

Prompt, Prove, Patch: The Neuro-Symbolic Loop for General Policy Synthesis

Dominik Drexler¹ Markus Fritzsche¹ Farid Musayev¹ Simon Ståhlberg²
29th of June, 2026

¹ Linköping University, Linköping, Sweden,

² RWTH Aachen University, Aachen, Germany

Generalized Classical Planning: One Controller for a Whole Domain

- ▶ A *general policy* solves *every* task of a possibly infinite domain family $\mathcal{Q}$ — objects, initial state, and goal all vary.
- ▶ It cannot refer to object names or counts; only to abstract, task-independent state properties: *is some package in the wrong city?*, *is a vehicle co-located with it?*
- ▶ **Our target:** compact *rule-based* policies over *relational features* — one abstract controller for a whole domain, not a per-task plan.

Three Approaches, Three Failure Modes

- ▶ **Symbolic synthesis:** compiles all training tasks into a single combinatorial search — a classical-planning episode, (weighted Max-)SAT, or ASP $\Rightarrow$ one monolithic problem, no refinement on failure; expressivity capped by a pre-enumerated feature pool.
- ▶ **GNN learners:** generalize across task sizes, but bounded by C_2 and systematically fail where more expressivity is required.
- ▶ **LLMs as one-shot generators:** broad world knowledge, but no formal guarantees — failure yields only a binary pass/fail, no structural cause.

- ▶ A **policy** π is a relation over state pairs: $(s, s') \in \pi$ means that transition is allowed.
- ▶ A policy π is **general** for a class of tasks $\mathcal{Q}$ iff for every task $\tau \in \mathcal{Q}$, every maximal π -path is finite and ends in a goal.
- ▶ We represent π by rules over relational Boolean and numerical features.
- ▶ Rules test the current abstract state and constrain qualitative change.

Why Not Generate Python?

- ▶ **Python:** Turing-complete and too unconstrained for automatic guarantees.
- ▶ Programs can use object identities, arbitrary control flow, and non-terminating loops.
- ▶ **Relational features:** no object names; policies act through task-independent state properties.
- ▶ **Rule-based qualitative effects:** restricted enough to check whether policy executions are finite.

The Feature Language: Compositional DL Syntax

- Concepts denote sets of objects; roles denote binary relations. Both are evaluated on $(s, \mathcal{G})$.

$$\begin{aligned} C ::= & \top \mid \perp \mid A_s \mid A_{\mathcal{G}}^+ \mid A_{\mathcal{G}}^- \mid \neg C \mid C \sqcap C' \mid C \sqcup C' \\ & \mid \exists R.C \mid \forall R.C \mid \geq n R.C \mid \leq n R.C \mid = n R.C \\ & \mid R \subseteq R' \mid R \doteq R' \mid R : \{i_1, \dots, i_k\} \mid \{i_1, \dots, i_k\} \end{aligned}$$

$$\begin{aligned} R ::= & U \mid P_s \mid P_{\mathcal{G}}^+ \mid P_{\mathcal{G}}^- \mid \neg R \mid R \sqcap R' \mid R \sqcup R' \mid R^{-1} \\ & \mid R \circ R' \mid R^+ \mid R^* \mid R|C \mid id(C) \end{aligned}$$

- **Features:** zero-arity tests, $C \neq \emptyset$, $R \neq \emptyset$, $|C|$, $|R|$, and $d_R(C, C')$.
- Role composition and transitive closure go beyond C_2 -style feature languages and GNNs.
- Unbounded nesting and composition give a feature space far beyond fixed pre-enumerated pools.

- ▶ The LLM agent authors the policy; symbolic tools analyze finite transition graphs.

Prompt → **Prove** → **Patch** → **Prompt**

- ▶ **ProvePolicy:** checks *all* maximal π -paths, not one rollout.
- ▶ Failed proofs return structural counterexamples:
 - *open state*: reachable non-goal with no accepted transition,
 - *cycle*: policy loops without reaching a goal.
- ▶ **GenerateFeatures:** turns counterexamples into candidate DL features for the next prompt.
- ▶ We do not provide distance estimators nor unsolvability detectors.
- ▶ The agent may also write small Python analyses to inspect counterexamples.

Learning Results: 15 Domains

Domain	#I	# Φ	$C^{\max}$	C^{sum}	#R	Proved	Valid.
Barman	69	18	49	221	18	98/98	77/77
Blocks-4	8	8	39	109	5	100/100	100/100
Childsnack	3	9	19	78	10	100/100	66/66
Driverlog	8	28	31	462	22	100/100	100/100
Floortile	329	12	100	465	11	97/100	100/100
Grid	18	12	28	141	14	100/100	100/100
Hiking-Bin.	28	21	41	419	31	95/95	0/0
Logistics	9	16	29	335	14	100/100	100/100
Satellite	3	9	24	125	7	100/100	100/100
Blocks-3	11	11	53	198	9	100/100	100/100
Delivery	1	6	18	74	4	100/100	100/100
Ferry	1	4	10	26	4	100/100	100/100
Goldminer	10	9	9	27	7	100/100	88/88
Gripper	2	5	10	33	4	5/5	5/5
Miconic	2	12	8	65	6	100/100	100/100

#I iterations, # Φ features, $C^{\max}/C^{\text{sum}}$ max/sum feature complexity, #R rules, Proved (train), Valid. (greedy).

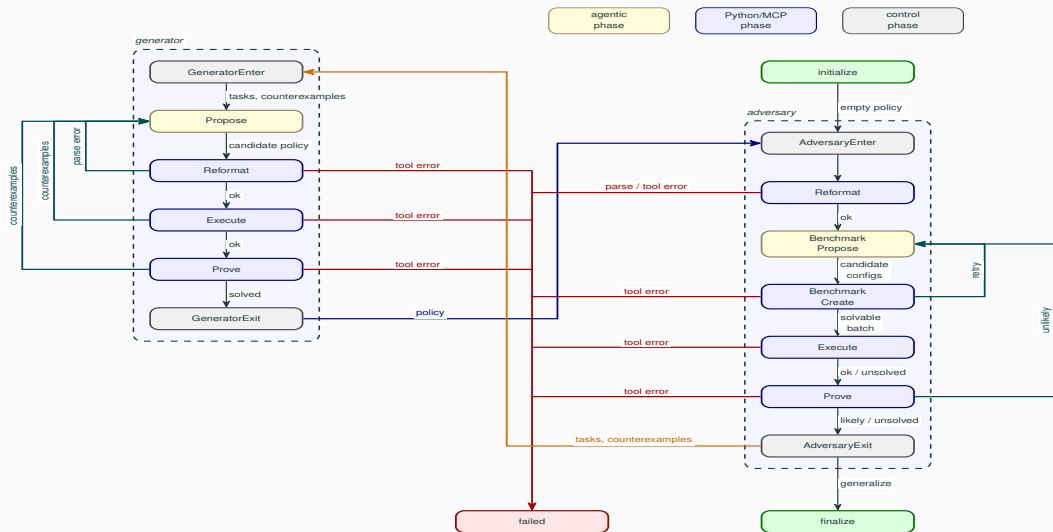
Test Evaluation vs. LAMA and Dual-BFWS: IPC Benchmarks

Domain	LAMA		BFWS		Policy	
	S	T	S	T	S	T
Barman-11 (20)	20	0.90	20	0.05	20	0.55
Barman-14 (20)	20	2.07	20	0.10	20	0.81
Blocks (35)	35	0.01	35	0.01	35	0.11
Childsnack (20)	6	0.21	9	0.83	20	1.01
Driverlog (20)	20	0.03	20	0.00	15	0.16
Floortile-11 (20)	7	41.84	4	0.07	20	0.16
Floortile-14 (20)	2	3.54	2	0.03	20	0.17
Grid (5)	5	0.08	5	0.03	5	0.61
Hiking-Bin. (20)	20	0.88	14	0.05	20	0.93
Logistics-00 (28)	28	0.01	28	0.01	28	0.17
Logistics-98 (35)	35	0.18	34	1.74	35	4.34
Satellite (36)	36	0.14	33	0.47	32	3.28
Gripper (20)	20	0.01	20	0.00	20	0.12
Miconic (150)	150	0.02	150	0.01	150	0.23
Total (449)	404	0.19	394	0.04	440	0.43

Greedy execution; 8 GB, 30 min. S = solved, T = geom. mean time (s) over *commonly* solved tasks.

- **Logistics:** unload / same-city truck / route to airport / fly / destination-city truck. Needs *role composition* to recognize “in the destination city” — usually excluded by bottom-up pools, here essential.
- **Driverlog:** no generalizing policy. Set-distance features cannot bind each driver to *its own* goal — one driver at another’s goal gives distance 0 but a wrong assignment. Remedies: per-driver distance sum, or indexical policies (Bonet, Drexler, and Geffner 2024).
- **Hiking-Binary:** 21 features (≤ 41), 31 rules. Relies on $\geq n$ *R.C* multiplicity (both partners co-located, two cars nearby) — hard for explicit enumeration to manage.

Work in Progress: Unfolding the loop



- ▶ Agents can synthesize rule-based general policies over complex relational feature grammars.
- ▶ Counterexamples provide a strong signal for the agent to improve its current policy.
- ▶ Target languages can preserve generalization invariance by construction, and can allow for formal verification of desired properties such as termination.