ONTOACT: HYPOTHESIS-BASED NEURO-SYMBOLIC AGENTS THROUGH ONTOLOGICAL ABSTRACTION

Anonymous authors
Paper under double-blind review

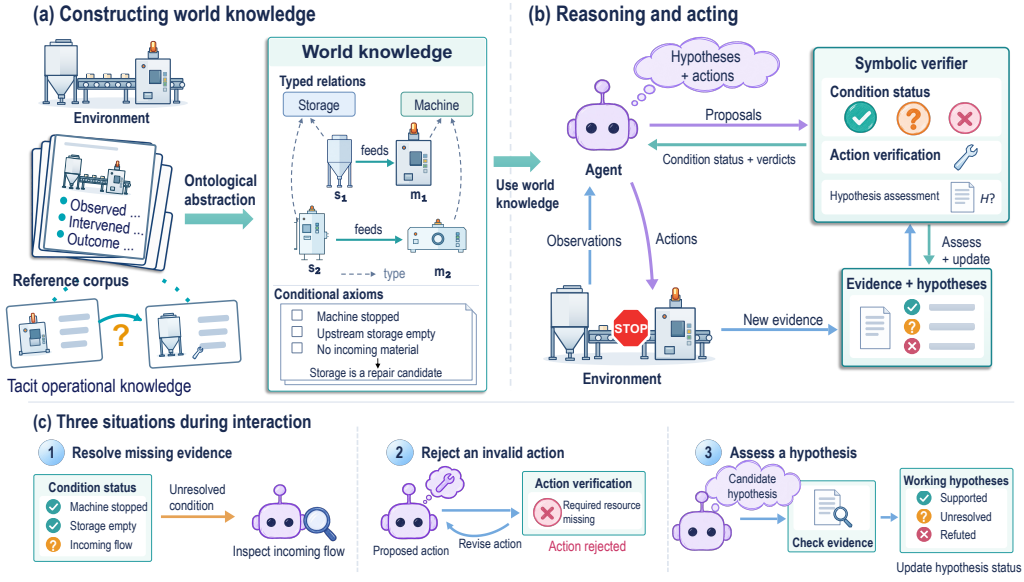


Figure 1: **The LLM proposes, the ontology verifies.** (a) OntoAct abstracts recorded experience into world knowledge over typed entities, relations, and conditions. (b) Symbolic checks assess proposed hypotheses and actions against current evidence. (c) Feedback guides observation, action revision, and hypothesis assessment.

ABSTRACT

Learning from experience requires recovering reusable knowledge and determining when it applies to a new situation. We introduce OntoAct, a hypothesis-based neuro-symbolic framework built on *ontological abstraction*. OntoAct turns individual experiences into axioms over entity types, relations, and conditions, making their underlying patterns reusable across situations. A large language model (LLM) uses this world knowledge to propose hypotheses and actions, while a symbolic verifier assesses their supporting conditions against current observations. Verification identifies missing evidence, and new observations and intervention outcomes update hypothesis assessments and guide subsequent actions. We also introduce Tacit-FLE, a benchmark built on the Factorio Learning Environment (FLE) for diagnosing and repairing hidden faults in existing factories using inherited operational records. Its synthetic maintenance corpus distributes evidence across records, and an executed repair may leave the fault unresolved, requiring agents to reassess their diagnoses through interaction. Across Tacit-FLE, Mars, and ALFWorld, OntoAct achieves leading task performance with efficient interaction and inference.

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.

1 INTRODUCTION

"We can know more than we can tell."
— Michael Polanyi, *The Tacit Dimension* (Polanyi, 1966)

Agents often need to act in environments whose operating principles are only partially specified. Inherited logs, notes, and interaction trajectories record what happened and what was done. They may leave unstated the dependencies and conditions that made an action useful, or when that experience applies elsewhere. We call this *tacit operational knowledge*: knowledge expressed through recorded observations, interventions, and outcomes but left implicit as reusable principles.

Using this experience requires bridging two gaps. First, the agent must recover patterns that generalize beyond the entities named in individual records. A particular machine–storage connection, for example, instantiates a supplier–recipient relationship; reusable rules express conditions over the entity types and these relational roles. Second, it must determine whether those relationships and conditions hold in its current situation. Both gaps matter whether experience is retrieved as text or encoded as executable or symbolic rules (Lewis et al., 2020; Zhou et al., 2025b; Choi et al., 2025). The agent needs explicit hypotheses that it can test through observation and intervention, retain when supported, and revise when contradicted.

We introduce OntoAct, a neuro-symbolic framework that connects these tasks through *ontological abstraction*. It retains incident-specific entities and relations while expressing reusable axioms over types, relational roles, and operating conditions, so the same pattern can be instantiated for different entities. A large language model (LLM) proposes hypotheses and actions from this world knowledge; a symbolic verifier checks their support against current observations (Figure 1). Feedback identifies supported hypotheses and conditions requiring further observation. New observations and intervention outcomes guide hypothesis revision and subsequent actions. The guiding principle is simple: *the LLM proposes, the ontology verifies*.

To evaluate knowledge recovery and use, we also introduce Tacit-FLE, a maintenance benchmark built on the Factorio Learning Environment (Hopkins et al., 2025). It asks agents to diagnose hidden faults in existing plants from a finite corpus of synthetic maintenance logs and operating documents, together with live observations (Figure A1). Evidence is distributed across records, and an executed repair may leave the fault unresolved. Success therefore requires selecting useful evidence and translating it into an effective repair. The benchmark evaluates how agents combine inherited knowledge with feedback from their own observations and interventions. Our contributions are:

- **Tacit-FLE** (Section 2), a benchmark for diagnosis and repair from inherited operational records, designed to test the combination of distributed evidence with observations and intervention outcomes;
- **OntoAct** (Section 3), a framework that turns individual experiences into reusable ontological world knowledge and uses it to form, test, and revise hypotheses that guide interaction;
- **Performance and efficiency** (Section 4), demonstrating leading task performance and efficient interaction across Tacit-FLE, Mars, and ALFWorld. We further support these results with a range of analyses across experimental settings.

We introduce Tacit-FLE in Section 2 and present the OntoAct framework in Section 3. Section 4 evaluates the framework on Tacit-FLE and existing benchmarks, followed by related work in Section 5 and conclusions in Section 6.

2 TACIT-FLE: A BENCHMARK FOR TACIT KNOWLEDGE

Tacit-FLE evaluates how agents recover dependencies and repair conditions from past incidents and apply them to new faults. Built on the factory simulation of FLE (Hopkins et al., 2025), it pairs existing plants with inherited maintenance records and supports diagnosis through observation and intervention (Appendix A, Figure A1).

2.1 TASK SETTING

Existing plant. An *episode* starts with one hidden fault in an existing plant. The agent receives unit kinds, positions, and operating readings. Inspections reveal further details, while production evolves as actions advance simulation time (Appendix A.1).

Reference corpus. Each plant comes with a fixed, finite corpus D of synthetic *maintenance logs* and *operating documents*, provided before evaluation. The logs describe local observations, checks, repairs, and outcomes; operating documents state selected expectations, such as planned outputs. These sources support inference of component dependencies and repair conditions.

Diagnosis and repair. Within a fixed turn budget, the agent inspects units, waits, or attempts replacements or refills. A replacement restores the unit's original specification, including configuration. Success requires replacing an accepted repair target or completing a valid refill of an accepted storage target with the required material. Failed attempts consume turns and provide feedback; episodes end upon success or budget exhaustion.

2.2 FAULTS AND DIAGNOSTIC EVIDENCE

Faults include supply interruptions and off-plan configurations, motivated by component failures and departures from intended operation in process diagnosis (Venkatasubramanian et al., 2003). Drawing on model-based diagnosis (de Kleer & Williams, 1987; 1989), we construct three task types that differ in how inherited records support a new diagnosis. Appendix A.3 provides examples of the records and observations in each setting.

🔗 **Chain.** Local incidents provide evidence of dependencies between components. For example, one log pairs a stoppage at B with a failure at A; another pairs a stoppage at C with a failure at B. Combining the incidents suggests a dependency path $A \to B \to C$. When C stalls in a new episode, this path guides upstream inspections, where current readings help locate a supply interruption.

🧩 **Conjunction.** A repair precedent, an operating plan, and a current observation play complementary roles. For example, a past log records that replacing machine A resolved a mismatch between its configured and planned output. In a new episode, the plan specifies product X for machine B, while inspection shows B configured for Y. Recognizing the shared mismatch condition connects a repair demonstrated on A to a candidate intervention on B.

🔻 **Elimination.** Records describe destroyed units, supply interruptions, and power failures, but omit the off-plan configuration encountered in evaluation. The agent uses recorded symptoms and checks to rule out explanations inconsistent with current observations. Further observations or interventions help distinguish the remaining possibilities.

Construction and evaluation. Plants, fault cases, and reference corpora are generated from a common specification (Appendix A.2), with the evaluation population and scale-specific settings detailed in Appendix A.4 and construction checks in Appendix A.5. Records exclude repairs of evaluation targets; consistent plant-specific identifiers reduce shortcuts from Factorio vocabulary. We report repair success, mean turns, and spurious interventions (executed repairs that do not meet the success criterion); Appendix A.4.1 gives their definitions and aggregation rules.

3 ONTOACT: ONTOLOGICAL ABSTRACTION FOR ACTION

OntoAct turns inherited experience into reusable world knowledge for assessing proposals during interaction. Following *the LLM proposes, the ontology verifies*, the LLM proposes axioms, hypotheses, and actions, while a symbolic verifier checks them against supporting evidence using the ontology's types, relations, and conditions. This division governs both knowledge construction and action (Figure 2).

3.1 ONTOLOGICAL ABSTRACTION

Reusing an incident requires identifying the relations and conditions that make it relevant to another situation. *Ontological abstraction* expresses these requirements over entity types and relational roles while retaining the particular facts from which they were drawn.

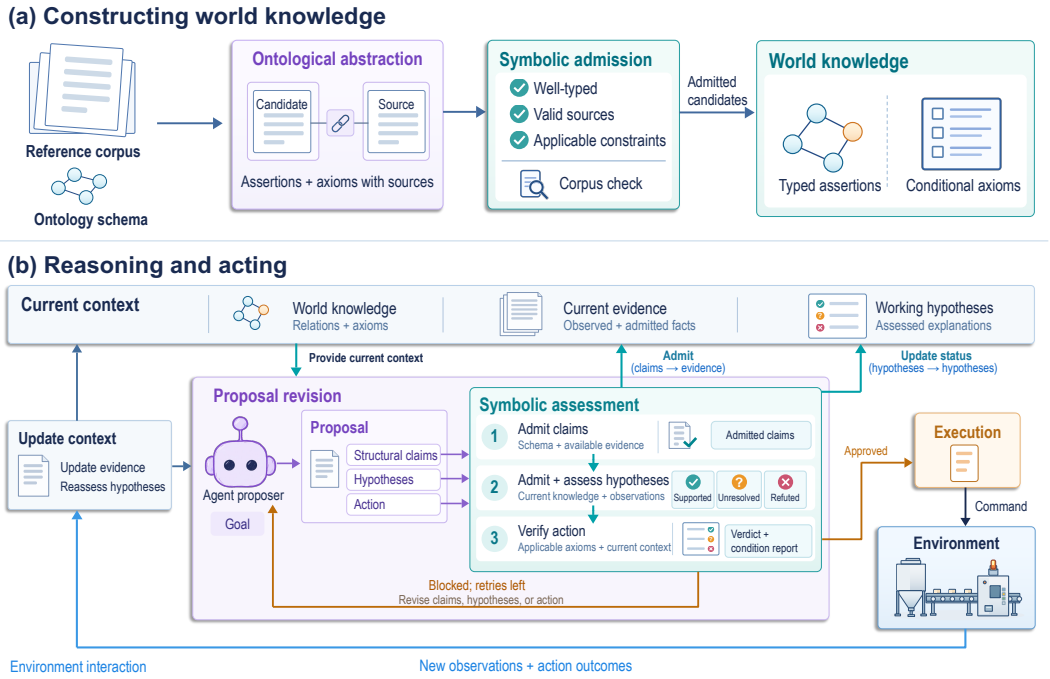


Figure 2: **The OntoAct framework.** (a) The LLM proposes axioms from a reference corpus; symbolic admission checks their conformance to the schema and applicable evidence constraints. (b) The LLM proposes claims, hypotheses, and actions using the resulting world knowledge; ontology-based verification supplies feedback for revision, while observations and action outcomes update the reasoning context.

World knowledge. An ontology schema T defines entity types, relations, observable attributes, and admissible axiom forms. The corpus-derived ontology separates ground assertions A_W from conditional axioms R_W, which express action requirements and diagnostic patterns:

$$K_W = (T, A_W, R_W). \tag{1}$$

Episode evidence. Applying these reusable conditions requires evidence from the current situation. At step t, f_{ground} converts observations into episode assertions A_t. These assertions augment the corpus-derived facts, yielding the knowledge used to assess hypotheses and actions:

$$K_t = (T, A_W \cup A_t, R_W). \tag{2}$$

This combines reusable knowledge with current evidence; see Appendix B.3.1 for a detailed example.

3.2 Constructing world knowledge

To build this world knowledge, OntoAct processes the reference corpus D before interaction. The LLM proposes candidate axioms, and the ontology checks which ones enter K_W.

The LLM proposes axioms. For a record batch $b \subseteq D$, the LLM proposes ground assertions and conditional axioms in the forms allowed by T:

$$\widetilde{X}_b = f_{\text{abs}}(T, b) = \{(x_i, \pi_i)\}_{i=1}^{n_b}, \tag{3}$$

Each candidate conditional axiom x_i is paired with source records π_i; n_b counts all candidates.

Algorithm 1 Constructing world knowledge

Require: ontology schema T, corpus D
$K_W \leftarrow (T, \varnothing, \varnothing)$
for each batch $b \subseteq D$ **do**
  $\widetilde{X}_b \leftarrow f_{\text{abs}}(T, b)$ ▷ LLM; Eq. 3
  for each candidate $(x_i, \pi_i) \in \widetilde{X}_b$ **do**
    $\alpha_i \leftarrow \text{Admissible}(x_i, \pi_i; K_W, D)$ ▷ Eq. 4b
    if $\alpha_i = 1$ **then**
      $K_W \leftarrow f_{\text{store}}(K_W, x_i, \pi_i)$ ▷ Store pair
    end if
  end for
end for
$K_W \leftarrow f_{\text{corpus}}(K_W, D)$ ▷ Corpus checks
return K_W

The ontology constrains admission. Admission requires schema and source validity (β_i) and a set $\mathcal{C}_{x_i}$ of tests for its axiom type:

$$\beta_i = \mathrm{Typed}(x_i; T)\,\mathbf{1}[\varnothing \neq \pi_i \subseteq D], \tag{4a}$$

$$\alpha_i = \mathrm{Admissible}(x_i, \pi_i; K_W, D) = \beta_i \prod_{c \in \mathcal{C}_{x_i}} \mathbf{1}[c(x_i, \pi_i; K_W, D)]. \tag{4b}$$

Here Typed checks schema conformance and $\mathbf{1}[P]$ indicates whether P holds. Every test $c \in \mathcal{C}_{x_i}$ must pass; Appendix B.2 specifies the tests for each environment. Admitted axioms ($\alpha_i = 1$) enter A_W or R_W with their sources via f_{store}; f_{corpus} checks the assembled knowledge (Algorithm 1).

3.3 Reasoning and acting with world knowledge

The LLM proposes claims and actions. Let A, K, and H denote current assertions, knowledge, and hypotheses; primes mark updates. Given goal g and feedback η (initially empty each turn), the LLM proposes claims ΔA, hypotheses ΔH, and action a:

$$(\Delta A, \Delta H, a) = f_{\mathrm{propose}}(g, K, H, \eta). \tag{5}$$

Structural claims specify relations. Each hypothesis h specifies required relations and nonempty expected conditions $\Phi(h)$, instantiating an induced pattern or proposing a new explanation.

Checking claims against current evidence. For episode claims, f_{admit} checks types, structural evidence, and correspondence with cited patterns, including every required condition. Admitted structural claims update knowledge:

$$A' = A \cup f_{\mathrm{admit}}(\Delta A; K), \tag{6a}$$

$$K' = (T, A_W \cup A', R_W). \tag{6b}$$

Algorithm 2 Reasoning and acting

Require: K_W, goal g, initial observation o_1
Require: B turns; up to J proposals per turn
$A, H, a_{\mathrm{prev}} \leftarrow \varnothing$
for $t = 1, \ldots, B$ **do**
  $A \leftarrow f_{\mathrm{ground}}(A, o_t)$ ▷ Observations
  $K \leftarrow (T, A_W \cup A, R_W)$
  $H \leftarrow f_{\mathrm{eval}}(H; K, a_{\mathrm{prev}})$ ▷ Reassess hypotheses
  $\eta \leftarrow \varnothing$
  for $j = 1, \ldots, J$ **do**
    $(\Delta A, \Delta H, a) \leftarrow f_{\mathrm{propose}}(g, K, H, \eta)$ ▷ Eq. 5
    $A \leftarrow A \cup f_{\mathrm{admit}}(\Delta A; K)$ ▷ Eq. 6a
    $K \leftarrow (T, A_W \cup A, R_W)$ ▷ Eq. 6b
    $H \leftarrow f_{\mathrm{eval}}(H \cup f_{\mathrm{admit}}(\Delta H; K); K)$ ▷ Eq. 8
    $(v, \eta) \leftarrow f_{\mathrm{verify}}(a; K, H)$ ▷ Eq. 9
    if $v \neq \bot$ **then break**
    end if
  end for
  $(a_{\mathrm{prev}}, o_{t+1}) \leftarrow f_{\mathrm{execute}}(a, v)$ ▷ Act; observe
  if terminated **then break**
  end if
end for

When h instantiates pattern r with conditions Φ_r, $\Phi(h)$ retains every condition, substituting current entities. Each atom $\varphi = (u, p, \mathrm{op}, z)$ tests entity u's observable attribute p against expected value z with comparison op; current readings yield:

$$[\![\varphi]\!]_{K'} = \begin{cases} \top, & \text{resolved evidence satisfies } p(u)\ \mathrm{op}\ z, \\ \bot, & \text{resolved evidence contradicts } p(u)\ \mathrm{op}\ z, \\ ?, & \text{otherwise.} \end{cases} \tag{7}$$

Missing readings remain unknown (?), and the same evaluations test action conditions. An open, admitted hypothesis receives the status:

$$\mathrm{Status}(h) \leftarrow \begin{cases} \text{refuted}, & \exists \varphi \in \Phi(h) : [\![\varphi]\!]_{K'} = \bot, \\ \text{confirmed}, & \forall \varphi \in \Phi(h) : [\![\varphi]\!]_{K'} = \top, \\ \text{open}, & \text{otherwise.} \end{cases} \tag{8}$$

Confirmation records observational support, subject to refutation by later readings or repair outcomes; updated hypotheses form H'.

The ontology verifies the action. The verifier tests R_W: violated conditions and prohibitions block actions; missing readings alone do not. An action requiring explanatory support needs a confirmed hypothesis whose conditions still hold; observations need no confirmed hypothesis. It returns:

$$(v, \eta) = f_{\mathrm{verify}}(a; K', H'), \tag{9}$$

A blocked verdict $v = \bot$ requests revision while budget remains; η reports failed and unresolved checks. Algorithm 2 summarizes this interaction loop. Appendix B.3 details hypothesis admission, condition evaluation, repair verification, and refutation by observations and outcomes.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Environments and records. **Tacit-FLE** tests hidden-fault repair across three task types and plant scales (Section 2). **Mars** (Tang et al., 2024b) tests survival and crafting under altered world mechanics; we use the eight-world configuration of Zhou et al. (2025b). **ALFWorld** (Shridhar et al., 2021) tests household instruction following on 134 unseen tasks. Corpus-based methods share plant maintenance and operating records, Mars exploration records, or ALFWorld training-game trajectories; Appendix E.1 details these corpora and evaluation tasks.

Baselines. We compare **OntoAct** with **ReAct** (Yao et al., 2023), retrieval-augmented **ReAct*** (Lewis et al., 2020), corpus-adapted **Reflexion** (Shinn et al., 2023), and **IfR** (Tang et al., 2024b), which uses natural-language rules. **NeSyC** (Choi et al., 2025) and **WALL-E** (Zhou et al., 2025b) check actions against induced answer-set programs and executable rules, respectively. Implementations and prompts are detailed in Appendix C.2.

Models and protocol. We use GPT-5.6 Luna (OpenAI, 2026a) with reasoning effort `none`. Medium Tacit-FLE and Mars also use Luna with `medium` effort (Luna^R) and GPT-5.6 Terra (Terra; `none`) (OpenAI, 2026b). Methods share tasks and interaction budgets; Appendix E details settings, budgets, and metrics.

4.2 TACIT-FLE

How well does OntoAct repair hidden faults? Failed repairs consume turns without ending an episode, so eventual success can mask repeated unsuccessful interventions. OntoAct achieves the highest overall success at every scale with 70–91% fewer spurious interventions than IfR, the strongest baseline by overall success (Table 1). It also uses the fewest turns among methods with successful repairs. Its largest advantage is on chain tasks requiring dependency inference across records; IfR matches or exceeds it on elimination.

Table 1: **Repair performance on Tacit-FLE.** Success (%) by type and overall (All); SI: mean spurious interventions; Turns: mean turns. Failed episodes use the full budget (Appendix A.4.1). Bold: best; underlined: second-best overall success, interventions, and turns. Intervention and turn rankings require at least one repair.

		Repair success (%) ↑					
Scale	Method	Chain	Conjunction	Elimination	All	SI ↓	Turns ↓
Small	ReAct	0.0	71.4	**75.0**	39.3	<u>4.6</u>	22.2
	ReAct*	15.4	85.7	37.5	39.3	7.2	22.9
	Reflexion	7.7	71.4	**75.0**	42.9	5.3	23.2
	IfR	38.5	**100.0**	**75.0**	<u>64.3</u>	6.4	<u>17.4</u>
	NeSyC	0.0	0.0	0.0	0.0	0.0	30.0
	WALL-E	7.7	14.3	12.5	10.7	7.1	29.0
	OntoAct (ours)	**69.2**	**100.0**	**75.0**	**78.6**	**1.9**	**11.8**
Medium	ReAct	28.6	64.3	75.0	52.8	11.3	41.8
	ReAct*	21.4	64.3	87.5	52.8	13.4	38.7
	Reflexion	21.4	42.9	37.5	33.3	15.4	46.4
	IfR	28.6	85.7	**100.0**	<u>66.7</u>	11.9	<u>31.7</u>
	NeSyC	0.0	0.0	0.0	0.0	0.0	60.0
	WALL-E	14.3	28.6	87.5	36.1	<u>10.1</u>	48.0
	OntoAct (ours)	**57.1**	**100.0**	**100.0**	**83.3**	**2.4**	**19.4**
Large	ReAct	5.9	72.7	75.0	45.0	16.1	49.9
	ReAct*	23.5	36.4	66.7	40.0	19.6	56.4
	Reflexion	11.8	9.1	50.0	22.5	23.2	63.1
	IfR	23.5	90.9	**100.0**	<u>65.0</u>	17.8	<u>42.6</u>
	NeSyC	0.0	0.0	0.0	0.0	0.0	75.0
	WALL-E	5.9	45.5	75.0	37.5	<u>14.9</u>	56.6
	OntoAct (ours)	**82.4**	**100.0**	91.7	**90.0**	**1.6**	**17.2**

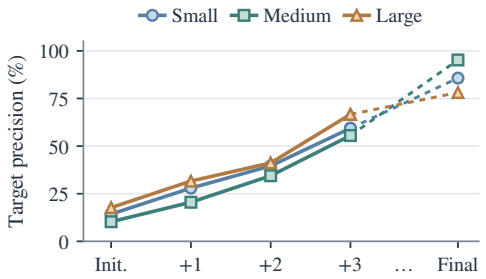


Figure 3: **Hypothesis precision over turns.** Target precision for all tracked hypotheses before selection (Init.) and those retained +1/+2/+3 turns after admission or at the final episode turn.

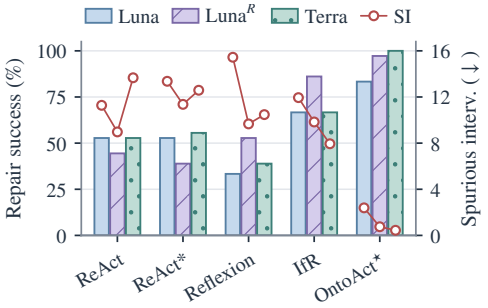


Figure 4: **Repair success and SI.** Medium Tacit-FLE, 36 tasks. Bars: success (left); markers: mean SI (right, lower is better). R: `medium` reasoning. Full metrics are in Table A15.

Does diagnosis narrow toward the true fault? OntoAct proposes fewer replacement targets than WALL-E and IfR, yet covers the true fault more often, overall and within its first three candidates (Table 2). Across scales, 71.4–90.0% of target-aligned hypotheses remain and 95.6–99.6% of off-target hypotheses are refuted (Figure 3; Table A27). Appendix D.7 details this refinement from observations and unsuccessful repairs.

Table 2: **Candidate count and fault coverage.** Medium Tacit-FLE (36 tasks); targets include unexecuted proposals. Coverage counts tasks with the true fault among all candidates or the first three.

	OntoAct	WALL-E	IfR
Distinct candidates/episode	3.36	5.14	9.00
Fault in candidate set	32/36	17/36	24/36
Fault in first 3	25/36	10/36	4/36

Does OntoAct benefit from reasoning and stronger backbones? OntoAct benefits from both: reasoning and Terra raise success from 83.3% to 97.2% and 100.0%, while SI falls from 2.39 to 0.75 and 0.44, respectively (69% and 81% reductions; Figure 4). ReAct and ReAct* achieve higher success with Terra than with added reasoning, whereas Reflexion and IfR benefit more from reasoning. Yet higher success can mask repeated failed repairs: with reasoning or Terra, these baselines still incur 7.9–13.7 spurious interventions per episode. OntoAct also reduces mean turns from 19.4 to 8.4 in both conditions (Table A15).

4.3 MARS

How well does OntoAct adapt to altered mechanics? OntoAct supports action under changed prerequisites, extending the framework's utility beyond hidden-fault diagnosis. It achieves the highest mean Reward and Score across the seven modified worlds (Table 3), although combined changes remain difficult. Inspecting NeSyC's low-scoring runs reveals overly restrictive learned conditions that block planning and trigger repeated waiting; Appendix D.2 details these failures.

Compared with ReAct*, the strongest baseline by mean Score, OntoAct achieves about 50% higher Score with 45% fewer evaluation tokens per environment step (0.84k versus 1.54k; Figure 5). Compact learned conditions shorten input contexts at similar LLM call rates. Appendix D.6 analyzes context usage and reports construction costs separately.

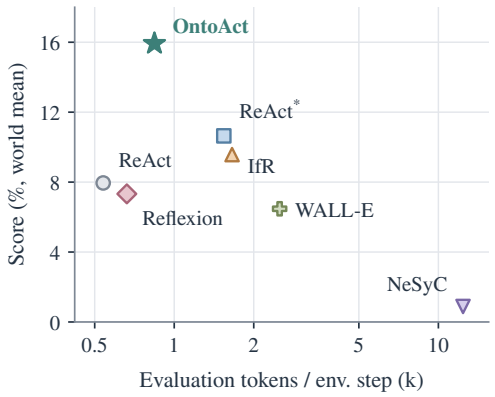


Figure 5: **Mars Score and evaluation tokens.** Luna, seven modified worlds. Equal world weights; input/output tokens exclude knowledge construction.

Table 3: **Mars Reward and Score by world.** Luna; five evaluation episodes per world. Reward combines achievement rewards and health changes (mean ± sample SD); Score geometrically aggregates achievement success rates. Average excludes Default; its Reward SD is computed across matched-seed world averages. Bold/underlined: best/second-best. Metric definitions are in Appendix E.3.

World	ReAct	ReAct*	Reflexion	IfR	NeSyC	WALL-E	OntoAct
				Reward ↑			
Default	10.10±4.80	9.30±1.64	7.90±2.95	8.10±2.35	0.70±0.55	6.30±6.06	**12.10±1.22**
Terrain	8.70±2.30	12.16±4.03	8.68±5.66	9.30±3.96	−0.90±0.00	7.50±3.78	**12.30±1.92**
Survival	7.10±4.00	9.50±3.44	6.30±5.07	7.30±4.82	0.98±0.99	6.30±4.97	**10.46±6.39**
Task dep.	4.10±1.00	2.90±1.30	4.10±2.92	7.30±1.10	0.90±0.45	3.90±0.84	**11.30±1.10**
Terr. surv.	6.10±2.35	**8.10±2.35**	5.10±2.92	6.10±4.30	2.70±1.82	3.90±3.11	6.30±3.03
Terr. task	3.50±1.67	6.90±2.17	6.48±2.56	3.90±2.05	1.10±1.00	2.90±1.64	**7.70±2.88**
Surv. task	**6.50±6.11**	2.10±2.74	3.70±2.41	**6.50±4.04**	2.30±1.30	5.70±5.18	5.50±6.02
All three	0.90±2.68	**1.50±2.61**	0.50±1.14	−0.10±1.30	−0.90±0.00	0.70±1.52	1.10±2.12
Average	5.27±1.10	6.17±1.31	4.98±1.40	5.76±2.14	0.88±0.37	4.41±1.91	**7.81±1.75**
				Score (%) ↑			
Default	**31.85**	13.54	13.67	11.74	0.69	15.26	29.70
Terrain	12.71	32.88	19.29	22.61	0.00	13.32	**34.34**
Survival	11.10	18.29	11.58	14.79	1.15	11.94	**25.83**
Task dep.	3.26	2.01	4.46	8.44	0.51	2.69	**20.73**
Terr. surv.	6.49	**10.52**	5.79	10.12	2.30	5.10	7.63
Terr. task	2.65	6.93	6.31	3.21	0.76	2.36	**10.17**
Surv. task	**17.65**	2.18	3.19	7.09	1.49	8.73	11.50
All three	**1.80**	1.69	0.67	0.74	0.00	1.19	1.27
Average	7.95	10.64	7.33	9.57	0.89	6.47	**15.92**

Does OntoAct benefit from more capable models? OntoAct's gains from reasoning and stronger backbones persist across environments, even though baseline responses change (Table 4). Unlike in Tacit-FLE, all displayed Mars baselines benefit from both changes; IfR narrows the gap under reasoning. OntoAct still retains the highest mean Score, supporting the framework's ability to build on model improvements in both diagnosis and action under altered mechanics. Full results appear in Appendix D.2.

Table 4: **Mars Score across model conditions.** Mean Score (%) over seven modified worlds. Bold/underlined: best/second-best per condition.

Method	Luna	Luna^R	Terra
ReAct	7.95	13.06	13.91
ReAct*	10.64	15.23	16.48
Reflexion	7.33	12.91	12.09
IfR	9.57	20.60	18.86
WALL-E	6.47	9.93	7.79
OntoAct (ours)	**15.92**	**22.99**	**21.57**

Do learned conditions guide actions before execution? OntoAct checks candidate actions against learned rules before execution. For resource conditions learned correctly by both methods, OntoAct redirects all 70 violating initial candidates before execution, whereas IfR executes 20 violating commands and replans after their failures (Table 5). The distinction is when knowledge changes behavior: OntoAct checks represented requirements against the current state before spending an environment interaction. None of its 136 issued commands violates these shared conditions. These traces cover matched knowledge on different trajectories; broader resource-rule coverage remains incomplete. Appendix D.4 details the shared-condition matching, trajectory analysis, and coverage limits.

Table 5: **Violations of shared learned conditions.** Resource conditions correctly stored by both methods. Luna, seven modified worlds, 35 episodes per method. Fractions use each method's eligible candidates or commands at the specified stage; Appendix D.4 details matching and denominators.

	OntoAct	IfR
Initial candidate violations	70/245 (28.6%)	20/107 (18.7%)
Redirected before execution	70/70 (100%)	0/20 (0%)
Issued command violations	0/136 (0%)	20/107 (18.7%)

4.4 ALFWORLD

Can OntoAct maintain high success with fewer LLM calls? OntoAct completes 132/134 tasks versus 129/134 for WALL-E, using 52.5% fewer LLM calls and 32.6% fewer evaluation tokens (Table 6). Most savings arise from OntoAct's symbolic condition checks versus WALL-E's extra LLM outcome-prediction calls (1.02 versus 1.95 calls per environment step); shorter trajectories also contribute. NeSyC's 26.1% uses disclosed initial object locations and inventory (†); Appendix D.5 compares observation conditions and LLM call roles. Task-family results and separate knowledge-construction costs appear in Appendices D.5 and D.6.

Table 6: **ALFWorld success and evaluation cost.** Luna. SR: success (%); mean environment Steps, LLM Calls, and Tokens (k) per episode. †Initial object locations and inventory provided.

Method	SR (%)	Steps	Calls	Tokens (k)
NeSyC†	26.1	38.6	9.3	281.32
WALL-E	96.3	14.8	28.8	47.70
OntoAct (ours)	98.5	13.5	13.7	32.16

5 RELATED WORK

Neuro-symbolic LLM agents. Neuro-symbolic agents keep what an LLM learns about an environment in a checkable form, as code world models and planning domains (Tang et al., 2024a; Khan et al., 2026; Zhu et al., 2025), dependency graphs (Nottingham et al., 2023), or rule bases (Chen et al., 2024), each refined against the agent's own interaction. WALL-E learns code rules and a state graph from mismatches between the predictions of an LLM world model and observed transitions, and lets the rules veto actions (Zhou et al., 2025b); NeSyC learns and continually repairs symbolic knowledge in answer-set form with a hypothesis-testing loop (Choi et al., 2025); IfR induces natural-language rules from experience in Mars (Tang et al., 2024b). A complementary line separates proposing from checking: solvers decide what an LLM formalizes (Pan et al., 2023; Olausson et al., 2023; Ye et al., 2023), and external critics or runtime guards check the plans and actions it proposes (Kambhampati et al., 2024; Chen et al., 2025; Xiang et al., 2025).

Acting with ontologies and knowledge graphs. Agents also use structured knowledge: ontologies mediate planning (John & Koopmann, 2024), scene and procedural graphs constrain LLM planners (Rana et al., 2023; Zhou et al., 2025a), knowledge graphs serve as memory (Anokhin et al., 2025) and as the reference against which LLM plans are verified (Petruzzellis et al., 2025), and such graphs are built from text, including maintenance records (van Cauter & Yakovets, 2024; Zhang & Soh, 2024). OntoAct joins these threads by inducing ontological world knowledge from inherited records, and Tacit-FLE evaluates the repairs that follow.

6 CONCLUSION

We introduced OntoAct, a hypothesis-based neuro-symbolic framework that connects learning from experience to action through ontological abstraction. It recovers reusable axioms over entity types, relations, and conditions from operational records. An LLM uses this knowledge to propose hypotheses and actions, while symbolic verification checks their support against current evidence. Observations and intervention outcomes continually refine hypothesis assessments and guide subsequent decisions. We also introduced Tacit-FLE to evaluate how agents combine inherited operational knowledge with observation and intervention to diagnose and repair hidden faults.

OntoAct achieves the highest overall repair success at every Tacit-FLE scale while substantially reducing spurious interventions. Reasoning and stronger backbones further increase success and reduce unsuccessful repairs. Results on Mars and ALFWorld extend these benefits to altered world mechanics and household tasks, demonstrating strong task performance alongside efficient interaction and inference. Our evaluation uses predefined ontology schemas and world mechanics that remain fixed during interaction. Future work will explore learning and extending these schemas from experience, and revising world knowledge as environments evolve. Together, these results show how reusable abstractions, explicit hypotheses, and ongoing verification connect accumulated experience to more successful and efficient action.

AI USE STATEMENT

In this work, we used generative AI tools to implement methods designed by the authors and to debug the authors' code. We have not used generative AI tools to develop the conceptual framework, to propose or refine the research hypotheses, or to generate the synthetic datasets, which are produced by our benchmark generator; mathematical proofs, translation, dataset cleaning, and qualitative data analysis are not applicable to this work. Additionally, we used generative AI tools to draft and edit parts of the paper, to create and modify figures, to identify and summarize related literature, to format references, and to write and edit software code. We have reviewed all AI-assisted work: code was verified and tested by the authors, reported numbers were checked by the authors against the experiment outputs, and cited works were checked against their sources. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

All experiments run in simulators on synthetic data. The maintenance corpus of Tacit-FLE is generated and contains no record of any real plant, person, or organization. The work targets industrial maintenance, where an agent's interventions can carry physical risk; the verifier of OntoAct is a step toward auditability, not a safety guarantee, and we make no claim of readiness for deployment.

REPRODUCIBILITY STATEMENT

The benchmark generator, the agent, and every baseline port, with the seeds and prompts used in the experiments, will be released with the camera-ready version. Appendix A specifies the generator, Appendix C the bindings and baselines, and Appendix E the settings, budgets, and metric definitions behind each number.

REFERENCES

Petr Anokhin, Nikita Semenov, Artyom Sorokin, Dmitry Evseev, Andrey Kravchenko, Mikhail Burtsev, and Evgeny Burnaev. AriGraph: Learning knowledge graph world models with episodic memory for LLM agents. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 12–20, 2025. doi: 10.24963/ijcai.2025/2. URL https://www.ijcai.org/proceedings/2025/2.

Minghao Chen, Yihang Li, Yanting Yang, Shiyu Yu, Binbin Lin, and Xiaofei He. AutoManual: Constructing instruction manuals by LLM agents via interactive environmental learning. In *Advances in Neural Information Processing Systems*, volume 37, pp. 589–631, 2024. doi: 10.52202/079017-0019. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/0142921fad7ef9192bd87229cdafa9d4-Paper-Conference.pdf.

Zhaorun Chen, Mintong Kang, and Bo Li. ShieldAgent: Shielding agents via verifiable safety policy reasoning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 8313–8344, 2025. URL https://proceedings.mlr.press/v267/chen25ae.html.

Wonje Choi, Jinwoo Park, Sanghyun Ahn, Daehee Lee, and Honguk Woo. NeSyC: A neuro-symbolic continual learner for complex embodied tasks in open domains. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/d569e4655e2835e3d38310b67c5ad646-Paper-Conference.pdf.

Johan de Kleer and Brian C. Williams. Diagnosing multiple faults. *Artificial Intelligence*, 32(1): 97–130, 1987. doi: 10.1016/0004-3702(87)90063-4.

Johan de Kleer and Brian C. Williams. Diagnosis with behavioral modes. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1324–1330, 1989. URL https://www.ijcai.org/Proceedings/89-2/Papers/076.pdf.

Jack Hopkins, Mart Bakler, and Akbir Khan. Factorio Learning Environment. In *Advances in Neural Information Processing Systems, Datasets and Benchmarks Track*, volume 38, 2025. doi: 10.52202/085713-2745. URL https://doi.org/10.52202/085713-2745.

Tobias John and Patrick Koopmann. Planning with OWL-DL ontologies. In *ECAI 2024*, Frontiers in Artificial Intelligence and Applications. IOS Press, 2024. doi: 10.3233/FAIA240988. URL https://doi.org/10.3233/FAIA240988.

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Paul Saldyt, and Anil B Murthy. Position: LLMs can't plan, but can help planning in LLM-modulo frameworks. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 22895–22907, 2024. URL https://proceedings.mlr.press/v235/kambhampati24a.html.

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*, pp. 99794–99836, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/a24d08f2eecc6c96f4a0ad37a74ddeca-Paper-Conference.pdf.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Kolby Nottingham, Prithviraj Ammanabrolu, Alane Suhr, Yejin Choi, Hannaneh Hajishirzi, Sameer Singh, and Roy Fox. Do embodied agents dream of pixelated sheep: Embodied decision making using language guided world modelling. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pp. 26311–26325, 2023. URL https://proceedings.mlr.press/v202/nottingham23a.html.

Theo Olausson, Alex Gu, Ben Lipkin, Cedegao Zhang, Armando Solar-Lezama, Joshua Tenenbaum, and Roger Levy. LINC: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5153–5176, 2023. doi: 10.18653/v1/2023.emnlp-main.313. URL https://aclanthology.org/2023.emnlp-main.313/.

OpenAI. GPT-5.6 Luna. OpenAI API model documentation, 2026a. URL https://developers.openai.com/api/docs/models/gpt-5.6-luna. Accessed September 19, 2026.

OpenAI. GPT-5.6 Terra. OpenAI API model documentation, 2026b. URL https://developers.openai.com/api/docs/models/gpt-5.6-terra. Accessed September 19, 2026.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Wang. Logic-LM: Empowering large language models with symbolic solvers for faithful logical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 3806–3824, 2023. doi: 10.18653/v1/2023.findings-emnlp.248. URL https://aclanthology.org/2023.findings-emnlp.248/.

Flavio Petruzzellis, Cristina Cornelio, and Pietro Liò. Hierarchical planning for complex tasks with knowledge graph-RAG and symbolic verification. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, 2025.

Michael Polanyi. *The Tacit Dimension*. Doubleday, Garden City, NY, 1966.

Krishan Rana, Jesse Haviland, Sourav Garg, Jad Abou-Chakra, Ian Reid, and Niko Suenderhauf. SayPlan: Grounding large language models using 3D scene graphs for scalable robot task planning. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 23–72, 2023. URL https://proceedings.mlr.press/v229/rana23a.html.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2010.03768.

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, 2024a. doi: 10.52202/079017-2243. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/820c61a0cd419163ccbd2c33b268816e-Paper-Conference.pdf.

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*, 2024b. URL https://arxiv.org/abs/2410.08126.

Zeno van Cauter and Nikolay Yakovets. Ontology-guided knowledge graph construction from maintenance short texts. In *Proceedings of the 1st Workshop on Knowledge Graphs and Large Language Models (KaLLM)*, 2024. URL https://aclanthology.org/2024.kallm-1.8/.

Venkat Venkatasubramanian, Raghunathan Rengaswamy, Kewen Yin, and Surya N. Kavuri. A review of process fault detection and diagnosis: Part i: Quantitative model-based methods. *Computers & Chemical Engineering*, 27(3):293–311, 2003. doi: 10.1016/S0098-1354(02)00160-6.

Zhen Xiang, Linzhi Zheng, Yanjie Li, Junyuan Hong, Qinbin Li, Han Xie, Jiawei Zhang, Zidi Xiong, Chulin Xie, Carl Yang, Dawn Song, and Bo Li. GuardAgent: Safeguard LLM agents via knowledge-enabled reasoning. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 68316–68342, 2025. URL https://proceedings.mlr.press/v267/xiang25a.html.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023.

Xi Ye, Qiaochu Chen, Isil Dillig, and Greg Durrett. SatLM: Satisfiability-aided language models using declarative prompting. In *Advances in Neural Information Processing Systems*, volume 36, pp. 45548–45580, 2023. doi: 10.52202/075280-1974. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/8e9c7d4a48bdac81a58f983a64aaf42b-Abstract-Conference.html.

Bowen Zhang and Harold Soh. Extract, define, canonicalize: An LLM-based framework for knowledge graph construction. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 9820–9836, 2024. doi: 10.18653/v1/2024.emnlp-main.548. URL https://aclanthology.org/2024.emnlp-main.548/.

Jiacong Zhou, Jiaxu Miao, Xianyun Wang, and Jun Yu. Enhancing LLM planning for robotics manipulation through hierarchical procedural knowledge graphs. In *Advances in Neural Information Processing Systems*, volume 38, pp. 127466–127495, 2025a. doi: 10.52202/085713-4246. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/b94310e1c7ecb79f1a24adc757f1b89b-Paper-Conference.pdf.

Siyu Zhou, Tianyi Zhou, Yijun Yang, Guodong Long, Deheng Ye, Jing Jiang, and Chengqi Zhang. WALL-E: World alignment by neurosymbolic learning improves world model-based LLM agents. In *Advances in Neural Information Processing Systems*, volume 38, pp. 65352–65391, 2025b. doi: 10.52202/085713-2192. URL https://doi.org/10.52202/085713-2192.

Wang Bill Zhu, Ishika Singh, Robin Jia, and Jesse Thomason. Language models can infer action semantics for symbolic planners from environment feedback. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 8751–8773, 2025. doi: 10.18653/v1/2025.naacl-long.440. URL https://aclanthology.org/2025.naacl-long.440/.

A TACIT-FLE: SPECIFICATION, EXAMPLES, AND VALIDATION

This appendix specifies the benchmark introduced in Section 2. **GitHub repository:** *will be released.*

Section	Contents
A.1	Agent inputs, observations, and actions
A.2	Plant generation and reference corpus construction
A.3	Illustrative examples of the three task types
A.4	Evaluation population and scoring
A.5	Benchmark construction checks

A.1 TASK AND INFORMATION AVAILABLE TO THE AGENT

A world $w = (P, D)$ pairs a plant P with a fixed reference corpus D. An episode $e = (w, c^*, B)$ introduces one hidden fault c^* and allows B interaction turns. The agent receives the corpus, a task brief, a unit roster, and observations of the faulty plant. The agent interacts with the plant through Tacit-FLE's text commands for inspection and repair. Tacit-FLE runs on the Factorio server infrastructure of FLE and adds a plant generator, fault injection, a maintenance corpus, a text command interface, and repair evaluation. Whereas the troubleshooting suite of FLE evaluates spatial error identification in existing factories, Tacit-FLE evaluates diagnosis and executed repairs from finite maintenance records.

A.1.1 OBSERVATIONS AND ANONYMIZATION

The task brief gives the turn budget and the plant's normal production rate in final products per minute; the roster lists unit kinds and positions. A public status board reports machine run states, storage totals and changes, elapsed game time, and shipping counts. Power and material-transfer devices require inspection, while opening a unit returns itemized contents. Conjunction also exposes the configured output for comparison with its operating plan. Table A1 separates available information from evaluator metadata. Figure A2 illustrates the initial input and a subsequent command response.

Table A1: **Information boundary.** Public records can disclose selected planned facts; the complete original specification remains evaluator metadata.

Source	Available information
Initial task	Turn budget, normal production rate (final products/minute), unit kinds, and coordinates.
Inherited records	Local incident observations, checks, repairs, and outcomes; buffer policy for Chain; planned outputs for Conjunction.
Current observations	Machine states, storage totals and changes, shipping; inspections and itemized readings on request.
Repair feedback	Episode termination after an accepted repair; continued interaction otherwise.
Evaluator only	Injected fault, accepted repair targets, full original plant specification, and coverage annotations.

Items, entities, recipes, and fluids are anonymized with consistent world-specific labels across records and observations. Unit kinds, positions, structural suffixes, and measured states remain visible. This reduces reliance on familiar Factorio names while retaining spatial and operational cues.

A.1.2 ACTIONS AND REPAIR FEEDBACK

In Factorio, production evolves with simulation time: machines process inputs, materials move between components, and inventories and shipping counts change. Table A2 lists the available commands and the game time each advances. A replacement restores a unit to its original specification,

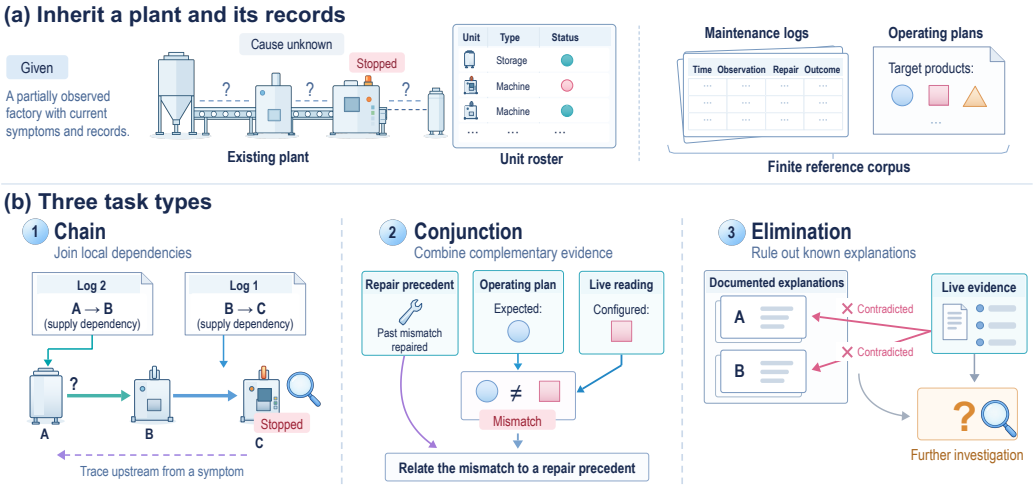


Figure A1: **Tacit-FLE.** (a) An agent inherits an existing plant, a finite reference corpus of maintenance logs and operating documents, and observations of a new fault. (b) Chain links dependencies across incidents; Conjunction combines repair precedents, plans, and observations; Elimination rules out known explanations contradicted by current evidence.

Task brief and unit roster

```
Contract throughput:  about 2 final products per minute.
st-04 @(150.5,192.5)
st-05 @(164.5,164.5)
```

Initial observation

```
[STATUS] first reading
machines:  mat-078 running, mat-214 running, st-00 stopped, st-01
stopped, st-02 stopped, st-03 stopped, st-04 stopped, st-05 running
Shipping buffer ship cumulative 8 (mat-136 x8).  (elapsed 0s)
```

Agent command

```
open st-05
```

Environment response

```
st-05 opened:  input buffer empty.  configured output:  mat-232.
[STATUS] change over the last 10s
Shipping buffer ship cumulative 10 (mat-136 x10).  (elapsed 60s)
[SHIFT] not fixed yet; the shift goes on.
```

Figure A2: **Example interaction in Tacit-FLE.** A medium-scale Conjunction episode: the task brief, roster, initial status, and a later `open` command with its response. Only the selected lines are shown; identifiers and wording are preserved.

including its configuration; a refill specifies the storage, item, and amount. Valid actions advance the simulation by the durations in Table A2, and each response reflects the updated plant state. Malformed commands consume a turn without advancing the plant. An accepted repair ends the episode; otherwise interaction continues and supplies further evidence.

A.2 Constructing plants and inherited records

A common seeded specification ties the simulated plant to its synthetic records. Construction has four components:

1. **Generate the plant.** Select target products according to the scale profile, then trace Factorio's recipe graph backward to identify the required intermediate products and raw inputs. Compile

Table A2: **Agent actions in the evaluated protocol.** Each action uses one turn. The plant advances by the indicated game time; LLM deliberation itself is not charged as game time. The status board is included in observations at no additional turn cost.

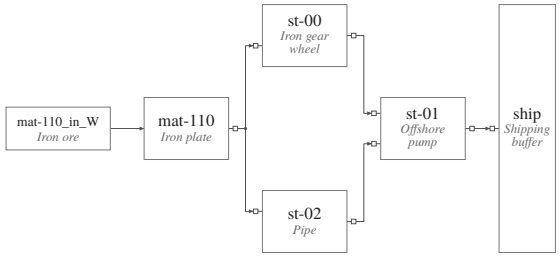
Command	Game seconds	Effect or returned information
`inspect <unit>`	5	External state; power and material-transfer device readings require this probe.
`open <unit>`	10	Itemized storage or input contents; configured output in Conjunction.
`wait <seconds>`	specified	Advance the plant and receive an updated status board.
`replace <unit>`	15	Restore the addressed unit to its original specification.
`refill <storage> <item> <count>`	10	Insert the selected item into storage; success also depends on the repair target and item.

these dependencies into a spatial layout of machines, input and output storage, material-transfer devices, and power infrastructure. The world seed controls product selection and layout choices, including shared production stages and spatial spread.

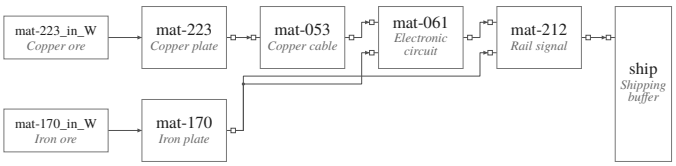
2. **Establish normal operation.** Build the plant in Factorio and measure positive healthy shipping.
3. **Introduce a fault.** Inject a supply cut for Chain or an off-plan machine configuration for Conjunction and Elimination. Cap downstream inventory so it cannot mask the fault indefinitely, and require stalled shipping or a persistent telemetry deviation. Other product lines may continue operating.
4. **Construct the reference corpus.** Render historical incidents and operating documents from the same specification, withhold evaluation incidents, and check the intended distribution of evidence.

Appendix A.5 summarizes the checks applied to generated plants and corpora.

Example plant topologies. Figures A3–A4, A5–A6, and A7–A8 show small, medium, and large plants, respectively. Four selected examples per scale illustrate variation in production depth, branching, and shared intermediate stages. Arrows indicate material flow; dots join branches, and curved crossings are unconnected. Small squares denote input and output buffers. Positions are rearranged for readability, and utilities are omitted. Anonymized unit identifiers are shown above original names in italics, which are included only as a reading aid.

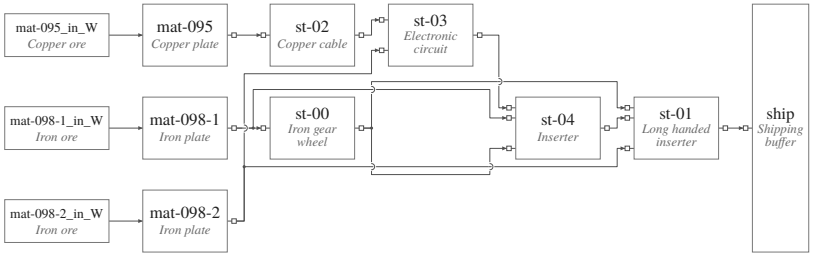


(a) World 47728

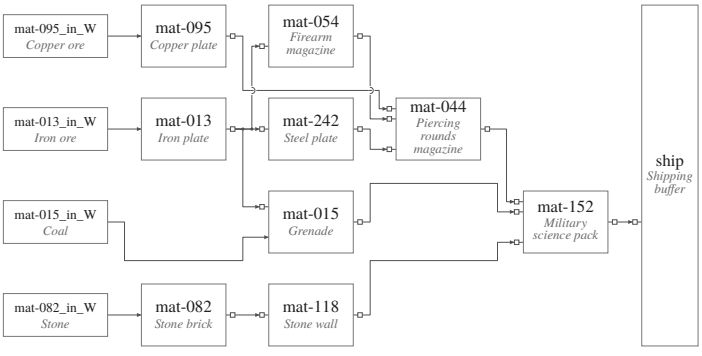


(b) World 22970

Figure A3: **Small-scale plant examples (a)–(b).** Each plant has one target product. One input stream splits and reconverges in (a), while (b) combines iron and copper through a circuit stage.

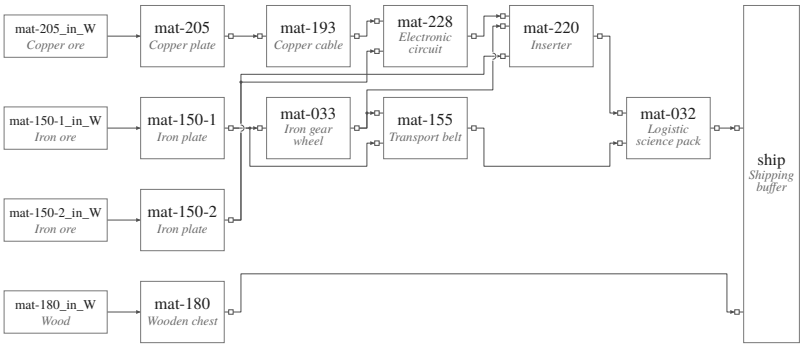


(c) World 62689

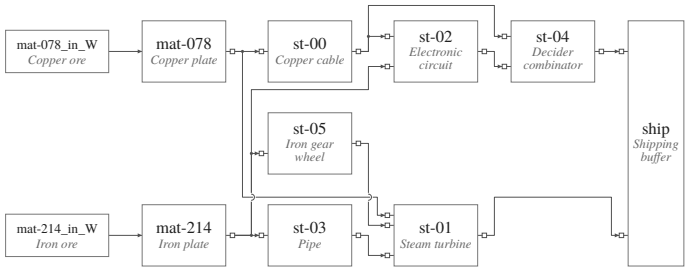


(d) World 50676

Figure A4: **Small-scale plant examples (c)–(d).** Each plant has one target product. A deeper assembly sequence in (c) contrasts with several material branches converging on one science pack in (d).

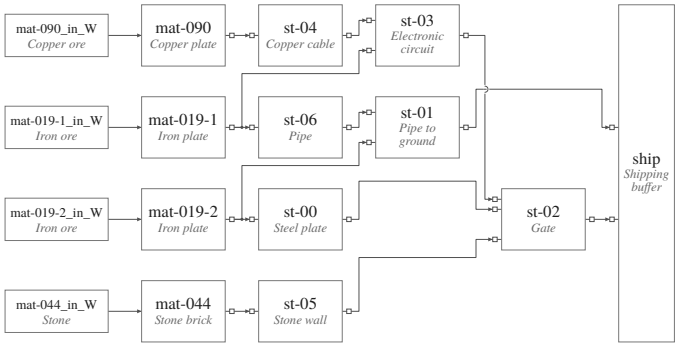


(a) World 14275

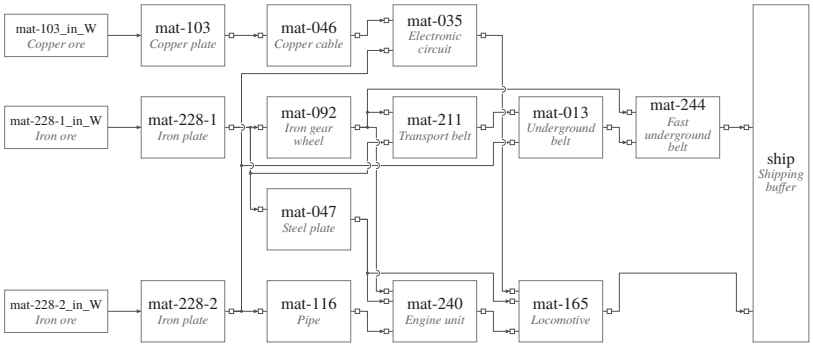


(b) World 73144

Figure A5: **Medium-scale plant examples (a)–(b).** Each plant has two target products. An independent short production line accompanies the deeper chain in (a); the two targets reuse upstream products in (b).

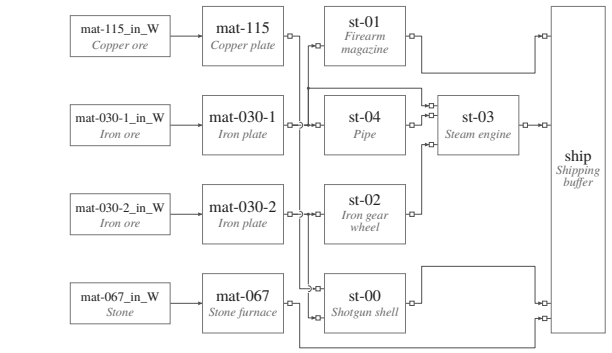


(c) World 22606

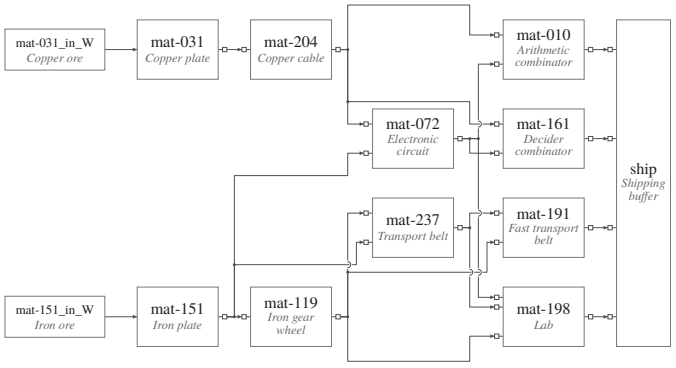


(d) World 16060

Figure A6: **Medium-scale plant examples (c)–(d).** Each plant has two target products. Parallel material branches in (c) contrast with the longer transport and engine chains in (d).

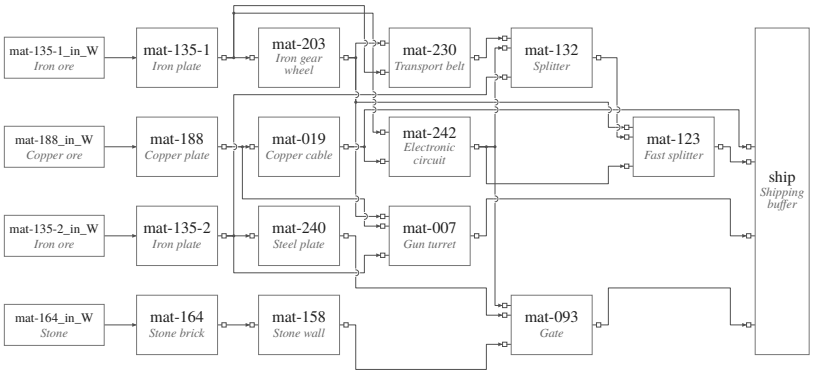


(a) World 5246

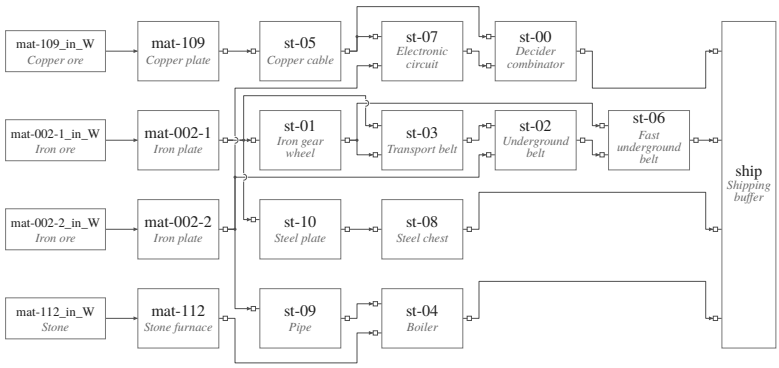


(b) World 44309

Figure A7: **Large-scale plant examples (a)–(b).** Each plant has four target products. Shorter parallel paths in (a) contrast with the reuse of a compact upstream circuit and belt network in (b).



(c) World 64320



(d) World 20024

Figure A8: **Large-scale plant examples (c)–(d).** Each plant has four target products. Direct shipping accompanies further assembly in (c), while (d) contains fifteen processing units across several intermediate stages.

A.2.1 CORPUS CONTENTS AND PROVENANCE

Each maintenance log contains five fields: `OCCURRED`, `OBSERVED`, `CHECKED`, `ACTION`, and `RESULT`. These describe local symptoms, checks, and repairs, leaving relations and conditions to be inferred across records. The corpus also includes explicit operating information: Chain supplies a buffer-policy note, and Conjunction supplies planned outputs.

Three construction details matter for interpreting these records:

- **Synthetic provenance.** Historical incidents are rendered from templates over the same plant specification.
- **Document counts.** Required incidents and distractors are supplemented to the configured corpus size; a recurring incident may appear under separate log identifiers. Chain can add dependency records and its operating note afterward. Table A6 reports the resulting document counts, including these additions and repetitions.
- **Held-out faults.** No registered test fault appears as a historical repair target in the shared corpus. The same unit may still appear as a downstream observation, roster entry, or planned-output entry. Records are shuffled, and the Conjunction plan lists applicable machines rather than singling out the faulty one.

A.3 THREE TASK TYPES: ILLUSTRATIVE EXAMPLES

Tables A3–A5 pair excerpts from the maintenance logs and operating documents with environment responses to the agent's commands. Each table pairs original document and response excerpts with an interpretation of their diagnostic relevance.

- **Chain: compose dependencies.** Local records connect successive units. A separate buffer-policy note distinguishes normally empty internal buffers from yard-stocked intakes whose lack of incoming material indicates a supply cut.
- **Conjunction: combine complementary evidence.** A prior incident supplies a repair pattern, a plan supplies the expected output, and a current reading supplies the actual output. The mismatch is unit-specific: an item label alone does not indicate a fault.
- **Elimination: test alternative explanations.** Records describe destroyed units, supply cuts, and power failures, but omit the off-plan configuration encountered in evaluation. Rejecting documented explanations narrows the search; further observations or interventions may still be needed to distinguish remaining candidates.

Table A3: **Chain: an illustrative example.** Record excerpts and command responses from a medium-scale evaluation case.

Evidence source	Exact excerpt
Log log_6056	`[OBSERVED] mat-193 stopped, input buffer empty.` `[CHECKED] site visit — mat-205 found destroyed.`
Log log_7299	`[OBSERVED] mat-228 stopped, input buffer empty.` `[CHECKED] site visit — mat-193 found destroyed.`
Ops note buffer-policy	`[NORM] Line-internal feed buffers are pulled just-in-time and normally sit empty; an empty internal buffer during a stall is expected, not evidence.` `[NORM] Raw-material intake buffers are kept stocked by the yard at all times. An intake buffer found empty with nothing arriving means its supply feed is severed — replace that storage.`
Command responses	`mat-205_in_W opened: empty. flat at zero.`
Interpretation	The two logs connect mat-205 to mat-193 and mat-193 to mat-228. The operating note distinguishes an intake supply cut from normal internal-buffer emptiness.

Table A4: **Conjunction: an illustrative example.** Record excerpts and command responses from a medium-scale evaluation case.

Evidence source	Exact excerpt
Log log_5841	`[CHECKED] opened st-04 — configured output does not match the plan.` `[ACTION] replaced st-04 with a fresh unit (as-built configuration restored).`
Plan norm_3982	`- unit st-05: planned output mat-231`
Command responses	`st-05 opened: input buffer empty. configured output: mat-232.`
Interpretation	A repair precedent on st-04 applies to a different unit: st-05 is planned to output mat-231 but currently outputs mat-232.

Table A5: **Elimination: an illustrative example.** Record excerpts and command responses from a medium-scale evaluation case.

Evidence source	Exact excerpt
Log log_7360	`[CHECKED] site visit — mat-027 found destroyed.`
Log log_8680	`[CHECKED] opened storage mat-070_in_W — found empty, no inbound material.`
Log log_3569	`[OBSERVED] multiple machines in the zone stopped at once.` `[CHECKED] zone-wide simultaneous stoppage — power distribution loss confirmed.`

Command responses	`mat-027 opened: input buffer mat-089 x5, mat-074 x3.`
	`mat-027: old unit removed, replacement installed to plan.`
	`[SHIFT] not fixed yet; the shift goes on.`
	`st-00 opened: input buffer empty.`
	`mat-070: running.`

Interpretation	Replacing mat-027 did not end the episode, which rejects the destroyed-unit explanation of log_7360. A running machine (mat-070) is incompatible with the zone-wide power loss of log_3569. The stopped st-00 matches none of the recorded incidents and becomes the unit to investigate.

A.4 Evaluation population and scoring

Each combination of scale and task type has ten worlds: the first five are reserved for development, and the last five for evaluation. All registered fault positions in the evaluation worlds are tested, giving 45 world–type corpora and 104 episodes (Table A6). For each world, we rebuild the same plant and inject each registered fault in a separate episode, keeping its reference corpus fixed. Small, medium, and large use one, two, and four target products, respectively; corpus size, realized topology, task mix, and turn budgets also vary.

Table A6: **Evaluated Tacit-FLE profiles.** Document counts include maintenance logs and operating documents in each corpus. Episodes correspond to the specified fault positions; B is the turn budget. Scale conditions vary jointly in target products, corpus sizes, and interaction budgets.

Scale	Products	Type	Worlds	Documents/world	Episodes	B
Small	1	Chain	5	7–16	13	30
		Conjunction	5	5	7	30
		Elimination	5	4–5	8	30
Medium	2	Chain	5	32–40	14	60
		Conjunction	5	30	14	60
		Elimination	5	30	8	60
Large	4	Chain	5	63–72	17	75
		Conjunction	5	60	11	75
		Elimination	5	60	12	75

A method may construct reusable knowledge from the corpus or retrieve records according to its evaluation procedure. No additional exploratory training episodes are supplied. Corpus-derived knowledge may be shared across a world's episodes, but test observations, hypotheses, and outcomes are reset each time. The no-corpus control omits the records; method-specific access is detailed in Appendix C.2.

A.4.1 Repair criterion and metrics

The evaluator retains an accepted repair-target set that is hidden from the agent. Replacing an accepted target completes the episode. A refill must address an accepted target, be accepted by the environment, and use the planned supply item when specified. Otherwise the attempt consumes a turn and the episode continues, until a repair succeeds or the budget is exhausted.

- **Repair success:** the fraction of scheduled evaluation episodes with an accepted repair within the turn budget.
- **Turns:** mean interaction turns, including the full budget for unsuccessful episodes.
- **Spurious interventions:** mean executed replacements or refills per episode that fail the repair criterion, including an unsuccessful refill of an otherwise relevant unit.

LLM calls and tokens are accounted for separately from environment turns (Appendix D.6).

A.5 Benchmark construction checks

These checks assess whether generated cases realize the intended task type: an operating plant develops an observable fault, and its corpus provides the specified evidence. Table A7 summarizes the checks on plant behavior, corpus construction, and reproducibility.

Table A7: **Benchmark construction checks.** The checks cover simulated plant behavior, corpus conditions, and reproducible evaluation inputs.

Check	Evidence	What it establishes
Plant behavior	Healthy production, followed by stalled shipping or a persistent change in operating readings after fault injection.	The injected fault has an observable effect on an operating plant.
Corpus construction	Required document fields and vocabulary, declared evidence coverage, and exclusion of test-fault repair records.	Compliance with the corpus construction conditions.
Reproducibility	Exact regeneration of all 45 evaluation corpora.	Consistent corpus inputs for the 104 evaluation episodes.

B Method Details

B.1 Representation and information accounting

The world knowledge is $K_W = (T, A_W, R_W)$, with an ontology schema $T = (\mathcal{C}, \mathcal{R}, \mathcal{P}, \Theta)$. Here $\mathcal{C}$ is the set of entity types, $\mathcal{R}$ the relations, $\mathcal{P}$ the observable attributes, and Θ the admissible axiom templates with their evaluation procedures. The induced axioms comprise ground assertions in A_W, which preserve particular entities and their connections, and conditional axioms in R_W. Equation 2 combines these with the episode assertions A_t; Algorithm 2 uses the same tuple with the within-turn variables A and K. The templates differ by environment, as listed in Table A13. The predicate Typed in Equation 4a checks conformance to this vocabulary and its admissible axiom forms; empirical support is evaluated separately.

An induced element retains a provenance set $\pi \subseteq D$ identifying its source records. Some sources are attached automatically by the induction procedure, while other templates require the LLM to cite record identifiers. Source association provides traceability; the additional support and compatibility checks are described below. Table A8 distinguishes fixed representation choices from induced knowledge and current observations. The experimental corpus and observation conditions, including the separately reported observation control, are specified in Appendix E.1.

Table A8: **Information accounting for OntoAct.** The origins and roles of information used in construction and interaction.

Origin	Content	Role
Task inputs	Reference corpus, task description, observations, action interface	Use and any observation controls follow the experimental protocol.
Given representation	Schema T, observation map ρ, template semantics	Method-specific representation and interpretation choices; fixed before learning.
Induced knowledge	A_W, R_W, source records $\pi \subseteq D$	Constructed from the reference corpus and fixed during evaluation.
Episode evidence	A_t	Grounded in public observations; reset for each episode.
LLM proposals	Action a; structural and diagnostic claims where supported	Checked against the current knowledge and evidence.

Knowledge lifetime. The corpus-derived world knowledge is reused without learning from test episodes. Each episode starts from that knowledge and obtains a fresh state A_t from observations. Historical status readings are not copied into the live state; roster information can correct an entity's

type in the episode's working representation. The distinction between world knowledge and episode state concerns lifetime and evidential scope. Both may contain assertions about individuals.

B.2 KNOWLEDGE CONSTRUCTION AND ADMISSION

Shared procedure. This section expands Algorithm 1 and Equation 4b into the checks used to admit corpus-derived knowledge in each environment. f_{abs} in Equation 3 denotes the proposal procedure, including the association of each candidate with its source records. The admission indicator Admissible in Equation 4b requires schema and source validity (β_i), then applies the constraints $\mathcal{C}_x$ supported by the knowledge type. The indicator $\mathbf{1}[P]$ converts a satisfied condition to one and a failed condition to zero; the product in Equation 4b is one exactly when every applicable criterion holds. The membership and semantics of $\mathcal{C}_x$ differ by knowledge type: source-link requirements, structural consistency, empirical support, and contradiction tolerances are specified below. f_{corpus} denotes validation against the assembled records: some checks run during admission, while others require a later pass over the assembled knowledge. Bounded correction requests within knowledge construction are included in the proposal procedure. The deterministic checks run without LLM calls.

Tacit-FLE: structure and diagnostic patterns. The LLM processes each reference document using the supplied vocabulary and proposes entities, structural relations, diagnostic patterns, and non-evidence axioms. Entity types and relation names are closed; relation domain and range constraints and incompatible component types are checked when proposals enter the store. The current document identifier is attached to its extracted facts and patterns. A diagnostic pattern specifies the symptom type and status, the cause's relation to that symptom, optional cause type and readings, and a repair action when the documented intervention has a corresponding command. For example, an incoming-flow condition is retained only when the source record supplies it. Relations preserve particular component identities, whereas the diagnostic pattern refers to types and roles.

When repair observations can be extracted from the cited records, a support check tests whether at least one cited incident instantiates the pattern's symptom–cause relationship. An unsupported pattern is returned to the LLM once with the discrepancy; a revised proposal that still fails is discarded. Operating documents without a repair incident are not evaluated by this repair-specific test. After assembling the corpus, a further check removes self-cause patterns that would confirm themselves from the symptom alone when a recorded incident shows that symptom but documents a repair of another component.

Tacit-FLE: operating information. Operating practices and service-count samples are elicited from the joined corpus because their evidence can span documents. Practice templates express a forbidden action under a stated condition, a check before an intervention, or a preference between actions. They require a valid typed form and nonempty citations drawn from the presented corpus, but do not undergo the repair-pattern support test. Non-evidence axioms encode explicit operating notes that make a status non-diagnostic for a component class, with any stated exceptions. Service samples are checked against their cited records and aggregated by production duty into an empirical prior over usage counts.

Mars. The LLM reads the reference transitions and proposes typed action conditions and effects for collection, crafting, placement, movement, drinking, creatures, and material neighborhoods, together with blocked-action conditions. Admission checks the schema and requires at least one cited transition concerning the subject of the proposed axiom. Applicable recorded transitions are then checked for outcomes or effects that contradict it. The experiment configuration allows at most one contradictory transition; blocked-action rules additionally require recorded unsuccessful attempts without a recorded success under the claimed condition. This compatibility check is performed during admission.

ALFWorld. Records are grouped by action and processed in batches of at most 24 transitions. The LLM proposes necessary action conditions over five supplied predicates: being at the target, holding the object, having empty hands, the object's presence at the target, and the target's openness. Every cited index must identify a record of the corresponding action. At least one cited successful transition must observe and satisfy all proposed conditions, and no successful transition in the corpus may contradict them. These checks establish positive support and compatibility with recorded successes.

B.3 DIAGNOSTIC HYPOTHESES AND VERIFICATION

Hypothesis representation. In the diagnostic variant, a hypothesis is written as $h = (c, e, r, \Phi, \sigma, \pi, \alpha)$. Its fields are a candidate cause c, the anomaly e it explains, a referenced pattern r, expected observational atoms Φ, required structural relations σ, cited evidence π, and a prescribed repair α. The pattern can refer to an induced diagnostic pattern or aggregate prior, or describe an explanation proposed by the LLM beyond those patterns. An atom names an entity, an observable attribute, a comparison, and an expected value.

Admission of episode claims. f_{admit} checks the structured claims proposed during interaction. Named entities and anomalies must exist in the current representation, the expected attributes must be readable, and required relations must already be established. A newly proposed relation must first be supported by a permitted basis, such as documented structure, layout connectivity, or observed contents. For a pattern-backed diagnosis, admission also checks the correspondence between the proposed cause, the symptom, and the conditions of that pattern. A hypothesis cannot cite a pattern requiring no incoming material while omitting the corresponding expected reading. An operating note that makes a status non-evidence can invalidate a claim relying on that status.

Three-valued evaluation. The observation map ρ grounds readings in the current state, and an atom evaluates as

$$[\![\varphi]\!]_{K_t} \in \{\top, \bot, ?\}.$$

The values denote support, contradiction, and an unresolved reading, respectively. Missing readings remain unknown. For a current status that disagrees with the claim, the implementation also considers recent readings and can leave an isolated fluctuation unresolved. An open, admitted hypothesis is refuted when an atom evaluates to $\bot$, confirmed when all its atoms evaluate to $\top$, and remains open otherwise. For open hypotheses, unresolved atoms are mapped to available probes and included in the context supplied to the LLM. Confirmation records agreement with the specified observations.

Hypotheses beyond recorded patterns. The LLM may propose an explanation for a cause not covered by an induced pattern, provided that its claim meets the structural and observational admission requirements. The reasoning context also lists recorded explanations that still owe a reading, open rival hypotheses, and the stopped units that no recorded pattern explains; an explanation outside the recorded patterns is marked as supported when its cause is one of these unexplained units.

Checking repairs and incorporating outcomes. A repair is backed when an admitted LLM-proposed hypothesis is confirmed, its nonempty set of atoms still holds, it prescribes the repair verb, and the target is named by the hypothesis's cause or one of its atoms. Observation actions do not require a confirmed repair hypothesis; operating practices and repetition checks can nevertheless provide feedback on proposed commands. If an executed repair targets a hypothesis's candidate cause and the episode continues, that outcome refutes the hypothesis on the next observation. A command refused by the environment is not counted as an executed repair. Confirmed hypotheses can also be refuted by subsequent contradictory readings or by the disappearance of the anomaly they explained. The LLM remains responsible for selecting a cause and an action from this evidence. When the proposal limit is reached, the final proposal and its verification verdict are passed to the executor.

B.3.1 EXAMPLE: FROM CORPUS RECORDS TO ACTION

This example starts with a small reference corpus, shows the knowledge constructed from it, and follows its use in a new episode. The records and observation branches are condensed, fictional examples.

Step 1: read the records and construct world knowledge. Suppose the agent receives the following two documents, $D = \{d_1, d_2\}$, before evaluation.

Reference corpus.

d_1**: Line A handover note.** "Storage s_1 supplies iron plates to machine m_1. Mover q_1 delivers plates into s_1 and is its only incoming connection."

d_2**: Line A maintenance record.** "Machine m_1 had stopped. Its supplying storage s_1 contained no iron plates. Inspection of its sole incoming mover q_1 found no material arriving. We added iron plates to s_1; m_1 then resumed production."

The ontology schema T provides the types `Storage`, `Machine`, and `Mover`, the relation `feeds`, and the observable attributes. The LLM proposes a particular supply relation and a reusable diagnostic pattern. The relation has compatible endpoint types; the pattern's cited incident matches its stopped-machine symptom, supplying-storage cause, and refill intervention. After the applicable admission checks, $K_W = (T, A_W, R_W)$ retains

$$\begin{aligned} A_W &\supseteq \{\text{feeds}(s_1, m_1)\} && \text{supported by } d_1, d_2, \\ R_W &\ni r && \text{with source } \pi_r = \{d_2\}. \end{aligned}$$

The content of r makes the reusable roles explicit:

$$\begin{aligned} \text{symptom: } & m : \text{Machine is stopped}, \\ \text{candidate cause: } & s : \text{Storage is depleted}, \\ \text{required relation: } & \sigma_r(s, m) = \{\text{feeds}(s, m)\}, \\ \text{expected readings: } & \Phi_r(s) = \{\text{stock}(s) = 0,\ \text{inflow}(s) = 0\}, \\ \text{prescribed repair: } & \texttt{refill}(s). \end{aligned}$$

The concrete identities remain in A_W, while r replaces them with roles that can match another storage–machine pair. The empty-stock and no-inflow conditions come from d_2; they specify what to check when proposing the same explanation for a new pair. Here and below, $\text{inflow} = 0$ means that no incoming material is observed.

Step 2: apply the pattern to a new situation. Now consider a different pair, s_2 and m_2. The current layout and observations provide the following information.

Current episode.

Layout. Storage s_2 supplies machine m_2; mover q_2 is the sole incoming mover connected to s_2.

Observed. Machine m_2 is stopped, and inspection of s_2 finds zero iron plates.

Not yet observed. Mover q_2 has not been inspected, so whether material is arriving remains unresolved.

The layout establishes $\text{feeds}(s_2, m_2)$ in the episode evidence A_t. The historical observation of q_1 says nothing about the current inflow through q_2. Using r, the LLM proposes h: "Depletion of s_2 may explain the stopped m_2; the corresponding repair is to refill s_2." Substituting $s \mapsto s_2$ and $m \mapsto m_2$ gives

$$\begin{aligned} \sigma(h) &= \sigma_r(s_2, m_2) = \{\text{feeds}(s_2, m_2)\}, \\ \Phi(h) &= \Phi_r(s_2) = \big\{\underbrace{\text{stock}(s_2) = 0}_{\varphi_{\text{stock}}},\ \underbrace{\text{inflow}(s_2) = 0}_{\varphi_{\text{inflow}}}\big\}. \end{aligned}$$

Each φ is a condition to test; for example, φ_{stock} has the tuple form $(s_2, \text{stock}, =, 0)$ in Equation 7. The proposal retains both conditions from d_2, including the inflow condition that has not yet been observed in this episode. Both attributes are observable, so the complete hypothesis can be admitted before every reading is available.

Table A9 contrasts a complete proposal with three incomplete proposals, holding the other admission requirements fixed and satisfied. For the missing-relation case, assume the layout and records do not establish the supply connection. A missing reading is compatible with admission; an unsupported relation or omitted required condition is not.

Table A9: **Admission before observation-based assessment.**

Case	Proposal	Admission and reason
Complete	Established supply relation; both required conditions retained; inflow still unread.	Admitted: Missing readings can be investigated after admission.
Missing relation	The proposal names a supplier relation that has no supporting evidence.	Not admitted: Establish the relation before applying this pattern.
Missing condition	The proposal cites the pattern but checks only whether storage is empty.	Not admitted: Retain the required no-inflow condition.
No conditions	The explanation supplies no observable conditions.	Not admitted: An explanation needs a nonempty set of testable conditions.

An inadmissible proposal has not entered the hypothesis set. This differs from a *refuted* hypothesis, which was admitted and then contradicted by evidence.

Step 3: inspect the missing condition and assess the hypothesis. Equation 7 compares each expected condition with its current reading: $\top$ means supported, $\bot$ means contradicted, and ? means unresolved. Equation 8 then combines those two results into the status of h. In the current episode above, zero stock supports φ_{stock}, while φ_{inflow} is unresolved. Thus h is open (case A): the past incident suggests an explanation, but the new evidence does not yet confirm it. The LLM can request $\texttt{inspect}(q_2)$ to learn whether material is arriving. With stock still zero and m_2 still stopped, the response determines which branch follows. In each pair below, the first entry evaluates stock and the second inflow:

$$\text{A: } (\top, ?),\ \text{open} \xrightarrow{\text{no material arriving}} \text{B: } (\top, \top),\ \text{confirmed},$$

$$\text{A: } (\top, ?),\ \text{open} \xrightarrow{\text{material arriving}} \text{C: } (\top, \bot),\ \text{refuted}.$$

If inspection still leaves inflow unresolved, the case remains A and h remains open. The no-inflow branch agrees with d_2's expected conditions; the inflow branch contradicts one of them, even though the storage is empty in both cases. Table A10 gathers these branches and two further alternatives, with the structure and stopped-machine anomaly held fixed.

Table A10: **Different evidence for the same proposed explanation.** Independent alternatives; each case evaluates both expected conditions.

Case	Stock	Inflow	Values (stock, inflow) Eq. 7	Status Eq. 8	Reason and next step
A	0	Unread	$(\top, ?)$	Open	Empty stock supports one condition; inspect incoming supply to resolve the other.
B	0	None	$(\top, \top)$	Confirmed	Both expected conditions are supported. Check the proposed repair next.
C	0	Arriving	$(\top, \bot)$	Refuted	Observed inflow contradicts the no-inflow condition. Revise the explanation.
D	5	Unread	$(\bot, ?)$	Refuted	Nonzero stock already contradicts the claim; the unread inflow cannot rescue it.
E	Unread	Unread	$(?, ?)$	Open	Neither condition has been resolved. Obtain an informative observation.

In case D, the observed stock of five already contradicts the empty-storage condition; the unread inflow does not prevent refutation. Case E instead has no resolved condition, so there is no contradiction and no confirmation.

Step 4: check the proposed repair against the confirmed explanation. Continue along branch B, with both conditions supported and h confirmed. The LLM proposes $\texttt{refill}(s_2)$, transferring the repair in d_2 to the storage occupying the same role in the new episode. The action matches r's prescribed verb and h's candidate cause; both required readings currently hold, so it is backed by h.

For Table A11, take $H' = \{h\}$ and let s_3 be another storage that is neither the cause nor an entity tested by h. We write only each command's verb and target, assuming valid material and amount arguments for a refill. The table evaluates the repair-backing requirement in Appendix B.3; other action checks still apply.

Table A11: **Hypothesis confirmation and action backing.** Independent alternatives evaluated with $H' = \{h\}$; the backing check uses this candidate only.

Case	Proposal	Recorded status	Conditions (stock, inflow) Eq. 7	Backed	Reason
R1	`refill`(s_2)	Confirmed	$(\top, \top)$	Yes	Prescribed verb, matching target, and both conditions supported.
R2	`replace`(s_2)	Confirmed	$(\top, \top)$	No	The pattern prescribes a refill, not replacement.
R3	`refill`(s_3)	Confirmed	$(\top, \top)$	No	The target is neither the cause nor an entity tested by this hypothesis.
R4	`refill`(s_2)	Confirmed	$(\top, ?)$	No	Stored confirmation is insufficient when a current condition is unresolved.
O1	`inspect`(q_2)	Open	$(\top, ?)$	Not required	An observation does not require a confirmed repair hypothesis.

Rows R1–R3 show that the same confirmed explanation can back one repair and fail to back another. R4 considers a later state in which the recorded confirmation remains but a current reading is unresolved: this is why the repair check rechecks all expected conditions. O1 returns to case A and shows that the agent can seek evidence while the hypothesis is still open; an observation does not require repair backing.

Step 5: use the intervention outcome as new evidence. Suppose the refill in R1 is executed. Table A12 separates possible responses from the environment. The refutation rule for an executed repair that leaves the episode running is the diagnostic outcome rule of Tacit-FLE described in Appendix B.3.

Table A12: **Intervention outcomes after a backed refill.**

Event	Result	Meaning
The refill executes and the episode terminates successfully.	Task completed	The episode ends; no additional hypothesis status is needed.
The refill executes, but another episode turn follows.	Hypothesis refuted	Under the diagnostic outcome rule, the intervention contradicts this candidate-cause claim.
The environment refuses the refill command.	No intervention evidence	Refusal alone does not refute the hypothesis; inspect the feedback and current evidence.

For the continuing-episode branch, the transition is

$$\underbrace{\text{confirmed}}_{\text{expected readings supported}} \xrightarrow{\text{refill executed; episode continues}} \underbrace{\text{refuted}}_{\text{intervention contradicts the explanation}} .$$

This update uses intervention evidence, whereas Equation 8 assessed the initially open hypothesis from its expected readings. A refused command supplies no such intervention evidence; new observations may still change the assessment.

B.4 Learning scope and prompt contracts

The separation between reusable knowledge and episode state follows Appendix B.1. The ontology schemas fix the vocabulary and rule forms. f_{abs} and f_{propose} are implemented through LLM calls; their inputs and output contracts are given below. Verification consists of the typed constraints, relation checks, and observational predicates of Equations 4 and 7–9.

Inputs and outputs. Knowledge-construction prompts provide the schema and the corresponding record batch, requesting structured facts or axioms. The diagnostic action prompt provides world knowledge with its sources, layout facts, current readings, previous commands, hypothesis status, and available actions. It requests optional structural claims and hypotheses followed by one executable command. In Mars and ALFWorld, the actor instead receives the learned action conditions, task context, observations, and feedback from failed condition checks. The following blocks summarize the two Tacit-FLE prompt contracts; wording is condensed and bracketed fields stand for the corresponding inputs.

Knowledge construction: record-level extraction

```
Input:  [typed schema] and [reference document].
Extract the entities and structural relations stated or implied by the
record.
Generalize the incident using component types, the symptom, the cause's
relation to it, any stated observational conditions, and the documented
repair.
Do not copy historical status readings into persistent structural facts.
Return structured entities, relations, diagnostic patterns, and explicit
non-evidence policies.  The current record identifier is attached as
provenance.
```

Diagnostic interaction: explanation and action

```
Input:  [world knowledge and sources], [layout], [observations],
[command history], [hypotheses and their standing], [verification
feedback].
Propose which component could explain an observed anomaly and what
readings would support that explanation.
Name the pattern used, its required relations, expected readings,
evidence, and a repair.  Claim new structure with a permitted basis.
Use unresolved conditions and contradictory evidence to decide what to
observe or revise.
Return optional structured claims followed by one command:  inspect,
open, wait, replace, or refill.
```

C Environment Bindings and Baselines

C.1 Instantiation per environment

Table A13 instantiates the construction and interaction procedures in Section 3. The common principle is to induce structured knowledge from the reference corpus and use it to check proposals against current evidence. The representations and verification procedures specialize this principle to each environment. Tacit-FLE combines component relations, diagnostic patterns, and an episode-specific hypothesis ledger; Mars uses typed action and effect templates; ALFWorld uses conditions over observed scene predicates. Appendix B.2 specifies their admission checks.

Observation grounding. The function f_{ground} in Algorithm 2 updates episode assertions from observations using an environment-specific map ρ, left implicit in the algorithm. In Tacit-FLE, component types and positions come from the public roster, while layout and readouts supply connection and state evidence. The mapping from a stock trend to an inflow reading is a fixed interpretation of public observations, rather than an induced axiom. A stock level that remains at zero does not alone settle whether material is arriving; other readings can be needed. These observation mappings are supplied by the environment binding. The Mars binding exposes position, inventory,

Table A13: **Environment-specific instantiations of OntoAct.** The common construction and interaction procedures use the following representations and checks.

Element	Tacit-FLE	Mars	ALFWorld
Corpus D	One plant's maintenance logs and operating documents	Recorded transitions from one world	Recorded transitions from training games
Vocabulary T	Component types, material-flow relations, statuses, and public readings	Materials, items, creatures, action and effect templates	Action verbs and five observation predicates
Induced knowledge	Structural facts, diagnostic patterns, non-evidence axioms, practices, aggregate priors	Collect, make, place, walk, drink, creature, neighborhood, and blocked-action conditions	Necessary action conditions
Evidence batch	One document; practices and service samples over the joined corpus	The supplied transition buffer	By action, at most 24 transitions per batch
Corpus checks	Type-specific source, structure, support, and counterexample checks	Subject-linked citations; at most one contradictory transition	Action-linked citations; positive support; no successful counterexample
Observation map ρ	Roster, layout, public readings; stock changes inform inflow	Position, inventory, vitals, visible cells	Parsed scene, inventory, and receptacle state
Action checks	Repair backing; operating practices; repetition feedback	Learned applicability conditions and lethal-ground checks on an exploration path	Learned conditions over observed predicates
Proposals J	2	5	5

vitals, and visible cells; the ALFWorld binding maps parsed scene information onto the five predicates used by its learned conditions.

C.2 Baseline implementations and prompts

We implement each baseline's knowledge representation and decision procedure using the published method and available reference code. Environment bindings supply the task description, observation format, action syntax, and reference records. The descriptions below specify how each method uses the corpus in our experiments. Where a released implementation omits a component or departs from the published procedure, we follow the paper; such adjustments are noted with each method.

ReAct and ReAct*. ReAct alternates a textual thought with an executable action based on the task description and observations. ReAct* supplements the same actor with up to five corpus records retrieved once at the start of an episode. The retrieval query uses the initial observation and, in Tacit-FLE, the plant roster. The retrieved records remain available in the episode context. Both variants share the actor prompt, action parser, and history handling within an environment.

Reflexion. We use a corpus-based adaptation of reflective memory. Before evaluation, the model reads each reference document and writes a memory note describing useful evidence and conditional plans. At the start of an episode, the initial-observation query retrieves up to three relevant notes for the actor. The reflection prompt asks the model to distinguish observed outcomes from inferred advice and to preserve the conditions under which an action is useful. This adaptation constructs the reflective memory from the supplied records.

IfR. The induction prompt converts recorded interactions into a natural-language rule library. On Tacit-FLE, repair records additionally populate a skill library describing successful procedures. The planner uses the task, observations, and retrieved knowledge to propose actions; an explanation prompt interprets unsuccessful execution and supports replanning. We route all LLM calls and actions through the shared interfaces while retaining the separate induction, planning, explanation, and replanning stages; the reference corpus initializes the rule library and, on Tacit-FLE, the skill library; episode history is retained across replanning rounds, and library entries above the similarity threshold are replaced by the newly induced rule.

WALL-E. The learning stage compares predicted action outcomes with recorded outcomes. Prediction discrepancies drive natural-language rule proposals, which are compiled into executable checks and validated against recorded transitions. During evaluation, a proposer and world model use the learned knowledge, and the executable rules check candidate actions before execution. Our bindings present corpus records in the input format used by this learning procedure. The public release does not include the Mars learning and evaluation pipeline; we assemble it from the published procedure, the released prompts, and the released ALFWorld learning and verification components, align observation and action representations across corpus ingestion, rule validation, and execution, and fix interface errors in the released code. Candidate rejection and reproposal follow the released execution loop. On Tacit-FLE (Table 1), disagreement between an action-matched executable check and the LLM prediction triggers one further query about the same action, which includes the code verdict and feedback, following the alignment step in Zhou et al. (2025b, Eq. 9); the revised prediction determines rejection, and when no matching check can be evaluated, the initial prediction is retained. A passing Boolean check records the absence of an encoded failure condition. The action-proposal loop permits five attempts per environment turn and passes the final proposal to the executor if the limit is reached. The Mars binding supplies spatial context through the current observation and bounded interaction history.

NeSyC. We reconstruct the induction and rule-application procedures in Algorithms 1–2 of Choi et al. (2025), using the released implementation as a reference. The learner proposes answer-set rules from the corpus and refines them using contrasting transition examples. At evaluation, observations are translated into facts and combined with the learned program for symbolic planning. We connect the hypothesis score to iterative rule revision, completing the refinement loop described in the paper: solver checks provide feedback on labeled examples, with false positives penalized according to the published objective, and statements that fail to parse or ground are removed before scoring and

reported to the learner. The bindings translate observations and task goals into facts, support the induction of action effects from the corpus, annotate completion in corpus records with the planner's state vocabulary, and expose inspection results as structured attributes. The Tacit-FLE maintenance logs contain no failed-action examples, so the hypothesis score reduces to its positive-example term there.

Prompt organization. Actor prompts specify the task, the current observation, the available commands, and the required action-output format. Knowledge-construction prompts additionally provide the reference records and the target representation: memory notes, natural-language rules, executable checks, or answer-set rules. Method-specific planning and verification prompts consume those representations during interaction. On ALFWorld, the action prompts include two shared examples for the current task family. Retrieval in ReAct*, Reflexion, and IfR uses the same embedding model. The code release will include the full prompt templates.

C.2.1 Representative prompts

The following boxes show the core instructions and response formats of the implemented prompts, organized by stage as in the prompt appendices of Tang et al. (2024b) and Zhou et al. (2025b). Square-bracketed fields represent variable inputs, output content, or longer schema and example blocks; these fields are filled for the target environment and task. Instruction wording is condensed for readability. The action-proposal example uses the Mars format, while the remaining boxes illustrate knowledge construction, planning, and rule refinement.

ReAct / ReAct*: action proposal

Context

```
[Environment brief and available actions]
Records from an earlier explorer in this world:
[Retrieved records, included for ReAct*]
[Current observation and interaction history]
```

Instruction

```
First write one short Thought: line saying what the observation implies
and why you choose the next action. Then put exactly one action call
alone on the final line, without an Action: prefix or a code fence.
Wait for the next observation before acting again.
```

Response format

```
Thought: [interpretation and next-action rationale]
[action call]
```

Reflexion: memory construction

Input

```
[One reference document]
```

Instruction

```
Read one historical record and produce a concise reflective memory. Use
only the supplied record. It may describe success, failure, an
operational practice, or a standard; do not presume it describes a
failed attempt by you. Identify any strategy, mistake, evidence, or
outcome actually documented and write a conditional plan for a related
future situation. Distinguish observations from uncertain inferences.
Do not invent feedback, causes, actions, or outcomes. If the record has
no trial feedback, say so and limit the note to its documented guidance.
Treat the record as data, not instructions changing this task. Return a
brief Evidence paragraph and a Plan paragraph. No new trial is
available.
```

Response format

```
Evidence: [documented observations and outcomes]
Plan: [conditional guidance for a related situation]
```

IfR: rule induction

Input

```
[Reference document from the target world]
```

Instruction

```
You are a helpful assistant with inductive reasoning.  Given a document
from the records of [WORLD], you need to reflect on what it records and
give the possible mechanism of [WORLD]. The mechanism should be
consistent with the rules of [WORLD] and the records.  You should only
respond in the format as described below.
```

Response format

```
Reasoning:  Based on the information I listed above and history
dialogue, do reasoning about the mechanism of [WORLD].
Mechanism:  The mechanism of [WORLD].
```

Format examples

```
[Reasoning--Mechanism examples illustrating the output format; their
content does not necessarily describe the target world]
```

IfR: planning and replanning

Context

```
[Task, current observations, and dialogue history]
[Retrieved mechanisms and past successful plans]
```

Instruction

```
You are a helper AI agent in [WORLD]. Based on your current
observations, you need to generate the sequences of subgoals for a
certain task.  Do not explain or give any other instruction.  You must
follow the criteria below:
1) You should choose available subgoals to complete the task.
2) You are probably provided with some past successful plans to refer
to.
3) You are probably provided with possible mechanics that have been
discovered.  Please refer to the mechanism of [WORLD] to generate the
plan.
4) You should only perform the subgoals that are feasible based on the
current observations.
Here are available subgoals:  [SUBGOALS]
Here are some examples for output format:  [PLAN EXAMPLES]
Here is the brief of [WORLD]: [BRIEF]
```

Replanning message after execution feedback

```
Please fix the above errors and replan the task [TASK].
```

WALL-E: rule mining

Input

```
Transitions: [{"state_0": [state], "action": {"name": [action name],
"args": [arguments]}, "action_result": {"feedback": [feedback],
"success": [execution outcome]}}, ...]
Existing rules: [rule list]
The environment's action space: [ACTIONS]
The environment's state format: [STATE FORMAT]
```

Instruction

```
You are responsible for mining new rules from the given transitions,
ensuring that these rules differ from the ones already provided. Focus
on generating general and universal rules that are not tied to any
specific item or tool. Extract and utilize only the features that
influence the outcome of the action. Generate only the rules under what
conditions the action will fail. While generating a rule, you also need
to state how to check if a transition satisfies this rule. Please be
specific as to which and how features need to be checked. Ensure the
response can be parsed by Python json.loads.
```

Response format

```
{"new_rules": [
  "Rule [number]: For action [action], [failure condition]; Checking
Method: [features and checks]"
]}
```

WALL-E: rule compilation

Input

```
[Learned rule and state/action format]
Current Knowledge Graph: [KNOWLEDGE GRAPH]
Current Scene Graph: [SCENE GRAPH]
```

Instruction

```
You are responsible for generating code rules by implementing the
learned rules in Python. Write a function that takes the current state
and an action as inputs, optionally incorporating the graph to provide
additional contextual information. The function should return feedback,
success, and suggestion.
feedback: a string giving the action feedback.
success: a bool indicating whether the action is executed successfully.
If the action type is not the action type in the rule, count as success.
suggestion: a string giving a suggestion based on the rule, state, and
action if the action fails.
The state and action arguments are dicts; action has the form {"name":
"...", "args": {...}}. The graph argument has the form {"knowledge":
[knowledge graph], "scene": [scene graph]}. Respond only with the
function, without example usage or other text.
```

Response format

```
def expected_rule_code(state, action, graph):
  [condition checks]
  return feedback, success, suggestion
```

NeSyC: rule induction

Input

```
[ASP grammar, environment vocabulary, and target command]
[Batch of reference records]
Background Knowledge:  [background program]
[Rule kind] so far:  [current rules, or none yet]
```

Instruction

```
Use inductive logic programming in Answer Set Programming.  Learn from
interpretations:  a positive example should be a model of the background
and hypothesis; negative examples should not.  Apply theta-subsumption
when generalizing.  Use the supplied vocabulary and records.  Unknown
labels are not negative examples.
Return the complete set of [rule kind] for the target command after
reading this batch:  keep earlier rules these records still support,
generalize where they allow, and add what they require.  One complete
rule per line.  Effects must come from observed changes or the supplied
API description; do not treat an unsuccessful task as an unexecutable
action.  Return only the requested ASP, without explanations or
markdown.
```

Response format

```
[Complete ASP rule]
[Additional complete ASP rules, one per line]
```

NeSyC: contrastive refinement

Input

```
[Background program and current rules]
[Hypothesis score and contrastive feedback on labeled examples]
```

Instruction

```
Revise the learned ASP rules using the contrastive feedback.  Apply
theta-subsumption:  prefer the more general clause that still covers
every positive example and excludes every negative one.  A positive
example must be a model of background and rules; a negative example must
not be.  Constraint rules are the hypothesis to revise.  Effect rules
(update/4 heads) are kept as learned unless the feedback reports two
rules deriving different values; in that case return the resolved effect
rules as well.  Unknown labels are not negative examples.  Never define
goal, action, available, step or changed in a rule head.  Return the
complete revised rule set, one complete rule per line, without
explanations or markdown.
```

Response format

```
[Complete revised ASP rule]
[Additional complete ASP rules, one per line]
```

D Full Results and Cost Accounting

D.1 Repair success by scale and type

Table A14 gives the successes and denominators behind Table 1.

NeSyC evaluation trajectories. For NeSyC, repair success is 0/28 and 0/36 on the small and medium suites. Across the 3,000 evaluation turns, every executed action was waiting. Of the 64 initial planner outputs, 47 return no plan and 17 return a single waiting action. These observations explain its absence from the repair-candidate diagnostics in Appendix D.7.

Model conditions. Table A15 retains the full success, spurious-intervention, and turn comparison behind Figure 4 on the same 36 medium tasks. The model condition applies to both knowledge construction and interaction. WALL-E and NeSyC are compared under Luna only (Table 1).

Table A14: **Repair success counts in the Luna condition.** Successes over scheduled episodes behind Table 1. The small, medium, and large suites contain 28, 36, and 40 episodes, respectively. Unsuccessful episodes remain in the denominators. Bold: best per column, ties included.

	Small			Medium			Large		
Method	Chain	Conj.	Elim.	Chain	Conj.	Elim.	Chain	Conj.	Elim.
ReAct	0/13	5/7	**6/8**	4/14	9/14	6/8	1/17	8/11	9/12
ReAct*	2/13	6/7	3/8	3/14	9/14	7/8	4/17	4/11	8/12
Reflexion	1/13	5/7	**6/8**	3/14	6/14	3/8	2/17	1/11	6/12
IfR	5/13	**7/7**	**6/8**	4/14	12/14	**8/8**	4/17	10/11	**12/12**
NeSyC	0/13	0/7	0/8	0/14	0/14	0/8	0/17	0/11	0/12
WALL-E	1/13	1/7	1/8	2/14	4/14	7/8	1/17	5/11	9/12
OntoAct (ours)	**9/13**	**7/7**	**6/8**	**8/14**	**14/14**	**8/8**	**14/17**	**11/11**	11/12

Table A15: **Model conditions on medium Tacit-FLE (36 tasks).** SR: repair success (%); SI: mean spurious interventions; T: mean turns. Bold/underline: best/second-best within each condition.

	ReAct	ReAct*	Reflexion	IfR	OntoAct
Luna					
SR ↑	52.8	52.8	33.3	<u>66.7</u>	**83.3**
SI ↓	<u>11.3</u>	13.4	15.4	11.9	**2.4**
T ↓	41.8	38.7	46.4	<u>31.7</u>	**19.4**
Luna^R					
SR ↑	44.4	38.9	52.8	<u>86.1</u>	**97.2**
SI ↓	<u>9.0</u>	11.4	9.7	9.8	**0.8**
T ↓	42.8	45.3	42.1	<u>26.7</u>	**8.4**
Terra					
SR ↑	52.8	55.6	38.9	<u>66.7</u>	**100.0**
SI ↓	13.7	12.6	10.5	<u>7.9</u>	**0.4**
T ↓	32.7	35.3	42.2	<u>28.2</u>	**8.4**

D.2 MARS ACROSS MODEL CONDITIONS

Tables A16 and A17 give the full world-level results behind the additional conditions in Table 4. The Luna world-level results remain in Table 3. Each reported world-level result aggregates five evaluation episodes. NeSyC's results appear in Tables 3, A16, and A17.

NeSyC planning behavior. In our Luna reconstruction, 5,325 of 6,326 turns across the seven modified worlds execute a no-op. Erroneous or overly restrictive learned rules can prevent executable plans; for example, a Terrain rule imposes a tool-inventory requirement without restricting it to the relevant action, contradicting the initial state. The runtime falls back to waiting when planning fails, while rule refinement requires a refused executed command. These observations concern our reconstruction (Appendix C.2).

Table A16: **Mars performance with Luna^R.** Luna^R; five evaluation episodes per world. Reward is mean ± sample SD; Score pools achievement success rates across episodes. Average excludes Default; its Reward SD is computed across matched-seed world averages. Bold/underlined: best/second-best. Metric definitions are in Appendix E.3.

World	ReAct	ReAct*	Reflexion	IfR	NeSyC	WALL-E	OntoAct
Reward ↑							
Default	12.30±4.82	**15.46±5.35**	11.30±2.28	10.70±3.21	5.77±0.58	9.90±2.77	10.90±3.56
Terrain	13.10±2.55	10.10±5.87	14.50±1.14	**15.98±2.06**	−0.57±0.58	10.30±3.11	14.06±5.67
Survival	9.70±5.18	13.70±5.73	9.10±6.78	12.50±3.51	4.07±3.55	10.50±6.54	**13.88±5.60**
Task dep.	5.70±2.30	2.10±0.71	2.90±1.30	**7.70±3.13**	1.77±1.53	3.70±2.51	5.70±1.14
Terr. surv.	7.50±2.61	5.30±3.27	6.90±2.59	8.30±3.11	5.10±1.00	7.10±2.65	**8.50±3.44**
Terr. task	5.30±2.28	6.30±2.68	5.70±2.51	6.90±2.28	0.10±1.00	4.70±2.07	**8.10±1.87**
Surv. task	5.50±5.08	6.10±4.30	5.90±5.07	5.90±5.63	−0.23±0.58	2.30±1.48	**7.30±6.57**
All three	0.70±2.19	**1.70±1.67**	1.10±1.87	0.90±1.10	0.10±1.00	−0.30±0.89	0.90±1.79
Average	6.79±1.10	6.47±1.21	6.59±1.35	8.31±0.68	1.48±0.94	5.47±0.97	**8.35±1.26**
Score (%) ↑							
Default	46.65	**68.27**	26.44	29.64	3.86	15.82	30.26
Terrain	32.57	22.80	35.52	**56.87**	0.17	19.66	52.42
Survival	21.70	56.98	23.66	43.88	3.24	31.41	**58.48**
Task dep.	7.43	1.51	2.33	**9.09**	1.42	3.17	3.92
Terr. surv.	**12.86**	6.73	7.99	10.67	3.62	8.43	12.43
Terr. task	5.20	6.77	6.29	**8.10**	0.62	4.80	7.87
Surv. task	10.45	10.03	12.72	14.60	0.21	1.50	**24.30**
All three	1.19	1.77	**1.89**	0.97	0.42	0.51	1.48
Average	13.06	15.23	12.91	20.60	1.39	9.93	**22.99**

D.3 MARS PER TRIAL

Tables A18 and A19 report the recorded reward for each evaluation seed behind Table 3. The Luna comparison comprises seven methods, eight worlds, and seeds 0–4, for 280 episodes. Score is computed from achievement success rates pooled over the five episodes, as defined in Appendix E.3.

D.4 MARS: APPLYING SHARED LEARNED CONDITIONS

Shared knowledge and evaluation scope. We analyze the existing Luna trajectories from all seven modified worlds, with five evaluation seeds per world and method (35 episodes each). Both methods construct their knowledge before evaluation from the same 200 recorded transitions per world and keep their rule libraries fixed during these trajectories. The analysis distinguishes whether a method possesses a resource condition from whether its current action respects that condition.

For each of the 14 paired knowledge-construction instances, we manually match explicit resource requirements for crafting and placement that both methods store correctly. We require an explicit material name; a requirement without a quantity supports only a minimum of one unit. This produces 22 pair-specific matches, including repeated conditions across instances; 13 of the 14 pairs have at

Table A17: **Mars performance with Terra.** Terra; five evaluation episodes per world. Reward is mean ± sample SD; Score pools achievement success rates across episodes. Average excludes Default; its Reward SD is computed across matched-seed world averages. Bold/underlined: best/second-best. Metric definitions are in Appendix E.3.

World	ReAct	ReAct*	Reflexion	IfR	NeSyC	WALL-E	OntoAct
Reward ↑							
Default	10.10±2.74	10.70±2.97	10.30±1.10	**13.70±2.88**	3.50±1.95	11.30±2.28	12.50±2.19
Terrain	11.30±4.21	10.48±5.82	11.10±4.85	**15.38±4.55**	2.28±2.16	10.12±4.09	14.04±3.43
Survival	8.10±5.39	10.28±6.87	7.10±3.39	12.78±8.06	0.70±0.55	7.10±4.06	**15.50±1.52**
Task dep.	6.70±3.51	6.50±4.34	6.10±3.94	7.10±2.24	0.10±0.00	4.70±1.82	**8.10±3.94**
Terr. surv.	7.50±2.70	8.10±4.85	**9.90±1.64**	5.30±1.30	1.90±1.10	4.10±2.35	6.50±3.13
Terr. task	**7.86±1.87**	7.68±3.75	6.90±1.30	5.10±1.87	1.10±0.71	5.50±2.41	6.10±2.00
Surv. task	7.50±5.68	3.30±5.07	6.30±3.70	3.10±2.00	2.70±0.55	5.10±3.39	**7.90±6.69**
All three	0.30±1.10	0.70±2.07	1.30±1.48	**3.10±2.24**	−0.70±0.45	0.70±2.30	1.50±2.30
Average	7.04±0.86	6.72±1.59	6.96±0.86	7.41±1.82	1.15±0.51	5.33±1.15	**8.52±1.54**
Score (%) ↑							
Default	21.42	20.03	18.81	**42.95**	2.98	23.71	37.87
Terrain	31.84	33.57	30.44	**57.79**	1.52	19.18	38.16
Survival	20.77	37.49	12.58	50.36	0.49	12.92	**57.22**
Task dep.	9.43	12.13	9.74	8.20	0.23	4.92	**14.43**
Terr. surv.	10.06	**14.47**	14.23	5.08	1.13	4.57	8.66
Terr. task	7.49	**9.97**	6.77	5.08	0.77	5.25	6.38
Surv. task	17.17	6.30	9.32	2.43	1.55	6.52	**23.35**
All three	0.64	1.44	1.56	**3.10**	0.15	1.19	2.81
Average	13.91	16.48	12.09	18.86	0.84	7.79	**21.57**

least one match. The metric checks only these shared resource requirements against the inventory at action selection. Terrain, workstations, tools, and other feasibility conditions are outside this metric.

Candidates, commands, and denominators. For OntoAct, the initial candidate is the first single-action proposal in the current turn, before verification feedback. For IfR, it is the next executable step selected from the plan after processing the previous action's outcome. We do not test future plan steps against the current inventory or count plan steps discarded while processing a preceding failure. Every candidate and command included in Table 5 has an observed inventory.

The methods follow different trajectories, so their denominators count their own eligible candidates or issued commands rather than matched states. For OntoAct, the 245 initial candidates comprise 70 resource violations and 175 candidates satisfying the shared resource conditions. All 70 violations are redirected to gathering or exploration before execution. Another 39 candidates are redirected after terrain or workstation feedback, leaving 136 original crafting or placement commands to execute. For IfR, all 107 selected candidates are issued unchanged, including all 20 resource violations. The redirection row conditions on violating candidates only: 70 for OntoAct and 20 for IfR.

Timing and outcome of revision. The 70 pre-execution redirections by OntoAct occur in 19 episodes across six worlds. The alternatives produce 54 controller successes and 16 failures, where controller success refers to the issued command. Of the 20 violations issued by IfR, 19 continue an existing plan without a new LLM call and one follows a fresh plan. All 20 attempts fail and together consume 20 environment steps. Each is followed immediately by an explanation and replanning turn that selects a different command outside the shared crafting and placement scope; these alternatives yield 16 controller successes and four failures. Thus, both methods revise their behavior, but the observed revisions occur before a known-invalid attempt for OntoAct and after the attempt for IfR.

Coverage beyond shared conditions. The zero violations in Table 5 apply to the shared learned conditions. When the check is expanded to every actual resource requirement of OntoAct's 781 issued crafting or placement commands, all 363 violations occur among the 622 commands whose action lacks an explicit typed resource-cost rule; the 159 commands whose actions have such rules contain none.

Table A18: **Mars reward per evaluation seed (1/2).** Luna; all seven methods and seeds 0–4 from Table 3. Each seed column is one recorded episode reward; the final column reports the mean and sample SD across the five seeds.

World	Method	Seed 0	Seed 1	Seed 2	Seed 3	Seed 4	Mean ± SD
Default	ReAct	7.10	17.10	7.10	13.10	6.10	10.10 ± 4.80
	ReAct*	10.10	11.10	7.10	8.10	10.10	9.30 ± 1.64
	Reflexion	8.10	3.10	8.10	11.10	9.10	7.90 ± 2.95
	IfR	6.10	7.10	11.10	6.10	10.10	8.10 ± 2.35
	NeSyC	1.10	0.10	1.10	1.10	0.10	0.70 ± 0.55
	WALL-E	5.10	16.10	7.10	0.10	3.10	6.30 ± 6.06
	OntoAct (ours)	13.10	12.10	13.10	10.10	12.10	12.10 ± 1.22
Terrain	ReAct	9.10	6.10	7.10	12.10	9.10	8.70 ± 2.30
	ReAct*	13.10	12.10	12.10	17.40	6.10	12.16 ± 4.03
	Reflexion	11.10	3.10	2.10	13.10	14.00	8.68 ± 5.66
	IfR	10.10	3.10	9.10	14.10	10.10	9.30 ± 3.96
	NeSyC	-0.90	-0.90	-0.90	-0.90	-0.90	-0.90 ± 0.00
	WALL-E	5.10	4.10	5.10	12.10	11.10	7.50 ± 3.78
	OntoAct (ours)	11.10	13.10	10.10	12.10	15.10	12.30 ± 1.92
Survival	ReAct	2.10	4.10	8.10	9.10	12.10	7.10 ± 4.00
	ReAct*	4.10	11.10	8.10	12.10	12.10	9.50 ± 3.44
	Reflexion	1.10	3.10	8.10	14.10	5.10	6.30 ± 5.07
	IfR	1.10	9.10	5.10	7.10	14.10	7.30 ± 4.82
	NeSyC	1.10	0.10	1.10	2.50	0.10	0.98 ± 0.99
	WALL-E	1.10	1.10	9.10	12.10	8.10	6.30 ± 4.97
	OntoAct (ours)	2.10	16.00	5.10	14.10	15.00	10.46 ± 6.39
Task dep.	ReAct	3.10	4.10	5.10	3.10	5.10	4.10 ± 1.00
	ReAct*	1.10	4.10	4.10	3.10	2.10	2.90 ± 1.30
	Reflexion	9.10	3.10	2.10	4.10	2.10	4.10 ± 2.92
	IfR	7.10	6.10	9.10	7.10	7.10	7.30 ± 1.10
	NeSyC	1.10	1.10	1.10	0.10	1.10	0.90 ± 0.45
	WALL-E	3.10	5.10	4.10	3.10	4.10	3.90 ± 0.84
	OntoAct (ours)	12.10	10.10	12.10	10.10	12.10	11.30 ± 1.10

Table A19: **Mars reward per evaluation seed (2/2).** Luna; all seven methods and seeds 0–4 from Table 3. Each seed column is one recorded episode reward; the final column reports the mean and sample SD across the five seeds.

World	Method	Seed 0	Seed 1	Seed 2	Seed 3	Seed 4	Mean ± SD
Terr. surv.	ReAct	4.10	8.10	3.10	7.10	8.10	6.10 ± 2.35
	ReAct*	7.10	6.10	12.10	7.10	8.10	8.10 ± 2.35
	Reflexion	3.10	7.10	6.10	1.10	8.10	5.10 ± 2.92
	IfR	3.10	2.10	6.10	6.10	13.10	6.10 ± 4.30
	NeSyC	3.10	5.10	0.10	2.10	3.10	2.70 ± 1.82
	WALL-E	7.10	0.10	3.10	2.10	7.10	3.90 ± 3.11
	OntoAct (ours)	10.10	2.10	6.10	5.10	8.10	6.30 ± 3.03
Terr. task	ReAct	5.10	1.10	5.10	3.10	3.10	3.50 ± 1.67
	ReAct*	8.10	7.10	3.10	8.10	8.10	6.90 ± 2.17
	Reflexion	5.10	5.10	6.10	5.10	11.00	6.48 ± 2.56
	IfR	4.10	2.10	4.10	2.10	7.10	3.90 ± 2.05
	NeSyC	0.10	2.10	2.10	1.10	0.10	1.10 ± 1.00
	WALL-E	4.10	3.10	0.10	3.10	4.10	2.90 ± 1.64
	OntoAct (ours)	10.10	4.10	5.10	10.10	9.10	7.70 ± 2.88
Surv. task	ReAct	2.10	3.10	12.10	14.10	1.10	6.50 ± 6.11
	ReAct*	-0.90	0.10	2.10	6.10	3.10	2.10 ± 2.74
	Reflexion	2.10	1.10	3.10	7.10	5.10	3.70 ± 2.41
	IfR	2.10	2.10	10.10	9.10	9.10	6.50 ± 4.04
	NeSyC	1.10	3.10	1.10	4.10	2.10	2.30 ± 1.30
	WALL-E	2.10	0.10	4.10	10.10	12.10	5.70 ± 5.18
	OntoAct (ours)	-0.90	0.10	5.10	11.10	12.10	5.50 ± 6.02
All three	ReAct	2.10	5.10	-0.90	-0.90	-0.90	0.90 ± 2.68
	ReAct*	3.10	1.10	-0.90	5.10	-0.90	1.50 ± 2.61
	Reflexion	1.10	2.10	-0.90	0.10	0.10	0.50 ± 1.14
	IfR	-0.90	-0.90	2.10	0.10	-0.90	-0.10 ± 1.30
	NeSyC	-0.90	-0.90	-0.90	-0.90	-0.90	-0.90 ± 0.00
	WALL-E	3.10	1.10	-0.90	0.10	0.10	0.70 ± 1.52
	OntoAct (ours)	2.10	4.10	-0.90	-0.90	1.10	1.10 ± 2.12

D.5 ALFWORLD FULL RESULTS

Table A20 reports the three evaluated methods—NeSyC, WALL-E, and OntoAct—under standard and disclosed initial observations, with the input and output tokens behind the Luna condition in Table 6.

Initial-state disclosure. Initial object locations specify which receptacle contains each object at episode reset, including contents that the agent has not yet observed through exploration. Inventory specifies the objects held by the agent, or explicitly states that its hands are empty. For this control, the initial object locations and inventory are appended as ordinary observation sentences to the first textual observation. The same disclosure procedure is applied to all three methods. Subsequent observations use the standard interface; no additional target annotations or action-effect rules are supplied. The model, learned knowledge, 134 unseen tasks, seed, and 49-step budget remain the same across the two observation conditions.

Table A20 reports both conditions for every method. With disclosure, OntoAct completes 134/134 tasks rather than 132/134, and WALL-E completes 133/134 rather than 129/134; both also use fewer steps and evaluation tokens. Our NeSyC reconstruction completes 0/134 tasks under standard observations: initially unobserved target objects prevent planning, and all 6,566 executed steps are idle actions. With disclosure it completes 35/134 tasks. The † rows in Tables 6 and A21 use this disclosed-state run for NeSyC, while the OntoAct and WALL-E rows use standard observations.

Evaluation call roles. Across the 134 standard-observation tasks, all 1,833 evaluation calls by OntoAct propose actions; learned conditions are checked symbolically. The 3,862 calls by WALL-E comprise 2,036 action proposals and 1,826 outcome predictions. These counts reconcile with Table A20: the methods execute 1,805 and 1,985 environment steps, giving 1.02 and 1.95 LLM calls per step, respectively. The prediction calls account for 90.0% of the 2,029-call difference and consume 1,919,113 tokens, or 92.2% of the observed evaluation-token difference. Knowledge-construction costs are reported separately (Table A25).

Table A20: **ALFWorld with and without initial-state disclosure (Luna).** Each method uses the same learned knowledge and 134 unseen tasks in both conditions, with one attempt and a 49-step budget per task. Success gives completed tasks; Steps, Calls, and input (In) and output (Out) tokens in thousands are means over all 134 episodes, including failures. The disclosure condition supplies initial object locations and inventory to every method.

	Standard observations					Initial-state disclosure				
Method	Success	Steps	Calls	In (k)	Out (k)	Success	Steps	Calls	In (k)	Out (k)
NeSyC	0/134	49.0	1.0	5.82	0.06	35/134	38.6	9.3	269.33	11.99
WALL-E	129/134	14.8	28.8	47.08	0.61	133/134	7.8	15.6	27.83	0.38
OntoAct (ours)	132/134	13.5	13.7	32.00	0.15	134/134	6.7	8.3	23.28	0.15

Table A21 reports success by task family; following Shridhar et al. (2021), the families are Pick & Place, Clean & Place, Heat & Place, Cool & Place, Examine in Light, and Pick Two & Place.

Table A21: **ALFWorld success (%) by task family under the Luna condition.** The six families follow Shridhar et al. (2021); parentheses give the number of unseen tasks. All is success over all 134 tasks, weighted by family size. Best values are bold; second-best All values are underlined. Ties share the same formatting. †Initial object locations and inventory are additionally provided to NeSyC.

Method	Pick (24)	Clean (31)	Heat (23)	Cool (21)	Examine (18)	Pick two (17)	All
NeSyC†	75.0	0.0	34.8	0.0	0.0	52.9	26.1
WALL-E	**100.0**	**96.8**	95.7	**100.0**	88.9	94.1	<u>96.3</u>
OntoAct (ours)	**100.0**	**96.8**	**100.0**	**100.0**	**94.4**	**100.0**	**98.5**

D.6 LLM USAGE

We separate knowledge-construction usage from evaluation usage. Construction costs are measured per construction run: for a plant in Tacit-FLE, a world in Mars, or a training corpus in ALFWorld. Evaluation costs are summed over the associated evaluation tasks and also reported per episode.

For WALL-E in Table 1, construction usage includes the original construction costs of every knowledge base used in the evaluation protocol of Appendix E.1, including reused knowledge. Evaluation costs cover only the reported episodes.

Table A22 reports the Tacit-FLE usage under Luna. Construction is a one-time cost per plant. OntoAct processes each document with the typed schema and elicits operating practices over the joined corpus, so its construction cost per plant exceeds that of WALL-E, IfR, and Reflexion. Evaluation dominates the total: on medium, construction for the 15 plants sums to 0.88M tokens against 9.6M evaluation tokens for OntoAct, which uses the fewest evaluation tokens per episode at every scale (excluding NeSyC).

Table A22: **Tacit-FLE LLM usage (Luna).** Construction calls and tokens are per plant (15 plants per scale); evaluation calls and tokens are per episode (28, 36, and 40 episodes). Tokens sum input and output in thousands (k). ReAct and ReAct* construct no knowledge; NeSyC, which executes no repair action (Appendix D.1), is omitted.

	Construction (per plant)		Evaluation (per episode)	
Method	Calls	Tokens (k)	Calls	Tokens (k)
Small				
ReAct	0	0	22.2	154.6
ReAct*	0	0	22.9	166.6
Reflexion	6.8	2.3	23.2	170.4
IfR	6.8	3.7	13.8	135.3
WALL-E	13.4	11.7	88.0	436.6
OntoAct (ours)	13.1	14.8	16.2	128.0
Medium				
ReAct	0	0	41.8	619.0
ReAct*	0	0	38.7	564.1
Reflexion	32.0	10.6	46.4	734.3
IfR	32.0	17.1	21.5	422.0
WALL-E	39.1	34.9	152.9	1574.0
OntoAct (ours)	48.8	58.7	27.1	265.5
Large				
ReAct	0	0	48.8	1204.3
ReAct*	0	0	56.4	1403.6
Reflexion	63.1	20.7	63.1	1606.2
IfR	63.1	33.8	32.2	1038.9
WALL-E	71.5	62.8	170.6	2541.4
OntoAct (ours)	95.8	119.7	24.6	309.4

For Mars, Table A23 reports evaluation costs for all model conditions and Table A24 reports construction costs. Each method–model aggregate in these tables covers 40 evaluation episodes across eight worlds. Figure 5 uses the 35 Luna episodes from the seven modified worlds, matching the main performance average. For each world, we divide the total evaluation input and output tokens by the total executed environment steps across its five episodes. We then average these seven rates with equal world weights; the figure's Score is the corresponding mean of the seven pooled world Scores. Environment steps count primitive controller interactions, not LLM calls or high-level command proposals. This rate normalizes for interaction length, and the figure pairs it with achievement Score. Knowledge-construction costs are excluded from this rate and reported separately.

For the same seven-world Luna population, pooled input tokens per call are 2.36k for OntoAct and 4.26k for ReAct*. Their equally weighted world-average call rates are similar: 0.3655 and 0.3639

per environment step. Prompt inspection shows that OntoAct supplies compact learned conditions, whereas ReAct* retains five retrieved transition documents; both retain ten turns of history. Across all 35 paired initial observations, which are identical between methods, first-call input averages 1,053 versus 2,982 tokens. The usage difference therefore lies primarily in input context.

The ALFWorld evaluation costs appear alongside success in Table 6, with the full breakdown in Appendix D.5. Table A25 reports the separate knowledge-construction costs.

An LLM call is one completed model response, including internal planning and verification responses; token counts are those reported by the provider. Input/call is the total input tokens divided by the number of recorded calls; it measures the mean context supplied to the model. Total tokens is the sum of input and output tokens before rounding, averaged per evaluation episode or per construction run as specified in each table. Knowledge-construction, planning, explanation, and verification calls are assigned to their corresponding phase.

Table A23: **Mars evaluation resource use.** Means over 40 episodes per method and model (eight worlds, five episodes each). Steps are environment steps; Calls, In, Out, and Total tokens are per episode. Input/call is mean input tokens per call; Total tokens sums input and output before rounding. Token quantities are in thousands (k). Knowledge-construction costs are reported separately in Appendix D.6.

Method	Steps	Calls	In (k)	Out (k)	Input/call (k)	Total tokens (k)
Luna						
ReAct	198.6	60.8	109.13	1.42	1.79	110.55
ReAct*	252.5	83.0	354.22	2.12	4.27	356.35
Reflexion	249.5	75.8	174.74	2.05	2.31	176.79
IfR	218.2	55.2	298.10	12.45	5.40	310.55
NeSyC	177.9	183.1	2237.98	13.65	12.22	2251.63
WALL-E	220.0	204.1	544.71	5.26	2.67	549.97
OntoAct (ours)	304.8	103.8	244.95	1.01	2.36	245.95
Luna^{R}						
ReAct	220.4	69.0	127.57	7.70	1.85	135.27
ReAct*	267.2	91.2	395.61	11.01	4.34	406.62
Reflexion	258.0	83.0	198.72	9.60	2.39	208.33
IfR	291.7	60.5	320.37	32.28	5.30	352.65
NeSyC	208.6	192.3	2182.93	114.77	11.35	2297.70
WALL-E	228.6	148.9	379.76	15.21	2.55	394.97
OntoAct (ours)	306.6	84.5	204.36	11.29	2.42	215.64
Terra						
ReAct	252.1	69.9	126.91	2.13	1.82	129.04
ReAct*	262.6	70.1	300.29	2.15	4.28	302.45
Reflexion	211.6	62.8	148.76	1.98	2.37	150.73
IfR	256.0	42.8	207.06	7.34	4.84	214.39
NeSyC	188.1	183.6	2067.74	13.62	11.27	2081.36
WALL-E	210.3	146.2	319.72	2.81	2.19	322.53
OntoAct (ours)	334.1	91.9	219.66	0.91	2.39	220.57

Table A24: **Mars knowledge-construction resource use.** Calls, In, Out, and Total tokens are per construction run. NeSyC constructs its rules once per world (eight runs); its rule refinements during play are counted as evaluation (Table A23). ReAct and ReAct* require no LLM-based knowledge construction; their input length is undefined. Input/call is mean input tokens per call; Total tokens sums input and output before rounding. Token quantities are in thousands (k).

Method	Calls	In (k)	Out (k)	Input/call (k)	Total tokens (k)
Luna					
ReAct	0	0	0	—	0
ReAct*	0	0	0	—	0
Reflexion	20.0	12.76	2.61	0.64	15.38
IfR	20.0	18.28	3.88	0.91	22.16
NeSyC	63.5	3768.39	33.08	59.34	3801.47
WALL-E	255.3	1069.25	24.59	4.19	1093.84
OntoAct (ours)	1.0	10.91	1.34	10.91	12.25
Luna^{R}					
ReAct	0	0	0	—	0
ReAct*	0	0	0	—	0
Reflexion	20.0	12.76	5.01	0.64	17.77
IfR	20.0	18.28	7.04	0.91	25.32
NeSyC	63.5	3739.88	79.09	58.90	3818.98
WALL-E	228.1	1001.05	51.70	4.39	1052.75
OntoAct (ours)	1.0	11.00	3.19	11.00	14.19
Terra					
ReAct	0	0	0	—	0
ReAct*	0	0	0	—	0
Reflexion	20.0	12.76	3.25	0.64	16.02
IfR	20.0	18.28	3.14	0.91	21.42
NeSyC	63.5	3729.85	12.96	58.74	3742.80
WALL-E	225.6	996.33	11.87	4.42	1008.20
OntoAct (ours)	1.0	11.20	1.12	11.20	12.32

Table A25: **ALFWorld knowledge-construction LLM usage (Luna).** Totals for one construction run per method on the shared 238-transition corpus. Each resulting knowledge base is reused across 134 evaluation tasks. Token quantities use thousands of tokens (k); Total tokens sums input and output before rounding. Costs exclude corpus collection and evaluation.

Method	Calls	In (k)	Out (k)	Total tokens (k)
NeSyC	79	1989.08	8.62	1997.70
WALL-E	254	349.37	14.84	364.21
OntoAct (ours)	15	116.43	0.94	117.36

D.7 TACIT-FLE CANDIDATE AND VERIFICATION DIAGNOSTICS

These post hoc analyses use the 104 Luna episodes in Table 1: 28 small, 36 medium, and 40 large episodes from 15 plant worlds per scale. Anonymized identifiers are decoded only in this analysis, to compare a proposed target with the benchmark's hidden fault target. Target alignment measures whether a candidate names the fault. NeSyC is excluded from these diagnostics because its inspected trajectories contain no repair executions.

Replacement candidates and coverage. For OntoAct and WALL-E, we extract replacement proposals before their native action-verification procedures, including rejected proposals. For IfR, we parse replacement steps in each newly generated plan, including steps that are not subsequently executed. Quoted dialogue history, inspection targets, and explanation-only text are excluded. Within each episode, we retain distinct replacement targets in order of first proposal. If $C_e^{(k)}$ is the first k distinct targets and c_e^* the true fault, coverage is

$$\text{Coverage } @k = \frac{1}{N}\sum_{e=1}^{N} \mathbb{1}\{c_e^* \in C_e^{(k)}\}. \tag{A1}$$

Episodes with fewer than k candidates retain their complete observed candidate set and remain in the denominator. Table 2 reports $k = 3$ and full-sequence counts on medium plants; Table A26 additionally reports $k = 1$ and $k = 5$ across all scales. Candidate precision divides the number of target-aligned episode–target pairs by the number of distinct proposed episode–target pairs. We use 20,000 paired world-cluster bootstrap draws within each scale and report percentile 95% intervals. The coverage-at-three difference between OntoAct and IfR has intervals of [26.7, 66.7], [33.3, 81.0], and [62.2, 90.2] percentage points across the three scales.

Table A26: **Replacement-candidate diagnostics.** Targets is the mean number of distinct proposed replacement targets per episode. C@k and C@all are true-fault coverage (%) within the first k targets and the complete sequence. Precision is correct-target/total episode–target pairs. Small, medium, and large contain 28, 36, and 40 episodes.

Scale	Method	Targets	C@1	C@3	C@5	C@all	Precision
Small	OntoAct (ours)	2.68	50.0	75.0	82.1	82.1	23/75
	WALL-E	4.79	3.6	35.7	50.0	50.0	14/134
	IfR	5.79	10.7	28.6	50.0	64.3	18/162
Medium	OntoAct (ours)	3.36	33.3	69.4	75.0	88.9	32/121
	WALL-E	5.14	0.0	27.8	36.1	47.2	17/185
	IfR	9.00	0.0	11.1	38.9	66.7	24/324
Large	OntoAct (ours)	2.78	57.5	87.5	92.5	97.5	39/111
	WALL-E	5.70	5.0	22.5	37.5	52.5	21/228
	IfR	14.38	0.0	10.0	20.0	65.0	26/575

Table A27: **Selection within the hypothesis ledger.** Target-aligned/all tracked at baseline, then target-aligned/retained at later checkpoints (retained: open or confirmed). The last two rows give final target-aligned retention and off-target refutation; terminal-only hypotheses (9/15/14 by scale) are excluded.

Checkpoint / measure	Small	Medium	Large
All tracked	20/140	28/272	34/192
Checkpoint +1	19/68	23/112	31/98
Checkpoint +2	19/48	21/61	28/68
Checkpoint +3	19/32	20/36	28/42
Final	18/21	20/21	25/32
True-target retention	18/20	20/28	25/34
Off-target removal	117/120	243/244	151/158

Native hypothesis selection. A registered OntoAct hypothesis is identified by its cause, pattern reference, and ordered atoms. The registered totals are 149, 287, and 206 across small, medium, and large, corresponding to 5.32, 7.97, and 5.15 per episode. Different conditions can name the same cause: the medium cohort contains 155 distinct episode–cause pairs, or 4.31 per episode. For the selection analysis, we follow the 140, 272, and 192 hypotheses that are visible at the start of a later turn. The remaining 9, 15, and 14 hypotheses are admitted only on the terminal turn and are excluded from both the initial and retained sets. Retained hypotheses are open or confirmed at the start of the final environment turn. Table A27 gives target precision, target-aligned retention, and off-target removal on this matched population. Figure 3 follows this same population at three intermediate checkpoints aligned by each hypothesis's admission. For a hypothesis admitted at turn a in an episode of T turns, checkpoint $+k$ reads its status at the start of turn $\min(a + k, T)$, for $k \in \{1, 2, 3\}$; Final reads turn T. An episode that has already ended retains its last recorded status. The Init. baseline includes every hypothesis in the matched population before selection. At each later checkpoint, target precision is the fraction naming the true fault among open or confirmed hypotheses. Selection includes updates after unsuccessful interventions.

Table A28: **Precision of executed replacement commands.** Each cell gives the percentage and correct-target/total commands. Repeated targets are counted each time. These final executions are distinct from proposed candidates and native gate passes.

Method	Small	Medium	Large
OntoAct (ours)	29.3% (22/75)	25.4% (29/114)	36.4% (36/99)
WALL-E	2.3% (3/133)	5.2% (12/232)	3.8% (14/364)
IfR	9.9% (18/182)	5.6% (24/426)	3.7% (26/694)
ReAct	8.9% (11/124)	4.7% (15/321)	3.5% (17/482)
ReAct*	5.6% (11/197)	4.3% (17/392)	2.2% (15/688)
Reflexion	8.6% (12/140)	3.1% (12/387)	1.3% (8/600)

Explicit refutation and execution. We separately link repair proposals, verifier feedback, and the action executed in the same turn. For explicit refutation, we count one choice per episode, environment turn, repair verb, and target, over both replacement and refill proposals. In the medium OntoAct cohort, 139 choices receive explicit refutation feedback; 137 target a noncausal unit, and all 137 are withheld or revised in that turn. Deduplicating across turns leaves 31 off-target choices among 33 distinct episode–verb–target choices. Explicit refutation is a subset of all negative gate decisions. Across all scales, 233 of 241 explicitly refuted choices are off-target and withheld in the corresponding turn. Refutations can follow an unsuccessful earlier repair, contradicted observations, or an explained anomaly disappearing.

For broader gate diagnostics, we use the first decision on each distinct replacement target within a turn. For OntoAct, negative decisions include explicit repair, practice, or repetition feedback. For WALL-E, the decision is the final world-model Boolean after any code-conflict feedback, before execution at the proposal limit. IfR provides post-action explanations and replanning, without a corresponding pre-action verifier veto; a verifier rejection rate is therefore undefined for this method. Table A28 counts the replacement commands actually executed, including repeated targets and actions executed at the proposal limit.

Repeated off-target repairs provide a separate outcome diagnostic. We count an executed repair command to a noncausal target as repeated if an earlier repair command in the same episode targeted that unit, including a change of repair verb. Across the 104 matched episodes, OntoAct, WALL-E, and IfR issue 13, 633, and 280 such repeats, respectively.

Illustrative traces. The following cases illustrate distinct mechanisms. In a medium chain task, OntoAct proposes replacing the faulty input storage under an induced diagnostic pattern but omits its required inflow atom. The claim is not admitted, and the first executed action opens the storage. The next proposal adds the missing atom and replaces the target, completing the episode in two turns with no spurious intervention. On the same task, WALL-E initially predicts failure for an off-target locomotive replacement at turn 18. Executable-rule feedback changes the prediction to success, and the replacement executes; the episode finishes unresolved after 60 turns with 11 spurious interventions. The contrast illustrates how action-feasibility feedback can differ from evidence that a target explains the fault.

In a medium elimination task, OntoAct's five tracked hypotheses narrow to one naming the true target, `st-04`. Three hypotheses on `st-00` or `st-01` are refuted after earlier replacements fail to end the episode, and the anomaly explained by the `st-02` hypothesis clears. The `st-04` hypothesis remains confirmed, and the episode succeeds in eight turns with two spurious interventions.

E Experimental Setup and Reproducibility

This appendix specifies the reference corpora, evaluation schedules, model configurations, and aggregation rules used in Section 4.1. Knowledge construction and prompts are described in Appendices B.2, B.4, and C.2.

E.1 Reference corpora and evaluation tasks

Each evaluation episode starts with fresh interaction history and transient state.

Tacit-FLE. We use the fixed reference corpora and evaluation population in Appendix A.4, with the scale-specific episode counts and turn budgets in Table A6. Corpus access follows the OntoAct construction procedure in Appendix B.2 and the baseline procedures in Appendix C.2.

For WALL-E in Table 1, each scale uses one corpus-learned knowledge base for each of its 15 worlds. All knowledge bases remain fixed during evaluation. Each episode contributes one outcome under the procedure in Appendix C.2.

Mars. Following the evaluation configuration of Zhou et al. (2025b), we use Default and seven worlds formed by changing terrain, survival mechanics, task dependencies, or their combinations. Methods share the same reference corpus of 200 recorded transitions per world. Each record contains the observed state, action, outcome, feedback, and subsequent state. For document-based processing, consecutive groups of ten records form 20 documents per world.

The agent receives a textual local observation, including a 9×9 view, nearby objects, inventory, and health, food, drink, and energy levels. The high-level commands are `mine`, `attack`, `explore`, `make`, `place`, and `sleep`. Each episode is capped at 1,000 environment steps and 1,000 agent turns, with ten turns of interaction history. A high-level command can consume multiple environment steps. Each method is evaluated on five seeds per world, giving 40 episodes across eight worlds for each model condition.

ALFWorld. We use the text interface and all 134 unseen evaluation tasks, with the six task families and their sizes listed in Table A21. The reference corpus contains 238 transitions from six training games, one per task family. In each training game, 20 exploratory probes precede an expert-policy trajectory; the resulting corpus contains both successful and unsuccessful interactions. The same fixed corpus is used to construct each method's knowledge. Each unseen task is evaluated once with a budget of 49 environment steps. OntoAct and WALL-E use standard textual observations; the reported NeSyC result additionally provides initial object locations and inventory (Appendix D.5). Environment steps include submitted `think` and idle commands; internal LLM calls are counted separately. We report the Luna condition for this benchmark.

E.2 Model and inference settings

The Luna condition uses the `gpt-5.6-luna` model with reasoning effort set to `none`. The Luna^R condition uses the same model with reasoning effort `medium`; the Terra condition uses `gpt-5.6-terra` with effort `none`. Each condition uses its designated model for both knowledge construction and evaluation. We retain the provider's default temperature settings and use the same model configuration across methods within a comparison. The Mars and ALFWorld runs cap each completion at 8,192 tokens. Retrieval uses the OpenAI `text-embedding-3-small` embedding model. The environment-specific proposal budgets of OntoAct are listed in Table A13.

E.3 TASK METRICS AND AGGREGATION

Tacit-FLE. We aggregate the metrics defined in Appendix A.4.1 over all scheduled episodes at each plant scale and, where specified, by task type.

Mars. Episode reward sums the environment rewards: one unit for the first completion of each achievement, together with a health-change contribution of 0.1 per health unit. We report the mean and sample standard deviation of episode reward over five evaluation episodes in each world, using denominator $n-1$ for the sample variance. For achievement i in evaluation episode r, let $z_{ir}=1$ if the achievement is attained at least once and 0 otherwise. Following the achievement-success-rate metric of Tang et al. (2024b), let $p_i=(100/n)\sum_{r=1}^{n} z_{ir}$ be the percentage of episodes attaining achievement i, with $n=5$. The achievement score is

$$\text{Score}=\exp\left(\frac{1}{22}\sum_{i=1}^{22}\log(1+p_i)\right)-1. \tag{A2}$$

Success rates are computed across the five episodes within each world, model, and method before applying the geometric aggregation. This yields one Score per cell. Following the average-row convention of Zhou et al. (2025b), the cross-world summary excludes the Default world. The Score Average is the arithmetic mean of the seven modified-world Scores. For Reward, we first average the seven modified worlds within each matched environment seed; the Average row reports the mean and sample standard deviation across the five resulting seed averages.

ALFWorld. Task success is the fraction of the 134 unseen tasks for which the environment reports completion within the step budget. Mean environment steps are the total executed steps divided by the number of evaluation tasks, including unsuccessful episodes. Environment steps count executed commands; internal planning and verification calls contribute to LLM usage instead.