Phase-Aware Source-Aligned Conformance Testing for Multi-Agent Environment Adapters

Safiullah Baig*
Independent Researcher
ORCID: 0009-0008-5547-6088
*Corresponding author
Postal and email details supplied privately at submission

Results manuscript — 2 September 2026

Running title: Phase-Aware MARL Adapter Conformance

Abstract

Adapters between multi-agent reinforcement-learning APIs may expose unequal event schedules: one simultaneous source decision can become several Agent Environment Cycle calls, while chance resolution can be hidden within one destination call. Destination-local and matched-timestep checks therefore do not establish source-task preservation. MARLRefine aligns a separately loaded native execution with variable-length adapter-call blocks while retaining destination-only buffer and cleanup phases. In a complete registry census of one OpenSpiel–Shimmy–PettingZoo stack, 105 reserved game types received eight fixed policies (840 traces). Of these, 808 produced semantic verdicts: 90 passed and 718 failed; 32 traces from four sampled-stochastic games were inapplicable. Findings in 99 games clustered into three implementation roots: stale reward exposure on non-advancing calls, a chance-inclusive destination clock that can truncate before the native decision horizon, and silent terminal projection of an unsupported mean-field node. The stock destination API test missed all three root witnesses. Isolated treatment succeeded for the reward root, failed the strict regression criterion for the clock root, and was not applicable as a mean-field implementation. In a sealed 24-mutant cohort, MARLRefine made 19 semantic kills, recorded four additional crash-only detections and one survivor, and raised no clean-reference alarms, missing the prespecified 20/24 strong-sensitivity threshold. The study is locally Git-frozen, not preregistered, and tests pathwise conformance conditional on recorded chance transcripts; it does not claim distributional equivalence or ecosystem-wide prevalence.

Keywords: conformance testing; differential testing; multi-agent reinforcement learning; environment adapters; trace alignment

1 Introduction

An environment adapter is part of an experiment's task specification. If it re-emits a reward, truncates after the wrong clock tick, changes a constructor parameter at reset, or projects an

unsupported source state into ordinary termination, a policy may train on a different process even when every value has a valid type and shape. Destination-side API tests remain essential and exercise dynamic lifecycle and reward bookkeeping, but they need not compare behavior with a separately executed source.

That comparison is not a standard lockstep test when the public schedules differ. A simultaneous move in a partially observable stochastic game (POSG) can be represented in the Agent Environment Cycle (AEC) model as several zero-reward action-queueing steps followed by an environment-resolution step [1]. Conversely, one adapter call can encompass a player action followed by one or more source chance transitions. Insisting on one source event per destination call either rejects legitimate conversions or compares unrelated events. Compressing the destination trace only at source boundaries avoids that problem, but discards observable intermediate behavior: a stale reward shown during a buffer-only call disappears under compression.

MARLRefine treats alignment and conformance as separate operations. An event-retaining aligner over the recorded ledgers uses independently replay-anchored source progress to group contiguous source and destination events into zero-to-many, one-to-many, and many-to-one blocks. A pure core comparator and boundary-local integration checks then evaluate only obligations whose contracts or explicit study policies and applicability conditions are established. A separately loaded native execution supplies expected values; adapter internals are used only to synchronize boundaries and are never accepted as an oracle.

This paper makes two methodological contributions:

1. It gives an executable event and alignment schema for testing adapters with unequal source and destination schedules. The schema operationalizes, but does not replace, established action refinement, input/output conformance, and stuttering/skipping refinement [2–4].
2. It defines an applicability-scoped MARL conformance checker from pinned source/destination contracts and explicit study policies, covering progress, two reward channels, decision clocks, lifecycle, discovery-only configuration provenance, source state kinds, agent scheduling, legal actions, and declared observations, including per-delivery reward accumulation and explicit strong/weak state-oracle accounting.

We evaluate those contributions through a complete registry census with a predeclared 7/1/105 development/exclusion/reserved partition and a 840-trace schedule. Protocol version 1.1 also mandates a separately manifested 48-candidate mutation pool with a fixed, outcome-blind selection target of 24 eligible mutants. The empirical contribution is a root-level account of three recurring semantic mechanisms, their visibility to weaker comparators, exact-call localization, replay, and isolated treatment, together with prespecified mutation sensitivity and clean-reference cost.

Our empirical scope is one source framework and one released adapter. We use "no observed violation" for passing finite traces and do not generalize to all MARL adapters.

2 Contracts and Claim Boundary

Source and destination events. Let a native execution produce atomic source events $U = \langle u_1, \ldots, u_m \rangle$. Each event records source progress, node kind, action or chance outcome, immediate reward vector, cumulative return, lifecycle, and a SHA-256 digest of OpenSpiel serialization where available, otherwise a tagged history/text fallback; the digest method tag is recorded with the value. Only canonical serialization can earn O1 pass credit; a fallback is diagnostic and

marks O1 unevaluated. Legal actions and the declared representation are compared online before action selection; complete expected/observed values are persisted when O8 fails rather than duplicated for every passing event. A *source boundary* follows one player or joint decision and any immediately resulting chance events needed to reach the next player, simultaneous, mean-field, or terminal node. Thus one decision macrotransition may contain several atomic events.

The adapter produces destination microevents $V = \langle v_1, \ldots, v_n \rangle$, one per AEC `step` call, including terminal cleanup calls. Each v_j records a progress annotation $p(v_j)$, instantaneous `env.rewards`, the reward returned to the selected agent by `last()`, termination, truncation, submitted action, and synchronization metadata. The two reward channels are not interchangeable: the AEC contract defines `rewards` for the most recent action, whereas `last()` exposes reward accumulated since the selected agent last acted [5]. The runner therefore maintains an independent native accumulator for each player, compares it at every observed delivery, and resets only that player's accumulator at the matching step boundary.

At a live source boundary, the source-side observation oracle follows the adapter's declared representation order: observation tensor, information-state tensor, observation string, then information-state string. The destination's raw container type, element count, shape, and dtype are recorded and compared exactly before any value normalization. Only then are the source expectation and destination values represented in a common numeric form for the floating-value comparison. Thus casting cannot repair a wrong public shape or dtype. The representation priority is an adapter-specific oracle choice, not a claim that the representations are interchangeable.

Progress annotation and trust boundary. For destination call v_j, $p(v_j)$ is the cumulative number of atomic source transitions successfully replayed by the harness on the separately loaded native state after that call. It is not the adapter's `game_length`, wrapped-history length, or a semantic value supplied by the adapter. The runner first verifies that the wrapped pre-call history is a prefix of the post-call history; it then checks the submitted player or joint action, replays every disclosed legal chance outcome, and verifies native/adapted boundary identity. Each event stores the progress before and after the call, replayed event count and event-progress sequence, wrapped-history delta, and the method identifier. A tag/anchor disagreement is an instrumentation failure rather than an adapter-semantic finding.

This instrumentation assumes that wrapped history disclosure is complete, ordered, append-only, and nonperturbing. A faulty but plausible disclosure can therefore defeat alignment; the supported claim is white-box source alignment, not black-box equivalence. Two prospectively sealed progress-history corruption controls exercise lag and overcount independently of semantic-output mutants. They are scored as instrumentation controls outside the 24-mutant denominator and cannot receive semantic adapter-defect credit.

Non-bijective alignment. Starting from $p_0 = 0$, the aligner scans V without deleting events. Calls with $p(v_j) = p_0$ are buffered. When a call advances to $p(v_j) = q > p_0$, the aligner emits

$$B_k = (U[p_0 : q],\ V[\ell : j{+}1])\,, \tag{1}$$

where ℓ is the first destination index after the preceding commit. It then sets $p_0 \leftarrow q$. A block may therefore be one-to-many when several destination calls buffer before a commit, or many-to-one when a destination call encompasses multiple atomic source events. An uncommitted

suffix becomes a zero-source stutter block or, after source terminality, a terminal-cleanup tail. Regressing, over-running, or incomplete progress is preserved as input and diagnosed rather than repaired by the aligner. Contiguous half-open spans cover every destination call and every consumed source-prefix event; any unmatched source suffix remains explicitly in the source ledger and is diagnosed as incomplete progress.

Under the stated instrumentation assumptions, if the destination progress sequence is nondecreasing, bounded by m, and each strict increase names the consecutive source events actually replayed, the increase indices uniquely determine the contiguous blocks in equation (1). Cursor induction gives original order, no destination-event deletion or duplication, and exactly-once coverage of every source event through $\max_j p(v_j)$. If the final progress equals m, the source ledger is covered completely. This elementary partition property characterizes the algorithm; it is not a new refinement theorem.

This construction is an executable testing device, not a new correctness relation. Action refinement in conformance testing already maps an abstract action to a lower-level action sequence [2]; matching partitions already accommodate both stuttering and skipping [4]; and the POSG-to-AEC construction already supplies the zero-reward buffering principle [1]. Our research question is whether a released implementation preserves the pinned MARL observables across a heterogeneous game registry under finite testing.

Obligations. Table 1 summarizes the eight study obligations. Exact integers, Boolean values, sets, parameters, histories, lifecycle values, raw containers, element counts, shapes, and dtypes use equality. Floating rewards use absolute and relative tolerances of 10^{-12}, a tight float64 roundoff allowance for native return-delta versus public-reward arithmetic. Floating observation values use absolute and relative tolerances of 10^{-7} only after exact raw-signature checks; discrete observations use equality. A prespecified sensitivity report reclassifies stored numeric residuals at 0.1, 1, and 10 times the primary tolerances without changing the primary verdict. Ambiguous contracts produce a mismatch for adjudication, not a counted defect. In particular, the OpenSpiel interface specifies maximum game length in player decisions, counts a simultaneous joint decision once, and excludes chance events [6]; O4 does not equate that clock with raw history length or AEC call count.

For example, if J_k is the destination side of a transition block and I_k its source side, O3 checks componentwise

$$\sum_{u_i \in I_k} r_i^S \;\approx\; \sum_{v_j \in J_k} r_j^{D,\mathrm{inst}}, \qquad R^S_{\mathrm{episode}} \;\approx\; R^{D,\mathrm{delivered}}_{\mathrm{episode}}. \tag{2}$$

The second equality is applicable only to completed episodes. This separation prevents persistent source reward snapshots or AEC accumulation from being mistaken for new immediate reward.

3 System Design

MARLRefine separates three logical layers while coupling the two executions only where synchronization requires it.

Separately loaded native oracle. A separately loaded OpenSpiel game receives harness-selected player or joint actions. It records immutable source events, deriving immediate rewards

Table 1: Contract- and study-policy-derived obligations. Each is evaluated only when its applicability predicate is satisfied.

ID	Obligation	Verdict condition (abridged)
O1	Stutter-state preservation	Projected source state and history do not advance during action buffering.
O2	Stutter reward neutrality	Instantaneous destination reward is zero on buffer-only and cleanup calls.
O3	Exactly-once reward/return	Within each aligned block, $\sum r^D = \sum r^S$; each `last()` delivery matches native per-player accumulation; completed consumer returns equal native returns.
O4	Source-decision clock	Normalized elapsed adapter `game_length` equals independently counted source decisions; a cited mapping is required before defect classification.
O5	Lifecycle preservation	Termination agrees at aligned boundaries; truncation requires a declared external budget; cleanup is inert.
O6	Configuration provenance	In discovery-only nondefault cases, canonical identity and supplied parameters survive construction and reset.
O7	State-kind soundness	Decision, simultaneous, chance, mean-field, and terminal nodes are supported according to contract or rejected explicitly.
O8	Agent schedule and interface projection	In sequential states the selected destination agent equals the source player; a simultaneous cycle contains every required live player exactly once before commit. The mapped legal-action set and raw observation container, count, shape, dtype, and value agree with the declared source representation.

from return deltas and keeping decision count distinct from history length. The oracle does not read the adapter's rewards, clock, lifecycle, or expected values. This matters for fault detection: if an adapter remaps a submitted action, replaying the adapter's internal action history as the native oracle would reproduce the fault on both sides.

Adapter runner. A fresh Shimmy environment receives the same player decisions in AEC order. The runner logs instantaneous and delivered reward vectors separately, as well as actions, lifecycle, masks, observations, configuration, and source-progress instrumentation. Explicit adapter chance outcomes are recorded as synchronization evidence and replayed on the native state. Such executions establish pathwise semantics conditional on that transcript; they do not test sampling probabilities, random-number generator identity, or distributional equivalence. Games whose stochastic events cannot be observed and coupled are marked inapplicable or unalignable, never silently passed. This conditional replay is deliberately weaker than testing a probabilistic trace or moment equivalence [7].

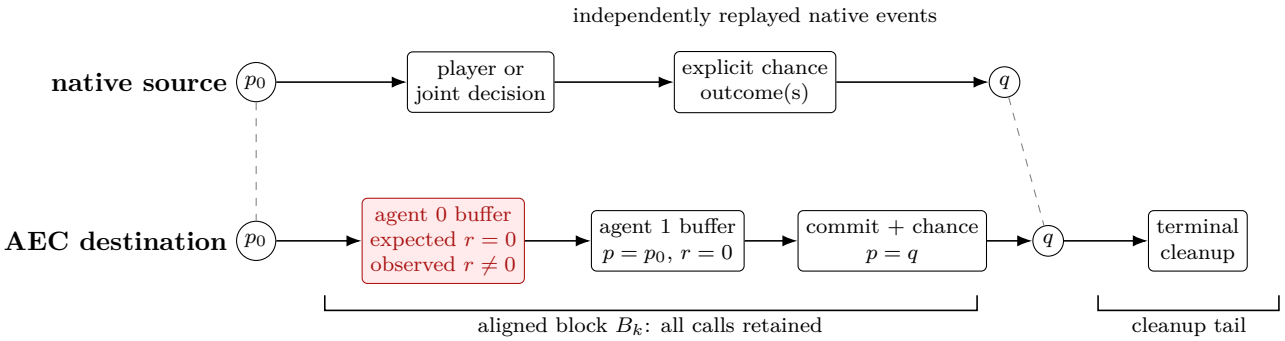

Figure 1: Illustrative discovery-only two-agent alignment. One source macrotransition spans two destination-only buffer calls and one committing call; a terminal cleanup call remains visible outside the transition block. The highlighted stale reward is detectable and localized before compression. Progress uses adapter-disclosed history only as replay-validated synchronization evidence; expected semantics come from the separately loaded native execution and contracts.

Aligner, comparator, and integration checks. Framework-specific objects are normalized into immutable numeric and mapping records before comparison. The aligner implements equation (1); pure reward, progress, cleanup, and aligned-lifecycle checks consume only those records. State-digest, clock, configuration, state-kind, interface, and consumer-delivered-return checks run at the live boundary where framework objects are available. Both paths return structured violations with source and destination spans, expected and observed values, and a stable code. Each case records its policies, chance transcript, boundary digests, and budgets; the replication metadata binds the environment, protocol, code, and manifest identities. Setup failures and inapplicable traces cannot serialize as empty passes.

Counterexamples and causal checks. Every recorded finding is localized to its original destination call or boundary, its buffer/commit/cleanup phase, and the source/destination prefix needed for a standalone replay. The chronologically first trace divergence remains derivable, while a later distinct finding can still serve as a root witness. The report distinguishes exact-call localization from a multi-call boundary and records the prefix/full-trace ratio only as witness position, not as accuracy. Replayability is claimed only after the per-root replay succeeds. Candidate mechanisms are evaluated through isolated treatments: a successful treatment must remove its targeted code under the same requested limits and input policy, introduce no unexpected code, and survive the relevant regression witnesses. Constructor exceptions are not accepted as semantic repairs unless explicit rejection is itself the required capability behavior. The localizer returns original prefixes without claiming delta debugging, semantic minimization, or a minimal counterexample.

AI-assisted research workflow. OpenAI Codex provided substantial implementation assistance, including drafting portions of the code and tests and supporting scripted execution, artifact-integrity checks, initial evidence summaries, citation discovery, and manuscript editing.

4 Study Design

Subject and population. The subject is OpenSpiel 2.0.2 through Shimmy 2.0.1 into PettingZoo 1.27.0 [8]. The construction-census runtime identity records those installed versions together with NumPy 2.5.2 and Python 3.13.2. The primary population is the sorted unique set of all 113 OpenSpiel registry records whose `GameType.default_loadable` flag is true. Every name remains in the denominator if loading, construction, reset, alignment, or bounded execution fails. The census contains 89 sequential, 20 simultaneous, and four mean-field games; 58 are deterministic, 51 expose explicit stochastic chance, and four use sampled stochastic chance.

Development/validation separation. The checker author inspected traces from `coop_box_pushing`, `go`, `kuhn_poker`, `matrix_rps`, `mfg_crowd_modelling`, `nim`, and `tic_tac_toe`. These 7 short names, including every alternate configuration, are discovery and implementation controls. The residual accounting partition contains 106 names. The already observed construction limitation for `crossword` is retained as one descriptive capability exclusion; the other 105 names form the prospective semantic cohort. A corrected observation-shape mismatch on `tic_tac_toe` was an oracle error and is not an adapter result. No pre-authorization output is relabeled held out. The prospective cases use default configurations, so O6 is not applicable there; this design makes no confirmatory released-adapter claim about nondefault-configuration preservation. A separately sealed mutation-only nondefault trigger panel evaluates O6 detector sensitivity without changing that claim boundary.

Trace schedule. Each default case is assigned eight source-legal policies: smallest legal, largest legal, and pseudo-random policies with seeds 0–5. An execution stops at native terminality or after the smaller of a finite positive declared maximum and 1,000 source player/joint decisions; destination calls have a separate 10,000-call safety budget. Seeds are namespaced by the canonical game and configuration hash. The product is exactly 840 game–policy traces. A first legal-action divergence terminates the shared path and is itself an O8 result. The exact protocol, implementation, manifest, lock, container identity, and analysis code are committed and hash-bound before any of the 105 semantic names or any sealed mutation candidate is executed. The selected gate is an explicit local authorization that records `preregistered=false` and `public_archive=false`; this study is prospectively specified but not preregistered. The main manifest binds the separate 48-candidate mutation manifest's exact hash. Outcome precedence keeps execution, alignment, infrastructure, inapplicability, fail, and pass states mutually exclusive. The protocol records the narrow exception-only retry rule and artifact-integrity mechanics; they are not empirical results.

The first authorized attempt completed and sealed the primary and external panels, but the mutation stage stopped before producing an artifact when its self-validator found an inconsistent O8 evidence count. Aggregate primary and external statuses had already been displayed. The narrow correction added an explicit "interface boundary evaluated" annotation to an existing early-stop finding; it changed no actions, replay, destination events, obligation mapping, or outcome rule. Nevertheless, we created a new source revision, regenerated both manifests and the local authorization, and reran all three panels from the beginning. The first attempt remains preserved. The results below are therefore an integrity replication under revision `640b1d89174e44fa7a0440f1d4d296e316cc5b41`, not a blinded or preregistered confirmatory run.

Coverage reporting. Coverage is descriptive rather than inferential. Before execution, registry metadata assigns five mutually exclusive strata: sequential deterministic, sequential stochastic, simultaneous deterministic, simultaneous stochastic, and mean-field. Explicit versus sampled chance remains a secondary tag. The analysis reports status and finding counts by stratum; separate semantic-evidence and execution-completeness axes; terminal-complete versus bounded-prefix runs; reached source node kinds and destination buffer, commit, stutter, and cleanup phases; unique offered and selected game–state–player–action tuples; and cumulative/marginal action coverage after each of the eight fixed policies. Site counts and correlated policy traces are never treated as independent defect samples.

Detection comparator and in-line ablations. The released Shimmy OpenSpiel tests provide contextual upstream evidence: their fixed game list is not the 105-game cohort, some paths sample unseeded action spaces, and the test module is available from the source distribution but is not installed by the wheel in our frozen environment [9]. We therefore freeze and hash the exact source-test bytes and command, but do not score that suite as a root-level cohort comparator [10]. The authorization-gated PettingZoo `api_test` is run once per prospective game for 1,000 cycles with captured diagnostics [11]. We additionally evaluate five project-defined schedule/information ablations: strict one-call lockstep, commit-sampling macro-boundary comparison, whole-block macro-aggregate comparison, endpoint state/lifecycle, and episode return only. The macro-aggregate comparator sums every destination microstep reward in an aligned transition block before comparing it with the source reward; unlike the obligation-aware checker, it does not attribute a discrepancy to a particular destination-only phase. Gymnasium 1.3.0 supplies a related lockstep utility that resets two environments with the same seed and feeds one sampled action to one step of each [12]. It is not executed here because native OpenSpiel and the PettingZoo AEC adapter do not share a Gymnasium `Env` interface; wrapping both only to invoke it would add another adapter under test. The five in-line ablations consume the same serialized paired trace; the stock API test is a separate destination-only execution. A baseline receives detection credit only for a failure causally attributable to the same reachable root. A pass counts as a miss only if the root witness was reached; inapplicability, an unreached path, and unrelated crashes are not scored.

Controls, sealed mutation validation, and adjudication. The pre-freeze clean controls are: a separately loaded native clone replay, the official OpenSpiel simultaneous-to-turn-based transform on the discovery game `matrix_rps` [13], and the canonical PettingZoo Parallel-to-AEC conversion on a synthetic two-agent fixture [14]. Any unexplained deterministic alarm narrows or invalidates the affected obligation before adapter results are interpreted. Previously exercised mutants and direct recreations remain development tests and are excluded from the score. The mandatory mutation manifest instead seals 48 unexecuted, single-hook candidates in a fixed priority order: eight candidates in each of six families covering O2/O3 reward, O1/O8 progress and interface projection, O4 decision clocks, O5 lifecycle, O6 configuration provenance, and O7 special node kinds. Each candidate is applied to the same composite semantic-preserving reference, rather than to the already-failing released adapter.

Screening is paired and outcome-blind. A candidate is eligible only when its clean reference is acceptable, its hook fires, its adapter-facing execution differs from the reference, and it is not behaviorally duplicate within its family. The fixed order selects four eligible candidates per family; reserves are never chosen using checker, ablation, or API-test verdicts. Exhausting a family

reports a short denominator and blocks the prespecified 24-mutant sensitivity claim. A semantic kill is a new full MARLRefine finding signature relative to the matched clean reference with the mutation site reached. Constructor and execution exceptions are reported separately as crash-only detections. The stock API test and all five in-line ablations are likewise reported as paired mutant-minus-reference deltas under their applicability rules. Paired clean-reference alarms are reported as validation cost. The two lag/overcount progress controls must produce named instrumentation failures and remain outside the mutation denominator. Synthetic mutants are never counted as real roots or evidence of real-world defect prevalence.

Candidate real findings require a primary-contract citation, an original first-divergence prefix and successful replay artifact, an isolated causal treatment, reversion evidence, regression across the full matrix, and duplicate clustering. Outcomes are classified as contract-backed defect, mismatch, unsupported capability, oracle issue, or not a defect. The unit of evidence is an implementation root, not a seed, trace, game, agent, or failed assertion.

Research questions. RQ1 asks which violation codes occur, on how many scheduled traces and distinct games, and with what O1–O8 applicability and evaluation coverage. RQ2 asks which causally isolated roots the in-line ablations and separately executed stock API test detect when their witness is reached. RQ3 asks whether MARLRefine identifies the exact destination call and phase, how long the reported prefix is, whether its obligation/code agrees with adjudication, whether replay reproduces the boundary, and what localization resolution each ablation retains. RQ4 records the semantic and public-consumer consequence of each root. RQ5 accounts for every root by repair disposition and reports before/after evidence for attempted isolated repairs. RQ6 asks what fraction of the selected reachable, behaviorally non-equivalent sealed mutants MARLRefine detects overall and within each family; which paired detections are retained by the API test and each ablation; which detections are semantic rather than crash-only; and what alarm cost occurs on the matched clean references.

5 Results

5.1 Construction Census and Discovery Controls

These results were inspected while the checker was developed. They establish feasibility and motivate the prospective experiment, but they are neither confirmatory findings nor held-out evidence.

All 113 registry-marked game types load natively. The released adapter constructs and resets 112; `crossword` fails during construction because its observation tensor shape is unimplemented. This is capability accounting, not a semantic-defect count; the full census is released as `artifacts/registry_census.json`. The stock PettingZoo `api_test`, run for 1,000 cycles per discovery name, passes all seven names (with warnings archived in `artifacts/discovery_api_baselines.json`). On the same selected cases, source-aligned discovery traces expose the candidate mechanisms in table 2. Thus destination-local API conformance and source-semantic conformance are empirically distinct on these development controls; the result does not imply that `api_test` was intended to establish source equivalence.

The three bounded clean semantic-preserving controls in `artifacts/discovery_controls.json` all pass with zero unexplained alarms. A native `tic_tac_toe` execution and separately loaded replay agree across seven one-to-one transitions. The official turn-based transform maps

Table 2: Discovery-only candidate mechanisms and development treatments. A row may cover several symptoms; these rows are not publication-grade defect counts.

Candidate mechanism	Development witness and obligation	Treatment evidence
Reward refresh during buffer/cleanup phases	`coop_box_pushing`, `nim`; O2/O3	Targeted reward codes removed on both witnesses; no unexpected codes.
Initial offset plus chance-inclusive rather than source-decision clock	`coop_box_pushing`, `matrix_rps`; O4/O5	Targeted clock/lifecycle codes were removed on development witnesses; the reserved evaluation classifies the recurring root as a contract mismatch.
Prebuilt configuration lost at reset	`go(board_size=5)`; O6	Parameter retention removes the mismatch; closely related to prior Shimmy configuration fixes [15, 16].
Mean-field node projected as ordinary AEC completion	`mfg_crowd_modelling`; O7	Explicit unsupported-game rejection replaces silent projection; classified as a capability treatment, not implemented mean-field support.

one `matrix_rps` joint transition to two calls: a reward-zero buffer and a committing call. A synthetic two-agent Parallel environment maps two source transitions to six AEC calls, including two inert cleanup calls, and preserves the delivered return $(2, -2)$. These controls establish only the exercised mappings, not a registry-wide false-positive rate.

Six targeted treatment cases in `artifacts/discovery_repairs.json` remove the intended codes under their fixed development witnesses. A combined treatment records zero violations on the four non-mean-field regression witnesses used to construct it. These are causal development checks, not a full-census repair evaluation. On a partial simultaneous cycle in `coop_box_pushing`, strict lockstep is inapplicable and the commit-sampling macro-boundary comparator passes after discarding the buffer phase, whereas both the whole-block macro-aggregate comparator and MARLRefine detect the resulting reward mismatch. Only MARLRefine attributes it to the exact nonzero destination-only call and the zero-reward buffering obligation. This witness therefore demonstrates a localization and diagnosis advantage over aggregate compression, not unique detection. The reserved census below establishes that the reward mechanism recurs beyond the development names.

5.2 Prospective Preservation Results

All counts in this section were emitted from the validated analysis artifact and its structured adjudication records into `results_macros.tex`; they were not transcribed from console output.

Across 105 distinct reserved game types and 840 scheduled game–policy traces, the analysis artifact contains 840 trace records. At the trace level, 808 completed semantically: 90 passed

Table 3: Prospective trace and game accounting. Trace outcomes are mutually exclusive. Game rows use the stated evidence-completeness precedence and are also mutually exclusive; overlapping "any trace" flags remain in the artifact.

Outcome	Count
Distinct prospective game types	105
Scheduled game–policy traces	840
Semantically completed traces (pass + fail)	808
Passing traces (no detected violation)	90
Terminal-complete traces	647
Passing bounded prefixes stopped at the source-decision limit	10
Failing traces	718
Semantic evidence: observed failure	718
Semantic evidence: no observed failure	90
Semantic evidence: no verdict	32
Execution completeness: semantic abort	151
Inapplicable traces	32
Unalignable traces	0
Infrastructure-failure traces	0
Games: all eight traces pass	2
Games: violation (without higher-precedence outcome)	99
Games: inapplicable (without unalignable/infrastructure)	4
Games: unalignable (without infrastructure)	0
Games: infrastructure outcome	0

and 718 failed. The remaining outcomes are 32 inapplicable, 0 unalignable, and 0 infrastructure failures. The analysis reports game-level coverage separately and never treats policies as independent game replications. Independently of that operational classifier, the semantic axis records 718 traces with observed failure evidence, 90 with no observed failure, and 32 with no semantic verdict; the execution axis records 151 semantic aborts.

RQ1 reports both finding incidence and descriptive obligation coverage. Table 4 lists each observed checker-family/code pair with its raw occurrence count, number of affected scheduled traces, and number of distinct affected games. A code absent from this table was not observed; its parent obligation's applicability and evaluation counts must be read from table 5, not inferred from absence alone.

5.3 Root, Comparison, Localization, and Repair Results

After contract adjudication and causal clustering, the complete study yields three prospective roots across three fault families and no discovery-only root. The macro-aggregate ablation misses one root and the stock destination API baseline misses all three. Isolated repair is attempted for two roots: one satisfies the full criterion and one fails it; repair is not applicable to the unsupported mean-field capability. The controls record no unexplained alarm on deterministic

Table 4: Prospective finding incidence emitted by the prespecified analysis. Occurrences are correlated symptoms and must not be read as independent bugs.

Checker family	Violation code	Occurrences	Traces	Games
boundary_lifecycle_preservation	`boundary_lifecycle_mismatch`	318	318	54
decision_clock_preservation	`premature_adapter_truncation`	127	127	21
decision_clock_preservation	`source_decision_clock_mismatch`	192	192	26
delivered_reward_conservation	`consumer_delivery_mismatch`	561	446	73
delivered_reward_conservation	`consumer_return_mismatch`	439	439	72
lifecycle_preservation	`agents_exhausted_before_source_terminal`	151	151	24
lifecycle_preservation	`pre_cleanup_lifecycle_mismatch`	318	318	54
lifecycle_preservation	`terminality_mismatch`	24	24	3
segment_reward_conservation	`segment_reward_mismatch`	19	7	1
state_kind_soundness	`mean_field_node_silently_terminated`	24	24	3
stutter_reward_neutrality	`nonzero_stutter_reward`	19	7	1
terminal_cleanup_reward_neutrality	`nonzero_terminal_cleanup_reward`	545	439	72

Table 5: Per-obligation trace coverage emitted from the frozen analysis. The four outcome columns are mutually exclusive within each obligation and sum to the scheduled trace count. Site checks are repeated boundary or microstep evaluations, not independent replications.

Obligation	Eval. pass	Eval. fail	Not applicable	Not evaluated	Site checks
O1	136	0	704	0	10308
O2	352	446	42	0	11992
O3	362	446	32	0	74425
O4	616	192	32	0	85770
O5	130	678	32	0	75462
O6	0	0	840	0	0
O7	784	24	32	0	90337
O8	808	0	32	0	84086

clean controls. The 718 failing traces and 99 affected games are therefore recurrence counts, not defect counts: causal clustering reduces them to three roots. Schema and cross-reference validation establishes artifact consistency, not the substantive truth of a contract interpretation or causal attribution. Root, consequence, comparison credit, and repair status are human causal adjudications. The frozen analyzer validates the adjudication schema, raw-batch identity, and internal consistency; it does not execute repairs or infer causality. Each label must cite a released runner artifact. A comparator receives same-root detection credit only when isolated-treatment evidence removes both the target root and its signal. Repair dispositions are derived from structured labels backed, when attempted, by before/after matrices; the analyzer validates the accounting but does not rerun or substantively adjudicate a repair.

5.4 Execution Cost

The median selected-final-record runner duration is 0.0143584 s, total selected-final-record runner duration is 134.133 s, and the median source-plus-destination ledger length (including zero-event infrastructure records) is 29 events. Runtime excludes archive verification, orchestration, and the superseded first attempt. The separately bound external-baseline panel took 64.5941 s: the stock API test passed on 96 games and failed on 9, while the released Shimmy suite status was `pass`. The mutation batch took 2.24754 s. Its peak memory is not measured (not instrumented); the

Table 6: Prospective path, structural, action, stratum, policy-saturation, and tolerance-sensitivity reporting. Counts describe recorded bounded paths; they are neither independent replications nor proof over unvisited behavior.

Dimension	Unit or comparison	Frozen result
Registry strata	scheduled traces	mean_field: 24; sequential___deterministic: 336; sequential___stochastic: 336; simultaneous___deterministic: 96; simultaneous___stochastic: 48
Source event origins	atomic events by native pre-event kind	chance: 15751; decision: 59358; simultaneous: 14420
Destination phases	calls retained by phase	buffer: 10308; commit: 73778; other stutter: 0; cleanup: 1684
Action coverage	unique selected/offered visited-state tuples	83759/2406731; marginal gains by policy: 12266, 10557, 9384, 10238, 11228, 9767, 9819, 10500
Tolerance sensitivity	sites within 0.1x/1x/10x threshold	reward 0.1x=73759, 1x=73759, 10x=73759 of 73778; observation 0.1x=81986, 1x=81986, 10x=81986 of 81986

primary-run peak-memory field is not measured (not instrumented). These separately identified costs are not folded into the primary 840-trace runner measurements.

5.5 Prospectively Sealed Mutation Validation

The sealed runner attempted 48 candidates and the fixed-order eligibility rule selected 24. MARLRefine produced 19 semantic kills and 4 crash-only detections; one selected mutant survived. The paired clean references raised 0 unexpected alarms among selected candidates and 0 across the attempted pool. The separately scored progress-history controls detected 2 of 2 and remain outside the mutant denominator. Exact 24-mutant cohort completion is yes, named progress-control validation is yes, and RQ6 reporting readiness is yes. The strong gate—RQ6 reporting readiness plus a complete cohort, 20/24 overall semantic kills, and 3/4 in every family—is no (semantic_kills_below_20_of_24); the explicit sensitivity-claim alias is no (semantic_kills_below_20_of_24). These gates are not interchangeable: a fully validated short-denominator or low-score result remains reportable without supporting the stronger claim. In this run, the cohort and both progress controls are complete, but 19/24 semantic kills falls one short of the aggregate threshold. Four crash-only detections do not convert that result into a semantic kill, so we make no strong detector-sensitivity claim. One candidate's O8 evidence-count inconsistency surfaced during the superseded attempt; the complete panel was rerun after the representation-only correction. Its result remains in the denominator but is not described as untouched or preregistered evidence.

Table 7: One row per distinct, causally adjudicated implementation root; do not duplicate rows by game, seed, trace, or symptom.

Root/family	Provenance, first witness, and effect	Ablation/baseline result, credit, attribution, and reach	Repair and adjudication
R3: mean-field capability	`mfg_crowd_modelling_2d`, smallest legal; a live mean-field node is projected as terminal	Macro boundary, aggregate, and endpoint signal; stock API passes and misses; strict/return inapplicable	Replay reproduced; explicit rejection; repair not applicable; unsupported capability
R1: reward replay	`laser_tag`, largest legal; non-advancing calls re-expose stale rewards, changing delivery and return	Macro boundary, aggregate, and endpoint signal; stock API passes and misses; strict/return inapplicable	Replay reproduced; isolated repair successful; contract-backed defect
R2: decision clock	`2048`, smallest legal; chance/microstep counting causes early truncation	All five in-line ablations miss or are inapplicable; stock API passes and misses	Replay reproduced; strict repair criterion failed; contract mismatch

Table 8: RQ3 localization and diagnostic specificity, emitted per confirmed root. Detection credit is separate from the resolution retained by each in-line ablation. Prefix length and ratio describe witness position, not localization accuracy.

Root	Destination phase/call	Original prefix	Obligation/code	Replay and ablation resolution
R3: mean-field	commit, call 0	1/2 calls	O7 / silently terminated mean-field node	replay reproduced; ablations retain no exact call attribution
R1: reward replay	buffer, call 2	3/1000 calls	O2 / nonzero stutter reward	replay reproduced; ablations retain no exact phase/call attribution
R2: decision clock	commit, call 0	1/250 calls	O4 / source decision-clock mismatch	replay reproduced; ablations retain no exact call attribution

6 Discussion

What the census found. The result is broad recurrence of a small number of mechanisms, not a catalog of hundreds of independent defects. Ninety-nine of 105 reserved game types had at least one failing trace, yet the 2,737 individual findings cluster into three roots. The dominant root re-exposes stale rewards during non-advancing buffer or cleanup calls. It changes the reward delivered through the public AEC interface and, on completed episodes, the consumer-visible return; an isolated treatment removed its target findings on all 446 affected reserved traces without introducing an unexpected finding. The clock root counts adapter-internal chance transitions and AEC microsteps against a horizon that the native source defines in player decisions. Its treatment removed the target on all 420 non-mean-field affected traces, but extending those paths revealed already known reward findings and the strict no-new-finding regression criterion was not met. The third root silently projects a live mean-field state as terminal. Explicit rejection is the appropriate capability disclosure, not evidence that mean-field semantics were implemented.

Complementarity rather than baseline defeat. The stock PettingZoo API test passed at each of the three canonical root witnesses, so all three are scored as missed under the predeclared reachability rule. This does not show that the stock test is defective: it checks destination-side protocol usability, while MARLRefine asks whether the destination still denotes the separately executed source task. The macro-aggregate ablation missed the decision-clock root but detected

Table 9: Sealed mutation results by prespecified contract family. Semantic kills exclude crash-only detections; clean alarms are measured on matched selected references.

Family	Selected	Semantic	Crash-only	Survived	Clean alarms
`reward_accounting`	4	4	0	0	0
`progress_and_interface`	4	3	1	0	0
`decision_clock`	4	3	0	1	0
`lifecycle`	4	3	1	0	0
`configuration_provenance`	4	3	1	0	0
`special_node_kind`	4	3	1	0	0

Table 10: Paired mutant-minus-reference signals for the five in-line ablations and the separately executed stock destination API test.

Comparator	Paired signals
`strict_lockstep`	0
`macro_boundary`	9
`macro_aggregate`	9
`endpoint`	5
`return_only`	4
`stock_pettingzoo_api_test`	4

signals associated with the reward and mean-field roots. Its loss is mainly diagnostic resolution: it cannot identify which destination-only call exposed the wrong value.

Sensitivity evidence. The mutation panel supports a qualified, not maximal, validation claim. Nineteen of 24 selected mutants produced new semantic signatures, four more were exposed only through execution failure, and one survived; the paired clean references raised no alarms. The complete cohort and two control injections make RQ6 reportable, but the semantic score missed the prespecified 20/24 bar by one. The panel therefore demonstrates substantial sensitivity to the declared fault model while preserving a visible negative result.

What alignment adds. Whole-block macro aggregation is a useful ablation: it preserves total reward across each completed source transition, while commit-only boundary sampling also checks state and lifecycle at the corresponding source boundary. Both compress away which value was exposed on which destination microstep. Non-bijective alignment retains those phases and labels their relationship to the nearest commit. This supports obligations such as zero instantaneous reward and no source-state advance during buffering without mistaking buffering itself for a semantic error. The many-to-one direction is equally important for adapter calls that internally resolve chance.

Testing, not proof. A passing trace shows only that the prespecified checker emitted no violation along that bounded action and chance path. Its obligation ledger records what was applicable and evaluated on that prefix; neither the trace verdict nor the ledger is a bisimulation proof or says anything about unvisited actions, configurations, or stochastic distributions. The benefit is executable counterevidence: one aligned failure can identify an exact recorded semantic discrepancy and produce a stable regression test. Consumer visibility is claimed only when the evidence shows that the discrepancy crosses a public destination-API boundary.

Table 11: Audit rows retained from the complete 48-candidate attempt ledger. The full candidate-level records remain in the bound mutation artifact.

First detecting obligation	Count
`boundary_lifecycle_preservation`	5
`configuration_provenance`	3
`decision_clock_preservation`	2
`interface_projection`	4
`segment_reward_conservation`	4
`state_projection`	1
`trace_execution`	4

First detecting phase	Count
`execution_control`	1
`online_destination_boundary`	8
`post_trace_alignment`	9
`setup_or_reset`	5

Replacement decision/reason	Count
`adapter_behavior_unchanged`	5
`family_quota_already_met`	19
`mutation_hook_not_reached`	2
`selected`	24

Portability. The immutable event records and pure comparator contain no live framework objects, and the span alignment retains every recorded ledger event. That architecture is adapter-parameterized; cross-framework reuse has not yet been demonstrated. Event extraction, player/action normalization, declared-representation choice, and progress instrumentation are adapter specific. The current integration uses private wrapped-state information for synchronization, so the supported claim is a white-box method evaluated on one adapter, not a black-box universal checker. Future reuse should report unchanged core code separately from adapter-specific instrumentation effort.

Practical use. Adapter maintainers can retain each original first-divergence source/destination prefix as a regression fixture alongside destination API tests. The two suites are complementary: destination tests protect protocol usability; source-aligned tests protect the mapping to the scientific task.

7 Threats to Validity

Construct and contract validity. The obligations combine general trace semantics with adapter-specific configuration, lifecycle, representation, and capability contracts. We pin primary sources and distinguish mismatches from adjudicated defects, but API documentation can remain ambiguous. In particular, failing fast on mean-field states is a capability treatment, not preservation of mean-field dynamics.

Oracle validity. The checker could model player/action mappings, reward channels, observations, or source boundaries incorrectly. The corrected `tic_tac_toe` observation oracle demonstrates that this risk is real. Separately loaded native clones, one bounded official-transform witness, adversarial harness mutants, separate reward channels, and source-tree/provenance

hashes mitigate but do not eliminate it. Moreover, alignment assumes that the adapter's private wrapped-history disclosure is complete, ordered, append-only, and nonperturbing. Native replay detects illegal deltas and inconsistent tags, but cannot prove that a plausible yet incomplete disclosure never occurred.

Design leakage and execution deviation. The obligations and implementation were informed by seven named games. We exclude every configuration of those names from prospective claims and label their outputs discovery-only. A validation recurrence of a known mechanism is prospective generalization, not a newly discovered causally distinct root. In addition, aggregate primary statuses from the first authorized attempt were visible before the mutation validator exposed its representational inconsistency. Although the correction did not alter semantic behavior and all stages were rerun under a new source identity, the final census was no longer blinded. We therefore report it as an integrity replication and do not use "preregistered" or "untouched held-out" language.

Coverage and stochasticity. Eight bounded traces cannot exhaust most games. Conditional replay of observed explicit chance outcomes checks a path, not the source distribution or seed identity. Sampled-stochastic and unsynchronizable cases remain visible in the denominator as inapplicable or unalignable outcomes.

External validity. The full registry census is broad within OpenSpiel, but it is still one source framework, one adapter, one destination protocol, and pinned releases. Results do not establish prevalence across the adapter ecosystem. Contract and version drift also limit conclusions; exact code, manifests, and primary sources therefore accompany the artifact.

Diagnosis and repair bias. The same researcher designed the checker, diagnosed failures, and evaluated repairs, using an AI-assisted implementation workflow. We preserve original traces, require contract citations, isolated treatments, reversion evidence, and full regression, and bind every manual root assignment to an exact finding. No upstream maintainer confirmation or second adjudicator was obtained, so the root labels remain author adjudications and no inter-rater reliability claim is made. The mandatory sealed mutation panel supplies detector-sensitivity evidence only under its stated contract-derived fault model. Its synthetic kills remain separate from real roots and do not estimate real-world prevalence.

8 Related Work

Input/output conformance supplies a general model-based testing foundation for checking an implementation against a labeled-transition specification [3]. Action refinement relates one abstract action to a concrete action sequence [2], while skipping refinement supports variable-rate matching with stuttering or skipping [4]. Semantic program alignment and runtime trace validators similarly align executions with programs or executable/formal models and can return counterexamples [17–23]. Formal adapter equivalence and synthesis are also established topics [24]. MARLRefine uses these ideas as foundations; it claims neither a new refinement relation nor a generic first instance of differential or trace-based testing.

Table 12: Positive capability comparison with the closest executable testing patterns. "Work-specific" indicates that capabilities vary across the cited family rather than being uniformly absent.

Approach	Reference/oracle and schedule relation	Retained checks	Diagnostic result
Gymnasium matcher; Karten et al. [12, 27]	Two executable environments under matched seeds/actions and a common step index	Reset/step outputs plus interaction and rollout properties	First failing common step or property
Official OpenSpiel/PettingZoo transformations [13, 14]	Framework-defined source/conversion relation, including joint-to-sequential schedules	Conversion-specific state, buffering, reward, and rollout behavior	Test failure on the exercised conversion
PettingZoo and Shimmy API tests [10, 11]	Destination execution without a separately loaded semantic source	AEC lifecycle, spaces, rewards, agent iteration, and adapter-specific smoke behavior	Assertion, warning, or exception at a destination call
Runtime refinement, trace validation, and RATE [18, 19, 23, 28]	Executable/formal model or reference trace with a work-specific correspondence	Work-specific state, transition, temporal, and coverage properties	Counterexample, failing test, aligned trace, or refinement evidence
MARLRefine	Separately loaded live source with replay-anchored variable-length blocks	O1–O8 at aligned boundaries and retained buffer, commit, stutter, and cleanup calls	Exact destination call and phase, obligation/code, original prefix, and gated replay

The AEC model and PettingZoo establish an equivalent POSG-to-AEC construction in which actions are queued across agents and reward is zero until environment resolution [1]. PettingZoo's canonical Parallel-to-AEC wrapper implements similar buffering [14]. OpenSpiel provides both the source game framework [25] and a simultaneous-to-turn-based transform [13]. Its v2.0.2 test source contains a broad native/transformed rollout-comparison routine, but that routine's call is disabled in the test entry point [26]. These supply semantic foundations and clean conversion witnesses for the present study.

The specific combination evaluated here is therefore a separately loaded native oracle, non-bijective live-API schedules, retained destination-only phases, contract-derived MARL obligations at boundaries and microsteps, and root-oriented localization/replay. Gymnasium's matcher and the translated-RL evaluation of Karten et al. provide the closest common-clock comparison patterns [12, 27]; the distinction is the schedule-changing relation and the evidence retained within a block. Gimitest instruments Gymnasium and PettingZoo to test trained policies [29], whereas the subject here is an adapter and the oracle is its separately executed source. Coverage-guided and cross-implementation RL differential testing address complementary test-generation and consistency questions [30, 31]. MARLlib and PufferLib demonstrate the wider importance of compatibility layers [32, 33]; cross-adapter reuse of MARLRefine remains future work.

Executable specifications and property-based testing are established for RL problems and algorithms [34]; RATE combines model refinement and conformance-test execution for ordinary software [28]. RL mutation testing predates this study [35, 36], and the later real-fault-derived μPRL taxonomy includes environment setup, termination flags, and state representation [37]. Those precedents motivate the prospectively sealed, contract-derived mutation benchmark used here. Its 48-candidate manifest, outcome-blind eligibility rule, paired composite references, and six-family selection target are a validation design rather than a novelty claim about mutation testing. The study reports semantic and crash-only detections separately and makes no general detector-sensitivity claim beyond this frozen mutation model.

9 Availability, Ethics, and Limitations

Data Availability Statement. The execution status is not preregistered (local Git freeze) at source revision `640b1d89174e44fa7a0440f1d4d296e316cc5b41`. A clean Git commit and explicit local authorization freeze the protocol, exact 113-name census and study manifests, mutation manifest, source and lock identities, analysis, and execution contract before any prospective semantic trace or sealed mutation outcome, but provide no public preregistration or independent timestamp. The result data bind the raw prospective batch at `b47aeb50ab15fc21418dfeb781e8224c77addaddddcb4b15cd40a2d148fc0ce0`, the external baseline at `1cde9d58e00ff60744d7496fc695733afe4424c2780123431d4428c7e4c1eae4`, and the mutation batch at `ab4fa4756ce56c510df2704c5aa272a61d041412d0ce2514575315130dd2a628`. The complete raw ledgers, paired mutation records, controls, replay artifacts, structured adjudication, treatment matrices, and generated tables are currently retained in a validated local result bundle. They have not yet been deposited or packaged for anonymous review; those submission logistics are deliberately separate from the completed analysis reported here. A repository or archival access statement will replace this sentence before journal submission. The current materials are not described as preregistered.

Ethics. This is software-artifact research and involves no recruited human participants or personal data. The AI-assisted workflow is disclosed in section 3. The supported scope remains the pinned OpenSpiel–Shimmy–PettingZoo stack and the exercised traces.

Acknowledgments. The author thanks the maintainers of OpenSpiel, Shimmy, and PettingZoo for the open-source systems studied here. AI-tool use is reported in section 3.

Funding. No external funding was received for this work.

Conflict of Interest Statement. The author declares no conflicts of interest.

10 Conclusion

Schedule-changing environment adapters require more than one-call lockstep, but unequal schedules do not require a new refinement theory. MARLRefine turns established action-refinement, stuttering/skipping, and POSG-to-AEC principles into an event-retaining alignment over recorded source/destination ledgers and checks pinned MARL obligations both at commits and during target-only phases. Discovery evidence showed why this question is distinct from destination-local API conformance; the completed 105-game, 840-trace census then found 718 failing traces across 99 games but only three causally adjudicated roots. A consumer-visible stale-reward root was successfully treated, a decision-clock mismatch failed the strict repair criterion, and a mean-field projection was classified as unsupported capability. The stock destination API test missed all three canonical witnesses. The sealed mutation panel produced 19/24 semantic

kills, four crash-only detections, one survivor, and no clean-reference alarms, falling one semantic kill short of the prespecified strong threshold. These results support source-aligned, phase-aware testing as a useful complement to destination API validation, while remaining bounded to one adapter, finite paths, one author adjudication, and a non-preregistered execution.

References

[1] J. K. Terry, Benjamin Black, Nathaniel Grammel, Mario Jayakumar, Ananth Hari, Ryan Sullivan, Luis Santos, Clemens Dieffendahl, Caroline Horsch, Rodrigo Perez-Vicente, Niall L. Williams, Yashas Lokesh, and Praveen Ravi. PettingZoo: Gym for multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 34, pages 15032–15043, 2021. URL `https://arxiv.org/abs/2009.14471`.

[2] Machiel van der Bijl, Arend Rensink, and Jan Tretmans. Action refinement in conformance testing. In *Testing of Communicating Systems*, volume 3502 of *Lecture Notes in Computer Science*, pages 81–96. Springer, 2005. doi: 10.1007/11430230_7. URL `https://doi.org/10.1007/11430230_7`.

[3] Jan Tretmans. Model based testing with labelled transition systems. In *Formal Methods and Testing*, volume 4949 of *Lecture Notes in Computer Science*, pages 1–38. Springer, 2008. doi: 10.1007/978-3-540-78917-8_1. URL `https://doi.org/10.1007/978-3-540-78917-8_1`.

[4] Mitesh Jain and Panagiotis Manolios. Skipping refinement. In *Computer Aided Verification*, volume 9206 of *Lecture Notes in Computer Science*, pages 103–119. Springer, 2015. doi: 10.1007/978-3-319-21690-4_7. URL `https://doi.org/10.1007/978-3-319-21690-4_7`.

[5] Farama Foundation. PettingZoo 1.27.0 AECEnv reward, lifecycle, and cleanup implementation. Tagged source code, 2026. URL `https://github.com/Farama-Foundation/PettingZoo/blob/1.27.0/pettingzoo/utils/env.py#L48-L53`. Reward channels at lines 48–53 and 161–196; dead-agent cleanup at lines 198–246; accessed 2026-08-31.

[6] Google DeepMind. OpenSpiel game and maximum-game-length interface. Tagged source code, 2026. URL `https://github.com/google-deepmind/open_spiel/blob/v2.0.2/open_spiel/spiel.h#L476-L500`. Immediate rewards and cumulative returns at lines 476–500; MaxGameLength contract at lines 1073–1078; accessed 2026-08-31.

[7] Josée Desharnais, François Laviolette, and Sami Zhioua. Testing probabilistic equivalence through reinforcement learning. *Information and Computation*, 227:21–57, 2013. doi: 10.1016/j.ic.2013.02.002. URL `https://doi.org/10.1016/j.ic.2013.02.002`.

[8] Farama Foundation. Shimmy 2.0.1 OpenSpiel compatibility adapter. Source code, 2026. URL `https://github.com/Farama-Foundation/Shimmy/blob/v2.0.1/shimmy/openspiel_compatibility.py#L33-L68`. Construction at lines 33–68, interface projection at lines 70–161 and 301–405, and reset at lines 190–218; accessed 2026-08-31.

[9] Farama Foundation. Shimmy 2.0.1 distribution files. Python Package Index release files, 2026. URL `https://pypi.org/project/Shimmy/2.0.1/#files`. Accessed 2026-08-31.

[10] Farama Foundation. Shimmy 2.0.1 OpenSpiel adapter tests. Source code, 2026. URL `https://github.com/Farama-Foundation/Shimmy/blob/v2.0.1/tests/test_openspiel.py`.

[11] Farama Foundation. PettingZoo 1.27.0 AEC api test. Tagged source code, 2026. URL https://github.com/Farama-Foundation/PettingZoo/blob/1.27.0/pettingzoo/test/api_test.py#L425-L550. Dynamic play checks at lines 425–550 and API-test entry point at lines 577–661; accessed 2026-08-31.

[12] Farama Foundation. Gymnasium environment matching utility. Source code, version 1.3.0, 2026. URL https://github.com/Farama-Foundation/Gymnasium/blob/v1.3.0/gymnasium/utils/env_match.py.

[13] Google DeepMind. OpenSpiel simultaneous-to-turn-based game transform. Tagged source code, 2026. URL https://github.com/google-deepmind/open_spiel/blob/v2.0.2/open_spiel/game_transforms/turn_based_simultaneous_game.cc#L107-L130. Action buffering and joint commit at lines 107–130; mid-rollout zero rewards at lines 167–170; accessed 2026-08-31.

[14] Farama Foundation. PettingZoo environment conversion implementations. Tagged source code, 2026. URL https://github.com/Farama-Foundation/PettingZoo/blob/1.27.0/pettingzoo/utils/conversions.py#L358-L397. Parallel-to-AEC action collection, reward clearing, commit, and delivery at lines 358–408; accessed 2026-08-31.

[15] Elliot Tower. Fix bugs in OpenSpiel wrapper (obs/action space, seeding, reset config). Shimmy pull request 96, 2023. URL https://github.com/Farama-Foundation/Shimmy/pull/96.

[16] Elliot Tower. OpenSpiel: Fix reset with no seed, make obs/act spaces match PettingZoo. Shimmy pull request 97, 2023. URL https://github.com/Farama-Foundation/Shimmy/pull/97.

[17] Berkeley Churchill, Oded Padon, Rahul Sharma, and Alex Aiken. Semantic program alignment for equivalence checking. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 1027–1040. ACM, 2019. doi: 10.1145/3314221.3314596. URL https://doi.org/10.1145/3314221.3314596.

[18] Tayfun Elmas, Serdar Tasiran, and Shaz Qadeer. VYRD: Verifying concurrent programs by runtime refinement-violation detection. In *Proceedings of the ACM SIGPLAN 2005 Conference on Programming Language Design and Implementation*, pages 27–37. ACM, 2005. doi: 10.1145/1064978.1065015. URL https://doi.org/10.1145/1064978.1065015.

[19] Horatiu Cirstea, Markus A. Kuppe, Benjamin Loillier, and Stephan Merz. Validating traces of distributed programs against TLA+ specifications. In *Software Engineering and Formal Methods*, volume 15280 of *Lecture Notes in Computer Science*, pages 126–143. Springer, 2025. doi: 10.1007/978-3-031-77382-2_8. URL https://doi.org/10.1007/978-3-031-77382-2_8. SEFM 2024; first online 26 November 2024.

[20] Ding Ding, Zhanghan Wang, Jinyang Li, and Aurojit Panda. Runtime protocol refinement checking for distributed protocol implementations. In *22nd USENIX Symposium on Networked Systems Design and Implementation*, pages 1305–1326. USENIX Association, 2025. URL https://www.usenix.org/conference/nsdi25/presentation/ding.

[21] Finn Hackett and Ivan Beschastnikh. Tracelinking implementations with their verified designs. *Proceedings of the ACM on Programming Languages*, 9(OOPSLA2):2171–2198, 2025. doi: 10.1145/3763128. URL https://doi.org/10.1145/3763128. Article 350.

[22] Finn Hackett, Evan Wrench, Peter Macko, A. Jesse Jiryu Davis, Yuanhao Wei, and Ivan Beschastnikh. Trace validation of unmodified concurrent systems with OmniLink. *arXiv preprint*

arXiv:2601.11836, 2026. doi: 10.48550/arXiv.2601.11836. URL https://arxiv.org/abs/2601.11836.

[23] Noah van der Vleuten, Anthony Flores, Shray Mathur, Max Rakitin, Thomas Hopkins, Kevin G. Yager, and Esther H. R. Tsai. EnvTrace: Simulation-based semantic evaluation of LLM code via execution trace alignment—demonstrated at synchrotron beamlines. *arXiv preprint arXiv:2511.09964*, 2025. doi: 10.48550/arXiv.2511.09964. URL https://arxiv.org/abs/2511.09964.

[24] Gal Amram, Suguman Bansal, Dror Fried, Lucas Martinelli Tabajara, Moshe Y. Vardi, and Gera Weiss. Adapting behaviors via reactive synthesis. In *Computer Aided Verification*, volume 12759 of *Lecture Notes in Computer Science*, pages 870–893. Springer, 2021. doi: 10.1007/978-3-030-81685-8_41. URL https://doi.org/10.1007/978-3-030-81685-8_41.

[25] Marc Lanctot, Edward Lockhart, Jean-Baptiste Lespiau, Vinicius Zambaldi, Satyaki Upadhyay, Julien Pérolat, Sriram Srinivasan, Finbarr Timbers, Karl Tuyls, Shayegan Omidshafiei, Daniel Hennes, Dustin Morrill, Paul Muller, Timo Ewalds, Ryan Faulkner, János Kramár, Bart De Vylder, Brennan Saeta, James Bradbury, David Ding, Sebastian Borgeaud, Matthew Lai, Julian Schrittwieser, Thomas Anthony, Edward Hughes, Ivo Danihelka, and Jonah Ryan-Davis. OpenSpiel: A framework for reinforcement learning in games. *CoRR*, abs/1908.09453, 2019. URL https://arxiv.org/abs/1908.09453.

[26] Google DeepMind. OpenSpiel simultaneous-to-turn-based transform tests. Tagged source code, 2026. URL https://github.com/google-deepmind/open_spiel/blob/v2.0.2/open_spiel/game_transforms/turn_based_simultaneous_game_test.cc. The registry-wide comparison routine is present but disabled in the test entry point; accessed 2026-08-31.

[27] Seth Karten, Rahul Dev Appapogu, and Chi Jin. Automatic generation of high-performance RL environments. *arXiv preprint arXiv:2603.12145*, 2026. doi: 10.48550/arXiv.2603.12145. URL https://arxiv.org/abs/2603.12145.

[28] Andrea Bombarda, Silvia Bonfanti, Angelo Gargantini, Yu Lei, and Feng Duan. RATE: A model-based testing approach that combines model refinement and test execution. *Software Testing, Verification and Reliability*, 33(2):e1835, 2023. doi: 10.1002/stvr.1835. URL https://doi.org/10.1002/stvr.1835.

[29] Dennis Gross, Quentin Mazouni, Helge Spieker, and Arnaud Gotlieb. Gimitest: A comprehensive tool for testing reinforcement learning policies. *arXiv preprint arXiv:2607.07029*, 2026. doi: 10.48550/arXiv.2607.07029. URL https://arxiv.org/abs/2607.07029.

[30] Martin Tappler, Edi Muškardin, Bernhard K. Aichernig, and Bettina Könighofer. Learning environment models with continuous stochastic dynamics—with an application to deep RL testing. In *2024 IEEE Conference on Software Testing, Verification and Validation*, pages 197–208. IEEE, 2024. doi: 10.1109/ICST60714.2024.00026. URL https://doi.org/10.1109/ICST60714.2024.00026.

[31] Rajdeep Singh Hundal, Yan Xiao, Xiaochun Cao, Jin Song Dong, and Manuel Rigger. On the mistaken assumption of interchangeable deep reinforcement learning implementations. In *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)*, pages 2225–2237. IEEE, 2025. doi: 10.1109/ICSE55347.2025.00222. URL https://doi.org/10.1109/ICSE55347.2025.00222.

[32] Siyi Hu, Yifan Zhong, Minquan Gao, Weixun Wang, Hao Dong, Xiaodan Liang, Zhihui Li, Xiaojun Chang, and Yaodong Yang. MARLlib: A scalable and efficient multi-agent reinforcement learning

library. *Journal of Machine Learning Research*, 24(315):1–23, 2023. URL https://jmlr.org/papers/v24/23-0378.html.

[33] Joseph Suarez. PufferLib: Making reinforcement learning libraries and environments play nice. *arXiv preprint arXiv:2406.12905*, 2024. doi: 10.48550/arXiv.2406.12905. URL https://arxiv.org/abs/2406.12905.

[34] Mahsa Varshosaz, Mohsen Ghaffari, Einar Broch Johnsen, and Andrzej Wąsowski. Formal specification and testing for reinforcement learning. *Proceedings of the ACM on Programming Languages*, 7(ICFP):125–158, 2023. doi: 10.1145/3607835. URL https://doi.org/10.1145/3607835. Article 193.

[35] Yuteng Lu, Weidi Sun, and Meng Sun. Towards mutation testing of reinforcement learning systems. *Journal of Systems Architecture*, 131:102701, 2022. doi: 10.1016/j.sysarc.2022.102701. URL https://doi.org/10.1016/j.sysarc.2022.102701.

[36] Florian Tambon, Vahid Majdinasab, Amin Nikanjam, Foutse Khomh, and Giuliano Antoniol. Mutation testing of deep reinforcement learning based on real faults. In *2023 IEEE Conference on Software Testing, Verification and Validation (ICST)*, pages 188–198. IEEE, 2023. doi: 10.1109/ICST57152.2023.00026. URL https://doi.org/10.1109/ICST57152.2023.00026.

[37] Deepak-George Thomas, Matteo Biagiola, Nargiz Humbatova, Mohammad Wardat, Gunel Jahangirova, Hridesh Rajan, and Paolo Tonella. μPRL: A mutation testing pipeline for deep reinforcement learning based on real faults. In *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)*, pages 2238–2250. IEEE, 2025. doi: 10.1109/ICSE55347.2025.00036. URL https://doi.org/10.1109/ICSE55347.2025.00036.