A POPPERIAN NEURO-SYMBOLIC AGENT THAT LEARNS THROUGH LLM-BASED CONJECTURE AND REFUTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent advancements in Large Language Models (LLMs) have enabled new manners of integration with decision systems and autonomous agents, although they offer no guarantees for deductive reasoning nor plan generation. However, their capacity to provide probable, plausible answers to given inputs makes them suitable to be used as abductive reasoners, and brings the field to the era of popperian agents that weed out invalid possibilities in order to quicken learning. In this paper, we present a popperian agent architecture that uses an LLM-based component for epistemic queries in order to learn the dynamics of its environment: the LLM conjectures hypotheses, which are validated through thorough experimentation, and ultimately form a planning model that can be reused at near-zero cost, without the intervention of the LLM. We also present a provenance ontology that allows connecting each learnt piece of information with the experiments that validated it and the phenomenon that caused the piece of information to be conjectured at all.

1 INTRODUCTION

Large language models (LLMs) have reached a level of capability at which they are increasingly used not merely as interfaces, but as components of autonomous agents. This creates a tension: LLMs remain inconsistent across semantically equivalent formulations and repeated queries Liu et al. (2026b); Shapira et al. (2026) provide no guarantees of deductive correctness Mirzadeh et al. (2025); Bertolazzi et al. (2026), and may in fact exhibit less goal-directed behavior than alternative agent architectures Montese et al. (2026). These limitations are particularly relevant when an LLM directly influences an agent's decision: errors in reasoning become errors in action. At the same time, learning explicit environment dynamics has the benefit of enabling agents that do not require extensive computation to act at deployment time, once learning has taken place.

Our stance is to pursue a different role for LLMs. Rather than making the LLM the agent, we propose to use it as an isolated epistemic component: a fallible mechanism for generating candidate explanations of observations that the agent can subsequently test. This distinction separates *hypothesis generation* from *decision making*. The LLM is invoked when the agent encounters something its current model cannot explain, but the resulting conjectures are not immediately incorporated into the agent's beliefs or its policy. Instead, they must survive an explicit process of empirical refutation. Once sufficient environment dynamics have been learned, planning and acting proceed entirely over the acquired symbolic model, without further LLM inference unless new observations render that model inadequate.

This architecture is motivated by a Popperian view of learning. According to Dennett, we are entering the era of Popperian agents Dennett (1975), because LLMs can be and are being used as a mechanism that weeds out clearly invalid hypotheses without requiring testing. Moreover, LLMs have been observed to have potential as abductive reasoners Iturgaiz et al. (2026).

Agents that do not attempt to refute their hypotheses have no guarantees regarding when, or whether, they converge to correct models, and may cause unintended situations because of a misunderstanding of the environment's dynamics. Adding a strong refutation component means that, even if a learned piece of information is ultimately incorrect, there is a stronger empirical basis for tentatively being accepted as correct Popper (1957; 2005). Rather than accepting the first plausible explanation

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

generated for an unexpected observation, the agent actively attempts to identify observations or experiments that would contradict it. The purpose of refutation is therefore not to establish certainty, but to reduce the space of plausible explanations and reinforce the ones that survive. This way, candidate explanations can be rejected internally before they are relied upon in consequential behaviour with real effects on the environment.

Being incapable of refuting a hypothesis, however, does not make it automatically true. While refutation process minimize the chances of learning incorrect information, they do not completely eliminate this possibility: a realistically feasible suite of experiments in a complex environment cannot in general be shown to generalize and cover every future situation the learned dynamics may confront. Agents operating in real-world scenarios should therefore be built with the capability to provide explanations to repair behavior, so that in case of failure, a supervisor user can inspect what caused the agent to learn incorrect dynamics.

We present HYPER (*Hypo*thesiser-*E*xperimente*r*), an agent architecture in which an isolated LLM-based component, as opposed to an LLM-based agent, is used to answer epistemic queries that support the learning of an explicit planning model. The agent is agnostic to the particular planning language and can in principle be adapted to different planning formalisms. It learns autonomously in response to novelties and unexpected situations, invoking the LLM only when such situations occur. After learning the relevant environment dynamics, the resulting model can be used to complete the same tasks without further LLM calls, drastically reducing computational cost at deployment time. The agent returns to the LLM only when changes in the environment, the goals, or newly observed situations reveal inconsistencies w.r.t. its current model. Even then, learned dynamics are held only as tentatively correct: the agent remains receptive to future novelties that may require its knowledge to be revise.

More specifically, our contributions are:

- An agent learning procedure inspired by Popper's notion of bold hypothesis and severe refutation. We formalize the concepts of novely, conjecture, refutation, and experiments in the context of classical planning. Beyond the specific setting considered here, these concepts may provide a basis for formalizing agent-driven experimentation in other planning paradigms. This perspective also allows us to identify desirable properties of learning agents that are largely absent from the current LLM-agent literature.
- An agent architecture that isolates the LLM from the agent's decision-making process. The LLM is used for epistemic queries that support model learning, rather than directly determining actions. This can be subsequently used by a conventional planner, making the architecture independent of both the LLM and the particular planning language.
- A provenance mechanism grounded in the W3C-PROV standard. This mechanism makes it possible to trace an individual learned piece of knowledge back to the observations, conjectures, experiments, and refutation processes that let to its validation, with the purpose of supporting *post-hoc* inspection of the agent's learning processes and the diagnosis of incorrectly learned dynamics.

2 RELATED WORK

Learning symbolic models Automated planning requires knowing the effects of actions. Outside of toy examples, it is not assumed that an expert human can provide a perfect description of them. Planning literature has studied how to learn action effects from compiled state transitions Arora et al. (2018); Silver et al. (2023) or through interaction Lamanna et al. (2021); Sarathy et al. (2021).

Learning through novelty The concept of novelty has taken two different meanings, coming from two different fields. In Intrinsically-Motivated Reinforcement Learning, novelty is understood as the property of a state describing how much it diverges from the states that the agent has already seen Metzen & Kirchner (2013); Gatsoulis & McGinnity (2015); Hester & Stone (2017). It serves as a utility to maximise in order to increase exploratory behaviour. In classic agent literature, it is defined as an unpredicted inconsistency between an observation and the agent's internal beliefs about the environment Muhammad et al. (2021); Piotrowski et al. (2023); Goel et al. (2024). In all cases, it serves as a trigger to update the agent's beliefs.

LLMs and planning LLMs have been used as a replacement for classical planners Huang et al. (2022); Verma et al. (2025), or as a means to decompose a complex task in sub-goals Peng et al. (2024); Kwon et al. (2025). Recent literature has shown that, although state-of-the-art LLMs are capable of generating valid complex plans, their performance remains sensitive to domain and representation as LLMs lack the properties and guarantees of classical planning methods Corrêa et al. (2026); Liu et al. (2026a), and more generally, deductive reasoners Mirzadeh et al. (2025); Bertolazzi et al. (2026).

LLMs have also served to generate environment transition models from textual descriptions Guan et al. (2023); Oswald et al. (2024); Hu et al. (2025a), compiled state transitions Tsividis et al. (2021); Dainese et al. (2024); Hu et al. (2025b); Khan et al. (2026) or online interaction Tang et al. (2024). The latter also introduces an explicit model revision mechanism. Some produce abstracted transition models that function a layer above low-level actuation Piriyakulkij et al. (2025); Ahmed et al. (2026).

Science and discovery Karl Popper expressed that the most rational procedure for scientific discovery is that of conjecture and refutation: boldly proposing a theory, extensively testing it, and tentatively accepting it while no observation is encountered that refutes it Popper (1957). Although later philosophers revised Lakatos (1976) or plainly rejected Kuhn & Hacking (2012) Popper's ideas, they remain a good fit for agents existing in deterministic environments. Popper also stated that all theories should be susceptible to scrutiny, and that it is natural to stop seeking new knowledge when the existing is enough to carry on for the time being Popper (2005).

Hypothesis-driven learning Hypothesis generation and validation is mostly found in literature on AI for scientific discovery. Here, hypothesis testing uses sequential falsification tests and auditable conjecture–test–evidence loops Huang et al. (2025); Takahara & Mizoguchi (2026), acting under the premise that passing a finite set of tests does not prove a rule true. Hypothesis-driven exploration has also been explored in open-world robotics to expand and update incomplete world models with previously unknown objects, attributes or environment dynamics Xiao et al. (2026), but not from completely empty domains.

Provenance in agent interactions Recently, provenance mechanisms have emerged that record agentic execution artifacts like prompts and model responses to support *post-hoc* queries in LLM-based agentic workflows Souza et al. (2025); Wang et al. (2026). There also exist more general data provenance ontologies like PROV Lebo et al. (2013).

3 METHOD

In Sec. 3.1, we present the necessary definitions and properties to understand our agent, based on established agent literature Wooldridge & Jennings (1995); Rao & Georgeff (1991). In Sec. 3.2, we discuss the components that form HYPER and the processes it performs, and in Sec. 3.3, we explain of our agent learning provenance ontology. Lastly, in Sec. 3.4, we give additional details related to the implementation of the agent.

3.1 DEFINITIONS

An **agent** a is a system that is able to perceive their environment, and react to it by acting without the direct intervention of a human. Specifically, **discrete-time environments** $\mathcal{W}$ contain discrete timesteps $t_1, ..., t_T$. At each timestep, a may observe its **state** s_t and perform an action α_t from the set of allowed actions $\mathcal{A}$.

An agent may have a **goal** g, which we define as a boolean condition expressed in a logic language. A state s_t may or may not satisfy g. Agent a satisfies or accomplishes g when it observes a state satisfying g. An agent a can be said to be **goal-directed** with respect to some goal g if it acts in such a way that its actions bring about g with some measure of efficiency (*i.e.*its course of action satisfies the goal reasonably fast compared to alternative courses of action).

A **plan** $\pi := \{\alpha_1, ..., \alpha_n\}$, $\alpha_i \in \mathcal{A}$ is an ordered sequence of actions that is executed in n consecutive timesteps and brings about a given goal g. A **planning** process is the automatised search of a

π, given g. These processes use solvers, which use descriptions of the effects of the actions, usually called an **action model**. For instance, in classical planning with PDDL, the domain description would correspond to the domain file.

Any description of $\mathcal{W}$ that is exploited by an agent to achieve a goal constitutes a **world model**. Action models, by describing effects of actions in the world, are world models.

A **novelty** is a recorded divergence between an agent's observation and its internal prediction of it. Taking the markovian assumption and assuming a deterministic, fully-observable environment, we formalise it as $\delta := \langle s, \alpha, s'_{pred}, s'_{real} \rangle$, a tuple containing the previous observation s, action performed α, and the differing predicted and observed next states s'_{pred} and s'_{real}. Assuming that the environment is fully observable and not probabilistic, novelties are detected by just testing the equality of two states. Defining novelty detection for other kinds of environments falls out of scope of this paper.

A **hypothesis** h is the proposed cause for a particular novelty δ, which when added as a belief to the model M would modify the predicted s'_{pred} to match the real s'_{real}.[1] To be a **valid hypothesis**, it mus be falsifiable: that is, under some conditions, it can be proven false.

A hypothesis' value can be validated empirically by trying to disprove it. Given h, an **experiment** $e_h := \langle c_e^{init}, \alpha_e, c_e^{pass} \rangle$ is a tuple containing an initial condition c_e^{init} under which the experiment can be started, an action α_e that is to be executed, and a passing condition c_e^{pass}. An experiment passes when α_e is performed in a state that satisfies c_e^{init}, and the resulting next observation satisfies c_e^{pass}. Experiments should be designed such that a false, untested hypothesis will not pass them, picking possible adversarial conditions where it is expected to fail. A hypothesis with enough value or merit becomes a new belief candidate.

We define **epistemic memory** as the capacity to relate individual beliefs to the elements that created it. In this work, a belief b_h is the result of a hypothetisation process that proposed it as a hypothesis h (including the novelty δ and the model M at the time), but also of its validation process (including all experiments e_h).

3.2 ARCHITECTURE

HYPER is a neuro-symbolic architecture that combines verifiable, classical planning with LLM-based processes during the learning stage.

Its behaviour can be separated in two main processes: goal-directed and conjecture-refutation processes. Briefly, HYPER uses classical planning in order to accomplish its goal (goal-directed). If it encounters unpredicted phenomena or it does not have enough knowledge to achieve said goal, it enters a conjecture-refutation process in order to explore and learn new environment dynamics. Goal-directed behaviour mainly uses algorithmic components, while conjecture and refutation are orchestrated by an experiment validation algorithm while mainly using queries to an LLM.

In this paper, HYPER is instantiated with zero knowledge about the environment. In this setting, it mainly works in a loop of conjecturing, refuting, and checking whether its knowledge is now enough to solve the task.

World model In order to integrate the results of LLM queries with a formal planning language, HYPER maintains two equivalent action models: a list of plain text statements M_{text} and its translation $M_{\mathcal{L}}$ to formal planning language $\mathcal{L}$ of choice.

Note that the equivalence between both models depends on the capability of the selected LLM to produce correct LLM in the selected formal planning of choice. In addition, the propertie of chosen language $\mathcal{L}$ also force certain properties in the environments to learn.

Goal-directed processes Goal-directed behaviour is manifested through the **Planner** and the **Plan runner** components.

[1] In this role we do not constrain this to monotonic logics: a hypothesis can reject or retract previous beliefs.

- The **Planner** is simply an algorithmic component that functions as defined in Sec. 3.1. It receives the current state s, a goal g, and the current action model $M_{\mathcal{L}}$, and it either generates a valid plan π that accomplishes g, or it returns a planning error signaling that it is not possible to reach g.
- The **Plan runner** is an algorithmic component that runs a given π, action by action, and detects novelties while the plan is running. After performing each action, it observes the environment and checks whether the new state is equal to the predicted state according to the current $M_{\mathcal{L}}$. In case of divergence, it halts the execution of π and returns the encountered novelty. Otherwise, it performs the entire π and finishes successfully.

In goal-directed processes, the agent translates its given extrinsic goal to the formal language of choice using the Domain translation component (explained below), uses the Planner component to generate a plan to achieve said goal and performs it via the Plan runner. In the cases of an inability to find a plan, or the detection of a novelty, a conjecture-refutation process is triggered in order to update the agent's world model.

Conjecture-refutation processes Conjecture and refutation processes are always performed as a couple. When conjecturing, an LLM is queried through the **Hypothesiser** in order to perform inference to the best hypothesis. When refuting, the **Experiment designer** queries an LLM to design experiments that validate the aforementioned hypothesis. These processes also involve other modules like the **Domain translator** and the **Belief integrator**.

- The **Hypothesiser** produces a plausible hypothesis under three different circumstances. On a planning failure, it is queried to propose a plausible environment behaviour that can be tested under the current circumstances (current state s and action models M_{text} and $M_{\mathcal{L}}$) that will bring the agent closer to extrinsic goal g. When a novelty is discovered, it is tasked with conjecturing an environment dynamic that explains the observed divergence, based on said novelty and M_{text}. Lastly, when an experiment e fails, the module is given e, the hypothesis that e rejects, and M_{text} to generate a new hypothesis that integrates the observed failure from e. In all three cases, the module is also given a list of failed experiments as input, to avoid repeating already rejected hypotheses.
- Given a hypothesis, the **Experiment designer** produces a list of experiments that validate it. The component first outputs an initial and passing conditions $\langle c_e^{init}, c_e^{pass} \rangle$ per experiment, both in plain text. It also selects an action to observe from the environment's available actions. If the hypothesis states various cases, this component is tasked to produce experiments to validate them all. It is given the current M_{text} to avoid testing behaviours exactly equal to those already learned.
- The **Domain translator** is a module that translates plain text descriptions to the formal planning language $\mathcal{L}$. It integrates a hypothesis h into the current $M_{\mathcal{L}}$ in order to obtain a new candidate model to validate alongside h. It also translates the extrinsic goal, and initial and passing conditions of experiemtns to goals. In all translations, the LLM is given as input at least one between $M_{\mathcal{L}}$ and the current state (expressed in $\mathcal{L}$ as well), as reference.
- Lastly, the **Belief integrator** receives the current action model M_{text} and an accepted hypothesis h. It determines whether d is a duplicate of an element in M_{text}, there is only partial overlap, or h is a completely new dynamic. In the second or third case, the module also integrates h in M_{text} (respectively, by modifying the existing overlapping elements with the information in h; or by adding h as is in M_{text}).

Conjecture processes consist in the generation of a hypothesis, and its translation to a candidate domain model, immediately starting a refutation process to validate them.

Refutation starts by generating a list of experiments E, and translating their initial and passing conditions. For each experiment, the Planner module checks whether its initial condition can be reached. From the reachable conditions, one $e := \langle c_e^{init}, \alpha_e, c_e^{pass} \rangle$ is randomly selected and a goal-directed sub-process towards c_e^{init} starts. Once reached, α_e is performed and observed. The new observation and the translated passing condition are given to one last small component, the **Experiment checker**, which returns true or false according to whether c_e^{pass} is satisfied after α_e. If e has failed, a new conjecture process is triggered. Otherwise, the refutation continues by checking

which other experiments' setup conditions can be reached. This continues in a loop until no more experiments can be performed. If all experiments pass, hypothesis h is integrated in M_{text}. If some but not all experiments pass, the LLM is used to modify h and its domain model $M^h_{\mathcal{L}}$ into a h' and $M^{h'}_{\mathcal{L}}$ that only describe the subset of h that has been validated through passing experiments[2], then integrated. If a novelty n is detected at any point during refutation, h is modified in the same manner with the currently passed experiments, and a new conjecture-refutation process is triggered to explain n.

Learning process HYPER starts learning from an empty action model, meaning that it starts unable to plan to reach the goal and needing to propose environment behaviours. Its learning loop consists on finding itself unable to accomplish its goal, conjecturing and validating beliefs, and expanding its action models in this manner while interrupted at times by a novelty or a failed experiment. Hypotheses proposed after failing to reach the goal are bold by nature, as they are novel and allow exploration and experimentation in new states Chalmers (2013). Hypotheses that are untestable at the time of its conjecturing are immediately disregarded, and very implausible hypotheses are never even proposed in the first time. The combination of these factors make HYPER a popperian agent Dennett (1975).

During learning, HYPER's action model will contain correct (sometimes over-generalised) beliefs but also may contain incorrect ones (by depending on LLMs, the agent loses some guarantees). Incorrect and over-generalised beliefs will produce novelties, which—assuming that, statistically, the system validates more correct than incorrect beliefs—will tend to correct wrong beliefs over introducing more noise in the action model, making it eventually converge into a model that is "correct enough" to accomplish the task.

After learning said model, HYPER can reuse it as many times as needed to solve tasks that contain the same dynamics as the learned one. That means that in deployment, it can perform a task repeatedly and efficiently without needing to involve the LLM[3] in new conjecture-refutation processes, only using its Planner and Plan runner modules. At the same time, if desired, HYPER can always keep automatically improving its action model via encountering novelties. By using a planner with a guarantee of correctness, HYPER avoids noise and possible distractions in its environment: if a plan to accomplish the task exists, it will find it regardless of other potentially novel but irrelevant factors.

3.3 ONTOLOGY

In order to capture not only LLM-based processes but the larger variety of processes that HYPER performs, we choose to expand provenance ontology PROV Lebo et al. (2013) instead of using existing agentic provenance mechanisms.

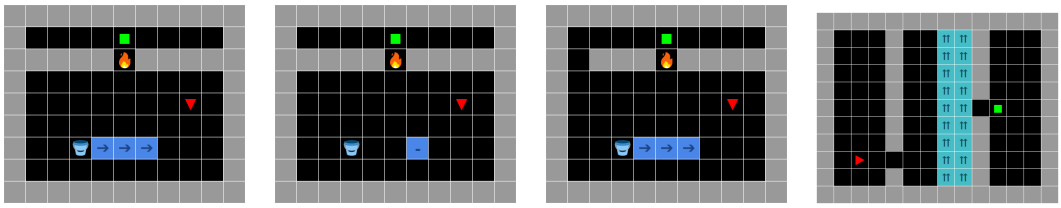
The PROV ontology defines the concepts of *Agent*, *Entity* (*i.e.*, any thing that is physical, digital, etc.) and *Activity* (*i.e.*, something that occurs over a period of time). We record every piece of information that the agent uses as separate entities. Among others, this includes hypotheses, observations, goals, plans, domain models, experiments, calls to LLM, etc. Every process (see Sec. 3.2) is recorded as an activity, including conjectures, refutations, goal-directed ones, and every call to an aforementioned component. Although technically unnecessary, we define a new type of activity for every type of process that the agent performs (*e.g.*, *ConjectureProcess*, *ExperimentDesignProcess*...) and a new type of entity for every type of data the agent uses (*e.g.*, *Hypothesis*, *Goal*, *Plan*...).

For every process, we use property *wasGeneratedBy* to describe what started it. As opposed to PROV, we allow actitivies to start other activities in order to describe subprocesses. We use properties *used* and *generated* to describe the data that a process took as input and produced as output, respectively.

We also introduce a new property, *wasCausedBy*, to describe the specific cases in which a process (and its outcome) directly cause other processes. The cases are: a) a *GoalDirectedProcess* planning

[2]If, somehow, no experiments can be performed, h is completely discarded as it is untestable under current M_{text} and $M^h_{\mathcal{L}}$.

[3]End users have the choice of giving the agent its goal g expressed in either plain text or language $\mathcal{L}$. The first option would still require using the LLM-based Domain translator to obtain g expressed in $\mathcal{L}$, but its cost is minimal.



(a) *fire_crossing* (b) *fire_crossing_easy* (c) *fire_crossing_shortcut* (d) *windy_corridor*

Figure 1: The four environments presented. The green square marks the goal cell, the red triangle corresponds to the agent, the blue cells mark water (whose arrow shows which direction it displaces objects), and the teal cells mark cells that automatically displace the agent upwards.

failure causing a *ConjectureProcess*, b) a *ConjectureProcess* always causing a *RefutationProcess*, and c) a *RefutationProcess* causing a *ConjectureProcess* when a novelty is encountered or an experiment does not pass.

Recording this information while the agent learns a task allows answering *post-hoc* queries about the process such as:

- **What experiments validated belief b?** Answered by compiling all passed experiments in the refutation process that integrated b.
- **What phenomenon caused the integration of belief b?** Answered by compiling the data used by every conjecture process that can be traced back via *wasCausedBy* relations. This may return multiple conjecture processes (each one with its own phenomenon to hypothesise about) in the case of arisen novelties or failed experiments inside refutation processes.

By being able of answering these questions, we can say that HYPER shows epistemic memory.

3.4 IMPLEMENTATION DETAILS

HYPER is implemented in Python 3.12, while LLM queries are design using the DSPy framework Khattab et al. (2024). Our ontology is implemented with Owlready2 Lamy (2017).

In this work, we showcase our agent with the Planning Domain Description Language (PDDL) McDermott et al. (1998) as the formal planning language of choice. Consequently, we assume environments to be deterministic and fully observable.

The planner component uses metric-ff Hoffmann (2003). The validator checks the syntactic and type correctness of generated PDDL code with the Parser and Validate programs from the VAL suite Howey et al. (2004) and the *pddl* Python package[4]. The domain translator module is given 5 tries to produce a valid PDDL in each text to translation process. After the first try, the module is also given the error message from the checker software to aid in fixing the invalid PDDL. Given goal g, we check whether it is satisfied by creating a planning problem that combines the current state and g, and checking if the planner returns the empty plan.

4 EXPERIMENTAL SETUP

In this work, we aim to show that the hypothesis-driven learning from HYPER is able to tackle complex dynamics that simpler learning paradigms like LLM-based planning cannot, and that it is improved by joining conjecture with refutation. We do so by comparing the performance of the full agent to a version of it without refutation and an LLM-based planner.

We choose to design environments inspired by the ones found on the HalluWorld benchmark Liu et al. (2026a), which feature complex, confusing and sometimes contradictory environment dynamics in order to study LLM hallucination under agentic interaction with the environment. Here, we

[4] https://ai-planning.github.io/pddl/

focus on environments that are semantically plausible and understandable, in which various dynamics come into play at once, while ensuring that they are deterministic and representable in formal language PDDL. We present the following environments (see also Fig.1), approximately ordered by complexity:

- **fire_crossing**: This environment features an empty bucket, a river and a cell on fire that blocks passage to the goal. The agent must learn how to pickup the bucket and fill it in the river, understanding that the river displaces objects in a fixed direction, and how to use the full bucket to extinguish the fire and cross to the goal.
- **fire_crossing_easy**: A variation of *fire_crossing* in which the river does not displace objects.
- **fire_crossing_shortcut**: A variation of *fire_crossing* in which there is a shortcut that allows direct passage since the start of the episode without needing to learn any dynamics other than movement. This one serves
- **windy_corridor**: This environment features cells that displace the agent in a fixed direction, who must take this into account to enter a 1 cell-wide corridor and reach the goal.

We evaluate 3 LLMs of different size and family: GPT 5.6 Luna[5], Kimi K2.7 Code[6] and Gemma 4 31B Team et al. (2026). For GPT, we perform 3 runs per combination of agent and environment; for Kimi and Gemma we perform 5. We track task success, and the number of generated output tokens and actions taken in the environment per run. Output tokens allow us to measure LLM-efficiency, while actions taken serves as a proxy for battery efficiency (useful, for example, in robotics).

We implement LLM-based planning by creating a new planner component that queries the LLM to directly generate a plan to serve the goal. For the version with no refutation, we directly accept every hypothesis coming from conjecture processes. We set a timeout limit of 200 goal-directed and/or conjecture-refutation processes for both verions of HYPER, and a limit of 100 goal-directed (planning) processes for the LLM planner. For all proposed environments, we understand these limits to be more than enough to learn all necessary dynamics.

5 RESULTS

Table 1 displays the performance of all three evaluated agent modalities in our four environments.

These results show that the full HYPER is the only modality that is capable of learning the most complex environment, *fire_crossing*. However, its success rate depends on the general quality of the LLM, at 2/3 for GPT 5.6 Luna and 0/5 for Gemma 4 31B. Cause of failure was timeout in all cases for GPT, 97% of all cases for Kimi and 95% for Gemma. The latter models had, respectively, 1 and 2 unrecoverable failures because of LLM output formatting.

Notably, we see that the LLM planner can only solve environments *windy_corridor* and *fire_crossing_shortcut*. Furthermore, in one instance, the GPT model was not able to identify that a shortcut existed on the latter environment and could not reach the goal.

In the more complex environments *fire_crossing* and *fire_crossing_easy*, HYPER without refutation processes was only able to match the full agent in 1/6 environment-model combinations. On the easier environments, it was capable of matching the full agent and use fewer output tokens and actions to do so.

In the cases in which both the LLM planner and HYPER were able to succeed, we see that the LLM planner generates less output tokens and in 3/4 cases, takes fewer actions in the environment to do so. Discrepancies in actions taken are explained by HYPER's need for additional experimentation in order to validate hypotheses. Regarding output tokens, given that once the task is learned, HYPER does not need to query the LLM in order to solve the task again, its learning cost with respect to the LLM planner is amortised by completing the task 3 to 46 times, depending on the specific task and model.

Among completed full HYPER runs of GPT 5.6 Luna, the best and most stable model from the ones tested, an average of 49.7% of the generated hypotheses ended up being accepted in all environments

[5] https://developers.openai.com/api/docs/models/gpt-5.6-luna

[6] https://huggingface.co/moonshotai/Kimi-K2.7-Code

Table 1: Performance of HYPERand variants. Data columns refer to number of successes, average millions of tokens spent per run and average number of actions taken in the environment. Tokens and actions are counted for successful runs. Results marked with (*) are invalid, as an error external to the agent contaminated these results. The table will be updated after reviews.

		Succ.	MTok	Actions
fire_crossing				
GPT 5.6 Luna	HYPER	2/3	0.399	358
	HYPER, no refut.	0/3	–	–
	LLM Planning	0/3	–	–
Kimi K2.7 Code	HYPER	1/5	0.568	153
	HYPER, no refut.	0/5	–	–
	LLM Planning	0/5	–	–
Gemma 4 31B	HYPER	0/5	–	–
	HYPER, no refut.	0/5	–	–
	LLM Planning	0/5	–	–
fire_crossing_easy				
GPT 5.6 Luna	HYPER	2/3	0.282	247
	HYPER, no refut.	0/3	–	–
	LLM Planning	0/3	–	–
Kimi K2.7 Code	HYPER	4/5	0.584	164
	HYPER, no refut.	4/5	0.242	44
	LLM Planning	0/5	–	–
Gemma 4 31B	HYPER	0/5	–	–
	HYPER, no refut.	0/5	–	–
	LLM Planning	0/5	–	–

		Succ.	MTok	Actions
fire_crossing_shortcut				
GPT 5.6 Luna	HYPER	3/3	0.109	109
	HYPER, no refut.	3/3	0.076	21
	LLM Planning	2/3	0.003	32
Kimi K2.7 Code	HYPER	*	*	*
	HYPER, no refut.	*	*	*
	LLM Planning	5/5	0.007	20
Gemma 4 31B	HYPER	*	*	*
	HYPER, no refut.	*	*	*
	LLM Planning	5/5	0.037	84
windy_corridor				
GPT 5.6 Luna	HYPER	3/3	0.091	125
	HYPER, no refut.	1/3	0.062	45
	LLM Planning	3/3	0.010	67
Kimi K2.7 Code	HYPER	5/5	0.379	81
	HYPER, no refut.	5/5	0.039	16
	LLM Planning	5/5	0.008	24
Gemma 4 31B	HYPER	4/5	0.325	70
	HYPER, no refut.	5/5	0.139	27
	LLM Planning	5/5	0.116	158

except *fire_crossing_shortcut*, for which 79.9% of them were. In the more complex environments *fire_crossing* and *fire_crossing_easy*, 70% to 80% of the hypotheses were conjectured after a failure to plan up to the goal. The percentage decreased to 40% to 50% in easier environments, in which novelties were more important and also generated 45% to 55% of hypotheses.

6 DISCUSSION AND CONCLUSIONS

Our experimentation results show that in environments that contain dynamics that are not very easily understandable and/or interact with each other, LLMs on their own struggle to understand them and act perform their tasks, at times even being distracted by them. Instead, the structure provided by HYPER, which exploits LLMs' capabilities to output probable answers to queries but does not commit to them, is a better fit. In addition, having the agent validate its conjectures is shown not only to be beneficial but also necessary. As has been seen, the cost of HYPER correctly learning a task with respect to using LLMs for direct actuation can be amortised in a relatively small number of repetitions, as after that it does not need to use more tokens. What this means is that our agent is especially suited for settings in which it will be deployed to repeatedly perform a task.

Given its learning loop, the agent mainly conjectures out of necessity, proposing new behaviours that will allow it to gradually explore more. Under other circumstances, for example by starting with a basic but limited action model, the agent may find more novelties than under the current setting. It is an interesting line for future research, to better understand the architecture. By being designed with language-agnostic components, HYPER can display different properties depending on planning language choice. We leave as future work to validate it in other planning languages that feature *e.g.* non-deterministic state transitions or partial visibility.

We believe that some of HYPER's shortcomings would be fixed by LLMs capable of processing inputs more coherently, as an inspection of its failed runs show that a number of learning processes were deviated by erroneous LLM output propagating to the action model. Given enough time, these erroneous models would be fixed–and before timeout, some were starting to be corrected–but there are no guarantees regarding the duration of that process. It is left as future work to review these phenomena and add additional learning stalling detection mechanisms.

Regardless, in this work we have proposed one of the first popperian agents as well as what is, to the best of our knowledge, the first agent that combines hypothesis-driven and novelty-driven learning with explicit belief validation, as well as a provenance ontology that gives it epistemic memory.

AI USE STATEMENT

In this work, generative AI has been used in order to find complementary literature to that which was found by humans. All literature suggested by it has been posteriorly reviewed by the authors of the work.

REFERENCES

Zergham Ahmed, Kazuki Irie, Joshua B Tenenbaum, Christopher J Bates, and Samuel J Gershman. Learning abstractions for hierarchical planning in program-synthesis agents. *arXiv preprint arXiv:2602.00929*, 2026.

Ankuj Arora, Humbert Fiorino, Damien Pellier, Marc Métivier, and Sylvie Pesty. A review of learning planning action models. *The Knowledge Engineering Review*, 33:e20, 2018.

Leonardo Bertolazzi, Sandro Pezzelle, and Raffaella Bernardi. How language models conflate logical validity with plausibility: A representational analysis of content effects. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 9316–9347, San Diego, California, United States, 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.454. URL https://aclanthology.org/2026.findings-acl.454/.

Alan F. Chalmers. *What is this thing called science?* Hackett Publishing Company, Inc, Indianapolis, 4. ed edition, 2013. ISBN 978-1-62466-038-2.

Augusto B. Corrêa, André G. Pereira, and Jendrik Seipp. Frontier large language models rival state-of-the-art planners, 2026. URL https://arxiv.org/abs/2511.09378.

Nicola Dainese, Matteo Merler, Minttu Alakuijala, and Pekka Marttinen. Generating code world models with large language models guided by monte carlo tree search. In *Advances in Neural Information Processing Systems*, volume 37, pp. 60429–60474, 2024.

D. C. Dennett. Why the law of effect will not go away*. *Journal for the Theory of Social Behaviour*, 5(2):169–188, October 1975. ISSN 0021-8308, 1468-5914. doi: 10.1111/j.1468-5914.1975.tb00350.x.

Y. Gatsoulis and T.M. McGinnity. Intrinsically motivated learning systems based on biologically-inspired novelty detection. *Robotics and Autonomous Systems*, 68:12–20, 2015. ISSN 09218890. doi: 10.1016/j.robot.2015.02.006.

Shivam Goel, Panagiotis Lymperopoulos, Ravenna Thielstrom, Evan Krause, Patrick Feeney, Pierrick Lorang, Sarah Schneider, Yichen Wei, Eric Kildebeck, Stephen Goss, Michael C. Hughes, Liping Liu, Jivko Sinapov, and Matthias Scheutz. A neurosymbolic cognitive architecture framework for handling novelties in open worlds. *Artificial Intelligence*, 331:104111, 2024. ISSN 00043702. doi: 10.1016/j.artint.2024.104111.

Lin Guan, Karthik Valmeekam, Sarath Sreedharan, and Subbarao Kambhampati. Leveraging pretrained large language models to construct and utilize world models for model-based task planning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 79081–79094, 2023.

Todd Hester and Peter Stone. Intrinsically motivated model learning for developing curious robots. *Artificial Intelligence*, 247:170–186, 2017. ISSN 00043702. doi: 10.1016/j.artint.2015.05.002.

Jörg Hoffmann. The metric-ff planning system: translating "ignoring delete lists" to numeric state variables. *J. Artif. Int. Res.*, 20(1):291–341, December 2003. ISSN 1076-9757.

R. Howey, D. Long, and M. Fox. Val: automatic plan validation, continuous effects and mixed initiative planning using pddl. In *16th IEEE International Conference on Tools with Artificial Intelligence*, pp. 294–301, 2004. doi: 10.1109/ICTAI.2004.120.

Mengkang Hu, Tianxing Chen, Yude Zou, Yuheng Lei, Qiguang Chen, Ming Li, Yao Mu, Hongyuan Zhang, Wenqi Shao, and Ping Luo. Text2World: Benchmarking large language models for symbolic world model generation. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 26043–26066, Vienna, Austria, July 2025a. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.1337. URL https://aclanthology.org/2025.findings-acl.1337/.

Mengkang Hu, Bowei Xia, Yuran Wu, Ailing Yu, Yude Zou, Qiguang Chen, Shijian Wang, Jiarui Jin, Kexin Li, Wenxiang Jiao, Yuan Lu, and Ping Luo. Agent2world: Learning to generate symbolic world models via adaptive multi-agent feedback, 2025b. URL https://arxiv.org/abs/2512.22336.

Kexin Huang, Ying Jin, Ryan Li, Michael Y. Li, Emmanuel Candes, and Jure Leskovec. Automated hypothesis validation with agentic sequential falsifications. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 25372–25437. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/huang25n.html.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International conference on machine learning*, pp. 9118–9147. PMLR, 2022.

Eneko Sabaté Iturgaiz, Victor Gimenez-Abalos, Adrián Tormos, Oriol Miro-Lopez-Feliu, and Sergio Alvarez-Napagao. Boldly propose, carefully verify: LLMs in agents should be used only for abduction. In *The Eighteenth Workshop on Adaptive and Learning Agents*, 2026. URL https://openreview.net/forum?id=roFMmPN96R.

Zaid Khan, Archiki Prasad, Elias Stengel-Eskin, Jaemin Cho, and Mohit Bansal. One life to learn: Inferring symbolic world models for stochastic environments from unguided exploration. In *International Conference on Learning Representations*, volume 2026, pp. 99794–99836, 2026.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. Dspy: Compiling declarative language model calls into self-improving pipelines. 2024.

Thomas Samuel Kuhn and Ian Hacking. *The structure of scientific revolutions*. University of Chicago press, Chicago, 4th ed edition, 2012. ISBN 978-0-226-45811-3.

Minseo Kwon, Yaesol Kim, and Young J. Kim. Fast and accurate task planning using neuro-symbolic language models and multi-level goal decomposition. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 16195–16201, Atlanta, GA, USA, May 2025. IEEE. ISBN 979-8-3315-4139-2. doi: 10.1109/ICRA55743.2025.11127617. URL https://ieeexplore.ieee.org/document/11127617/.

Imre Lakatos. *Falsification and the Methodology of Scientific Research Programmes*, pp. 205–259. Springer Netherlands, Dordrecht, 1976. ISBN 978-90-277-0630-0. doi: 10.1007/978-94-010-1863-0_14. URL http://link.springer.com/10.1007/978-94-010-1863-0_14.

Leonardo Lamanna, Alessandro Saetti, Luciano Serafini, Alfonso Gerevini, and Paolo Traverso. Online learning of action models for pddl planning. In *IJCAI*, pp. 4112–4118, 2021.

Jean-Baptiste Lamy. Owlready: Ontology-oriented programming in python with automatic classification and high level constructs for biomedical ontologies. *Artificial Intelligence in Medicine*, 80: 11–28, 2017. ISSN 0933-3657. doi: https://doi.org/10.1016/j.artmed.2017.07.002. URL https://www.sciencedirect.com/science/article/pii/S0933365717300271.

Timothy Lebo, Satya Sahoo, Deborah McGuinness, Khalid Belhajjame, James Cheney, David Corsar, Daniel Garijo, Stian Soiland-Reyes, Stephan Zednik, and Jun Zhao. Prov-o: The prov ontology. 2013.

Emmy Liu, Varun Gangal, Michael Yu, Zhuofu Tao, Karan Singh, Sachin Kumar, and Steven Y Feng. Halluworld: A controlled benchmark for hallucination via reference world models. *arXiv preprint arXiv:2605.19341*, 2026a.

Emmy Liu, Varun Gangal, Chelsea Zou, Michael Yu, Xiaoqi Huang, Alex Chang, Zhuofu Tao, Karan Singh, Sachin Kumar, and Steven Y. Feng. A unified definition of hallucination: It's the world model, stupid!, 2026b. URL https://arxiv.org/abs/2512.21577.

D. McDermott, M. Ghallab, A. Howe, Craig A. Knoblock, A. Ram, M. Veloso, Daniel S. Weld, and D. Wilkins. *PDDL - the planning domain definition language*. Number CVC TR-98-003/DCS TR-1165. 1998. URL https://www.semanticscholar.org/paper/PDDL-the-planning-domain-definition-language-McDermott-Ghallab/d82c6b8081343b2eae63d45feefe630233ad60e1.

Jan H. Metzen and Frank Kirchner. Incremental learning of skill collections based on intrinsic motivation. *Frontiers in Neurorobotics*, 7, 2013. ISSN 1662-5218. doi: 10.3389/fnbot.2013.00011. URL http://journal.frontiersin.org/article/10.3389/fnbot.2013.00011/abstract.

Seyed Iman Mirzadeh, Keivan Alizadeh, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. In *International Conference on Learning Representations*, 2025. URL https://mlanthology.org/iclr/2025/mirzadeh2025iclr-gsmsymbolic/.

Sara Montese, Sergio Alvarez-Napagao, Victor Gimenez-Abalos, and Ulises Cortés. The illusion of explainability with llms and llm- agents, April 2026. URL https://doi.org/10.5281/zenodo.19353068.

Faizan Muhammad, Vasanth Sarathy, Gyan Tatiya, Shivam Goel, Saurav Gyawali, Mateo Guaman, Jivko Sinapov, and Matthias Scheutz. A novelty-centric agent architecture for changing worlds. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 925–933, 2021.

James Oswald, Kavitha Srinivas, Harsha Kokel, Junkyu Lee, Michael Katz, and Shirin Sohrabi. Large language models as planning domain generators. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 34, pp. 423–431, 2024.

Shaohui Peng, Xing Hu, Qi Yi, Rui Zhang, Jiaming Guo, Di Huang, Zikang Tian, Ruizhi Chen, Zidong Du, Qi Guo, Yunji Chen, and Ling Li. Hypothesis, verification, and induction: Grounding large language models with self-driven skill learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(13):14599–14607, March 2024. doi: 10.1609/aaai.v38i13.29376.

Wiktor Piotrowski, Roni Stern, Yoni Sher, Jacob Le, Matthew Klenk, Johan deKleer, and Shiwali Mohan. Learning to operate in open worlds by adapting planning models. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '23, pp. 2610–2612, Richland, SC, 2023. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450394321.

Top Piriyakulkij, Yichao Liang, Hao Tang, Adrian Weller, Marta Kryven, and Kevin Ellis. Poeworld: Compositional world modeling with products of programmatic experts. In *Advances in Neural Information Processing Systems*, volume 38, pp. 26609–26638, 2025.

Karl Popper. *The Logic of Scientific Discovery*. Routledge, 0 edition, November 2005. ISBN 978-1-134-47002-0. doi: 10.4324/9780203994627. URL https://www.taylorfrancis.com/books/9781134470020.

Karl R Popper. Philosophy of science: A personal report. 1957.

Anand S. Rao and Michael P. Georgeff. Modeling rational agents within a bdi-architecture. In James Allen, Richard Fikes, and Erik Sandewall (eds.), *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning*, pp. 473–484, San Mateo, CA, USA, 1991. Morgan Kaufmann publishers Inc. URL http://jmvidal.cse.sc.edu/library/rao91a.pdf.

Vasanth Sarathy, Daniel Kasenberg, Shivam Goel, Jivko Sinapov, and Matthias Scheutz. Spotter: Extending symbolic planning operators through targeted reinforcement learning. In *AAMAS*, pp. 1118–1126, 2021. URL https://www.ifaamas.org/Proceedings/aamas2021/pdfs/p1118.pdf.

Natalie Shapira, Chris Wendler, Avery Yen, Gabriele Sarti, Koyena Pal, Olivia Floody, Adam Belfki, Alex Loftus, Aditya Ratan Jannali, Nikhil Prakash, Jasmine Cui, Giordano Rogers, Jannik Brinkmann, Can Rager, Amir Zur, Michael Ripa, Aruna Sankaranarayanan, David Atkinson, Rohit Gandikota, Jaden Fiotto-Kaufman, EunJeong Hwang, Hadas Orgad, P Sam Sahil, Negev Taglicht, Tomer Shabtay, Atai Ambus, Nitay Alon, Shiri Oron, Ayelet Gordon-Tapiero, Yotam Kaplan, Vered Shwartz, Tamar Rott Shaham, Christoph Riedl, Reuth Mirsky, Maarten Sap, David Manheim, Tomer Ullman, and David Bau. Agents of chaos, 2026. URL https://arxiv.org/abs/2602.20021.

Tom Silver, Ashay Athalye, Joshua B. Tenenbaum, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Learning neuro-symbolic skills for bilevel planning. In Karen Liu, Dana Kulic, and Jeff Ichnowski (eds.), *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 701–714. PMLR, December 2023. URL https://proceedings.mlr.press/v205/silver23a.html.

Renan Souza, Amal Gueroudji, Stephen DeWitt, Daniel Rosendo, Tirthankar Ghosal, Robert Ross, Prasanna Balaprakash, and Rafael Ferreira Da Silva. Prov-agent: Unified provenance for tracking ai agent interactions in agentic workflows. In *2025 IEEE International Conference on eScience (eScience)*, pp. 467–473. IEEE, 2025.

Izumi Takahara and Teruyasu Mizoguchi. Toward auditable ai scientists: A hypothesis evolution protocol for llm agents. *arXiv preprint arXiv:2607.09195*, 2026.

Hao Tang, Darren Key, and Kevin Ellis. Worldcoder, a model-based llm agent: Building world models by writing code and interacting with the environment. In *Advances in Neural Information Processing Systems*, volume 37, pp. 70148–70212, 2024.

Gemma Team, Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, Ian Ballantyne, Cormac Brick, Victor Cărbune, Michelle Casbon, Mayank Chaturvedi, Aditya Chawla, Victor Cotruta, Alice Coucke, Phil Culliton, Robert Dadashi, Lucas Dixon, Mohamed Elhawaty, Utku Evci, Clément Farabet, Johan Ferret, Filippo Galgani, Sertan Girgin, Jean-Bastien Grill, Maarten Grootendorst, Jiaxian Guo, Cassidy Hardin, Yanzhang He, Steven M. Hernandez, Omri Homburger, Léonard Hussenot, Juyeong Ji, Armand Joulin, Aishwarya Kamath, Parnian Kassraie, Olivier Lacombe, Preethi Lahoti, Gaël Liu, Gus Martins, Luciano Martins, Tatiana Matejovicova, Ramona Merhej, Nikola Momchev, Sneha Mondal, Ryan Mullins, Sindhu Raghuram Panyam, Shreya Pathak, Sarah Perrin, André Susano Pinto, Etienne Pot, Angéline Pouget, Alexandre Ramé, Sabela Ramos, Douglas Reid, David Rim, Morgane Rivière, Karsten Roth, Louis Rouillard, Omar Sanseviero, Pier Giuseppe Sessa, Shane Settle, Danila Sinopalnikov, Sara Smoot, Piotr Stanczyk, Andreas Steiner, Lawrence Stewart, Ilya Tolstikhin, Michael Tschannen, Anton Tsitsulin, Nino Vieillard, Renjie Wu, Pingmei Xu, Haichuan Yang, Edouard Yvinec, Biao Zhang, Li Zhang, Joe Zou, Nicolas Aagnes, Abdelrahman Abdelhamed, Jakub Adamek, Shivani Agrawal, Shubham Agrawal, Ibrahim Alabdulmohsin, Jean Baptiste Alayrac, Uri Alon, Chandramouli Amarnath, Ankesh Anand, Chrysovalantis Anastasiou, Setareh Ariafar, François-Xavier Aubet, Kyriakos Axiotis, Federico Barbero, Joelle Barral, Alexei Bendebury, Urs Bergmann, Stanley Bileschi, Kat Black, Mathieu Blondel, Sebastian Borgeaud, Arthur Bražinskas, Ryan Burnell, Robert Busa-Fekete, Mu Cai, Daniele Calandriello, Glenn Cameron, Charlotte Caucheteux, Rahma Chaabouni, Garima Chadha, Jetha Chan, Blake Jianhang Chen, Jesse Chen, Lin Chen, Xu Chen, Derek Cheng, Tzu hsiang Chien, Nikolai Chinaev, Yi Chou, Zhaohui Chu, Benjamin Coleman, Pooja Consul, Sam Conway-Rahman, Scott Crowell, Dylan Cutler, Vivek Dani, Samira Daruki, Anil Das, Daniel Deutsch, Nishanth Dikkala, Li Ding, Qiuhan Ding, Shenil Dodhia, Konstantin Donhauser, Tulsee Doshi, Anca Dragan, Alex Druinsky, Sahil Dua, Zoltan Egyed, Danielle Eisenbud, Daniel Eppens, Cindy Fan, Bahare Fatemi, Yassir Fathullah, Vlad Feinberg, Milen Ferev, Sebastian Flennerhag, Takumi Fujimoto, João Gabriel Oliveira, Isaac Galatzer-Levy, João Gante, Simon Geisler, Soham Ghosal, Antonious M. Girgis, Tamara von Glehn, Alec Go, Alhaad Gokhale, Alex Grills, Yiming Gu, Mayank Gupta, Pramod Gupta, Guru Guruganesh, Raia Hadsell, Hamza Harkous, Jitendra Harlalka, Demis Hassabis, Anja Hauth, Joe

Heyward, Arian Hosseini, Chih-Yang Hsia, I-Hung Hsu, Xiaopeng Huang, Yangsibo Huang, Kevin Hui, Adrian Hutter, Te I, Fotis Iliopoulos, Advait Jain, Ganesh Jawahar, Ziwei Ji, Qilin Jin, Melvin Johnson, Kandarp Joshi, Arun Kandoor, Wang-Cheng Kang, Koray Kavukcuoglu, Mehran Kazemi, Kathleen Kenealy, Amr Khalifa, Phoebe Kirk, Ivan Korotkov, Suraj Kothawade, Vitaly Kovalev, Neel Kovelamudi, Adam Kraft, Ravin Kumar, Vivek Kumar, Harish Kuppam, Justin Lannin, Chen-Yu Lee, Seungji Lee, Dmitry Lepikhin, Alon Levkovitch, Dongdong Li, Qiujia Li, Valentin Liévin, Ethan Lin, Ziqian Lin, Casper Liu, Tianlin Liu, Tianqi Liu, Xin Liu, Ivan Lobov, Mayank Lunayach, Min Ma, Gagan Madan, Andrii Maksai, Eric Malmi, Michal Matuszak, Daniel McDuff, Gaurav Menghani, Maciej Mikuła, Daniil Mirylenka, Karolis Misiunas, Vedant Misra, Andreea Mitran, Kareem Mohamed, Maksim Mukha, Eric Noland, James O'Donnell, Brendan O'Donoghue, Kate Olszewska, Bernett Orlando, Wanqiong Pan, Rina Panigrahy, Unnati Parekh, Nicolas Perez-Nieves, Chunjong Park, Eric Paskie, Liqian Peng, Bryce Petrini, Slav Petrov, Jonas Pfeiffer, Bilal Piot, Martyna Plomecka, Siim Poder, Octavio Ponce, Arijit Pramanik, David Racz, Anish Rajan, Michelle Ramanovich, Anand Rao, Marvin Ritter, Vitor Rodrigues, Evan Rosen, Mikołaj Rybiński, Noveen Sachdeva, Michaël E. Sander, Rohit Sathyanarayana, Sagar Savla, Samuel Schmidgall, Tal Schuster, George Scrivener, Benoit Seguin, Andrew Sellergren, Aliaksei Severyn, Izhak Shafran, Dhruv Shah, Bobak Shahriari, Yuan Shangguan, Ashish Shenoy, Pradeep Shenoy, Rakesh Shivanna, Pauline Sho, Lucas Spangher, Wojciech Stokowiec, Tim Strother, Yao Su, Yinghao Sun, Mukund Sundararajan, Andrea Tacchetti, Mor Hazan Taege, Pouya Tafti, Jean Tarbouriech, Chetan Tekur, Shantanu Thakoor, Rahul Thapa, Madeleine Traverse, Lenart Treven, Tao Tu, Chien Te Tung, Çağlar Ünlü, Petar Veličković, Malini Pooni Venkat, Sagar Gubbi Venkatesh, Vidya Venkiteswaran, Francesco Visin, Alex Vitvitskyi, Kiran Vodrahalli, Weiyi Wang, Xin Wang, Tris Warkentin, Jan Wassenberg, John Wieting, Cindy Wu, Lechao Xiao, Hao Xu, Yuhui Xu, Fuzhao Xue, Arun Yadav, Jun Yan, Antoine Yang, Lin Yang, Ming-Hsuan Yang, Ziyu Ying, Jae Hyeon Yoo, Morteza Zadimoghaddam, Sajjad Zafar, Fred Zhang, Jiageng Zhang, Jianyi Zhang, Xiaofan Zhang, Chao Zhao, David Zhou, and Chen Zou. Gemma 4 technical report, 2026. URL https://arxiv.org/abs/2607.02770.

Pedro A Tsividis, Joao Loula, Jake Burga, Nathan Foss, Andres Campero, Thomas Pouncy, Samuel J Gershman, and Joshua B Tenenbaum. Human-level reinforcement learning through theory-based modeling, exploration, and planning. *arXiv preprint arXiv:2107.12544*, 2021.

Pulkit Verma, Ngoc La, Anthony Favier, Swaroop Mishra, and Julie A. Shah. Teaching llms to plan: Logical chain-of-thought instruction tuning for symbolic planning. (arXiv:2509.13351), 2025. doi: 10.48550/arXiv.2509.13351. URL http://arxiv.org/abs/2509.13351. arXiv:2509.13351 [cs].

Yiqi Wang, Jiaqi Zhang, Taotao Cai, Zirui Liu, Qingqiang Sun, Zequn Sun, Zhangkai Wu, Manqing Dong, Mingkai Zheng, Xuefei Yin, and Yanming Zhu. From agent traces to trust: A survey of evidence tracing and execution provenance in llm agents, 2026. URL https://arxiv.org/abs/2606.04990.

Michael Wooldridge and Nicholas R. Jennings. Intelligent agents: theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995. ISSN 1469-8005, 0269-8889. doi: 10.1017/S0269888900008122.

Anxing Xiao, Hanbo Zhang, Tianrun Hu, and David Hsu. Hypothesis-driven Model Expansion under Uncertainty for Open-World Robot Planning. In *Proceedings of Robotics: Science and Systems*, Sydney, Australia, July 2026. doi: 10.15607/RSS.2026.XXII.181.

Author Contributions

Acknowledgments

A Appendix