

# Understanding the Logic of Generative AI through Logic

**Kyle Richardson**

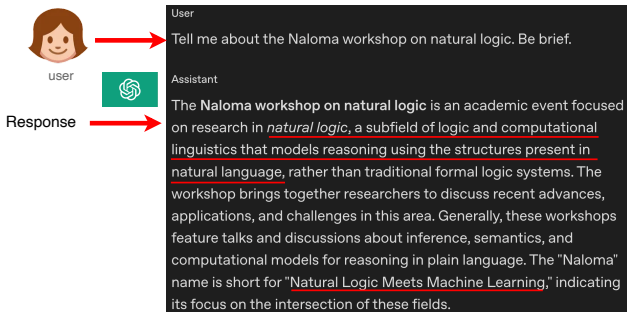
Allen Institute for AI (AI2)

August 2025

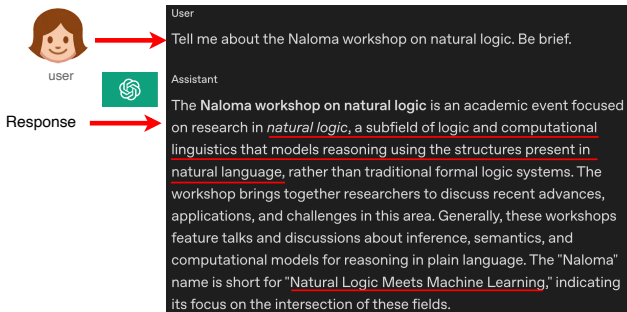
**Collaborators:** Ashish Sabharwal (AI2), Vivek Srimumar (University of Utah)



# General purpose large language models (LLMs)

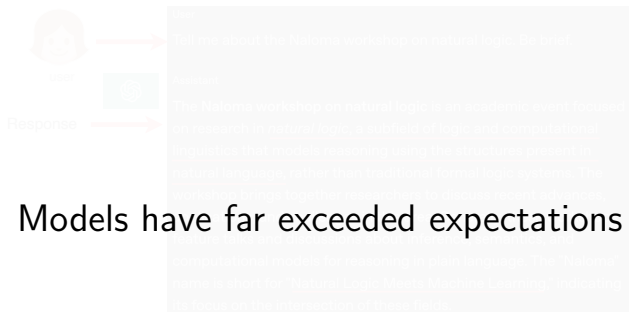


# General purpose large language models (LLMs)



- ▶ **General purpose models:** Trained at massive scales, used *as-is* and directly for a wide range of problems.

# General purpose large language models (LLMs)



Models have far exceeded expectations

- **General purpose models:** Trained at massive scales, used *as-is* and directly for a wide range of problems.

# Language models as agent simulators

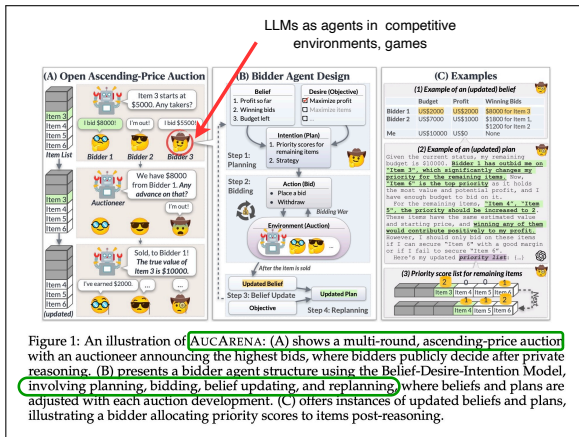


Figure 1: An illustration of AUCARENA: (A) shows a multi-round, ascending-price auction with an auctioneer announcing the highest bids, where bidders publicly decide after private reasoning. (B) presents a bidder agent structure using the Belief-Desire-Intention Model, involving planning, bidding, belief 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? (Chen et al., 2023; Zhang et al., 2024; Yang et al., 2025)

# Language models as agent simulators

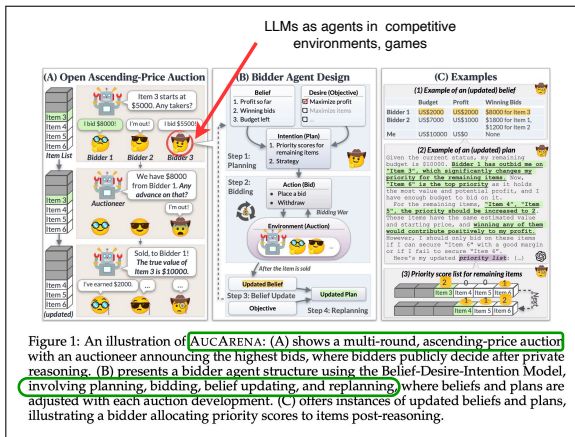
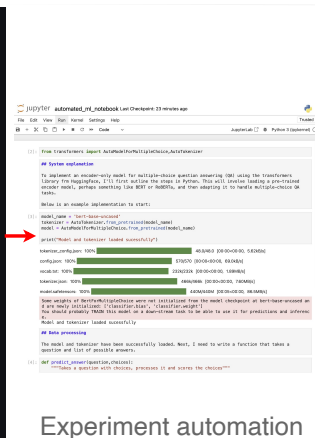
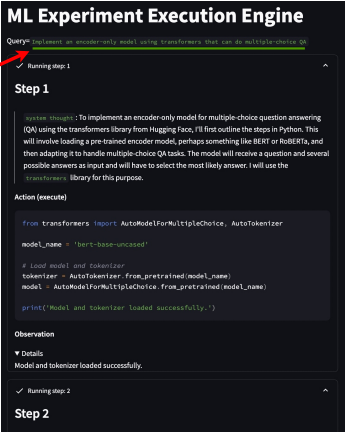


Figure 1: An illustration of AUCARENA: (A) shows a multi-round, ascending-price auction with an auctioneer announcing the highest bids, where bidders publicly decide after private reasoning. (B) presents a bidder agent structure using the Belief-Desire-Intention Model, involving planning, bidding, belief 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, adversarial language experts.

## Language models as part of complex systems

Model  
generated  
code

- ▶ SUPER (Bogin et al., 2024), benchmark for setting up and executing research code repositories.

# Language models as part of complex systems

Machine learning experiment



Model generated code



## 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 QA 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

## Experiment automation

jupyter automated\_jml\_notebook Local Checkpoint 33 minutes ago

File Edit View Help

Automated JML Notebook Python 3 (Data Science)

```
[1]: from transformers import AutoModelForMultipleChoice, AutoTokenizer

# System explanation
'''To implement an encoder-only model for multiple-choice question answering (QA) using the transformers library from HuggingFace, 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 QA tasks.

Below is an example implementation to start:'''

[2]: model_name = 'bert-base-uncased'
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForMultipleChoice.from_pretrained(model_name)
print('Model and tokenizer loaded successfully.')
```

Model and tokenizer loaded successfully.


Observation

▼ Details


Model and tokenizer loaded successfully.

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


# Language models as part of complex systems




Machine learning experiment



ChatGPT



Model generated code



## ML Experiment Execution Engine

Query:

Running step 1

### Step 1

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 BERT 2.0, and then adapting it to handle multiple-choice QA 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 (executed)

```
from transformers import AutoTokenizer, AutoModelForSequenceClassification
from transformers import logging
import torch

model = AutoModelForSequenceClassification.from_pretrained('bert-base-uncased', num_labels=5)
tokenizer = AutoTokenizer.from_pretrained('bert-base-uncased')
```

Observation

✖ Details

Model and tokenizer loaded successfully.

Running step 1

### Step 2

## Experiment automation

```
from transformers import AutoModelForSequenceClassification
from transformers import logging
import torch

model = AutoModelForSequenceClassification.from_pretrained('bert-base-uncased', num_labels=5)
tokenizer = AutoTokenizer.from_pretrained('bert-base-uncased')
```

How to use tokenizer to encode input text

To generate an embedding vector for a text-based question, we need to use the tokenizer. There are two methods: `tokenizer.tokenize()` and `tokenizer.encode()`. The former returns a list of tokens, while the latter returns a single integer representing the token index. We will use the latter method for encoding.

How to use tokenizer to encode input text

We will use the `tokenizer.encode()` method to generate a single integer representing the token index. This integer can then be used to look up the corresponding word in the vocabulary and generate the output.

How to use tokenizer to encode input text


We will use the `tokenizer.encode()` method to generate a single integer representing the token index. This integer can then be used to look up the corresponding word in the vocabulary and generate the output.

How to use tokenizer to encode input text


We will use the `tokenizer.encode()` method to generate a single integer representing the token index. This integer can then be used to look up the corresponding word in the vocabulary and generate the output.

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

# Language models as part of complex systems




Machine learning experiment



ChatGPT

Model generated code



## ML Experiment Execution Engine

Query:

Running step 1

### Step 1

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 QA 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 `torch.nn` library for this purpose.

Action (executed)

```
from transformers import AutoModelForSequenceClassification, AutoTokenizer
model = AutoModelForSequenceClassification.from_pretrained('bert-base-uncased')
tokenizer = AutoTokenizer.from_pretrained('bert-base-uncased')
```

Model's output


Observation

\* Details

Model and tokenizer loaded successfully.

Running step 1

### Step 2

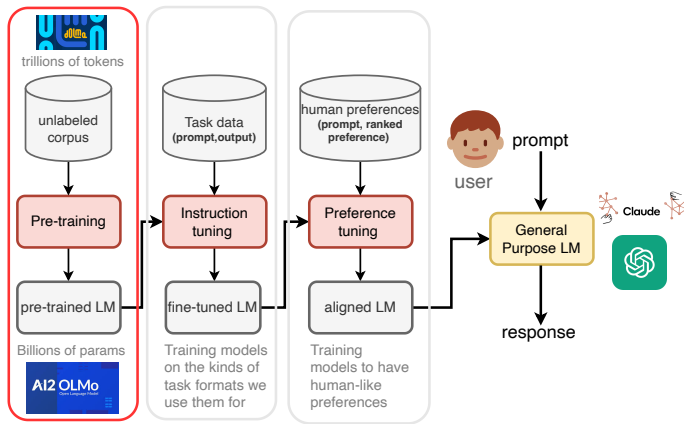


Experiment automation

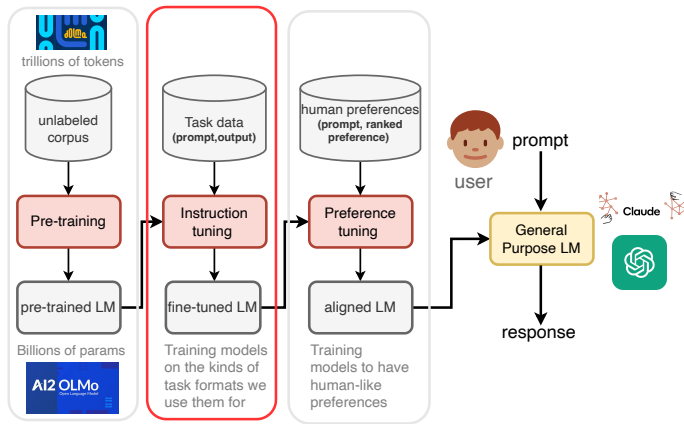
Missing algorithmic and semantic foundations.

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

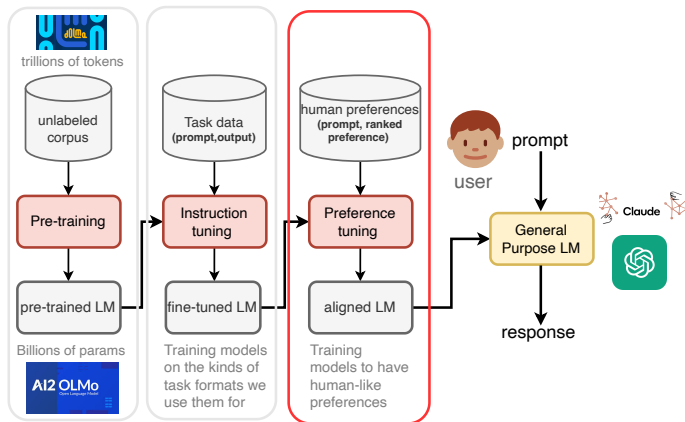
# How do we get to general purpose LLMs? **recipe**



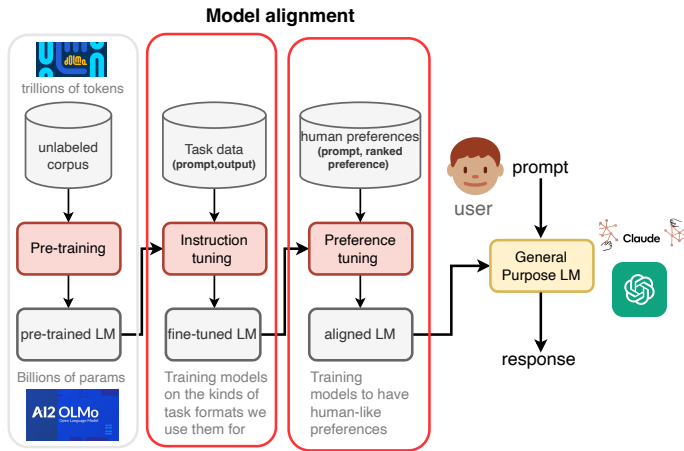
# How do we get to general purpose LLMs? **recipe**



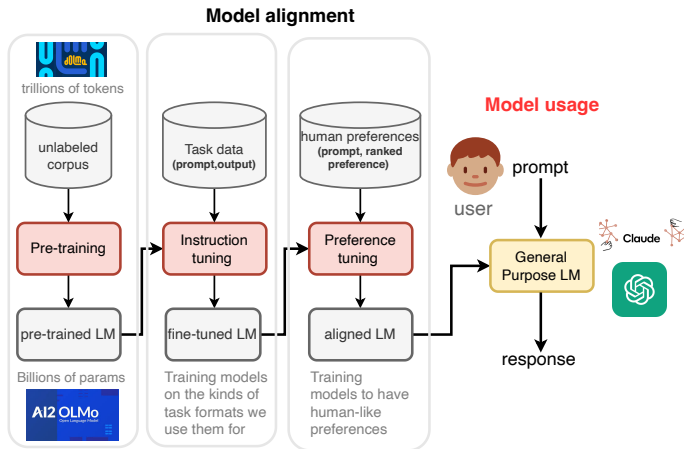
# How do we get to general purpose LLMs? **recipe**



# How do we get to general purpose LLMs? **recipe**



# How do we get to general purpose LLMs? **recipe**



# OLMo: fully open-source general purpose LMs

The screenshot shows the Hugging Face model card for `allennai/OLMo-2-1124-13B-Instruct`. The card features the OLMo logo, a note about a 1/3/2025 update, and release documentation. On the right, there's a sidebar with statistics (10,023 downloads last month), safetensors information (11.7B params), inference providers, and a model tree. The model tree is highlighted with a red box and shows a hierarchy of models derived from the base model, including finetuned versions and quantizations.

**allennai/OLMo-2-1124-13B-Instruct** 👍 like 46 👤 following 3.68k

Text Generation Transformers Safetensors allennai/RLVR-MATH English olmo2 conversational anirv-2501.00656 anirv-2411.15124 License: apache-2.0

Model card Files and versions 🔗 set Community ⚙️ Settings 🚀 Train 🚀 Deploy 📄 Use this model

**OLMo-2-1124-13B-Instruct**

**NOTE: 1/3/2025 UPDATE:**

Upon the initial release of OLMo-2 models, we realized the post-trained models did not share the pre-tokenization logic that the base models use. As a result, we have trained new post-trained models. The new models are available under the same names as the original models, but we have made the old models available with a postfix "-preview". See [OLMo 2 Preview Post-trained Models](#) for the collection of the legacy models.

**Release Documentation**

OLMo 2 13B Instruct November 2024 is post-trained variant of the [OLMo-2 13B November 2024](#) model, which has undergone supervised finetuning on an OLMo-specific variant of the Tulu 3 dataset (<https://huggingface.co/datasets/allennai/tulu-3-sft-olmo-2-mixture>) and further DPO training on [this dataset](#), and finally RLVR training using [this data](#). Tulu 3 is designed for state-of-the-art performance on a diversity of tasks in addition to chat, such as MATH, GSM8K, and IFEval. Check out the [OLMo 2 paper](#) or [Tulu 3 paper](#) for more details!

Downloads last month  
**10,023**  
[View full history](#)

**Safetensors**  
Model size: 11.7B params Tensor type: BF16 Chat template: Files info

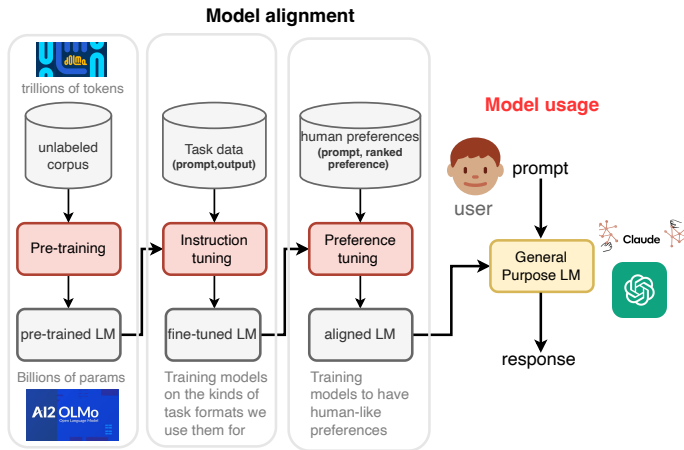
**Inference Providers**  
Text Generation  
This model isn't deployed by any Inference Provider. [Ask for provider support](#)

**Model tree for allennai/OLMo-2-1124-13B-Instruct**

- Base model: allennai/OLMo-2-1124-7B
  - Finetuned: allennai/OLMo-2-1124-7B-SFT
    - Finetuned: allennai/OLMo-2-1124-7B-DPO
      - Finetuned: allennai/OLMo-2-1124-13B-Instruct-RLVR1
        - Finetuned (1): allennai/OLMo-2-1124-13B-Instruct-RLVR2 (this model)
        - Adapters: 4 models
        - Finetunes: 2 models
        - Quantizations: 29 models

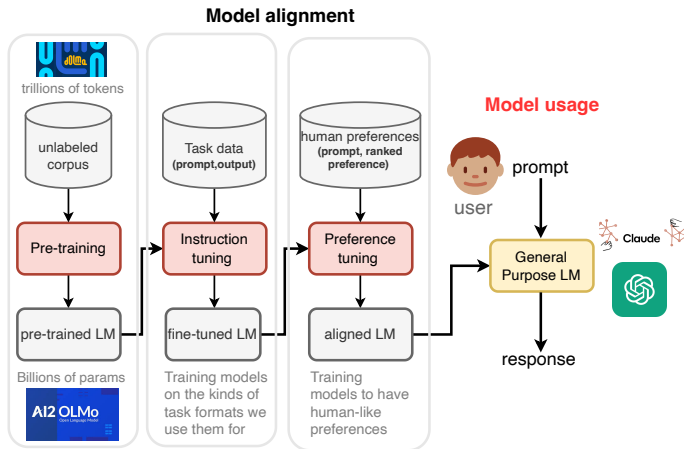
<https://allennai.org/olmo>

# How do we get to general purpose LLMs? **recipe**



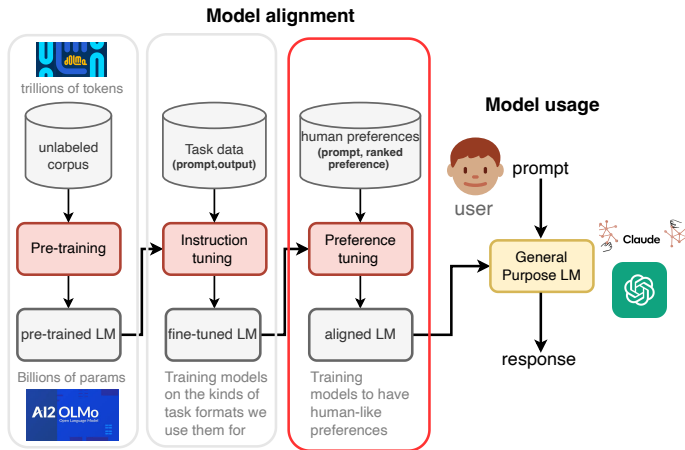
- **Dilemma:** we know vanishingly little about commercial models, models and datasets in general are huge, opaque.

# How do we get to general purpose LLMs? **recipe**



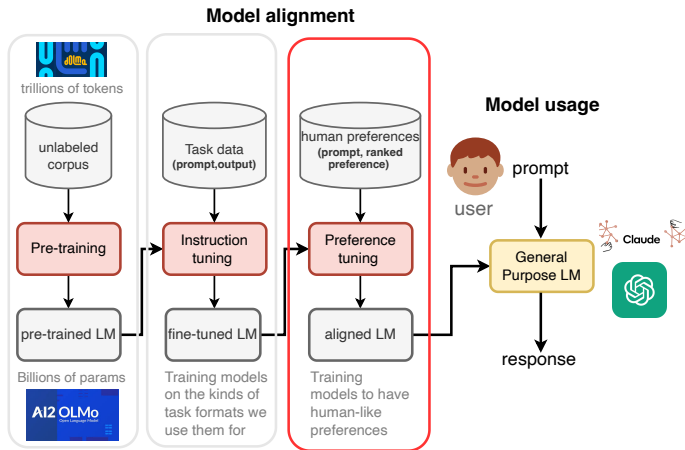
An obvious problem for safety and applications, but also for deciding what research to do, how to innovate.

# Modeling the formal semantics of LLM algorithms

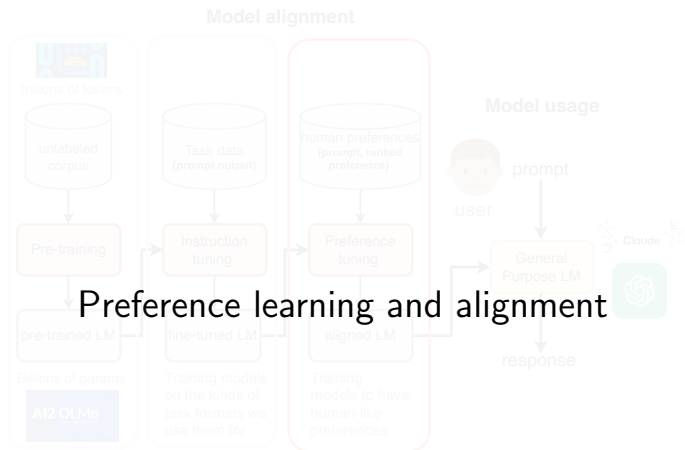


**Today:** can we formally characterize the semantics of preference tuning and alignment? Both for understanding and innovation; **armchair NLP**.

# 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?



## Preference learning and alignment

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

# Offline preference alignment in a nutshell

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

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

optimize a policy model  $y \sim \pi_\theta(\cdot | x)$  (**LLM**) to such preferences.

# Offline preference alignment in a nutshell

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

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

optimize a policy model  $y \sim \pi_\theta(\cdot | x)$  (**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, ....*

# Offline preference alignment in a nutshell

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

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

optimize a policy model  $y \sim \pi_\theta(\cdot | x)$  (**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, ....*

**Note:** What constitutes a *winner* or *loser* is fuzzy.

# Direct Preference Alignment (DPA) approaches

---

## Direct Preference Optimization: Your Language Model is Secretly a Reward Model

---

Rafael Rafailov<sup>\*†</sup>

Archit Sharma<sup>\*†</sup>

Eric Mitchell<sup>\*†</sup>

Stefano Ermon<sup>‡‡</sup>

Christopher D. Manning<sup>†</sup>

Chelsea Finn<sup>†</sup>

<sup>†</sup>Stanford University <sup>‡</sup>CZ Biohub  
{rafailov,architsh,eric.mitchell}@cs.stanford.edu

### Abstract

While large-scale unsupervised language models (LMs) learn broad world knowledge and some reasoning skills, achieving precise control of their behavior is difficult due to the completely unsupervised nature of their training. Existing methods for gaining such steerability collect human labels of the relative quality of model generations and fine-tune the unsupervised LM to align with these preferences, often with reinforcement learning from human feedback (RLHF). However, RLHF is a complex and often unstable procedure, first fitting a reward model that reflects the human preferences, and then fine-tuning the large unsupervised LM using reinforcement learning to maximize this estimated reward without drifting too far from the original model. In this paper we introduce a new parameterization of the reward model in RLHF that enables extraction of the corresponding optimal policy in closed form, allowing us to solve the standard RLHF problem with only a simple classification loss. The resulting algorithm, which we call *Direct Preference Optimization* (DPO), is stable, performant, and computationally lightweight, eliminating the need for sampling from the LM during fine-tuning or performing significant hyperparameter tuning. Our experiments show that DPO can fine-tune LMs to align with human preferences as well as or better than existing methods. Notably, fine-tuning with DPO exceeds PPO-based RLHF in ability to control sentiment of generations, and matches or improves response quality in summarization and single-turn dialogue while being substantially simpler to implement and train.

# Direct Preference Alignment (DPA) approaches

## Direct Preference Optimization: Your Language Model is Secretly a Reward Model

Rafael Rafailov<sup>1†</sup>

Archit Sharma<sup>1†</sup>

Eric Mitchell<sup>1†</sup>

Stefano Ermon<sup>1</sup> Christopher Manning<sup>1</sup> Chelsea Finn<sup>1</sup>

### DPO loss function

<sup>1</sup>Stanford University <sup>2</sup>CZ Biohub

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

**Intuitively:** reasoning about relationship between predictions of policy  $\pi_{\theta}$  and reference  $\pi_{\text{ref}}$ .

# Direct Preference Alignment (DPA) approaches

## Direct Preference Optimization: Your Language Model is Secretly a Reward Model

Rafael Rafailov<sup>1</sup>

Archit Sharma<sup>1</sup>

Eric Mitchell<sup>1\*</sup>

Stefano Ermon<sup>1</sup>

Christopher D. Manning<sup>1</sup>

Chelsea Finn<sup>1</sup>

<sup>1</sup>Stanford University <sup>2</sup>CZ Biohub  
{rafailov,architah,eric.mitchell}@cs.stanford.edu

<sup>\*</sup>best paper

# These equations are not easy to understand

While large-scale unsupervised language models (LMs) learn broad world knowledge and some reasoning skills, achieving precise control of their behavior is difficult due to the completely unsupervised nature of their training. Existing methods for gaining such steerability collect human labels of the relative quality of model generations and fine-tune the unsupervised LM to align with these preferences, often with reinforcement learning from human feedback (RLHF). However, RLHF is a complex and often unstable procedure, first fitting a reward model that reflects the human preferences, and then fine-tuning the large unsupervised LM using reinforcement learning to maximize this estimated reward without drifting too far from the original model. In this paper we introduce a new parameterization of the reward model in RLHF that enables extraction of the corresponding optimal policy in closed form, allowing us to solve the standard RLHF problem with only a simple classification loss. The resulting algorithm, which we call *Direct Preference Optimization* (DPO), is stable, performant, and computationally lightweight, eliminating the need for sampling from the LM during fine-tuning or performing significant hyperparameter tuning. Our experiments show that DPO can fine-tune LMs to align with human preferences as well as or better than existing methods. Notably, fine-tuning with DPO exceeds PPO-based RLHF in ability to control sentiment of generations, and matches or improves response quality in summarization and single-turn dialogue while being substantially simpler to implement and train.

# Direct Preference Alignment (DPA) approaches

## Direct Preference Optimization: Your Language Model is Secretly a Reward Model

Rafael Rafailov<sup>1†</sup>

Archit Sharma<sup>1†</sup>

Eric Mitchell<sup>1†</sup>

Stefano Ermon<sup>1</sup> Christopher Manning<sup>1</sup> David D. Osherson<sup>2</sup> Chelsea Finn<sup>1</sup>

### DPO loss function

<sup>1</sup>Stanford University <sup>2</sup>CZ Biohub

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

**Question:** What kind of discrete reasoning problems do these losses encode?

model generations and fine-tune the unsupervised LM to align with these preferences, often with reinforcement learning from human feedback (RLHF). However, RLHF is a complex and often unstable procedure, first fitting a reward model that reflects human preferences, and then fine-tuning the large unsupervised LM using this reward model. In this paper, we show that the RLHF procedure is not too far from the original model. In this paper we introduce a new parameter-free reward model in RLHF that enables extraction of the corresponding optimal policy. The resulting DPO loss is a simple closed-form expression, which we call *Direct Preference Optimization* (DPO), is stable, performant, and computationally lightweight, eliminating the need for sampling from the LM during fine-tuning or performing significant hyperparameter tuning. Our experiments show that DPO can fine-tune LMs to align with human preferences as well as or better than existing methods. Notably, fine-tuning with DPO exceeds PPO-based RLHF in ability to control sentiment of generations, and matches or improves response quality in summarization and single-turn dialogue while being substantially simpler to implement and train.

# The many varieties of DPO

## Direct Preference Optimization: Your Language Model is Secretly a Reward Model

Rafael Rafailov<sup>\*1</sup>

Archit Sharma<sup>\*1</sup>

Eric Mitchell<sup>\*1</sup>

Stefano Ermon<sup>†2</sup>

Christopher D. Manning<sup>‡</sup>

Chelsea Finn<sup>‡</sup>

<sup>1</sup>Stanford University <sup>2</sup>CZ Biohub  
{rafailov,architsh,eric.mitchell}@cs.stanford.edu

### Abstract

While large-scale unsupervised language models (LMs) learn broad world knowledge and some reasoning skills, achieving precise control of their behavior is difficult due to the completely unsupervised nature of their training. Existing methods for gaining such steerability collect human labels of the relative quality of model generations and fine-tune the unsupervised LM to align with these preferences, often with reinforcement learning from human feedback (RLHF). However, RLHF is a complex and often unstable procedure, first fitting a reward model that reflects the human preferences, and then fine-tuning the large unsupervised LM using reinforcement learning to maximize this estimated reward without drifting too far from the original model. In this paper we introduce a new parameterization of the reward model in RLHF that enables extraction of the corresponding optimal policy in closed form, allowing us to solve the standard RLHF problem with only a simple classification loss. The resulting algorithm, which we call *Direct Preference Optimization* (DPO), is stable, performant, and computationally lightweight, eliminating the need for sampling from the LM during fine-tuning or performing significant hyperparameter tuning. Our experiments show that DPO can fine-tune LMs to align with human preferences as well as or better than existing methods. Notably, fine-tuning with DPO exceeds PPO-based RLHF in ability to control sentiment of generations, and matches or improves response quality in summarization and single-turn dialogue while being substantially simpler to implement and train.

## DPO loss

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

# The many varieties of DPO

## Direct Preference Optimization: Your Language Model is Secretly a Reward Model

Rafael Rafailov<sup>\*1</sup> Archit Sharma<sup>\*1</sup> Eric Mitchell<sup>\*1</sup>  
Stefano Ermon<sup>1‡</sup> Christopher D. Manning<sup>1</sup> Chelsea Finn<sup>1</sup>

<sup>1</sup>Stanford University <sup>‡</sup>CZ Biohub  
{rafailov,architsh,eric.mitchell}@cs.stanford.edu

### Abstract

While large-scale unsupervised language models (LMs) learn broad world knowledge and some reasoning skills, achieving precise control of their behavior is difficult due to the completely unsupervised nature of their training. Existing methods for gaining such steerability collect human labels of the relative quality of model generations and fine-tune the unsupervised LM to align with these preferences, often with reinforcement learning from human feedback (RLHF). However, RLHF is a complex and often unstable procedure, first fitting a reward model that reflects the human preferences, and then fine-tuning the large unsupervised LM using reinforcement learning to maximize this estimated reward without drifting too far from the original model. In this paper we introduce a new parameterization of the reward model in RLHF that enables extraction of the corresponding optimal policy in closed form, allowing us to solve the standard RLHF problem with only a simple classification loss. The resulting algorithm, which we call *Direct Preference Optimization* (DPO), is stable, performant, and computationally lightweight, eliminating the need for sampling from the LM during fine-tuning or performing significant hyperparameter tuning. Our experiments show that DPO can fine-tune LMs to align with human preferences as well as or better than existing methods. Notably, fine-tuning with DPO exceeds PPO-based RLHF in ability to control sentiment of generations, and matches or improves response quality in summarization and single-turn dialogue while being substantially simpler to implement and train.

## DPO loss

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

## DPO variants

Method	Objective
RRHF [91]	$\max \left( 0, -\frac{1}{ \mathcal{D}_w } \log \pi_{\theta}(y_w x) + \frac{1}{ \mathcal{D}_l } \log \pi_{\theta}(y_l x) \right) - \lambda \log \pi_{\theta}(y_w x)$
SLIC-HF [96]	$\max \left( 0, \delta - \log \pi_{\theta}(y_w x) + \log \pi_{\theta}(y_l x) - \lambda \log \pi_{\theta}(y_w x) \right)$
DPO [66]	$-\log \sigma \left( \beta \log \frac{\pi_{\theta}(y_w x)}{\pi_{\text{ref}}(y_w x)} - \beta \log \frac{\pi_{\theta}(y_l x)}{\pi_{\text{ref}}(y_l x)} \right)$
IPO [6]	$\left( \log \frac{\pi_{\theta}(y_w x)}{\pi_{\text{ref}}(y_w x)} - \log \frac{\pi_{\theta}(y_l x)}{\pi_{\text{ref}}(y_l x)} - \frac{1}{\beta r} \right)^2$
CPO [88]	$-\log \sigma \left( \beta \log \pi_{\theta}(y_w x) - \beta \log \pi_{\theta}(y_l x) \right) - \lambda \log \pi_{\theta}(y_w x)$
KTO [29]	$-\lambda_w \sigma \left( \beta \log \frac{\pi_{\theta}(y_w x)}{\pi_{\text{ref}}(y_w x)} - z_{\text{ref}} \right) + \lambda_l \sigma \left( z_{\text{ref}} - \beta \log \frac{\pi_{\theta}(y_l x)}{\pi_{\text{ref}}(y_l x)} \right)$ , where $z_{\text{ref}} = \mathbb{E}_{(x,y) \sim \mathcal{D}} [\beta \text{KL}(\pi_{\theta}(y x)    \pi_{\text{ref}}(y x))]$
ORPO [42]	$-\log p_{\theta}(y_w x) - \lambda \log \sigma \left( \log \frac{p_{\theta}(y_w x)}{1 - p_{\theta}(y_w x)} - \log \frac{p_{\theta}(y_l x)}{1 - p_{\theta}(y_l x)} \right)$ , where $p_{\theta}(y x) = \exp \left( \frac{1}{ \mathcal{V} } \log \pi_{\theta}(y x) \right)$
R-DPO [64]	$-\log \sigma \left( \beta \log \frac{\pi_{\theta}(y_w x)}{\pi_{\text{ref}}(y_w x)} - \beta \log \frac{\pi_{\theta}(y_l x)}{\pi_{\text{ref}}(y_l x)} + (\alpha y_w  - \alpha y_l ) \right)$
SimPO	$-\log \sigma \left( \frac{\beta}{ \mathcal{D}_w } \log \pi_{\theta}(y_w x) - \frac{\beta}{ \mathcal{D}_l } \log \pi_{\theta}(y_l x) - \gamma \right)$

from Meng et al. (2024)

- ▶ **No reference approaches** (e.g., CPO, ORPO, *only involves a single model*) versus multi-model, **reference approaches** (DPO).

# The many varieties of DPO

## Direct Preference Optimization: Your Language Model is Secretly a Reward Model

Rafael Rafailov<sup>\*1</sup> Archit Sharma<sup>\*1</sup> Eric Mitchell<sup>\*1</sup>  
Stefano Ermon<sup>1‡</sup> Christopher D. Manning<sup>1</sup> Chelsea Finn<sup>1</sup>

<sup>1</sup>Stanford University <sup>‡</sup>CZ Biohub  
{rafailov,architsh,eric.mitchell}@cs.stanford.edu

### Abstract

While large-scale unsupervised language models (LMs) learn broad world knowledge and some reasoning skills, achieving precise control of their behavior is difficult due to the completely unsupervised nature of their training. Existing methods for gaining such steerability collect human labels of the relative quality of model generations and fine-tune the unsupervised LM to align with these preferences, often with reinforcement learning from human feedback (RLHF). However, RLHF is a complex and often unstable procedure, first fitting a reward model that reflects the human preferences, and then fine-tuning the large unsupervised LM using reinforcement learning to maximize this estimated reward without drifting too far from the original model. In this paper we introduce a new parameterization of the reward model in RLHF that enables extraction of the corresponding optimal policy in closed form, allowing us to solve the standard RLHF problem with only a simple classification loss. The resulting algorithm, which we call *Direct Preference Optimization* (DPO), is stable, performant, and computationally lightweight, eliminating the need for sampling from the LM during fine-tuning or performing significant hyperparameter tuning. Our experiments show that DPO can fine-tune LMs to align with human preferences as well as or better than existing methods. Notably, fine-tuning with DPO exceeds PPO-based RLHF in ability to control sentiment of generations, and matches or improves response quality in summarization and single-turn dialogue while being substantially simpler to implement and train.

## DPO loss

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

## DPO variants

Method	Objective
RRHF [91]	$\max \left( 0, -\frac{1}{ \mathcal{D}_w } \log \pi_{\theta}(y_w x) + \frac{1}{ \mathcal{D}_l } \log \pi_{\theta}(y_l x) \right) - \lambda \log \pi_{\theta}(y_w x)$
SLiC-HF [96]	$\max \left( 0, \delta - \log \pi_{\theta}(y_w x) + \log \pi_{\theta}(y_l x) \right) - \lambda \log \pi_{\theta}(y_w x)$
DPO [66]	$-\log \sigma \left( \beta \log \frac{\pi_{\theta}(y_w x)}{\pi_{\text{ref}}(y_w x)} - \beta \log \frac{\pi_{\theta}(y_l x)}{\pi_{\text{ref}}(y_l x)} \right)$
IPO [6]	$\left( \log \frac{\pi_{\theta}(y_w x)}{\pi_{\text{ref}}(y_w x)} - \log \frac{\pi_{\theta}(y_l x)}{\pi_{\text{ref}}(y_l x)} - \frac{1}{\beta r} \right)^2$
CPO [88]	$-\log \sigma \left( \beta \log \pi_{\theta}(y_w x) - \beta \log \pi_{\theta}(y_l x) \right) - \lambda \log \pi_{\theta}(y_w x)$
KTO [29]	$-\lambda_w \sigma \left( \beta \log \frac{\pi_{\theta}(y_w x)}{\pi_{\text{ref}}(y_w x)} - z_{\text{ref}} \right) + \lambda_l \sigma \left( z_{\text{ref}} - \beta \log \frac{\pi_{\theta}(y_l x)}{\pi_{\text{ref}}(y_l x)} \right)$ , where $z_{\text{ref}} = \mathbb{E}_{(x,y) \sim \mathcal{D}} [\beta \text{KL}(\pi_{\theta}(y x)    \pi_{\text{ref}}(y x))]$
ORPO [42]	$-\log p_{\theta}(y_w x) - \lambda \log \sigma \left( \log \frac{p_{\theta}(y_w x)}{1 - p_{\theta}(y_w x)} - \log \frac{p_{\theta}(y_l x)}{1 - p_{\theta}(y_l x)} \right)$ , where $p_{\theta}(y x) = \exp \left( \frac{1}{ \mathcal{V} } \log \pi_{\theta}(y x) \right)$
R-DPO [64]	$-\log \sigma \left( \beta \log \frac{\pi_{\theta}(y_w x)}{\pi_{\text{ref}}(y_w x)} - \beta \log \frac{\pi_{\theta}(y_l x)}{\pi_{\text{ref}}(y_l x)} + (\alpha  y_w  - \alpha  y_l ) \right)$
SimPO	$-\log \sigma \left( \frac{\beta}{ \mathcal{D}_w } \log \pi_{\theta}(y_w x) - \frac{\beta}{ \mathcal{D}_l } \log \pi_{\theta}(y_l x) - \gamma \right)$

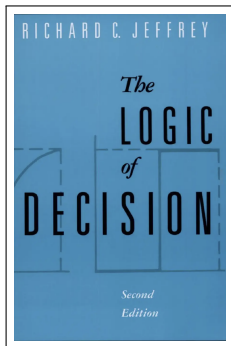
from Meng et al. (2024)

**Questions:** How are all these variations related to one another, nature of the space of losses?



# Haven't these semantic questions been looked at before?

**Analytic philosophy:** Much work on the semantics of pairwise preference, rich languages for expressing ideas.



(Jeffrey, 1965)

THE STATUS OF VARIOUS PREFERENCE PRINCIPLES						
Preference Principle	Von Wright	Chisholm Sosa	Martin	$P^\#$	$P^\star$	$P^\omega$
1. $pPq \rightarrow \sim(qPp)$	✓	✓	✓	+	+	+
2. $(pPq \ \& \ qPr) \rightarrow pPr$	✓	✓	✓	+	+	+
3. $pPq \rightarrow \sim qP\sim p$		x	✓	$(+)^1$	+	+
4. $\sim qP\sim p \rightarrow pPq$		x	✓	$(+)^1$	+	+
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. $pPq \rightarrow [(p \ \& \ r) P(q \ \& \ r) \ \& \ (p \ \& \ \sim r) P(q \ \& \ \sim r)]$	✓			—	—	+
11. $[(p \ \& \ r) P(q \ \& \ r) \ \& \ (p \ \& \ \sim r) P(q \ \& \ \sim r)] \rightarrow pPq$	✓			$(+)^2$	$(+)^2$	+
12. $[\sim(pPq) \ \& \ \sim(qPr)] \rightarrow \sim(pPr)$		✓		—	—	—
13. $(pPr \vee qPr) \rightarrow (p \vee q) Pr$			✓	—	—	—
14. $(p \vee q) Pr \rightarrow [pPr \ \& \ qPr]$	✓			—	—	—
15. $[pPr \ \& \ qPr] \rightarrow (p \vee q) Pr$	✓			—	—	—
16. $(p \vee q) Pr \rightarrow (pPr \vee qPr)$				—	—	—
17. $pP(q \vee r) \rightarrow (pPq \ \vee \ pPr)$			✓	—	—	—
18. $(pPq \ \& \ pPr) \rightarrow pP(q \vee r)$				—	—	—
19. $(pPr \ \& \ qPr) \rightarrow (p \ \& \ q) Pr$				—	—	—

Semantic foundations for the logic of preference Rescher (1967)

# The language of machine learning

## Loss functions

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

# The language of machine learning

## Loss functions

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

- 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

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



Specification or theory of preference?

- 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

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



Specification or theory of preference?

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

**Broader goal:** High-level modeling languages for specifying and better understanding LLMs and their algorithms.

# The language of machine learning

## Loss functions

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

## Formalization of preference losses

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

**Broader goal:** High-level modeling languages for specifying and better understanding LLMs and their algorithms.

# The language of machine learning

## Loss functions

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

Going away from these opaque equations

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

**Broader goal:** High-level modeling languages for specifying and better understanding LLMs and their algorithms.

# Preference learning as a discrete reasoning problem

## Loss Function

$$-\log \sigma \left( \log \frac{\pi_{\theta}(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \log \frac{\pi_{\theta}(y_l|x)}{\pi_{\text{ref}}(y_l|x)} \right)$$

# Preference learning as a discrete reasoning problem

## Loss Function

$$-\log \sigma \left( \log \frac{\pi_{\theta}(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \log \frac{\pi_{\theta}(y_l|x)}{\pi_{\text{ref}}(y_l|x)} \right)$$

Two models, four predictions

# Preference learning as a discrete reasoning problem

## Loss Function

$$-\log \sigma \left( \log \frac{\pi_{\theta}(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \log \frac{\pi_{\theta}(y_l|x)}{\pi_{\text{ref}}(y_l|x)} \right)$$

Two models, four 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

## Symbolic Program

```
Implies(  
  And(M(x, yI), Ref(x, yw)),  
  And(M(x, yw), Ref(x, yI))  
)
```

## Loss Function

$$-\log \sigma \left( \log \frac{\pi_{\theta}(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \log \frac{\pi_{\theta}(y_I|x)}{\pi_{\text{ref}}(y_I|x)} \right)$$

High-level model behavior

- **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

## Symbolic Program

```
Implies(  
  And(M(x, yI), Ref(x, yw)),  
  And(M(x, yw), Ref(x, yI))  
)
```

## Loss Function

$$-\log \sigma \left( \log \frac{\pi_{\theta}(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \log \frac{\pi_{\theta}(y_I|x)}{\pi_{\text{ref}}(y_I|x)} \right)$$

Decompilation  $\longleftrightarrow$  Compilation

- **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

## Symbolic Program

```
Implies(  
  And(M(x, yI), Ref(x, yw)),  
  And(M(x, yw), Ref(x, yI))  
)
```

## Loss Function

$$-\log \sigma \left( \log \frac{\pi_{\theta}(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \log \frac{\pi_{\theta}(y_I|x)}{\pi_{\text{ref}}(y_I|x)} \right)$$

Decompilation  $\longleftrightarrow$  Compilation

- **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

## Symbolic Program

```
Implies(  
  And(M(x, yI), Ref(x, yw)),  
  And(M(x, yw), Ref(x, yI))  
)
```

## Loss Function

$$-\log \sigma \left( \log \frac{\pi_{\theta}(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \log \frac{\pi_{\theta}(y_I|x)}{\pi_{\text{ref}}(y_I|x)} \right)$$

Decompilation  $\longleftrightarrow$  Compilation

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

1. **Compilation:** Translating specifications into loss, well studied.

# Preference learning as a discrete reasoning problem

## Symbolic Program

```
Implies(  
  And(M(x, yI), Ref(x, yw)),  
  And(M(x, yw), Ref(x, yI))  
)
```

## Loss Function

$$-\log \sigma \left( \log \frac{\pi_{\theta}(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \log \frac{\pi_{\theta}(y_I|x)}{\pi_{\text{ref}}(y_I|x)} \right)$$

Decompilation  $\longleftrightarrow$  Compilation

- **Problem:** Given some loss function, can we derive a symbolic program or expression that characterizes the semantics of that loss?
  1. **Compilation:** Translating specifications into loss, well studied.
  2. **Decompilation:** Losses to specifications (inverse), less explored.

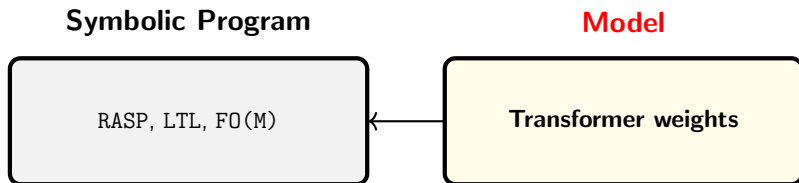
# Formal analysis via decompilation in general

**Model**

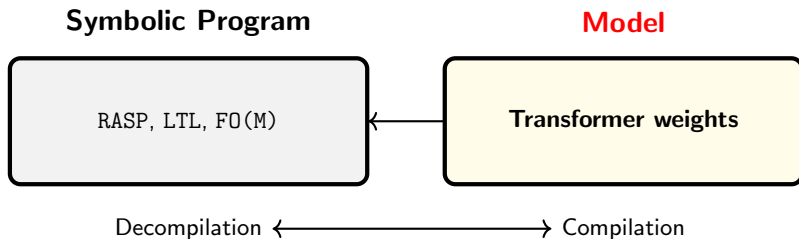
**Transformer weights**



## Formal analysis via decompilation in general



# Formal analysis via decompilation in general



- We know what the *target languages* are (Weiss et al., 2021; Merrill and Sabharwal, 2023; Yang and Chiang, 2024), how to compile, decompile (Friedman et al., 2023).

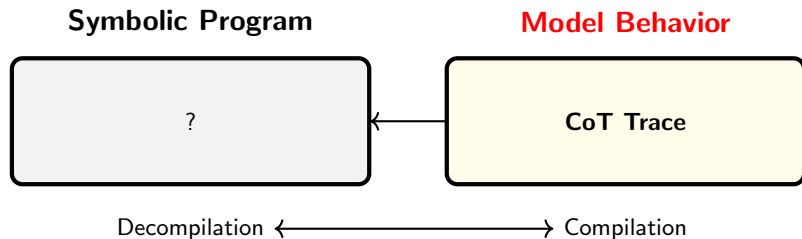
# Formal analysis via decompilation in general

**Model Behavior**

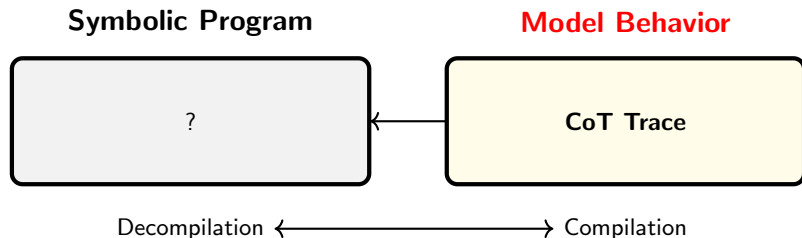


**CoT Trace**

## Formal analysis via decompilation in general



# Formal analysis via decompilation in general



- Not always clear what the target language is or should be.

# Language model programming: ESSLLI 2025

## Lecturers

[Kyle Richardson](#) (Allen Institute for AI)

[Gijs Wijnholds](#) (Leiden Institute of Advanced Computer Science)

## Slides

[lecture 1](#): course overview, language modeling basics, [transformers](#), [RASP](#).

[lecture 2](#): declarative approaches to model training and fine-tuning, the [semantic loss](#) and [weighted model counting](#), [other](#) approaches.

[lecture 3](#): high-level programming techniques for [direct preference alignment](#) and [LLM alignment](#), [formal characterizations](#) of known loss functions.

[lecture 4](#): [declarative and probabilistic approaches](#) to test-time inference, [LLM self-correction](#), [consistency](#), distilling LLMs to tractable models, [logic programming](#).

[lecture 5](#): Advanced prompting, [chain-of-thought](#), [imperative model programming](#), [\(discrete\) probabilistic programming](#).

background [logic notes](#), [extended notes on transformers](#)

extra lectures [Prompting as programming](#), [Grammar-constrained decoding](#)

## Helpful Resources

Below are some pointers to code resources:

- languages [\[scallop\]](#), [\[problog\]](#), [\[pyDataLog\]](#), [\[lmql\]](#), [\[rasp\]](#), [\[NumPy RASP\]](#), [\[deepproblog\]](#)
- automated reasoning tools/circuits [\[Z3 solver\]](#), [\[python-sat\]](#), [\[pysdd\]](#), [\[circuit\]](#)
- NLP and general ML [\[transformers\]](#), [\[PyTorch\]](#), [\[pylon-lib\]](#), [\[hf datasets\]](#), [\[hf hub\]](#)
- other useful utilities [\[sympy\]](#)

Useful tutorials: [Transformers from scratch](#) (some examples/ideas used in lecture 1), [Lectures on Probabilistic Programming](#), [Tractable Probabilistic Models](#)

<https://github.com/yakazimir/LMProgramming>

## Lecturers

[Yury Richardson](#) (Allen Institute for AI)

[Rita Vrijenhoek](#) (Leiden Institute of Advanced Computer Science)

## Slides

[lecture 1](#): course overview, language modeling basics, [transformers](#), [RASP](#)

[lecture 2](#): declarative approaches to model training and fine-tuning, the [cross-entropy loss](#) and [weighted model counting](#), other approaches.

[lecture 3](#): high-level programming techniques for [direct preference alignment](#) and [LLM alignment](#), formal characterizations of known loss functions.

[lecture 4](#): [declarative and probabilistic approaches](#) to test-time inference, [LLM self-correction](#), [consistency](#), distilling LLMs to tractable models, [logic programming](#).

[background](#) [logic](#), [syntax](#), [semantics](#), [losses](#), [on transformers](#)

extra lectures [Formalizing as programming](#), [Quantifier elimination](#), [decoding](#)

## Helpful Resources

Below are some pointers to code resources:

- languages: [pytorch](#), [pytorch-lightning](#), [pytorch-train](#), [pytorch-train](#), [PyTorch Deep Learning](#)
- automated reasoning toolchains: [Z3 solver](#), [python-smt](#), [smtlib](#), [smtlib](#)
- NLP and general ML: [transformers](#), [PyTorch](#), [datautils](#), [NLTK](#), [NLTK](#), [NLTK](#)
- other useful utilities: [pymc3](#)

Useful tutorials: [Transformers from scratch](#) (some examples/ideas used in lecture 1), [Lectures on Probabilistic Programming](#), [Tractable Probabilistic Models](#)

# What is the right programming language for preference?

<https://github.com/yakazimir/LMProgramming>

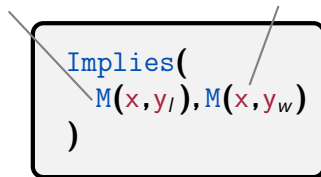
## Declarative models of preference

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

# Declarative models of preference

Model predicts loser

Model predicts winner

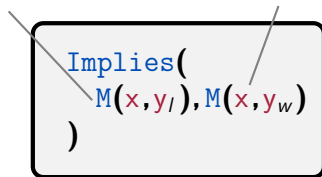


**Conceptually:** Model predications are logical propositions, Boolean variables inside of formulas, weighted by prediction probability.

# Declarative models of preference

Model predicts loser

Model predicts winner



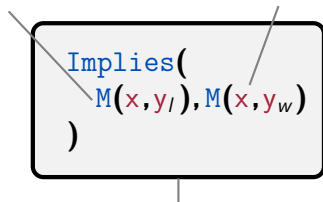
$$w(\mathbf{M}(\mathbf{x}, y)) = \pi_{\mathbf{M}}(y \mid \mathbf{x})$$

**Conceptually:** Model predications are logical propositions, Boolean variables inside of formulas, weighted by prediction probability.

# Declarative models of preference

Model predicts loser

Model predicts winner

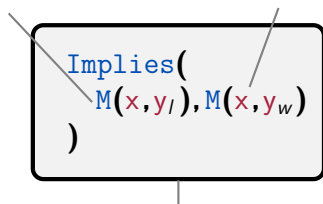


*Whenever the model deems the loser to be a valid generation, it should deem the winner to be valid too.*

**Conceptually:** Predictions are connected through Boolean operators, express constraints on predictions;  $\rho_\theta$  as formulas.

# Uncovering the natural logic of these algorithms

Model predicts loser      Model predicts winner



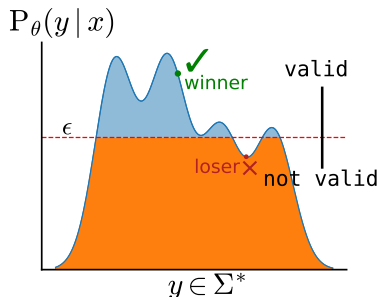
*Whenever the model deems the loser to be a valid generation, it should deem the winner to be valid too.*

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

# Uncovering the natural logic of these algorithms

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

Whenever the model deems the loser to be a valid generation, it should deem the winner to be valid too.

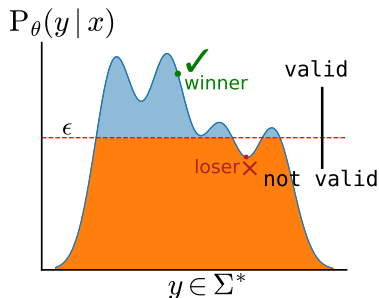


**Assumption:** Every loss function has an internal logic that can be expressed in this way, we want to uncover that logic.

# Uncovering the natural logic of these algorithms

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

```
And(  
  M(x, yw),  
  Not(M(x, yl)))
```

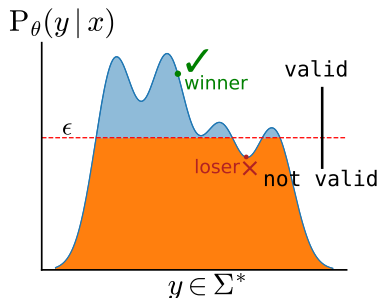


**Assumption:** Every loss function has an internal logic that can be expressed in this way, we want to uncover that logic.

# Uncovering the natural logic of these algorithms

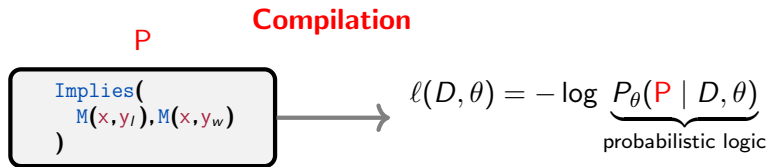
```
Implies(  
  M(x, yl), M(x, yw)  
)
```

```
And(  
  M(x, yw),  
  Not(M(x, yl)))
```



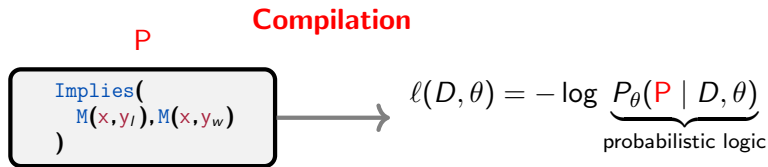
**Observation:** The second program is more strict than the first, involves semantic entailment.

# Compilation and decompilation again



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

# Compilation and decompilation again

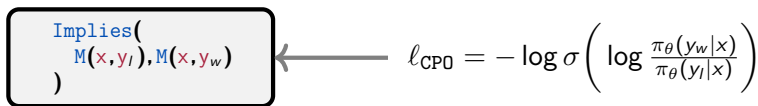


*Whenever the model deems the loser to be a valid generation, it should deem the winner to be valid too.*

**What we did:** defined a novel probabilistic logic for preference modeling,  
**note:** logic useful not only for learning and loss.

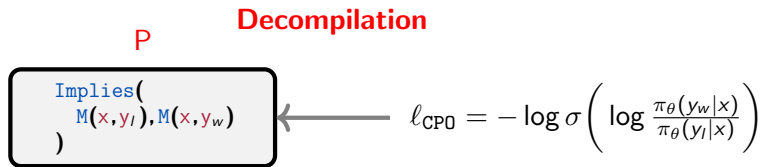
# Compilation and decompilation again

P **Decompilation**



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

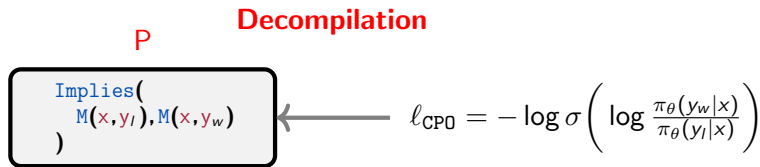
# Compilation and decompilation again



Whenever the model deems the loser to be a valid generation, it should deem the winner to be valid too.

$$\underbrace{\ell_{\text{CPO}}(D, \theta) = -\log P_{\theta}(\mathbf{P} \mid D, \theta)}_{\text{correctness property}}$$

# Compilation and decompilation again



Whenever the model deems the loser to be a valid generation, it should deem the winner to be valid too.

$$\underbrace{\ell_{\text{CPO}}(D, \theta) = -\log P_{\theta}(\mathbf{P} \mid D, \theta)}_{\text{correctness property}}$$

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

# Illustration of approach and results

**Input Loss**  $\ell_{\text{ORPO}}$

$$-\log \sigma \left( \log \frac{\text{Odds}_{\theta}(y_w|x)}{\text{Odds}_{\theta}(y_l|x)} \right)$$

# Illustration of approach and results

**Input Loss**  $\ell_{\text{ORPO}}$

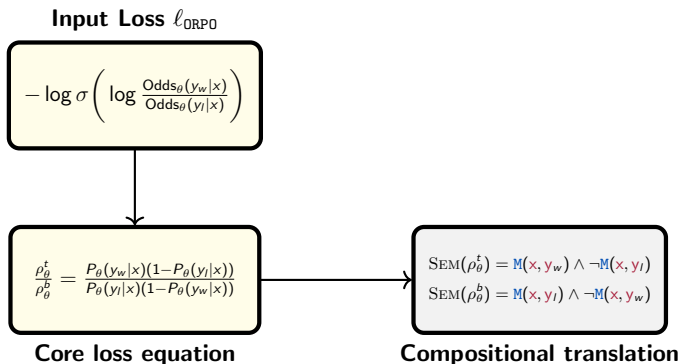
$$-\log \sigma \left( \log \frac{\text{Odds}_{\theta}(y_w|x)}{\text{Odds}_{\theta}(y_l|x)} \right)$$



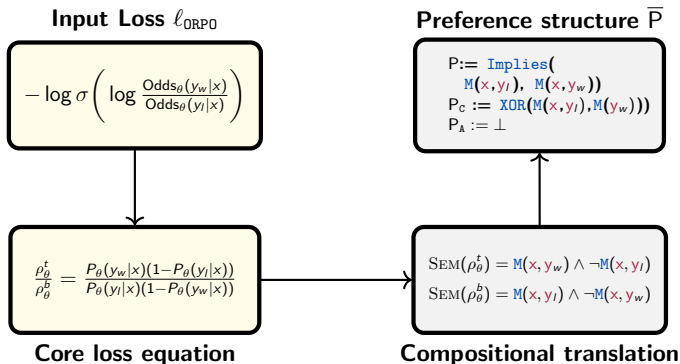
$$\frac{\rho_{\theta}^t}{\rho_{\theta}^b} = \frac{P_{\theta}(y_w|x)(1-P_{\theta}(y_l|x))}{P_{\theta}(y_l|x)(1-P_{\theta}(y_w|x))}$$

**Core loss equation**

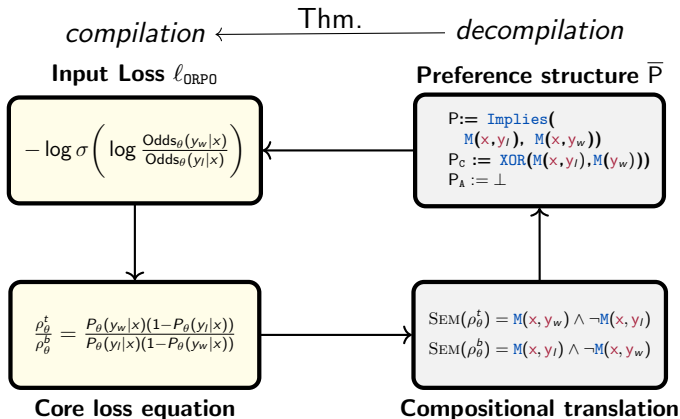
# Illustration of approach and results



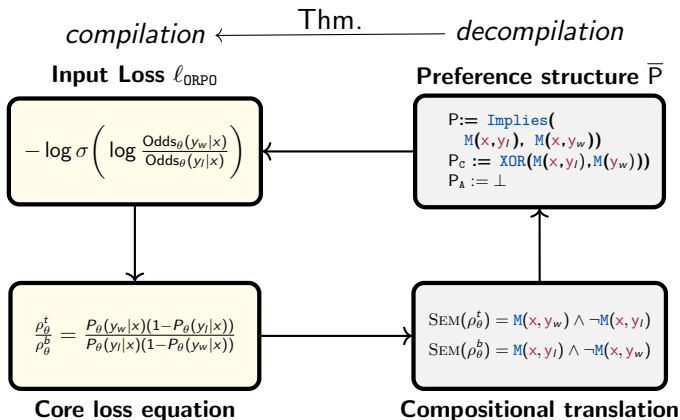
# Illustration of approach and results



# Illustration of approach and results

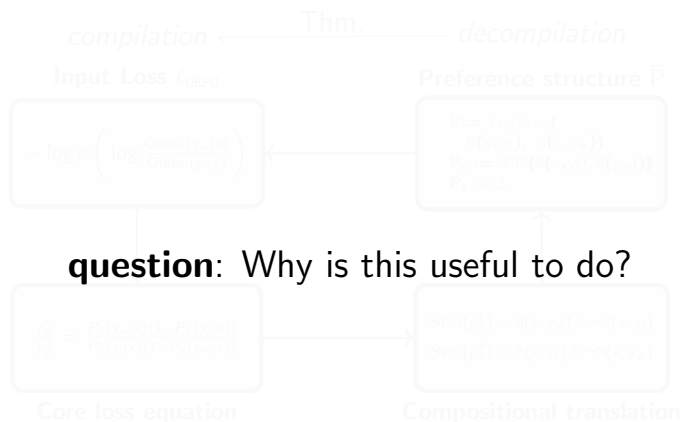


# Illustration of approach and results



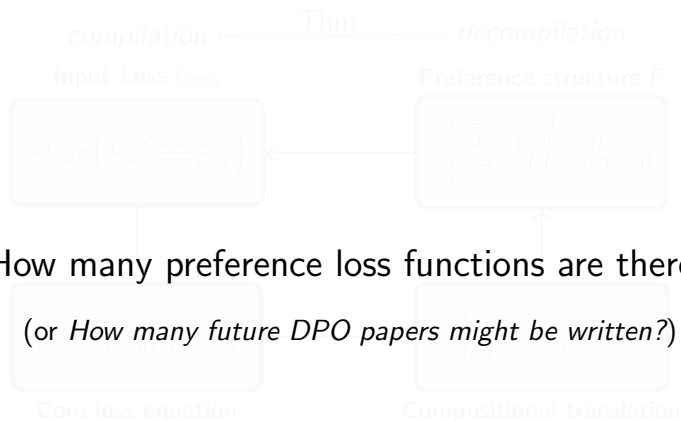
- **Preference structure**, a core construct in our logic, encoding for preference losses, has a natural Boolean interpretation.

# Illustration of approach and results



- Preference structure, a core construct in our logic, encoding for preference losses, has a natural Boolean interpretation.

# Illustration of approach and results

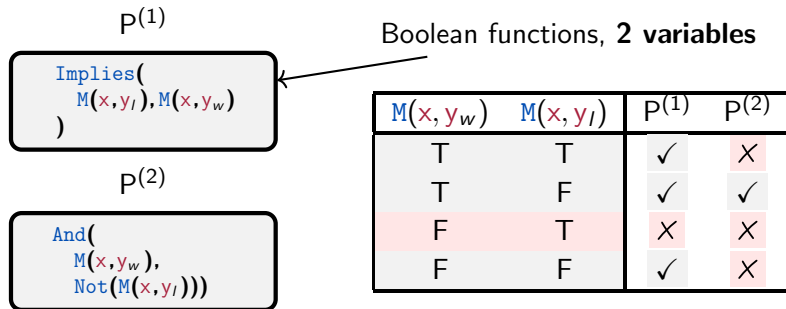


How many preference loss functions are there?

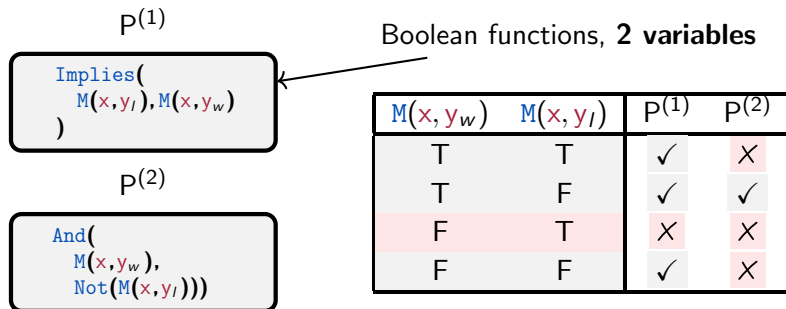
(or *How many future DPO papers might be written?*)

- Preference structure, a core construct in our logic, encoding for preference losses, has a natural Boolean interpretation.

## Why is this useful? understanding the space

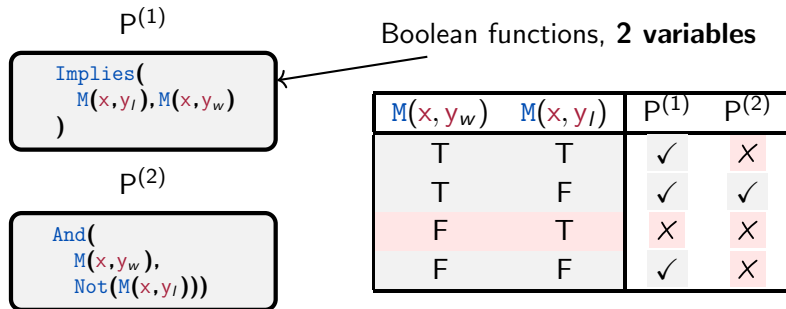


# Why is this useful? understanding the space



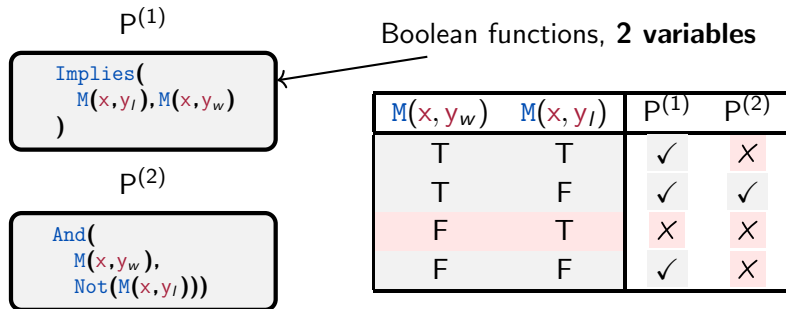
- ▶ Every program (in our logic) is pair of Boolean functions (in  $n$  variables),  
corr. to ✓ and X, leads to  $4^{2^n}$  possible loss functions.

## Why is this useful? understanding the space



Loss creation will end up being equivalent to drawing different sets of ✓ s and X (or blank marks) in a truth table.

## Why is this useful? understanding the space



**no reference:** 256 losses

Loss creation will end up being equivalent to drawing different sets of  
✓ s and X (or blank marks) in a truth table.

# Loss functions as truth tables

```
Implies(  
  And(M(x,yI),Ref(x,yw)),  
  And(M(x,yw),Ref(x,yI))  
)
```

**4 variables**

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

**w/ reference:** 4,294,967,296 losses

# Loss functions as truth tables

```
Implies(  
  And(M(x,yI),Ref(x,yw)),  
  And(M(x,yw),Ref(x,yI))  
)
```

4 variables

**answer: loads.**

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

w/ reference: 4,294,967,296 losses

## Loss functions as truth tables

```
Implies(  
  And(M(x,y_l),Ref(x,y_w)),  
  And(M(x,y_w),Ref(x,y_l))  
)
```

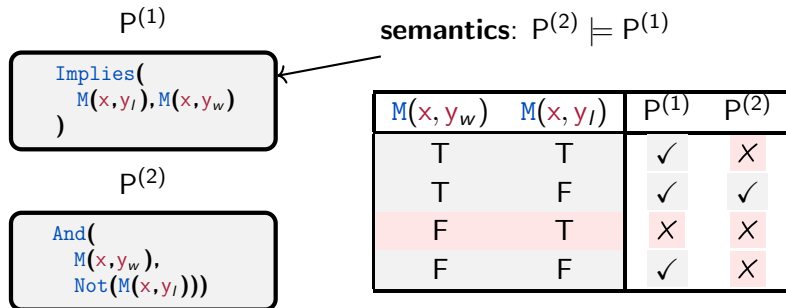
**question:** How are losses related to one another?

4 variables

Ref(x,y_w)	M(x,y_l)	Ref(x,y_l)	M(x,y_w)
F	F	F	F
F	F	F	T
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	T	F
T	F	T	T
T	T	F	F
T	T	F	T
T	T	T	F
T	T	T	T

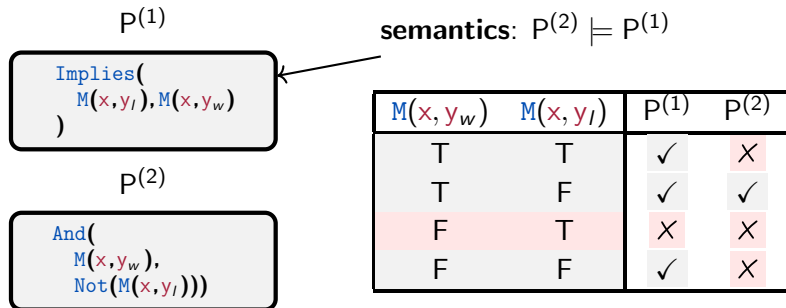
w/ reference: 4,294,967,296 losses

## Why is this useful? understanding the structure



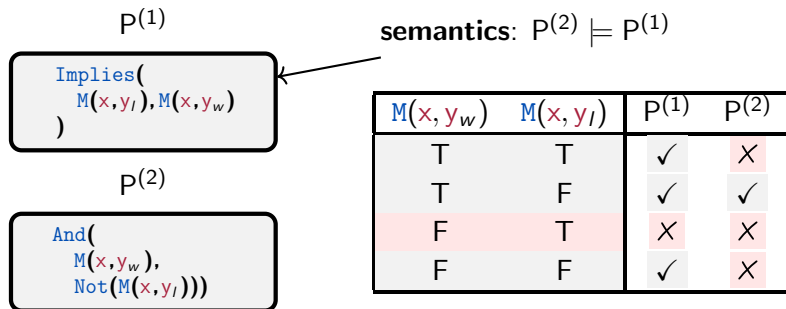
**Proposition** (Xu et al., 2018): Loss behavior is monotonic w.r.t semantic entailment: if  $P^{(2)} \models P^{(1)}$  then  $\ell(D, \theta, P^{(2)}) \geq \ell(D, \theta, P^{(1)})$ .

# Why is this useful? understanding the structure



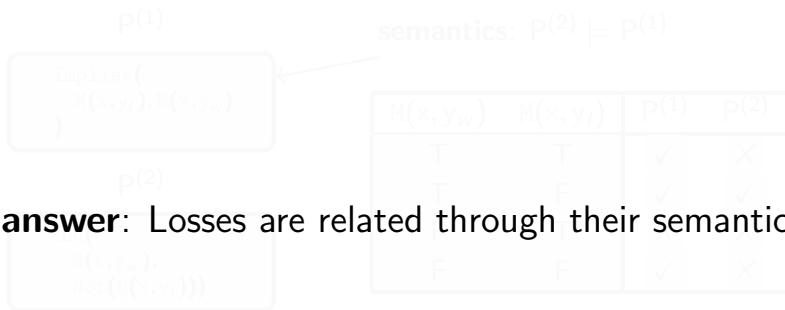
**Proposition** (Xu et al., 2018): Loss is equivalent under semantic equivalence: If  $P^{(2)} \equiv P^{(1)}$  then  $\ell(D, \theta, P^{(2)}) = \ell(D, \theta, P^{(1)})$ .

# Why is this useful? understanding the structure



**Theorem:**  $\ell(D, \theta, P^{(2)}) > \ell(D, \theta, P^{(1)})$  (the loss of  $P^{(1)}$  is contained in the loss of  $P^{(2)}$ ).

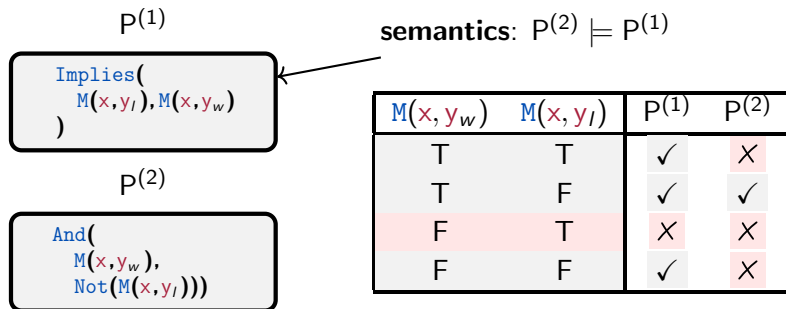
## Why is this useful? understanding the structure



**answer:** Losses are related through their semantics

**Theorem:**  $\ell(D, \theta, P^{(2)}) \geq \ell(D, \theta, P^{(1)})$  (the loss of  $P^{(1)}$  is contained in the loss of  $P^{(2)}$ ).

# Why is this useful? understanding the structure



**Practical strategy:** Start with empirically successful losses, modify semantics (make more or less constrained), then experiment accordingly.

# Deriving new losses symbolically, from first principles

## Symbolic Program

```
Implies(  
  And(M(x, yI), Ref(x, yw)),  
  And(M(x, yw), Ref(x, yI))  
)
```

## DPO Loss

$$-\log \sigma \left( \log \frac{\pi_{\theta}(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \log \frac{\pi_{\theta}(y_I|x)}{\pi_{\text{ref}}(y_I|x)} \right)$$



# Deriving new losses symbolically, from first principles

## Symbolic Program

```
Implies(  
  And(M(x, yI), Ref(x, yw)),  
  And(M(x, yw), Ref(x, yI))  
)
```

↓ modify

```
Implies(  
  And(M(x, yI), Ref(x, yw)),  
  M(x, yw)  
)
```

## DPO Loss

$$-\log \sigma \left( \log \frac{\pi_{\theta}(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \log \frac{\pi_{\theta}(y_I|x)}{\pi_{\text{ref}}(y_I|x)} \right)$$

# Deriving new losses symbolically, from first principles

## Symbolic Program

```
Implies(  
  And(M(x, yI), Ref(x, yw)),  
  And(M(x, yw), Ref(x, yI))  
)
```

modify

```
Implies(  
  And(M(x, yI), Ref(x, yw)),  
  M(x, yw)  
)
```

## DPO Loss

$$-\log \sigma \left( \log \frac{\pi_{\theta}(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \log \frac{\pi_{\theta}(y_I|x)}{\pi_{\text{ref}}(y_I|x)} \right)$$

## Novel loss

$$-\log \sigma \left( \log \frac{\pi_{\theta}(y_w|x)\pi_{\text{ref}}(y_I|x)(1-\pi_{\theta}(y_I|x))}{\pi_{\theta}(y_I|x)\pi_{\text{ref}}(y_w|x)(1-\pi_{\theta}(y_w|x))} \right)$$

# Deriving new losses symbolically, from first principles

## Symbolic Program

```
Implies(  
  And(M(x, yI), Ref(x, yw)),  
  And(M(x, yw), Ref(x, yI))  
)
```

↓ modify

```
Implies(  
  And(M(x, yI), Ref(x, yw)),  
  M(x, yw)  
)
```

## DPO Loss

$$-\log \sigma \left( \log \frac{\pi_{\theta}(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \log \frac{\pi_{\theta}(y_I|x)}{\pi_{\text{ref}}(y_I|x)} \right)$$

## Novel loss

$$-\log \sigma \left( \log \frac{\pi_{\theta}(y_w|x)\pi_{\text{ref}}(y_I|x)(1-\pi_{\theta}(y_I|x))}{\pi_{\theta}(y_I|x)\pi_{\text{ref}}(y_w|x)(1-\pi_{\theta}(y_w|x))} \right)$$

- High-level programming language for defining new losses.

# Deriving new losses symbolically, from first principles

## Symbolic Program

```
Implies(  
  And(M(x,y_l),Ref(x,y_w)),  
  And(M(x,y_w),Ref(x,y_l))  
)
```

## DPO Loss

$$-\log \sigma \left( \log \frac{\pi_{\theta}(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \log \frac{\pi_{\theta}(y_l|x)}{\pi_{\text{ref}}(y_l|x)} \right)$$

**questions:** How does our logic work? What do we see?

```
Implies(  
  And(M(x,y_l),Ref(x,y_w)),  
  M(x,y_w)  
)
```

$$-\log \sigma \left( \log \frac{\pi_{\theta}(y_w|x)\pi_{\text{ref}}(y_l|x)(1-\pi_{\theta}(y_l|x))}{\pi_{\theta}(y_l|x)\pi_{\text{ref}}(y_w|x)(1-\pi_{\theta}(y_w|x))} \right)$$

## How does the logic work? **compilation**

P

$M(x, y_w)$	$M(x, y_l)$	CPO	ORPO	unCPO
T	T	✓ X		✓
T	F	✓	✓	✓
F	T	X	X	X
F	F			✓

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

Whenever the model deems the loser to be a valid generation, it should deem the winner to be valid too.

# How does the logic work? **compilation**

P

$M(x, y_w)$	$M(x, y_l)$	CPO	ORPO	unCPO
T	T			
T	F			
F	T			
F	F			

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

Whenever the model deems the loser to be a valid generation, it should deem the winner to be valid too.

$$\{\checkmark, \text{X}\}_w := \prod_{w \models M(x, y)} \pi_{\theta}(y \mid x) \cdot \prod_{w \models \neg M(x, y)} 1 - \pi_{\theta}(y \mid x)$$

# How does the logic work? **compilation**

		P		
$M(x, y_w)$	$M(x, y_l)$	CPO	ORPO	unCPO
T	T	✓ X		✓
T	F	✓	✓	✓
F	T	X	X	X
F	F			✓











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

Whenever the model deems the loser to be a valid generation, it should deem the winner to be valid too.

$$\{\checkmark, X\}_w := \prod_{w \models M(x, y)} \pi_\theta(y | x) \cdot \prod_{w \models \neg M(x, y)} 1 - \pi_\theta(y | x)$$

- Formula probability computed as a weighted count  $\sum \checkmark_w$  (Chavira and Darwiche, 2008), loss is  $-\log$ , *semantic loss* (Xu et al., 2018).

# How does the logic work? **compilation**

$M(x, y_w)$	$M(x, y_l)$	CPO	ORPO	unCPO
T	T	 		
T	F			
F	T			
F	F			

P

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

Whenever the model deems the loser to be a valid generation, it should deem the winner to be valid too.

$$\{\checkmark, \text{X}\}_w := \prod_{w \models M(x, y)} \pi_\theta(y \mid x) \cdot \prod_{w \models \neg M(x, y)} 1 - \pi_\theta(y \mid x)$$

$$\underbrace{\ell_x}_{\text{column}} := -\log \sigma \left( \underbrace{\log \frac{\sum \checkmark}{\sum \text{X}}}_{\text{arbitrary } \text{X}_w} \right)$$

# How does the logic work? **compilation**

$M(x, y_w)$	$M(x, y_l)$	CPO	ORPO	unCPO
T	T	✓ X		✓
T	F	✓	✓	✓
F	T	X	X	X
F	F			✓

P

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

Whenever the model deems the loser to be a valid generation, it should deem the winner to be valid too.

$$\{\textcolor{gray}{\checkmark}, \textcolor{pink}{X}\}_w := \prod_{w \models M(x, y)} \pi_\theta(y \mid x) \cdot \prod_{w \models \neg M(x, y)} 1 - \pi_\theta(y \mid x)$$

$$\underbrace{\ell_x}_{\text{column}} := -\log \sigma \left( \log \frac{\sum \textcolor{gray}{\checkmark}_w}{\sum \textcolor{pink}{X}_w} \right)$$

$$= \underbrace{-\log \sigma \left( \log \frac{\pi_\theta(y_w \mid x)(1 - \pi_\theta(y_l \mid x))}{\pi_\theta(y_l \mid x)(1 - \pi_\theta(y_w \mid x))} \right)}_{\ell_{\text{ORPO}}, P_\theta(P \mid \text{one hot})}$$

# How does the logic work? **compilation**

		P		
$M(x, y_w)$	$M(x, y_l)$	CPO	ORPO	unCPO
T	T	✓ X		✓
T	F	✓	✓	✓
F	T	X	X	X
F	F			✓

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

Whenever the model deems the loser to be a valid generation, it should deem the winner to be valid too.

$$\{\checkmark, X\}_w := \prod_{w \models M(x, y)} \pi_\theta(y \mid x) \cdot \prod_{w \models \neg M(x, y)} 1 - \pi_\theta(y \mid x)$$

$$\underbrace{\ell_x}_{\text{column}} := -\log \sigma \left( \log \frac{\sum_w \checkmark_w}{\sum_w X_w} \right)$$

$$= \underbrace{-\log \sigma \left( \log \frac{\pi_\theta(y_w \mid x)}{\pi_\theta(y_l \mid x)} \right)}_{\ell_{\text{CPO}}, \sim P_\theta(P \mid \text{one true})}$$

# How does the logic work? compilation

		P		
$M(x, y_w)$	$M(x, y_l)$	CPO	ORPO	unCPO
T	T	✓ X		✓
T	F	✓	✓	✓
F	T	X	X	X
F	F			✓

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

Whenever the model deems the loser to be a valid generation, it should deem the winner to be valid too.

**observation:** losses differ in hard constraints

$$\underbrace{\ell_x}_{\text{column}} := -\log \sigma \left( \log \frac{\sum_w \checkmark_w}{\sum_w X_w} \right)$$

$$= \underbrace{-\log \sigma \left( \log \frac{\pi_\theta(y_w | x)}{\pi_\theta(y_l | x)} \right)}_{\ell_{\text{CPO}}, \sim P_\theta(P | \text{one true})}$$

# How does the logic work? compilation

		P		
$M(x, y_w)$	$M(x, y_l)$	CPO	ORPO	unCPO
T	T	✓ X		✓
T	F	✓	✓	✓
F	T	X	X	X
F	F			✓

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

Whenever the model deems the loser to be a valid generation, it should deem the winner to be valid too.

$$\{\checkmark, X\}_w := \prod_{w \models M(x, y)} \pi_\theta(y \mid x) \cdot \prod_{w \models \neg M(x, y)} 1 - \pi_\theta(y \mid x)$$

Loss	Representation $\bar{P}$
CE	$P := M(x, y_w), P_C := \perp$
CEUnl	$P := \text{And}(M(x, y_w), \text{Not}(M(x, y_l)))$ $P_C := \perp$
CPO	$;;$ core semantic formula $P := \text{Implies}(M(x, y_l), M(x, y_w))$ $;;$ one-true constraint $P_C := \text{Or}(M(x, y_l), M(x, y_w))$
ORPO	$P := \text{Implies}(M(x, y_l), M(x, y_w))$ $;;$ one-hot constraint $P_C := \text{XOR}(M(x, y_l), M(x, y_w))$

# How does the logic work? **compilation**

		P		
$M(x, y_w)$	$M(x, y_l)$	CPO	ORPO	unCPO
T	T	✓ X		✓
T	F	✓	✓	✓
F	T	X	X	X
F	F			✓

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

Whenever the model deems the loser to be a valid generation, it should deem the winner to be valid too.

$$\{\checkmark, X\}_w := \prod_{w \models M(x, y)} \pi_\theta(y \mid x) \cdot \prod_{w \models \neg M(x, y)} 1 - \pi_\theta(y \mid x)$$

- **Preference structure:** equivalent way of expressing truth table representations (Richardson et al., 2025),

$$\bar{P} := \left( \underbrace{P}_{\text{core}}, \underbrace{P_C, P_A}_{\text{constraints}} \right)$$

# How does the logic work? **compilation**

$M(x, y_w)$	$M(x, y_l)$	CPO	ORPO	unCPO
T	T	✓ X		✓
T	F	✓	✓	✓
F	T	X	X	X
F	F			✓

P

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

Whenever the model deems the loser to be a valid generation, it should deem the winner to be valid too.

$$\{\checkmark, X\}_w := \prod_{w \models M(x, y)} \pi_{\theta}(y \mid x) \cdot \prod_{w \models \neg M(x, y)} 1 - \pi_{\theta}(y \mid x)$$

$$\underbrace{\ell_x}_{\text{column}} := -\log \sigma \left( \log \frac{\sum \checkmark_w}{\sum X_w} \right)$$

$$= -\log \sigma \left( \log \frac{\pi_{\theta}(y_l \mid x) \pi_{\theta}(y_w \mid x) + (1 - \pi_{\theta}(y_l \mid x))}{\pi_{\theta}(y_l \mid x) (1 - \pi_{\theta}(y_w \mid x))} \right)$$

novel loss without constraints,  $P_{\theta}(P \mid T)$

# How does the logic work? compilation

$M(x, y_w)$	$M(x, y_l)$	CPO	ORPO	unCPO
T	T	✓ X		✓
T	F	✓	✓	✓
F	T	X	X	X
F	F			✓

P

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

Whenever the model deems the loser to be a valid generation, it should deem the winner to be valid too.

**note:**  $M(x, y_l) \rightarrow M(x, y_w) \equiv \neg M(x, y_l) \vee M(x, y_w)$

$\underbrace{M(x, y_l)}_{w \models M(x, y)}$ 
 $\underbrace{M(x, y_w)}_{w \models \neg M(x, y)}$

$$\underbrace{\ell_x}_{\text{column}} := -\log \sigma \left( \log \frac{\sum_w \checkmark_w}{\sum_w X_w} \right)$$

$$= \underbrace{-\log \sigma \left( \log \frac{\pi_\theta(y_l | x) \pi_\theta(y_w | x) + (1 - \pi_\theta(y_l | x))}{\pi_\theta(y_l | x) (1 - \pi_\theta(y_w | x))} \right)}_{\text{novel loss without constraints, } P_\theta(P|T)}$$

# What properties do real losses have?

$M(x, y_w)$	$M(x, y_l)$	CPO	ORPO	unCPO
T	T	✓ X		✓
T	F	✓	✓	✓
F	T	X	X	X
F	F			✓

P

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

Whenever the model deems the loser to be a valid generation, it should deem the winner to be valid too.

$$\{\checkmark, X\}_w := \prod_{w \models M(x, y)} \pi_\theta(y \mid x) \cdot \prod_{w \models \neg M(x, y)} 1 - \pi_\theta(y \mid x)$$

$$\underbrace{\ell_x}_{\text{column}} := -\log \sigma \left( \log \frac{\sum \checkmark_w}{\sum X_w} \right)$$

$$= \underbrace{-\log \sigma \left( \log \frac{\pi_\theta(y_l \mid x) \pi_\theta(y_w \mid x) + (1 - \pi_\theta(y_l \mid x))}{\pi_\theta(y_l \mid x) (1 - \pi_\theta(y_w \mid x))} \right)}_{\text{novel loss without constraints, } P_\theta(P \mid T)}$$

# What properties do real losses have?

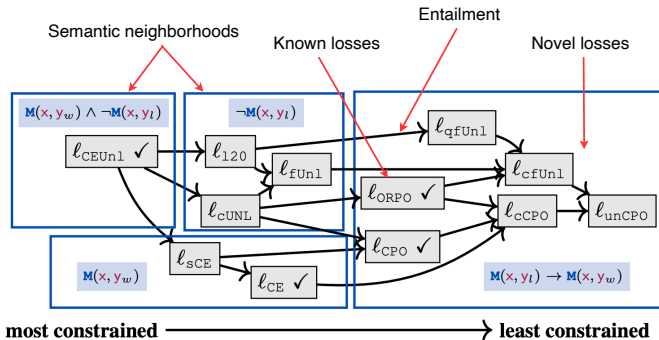
$M(x, y_w)$	$M(x, y_l)$	CPO	ORPO	unCPO	P
T	T	✓ X		✓	<div>                     Implies(  <math>M(x, y_l), M(x, y_w)</math>                      )                 </div> <p>Whenever the model deems the loser to be a <u>valid</u> generation, it should deem the winner to be <u>valid</u> too.</p>
T	F	✓	✓	✓	
F	T	X	X	X	
F	F			✓	

Mapping out these loss spaces semantically

$$\underbrace{\ell_x}_{\text{column}} := -\log \sigma \left( \log \frac{\sum_w \text{✓}_w}{\sum_w \text{X}_w} \right)$$

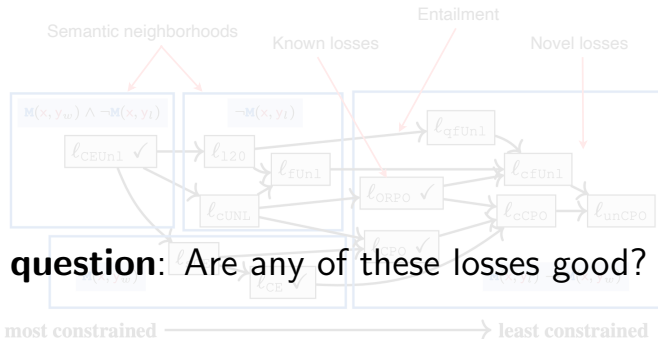
$$= \underbrace{-\log \sigma \left( \log \frac{\pi_\theta(y_l | x) \pi_\theta(y_w | x) + (1 - \pi_\theta(y_l | x))}{\pi_\theta(y_l | x) (1 - \pi_\theta(y_w | x))} \right)}_{\text{novel loss without constraints, } P_\theta(P|T)}$$

# The no reference loss landscape

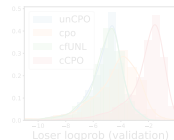
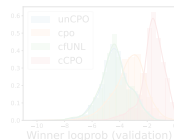


- **Loss lattice:** semantic structure of space, ordering.

# The no reference loss landscape

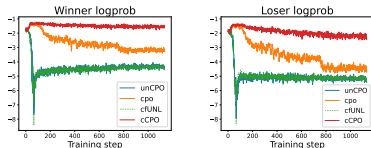
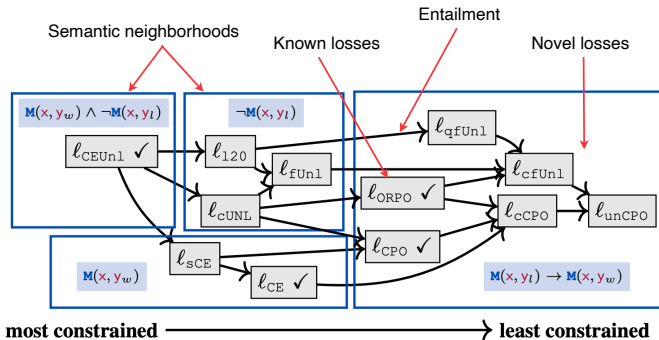


Training dynamics

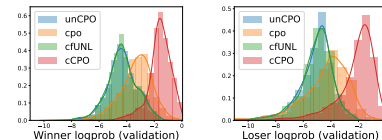


Inference

# The no reference loss landscape

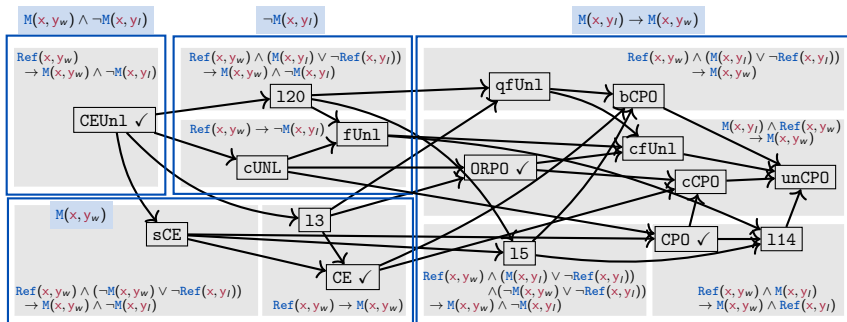


Training dynamics



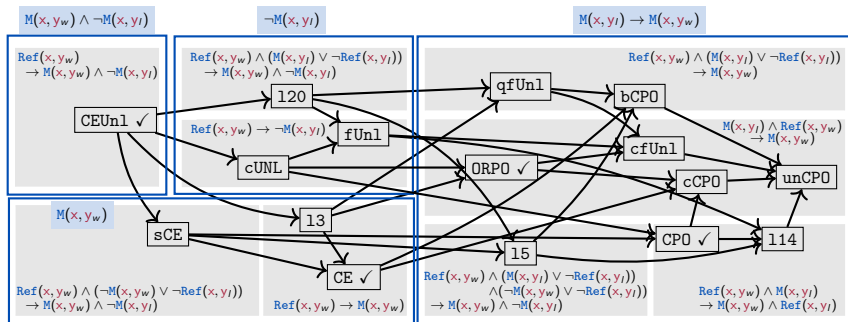
Inference

# The full landscape, reference approaches



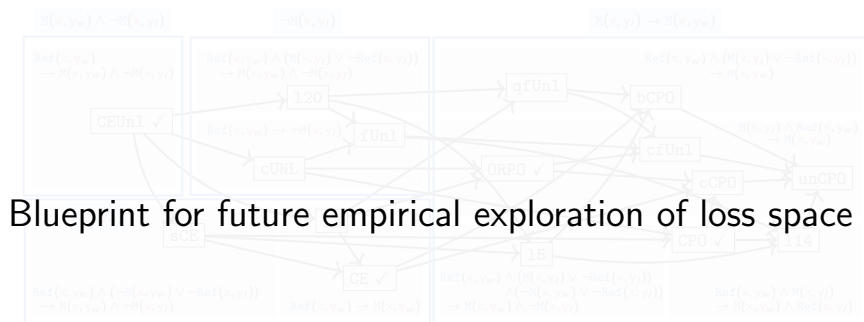
- The semantics of DPO-style reference losses can be straightforwardly computed from no reference approaches.

# The full landscape, reference approaches



- Many new losses to explore and experiment with!

# The full landscape, reference approaches



# Conclusions

- ▶ New ideas about using symbolic techniques to formally characterize the semantics of LLM algorithms, preference learning.

# Conclusions

- ▶ New ideas about using symbolic techniques to formally characterize the semantics of LLM algorithms, preference learning.
  1. Understanding the full space of loss functions (finding: it's a huge space, many novel variations yet to be explored)
  2. Understanding the structure of the space and relationships between different losses (finding: tied to the semantics of the losses).

# Conclusions

- ▶ New ideas about using symbolic techniques to formally characterize the semantics of LLM algorithms, preference learning.
  1. Understanding the full space of loss functions (finding: it's a huge space, many novel variations yet to be explored)
  2. Understanding the structure of the space and relationships between different losses (finding: tied to the semantics of the losses).

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

# Conclusions

- ▶ New ideas about using symbolic techniques to formally characterize the semantics of LLM algorithms, preference learning.
  1. Understanding the full space of loss functions (finding: it's a huge space, many novel variations yet to be explored)
  2. Understanding the structure of the space and relationships between different losses (finding: tied to the semantics of the losses).

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

- ▶ **many other areas to look at:** *analysis of transformers, semantics of data, reinforcement learning, chain-of-thought, LLM agents ...*

Thank you.

## Adding a reference model

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

Whenever the model being tuned deems the loser to be a valid generation and the reference model deems the winner to be valid, the tuned model should deem the winner to be valid too, and *the reference should deem the loser to be valid.*

## Adding a reference model

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

Whenever the model being tuned deems the loser to be a valid generation and the reference model deems the winner to be valid, the tuned model should deem the winner to be valid too, and *the reference should deem the loser to be valid.*

- **Peculiar semantics**, but the logic makes sense, e.g., we want to maximize

$$\sigma \left( \log \frac{\pi_{\theta}(y_w | x)}{\pi_{\theta}(y_l | x)} - \log \frac{\pi_{\text{ref}}(y_w | x)}{\pi_{\text{ref}}(y_l | x)} \right)$$

negating left side of implication (i.e., making  $M(x, y_l)$  and  $Ref(x, y_w)$  false) and making the right side true is logical.

# References I

- 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. *Proceedings of EMNLP*.
- 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 llm 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*.
- Friedman, D., Wettig, A., and Chen, D. (2023). Learning transformer programs. *Advances in Neural Information Processing Systems*, 36:49044–49067.
- Jeffrey, R. C. (1965). *The logic of decision*. University of Chicago press.
- 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.
- Meng, Y., Xia, M., and Chen, D. (2024). Simpo: Simple preference optimization with a reference-free reward. *arXiv preprint arXiv:2405.14734*.
- Merrill, W. and Sabharwal, A. (2023). A logic for expressing log-precision transformers. *Advances in neural information processing systems*, 36:52453–52463.

# References II

- Rescher, N. (1967). *The logic of decision and action*. University of Pittsburgh Pre.
- Richardson, K., Srikumar, V., and Sabharwal, A. (2025). Understanding the logic of direct preference alignment through logic. *Proceedings of ICML*.
- Weiss, G., Goldberg, Y., and Yahav, E. (2021). Thinking like transformers. In *International Conference on Machine Learning*, pages 11080–11090. PMLR.
- 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, A. and Chiang, D. (2024). Counting like transformers: Compiling temporal counting logic into softmax transformers. *arXiv preprint arXiv:2404.04393*.
- Yang, R., Chen, J., Zhang, Y., Yuan, S., Chen, A., Richardson, K., Xiao, Y., and Yang, D. (2025). Selfgoal: Your language agents already know how to achieve high-level goals. *Proceedings of NAACL*.
- 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. *Proceedings of ACL*.