CONTINUAL LEARNING AND REFINEMENT OF CAUSAL MODELS THROUGH DYNAMIC PREDICATE INVENTION

Enrique Crespo-Fernández **Oliver Ray** **Telmo de Menezes e Silva Filho** **Peter Flach**
enrique.crespofernandez@bristol.ac.uk
University of Bristol
Bristol, UK

ABSTRACT

Efficiently navigating complex environments requires agents to internalize the underlying logic of their world, yet standard world modelling methods often struggle with sample inefficiency, lack of transparency, and poor scalability. We propose a framework for constructing symbolic causal world models entirely online by integrating continuous model learning and repair into the agent's decision loop, by leveraging the power of Meta-Interpretive Learning and predicate invention to find semantically meaningful and reusable abstractions, allowing an agent to construct a hierarchy of disentangled, high-quality concepts from its observations. We demonstrate that our lifted inference approach scales to domains with complex relational dynamics, where propositional methods suffer from combinatorial explosion, while achieving sample-efficiency orders of magnitude higher than the established PPO neural-network-based baseline.

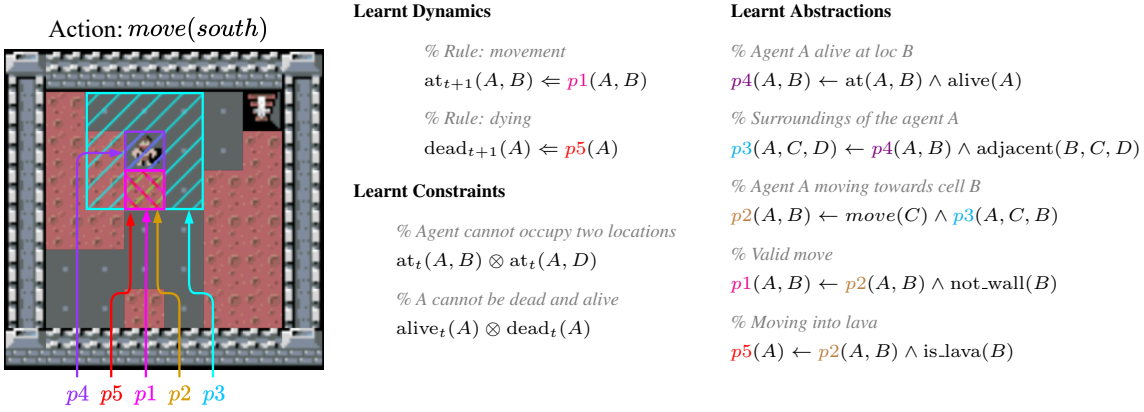

Figure 1: Visualization of a selected set of rules from the learnt symbolic causal model on the MiniHack 'Lava Crossing' task. The environment is modelled as a hierarchy of interpretable concepts, transition rules, and physical constraints. (Left) State Interpretation: Colored overlays illustrate how the Learnt Abstractions (Right) ground to specific regions of the state space. The agent ($p4$, purple) senses its neighborhood ($p3$, cyan). The concept of "moving" ($p2$, orange) is reused in two rules: the one modelling movement ($p1$) and the one modelling death ($p5$). (Center) Dynamics & Constraints: The Learnt Dynamics use these high-level abstractions to predict state evolution. For example, the dying rule is triggered only when the abstract condition $p5$ (moving into lava) is met. Learnt Constraints enforce physical consistency, such as mutual exclusion ($\otimes$), ensuring the agent cannot be simultaneously alive and dead or occupy multiple coordinates.

1 INTRODUCTION

Intelligent agents deployed in dynamic environments must learn compact world models from scarce experience, reason with them, and adapt when predictions fail. Modern deep model-based reinforcement learning offers powerful function approximation but remains data-hungry and opaque (Sutton & Barto, 2018; Moerland et al., 2022; Zhang & Yu, 2020). Conversely, traditional symbolic systems possess structural properties—specifically, inherent compositionality, the capacity for arbitrary-depth computation, and interpretability—that make them ideal candidates for the backbone of a causal agent. However, classical automated planning typically presupposes a hand-engineered symbolic world model (Ghallab et al., 2004; Fikes & Nilsson, 1971; Yu et al., 2023; Smet et al., 2025).

Bridging these paradigms requires a framework that acquires a symbolic dynamics model from experience while retaining the deliberative strengths of classical planning. The acquisition of such models requires an agent to formulate hypotheses that rationalize environmental changes as they occur, to test their consistency over time, and to update beliefs when they are falsified (Sutton & Barto, 2018; Evans et al., 2021b). Crucially, if the aim is general-purpose agency, the learner must be able to formulate and refine these hypotheses during interaction without reliance on pre-collected trajectories, batched histories, or external oracles, as is common in many existing Inductive Logic Programming (ILP) approaches (Evans et al., 2021b; Cropper et al., 2020; Cropper & Morel, 2021). To this end, we present a self-supervised, online framework for learning symbolic world models. Our agent operates in a continuous loop: it (i) incrementally **induces** a symbolic transition theory from streams of experience, (ii) **predicts** state evolution to plan actions, and (iii) **reacts** to prediction–observation mismatches by revising its model in real-time.

Building on recent developments in Meta-Interpretive Learning (MIL) (Muggleton et al., 2015; Cropper & Tourret, 2020; Patsantzis & Muggleton, 2021), our system generates logical explanations for unpredicted transitions in the language of First-Order Logic (see Appendix A for a primer on First-Order Logic and ILP). We employ a template-based approach to manage the combinatorial complexity of the search space. Furthermore, we introduce a dynamic predicate invention that composes short, general abstract rules to define complex relationships. This hierarchical structure ensures that the learned dynamic rules remain concise, general, and interpretable while remaining sufficiently robust to capture deep causal dependencies.

The main contributions of this work are summarized as follows. We introduce a **Continuous Model Repair** framework that utilizes prediction error to refine symbolic theories incrementally, eliminating the need for retraining. By using lifted inference and predicate invention, the complexity of our **Scale-Invariant Learning** mechanism depends only on the logic depth and vocabulary size and not on the grounding size, a standard limitation of propositional methods (ASP- and SAT-based). We empirically validate the **Sample Efficiency** of our system on grid-world environments, where it achieves superior sample efficiency compared to standard PPO Schulman et al. (2017) baselines.

By integrating symbolic learning and model repair into a single loop, we enable agents that are not only effective but also transparent and data-efficient. The remainder of this paper is structured as follows: Section 2 formalizes the MIL framework and the self-supervised Predict-Verify-Refine cycle. Section 3 presents an empirical evaluation of the system's sample efficiency and model repair capabilities in the MiniHack environment. Section 4 situates our work within the broader context of symbolic and neurosymbolic world models, and Section 5 concludes with a discussion of future research directions.

2 METHOD

This section details the MIL framework for acquiring predictive world models. We employ a self-supervised continuous learning paradigm in which the agent induces a logic program H to predict state transitions and iteratively refines it based on prediction failures.

2.1 PROBLEM DEFINITION

The learning task is to induce a hypothesis H that rationalizes transitions in a sequence $S = \langle S_0, S_1, \ldots, S_n \rangle$ within a deterministic, fully observable environment. A state S_t is a set of ground atoms. The goal is to learn a transition function $T : S \times A \rightarrow S'$ as a logic program $H = \langle Abs, Dyn, Con \rangle$. Here, Abs is a set of definite clauses defining invented predicates, enabling compositional representation of spatial or semantic relations of the form $p_i \leftarrow Body$. Dyn is a set of transition rules governing the addition of atoms, taking the form $add(P) \leftarrow Body$. Con is a set of rules governing the removal of atoms, taking the form $del(P) \leftarrow Body$. $Body$ is a conjunction of literals drawn from the background knowledge $\mathcal{B}$, the current state S_t, and the set of invented predicates defined in Abs.

The transition logic is defined by:

$$S_{t+1} = (S_t \setminus \mathcal{D}) \cup \mathcal{A}$$

where $\mathcal{A} = \{h | H_{Dyn} \cup S_t \cup \mathcal{B} \models h\}$ and $\mathcal{D} = \{h | H_{Con} \cup S_t \cup \mathcal{B} \models h\}$, with $\mathcal{B}$ representing background knowledge (static predicates). We assume **lifted dynamics**, meaning rules depend on object relations rather than identities. The search space is the power set of the logic language $\mathcal{L}$. To manage the super-exponential complexity online, we use a **Predict-Verify-Refine** cycle rather than batch learning. We maintain a current hypothesis H_t; when H_t fails to predict S_{t+1}, we trigger localized MIL search to repair the theory via generalization (metarule-guided construction) or specialization (clause pruning), see Algorithm 2 in Appendix C.

2.2 METARULE-GUIDED HYPOTHESIS CONSTRUCTION AND REFINEMENT

To overcome the combinatorial explosion typical of inductive synthesis, we restrict the search space using rule templates (Muggleton et al., 2015). Rather than searching the space of all Horn clauses, we search the space of proofs generated by a set of second-order templates called **metarules** $\mathcal{M}$.

A metarule $M \in \mathcal{M}$ is a second-order clause defining a valid syntactic structure for a rule. For example, the Chain metarule allows the agent to discover transitive relationships, and the absorption metarule to restrict a relation with a property:

$$M_{chain} : P(X, Y) \leftarrow Q(X, Z), R(Z, Y)$$
$$M_{absorption} : P(X, Y) \leftarrow Q(X, Y), R(Y)$$

where $\{P, Q, R\}$ are binded to predicates and $\{X, Y\}$ to first-order variables. While users select metarules, these typically have a general structure and can be reused across domains. Additionally, the system can compensate for a missing metarule by composing the available metarules through predicate invention.

Top Program Construction. Building on the Louise system (Patsantzis & Muggleton, 2019), we avoid searching for a single hypothesis directly. Instead, we construct the **Top Program** $\top$, the most general logic program entailing observed positive examples (E^+) within constraints:

$$\top = \{M\theta \mid M \in \mathcal{M}, \exists(E_i^+ \in E^+) : M\theta \models E_i^+\} \tag{1}$$

where θ is a choice of grounding. In our online setting, $\top$ represents the set of all plausible explanations. Learning reduces to generalizing $\top$ to explain new changes and specializing $\top$ by pruning falsified clauses (see Figure 3 in Appendix B).

Recursive Abduction and Predicate Invention. Hypotheses are constructed via abductive instantiation. Given an unexplained observation O, the system identifies a metarule M and a substitution θ such that $head(M)\theta = O$. Body literals are resolved against $\mathcal{B} \cup S_t$ or further abduced. These unexplained literals become targets for further abduction using abstraction metarules $\mathcal{M}_{abs}$ (see Algorithm 1 in Appendix C). This process builds a hierarchical derivation chain in which higher-level predicates are defined in terms of

lower-level ones, continuing recursively until the chain is grounded in $\mathcal{B}$ or the maximum depth D_{max} is reached. To bound complexity, we impose a type system $\mathcal{T}$, considering only instantiations that respect predicate typing in $\mathcal{B}$. The user manually assigns a type to primary predicates; invented predicates inherit the type from the predicates in their body.

Error Signals and Refinement. For any state transition $(S_t \to S_{t+1})$, the learner derives error signals via set difference operations on the observed states. We distinguish between two types of prediction errors that drive the lattice search, see Figure 3 in Appendix B:

- **False Negatives (FN):** These represent observed changes in the environment that the current hypothesis H failed to predict (i.e., $S_{t+1} \setminus S_{predicted}$). These examples trigger the **generalization** of $\top$ by invoking the abductive engine to generate new metarule instantiations that entail the missing atoms, adding them to $\top$ and H.
- **False Positives (FP):** These represent changes predicted by H that did not occur in reality (i.e., $S_{predicted} \setminus S_{t+1}$). When the model hallucinates an effect, the system identifies the specific clauses in H responsible for the hallucination and **specialises** $\top$ by pruning them.

Explicitly constructing the set of all non-changes in a large environment is intractable. To mitigate it without intractable global checks, we employ an Inertia Assumption (Fikes & Nilsson, 1971): the state is assumed invariant unless a specific rule in H predicts a change. This reduces the verification step to checking only predicted effects ($S_{predicted} \setminus S_{t+1}$).

2.3 Complexity Analysis

A defining characteristic of our framework is its scale invariance: the computational cost of hypothesis generation depends on the complexity of the underlying logic $\mathcal{D}_{max}$ rather than on the size of the state space. By operating on lifted predicates with intensional background knowledge, our system evaluates rules such as $adjacent/3$ via procedural attachment rather than by scanning a state matrix that grows exponentially with grid size.

While the search space for logic programs is theoretically infinite, our approach bounds the search through three structural constraints: Fixed Metarules, a Strong Type System, and Canonicalization. The worst-case complexity of expanding the hypothesis space for a single generalization step can be stated as:

$$O(|\mathcal{M}| \cdot |\mathcal{P}_{typed}|^k \cdot D_{max}) \tag{2}$$

Here, $|\mathcal{M}|$ is the finite set of second-order metarules. $|\mathcal{P}_{typed}|$ is the effective branching factor, constrained by the type system. k is the arity of the metarules (typically $k = 2$).

Our algorithm employs Top Program Construction with memorization, which effectively converts the search from a Tree Traversal into a **Directed Acyclic Graph Construction**. We maintain a **Global Predicate Registry** (Ω) of all discovered predicate signatures. When the abductive engine generates a new rule body (e.g., $P \leftarrow Q, R$), it computes a canonical hash of the body literals. If a semantically equivalent predicate already exists in Ω, the system reuses it rather than inventing a duplicate. We cache the results of abductive queries. If the system encounters a subgoal that has already been solved within the current "budget" of depth, it retrieves the solution in $O(1)$ time. By collapsing redundant branches and preventing the re-derivation of known concepts, the total work becomes the *sum* of the work required to construct each layer of the hierarchy, rather than the product.

3 Experiments

We evaluate our framework against three core desiderata for concept learning: the ability to perform continuous model repair; sample efficiency relative to neural baselines; and the robustness of the learned abstractions

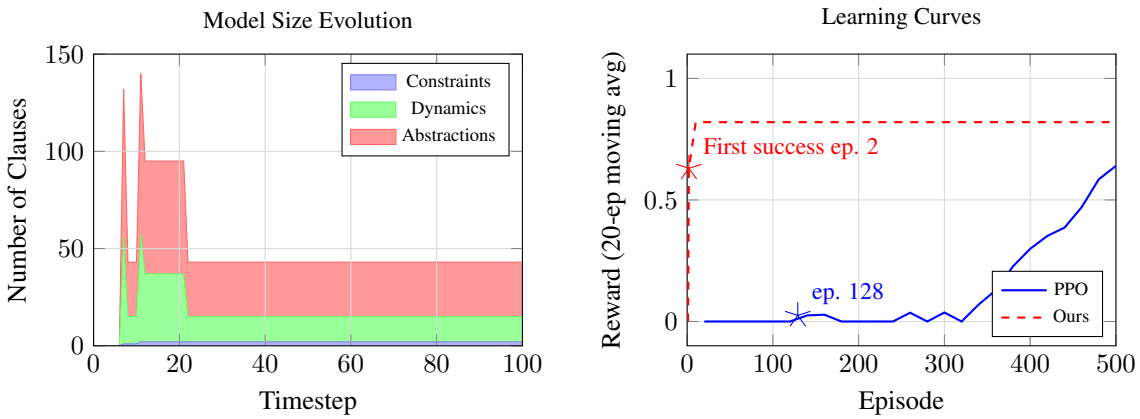

Figure 2: Our system vs PPO on the 10×10 grid version of the MiniHack 'Lava Crossing' task. (a) Our system converges to 43 clauses (28 abstractions, 13 dynamics, 2 constraints) by step 23. (b) Our system reaches the goal at episode 2 since then it is able to consistently navigate the environment to it relying on it learn model; PPO requires 128 episodes for first success and it is not until episode 300 that it starts to converge.

across varying domain sizes and semantic alignment. For these experiments, we equipped the agent with the ability to plan using its learnt model and take actions accordingly.

3.1 Online Model Repair and Convergence

To validate the Predict-Verify-Refine cycle, we track the evolution of the symbolic hypothesis H during a single run in the MiniHack environment. Figure 2 visualizes the internal model evolution. We monitor Model Size $|Dyn \cup Cons \cup Abs|$. Spikes in model size correspond to generalization events, where unexpected observations FN trigger the invention of new candidate rules. Subsequent sharp drops correspond to specialization, where the agent prunes rules that lead to FP. By time step 20, the model stabilizes into a set of plausible rules that have not been disproven. Prediction error drops to zero, demonstrating that the system successfully disentangles valid causal mechanisms from transient noise.

3.2 Sample Efficiency and Scale Invariance

We compare the sample efficiency of our Online MIL agent with that of a standard PPO baseline initialized with pre-trained CNN weights. As shown in Figure 2, the symbolic agent solves the environment in Episode 2 (one-shot learning after a single failure), whereas PPO requires approx. 128 episodes.

Although a logic-based learner is expected to outperform a gradient-based learner on grid worlds, this result highlights the representational efficiency of lifted concepts. Unlike neural weights, which approximate the state manifold, our agent immediately lifts observations into a relational space. Consequently, the "danger" concept learned on a 10×10 grid is structurally identical to one on a 100×100 grid. We confirmed that the model learned in the small environment generalizes zero-shot to the larger grid, demonstrating that the learned concepts are invariant to the state-space scale.

3.3 Semantic alignment and Interpretability

A key advantage of our approach over neural world models is the immediate interpretability of the latent space. The system invents predicates (p_n) to compress the state space. Figure 1 illustrates the semantic alignment between the automatically invented predicates and human concepts for the 'Lava Crossing' task. The predicate p_2, for example, is reused to define the death and movement transition rules, demonstrating that the system creates hierarchical abstractions rather than flat correlations.

4 Related Work

Symbolic and Neurosymbolic World Models. Learning symbolic dynamics is traditionally framed as batch constraint satisfaction. While the Apperception Engine (Evans et al., 2021b) offers robust unification, however it is framed as a constraint satisfaction problem, which makes it computationally prohibitive for real-time agents in big environments. Neurosymbolic alternatives (Athalye et al., 2025; Silver et al., 2021; 2023; Piriyakulkij et al., 2025) improve speed but often rely on expensive pre-trained LLMs, which poses a challenge for systems that work offline or are limited on compute. Our framework bridges this gap by framing the problem as local, iterative refinement, updating beliefs only when immediate predictions are falsified. Our work system exclusively relies on pure logic programming rather than gradient-based search. This ensures the automated acquisition of the transition model results in a transparent, verifiable First-Order Logic theory at every step.

Non-Monotonic Learning. Modern Inductive learners such as FastLAS (Law et al., 2020) and Popper (Cropper & Morel, 2021; Cropper et al., 2020) typically operate in a batch setting, requiring full traces or labelled examples. We extend the Lousie Patsantzis & Muggleton (2021) system by enabling incremental repair of transition rules during exploration via a self-supervision framework and by maintaining a persistent predicate registry, we enable the incremental repair of transition rules during exploration. Recently, PyGol (Varghese et al., 2025) outperformed Deep RL in sample efficiency, taking a similar approach to ours. However, they focused on learning policies and are not able to perform predicate invention.

Abstraction and Program Synthesis. Systems like DreamCoder (Ellis et al., 2021) compress solution spaces via offline "wake-sleep" cycles, creating a lag between observation and adaptation. In contrast, our system performs abstraction online. By inventing reusable predicates *during* the decision loop, we merge the wake and sleep phases, making hierarchical concepts immediately available for planning.

5 Conclusion

This work introduces a novel online learning framework that leverages the representational power of invented predicates to generate compact, interpretable transition functions online. We show that this system is effective at learning models of classic RL environments. Additionally, we show that, when combined with an off-the-shelf planning algorithm, the agent efficiently solves the MiniHack lava environment, surpassing established RL benchmarks. These results suggest that this approach opens avenues for research into methods to address longstanding RL challenges in which agents must interact with diverse objects and reason about their behaviour. A promising research direction is to enforce active model refinement, in which the agent sets the rule body as a goal and tests whether the rule is valid. Other promising avenues of research include moving towards probabilistic logic and the use of neural predicates (Manhaeve et al., 2021; Evans et al., 2021a; Smet et al., 2025). This would allow a transition from hand-engineered domain encoding to a more autonomous learning framework in which primary predicates are also learnt. Combining this method with a neural policy could enable the agent to plan and react efficiently and robustly.

REFERENCES

Ashay Athalye, Nishanth Kumar, Tom Silver, Yichao Liang, Jiuguang Wang, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. From Pixels to Predicates: Learning Symbolic World Models via Pretrained Vision-Language Models, June 2025. URL http://arxiv.org/abs/2501.00296. arXiv:2501.00296 [cs].

Andrew Cropper and Rolf Morel. Learning programs by learning from failures. *Machine Learning*, 110(4): 801–856, April 2021. ISSN 0885-6125, 1573-0565. doi: 10.1007/s10994-020-05934-z. URL https://link.springer.com/10.1007/s10994-020-05934-z.

Andrew Cropper and Sophie Tourret. Logical reduction of metarules. *Machine Learning*, 109(7):1323–1369, July 2020. ISSN 0885-6125, 1573-0565. doi: 10.1007/s10994-019-05834-x. URL http://link.springer.com/10.1007/s10994-019-05834-x.

Andrew Cropper, Richard Evans, and Mark Law. Inductive general game playing. *Machine Learning*, 109(7):1393–1434, July 2020. ISSN 1573-0565. doi: 10.1007/s10994-019-05843-w. URL https://doi.org/10.1007/s10994-019-05843-w.

Kevin Ellis, Catherine Wong, Maxwell Nye, Mathias Sablé-Meyer, Lucas Morales, Luke Hewitt, Luc Cary, Armando Solar-Lezama, and Joshua B. Tenenbaum. DreamCoder: bootstrapping inductive program synthesis with wake-sleep library learning. In *Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, pp. 835–850, Virtual Canada, June 2021. ACM. ISBN 978-1-4503-8391-2. doi: 10.1145/3453483.3454080. URL https://dl.acm.org/doi/10.1145/3453483.3454080.

Richard Evans, Matko Bošnjak, Lars Buesing, Kevin Ellis, David Pfau, Pushmeet Kohli, and Marek Sergot. Making sense of raw input. *Artificial Intelligence*, 299:103521, October 2021a. ISSN 0004-3702. doi: 10.1016/j.artint.2021.103521. URL https://www.sciencedirect.com/science/article/pii/S0004370221000722.

Richard Evans, José Hernández-Orallo, Johannes Welbl, Pushmeet Kohli, and Marek Sergot. Making sense of sensory input. *Artificial Intelligence*, 293:103438, April 2021b. ISSN 0004-3702. doi: 10.1016/j.artint.2020.103438. URL https://www.sciencedirect.com/science/article/pii/S0004370220301855.

Richard E. Fikes and Nils J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3):189–208, December 1971. ISSN 0004-3702. doi: 10.1016/0004-3702(71)90010-5. URL https://www.sciencedirect.com/science/article/pii/0004370271900105.

Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning: Theory and Practice*. Elsevier, May 2004. ISBN 978-0-08-049051-9. Google-Books-ID: uYnpze57MSgC.

Mark Law, Alessandra Russo, Elisa Bertino, Krysia Broda, and Jorge Lobo. FastLAS: Scalable Inductive Logic Programming Incorporating Domain-Specific Optimisation Criteria. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(03):2877–2885, April 2020. ISSN 2374-3468. doi: 10.1609/aaai.v34i03.5678. URL https://ojs.aaai.org/index.php/AAAI/article/view/5678.

Robin Manhaeve, Sebastijan Dumančić, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Neural probabilistic logic programming in DeepProbLog. *Artificial Intelligence*, 298:103504, September 2021. ISSN 0004-3702. doi: 10.1016/j.artint.2021.103504. URL https://www.sciencedirect.com/science/article/pii/S0004370221000552.

Thomas M. Moerland, Joost Broekens, Aske Plaat, and Catholijn M. Jonker. Model-based Reinforcement Learning: A Survey, March 2022. URL http://arxiv.org/abs/2006.16712. arXiv:2006.16712 [cs].

Stephen H. Muggleton, Dianhuan Lin, and Alireza Tamaddoni-Nezhad. Meta-interpretive learning of higher-order dyadic datalog: predicate invention revisited. *Machine Learning*, 100(1):49–73, July 2015. ISSN 0885-6125, 1573-0565. doi: 10.1007/s10994-014-5471-y. URL http://link.springer.com/10.1007/s10994-014-5471-y.

S. Patsantzis and S. H. Muggleton. Top program construction and reduction for polynomial time Meta-Interpretive learning. *Machine Learning*, 110(4):755–778, April 2021. ISSN 1573-0565. doi: 10.1007/s10994-020-05945-w. URL https://doi.org/10.1007/s10994-020-05945-w.

Stassa Patsantzis and Stephen H. Muggleton. Louise: A Meta-Interpretive Learner for Efficient Multi-clause Learning of Large Programs. Plovdiv, Bulgaria, September 2019. York. URL https://eprints.whiterose.ac.uk/id/eprint/157823/. Num Pages: 4.

Wasu Top Piriyakulkij, Yichao Liang, Hao Tang, Adrian Weller, Marta Kryven, and Kevin Ellis. PoE-World: Compositional World Modeling with Products of Programmatic Experts, May 2025. URL http://arxiv.org/abs/2505.10819. arXiv:2505.10819 [cs].

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, August 2017. URL http://arxiv.org/abs/1707.06347. arXiv:1707.06347.

Tom Silver, Rohan Chitnis, Joshua Tenenbaum, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Learning Symbolic Operators for Task and Motion Planning. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3182–3189, September 2021. doi: 10.1109/IROS51168.2021.9635941. URL https://ieeexplore.ieee.org/document/9635941/?arnumber=9635941.

Tom Silver, Ashay Athalye, Joshua B. Tenenbaum, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Learning Neuro-Symbolic Skills for Bilevel Planning. In *Proceedings of The 6th Conference on Robot Learning*, pp. 701–714. PMLR, March 2023. URL https://proceedings.mlr.press/v205/silver23a.html.

Lennert De Smet, Gabriele Venturato, Luc De Raedt, and Giuseppe Marra. Relational Neurosymbolic Markov Models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(15):16181–16189, April 2025. ISSN 2374-3468. doi: 10.1609/aaai.v39i15.33777. URL https://ojs.aaai.org/index.php/AAAI/article/view/33777.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, October 2018. ISBN 978-0-262-03924-6.

Dany Varghese, Daniel Cyrus, Stassa Patsantzis, James Trewern, Alfie Anthony Treloar, Alan Hunter, and Alireza Tamaddoni-Nezhad. One-Shot Learning of Autonomous Behaviour: A Meta Inverse Entailment Approach. In *Learning and Reasoning: 4th International Joint Conference on Learning and Reasoning, IJCLR 2024, and 33rd International Conference on Inductive Logic Programming, ILP 2024, Nanjing, China, September 20–22, 2024, Proceedings*, pp. 48–65, Berlin, Heidelberg, November 2025. Springer-Verlag. ISBN 978-3-032-09086-7. doi: 10.1007/978-3-032-09087-4_4. URL https://doi.org/10.1007/978-3-032-09087-4_4.

Zhongwei Yu, Jingqing Ruan, and Dengpeng Xing. Explainable Reinforcement Learning via a Causal World Model. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, pp. 4540–4548, Macau, SAR China, August 2023. International Joint Conferences on Artificial Intelligence Organization. doi: 10.24963/ijcai.2023/505. URL https://www.ijcai.org/proceedings/2023/505.

Hongming Zhang and Tianyang Yu. AlphaZero. In Hao Dong, Zihan Ding, and Shanghang Zhang (eds.), *Deep Reinforcement Learning: Fundamentals, Research and Applications*, pp. 391–415. Springer, Singapore, 2020. ISBN 978-981-15-4095-0. doi: 10.1007/978-981-15-4095-0_15. URL https://doi.org/10.1007/978-981-15-4095-0_15.

A General primer on logic programming

This section provides brief, general definitions of standard First-Order Logic and Inductive Logic Programming terminology to assist readers less familiar with symbolic AI.

- **Atoms and Predicates:** In logic programming, knowledge is represented relationally. A *predicate* defines a property of an object or a relationship between multiple objects (e.g., $color(X)$ or $adjacent(X, Y)$). An *atom* is the application of a predicate to specific arguments. A *ground atom* is one where all arguments are specific constants rather than variables (e.g., $adjacent(roomA, roomB)$).
- **Definite Clauses and Horn Clauses:** A logic program is primarily composed of rules. A *definite clause* (often referred to in this context as a Horn clause) is a strict logical implication written in the form $Head \leftarrow Body$. The $Head$ contains a single positive conclusion, and the $Body$ is a conjunction of conditions. If all conditions in the body are satisfied, the head is logically asserted to be true.
- **Propositional vs. Lifted Representation:** A *propositional* representation evaluates specific, grounded instances (e.g., a specific agent at a specific coordinate), which often leads to a combinatorial explosion as environments grow. A *lifted* representation uses variables (e.g., $agent(X)$ and $location(Y)$) to express generalized, structural rules that apply universally to any objects in the domain, independent of their specific identities or the overall size of the state space.

A.1 Modes of Logical Inference

Logic programming formally distinguishes between the semantic truth of statements and the procedural algorithms used to reason about them:

- **Entailment ($\models$):** A semantic relationship denoting logical consequence. A set of rules and facts *entails* a conclusion if, in every possible scenario where the premises are true, the conclusion is also true. In ILP, we evaluate whether a learned hypothesis successfully entails the observed data.
- **Deduction ($\vdash$):** The procedural process of applying formal inference rules to derive conclusions from known premises. It is the "forward pass" of a symbolic system. Given a set of established rules and a current set of facts, deduction mechanically computes the guaranteed consequences (e.g., predicting the exact next state of an environment).
- **Induction:** The process of reasoning from specific examples to general rules. In ILP, induction involves synthesizing a generalized hypothesis (a logic program) from a set of observed data points (positive and negative examples) and static background knowledge. The goal is to discover the underlying rules that govern the observations.

- **Abduction:** The process of reasoning backward from effects to plausible causes. Given an observation and a set of rules, abduction infers the missing facts, assumptions, or intermediate steps that would make the observation true. It is the formulation of a specific hypothesis to explain a novel data point.

B Theoretical Guarantees

This section analyzes the completeness of the search procedure and compares the computational complexity of this lifted inference approach with that of propositional methods.

B.1 Lattice-Based Refinement

The classification metrics map directly to operators on the subset lattice, see 3:

- **Specialization (Handling FP):** When the hypothesis is overly general (predicting events that do not occur), the system prunes the incorrect clauses from the Top Program. This reduces the entailment set, effectively moving down the lattice.
- **generalization (Handling FN):** When the hypothesis is overly specific (failing to predict observed events), the system invokes the metarule engine to induce (add) new clauses. This expands the entailment set to cover the unexplained observations, effectively moving up the lattice.

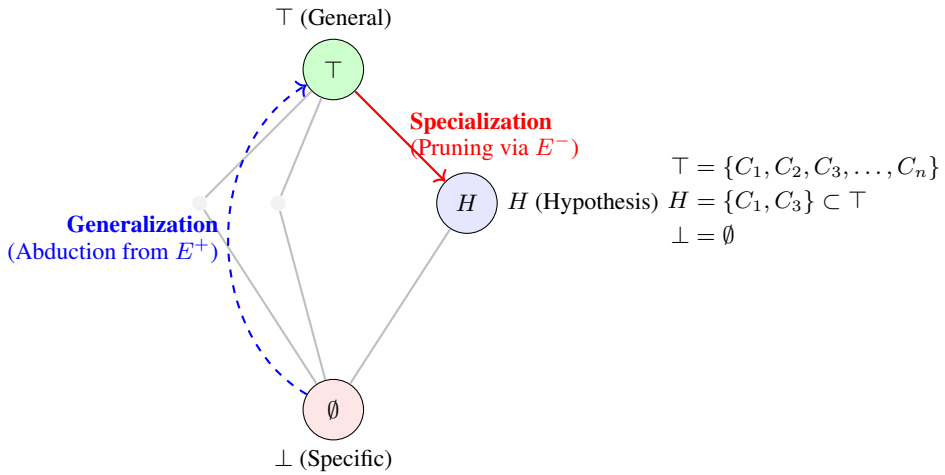

Figure 3: Lattice traversal dynamics. Prediction errors trigger **Generalization** (moving to $\top$) or **Specialization** (pruning to H).

The search for a hypothesis H corresponds to a traversal of the refinement graph defined by the metarules $\mathcal{M}$ and background knowledge $\mathcal{B}$. We define the expressible hypothesis space $\mathcal{H}_{\mathcal{M},d}$ as the set of all logic programs derivable by composing metarules from $\mathcal{M}$ up to a maximum derivation depth d.

B.2 Completeness and Expressivity

Proposition (Completeness): *If there exists a target hypothesis $H^* \in \mathcal{H}_{\mathcal{M},d}$ that correctly entails the observed state transitions, the algorithm is guaranteed to find it.*

Proof: The proof follows the properties of Top Program Construction established in (Patsantzis & Muggleton, 2021), adapted to our depth-bounded search:

1. **Completeness of generalization (Moving Up):** The `AbduceChain` procedure exhaustively generates all metarule instantiations that entail the current observation O within depth d. If the target hypothesis H^* exists in $\mathcal{H}_{\mathcal{M},d}$, then every clause $C \in H^*$ is technically derivable. Consequently, during the generalization phase, all clauses constituting H^* are added to the initial Top Program $\top_{init}$. Thus, $H^* \subseteq \top_{init}$.
2. **Soundness of specialization (Moving Down):** The refinement phase iterates through $\top_{init}$ and removes any clause C such that $C \wedge \mathcal{B} \models E^-$. By definition, the target hypothesis H^* is consistent with the environment, meaning no clause in H^* entails a negative example. Therefore, the pruning operator never removes a clause belonging to H^*.
3. **Convergence:** Since H^* is captured during generalization and preserved during Specialization, the final hypothesis $\top_{final}$ satisfies $H^* \subseteq \top_{final}$. Because H^* entails the positive examples, $\top_{final}$ necessarily entails them as well.

□

C ALGORITHMS

Algorithm 1: Metarule-Guided Abduction with Predicate Reuse

Function *MetaruleInduction*($Literal\ L, Depth\ k$):
if $k > D_{max}$ **then**
  return failure ; `// Terminate at max abstraction depth`
$\mathcal{H} \leftarrow \emptyset, \mathcal{T} \leftarrow \emptyset$;
foreach *metarule* $m \in \mathcal{M}$ *(*$H \leftarrow B_1, \ldots, B_n$*)* **do**
  Apply substitution θ such that $H\theta = L$;
  $Body \leftarrow [\,]$;
  foreach *literal* $lit \in [B_1\theta, \ldots, B_n\theta]$ **do**
    if lit *is primitive OR entails* $\mathcal{B}$ **then**
      $Body$.append(lit);
    else
      `// Recursive abduction for predicate invention`
      $(P, Rule_p, Type_p) \leftarrow$ ***MetaruleInduction***$(lit, k+1)$;
      $Body$.append(P) ; `// Invented predicate P added to the body`
      $\mathcal{H} \leftarrow \mathcal{H} \cup Rule_p$; `// Rule that explains P is stored`
      $\mathcal{T} \leftarrow \mathcal{T} \cup Type_p$; `// Type signature of P is stored`
  if $k == 0$ **then**
    $\mathcal{H} \leftarrow \mathcal{H} \cup \{H \leftarrow Body\}$;
    $P_{ret} \leftarrow L$;
  else
    $P_{ret} \leftarrow$ ***ReuseOrRegister***$(Body, \mathcal{H})$;
return $(P_{ret}, \mathcal{H}, \mathcal{T})$;

Algorithm 2: Self-Supervised Online MIL Loop

Input : Stream of states $S_0, S_1, \ldots$, Metarules $\mathcal{M}$, Background Knowledge $\mathcal{B}$
Output: Evolving Hypothesis $H = \langle Abs, Dyn, Con \rangle$

$H \leftarrow \langle \emptyset, \emptyset, \emptyset \rangle$;
$Context \leftarrow S_0$;
for $t \leftarrow 1$ **to** ∞ **do**
  Receive S_t;
  // Derive Examples via Set Difference
  $E^+ \leftarrow S_t \setminus S_{t-1}$; // Observed Additions
  $E^- \leftarrow S_{t-1} \setminus S_t$; // Observed Removals
  // Phase 1: Predict (using compiled H)
  $P_{add} \leftarrow$ Predict$(H, Context)$;
  $P_{rem} \leftarrow$ DeriveRemovals(P_{add}, H);
  // Phase 2: Verify and Calculate Error
  $FP_{add} \leftarrow P_{add} \setminus E^+$;
  $FN_{add} \leftarrow E^+ \setminus P_{add}$;
  $FP_{rem} \leftarrow P_{rem} \setminus E^-$;
  $FN_{rem} \leftarrow E^- \setminus P_{rem}$;
  // Phase 3: Refine
  if $FP_{add} \neq \emptyset$ **then**
    $Dyn \leftarrow$ Prune(Dyn, FP_{add});
  if $FP_{rem} \neq \emptyset$ **then**
    $Con \leftarrow$ Prune(Con, FP_{rem});
  if $FN_{add} \neq \emptyset$ **then**
    $\langle \Delta Dyn, \Delta Abs \rangle \leftarrow$ MetaruleInduction$(FN_{add}, Context, \mathcal{M}, \mathcal{B})$;
    $Dyn \leftarrow Dyn \cup \Delta Dyn$; $Abs \leftarrow Abs \cup \Delta Abs$;
  if $FN_{rem} \neq \emptyset$ **then**
    $\langle \Delta Con, \Delta Abs' \rangle \leftarrow$ MetaruleInduction$(FN_{rem}, Context, \mathcal{M}, \mathcal{B})$;
    $Con \leftarrow Con \cup \Delta Con$; $Abs \leftarrow Abs \cup \Delta Abs'$;
  // Maintenance
  $H \leftarrow$ CompressAndGC(H);
  $Context \leftarrow S_t$;