Phase 0 : Detailed Research Documentation

Project: Continual RL - Phase-0 (Symbolic GridWorld Sanity)

Prepared by: Rishav Tewari

Date: 10/01/2026

Introduction

This document records Phase-0 of the Continual RL pipeline: a minimal, deterministic validation of the symbolic planning + low-level execution architecture. The purpose of Phase-0 is to (a) establish a reproducible experimental skeleton, (b) validate reward semantics and reachability, and (c) provide diagnostic baselines that reveal pathologies before attempting learned policies or world models.

All artifacts (notebooks, logs, and images) are attached to this workspace. The canonical notebook for Phase-0 is `Phase0_Symbolic_Gridworld_Sanity.ipynb`. The full research log is available as `Phase0_Research_Log.txt`.

My Role in this Research

I acted as the research lead and experimental designer for Phase-0. Responsibilities performed:

- Defined the minimal, deterministic MDP semantics used for sanity checks.
- Implemented a reproducible environment (`SimpleGridEnv`) with deterministic reset and sparse terminal reward.
- Designed and implemented a symbolic planner interface and a low-level greedy executor to validate end-to-end behavior.
- Implemented formal reachability checks (BFS) to guarantee that the environment is solvable and to prevent silent failures in later stages.
- Executed diagnostics comparing random low-level action sequences and random symbolic plan sampling; interpreted results and generated recommendations.

What Happened at Every Step (and Why)

Step 1 : Workspace setup

- Action: Mounted Google Drive and created canonical project directories under `/content/drive/MyDrive/Continual_RL_Phi2`.
- Why: Maintain reproducibility, versioned checkpoints, and a consistent path structure for collaborations.
- Verification: Directory tree created successfully.

Step 2 : Dependency installation and smoke tests

- Action: Installed core libraries (gymnasium, stable-baselines3, PyTorch, matplotlib, pandas, seaborn).
- Why: Ensure a stable runtime and reproducible package versions for downstream experiments.

- Verification: GPU available (Tesla T4); CartPole smoke test passed.

Step 3 : Deterministic environment implementation

- Action: Implemented `SimpleGridEnv` (fixed start `(0,0)`, fixed goal `(W-1,H-1)`, deterministic transitions, terminal reward `r=1` if and only if `agent_pos == goal_pos`).
- Why: Base MDP for sanity, remove stochasticity and partial rewards to isolate planner/executor behavior.
- Verification: Deterministic trajectory produced `Total reward: 1` upon reaching goal.

Step 4 : Symbolic planner + executor (Phase-0 canonical plan)

- Action: Implemented `symbolic_plan(obs)` that returns `[('goto','green','goal')]`. Implemented `execute_plan(env, plan)` that greedily translates the single symbol to primitive moves.
- Why: Provide the minimal symbolic interface and oracle grounding for a reproducible baseline.
- Verification: Final Plan and Total Reward printed as expected.

Step 5 : Formal reachability (BFS)

- Action: Implemented a deterministic BFS reachability function to confirm solvability of the grid from the start position.
- Why: Prevents silent failures later (e.g., unreachable goals due to obstacles or state inconsistencies).
- Verification: `Reachable: True`.

Diagnostics : Random Behavior Experiments

- Action: Monte Carlo diagnostics comparing:
- Random low-level action sequences of length 6.
- Random symbolic plans sampled from a small symbolic set, with naive grounding.
- Why: Measure baseline success rates and expose any oracle effects.
- Observed results (verbatim):
- `Random low-level sequences success rate (seq_len=6): 0.0036`
- `Random symbolic plans success rate (naive grounding): 0.5355`

These results reveal a low probability of accidental success under random primitive actions, and a high success rate for random symbolic plans **only** because the sampling distribution included the correct operator and grounding was oracle-like.

How I Implemented and Resolved Issues

- **Determinism:** I fixed both the reset and goal positions to remove nondeterminism. This ensures every run of the same code produces identical outputs for Phase-0.

- **Oracle grounding clarity:** I intentionally separated planner (symbol emission) and executor (low-level controller). The Phase-0 executor is an oracle greedy controller; this separation was implemented deliberately to measure the distinction between symbol correctness and execution competence.

- **Reproducibility guarantees:** I recorded package versions, runtime GPU, directory layout, and printed the exact outputs (see `Phase0_Research_Log.txt`) to ensure others can replicate the Phase-0 results.

- **Diagnostics design:** To expose the oracle effect, I added Monte Carlo sampling of both low-level sequences and symbolic plan emissions and recorded success rates, illustrating the difference in semantics.

Failures Faced and How I Resolved Them

1. **Perceptual/dataset mismatch (design uncertainty)**

2. *Issue:* Initial confusion whether reward was probabilistic or deterministic.

3. *Resolution:* Re-examined reward code and confirmed `r = 1 iff agent_pos == goal_pos`. Documented this explicitly in the research log.

4. **Oracle grounding inflating symbolic baselines**

5. *Issue:* Random symbolic plan success appeared deceptively high due to an oracle executor.

6. *Resolution:* Performed controlled diagnostics to quantify the effect and recommended rigorous experimental corrections (learned or stochastic grounding or separate performance metrics).

7. **Potential hidden failure modes in later phases**

8. *Issue:* If Phase-0 had silent unsolvable configurations, downstream training would silently fail.

9. *Resolution:* Implemented BFS reachability checks and enforced explicit assertions to ensure solvability prior to launching training runs.

Results and How They Were Achieved

- **Canonical Phase-0 result (reproducible):**

```
Final Plan: [('goto', 'green', 'goal')]
Total Reward: 1
```

- **Environment reset and final state (verbatim):**

```
Initial obs: {'agent_pos': (0, 0), 'goal_pos': (3, 3)}
Final obs: {'agent_pos': (3, 3), 'goal_pos': (3, 3)}
Total reward: 1
```

- **Reachability check:** `Reachable: True`
- **Diagnostics:**
- `Random low-level sequences success rate (seq_len=6): 0.0036`
- `Random symbolic plans success rate (naive grounding): 0.5355`

Interpretation: The MDP and reward are strictly deterministic. The high symbolic success rate is an artifact of the small symbolic sample space plus oracle grounding, not evidence of learning competence.