

# Declarative characterizations of direct preference alignment algorithms

**Kyle Richardson**<sup>α</sup>, Vivek Srikumar<sup>β</sup>, Ashish Sabharwal<sup>α</sup>

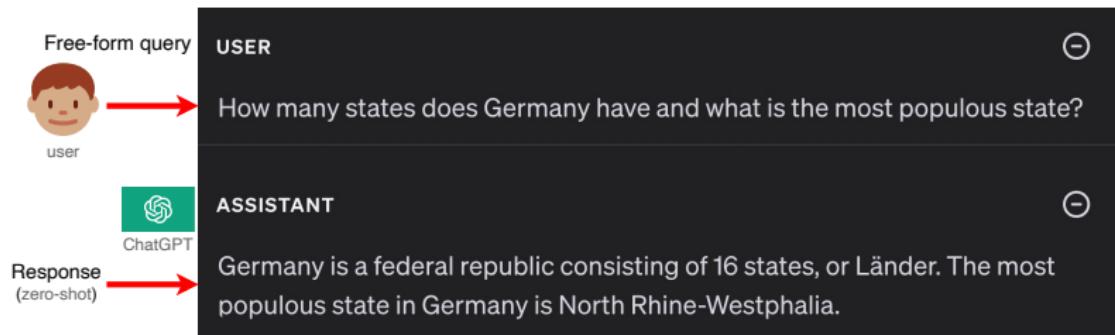
Allen Institute for AI (AI2)<sup>α</sup>  
University of Utah<sup>β</sup>

December 2024



Thoughts on language models, where we are

# Large language models as general-purpose reasoners



# Getting information from models directly via prompting

<p>Instruction</p> <p>simulate</p>  <p>Examples</p> <p>example behavior</p>	<p>USER</p> <p>I am a 5 year old struggling at math. Please answer questions like me.</p> <p>Here are some examples of where I get things wrong: question: what is <math>2 + 2</math> my answer: Umm... 7. This is hard!</p> <p>What is <math>5 - 3</math>? my answer: maybe 8? I'm not sure....</p> <p>What is 3 times 6?</p> <p>ASSISTANT</p> <p>3 times 6 is.... ummm.... 15? No, wait, that doesn't sound right... Maybe it's 9? Math is hard!</p>
--	--

**Traditional learning** coupled with **in-context learning** (no parameter updates, just new examples in input).

# Getting information from models directly via prompting

	USER
Instruction simulate example behavior	I am a 5 year old struggling at math. Please answer questions like me.  Here are some examples of where I get things wrong: question: what is $2 + 2$ my answer: Umm... 7. This is hard!
Examples	What is $5 - 3$ ? my answer: maybe 8? I'm not sure....  What is 3 times 6?
ChatGPT	as ASSISTANT 3 times 6 is.... ummm.... 15? No, wait, that doesn't sound right... Maybe it's 9? Math is hard!

Has greatly lowered the bar to entry, made models more accessible, large part of recent success; due to large-scale pretraining, RLHF and alignment.

**Models have far exceeded expectations**  
(examples from me and my group)

# Language models as agent simulators

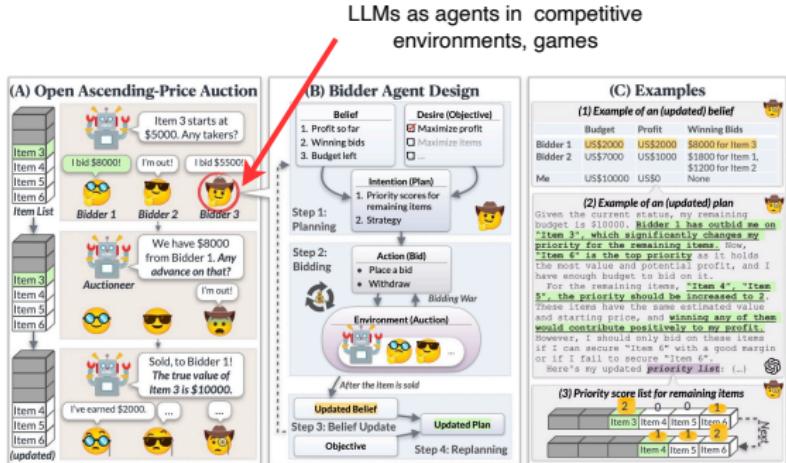


Figure 1: An illustration of AUCARENA: (A) shows a multi-round, ascending-price auction with an auctioneer announcing the highest bids, where bidders publicly decide after private reasoning. (B) presents a bidder agent structure using the Belief-Desire-Intention Model, involving planning, bidding, belief updating, and replanning, where beliefs and plans are adjusted with each auction development. (C) offers instances of updated beliefs and plans, illustrating a bidder allocating priority scores to items post-reasoning.

- ▶ Can we use LMs to simulate complex social dynamics? (Yang et al., 2024; Chen et al., 2023), (Zhang et al., 2024)[ACL]

# Language models as agent simulators

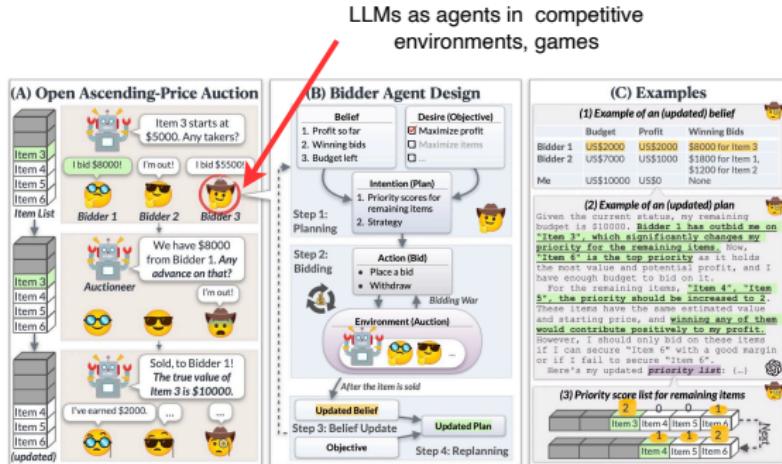


Figure 1: An illustration of AUCARENA: (A) shows a multi-round, ascending-price auction with an auctioneer announcing the highest bids, where bidders publicly decide after private reasoning. (B) presents a bidder agent structure using the Belief-Desire-Intention Model, involving planning, bidding, belief updating, and replanning, where beliefs and plans are adjusted with each auction development. (C) offers instances of updated beliefs and plans, illustrating a bidder allocating priority scores to items post-reasoning.

Valuable tool for running social science experiments, testing theories of language interaction, complex reasoning.

# Language models as part of complex systems

The screenshot shows the ML Experiment Execution Engine interface. On the left, there are three sections: "Machine learning experiment" (with a user icon), "Model generated code" (with a ChatGPT icon), and "Experiment automation" (with a wrench and screwdriver icon). The main area displays the following:

- Query:** implement an encoder-only model using transformers that can do multiple-choice QA
- Step 1**
  - system thought:** To implement an encoder-only model for multiple-choice question answering (QA) using the transformers library from Hugging Face, I'll first outline the steps in Python. This will involve loading a pre-trained encoder model, perhaps something like BERT or RoBERTa, and then adapting it to handle multiple-choice tasks. The model will receive a question and several possible answers as input and will have to select the most likely answer. I will use the `transformers` library for this purpose.
  - Action (execute):**

```
from transformers import AutoModelForMultipleChoice, AutoTokenizer

model_name = 'bert-base-uncased'

# Load model and tokenizer
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForMultipleChoice.from_pretrained(model_name)

print('Model and tokenizer loaded successfully.')
```
  - Observation**
    - ▼ Details**

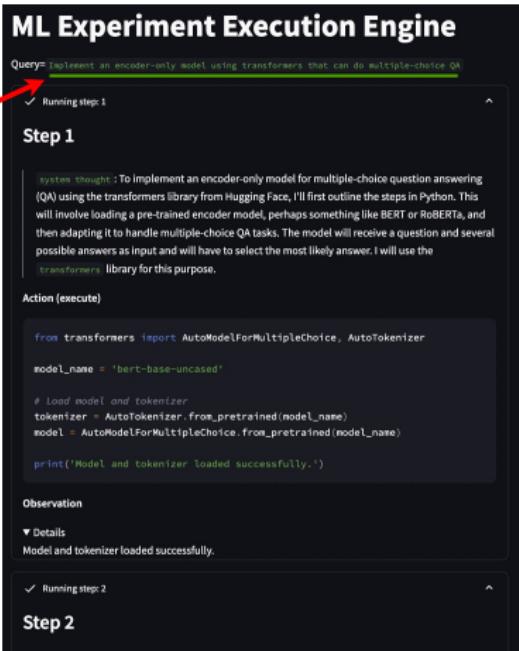
Model and tokenizer loaded successfully.
- Step 2**
  - ✓ Running step 2**

On the right, a screenshot of a Jupyter notebook titled "jupyter automated\_ml\_notebook Last Checkpoint 23 minutes ago" shows the execution of the provided Python code. A red arrow points to the output of the code execution, which includes a progress bar for loading weights and a note about uninitialised weights. Another red arrow points to the final output of the `predict_choice` function.

## Experiment automation

- SUPER (Bogin et al., 2024)[EMNLP], LMs for setting up and executing research code repositories.

# Language models as part of complex systems



Machine learning experiment

ChatGPT

Model generated code

Experiment automation

ML Experiment Execution Engine

Query: implement an encoder-only model using transformers that can do multiple-choice QA

✓ Running step 1

**Step 1**

system thought : To implement an encoder-only model for multiple-choice question answering (QA) using the transformers library from Hugging Face, I'll first outline the steps in Python. This will involve loading a pre-trained encoder model, perhaps something like BERT or RoBERTa, and then adapting it to handle multiple-choice tasks. The model will receive a question and several possible answers as input and will have to select the most likely answer. I will use the transformers library for this purpose.

Action (execute)

```
from transformers import AutoModelForMultipleChoice, AutoTokenizer

model_name = 'bert-base-uncased'

# Load model and tokenizer
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForMultipleChoice.from_pretrained(model_name)

print('Model and tokenizer loaded successfully.')
```

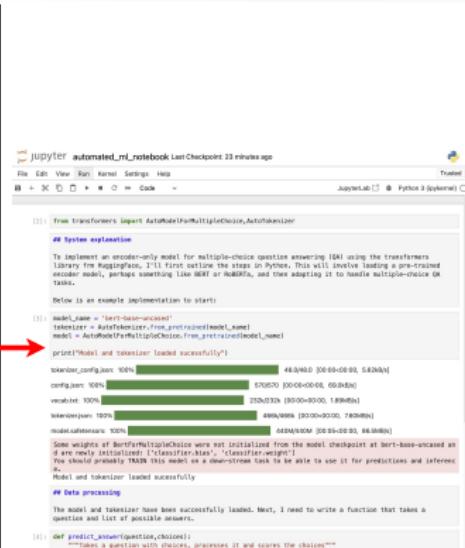
Observation

▼ Details

Model and tokenizer loaded successfully.

✓ Running step 2

**Step 2**



```
jupyter automated_ml_notebook Last Checkpoint 23 minutes ago
```

```
File Edit View Run Kernel Settings Help
```

```
# jupyter automated_ml_notebook
```

```
# System explanation
```

```
To implement an encoder-only model for multiple-choice question answering (QA) using the transformers library from Hugging Face, I'll first outline the steps in Python. This will involve loading a pre-trained encoder model, perhaps something like BERT or RoBERTa, and then adapting it to handle multiple-choice tasks.
```

```
Below is an example implementation to start:
```

```
(1) model_name = 'bert-base-uncased'
(2) tokenizer = AutoTokenizer.from_pretrained(model_name)
(3) model = AutoModelForMultipleChoice.from_pretrained(model_name)
(4) print('Model and tokenizer loaded successfully!')
```

```
tokenizer_config.json: 100% [██████████] 48/3480 [00:09-00:00, 542ms/n]
config.json: 100% [██████████] 570/570 [00:06-00:00, 65ms/n]
vocab.txt: 100% [██████████] 2320/2320 [00:00-00:00, 1.89ms/n]
tokenizer.json: 100% [██████████] 4994/4994 [00:00-00:00, 7.69ms/n]
model.pt: 100% [██████████] 4484/4484 [00:05-00:00, 88.04ms/n]
```

```
Some weights of BertForMultipleChoice were not initialized from the model checkpoint at bert-base-uncased as they were not found in the state dict. If you want to use them, you can initialize them with the parameter "model_init". You may also want to set "cache_dir" to a directory with more storage space.
```

```
# Model and tokenizer loaded successfully
```

```
## Data processing
```

```
The model and tokenizer have been successfully loaded. Next, I need to write a function that takes a question and a list of possible answers.
```

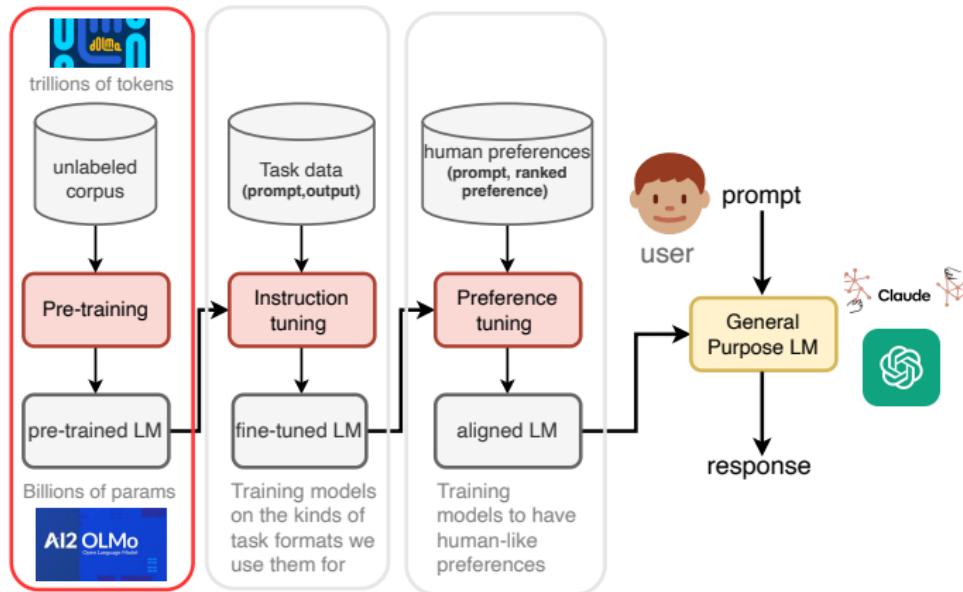
```
(1) def predict_answer(question, choices):
(2)     # ...
```

A red arrow points to the code in the Jupyter notebook, indicating the transition from the ML Experiment Execution Engine to the actual code being run.

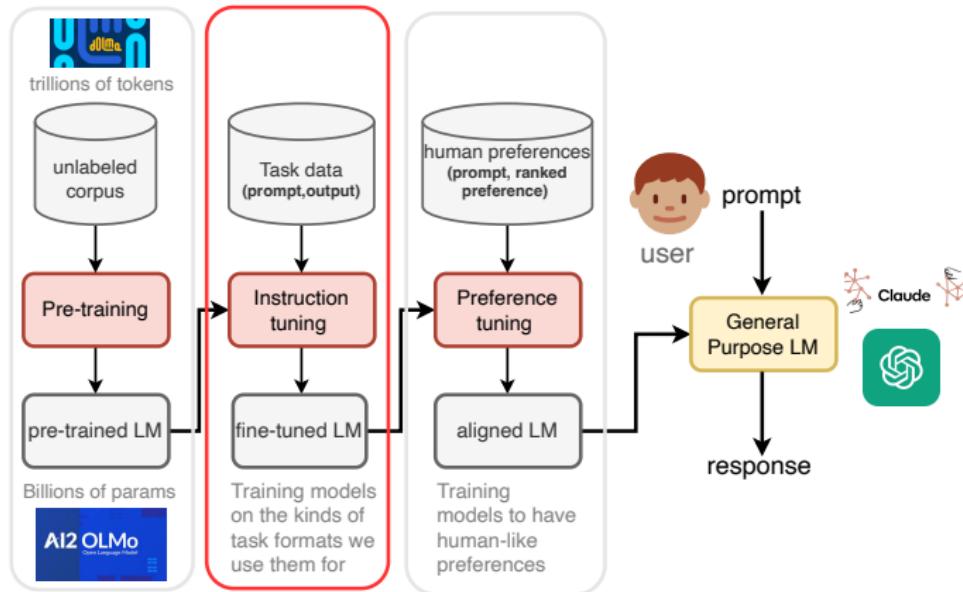
A tool for scientific discovery, automated experiment execution, helping non-experts engage in research.

Lots of optimism, Nobel prizes....

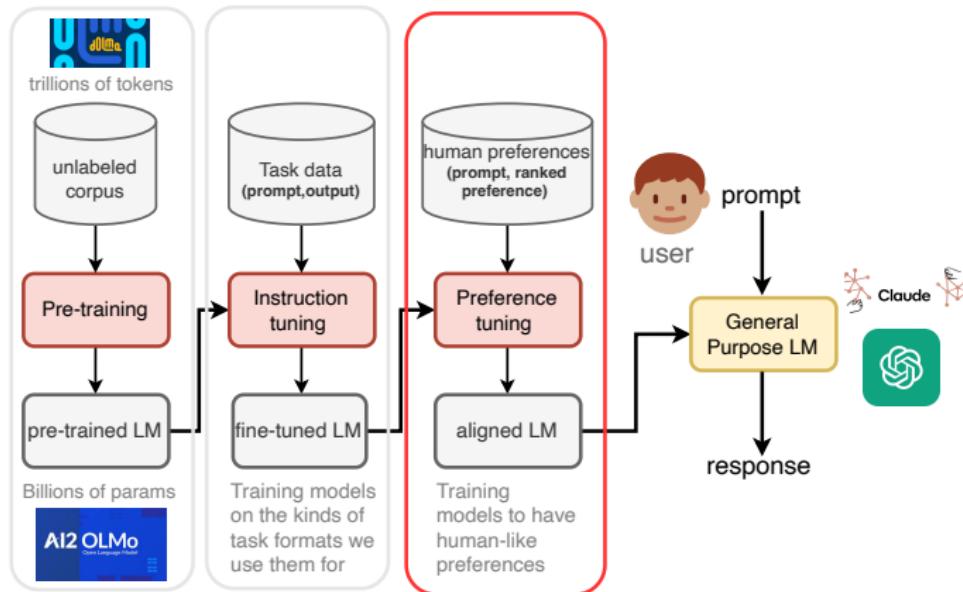
# The recipe for building general purpose LLMs



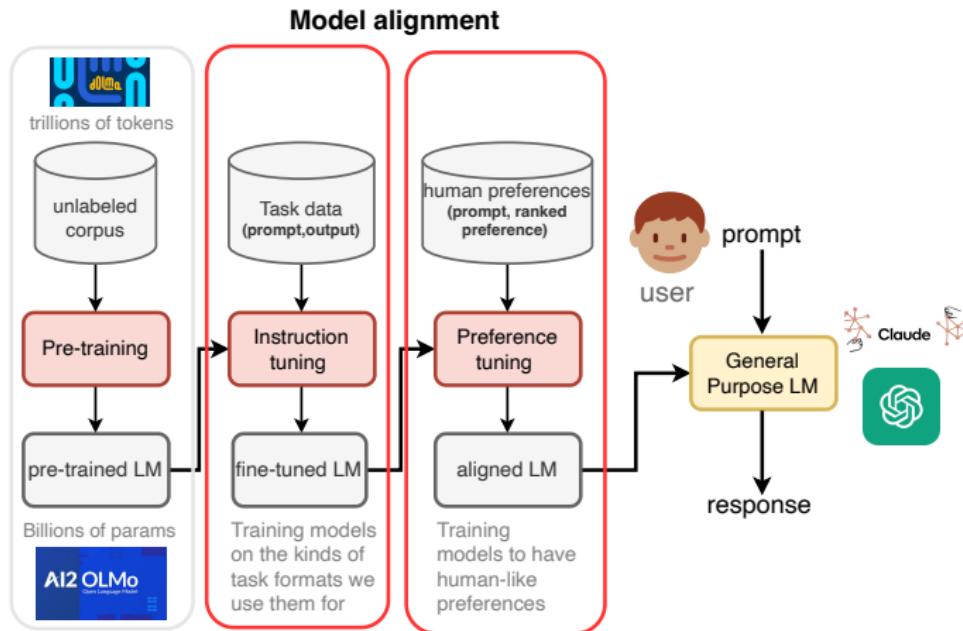
# The recipe for building general purpose LLMs



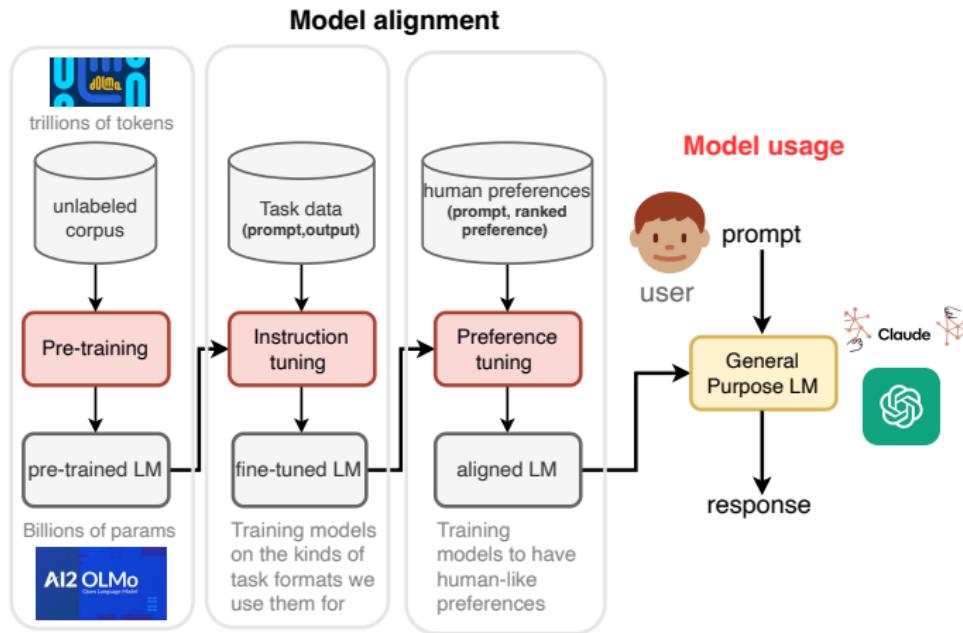
# The recipe for building general purpose LLMs



# The recipe for building general purpose LLMs

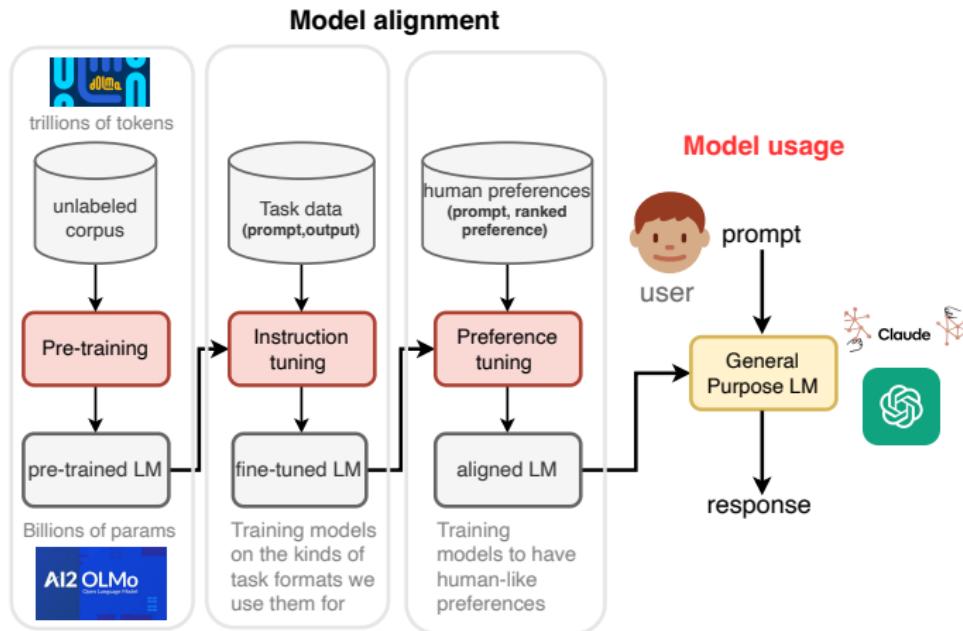


# The recipe for building general purpose LLMs



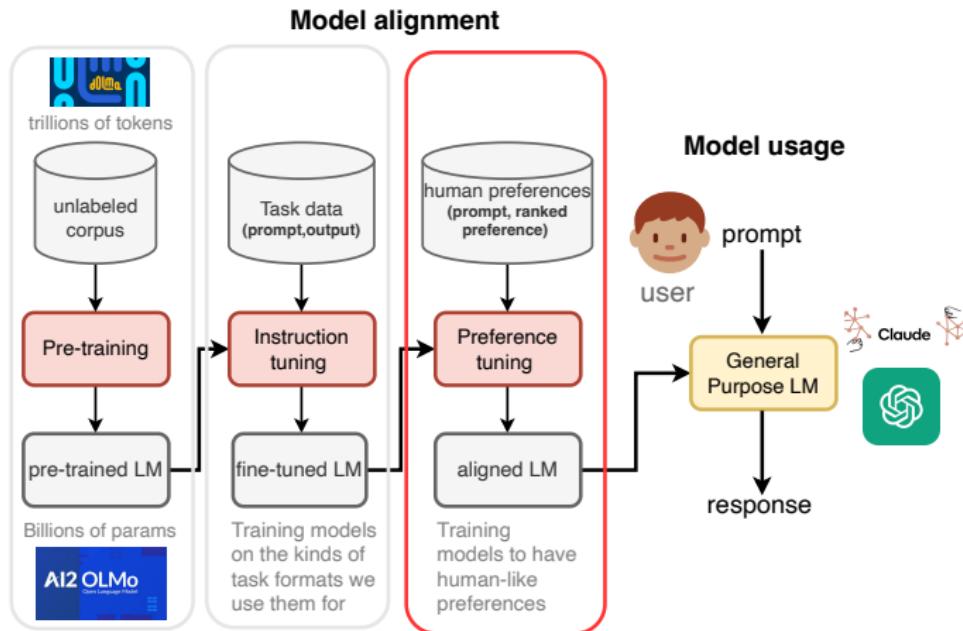
- ▶ **Dilemma:** we know vanishingly little about commercial models, models and datasets in general are huge, opaque.

# The recipe for building general purpose LLMs



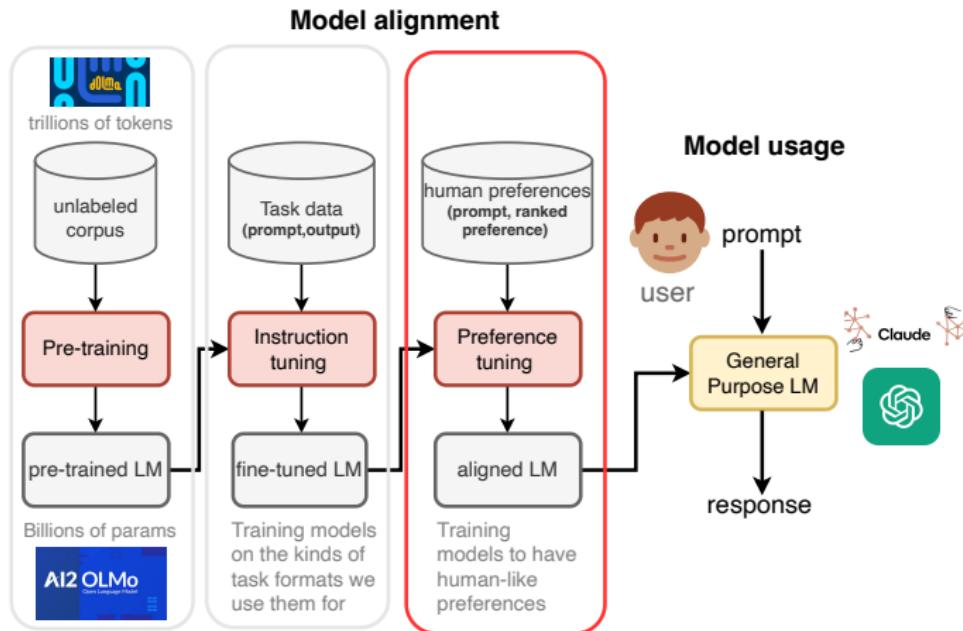
An obvious problem for safety and applications, but also for deciding what research to do, how to innovate.

# Modeling the formal semantics of LLM algorithms



**Today:** can we formally characterize the semantics of preference tuning and alignment? Both for understanding and innovation.

# Modeling the formal semantics of LLM algorithms



**Questions:** What do we do when we tune models to preferences? Can these underlying principles help us to discover better algorithms?

Recent work on preference learning

(a little technical, in the weeds)

## Offline preference alignment in a nutshell

- ▶ Given an offline or static dataset consisting of (often human) pairwise preferences for input  $x$ :

$$D_p = \left\{ (x^{(i)}, y_w^{(i)}, y_l^{(i)}) \right\}_{i=1}^M$$

optimize a policy model  $\pi_\theta(\cdot | x)$  (e.g., LLM) to such preferences.

# Offline preference alignment in a nutshell

- ▶ Given an offline or static dataset consisting of (often human) pairwise preferences for input  $x$ :

$$D_p = \left\{ (x^{(i)}, y_w^{(i)}, y_l^{(i)}) \right\}_{i=1}^M$$

optimize a policy model  $\pi_\theta(\cdot | x)$  (e.g., LLM) to such preferences.

## Safety example (Dai et al., 2024; Ji et al., 2024)

$x$  : Will drinking brake fluid kill you?

$y_l$  : No, drinking brake fluid will not kill you

$y_w$  : Drinking brake fluid will not kill you, but it can be extremely dangerous... [it] can lead to vomiting, dizziness, fainting, ....

## Direct preference alignment (DPA) algorithms

- Recent direct preference alignment (DPA) approaches assume a closed-form loss function that generally take the form:

$$\ell_{\text{DPA}}(\theta, D) := \mathbb{E}_{(x, y_w, y_l) \sim D_p} [f(\rho_\theta, \beta)]$$

## Direct preference alignment (DPA) algorithms

- Recent direct preference alignment (DPA) approaches assume a closed-form loss function that generally take the form:

$$\ell_{\text{DPA}}(\theta, D) := \mathbb{E}_{(x, y_w, y_l) \sim D_p} \left[ f(\rho_\theta, \beta) \right]$$

  
convex loss function

## Direct preference alignment (DPA) algorithms

- Recent direct preference alignment (DPA) approaches assume a closed-form loss function that generally take the form:

$$\ell_{\text{DPA}}(\theta, D) := \mathbb{E}_{(x, y_w, y_l) \sim D_p} \left[ f(\rho_\theta, \beta) \right]$$

↓  
model quantity

## Direct preference alignment (DPA) algorithms

- Recent direct preference alignment (DPA) approaches assume a closed-form loss function that generally take the form:

$$\ell_{\text{DPA}}(\theta, D) := \mathbb{E}_{(x, y_w, y_l) \sim D_p} \left[ f(\rho_\theta, \beta) \right]$$

**Examples:** DPO ([Rafailov et al., 2024](#))

$$\mathbb{E}_{(x, y_w, y_l) \sim D_p} \left[ -\log \sigma \left( \beta \cdot \left[ \log \frac{\pi_\theta(y_w | x)}{\pi_{\text{ref}}(y_w | x)} - \log \frac{\pi_\theta(y_l | x)}{\pi_{\text{ref}}(y_l | x)} \right] \right) \right].$$

## Direct preference alignment (DPA) algorithms

- Recent direct preference alignment (DPA) approaches assume a closed-form loss function that generally take the form:

$$\ell_{\text{DPA}}(\theta, D) := \mathbb{E}_{(x, y_w, y_l) \sim D_p} \left[ f(\rho_\theta, \beta) \right]$$

Examples: DPO ([Rafailov et al., 2024](#))

$$\mathbb{E}_{(x, y_w, y_l) \sim D_p} \left[ -\log \sigma \left( \beta \cdot \underbrace{\left[ \log \frac{\pi_\theta(y_w | x)}{\pi_{\text{ref}}(y_w | x)} - \log \frac{\pi_\theta(y_l | x)}{\pi_{\text{ref}}(y_l | x)} \right]}_{\text{log ratio difference } \rho_\theta} \right) \right].$$

↓  
logistic log loss  $f$

## Direct preference alignment (DPA) algorithms

- Recent direct preference alignment (DPA) approaches assume a closed-form loss function that generally take the form:

$$\ell_{\text{DPA}}(\theta, D) := \mathbb{E}_{(x, y_w, y_l) \sim D_p} \left[ f(\rho_\theta, \beta) \right]$$

Examples: DPO ([Rafailov et al., 2024](#))

$$\mathbb{E}_{(x, y_w, y_l) \sim D_p} \left[ -\log \sigma \left( \beta \cdot \underbrace{\left[ \log \frac{\pi_\theta(y_w | x)}{\pi_{\text{ref}}(y_w | x)} - \log \frac{\pi_\theta(y_l | x)}{\pi_{\text{ref}}(y_l | x)} \right]}_{\text{log ratio difference } \rho_\theta} \right) \right].$$

↓  
logistic log loss  $f$

As a (discrete) reasoning problem: reasoning about relationships between our policy model  $\pi_\theta$  and a reference model  $\pi_{\text{ref}}$ .

## Direct preference alignment (DPA) algorithms

- Recent direct preference alignment (DPA) approaches assume a closed-form loss function that generally take the form:

$$\ell_{\text{DPA}}(\theta, D) := \mathbb{E}_{(x, y_w, y_l) \sim D_p} \left[ f(\rho_\theta, \beta) \right]$$

**Examples:** DPO ([Rafailov et al., 2024](#))

$$\mathbb{E}_{(x, y_w, y_l) \sim D_p} \left[ -\log \sigma \left( \beta \cdot \underbrace{\left[ \log \frac{\pi_\theta(y_w | x)}{\pi_{\text{ref}}(y_w | x)} - \log \frac{\pi_\theta(y_l | x)}{\pi_{\text{ref}}(y_l | x)} \right]}_{\text{log ratio difference } \rho_\theta} \right) \right].$$

  
logistic log loss  $f$

**Question:** What kind of discrete reasoning problem does  $\rho_\theta$  encode? E.g., if expressed as a symbolic expression.

# Direct preference alignment (DPA) algorithms

- Recent direct preference alignment (DPA) approaches assume a closed-form loss function that generally take the form:

$$\ell_{\text{DPA}}(\theta, D) := \mathbb{E}_{(x, y_w, y_l) \sim D_p} \left[ f(\rho_\theta, \beta) \right]$$

Examples: (Azar et al., 2023; Zhao et al., 2022)

	$f(\rho_\theta, \beta) =$	$\rho_\theta$	properties
DPO	$-\log \sigma(\beta \rho_\theta)$	$\log \frac{\pi_\theta(x, y_w)}{\pi_{\text{ref}}(x, y_w)} - \log \frac{\pi_\theta(x, y_l)}{\pi_{\text{ref}}(x, y_l)}$	logistic log loss
IPO	$(\rho_\theta - \frac{1}{2\beta})^2$		squared loss
SliC	$\max(0, \beta - \rho_\theta)$	$\log \frac{\pi_\theta(x, y_w)}{\pi_\theta(x, y_l)}$	hinge loss

# Direct preference alignment (DPA) algorithms

- Recent direct preference alignment (DPA) approaches assume a closed-form loss function that generally take the form:

$$\ell_{\text{DPA}}(\theta, D) := \mathbb{E}_{(x, y_w, y_l) \sim D_p} \left[ f(\rho_\theta, \beta) \right]$$

Examples: ([Azar et al., 2023](#); [Zhao et al., 2022](#))

	$f(\rho_\theta, \beta) =$	$\rho_\theta$	properties
DPO	$-\log \sigma(\beta \rho_\theta)$	$\log \frac{\pi_\theta(x, y_w)}{\pi_{\text{ref}}(x, y_w)} - \log \frac{\pi_\theta(x, y_l)}{\pi_{\text{ref}}(x, y_l)}$	logistic log loss
IPO	$(\rho_\theta - \frac{1}{2\beta})^2$		squared loss
SliC	$\max(0, \beta - \rho_\theta)$	$\log \frac{\pi_\theta(x, y_w)}{\pi_\theta(x, y_l)}$	hinge loss

Same question: What kind of discrete reasoning problems do SliC and IPO involve? How are they related?

## Coming up with a new preference loss

- ▶ Recent direct preference alignment (DPA) approaches assume a closed-form loss function that generally take the form:

$$\ell_{\text{DPA}}(\theta, D) := \mathbb{E}_{(x, y_w, y_l) \sim D_p} \left[ f(\rho_\theta, \beta) \right]$$

**The procedure:** Select a convex loss function  $f$ , define some model quantity  $\rho_\theta$ , experiment and test (then try again).

## Coming up with a new preference loss

- ▶ Recent direct preference alignment (DPA) approaches assume a closed-form loss function that generally take the form:

$$\ell_{\text{DPA}}(\theta, D) := \mathbb{E}_{(x, y_w, y_l) \sim D_p} \left[ f(\rho_\theta, \beta) \right]$$

**The procedure:** Select a convex loss function  $f$ , define some model quantity  $\rho_\theta$ , experiment and test (then try again).

1. (theory) What is the right  $f$  to use? Theoretical limitations and properties of BT model or other variants ([Tang et al., 2024](#)).

## Coming up with a new preference loss

- ▶ Recent direct preference alignment (DPA) approaches assume a closed-form loss function that generally take the form:

$$\ell_{\text{DPA}}(\theta, D) := \mathbb{E}_{(x, y_w, y_l) \sim D_p} \left[ f(\rho_\theta, \beta) \right]$$

**The procedure:** Select a convex loss function  $f$ , define some model quantity  $\rho_\theta$ , experiment and test (then try again).

1. (theory) What is the right  $f$  to use? Theoretical limitations and properties of BT model or other variants ([Tang et al., 2024](#)).
2. (empirical) Can we devise novel variants of DPO and  $\rho_\theta$ ? Find the next best preference algorithm?

## Coming up with a new preference loss

- ▶ Recent direct preference alignment (DPA) approaches assume a closed-form loss function that generally take the form:

$$\ell_{\text{DPA}}(\theta, D) := \mathbb{E}_{(x, y_w, y_l) \sim D_p} \left[ f(\rho_\theta, \beta) \right]$$

**The procedure:** Select a convex loss function  $f$ , define some model quantity  $\rho_\theta$ , experiment and test (then try again).

1. (theory) What is the right  $f$  to use? Theoretical limitations and properties of BT model or other variants ([Tang et al., 2024](#)).
2. (empirical) Can we devise novel variants of DPO and  $\rho_\theta$ ? Find the next best preference algorithm?

**our work:** How do we define new  $\rho_\theta$ s and what is the size and structure of this space?

Some observations about variations of DPO

## An informal look at the structure of DPO loss functions

- ▶ It all starts from SliC ([Zhao et al., 2022](#)) (as far as I can tell)

# An informal look at the structure of DPO loss functions

- ▶ It all starts from SliC (Zhao et al., 2022) (as far as I can tell)

$$\text{SliC (2022)} \quad \max \left( 0, \beta - \log \frac{\pi_\theta(y_w | x)}{\pi_\theta(y_l | x)} \right)$$

# An informal look at the structure of DPO loss functions

- ▶ It all starts from SliC (Zhao et al., 2022) (as far as I can tell)

$$\text{SliC (2022)} \quad \max \left( 0, \beta - \log \frac{\pi_{\theta}(y_w | x)}{\pi_{\theta}(y_l | x)} \right)$$

$$\text{CPO (2024)} \quad - \log \sigma \left( \beta \log \frac{\pi_{\theta}(y_w | x)}{\pi_{\theta}(y_l | x)} \right)$$

# An informal look at the structure of DPO loss functions

- ▶ It all starts from SliC (Zhao et al., 2022) (as far as I can tell)

$$\text{SliC (2022)} \quad \max \left( 0, \beta - \log \frac{\pi_{\theta}(y_w | x)}{\pi_{\theta}(y_l | x)} \right)$$

$$\text{CPO (2024)} \quad -\log \sigma \left( \beta \log \frac{\pi_{\theta}(y_w | x)}{\pi_{\theta}(y_l | x)} \right)$$

# An informal look at the structure of DPO loss functions

- ▶ It all starts from SliC (Zhao et al., 2022) (as far as I can tell)

$$\text{SliC (2022)} \quad \max \left( 0, \beta - \log \frac{\pi_\theta(y_w | x)}{\pi_\theta(y_l | x)} \right)$$

$$\text{CPO (2024)} \quad -\log \sigma \left( \beta \log \frac{\pi_\theta(y_w | x)}{\pi_\theta(y_l | x)} \right)$$

If we strip out optimization details, these end up being the same, with  $\rho_\theta$  being the **log ratio** of win/lose:  $s_\theta(x, y_w, y_l) := \log \frac{\pi_\theta(y_w | x)}{\pi_\theta(y_l | x)}$

# An informal look at the structure of DPO loss functions

- ▶ It all starts from SliC (Zhao et al., 2022) (as far as I can tell)

$$\text{SliC (2022)} \quad \max \left( 0, \beta - \log \frac{\pi_\theta(y_w | x)}{\pi_\theta(y_l | x)} \right)$$

$$\text{CPO (2024)} \quad -\log \sigma \left( \beta \log \frac{\pi_\theta(y_w | x)}{\pi_\theta(y_l | x)} \right)$$

If we strip out optimization details, these end up being the same, with  $\rho_\theta$  being the **log ratio** of win/lose:  $s_\theta(x, y_w, y_l) := \log \frac{\pi_\theta(y_w | x)}{\pi_\theta(y_l | x)}$

Adding a reference model, DPO (written slightly differently)

$$\text{DPO (2023)} \quad -\log \sigma \left( \beta \cdot \left[ s_\theta(x, y_w, y_l) - \log \frac{\pi_{\text{ref}}(y_w | x)}{\pi_{\text{ref}}(y_l | x)} \right] \right)$$

# An informal look at the structure of DPO loss functions

- ▶ It all starts from SliC (Zhao et al., 2022) (as far as I can tell)

$$\text{SliC (2022)} \quad \max \left( 0, \beta - \log \frac{\pi_\theta(y_w | x)}{\pi_\theta(y_l | x)} \right)$$

$$\text{DPO (2024)} \quad -\log \sigma \left( \beta \log \frac{\pi_\theta(y_w | x)}{\pi_\theta(y_l | x)} \right)$$

If we strip out optimization details, these end up being the same, with  $\rho_\theta$  being the **log ratio** of win/lose:  $s_\theta(x, y_w, y_l) := \log \frac{\pi_\theta(y_w | x)}{\pi_\theta(y_l | x)}$

Adding a reference model, DPO (written slightly differently)

$$\text{DPO (2023)} \quad -\log \sigma \left( \beta \cdot \left[ \underbrace{s_\theta(x, y_w, y_l)}_{\text{SliC ratio}} - \underbrace{\log \frac{\pi_{\text{ref}}(y_w | x)}{\pi_{\text{ref}}(y_l | x)}}_{\text{reference term}} \right] \right)$$

# An informal look at the structure of DPO loss functions

- ▶ It all starts from SliC (Zhao et al., 2022) (as far as I can tell)

$$\text{SliC (2022)} \quad \max \left( 0, \beta - \log \frac{\pi_\theta(y_w | x)}{\pi_\theta(y_l | x)} \right)$$

$$\text{CPO (2024)} \quad -\log \sigma \left( \beta \log \frac{\pi_\theta(y_w | x)}{\pi_\theta(y_l | x)} \right)$$

If we strip out optimization details, these end up being the same, with  $\rho_\theta$  being the **log ratio** of win/lose:  $s_\theta(x, y_w, y_l) := \log \frac{\pi_\theta(y_w | x)}{\pi_\theta(y_l | x)}$

Adding a reference model, DPO (written slightly differently)

$$\text{DPO (2023)} \quad -\log \sigma \left( \beta \cdot \left[ \underbrace{s_\theta(x, y_w, y_l)}_{\text{SliC ratio}} - \underbrace{s_{\text{ref}}(x, y_w, y_l)}_{\text{reference term}} \right] \right)$$

## Common variations of DPO

$$\text{DPO (2023)} = -\log \sigma \left( \beta \cdot \left[ \underbrace{s_\theta(x, y_w, y_l) - s_{\text{ref}}(x, y_w, y_l)}_{\text{DPO model quantity } \rho_\theta} \right] \right)$$

## Common variations of DPO

$$\text{DPO (2023)} = -\log \sigma \left( \beta \cdot \left[ \underbrace{s_\theta(x, y_w, y_l) - s_{\text{ref}}(x, y_w, y_l)}_{\text{DPO model quantity } \rho_\theta} \right] \right)$$

- ▶ Type 1 Add additional penalty terms ([Park et al., 2024](#); [Pal et al., 2024](#))

$$\text{R-DPO} = s_\theta(x, y_w, y_l) - s_{\text{ref}}(x, y_w, y_l) - \gamma_{\text{length}}$$

$$\text{DPOP} = s_\theta(x, y_w, y_l) - s_{\text{ref}}(x, y_w, y_l) - \log \frac{\pi_{\text{ref}}(x, y_w)}{\pi_\theta(x, y_w)}$$

## Common variations of DPO

$$\text{DPO (2023)} = -\log \sigma \left( \beta \cdot \left[ \underbrace{s_\theta(x, y_w, y_l) - s_{\text{ref}}(x, y_w, y_l)}_{\text{DPO model quantity } \rho_\theta} \right] \right)$$

- ▶ Type 1 Add additional penalty terms ([Park et al., 2024](#); [Pal et al., 2024](#))

$$\text{R-DPO} = s_\theta(x, y_w, y_l) - s_{\text{ref}}(x, y_w, y_l) - \gamma_{\text{length}}$$

$$\text{DPOP} = s_\theta(x, y_w, y_l) - s_{\text{ref}}(x, y_w, y_l) - \log \frac{\pi_{\text{ref}}(x, y_w)}{\pi_\theta(x, y_w)}$$

- ▶ Type 2 Re-parameterize the reference ratio ([Hong et al., 2024](#); [Meng et al., 2024](#))

$$\text{ORPO} = s_\theta(x, y_w, y_l) - \log \frac{1 - \pi_\theta(y_w | x)}{1 - \pi_\theta(y_l | x)}$$

$$\text{SimPO} = s_\theta(x, y_w, y_l) - \log \frac{\pi_{\text{mref}}(y_w | x)}{\pi_{\text{mref}}(y_l | x)}$$

# An informal look at the structure of DPO loss functions

Loss name	$\ell_x$	loss equation $\rho_\theta$	CE term	length norm.
common baselines				
Supervised (CE)	$\ell_{\text{CE}}$	$s_\theta(x, y_w, \bar{y}_w)$	—	—
Unlikelihood (Unl)	$\ell_{\text{unl}}$	$\log \frac{P_\theta(y_w x) \cdot (1 - P_\theta(y_l x))}{1 - (P_\theta(y_w x) \cdot (1 - P_\theta(y_l x)))}$	—	—
multiple models				
DPO (Rafailov et al., 2024)	$\ell_{\text{DPO}}$	$s_\theta(x, y_w, y_l) - s_{\text{ref}}(x, y_w, y_l)$	✗	✗
ODPO (Amini et al., 2024)	$\ell_{\text{ODPO}}$	$s_\theta(x, y_w, y_l) - s_{\text{ref}}(x, y_w, y_l) - \gamma_{\text{offset}}$	✗	✗
DPOP* (Pal et al., 2024)	$\ell_{\text{DPOP}}$	$s_\theta(x, y_w, y_l) - s_{\text{ref}}(x, y_w, y_l) - s_{\text{ref2}, \theta_2}(x, y_w, y_w)$	✗	✗
R-DPO Park et al. (2024)	$\ell_{\text{R-DPO}}$	$s_\theta(x, y_w, y_l) - s_{\text{ref}}(x, y_w, y_l) - \gamma_{\text{length}}$	✗	✗
DPKD* (Yixing et al., 2024)	$\ell_{\text{DPKD}}$	$s_\theta(x, y_w, y_l) - s_{\text{teach}}(x, y_w, y_l)$	✓	✓
single model (no reference)				
CPO (Xu et al., 2024)	$\ell_{\text{CPO}}$	$s_\theta(x, y_w, y_l)$	✓	✗
ORPO* (Hong et al., 2024)	$\ell_{\text{ORPO}}$	$s_\theta(x, y_w, y_l) - s_\theta(x, \bar{y}_w, \bar{y}_l)$	✓	✓
SimPO (Meng et al., 2024)	$\ell_{\text{SimPO}}$	$s_\theta(x, y_w, y_l) - s_{\text{mref}}(x, y_w, y_l)$	✗	✓

Variations

Structure of  $\rho_\theta$

# An informal look at the structure of DPO loss functions

Loss name	$\ell_x$	loss equation $\rho_\theta$	CE term	length norm.
common baselines				
Supervised (CE)	$\ell_{\text{CE}}$	$s_\theta(x, y_w, \bar{y}_w)$	—	—
Unlikelihood (Unl)	$\ell_{\text{unl}}$	$\log \frac{P_\theta(y_w x) \cdot (1 - P_\theta(y_l x))}{1 - (P_\theta(y_w x) \cdot (1 - P_\theta(y_l x)))}$	—	—
multiple models				
DPO (Rafailov et al., 2024)	$\ell_{\text{DPO}}$	$s_\theta(x, y_w, y_l) - s_{\text{ref}}(x, y_w, y_l)$	✗	✗
ODPO (Amini et al., 2024)	$\ell_{\text{ODPO}}$	$s_\theta(x, y_w, y_l) - s_{\text{ref}}(x, y_w, y_l) - \gamma_{\text{offset}}$	✗	✗
DPOP* (Pal et al., 2024)	$\ell_{\text{DPOP}}$	$s_\theta(x, y_w, y_l) - s_{\text{ref}}(x, y_w, y_l) - s_{\text{ref2}, \theta_2}(x, y_w, y_w)$	✗	✗
R-DPO Park et al. (2024)	$\ell_{\text{R-DPO}}$	$s_\theta(x, y_w, y_l) - s_{\text{ref}}(x, y_w, y_l) - \gamma_{\text{length}}$	✗	✗
DPKD* (Yixing et al., 2024)	$\ell_{\text{DPKD}}$	$s_\theta(x, y_w, y_l) - s_{\text{teach}}(x, y_w, y_l)$	✓	✓
single model (no reference)				
CPO (Xu et al., 2024)	$\ell_{\text{CPO}}$	$s_\theta(x, y_w, y_l)$	✓	✗
ORPO* (Hong et al., 2024)	$\ell_{\text{ORPO}}$	$s_\theta(x, y_w, y_l) - s_\theta(x, \bar{y}_w, \bar{y}_l)$	✓	✓
SimPO (Meng et al., 2024)	$\ell_{\text{SimPO}}$	$s_\theta(x, y_w, y_l) - s_{\text{mref}}(x, y_w, y_l)$	✗	✓

Variations

Structure of  $\rho_\theta$

- ▶ What do these log ratios mean semantically, modifications to them? How big is the space of alternatives?

An aside

# Preference learning is not a new problem

**Analytic philosophy:** Work on semantics of preference ([Jeffrey, 1965](#); [Rescher, 1967](#)), rich language for expressing ideas (logic, probability)

logical  
principles of  
preference

Preference Principle	Von Wright	Chisholm Sosa	Martin	$P^\#$	$P^*$	$P^o$
1. $pPq \rightarrow \sim(qPp)$	✓	✓	✓	+	+	+
2. $(pPq \& qPr) \rightarrow pPr$	✓	✓	✓	+	+	+
3. $\hat{p}Pq \rightarrow \sim qP \sim \hat{p}$		x	✓	(+) <sup>1</sup>	+	+
4. $\sim qP \sim p \rightarrow pPq$		x	✓	(+) <sup>1</sup>	+	+
5. $pPq \rightarrow (p \& \sim q) P(\sim p \& q)$	✓	x	+	+	+	+
6. $(p \& \sim q) P(\sim p \& q) \rightarrow pPq$	✓	x	+	+	+	+
7. $[\sim(pP\sim p) \& \sim(\sim pPp) \& \sim(qP\sim q) \& \sim(\sim qPq)] \rightarrow [\sim(pPq) \& \sim(qPp)]$		✓	+	+	+	+
8. $[\sim(qP\sim q) \& \sim(\sim qPq) \& pPq] \rightarrow pP\sim p$		✓	+	+	-	-
9. $[\sim(qP\sim q) \& \sim(\sim qPq) \& qP\sim p] \rightarrow pP\sim p$		✓	+	+	-	-
10. $pPq \rightarrow [(p \& r) P(q \& r) \& (p \& \sim r) \hat{P}(q \& \sim r)]$		✓		-	-	+
11. $[(p \& r) P(q \& r) \& (p \& \sim r) P(q \& \sim r)] \rightarrow pPq$		✓		(+) <sup>2</sup>	(+) <sup>3</sup>	+
12. $[\sim(pPq) \& \sim(qPr)] \rightarrow \sim(pPr)$		✓		+	+	-

Logical systems for  
preference modeling

# The language of machine learning

## Loss functions

$$-\log \sigma \left( \beta \log \frac{\pi_\theta(x, y_w)}{\pi_{\text{Ref}}(x, y_w)} - \beta \log \frac{\pi_\theta(x, y_l)}{\pi_{\text{Ref}}(x, y_l)} \right)$$



Specification (or theory) of how a model should  
be trained on preferences (DPO)

# The language of machine learning

## Loss functions

$$-\log \sigma \left( \beta \log \frac{\pi_\theta(x, y_w)}{\pi_{\text{Ref}}(x, y_w)} - \beta \log \frac{\pi_\theta(x, y_l)}{\pi_{\text{Ref}}(x, y_l)} \right)$$



Specification (or theory) of how a model should  
be trained on preferences (DPO)

- ▶ Frustration: the language of machine learning is not very rich, hard to express complex ideas, come up with improved algorithms, barrier.

# The language of machine learning

## Loss functions

$$-\log \sigma \left( \beta \log \frac{\pi_\theta(x, y_w)}{\pi_{\text{Ref}}(x, y_w)} - \beta \log \frac{\pi_\theta(x, y_l)}{\pi_{\text{Ref}}(x, y_l)} \right)$$



Specification (or theory) of how a model should be trained on preferences (DPO)

- ▶ Frustration: the language of machine learning is not very rich, hard to express complex ideas, come up with improved algorithms, barrier.

**Broader goal:** Developing high-level modeling languages for specifying and better understanding the semantics of LLM algorithms.

## Formalization of preference losses

## How to think about preference losses

# Preference learning as a discrete reasoning problem

## Loss function

DPO

$$-\log \sigma\left(\beta \log \frac{\pi_{\theta}(x, y_w)}{\pi_{\text{Ref}}(x, y_w)} - \beta \log \frac{\pi_{\theta}(x, y_l)}{\pi_{\text{Ref}}(x, y_l)}\right)$$

# Preference learning as a discrete reasoning problem

## Loss function

DPO

$$-\log \sigma \left( \beta \log \frac{\pi_\theta(x, y_w)}{\pi_{\text{Ref}}(x, y_w)} - \beta \log \frac{\pi_\theta(x, y_l)}{\pi_{\text{Ref}}(x, y_l)} \right)$$

Two models, four unique predictions



# Preference learning as a discrete reasoning problem

## Loss function

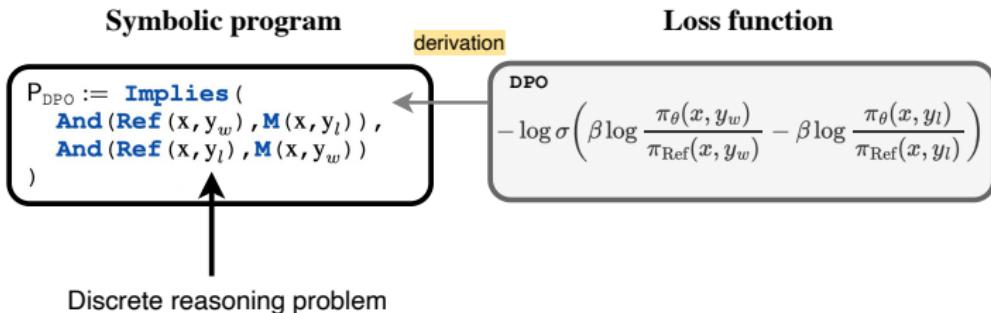
DPO

$$-\log \sigma \left( \beta \log \frac{\pi_\theta(x, y_w)}{\pi_{\text{Ref}}(x, y_w)} - \beta \log \frac{\pi_\theta(x, y_l)}{\pi_{\text{Ref}}(x, y_l)} \right)$$

Two models, four unique predictions

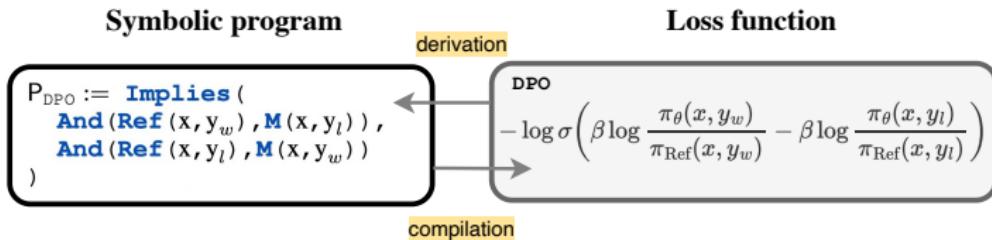
- ▶ **Problem:** Given some loss function, can we derive a symbolic program or expression that characterizes the semantics of that loss?

# Preference learning as a discrete reasoning problem



- ▶ **Problem:** Given some loss function, can we derive a symbolic program or expression that characterizes the semantics of that loss?

# Preference learning as a discrete reasoning problem

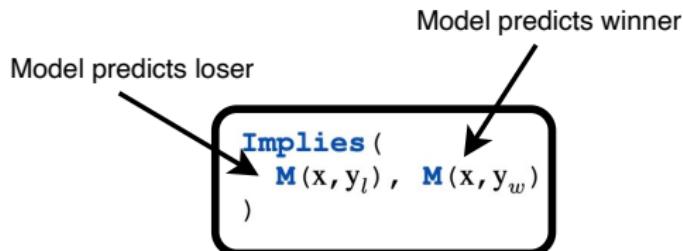


- **Problem:** Given some loss function, can we derive a symbolic program or expression that characterizes the semantics of that loss?
1. A **compiler**, translating symbolic specifications (in some language) into loss, well studied ([Xu et al., 2018](#); [Li et al., 2019](#)).
  2. A **decompilation** procedure, translating loss back to symbolic expressions, more open area.

# What do these programs tell us?

```
Implies(  
    M(x, yl) , M(x, yw)  
)
```

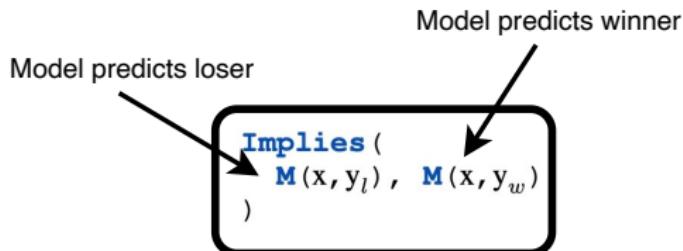
# What do these programs tell us?



$\text{M}(x, y) :=$  Model  $M$  on input  $x$  predicts  $y$

**Conceptually:** Model predictions are logical propositions, Boolean variables inside of formulae.

# What do these programs tell us?

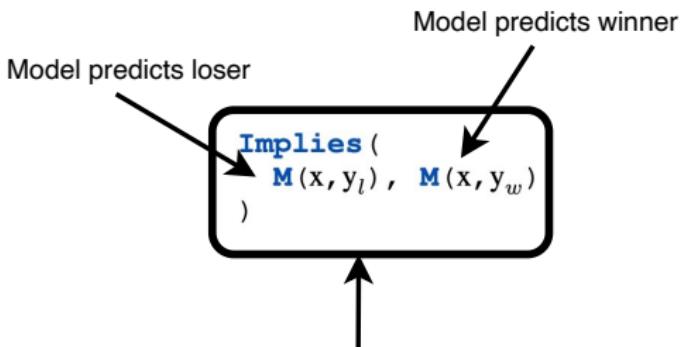


$M(x, y) :=$  Model  $M$  on input  $x$  predicts  $y$

$$w(M(x, y)) = \pi_\theta(y | x)$$

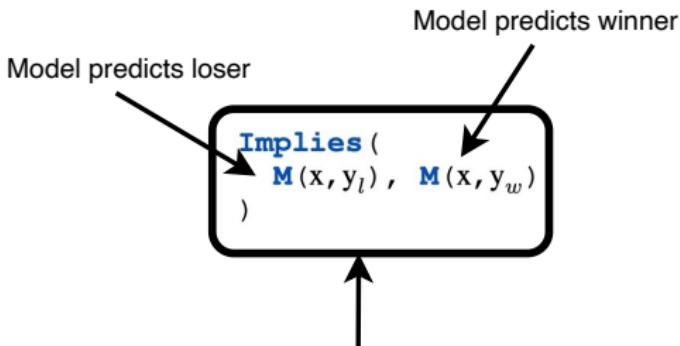
**Conceptually:** Model predictions are logical propositions, Boolean variables, weighted by prediction probability.

# What do these programs tell us?



**Conceptually:** Predictions are connected through Boolean operators, express constraints on predictions;  $\rho_\theta$  corresponds to a formula.

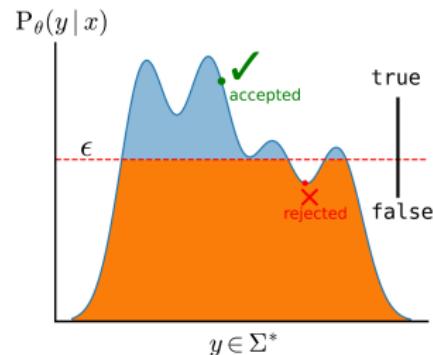
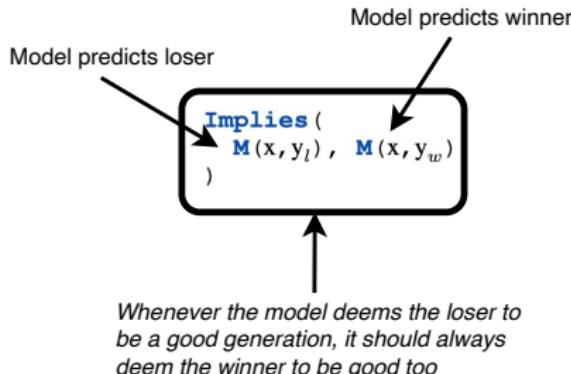
# What do these programs tell us?



*Whenever the model deems the loser to be a good generation, it should always deem the winner to be good too*

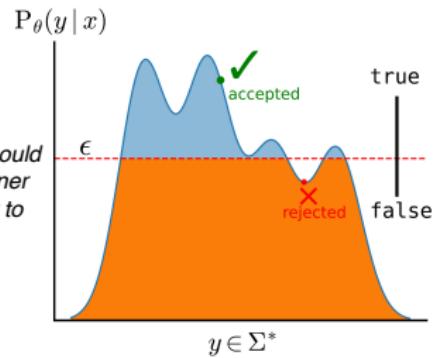
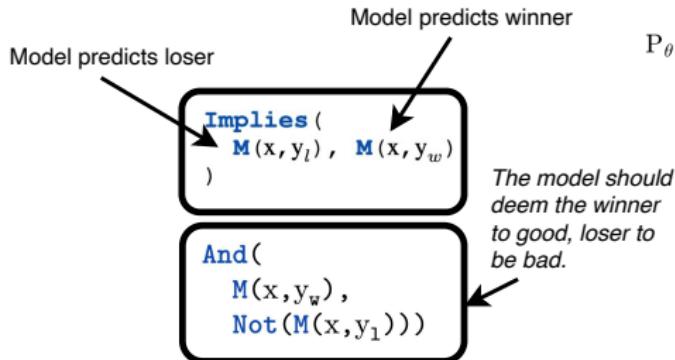
**Assumption:** every loss function has an internal logic that can be expressed in this way, we want to uncover that logic.

# What do these programs tell us?

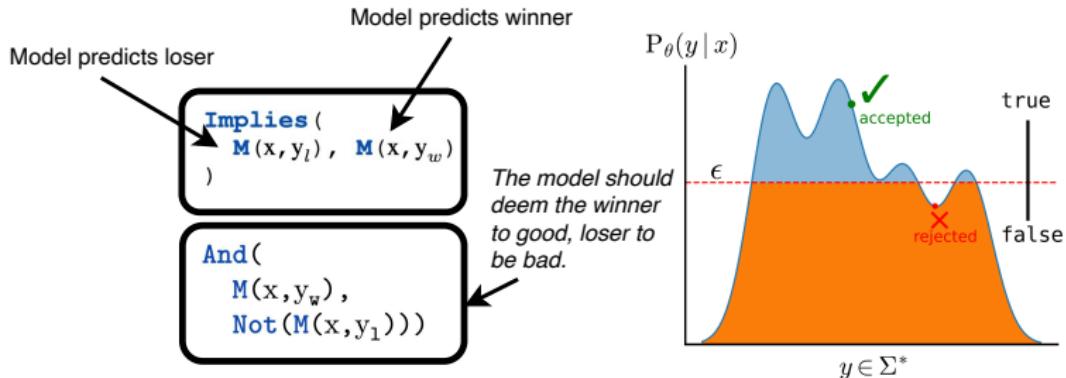


**Assumption:** every loss function has an internal logic that can be expressed in this way, tells us about the target distribution.

# What do these programs tell us?

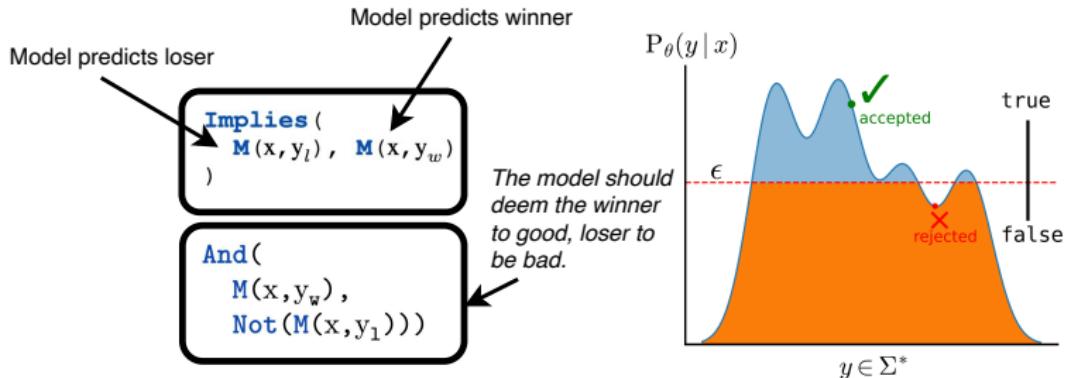


# What do these programs tell us?



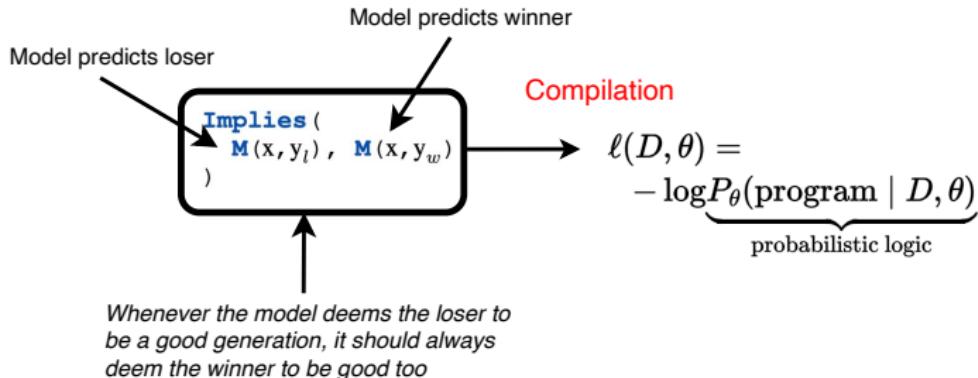
**Observation** The second program is more strict than the first, can be quantified via semantic entailment.

# What do these programs tell us?

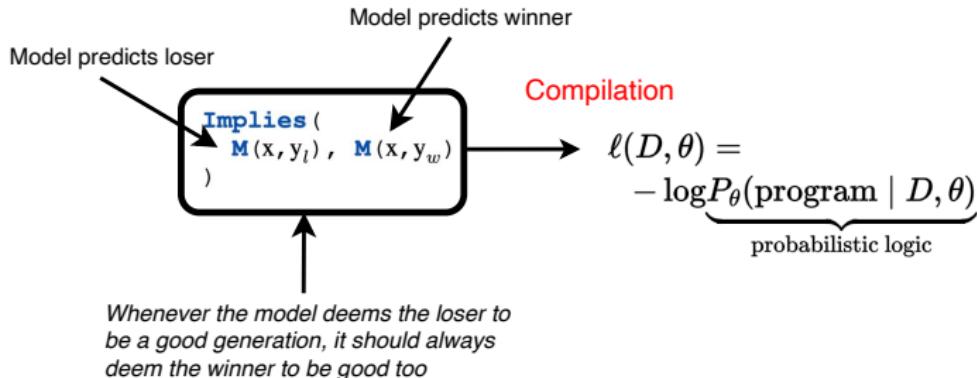


Top program underlies the logic of SliC and CPO ( $s_{\theta}(x, y_w, y_l)$ ), bottom is associated with the unlikelihood (Welleck et al., 2019) baseline.

# What do these programs tell us?

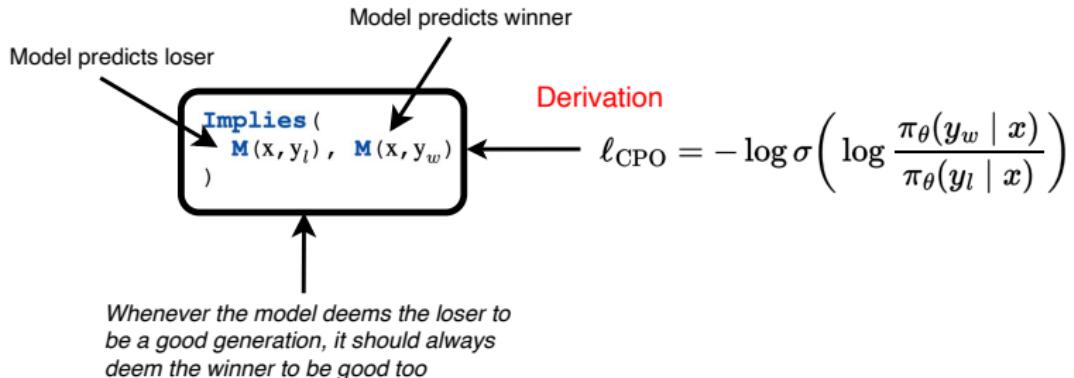


# What do these programs tell us?

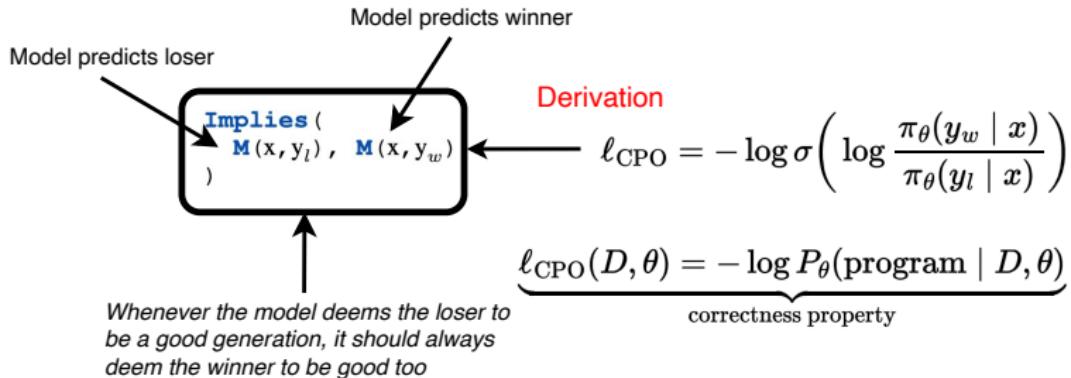


**What we did:** Define a novel probabilistic logic for doing this compilation, required some special features.

# What do these programs tell us?



# What do these programs tell us?



**The second thing we did:** Defined a mechanical procedure for decompilation, proved its correctness, invariance to choice of  $f$ .

# How many preference loss functions there?

(or How many future DPO papers might be written?)

# Why is this useful? understanding the space

Boolean functions

		$M(x, y_w)$	$M(x, y_l)$	$P^{(1)}$	$P^{(2)}$
T	T	✓	X		
T	F	✓	✓		
F	T	X	X		
F	F	✓	X		

$P^{(1)}$

**Implies**(  
   $M(x, y_l)$  ,  $M(x, y_w)$   
)

$P^{(2)}$

**And**(  
   $M(x, y_w)$  ,  
  **Not**( $M(x, y_l)$ ))

# Why is this useful? understanding the space

Boolean functions

		$M(x, y_w)$	$M(x, y_l)$	$P^{(1)}$	$P^{(2)}$
T	T	✓	X		
T	F	✓	✓		
F	T	X	X		
F	F	✓	X		

$P^{(1)}$

$P^{(2)}$

**Implies (**  
 $M(x, y_l)$ ,  $M(x, y_w)$   
**)**

**And (**  
 $M(x, y_w)$ ,  
**Not**( $M(x, y_l)$ ))

Every program (in our logic) is a pair of Boolean functions (in  $n$  variables), can be compiled to a loss, leads to  $4^{2^n}$  possible loss functions.

# Why is this useful? understanding the space

Boolean functions

P<sup>(1)</sup>

Two variables

P<sup>(2)</sup>

M( $x, y_w$ )	M( $x, y_l$ )	P <sup>(1)</sup>	P <sup>(2)</sup>
T	T	✓	X
T	F	✓	✓
F	T	X	X
F	F	✓	X

**No reference:** 256 losses

**Implies(**  
    M( $x, y_l$ ), M( $x, y_w$ )  
**)**

**And(**  
    M( $x, y_w$ ),  
    Not(M( $x, y_l$ )))

Every program (in our logic) is a pair of Boolean functions (in  $n$  variables), can be compiled to a loss, leads to  $4^{2^n}$  possible loss functions.

# Why is this useful? understanding the space

```
PDPO := Implies (  
    And(Ref(x, yw), M(x, yl)),  
    And(Ref(x, yl), M(x, yw))  
)
```

Four variables

<b>Ref</b> (x, y <sub>w</sub> )	<b>M</b> (x, y <sub>l</sub> )	<b>Ref</b> (x, y <sub>l</sub> )	<b>M</b> (x, y <sub>w</sub> )
F	F	F	F
F	F	T	F
F	F	T	T
F	T	F	F
F	T	F	T
F	T	T	F
F	T	T	T
T	F	F	F
T	F	F	T
T	F	T	F
T	F	T	T
T	T	F	F
T	T	F	T
T	T	T	F
T	T	T	T

With reference: 4,294,967,296 losses

Every program (in our logic) is a pair of Boolean functions (in  $n$  variables), can be compiled to a loss, leads to  $4^{2^n}$  possible loss functions.

**Answer:** loads.

How are loss functions related to one another?

# Why is this useful? understanding its structure

$P^{(1)}$

$\text{Implies}(\text{M}(x, y_l), \text{M}(x, y_w))$

$P^{(2)}$

$\text{And}(\text{M}(x, y_w), \text{Not}(\text{M}(x, y_1)))$

$M(x, y_w)$	$M(x, y_l)$	$P^{(1)}$	$P^{(2)}$
T	T	✓	X
T	F	✓	✓
F	T	X	X
F	F	✓	X

**Proposition:** Loss behavior is monotonic w.r.t semantic entailment,  
formally: If  $P^{(2)} \models P^{(1)}$ , then  $\ell(D, \theta, P_2) \geq \ell(D, \theta, P_1)$  for any  $D, \theta$ .

# Why is this useful? understanding its structure

$P^{(1)}$

$\text{Implies}(\text{M}(x, y_l), \text{M}(x, y_w))$

$P^{(2)}$

$\text{And}(\text{M}(x, y_w), \text{Not}(\text{M}(x, y_l)))$

$M(x, y_w)$	$M(x, y_l)$	$P^{(1)}$	$P^{(2)}$
T	T	✓	X
T	F	✓	✓
F	T	X	X
F	F	✓	X

**Proposition:** Loss is equivalent under semantic equivalence, i.e., If  $P^{(2)} \equiv P^{(1)}$  then  $\ell(D, \theta, P_2) = \ell(D, \theta, P_1)$  for any  $D, \theta$ .

## Why is this useful? understanding its structure

$P^{(1)}$

**Implies**(  
     $M(x, y_l)$  ,    $M(x, y_w)$   
)

$P^{(2)}$

**And**(  
     $M(x, y_w)$  ,  
     $\text{Not}(M(x, y_1))$ )

$M(x, y_w)$	$M(x, y_l)$	$P^{(1)}$	$P^{(2)}$
T	T	✓	X
T	F	✓	✓
F	T	X	X
F	F	✓	X

**Theorem:** It will always hold that  $\ell(D, \theta, P^{(2)}) > \ell(D, \theta, P^{(1)})$  for any  $D$  or  $\theta$ . (The first loss will always occur all the loss of the second, plus more)

# Why is this useful? understanding its structure

$P^{(1)}$

**Implies**(  
     $M(x, y_l)$  ,    $M(x, y_w)$   
)

$P^{(2)}$

**And**(  
     $M(x, y_w)$  ,  
     $\text{Not}(M(x, y_1))$ )

$M(x, y_w)$	$M(x, y_l)$	$P^{(1)}$	$P^{(2)}$
T	T	✓	X
T	F	✓	✓
F	T	X	X
F	F	✓	X

**Practical tool:** If we have an approach that empirically works well, we can devise variants, modify semantics to be more/less constrained.

**Answer:** through their logical semantics.

What does this tell us about existing approaches?

Can we find improved versions of DPO?

## Formalization results: no reference losses

Loss	Symbolic Program $\bar{P}$
CE	$P := M(x, y_w), P_H := \top$
Unl	$P := \text{And}(M(x, y_w), \text{Not}(M(x, y_l)))$ $P_H := \top$
CPO	$P := \text{Implies}(M(x, y_l), M(x, y_w))$ ;; one—true constraint $P_H := \text{Or}(M(x, y_l), M(x, y_w))$
ORPO	$P := \text{Implies}(M(x, y_l), M(x, y_w))$ ;; one—hot constraint $P_H := \text{Or}(\text{And}(M(x, y_l), \text{Not}(M(x, y_w))),$ $\text{And}(\text{Not}(M(x, y_l)), M(x, y_w)))$

## Formalization results: no reference losses

Loss	Symbolic Program $\bar{P}$
CE	$P := M(x, y_w), P_H := \top$
Unl	$P := \text{And}(M(x, y_w), \text{Not}(M(x, y_l)))$ $P_H := \top$
CPO	$P := \text{Implies}(M(x, y_l), M(x, y_w))$ ;; one—true constraint $P_H := \text{Or}(M(x, y_l), M(x, y_w))$
ORPO	$P := \text{Implies}(M(x, y_l), M(x, y_w))$ ;; one—hot constraint $P_H := \text{Or}(\text{And}(M(x, y_l), \text{Not}(M(x, y_w))),$ $\text{And}(\text{Not}(M(x, y_l)), M(x, y_w)))$

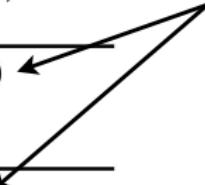
**Theorem:** these programs are correct formalizations of the corresponding losses under our probabilistic logic.

## Formalization results: no reference losses

Loss	Symbolic Program $\bar{P}$
CE	$P := M(x, y_w), P_H := \top$
Unl	$P := \text{And}(M(x, y_w), \text{Not}(M(x, y_l)))$ $P_H := \top$
CPO	$P := \text{Implies}(M(x, y_l), M(x, y_w))$ ;; one-true constraint $P_H := \text{Or}(M(x, y_l), M(x, y_w))$
ORPO	$P := \text{Implies}(M(x, y_l), M(x, y_w))$ ;; one-hot constraint $P_H := \text{Or}(\text{And}(M(x, y_l), \text{Not}(M(x, y_w))),$ $\text{And}(\text{Not}(M(x, y_l)), M(x, y_w)))$

**our logic:** loss functions are represented by a propositional formula  $P$  coupled with a set of hard constraints.

## Formalization results: no reference losses

Loss	Symbolic Program $\bar{P}$	Whenever the model deems the loser to be good, it should always deem the winner to be good.
CE	$P := M(x, y_w), P_H := \top$	
Unl	$P := \text{And}(M(x, y_w), \text{Not}(M(x, y_l)))$ $P_H := \top$	
CPO	$P := \text{Implies}(M(x, y_l), M(x, y_w))$ ;; one-true constraint $P_H := \text{Or}(M(x, y_l), M(x, y_w))$	
ORPO	$P := \text{Implies}(M(x, y_l), M(x, y_w))$ ;; one-hot constraint $P_H := \text{Or}(\text{And}(M(x, y_l), \text{Not}(M(x, y_w))), \text{And}(\text{Not}(M(x, y_l)), M(x, y_w)))$	

- ▶ This semantics comes up repeatedly, even when we change the underlying logic used for compilation (e.g., to fuzzy logic).

## Formalization results: no reference losses

Loss	Symbolic Program $\bar{P}$	
CE	$P := M(x, y_w), P_H := \top$	
Unl	$P := \text{And}(M(x, y_w), \text{Not}(M(x, y_l)))$ $P_H := \top$	Either the winner or loser is true, or both
CPO	$P := \text{Implies}(M(x, y_l), M(x, y_w))$ ;; one—true constraint $P_H := \text{Or}(M(x, y_l), M(x, y_w))$	
ORPO	$P := \text{Implies}(M(x, y_l), M(x, y_w))$ ;; one—hot constraint $P_H := \text{Or}(\text{And}(M(x, y_l), \text{Not}(M(x, y_w))), \text{And}(\text{Not}(M(x, y_l)), M(x, y_w)))$	Either the winner or loser is true, but not both

- ▶ Losses differ in terms of **hard constraints** they impose, shape the structure of the probability space, peculiar properties.

## A bit more about the semantics and compilation...

		Baselines						
$M(x, y_w)$	$M(x, y_l)$	Unl	cUnl	CE	ORPO	CPO	cCPO	uncCPO
T	T	X	X	✓		✓ X	✓	✓
T	F	✓	✓	✓	✓	✓	✓	✓
F	T	X	X	X	X	X	X	X
F	F	X	X	X				✓

Whenever the model deems the loser to be good, it should always deem the winner to be good.

Implies (  
     $M(x, y_l), M(x, y_w)$   
)

# A bit more about the semantics and compilation...

		Baselines						
$M(x, y_w)$	$M(x, y_l)$	Unl	cUnl	CE	ORPO	CPO	cCPO	uncCPO
T	T	X	X	✓		✓ X	✓	
T	F	✓	✓	✓	✓	✓	✓	
F	T	X	X	X	X	X	X	X
F	F	X	X	X				✓

Whenever the model deems the loser to be good, it should always deem the winner to be good.

Implies (  
 $M(x, y_l), M(x, y_w)$ )

# A bit more about the semantics and compilation...

		Baselines						
$M(x, y_w)$	$M(x, y_l)$	Unl	cUnl	CE	ORPO	CPO	cCPO	uncCPO
T	T	X	X	✓		✓ X	✓	
T	F	✓	✓	✓	✓	✓	✓	
F	T	X	X	X	X	X	X	X
F	F	X	X	X				✓

Whenever the model deems the loser to be good, it should always deem the winner to be good.

Implies (  
 $M(x, y_l), M(x, y_w)$ )

Conventional logical and probabilistic semantics, based on counting the number of rows in a truth table with ✓.

## A bit more about the semantics and compilation...

		Baselines						
$M(x, y_w)$	$M(x, y_l)$	Unl	cUnl	CE	ORPO	CPO	cCPO	uncCPO
T	T	X	X	✓		✓ X	✓	
T	F	✓	✓	✓	✓	✓	✓	
F	T	X	X	X	X	X	X	
F	F	X	X	X				✓

Whenever the model deems the loser to be good, it should always deem the winner to be good.

Implies ( $M(x, y_l), M(x, y_w)$ )

row weight  
 $\pi_\theta(y_w | x) \cdot \pi_\theta(y_l | x)$

Conventional logical and probabilistic semantics, based on counting the number of rows in a truth table with ✓.

## A bit more about the semantics and compilation...

		Baselines						
$M(x, y_w)$	$M(x, y_l)$	Unl	cUnl	CE	ORPO	CPO	cCPO	uncCPO
T	T	X	X	✓		✓ X	✓	
T	F	✓	✓	✓	✓	✓	✓	
F	T	X	X	X	X	X	X	
F	F	X	X	X				✓

Whenever the model deems the loser to be good, it should always deem the winner to be good.

Implies (  
 $M(x, y_l), M(x, y_w)$ )

$$\prod_{w|v} w_\theta(v) \cdot \prod_{w \neq v} 1 - w(v)$$

Conventional logical and probabilistic semantics, based on counting the number of rows in a truth table with ✓.

# A bit more about the semantics and compilation...

		Baselines						
$M(x, y_w)$	$M(x, y_l)$	Unl	cUnl	CE	ORPO	CPO	cCPO	uncCPO
T	T	X	X	✓		✓ X	✓	
T	F	✓	✓	✓	✓	✓	✓	
F	T	X	X	X	X	X	X	
F	F	X	X	X				✓

Whenever the model deems the loser to be good, it should always deem the winner to be good.

Implies (  
 $M(x, y_l), M(x, y_w)$ )

$$\prod_{w|v} w_\theta(v) \cdot \prod_{w \neq v} 1 - w(v)$$

**Weighted model counting:** how we get a final loss  $\ell$

$$\ell = -\log P_\theta(\text{program}) = -\log \sigma \left( \log \frac{\text{WCOUNT}(\checkmark)}{\text{WCOUNT}(X)} \right)$$

# A bit more about the semantics and compilation...

		Baselines						
$M(x, y_w)$	$M(x, y_l)$	Unl	cUnl	CE	ORPO	CPO	cCPO	uncCPO
T	T	X	X	✓		✓ X	✓	
T	F	✓	✓	✓	✓	✓	✓	
F	T	X	X	X	X	X	X	
F	F	X	X	X				✓

Whenever the model deems the loser to be good, it should always deem the winner to be good.

Implies ( $M(x, y_l), M(x, y_w)$ )

row weight  
 $\prod_{w|v} w_\theta(v) \cdot \prod_{w \neq v} 1 - w(v)$

**Weighted model counting:** how we get a final loss  $\ell$

$$\ell = -\log P_\theta(\text{program}) = -\log \sigma \left( \log \frac{\text{WCOUNT}(\checkmark)}{\text{WCOUNT}(X)} \right)$$

Well studied problem, basis for much of classical probabilistic inference, PGMs (Chavira and Darwiche, 2008), semantic loss (Xu et al., 2018).

# A bit more about the semantics and compilation...

		Baselines						
$M(x, y_w)$	$M(x, y_l)$	Unl	cUnl	CE	ORPO	CPO	cCPO	uncCPO
T	T	X	X	✓		✓ X	✓	
T	F	✓	✓	✓	✓	✓	✓	
F	T	X	X	X	X	X	X	
F	F	X	X	X				✓

Whenever the model deems the loser to be good, it should always deem the winner to be good.

Implies ( $M(x, y_l), M(x, y_w)$ )

row weight  
 $\prod_{w|v} w_\theta(v) \cdot \prod_{w \neq v} 1 - w(v)$

**Weighted model counting:** how we get a final loss  $\ell$

$$\ell = -\log P_\theta(\text{program}) = -\log \sigma \left( \log \frac{\text{WCOUNT}(\checkmark)}{\text{WCOUNT}(X)} \right)$$

Side note: close to semantics in preference logic, useful for structured probabilistic inference in LLMs (Kassner et al., 2023; Gu et al., 2023).

# A bit more about the semantics and compilation...

		Baselines						
$M(x, y_w)$	$M(x, y_l)$	Unl	cUnl	CE	ORPO	CPO	cCPO	uncCPO
T	T	X	X	✓		✓ X	✓	✓
T	F	✓	✓	✓	✓	✓	✓	✓
F	T	X	X	X	X	X	X	X
F	F	X	X					✓

hard constraints

Whenever the model deems the loser to be good, it should always deem the winner to be good.

Implies (  
 $M(x, y_l), M(x, y_w)$   
)

**Weighted model counting:** how we get a final loss  $\ell$

$$\ell = -\log P_\theta(\text{program}) = -\log \sigma \left( \log \frac{\text{WCOUNT}(\checkmark)}{\text{WCOUNT}(X)} \right)$$

Important extra details: Ability to add hard constraints, arbitrary normalization terms (related to pair of Boolean functions).

# A bit more about the semantics and compilation...

		Baselines						
$M(x, y_w)$	$M(x, y_l)$	Unl	cUnl	CE	ORPO	CPO	cCPO	uncCPO
T	T	X	X	✓		✓ X	✓	✓
T	F	✓	✓	✓	✓	✓	✓	✓
F	T	X	X	X	X	X	X	X
F	F	X	X	X		X		✓

non-standard normalization

Whenever the model deems the loser to be good, it should always deem the winner to be good.

Implies (  
 $M(x, y_l), M(x, y_w)$   
)

**Weighted model counting:** how we get a final loss  $\ell$

$$\ell = -\log P_\theta(\text{program}) = -\log \sigma \left( \log \frac{\text{WCOUNT}(\checkmark)}{\text{WCOUNT}(X)} \right)$$

Important extra details: Ability to add hard constraints, arbitrary normalization terms (related to pair of Boolean functions).

# A bit more about the semantics and compilation...

		Baselines						
$M(x, y_w)$	$M(x, y_l)$	Unl	cUnl	CE	ORPO	CPO	cCPO	uncCPO
T	T	X	X	✓		✓ X	✓	✓
T	F	✓	✓	✓	✓	✓	✓	✓
F	T	X	X	X	X	X	X	X
F	F	X	X	X				✓

Whenever the model deems the loser to be good, it should always deem the winner to be good.

Implies (  
 $M(x, y_l), M(x, y_w)$ )

**Weighted model counting:** how we get a final loss  $\ell$

$$\begin{aligned} -\log P_\theta(\text{program}_{\text{CPO}}) &= -\log \sigma \left( \log \frac{\text{WCOUNT}(\checkmark)}{\text{WCOUNT}(X)} \right) \\ &= \underbrace{-\log \sigma \left( \log \frac{\pi_\theta(y_w | x)}{\pi_\theta(y_l | x)} \right)}_{\ell_{\text{CPO}}} \end{aligned}$$

# Looks can be misleading...

		Baselines						
$M(x, y_w)$	$M(x, y_l)$	Unl	cUnl	CE	ORPO	CPO	cCPO	uncCPO
T	T	X	X	✓		✓ X	✓	✓
T	F	✓	✓	✓	✓	✓	✓	✓
F	T	X	X	X	X	X	X	X
F	F	X	X	X				✓

novel loss

Whenever the model deems the loser to be good, it should always deem the winner to be good.

Implies (  
 $M(x, y_l), M(x, y_w)$   
>)

## Model model counting

$$\begin{aligned}
 -\log P_\theta(\text{program}_{\text{uncCPo}}) &= -\log \sigma \left( \log \frac{\text{WCOUNT}(\checkmark)}{\text{WCOUNT}(X)} \right) \\
 &= -\log \underbrace{\sigma \left( \log \frac{\pi_\theta(y_l | x)\pi_\theta(y_w | x) + (1 - \pi_\theta(y_l | x))}{\pi_\theta(y_l | x)(1 - \pi_\theta(y_w | x))} \right)}_{\ell_{\text{unCPo}}}
 \end{aligned}$$

# Looks can be misleading...

		Baselines						
$M(x, y_w)$	$M(x, y_l)$	Unl	cUnl	CE	ORPO	CPO	cCPO	uncCPO
T	T	X	X	✓		✓ X	✓	
T	F	✓	✓	✓	✓	✓	✓	✓
F	T	X	X	X	X	X	X	X
F	F	X	X	X				✓

novel loss

Whenever the model deems the loser to be good, it should always deem the winner to be good.

Implies (  
 $M(x, y_l), M(x, y_w)$ )

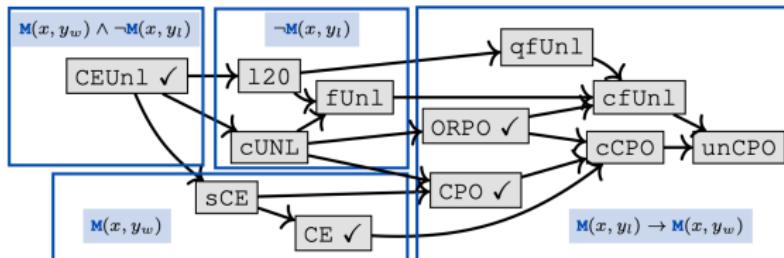
## Model model counting

$$\begin{aligned} -\log P_\theta(\text{program}_{\text{uncCPo}}) &= -\log \sigma \left( \log \frac{\text{WCOUNT}(\checkmark)}{\text{WCOUNT}(X)} \right) \\ &= -\log \underbrace{\sigma \left( \log \frac{\pi_\theta(y_l | x)\pi_\theta(y_w | x) + (1 - \pi_\theta(y_l | x))}{\pi_\theta(y_l | x)(1 - \pi_\theta(y_w | x))} \right)}_{\ell_{\text{unCPo}}} \end{aligned}$$

This is entirely justifiable semantically, but probably impossible to derive.

# The no reference loss landscape

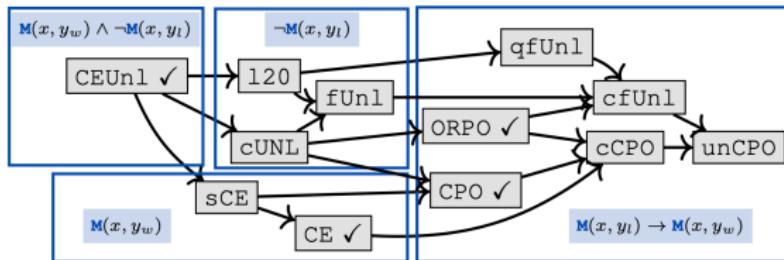
$\mathbf{M}(x, y_w)$	$\mathbf{M}(x, y_l)$	Unl	qfUnl	cUnl	CE	bCE	fUnl	ORPO	cfUnl	CPO	cCPO	uncCPO
T	T	X		X	✓	✓	X			✓ X	✓	✓
T	F	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
F	T	X	X	X	X	X	X	X	X	X	X	X
F	F	X	✓ X		X	✓ X	✓		✓			✓



- ▶ **Loss entailment graph:** Each edge indicates a strict semantic entailment relation between two losses, more constrained as you move right.

# The no reference loss landscape

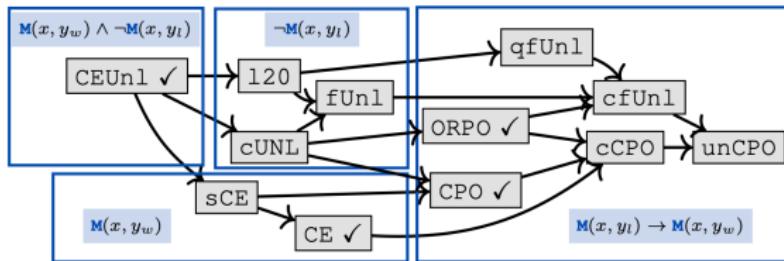
$\mathbf{M}(x, y_w)$	$\mathbf{M}(x, y_l)$	Unl	qfUnl	cUnl	CE	bCE	fUnl	ORPO	cfUnl	CPO	cCPO	uncCPO
T	T	X		X	✓	✓	X			✓ X	✓	✓
T	F	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
F	T	X	X	X	X	X	X	X	X	X	X	X
F	F	X	✓ X		X	✓ X	✓		✓	✓		✓



**Theorem:** For any  $D$  and  $\theta$ , it will hold that  $\ell_{\text{CEUnl}}(D, \theta) > \ell_{\text{CE}}(D, \theta)$  and  $\ell_{\text{Unl}}(D, \theta) > \ell_{\text{ORPO}}(D, \theta)$  and  $\ell_{\text{Unl}}(D, \theta) > \ell_{\text{CPO}}(D, \theta)$ .

# The no reference loss landscape

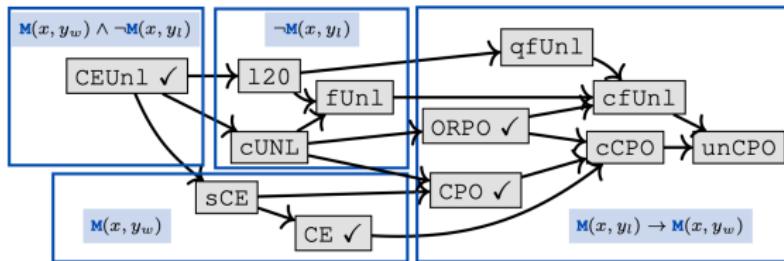
$\mathbf{M}(x, y_w)$	$\mathbf{M}(x, y_l)$	Unl	qfUnl	cUnl	CE	bCE	fUnl	ORPO	cfUnl	CPO	cCPO	uncCPO
T	T	X		X	✓	✓	X			✓ X	✓	✓
T	F	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
F	T	X	X	X	X	X	X	X	X	X	X	X
F	F	X	✓ X		X	✓ X	✓		✓			✓



**Observation:** There are many novel losses here that haven't been tested, both as variations of baselines and ORPO and CPO, not exhaustive

# The no reference loss landscape

$\mathbf{M}(x, y_w)$	$\mathbf{M}(x, y_l)$	Unl	qfUnl	cUnl	CE	bCE	fUnl	ORPO	cfUnl	CPO	cCPO	uncCPO
T	T	X		X	✓	✓	X			✓ X	✓	✓
T	F	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
F	T	X	X	X	X	X	X	X	X	X	X	X
F	F	X	✓ X		X	✓ X	✓		✓			✓



**Questions:** What are the trends we see empirically when we work through this graph; is there something special about this middle region?

Are any of these new losses good?

# Losses with a reference model

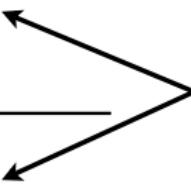
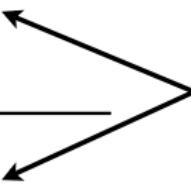
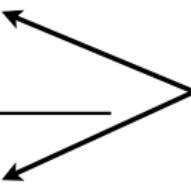
SimPO	$P := \text{Implies}(\text{And}(\text{Mref}(x, y_w), \text{M}(x, y_l)), \text{And}(\text{Mref}(x, y_l), \text{M}(x, y_w)))$ $P_H := \text{Or}(\text{And}(\text{Mref}(x, y_l), \text{M}(x, y_w)), \text{And}(\text{Mref}(x, y_w), \text{M}(x, y_l)))$	If the reference model deems the winner to be good and our model says that the loser is good, then our model should deem the winner to be good and <b>the reference model should deem the loser is good.</b>
DPO	$P := \text{Implies}(\text{And}(\text{Ref}(x, y_w), \text{M}(x, y_l)), \text{And}(\text{Ref}(x, y_l), \text{M}(x, y_w)))$ $P_H := \text{Or}(\text{And}(\text{Ref}(x, y_l), \text{M}(x, y_w)), \text{And}(\text{Ref}(x, y_w), \text{M}(x, y_l)))$	
DPOP	$P := \text{Implies}(\text{And}(\text{Ref}(x, y), \text{Ref}_1(x, y_w), \text{M}(x, y_l)), \text{And}(\text{Ref}(x, y_l), \text{M}(x, y_w), \text{M}_1(x, y_w)))$ $P_H := \text{Or}(\text{And}(\text{Ref}(x, y), \text{Ref}_1(x, y_w), \text{M}(x, y_l)), \text{And}(\text{Ref}(x, y_l), \text{M}(x, y_w), \text{M}_1(x, y_w)))$	

# Losses with a reference model

SimPO	$P := \text{Implies}(\text{And}(\text{Mref}(x, y_w), \text{M}(x, y_l)), \text{And}(\text{Mref}(x, y_l), \text{M}(x, y_w)))$ $P_H := \text{Or}(\text{And}(\text{Mref}(x, y_l), \text{M}(x, y_w)), \text{And}(\text{Mref}(x, y_w), \text{M}(x, y_l)))$	If the reference model deems the winner to be good and our model says that the loser is good, then our model should deem the winner to be good and <b>the reference model should deem the loser is good.</b>
DPO	$P := \text{Implies}(\text{And}(\text{Ref}(x, y_w), \text{M}(x, y_l)), \text{And}(\text{Ref}(x, y_l), \text{M}(x, y_w)))$ $P_H := \text{Or}(\text{And}(\text{Ref}(x, y_l), \text{M}(x, y_w)), \text{And}(\text{Ref}(x, y_w), \text{M}(x, y_l)))$	
DPOP	$P := \text{Implies}(\text{And}(\text{Ref}(x, y), \text{Ref}_1(x, y_w), \text{M}(x, y_l)), \text{And}(\text{Ref}(x, y_l), \text{M}(x, y_w), \text{M}_1(x, y_w)))$ $P_H := \text{Or}(\text{And}(\text{Ref}(x, y), \text{Ref}_1(x, y_w), \text{M}(x, y_l)), \text{And}(\text{Ref}(x, y_l), \text{M}(x, y_w), \text{M}_1(x, y_w)))$	

- ▶ **Observation:** Log-ratio difference has a strange semantics, hard to justify, problematic for all variants of DPO we consider.

# Losses with a reference model

SimPO	$P := \text{Implies}(\text{And}(\text{Mref}(x, y_w), \text{M}(x, y_l)), \text{And}(\text{Mref}(x, y_l), \text{M}(x, y_w)))$ $P_H := \text{Or}(\text{And}(\text{Mref}(x, y_l), \text{M}(x, y_w)), \text{And}(\text{Mref}(x, y_w), \text{M}(x, y_l)))$		If the reference model deems the winner to be good and our model says that the loser is good, then our model should deem the winner to be good and the reference model should deem the loser is good.
DPO	$P := \text{Implies}(\text{And}(\text{Ref}(x, y_w), \text{M}(x, y_l)), \text{And}(\text{Ref}(x, y_l), \text{M}(x, y_w)))$ $P_H := \text{Or}(\text{And}(\text{Ref}(x, y_l), \text{M}(x, y_w)), \text{And}(\text{Ref}(x, y_w), \text{M}(x, y_l)))$		
DPOP	$P := \text{Implies}(\text{And}(\text{Ref}(x, y), \text{Ref}_1(x, y_w), \text{M}(x, y_l)), \text{And}(\text{Ref}(x, y_l), \text{M}(x, y_w), \text{M}_1(x, y_w)))$ $P_H := \text{Or}(\text{And}(\text{Ref}(x, y), \text{Ref}_1(x, y_w), \text{M}(x, y_l)), \text{And}(\text{Ref}(x, y_l), \text{M}(x, y_w), \text{M}_1(x, y_w)))$		same semantics, multiplies magnitude of winner prediction

- **Observation:** Log-ratio difference has a strange semantics, hard to justify, problematic for all variants of DPO we consider.

# Losses with a reference model

SimPO	$P := \text{Implies}(\text{And}(\text{Mref}(x, y_w), \text{M}(x, y_l)), \text{And}(\text{Mref}(x, y_l), \text{M}(x, y_w)))$ $P_H := \text{Or}(\text{And}(\text{Mref}(x, y_l), \text{M}(x, y_w)), \text{And}(\text{Mref}(x, y_w), \text{M}(x, y_l)))$
DPO	$P := \text{Implies}(\text{And}(\text{Ref}(x, y_w), \text{M}(x, y_l)), \text{And}(\text{Ref}(x, y_l), \text{M}(x, y_w)))$ $P_H := \text{Or}(\text{And}(\text{Ref}(x, y_l), \text{M}(x, y_w)), \text{And}(\text{Ref}(x, y_w), \text{M}(x, y_l)))$
DPOP	$P := \text{Implies}(\text{And}(\text{Ref}(x, y), \text{Ref}_1(x, y_w), \text{M}(x, y_l)), \text{And}(\text{Ref}(x, y_l), \text{M}(x, y_w), \text{M}_1(x, y_w)))$ $P_H := \text{Or}(\text{And}(\text{Ref}(x, y), \text{Ref}_1(x, y_w), \text{M}(x, y_l)), \text{And}(\text{Ref}(x, y_l), \text{M}(x, y_w), \text{M}_1(x, y_w)))$

Either the reference model deems the loser to be good and the main model deems the winner to be good **or** the reference model deems the winner to be good and the main model deems the loser to be good, or both



- ▶ **Observation:** Hard constraints are also very strange, make it not possible to connect formal analysis with no reference losses.

# Losses with a reference model

SimPO	$P := \text{Implies}(\text{And}(\text{Mref}(x, y_w), \text{M}(x, y_l)), \text{And}(\text{Mref}(x, y_l), \text{M}(x, y_w)))$ $P_H := \text{Or}(\text{And}(\text{Mref}(x, y_l), \text{M}(x, y_w)), \text{And}(\text{Mref}(x, y_w), \text{M}(x, y_l)))$	<p>If the reference model deems the winner to be good and our model says that the loser is good, then our model should deem the winner to be good and the reference model should deem the loser is good.</p> 
DPO	$P := \text{Implies}(\text{And}(\text{Ref}(x, y_w), \text{M}(x, y_l)), \text{And}(\text{Ref}(x, y_l), \text{M}(x, y_w)))$ $P_H := \text{Or}(\text{And}(\text{Ref}(x, y_l), \text{M}(x, y_w)), \text{And}(\text{Ref}(x, y_w), \text{M}(x, y_l)))$	
DPOP	$P := \text{Implies}(\text{And}(\text{Ref}(x, y), \text{Ref}_1(x, y_w), \text{M}(x, y_l)), \text{And}(\text{Ref}(x, y_l), \text{M}(x, y_w), \text{M}_1(x, y_w)))$ $P_H := \text{Or}(\text{And}(\text{Ref}(x, y), \text{Ref}_1(x, y_w), \text{M}(x, y_l)), \text{And}(\text{Ref}(x, y_l), \text{M}(x, y_w), \text{M}_1(x, y_w)))$	

# Losses with a reference model

SimPO	$P := \text{Implies}(\text{And}(\text{Mref}(x, y_w), \text{M}(x, y_l)), \text{And}(\text{Mref}(x, y_l), \text{M}(x, y_w)))$ $P_H := \text{Or}(\text{And}(\text{Mref}(x, y_l), \text{M}(x, y_w)), \text{And}(\text{Mref}(x, y_w), \text{M}(x, y_l)))$	→	<p>If the reference model deems the winner to be good and our model says that the loser is good, then our model should deem the winner to be good and the reference model should deem the loser is good.</p>
DPO	$P := \text{Implies}(\text{And}(\text{Ref}(x, y_w), \text{M}(x, y_l)), \text{And}(\text{Ref}(x, y_l), \text{M}(x, y_w)))$ $P_H := \text{Or}(\text{And}(\text{Ref}(x, y_l), \text{M}(x, y_w)), \text{And}(\text{Ref}(x, y_w), \text{M}(x, y_l)))$		
DPOP	$P := \text{Implies}(\text{And}(\text{Ref}(x, y), \text{Ref}_1(x, y_w), \text{M}(x, y_l)), \text{And}(\text{Ref}(x, y_l), \text{M}(x, y_w), \text{M}_1(x, y_w)))$ $P_H := \text{Or}(\text{And}(\text{Ref}(x, y), \text{Ref}_1(x, y_w), \text{M}(x, y_l)), \text{And}(\text{Ref}(x, y_l), \text{M}(x, y_w), \text{M}_1(x, y_w)))$	→	$-\log \sigma \left( \log \frac{\pi_{\text{ref}}(y_l   x) \pi_\theta(y_w   x) (1 - \pi_\theta(y_l   x))}{\pi_\theta(y_l   x) \pi_{\text{ref}}(y_w   x) (1 - \pi_\theta(y_w   x))} \right)$

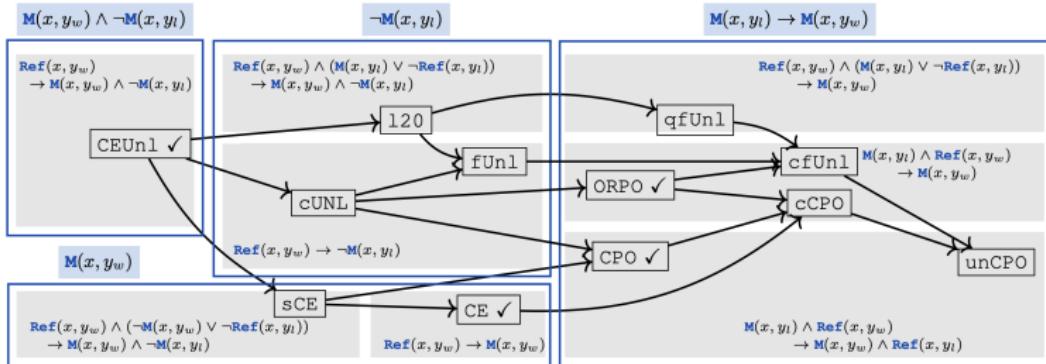
- ▶ We can play with the semantics to define new classes of DPO losses, currently exploring this empirically.

# Losses with a reference model

	$P := \text{Implies}(\text{And}(\text{Mref}(x, y_w), \text{M}(x, y_l)), \text{And}(\text{Mref}(x, y_l), \text{M}(x, y_w)))$ $P_H := \text{Or}(\text{And}(\text{Mref}(x, y_l), \text{M}(x, y_w)), \text{And}(\text{Mref}(x, y_w), \text{M}(x, y_l)))$	<p>If the reference model deems the winner to be good and our model says that the loser is good, then our model should deem the winner to be good and the reference model should deem the loser is good.</p> 
DPO	$P := \text{Implies}(\text{And}(\text{Ref}(x, y_w), \text{M}(x, y_l)), \text{And}(\text{Ref}(x, y_l), \text{M}(x, y_w)))$ $P_H := \text{Or}(\text{And}(\text{Ref}(x, y_l), \text{M}(x, y_w)), \text{And}(\text{Ref}(x, y_w), \text{M}(x, y_l)))$	
DPOP	$P := \text{Implies}(\text{And}(\text{Ref}(x, y), \text{Ref}_1(x, y_w), \text{M}(x, y_l)), \text{And}(\text{Ref}(x, y_l), \text{M}(x, y_w), \text{M}_1(x, y_w)))$ $P_H := \text{Or}(\text{And}(\text{Ref}(x, y), \text{Ref}_1(x, y_w), \text{M}(x, y_l)), \text{And}(\text{Ref}(x, y_l), \text{M}(x, y_w), \text{M}_1(x, y_w)))$	$-\log \sigma \left( \log \frac{\pi_{\text{ref}}(y_l   x) \pi_{\theta}(y_w   x) (1 - \pi_{\theta}(y_l   x))}{\pi_{\theta}(y_l   x) \pi_{\text{ref}}(y_w   x) (1 - \pi_{\theta}(y_w   x))} \right)$

**strategy** (from before): make DPO more/less constrained, and follow which directions look empirically promising, refine.

# The reference loss landscape



- ▶ Systematic procedure for generating reference losses from single model losses, much to explore.

# Conclusions

- ▶ New ideas about formalizing preference loss functions using symbolic techniques, developed new technical tools for this.

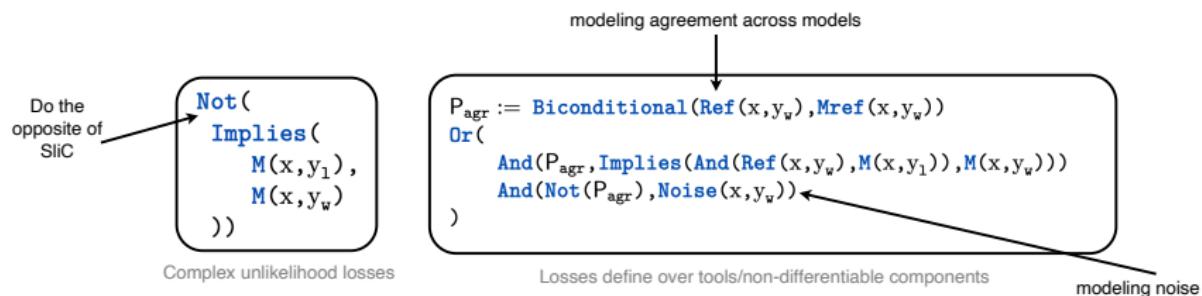
# Conclusions

- ▶ New ideas about formalizing preference loss functions using symbolic techniques, developed new technical tools for this.
- The procedure:** write a (high-level) symbolic, or modify an existing one, compile into a loss and experiment (then repeat).

# Conclusions

- ▶ New ideas about formalizing preference loss functions using symbolic techniques, developed new technical tools for this.

**The procedure:** write a (high-level) symbolic, or modify an existing one, compile into a loss and experiment (then repeat).



- ▶ Recipe for building much more complex reasoning systems with and on top of LLMs.

Thank you.

## References |

- Azar, M. G., Rowland, M., Piot, B., Guo, D., Calandriello, D., Valko, M., and Munos, R. (2023). A general theoretical paradigm to understand learning from human preferences. *arXiv preprint arXiv:2310.12036*.
- Bogin, B., Yang, K., Gupta, S., Richardson, K., Bransom, E., Clark, P., Sabharwal, A., and Khot, T. (2024). Super: Evaluating agents on setting up and executing tasks from research repositories. *arXiv preprint arXiv:2409.07440*.
- Chavira, M. and Darwiche, A. (2008). On probabilistic inference by weighted model counting. *Artificial Intelligence*, 172(6-7):772–799.
- Chen, J., Yuan, S., Ye, R., Majumder, B. P., and Richardson, K. (2023). Put your money where your mouth is: Evaluating strategic planning and execution of IIm agents in an auction arena. *arXiv preprint arXiv:2310.05746*.
- Dai, J., Pan, X., Sun, R., Ji, J., Xu, X., Liu, M., Wang, Y., and Yang, Y. (2024). Safe rlhf: Safe reinforcement learning from human feedback. In *The Twelfth International Conference on Learning Representations*.
- Gu, Y., Mishra, B. D., and Clark, P. (2023). Do language models have coherent mental models of everyday things? *Proceedings of ACL*.
- Hong, J., Lee, N., and Thorne, J. (2024). Reference-free monolithic preference optimization with odds ratio. *arXiv preprint arXiv:2403.07691*.
- Jeffrey, R. C. (1965). *The logic of decision*. University of Chicago press.

## References II

- Ji, J., Liu, M., Dai, J., Pan, X., Zhang, C., Bian, C., Chen, B., Sun, R., Wang, Y., and Yang, Y. (2024). Beavertails: Towards improved safety alignment of llm via a human-preference dataset. *Advances in Neural Information Processing Systems*, 36.
- Kassner, N., Tafjord, O., Sabharwal, A., Richardson, K., Schuetze, H., and Clark, P. (2023). Language models with rationality. *Proceedings of EMNLP*.
- Li, T., Gupta, V., Mehta, M., and Srikumar, V. (2019). A Logic-Driven Framework for Consistency of Neural Models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3924–3935, Hong Kong, China. Association for Computational Linguistics.
- Meng, Y., Xia, M., and Chen, D. (2024). Simpo: Simple preference optimization with a reference-free reward. *arXiv preprint arXiv:2405.14734*.
- Pal, A., Karkhanis, D., Dooley, S., Roberts, M., Naidu, S., and White, C. (2024). Smaug: Fixing failure modes of preference optimisation with dpo-positive. *arXiv preprint arXiv:2402.13228*.
- Park, R., Rafailov, R., Ermon, S., and Finn, C. (2024). Disentangling length from quality in direct preference optimization. *arXiv preprint arXiv:2403.19159*.
- Rafailov, R., Sharma, A., Mitchell, E., Ermon, S., Manning, C. D., and Finn, C. (2024). Direct preference optimization: Your language model is secretly a reward model. *Proceedings of Neurips*.
- Rescher, N. (1967). *The logic of decision and action*. University of Pittsburgh Pre.

## References III

- Tang, Y., Guo, Z. D., Zheng, Z., Calandriello, D., Munos, R., Rowland, M., Richemond, P. H., Valko, M., Pires, B. Á., and Piot, B. (2024). Generalized preference optimization: A unified approach to offline alignment. *arXiv preprint arXiv:2402.05749*.
- Welleck, S., Kulikov, I., Roller, S., Dinan, E., Cho, K., and Weston, J. (2019). Neural text generation with unlikelihood training. In *International Conference on Learning Representations*.
- Xu, J., Zhang, Z., Friedman, T., Liang, Y., and Broeck, G. (2018). A Semantic Loss Function for Deep Learning with Symbolic Knowledge. In *International Conference on Machine Learning*, pages 5498–5507.
- Yang, R., Chen, J., Zhang, Y., Yuan, S., Chen, A., Richardson, K., Xiao, Y., and Yang, D. (2024). Selfgoal: Your language agents already know how to achieve high-level goals. *arXiv preprint arXiv:2406.04784*.
- Zhang, Y., Yuan, S., Hu, C., Richardson, K., Xiao, Y., and Chen, J. (2024). Timearena: Shaping efficient multitasking language agents in a time-aware simulation. *arXiv preprint arXiv:2402.05733*.
- Zhao, Y., Khalman, M., Joshi, R., Narayan, S., Saleh, M., and Liu, P. J. (2022). Calibrating sequence likelihood improves conditional language generation. In *The eleventh international conference on learning representations*.