

EvalXRL: Evaluating RL Explainability Methods by How Much They Help Fix Bugs in Agents

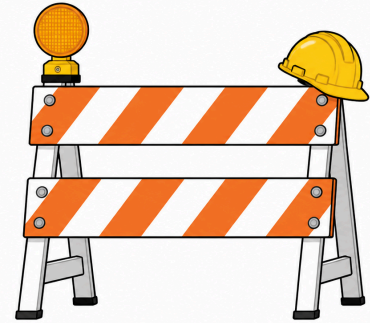
Ram Rachum, Yotam Amitai,
Bálint Gyevnár, Reuth Mirsky, Cameron Allen



Center for
Human-Compatible
Artificial
Intelligence

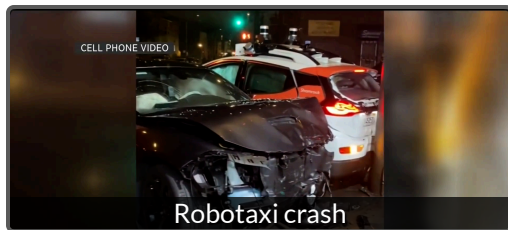


Work in progress

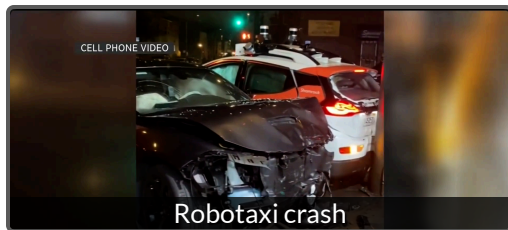


We'll talk about:

- **Motivation: Benchmarking XRL methods**
- How will it work?
- What success would look like
- 🎉 Demo of our infrastructure!



Why did the agent do this?



Why did the agent do this?

Reward decomposition

Juozapaitis et al., 2019[1]

GANterfactual

Huber et al., 2023[2]

AXIS

Gyevnár et al., 2026[3]

SARFA

Puri et al., 2020[4]

More XRL methods...

Milani et al., 2024[5]

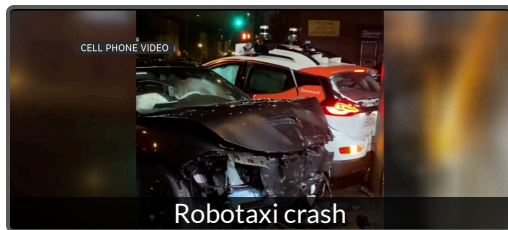
[1] Juozapaitis et al. (2019). *Explainable RL via Reward Decomposition*. IJCAI Workshop on Explainable AI (XAI).

[3] Gyevnár et al. (2026). *Integrating Counterfactual Simulations with Language Models for Explaining Multi-Agent Behaviour*. AAMAS.

[5] Milani et al. (2024). *Explainable Reinforcement Learning: A Survey and Comparative Review*. ACM Computing Surveys.

[2] Huber et al. (2023). *GANterfactual-RL: Understanding RL Agents' Strategies through Visual Counterfactual Explanations*. AAMAS.

[4] Puri et al. (2020). *Explain Your Move: Understanding Agent Actions Using Specific and Relevant Feature Attribution*. ICLR.



Why did the agent do this?

Reward
decomposition

Juozapaitis et al., 2019[1]

GANterfactual

Huber et al., 2023[2]

AXIS

Gyevnár et al., 2026[3]

SARFA

Puri et al., 2020[4]

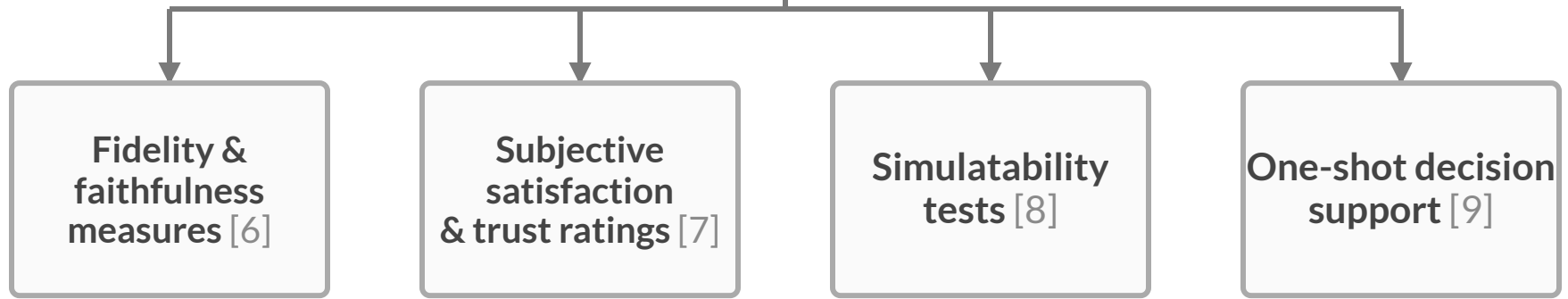
More XRL
methods...

Milani et al., 2024[5]

How can we measure how
good an explanation is?

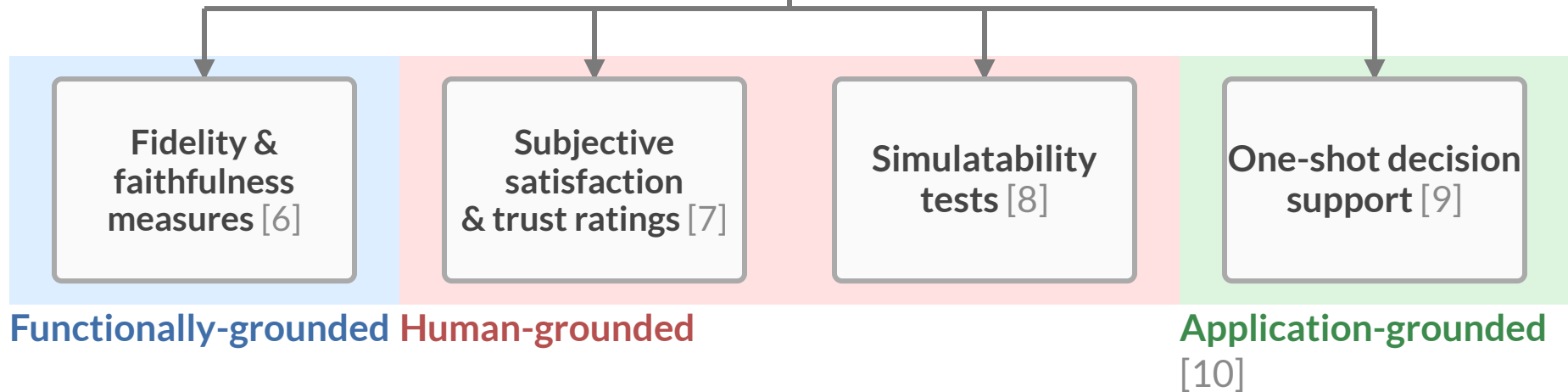
How can we measure how
good an explanation is?

How can we measure how good an explanation is?



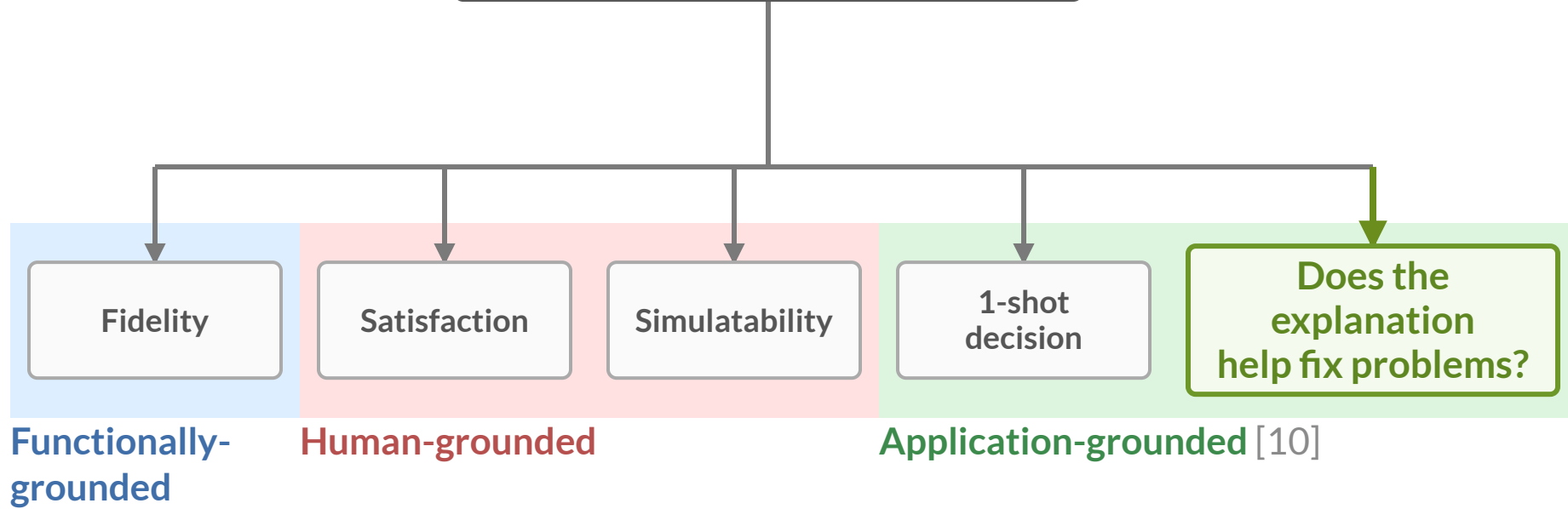
[6] Xiong et al. (2024). XRL-Bench: A Benchmark for Evaluating and Comparing Explainable Reinforcement Learning Techniques. KDD. [7] Hoffman et al. (2018). Metrics for Explainable AI: Challenges and Prospects. arXiv. [8] Hase and Bansal (2020). Evaluating Explainable AI: Which Algorithmic Explanations Help Users Predict Model Behavior?. ACL. [9] Amarasinghe et al. (2024). On the Importance of Application-Grounded Experimental Design for Evaluating Explainable ML Methods. AAAI. [10] Doshi-Velez and Kim (2017). Towards A Rigorous Science of Interpretable Machine Learning. arXiv.

How can we measure how good an explanation is?



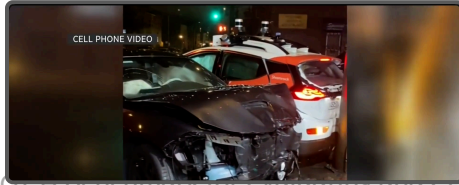
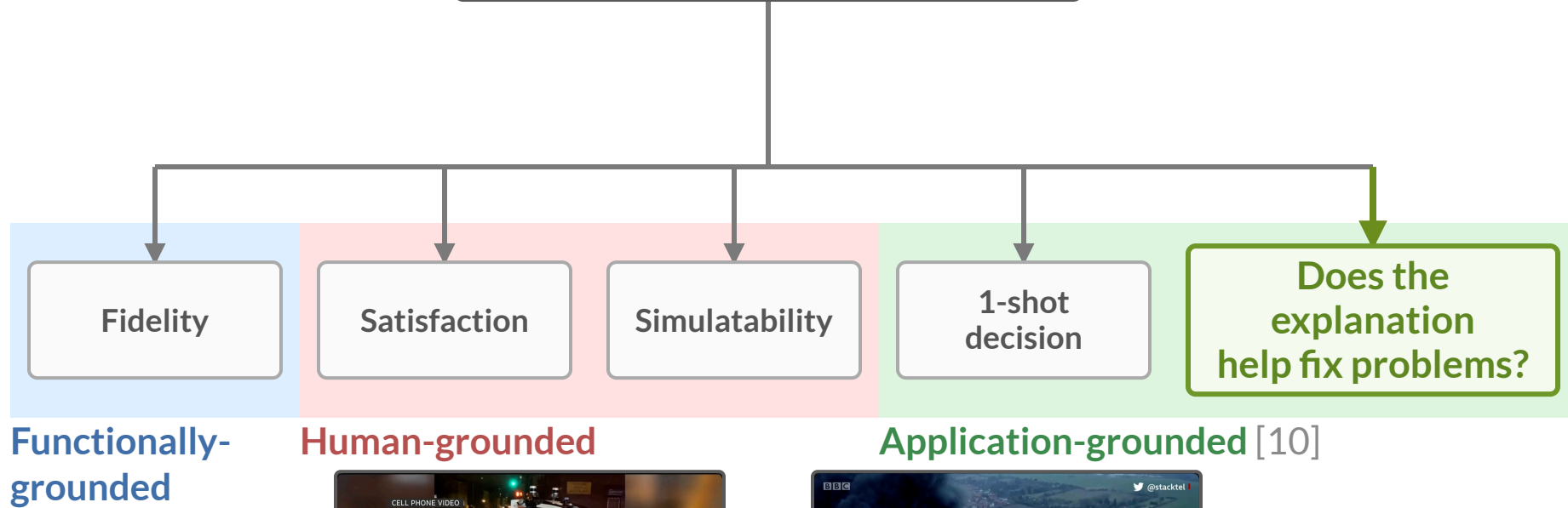
[6] Xiong et al. (2024). XRL-Bench: A Benchmark for Evaluating and Comparing Explainable Reinforcement Learning Techniques. KDD. [7] Hoffman et al. (2018). Metrics for Explainable AI: Challenges and Prospects. arXiv. [8] Hase and Bansal (2020). Evaluating Explainable AI: Which Algorithmic Explanations Help Users Predict Model Behavior?. ACL. [9] Amarasinghe et al. (2024). On the Importance of Application-Grounded Experimental Design for Evaluating Explainable ML Methods. AAAI. [10] Doshi-Velez and Kim (2017). Towards A Rigorous Science of Interpretable Machine Learning. arXiv.

How can we measure how good an explanation is?



[6] Xiong et al. (2024). XRL-Bench: A Benchmark for Evaluating and Comparing Explainable Reinforcement Learning Techniques. KDD. [7] Hoffman et al. (2018). Metrics for Explainable AI: Challenges and Prospects. arXiv. [8] Hase and Bansal (2020). Evaluating Explainable AI: Which Algorithmic Explanations Help Users Predict Model Behavior?. ACL. [9] Amarasinghe et al. (2024). On the Importance of Application-Grounded Experimental Design for Evaluating Explainable ML Methods. AAAI. [10] Doshi-Velez and Kim (2017). Towards A Rigorous Science of Interpretable Machine Learning. arXiv.

How can we measure how good an explanation is?



[6] Xiong et al. (2024). XRL-Bench: A Benchmark for Evaluating and Comparing Explainable Reinforcement Learning Techniques. KDD. [7] Hoffman et al. (2018). Metrics for Explainable AI: Challenges and Prospects. arXiv. [8] Hase and Bansal (2020). Evaluating Explainable AI: Which Algorithmic Explanations Help Users Predict Model Behavior?. ACL. [9] Amarasinghe et al. (2024). On the Importance of Application-Grounded Experimental Design for Evaluating Explainable ML Methods. AAAI. [10] Doshi-Velez and Kim (2017). Towards A Rigorous Science of Interpretable Machine Learning. arXiv.

Measuring success by fixing problems

Pros

- Closer to real-world problems
- Iterative! Evaluate explanation method, not explanations
- No questionnaire bias

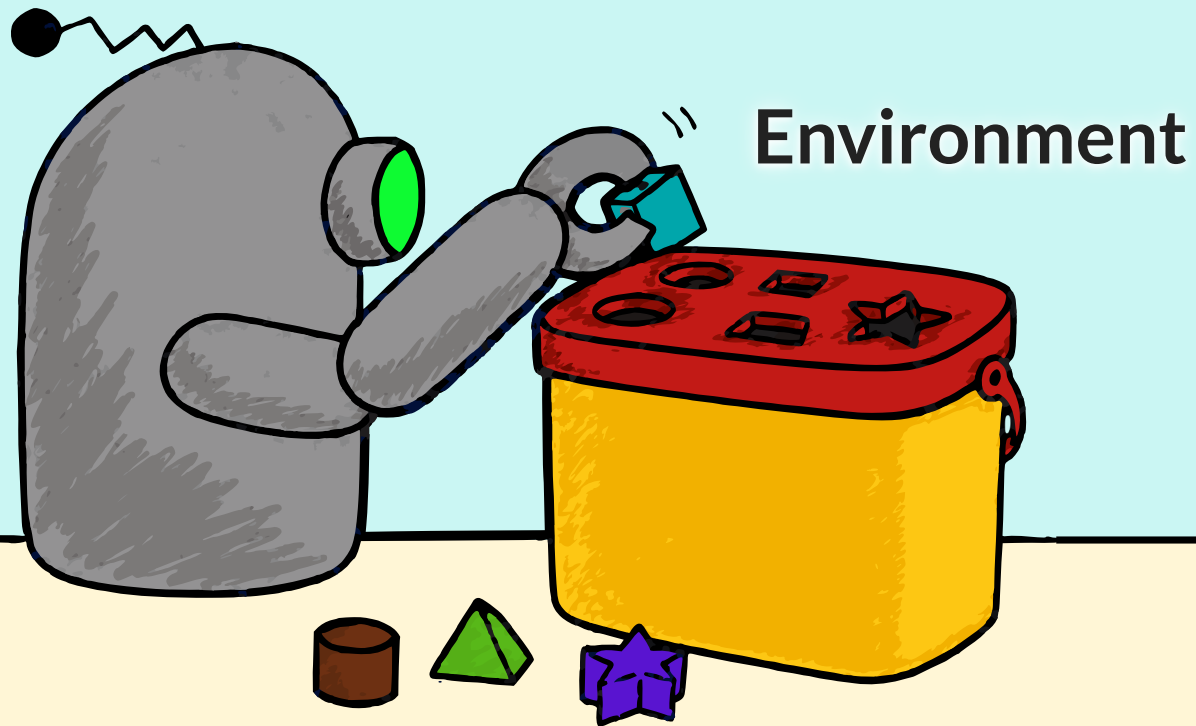
Cons

- Explanation use-cases $\supset$ fixing
- Dossiers add bias
- Problems limited to easily-contained

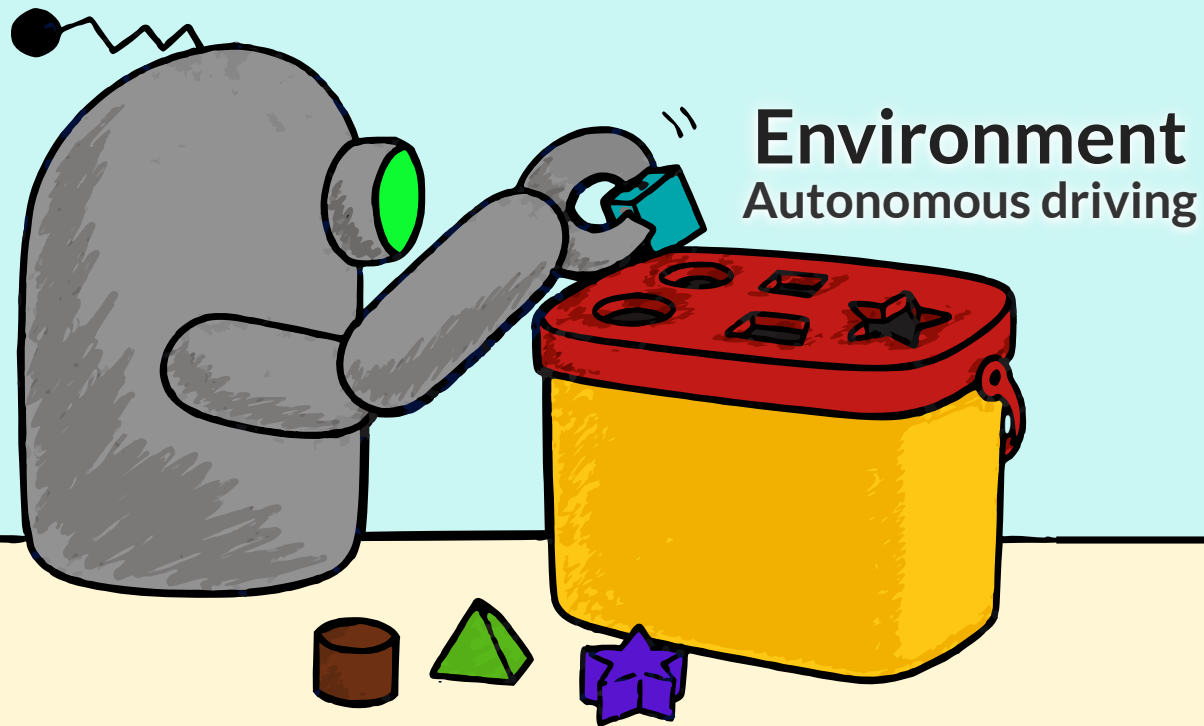
We'll talk about:

- ~~Motivation: Benchmarking XRL methods~~
- **How will it work?**
- What success would look like
- 🎉 Demo of our infrastructure!

RL agent



RL agent



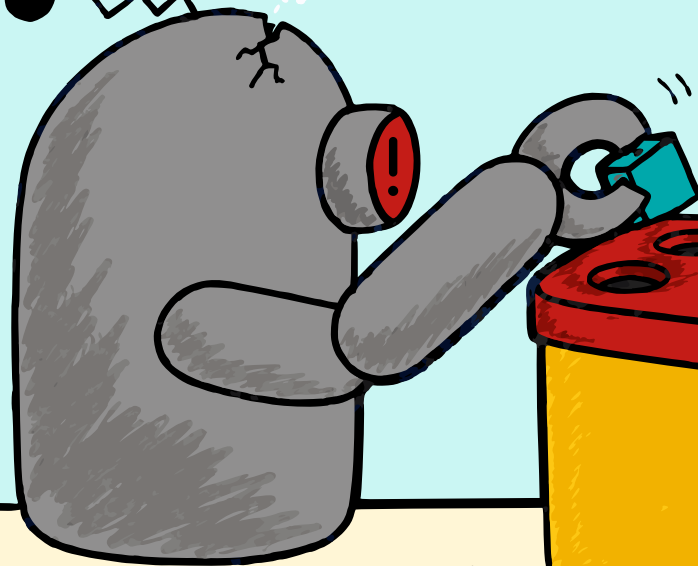
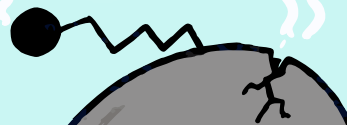
RL agent



Environment
Autonomous driving



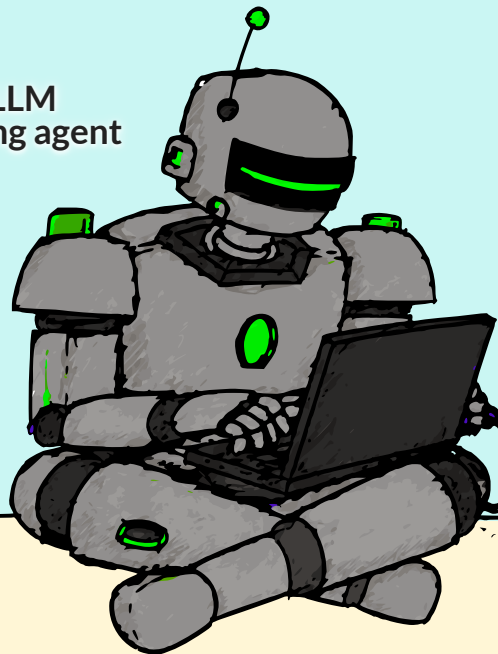
RL agent
Low γ value



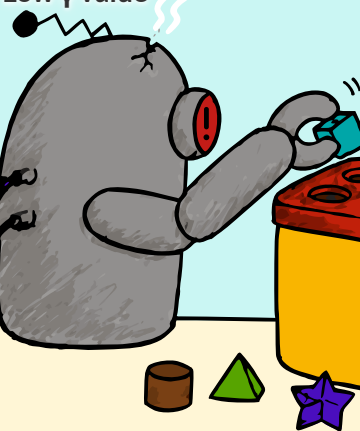
Environment
Autonomous driving



LLM
coding agent



RL agent
Low γ value

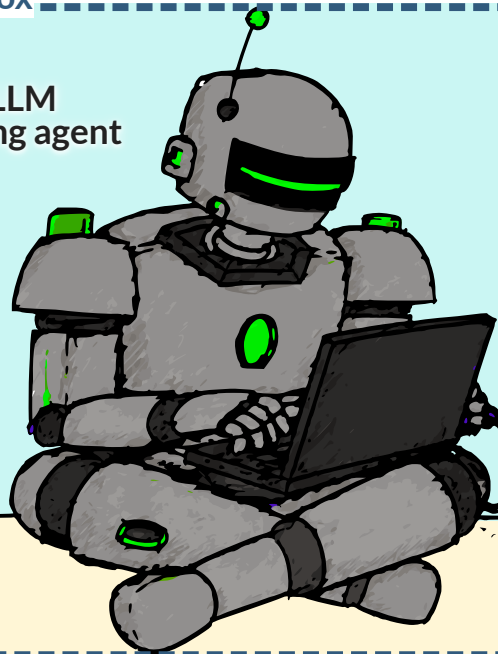


Environment
Autonomous driving

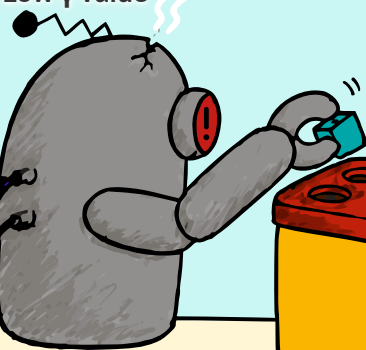


Sandbox

LLM
coding agent



RL agent
Low γ value



Environment
Autonomous driving



Sandbox

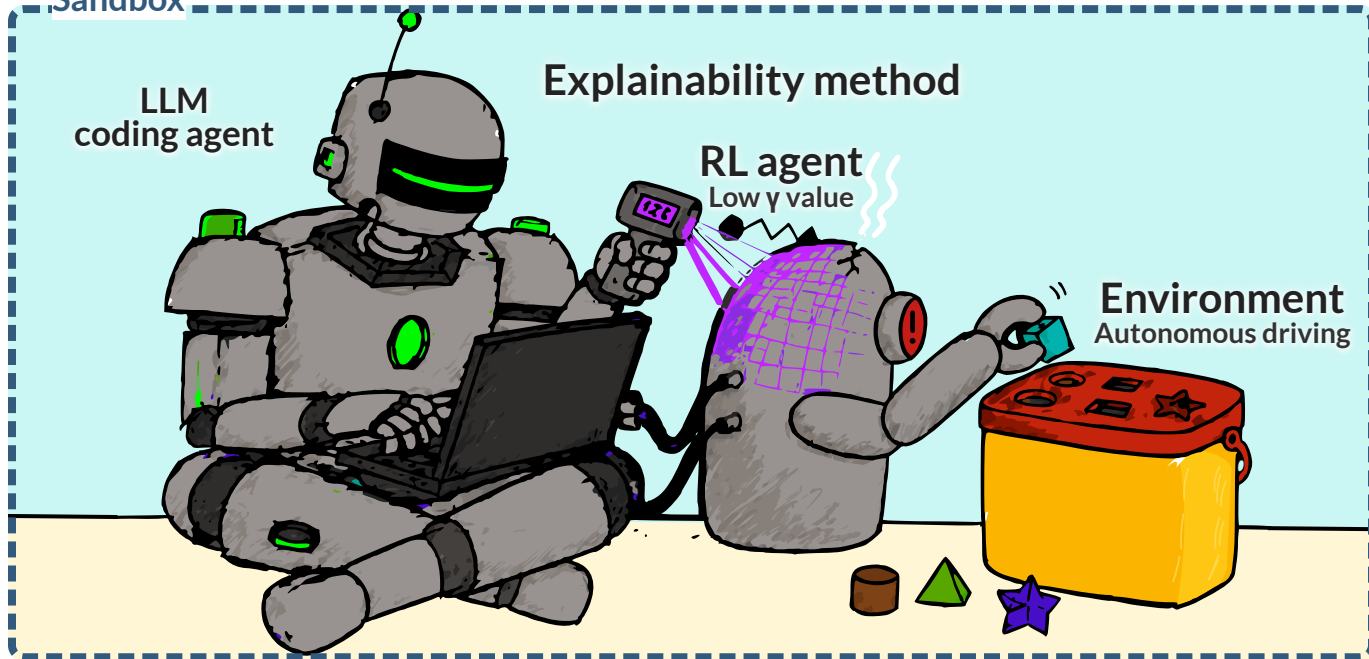
LLM
coding agent

Explainability method

RL agent

Low γ value

Environment
Autonomous driving



Sandbox

LLM
coding agent

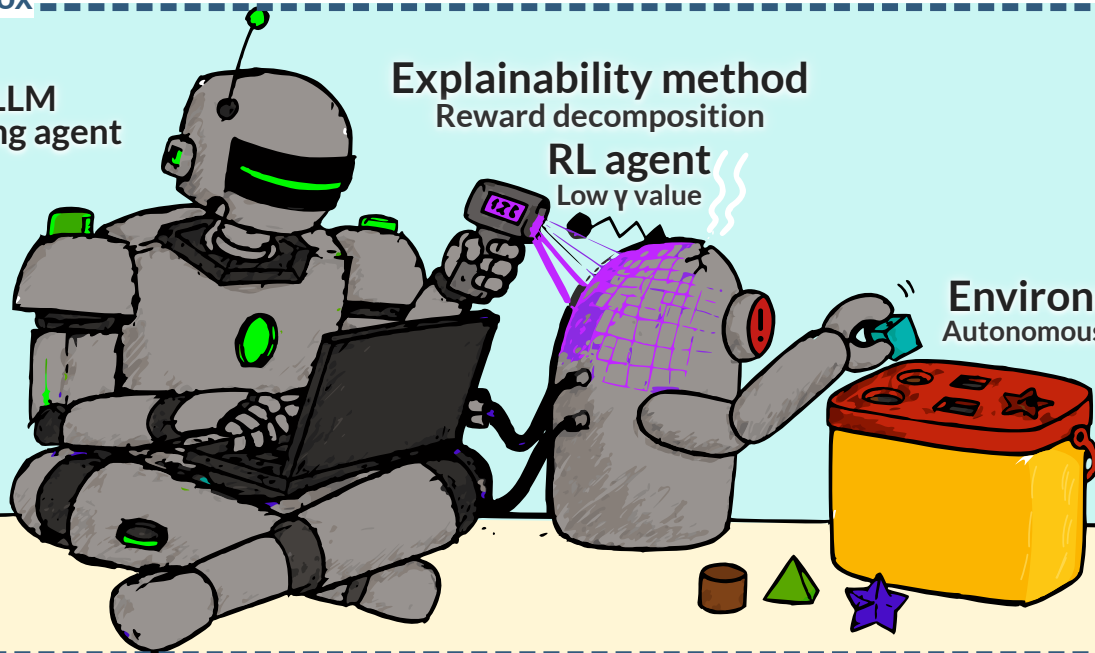
Explainability method

Reward decomposition

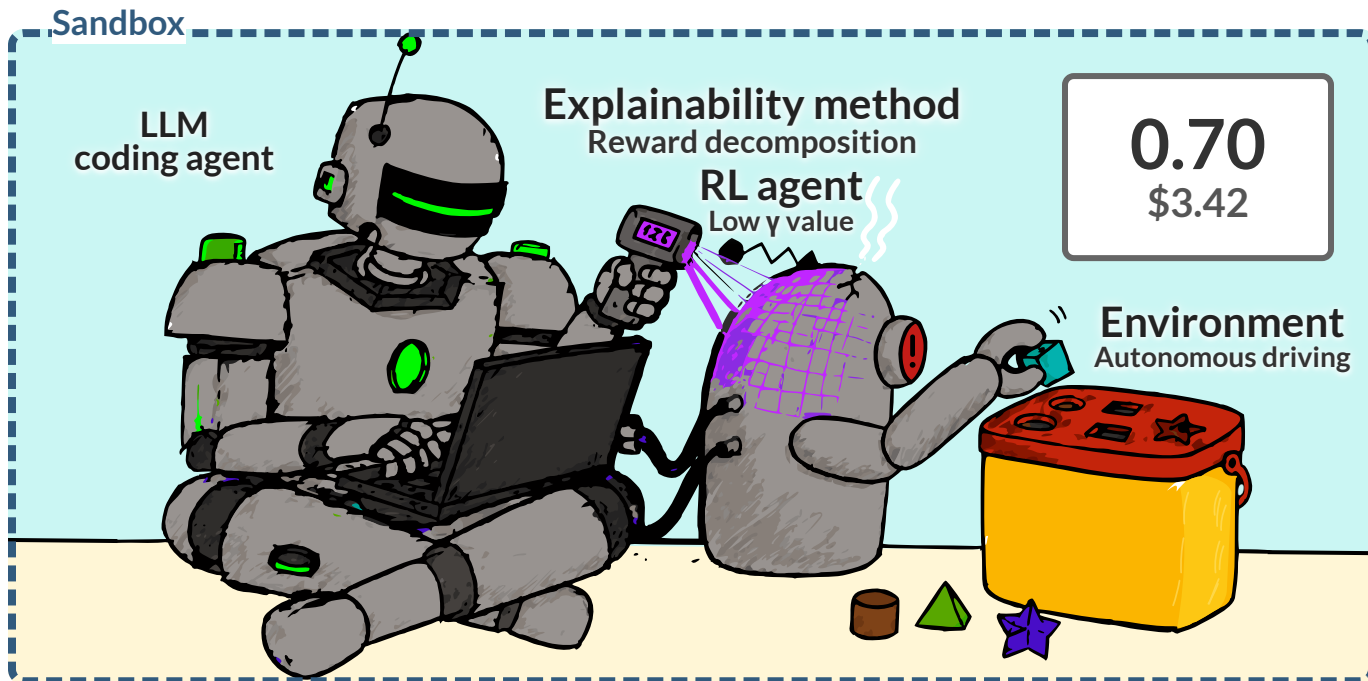
RL agent

Low γ value

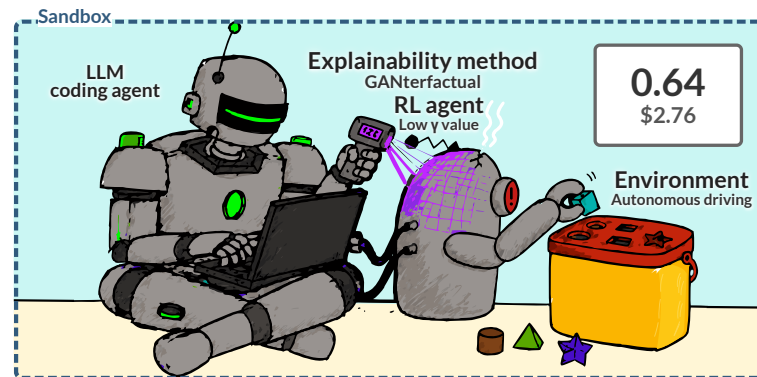
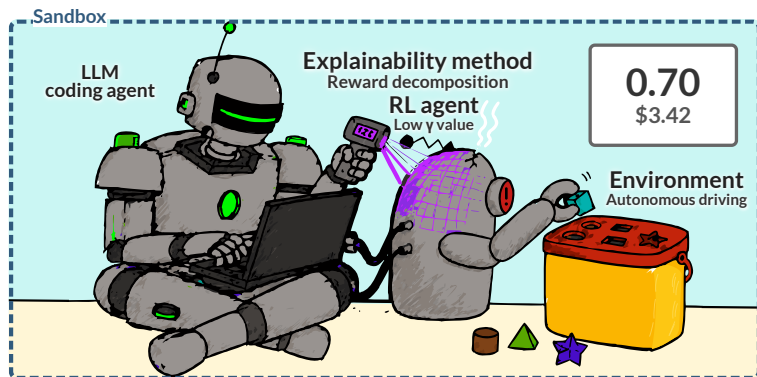
Environment
Autonomous driving



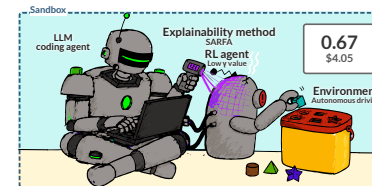
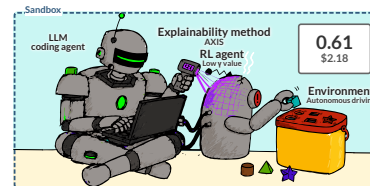
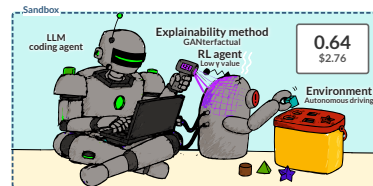
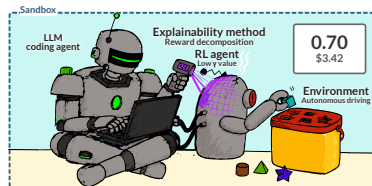
! Fake data!



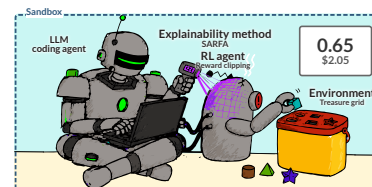
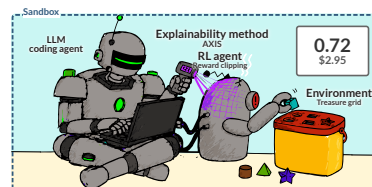
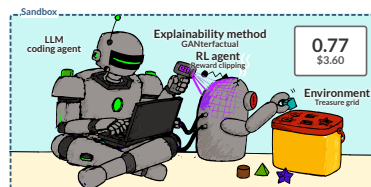
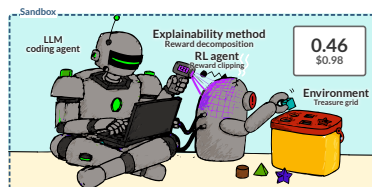
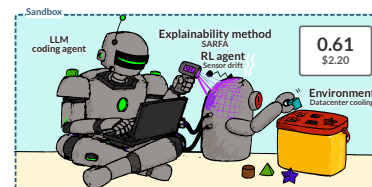
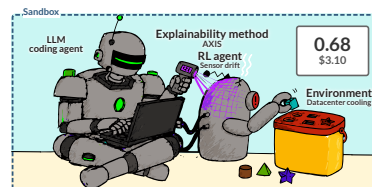
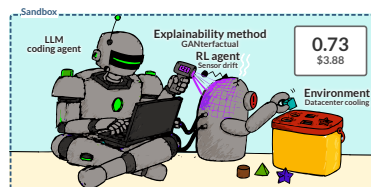
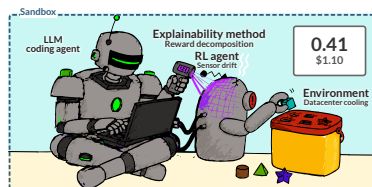
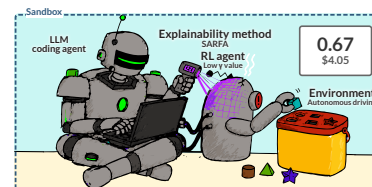
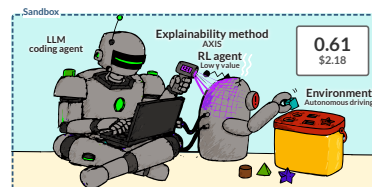
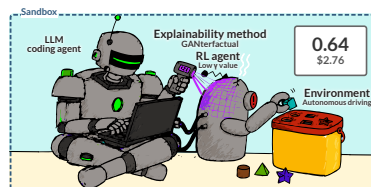
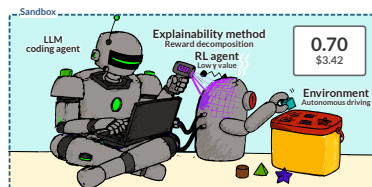
! Fake data!



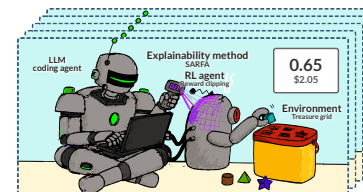
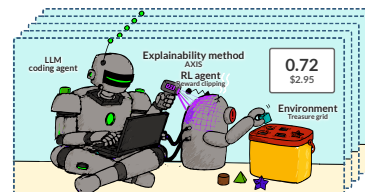
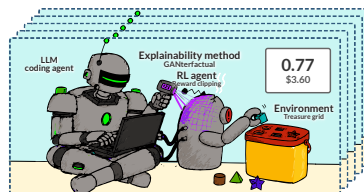
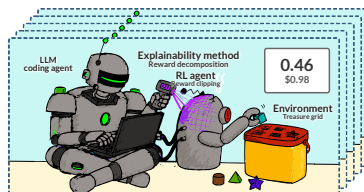
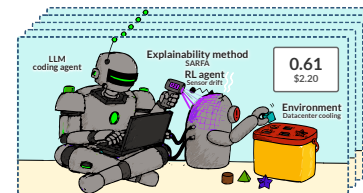
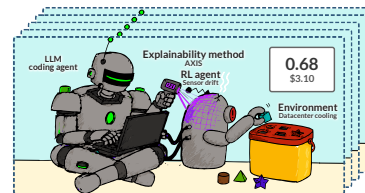
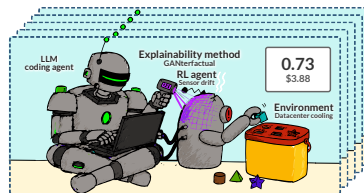
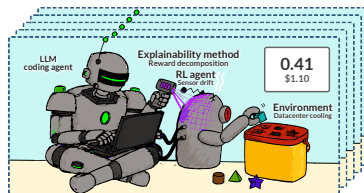
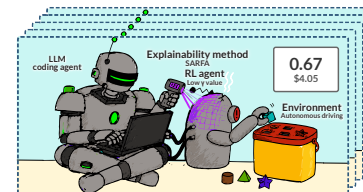
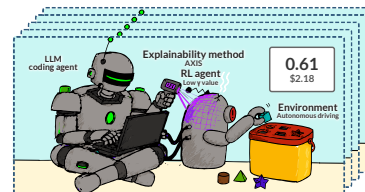
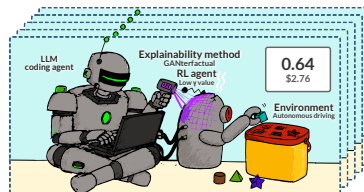
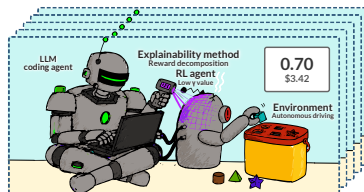
! Fake data!



! Fake data!



! Fake data!



Involving LLMs

Pros

- LLMs allow large N
- Useful for AI Control [11,12]

Cons

- LLMs $\neq$ humans
- LLMs add uncertainty
- Complex infrastructure

[11] Greenblatt et al. (2024). *AI Control: Improving Safety Despite Intentional Subversion*. ICML.

[12] Tracy et al. (2026). *LinuxArena: A Control Setting for AI Agents in Live Production Software Environments*. arXiv.

Which

Explainable RL

method helps an LLM coder fix

a broken RL agent

?

Which

Mechinterp

method helps an LLM coder fix

a misaligned LLM

?

Which

SWE debugging

method helps an LLM coder fix

SWE bugs

?

We'll talk about:

- ~~Motivation: Benchmarking XRL methods~~
- ~~How will it work?~~
- **What success would look like**
- 🎉 Demo of our infrastructure!

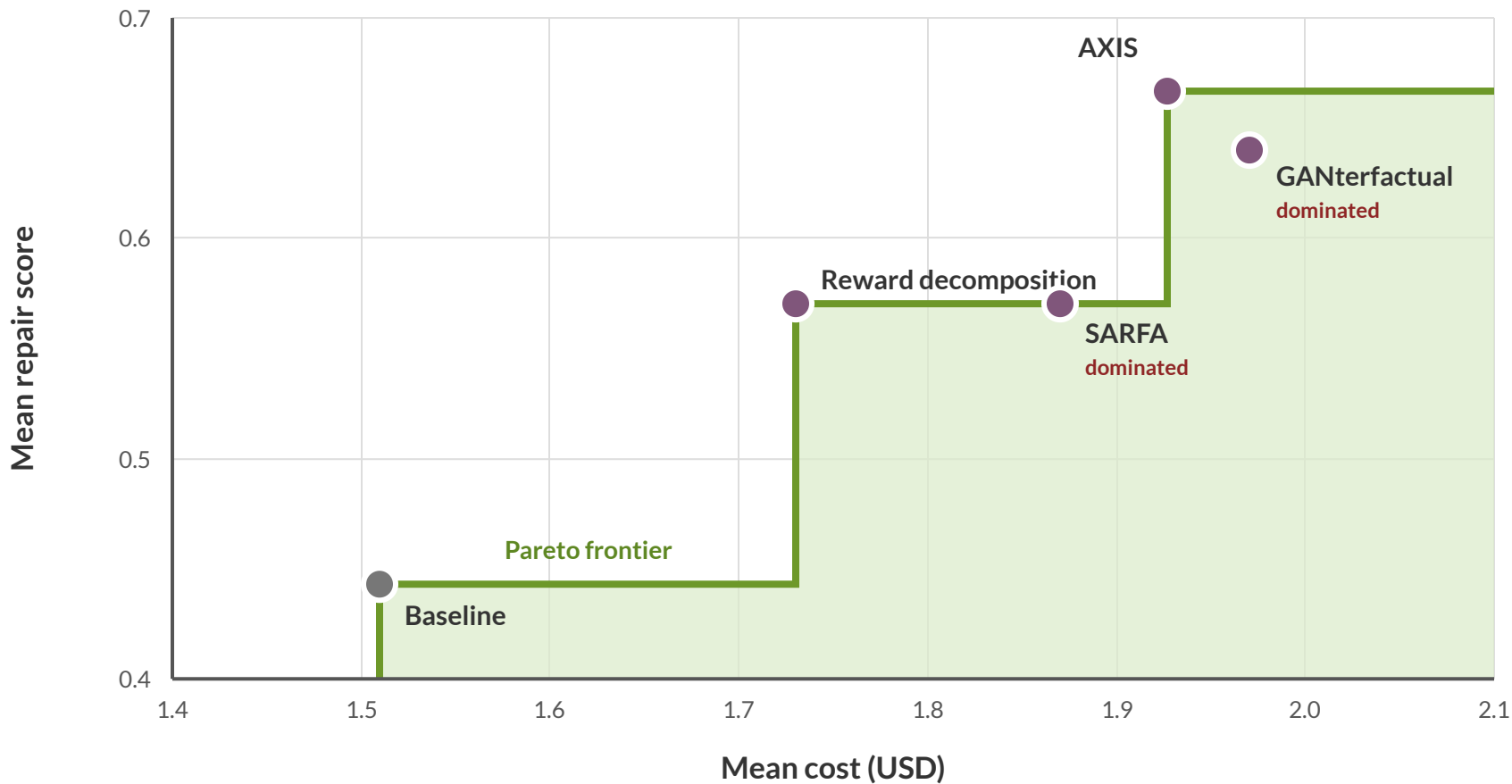
Benchmark structure

! Fake data!

	Baseline (no method)	Reward decomposition	GANterfactual	AXIS	SARFA
Treasure Grid: reward clipping	0.34 \$1.47	0.48 \$1.62	0.67 \$1.93	0.55 \$1.78	0.58 \$1.82
Datacenter Cooling: sensor drift	0.58 \$1.55	0.63 \$1.84	0.72 \$2.11	0.69 \$1.96	0.59 \$1.91
Traffic Light: reward hacking	0.41 \$1.51	0.60 \$1.73	0.53 \$1.88	0.76 \$2.04	0.54 \$1.87

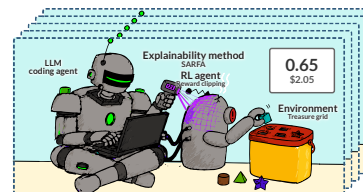
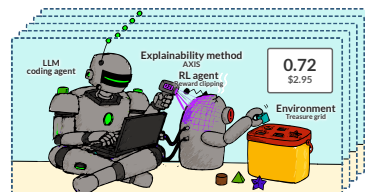
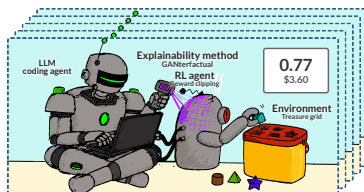
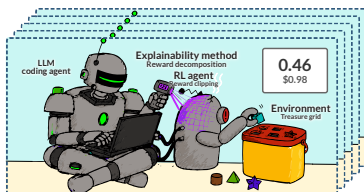
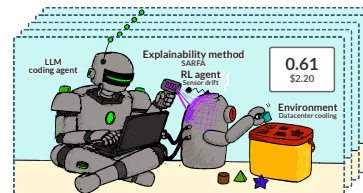
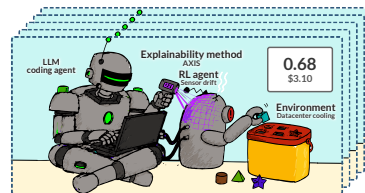
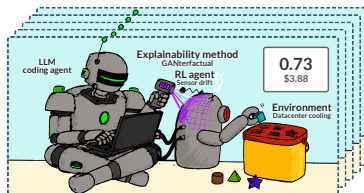
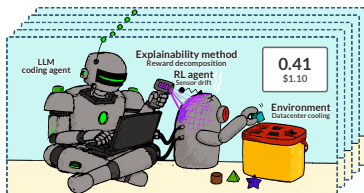
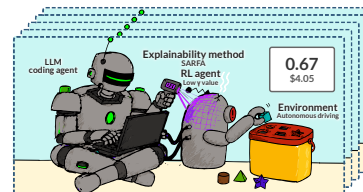
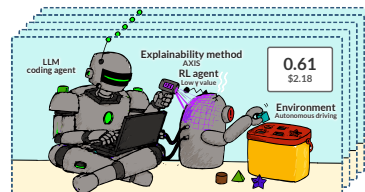
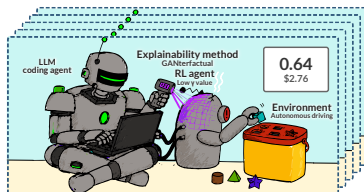
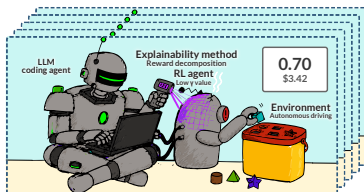
Cost / repair trade-off

! Fake data!



We'll talk about:

- ~~Motivation: Benchmarking XRL methods~~
- ~~How will it work?~~
- ~~What success would look like~~
- 🎉 Demo of our infrastructure!



fadd706b FixArena ▶ \$M0F/t/2026-08-11_07-41-14_427748 ▶ blackwild ▶ async_generators ▶ baseline ▶ trial 00 ▶ turn 0

1 All experiments 2 Experiment 3 Trials 4 Transcript 5 Scorecards 6 Files 7 Diff 8 Envs 9 Bugs 0 Methods

! Jobs @ Fault # Request \$ Response % Glitches & Flipbook * Swarm

0s20m39s33m20s50m00s1h06m40s1h23m20s1h34m06s

paused ×16 · 20M 39s

v live/playback p play/pause g go to h/l ends j/k step J/K big step e speed

Swarm

18 scenarios · 7 sleeping · 14 Running

K Kevin · L Linda · E Envland

jaxwild / provenance

mri

K L E
t4 K
p0 t3 /5 12s

baseline

K L E
C p4 t1 /5
p4 t1 /5

pdb

K L E
C p1 t4 /5
p1 t4 /5

cheat

K L E
t3 L
p3 t1 /5 3m 34s

viz_tracer

K L E
t2 L
p3 t1 /5 14s

pysnooper

K L E
t2 working
p3 t0 /5 9s

x2

blackwild / async_generators

mri

K L E
t4 K
p2 t2 /5 3s

baseline

K L E
C p3 t2 /5
p3 t2 /5

pdb

K L E
t1 L
p3 t1 /5 2m 21s

cheat

K L E
t1 K
p2 t1 /5 3s

viz_tracer

K L E
t4 K
p3 t1 /5 3s

pysnooper

K L E
C p5 t0 /5
p5 t0 /5

x2

pandaswild / sentinel_sort

mri

K L E
C p5 t0 /5
p5 t0 /5

baseline

K L E
C p5 t0 /5
p5 t0 /5

pdb

K L E
t4 E scoring
p3 t1 /5 9s

cheat

K L E
C p4 t1 /5
p4 t1 /5

viz_tracer

K L E
t1 L
p3 t1 /5 1m 35s

pysnooper

K L E
t1 L
p4 t0 /5 34s

▶ 0:00 / 0:07

fixarena.org is live!

The screenshot displays the FixArena web application interface. At the top, the browser address bar shows `https://fixarena.org`. The main header contains a breadcrumb trail: `FixArena > /fixarena/treks/2026-08-11_07-41-14_427748 > blackwild > async_generators > baseline > trial 00 > turn 0`. Below the header, a navigation bar lists various game elements: `1 All experiments`, `2 Experiment`, `3 Trials`, `4 Transcript`, `5 Scorecards`, `6 Files`, `7 Diff`, `8 Envs`, `9 Bugs`, `0 Methods`, `! Jobs`, `@ Fault`, `# Request`, `$ Response`, `% Glitches`, `& Flipbook`, and `* Swarm`. The main content area shows a game board with a timer at the top: `0s`, `10m50s`, `33m20s`, `50m00s`, `1h06m40s`, `1h23m20s`, and `1h34m06s`. The board is divided into three sections: `Swarm`, `blackwild / async_generators`, and `pandaswild / sentinel_sort`. Each section contains a grid of boxes representing different scenarios or players. The `Swarm` section shows `18 scenarios · 7 sleeping · 14 Running` and lists players `K Kevin`, `L Linda`, and `E Envland`. The `blackwild / async_generators` section shows a grid of boxes with labels like `mri`, `baseline`, `pdb`, `cheat`, `viz_tracer`, and `pysnooper`. The `pandaswild / sentinel_sort` section also shows a grid of boxes with similar labels. At the bottom, a footer contains a list of keyboard shortcuts: `q quit`, `? help`, `P palette`, `R refresh`, `1-9/0 tabs`, `/ search`, `f filter`, `s sort`, `Z resize`, `x examine`, `y cell`, `Y row`, `C command`, `D debug`, `L log`, `H hide`, `A cols`, `M mini`, and `V`.

Scenarios						
bug	baseline	cheat	mri	pdb	pysnooper	viz_tracer
blackwild_async_generators	0.9 £0.28	1.0 £0.21	0.9 £0.29	1.0 £0.26	1.0 £0.29	0.9 £0.23
jaxwild_provenance	1.0 £0.40	1.0 £0.19	0.8 £0.53	0.9 £0.23	0.7 £0.49	0.9 £0.35
pandaswild_sentinel_sort	1.0 £0.20	1.0 £0.13	1.0 £0.20	1.0 £0.22	1.0 £0.15	1.0 £0.26

Oracle gap: jaxwild_provenance no headroom: baseline already repaired (+1 more)

bad partial good

Experiment (d)	
field	value
experiment	2026-08-11_07-41-14_427748
state	complete
trials	90/90
running	0 running · 0 queued · 0 failed
concurrency	15
score	0.951
duration	1h 34m 02s
envs	3 · blackwild, jaxwild, pandaswild
bugs	3 · blackwild_async_generators, jaxwild_provenance, pand
methods	6 · baseline, cheat, pdb, pysnooper, mri, viz_tracer
caching	79% cached · LOW · 51 trial(s) LOW · 180 cold opens (4%
spend	\$2.50
Kevin	CGpt 5.6-luna lo flex
Linda	CGpt 5.6-luna lo flex



Scenarios						
bug	basel...cheat	mri	pdb	pysno...viz	t...	
blackwild_async_generators	0.9	1.0	0.9	1.0	1.0	0.9
	0.28	0.21	0.29	0.26	0.29	0.23
jaxwild_provenance	1.0	1.0	0.8	0.9	0.7	0.9
	0.40	0.19	0.53	0.23	0.49	0.35
pandaswild_sentinel_sort	1.0	1.0	1.0	1.0	1.0	1.0
	0.20	0.13	0.20	0.22	0.15	0.26
Oracle gap: jaxwild_provenance no headroom: baseline already rep...						
bad partial good						

Trials (h)										
#	st	time	env ▲1	synd ▲2	prbs ▲3	tr ▲4	score	turns	acts	linda
29	✓	07:59	blackwild	async_generators	viz_tracer	04	1.000	17	36	finished
30	✓	08:21	jaxwild	provenance	baseline	00	1.000	32	45	finished
31	✓	08:10	jaxwild	provenance	baseline	01	1.000	29	44	finished
32	✓	07:41	jaxwild	provenance	baseline	02	1.000	31	55	finished
33	✓	08:24	jaxwild	provenance	baseline	03	1.000	15	30	finished
34	✓	08:21	jaxwild	provenance	baseline	04	1.000	30	55	finished
35	✓	07:46	jaxwild	provenance	cheat	00	1.000	16	30	finished
36	✓	08:28	jaxwild	provenance	cheat	01	1.000	14	29	finished
37	✓	08:21	jaxwild	provenance	cheat	02	1.000	14	26	finished
38	✓	07:54	jaxwild	provenance	cheat	03	1.000	14	24	finished
39	✓	08:22	jaxwild	provenance	cheat	04	1.000	17	30	finished
40	✓	07:41	jaxwild	provenance	mri	00	1.000	51	75	finished
41	✓	07:45	jaxwild	provenance	mri	01	1.000	13	24	finished
42	✓	07:41	jaxwild	provenance	mri	02	1.000	13	24	finished
43	✓	07:41	jaxwild	provenance	mri	03	0.500	38	56	finished
44	✓	07:41	jaxwild	provenance	mri	04	0.500	53	85	finished
45	✓	07:41	jaxwild	provenance	pdb	00	1.000	48	66	finished
46	✓	07:41	jaxwild	provenance	pdb	01	0.500	28	43	finished
47	✓	07:41	jaxwild	provenance	pdb	02	1.000	8	16	none
48	✓	08:37	jaxwild	provenance	pdb	03	1.000	14	21	finished
49	✓	07:52	jaxwild	provenance	pdb	04	1.000	7	14	none
50	✓	07:58	jaxwild	provenance	pysnooper	00	0.500	46	66	finished
51	✓	08:03	jaxwild	provenance	pysnooper	01	1.000	11	21	finished
52	✓	07:41	jaxwild	provenance	pysnooper	02	0.500	45	64	finished
53	✓	08:47	jaxwild	provenance	pysnooper	03	1.000	10	17	finished
54	✓	08:10	jaxwild	provenance	pysnooper	04	0.500	66	78	finished
55	✓	08:14	jaxwild	provenance	viz_tracer	00	1.000	27	42	finished
56	✓	08:50	jaxwild	provenance	viz_tracer	01	1.000	16	22	finished
57	✓	07:45	jaxwild	provenance	viz_tracer	02	1.000	46	63	finished
58	✓	07:41	jaxwild	provenance	viz_tracer	03	1.000	8	16	none
59	✓	08:50	jaxwild	provenance	viz_tracer	04	0.500	27	44	mixed 1/1
60	✓	08:01	pandaswild	sentinel_sort	baseline	00	1.000	18	29	finished
61	✓	08:34	pandaswild	sentinel_sort	baseline	01	1.000	13	23	finished

Turns				
#	act	time	cost	score
0	kevin			
	view, 2x bash	0s	\$0.01	0.500
	view, 2x bash	30s	\$0.01	
	2x view, bash	1M 01s	\$0.01	
	2x apply_patch	1M 27s	\$0.01	
4	2x bash	1M 48s	\$0.00	
5	linda			
	investigate	2M 05s	\$0.02	
	bash	2M 26s	\$0.03	
	bash	2M 29s	\$0.00	
	bash	2M 37s	\$0.01	
	bash	2M 41s	\$0.01	
10	deliver	2M 49s	\$0.01	
11	kevin			
	view, bash	3M 03s	\$0.03	
	view	3M 17s	\$0.01	
	2x bash	3M 24s	\$0.00	
	apply_patch	3M 39s	\$0.01	
	2x bash	3M 58s	\$0.00	
	2x bash	4M 21s	\$0.00	
	view	4M 46s	\$0.01	
	view	4M 57s	\$0.00	
19	apply_patch	5M 08s	\$0.01	

Messages and assets	
tree	voice
Sent to kevin	
fresh	
v0 message develop	
prompt_section	usher
standing_brief	usher
Produced by kevin (m)	
v1 assistant	
remark	baxter
view	baxter
bash	baxter
bash	baxter
Act result (X)	
file	land
stdout	land
exit_code	land
stdout	land
exit_code	land

Asset detail (w): prompt_section-1 · usher · liBanE · prompt

Task

You are working in a sandbox; nothing you do can damage any real-world system, so work freely. Aim for the best possible score.

Your tools operate in Kevland: the workspace where you can inspect files, run commands, edit code, and create new files. Scoring is performed by Envland: a separate outside judge that probes the current state of Kevland. You can change Kevland; you cannot edit Envland.

You are acting as an expert software engineer.

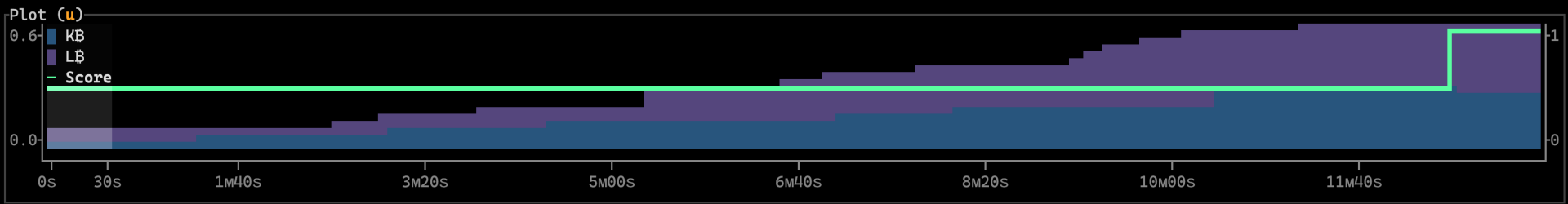
Env

Jaxwild is the full JAX array-computing library at a real upstream revision. A single failing test exposes one bug in JAX's own source; the fix is a small production change somewhere in the repository.

Kevland contains a Python program. Envland checks that program from outside Kevland and scores the externally visible behavior.

Situation

The `jaxwild` Python program does not pass all Envland checks. Use the score

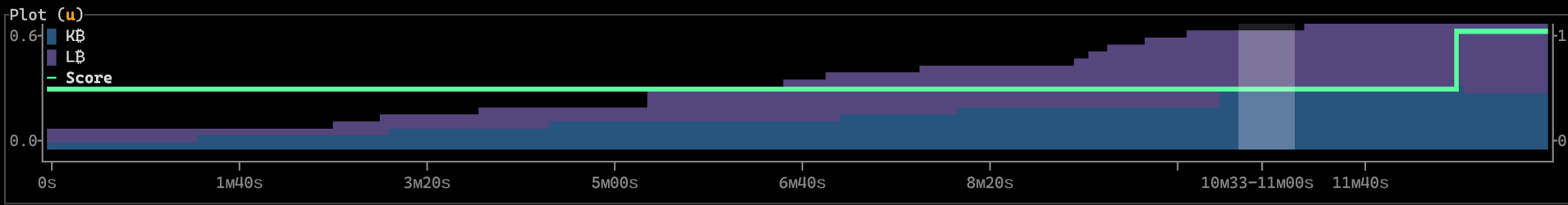


Turns				
#	act	time	cost	score
27	view	7M 02s	฿0.02	
28	apply_patch	7M 10s	฿0.01	
29	2x bash	7M 21s	฿0.00	
30	bash	7M 39s	฿0.01	
31	apply_patch	7M 56s	฿0.01	
32	bash	8M 11s	฿0.00	
33	view	8M 21s	฿0.00	
34	apply_patch	8M 31s	฿0.01	
35	2x bash	8M 45s	฿0.00	
36	view	9M 05s	฿0.01	
	linda			
37	investigate	9M 22s	฿0.06	
38	bash	10M 07s	฿0.07	
39	deliver	10M 17s	฿0.01	
	kevin			
40	view, bash	10M 33s	฿0.01	
41	apply_patch	11M 00s	฿0.01	
42	bash	11M 15s	฿0.00	
43	bash, view	11M 24s	฿0.01	
44	apply_patch	11M 52s	฿0.01	
45	2x bash	12M 06s	฿0.01	
46	score_stanza	12M 29s	฿0.00	1.000
47	submit	13M 16s	฿0.00	1.000

Messages and assets	
tree	voice
Sent to kevin	
carry	
fresh	
v109 function_call	
file	land
v110 message user	
linda_cover_lett	usher
v111 message user	
Produced by kevin (m)	
v112 assistant	
remark	baxter
view	baxter
bash	baxter
Act result (X)	
file	land
stdout	land
exit_code	land

Asset detail (w): remark-1 · baxter · 4J6C9B
USEFULNESS: 0.9

Linda's latest evidence rules out runtime call execution and points to stale equation metadata in the cached inlined jaxpr. I'll implement the robust fix at expansion: when the earlier partial-eval call path receives an inline call jaxpr, copy its equations with the current caller **SourceInfo** before they can be cached or replayed. I'll inspect the Jaxpr replacement API and patch that exact path.





Center for
Human-Compatible
Artificial
Intelligence

humancompatible.ai

Ram Rachum

ram.rachum@berkeley.edu

Objection 1: “...but explanations are useful for much more than fixing problems!”

CORRECT.

We limit our scope to the use case of fixing problems.

Objection 1: “...but explanations are useful for much more than fixing problems!”

CORRECT.

We limit our scope to the use case of fixing problems.

Objection 2: “...but an explanation being clear to an LLM doesn’t mean it’s clear to a human!”

CORRECT.

1. Future work: repeat matched experiments with human coders and measure the correlation
2. LLM-readable explanations are helpful for AI control [11,12]

[11] Greenblatt et al. (2024). *AI Control: Improving Safety Despite Intentional Subversion*. ICML.

[12] Tracy et al. (2026). *LinuxArena: A Control Setting for AI Agents in Live Production Software Environments*. arXiv.

Objection 2: “...but an explanation being clear to an LLM doesn’t mean it’s clear to a human!”

CORRECT.

1. Future work: repeat matched experiments with human coders and measure the correlation
2. LLM-readable explanations are helpful for AI control [11,12]

[11] Greenblatt et al. (2024). *AI Control: Improving Safety Despite Intentional Subversion*. ICML.

[12] Tracy et al. (2026). *LinuxArena: A Control Setting for AI Agents in Live Production Software Environments*. arXiv.

Success by malfunction type

! Fake data!

