Understanding the Logic of Generative AI through Logic and Programming

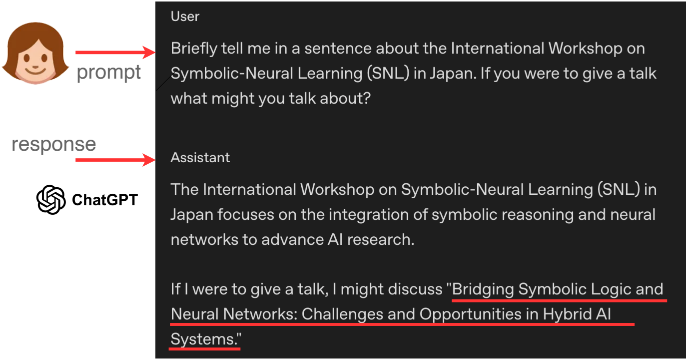
Kyle Richardson

Allen Institute for AI (AI2)

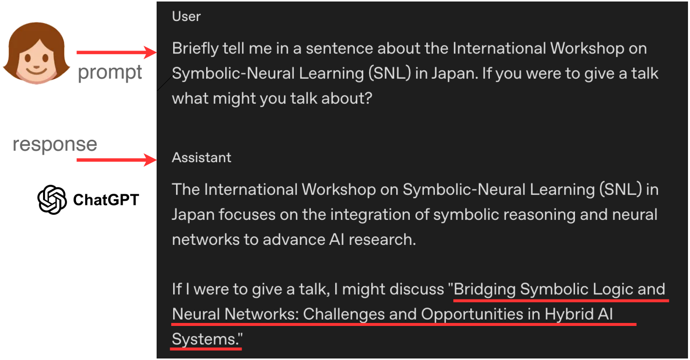
October 2025

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

General purpose large language models (LLMs)

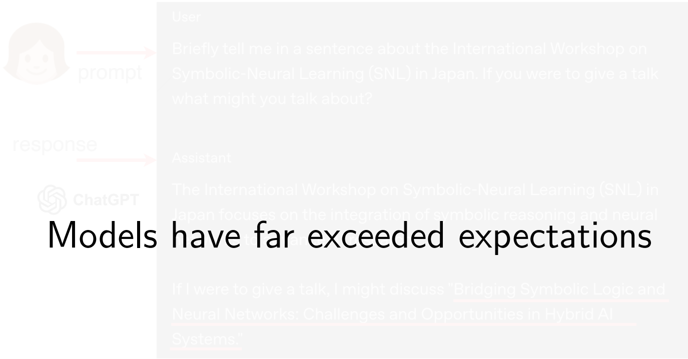


General purpose large language models (LLMs)



- **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

LLMs as agents in competitive environments, games

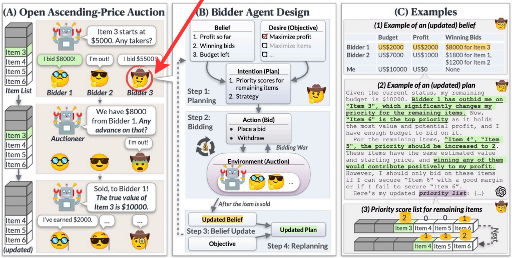


Figure 1: An illustration of AucArena: (A) shows a multi-round, ascending-price auction with an auctioneer announcing the highest bids, where bidders publicly decide after private reasoning. (B) presents a bidder agent structure using the Belief-Desire-Intention Model, involving planning, bidding, belief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

LLMs as agents in competitive environments, games

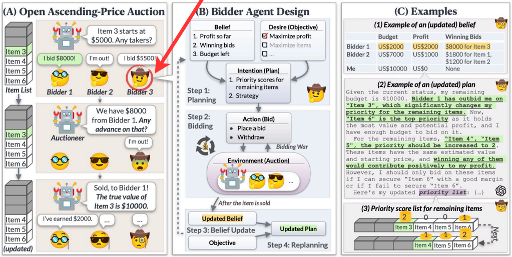
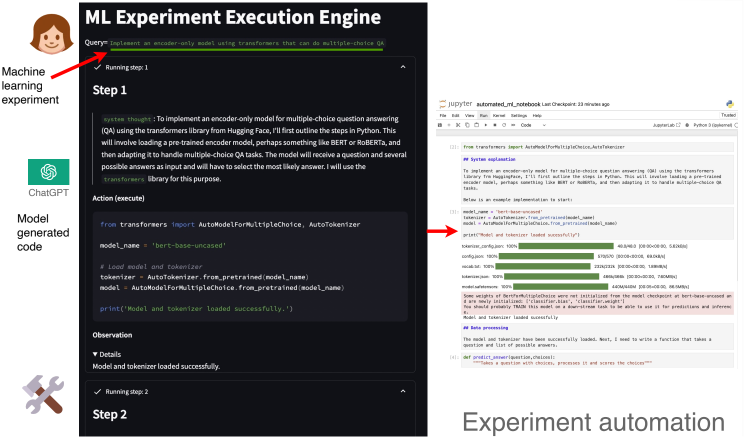


Figure 1: An illustration of AucArena: (A) shows a multi-round, ascending-price auction with an auctioneer announcing the highest bids, where bidders publicly decide after private reasoning. (B) presents a bidder agent structure using the Belief-Desire-Intention Model, involving planning, bidding, belief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



- SUPER (Bogin et al., 2024), benchmark for setting up and executing research code repositories, agent benchmarking (Bragg et al., 2025).

Language models as part of complex systems, **agents**

Language Modeling by Language Models

Junyan Cheng[α*] **Peter Clark**[β] **Kyle Richardson**[β]
Allen Institute for AI[β] Dartmouth College[α]
jc.th@dartmouth.edu kyler@allenai.org
https://github.com/allenai/genesys

Abstract

Can we leverage LLMs to model the process of discovering novel language model architectures? Inspired by real research, we propose a multi-agent LLM approach that simulates the conventional stages of research, from ideation and literature search (proposal stage) to design implementation (code generation), generative pre-training, and downstream evaluation (verification). Using ideas from scaling laws, our system *Genesys* employs a *Ladder of Scales* approach; new designs are proposed, adversarially reviewed, implemented, and selectively verified at increasingly larger model scales (14M∼350M parameters) with a narrowing budget (the number of models we can train at each scale). To help make discovery efficient and factorizable, Genesys uses a novel genetic programming backbone, which we show has empirical advantages over commonly used direct prompt generation workflows (e.g., ∼86% percentage point improvement in successful design generation, a key bottleneck). We report experiments involving 1,162 newly discovered designs (1,062 fully verified through pre-training) and find the best designs to be highly competitive with known architectures (e.g., outperform `GPT2`, `Mamba2`, etc., on 6/9 common benchmarks). We couple these results with comprehensive system-level ablations and formal results, which give broader insights into the design of effective autonomous LLM-driven discovery systems.

Cheng et al. (2025)

TinyScientist: An Interactive, Extensible, and Controllable Framework for Building Research Agents

Haofei Yu[1*] **Keyang Xuan**[1*] **Fenghai Li**[1*]
Kunlun Zhu[1] **Zijie Lei**[1] **Jiaxun Zhang**[1] **Ziheng Qi**[1]
Kyle Richardson[2] **Jiaxuan You**[1]
[1]University of Illinois Urbana-Champaign,
[2]Allen Institute for Artificial Intelligence

Abstract

Automatic research with Large Language Models (LLMs) is rapidly gaining importance, driving the development of increasingly complex workflows involving multi-agent systems, planning, tool usage, code execution, and human-agent interaction to accelerate research processes. However, as more researchers and developers begin to use and build upon these tools and platforms, the complexity and difficulty of extending and maintaining such agentic workflows have become a significant challenge, particularly as algorithms and architectures continue to advance. To address this growing complexity, TinyScientist identifies the essential components of the automatic research workflow and proposes an interactive, extensible, and controllable framework that easily adapts to new tools and supports iterative growth. We provide an open-source codebase[1], an interactive web demonstration[2], and a PyPI Python package[3] to make state-of-the-art auto-research pipelines broadly accessible to every researcher and developer.

end research pipelines (Jansen et al., 2025; Lu et al., 2024; Yamada et al., 2025; Li et al., 2024b; Cheng et al., 2025). Recent advances in this area leverage methods including multi-agent collaboration (Schmidgall et al., 2025), tool using (Skarlinski et al., 2024), and tree-based search (Yamada et al., 2025) to augment its performance.

In spite of this success, however, existing automatic research systems often design and use agentic frameworks that are overly complex and difficult to use and extend without significant technical expertise. These challenges stem from three key issues: (1) ***lack of interactivity***: human researchers struggle to engage with the specific agent's research progress due to the complexity of research intents and unclear communication interfaces (Zou et al., 2025; Liu et al., 2025b), making feedback incorporation challenging. (2) ***limited extensibility***: existing representative frameworks rely on rigid, tool-specific designs (Zhang et al., 2025), making it hard to integrate new tools or adapt to different research domains. (3) ***insufficient controllability***: many

Yu et al. (2025)

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

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

Lots of optimism, hubris, Nobel prizes....

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

Missing **semantic** and **algorithmic** foundations.

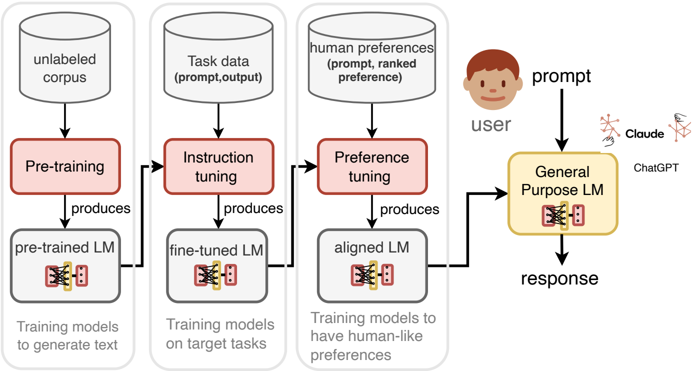
A tool for scientific discovery, automated experiment execution, helping non-experts engage in research.

Missing **semantic** and **algorithmic** foundations.

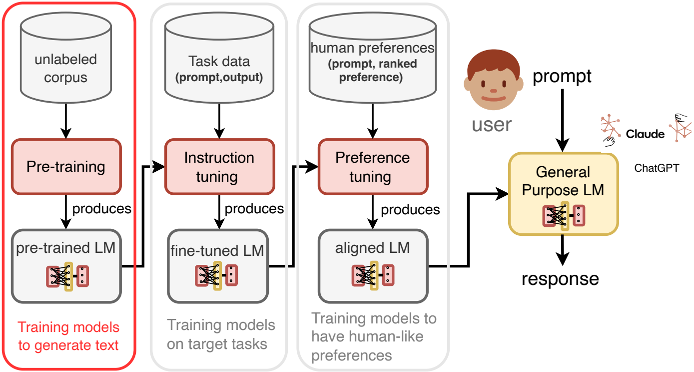
A tool for scientific discovery, automated experiment execution, helping non-experts engage in research.

Can symbolic techniques help?

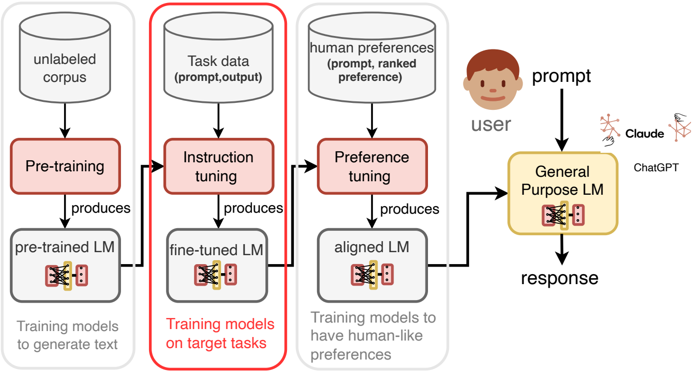
How do we get to general purpose LLMs? **recipe**



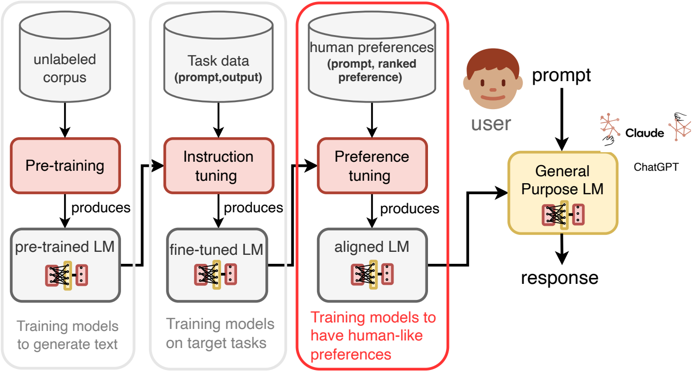
How do we get to general purpose LLMs? **recipe**



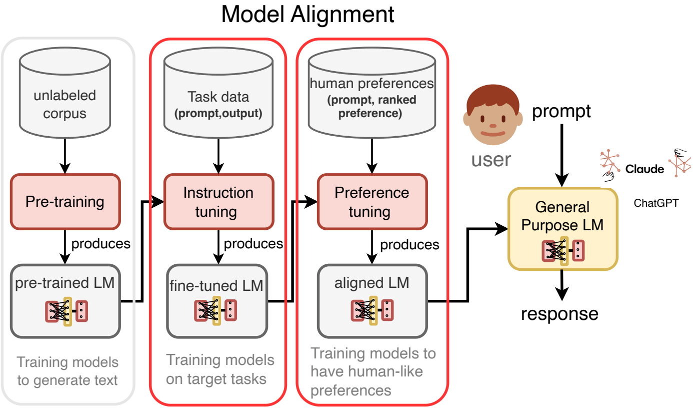
How do we get to general purpose LLMs? **recipe**



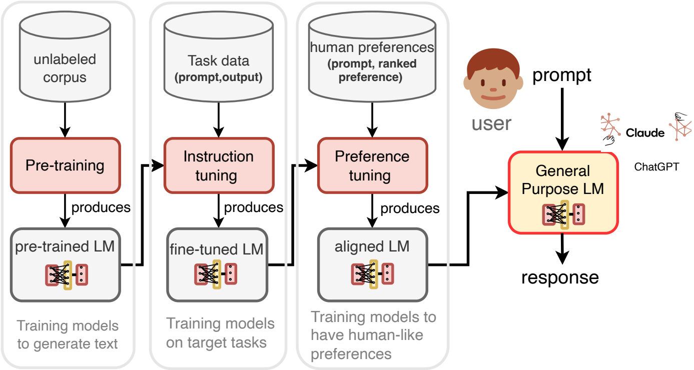
How do we get to general purpose LLMs? **recipe**



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

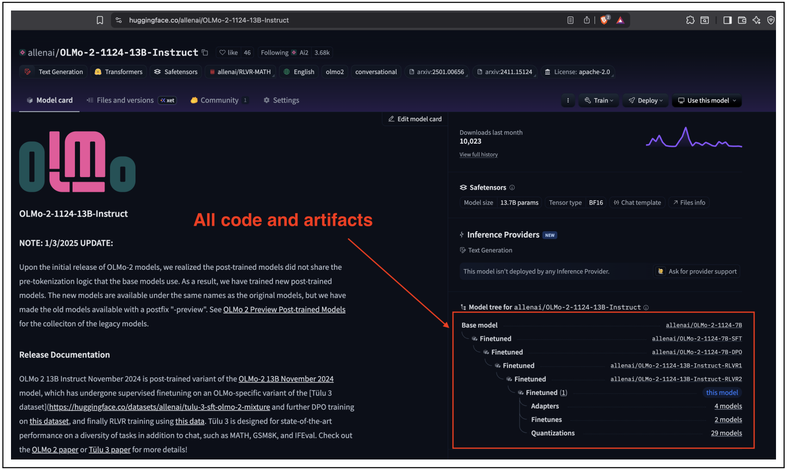


How do we get to general purpose LLMs? **recipe**



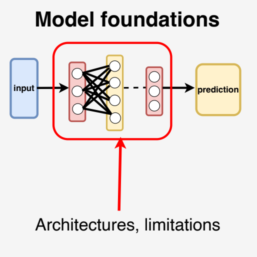
- Rough approximation of the kinds of general purpose models we use.

OLMo: fully open-source general purpose LMs

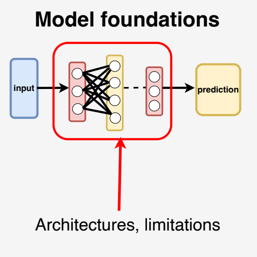


https://allenai.org/olmo

The landscape of Generative AI research

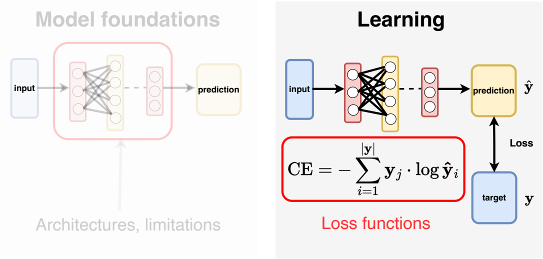


The landscape of Generative AI research



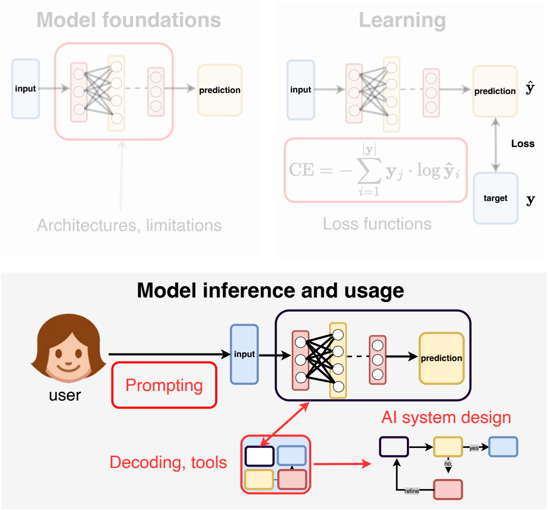
- What model to use? What kinds of computational problems can models solve? Limitations

The landscape of Generative AI research

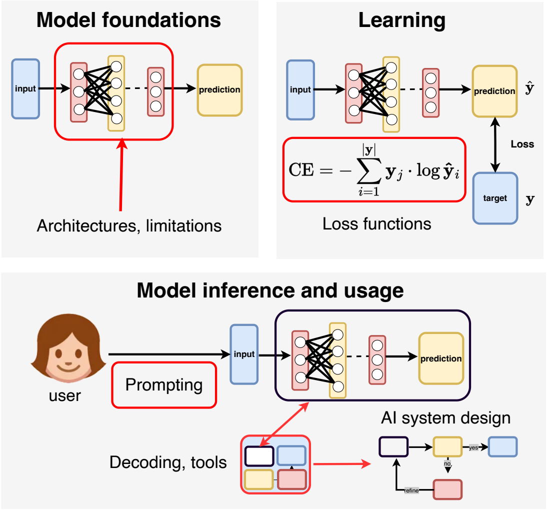


- How do we train and tune models? Loss function design, optimization algorithms.

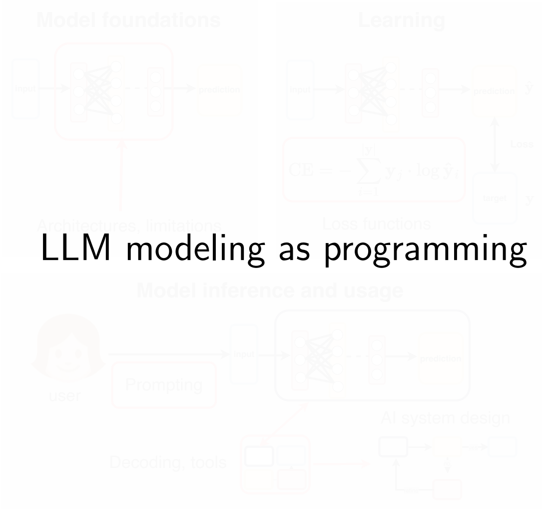
The landscape of Generative AI research



The landscape of Generative AI research

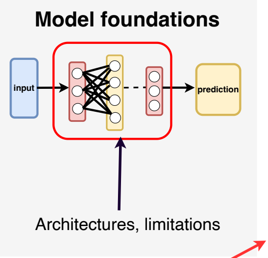


The landscape of Generative AI research



LLM modeling as programming

Programming techniques for model development



Functional Programming (Rasp)

```
opp_index = length - indices - 1;
flip = select(indices, opp_index,==);
reverse = aggregate(flip, tokens);
```

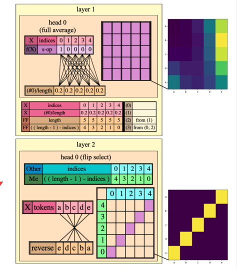
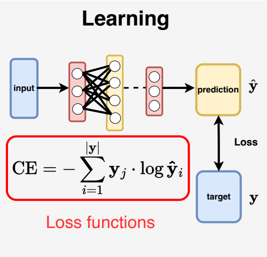


Figure 4: Top: RASP code for computing reverse (e.g., `reverse("abc")="cba"`). Below, its compilation to a transformer architecture (left, obtained through `draw(reverse,"abcde")` in the RASP REPL), and the attention heatmaps of a transformer trained on the same task (right), both visualised on the same input. Visually, the atten-

- Programming languages for expressing transformer computation (Weiss et al., 2021; Yang et al., 2024; Yang and Chiang, 2024).

Programming techniques for model development

Declarative style programming (Scallop)

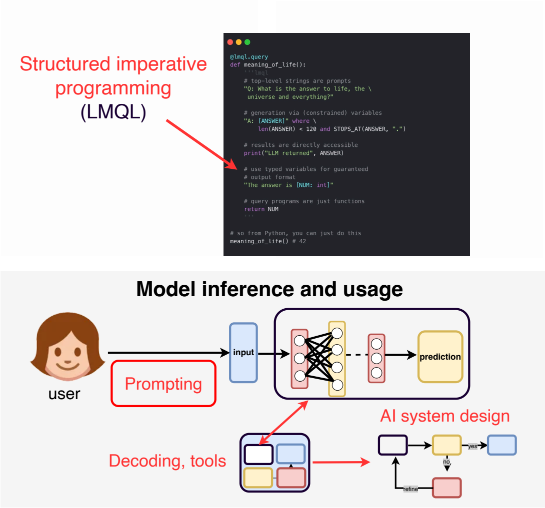



```
// File path_planner.scl
type actor(x: i32, y: i32), goal(x: i32, y: i32), enemy(x: i32, y: i32)

const UP = 0, DOWN = 1, RIGHT = 2, LEFT = 3
rel safe_cell(x, y) = range(0, 5, x), range(0, 5, y), not enemy(x, y)
rel edge(x, y, x, yp, UP) = safe_cell(x, y), safe_cell(x, yp), yp == y + 1
// Rules for DOWN, RIGHT, and LEFT edges are omitted...

rel next_pos(p, q, a) = actor(x, y), edge(x, y, p, q, a)
rel path(x, y, x, y) = next_pos(x, y, _)
rel path(x1, y1, x3, y3) = path(x1, y1, x2, y2), edge(x2, y2, x3, y3, _)
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.

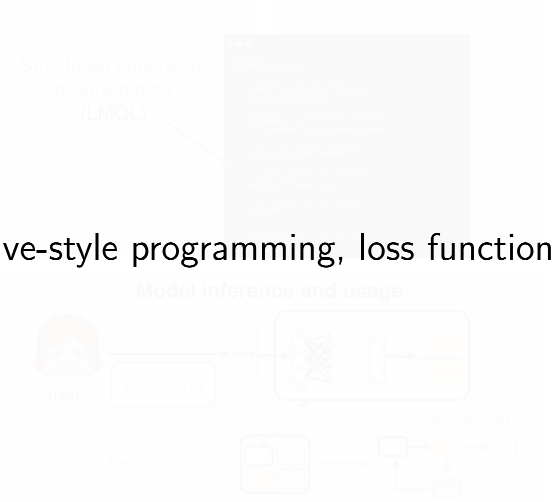
- Loss design via logical and probabilistic programming, neuro-symbolic modeling (Li et al., 2023; Manhaeve et al., 2018).

Programming techniques for model development



- Prompting as (imperative) programming (Beurer-Kellner et al., 2023).

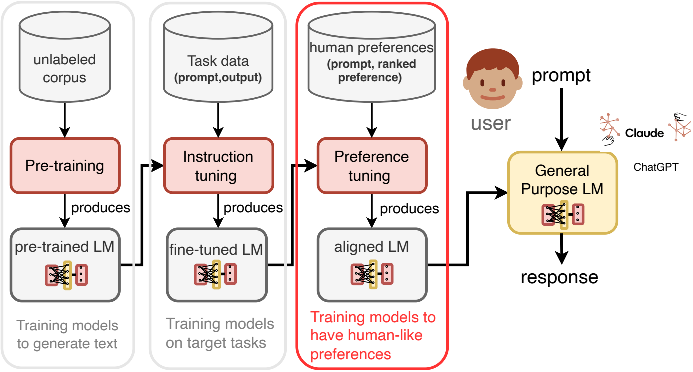
Programming techniques for model development



Declarative-style programming, loss function design.

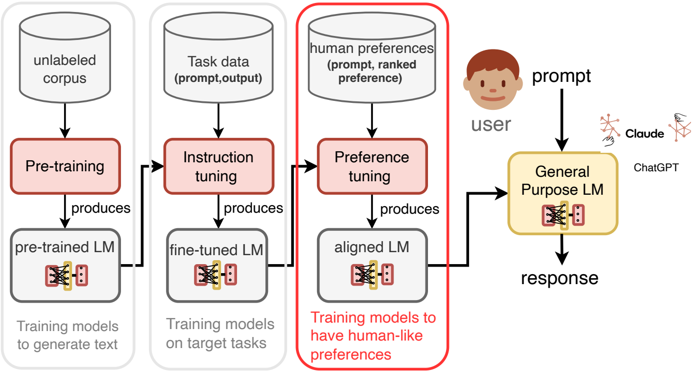
- Prompting as (imperative) programming (Beurer-Kellner et al., 2023).

Declarative-style programming for preference modeling



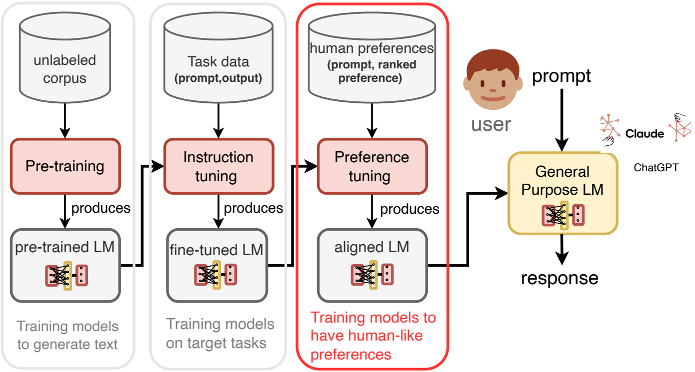
Today: Logical and probabilistic programming of preference losses, semantic characterizations.

Declarative-style programming for preference modeling



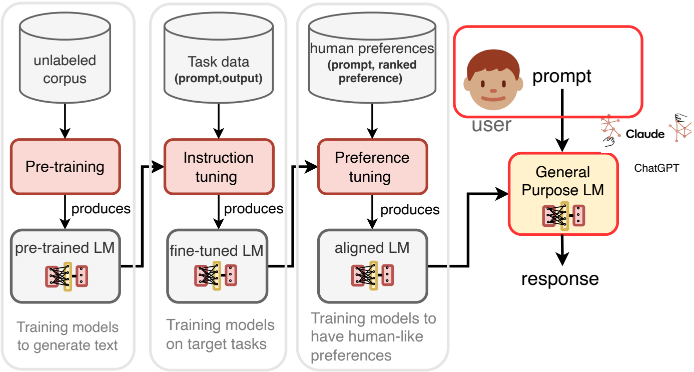
Questions: What do we do when we tune models to preferences? Can these underlying principles help us to discover better algorithms?

Declarative-style programming for preference modeling



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

~~Probabilistic programming for other applications~~



~~**If time permits**: Probabilistic programming techniques and languages for prompting and chain-of-thought.~~

Probabilistic programming for other applications

Preference learning and alignment

Offline preference alignment in a nutshell

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

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

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

Offline preference alignment in a nutshell

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

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

optimize a policy model $y \sim \pi_\theta(\cdot \mid x)$ (**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:

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

optimize a policy model $y \sim \pi_\theta(\cdot \mid x)$ (**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, datasets are very noisy and aggregate many different kinds of preferences.

Offline preference alignment in a nutshell

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

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

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

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

1. Unclear what is actually in our datasets

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, datasets are very noisy and aggregate many different kinds of preferences.

Direct Preference Alignment (DPA) approaches

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

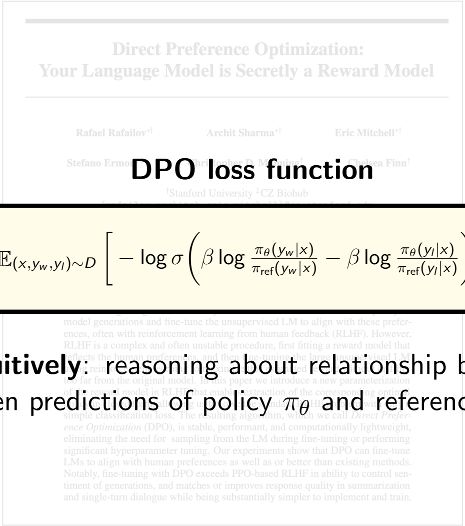
Rafael Rafailov*† **Archit Sharma***† **Eric Mitchell***†

Stefano Ermon†‡ **Christopher D. Manning**† **Chelsea Finn**†

†Stanford University ‡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 Optimization:
Your Language Model is Secretly a Reward Model

Rafael Rafailov*† Archit Sharma*† Eric Mitchell*†

†Stanford University ‡CZ Biohub

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

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 function

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

Intuitively: reasoning about relationship between predictions of policy π_θ and reference π_{ref}.

Direct Preference Alignment (DPA) approaches

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

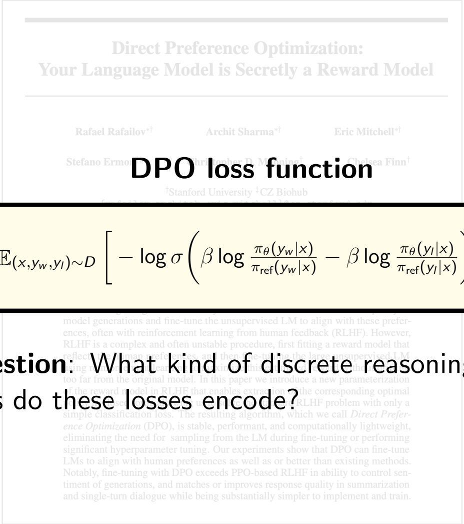
Rafael Rafailov*† Archit Sharma*† Eric Mitchell*†
Stefano Ermon†‡ Christopher D. Manning† Chelsea Finn†
†Stanford University ‡CZ Biohub
{rafailov,architsh,eric.mitchell}@cs.stanford.edu

2. These equations are not easy to understand

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



DPO loss function

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

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

The many varieties of DPO

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

Rafael Rafailov[*†] **Archit Sharma**[*†] **Eric Mitchell**[*†]

Stefano Ermon[†‡] **Christopher D. Manning**[†] **Chelsea Finn**[†]

[†]Stanford University [‡]CZ Biohub
{rafailov,architsh,eric.mitchell}@cs.stanford.edu

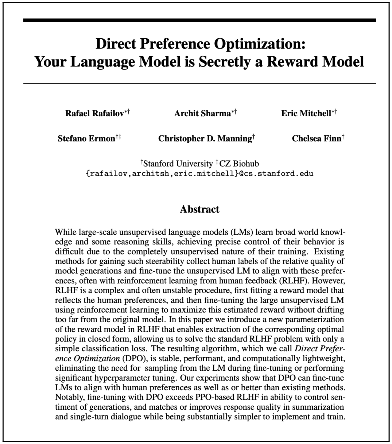
Abstract

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[*†] **Archit Sharma**[*†] **Eric Mitchell**[*†]

Stefano Ermon[†‡] **Christopher D. Manning**[†] **Chelsea Finn**[†]

[†]Stanford University [‡]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]	$\max\left(0, -\frac{1}{\lvert y_w\rvert}\log \pi_\theta(y_w\|x) + \frac{1}{\lvert y_l\rvert}\log \pi_\theta(y_l\|x)\right) - \lambda \log \pi_\theta(y_w\|x)$
SLiC-HF [96]	$\max(0, \delta - \log \pi_\theta(y_w\|x) + \log \pi_\theta(y_l\|x)) - \lambda \log \pi_\theta(y_w\|x)$
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]	$\left(\log \frac{\pi_\theta(y_w\|x)}{\pi_{\text{ref}}(y_w\|x)} - \log \frac{\pi_\theta(y_l\|x)}{\pi_{\text{ref}}(y_l\|x)} - \frac{1}{2\tau}\right)^2$
CPO [88]	$-\log \sigma(\beta \log \pi_\theta(y_w\|x) - \beta \log \pi_\theta(y_l\|x)) - \lambda \log \pi_\theta(y_w\|x)$
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 $z_{\text{ref}} = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\beta \mathrm{KL}\left(\pi_\theta(y\|x) \|\| \pi_{\text{ref}}(y\|x)\right)\right]$
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 $p_\theta(y\|x) = \exp\left(\frac{1}{\lvert y\rvert}\log \pi_\theta(y\|x)\right)$
R-DPO [64]	$-\log \sigma\left(\beta \log \frac{\pi_\theta(y_w\|x)}{\pi_{\text{ref}}(y_w\|x)} - \beta \log \frac{\pi_\theta(y_l\|x)}{\pi_{\text{ref}}(y_l\|x)} + (\alpha\lvert y_w\rvert - \alpha\lvert y_l\rvert)\right)$
SimPO	$-\log \sigma\left(\frac{\beta}{\lvert y_w\rvert}\log \pi_\theta(y_w\|x) - \frac{\beta}{\lvert y_l\rvert}\log \pi_\theta(y_l\|x) - \gamma\right)$

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*† Archit Sharma*† Eric Mitchell*†
Stefano Ermon†‡ Christopher D. Manning† Chelsea Finn†
†Stanford University ‡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

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

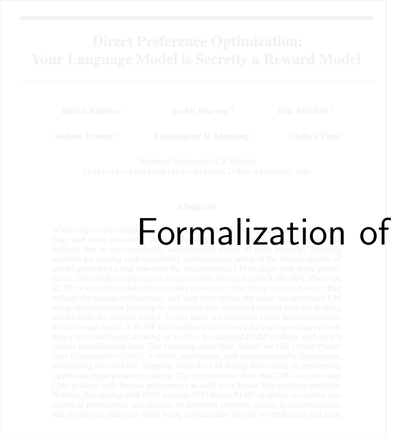
DPO variants

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

from Meng et al. (2024)

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

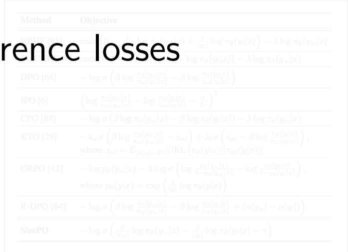
The many varieties of DPO



DPO loss

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

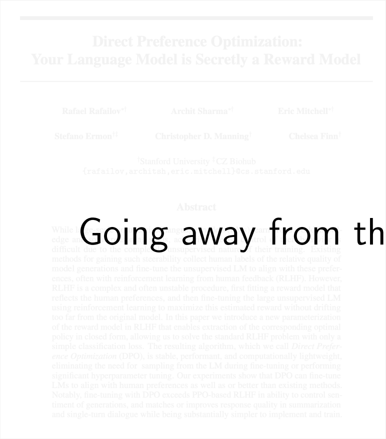
DPO variants



from Meng et al. (2024)

Formalization of preference losses

The many varieties of DPO


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

Rafael Rafailov*† Archit Sharma*† Eric Mitchell*†

Stefano Ermon†‡ Christopher D. Manning† Chelsea Finn†

†Stanford University ‡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]	$\max\left(0, -\frac{1}{\lvert y_w\rvert}\log \pi_\theta(y_w\|x) + \frac{1}{\lvert y_l\rvert}\log \pi_\theta(y_l\|x)\right) - \lambda \log \pi_\theta(y_w\|x)$
SLiC-HF [96]	$\max(0, \delta - \log \pi_\theta(y_w\|x) + \log \pi_\theta(y_l\|x)) - \lambda \log \pi_\theta(y_w\|x)$
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]	$\left(\log \frac{\pi_\theta(y_w\|x)}{\pi_{\text{ref}}(y_w\|x)} - \log \frac{\pi_\theta(y_l\|x)}{\pi_{\text{ref}}(y_l\|x)} - \frac{1}{2\tau}\right)^2$
CPO [88]	$-\log \sigma(\beta \log \pi_\theta(y_w\|x) - \beta \log \pi_\theta(y_l\|x)) - \lambda \log \pi_\theta(y_w\|x)$
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 $p_\theta(y\|x) = \exp\left(\frac{1}{\lvert y\rvert}\log \pi_\theta(y\|x)\right)$
R-DPO [64]	$-\log \sigma\left(\beta \log \frac{\pi_\theta(y_w\|x)}{\pi_{\text{ref}}(y_w\|x)} - \beta \log \frac{\pi_\theta(y_l\|x)}{\pi_{\text{ref}}(y_l\|x)} + (\alpha\lvert y_w\rvert - \alpha\lvert y_l\rvert)\right)$
SimPO	$-\log \sigma\left(\frac{\beta}{\lvert y_w\rvert}\log \pi_\theta(y_w\|x) - \frac{\beta}{\lvert y_l\rvert}\log \pi_\theta(y_l\|x) - \gamma\right)$

from Meng et al. (2024)

Going away from these opaque equations

Preference learning as a discrete reasoning problem

Loss Function ℓ

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

Preference learning as a discrete reasoning problem

Loss Function ℓ

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

Two models, four predictions

Preference learning as a discrete reasoning problem

Loss Function ℓ

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

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 P

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

Loss Function ℓ

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

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 P

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

Loss Function ℓ

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

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 P

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

Loss Function ℓ

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

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 P

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

Loss Function ℓ

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

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 P

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

Loss Function ℓ

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

Formula P $\xrightarrow{\text{interpret}}$ differential logic $\xrightarrow{\text{derive}}$ **Loss** ℓ

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.

Logic as loss, learning to satisfy (Marra et al., 2024)

Preference learning as a discrete reasoning problem

Symbolic Program P

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

Loss Function ℓ

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

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.

Preference learning as a discrete reasoning problem

Symbolic Program P

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

Loss Function ℓ

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

Decompilation $\longleftrightarrow$ Compilation

$$\begin{tikzcd}
\textbf{Formula } P \arrow[r, "\text{interpret}"] & \text{differential logic} \arrow[r, "\text{derive}"] & \textbf{Loss } \ell \arrow[ll, bend left, red]
\end{tikzcd}$$

Loss as logic (Richardson et al., 2025)

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

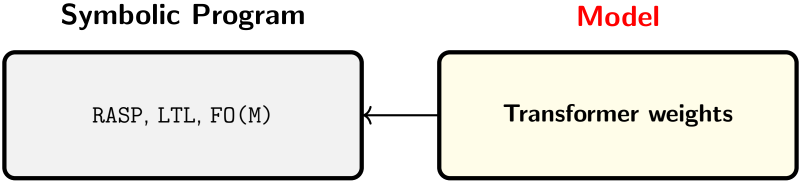
  1. **Compilation**: Translating specifications into loss, well studied.
  2. **Decompilation**:Losses to specifications (inverse), less explored.

Formal analysis via decompilation in general

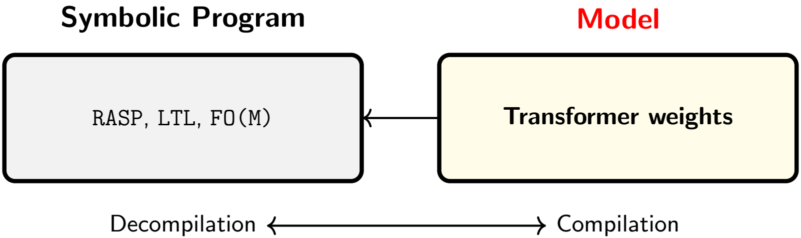
Model

Transformer weights

Formal analysis via decompilation in general



Formal analysis via decompilation in general



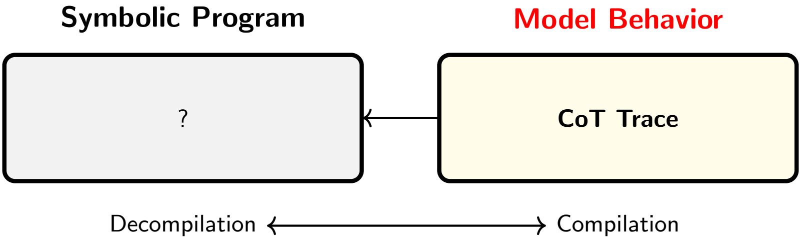
- We know what the *target languages* are (Weiss et al., 2021; Merrill and Sabharwal, 2023; Yang and Chiang, 2024), how to compile, decompile (Friedman et al., 2023).

Formal analysis via decompilation in general

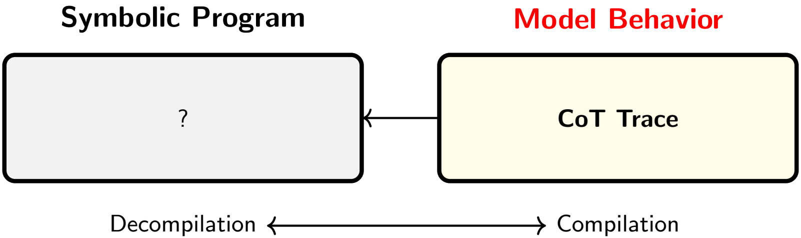
Model Behavior

CoT Trace

Formal analysis via decompilation in general



Formal analysis via decompilation in general



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

  Language model programming: the languages and formal interfaces used for for doing such analysis (Richardson and Wijnholds, 2025).

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

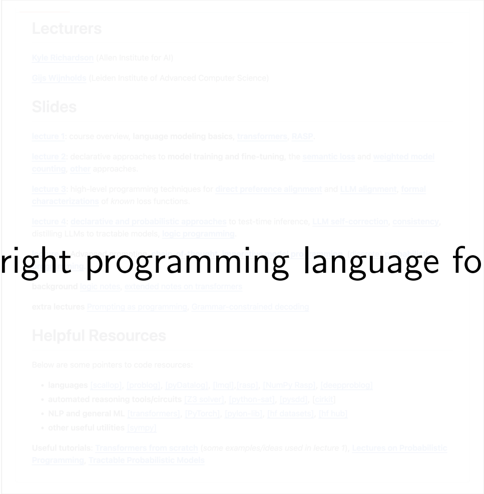
Below are some pointers to code resources:

- **languages** [scallop], [problog], [pyDatalog], [lmql],[rasp], [NumPy Rasp], [deepproblog]
- **automated reasoning tools/circuits** [Z3 solver], [python-sat], [pysdd], [cirkit]
- **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

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.

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], [cirkit]
- **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

What is the right programming language for preference?

https://github.com/yakazimir/LMProgramming

What do these programs tell us?

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

What do these programs tell us?

Model predicts loser   Model predicts winner

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

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

What do these programs tell us?

Model predicts loser

Model predicts winner

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

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

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

What do these programs tell us?

Model predicts loser

Model predicts winner

```
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.

Conceptually: Predictions are connected through Boolean operators, express constraints on predictions; ρ_θ as formulas.

Uncovering the natural logic of these algorithms

Model predicts loser

Model predicts winner

```
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

Model predicts loser

Model predicts winner

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

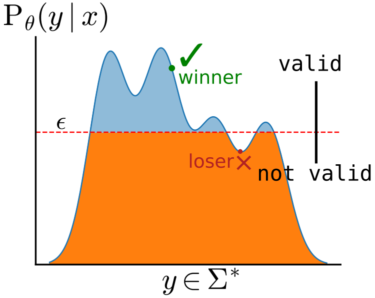
Whenever the model deems the loser to be a valid generation, it should deem the winner to be valid too.

Running example: This program and semantics is foundational to many DPO-style losses.

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.

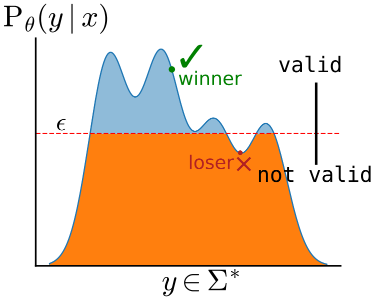


Model behavior: Programs tell us about the structure of the model's output distribution (right).

Uncovering the natural logic of these algorithms

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

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

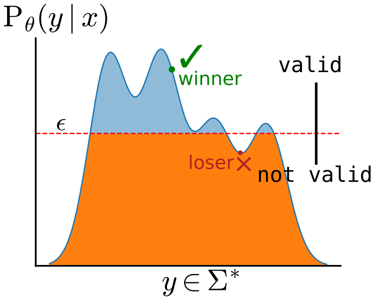


Model behavior: Programs tell us about the structure of the model's output distribution (right).

Uncovering the natural logic of these algorithms

```
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.

Compilation and decompilation again

Compilation

P

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

$$\ell(D,\theta) = -\log \underbrace{P_\theta(\mathrm{P} \mid D,\theta)}_{\text{probabilistic logic}}$$

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

Compilation and decompilation again

Compilation

P

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

$$\ell(D, \theta) = -\log \underbrace{P_\theta(\mathsf{P} \mid D, \theta)}_{\text{probabilistic logic}}$$

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

What we did: defined a novel probabilistic logic for preference modeling, interpret formulas in that logic to derive differentiable losses.

Compilation and decompilation again

Decompilation

P

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

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

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

Compilation and decompilation again

Decompilation

P

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

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

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

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

Compilation and decompilation again

Decompilation

P

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

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

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

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

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

Illustration of approach and results (Richardson et al., 2025)

Input Loss $\ell_{\texttt{ORPO}}$

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

Illustration of approach and results (Richardson et al., 2025)

Input Loss $\ell_{\texttt{ORPO}}$

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

$$\downarrow$$

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

Core loss equation

Illustration of approach and results (Richardson et al., 2025)

Input Loss $\ell_{\texttt{ORPO}}$

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

$$\frac{\rho^t_\theta}{\rho^b_\theta} = \frac{P_\theta(y_w|x)(1-P_\theta(y_l|x))}{P_\theta(y_l|x)(1-P_\theta(y_w|x))}$$

Core loss equation

$$\textsc{Sem}(\rho^t_\theta) = \texttt{M}(\mathsf{x}, \mathsf{y}_w) \wedge \neg\texttt{M}(\mathsf{x}, \mathsf{y}_l)$$
$$\textsc{Sem}(\rho^b_\theta) = \texttt{M}(\mathsf{x}, \mathsf{y}_l) \wedge \neg\texttt{M}(\mathsf{x}, \mathsf{y}_w)$$

Compositional translation

Illustration of approach and results (Richardson et al., 2025)

Input Loss $\ell_{\texttt{ORPO}}$

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

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

Core loss equation

$$\textsc{Sem}(\rho_\theta^t) = \texttt{M}(\mathsf{x},\mathsf{y}_w) \wedge \neg\texttt{M}(\mathsf{x},\mathsf{y}_l)$$

$$\textsc{Sem}(\rho_\theta^b) = \texttt{M}(\mathsf{x},\mathsf{y}_l) \wedge \neg\texttt{M}(\mathsf{x},\mathsf{y}_w)$$

Compositional translation

P:= $\texttt{Implies(}$
$\texttt{M(x,}y_l\texttt{), M(x,}y_w\texttt{))}$
$\mathsf{P}_{\texttt{C}}$:= $\texttt{XOR(M(x,}y_l\texttt{),M(}y_w\texttt{)))}$
$\mathsf{P}_{\texttt{A}} := \perp$

Preference structure $\overline{\mathsf{P}}$

Illustration of approach and results (Richardson et al., 2025)

compilation $\xleftarrow{\text{Thm.}}$ *decompilation*

Input Loss $\ell_{\texttt{ORPO}}$

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

Preference structure $\overline{\mathsf{P}}$

```
P:= Implies(
  M(x,y_l), M(x,y_w))
P_C := XOR(M(x,y_l),M(y_w)))
P_A := ⊥
```

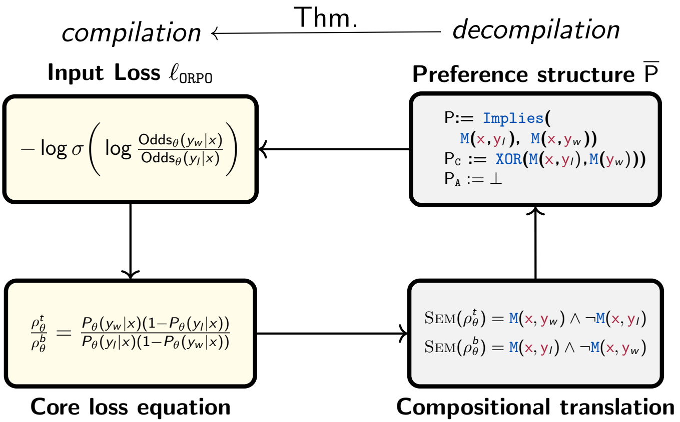
Core loss equation

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

Compositional translation

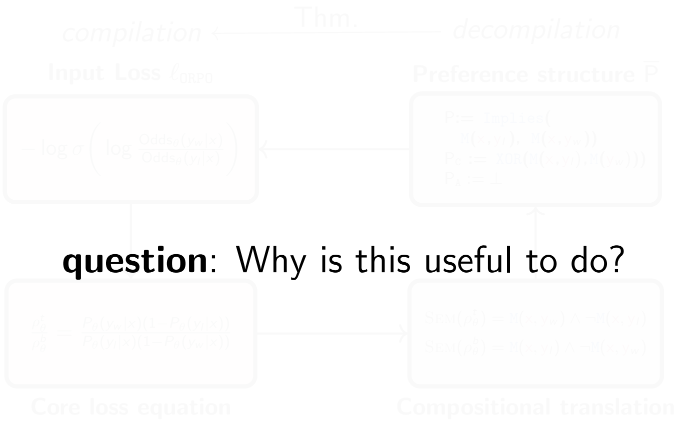
$$\textsc{Sem}(\rho_\theta^t) = \texttt{M}(x, y_w) \wedge \neg\texttt{M}(x, y_l)$$
$$\textsc{Sem}(\rho_\theta^b) = \texttt{M}(x, y_l) \wedge \neg\texttt{M}(x, y_w)$$

Illustration of approach and results (Richardson et al., 2025)



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

Illustration of approach and results (Richardson et al., 2025)



question: Why is this useful to do?

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

Illustration of approach and results (Richardson et al., 2025)

compilation $\xleftarrow{\text{Thm.}}$ *decompilation*

Input Loss $\ell_{\mathtt{ORPO}}$

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

Preference structure $\overline{\mathsf{P}}$

$\mathsf{P} := \mathtt{Implies(}$
$\quad \mathtt{M}(x,y_l),\ \mathtt{M}(x,y_w))$
$\mathsf{P}_C := \mathtt{XOR}(\mathtt{M}(x,y_l),\mathtt{M}(y_w)))$
$\mathsf{P}_A := \bot$

How many preference loss functions are there?

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

Core loss equation

$\textsc{Sem}(\rho_\theta^t) = \mathtt{M}(x,y_w) \wedge \neg\mathtt{M}(x,y_l)$
$\textsc{Sem}(\rho_\theta^b) = \mathtt{M}(x,y_l) \wedge \neg\mathtt{M}(x,y_w)$

Compositional translation

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

Why is this useful? understanding the space

$P^{(1)}$

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

$P^{(2)}$

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

Boolean functions, **2 variables**

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

Why is this useful? understanding the space

$P^{(1)}$

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

$P^{(2)}$

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

Boolean functions, **2 variables**

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

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

Why is this useful? understanding the space

$\mathsf{P}^{(1)}$

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

$\mathsf{P}^{(2)}$

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

Boolean functions, **2 variables**

$\mathtt{M}(x, y_w)$	$\mathtt{M}(x, y_l)$	$\mathsf{P}^{(1)}$	$\mathsf{P}^{(2)}$
T	T	✓	✗
T	F	✓	✓
F	T	✗	✗
F	F	✓	✗

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

Why is this useful? understanding the space

$P^{(1)}$

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

$P^{(2)}$

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

Boolean functions, **2 variables**

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

no reference: 256 losses

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

Loss functions as truth tables

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

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	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))
)
```

4 variables

$\mathtt{Ref}(x, y_w)$	$\mathtt{M}(x, y_l)$	$\mathtt{Ref}(x, y_l)$	$\mathtt{M}(x, y_w)$
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

answer: loads.

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))
```

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
[illegible]	[illegible]	[illegible]	[illegible]
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

question: How are losses related to one another?

Why is this useful? understanding the structure

$P^{(1)}$

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

$P^{(2)}$

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

semantics: $P^{(2)} \models P^{(1)}$

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

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

$\mathsf{P}^{(1)}$

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

$\mathsf{P}^{(2)}$

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

semantics: $\mathsf{P}^{(2)} \models \mathsf{P}^{(1)}$

$\mathtt{M}(x, y_w)$	$\mathtt{M}(x, y_l)$	$\mathsf{P}^{(1)}$	$\mathsf{P}^{(2)}$
T	T	✓	✗
T	F	✓	✓
F	T	✗	✗
F	F	✓	✗

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

Why is this useful? understanding the structure

$\mathsf{P}^{(1)}$

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

$\mathsf{P}^{(2)}$

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

semantics: $\mathsf{P}^{(2)} \models \mathsf{P}^{(1)}$

$\mathtt{M}(x, y_w)$	$\mathtt{M}(x, y_l)$	$\mathsf{P}^{(1)}$	$\mathsf{P}^{(2)}$
T	T	$\checkmark$	$\times$
T	F	$\checkmark$	$\checkmark$
F	T	$\times$	$\times$
F	F	$\checkmark$	$\times$

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

Why is this useful? understanding the structure

$P^{(1)}$

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

$P^{(2)}$

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

semantics: $P^{(2)} \models P^{(1)}$

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

answer: Losses are related through their semantics

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

$P^{(1)}$

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

$P^{(2)}$

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

semantics: $P^{(2)} \models P^{(1)}$

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

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

DPO Loss

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

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))
)
```

modify

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

DPO Loss

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

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

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

modify

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

Novel loss

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

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

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

modify

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

Novel loss

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

- High-level programming language for defining new losses.

Deriving new losses symbolically, from first principles

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

How does the logic work?

$\mathtt{M}(x, y_w)$	$\mathtt{M}(x, y_l)$	CPO	ORPO	unCPO
T	T	✓ ✗		✓
T	F	✓	✓	✓
F	T	✗	✗	✗
F	F			✓

P

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

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?

$\texttt{M}(x, y_w)$	$\texttt{M}(x, y_l)$	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.

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

How does the logic work?

$\texttt{M}(x, y_w)$	$\texttt{M}(x, y_l)$	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.

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

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

ELSEVIER

Available online at www.sciencedirect.com

ScienceDirect

Artificial Intelligence 172 (2008) 772–799

Artificial Intelligence

www.elsevier.com/locate/artint

On probabilistic inference by weighted model counting ☆

Mark Chavira *, Adnan Darwiche

algorithmic foundation

Computer Science Department, UCLA, Los Angeles, CA 90095, USA

Received 25 August 2006; received in revised form 22 July 2007; accepted 5 November 2007

A Semantic Loss Function for Deep Learning with Symbolic Knowledge

Losses computed from weighted model counts

Jingyi Xu[1] **Zilu Zhang**[2] **Tal Friedman**[1] **Yitao Liang**[1] **Guy Van den Broeck**[1]

Abstract

This paper develops a novel methodology for using symbolic knowledge in deep learning. From first principles, we derive a semantic loss function that bridges between neural output vectors and logical constraints. This loss function captures how close the neural network is to satisfying the constraints on its output. An experimental evaluation shows that it effectively guides the learner to achieve (near-)state-of-the-art results on semi-supervised multi-class classification. Moreover, it significantly increases the ability of the neural network to predict structured objects, such as rankings and paths. These discrete concepts are tremendously difficult to learn, and benefit from a tight integration of deep learning and symbolic reasoning methods.

This paper considers learning in domains where we have symbolic knowledge connecting the different outputs of a neural network. This knowledge takes the form of a constraint (or sentence) in Boolean logic. It can be as simple as an exactly-one constraint for one-hot output encodings, or as complex as a structured output prediction constraint for intricate combinatorial objects such as rankings, subgraphs, or paths. Our goal is to augment neural networks with the ability to learn how to make predictions subject to these constraints, and use the symbolic knowledge to improve the learning performance.

Most neuro-symbolic approaches aim to simulate or learn symbolic reasoning in an end-to-end deep neural network, or capture symbolic knowledge in a vector-space embedding. This choice is partly motivated by the need for smooth *differentiable* models; adding symbolic reasoning code (e.g., SAT solvers) to a deep learning pipeline de-

How does the logic work?

$\texttt{M}(x, y_w)$	$\texttt{M}(x, y_l)$	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.

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

$$\underbrace{\ell_x}_{\text{column}} := \underbrace{-\log \sigma\left(\log \frac{\sum \checkmark_w}{\sum \times_w}\right)}_{\text{log ratio of } \checkmark_w \text{ and } \times_w \textit{ counts}}$$

How does the logic work?

$\mathtt{M}(x, y_w)$	$\mathtt{M}(x, y_l)$	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.

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

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

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

How does the logic work?

$\texttt{M}(x, y_w)$	$\texttt{M}(x, y_l)$	CPO	ORPO	unCPO
T	T	✓ ✗		✓
T	F	✓	✓	✓
F	T	✗	✗	✗
F	F			✓

P

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

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

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

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

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

observation: losses differ in hard constraints

How does the logic work?

$\texttt{M}(x, y_w)$	$\texttt{M}(x, y_l)$	CPO	ORPO	unCPO
T	T	✓ ✗		✓
T	F	✓	✓	✓
F	T	✗	✗	✗
F	F			✓

P

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

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

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

Loss	Representation $\overline{\mathsf{P}}$
CE	$\mathsf{P} := \textbf{M}(x, y_w),\ \mathsf{P_C} := \bot$
CEUnl	$\mathsf{P} := \textbf{And}(\textbf{M}(x, y_w), \textbf{Not}(\textbf{M}(x, y_l)))$ $\mathsf{P_C} := \bot$
CPO	;; core semantic formula $\mathsf{P} := \textbf{Implies}(\textbf{M}(x, y_l), \textbf{M}(x, y_w))$;; one−true constraint $\mathsf{P_C} := \textbf{Or}(\textbf{M}(x, y_l), \textbf{M}(x, y_w))$
ORPO	$\mathsf{P} := \textbf{Implies}(\textbf{M}(x, y_l), \textbf{M}(x, y_w))$;; one−hot constraint $\mathsf{P_C} := \textbf{XOR}(\textbf{M}(x, y_l), \textbf{M}(x, y_w))$

How does the logic work?

$\texttt{M}(x, y_w)$	$\texttt{M}(x, y_l)$	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.

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

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

$$\overline{\mathsf{P}} := \Big(\underbrace{\mathsf{P}}_{\text{core}}, \underbrace{\mathsf{P}_{\mathbf{C}}, \mathsf{P}_{\mathbf{A}}}_{\text{constraints}} \Big)$$

How does the logic work?

$\texttt{M}(x, y_w)$	$\texttt{M}(x, y_l)$	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.

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

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

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

How does the logic work?

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.

observation: real losses are highly constrained

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

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

How does the logic work?

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: $\underbrace{\mathtt{M}(x, y_l) \rightarrow \mathtt{M}(x, y_w)}_{\text{goal of optimization}} \equiv \neg \mathtt{M}(x, y_l) \vee \mathtt{M}(x, y_w)$

$$\{✓, X\}_w := \prod \pi_\theta(y \mid x) \cdot \prod_{\neg \mathtt{M}(x,y)} 1 - \pi_\theta(y \mid x)$$

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

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

How does the logic work?

$\mathtt{M}(x, y_w)$	$\mathtt{M}(x, y_l)$	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.

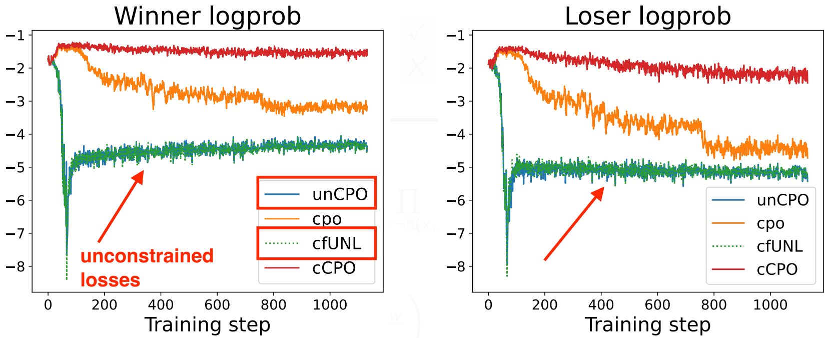
note: $\mathtt{M}(x, y_l) \rightarrow \mathtt{M}(x, y_w) \equiv \underbrace{\neg \mathtt{M}(x, y_l) \vee \mathtt{M}(x, y_w)}_{\text{drive probability to zero}}$

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

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

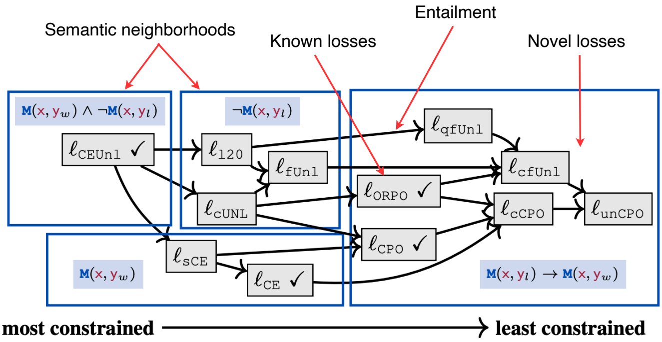
$$= \underbrace{-\log \sigma \left(\log \frac{\pi_\theta(y_l \mid x)\pi_\theta(y_w \mid x) + (1 - \pi_\theta(y_l \mid x))}{\pi_\theta(y_l \mid x)(1 - \pi_\theta(y_w \mid x))} \right)}_{\text{novel loss without constraints, } P_\theta(\mathsf{P} \mid \top)}$$

Training dynamics



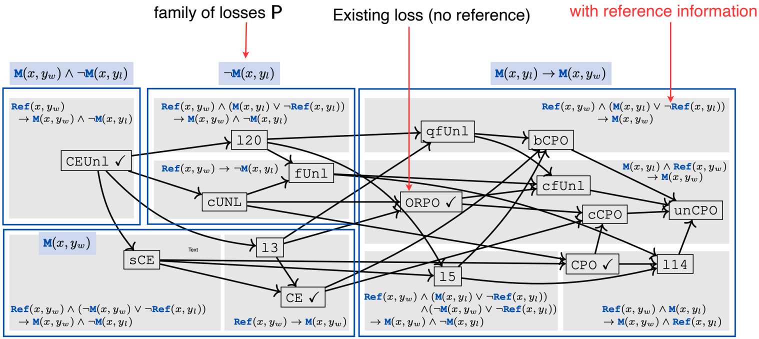
Constrainedness is an important property

The **no reference** loss landscape



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

The **reference** loss landscape



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

question: Are any of these losses good?

Characterizing dataset semantics?

known loss (baseline)

data sub-categories

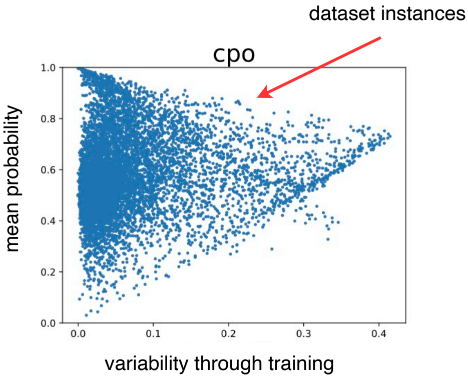
novel losses

loss	WR% (ℓ_{cpo})	evol	false-qa	flan	sharegpt	ultrachat
$\ell_{\texttt{cfUNL}}$	46.1 (±0.4)	46.1 (±2.2)	51.6 (±2.9)	46.4 (±1.7)	46.2 (±1.2)	44.1 (±1.0)
$\ell_{\texttt{qfUNL}}$	48.9 (±0.8)	45.3 (±1.9)	34.7 (±6.3)	57.9 (±1.2)	46.8 (±2.4)	41.3 (±1.4)
$\ell_{\texttt{cCPO}}$	52.0 (±0.6)	50.7 (±0.5)	50.2 (±0.7)	57.2 (±1.1)	47.2 (±1.8)	53.1 (±1.9)
$\ell_{\texttt{unCPO}}$	46.0 (±0.2)	45.8 (±0.3)	52.1 (±3.0)	45.7 (±0.6)	46.2 (±2.1)	44.8 (±2.1)

Table 5: Comparing performance of `Qwen-0.5B` tuned on new losses (rows) against $\ell_{\texttt{CPO}}$ based on aggregate win-rate (WR % (std)) on `ultrafeedback` test (second column) and different test subsets (columns 2-6).

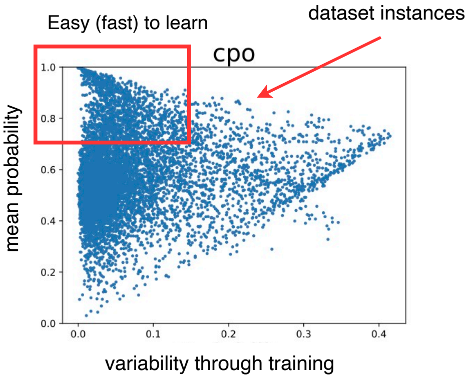
- **Finding**: Different losses perform better/worse on different subsets of data, reflecting the different semantics in preference data.

Characterizing dataset semantics?



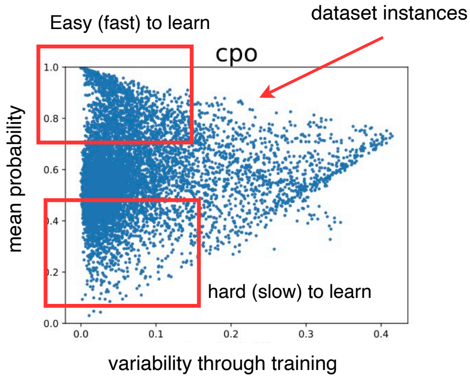
- Tracking instance-level training dynamics (Swayamdipta et al., 2020) and behavior across losses, use to reverse engineer data semantics.

Characterizing dataset semantics?



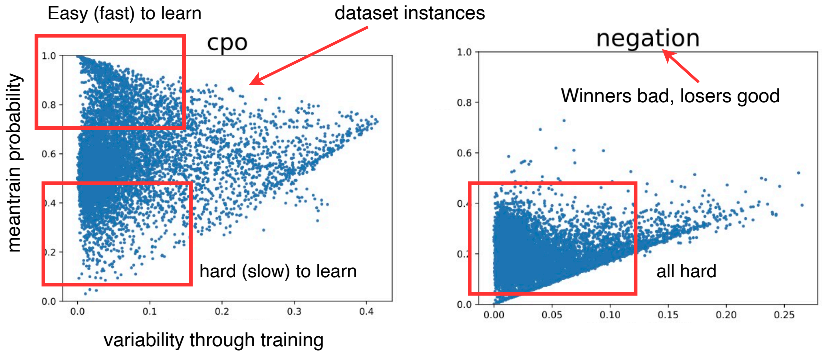
- **Intuition**: The speed/ease of training is a proxy for *goodness of semantic fit*, similar to small-loss criterion in noisy-label learning (Gui et al., 2021).

Characterizing dataset semantics?



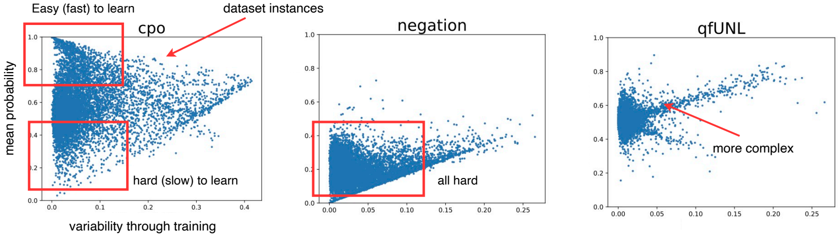
- **Intuition**: The speed/ease of training is a proxy for *goodness of semantic fit*, similar to small-loss criterion in noisy-label learning (Gui et al., 2021).

Characterizing dataset semantics?



- **Intuition**: The speed/ease of training is a proxy for *goodness of semantic fit*, similar to small-loss criterion in noisy-label learning (Gui et al., 2021).

Characterizing dataset semantics?



- **Intuition**: The speed/ease of training is a proxy for *goodness of semantic fit*, similar to small-loss criterion in noisy-label learning (Gui et al., 2021).

Blueprint for much empirical exploration of loss space

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).

  High-level programming: 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).

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

- **Decompiling models to symbolic programs**: *semantics of data*, *reinforcement learning*, *chain-of-thought*, *LLM agents* ...

Thank you.

Adding a reference model

```
P:= Implies(
  And(M(x,y_l),Ref(x,y_w)),
  And(M(x,y_w),Ref(x,y_l))
)
```

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

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.

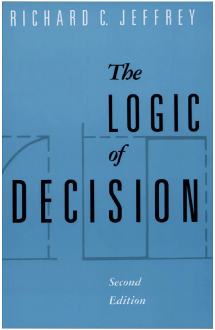
- **Peculiar semantics**, but the logic makes sense, e.g., we want to maximize

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

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

Classical work on preference

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	$P^{\sharp}$	$P^{\star}$	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. $pPq \rightarrow (p \,\&\, \sim q)\, P(\sim p \,\&\, q)$	✓	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. $[(p \,\&\, r)\, P(q \,\&\, r) \,\&\, (p \,\&\, \sim r)\, P(q \,\&\, \sim r)] \rightarrow pPq$	✓			$(+)^2$	$(+)^3$	+
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. $pP(q \vee r) \rightarrow (pPq \,\&\, pPr)$			✓	—	—	—
18. $(pPq \,\&\, pPr) \rightarrow pP(q \vee r)$				—	—	—
19. $(pPr \,\&\, qPr) \rightarrow (p \,\&\, q)\, Pr$				—	—	—

Semantic foundations for the logic of preference Rescher (1967)

References I

Beurer-Kellner, L., Fischer, M., and Vechev, M. (2023). Prompting is programming: A query language for large language models. *Proceedings of the ACM on Programming Languages*, 7(PLDI):1946–1969.

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*.

Bragg, J., D'Arcy, M., Balepur, N., Bareket, D., Dalvi, B., Feldman, S., Haddad, D., Hwang, J. D., Jansen, P., Kishore, V., Majumder, B. P., Naik, A., Rahamimov, S., Richardson, K., Singh, A., Surana, H., Tiktinsky, A., Vasu, R., Wiener, G., Anastasiades, C., Candra, S., Dunkelberger, J., Emery, D., Evans, R., Hamada, M., Huff, R., Kinney, R., Latzke, M., Lochner, J., Lozano-Aguilera, R., Nguyen, C., Rao, S., Tanaka, A., Vlahos, B., Clark, P., Downey, D., Goldberg, Y., Sabharwal, A., and Weld, D. S. (2025). Astabench: Rigorous benchmarking of ai agents with a scientific research suite.

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*.

Cheng, J., Clark, P., and Richardson, K. (2025). Language modeling by language models. *Proceedings of Neurips*.

References II

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.

Gui, X.-J., Wang, W., and Tian, Z.-H. (2021). Towards understanding deep learning from noisy labels with small-loss criterion. *arXiv preprint arXiv:2106.09291*.

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.

Li, Z., Huang, J., and Naik, M. (2023). Scallop: A language for neurosymbolic programming. *Proceedings of the ACM on Programming Languages*, 7(PLDI):1463–1487.

Manhaeve, R., Dumancic, S., Kimmig, A., Demeester, T., and De Raedt, L. (2018). Deepproblog: Neural probabilistic logic programming. *Advances in neural information processing systems*, 31.

Marra, G., Dumančić, S., Manhaeve, R., and De Raedt, L. (2024). From statistical relational to neurosymbolic artificial intelligence: A survey. *Artificial Intelligence*, page 104062.

References III

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.

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*.

Richardson, K. and Wijnholds, G. (2025). Lectures on language model programming.

Swayamdipta, S., Schwartz, R., Lourie, N., Wang, Y., Hajishirzi, H., Smith, N. A., and Choi, Y. (2020). Dataset cartography: Mapping and diagnosing datasets with training dynamics. *arXiv preprint arXiv:2009.10795*.

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, A., Chiang, D., and Angluin, D. (2024). Masked hard-attention transformers recognize exactly the star-free languages. *Advances in Neural Information Processing Systems*, 37:10202–10235.

References IV

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*.

Yu, H., Xuan, K., Li, F., Zhu, K., Lei, Z., Zhang, J., Qi, Z., Richardson, K., and You, J. (2025). Tinyscientist: An interactive, extensible, and controllable framework for building research agents. *Proceedings of EMNLP*.

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*.