ATLAS: FROM WORLD LAWS TO SKILLS IN LLM-GUIDED HIERARCHICAL REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-horizon reinforcement learning (RL) often relies on structured skill specifications to overcome sparse rewards and long dependency chains. Existing LLM-Hierarchical Reinforcement Learning (LLM-HRL) methods typically ask language models to both infer environment dynamics and derive skill preconditions, effects, and dependencies, making specification errors difficult to avoid. We propose ATLAS (Acquiring and Testing Laws for Actionable Skills), a *world-law-driven* LLM-HRL framework in which explicit world laws connect language-model reasoning with skill construction and learning. ATLAS represents environment dynamics as a BC+ world model and deterministically derives executable skill specifications from accepted laws for hierarchical RL. Test-driven development provides validation constraints on law construction and revision through symbolic unit tests and environment-level tests. To acquire and revise laws from experience, ATLAS uses successful and failed trajectories together with counterfactual experiments to check candidate world laws. Accepted revisions propagate to skill specifications and training, translating evolving environment knowledge into continually updated execution capabilities. With detailed Craftax-Classic materials, the generated laws support operator grounding and achievement completion across resource and tool dependencies; in an unfamiliar Mars world, interaction evidence supports law acquisition without supplied environment-specific rules.

1 INTRODUCTION

Long-horizon decision-making tasks often involve extended chains of prerequisite skills and sparse rewards, making it difficult for reinforcement learning (RL) to acquire the required behaviors through end-to-end exploration. Recent approaches use large language models (LLMs) to define what to learn and hierarchical reinforcement learning (HRL) to acquire and compose reusable skills. LEAGUE++ combines LLM-guided decomposition with symbolic planning and skill learning (Li et al., 2024); L2S generates skill rewards and termination conditions and adapts their numerical parameters through downstream learning (Mao et al., 2025). CODE-SHARP continually expands and refines a skill archive represented as hierarchical reward programs (Bornemann et al., 2026), while SCALAR uses execution trajectories to correct symbolic skill specifications (Zabounidis et al., 2026). Together, these approaches advance the specification-driven LLM-HRL paradigm along complementary directions: reward design, skill discovery, and specification refinement.

However, the effectiveness of this paradigm depends critically on the reliability of the generated specifications. Evidence from symbolic world-model generation highlights this challenge. Text2World reports that, without error correction, even its best-performing evaluated model achieves F1 scores below 0.60 for both preconditions and effects Hu et al. (2025). LLM+AL finds that only approximately one fifth of one-shot action-language generations pass syntax checks Ishay & Lee (2025). Related failures also occur within LLM-HRL systems: CODE-SHARP reports incorrect skill specifications despite access to the simulator's complete source code Bornemann et al. (2026). These findings indicate that access to rich environmental information does not, by itself, ensure its reliable conversion into formal models or skill specifications.

We argue that this unreliability stems from the coupling of world-law understanding and skill-specification derivation. Existing systems typically require LLMs to solve two problems in a single

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.

generation step: determining which conditions and relations hold in the environment to correctly understand its governing laws, and organizing this knowledge into precise skill specifications that satisfy strict structural constraints. Language models are well suited to identifying objects, conditions, and action-outcome relations from descriptions and trajectories. However, precisely combining these laws to derive skill preconditions and effects under numerous complex constraints and dependencies is not among their strengths, leading to reduced specification accuracy.

We propose ATLAS (Acquiring and Testing Laws for Actionable Skills), a *world-law-driven* LLM-HRL framework in which explicit world laws connect language-model reasoning with skill construction and learning. **First**, we introduce an explicit world-knowledge layer expressed in BC+, a declarative action language for describing state constraints and how actions and environmental dynamics change the world. This layer separates LLM-based world-law modeling from symbolic specification derivation. **Second**, because LLM-proposed laws may contain errors, we adopt test-driven development's idea: a testing agent constructs BC+ unit tests and environment-level end-to-end tests. These tests constrain subsequent law generation and revision. **Third**, a deterministic symbolic module derives skill preconditions, effects, and dependencies from the accepted world model, replacing LLM-based specification generation with explicit computation. **Finally**, ATLAS extracts candidate world laws from execution trajectories through simplification and semantic analysis, then checks them with counterfactual experiments. Accepted laws and revisions propagate to affected specifications and skill training, translating evolving environment knowledge into continually updated execution capabilities.

We evaluate ATLAS through three complementary studies. The first examines how much test-guided revision improves the accuracy of generated world laws on predefined behavioral tests. The second evaluates whether law-derived skill specifications support policy learning and execution across long achievement chains. The third investigates whether interaction experience supports law acquisition, validation, and operator learning in an unfamiliar world.

Our main contributions are threefold:

- We propose ATLAS, a world-law-centered representation for LLM-HRL, and a deterministic method for deriving HRL skill specifications and dependencies from accepted laws before grounding them as learned policies.
- Within ATLAS, we develop a method for extracting and revising world laws from execution experience, using trajectory evidence and counterfactual validation to support operation in environments without supplied rules.
- We conduct complementary experiments demonstrating improved law-test agreement, skill execution across long prerequisite chains, and interaction-based law acquisition with operator-level outcomes in an unfamiliar world.

2 Related Work

LLM-guided hierarchical reinforcement learning. Options provide a framework for learning and composing temporally extended behaviors (Sutton et al., 1999). LLM-guided HRL uses language knowledge to define what these behaviors should achieve. LEAGUE++ combines LLM-guided task decomposition, symbolic planning, and RL to accumulate reusable skills (Li et al., 2024). L2S generates reward functions and termination conditions, trains parameter-conditioned policies, and selects their parameters during subsequent skill learning (Mao et al., 2025). These methods reduce manual specification of skill objectives. ATLAS additionally represents the underlying environment laws explicitly and computes skill preconditions, effects, and dependencies from the accepted theory.

Skill discovery and specification refinement. Beyond defining individual skills, systems must determine which skills to acquire and how to revise them. SPOTTER and SORL discover symbolic operators or options through interaction (Sarathy et al., 2021; Jin et al., 2022). CODE-SHARP uses a foundation model to continually expand a skill archive with hierarchical reward programs and refine them through evaluation and mutation (Bornemann et al., 2026). SCALAR uses successful and failed trajectories to revise LLM-proposed preconditions and effects (Zabounidis et al., 2026). CODE-SHARP emphasizes open-ended expansion, while SCALAR emphasizes specification cor-

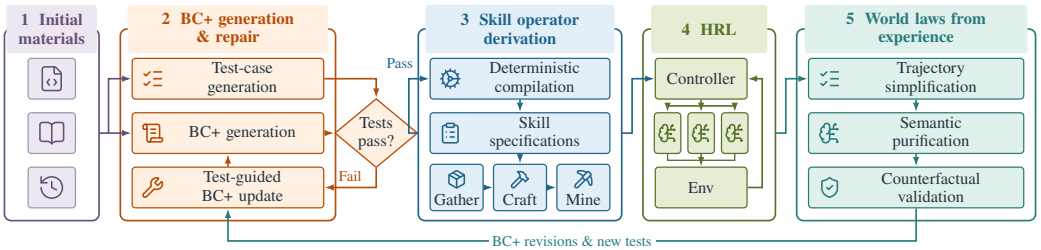


Figure 1: The ATLAS world-law-driven LLM-HRL pipeline.

rection; both support refinement. We place this feedback at the shared world-law layer, testing and revising environmental relations before propagating accepted changes to derived specifications.

Action languages and symbolic reinforcement learning. Action languages describe state constraints and transition dynamics for automated reasoning. BC and BC+ support causal laws, defaults, and recursive definitions under formal semantics (Lee et al., 2013; Babb & Lee, 2020). PEORL and SDRL connect symbolic action descriptions with hierarchical policy learning (Yang et al., 2018; Lyu et al., 2019). LLM+AL further translates natural-language descriptions into BC+ theories for solver-based reasoning (Ishay & Lee, 2025). Text2World evaluates LLM-generated symbolic world models and reveals persistent errors in preconditions and effects (Hu et al., 2025). ATLAS connects these ideas through a test-driven world-law layer: symbolic and environment-level tests constrain model construction, and deterministic synthesis derives specifications for HRL. Environment tests assess candidate laws against tested behaviors, while symbolic computation derives their consequences.

3 Preliminaries

A Markov decision process is defined by $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, r, \gamma \rangle$, where $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces, P is the transition kernel, r is the reward function, and $\gamma \in [0, 1)$ is the discount factor. Reinforcement learning seeks a policy π maximizing the expected discounted return $\mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$. In long-horizon tasks with sparse rewards, discovering extended prerequisite chains through primitive-action exploration is difficult, motivating hierarchical decomposition into reusable skills.

Action Languages. Action languages provide formal representations of how actions change the world. In language $\mathcal{A}$, causal laws are tied to specific actions: a **causes** F **if** G states that action a produces effect F when condition G holds (Gelfond & Lifschitz, 1993). Prior specification-driven LLM-HRL methods commonly follow this action-centered representation pattern, describing skills through their preconditions and effects (Zabounidis et al., 2026; Bornemann et al., 2026). BC+ can additionally express laws independent of any particular action: **caused** F **if** G states that condition G causes F to hold within the same state, regardless of which action establishes G (Babb & Lee, 2015).

Options and Specification-Guided HRL. An option is a temporally extended action $o = \langle I_o, \pi_o, \beta_o \rangle$, comprising an initiation set, an internal policy, and a termination rule (Sutton et al., 1999). A symbolic skill specification describes prerequisites and intended effects: the prerequisites define admissible starting states, while the effects provide success criteria used to construct rewards and termination rules. Termination may additionally account for failure or time limits. In specification-driven LLM-HRL, an LLM proposes these descriptions from environmental information, RL learns the corresponding policies, and a high-level controller composes skills by matching effects to prerequisites (Zabounidis et al., 2026; Bornemann et al., 2026). ATLAS retains this learning interface while deriving the specifications from a separately tested BC+ world model.

4 METHODOLOGY

Figure 1 summarizes ATLAS, from test-driven world-law construction to skill synthesis and hierarchical policy learning, with execution experience feeding back into law refinement.

4.1 WORLD-LAW MODEL

We maintain a *world-law model* $\mathcal{W}$: a collection of BC+ laws describing action conditions, effects, resource changes, and shared state constraints. These laws represent environment knowledge maintained independently of individual skill specifications, so that multiple skills can draw on the same model. For example, the model can encode the tool requirement and inventory change for mining iron, together with the conditions and resource costs of crafting a stone pickaxe. From these laws, skill synthesis derives the corresponding specifications and links the crafting effect to the mining prerequisite.

This shared world-law layer separates environment modeling from skill-specification derivation. The LLM proposes and revises laws from available environment materials and interaction evidence, but it may misunderstand how the environment behaves when formulating candidate BC+ laws. We therefore test and revise candidates before admitting them to the world model. A deterministic symbolic compiler then derives skill specifications from accepted laws within the supported language fragment. BC+ supports this division through a declarative representation of action effects and shared state relations, with formal semantics for symbolic reasoning (Babb & Lee, 2015). Environment tests provide empirical support for the laws, while symbolic synthesis computes their consequences for skill construction; neither step establishes that the accepted laws cover every environment behavior.

The model is developed through a testing and learning loop. During warmup, an initial candidate theory is generated and checked against independently constructed behavioral tests; only a candidate passing the validation procedure is accepted (Section 4.2). Accepted laws supply skill specifications and dependencies for RL training (Section 4.3). Trajectories from skill training and execution then inform experience-based world-law acquisition and revision, which proposes targeted changes (Section 4.4). Each revision passes through the same validation procedure before updating affected specifications and training, allowing environment knowledge and learned capabilities to develop together.

4.2 TEST-DRIVEN WORLD-LAW DEVELOPMENT

Before world laws enter skill synthesis, their behavior must be checked against the environment. Inspired by the *red-green-refactor* cycle of test-driven development, we separate test construction from law authoring and use test feedback to guide revisions. The two modules below provide a shared validation procedure for both initial generation and subsequent law updates.

The tests themselves require an independent source of expected outcomes. For the Craftax study, we first constructed 109 behavioral cases from the available source code and domain interface, without consulting candidate BC+ laws. Before using them to assess generated theories, we ran the cases in the environment and checked their expected outcomes against the observed transitions. This preliminary check supports the test expectations in the exercised situations; the resulting suite was fixed before law generation and repair. It does not establish that the suite covers every environment behavior.

Independent test construction. The testing agent receives the materials and domain interface, without candidate laws, and produces structured test specifications. A BC+ unit test queries a symbolic initial state and action sequence; an environment-level end-to-end test sets up the corresponding situation, executes actions, and checks observable outcomes. Here, end-to-end testing concerns whether a law holds in the environment. Separate symbolic and environment executors run these tests through a shared test interface. Tests cover effects, resource changes, and missing prerequisites.

Test-constrained generation and revision. The law author follows a BC+ authoring procedure to produce candidate theories or targeted revisions from the available evidence. Each candidate

enters the same validation pipeline: parsing and compilation checks precede BC+ unit tests and environment tests. Compilation errors or failed tests return feedback to the law author; empty suites, execution errors, and undecidable queries cannot count as passes. Within a fixed revision budget, only candidates passing all symbolic and environment tests are accepted. Failed revisions leave the previous accepted model intact.

Warmup. Warmup generates the initial world-law theory $\mathcal{W}_0$ from available materials and checks it against independently constructed behavioral tests before an accepted theory enters skill synthesis; RQ1 compares test pass rates before and after feedback-guided revision.

4.3 Deterministic Skill Synthesis and Grounding

Skill specification synthesis. Prior LLM-HRL work has established a useful way to ground long-horizon tasks: generate skill specifications, decompose the task into options, and train a policy for each option (Zabounidis et al., 2026). We retain this grounding structure. Given the accepted BC+ world theory $\mathcal{W}$, our symbolic compiler derives a set of skill specifications $\mathcal{S}$ and a dependency graph $\mathcal{G}$ without another LLM generation step:

$$(\mathcal{S}, \mathcal{G}) = \mathrm{Compile}(\mathcal{W}, g), \qquad \sigma_i = \langle P_i, E_i, C_i, L_i \rangle \in \mathcal{S}. \tag{1}$$

Here P_i specifies prerequisites, E_i the intended effects, C_i resource consumption, and L_i the supporting laws. Each node in $\mathcal{G}$ is a skill specification, and an edge $i \to j$ means that completing skill i can establish a prerequisite of skill j. For example, if mining iron requires a stone pickaxe, the graph links CraftStonePickaxe to MineIron, telling the controller to acquire the pickaxe before attempting to mine iron. The conversion consists of rule-based extraction and dependency matching; Appendix B gives the implementation and its scope. Its output is the set of skill specifications used for learning.

Policy grounding and climb. Each skill specification is grounded as a separately trained RL *operator* with its own checkpoint. Operators receive initial training in parallel; climb then trains deeper operators after the controller executes their prerequisites and saves states where the target can start. Improved prerequisite policies reopen downstream training, while low-success operators supply trajectories for world-law revision (Section 4.4). Appendix B specifies the controller, training thresholds, budgets, frontier-state mixture, and evaluation boundaries, and Appendix B.3 compares prior schedules.

4.4 Learning World Laws from Experience

Training produces successful and failed trajectories that the system can collect, but a trajectory is not itself a world law. Long episodes contain many irrelevant transitions, while high-dimensional observations contain variables that change together without all governing the outcome. Extracting a reliable law therefore raises three questions: which transitions are informative, which observed variables are relevant to the underlying mechanism, and whether the resulting candidate survives controlled tests in the environment. Within ATLAS, trajectory simplification, semantic purification, and counterfactual validation address these questions, respectively. Candidate revisions return to the shared testing procedure before they can update downstream skills.

Trajectory simplification. To locate informative transitions, an operator with evaluated success rate below 50% triggers collection of successful and failed trajectories after each climb round. This threshold initiates diagnosis. We extract complete transitions (x_t, a_t, x_{t+1}), remove duplicates, and exclude reset-crossing or conflicting records. Grouping transitions by action and target effect yields positive and negative instances and factor statistics for candidate conditions. This reduces episode-level histories to evidence relevant to the suspected law, while preserving local successes from unsuccessful episodes.

Semantic purification. To identify variables relevant to the law, an LLM uses these transitions and their factor statistics to group factors as mechanism-relevant, incidental, or unresolved and propose a candidate condition-effect relation. The proposal includes supporting transition identifiers, predicted effects and resource changes, and its applicable context. This step reduces the factors submitted to intervention a proposed removal must explicitly predict zero consumption so that it can be tested.

Counterfactual validation. When causal attribution is difficult, human investigators design multiple controlled comparisons to distinguish competing explanations. Our counterfactual validation follows this logic: for each candidate law, we compare a baseline with interventions that falsify one proposed prerequisite while preserving the others. Unrealized setups or unexecuted actions yield insufficient evidence; contradictory realized outcomes yield counterexamples and prompt candidate revision. Candidates passing interventions and independent repetitions become local BC+ edits. Their new symbolic and environment tests join the regression suite, and the revised model returns to compilation and both test suites before acceptance.

5 Experimental Setup

We evaluate ATLAS through three research questions. **RQ1** asks to what extent test-guided revision improves the accuracy of LLM-generated BC+ world laws on predefined behavioral tests. **RQ2** asks how well operators synthesized from the RQ1 laws can be grounded in policies and composed across Craftax-Classic achievement chains when a detailed game manual is available. **RQ3** tests whether interaction-derived laws match an unfamiliar Mars world's behavior and support skill learning without supplied environment-specific rules.

5.1 Environments

Craftax and Craftax-Classic. Craftax (Matthews et al., 2024) is a procedurally generated survival environment in which agents explore, gather resources, craft tools, and maintain health, food, and water while completing achievements. Craftax-Classic contains a single 64×64 overworld; reaching diamond requires a long chain of exploration, mining, and increasingly capable tools within a 10,000-step episode. Full Craftax extends the setting to nine floors with scarcer resources and combat-gated progression. RQ2 uses Craftax-Classic with a detailed game manual for BC+ generation, symbolic observations, and primitive actions. Seven achievements span resource collection and successive tool tiers (Zabounidis et al., 2026; Bornemann et al., 2026).

Mars. Mars (Tang et al., 2024) is a Crafter-based open-world benchmark for situated inductive reasoning. Its worlds can change terrain, survival interactions, resource yields, and crafting dependencies, deliberately violating familiar game rules. Environment-specific rules must be established from interaction.

5.2 Baselines

CODE-SHARP. CODE-SHARP (Bornemann et al., 2026) uses language models to discover and evolve hierarchical reward programs, then trains a goal-conditioned policy to execute the discovered skills. CODE-FRP is CODE-SHARP's flat reward-program ablation.

ELLM. ELLM (Du et al., 2023) uses an LLM to suggest context-dependent language goals from a caption of the current observation and the available actions. A transition captioner describes what the agent actually did; semantic similarity between this description and a suggested goal supplies intrinsic reward for training a goal-conditioned policy.

OMNI. OMNI (Zhang et al., 2024) uses a foundation model to identify tasks that are both interesting and within the agent's current learning frontier, then samples tasks according to learning progress. Its open-ended formulation can express a proposed task as an ordered sequence of environment-state conditions whose completion is checked in order.

SCALAR. SCALAR (Zabounidis et al., 2026) uses an LLM to propose symbolic skill operators, trains the corresponding options with reinforcement learning, and composes those options to pursue long-horizon goals. However, it augments Craftax's symbolic observations with relative-position offsets to nearby relevant blocks and mobs, providing distance-oriented spatial cues beyond the native observation protocol. Since SCALAR's source code has not been released, we cannot reproduce this augmentation and establish a matched observation setting. We therefore exclude SCALAR from our quantitative comparisons.

Table 1: Craftax behavioral-test pass rates and generation costs, averaged over three runs.

Model	Tests passed (%)						Generation cost		Time
	Initial	1	2	3	4	5	Tokens	USD	(min)
deepseek-v4.1-flash	0.00	59.94	96.94	96.33	100.00	100.00	954K	$0.21	20.20
glm-5.3-flash	91.74	99.08	88.99	33.03	99.08	99.39	705K	$0.11	26.94
gpt-6-luna	28.44	54.43	61.47	61.47	88.99	99.08	402K	$0.05	17.80
gpt-6-sol	89.91	100.00	100.00	100.00	100.00	100.00	336K	$0.98	19.23

5.3 Implementation and Evaluation Details

Implementation. We run Craftax through C++ EnvPool (Weng et al., 2022) and train separate PPO operator policies in PyTorch on Intel Xeon Platinum 8280 CPUs at 2.70 GHz, with 112 logical CPUs and 100 available to the container. In a calibration with 48 parallel training jobs, each using 256 environments and 64-step rollouts, aggregate training throughput reached 94,223 environment steps per second. This measures the parallel policy-training workload, not the complete world-law pipeline. Our earlier JAX implementation required more than 30 seconds for a cold compilation whenever a changed operator altered the compiled graph, reducing effective throughput for short, frequently revised training jobs. Small independent networks and checkpoints avoid this repeated cost; Craftax-Classic policy training takes approximately 2.5 hours. Following CODE-SHARP, our Craftax runs use a death-prevention setting while retaining task prerequisites and episode limits.

Training and interaction budgets. Our training uses 256 parallel environments, 64 rollout steps per update, four PPO epochs, 16 minibatches, a learning rate of 3×10^{-4}, and a discount factor of 0.999. Operators receive an initial 10^6 steps and, when they pass the continuation gate, another 3×10^6 steps; each subsequent climb round allocates 3×10^6 steps to a selected operator. A recorded policy-grounding run used approximately 1.3×10^8 environment steps in total. This is an observed run size, not a fixed budget: improvements in prerequisites can trigger further climb rounds, so the total number of training steps depends on the run. Language-model token use and API cost for BC+ generation and repair are summarized in Table 1, with detailed counts and pricing in Appendix A.

Evaluation. Each goal is evaluated over 256 episodes in 256 parallel environments, with actions sampled from the policy and a 10,000-step episode limit. Evaluation starts from natural initial states without injecting training-frame pools. Success requires completing the designated achievement within the episode. The recorded wood-collection and wooden-pickaxe evaluations use a warmup setting that ignores operator preconditions; the other five retain them. Appendix C reports additional item and achievement targets and marks the warmup evaluations. RQ3 permits only interface declarations and newly collected interaction evidence; controlled state construction is confined to development experiments and is never used in final evaluation.

6 Results

6.1 RQ1: Improving BC+ World-Law Accuracy through Test-Guided Revision

RQ1 asks to what extent test-guided revision improves the accuracy of LLM-generated BC+ world laws on predefined behavioral tests. Four models generate from the same Craftax materials; the 109 environment-checked cases described in Section 4.2 are fixed before law generation. Table 1 reports the initial theories and five feedback rounds, averaged over three runs.

Initial candidates contain test-detected errors. Test pass rates range from 0.00% to 91.74%, with a mean of 52.52%. Even the strongest initial theory leaves some tested behaviors incorrect. This variation motivates explicit behavioral testing and feedback-guided repair before generated laws are used for skill synthesis. **Test-driven repair raises pass rates.** The mean reaches 99.62% after five rounds, an improvement of 47.10 percentage points, with all four models reaching 99.08-100.00%. **Lower-cost models also reach high final pass rates.** The stronger proprietary model, gpt-6-sol, reaches 100.00% after the first repair round. The three lower-cost models reach 99.08-100.00% by

Table 2: Craftax-Classic achievement success rates (%).

Achievement	CODE-SHARP	CODE-FRP	ELLM	OMNI	ATLAS
Collect wood	89.2	100.0	100.0	99.7	99.6
Collect stone	99.8	20.0	9.7	69.9	99.6
Make wood pickaxe	99.9	99.4	72.7	77.9	100.0
Make stone pickaxe	99.7	53.2	0.0	77.9	99.2
Collect iron	98.1	18.1	0.0	0.0	98.0
Make iron pickaxe	94.7	16.5	0.0	0.0	94.1
Collect diamond	16.0	0.1	0.0	0.0	45.7

the fifth round. Their mean API costs for initial generation and all five repair rounds are $0.21, $0.11, and $0.05 per run, respectively; even gpt-6-sol remains below $1.

Together, these results show that test-guided revision improves accuracy on the predefined behavioral cases across models and enables inexpensive models to approach the final test performance of a stronger model. The pass rates do not establish accuracy on untested behaviors.

6.2 RQ2: Operator Grounding and Composition with Craftax Laws

Starting from the BC+ laws generated and repaired in RQ1, we synthesize operators and ground their skill specifications in HRL policies.

Bottom-up grounding works well, particularly on the long-horizon diamond task. Our bottom-up grounding procedure turns law-derived skill specifications into executable policies, with climb revisiting downstream operators as their prerequisite policies become usable or improve. The complete training trace in Appendix D records progression from stone tools, coal, and furnace construction to iron acquisition, iron tools, and diamond collection. Diamond collection is particularly demanding: the agent must acquire resources, craft successive tool tiers, explore to locate diamond, navigate to it, and mine it with the required tool. Successful execution therefore depends on both learned operators and their composition across a long prerequisite chain. From natural initial states ATLAS achieves 45.7% diamond-collection succes, compared with CODE-SHARP's reported 16.0% (Table 2). The six preceding resource and tool targets achieve 94.1-100.0% success. Together, these results demonstrate that the world-law framework, coupled with bottom-up grounding and climb, supports end-to-end execution of the deepest evaluated Craftax-Classic goal. Appendix B details the grounding procedure, and the operator-results table in Appendix C reports additional operator outcomes.

Although our central focus is world-law construction and specification derivation, grounding requires a design that preserves usable prerequisite skills while learning downstream ones. CODE-SHARP trains a shared goal-conditioned Transformer-XL policy across its evolving skill archive (Bornemann et al., 2026); Instead, in ATLAS, each operator has its own independent policy network and checkpoint. We argue that such design relieves the catastrophic forgetting problem in long-term continual learning, which may be the reason for code-sharp's performance drop on the diamond-mining task.

Table 3: Counterfactual tests of the acquired stone-pickaxe law.

Condition removed	Stone	Wood	Table	Furnace	Prediction	Observed
None	4	4	Yes	Yes	Gain one	$4 \to 5$
Stone	0	4	Yes	Yes	No gain	$4 \to 4$
Wood	4	0	Yes	Yes	No gain	$4 \to 4$
Table	4	4	No	Yes	No gain	$4 \to 4$
Furnace	4	4	Yes	No	No gain	$4 \to 4$

Table 4: Selected achievement success rates and scores (%). Scores use the same 14 targets in Appendix E; world configurations and evaluation procedures differ.

Method	Collect stone	Place table	Make wood pickaxe	Make stone pickaxe	Make wood sword	Score (14 targets)
Mars	0.00	14.29	14.29	0.00	7.14	2.45
ATLAS (ours)	100.0	99.6	96.5	8.2	96.5	19.69

6.3 RQ3: Zero-Prior Knowledge, Acquiring World Laws in Unfamiliar Mars Worlds

Craftax rules may already be represented in an LLM's training knowledge, making it difficult to attribute successful law construction to knowledge acquired during the experiment. RQ3 therefore uses Mars, a Crafter variant that generates unconventional combinations of crafting and resource-acquisition rules. We select its terrain-and-task-dependency world `world_terr_ach`, where cutting a tree yields coal instead of wood, while mining stone yields both stone and wood. These counter-commonsense combinations help distinguish knowledge acquired from interaction with the current world from familiar rules recalled from the LLM's training knowledge.

Cold-start law discovery. The run begins without an environment-specific BC+ theory or derived skill specifications, so its warmup starts with RL random exploration rather than law-guided skill training. We extract action-relevant transitions from the resulting trajectories, retaining both local successes and failures. Trajectory simplification removes irrelevant and duplicate transitions. Because this world's action outcomes deliberately violate familiar game semantics, we disable semantic purification for RQ3. Candidate laws drawn from the simplified trajectories proceed to counterfactual validation in new environment interactions.

Counterfactual validation. To illustrate the counterfactual validation process, the stone-pickaxe candidate states that crafting requires stone, wood, a nearby table, and a nearby furnace, and consumes one unit of each material. The environment experimenter designs controlled comparisons that remove one proposed condition at a time while retaining the others, then compares the observed outcomes with the candidate's predictions. Table 3 shows the outcome: the pickaxe count changes from four to five when all four conditions hold and remains four when any one condition is removed. These contrasts support the candidate's proposed prerequisites within the tested situations.

Policy evaluation. After candidate laws pass the recorded checks, they are compiled into skill specifications for operator training and natural-start evaluation. Table 4 reports five selected achievement outcomes and scores aggregated over the same 14 targets for Mars Skill Library and ATLAS. Mars denotes the official Skill Library results in Task Dep., digitized from Figure 4 of the Mars benchmark (Tang et al., 2024) as detailed in Appendix E; zeros indicate entries without a visible bar. Our entries follow the same appendix's operator evaluations in Terrain + Task Dependency, using the evaluation counts and 20-episode exceptions documented there.

Scores in Table 4 use the Mars benchmark's geometric aggregation over the same 14 targets in Appendix E, yielding approximately 2.45% for Mars Skill Library and 19.69% for ATLAS. The ATLAS score uses unrounded evaluation rates and counts Collect Coal once through its highest-performing operator. The shared target set aligns score coverage; world configurations and evaluation procedures differ as described above.

7 Conclusion

We presented ATLAS, a world-law-driven LLM-HRL framework that maintains explicit BC+ laws, derives skill specifications through deterministic symbolic computation, and uses execution evidence to refine the laws and retrain affected skills. Independent symbolic and environment tests constrain law acceptance, while counterfactual probes support targeted revisions. With detailed Craftax-Classic materials, the resulting operators performed well across the selected achievement chains; Mars probes demonstrated law acquisition without supplied environment-specific rules. Future work will evaluate law generalization and the independent contributions of refinement and synthesis across more environments and training seeds.

AI USE STATEMENT

Generative AI tools were used as part of the research method: language models constructed candidate behavioral tests, generated and revised candidate BC+ world laws, and proposed evidence-grounded law refinements. These outputs were evaluated through the compilation, symbolic-testing, and environment-testing procedures described in the paper. Generative AI tools were also used to polish the manuscript's language and improve readability. The authors reviewed the AI-assisted work and take responsibility for the final text, claims, and research artifacts.

REFERENCES

Joseph Babb and Joohyung Lee. Action language BC+: Preliminary report. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015. doi: 10.1609/aaai.v29i1.9405. URL https://ojs.aaai.org/index.php/AAAI/article/view/9405.

Joseph Babb and Joohyung Lee. Action language BC+. *Journal of Logic and Computation*, 30 (4):899–922, 2020. doi: 10.1093/logcom/exv062. URL https://doi.org/10.1093/logcom/exv062.

Richard Bornemann, Pierluigi Vito Amadori, and Antoine Cully. Code-sharp: Continuous open-ended discovery and evolution of skills as hierarchical reward programs. *arXiv preprint arXiv:2602.10085*, 2026. URL https://arxiv.org/abs/2602.10085.

Yuqing Du, Olivia Watkins, Zihan Wang, Cédric Colas, Trevor Darrell, Pieter Abbeel, Abhishek Gupta, and Jacob Andreas. Guiding pretraining in reinforcement learning with large language models. In *International Conference on Machine Learning*, pp. 8657–8677. PMLR, 2023. URL https://proceedings.mlr.press/v202/du23f.html.

Michael Gelfond and Vladimir Lifschitz. Representing action and change by logic programs. *The Journal of Logic Programming*, 17(2–4):301–321, 1993. doi: 10.1016/0743-1066(93)90035-F. URL https://doi.org/10.1016/0743-1066(93)90035-F.

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 *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL 2025)*, 2025. URL https://arxiv.org/abs/2502.13092.

Adam Ishay and Joohyung Lee. LLM+AL: Bridging large language models and action languages for complex reasoning about actions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025. URL https://arxiv.org/abs/2501.00830.

Mu Jin, Zhihao Ma, Kebing Jin, Hankz Hankui Zhuo, Chen Chen, and Chao Yu. Creativity of AI: Automatic symbolic option discovery for facilitating deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 7042–7050, 2022. doi: 10.1609/aaai.v36i6.20663. URL https://ojs.aaai.org/index.php/AAAI/article/view/20663.

Joohyung Lee, Vladimir Lifschitz, and Fangkai Yang. Action language BC: Preliminary report. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 983–989, 2013. URL https://asu.elsevierpure.com/en/publications/action-language-bc-preliminary-report/.

Zhaoyi Li, Kelin Yu, Shuo Cheng, and Danfei Xu. League++: Empowering continual robot learning through guided skill acquisition with large language models. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*, 2024.

Daoming Lyu, Fangkai Yang, Bo Liu, and Steven Gustafson. SDRL: Interpretable and data-efficient deep reinforcement learning leveraging symbolic planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 2970–2977, 2019. doi: 10.1609/aaai.v33i01.33012970. URL https://ojs.aaai.org/index.php/AAAI/article/view/4153.

Wensen Mao, Wenjie Qiu, Yuanlin Duan, and He Zhu. Skill discovery using language models, 2025. URL https://openreview.net/forum?id=DBbgasVgyQ. ICLR 2025 submission.

Michael Matthews, Michael Beukman, Benjamin Ellis, Mikayel Samvelyan, Matthew Thomas Jackson, Samuel Coward, and Jakob Nicolaus Foerster. Craftax: A lightning-fast benchmark for open-ended reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 35104–35137, 2024. URL https://proceedings.mlr.press/v235/matthews24a.html.

Vasanth Sarathy, Daniel Kasenberg, Shivam Goel, Jivko Sinapov, and Matthias Scheutz. SPOTTER: Extending symbolic planning operators through targeted reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2021. URL https://arxiv.org/abs/2012.13037.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1–2): 181–211, 1999. URL https://www.sciencedirect.com/science/article/pii/S0004370299000521.

Xiaojuan Tang, Jiaqi Li, Yitao Liang, Song-chun Zhu, Muhan Zhang, and Zilong Zheng. Mars: Situated inductive reasoning in an open-world environment. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, 2024. URL https://marscrafter.github.io/.

Jiayi Weng, Min Lin, Shengyi Huang, Bo Liu, Denys Makoviichuk, Viktor Makoviychuk, Zichen Liu, Yufan Song, Ting Luo, Yukun Jiang, Zhongwen Xu, and Shuicheng Yan. EnvPool: A highly parallel reinforcement learning environment execution engine. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://arxiv.org/abs/2206.10558.

Fangkai Yang, Daoming Lyu, Bo Liu, and Steven Gustafson. PEORL: Integrating symbolic planning and hierarchical reinforcement learning for robust decision-making. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 4860–4866, 2018. doi: 10.24963/ijcai.2018/675. URL https://www.ijcai.org/proceedings/2018/675.

Renos Zabounidis, Yue Wu, Simon Stepputtis, Woojun Kim, Yuanzhi Li, Tom Mitchell, and Katia Sycara. Scalar: Learning and composing skills through LLM guided symbolic planning and deep RL grounding. *arXiv preprint arXiv:2603.09036*, 2026. URL https://arxiv.org/abs/2603.09036.

Jenny Zhang, Joel Lehman, Kenneth Stanley, and Jeff Clune. OMNI: Open-endedness via models of human notions of interestingness. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/pdf/99084a826726857a7d2257afd1c3fc5de3dff4d3.pdf.