

Language Model Programming: Lecture 3

Kyle Richardson^α, Gijs Wijnholds^β

Allen Institute for AI (AI2)^α
Leiden Institute of Advanced Computer Science (LIACS)^β

July 2025

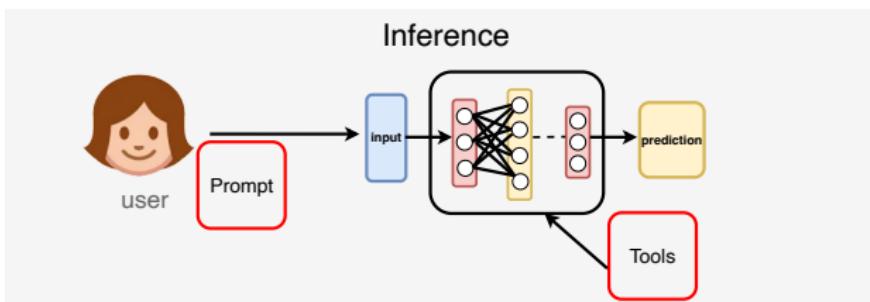
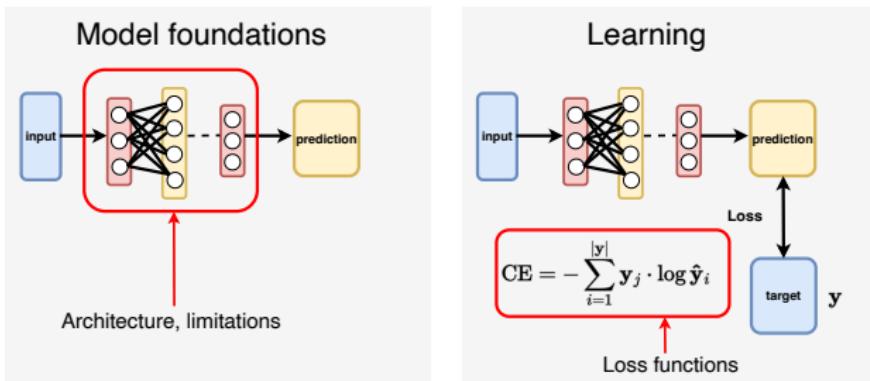


Leiden Institute of
Advanced Computer
Science

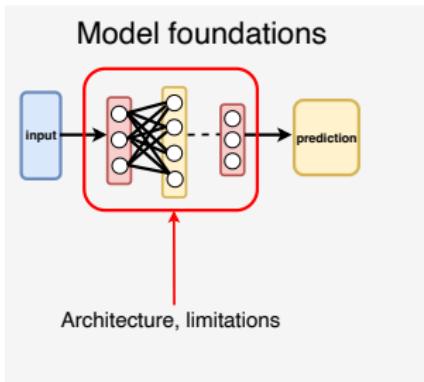


Universiteit
Leiden

Reminder of bigger picture



Reminder of bigger picture



RASPy

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

Input h e l l o

1	1	1	1	1	1
1					
1					
1					
1					
5	5	5	5	5	5

Layer 1

0	1	2	3	4	h
3					
2					
1					
0					

Layer 2

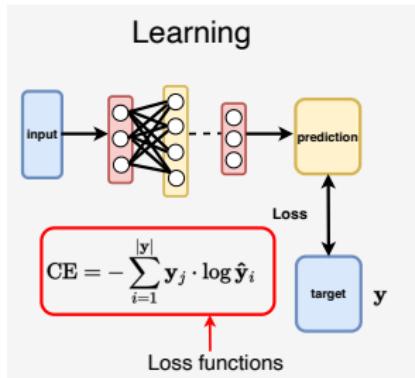
Final o l l e h

Reminder of bigger picture

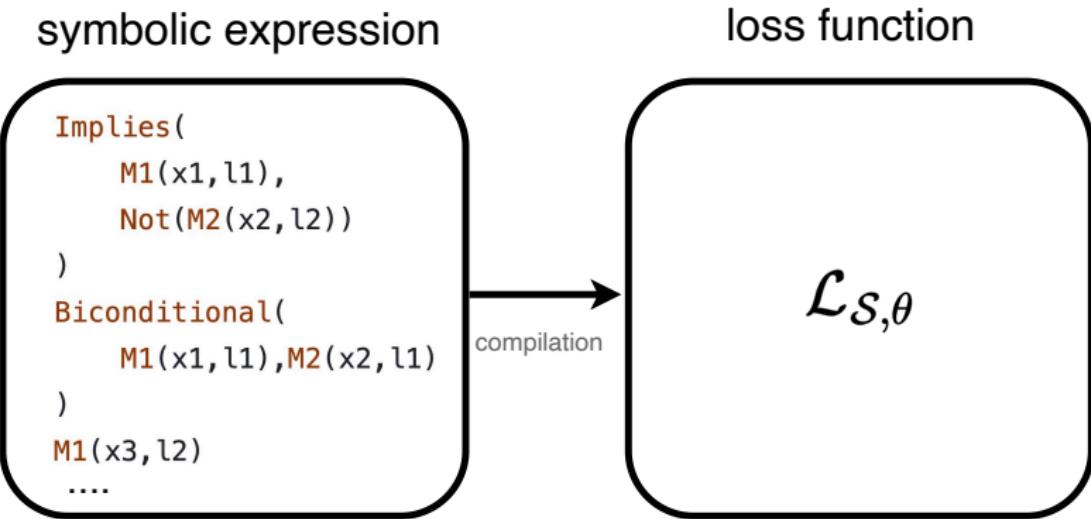
Declarative Programming

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

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



Reminder of bigger picture



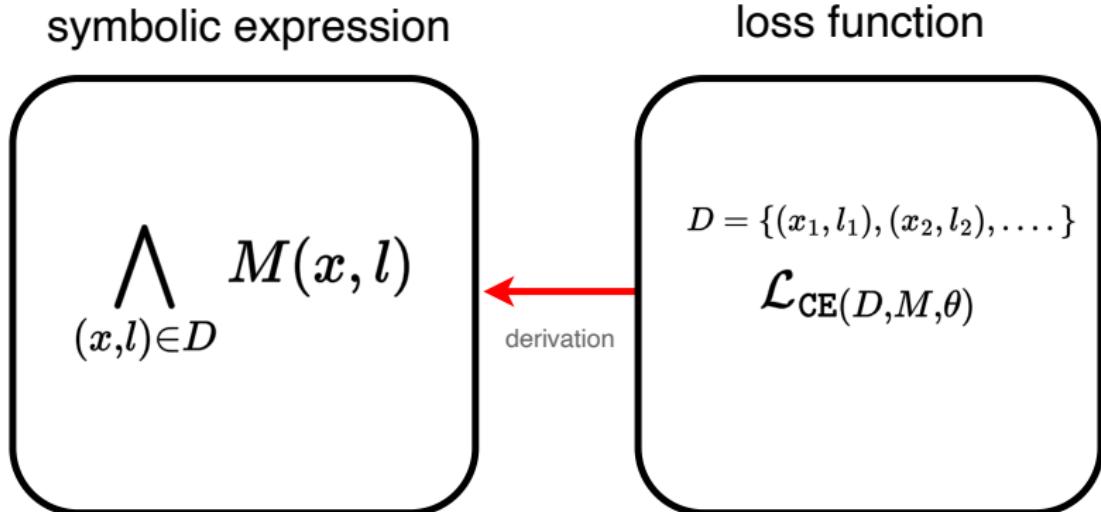
- ▶ **Declarative model programming**, compiling symbolic specifications of model behavior into loss, task specific **model fine-tuning**.

Task specific model fine-tuning

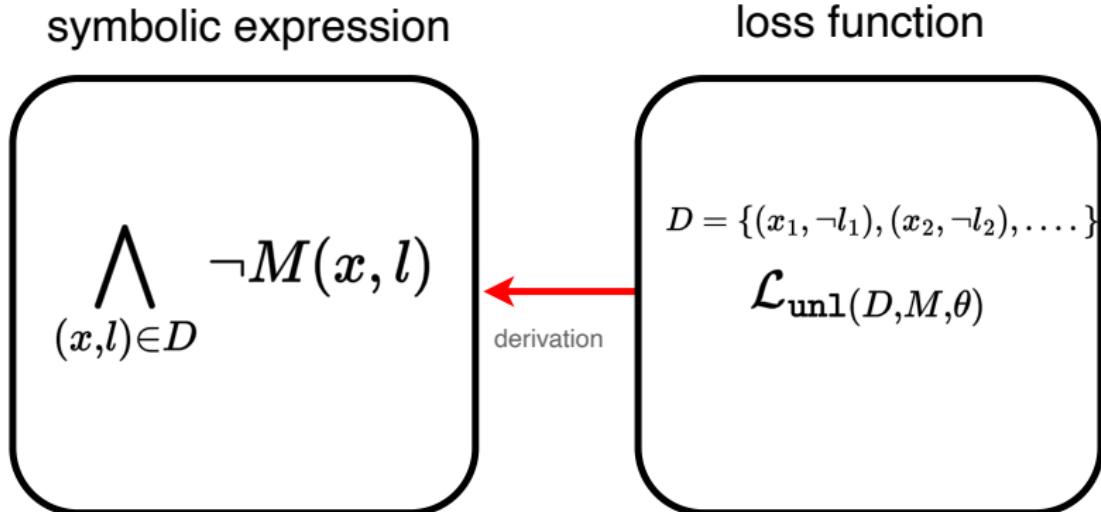
```
1 import torch
2 from transformers import \
3     AutoModelForSequenceClassification as hf_model
4
5 class NLIModel(torch.nn.Module):
6     def __init__(self,model_name):
7         super().__init__()
8         self.model = hf_model.from_pretrained(
9             model_name, ## base transformer
10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

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

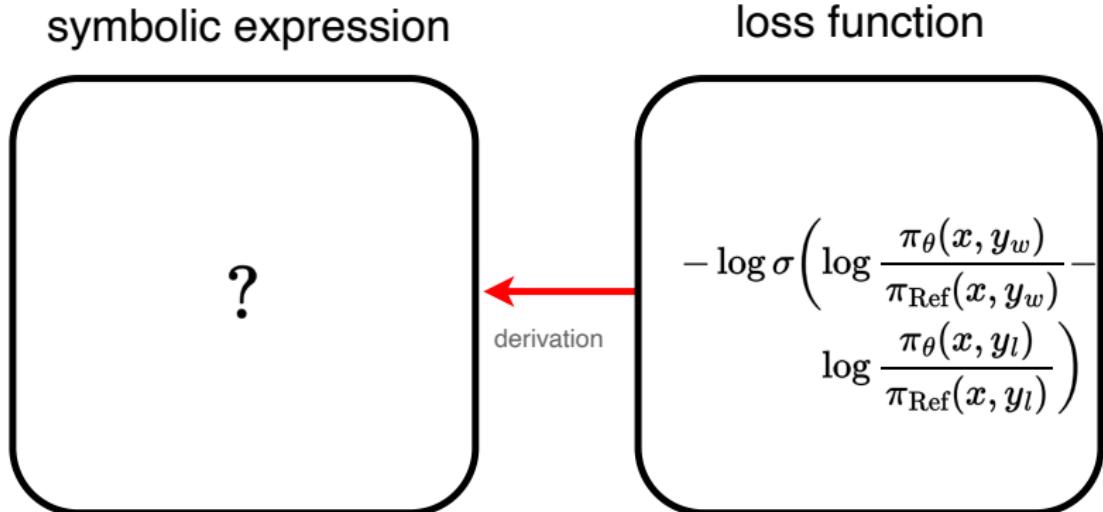
Deriving the semantics of known loss functions



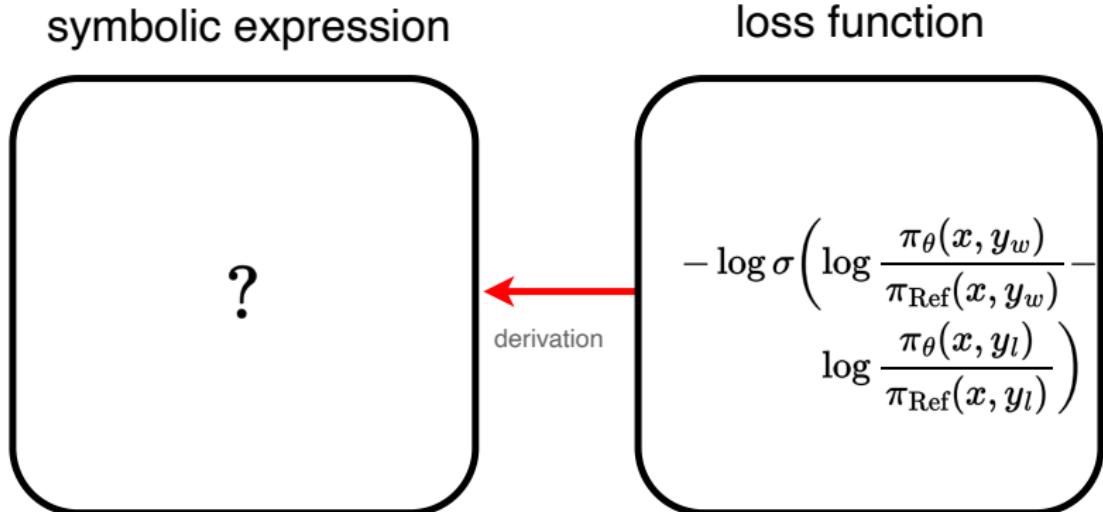
Deriving the semantics of known loss functions



Deriving the semantics of known loss functions



Goals for today



- ▶ **Decompiling** known loss functions into symbolic programs.
- ▶ High-level programming for studying **preference alignment** algorithms.
- ▶ **next lecture:** Test-time inference via probabilistic reasoning.

Background reading

Understanding the Logic of Direct Preference Alignment through Logic

Kyle Richardson¹ Vivek Srikanth² Ashish Sabharwal¹

Abstract

Recent direct preference alignment algorithms (DPA), such as DPO, have shown great promise in aligning large language models to human preferences. While this has motivated the development of many new variants of the original DPO loss, understanding the differences between these recent proposals, as well as developing new DPA loss functions, remains difficult given the lack of a technical and conceptual framework for reasoning about the underlying semantics of these algorithms. In this paper, we attempt to remedy this by formalizing DPA losses in terms of discrete reasoning problems. Specifically, we ask: *Given an existing DPA loss, can we systematically derive a symbolic program that characterizes its semantics?* We propose a novel formalism for characterizing preference losses for single model and reference model based approaches, and identify symbolic forms for a number of commonly used DPA variants. Further, we show how this formal view of preference learning sheds new light on both the size and structure of the DPA loss landscape, making it possible to not only rigorously characterize the relationships between recent loss proposals but also to systematically explore the landscape and derive new loss functions from first principles. We hope our framework and findings will help provide useful guidance to those working on human AI alignment.

1. Introduction

Symbolic logic has long served as the de-facto language for expressing complex knowledge throughout computer science (Halpern et al., 2001), including in AI (McCarthy et al., 1960; Nilsson, 1991) and early ML (McCulloch & Pitts, 1943), owing to its clean semantics. Symbolic approaches to

¹Allen Institute for AI, University of Utah. Correspondence to: Kyle Richardson <kyle@allenai.org>.

Proceedings of the 42nd International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

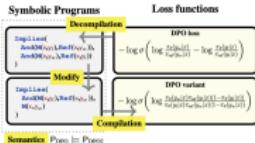


Figure 1. Can we uncover the hidden logic of DPO? Here we show the decomposition of the DPO loss into a symbolic expression that expresses its high-level model behavior, along with a semantically modified version that we can compile into a novel DPO variant. We study how to translate between these two spaces to better understand the semantics of existing preference learning algorithms and to derive new ones from first principles.

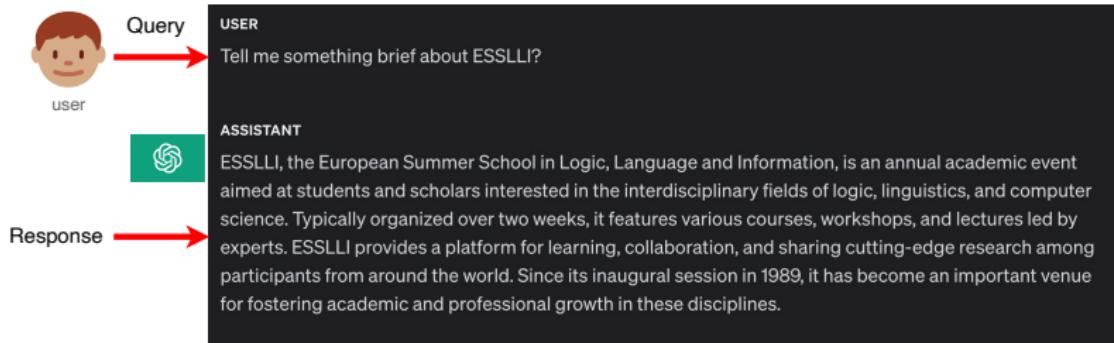
reasoning that are driven by declarative knowledge, in sharp contrast to purely machine learning-based approaches, have the advantage of allowing us to reason transparently about the behavior and correctness of the resulting systems. In this paper we focus on the broad question: *Can the declarative approach be leveraged to better understand and formally specify algorithms for large language models (LLMs)?*

We specifically investigate **direct preference optimization** (DPO) algorithms, such as direct preference optimization (DPO), Rafailov et al., 2023), for pairwise preference learning, which are currently at the forefront of research on LLM alignment and learning from human preferences (Ouyang et al., 2022; Wang et al., 2023). While there has been much recent work on algorithmic variations of DPO (Azar et al., 2024; Hong et al., 2024; Meng et al., 2024) that modify or add new terms to the original loss, understanding the differences between these new proposals, as well as coming up with new variants, remains a formidable challenge due to the lack of a conceptual and technical framework for reasoning about their underlying semantics.

Our study attempts to remedy this problem by formalizing the corresponding loss functions in terms of logic, trying to answer the question: *Given an existing loss function, such as DPO (see Figure 1), can we derive a symbolic expression that captures the core semantics of that loss function (i.e., one that we can then systematically compile back into*

Questions about last lecture?

General purpose (large) language models

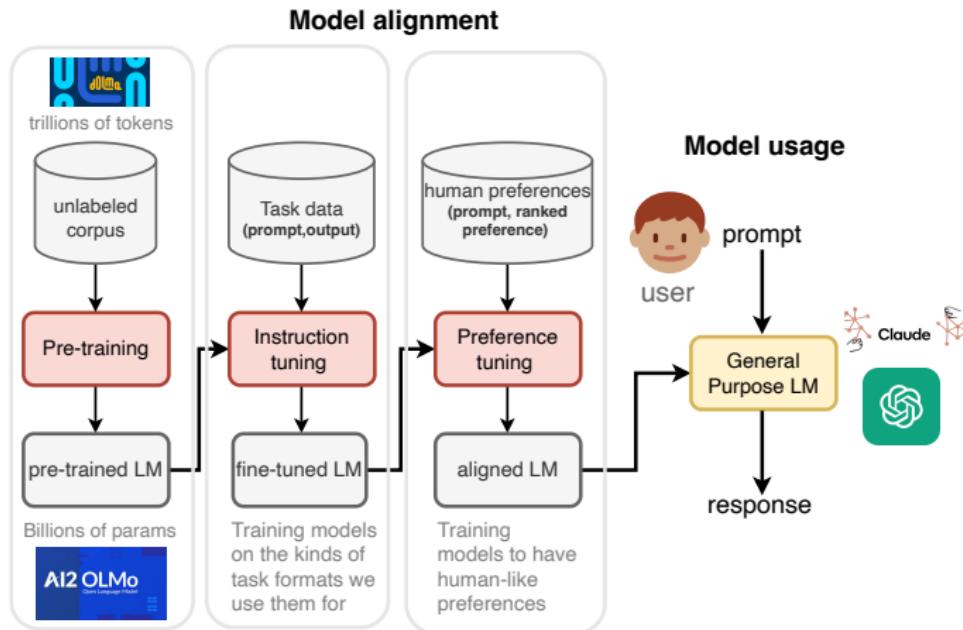


- ▶ Decoder-only models trained at very large scales, used directly via *prompts* and *in-context learning*.

AI2 OLMo models (<https://playground.allenai.org/>)

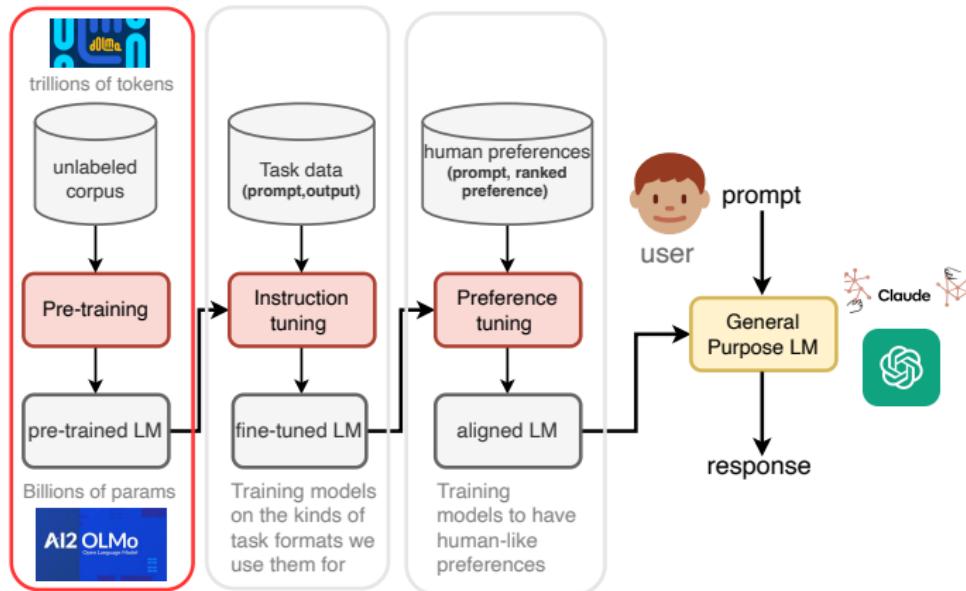
The screenshot shows the AI2 OLMo playground interface. At the top, there's a navigation bar with back, forward, and search icons, and the URL https://playground.allenai.org/. The main header features a stylized logo and the word "Playground". Below the header, a "New chat" button is visible. On the left sidebar, there are links for "Thread history" and "FAQ". A "Log in" button is located at the bottom of the sidebar. The main content area has a "Model:" dropdown set to "OLMo 213B Instruct". A note below the dropdown states: "AI2 Playground is a free scientific and educational tool and by using it you agree to AI2's [Terms of use](#), [Privacy policy](#), and [Responsible use guidelines](#). This site is protected by reCAPTCHA and the Google [Privacy Policy](#) and [Terms of Service](#) apply." A search bar contains the query "Can you tell me about the ESSLLI summer school? Be brief.". The response is provided by the model: "The ESSLLI (European Summer School in Logic, Language and Information) is an annual summer school that provides an intensive and interdisciplinary program focusing on theoretical and computational linguistics, logic, and their applications in computer science. The school brings together students and researchers to present and discuss the latest research in these areas. It aims to foster collaboration and interdisciplinary understanding among participants from various academic backgrounds. The event usually lasts for about three weeks, including lectures, workshops, and social activities, and is known for its high intellectual level and stimulating environment. Participation is typically by invitation, with slots for a limited number of students and postdocs from around the world." Below the response are several interaction icons: a thumbs up, a thumbs down, a reply arrow, a refresh, a square, and a document icon labeled "Show OLMoTrace". At the bottom, there's a "Reply to the model" input field with a microphone icon, and a footer note: "Always fact-check your results. OLMo is primarily designed to handle English queries."

General purpose (large) language models: how?



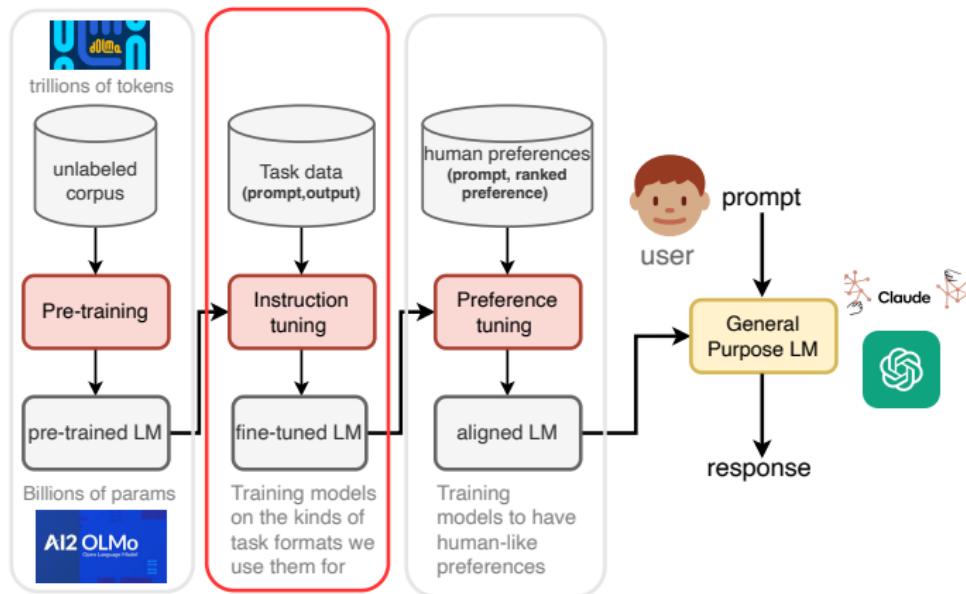
- ▶ Often trained via three stages:

General purpose (large) language models: how?



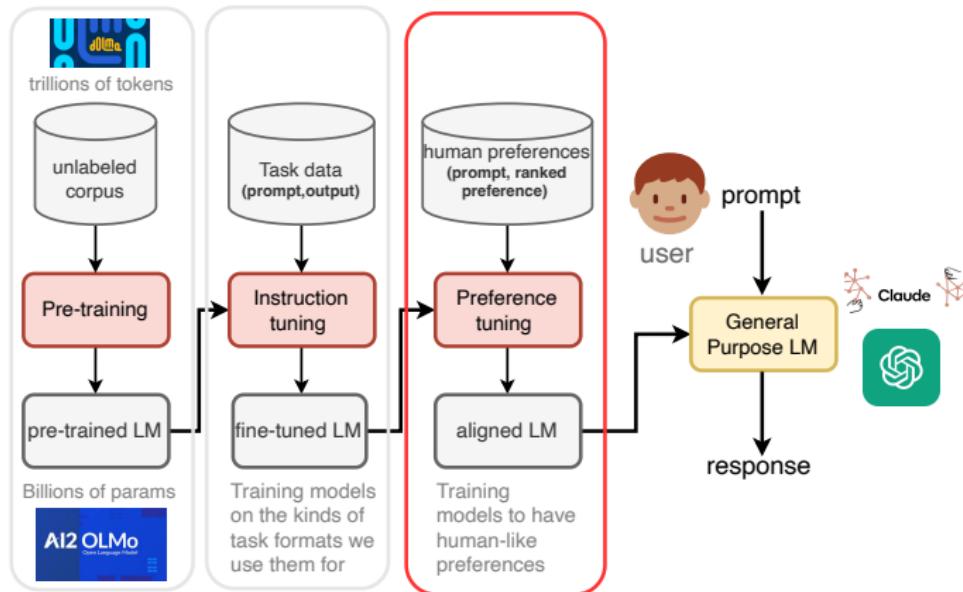
- ▶ Often trained via three stages: **pre-training**,

General purpose (large) language models: how?



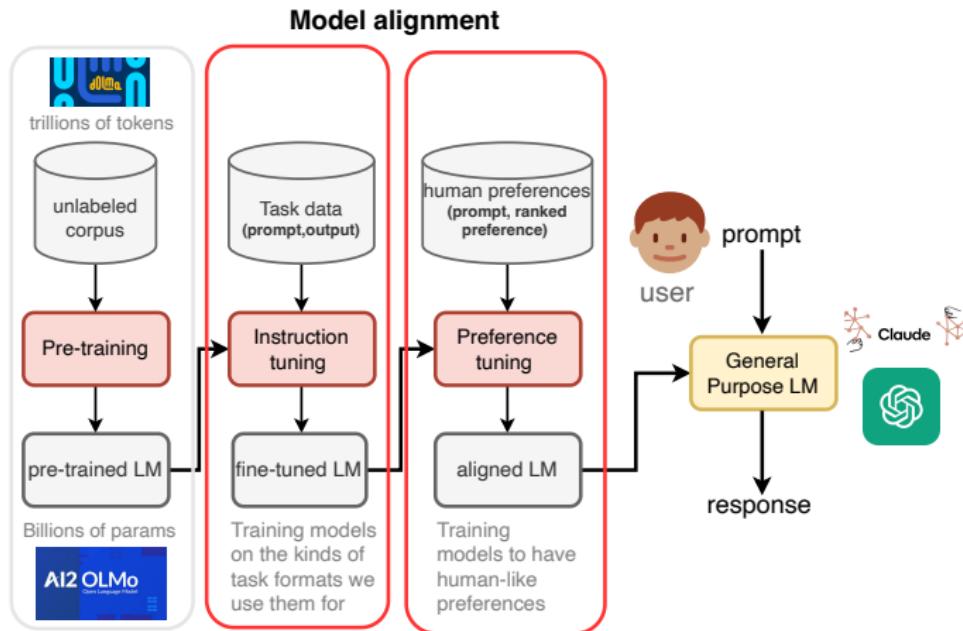
- ▶ Often trained via three stages: **pre-training**, **instruction tuning**

General purpose (large) language models: how?



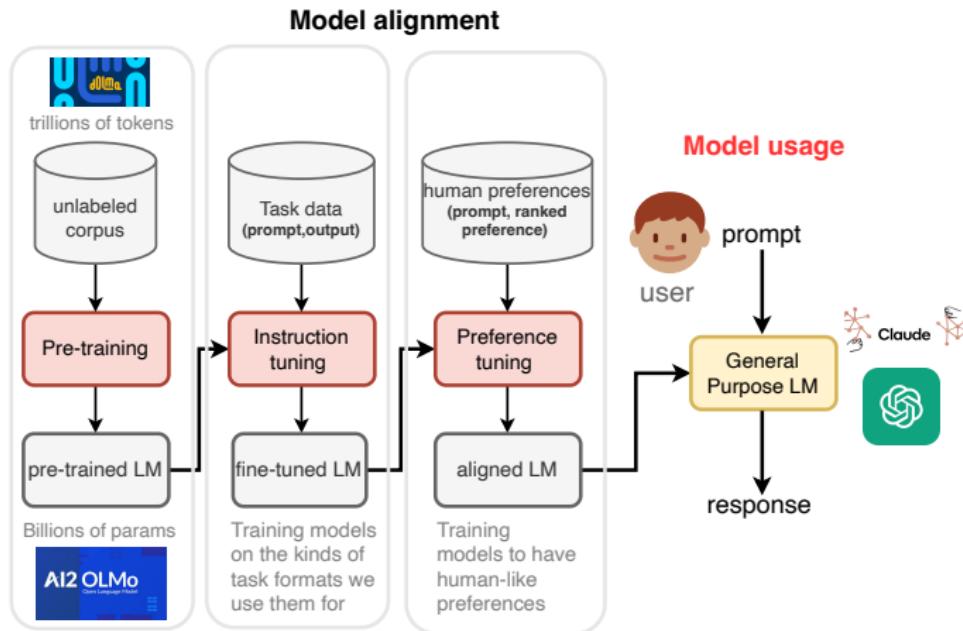
- ▶ Often trained via three stages: **pre-training**, **instruction tuning** and **preference alignment**.

General purpose (large) language models: how?



- ▶ Often trained via three stages: **pre-training**, **instruction tuning** and **preference alignment**.

General purpose (large) language models: **how?**



- ▶ Often trained via three stages: **pre-training**, **instruction tuning** and **preference alignment**.

OLMo on Huggingface hub (<https://huggingface.co/>)

The screenshot shows the Huggingface hub page for the `allenai/OLMo-2-1124-13B-Instruct` model. The top navigation bar includes links for Text Generation, Transformers, Safetensors, English, olimo2, conversational, arxiv2501.00656, arxiv2411.15124, and License: apache-2.0. Below the navigation are buttons for Model card, Files and versions, Community, and Settings.

The main content area features the `OLMo` logo, a large heading for `OLMo-2-1124-13B-Instruct`, and a note about the initial release. A "NOTE: 1/3/2025 UPDATE:" section discusses the realization that post-trained models did not share the pre-tokenization logic of the base models and details the training of new post-trained models. It also mentions the availability of old models with a "-preview" suffix.

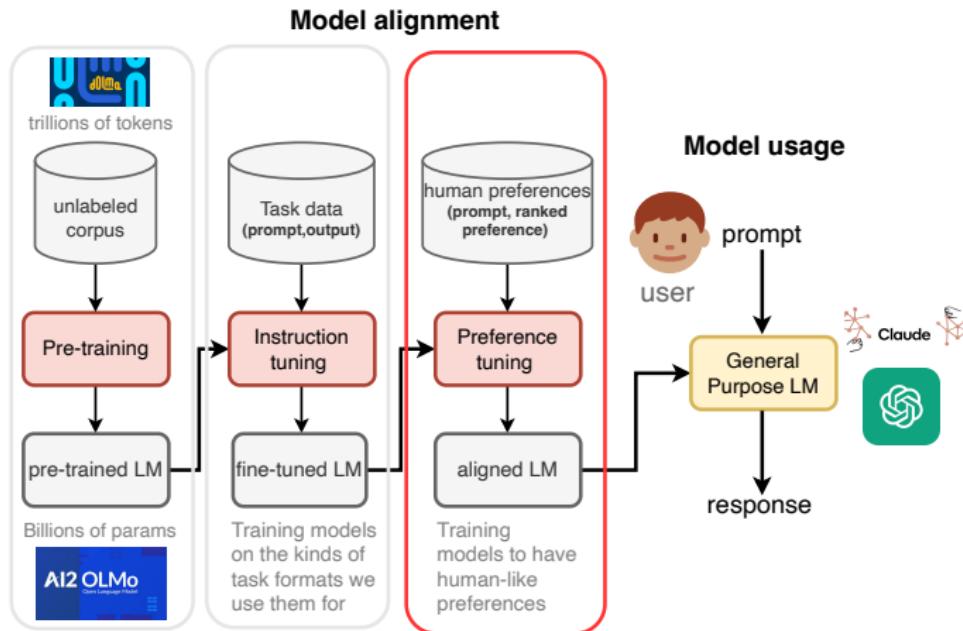
On the right side, there's an "Edit model card" button, a chart showing "Downloads last month" at 10,023, and a "View full history" link. Below this is a "Safetensors" section with model size (13.78 MB), tensor type (BF16), and a "Chat template" link. Further down are sections for "Inference Providers" (Text Generation) and "Model tree for allenai/OLMo-2-1124-13B-Instruct".

The "Model tree" section is highlighted with a red box and shows the following structure:

- Base model
 - Finetuned
 - Finetuned
 - Finetuned
 - Finetuned (1)
 - Adapters
 - Finetunes
 - Quantizations

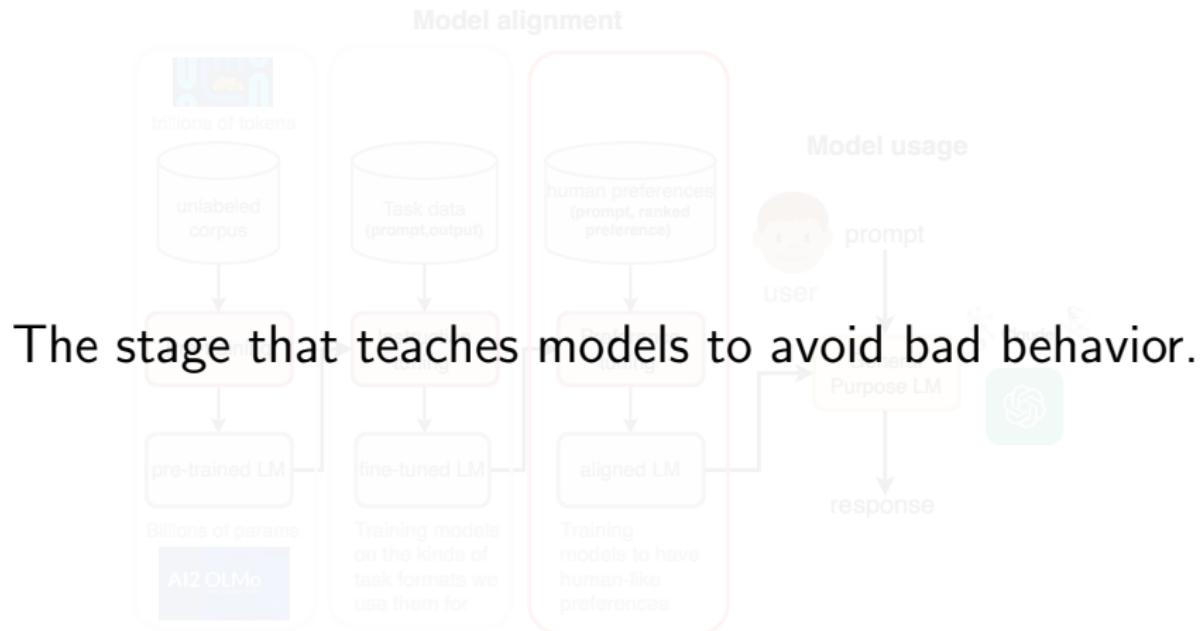
Path	Count
<code>allenai/OLMo-2-1124-7B</code>	1 model
<code>allenai/OLMo-2-1124-7B-SFT</code>	1 model
<code>allenai/OLMo-2-1124-7B-DPO</code>	1 model
<code>allenai/OLMo-2-1124-13B-Instruct-RLVR1</code>	1 model
<code>allenai/OLMo-2-1124-13B-Instruct-RLVR2</code>	1 model
<code>this model</code>	4 models
<code>allenai/OLMo-2-1124-13B-Instruct</code>	2 models
<code>allenai/OLMo-2-1124-13B-Instruct-RLVR1</code>	29 models

Preference alignment



- ▶ Often trained via three stages: **pre-training**, **instruction tuning** and **preference alignment**.

Preference alignment



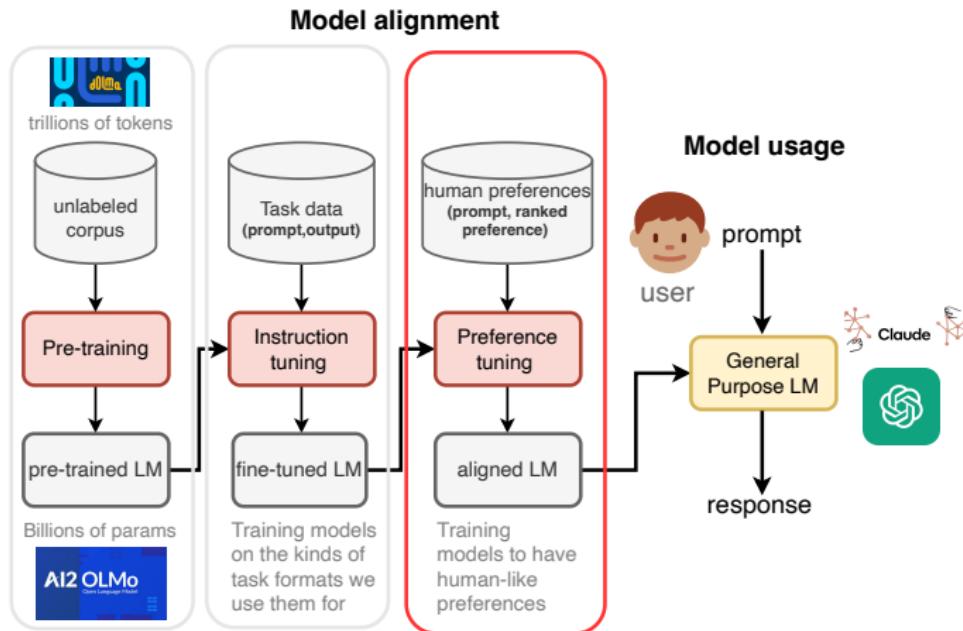
- ▶ Often trained via three stages: **pre-training**, **instruction tuning** and **preference alignment**.

Direct preference optimization (DPO)

The screenshot shows the Hugging Face Model Card for the **allenai/OLMo-2-1124-13B-Instruct** model. The card includes the following sections:

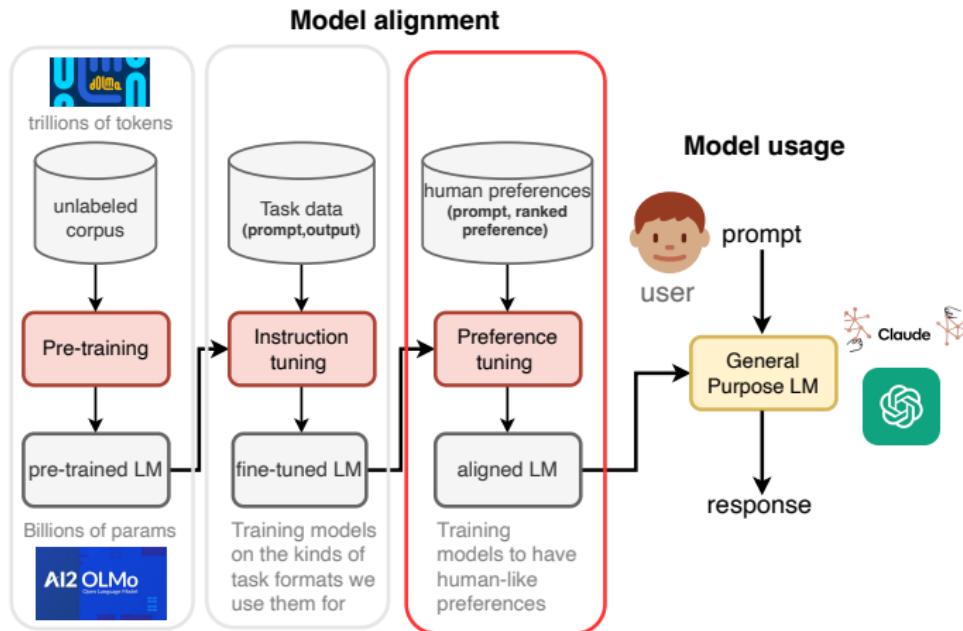
- Model card**: Includes the model logo, name, and a note about the initial release.
- Downloads last month**: Shows 10,023 downloads.
- Safetensors**: Provides details like model size (13.78 params), tensor type (BF16), and chat template.
- Inference Providers**: Shows that the model isn't deployed by any provider.
- Model tree for allenai/OLMo-2-1124-13B-Instruct**: A hierarchical tree showing the base model and its finetuned variants:
 - Base model**: allenai/OLMo-2-1124-7B
 - Finetuned**:
 - Finetuned**: allenai/OLMo-2-1124-7B-SFT
 - Finetuned**: allenai/OLMo-2-1124-7B-DPO (highlighted with a red box)
 - Finetuned**: allenai/OLMo-2-1124-13B-Instruct-RLVR1
 - Finetuned**: allenai/OLMo-2-1124-13B-Instruct-RLVR2
 - Adapters**: 4 models
 - Finetunes**: 2 models
 - Quantizations**: 29 models

Preference alignment



Today: Programming techniques for studying the semantics of direct preference alignment algorithms.

Preference alignment



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

warning: will get a bit technical...

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,

Preference alignment datasets

```
1  ### "pip install datasets"
2  from datasets import load_dataset
3
4  ## Ultrafeedback binary preferences
5  ultra = load_dataset("HuggingFaceH4/ultrafeedback_binarized")
6
7  ## random training instance
8  train_instance = ds['train_prefs'][14074]
9
10 ## input x
11 train_instance["prompt"]
12 ## => "write a paragraph with an angry tone"
13
14 ## preferred response, winner
15 train_instance["chosen"][-1]["content"]
16 #=> "I am absolutely fed up with this infuriating situation! ..."
17
18 ## dispreferred response, loser
19 train_instance["rejected"][-1]["content"]
20 #=> "Why do you always have to be so damn lazy? ..."
```

Direct preference alignment (DPA) algorithms

- ▶ Recent direct preference alignment (DPA) approaches assume a closed-form loss function, takes the form ([Tang et al., 2024](#)):

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

Direct preference alignment (DPA) algorithms

- Recent direct preference alignment (DPA) approaches assume a closed-form loss function, takes the form (Tang et al., 2024):

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


convex loss function

Direct preference alignment (DPA) algorithms

- Recent direct preference alignment (DPA) approaches assume a closed-form loss function, takes the form (Tang et al., 2024):

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

↓
model quantity

Direct preference alignment (DPA) algorithms

- Recent direct preference alignment (DPA) approaches assume a closed-form loss function, takes the form ([Tang et al., 2024](#)):

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

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, takes the form (Tang et al., 2024):

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

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, takes the form (Tang et al., 2024):

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

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

Intuitively: reasoning about relationships between our policy model π_θ and a reference model π_{ref} .

Example implementation of the DPO loss

```
1 import torch.nn.functional as F
2
3 def compute_loss(
4     chosen_logps: torch.FloatTensor, # winner
5     rejected_logps: torch.FloatTensor, # loser
6     ref_chosen_logps: torch.FloatTensor,
7     ref_rejected_logps: torch.FloatTensor,
8     beta
9 ):
10    ## model quantities
11    logratios = chosen_logps - rejected_logps
12    ref_logratios = ref_chosen_logps - ref_rejected_logps
13
14    diff = logratios - ref_logratios
15
16    ## sigmoid(x) = 1 / (1 + e^-x)
17    loss = -F.logsigmoid(beta*diff)
18
19    return loss
```

Adapted from TRL DPO implementation.

Direct preference alignment (DPA) algorithms

- Recent direct preference alignment (DPA) approaches assume a closed-form loss function, takes the form (Tang et al., 2024):

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

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 ρ_θ 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, takes the form ([Tang et al., 2024](#)):

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

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

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

Direct preference alignment (DPA) algorithms

- Recent direct preference alignment (DPA) approaches assume a closed-form loss function, takes the form ([Tang et al., 2024](#)):

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

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

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

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

Some variants of DPO

```
1 def compute_loss(
2     chosen_logps: torch.FloatTensor, # winner
3     rejected_logps: torch.FloatTensor, # loser
4     ref_chosen_logps: torch.FloatTensor,
5     ref_rejected_logps: torch.FloatTensor,
6     beta,
7     loss_type, ## function f
8 ):
9     logratios = chosen_logps - rejected_logps
10    ref_logratios = ref_chosen_logps - ref_rejected_logps
11
12    diff = logratios - ref_logratios
13    if loss_type == "dpo":
14        loss = -F.logsigmoid(beta*diff)
15    elif loss_type == "ipo":
16        loss = (diff - 1 / (2 * beta)) ** 2
17    elif loss_type == "slic":
18        loss = torch.relu(beta - logratios)
19
20    return loss
```

Some variants of DPO

```
1 def compute_loss(  
2     chosen_logps: torch.FloatTensor, # winner  
3     rejected_logps: torch.FloatTensor, # loser  
4     ref_chosen_logps: torch.FloatTensor,  
5     ref_rejected_logps: torch.FloatTensor,  
6     beta,  
7     loss_type, ## function f  
8 ):  
9     logratios = chosen_logps - rejected_logps  
10    diff = logratios - ref_logratios  
11    if loss_type == "dpo":  
12        loss = -F.logsigmoid(beta*diff)  
13    elif loss_type == "ipo":  
14        loss = (diff - 1 / (2 * beta)) ** 2  
15    elif loss_type == "slic":  
16        loss = torch.relu(beta - logratios)  
17    return loss
```

Exercise: implement function from scratch.

Computing output probabilities

```
1 from transformers import AutoTokenizer, AutoModelForCausallLM
2
3 model_name = "HuggingFaceTB/SmollM2-135M"
4 tokenizer = AutoTokenizer.from_pretrained(model_name)
5 model = AutoModelForCausallLM.from_pretrained(model_name)
6 ### example data
7 prompt = "Will drinking brake fluid kill you?"
8 loser = "No, it won't kill you"
9 winner = "No, but this could cause you serious harm"
10 loser_input = tokenizer.encode(
11     f"{prompt} {loser}", return_tensors="pt"
12 )
13 winner_input = tokenizer.encode(
14     f"{prompt} {winner}", return_tensors="pt"
15 )
16
17 ## how to pass through model, get (log) probabilities?
18 ## how is it related to TRL implementation?
```

Coming up with a new preference loss

- Recent direct preference alignment (DPA) approaches assume a closed-form loss function, takes the form (Tang et al., 2024):

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

The procedure: Select a convex loss function f , define some model quantity ρ_θ , experiment.

Coming up with a new preference loss

- ▶ Recent direct preference alignment (DPA) approaches assume a closed-form loss function, takes the form (Tang et al., 2024):

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

The procedure: Select a convex loss function f , define some model quantity ρ_θ , experiment.

1. (theory) *What is the right f to use?* Theoretical limitations and properties of Bradley-Terry model or other variants.

Coming up with a new preference loss

- ▶ Recent direct preference alignment (DPA) approaches assume a closed-form loss function, takes the form ([Tang et al., 2024](#)):

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

The procedure: Select a convex loss function f , define some model quantity ρ_θ , experiment.

1. (theory) *What is the right f to use?* Theoretical limitations and properties of Bradley-Terry model or other variants.
2. (empirical) *Can we devise novel algorithmic variants of DPO and ρ_θ ?* 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, takes the form (Tang et al., 2024):

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

The procedure: Select a convex loss function f , define some model quantity ρ_θ , experiment.

1. (theory) *What is the right f to use?* Theoretical limitations and properties of Bradley-Terry model or other variants.
2. (empirical) *Can we devise novel algorithmic variants of DPO and ρ_θ ?* Find the next best preference algorithm?

today: How do we define new ρ_θ s and what is the size and structure of this space?

Coming up with a new preference loss

- ▶ Recent direct preference alignment (DPA) approaches assume a closed-form loss function, takes the form (Tang et al., 2024):

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

The procedure: Select a convex loss function f , defining some model quantity ρ_θ , experiment.

1. (theory) *What is the right f to use?* Theoretical limitations and properties of Bradley-Terry model or other variants.
2. (empirical) *Can we devise novel algorithmic variants of DPO and ρ_θ ?* Find the next best preference algorithm?

today: How do we define new ρ_θ s and what is the size and structure of this space?

A look at the structure of DPO loss functions

- ▶ Looking at different preference approaches through time.

A look at the structure of DPO loss functions

- ▶ Looking at different preference approaches through time.

Approach	Loss equation $f(\rho_\theta, \beta)$
S1iC (Zhao et al., 2022)	$\max \left(0, \beta - \log \frac{\pi_\theta(y_w x)}{\pi_\theta(y_l x)} \right)$

A look at the structure of DPO loss functions

- ▶ Looking at different preference approaches through time.

Approach	Loss equation $f(\rho_\theta, \beta)$
S1iC (Zhao et al., 2022)	$\max \left(0, \beta - \log \frac{\pi_\theta(y_w x)}{\pi_\theta(y_l x)} \right)$
CPO (Xu et al., 2024)	$-\log \sigma \left(\beta \log \frac{\pi_\theta(y_w x)}{\pi_\theta(y_l x)} \right)$

A look at the structure of DPO loss functions

- ▶ Looking at different preference approaches through time.

Approach	Loss equation $f(\rho_\theta, \beta)$
S1iC (Zhao et al., 2022)	$\max\left(0, \beta - \log \frac{\pi_\theta(y_w x)}{\pi_\theta(y_l x)}\right)$
CPO (Xu et al., 2024)	$-\log \sigma\left(\beta \log \frac{\pi_\theta(y_w x)}{\pi_\theta(y_l x)}\right)$

A look at the structure of DPO loss functions

- ▶ Looking at different preference approaches through time.

Approach	Loss equation $f(\rho_\theta, \beta)$
S1iC (Zhao et al., 2022)	$\max\left(0, \beta - \log \frac{\pi_\theta(y_w x)}{\pi_\theta(y_l x)}\right)$
CPO (Xu et al., 2024)	$-\log \sigma\left(\beta \log \frac{\pi_\theta(y_w x)}{\pi_\theta(y_l x)}\right)$

Remove optimization details, log ratio $s_\theta(y_w, y_l) := \log \frac{\pi_\theta(y_w|x)}{\pi_\theta(y_l|x)}$,
regularizer terms, length normalization, **core loss equations** ρ_θ .

A look at the structure of DPO loss functions

- ▶ Looking at different preference approaches through time.

Approach	Loss equation $f(\rho_\theta, \beta)$
SliC (Zhao et al., 2022)	$\max\left(0, \beta - \log \frac{\pi_\theta(y_w x)}{\pi_\theta(y_l x)}\right)$
CPO (Xu et al., 2024)	$-\log \sigma\left(\beta \log \frac{\pi_\theta(y_w x)}{\pi_\theta(y_l x)}\right)$
RRHF (Yuan et al., 2024)	$\max\left(0, -\log \frac{\pi_\theta(y_w x)}{\pi_\theta(y_l x)} \frac{ y_w }{ y_l }\right)$

note: This is a messy area, idiosyncratic.

Remove optimization details, log ratio $s_\theta(y_w, y_l) := \log \frac{\pi_\theta(y_w|x)}{\pi_\theta(y_l|x)}$, regularizer terms, length normalization, core loss equations ρ_θ .

A look at the structure of DPO loss functions

- ▶ Looking at different preference approaches through time.

Approach	Loss equation $f(\rho_\theta, \beta)$
S1iC (Zhao et al., 2022)	$\max\left(0, \beta - \log \frac{\pi_\theta(y_w x)}{\pi_\theta(y_l x)}\right)$
CPO (Xu et al., 2024)	$-\log \sigma\left(\beta \log \frac{\pi_\theta(y_w x)}{\pi_\theta(y_l x)}\right)$
RRHF (Yuan et al., 2024)	$\max\left(0, -\log \frac{\pi_\theta(y_w x)}{\pi_\theta(y_l x)} \frac{ y_w }{ y_l }\right)$

Remove optimization details, log ratio $s_\theta(y_w, y_l) := \log \frac{\pi_\theta(y_w|x)}{\pi_\theta(y_l|x)}$, regularizer terms, length normalization, **core loss equations** ρ_θ .

A look at the structure of DPO loss functions

- ▶ Looking at different preference approaches through time.

Approach	Loss equation $f(\rho_\theta, \beta)$
SliC (Zhao et al., 2022)	$\max\left(0, \beta - \log \frac{\pi_\theta(y_w x)}{\pi_\theta(y_l x)}\right)$
CPO (Xu et al., 2024)	$-\log \sigma\left(\beta \log \frac{\pi_\theta(y_w x)}{\pi_\theta(y_l x)}\right)$
RRHF (Yuan et al., 2024)	$\max\left(0, \beta - \log \frac{\pi_\theta(y_w x)}{\pi_\theta(y_l x)} \cdot \frac{ y_w }{ y_l }\right)$

observation: same log ratios keep coming up

Remove optimization details, log ratio $s_\theta(y_w, y_l) := \log \frac{\pi_\theta(y_w|x)}{\pi_\theta(y_l|x)}$, regularizer terms, length normalization, core loss equations ρ_θ .

A look at the structure of DPO loss functions

- ▶ Looking at different preference approaches through time.

Approach	Loss equation $f(\rho_\theta, \beta)$
SliC (Zhao et al., 2022)	$\max\left(0, \beta - \log \frac{\pi_\theta(y_w x)}{\pi_\theta(y_l x)}\right)$
CPO (Xu et al., 2024)	$-\log \sigma\left(\beta \log \frac{\pi_\theta(y_w x)}{\pi_\theta(y_l x)}\right)$
RRHF (Yuan et al., 2024)	$\max\left(0, -\log \frac{\pi_\theta(y_w x)}{\pi_\theta(y_l x)}\right)$

question: What do these log ratios mean semantically?

Remove optimization details, log ratio $s_\theta(y_w, y_l) := \log \frac{\pi_\theta(y_w|x)}{\pi_\theta(y_l|x)}$, regularizer terms, length normalization, core loss equations ρ_θ .

A look at the structure of DPO loss functions

- ▶ Looking at different preference approaches through time.

Approach	Loss equation $f(\rho_\theta, \beta)$
S1iC (Zhao et al., 2022)	$\max\left(0, \beta - \log \frac{\pi_\theta(y_w x)}{\pi_\theta(y_l x)}\right)$
CPO (Xu et al., 2024)	$-\log \sigma\left(\beta \log \frac{\pi_\theta(y_w x)}{\pi_\theta(y_l x)}\right)$
RRHF (Yuan et al., 2024)	$\max\left(0, -\log \frac{\pi_\theta(y_w x)^{ y_w }}{\pi_\theta(y_l x)^{ y_l }}\right)$

Adding a reference model, involves adding an additional term:

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

The recipe for creating new variants of DPO

DPO

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

The recipe for creating new variants of DPO

DPO

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

- ▶ Type 1: Add additional terms to loss.

Variant	Core Loss equation ρ_θ
R-DPO (Park et al., 2024)	$s_\theta(y_w, y_l) - s_{\text{ref}}(y_w, y_l) + \gamma_{\text{length}}$
ODPO (Amini et al., 2024)	$s_\theta(y_w, y_l) - s_{\text{ref}}(y_w, y_l) - \gamma_{\text{offset}}$
DPOP (Pal et al., 2024)	$s_\theta(y_w, y_l) - s_{\text{ref}}(y_w, y_l) - \log \frac{\pi_{\text{ref}}(x, y_w)}{\pi_\theta(x, y_w)}$

The recipe for creating new variants of DPO

DPO

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

- ▶ Type 1: Add additional terms to loss.

Variant	Core Loss equation ρ_θ
R-DPO (Park et al., 2024)	$s_\theta(y_w, y_l) - s_{\text{ref}}(y_w, y_l) + \gamma_{\text{length}}$
ODPO (Amini et al., 2024)	$s_\theta(y_w, y_l) - s_{\text{ref}}(y_w, y_l) - \gamma_{\text{offset}}$
DPOP (Pal et al., 2024)	$s_\theta(y_w, y_l) - s_{\text{ref}}(y_w, y_l) - \log \frac{\pi_{\text{ref}}(x, y_w)}{\pi_\theta(x, y_w)}$

- ▶ Type 2: Change or re-parameterize reference ratio.

Variant	Core Loss equation ρ_θ
ORPO (Hong et al., 2024)	$s_\theta(y_w, y_l) - \log \frac{1 - \pi_\theta(y_w x)}{1 - \pi_\theta(y_l x)}$
SimPO (Meng et al., 2024)	$s_\theta(y_w, y_l) - \gamma_{\text{penalty}}$

The recipe for creating new variants of DPO

DPO

$$-\log \alpha \left(\beta \cdot \left[\underbrace{\log \frac{\pi_\theta(y_w | x)}{\pi_\theta(y_l | x)}}_{s_\theta(y_w, y_l)} - \underbrace{\log \frac{\pi_{\text{ref}}(y_w | x)}{\pi_{\text{ref}}(y_l | x)}}_{s_{\text{ref}}(y_w, y_l)} \right] \right)$$

- ▶ Type 1: Add additional terms to loss.

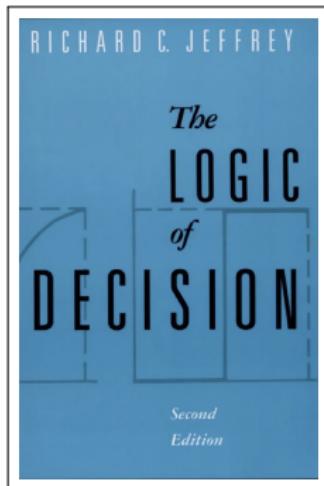
Variant	Core Loss equation ρ_θ
question: What happens when we add new terms?	
ODPO (Amini et al., 2024)	$s_\theta(y_w, y_l) - s_{\text{ref}}(y_w, y_l) - \gamma_{\text{offset}}$
DPOP (Pal et al., 2024)	$s_\theta(y_w, y_l) - s_{\text{ref}}(y_w, y_l) - \log \frac{\pi_{\text{ref}}(x, y_w)}{\pi_\theta(x, y_w)}$

- ▶ Type 2: Change or re-parameterize reference ratio.

Variant	Core Loss equation ρ_θ
ORPO (Hong et al., 2024)	$s_\theta(y_w, y_l) - \log \frac{1 - \pi_\theta(y_w x)}{1 - \pi_\theta(y_l x)}$
SimPO (Meng et al., 2024)	$s_\theta(y_w, y_l) - \gamma_{\text{penalty}}$

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)

Preference Principle	THE STATUS OF VARIOUS PREFERENCE PRINCIPLES					
	Von Wright	Chisholm Sosa	Martin	$P^\#$	P^*	P^w
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. $pPq \rightarrow (p \& \sim q) P(\sim p \& q)$	✓	✓		+	+	+
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. $pPq \rightarrow [(p \& r) P(q \& r) \& (p \& \sim r)]$	✓			-	-	+
11. $[(p \& r) P(q \& r) \& (p \& \sim r) P(q \& \sim r)] \rightarrow pPq$	✓			(+) ²	(+) ³	+
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. $pP(g \vee r) \rightarrow (pPp \& pPr)$	✓			-	-	-
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.

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

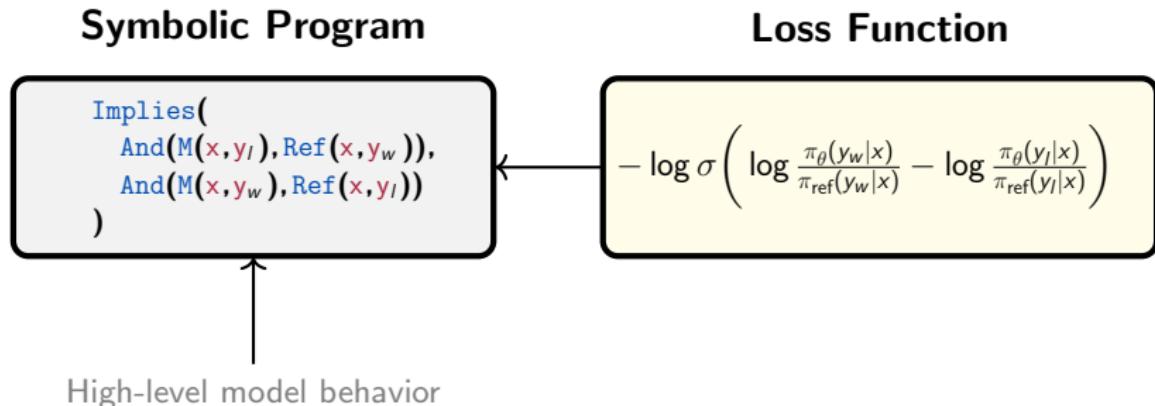
Loss Function

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

Two models, four predictions

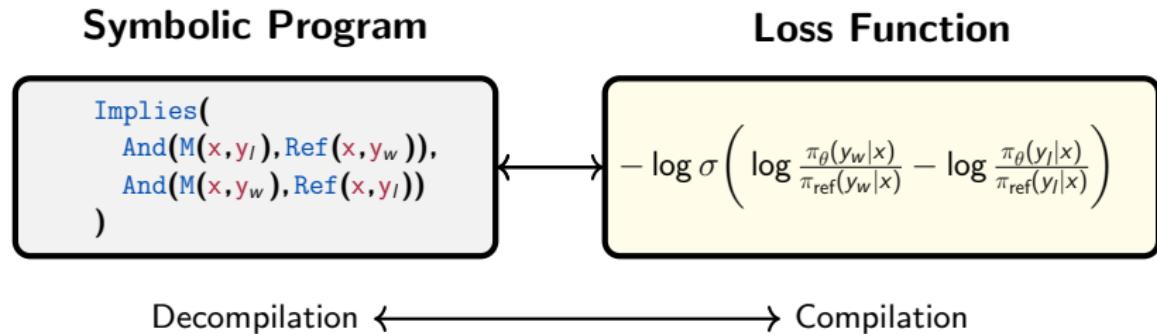
- ▶ **Problem:** Given some loss function, can we derive a symbolic program or expression that characterizes the semantics of that loss?

Preference learning as a discrete reasoning problem



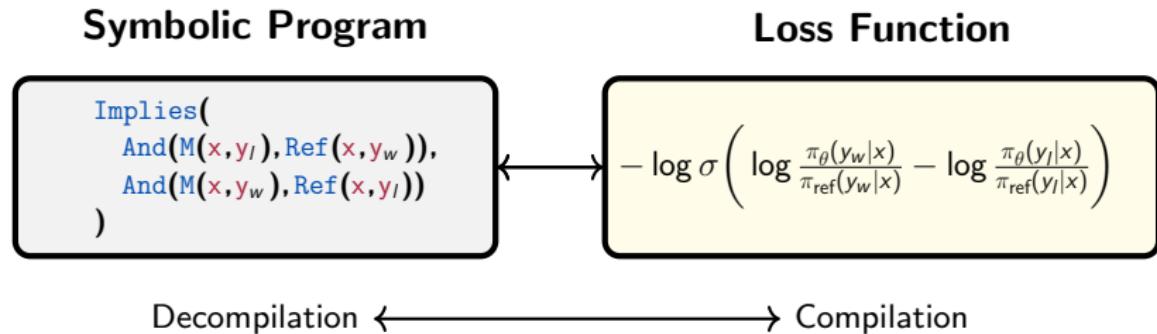
- ▶ **Problem:** Given some loss function, can we derive a symbolic program or expression that characterizes the semantics of that loss?

Preference learning as a discrete reasoning problem



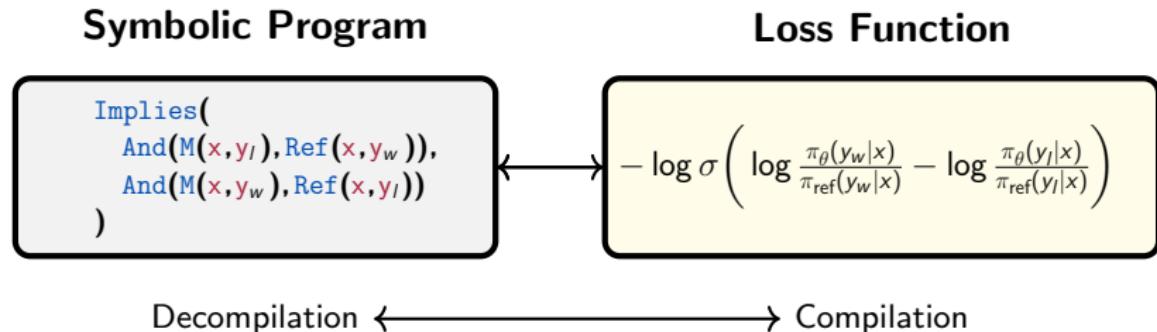
- ▶ **Problem:** Given some loss function, can we derive a symbolic program or expression that characterizes the semantics of that loss?

Preference learning as a discrete reasoning problem



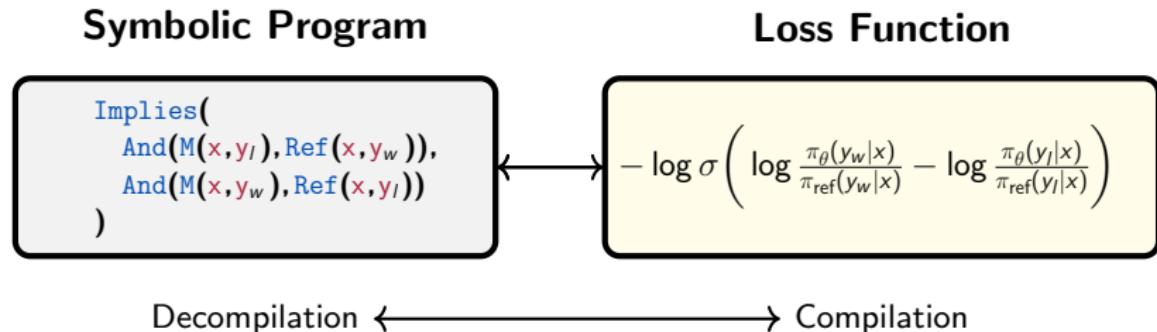
- ▶ **Problem:** Given some loss function, can we derive a symbolic program or expression that characterizes the semantics of that loss?

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. **Compilation:** Translating specifications into loss, [last lecture](#).

Preference learning as a discrete reasoning problem



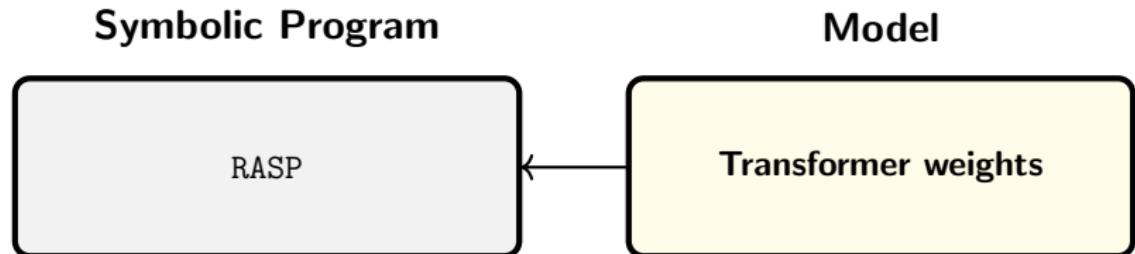
- ▶ **Problem:** Given some loss function, can we derive a symbolic program or expression that characterizes the semantics of that loss?
 1. **Compilation:** Translating specifications into loss, [last lecture](#).
 2. **Decompilation:** Losses to specifications (inverse), less explored.

Distilling LLMs to symbolic programs in general

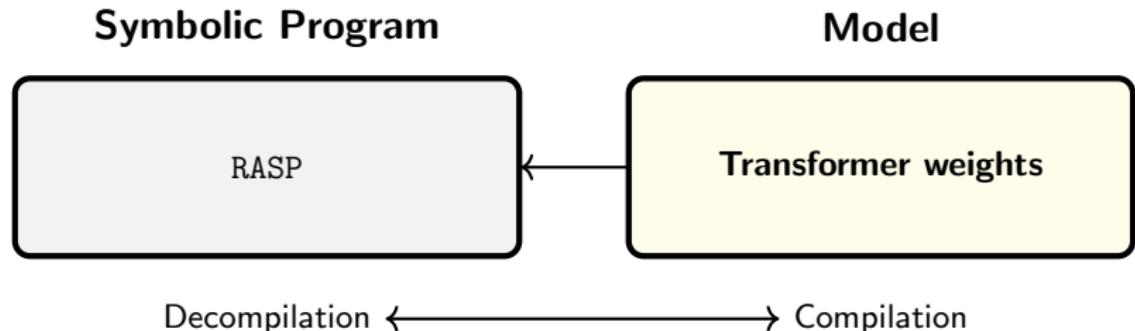
Model

Transformer weights

Distilling LLMs to symbolic programs in general



Distilling LLMs to symbolic programs 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](#)).

Distilling LLMs to symbolic programs in general

Learning Transformer Programs

Dan Friedman Alexander Wettig Danqi Chen

Department of Computer Science & Princeton Language and Intelligence

Princeton University

{dfriedman,awettig,danqic}@cs.princeton.edu

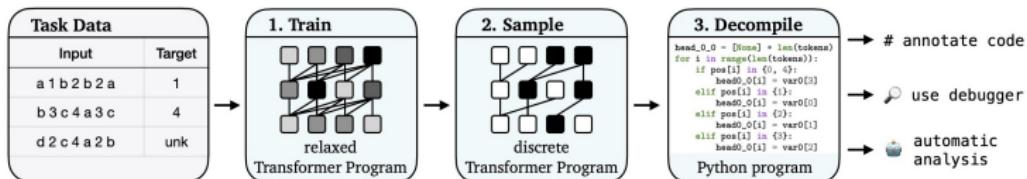


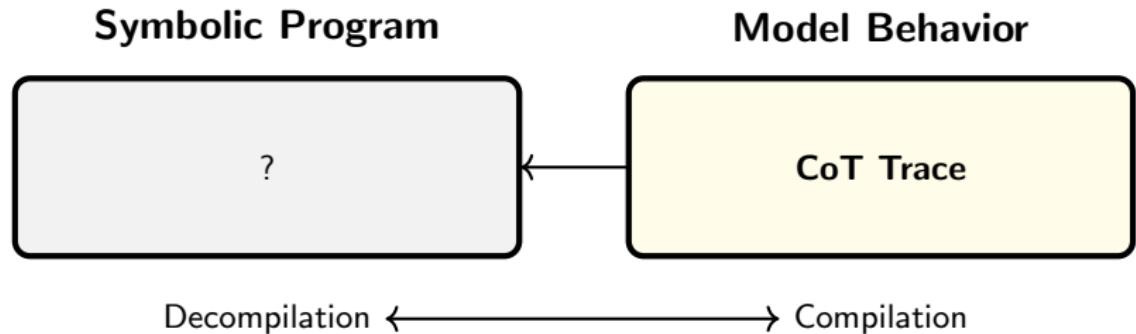
Figure 1: We design a modified Transformer that can be trained on data and then automatically discretized and converted into a human-readable program. The program is functionally identical to the Transformer, but easier to understand—for example, using an off-the-shelf Python debugger.

Distilling LLMs to symbolic programs in general

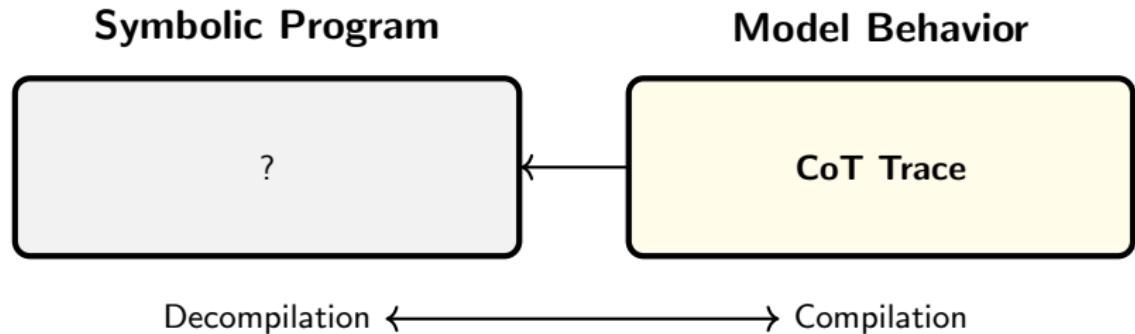
Model Behavior

CoT Trace

Distilling LLMs to symbolic programs in general



Distilling LLMs to symbolic programs in general



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

Distilling LLMs to symbolic programs in general



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

Distilling LLMs to symbolic programs in general

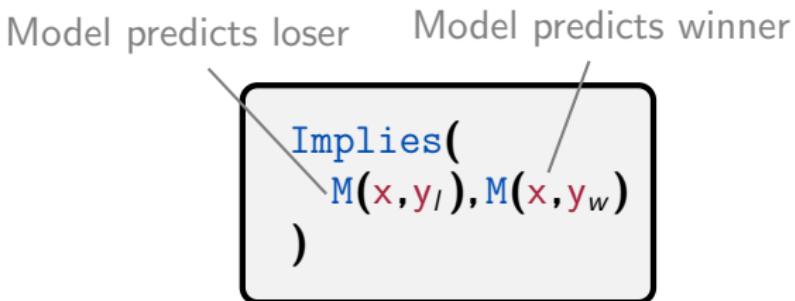


- ▶ Not always clear what the target language is or should be.

What do these programs tell us?

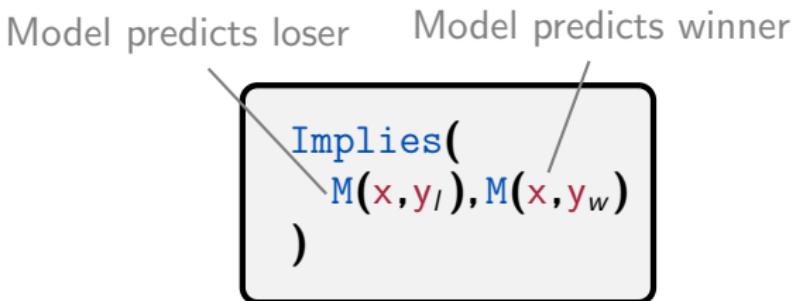
```
Implies(  
    M(x,yl), M(x,yw)  
)
```

What do these programs tell us?



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

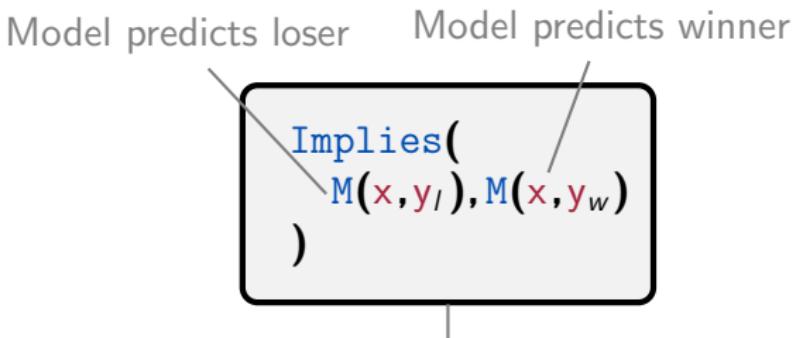
What do these programs tell us?



$$w(M(x, y)) = \pi_M(y | x)$$

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

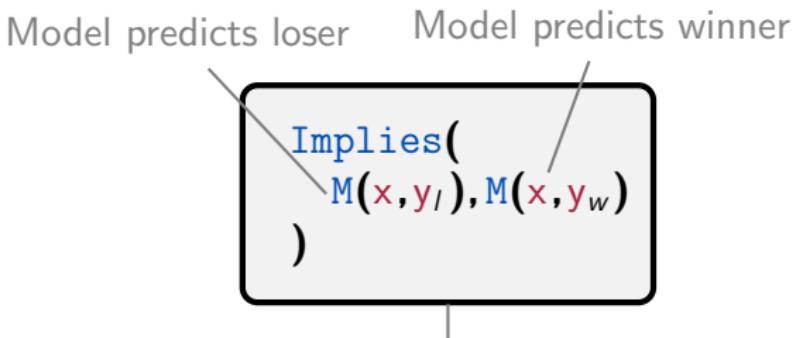
What do these programs tell us?



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; ρ_θ as formulas.

What do these programs tell us?



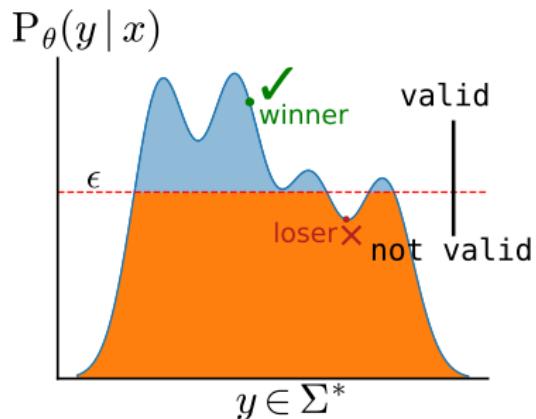
Whenever the model deems the loser to be a valid generation, it should deem the winner to be valid too.

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

What do these programs tell us?

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.

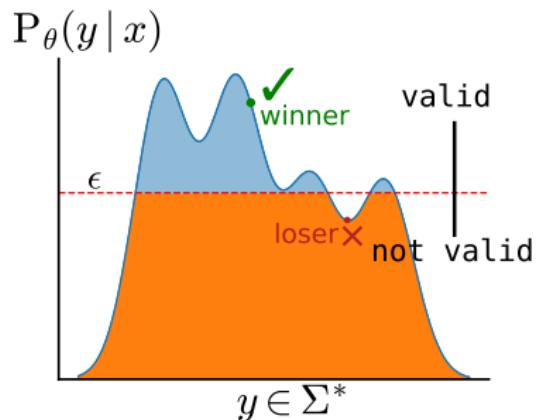


Assumption: Every loss function has an internal logic that can be expressed in this way, we want to uncover that logic.

What do these programs tell us?

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

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

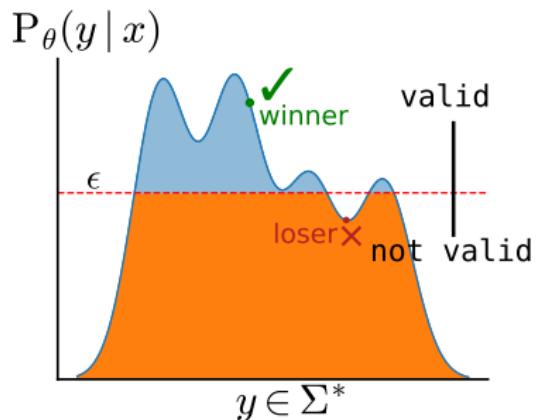


Assumption: Every loss function has an internal logic that can be expressed in this way, we want to uncover that logic.

What do these programs tell us?

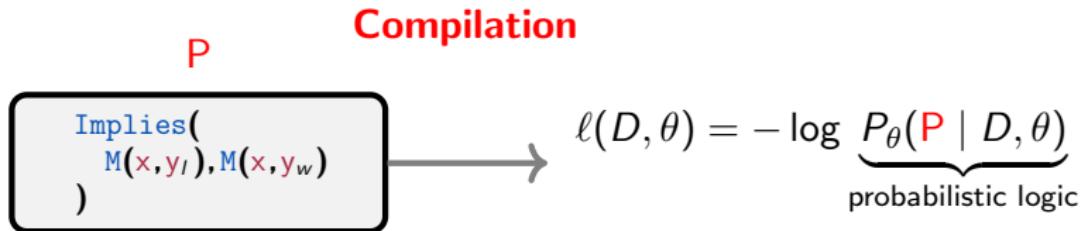
Implies(
M(x, y_l), M(x, y_w))

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



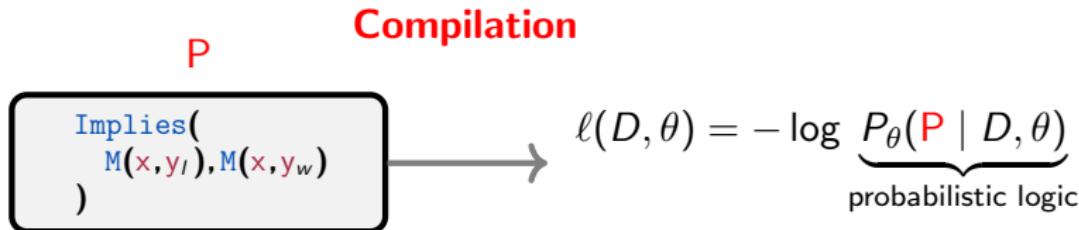
Observation: The second program is more strict than the first, involves semantic entailment.

What do these programs tell us?



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

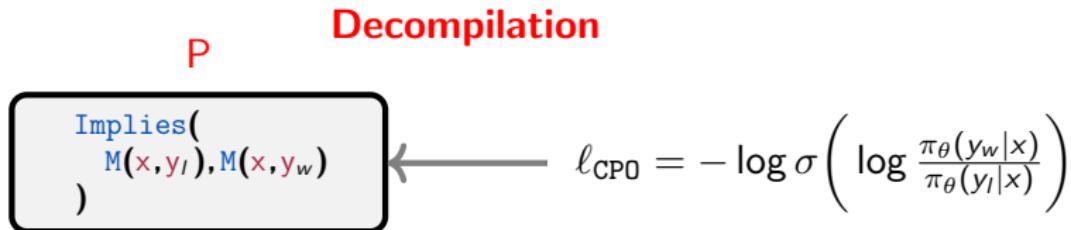
What do these programs tell us?



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

We we need: a probabilistic logic for preference modeling, **note:** logic useful not only for learning and loss.

What do these programs tell us?



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

What do these programs tell us?

P

Decompilation

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



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

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

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

What do these programs tell us?

Decompilation

P

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

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

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

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

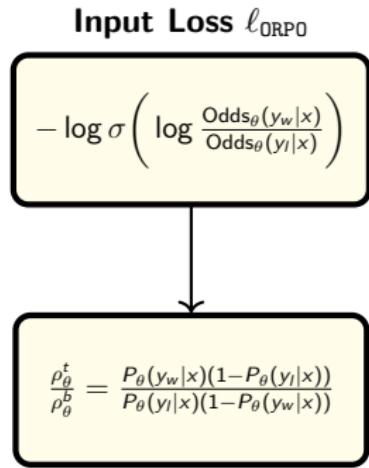
Second things we need: Defined a mechanical procedure for decompilation, proved its correctness, invariance to choice of f .

Bigger picture

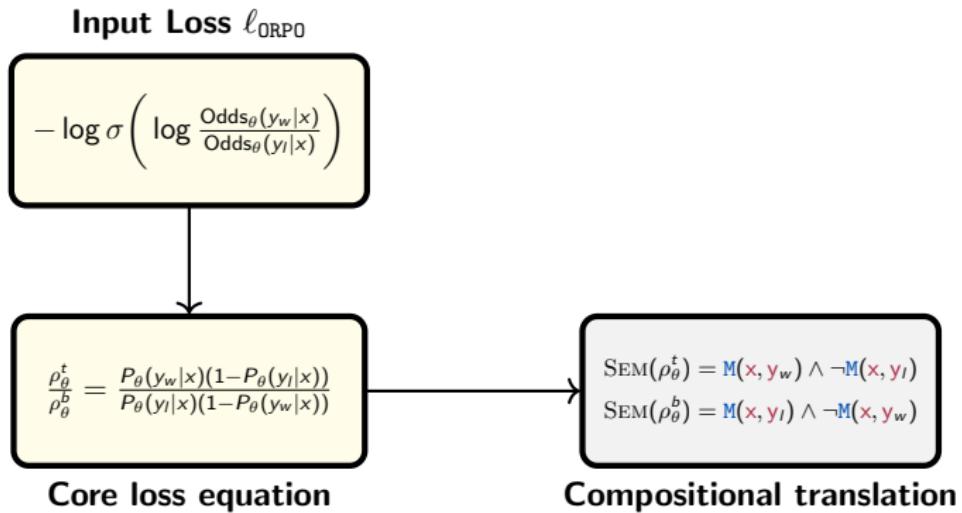
Input Loss ℓ_{ORPO}

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

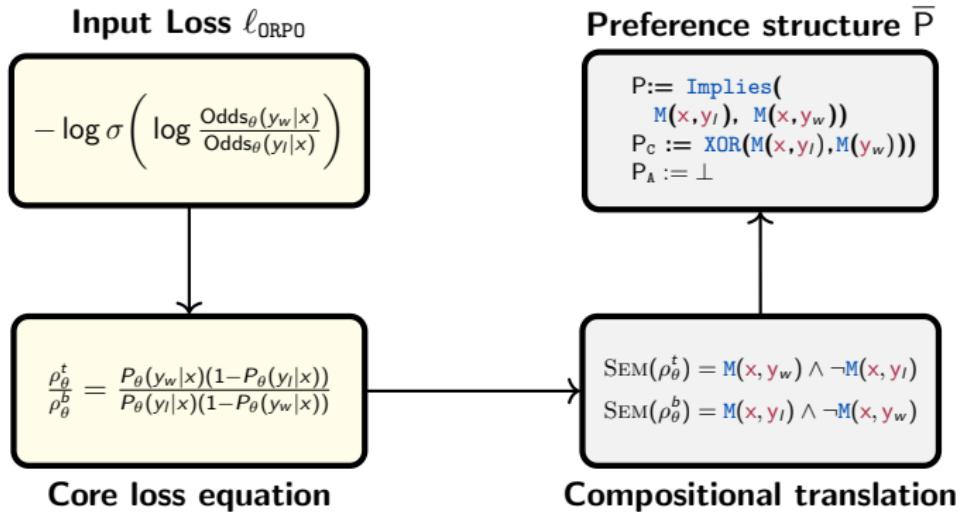
Bigger picture



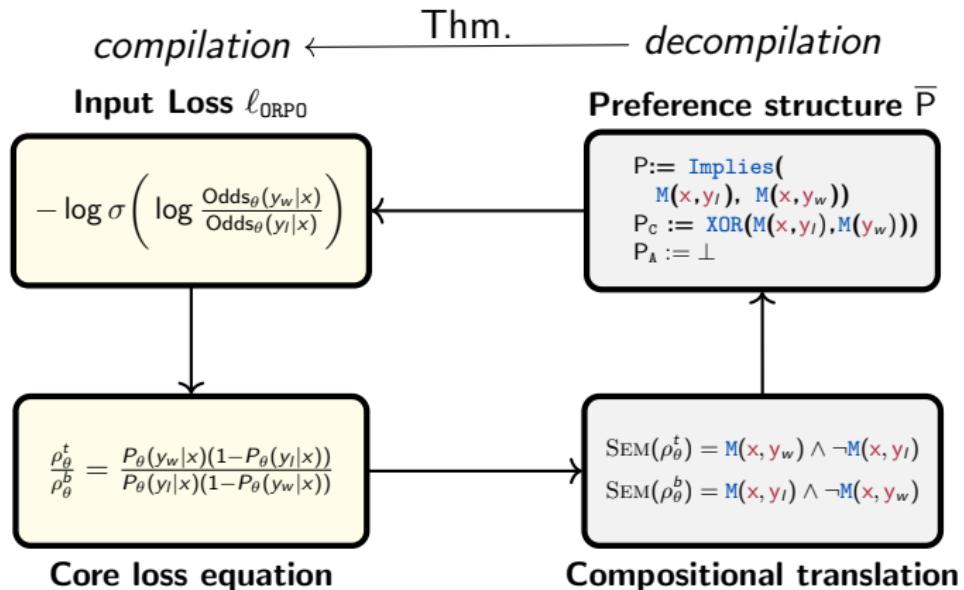
Bigger picture



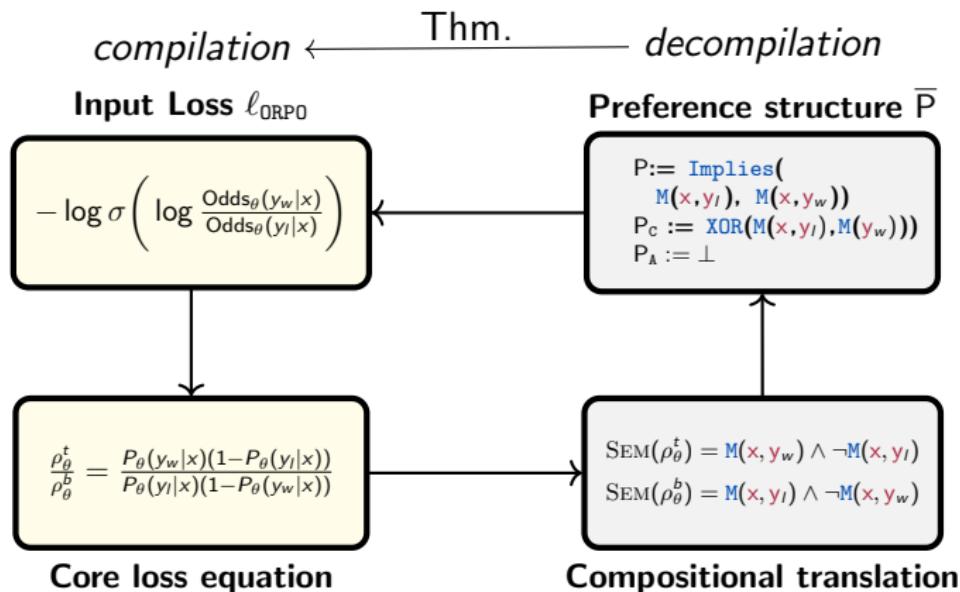
Bigger picture



Bigger picture

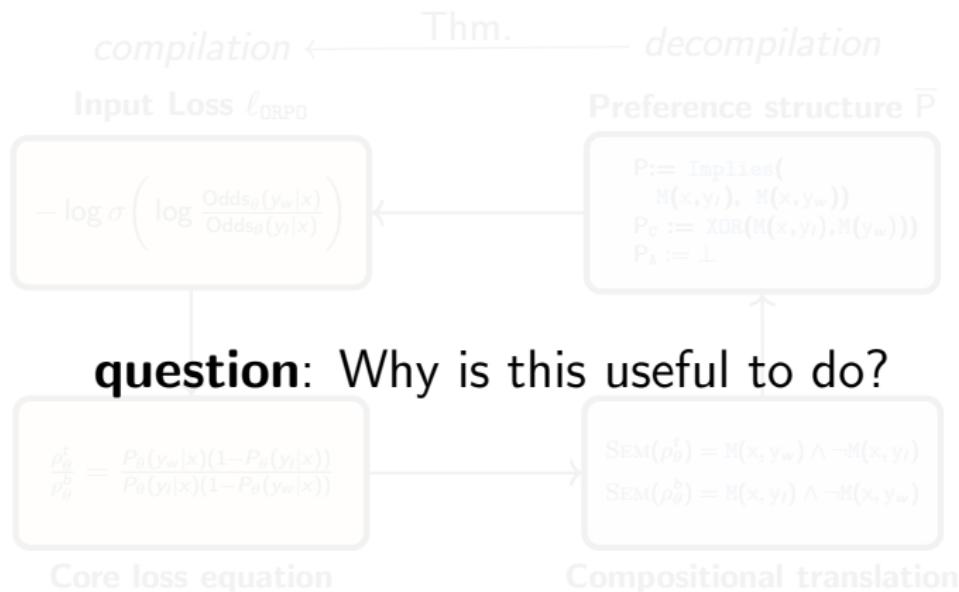


Bigger picture



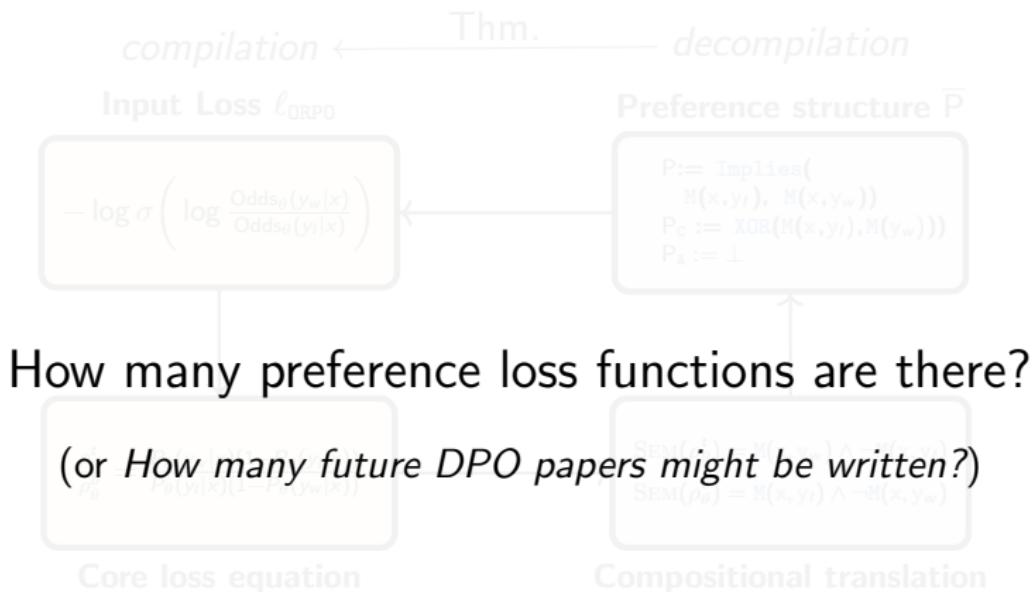
- ▶ Approach of Richardson et al. (2025), **preference structure**, special encoding, has a natural Boolean interpretation.

Bigger picture



- Approach of Richardson et al. (2025), preference structure, special encoding, has a natural Boolean interpretation.

Bigger picture



- Approach of Richardson et al. (2025), preference structure, special encoding, has a natural Boolean interpretation.

Why is this useful? understanding the space

$P^{(1)}$

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

$P^{(2)}$

And(
 $M(x, y_w),$
 Not($M(x, y_I)$)))

Boolean functions, 2 variables

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

Why is this useful? understanding the space

Boolean functions, 2 variables

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

$P^{(1)}$

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

$P^{(2)}$

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

- 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

Boolean functions, 2 variables

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

$P^{(2)}$

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

no reference: 256 losses

- ▶ 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

Boolean functions, 2 variables

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

$P^{(2)}$

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

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

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($M(x, y_I)$, $Ref(x, y_w)$),
And($M(x, y_w)$, $Ref(x, y_I)$))
)

4 variables

$Ref(x, y_w)$	$M(x, y_I)$	$Ref(x, y_I)$	$M(x, y_w)$
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	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($M(x, y_I)$, Ref(x, y_w)),
And($M(x, y_w)$, Ref(x, y_I)))

)

4 variables

answer: loads.

Ref(x, y_w)	$M(x, y_I)$	Ref(x, y_I)	$M(x, y_w)$
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
F	T	T	T
T	F	F	F
T	F	T	F
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($M(x, y_I)$, Ref(x, y_w)),
And($M(x, y_w)$, Ref(x, y_I)))

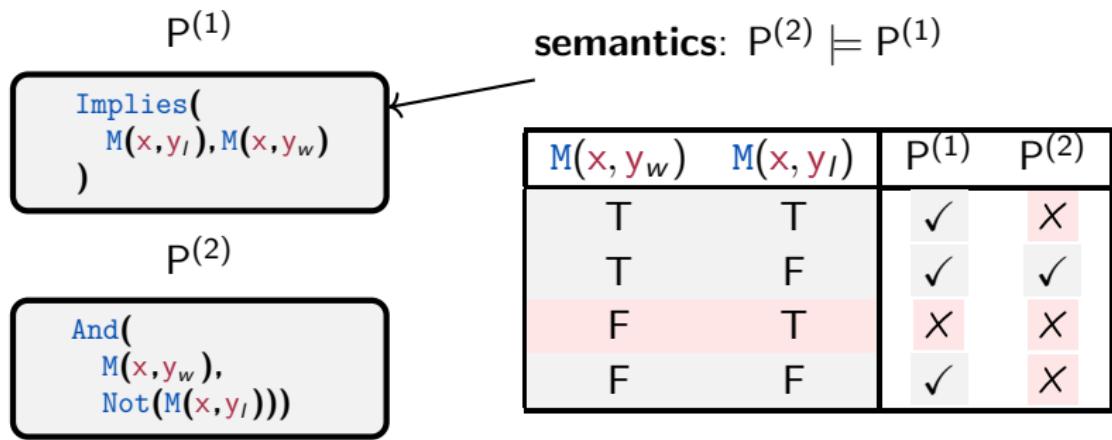
question: How are losses related to one another?

4 variables

Ref(x, y_w)	$M(x, y_I)$	Ref(x, y_I)	$M(x, y_w)$
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
F	T	T	T
T	F	F	F
T	F	T	T
T	T	F	F
T	T	F	T
T	T	T	F
T	T	T	T

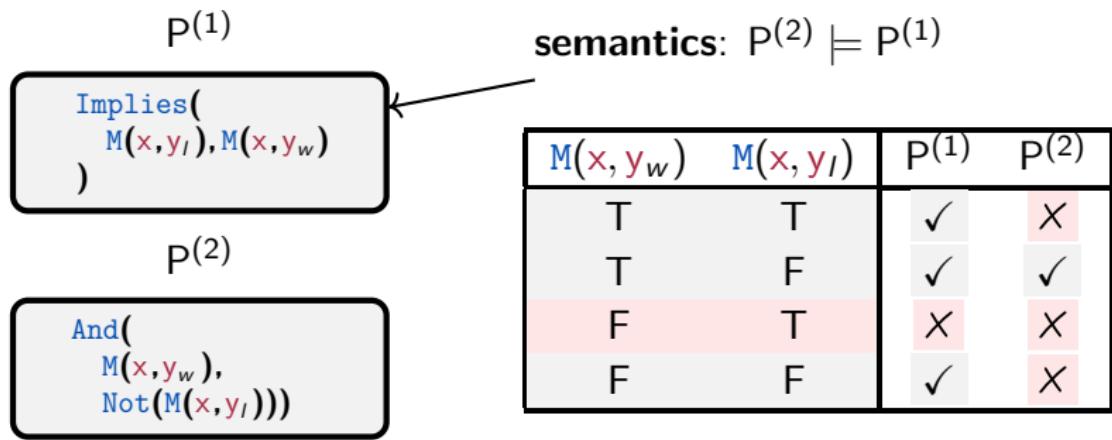
w/ reference: 4,294,967,296 losses

Why is this useful? understanding the structure



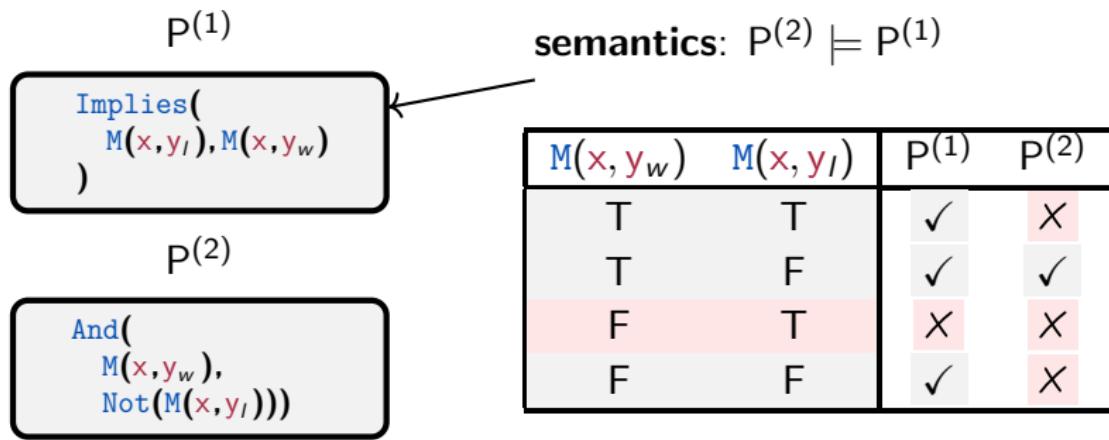
Proposition (Xu et al., 2018): Loss behavior is monotonic w.r.t semantic entailment: if $P^{(2)} \models P^{(1)}$ then $\ell(D, \theta, P^{(2)}) \geq \ell(D, \theta, P^{(1)})$.

Why is this useful? understanding the structure



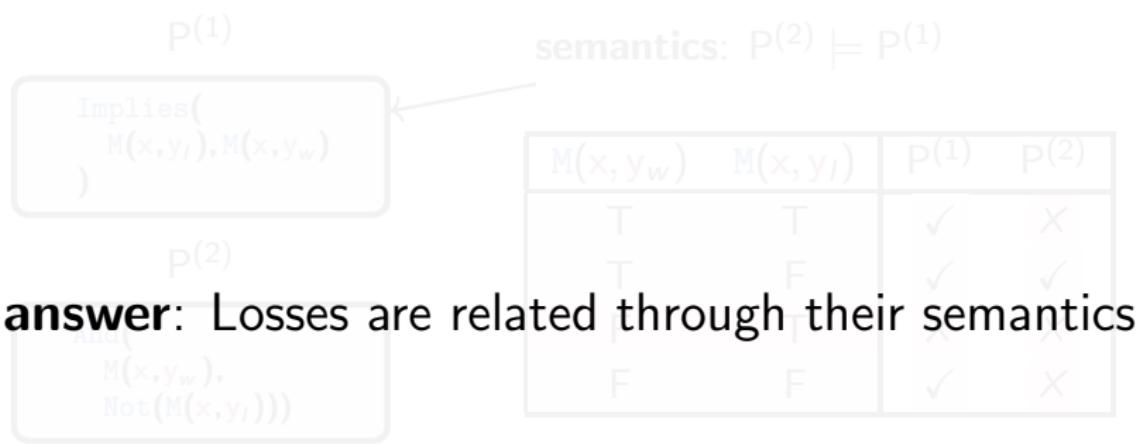
Proposition (Xu et al., 2018): Loss is equivalent under semantic equivalence: If $P^{(2)} \equiv P^{(1)}$ then $\ell(D, \theta, P^{(2)}) = \ell(D, \theta, P^{(1)})$.

Why is this useful? understanding the structure



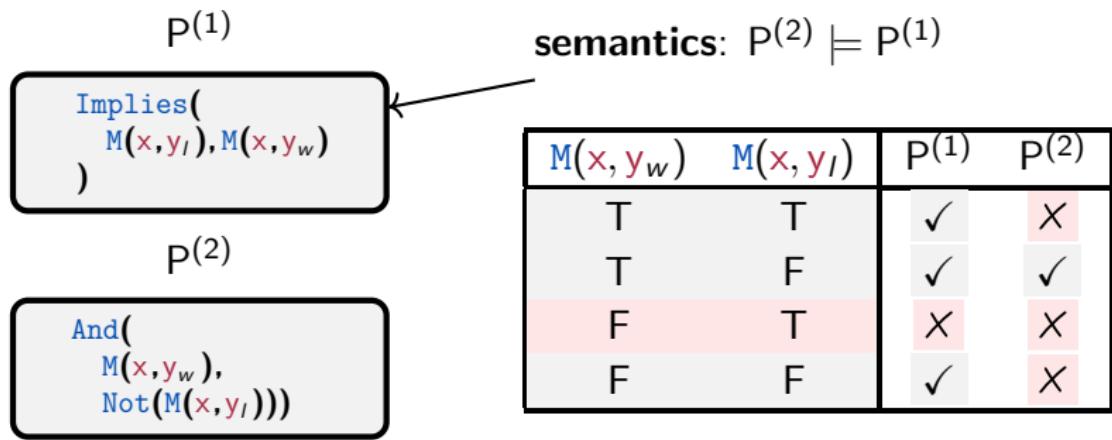
Theorem: $\ell(D, \theta, P^{(2)}) > \ell(D, \theta, P^{(1)})$ (the loss of $P^{(1)}$ is contained in the loss of $P^{(2)}$).

Why is this useful? understanding the structure



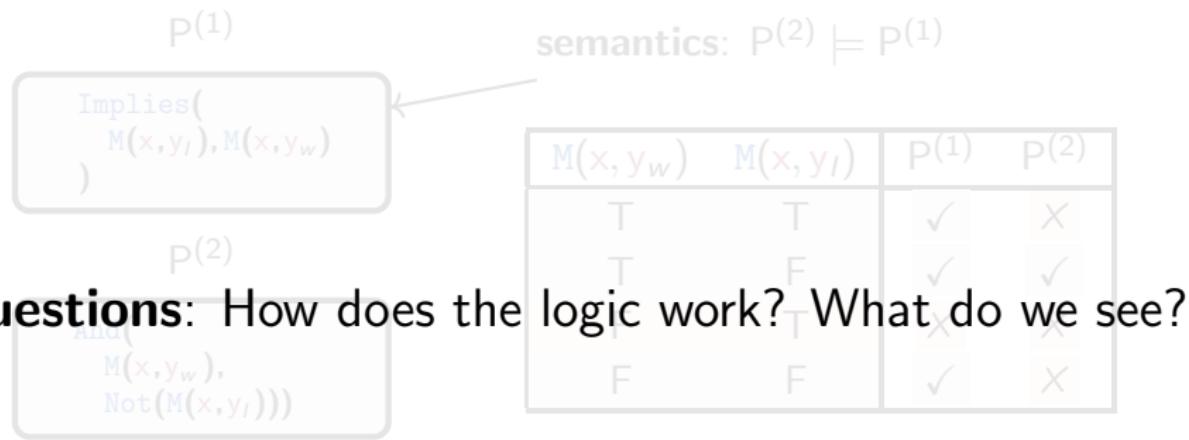
Theorem: $\ell(D, \theta, P^{(2)}) > \ell(D, \theta, P^{(1)})$ (the loss of $P^{(1)}$ is contained in the loss of $P^{(2)}$).

Why is this useful? understanding the structure



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

Why is this useful? understanding the structure



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

Revisiting the semantic loss

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

Revisiting the semantic loss

$M(x, y_w)$	$M(x, y_l)$	CPO	ORPO	unCPO	P
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.

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

Revisiting the semantic loss

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

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

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

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

$$\underbrace{\ell(P, \theta)}_{\text{semantic loss}} := -\log P_\theta(P) = -\log \frac{\text{WMC}(P, \theta)}{\text{WMC}(P, \theta) + \text{WMC}(\neg P, \theta)}$$

Revisiting the semantic loss

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

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

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

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

$$\underbrace{\ell(P, \theta)}_{\text{semantic loss}} := -\log P_\theta(P) = -\log \frac{\text{WMC}(P, \theta)}{\text{WMC}(P, \theta) + \text{WMC}(\neg P, \theta)}$$

$$= \underbrace{-\log \sigma \left(\log \frac{\text{WMC}(P, \theta)}{\text{WMC}(\neg P, \theta)} \right)}_{\text{DPO-style losses}} - \log \sigma(\rho_\theta)$$

Generalizing the semantic loss

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

P
Implies(
 $M(x, y_I), M(x, y_w)$)

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

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

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

Generalizing the semantic loss

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

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

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

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

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

Generalizing the semantic loss

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

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

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

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

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

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

Generalizing the semantic loss

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

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

$$w \models l(x, y)$$

$$w \not\models \neg l(x, y)$$

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

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

Representing losses as declarative programs

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

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

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

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 $P_C := \text{XOR}(M(x, y_l), M(x, y_w))$

Deriving new losses from first principles

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

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

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

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

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

$$\{\checkmark, X\}_w := \prod_{w \models M(x, y)} \pi_\theta(y | x) \cdot \prod_{w \not\models M(x, y)} 1 - \pi_\theta(y | x)$$

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

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

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

How does decompilation work? intuition

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

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

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

underlined
 ℓ_x
generalized

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

Generalizing even further

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

P

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

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

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

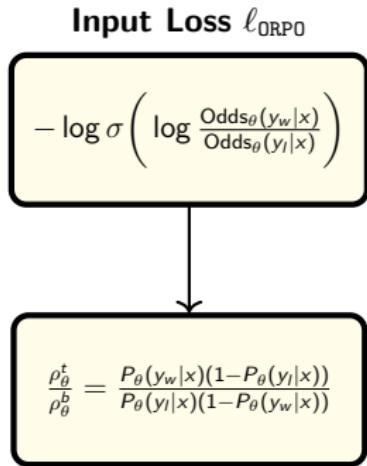
$$\underbrace{\ell_x}_{\text{generalized}} := \underbrace{f \left(\log \frac{\sum \text{✓}_w}{\sum \text{X}_w} \right)}_{\text{convex function } f}$$

Backtranslation: sketch

Input Loss ℓ_{ORPO}

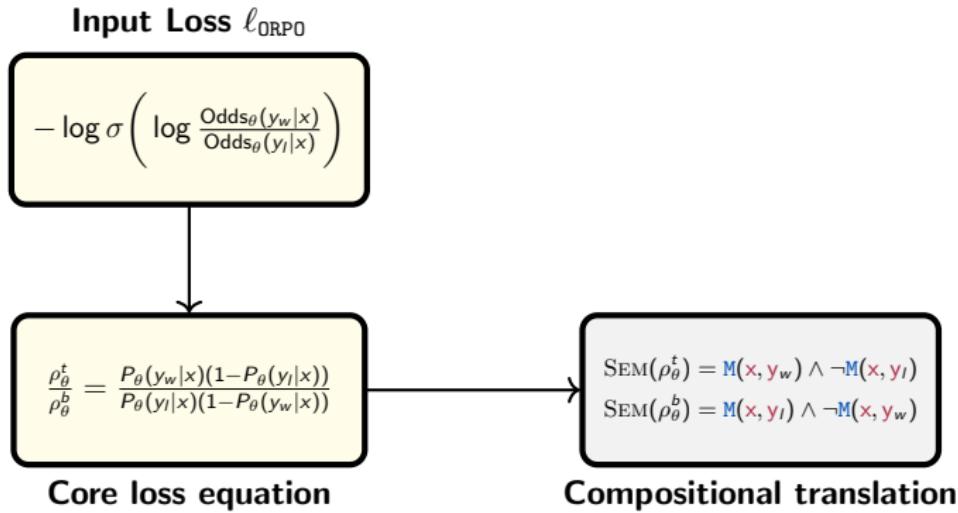
$$-\log \sigma \left(\log \frac{\text{Odds}_\theta(y_w|x)}{\text{Odds}_\theta(y_l|x)} \right)$$

Backtranslation: sketch



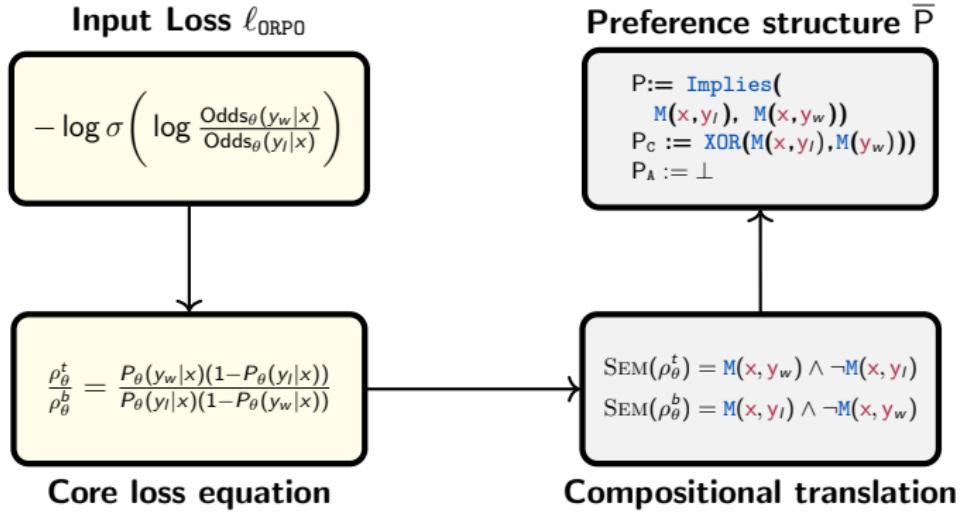
- ▶ Ratio of two (disjoint) multilinear polynomials.

Backtranslation: sketch



- ▶ Ratio of two (disjoint) multilinear polynomials; compositionally translate each polynomial into logic.

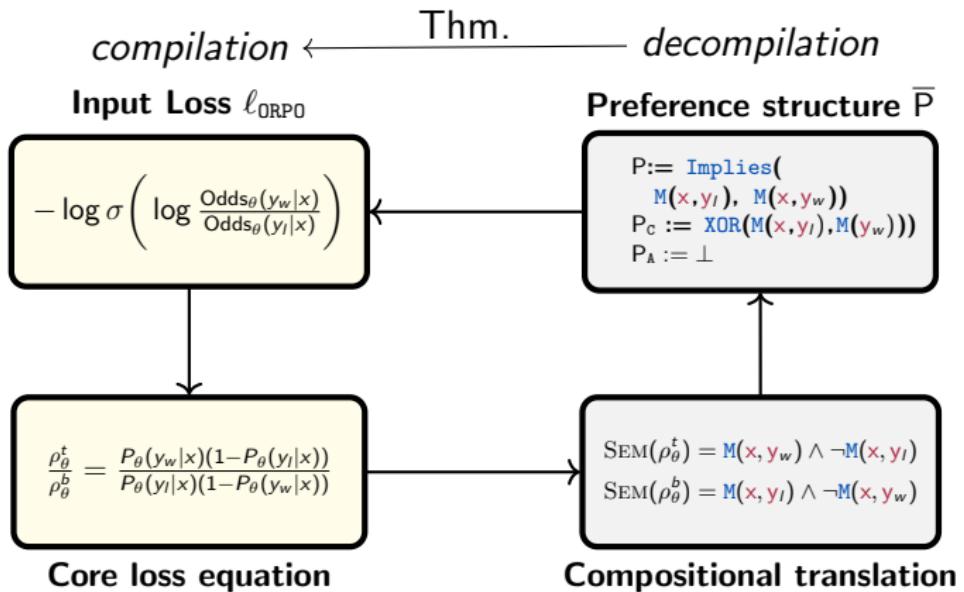
Backtranslation: sketch



- ▶ **Preference structure:** equivalent way of expressing truth table representations,

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

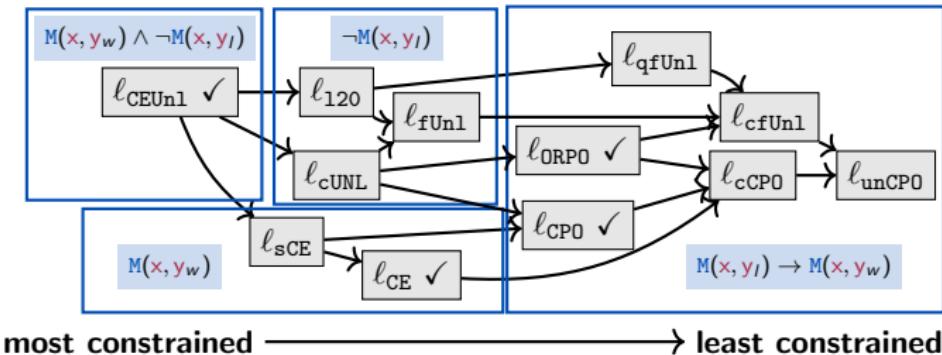
Backtranslation: sketch



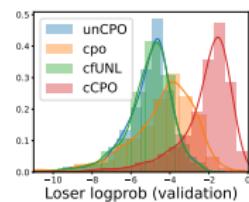
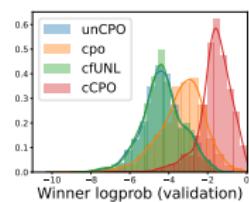
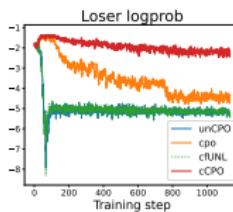
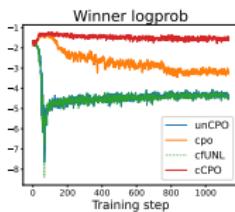
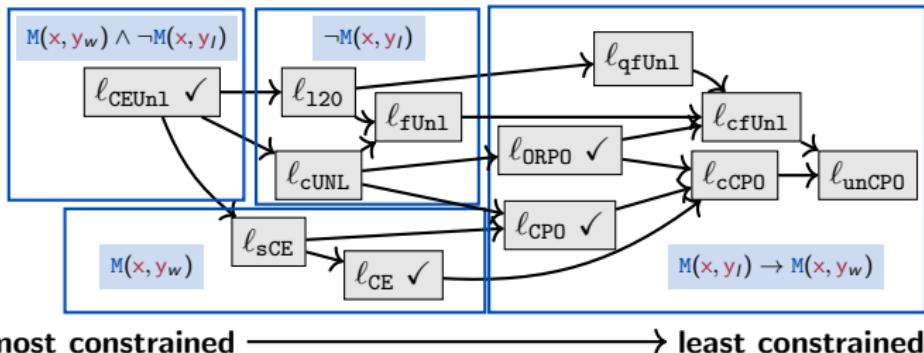
- ▶ **Preference structure:** equivalent way of expressing truth table representations,

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

The no reference loss landscape



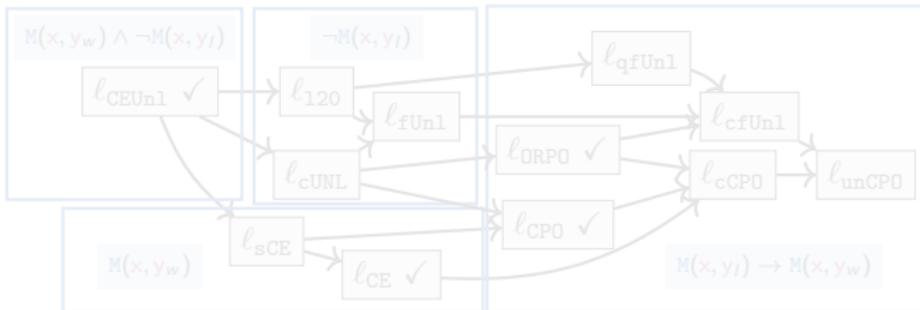
The no reference loss landscape



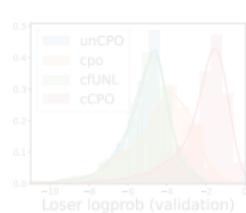
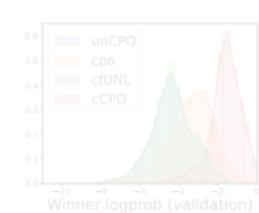
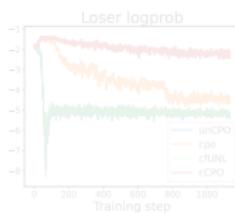
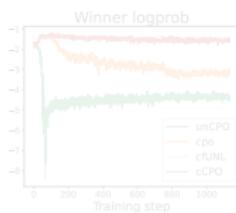
Training dynamics

Inference

The no reference loss landscape



question: Are any of these losses good?



Training dynamics

Inference

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

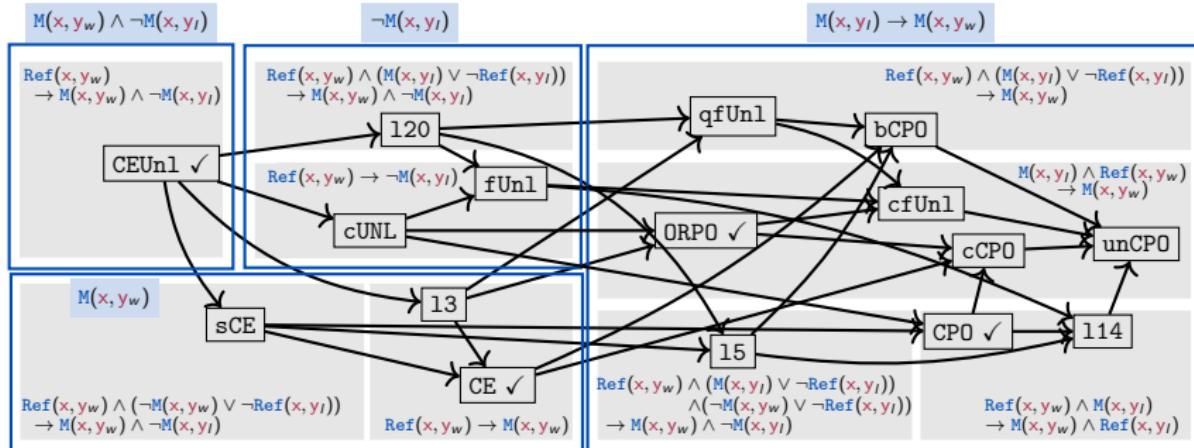
Whenever the model being tuned deems the loser to be a valid generation and the reference model deems the winner to be valid, the tuned model should deem the winner to be valid too, and **the reference should deem the loser to be valid.**

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

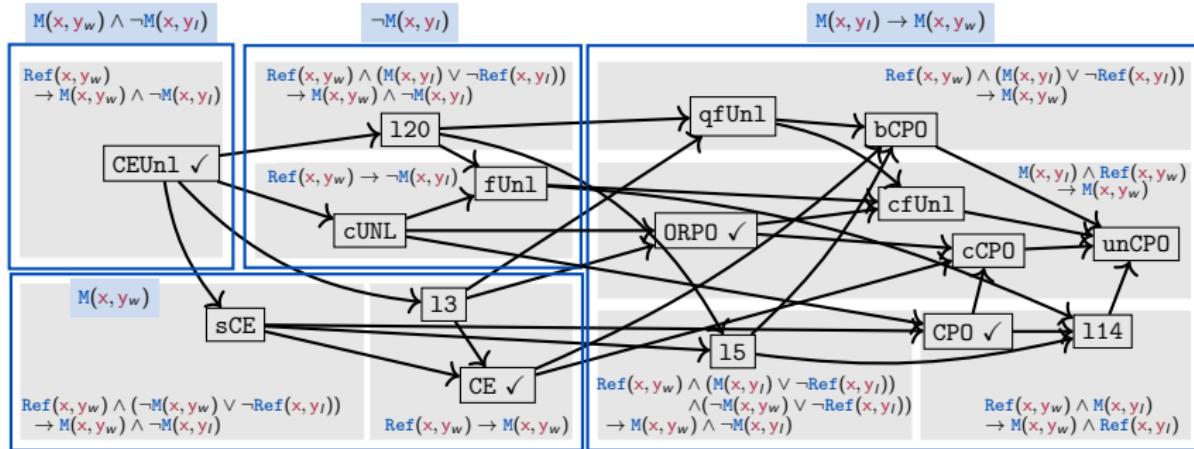
$$\sigma \left(\log \frac{\pi_{\theta}(y_w | x)}{\pi_{\theta}(y_I | x)} - \frac{\pi_{\text{ref}}(y_w | x)}{\pi_{\text{ref}}(y_I | x)} \right)$$

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

The full landscape, reference approaches



The full landscape, reference approaches



- ▶ Many new losses to explore and experiment with!

Conclusion

- ▶ Discussed the idea of decompilation, or working backwards from known losses to symbolic programs, *program analysis*.

Conclusion

- ▶ Discussed the idea of decompilation, or working backwards from known losses to symbolic programs, *program analysis*.
- ▶ **Preference learning:** tuning models to pairwise preference data, DPO.
 - Used to uncover formal semantics, *program new losses*.

Conclusion

- ▶ Discussed the idea of decompilation, or working backwards from known losses to symbolic programs, *program analysis*.
- ▶ **Preference learning:** tuning models to pairwise preference data, DPO.
 - Used to uncover formal semantics, *program* new losses.
- ▶ **Next time:** Probabilistic inference for test-time inference.

Thank you.

References I

- Amini, A., Vieira, T., and Cotterell, R. (2024). Direct preference optimization with an offset. *arXiv preprint arXiv:2402.10571*.
- 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*.
- Chavira, M. and Darwiche, A. (2008). On probabilistic inference by weighted model counting. *Artificial Intelligence*, 172(6-7):772–799.
- 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.
- 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.
- 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*.

References II

- Merrill, W. and Sabharwal, A. (2023). A logic for expressing log-precision transformers. *Advances in neural information processing systems*, 36:52453–52463.
- 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.
- Richardson, K., Srikumar, V., and Sabharwal, A. (2025). Understanding the logic of direct preference alignment through logic. *Proceedings of ICML*.
- 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*.
- Weiss, G., Goldberg, Y., and Yahav, E. (2021). Thinking like transformers. In *International Conference on Machine Learning*, pages 11080–11090. PMLR.

References III

- Xu, H., Sharaf, A., Chen, Y., Tan, W., Shen, L., Van Durme, B., Murray, K., and Kim, Y. J. (2024). Contrastive preference optimization: Pushing the boundaries of llm performance in machine translation. *arXiv preprint arXiv:2401.08417*.
- 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*.
- Yuan, H., Yuan, Z., Tan, C., Wang, W., Huang, S., and Huang, F. (2024). Rrhf: Rank responses to align language models with human feedback. *Advances in Neural Information Processing Systems*, 36.
- 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*.