_1) \rightarrow \mathbf{P}_2(t_2) \qquad \text{(learning from pairs)}$$

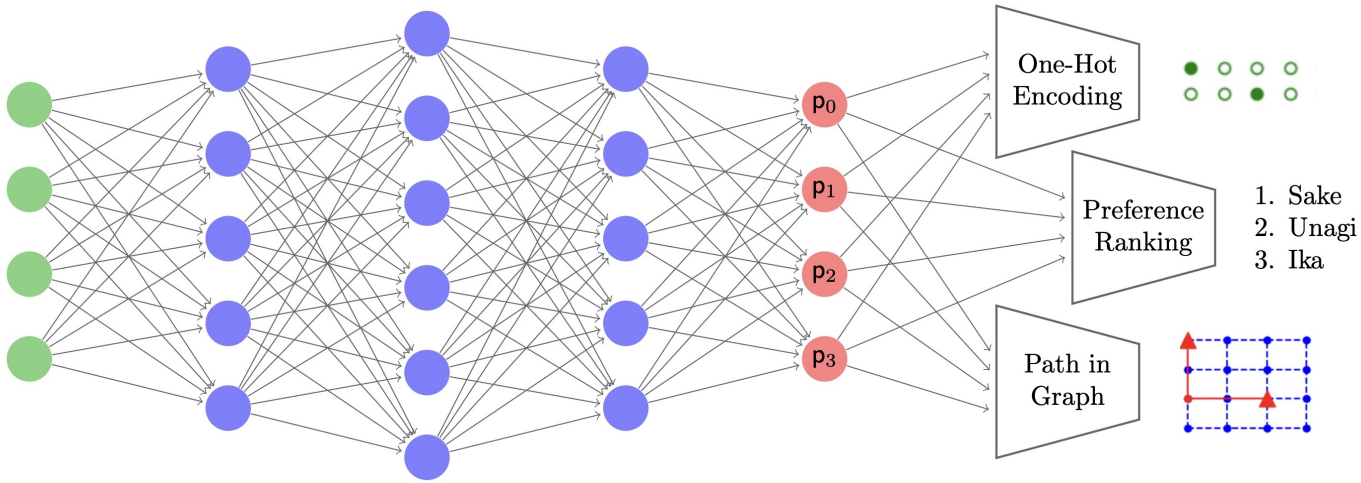
Other uses of semantic loss



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 $\mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3$, binary matrix $\mathbf{P}_{i,j}$ $(i, j \in \{1, ..., 3\})$ and a total order:

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

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

How the semantic loss is used

```
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)
loss = label_loss + wmc_loss
```

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*

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda \mathcal{L}_{\text{sem}(\cdot)}$$

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*

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda \mathcal{L}_{\text{sem}(\cdot)}$$

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"**)

How the semantic loss is used: pitfalls

- Why is it used in this way?

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda \mathcal{L}_{\text{sem}(\cdot)}$$

consider again $\forall(p, h).\, \mathbf{C}(p, h) \leftrightarrow \mathbf{C}(h, p)$ with worlds:

	Worlds	Satisfies F?	Correct?
$\mathbf{w}_1$	$\neg\mathbf{E}_1, \mathbf{C}_1, \neg\mathbf{N}_1, \neg\mathbf{E}_2, \mathbf{C}_2, \neg\mathbf{N}_2$	yes	yes
$\mathbf{w}_2$	$\neg\mathbf{E}_1, \mathbf{C}_1, \neg\mathbf{N}_1, \neg\mathbf{E}_2, \neg\mathbf{C}_2, \mathbf{N}_2$	no	no
$\mathbf{w}_3$	$\neg\mathbf{E}_1, \mathbf{C}_1, \neg\mathbf{N}_1, \mathbf{E}_2, \neg\mathbf{C}_2, \neg\mathbf{N}_2$	no	no
$\mathbf{w}_4$	$\neg\mathbf{E}_1, \neg\mathbf{C}_1, \mathbf{N}_1, \neg\mathbf{E}_2, \mathbf{C}_2, \neg\mathbf{N}_2$	no	no
$\mathbf{w}_5$	$\neg\mathbf{E}_1, \neg\mathbf{C}_1, \mathbf{N}_1, \neg\mathbf{E}_2, \neg\mathbf{C}_2, \mathbf{N}_2$	yes	no
$\mathbf{w}_6$	$\neg\mathbf{E}_1, \neg\mathbf{C}_1, \mathbf{N}_1, \mathbf{E}_2, \neg\mathbf{C}_2, \neg\mathbf{N}_2$	yes	no
$\mathbf{w}_7$	$\mathbf{E}_1, \neg\mathbf{C}_1, \neg\mathbf{N}_1, \neg\mathbf{E}_2, \mathbf{C}_2, \neg\mathbf{N}_2$	no	no
$\mathbf{w}_8$	$\mathbf{E}_1, \neg\mathbf{C}_1, \neg\mathbf{N}_1, \neg\mathbf{E}_2, \neg\mathbf{C}_2, \mathbf{N}_2$	yes	no
$\mathbf{w}_9$	$\mathbf{E}_1, \neg\mathbf{C}_1, \neg\mathbf{N}_1, \mathbf{E}_2, \neg\mathbf{C}_2, \neg\mathbf{N}_2$	yes	no
		count: 5	

How the semantic loss is used: pitfalls

- Why is it used in this way?

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda \mathcal{L}_{\text{sem}(\cdot)}$$

consider again $\forall(p, h).\, \mathbf{C}(p, h) \leftrightarrow \mathbf{C}(h, p)$ with worlds:

	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	

Some pitfalls: We might incentivize the model to never predict $\mathbf{C}$, or for rules $\mathbf{P}_1 \rightarrow \mathbf{P}_2$ always negate $\mathbf{P}_1$; tuning of λ is important.

How the semantic loss is used: pitfalls

- It can sometimes provide a weak, or even distracting, signal

$$F = \mathbf{P}_1 \rightarrow \mathbf{P}_2,\ \mathbf{p} = [0.5, 0.5]$$

	Worlds	Satisfies F?	Score
$\mathbf{w}_1$	$\mathbf{P}_1, \mathbf{P}_2$	**yes**	0.25
$\mathbf{w}_2$	$\mathbf{P}_1, \neg\mathbf{P}_2$	**no**	**0.25**
$\mathbf{w}_3$	$\neg\mathbf{P}_1, \mathbf{P}_2$	**yes**	**0.25**
$\mathbf{w}_4$	$\neg\mathbf{P}_1, \neg\mathbf{P}_2$	**yes**	**0.25**

WMC: 0.75

3/4 of the models get counted, usually high score in spite of having a high **semantic entropy** (Ahmed et al., 2022b):

How the semantic loss is used: pitfalls

- It can sometimes provide a weak, or even distracting, signal

$$F = \mathbf{P}_1 \rightarrow \mathbf{P}_2,\ \mathbf{p} = [0.5, 0.5]$$

	Worlds	Satisfies F?	Score
$\mathbf{w}_1$	$\mathbf{P}_1, \mathbf{P}_2$	**yes**	0.25
$\mathbf{w}_2$	$\mathbf{P}_1, \neg\mathbf{P}_2$	**no**	**0.25**
$\mathbf{w}_3$	$\neg\mathbf{P}_1, \mathbf{P}_2$	**yes**	**0.25**
$\mathbf{w}_4$	$\neg\mathbf{P}_1, \neg\mathbf{P}_2$	**yes**	**0.25**

WMC: 0.75

3/4 of the models get counted, usually high score in spite of having a high **semantic entropy** (Ahmed et al., 2022b):

$$\mathbf{H}(W \mid F) = -\sum_{\mathbf{w} \models F} p(\mathbf{w} \mid F) \log p(\mathbf{w} \mid F)$$

$$\text{with } p(\mathbf{w} \mid F) = \frac{p(F \mid \mathbf{w}) p(\mathbf{w})}{p(F)} = \frac{\mathbb{1}\{\mathbf{w} \models F\} p(\mathbf{w})}{p(F)}$$

which is a quantity we can add to our loss.

Semantic entropy

Given the **semantic entropy** (Ahmed et al., 2022b):

$$\mathbf{H}(W \mid F) = -\sum_{\mathbf{w} \models F} p(\mathbf{w} \mid F) \log p(\mathbf{w} \mid F)$$

$$\text{with } p(\mathbf{w} \mid F) = \frac{p(F \mid \mathbf{w}) p(\mathbf{w})}{p(F)} = \frac{\mathbb{1}\{\mathbf{w} \models F\} p(\mathbf{w})}{p(F)}$$

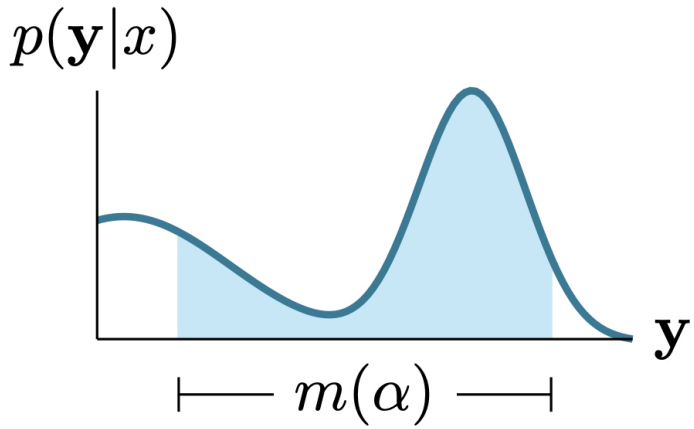
We can amend our loss to be:

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \lambda_1 \mathcal{L}_{\text{sem}(\cdot)} + \lambda_2 \mathbf{H}(\cdot)$$

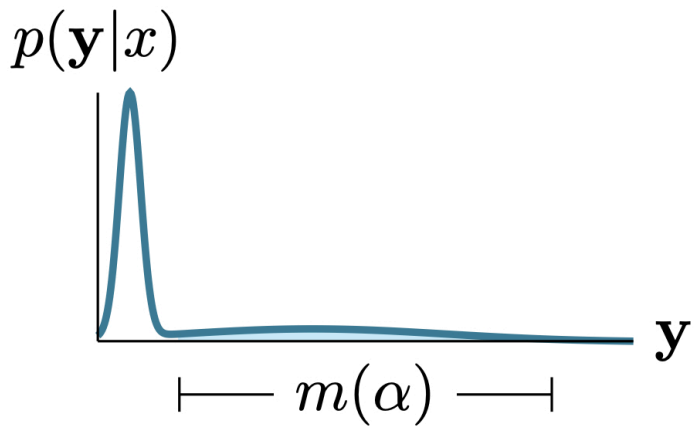
Also provides a means for measuring model uncertainity more broadly, beyond training.

Semantic entropy: some intuitions

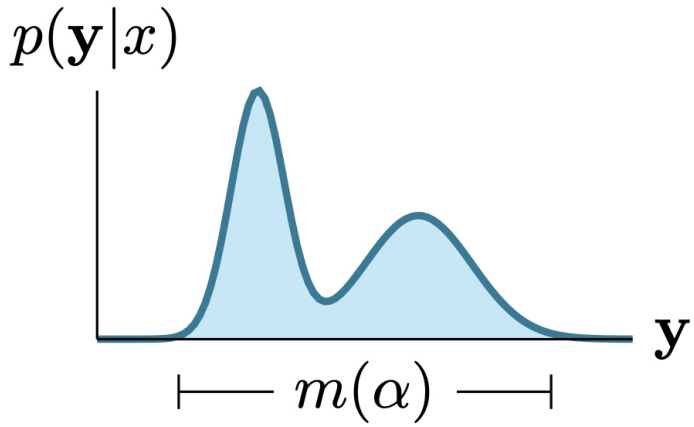
(Ahmed et al., 2022b)

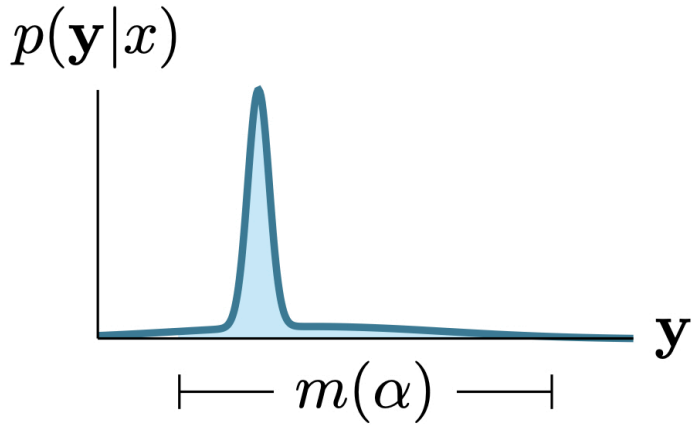


(a) Network uncertain over both valid and invalid predictions

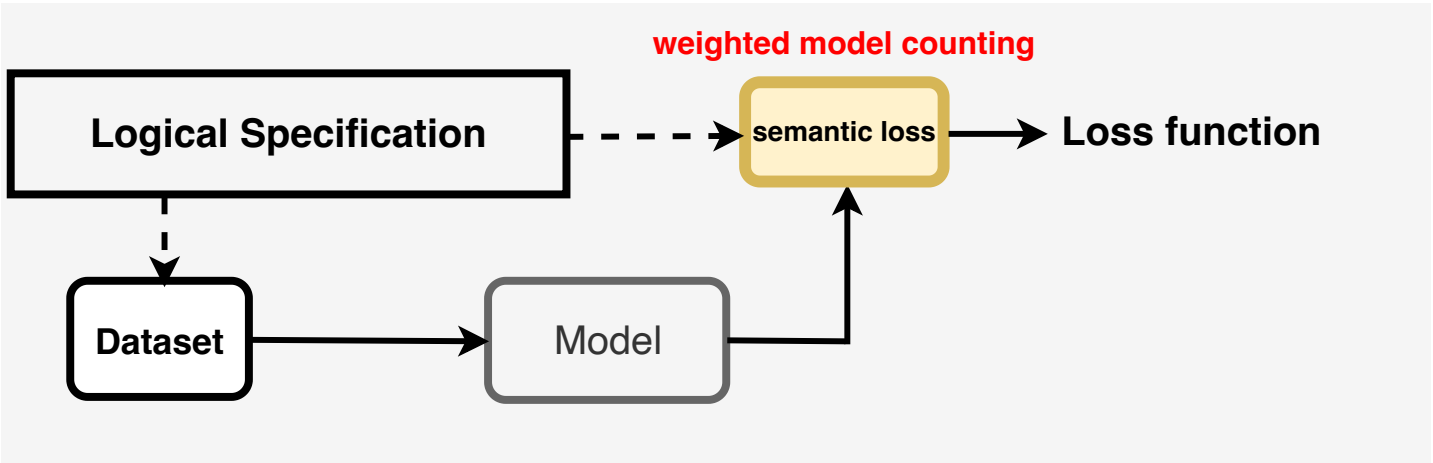


(b) Network allocating most mass to one invalid prediction





Interim summary



- Semantic loss, an approach to compiling logic into loss functions via weighted model counting, clean probabilistic semantics.

  Operationally: commonly added as a regularizer.

  As a framework, captures some classical loss function as special cases, *declaratively*; allows for much more.

Considering this case again

USER	Name me a country that is physically located between France and the Czech Republic.
ASSISTANT	Germany is a country that is physically located between France and the Czech Republic.

USER	Is Germany physically located between France and the Czech Republic?
ASSISTANT	No, Germany is not physically located between France and the Czech Republic. France is located to the west of Germany, while the Czech Republic is located to the east.

Considering this case again

USER	Name me a country that is physically located between France and the Czech Republic.
ASSISTANT	Germany is a country that is physically located between France and the Czech Republic.

USER	Is Germany physically located between France and the Czech Republic?
ASSISTANT	No, Germany is not physically located between France and the Czech Republic. France is located to the west of Germany, while the Czech Republic is located to the east.

We expect models to obey such invariances, but we don't train them to be invariant.

Model Counting

Model Counting

- **Model counting**: Counting the number of satisfying assignments for a formula, #P-complete (Valiant, 1979), weighted model counting (**WMC**)

  Problem: how do we compute our weighted model counts efficiently?

  Tractable Circuits *(from lecture 1)*

	Worlds	Satisfies F?	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

Model Counting: DPLL again

- Reminder about how to solve SAT (*lecture 1*), **splitting rule**.

input: A CNF formula Δ

```
1  Function DPLL(Δ):
2      if {} ∈ Δ then
3          return unsat                                   // empty clause
4      else if Δ = {} then
5          return sat                                  // No more clauses
6      select literal l from Δ                  // Branching rule ;
7      return (l ∧ DPLL(Δ|l)) ∨ (¬ l ∧ DPLL(Δ|¬l))     // Splitting Rule ;
```

Model Counting: DPLL again

- Reminder about how to solve SAT (*lecture 1*), **splitting rule**.

input: A CNF formula Δ

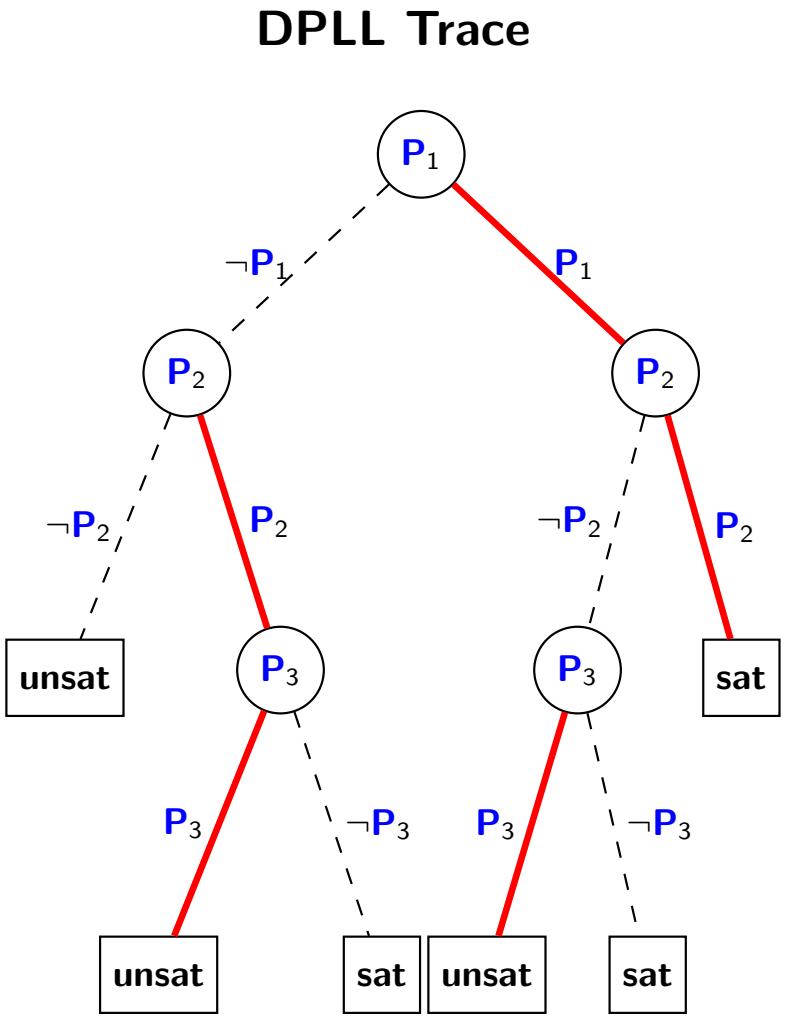
```
1  Function DPLL(Δ):
2      if {} ∈ Δ then
3          return unsat                                   // empty clause
4      else if Δ = {} then
5          return sat                                  // No more clauses
6      select literal l from Δ                  // Branching rule ;
7      return (l ∧ DPLL(Δ|l)) ∨ (¬ l ∧ DPLL(Δ|¬l))     // Splitting Rule ;
```

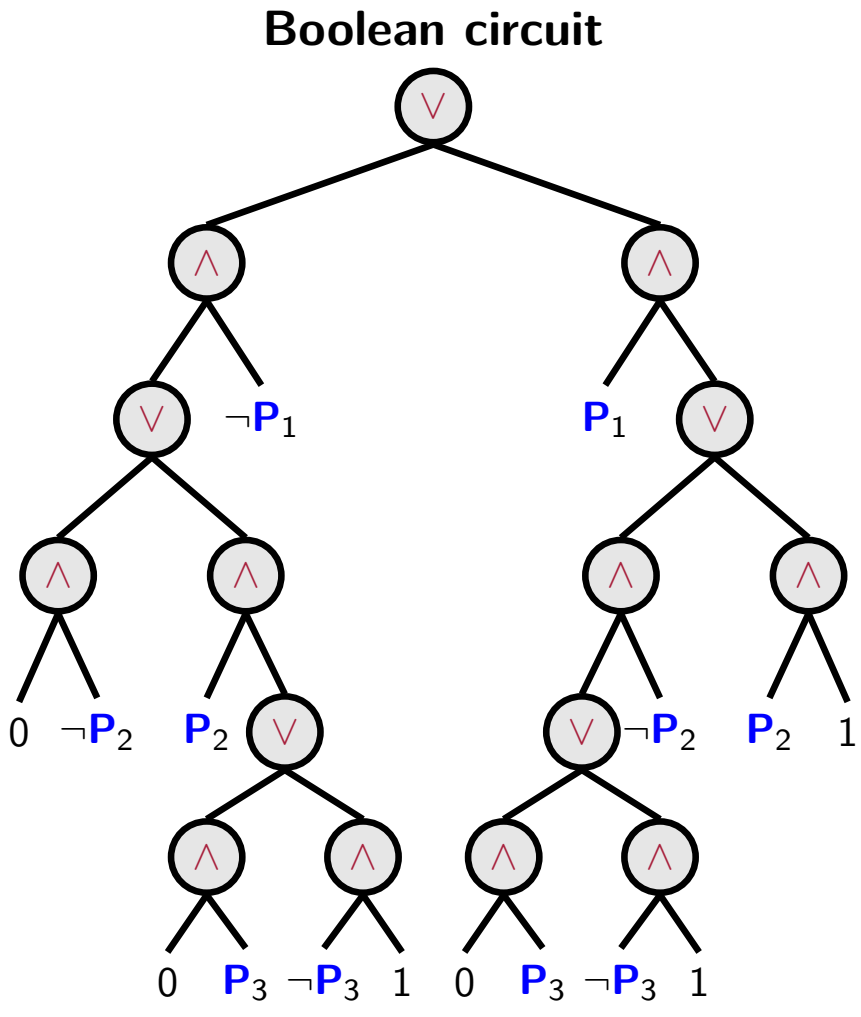
Variant for WMC (Birnbaum and Lozinskii, 1999), from Raedt et al. (2016)

input: A CNF formula Δ, literal weight function w, set of atoms A

```
1  Function WMC(Δ, A):
2      if {} ∈ Δ then
3          return 0
4      else if Δ = {} then
5          return ∏_{a∈A} w(a) + w(¬a)
6      select literal l from Δ ;
7      A ← A with l removed ;
8      return (w(l)× WMC(Δ|l, A)) + (w(¬l)× WMC(Δ|¬l, A)) ;
```

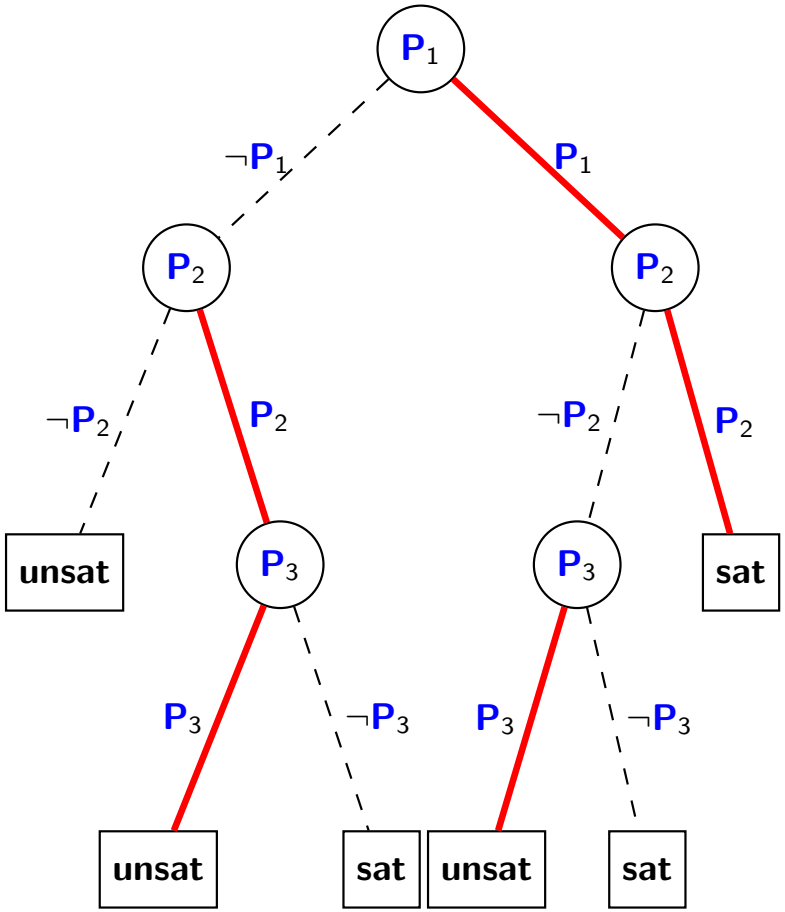
Visualizing DPLL again



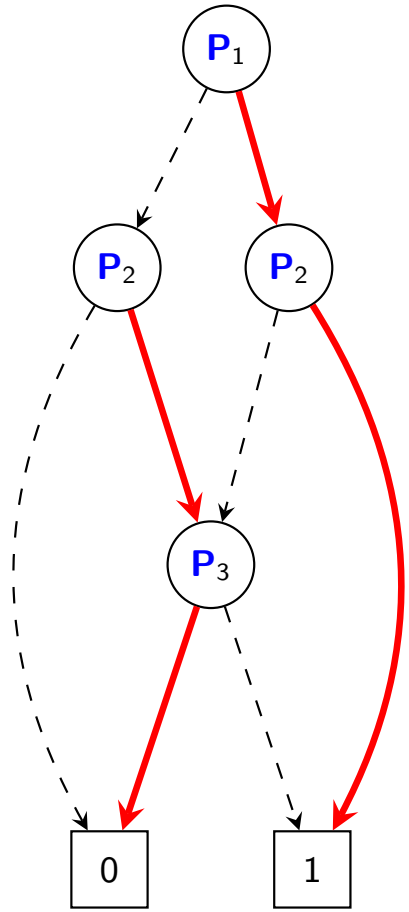


Visualizing DPLL Again

DPLL Trace



Binary Decision Diagram (BDD)

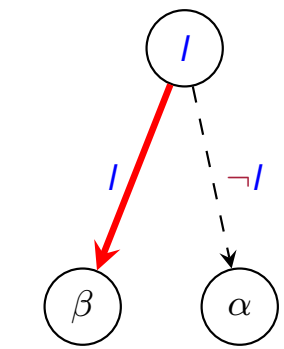


Model Counting: DPLL again

input: A CNF formula Δ, literal weight function w, set of atoms A

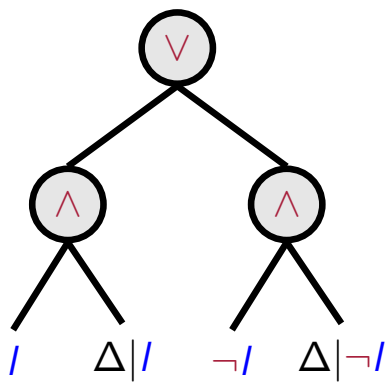
```
1  Function WMC(Δ, A):
2      if {} ∈ Δ then
3          return 0
4      else if Δ = {} then
5          return ∏_{a∈A} w(a) + w(¬a)
6      select literal l from Δ ;
7      A ← A with l removed ;
8      return (w(l)× WMC(Δ|l, A)) + (w(¬l)× WMC(Δ|¬l, A)) ;
```

BDD

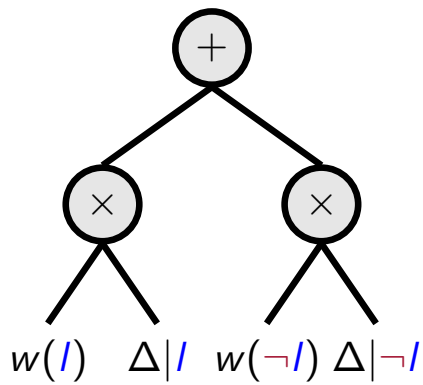


$(l \wedge \beta) \vee (\neg l \wedge \alpha)$

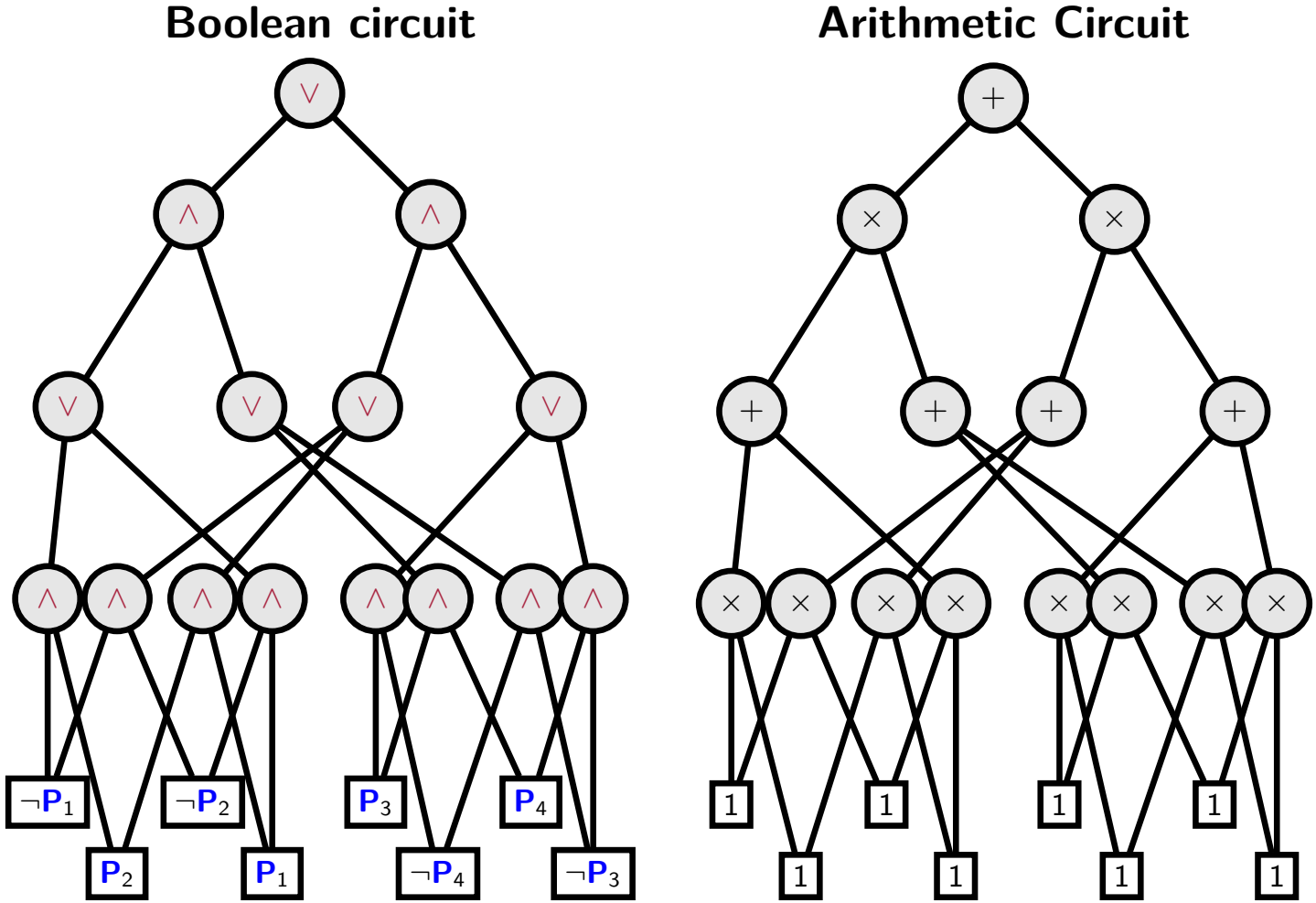
Boolean Circuit



Arithmetic Circuit

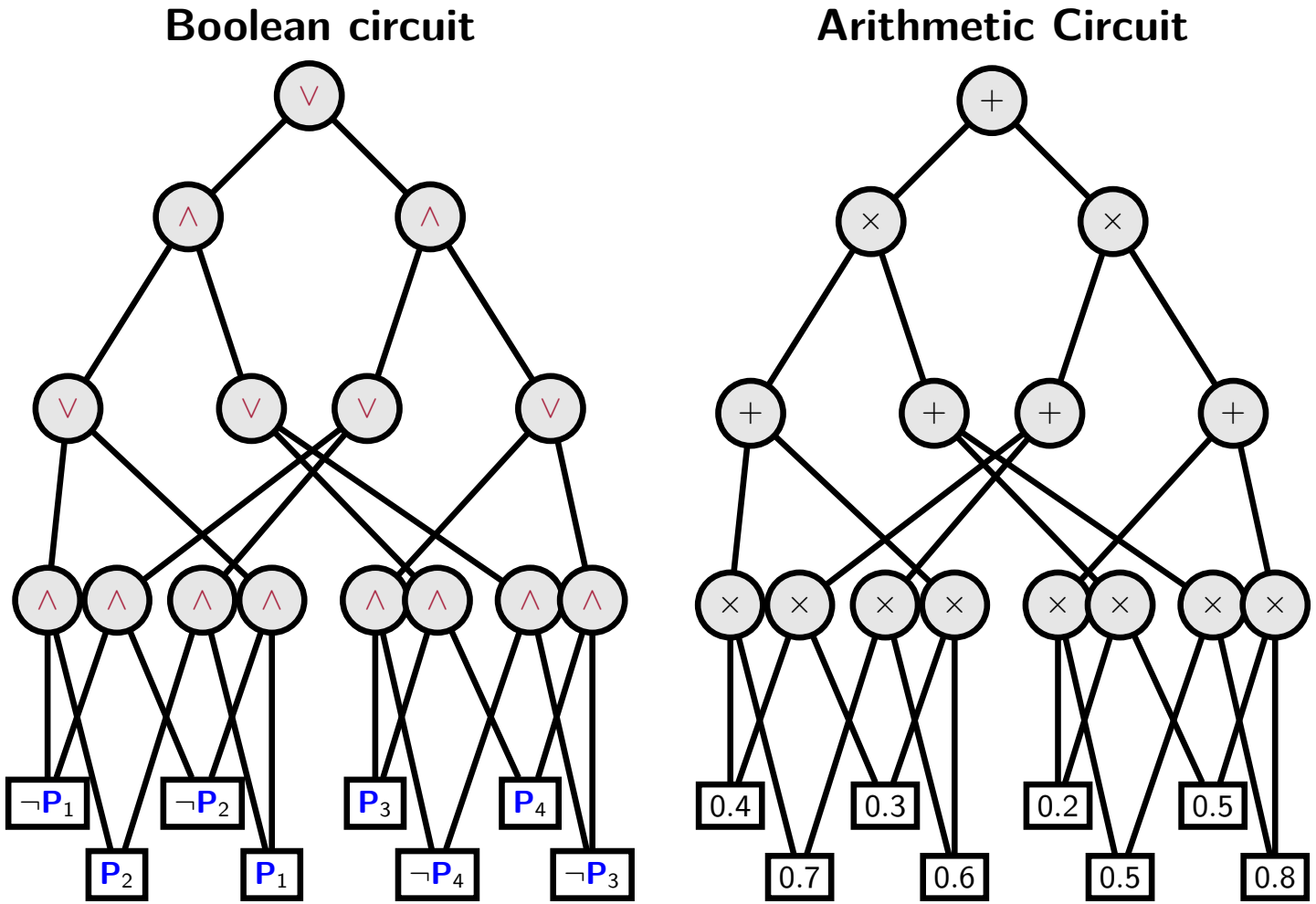


Tractable Circuits



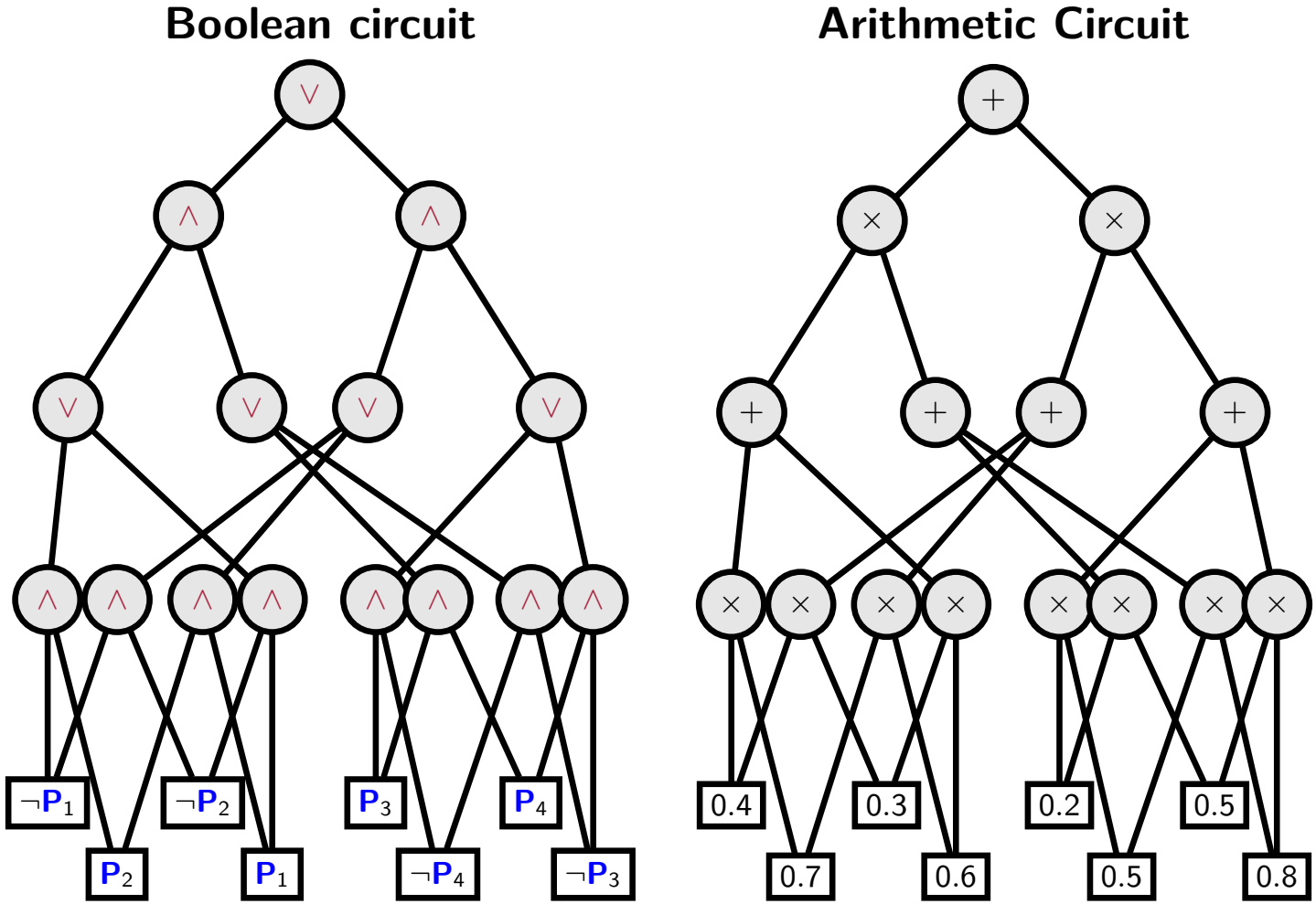
- **Decomposability** *(no shared variables in **and**-gates)*, **Determinism** (**or**-gates mutually exclusive) and **Smoothness** (**or**-gates share variables).

Tractable Circuits: Weighted Model Counting



- **Approach**: weight the literals nodes according to their weights, then evaluate; *linear time*.

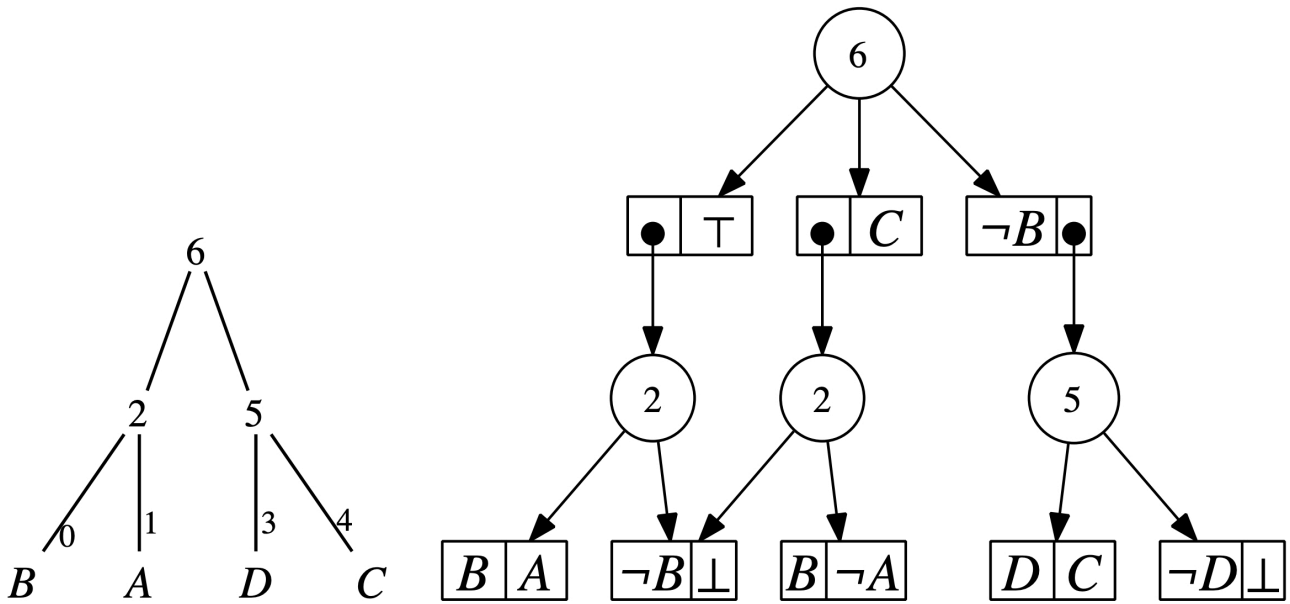
Tractable Circuits: Weighted Model Counting



- **Overall approach**: take target formula, compile into circuit (e.g., *top-down approach via SAT methods*, use to compute counts.

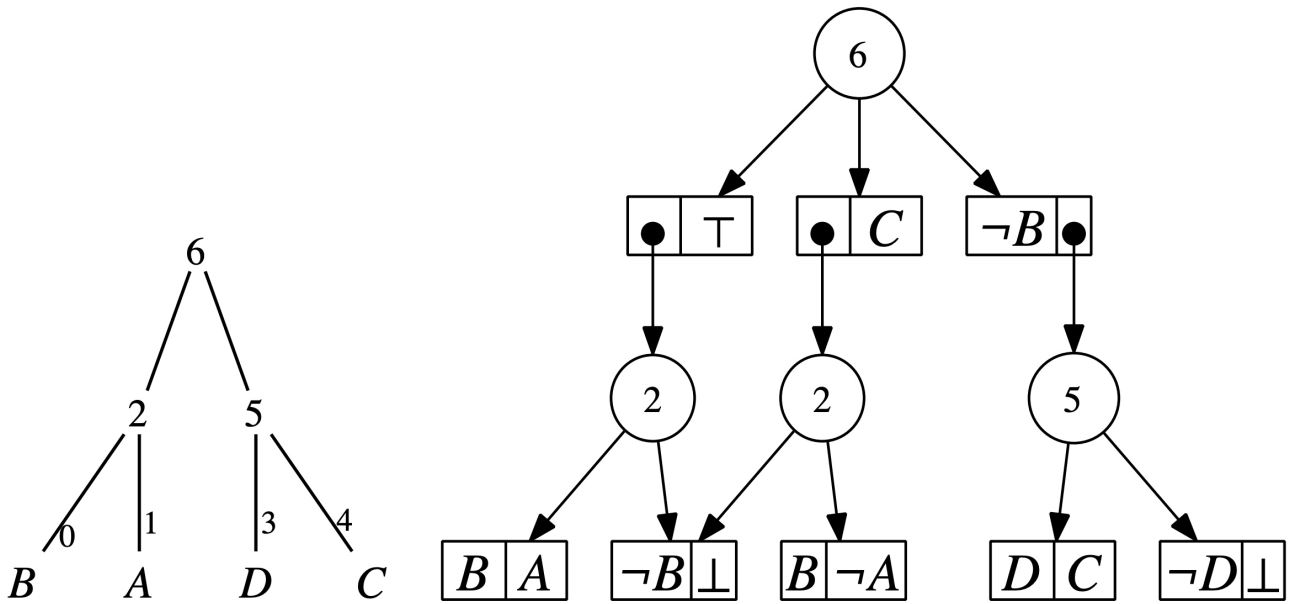
Example Sentential Decision Diagrams

$$F = (A \wedge B) \vee (B \wedge C) \vee (C \wedge D)$$



Example Sentential Decision Diagrams

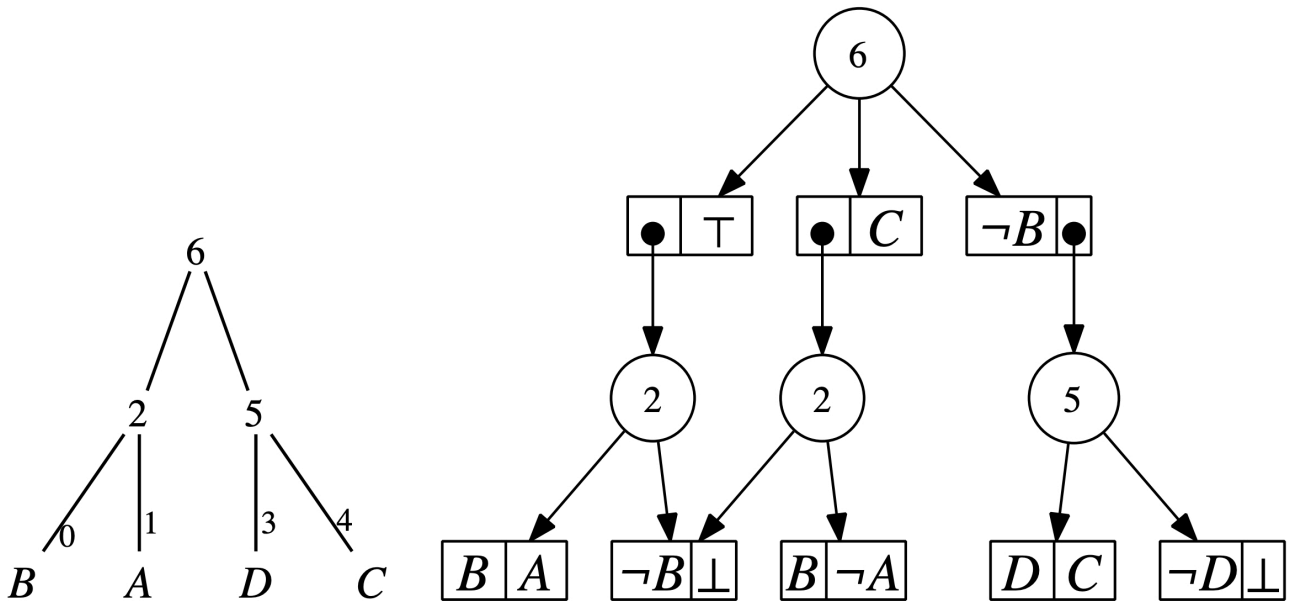
$$F = (A \wedge B) \vee (B \wedge C) \vee (C \wedge D)$$



- Notationally different, circles are **or**-nodes, boxes are **and**-nodes.

Example Sentential Decision Diagrams

$$F = (A \wedge B) \vee (B \wedge C) \vee (C \wedge D)$$



- Notationally different, circles are **or**-nodes, boxes are **and**-nodes.

  Structured Decomposability: Each **And**-gate has exactly two inputs i_1, i_2 that corresponds to nodes v in a binary tree (*vtree*).

  Partitioned Determinism: **Or**-gates of form $(p_1 \wedge s_1) \vee (p_2 \wedge s_2) \wedge \ldots$ *(prime/sub)*, precisely one prime can be true, exhaustive.

Sentential Decision Diagrams: bottom-up compilation

```
from pysdd.sdd import SddManager

sdd = SddManager(var_count=3)
p1, p2, p3 = sdd.vars
parity_1 = (p1 & -p2 & -p3) | (-p1 & p2 & -p3) \
         | (-p1 & -p2 & p3)
parity_3 =  (p1 & p2 & p3)
parity = sdd.disjoin(parity_1,parity_3)

count = parity.wmc(log_mode=False)
print(f"model count: {count.propagate()}")
## 4.0
```

- **Properties**: Can build boolean combinations of circuits in poly-time, canonical under fixed vtree, exponentially more succinct than OBDDs.

Integrating SDDs with Pytorch

```python
def compute_wmc(sdd,probs):
    """
    :type sdd: sdd representation of formula
    :type probs: pytorch probability tensor
    """
    if sdd.is_false():
        return 0.0
    elif sdd.is_true():
        return 1.0
    elif sdd.is_literal() and sdd.literal > 0:
        return lit_probs[0][sdd.literal-1][1]
    elif sdd.is_literal() and sdd.literal < 0:
        return lit_probs[0][-sdd.literal-1][0]
    elif sdd.is_decision():
        p = torch.tensor(0.0)
        for prime, sub in sdd.elements():
            p += self.prob(prime, lit_probs) * \
                 self.prob(sub, lit_probs)
        return p
```

Adapted from pylon-lib, see Ahmed et al. (2022a).

Conclusion

- Discussed the idea of modeling with declarative knowledge, probabilistic approach.
- **Semantic loss**: A loss function based on this semantics, based on weighted model counting (WMC).

  Logic as a loss function, learning to satisfy, regularizer.
- Efficient computation of model counts using tractable circuits, building on content from *lecture 1*.

Credits and more reading

- **NLP**: NLI examples based on Minervini and Riedel (2018); Li et al. (2019); semantic loss based on Xu et al. (2018)

  Other relevant work: Li and Srikumar (2019); Asai and Hajishirzi (2020); Rocktäschel et al. (2015)

- **Misc.**: Intro to SDDs *(Guy van den Broeck)*, see again Darwiche (2022).

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.

Asai, A. and Hajishirzi, H. (2020). Logic-guided data augmentation and regularization for consistent question answering. *arXiv preprint arXiv:2004.10157*.

Birnbaum, E. and Lozinskii, E. L. (1999). The good old davis-putnam procedure helps counting models. *Journal of Artificial Intelligence Research*, 10:457–477.

Choi, A., Van den Broeck, G., and Darwiche, A. (2015). Tractable learning for structured probability spaces: A case study in learning preference distributions. In *Proceedings of 24th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2861–2868.

Darwiche, A. (2022). Tractable boolean and arithmetic circuits. *arXiv preprint arXiv:2202.02942*.

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

Li, T., Gupta, V., Mehta, M., and Srikumar, V. (2019). A logic-driven framework for consistency of neural models. *arXiv preprint arXiv:1909.00126*.

References II

Li, T. and Srikumar, V. (2019). Augmenting neural networks with first-order logic. *Proceedings of ACL*.

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*.

Raedt, L. D., Kersting, K., Natarajan, S., and Poole, D. (2016). Statistical relational artificial intelligence: Logic, probability, and computation. *Synthesis lectures on artificial intelligence and machine learning*, 10(2):1–189.

Rocktäschel, T., Singh, S., and Riedel, S. (2015). Injecting logical background knowledge into embeddings for relation extraction. In *Proceedings of the 2015 conference of the north American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1119–1129.

Valiant, L. G. (1979). The complexity of computing the permanent. *Theoretical computer science*, 8(2):189–201.

Welleck, S., Kulikov, I., Roller, S., Dinan, E., Cho, K., and Weston, J. (2019). Neural text generation with unlikelihood training. *arXiv preprint arXiv:1908.04319*.

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.