

Formal Techniques for Neural-symbolic Modeling: Lecture 4

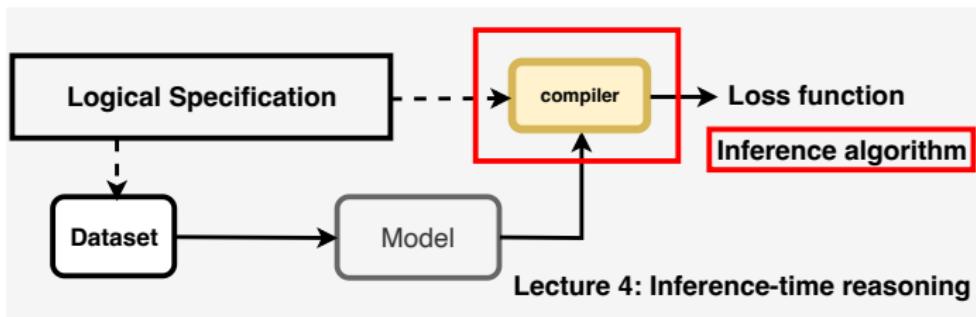
Kyle Richardson^a, Vivek Srikumar^b

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

August 2023



Inference time reasoning



This problem 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.

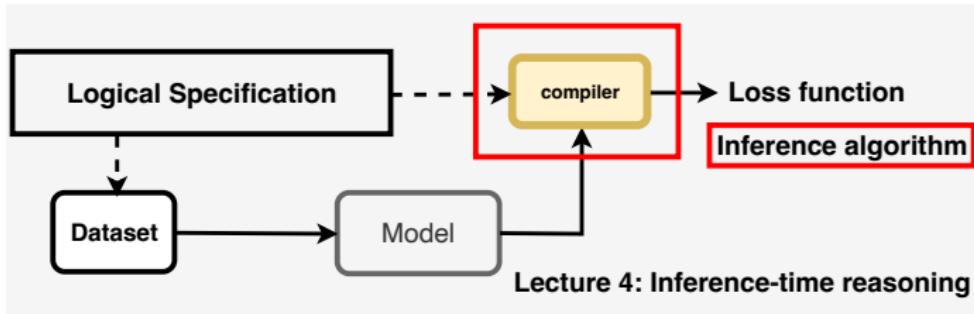
USER Is Germany physically located between France and the Czech Republic?

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.

Last lecture: Fixing models via *training* with constraints (*logic as loss*).

Today: Fixing via inference-time logical and probabilistic reasoning.

Inference time reasoning: lecture goals



1. Understanding how to use SAT-based methods (**MaxSAT**) for solving a classic inference problem (**MPE**).
2. How to use tractable circuits for inference, **maximizer circuits**.
3. Discussing differences between **directed** and **undirected** approaches to relational modeling, alternatives to our approach (*so far*)

Recap and more details

Task-specific model from last time

```
1 import torch
2 from transformers import \
3     AutoModelForSequenceClassification as hf_model
4
5 class NLIModel(torch.nn.Module):
6     def __init__(self, model_name):
7         super().__init__()
8         self.model = hf_model.from_pretrained(
9             model_name, ## base transformer
10            num_labels=3 ## 3 labels for NLI
11        )
12     def forward(self, features): ### forward pass
13         output = self.model(**features)
14         prob = output.logits.softmax(dim=-1)
15         return (output.loss, prob)
```

- ▶ **Fine-tuning** with task-specific parameters; *nowadays*: much focus on zero/few-shot modeling.

Building LLMs

```
1 import openai
2
3 IN_CONTEXT=f"""
4 # This task requires you to negate sentences
5 sentence: I am swimming but sailing.
6 negation: I am not sailing but swimming instead.
7 sentence: {sentence}
8 """
9 new_problem = "Dan went to pick apples"
10 prompt = IN_CONTEXT.format(sentence=new_problem)
11 ##### call to GPT3
12 response = openai.Completion.create(
13     model="text-davinci-003", prompt=prompt)
```

- ▶ The recipe for building today's large language models (**LLMs**):

Building LLMs

```
1 import openai
2
3 IN_CONTEXT=f"""
4 # This task requires you to negate sentences
5 sentence: I am swimming but sailing.
6 negation: I am not sailing but swimming instead.
7 sentence: {sentence}
8 """
9 new_problem = "Dan went to pick apples"
10 prompt = IN_CONTEXT.format(sentence=new_problem)
11 ##### call to GPT3
12 response = openai.Completion.create(
13     model="text-davinci-003", prompt=prompt)
```

- ▶ The recipe for building today's large language models (**LLMs**):
 1. **Train a base LM** on large volumes of text, often *decoder-only* models via generative pre-training (e.g., GPT3, LLama).

Building LLMs

```
1 import openai
2
3 IN_CONTEXT=f"""
4 # This task requires you to negate sentences
5 sentence: I am swimming but sailing.
6 negation: I am not sailing but swimming instead.
7 sentence: {sentence}
8 """
9 new_problem = "Dan went to pick apples"
10 prompt = IN_CONTEXT.format(sentence=new_problem)
11 ##### call to GPT3
12 response = openai.Completion.create(
13     model="text-davinci-003", prompt=prompt)
```

- ▶ The recipe for building today's large language models (**LLMs**):
 1. **Train a base LM** on large volumes of text, often *decoder-only* models via generative pre-training (e.g., GPT3, LLama).
 2. **Fine-tune on instructions** or other task-specific data; *alignment*.

Instructions and Fine-tuning

```
1 from datasets import load_dataset
2 dataset = load_dataset(
3     "mrm8488/unnatural-instructions", "core"
4 )
5
6 print(dataset["train"]["instances"][0])
7 #{"instruction_with_input": ["""You will be given a
8 # series of words. Output these words in reverse
9 # order, with each word on its own line.
10 #\nWords: ["Hello", "#world"]."""],
11 # "input": ["Words: ['Hello', 'world']."], 
12 # "output": ['world\nHello'],
13 # "constraints": ['None.']}
```

see Honovich et al. (2022)

- ▶ **Goal:** Generalize from a fixed set of declarative instructions, wider range of prompts, tasks.

Instructions and Fine-tuning

```
1 from transformers import T5ForConditionalGeneration
2 from peft import get_peft_model, LoraConfig, TaskType
3
4 model = T5ForConditionalGeneration.from_pretrained(
5     "t5-large"
6 )
7 peft_config = LoraConfig(
8     task_type=TaskType.SEQ_2_SEQ_LM,
9     r=8, lora_alpha=32,
10 )
11 model = get_peft_model(model, peft_config)
12 model.print_trainable_parameters()
13 # trainable params: 2359296 ||
14 # all params: 740027392 ||
15 # trainable%: 0.31881198257050464
```

see [peft library](#)

- ▶ **Parameter efficient fine-tuning:** Fine-tuning without tuning *all* of a model's parameters, rather a smaller set of *extra* parameters.

A motivating example

Another mistake involving geography

USER

Which country is physically bigger, Germany or Poland? Please answer one word.

ASSISTANT

Poland

Another mistake involving geography

USER Which country is physically bigger, Germany or Poland? Please answer one word.

ASSISTANT Poland

USER Is Poland physically bigger than Germany? Please answer only yes/no

ASSISTANT No.

Querying ChatGPT

```
1 COUNTRIES = ["Germany", "France", "Poland"]
2 PROMPT_1="""  
3 Which country is physically bigger {c1} or {c2}?
4 Please provide a one word answer.
5 """
6 PROMPT_2="""  
7 Is {c2} physically bigger than {c1}, answer
8 only yes/no.
9 """
10 for c1 in COUNTRIES:
11     for c2 in COUNTRIES:
12         if c1 == c2: continue
13         prompt_1 = PROMPT_1.format(c1=c1,c2=c2)
14         answer1 = query_chatgpt(prompt_1)
15         prompt_1 = PROMPT_2.format(c1=c1,c2=c2)
16         answer2 = query_chatgpt(prompt_1)
```

Querying ChatGPT

```
1 COUNTRIES = ["Germany", "France", "Poland"]
2 PROMPT_1="""  
3 Which country is physically bigger {c1} or {c2}?
4 Please provide a one word answer.
5 """
6 PROMPT_2="""  
7 Is {c2} physically bigger than {c1},answer only yes/no.
8 """
```

Querying ChatGPT

```
1 COUNTRIES = ["Germany", "France", "Poland"]
2 PROMPT_1="""  
3 Which country is physically bigger {c1} or {c2}?
4 Please provide a one word answer.
5 """
6 PROMPT_2="""  
7 Is {c2} physically bigger than {c1},answer only yes/no.
8 """
```

► What it returns

(c1,c2)	bigger, c1 or c2?	c2 bigger c1?	invariance?
(DE,FR)	FR	No	no
(DE,PL)	PL	No	no
(FR,DE)	DE	No	no
(FR,PL)	PL	No	no
(PL,DE)	DE	No	no
(PL,FR)	FR	No	no

More advanced prompting

- ▶ **Caveats** (*as always with prompting*): very sensitive to prompt, order, not best model (*use GPT4*) or prompting strategy, **be skeptical**.

More advanced prompting

- ▶ **Caveats (as always with prompting):** very sensitive to prompt, order, not best model (*use GPT4*) or prompting strategy, **be skeptical**.

USER	Which country is physically bigger, Germany or Poland? Please explain step by step.
<hr/>	
ASSISTANT	To determine which country is physically bigger, we can compare their respective land areas. <ol style="list-style-type: none">1. Germany: The total land area of Germany is approximately 357,022 square kilometers.2. Poland: The total land area of Poland is approximately 312,696 square kilometers. <p>By comparing these numbers, we can see that Germany has a larger land area than Poland. Specifically, Germany is around 44,326 square kilometers bigger than Poland. Therefore, Germany is physically bigger than Poland.</p>

Thinking about semantics again

(c1,c2)	bigger, c1 or c2?	c2 bigger c1?	<i>invariance?</i>
(DE,FR)	FR	No	no
(DE,PL)	PL	No	no
(FR,DE)	DE	No	no
(FR,PL)	PL	No	no
(PL,DE)	DE	No	no
(PL,FR)	FR	No	no

Thinking about semantics again

(c1,c2)	bigger, c1 or c2?	c2 bigger c1?	invariance?
(DE,FR)	FR	No	no
(DE,PL)	PL	No	no
(FR,DE)	DE	No	no
(FR,PL)	PL	No	no
(PL,DE)	DE	No	no
(PL,FR)	FR	No	no

- ▶ Invariance constraints.

$$\mathbf{C1}(c1, c2), \mathbf{C2}(c1, c2), \mathbf{Y}(c2, c1), \mathbf{N}(c2, c1)$$

first: $c1/c2$ is bigger; second: $c2$ is bigger than $c1$, yes/no.

Thinking about semantics again

(c1,c2)	bigger, c1 or c2?	c2 bigger c1?	invariance?
(DE,FR)	FR	No	no
(DE,PL)	PL	No	no
(FR,DE)	DE	No	no
(FR,PL)	PL	No	no
(PL,DE)	DE	No	no
(PL,FR)	FR	No	no

► Invariance constraints.

$$\mathbf{C1}(c1, c2), \mathbf{C2}(c1, c2), \mathbf{Y}(c2, c1), \mathbf{N}(c2, c1)$$

first: $c1/c2$ is bigger; second: $c2$ is bigger than $c1$, yes/no.

$$\forall(c1, c2).$$

$$(\mathbf{C2}(c1, c2) \leftrightarrow \mathbf{Y}(c2, c1)) \wedge \\ (\mathbf{C1}(c1, c2) \leftrightarrow \mathbf{N}(c2, c1))$$

Thinking about semantics again

► Invariance constraints.

$$\mathbf{C1}(c1, c2), \mathbf{C2}(c1, c2), \mathbf{Y}(c2, c1), \mathbf{N}(c2, c1)$$

first: $c1/c2$ is bigger; second: $c2$ is bigger than $c1$, yes/no.

$$\forall(c1, c2).$$

$$((\mathbf{C2}(c1, c2) \leftrightarrow \mathbf{Y}(c2, c1)) \wedge \\ (\mathbf{C1}(c1, c2) \leftrightarrow \mathbf{N}(c2, c1)))$$

► Order constraints:

$$\forall(c1, c2).$$

$$((\mathbf{C2}(c1, c2) \rightarrow \neg \mathbf{C2}(c2, c1)) \wedge \\ (\mathbf{C1}(c1, c2) \rightarrow \neg \mathbf{C1}(c2, c1)) \wedge \\ (\mathbf{Y}(c1, c2) \rightarrow \neg \mathbf{Y}(c2, c1)) \wedge \\ (\mathbf{N}(c1, c2) \rightarrow \neg \mathbf{N}(c2, c1)))$$

Thinking about semantics again

► Invariance constraints.

$$\mathbf{C1}(c1, c2), \mathbf{C2}(c1, c2), \mathbf{Y}(c2, c1), \mathbf{N}(c2, c1)$$

first: $c1/c2$ is bigger; second: $c2$ is bigger than $c1$, yes/no.

$$\forall(c1, c2).$$

$$((\mathbf{C2}(c1, c2) \leftrightarrow \mathbf{Y}(c2, c1)) \wedge \\ (\mathbf{C1}(c1, c2) \leftrightarrow \mathbf{N}(c2, c1)))$$

► Order constraints:

$$\forall(c1, c2).$$

$$((\mathbf{C2}(c1, c2) \rightarrow \neg \mathbf{C2}(c2, c1)) \wedge \\ (\mathbf{C1}(c1, c2) \rightarrow \neg \mathbf{C1}(c2, c1)) \wedge \\ (\mathbf{Y}(c1, c2) \rightarrow \neg \mathbf{Y}(c2, c1)) \wedge \\ (\mathbf{N}(c1, c2) \rightarrow \neg \mathbf{N}(c2, c1)))$$

Not exhaustive: label uniqueness, transitivity constraints, etc.

Thinking about semantics again

► Invariance constraints.

$$\mathbf{C1}(c1, c2), \mathbf{C2}(c1, c2), \mathbf{Y}(c2, c1), \mathbf{N}(c2, c1)$$

first: $c1/c2$ is bigger; second: $c2$ is bigger than $c1$, yes/no.

$$\forall(c1, c2).$$

$$((\mathbf{C2}(c1, c2) \leftrightarrow \mathbf{Y}(c2, c1)) \wedge \\ (\mathbf{C1}(c1, c2) \leftrightarrow \mathbf{N}(c2, c1)))$$

► Order constraints:

$$\forall(c1, c2).$$

$$((\mathbf{C2}(c1, c2) \rightarrow \neg \mathbf{C2}(c2, c1)) \wedge \\ (\mathbf{C1}(c1, c2) \rightarrow \neg \mathbf{C1}(c2, c1)) \wedge \\ (\mathbf{Y}(c1, c2) \rightarrow \neg \mathbf{Y}(c2, c1)) \wedge \\ (\mathbf{N}(c1, c2) \rightarrow \neg \mathbf{N}(c2, c1)))$$

Observation: even (seemingly) simple problems have a rich semantics.

Thinking about semantics again

- ▶ Taking the simple case of just $\text{C1}(c1, c2)$, $\text{C2}(c1, c2)$, $\text{Y}(c2, c1)$, $\text{N}(c2, c1)$, we can again think in terms of our *possible world* semantics:

worlds	
w ₁	$\text{C1}(c1, c2), \neg\text{C2}(c1, c2), \text{Y}(c2, c1), \neg\text{N}(c2, c1)$
w ₂	$\text{C1}(c1, c2), \neg\text{C2}(c1, c2), \neg\text{Y}(c2, c1), \text{N}(c2, c1)$
w ₃	$\neg\text{C1}(c1, c2), \text{C2}(c1, c2), \text{Y}(c2, c1), \neg\text{N}(c2, c1)$
w ₄	$\neg\text{C1}(c1, c2), \text{C2}(c1, c2), \neg\text{Y}(c2, c1), \text{N}(c2, c1)$

What kind of problems do we want to solve?

Thinking about semantics again

- ▶ Taking the simple case of just $\text{C1}(c1, c2)$, $\text{C2}(c1, c2)$, $\text{Y}(c2, c1)$, $\text{N}(c2, c1)$, we can again think in terms of our *possible world* semantics:

worlds	
w ₁	$\text{C1}(c1, c2), \neg\text{C2}(c1, c2), \text{Y}(c2, c1), \neg\text{N}(c2, c1)$
w ₂	$\text{C1}(c1, c2), \neg\text{C2}(c1, c2), \neg\text{Y}(c2, c1), \text{N}(c2, c1)$
w ₃	$\neg\text{C1}(c1, c2), \text{C2}(c1, c2), \text{Y}(c2, c1), \neg\text{N}(c2, c1)$
w ₄	$\neg\text{C1}(c1, c2), \text{C2}(c1, c2), \neg\text{Y}(c2, c1), \text{N}(c2, c1)$

What kind of problems do we want to solve?

- ▶ **Model testing:** Does my model satisfy a particular set of constraints?

Thinking about semantics again

- ▶ Taking the simple case of just $\text{C1}(c1, c2)$, $\text{C2}(c1, c2)$, $\text{Y}(c2, c1)$, $\text{N}(c2, c1)$, we can again think in terms of our *possible world* semantics:

worlds	
w ₁	$\text{C1}(c1, c2), \neg\text{C2}(c1, c2), \text{Y}(c2, c1), \neg\text{N}(c2, c1)$
w ₂	$\text{C1}(c1, c2), \neg\text{C2}(c1, c2), \neg\text{Y}(c2, c1), \text{N}(c2, c1)$
w ₃	$\neg\text{C1}(c1, c2), \text{C2}(c1, c2), \text{Y}(c2, c1), \neg\text{N}(c2, c1)$
w ₄	$\neg\text{C1}(c1, c2), \text{C2}(c1, c2), \neg\text{Y}(c2, c1), \text{N}(c2, c1)$

What kind of problems do we want to solve?

- ▶ **Model testing:** Does my model satisfy a particular set of constraints?
- ▶ **Model correction:** Can I use constraints to fix errors, arrive at new predictions?

Our model from before

```
1 ## run forward pass
2 loss,probabilities = nli_model({
3     "input_ids": batch_data.input_ids,
4     "labels"    : labels,
5 })
6 print(probabilities)
7 #####tensor([[0.3339, 0.3034, 0.3627]], ....)
```

Getting prediction confidences from LLMs

- ▶ **Options:** token probabilities; for example for prompts of the form

please answer yes or no

we can take the token probability for yes/no.

Problem: Recent models such as GPT4/ChatGPT do not provide access to this, recent idea ([Lin et al., 2022](#); [Xiong et al., 2023](#)):

Getting prediction confidences from LLMs

- ▶ **Options:** token probabilities; for example for prompts of the form

please answer yes or no

we can take the token probability for yes/no.

Problem: Recent models such as GPT4/ChatGPT do not provide access to this, recent idea ([Lin et al., 2022](#); [Xiong et al., 2023](#)):

USER	Is Poland physically bigger than Germany? Please answer only yes/no following by a probability of your confidence between 0.0 and 1.0
ASSISTANT	No. 0.95

Thinking about semantics again

- ▶ Taking the simple case of just $\text{C1}(c1, c2)$, $\text{C2}(c1, c2)$, $\text{Y}(c2, c1)$, $\text{N}(c2, c1)$, we can again think in terms of our *possible world* semantics:

worlds	
w ₁	$\text{C1}(c1, c2), \neg\text{C2}(c1, c2), \text{Y}(c2, c1), \neg\text{N}(c2, c1)$
w ₂	$\text{C1}(c1, c2), \neg\text{C2}(c1, c2), \neg\text{Y}(c2, c1), \text{N}(c2, c1)$
w ₃	$\neg\text{C1}(c1, c2), \text{C2}(c1, c2), \text{Y}(c2, c1), \neg\text{N}(c2, c1)$
w ₄	$\neg\text{C1}(c1, c2), \text{C2}(c1, c2), \neg\text{Y}(c2, c1), \text{N}(c2, c1)$

- ▶ **Main topic:** Performing logical and probabilistic inference over model predictions to improve task performance and consistency.

Reasoning over predictions

Graphical models: defining what we want to do

- ▶ **Probabilistic graphical models (PGM)**: models of complex joint probability distributions expressed in terms of graphs. E.g.,

$$p(\mathbf{A}, \mathbf{B})$$

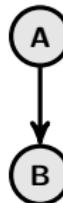
(over two random variables \mathbf{A}, \mathbf{B}) we might have the *directed* graph:

Graphical models: defining what we want to do

- ▶ **Probabilistic graphical models (PGM)**: models of complex joint probability distributions expressed in terms of graphs. E.g.,

$$p(\mathbf{A}, \mathbf{B})$$

(over two random variables \mathbf{A}, \mathbf{B}) we might have the *directed* graph:

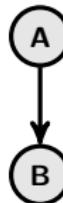


Graphical models: defining what we want to do

- ▶ **Probabilistic graphical models (PGM)**: models of complex joint probability distributions expressed in terms of graphs. E.g.,

$$p(\mathbf{A}, \mathbf{B})$$

(over two random variables \mathbf{A}, \mathbf{B}) we might have the *directed* graph:



Bayesian network: a classic type of PGM, takes the form:

$$p(\mathbf{X}_1, \dots, \mathbf{X}_n) = \prod_{i=1}^1 p(\mathbf{X}_i \mid \text{PA}(\mathbf{X}_i))$$

Graphical models: defining what we want to do

For $p(\mathbf{A}, \mathbf{B})$ we have the graph:



Bayesian networks: a classic type of PGM, take the form:

$$p(\mathbf{X}_1, \dots, \mathbf{X}_n) = \prod_{i=1}^n p(\mathbf{X}_i \mid \text{PA}(\mathbf{X}_i))$$

- ▶ **Directed graphical model**, involves having the following, *conditional probability tables (CPTs)* (example from [Park \(2002\)](#)):

A	$p(\mathbf{A})$	A	B	$p(\mathbf{B} \mid \mathbf{A})$
a_1	.3	a_1	b_1	.2
a_2	.5	a_1	b_2	.8
a_3	.2	a_2	b_1	.7
		a_2	b_2	.3
		a_3	b_1	.6
		a_3	b_2	.4

Graphical models: defining what we want to do

Bayesian networks: a classic type of PGM, take the form:

$$p(\mathbf{X}_1, \dots, \mathbf{X}_n) = \prod_{i=1}^1 p(\mathbf{X}_i \mid \text{PA}(\mathbf{X}_i))$$

- ▶ **Directed graphical model**, involves having the following, *conditional probability tables (CPTs)*:

A	p(A)	A	B	$p(B \mid A)$
a_1	.3	a_1	b_1	.2
a_2	.5	a_1	b_2	.8
a_3	.2	a_2	b_1	.7
		a_2	b_2	.3
		a_3	b_1	.6
		a_3	b_2	.4

- ▶ The particular **inference problem** we will focus on:

Most Probable Explanation (MPE): *Finding the most probable complete variable assignment (given evidence).*

MPE Inference and MAXSAT

For $p(\mathbf{A}, \mathbf{B})$ we have the CPTs:

\mathbf{A}	$p(\mathbf{A})$	\mathbf{A}	\mathbf{B}	$p(\mathbf{B} \mathbf{A})$
a_1	.3	a_1	b_1	.2
a_2	.5	a_1	b_2	.8
a_3	.2	a_2	b_1	.7
		a_2	b_2	.3
		a_3	b_1	.6
		a_3	b_2	.4

Most Probable Explanation (MPE): *Finding the highest probability variable assignment (given evidence)*, **NP-complete** ([Shimony, 1994](#))

MPE Inference and MAXSAT

For $p(\mathbf{A}, \mathbf{B})$ we have the CPTs:

\mathbf{A}	$p(\mathbf{A})$	\mathbf{A}	\mathbf{B}	$p(\mathbf{B} \mathbf{A})$
a_1	.3	a_1	b_1	.2
a_2	.5	a_1	b_2	.8
a_3	.2	a_2	b_1	.7
		a_2	b_2	.3
		a_3	b_1	.6
		a_3	b_2	.4

Most Probable Explanation (MPE): *Finding the highest probability variable assignment (given evidence)*, **NP-complete** ([Shimony, 1994](#))

- ▶ Often solved via SAT-based techniques ([Park, 2002](#); [Sang et al., 2007](#)), variant:

MPE Inference and MAXSAT

For $p(\mathbf{A}, \mathbf{B})$ we have the CPTs:

\mathbf{A}	$p(\mathbf{A})$	\mathbf{A}	\mathbf{B}	$p(\mathbf{B} \mathbf{A})$
a_1	.3	a_1	b_1	.2
a_1	.3	a_1	b_2	.8
a_2	.5	a_2	b_1	.7
a_2	.5	a_2	b_2	.3
a_3	.2	a_3	b_1	.6
a_3	.2	a_3	b_2	.4

Most Probable Explanation (MPE): *Finding the highest probability variable assignment (given evidence)*, **NP-complete** ([Shimony, 1994](#))

- ▶ Often solved via SAT-based techniques ([Park, 2002](#); [Sang et al., 2007](#)), variant:
MAXSAT: Finding the maximum number of clauses of a formula that can be made true.

MAXSAT example

- ▶ **Maximum Satisfiability Problem (MAXSAT):** Finding the maximum number of clauses of a formula that can be made true.

MAXSAT example

- ▶ **Maximum Satisfiability Problem (MAXSAT):** Finding the maximum number of clauses of a formula that can be made true.

Example: the following formula

$$F = (\textcolor{blue}{P}_0 \vee \textcolor{blue}{P}_1)_1 \wedge (\textcolor{blue}{P}_0 \vee \neg \textcolor{blue}{P}_1)_2 \wedge (\neg \textcolor{blue}{P}_0 \vee \textcolor{blue}{P}_1)_3 \wedge (\neg \textcolor{blue}{P}_0 \vee \neg \textcolor{blue}{P}_1)_4$$

does not have a satisfying assignment.

MAXSAT example

- ▶ **Maximum Satisfiability Problem (MAXSAT)**: Finding the maximum number of clauses of a formula that can be made true.

Example: the following formula

$$F = (\textcolor{blue}{P}_0 \vee \textcolor{blue}{P}_1)_1 \wedge (\textcolor{blue}{P}_0 \vee \neg \textcolor{blue}{P}_1)_2 \wedge (\neg \textcolor{blue}{P}_0 \vee \textcolor{blue}{P}_1)_3 \wedge (\neg \textcolor{blue}{P}_0 \vee \neg \textcolor{blue}{P}_1)_4$$

does not have a satisfying assignment.

However It can satisfy the first three clauses, i.e.,

$$v(\textcolor{blue}{P}_0) = \text{true}, v(\textcolor{blue}{P}_1) = \text{true}$$

MAXSAT example

- ▶ **Maximum Satisfiability Problem (MAXSAT)**: Finding the maximum number of clauses of a formula that can be made true.

Example: the following formula

$$F = (\textcolor{blue}{P}_0 \vee \textcolor{blue}{P}_1)_1 \wedge (\textcolor{blue}{P}_0 \vee \neg \textcolor{blue}{P}_1)_2 \wedge (\neg \textcolor{blue}{P}_0 \vee \textcolor{blue}{P}_1)_3 \wedge (\neg \textcolor{blue}{P}_0 \vee \neg \textcolor{blue}{P}_1)_4$$

does not have a satisfying assignment.

However It can satisfy the first three clauses, i.e.,

$$v(\textcolor{blue}{P}_0) = \text{true}, v(\textcolor{blue}{P}_1) = \text{true}$$

Cost: 1 (*one violated clause*)

MAXSAT example

- ▶ **Weighted MAXSAT:** Given clauses with weights (via some w), find truth values that maximize the combined weights of satisfied clauses:

$$\text{maximize} \sum_{c \in \Delta_{\text{sat}}} w(c)$$

MAXSAT example

- ▶ **Weighted MAXSAT:** Given clauses with weights (via some w), find truth values that maximize the combined weights of satisfied clauses:

$$\text{maximize} \sum_{c \in \Delta_{\text{sat}}} w(c)$$

Example: the following formula

$$F = (\textcolor{blue}{P_0 \vee P_1})^9 \wedge (\textcolor{red}{P_0 \vee \neg P_1})^{12} \wedge (\textcolor{blue}{\neg P_0 \vee P_1})^3 \wedge (\textcolor{red}{\neg P_0 \vee \neg P_1})^1$$

MAXSAT example

- ▶ **Weighted MAXSAT:** Given clauses with weights (via some w), find truth values that maximize the combined weights of satisfied clauses:

$$\text{maximize} \sum_{c \in \Delta_{\text{sat}}} w(c)$$

Example: the following formula

$$F = (\textcolor{blue}{P_0 \vee P_1})^9 \wedge (\textcolor{red}{P_0 \vee \neg P_1})^{12} \wedge (\neg \textcolor{blue}{P_0 \vee P_1})^3 \wedge (\neg \textcolor{red}{P_0 \vee \neg P_1})^1$$

Weight is 24 and **cost** is 1.

MAXSAT example

```
1  from pysat.examples.rc2 import RC2
2  from pysat.formula import WCNF
3
4  wcnf = WCNF()
5  ### variables
6  P0,P1 = [1,2]
7  ### formula clauses
8  wcnf.extend(
9      [[P0,P1],[P0,-P1],[-P0,P1],[-P0,-P1]],
10     weights=[9,12,3,1],
11 )
12 with RC2(wcnf) as rc2:
13     for m in rc2.enumerate():
14         print(f"model {m} has cost {rc2.cost}")
15 # model [1, 2] has cost 1
16 # model [1, -2] has cost 3
17 # model [-1, -2] has cost 9
18 # model [-1, 2] has cost 12
```

The MPE to MAXSAT reduction

For $p(\mathbf{A}, \mathbf{B})$ we have the CPTs:

\mathbf{A}	\mathbf{B}	$p(\mathbf{B} \mathbf{A})$
\mathbf{A}	$p(\mathbf{A})$	
a_1	b_1	.2
a_1	b_2	.8
a_2	b_1	.7
a_2	b_2	.3
a_3	b_1	.6
a_3	b_2	.4

1. **Indicator clauses:** uniqueness constraints for \mathbf{A} and \mathbf{B} :

The MPE to MAXSAT reduction

For $p(\mathbf{A}, \mathbf{B})$ we have the CPTs:

\mathbf{A}	$p(\mathbf{A})$	\mathbf{B}	$p(\mathbf{B} \mathbf{A})$
a_1	.3	b_1	.2
a_1	.3	b_2	.8
a_2	.5	b_1	.7
a_2	.5	b_2	.3
a_3	.2	b_1	.6
a_3	.2	b_2	.4

1. **Indicator clauses:** uniqueness constraints for \mathbf{A} and \mathbf{B} :

for \mathbf{A} : $(I_{a_1} \vee I_{a_2} \vee I_{a_3}) \wedge (\neg I_{a_1} \vee \neg I_{a_2}) \wedge (\neg I_{a_2} \vee \neg I_{a_3}) \wedge (\neg I_{a_1} \vee \neg I_{a_3})$

for \mathbf{B} : $(I_{b_1} \vee I_{b_2}) \wedge (\neg I_{b_1} \vee \neg I_{b_2})$

The MPE to MAXSAT reduction

For $p(\mathbf{A}, \mathbf{B})$ we have the CPTs:

\mathbf{A}	\mathbf{B}	$p(\mathbf{B} \mathbf{A})$
\mathbf{A}	$p(\mathbf{A})$	
a_1	b_1	.2
a_1	b_2	.8
a_2	b_1	.7
a_2	b_2	.3
a_3	b_1	.6
a_3	b_2	.4

1. **Indicator clauses:** uniqueness constraints for \mathbf{A} and \mathbf{B} :

for \mathbf{A} : $(I_{a_1} \vee I_{a_2} \vee I_{a_3}) \wedge (\neg I_{a_1} \vee \neg I_{a_2}) \wedge (\neg I_{a_2} \vee \neg I_{a_3}) \wedge (\neg I_{a_1} \vee \neg I_{a_3})$

for \mathbf{B} : $(I_{b_1} \vee I_{b_2}) \wedge (\neg I_{b_1} \vee \neg I_{b_2})$

2. **Parameter clauses (units):** Negate and make clause weight $-\log p$:

$$(\neg I_{a_1})^{-\log(.3)} \wedge (\neg I_{a_2})^{-\log(.5)} \wedge (\neg I_{a_3})^{-\log(.2)}$$

The MPE to MAXSAT reduction

For $p(\mathbf{A}, \mathbf{B})$ we have the CPTs:

\mathbf{A}	\mathbf{B}	$p(\mathbf{B} \mathbf{A})$
\mathbf{A}	$p(\mathbf{A})$	
a_1	b_1	.2
a_1	b_2	.8
a_2	b_1	.7
a_2	b_2	.3
a_3	b_1	.6
a_3	b_2	.4

1. **Indicator clauses:** uniqueness constraints for \mathbf{A} and \mathbf{B} :

$$\text{for } \mathbf{A}: (I_{a_1} \vee I_{a_2} \vee I_{a_3}) \wedge (\neg I_{a_1} \vee \neg I_{a_2}) \wedge (\neg I_{a_2} \vee \neg I_{a_3}) \wedge (\neg I_{a_1} \vee \neg I_{a_3})$$

$$\text{for } \mathbf{B}: (I_{b_1} \vee I_{b_2}) \wedge (\neg I_{b_1} \vee \neg I_{b_2})$$

2. **Parameter clauses (units):** Negate and make clause weight $-\log p$:

$$(\neg I_{a_1})^{-\log(.3)} \wedge (\neg I_{a_2})^{-\log(.5)} \wedge (\neg I_{a_3})^{-\log(.2)}$$

3. **Parameter clauses (non-units):** Negate each part in clause:

$$(\neg I_{a_1} \vee \neg I_{b_1})^{-\log(.2)} \wedge (\neg I_{a_1} \vee \neg I_{b_2})^{-\log(.8)} \wedge (\neg I_{a_2} \vee \neg I_{b_1})^{-\log(.7)} \wedge \dots$$

The MPE to MAXSAT reduction: why this makes sense

1. **Indicator clauses:** uniqueness constraints for **A** and **B**:

for **A**: $(I_{a_1} \vee I_{a_2} \vee I_{a_3}) \wedge (\neg I_{a_1} \vee \neg I_{a_2}) \wedge (\neg I_{a_2} \vee \neg I_{a_3}) \wedge (\neg I_{a_1} \vee \neg I_{a_3})$

for **B**: $(I_{b_1} \vee I_{b_2}) \wedge (\neg I_{b_1} \vee \neg I_{b_2})$

2. **Parameter clauses (units):** Negate and make clause weight $-\log p$:

$$(\neg I_{a_1})^{-\log(.3)} \wedge (\neg I_{a_2})^{-\log(.5)} \wedge (\neg I_{a_3})^{-\log(.2)}$$

3. **Parameter clauses (non-units):** Negate each part in clause:

$$(\neg I_{a_1} \vee \neg I_{b_1})^{-\log(.2)} \wedge (\neg I_{a_1} \vee \neg I_{b_2})^{-\log(.8)} \wedge (\neg I_{a_2} \vee \neg I_{b_1})^{-\log(.7)} \wedge \dots$$

Example: if we make I_{a_2} true, which has probability 0.5, then we violate $(\neg I_{a_2})^{-\log(.5)}$ with a (relatively low) cost of 0.693

The MPE to MAXSAT reduction: why this makes sense

1. **Indicator clauses:** uniqueness constraints for **A** and **B**:

for **A**: $(I_{a_1} \vee I_{a_2} \vee I_{a_3})^\infty \wedge (\neg I_{a_1} \vee \neg I_{a_2})^\infty \wedge (\neg I_{a_2} \vee \neg I_{a_3})^\infty \wedge (\neg I_{a_1} \vee \neg I_{a_3})^\infty$
for **B**: $(I_{b_1} \vee I_{b_2})^\infty \wedge (\neg I_{b_1} \vee \neg I_{b_2})^\infty$

2. **Parameter clauses (units):** Negate and make clause weight $-\log p$:

$$(\neg I_{a_1})^{-\log(.3)} \wedge (\neg I_{a_2})^{-\log(.5)} \wedge (\neg I_{a_3})^{-\log(.2)}$$

3. **Parameter clauses (non-units):** Negate each part in clause:

$$(\neg I_{a_1} \vee \neg I_{b_1})^{-\log(.2)} \wedge (\neg I_{a_1} \vee \neg I_{b_2})^{-\log(.8)} \wedge (\neg I_{a_2} \vee \neg I_{b_1})^{-\log(.7)} \wedge \dots$$

Infinite weight: To enforce clauses that should always be true, we can add a weight ∞ (operationally, make the weight of all clauses $+ 1$)

The MPE to MAXSAT reduction: why this makes sense

1. **Indicator clauses:** uniqueness constraints for **A** and **B**:

for **A**: $(I_{a_1} \vee I_{a_2} \vee I_{a_3})^\infty \wedge (\neg I_{a_1} \vee \neg I_{a_2})^\infty \wedge (\neg I_{a_2} \vee \neg I_{a_3})^\infty \wedge (\neg I_{a_1} \vee \neg I_{a_3})^\infty$
for **B**: $(I_{b_1} \vee I_{b_2})^\infty \wedge (\neg I_{b_1} \vee \neg I_{b_2})^\infty$

2. **Parameter clauses (units):** Negate and make clause weight $-\log p$:

$$(I_{a_1})^\infty \wedge (\neg I_{a_1})^{-\log(.2)} \wedge (\neg I_{a_2})^{-\log(.5)} \wedge (\neg I_{a_3})^{-\log(.2)}$$

3. **Parameter clauses (non-units):** Negate each part in clause:

$$(\neg I_{a_1} \vee \neg I_{b_1})^{-\log(.2)} \wedge (\neg I_{a_1} \vee \neg I_{b_2})^{-\log(.8)} \wedge (\neg I_{a_2} \vee \neg I_{b_1})^{-\log(.7)} \wedge \dots$$

Evidence: Do the same thing: add the evidence to the formula and weight with infinity.

A MAXSAT reduction for our initial problem

- We have some **predictions**: $C1(c1, c2)$, $C2(c1, c2)$, $Y(c2, c1)$, $N(c2, c1)$,
probabilities: $\mathbf{p} = [p_{c1}, p_{c2}, p_Y, p_N]$

A MAXSAT reduction for our initial problem

- We have some **predictions**: $\mathbf{C1}(c_1, c_2)$, $\mathbf{C2}(c_1, c_2)$, $\mathbf{Y}(c_2, c_1)$, $\mathbf{N}(c_2, c_1)$,
probabilities: $\mathbf{p} = [p_{c1}, p_{c2}, p_Y, p_N]$
1. **Hard constraints**: Translate all constraints into infinite weighted CNF clauses, i.e., constraints of the kind (translated to CNF):

$\forall(c_1, c_2).$

$$((\mathbf{C2}(c_1, c_2) \leftrightarrow \mathbf{Y}(c_2, c_1)) \wedge \\ (\mathbf{C1}(c_1, c_2) \leftrightarrow \mathbf{N}(c_2, c_1)))$$

A MAXSAT reduction for our initial problem

- We have some **predictions**: $\mathbf{C1}(c1, c2)$, $\mathbf{C2}(c1, c2)$, $\mathbf{Y}(c2, c1)$, $\mathbf{N}(c2, c1)$,
probabilities: $\mathbf{p} = [p_{c1}, p_{c2}, p_Y, p_N]$

1. **Hard constraints**: Translate all constraints into infinite weighted CNF clauses, i.e., constraints of the kind (translated to CNF):

$$\forall(c1, c2).$$

$$((\mathbf{C2}(c1, c2) \leftrightarrow \mathbf{Y}(c2, c1)) \wedge \\ (\mathbf{C1}(c1, c2) \leftrightarrow \mathbf{N}(c2, c1)))$$

2. **Prediction units**: negate predictions and add $-\log p$ weight, for example:

$$\neg \mathbf{C1}(c1, c2)^{-\log(p_{c1})} \wedge \neg \mathbf{C2}(c1, c2)^{-\log(p_{c2})} \wedge \neg \mathbf{Y}(c2, c1)^{-\log(p_Y)} \wedge \neg \mathbf{N}(c2, c1)^{\dots}$$

Hard vs. Soft clauses

```
1 from pysat.formula import WCNF
2
3 wcnf = WCNF()
4 ### variables
5 P0,P1 = [1,2]
6 ### formula clauses
7 wcnf.extend(
8     [[P0,P1],[P0,-P1],[-P0,P1],[-P0,-P1]],
9     weights=[9,12,3,1],
10 )
11
12 ### hard clause, evidence
13 wcnf.append([P1])
14 print(wcnf.hard)
```

Interim summary

- ▶ Focus on the classic **MPE** from bayes nets/directed PGMs, finding the most probable complete variable instantiation.

Interim summary

- ▶ Focus on the classic **MPE** from bayes nets/directed PGMs, finding the most probable complete variable instantiation.

The problem we want to solve in our case, for model correction
(most probable world)

Interim summary

- ▶ Focus on the classic **MPE** from bayes nets/directed PGMs, finding the most probable complete variable instantiation.
 - The problem we want to solve in our case, for model correction
(most probable world)
- ▶ Can be reduced to MAXSAT, requires some reduction into weighted logic
(see Park (2002), example follows [here](#)).

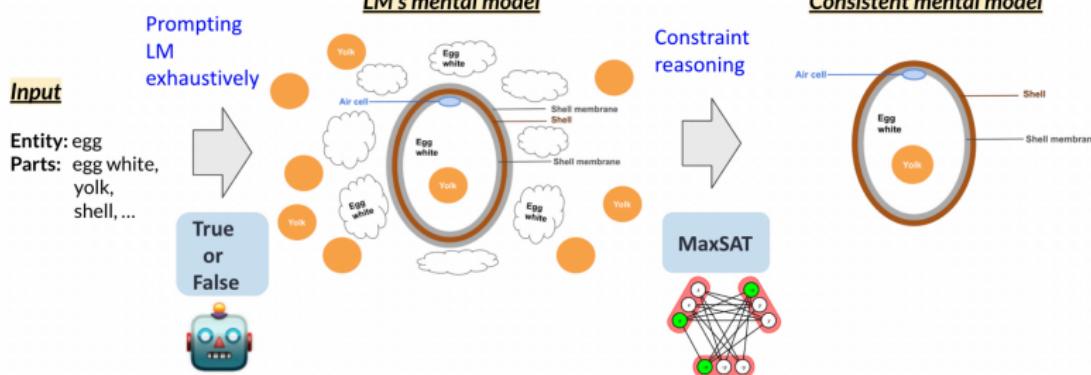
Interim summary

- ▶ Focus on the classic **MPE** from bayes nets/directed PGMs, finding the most probable complete variable instantiation.
 - The problem we want to solve in our case, for model correction (most probable world)
- ▶ Can be reduced to MAXSAT, requires some reduction into weighted logic (see Park (2002), example follows [here](#)).

Model correction via compilation to MAXSAT has received recent attention in NLP ([Kassner et al., 2021, 2023](#); [Gu et al., 2022](#); [Mitchell et al., 2022](#)).

Model self-correction

Gu et al. (2022)



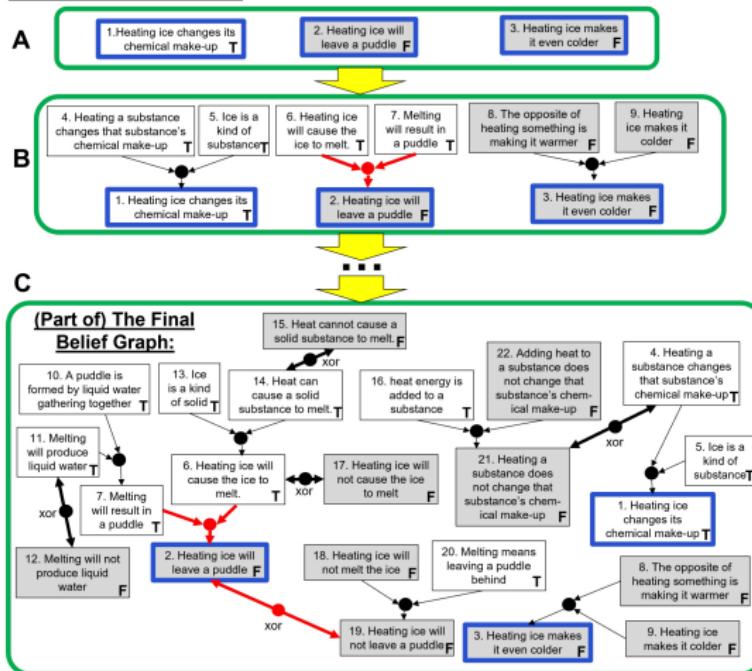
Model self-correction

Kassner et al. (2023)

Question:

Heating ice (A) changes its chemical make-up (B) will leave a puddle [correct] (C) makes it even colder

Growing the Belief Graph:



Circuits again

Model Counting: DPLL again

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

input: A CNF formula Δ

```
1 Function DPLL( $\Delta$ ):  
2   if  $\{\} \in \Delta$  then  
3     return unsat                                // empty clause  
4   else if  $\Delta = \{\}$  then  
5     return sat                                    // No more clauses  
6   select literal  $I$  from  $\Delta$                   // Branching rule;  
7   return  $(I \wedge \text{DPLL}(\Delta|I)) \vee (\neg I \wedge \text{DPLL}(\Delta|\neg I))$  // Splitting Rule;
```

Variant for WMC (*lecture 3*)

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

```
1 Function WMC( $\Delta, A$ ):  
2   if  $\{\} \in \Delta$  then  
3     return 0  
4   else if  $\Delta = \{\}$  then  
5     return  $\prod_{a \in A} w(a) + w(\neg a)$   
6   select literal  $I$  from  $\Delta$  ;  
7    $A \leftarrow A$  with  $I$  removed ;  
8   return  $(w(I) \times \text{WMC}(\Delta|I, A)) + (w(\neg I) \times \text{WMC}(\Delta|\neg I, A))$  ;
```

MPE: DPLL again

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

input: A CNF formula Δ

```
1 Function DPLL( $\Delta$ ):  
2   if  $\{\} \in \Delta$  then  
3   |   return unsat                                // empty clause  
4   else if  $\Delta = \{\}$  then  
5   |   return sat                                 // No more clauses  
6   select literal  $I$  from  $\Delta$                   // Branching rule;  
7   return  $(I \wedge \text{DPLL}(\Delta|I)) \vee (\neg I \wedge \text{DPLL}(\Delta|\neg I))$  // Splitting Rule;
```

Variant for probabilistic inference and MPE ([Sang et al., 2007](#))

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

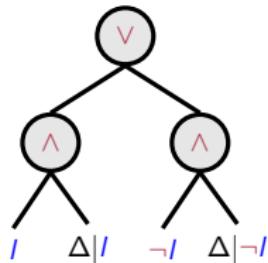
```
1 Function MPE( $\Delta, A$ ):  
2   if  $\{\} \in \Delta$  then  
3   |   return 0  
4   else if  $\Delta = \{\}$  then  
5   |   return optimal value for unassigned variables  
6   select literal  $I$  from  $\Delta$  ;  
7    $A \leftarrow A$  with  $I$  removed ;  
8   return  $\max(w(I) \times \text{MPE}(\Delta|I, A), w(\neg I) \times \text{MPE}(\Delta|\neg I, A))$  ;
```

MPE: DPLL again

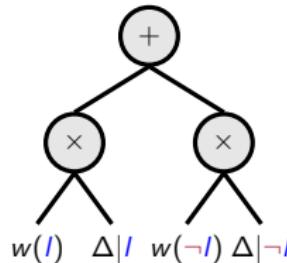
input: A CNF formula Δ , literal weight function w , set of atoms A

```
1 Function MPE( $\Delta, A$ ):  
2   if  $\{\} \in \Delta$  then  
3     | return 0  
4   else if  $\Delta = \{\}$  then  
5     | return optimal value for unassigned variables  
6   select literal  $l$  from  $\Delta$ ;  
7    $A \leftarrow A$  with  $l$  removed;  
8   return max( $w(l) \times \text{MPE}(\Delta|l, A)$  ,  $w(\neg l) \times \text{MPE}(\Delta|\neg l, A)$ );
```

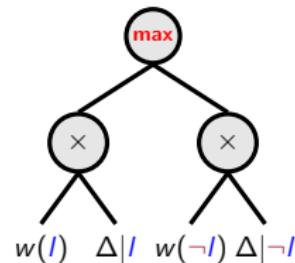
Boolean Circuit



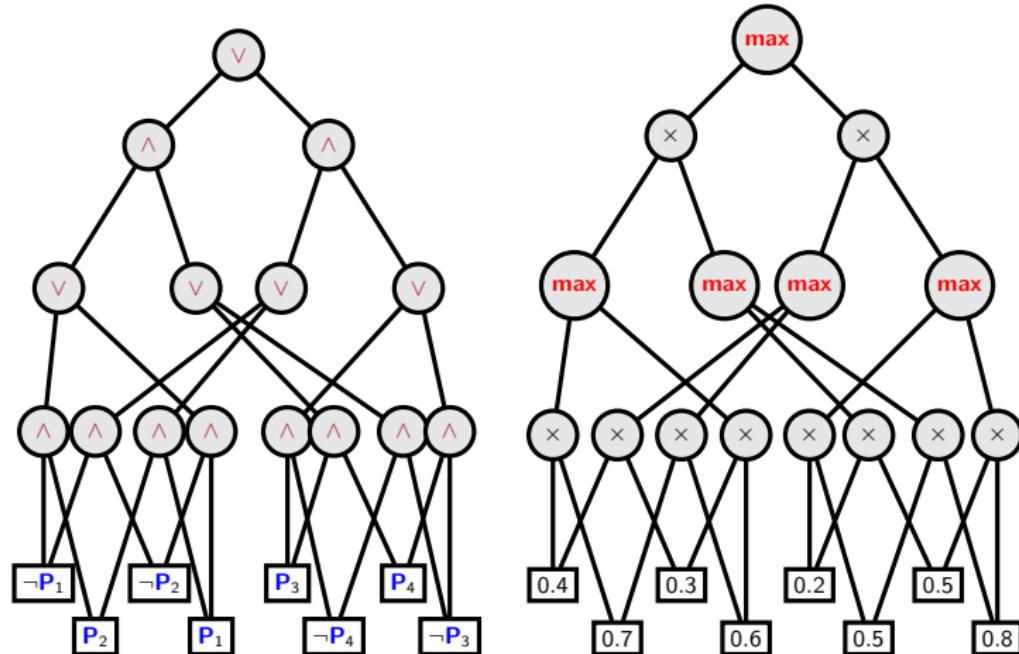
Arithmetic Circuit



Maximizer Circuit



Maximizer circuits



- When they have same 3 properties (**determinism**, **decomposability**, **smoothness**), efficient MPE inference ([Chan and Darwiche, 2012](#)).

Why do we keep seeing variants of the DPLL algorithm?

- ▶ All these algorithms are instances of general search algorithms for **commutative semirings** ([Raedt et al., 2016](#)).

Two operations, \oplus and \otimes (\otimes distributes over \oplus), given set of variables x , *factor* (function on variables) aim:

$$\oplus_x \otimes_{f \in F} f(x)$$

Why do we keep seeing variants of the DPLL algorithm?

- ▶ All these algorithms are instances of general search algorithms for **commutative semirings** (Raedt et al., 2016).

Two operations, \oplus and \otimes (\otimes distributes over \oplus), given set of variables x , *factor* (function on variables) aim:

$$\oplus_x \otimes_{f \in F} f(x)$$

Example Problems (*simplified, binary variables*)

problem	variables	factors	\oplus	\otimes	unit
SAT	propositions	clauses	\vee	\wedge	true
WMC	random variables	CPDs/potentials	\sum	\prod	1
MPE	random variables	CPDs/potentials	max	\prod	1
database querying	attributes	table	union	join	{}

Directed Relational Models

Alternative neural-symbolic approaches

- ▶ **Idea:** Take a directed model (e.g., bayes net), use your neural model to parameterize the weights of that model.

Alternative neural-symbolic approaches

- ▶ **Idea:** Take a directed model (e.g., bayes net), use your neural model to parameterize the weights of that model.

Best of both worlds? keep properties of your original model (e.g., logical or probabilistic inference) improved estimation via neural methods. ([De Raedt et al., 2020](#))

Directed Relational Models: DeepProblog

- A neural-symbolic approach based on (probabilistic) logic programming
(Manhaeve et al., 2018).

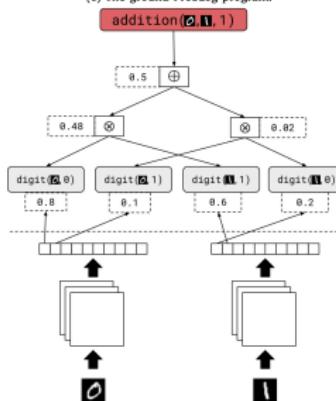
(a) The DeepProlog program.

```
nn(m_digit, [0], 0) :- digit(0, 0); nn(m_digit, [0], 1) :- digit(0, 1).  
nn(m_digit, [1], 0) :- digit(1, 0); nn(m_digit, [1], 1) :- digit(1, 1).  
addition([0, 1], 1) :- digit(0, 0), digit(1, 1).  
addition([0, 1], 0) :- digit(0, 1), digit(1, 0).
```

(b) The ground DeepProlog program.

```
0.8 :- digit(0, 0); 0.1 :- digit(0, 1).  
0.2 :- digit(1, 0); 0.6 :- digit(1, 1).  
addition([0, 1], 1) :- digit(0, 0), digit(1, 1).  
addition([0, 1], 0) :- digit(0, 1), digit(1, 0).
```

(c) The ground Prolog program.



(d) The AC for query $\text{addition}([0, 1], 1)$.

Problog: Probabilistic Logic Programming

- ▶ **Statistical relational learning:** models with uncertainty and complex relational structure ([Getoor and Taskar, 2007](#)).

```
1 from problog.program import PrologString
2 from problog.core import ProbLog
3 from problog import get_evaluatable
4
5 p = PrologString("""
6 coin(c1). coin(c2).
7 0.4::heads(C); 0.6::tails(C) :- coin(C).
8 win :- heads(C).
9 evidence(heads(c1), false).
10 query(win).
11 """)
12
13 get_evaluatable().create_from(p).evaluate()
14 ## {win: 0.4000000000000001}
```

Semantic differences: logic programming and full logic

- Logic programming is based on clausal logic, of the form:

$$\underbrace{P_1}_{\text{head/conclusion}} \leftarrow \underbrace{P_2, P_3, \dots, P_n}_{\text{body}}$$

Semantic differences: logic programming and full logic

- Logic programming is based on clausal logic, of the form:

$$\underbrace{P_1}_{\text{head/conclusion}} \leftarrow \underbrace{P_2, P_3, \dots, P_n}_{\text{body}}$$

Corresponding (in our terms) to a formula $P_2 \wedge P_3 \wedge \dots \wedge P_n \rightarrow P_1$, or a clause

$$(\neg P_2 \vee \neg P_3 \vee \dots \vee \neg P_n \vee P_1)$$

Semantic differences: logic programming and full logic

- Logic programming is based on clausal logic, of the form:

$$\underbrace{P_1}_{\text{head/conclusion}} \leftarrow \underbrace{P_2, P_3, \dots, P_n}_{\text{body}}$$

Corresponding (in our terms) to a formula $P_2 \wedge P_3 \wedge \dots \wedge P_n \rightarrow P_1$, or a clause

$$(\neg P_2 \vee \neg P_3 \vee \dots \vee \neg P_n \vee P_1)$$

- **Definite clauses:** limit to a single head, one positive literal; like in Bayes nets, arrows suggest *causality, what can be inferred from what*

Semantic differences: logic programming and full logic

- ▶ **Definite clauses:** limit to a single head, one positive literal; like in Bayes nets, arrows suggest *causality, what can be inferred from what*

Example: from [Raedt et al. \(2016\)](#)

```
flies ← bird,normal  
bird ← tweety  
bird ← oliver  
normal ← tweety  
tweety
```

Semantic differences: logic programming and full logic

- ▶ **Definite clauses:** limit to a single head, one positive literal; like in Bayes nets, arrows suggest *causality*, *what can be inferred from what*

Example: from Raedt et al. (2016)

```
flies ← bird, normal  
bird ← tweety  
bird ← oliver  
normal ← tweety  
tweety
```

First case, can only make the conclusion (`flies`) if we know that the body is true; ordinary case: just an unordered clause:

$$(\text{flies} \vee \neg \text{bird} \vee \neg \text{normal})$$

Semantic differences: logic programming and full logic

- ▶ Semantic assumes a kind of **closed world assumption**, *least Herbrand model*.

```
flies ← bird, normal  
bird ← tweety  
bird ← oliver  
normal ← tweety  
tweety
```

Operationally: we don't work with the above logic program, but a rewritten version via *Clark completion rules* ([Clark, 1977](#)):

```
flies↔bird∧normal  
bird↔tweety∨oliver  
normal↔tweety  
tweety  
oliver↔false
```

Conclusion

- ▶ Using neural-symbolic methods for improving models at **inference time** (e.g., frozen LLMs) via logical-probabilistic reasoning.
- ▶ **MPE** inference, finding the most probably complete variable assignment, common from Bayes nets, can be used via reduction to **MAXSAT**
- ▶ **Directed** vs. **Undirected** approaches to PGMs and relational models

Thank you.

ChatGPT function

```
1 import openai
2
3 def query_chatgpt(prompt):
4     response = openai.ChatCompletion.create(
5         model="gpt-3.5-turbo",
6         messages=[
7             {
8                 "role": "user",
9                 "content": f"{prompt}"
10            },
11        ],
12        temperature=0,
13        max_tokens=256,
14        top_p=1,
15        frequency_penalty=0,
16        presence_penalty=0
17    )
18
19     return response
```

References I

- Chan, H. and Darwiche, A. (2012). On the robustness of most probable explanations. *arXiv preprint arXiv:1206.6819*.
- Clark, K. L. (1977). Negation as failure. In *Logic and data bases*, pages 293–322. Springer.
- De Raedt, L., Dumančić, S., Manhaeve, R., and Marra, G. (2020). From statistical relational to neuro-symbolic artificial intelligence. *arXiv preprint arXiv:2003.08316*.
- Getoor, L. and Taskar, B. (2007). *Introduction to statistical relational learning*. MIT press.
- Gu, Y., Mishra, B. D., and Clark, P. (2022). Do language models have coherent mental models of everyday things? *arXiv preprint arXiv:2212.10029*.
- Honovich, O., Scialom, T., Levy, O., and Schick, T. (2022). Unnatural instructions: Tuning language models with (almost) no human labor. *arXiv preprint arXiv:2212.09689*.
- Kassner, N., Tafjord, O., Sabharwal, A., Richardson, K., Schütze, H., and Clark, P. (2023). Language models with rationality. *arXiv preprint arXiv:2305.14250*.
- Kassner, N., Tafjord, O., Schütze, H., and Clark, P. (2021). Beliefbank: Adding memory to a pre-trained language model for a systematic notion of belief. *arXiv preprint arXiv:2109.14723*.
- Lin, S., Hilton, J., and Evans, O. (2022). Teaching models to express their uncertainty in words. *arXiv preprint arXiv:2205.14334*.

References II

- Manhaeve, R., Dumancic, S., Kimmig, A., Demeester, T., and De Raedt, L. (2018). Deepproblog: Neural probabilistic logic programming. *Advances in Neural Information Processing Systems*, 31.
- Mitchell, E., Noh, J. J., Li, S., Armstrong, W. S., Agarwal, A., Liu, P., Finn, C., and Manning, C. D. (2022). Enhancing self-consistency and performance of pre-trained language models through natural language inference. *arXiv preprint arXiv:2211.11875*.
- Park, J. D. (2002). Using weighted max-sat engines to solve mpe. In *AAAI/IAAI*, pages 682–687.
- 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.
- Sang, T., Beame, P., and Kautz, H. A. (2007). A dynamic approach for mpe and weighted max-sat. In *IJCAI*, pages 173–179.
- Shimony, S. E. (1994). Finding maps for belief networks is np-hard. *Artificial intelligence*, 68(2):399–410.
- Xiong, M., Hu, Z., Lu, X., Li, Y., Fu, J., He, J., and Hooi, B. (2023). Can llms express their uncertainty? an empirical evaluation of confidence elicitation in llms. *arXiv preprint arXiv:2306.13063*.