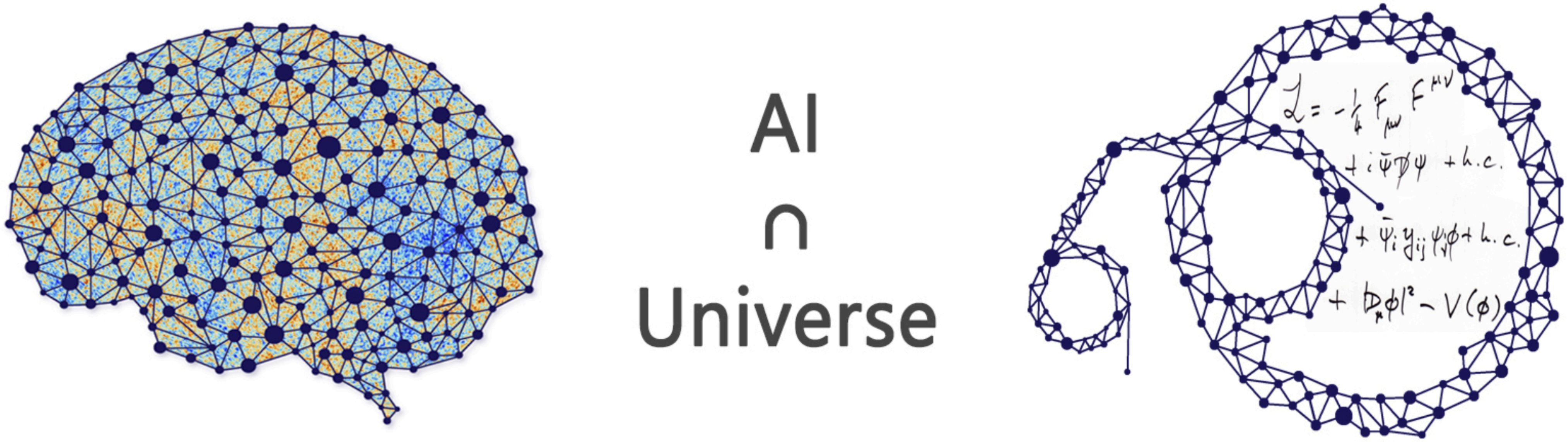
Physics 361 - Machine Learning in Physics

Lecture 18 – Reinforcement Learning

March 20th 2025

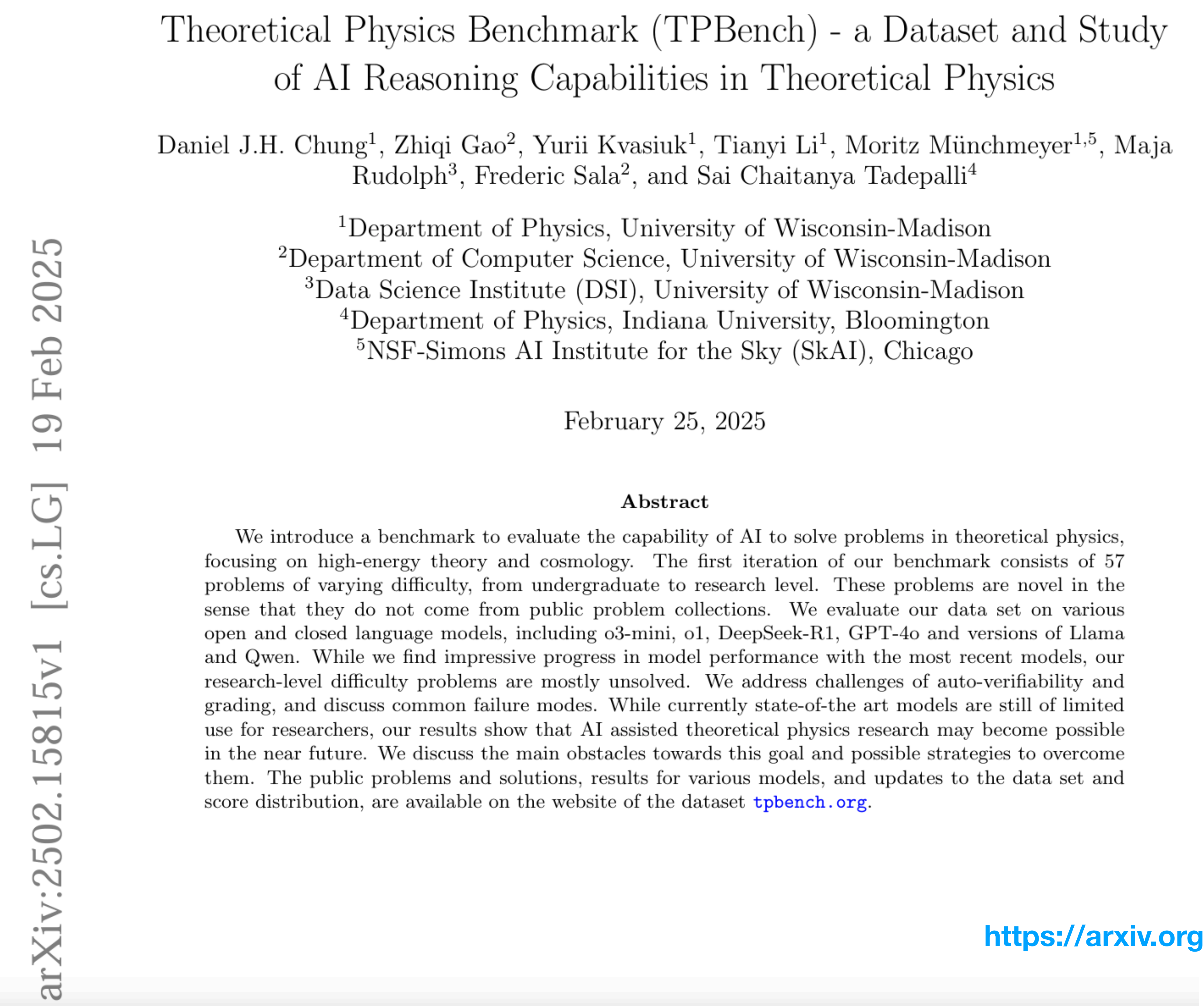

AI
∩
Universe
$\mathcal{L} = -\frac{1}{4}F_{\mu\nu}F^{\mu\nu}$
$+ i\bar{\psi}\not{D}\psi + h.c.$
$+ \bar{\psi}_i y_{ij}\psi_j\phi + h.c.$
$+ |D_\mu\phi|^2 - V(\phi)$

Moritz Münchmeyer

Motivation

LLMs for Reasoning at the example of TPBench (example from my group)

Recently on Arxiv



Theoretical Physics Benchmark (TPBench) - a Dataset and Study of AI Reasoning Capabilities in Theoretical Physics

Daniel J.H. Chung[1], Zhiqi Gao[2], Yurii Kvasiuk[1], Tianyi Li[1], Moritz Münchmeyer[1,5], Maja Rudolph[3], Frederic Sala[2], and Sai Chaitanya Tadepalli[4]

[1]Department of Physics, University of Wisconsin-Madison
[2]Department of Computer Science, University of Wisconsin-Madison
[3]Data Science Institute (DSI), University of Wisconsin-Madison
[4]Department of Physics, Indiana University, Bloomington
[5]NSF-Simons AI Institute for the Sky (SkAI), Chicago

February 25, 2025

Abstract

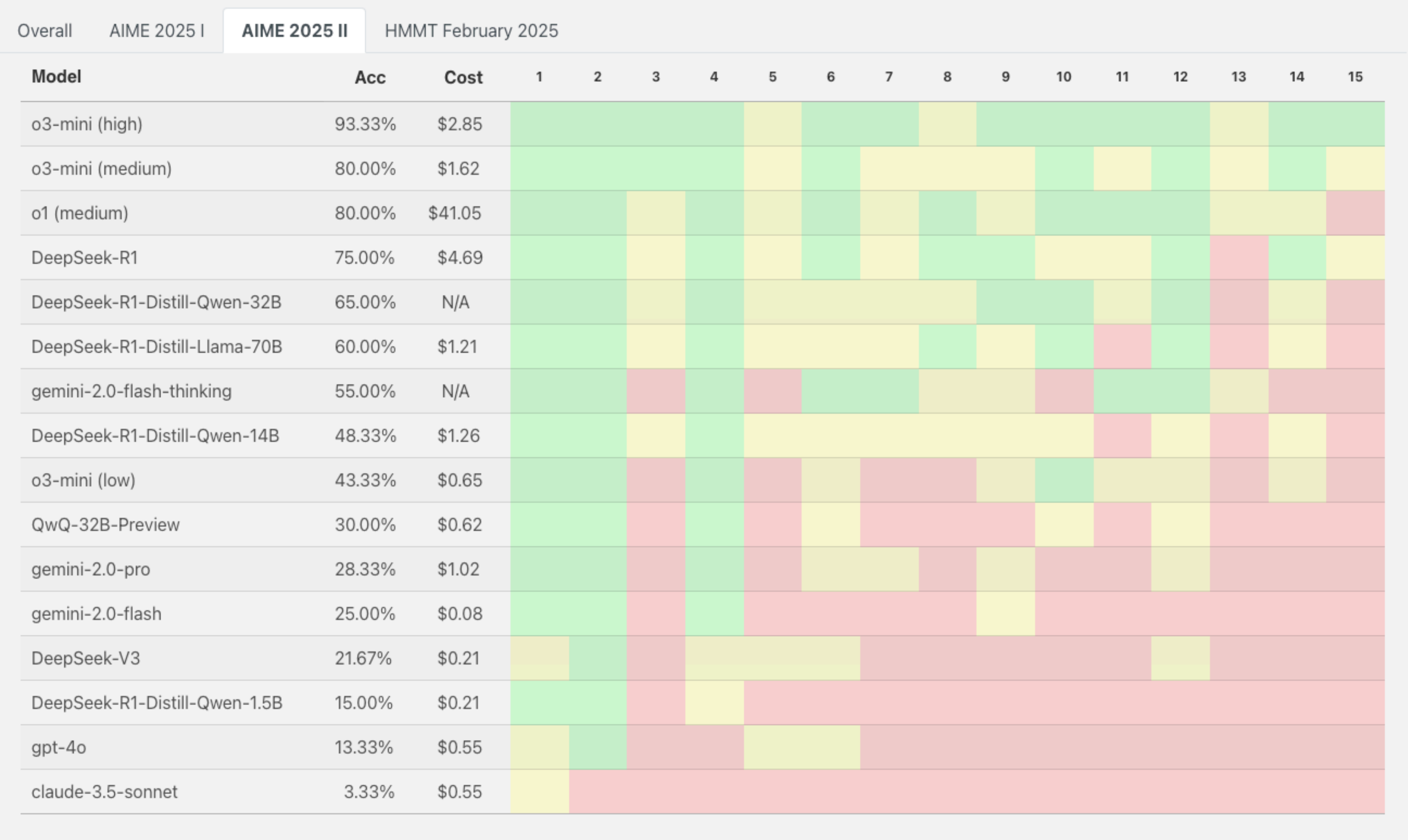
We introduce a benchmark to evaluate the capability of AI to solve problems in theoretical physics, focusing on high-energy theory and cosmology. The first iteration of our benchmark consists of 57 problems of varying difficulty, from undergraduate to research level. These problems are novel in the sense that they do not come from public problem collections. We evaluate our data set on various open and closed language models, including o3-mini, o1, DeepSeek-R1, GPT-4o and versions of Llama and Qwen. While we find impressive progress in model performance with the most recent models, our research-level difficulty problems are mostly unsolved. We address challenges of auto-verifiability and grading, and discuss common failure modes. While currently state-of-the art models are still of limited use for researchers, our results show that AI assisted theoretical physics research may become possible in the near future. We discuss the main obstacles towards this goal and possible strategies to overcome them. The public problems and solutions, results for various models, and updates to the data set and score distribution, are available on the website of the dataset tpbench.org.

https://arxiv.org/abs/2502.15815

LLM progress on Math

- High-school and undergraduate math competition problems are now basically solved.

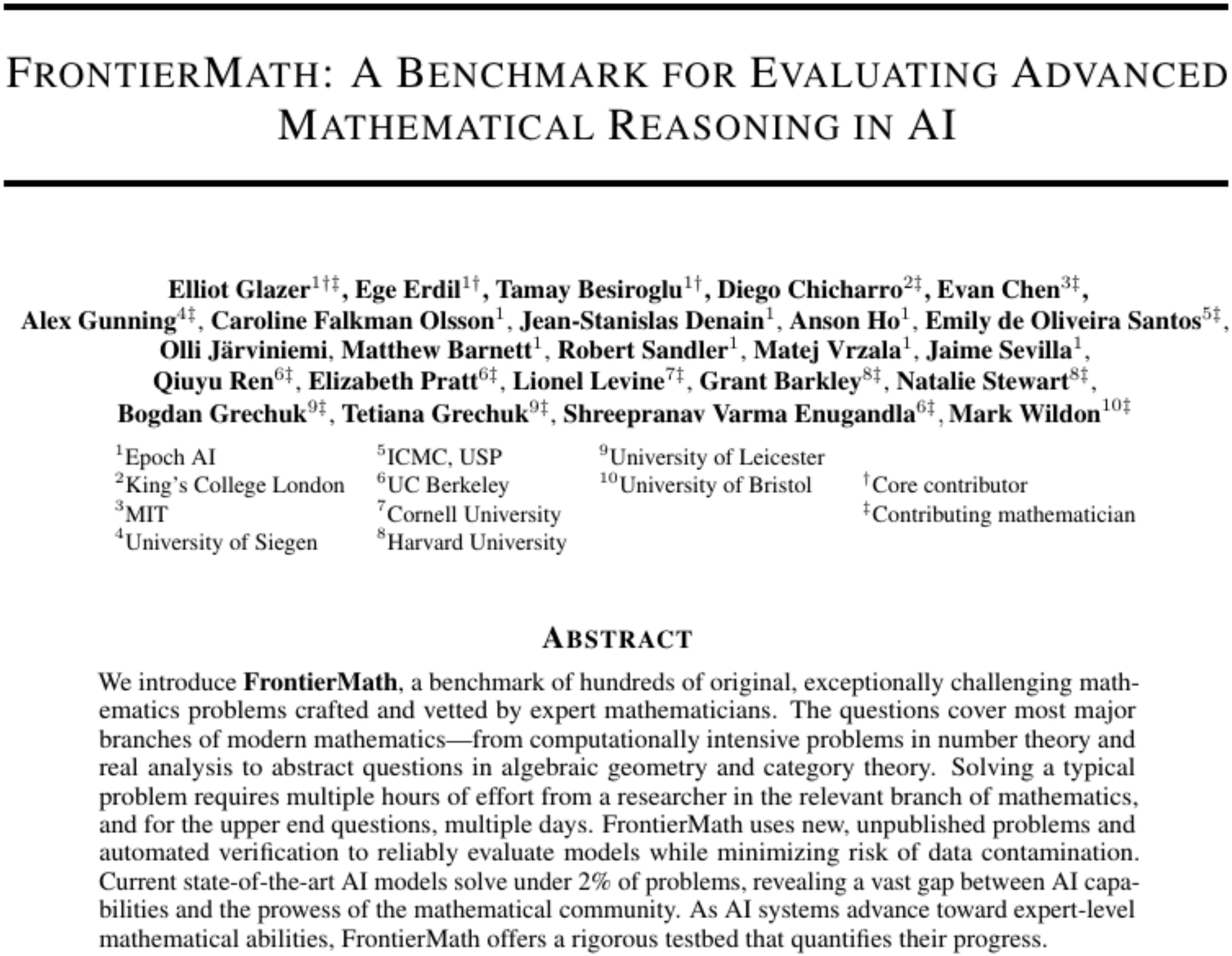
MathArena: Evaluating LLMs on Uncontaminated Math Competitions



Model	Acc	Cost	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
o3-mini (high)	93.33%	$2.85	🟩	🟩	🟩	🟩	🟨	🟩	🟩	🟨	🟩	🟩	🟩	🟩	🟨	🟩	🟩
o3-mini (medium)	80.00%	$1.62	🟩	🟩	🟩	🟩	🟨	🟩	🟨	🟨	🟨	🟩	🟨	🟩	🟨	🟩	🟨
o1 (medium)	80.00%	$41.05	🟩	🟩	🟨	🟩	🟨	🟩	🟨	🟩	🟨	🟩	🟩	🟩	🟨	🟨	🟥
DeepSeek-R1	75.00%	$4.69	🟩	🟩	🟨	🟩	🟨	🟩	🟨	🟩	🟩	🟨	🟨	🟩	🟥	🟩	🟨
DeepSeek-R1-Distill-Qwen-32B	65.00%	N/A	🟩	🟩	🟨	🟩	🟨	🟨	🟨	🟨	🟩	🟩	🟨	🟩	🟥	🟨	🟥
DeepSeek-R1-Distill-Llama-70B	60.00%	$1.21	🟩	🟩	🟨	🟩	🟨	🟨	🟨	🟩	🟨	🟩	🟥	🟩	🟥	🟨	🟥
gemini-2.0-flash-thinking	55.00%	N/A	🟩	🟩	🟥	🟩	🟥	🟩	🟩	🟨	🟨	🟥	🟩	🟩	🟨	🟥	🟥
DeepSeek-R1-Distill-Qwen-14B	48.33%	$1.26	🟩	🟩	🟨	🟩	🟨	🟨	🟨	🟨	🟨	🟨	🟥	🟨	🟥	🟨	🟥
o3-mini (low)	43.33%	$0.65	🟩	🟩	🟥	🟩	🟥	🟨	🟥	🟥	🟨	🟩	🟨	🟨	🟥	🟨	🟥
QwQ-32B-Preview	30.00%	$0.62	🟩	🟩	🟥	🟩	🟥	🟨	🟥	🟥	🟥	🟨	🟥	🟨	🟥	🟥	🟥
gemini-2.0-pro	28.33%	$1.02	🟩	🟩	🟥	🟩	🟥	🟨	🟨	🟥	🟨	🟥	🟥	🟨	🟥	🟥	🟥
gemini-2.0-flash	25.00%	$0.08	🟩	🟩	🟥	🟩	🟥	🟥	🟥	🟥	🟨	🟥	🟥	🟥	🟥	🟥	🟥
DeepSeek-V3	21.67%	$0.21	🟨	🟩	🟥	🟨	🟨	🟨	🟥	🟥	🟥	🟥	🟥	🟨	🟥	🟥	🟥
DeepSeek-R1-Distill-Qwen-1.5B	15.00%	$0.21	🟩	🟩	🟥	🟨	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
gpt-4o	13.33%	$0.55	🟨	🟩	🟥	🟥	🟨	🟨	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
claude-3.5-sonnet	3.33%	$0.55	🟨	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥

FrontierMath

- First research-level math data set.



FrontierMath: A Benchmark for Evaluating Advanced Mathematical Reasoning in AI

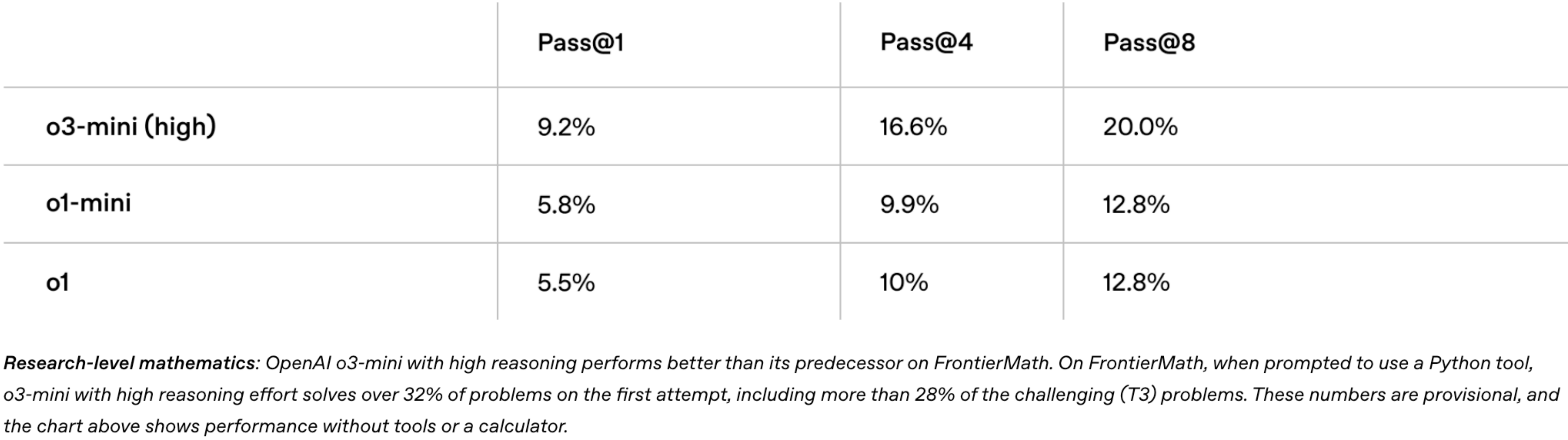
Elliot Glazer[1†‡], Ege Erdil[1†], Tamay Besiroglu[1†], Diego Chicharro[2‡], Evan Chen[3‡], Alex Gunning[4‡], Caroline Falkman Olsson[1], Jean-Stanislas Denain[1], Anson Ho[1], Emily de Oliveira Santos[5‡], Olli Järviniemi, Matthew Barnett[1], Robert Sandler[1], Matej Vrzala[1], Jaime Sevilla[1], Qiuyu Ren[6‡], Elizabeth Pratt[6‡], Lionel Levine[7‡], Grant Barkley[8‡], Natalie Stewart[8‡], Bogdan Grechuk[9‡], Tetiana Grechuk[9‡], Shreepranav Varma Enugandla[6‡], Mark Wildon[10‡]

[1]Epoch AI
[2]King's College London
[3]MIT
[4]University of Siegen
[5]ICMC, USP
[6]UC Berkeley
[7]Cornell University
[8]Harvard University
[9]University of Leicester
[10]University of Bristol
†Core contributor
‡Contributing mathematician

Abstract

We introduce **FrontierMath**, a benchmark of hundreds of original, exceptionally challenging mathematics problems crafted and vetted by expert mathematicians. The questions cover most major branches of modern mathematics—from computationally intensive problems in number theory and real analysis to abstract questions in algebraic geometry and category theory. Solving a typical problem requires multiple hours of effort from a researcher in the relevant branch of mathematics, and for the upper end questions, multiple days. FrontierMath uses new, unpublished problems and automated verification to reliably evaluate models while minimizing risk of data contamination. Current state-of-the-art AI models solve under 2% of problems, revealing a vast gap between AI capabilities and the prowess of the mathematical community. As AI systems advance toward expert-level mathematical abilities, FrontierMath offers a rigorous testbed that quantifies their progress.

FrontierMath Results



	Pass@1	Pass@4	Pass@8
o3-mini (high)	9.2%	16.6%	20.0%
o1-mini	5.8%	9.9%	12.8%
o1	5.5%	10%	12.8%

***Research-level mathematics**: OpenAI o3-mini with high reasoning performs better than its predecessor on FrontierMath. On FrontierMath, when prompted to use a Python tool, o3-mini with high reasoning effort solves over 32% of problems on the first attempt, including more than 28% of the challenging (T3) problems. These numbers are provisional, and the chart above shows performance without tools or a calculator.*

https://openai.com/index/openai-o3-mini/

TPBench

- There is **no comparable Benchmark for theoretical physics**. Our goals was to create one, and make it community owned (rather than industry driven). TPBench is considerably smaller than FrontierMath, but sufficiently large to gauge TP reasoning progress.

- We wanted to answer:

  - How good is the current state-of-the-art AI for problem-solving in TP? Are existing models useful for research-level reasoning?

  - What are the most common failure modes? For example, are models performing correct reasoning but fail mostly at algebra (at which LLMs are known to perform poorly)?

Differences between Math and Physics

- Mathematical reasoning tends to focus on establishing exact statements constructed within a rigid logical framework, while TP reasoning mostly deals with approximate statements constructed within a **“softer” logical framework** in which some of the less quantitatively relevant details are left unspecified but ``most likely" can be filled in such that the statements can be made arbitrarily precisely if desired.

- TP reasoning primarily **relies on techniques of direct computations**, while mathematical reasoning tends to use more often indirect techniques such as contradiction and induction. TP computations often utilize algorithmic methods in calculus, linear algebra, complex analysis, differential equations, differential geometry, and group representation theory.

- TP reasoning often **focuses on derivations** of formulas whose parametric dependences as well as the overall normalization are implicitly defined in a narrow domain of physical relevance.

- TP typically focuses on **approximations whose quantitative uncertainties are often left unspecified**.

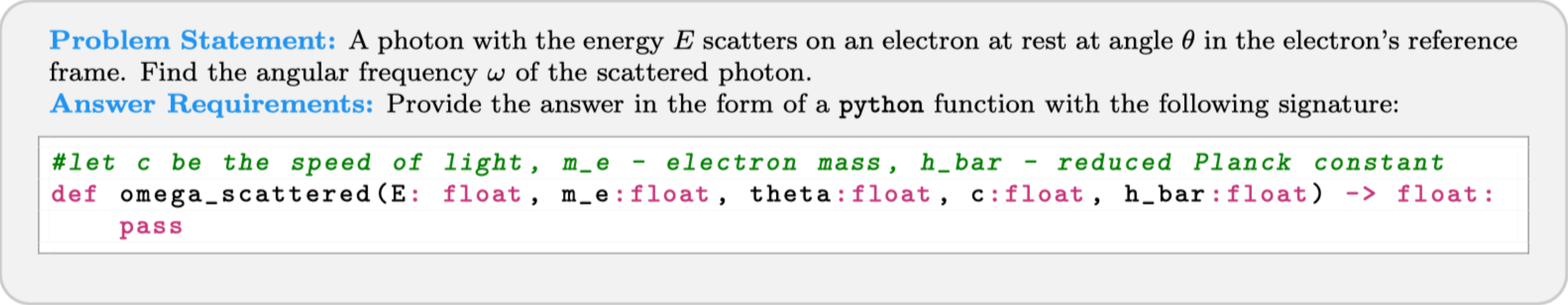
Properties of our data set

- **What properties should problems in the dataset have?**
  - The problem is **well-posed** and the solution to the problem is unambiguous. An expert in the field, after reading the solution, should not have any objections.
  - The problem is **original**. The solution to the problem cannot be easily found in the existing literature.
  - The answer should be **auto-verifiable**. This is easily achieved for numerical answers or simple algebraic expressions, but more difficult for tensor expressions. We discuss this property further below.
  - It should **not be possible to guess the answer** or remember it from the literature, despite a wrong reasoning chain.

Auto-verification

- We need to run a large number of problems many times, on many models. Human grading is not feasible.
- We did experiment with **LLM grading**, where the grader model gets the answer from the solver model and the expert answer, compares them, and assigns a grade. However, we found that this method is too noisy.
- Instead, we constructed problems for which we can **verify the answer with code**, ignoring the derivation.

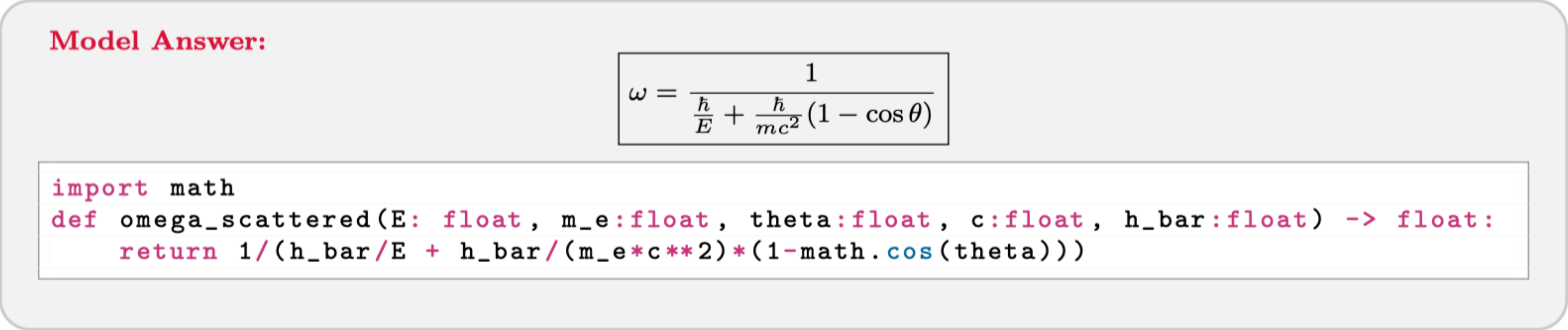
Auto-verification



Problem Statement: A photon with the energy E scatters on an electron at rest at angle θ in the electron's reference frame. Find the angular frequency ω of the scattered photon.

Answer Requirements: Provide the answer in the form of a **python** function with the following signature:

```python
#let c be the speed of light, m_e - electron mass, h_bar - reduced Planck constant
def omega_scattered(E: float, m_e:float, theta:float, c:float, h_bar:float) -> float:
    pass
```



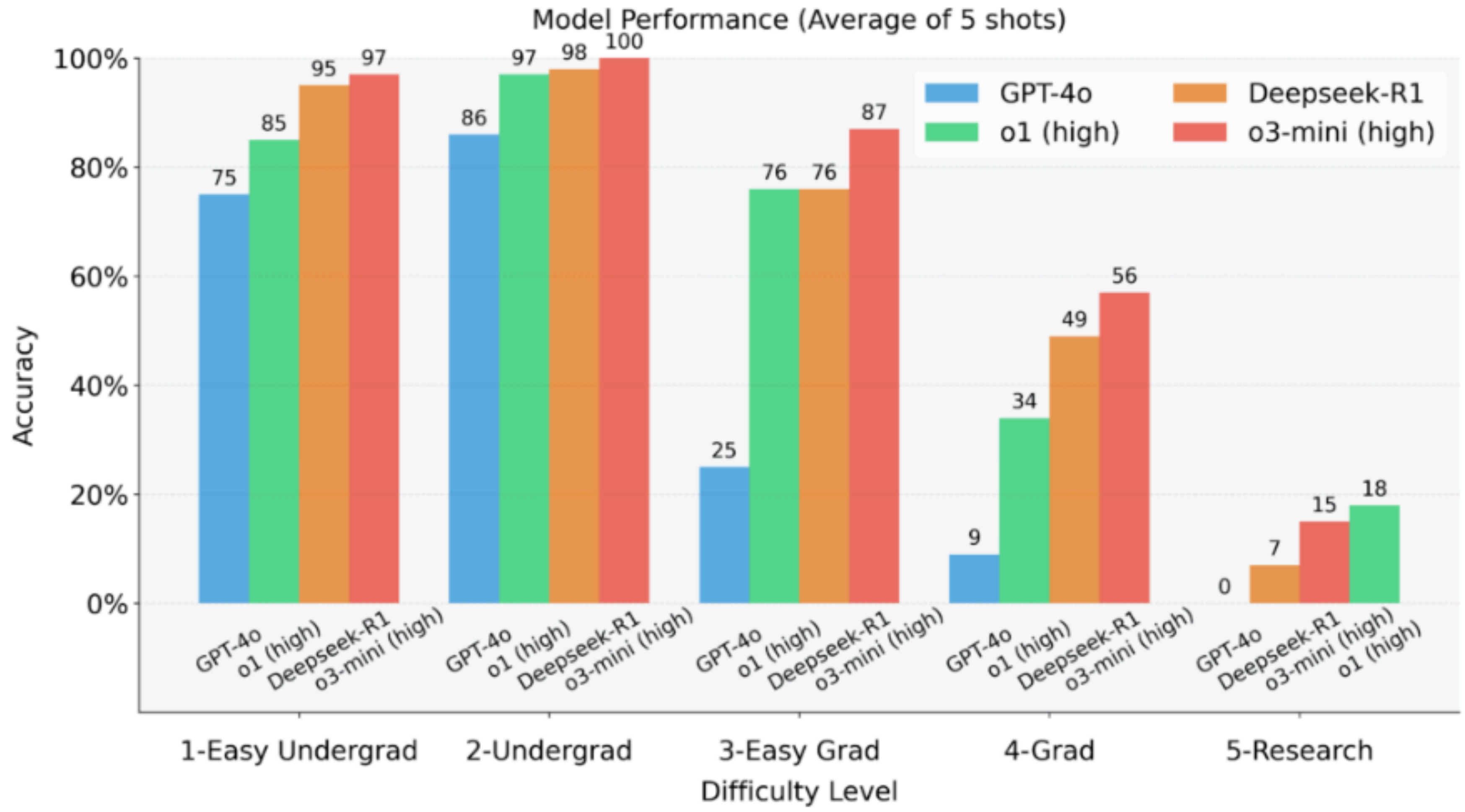
Model Answer:

$$\boxed{\omega = \frac{1}{\frac{\hbar}{E} + \frac{\hbar}{mc^2}(1 - \cos\theta)}}$$

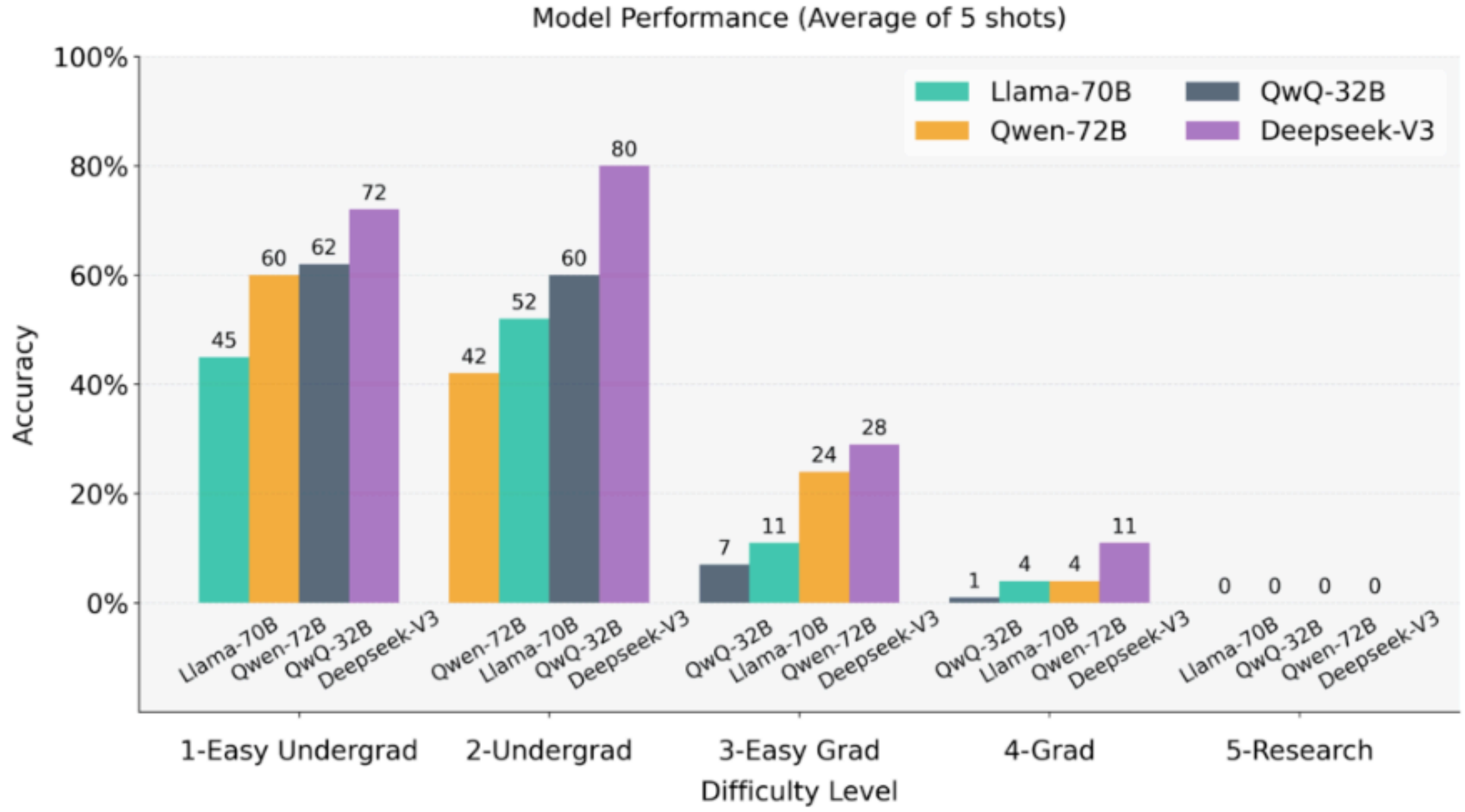
```python
import math
def omega_scattered(E: float, m_e:float, theta:float, c:float, h_bar:float) -> float:
    return 1/(h_bar/E + h_bar/(m_e*c**2)*(1-math.cos(theta)))
```

- Works well for algebraic expressions. Integral and derivative expressions pose problems.

Results



Model Performance (Average of 5 shots)
Accuracy
100%
80%
60%
40%
20%
0%
GPT-4o
o1 (high)
Deepseek-R1
o3-mini (high)
75
85
95
97
86
97
98
100
25
76
76
87
9
34
49
56
0
7
15
18
GPT-4o
o1 (high)
Deepseek-R1
o3-mini (high)
GPT-4o
o1 (high)
Deepseek-R1
o3-mini (high)
GPT-4o
o1 (high)
Deepseek-R1
o3-mini (high)
GPT-4o
o1 (high)
Deepseek-R1
o3-mini (high)
GPT-4o
Deepseek-R1
o3-mini (high)
o1 (high)
1-Easy Undergrad
2-Undergrad
3-Easy Grad
4-Grad
5-Research
Difficulty Level



Model Performance (Average of 5 shots)
Accuracy
100%
80%
60%
40%
20%
0%
Llama-70B
Qwen-72B
QwQ-32B
Deepseek-V3
45
60
62
72
42
52
60
80
7
11
24
28
1
4
4
11
0
0
0
0
Llama-70B
Qwen-72B
QwQ-32B
Deepseek-V3
Qwen-72B
Llama-70B
QwQ-32B
Deepseek-V3
QwQ-32B
Llama-70B
Qwen-72B
Deepseek-V3
QwQ-32B
Llama-70B
Qwen-72B
Deepseek-V3
Llama-70B
QwQ-32B
Qwen-72B
Deepseek-V3
1-Easy Undergrad
2-Undergrad
3-Easy Grad
4-Grad
5-Research
Difficulty Level

Results

- Progress has been **very rapid with the most recent models**. When we initiated this project, GPT-4o(released on May 2024) was state-of-the-art and unable to solve almost any TP problem beyond undergraduate level. When the o1-preview model (released on Sep 2024) appeared, it could solve many easy graduate level problems, but rarely any harder ones. The o3-mini series (released on Jan 2025), is able to solve about half of our advanced graduate level problems and even a few research problems.

- Nevertheless, **research problems involving long mathematical arguments are generally unsolved.**

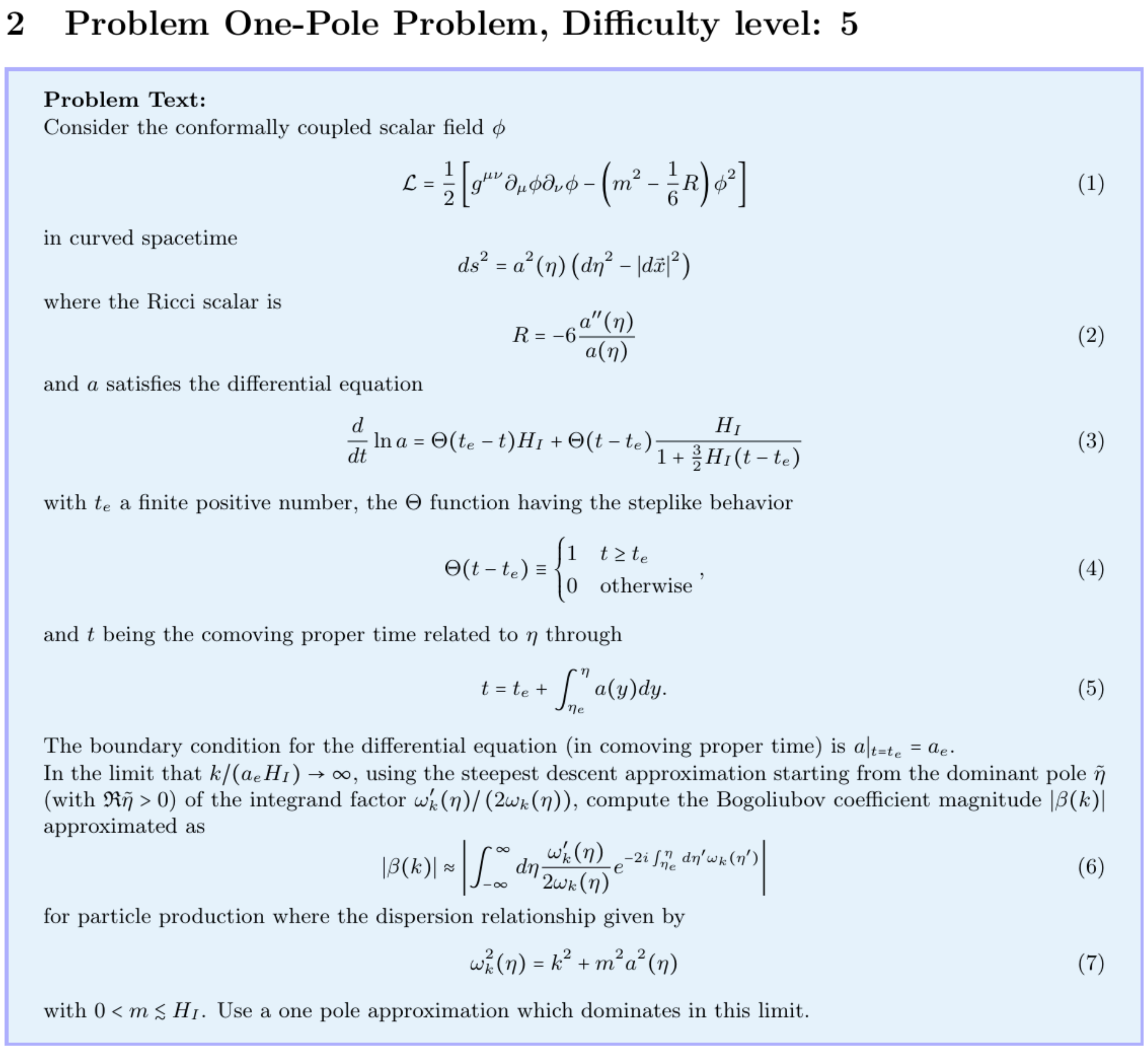
- Both symbolic calculation mistakes and logical reasoning mistakes are common, but have decreased with the newest models.

  - We provide a detailed **error analysis in the paper**.

Problem Samples

- Let’s look at some example problems, and model solutions.

https://tpbench.org/?page_id=2



2 Problem One-Pole Problem, Difficulty level: 5

Problem Text:
Consider the conformally coupled scalar field ϕ

$$\mathcal{L} = \frac{1}{2}\left[g^{\mu\nu}\partial_\mu\phi\partial_\nu\phi - \left(m^2 - \frac{1}{6}R\right)\phi^2\right] \tag{1}$$

in curved spacetime

$$ds^2 = a^2(\eta)\left(d\eta^2 - |d\vec{x}|^2\right)$$

where the Ricci scalar is

$$R = -6\frac{a''(\eta)}{a(\eta)} \tag{2}$$

and a satisfies the differential equation

$$\frac{d}{dt}\ln a = \Theta(t_e - t)H_I + \Theta(t - t_e)\frac{H_I}{1 + \frac{3}{2}H_I(t - t_e)} \tag{3}$$

with t_e a finite positive number, the Θ function having the steplike behavior

$$\Theta(t - t_e) \equiv \begin{cases} 1 & t \geq t_e \\ 0 & \text{otherwise} \end{cases}, \tag{4}$$

and t being the comoving proper time related to η through

$$t = t_e + \int_{\eta_e}^{\eta} a(y)dy. \tag{5}$$

The boundary condition for the differential equation (in comoving proper time) is $a|_{t=t_e} = a_e$.
In the limit that $k/(a_e H_I) \to \infty$, using the steepest descent approximation starting from the dominant pole $\tilde{\eta}$ (with $\Re\tilde{\eta} > 0$) of the integrand factor $\omega_k'(\eta)/(2\omega_k(\eta))$, compute the Bogoliubov coefficient magnitude $|\beta(k)|$ approximated as

$$|\beta(k)| \approx \left|\int_{-\infty}^{\infty} d\eta \frac{\omega_k'(\eta)}{2\omega_k(\eta)} e^{-2i\int_{\eta_e}^{\eta} d\eta' \omega_k(\eta')}\right| \tag{6}$$

for particle production where the dispersion relationship given by

$$\omega_k^2(\eta) = k^2 + m^2 a^2(\eta) \tag{7}$$

with $0 < m \lesssim H_I$. Use a one pole approximation which dominates in this limit.

How can models be so strong?

- We agreed to summarize as follows: **“In summary, current model performance perhaps resembles a student with superhuman literature knowledge but low intellectual rigor and technical expertise.”**

- However, superhuman literature knowledge is not cheating**. Humans use as much relevant literature as they can**, and adapt it to their problems. Models now are strong enough to recognize relevant literature, and adapt it (including non-trivial modifications).

- Of course our problems are not “entirely new”. They are not from public problem collections but they are constructed using research papers (at high difficulty level). You can look up their origin for the public problems.

- Some problems are more independent from publications than others. But to me it is absolutely **clear that models are not just spitting out text they remember without understanding**.

Interesting followup directions

- **Make larger benchmarks and larger training data.**
    - E.g. Problems extracted from very new arxiv papers that have not yet been used in pre-training.
- **Automatic verification** for non-algebraic expressions.
    - Difficult to solve in general.
- **Improving reasoning methods** for TP.
    - Better training data for Reinforcement Learning
    - Better tool usage such as Mathematica.
    - Test time scaling, better inference algorithms.
- Is it possible **to do truly novel research** with these models?

Outlook for next lectures

- In the next three lectures we want to **understand better how reasoning models can be built**.

- We **first core Reinforcement Learning in general** (not specifically for LLMs). Then we will discuss Reinforcement Learning for Reasoning models, including the Deepseek R1 result (e.g. PPO, GRPO).

- We will **also discuss other methods to improve reasoning** (supervised fine tuning, tool usage, test time scaling).

- While Reasoning is not (yet) in the mainstream of AI for physics, I think these topics are exciting and important for the future.

Reinforcement Learning

Introduction

Introduction

- Reinforcement leaning is a sequential decision making framework in which agents learned to perform actions in an environment with the goal of maximizing rewards.

- RL controls the **actions** of an **agent** in an **environment** to maximize the **reward**.

- RL applications: Go/Chess/Atari, robotics, financial trading, string theory, optimal experimental design, robotics, reasoning, ….

- RL is often used when problem involves searching a large configuration space.

- References:

  - Sutton and Barto: http://incompleteideas.net/book/the-book-2nd.html,

  - **Simon Prince, Understanding Deep Learning: https://udlbook.github.io/udlbook/ (primary reference used here)**

- https://github.com/Farama-Foundation/Gymnasium (formerly https://github.com/openai/gym)

Challenges of RL

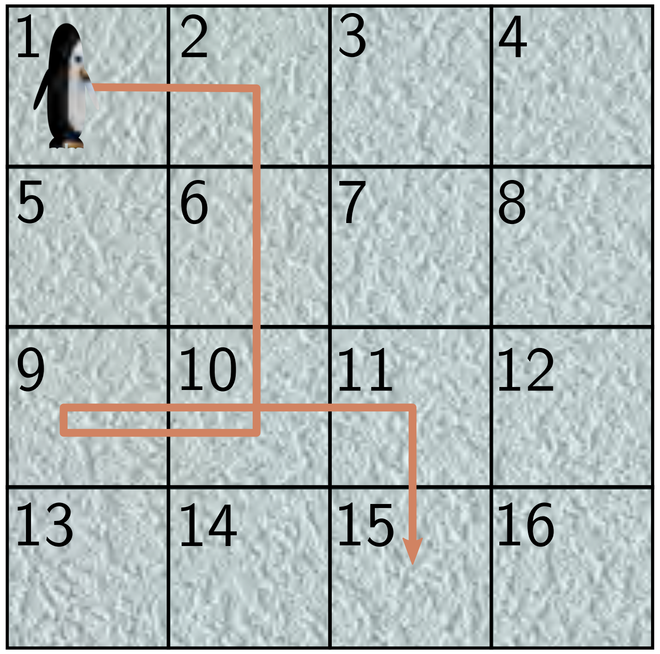
- Illustrate the challenges with **chess game**. A reward of +1, −1, or 0 is given at the end of the game if the agent wins, loses, or draws and 0 at every other time step. The challenges:
  - The reward is sparse; we must play an entire game to receive feedback.
  - **Temporal credit assignment problem**: The reward is temporally offset from the action that caused it; a decisive advantage might be gained thirty moves before victory. We must associate the reward with this critical action. (other examples?)
  - **The environment is stochastic**; the opponent doesn't always make the same move in the same situation, so it's hard to know if an action was truly good or just lucky.
  - **Exploration-exploitation trade-off**: The agent must balance exploring the environment (e.g., trying new opening moves) with exploiting what it already knows .

Reinforcement Learning

General Definitions

Markov Processes

- In RL, we learn a **policy** that maximizes the expected return in a Markov decision process.
- The word Markov implies that the probability of being in a state depends only on the previous state and not on the states before.
- The changes between states are captured by the transition probabilities $Pr(s_{t+1} \mid s_t)$ of moving to the next state s_{t+1} given the current state s_t, where t indexes the time step.
- A Markov process is an evolving system that produces a sequence $s_1, s_2, s_3, \ldots$ of states.
- τ = [s1, s2, s3, . . .] is called the **trajectory**



$s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8$

$\tau = [1, 2, 6, 10, 9, 10, 11, 15]$

Markov Reward Processes

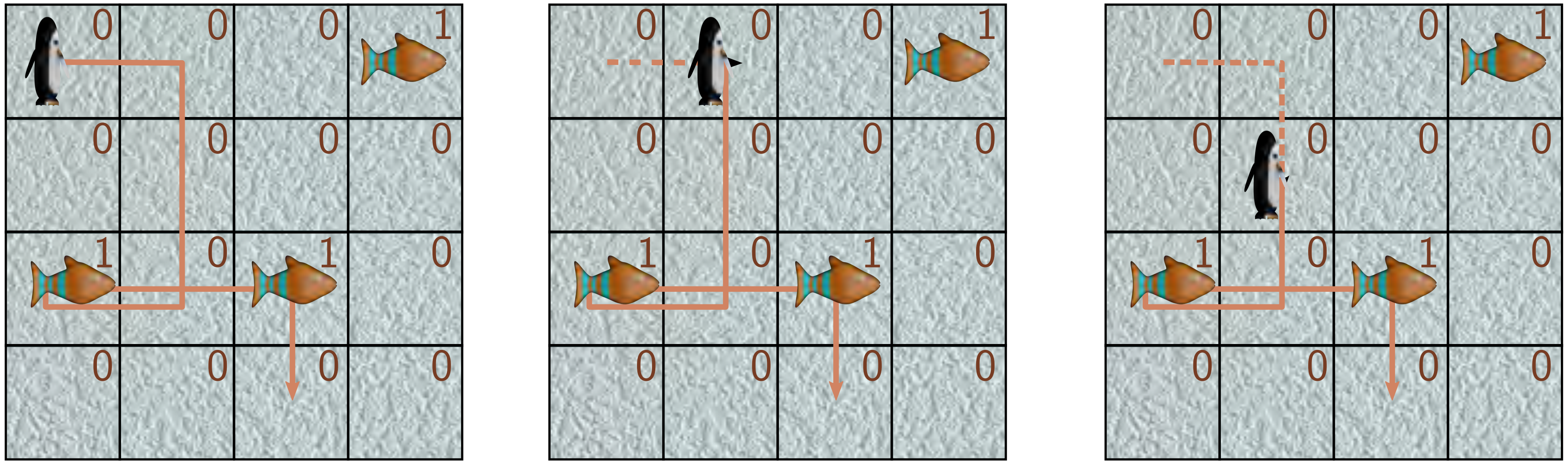
- A Markov reward process also includes a distribution $Pr(r_{t+1} \mid s_t)$ over the possible rewards r_{t+1} received at the next step, given s_t.
- Introduce a discount factor $\gamma \in (0,1]$ to compute the (cumulative) **return** G_t:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}.$$

a) $G_1 = 0 + \gamma \cdot 0 + \gamma^2 \cdot 0 + \gamma^3 \cdot 0 + \gamma^4 \cdot 1 + \gamma^5 \cdot 0 + \gamma^6 \cdot 1 + \gamma^7 \cdot 0 = 1.19$

b) $G_2 = 0 + \gamma \cdot 0 + \gamma^2 \cdot 0 + \gamma^3 \cdot 1 + \gamma^4 \cdot 0 + \gamma^5 \cdot 1 + \gamma^6 \cdot 0 = 1.31$

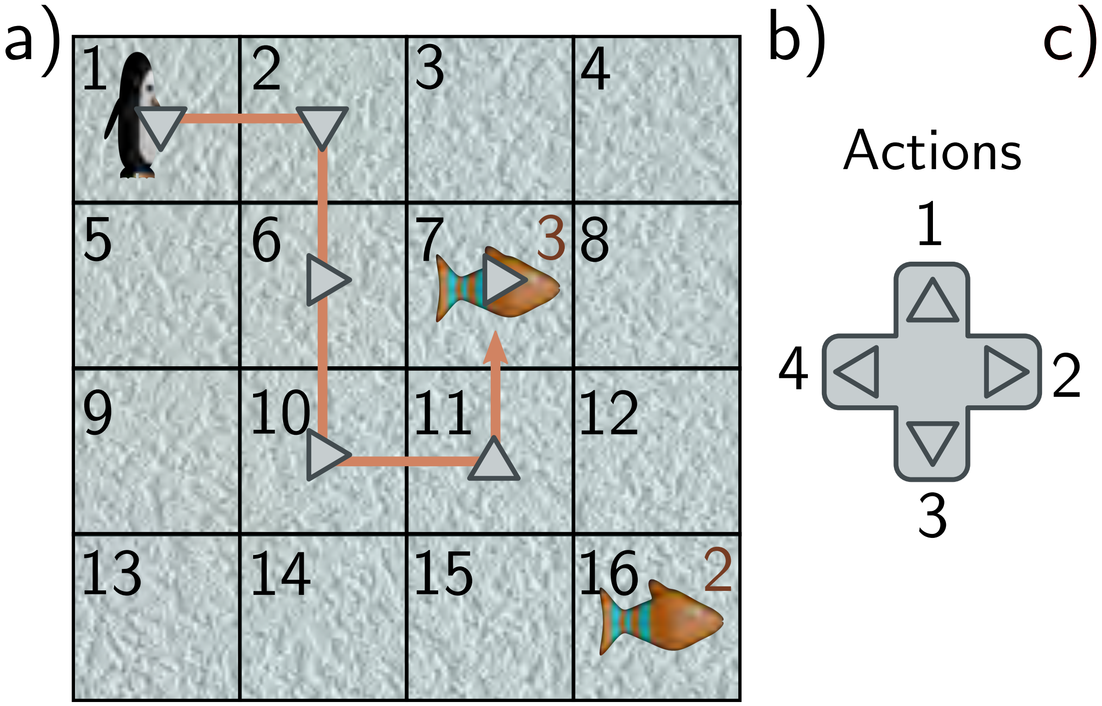
c) $G_3 = 0 + \gamma \cdot 0 + \gamma^2 \cdot 1 + \gamma^3 \cdot 0 + \gamma^4 \cdot 1 + \gamma^5 \cdot 0 = 1.47$



$$\boldsymbol{\tau} = [\overset{s_1}{1}, \overset{r_2}{0}, \overset{s_2}{2}, \overset{r_3}{0}, \overset{s_3}{6}, \overset{r_4}{0}, \overset{s_4}{10}, \overset{r_5}{0}, \overset{s_5}{9}, \overset{r_6}{1}, \overset{s_6}{10}, \overset{r_7}{0}, \overset{s_7}{11}, \overset{r_8}{1}, \overset{s_8}{15}, \overset{r_9}{0}]$$

Markov Decision Processes

- A Markov decision process (MDP) adds a set of possible action a_t at each step which changes the transition probabilities $Pr(s_{t+1} \,|\, s_t, a_t)$.
- The rewards can also depend on the action: $Pr(r_{t+1} \,|\, s_t, a_t)$.
- MDP produces a sequence $s_1, a_1, r_2, s_2, a_2, r_3, s_3, a_3, \ldots$ of states, actions & rewards. The entity that performs the actions is the **agent**.



$$\begin{array}{cccccccccccccccccc} s_1 & a_1 & r_2 & s_2 & a_2 & r_3 & s_3 & a_3 & r_4 & s_4 & a_4 & r_5 & s_5 & a_5 & r_6 & s_6 & a_6 & r_7 \end{array}$$
$$\boldsymbol{\tau} = [1, 3, 0, 2, 3, 0, 6, 2, 0, 10, 2, 0, 11, 1, 0, 7, 2, 3]$$

c)

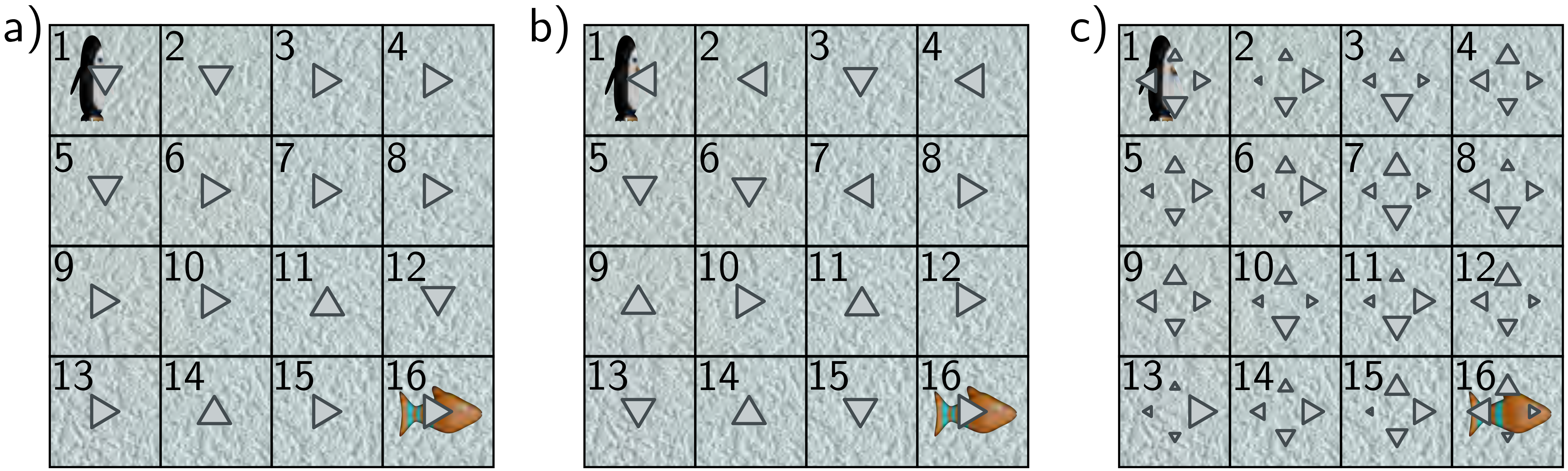
$$\begin{bmatrix} 0 \\ 0.5 \\ 0 \\ 0 \\ 0.17 \\ 0 \\ 0.17 \\ 0 \\ 0 \\ 0.17 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 0.17 \\ 0 \\ 0 \\ 0.17 \\ 0 \\ 0.5 \\ 0 \\ 0 \\ 0.17 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 0.17 \\ 0 \\ 0 \\ 0.17 \\ 0 \\ 0.17 \\ 0 \\ 0 \\ 0.5 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 \\ 0.17 \\ 0 \\ 0 \\ 0.5 \\ 0 \\ 0.17 \\ 0 \\ 0 \\ 0.17 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$Pr(s_{t+1}|s_t\!=\!6, a_t\!=\!1)$ $\quad$ $Pr(s_{t+1}|s_t\!=\!6, a_t\!=\!2)$ $\quad$ $Pr(s_{t+1}|s_t\!=\!6, a_t\!=\!3)$ $\quad$ $Pr(s_{t+1}|s_t\!=\!6, a_t\!=\!4)$

Here: The penguin moves in the intended direction with 50% probability, but the ice is slippery, so it may slide to one of the other adjacent positions with equal probability.

Policy

- The rules that determine the agent’s action are known as the **policy**.
- The policy can be **deterministic** (one action for a given state) or **stochastic** (a probability distribution over each possible action):

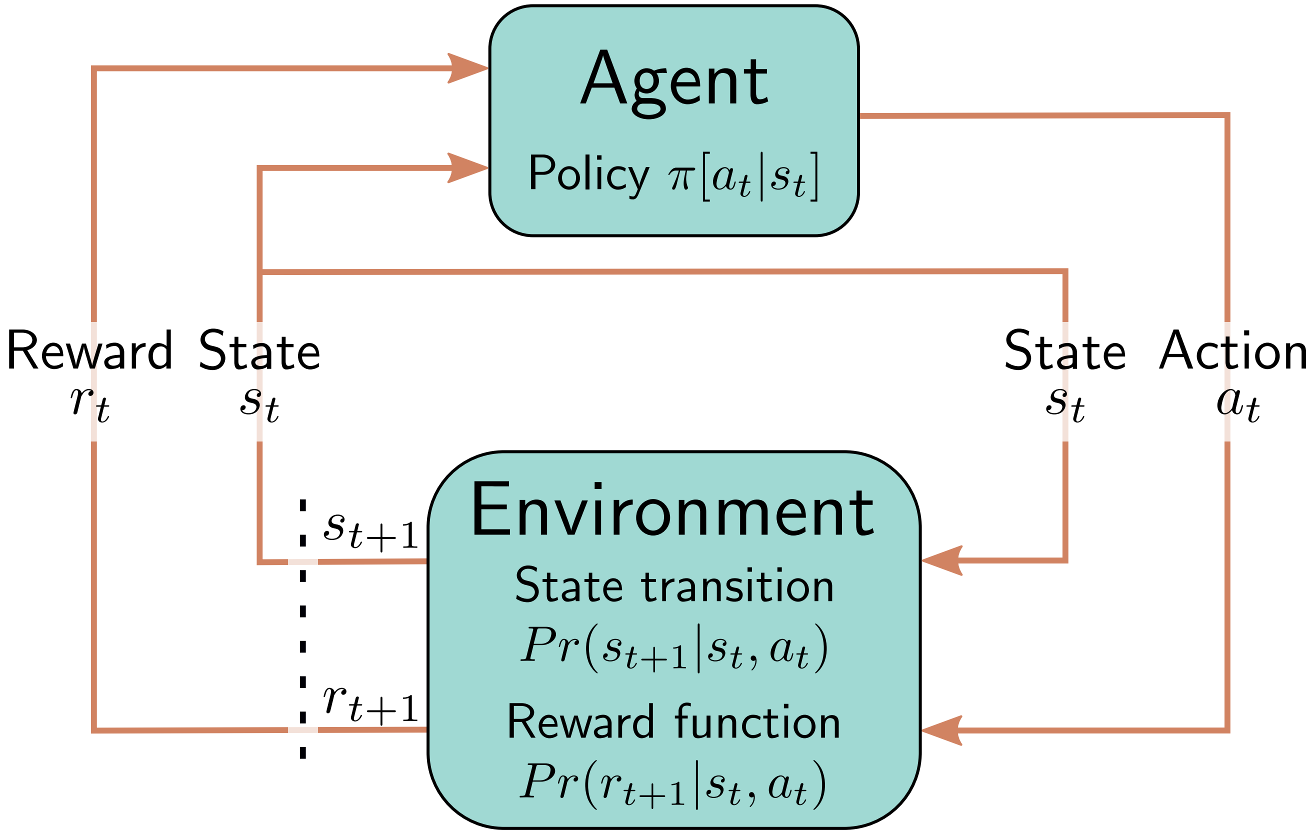


Deterministic

Stochastic

Reinforcement Learning Loop

- The environment and the agent form a loop:



- The agent receives the state and reward from the last time step. Based on the policy, the agent chooses the next action.
- The environment then assigns the next state according to $Pr(s_{t+1} \,|\, s_t, a_t)$ and the reward according to $Pr(r_{t+1} \,|\, s_t, a_t)$.

Expected return: state and action values

- The return G_t depends on the state s_t and the policy $\pi[a \mid s]$
- Characterize how “good” a state is under a given policy π by considering the expected return $v[s_t \mid \pi]$. **State-value function** (long-term return on average from sequences that starts from s_t):

$$v[s_t|\pi] = \mathbb{E}\Big[G_t|s_t, \pi\Big].$$

- **Action value** or state-action value function $q[s_t, a_t \mid \pi]$ is the expected return from executing action a_t in state s_t:

$$q[s_t, a_t|\pi] = \mathbb{E}\Big[G_t|s_t, a_t, \pi\Big].$$

- Through this quantity, RL algorithms connect future rewards to current actions (i.e., resolve the temporal credit assignment problem).

Optimal Policy

- We want a policy that maximizes the expected return.
- For MDPs, there ∃ a deterministic, stationary (depends only on the current state, not the time step) policy that maximizes the value of every state.
- If we know this optimal policy, then we get the optimal state-value function:

$$v^*[s_t] = \max_{\pi}\Big[\mathbb{E}\Big[G_t|s_t,\pi\Big]\Big].$$

- Similarly, the optimal state-action value function:

$$q^*[s_t,a_t] = \max_{\pi}\left[\mathbb{E}\Big[G_t|s_t,a_t,\pi\Big]\right].$$

- Turning this around, if we knew the optimal action-values, we can derive the optimal policy.

$$\pi[a_t|s_t] \leftarrow \underset{a_t}{\operatorname{argmax}}\Big[q^*[s_t,a_t]\Big].$$

Consistency of state and action values

- We may not know the state values v or action values q for any policy. However, we know that they must be consistent with one another, and it’s easy to write relations between these quantities.

- The state values are given by

$$\underset{\text{State value}}{v[s_t]} = \sum_{a_t} \underset{\text{Prob of action}}{\pi[a_t|s_t]} \cdot \underset{\text{Action value}}{q[s_t, a_t]}$$

- The action values are given by

$$\underset{\text{Action value}}{q[s_t, a_t]} = \underset{\text{Reward for action}}{r[s_t, a_t]} + \underset{\text{Discount factor}}{\gamma} \cdot \sum_{s_{t+1}} \underset{\text{Prob of next state}}{Pr(s_{t+1}|s_t, a_t)} \cdot \underset{\text{Value of next state}}{v[s_{t+1}]}$$

- These relations lead to the “Bellman equations”, a central concept in RL.

Reinforcement Learning

Tabular RL Methods

Tabular RL

- **"Tabular RL"** refers to **reinforcement learning methods that use explicit tables to store value functions (or policies)**. This means that every state (or state-action pair) is explicitly represented in a **lookup table** rather than being approximated by a function (e.g., a neural network).

- This a**pproach is feasible when the state space is small because all possible states and actions can be stored explicitly**.

- We start with this setup.

- We will later contrast tabular RL algorithms with the use of deep learning in RL that does not require storing the large transition matrix.

Course logistics

- **Reading for this lecture:**
  - This lecture was based in part on the book by Prince, linked on the website. Many figures were taken from this book.