

Layered Online Service and Replay Control for Verified AI R and D Acceleration

K. Takahashi

ORCID: [0009-0004-4273-3365](https://orcid.org/0009-0004-4273-3365)

April 28. 2026

Abstract

AI-assisted research and development can produce code, experiments, proofs, evaluators, tools, datasets, monitors, and reusable procedures. These outputs accelerate a development loop only when verified progress grows faster than hidden work, evaluator failure, benchmark contamination, unsafe artifacts, unfinished work, service overload, maintenance burden, and misleading claims. This paper develops *Layered Online Service and Certified Replay Control* (LOSCR), a model-independent theory for controlling scoped AI R&D acceleration under live operational constraints.

LOSCR separates cheap observation from strong claims. Layer 0 records an always-on event envelope and activates heavier sidecars only when required by contracts, adapters, audits, incidents, service claims, or reusable-library claims. Layer 1 represents validation, audit, replay, maintenance, and registry work as service obligations with calibrated capacity envelopes, reservation rules, queue-age limits, and load-contract state machines. Layer 2 admits reusable artifacts only through replay records, trusted-base registries, typed interfaces, contracts, protected witnesses, dependency graphs, maintenance envelopes, and measured promotion evidence.

The core mechanism is a deterministic claim checker. Canonical claim profiles, failure-code transitions, append-only ledgers, reference reducers, implementation adapters, evaluator audit profiles, baseline and frontier governance, service-obligation accounting, and falsification-to-checker maps determine the strongest claim level currently supported. The theory does not assume that larger controllers, stronger models, or more automation by themselves imply progress. It gives a lightweight operational path from ordinary AI-assisted work to controlled, service-aware, audited, production, frontier, and reinvestment claims, while automatically downgrading or quarantining claims when evidence, capacity, replay, evaluator health, or dependency integrity fails.

Keywords. AI R&D acceleration; online control; lightweight telemetry; service queues; certified replay; reusable libraries; validation throughput; evaluator audits; work-in-process aging; sequential testing.

1 Purpose

LOSCR is a theory of operational acceleration, not a theory of intelligence. It asks when a changing AI-assisted production loop increases verified useful progress per scarce resource under realistic constraints. A production loop may include humans, language models, tool-using agents, proof assistants, evaluators, CI systems, experiment runners, retrieval systems, replay stores, registry checkers, and governance checks. No component is treated as intrinsically special; each is represented by the policy it executes, the capacity it consumes, and the evidence it leaves.

The theory is designed for environments where models, prompts, scaffolds, tools, data, benchmarks, tasks, teams, and artifact libraries drift during development. Therefore, the primary object is an online

sequence of events and service queues. The practical aim is to decide, during development, whether to continue, throttle, route, audit, bridge, reserve capacity, replay, refresh, quarantine, roll back, retire, or escalate a claim.

2 Position Relative to Existing Work

Scaling laws relate training compute, data, and model size to loss or benchmark performance [1, 2]. LOSCR addresses a different object: whether an AI-assisted R&D production policy improves verified finite-horizon progress under resource, evaluation, safety, replay, validation, audit, and maintenance constraints.

Generator-evaluator systems such as FunSearch and AlphaEvolve show that model-guided search can generate useful mathematical, algorithmic, or infrastructure objects [3, 4]. AI Scientist-style systems, automated research pipelines, and self-modifying agent archives automate research stages or agent improvement [5, 6, 7]. LOSCR asks whether such outputs become durable production capital after independent validation, assimilation, replay preservation, retrieval, context, security, interference, and retirement costs are charged.

Benchmarks such as MLE-bench, RE-Bench, PaperBench, MLR-Bench, AIRS-Bench, SWE-CI, and long-task evaluations measure aspects of AI R&D competence, research engineering, replication, open-ended ML work, codebase maintenance, and long-horizon autonomy [8, 9, 10, 11, 12, 13, 14]. They are useful task sources and calibration environments. They do not by themselves provide a live production-control theory.

Recent empirical results motivate the conservative design. SWE-Skills-Bench reports that many injected software-engineering skills produce no pass-rate gain, can add large token overhead, and can degrade performance when guidance mismatches the local context [15]. OpenAI’s analysis of SWE-bench Verified reports flawed tests and contamination signals, and argues that the benchmark no longer measures frontier coding capability [16]. GDPval emphasizes expert-authored tasks and blind expert grading while reporting that automated graders are not yet a substitute for expert graders [17]. These results support treating skills, benchmarks, evaluators, and artifacts as fallible production assets rather than automatic progress.

The statistical and operational background includes randomized experiments, potential outcomes, inverse-propensity estimation, doubly robust evaluation, false-discovery control, time-uniform confidence sequences, online controlled experiments, queueing theory, bottleneck management, conservative bandits, technical-debt analysis, and Goodhart/Campbell measurement warnings [18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31].

Prior object	Imported constraint	Added object in LOSCR
Scaling laws	capability may improve with resources, but loss reduction is not production acceleration	service-aware verified progress per scarce resource
Generator-evaluator systems	search can create useful candidates, but evaluators and reuse can fail	evaluator audits, replay, admission, promotion, and retirement
AI R&D benchmarks	task suites calibrate capability under controlled conditions	live task-stream control, baselines, bridges, and frontier sampling frames
Long-horizon agent studies	controller persistence alone is insufficient under drift and external bottlenecks	service queues, age envelopes, reservations, and incident-scope invalidation
Causal inference and sequential testing	strong estimates require logged assignment, positivity, bounded outcomes, and uncertainty rules	claim-level caps, estimator profiles, and delayed-label ledgers
Queueing and bottleneck theory	throughput is limited by non-controller service capacity	vector service-load contracts and capacity ceilings for certification
Goodhart/Campbell warnings	measured proxies become unsafe when optimized without audit	evaluator state machines, sentinel rotation, and hard constrained ledgers

3 Layered Architecture

Definition 1 (Layer 0: Edge telemetry). *Layer 0 records cheap factual events and queue ages. It does not record value, causality, utility, evaluator trust, library capital, or scalar prices.*

Definition 2 (Layer 1: Online service control). *Layer 1 tracks validation, audit, replay, maintenance, and registry services as queues with capacity envelopes and per-action service-load contracts. It controls admission, throttling, routing, and reservation.*

Definition 3 (Layer 2: Certified reusable library). *Layer 2 applies only to outputs claimed as reusable certified capital. It requires replay records, interface signatures, contracts or acceptance conditions, checker identity, replay tier, protected witness, dependencies, and maintenance envelope.*

Protocol 1 (Layer activation). *Use Layer 0 for all ordinary AI-assisted work. Activate Layer 1 at the first observable trigger in this default list: a service queue has positive age; a claim is production-level or stronger; an output is reused in at least two downstream items; an action is external-facing; an evaluator, safety, replay, maintenance, registry, permission, or security dependency is present; an action enters a baseline, frontier, or ledger claim; or material Layer 0 missingness exceeds its declared tolerance. A scope may set stricter triggers before observation, but a production or stronger claim may not weaken these defaults. Activate Layer 2 only when a result is reused as certified production capital or supports certified-library or reinvestment claims.*

This separation is essential. A low-burden event log should never imply a high-burden acceleration claim. Conversely, a strong certified-library claim should not make ordinary daily telemetry prohibitively expensive.

4 Core Objects

Definition 4 (Execution substrate fingerprint). *An execution substrate fingerprint records the conditions under which an actor executes a run:*

$$S = (a, m, b, \theta, h, q, t, p, o),$$

where a is actor class and actor-process identifier, m is model or non-model executor identifier, b is model, provider, tool, or team build, θ is sampling, scheduling, or execution configuration, h is harness, scaffold, workflow, or procedure commit, q is prompt, instruction, rubric, or operating-procedure hash, t is tool and environment manifest, p is permission tier, and o is organization or service context. Human-only, model-only, and hybrid executions use the same fields; unavailable model-specific fields are recorded as not applicable rather than being omitted.

Definition 5 (Production policy). *A production policy π maps an item, observed state, substrate, budget, service state, and library state to an action procedure. It may generate work, verify work, repair an evaluator, retrieve a library entry, allocate review, modify a tool, reserve validation, refresh replay records, retire artifacts, or choose another policy.*

Definition 6 (Online event stream). *Events e_t arrive in time order and generate a filtration $\mathcal{F}_t = \sigma(e_1, \dots, e_t)$. A valid online decision at time t is measurable with respect to \mathcal{F}_t and cannot depend on later certification, later audits, future task difficulty, or future service capacity.*

Model independence means that claims are made about policies under recorded substrates and service states. If the substrate or service system changes without a bridge design, the claim is about the combined policy-substrate-service change.

Definition 7 (Claim levels). *Claim strength is ordered from weakest to strongest as observable, controlled, service-controlled, audited, operational production, causal production, transfer, frontier, and reinvestment. A claim may use evidence from stronger layers, but it cannot inherit a stronger level unless all required contracts, ledgers, gates, service reservations, baselines, and uncertainty rules for that level are satisfied.*

Definition 8 (Claim-level profile). *Each claim level has a machine-readable profile:*

```
claim_level_profile:
  claim_level:
  required_event_fields:
  required_contract_fields:
  required_ledgers:
  allowed_estimators:
  telemetry_coverage_floor:
  evaluator_state_floor:
  service_contract_states:
  dependency_unknown_budget:
  baseline_requirements:
  frontier_requirements:
  library_requirements:
  hard_constraints:
  downgrade_target:
```

The default profiles are the following. A stronger profile inherits all requirements of weaker profiles unless a field is explicitly marked not applicable before observation.

Level	Required evidence	Automatic downgrade trigger
Observable	Layer 0 envelope integrity, append-only correction links, state reconstruction, and telemetry coverage report	missing identifier, broken hash chain, or unreconstructable item trajectory
Controlled	observable evidence, gate state, WIP conservation, rollback rule, bounded intervention class, and resource-vector accounting	unresolved hard constraint, absent rollback rule, or unexplained WIP growth material to the claim
Service-controlled	controlled evidence, service channels, service-unit ontology, active or recalibrated load contracts, reservation records, and age envelopes	uncalibrated material channel, failed reservation, exceeded age envelope, or quarantined service contract
Audited	service-controlled evidence, monitored or audited evaluator state, logged audit sampling, delayed-label rules, and audit-age compliance	evaluator below floor, unknown assignment probability for audited labels, or stale audit
Operational production	audited evidence, complete sealed claim contract, shadow or bridged baseline, service envelopes, dependency graph, hard-constraint ledger, and no unresolved incident in scope	post-hoc contract field, missing bridge, material baseline debt, or unresolved incident reachability
Causal production	operational production evidence plus logged assignment, interference handling, exposure mapping, outcome cap, missingness rule, and admissible causal design	missing causal design, unknown assignment probability, or unsupported interference handling
Transfer	operational or causal production evidence plus bridged transfer target, target-stratum coverage, and substrate-service bridge	missing transfer bridge, unsupported target stratum, or changed target after outcomes
Frontier	operational or causal production or transfer evidence plus sealed frontier source, quota, weights, leakage screen, deduplication, and minimum task mass	self-selected frontier item without admission, quota change, leakage, or insufficient task mass
Reinvestment	promoted-library evidence, signed lineage, causal or cohort-bounded attribution, maintenance drag, negative lineage audit, and positive lower-bound net return	descriptive lineage only, unattributed co-use, negative net return, or stale maintenance envelope

Definition 9 (Canonical claim profiles). The following compact canonical profiles are normative for the reference checker. A local profile may only strengthen them unless a lower maximum supported claim level is declared before observation.

```
canonical_claim_profiles:
  observable:
    required_event_fields:
      [event_id, event_type, timestamp, item_id, parent_event_id,
       station_id, policy_id, action_type, substrate_fingerprint,
       status_raw, resource_raw, queue_channel, queue_age_raw,
       dependency_flag, reuse_count, input_hash, output_hash,
```

```

    claim_scope_id, integrity_hash]
required_ledgers: [edge_log]
telemetry_coverage_floor: 0.95
max_on_identifier_missing: invalid
controlled:
    inherits: observable
    required_ledgers: [edge_log, gate_ledger, wip_ledger, resource_ledger]
    required_contract_fields:
        [scope, station_set, task_strata, freeze_rule, downgrade_rule,
         escalation_rule, hard_constraints]
    max_on_missing_rollback: observable
    max_on_unexplained_wip_growth: observable
service_controlled:
    inherits: controlled
    required_ledgers: [service_ledger, service_obligation_ledger]
    service_contract_states: [active, recalibrated]
    telemetry_coverage_floor: 0.95
    max_on_uncalibrated_material_channel: controlled
audited:
    inherits: service_controlled
    required_ledgers: [audit_ledger, delayed_label_ledger]
    evaluator_state_floor: monitored
    telemetry_coverage_floor: 0.98
    max_on_unknown_assignment_probability: controlled
production_operational:
    inherits: audited
    claim_form: operational
    required_ledgers:
        [baseline_registry, dependency_graph, hard_constraint_ledger,
         checker_result_ledger]
    evaluator_state_floor: audited
    telemetry_coverage_floor: 0.99
    baseline_requirements: [shadow_or_bridged_baseline, contamination_test]
    service_contract_states: [active, recalibrated]
    dependency_unknown_budget: 0.01
    max_on_material_baseline_debt: audited
    max_on_unresolved_incident: quarantined
production_causal:
    inherits: production_operational
    claim_form: causal
    required_design:
        [randomized_assignment_or_shadow_assignment_or_switchback_or
         stepped_wedge_or_cluster_rollout_or_valid_instrument_or_bridge]
    required_logs:
        [assignment_probability, interference_handling, exposure_mapping,
         outcome_cap, missingness_rule]
    max_on_missing_causal_design: production_operational
transfer:
    inherits: production_operational
    bridge_requirements: [target_strata, substrate_service, evaluator]

```

```

    dependency_unknown_budget: 0
    max_on_missing_transfer_bridge: production_operational
frontier:
    inherits: production_operational
    frontier_requirements:
        [sealed_source, quota, weights, blinding, deduplication,
         leakage_screen, minimum_task_mass, governance_contract]
    dependency_unknown_budget: 0
    max_on_frontier_self_selection: production_operational
reinvestment:
    inherits: production_causal
    library_requirements:
        [promoted_entries, signed_lineage, negative_lineage_audit,
         causal_or_cohort_bounded_attribution, finite_horizon_lower_bound]
    dependency_unknown_budget: 0
    max_on_descriptive_lineage_only: production_operational

```

The production level has two claim forms. Operational production means the sealed loop achieved certified progress under logged conditions. Causal production additionally claims that the intervention caused the improvement relative to a counterfactual policy. When a canonical profile and a claim contract conflict, the stricter requirement is used. If strictness cannot be ordered, the checker downgrades to the strongest level whose canonical profile is unambiguous.

Definition 10 (Claim contract). *Every non-descriptive claim has a machine-readable contract:*

```

claim_contract:
    claim_id:
    contract_epoch:
    claim_level:
    claim_form:
    scope:
    station_set:
    task_strata:
    active_calendar_horizon:
    minimum_task_mass:
    minimum_service_exposure:
    materiality_floor:
    ledger_schema_ids:
    baseline_ids:
    frontier_source_ids:
    allowed_evidence:
    layer_activation_triggers:
    telemetry_coverage_floor:
    service_envelopes:
    evaluator_state_floor:
    uncertainty_rule:
    bridge_requirements:
    freeze_rule:
    downgrade_rule:
    escalation_rule:
    owner:

```

integrity_hash:

Definition 11 (Contract validity checker). *Let $\text{Check}_t(K, X_t)$ be the deterministic checker applied at time t to claim contract K and online state X_t . It returns*

$$\{\text{valid}, \text{invalid}, \text{downgraded}, \text{quarantined}\} \times L,$$

where L is the strongest claim level and claim form currently supported. The checker verifies field presence, type, unit, epoch, integrity hash, claim-level profile, claim-form requirements, ledger compatibility, service capacity, evaluator state, dependency coverage, baseline and frontier eligibility, estimator admissibility, hard constraints, and freeze state. Invalid means the contract cannot support any non-descriptive claim. Downgraded means it supports only level L below the requested level. Quarantined means a hard stop, integrity failure, revoked trusted base, evaluator revocation, corrupted replay, or incident-scope rule blocks positive credit until resolved.

Definition 12 (Checker result record). *The checker emits an append-only result record:*

```
checker_result:
  check_id:
  claim_id:
  contract_epoch:
  checked_at:
  requested_level:
  supported_level:
  supported_claim_form:
  status:
  failure_codes:
  violated_fields:
  incident_nodes:
  required_actions:
  frozen_intervals:
  dependency_hash:
  state_hash:
  checker_version_hash:
  integrity_hash:
```

The default failure-code families are:

```
type, missing, epoch, integrity, profile, ledger, service,
evaluator, baseline, frontier, dependency, estimator,
hard_constraint, freeze, burden
```

A failure code changes a claim only through the profile table, contract rules, or incident scope rule; it is never interpreted by informal judgment.

Definition 13 (Failure-code transition matrix). *The reference checker maps failure codes to status and maximum supported level by the following default matrix. A contract may strengthen a row but cannot weaken it for production or stronger claims.*

<i>Failure family</i>	<i>Default status</i>	<i>Default maximum level</i>	<i>Required action</i>
<i>type or schema conflict</i>	<i>invalid</i>	<i>observable only if envelope remains reconstructable</i>	<i>repair schema or create bridge</i>
<i>missing identifier</i>	<i>invalid</i>	<i>none above descriptive</i>	<i>reconstruct from signed source or discard claim</i>
<i>missing non-identifier</i>	<i>downgraded</i>	<i>strongest profile not using the field</i>	<i>add sidecar, audit, or lower claim</i>
<i>epoch without bridge</i>	<i>downgraded</i>	<i>descriptive comparison</i>	<i>create valid bridge or separate epochs</i>
<i>integrity failure</i>	<i>quarantined</i>	<i>none until resolved</i>	<i>incident review and revalidation</i>
<i>profile violation</i>	<i>downgraded</i>	<i>next satisfied canonical profile</i>	<i>satisfy profile or narrow claim</i>
<i>ledger incompatibility</i>	<i>downgraded</i>	<i>controlled unless ledger is repaired</i>	<i>reconcile ledger or mark inapplicable before use</i>
<i>service failure</i>	<i>downgraded or quarantined</i>	<i>controlled if no capacity, operational production if nonmaterial</i>	<i>reserve capacity, throttle, or recalibrate</i>
<i>evaluator failure</i>	<i>quarantined for affected strata</i>	<i>controlled for unaffected strata</i>	<i>stop optimization and pass bridge audit</i>
<i>baseline or frontier failure</i>	<i>downgraded</i>	<i>audited or operational production depending on affected scope</i>	<i>repair source, bridge, or narrow target</i>
<i>dependency failure</i>	<i>quarantined if incident-reachable, otherwise downgraded</i>	<i>boundary-dependent</i>	<i>expand boundary or audit edge</i>
<i>estimator failure</i>	<i>downgraded</i>	<i>estimator profile failure level</i>	<i>use fallback estimator or descriptive report</i>
<i>hard constraint failure</i>	<i>quarantined</i>	<i>none for affected scope</i>	<i>rollback, incident closure, and revalidation</i>
<i>freeze violation</i>	<i>downgraded</i>	<i>pre-freeze supported level</i>	<i>exclude frozen periods or bridge restart</i>
<i>burden violation</i>	<i>downgraded</i>	<i>operational production unless excess is charged</i>	<i>charge instrumentation or reduce burden</i>

If multiple failures occur, the checker applies the most restrictive status under the transition priority order. Quarantine dominates downgrade, and invalid dominates both when required identity or schema cannot be reconstructed.

Protocol 2 (Reference checker order). The reference checker is deterministic under a fixed event prefix and fixed trusted-base registry:

```

check(K, X):
  validate_schema_and_types(K, X)
  verify_integrity_hashes(K, X)
  verify_contract_epoch_and_bridge(K, X)
  reconstruct_state_from_append_only_events(X)
  apply_incident_scope_rule(K, X)
  test_hard_constraints(K, X)
  compute_telemetry_coverage(K, X)

```

```

test_claim_level_profile(K.claim_level, K, X)
test_claim_form_requirements(K.claim_form, K, X)
test_service_reservations_and_age_envelopes(K, X)
test_evaluator_state_and_audit_age(K, X)
test_baseline_and_frontier_requirements(K, X)
test_dependency_coverage_and_boundary_certificates(K, X)
test_estimator_admissibility(K, X)
test_instrumentation_burden(K, X)
return strongest_supported_level_with_codes()

```

Every step is pure: it reads the event prefix, registries, contracts, and reducer outputs, and writes only a checker result. State-changing repairs require separate intervention events.

Protocol 3 (Contract epoch changes). *A material change to scope, horizon, frontier source, baseline, evaluator, ledger, service envelope, estimator, task strata, or substrate creates a new contract epoch. Claims cannot pool numerator, denominator, duration, or service compliance across epochs unless a bridge record covers all changed fields. If a bridge is missing, the earlier and later epochs are reported separately and the combined claim is downgraded to descriptive comparison.*

Definition 14 (Epoch bridge record). *A bridge record is valid only if it is sealed before pooled outcomes are used:*

```

epoch_bridge:
  bridge_id:
  source_epoch:
  target_epoch:
  changed_fields:
  bridge_design:
  assignment_or_sampling_rule:
  covered_strata:
  minimum_overlap:
  accepted_error_bound:
  diagnostics:
  max_supported_claim_level:
  owner:
  integrity_hash:

```

If diagnostics fail, overlap is below the sealed minimum, or the accepted error bound exceeds the materiality floor of the target claim, the bridge cannot support pooling above descriptive comparison.

Protocol 4 (Contract precedence). *A claim contract is sealed before outcome data for the claim are used. If a field required by the claim level is absent, ambiguous, or self-inconsistent, Check_t automatically downgrades the claim to the strongest lower level whose profile is satisfied. Terms such as material, conservative, strong, acceptable, and out of scope have force only through contract fields, default thresholds in this paper, or independently auditable rules. Post-hoc narrowing is allowed only as a downgrade or quarantine, never as support for escalation.*

5 Layer 0: Edge Telemetry

Definition 15 (Edge event envelope and sidecar). *The always-on Layer 0 record is the envelope:*

```

edge_event_envelope:
  event_id:
  event_type:
  timestamp:
  item_id:
  parent_event_id:
  station_id:
  policy_id:
  action_type:
  substrate_fingerprint:
  status_raw:
  resource_raw:
    wall_time:
    compute_seconds:
    token_count:
    tool_call_count:
  queue_channel:
  queue_age_raw:
  dependency_flag:
  reuse_count:
  input_hash:
  output_hash:
  claim_scope_id:
  integrity_hash:

```

Additional fields are sidecar fields activated by claim contracts, audits, or adapters:

```

edge_event_sidecar:
  event_source:
  schema_id:
  state_before_raw:
  state_after_raw:
  queue_id:
  entered_queue_at:
  left_queue_at:
  adapter_id:
  adapter_source_uri:
  actor_process_id:
  service_unit_id:
  deadline_class:
  baseline_assignment_id:
  evaluator_id:
  incident_id:
  field_status_map:
    <field_name>:
      status:
      estimator_id:
      error_bound:

```

Envelope fields are mandatory for observable and stronger claims. Sidecar fields become mandatory only when required by a claim-level profile, adapter contract, service contract, evaluator audit, baseline

assignment, incident rule, or reusable-library claim.

Definition 16 (Edge sufficiency). *Layer 0 is sufficient when envelope records can reconstruct which policy acted on which item, at which station and queue channel, under which substrate, with which action type, with what raw status, basic resource use, queue age, dependency flag, reuse count, and intact input and output hashes. Event identifiers, timestamps, parent links, substrate fingerprints, input hashes, output hashes, claim scope, and integrity hashes are identifier fields and cannot be estimated for any claim above observable level. Stronger claims may require sidecar fields, but absence of a sidecar field downgrades only claims whose profile uses that field.*

Layer 0 does not certify improvement. It makes ordinary AI-assisted work observable with low implementation burden. It deliberately excludes value judgments, grader scores, causal estimates, service reservations, library admission, and scalar prices.

Definition 17 (Field status and telemetry coverage). *Every non-identifier field used by a claim has status **observed**, **missing**, **not_applicable**, or **estimated**. Estimated fields record estimator identity, source fields, estimation time, and a conservative error bound. For claim scope c and stratum s , telemetry coverage is the fraction of contract-required fields whose status is observed or justified as not applicable. The default coverage floor is 0.95 for service-controlled claims, 0.98 for audited claims, and 0.99 for production or stronger claims; a contract may choose stricter floors. If a required field is missing beyond the floor or an identifier field is missing, the claim is downgraded to observable or quarantined until a bridge, audit, or reconstruction rule covers the gap.*

Definition 18 (Default materiality and burden floors). *For production or stronger claims, the default materiality floor is any item, event class, service channel, dependency, or resource component that can change a primary ledger, hard constraint, service envelope, or claim-level decision by at least 1%, or can trigger a hard stop. For service-controlled and audited claims the default materiality floor is 2% unless the claim contract declares stricter floors. For controlled local claims, materiality is limited to hard constraints, WIP, rollback, and declared local resources. The default baseline-debt ceiling is 5% of claimed absolute gain. The default unknown-dependency budget is 1% of contract-required nodes for production and zero for transfer, frontier, and reinvestment. The default instrumentation burden ceiling for the daily profile is 3% wall time, 5% token count, 3% compute time, 1% storage growth in the claim scope, and one minute of manual annotation per event class, with a report of estimated human seconds per event. Burden above the ceiling must be charged as instrumentation cost and prevents dramatic claims unless the claim remains positive after the charge.*

Protocol 5 (Append-only telemetry invariant). *Layer 0 records are append-only. Corrections create new events that point to the corrected event and state the correction reason. A downstream claim uses the latest non-quarantined correction reachable at its seal time, and both the original and correction remain in the dependency graph.*

Definition 19 (Implementation adapter contract). *An adapter maps an existing operational log into envelope and sidecar fields:*

```
adapter_contract:
  adapter_id:
  source_system:
  source_event_type:
  field_map:
```

```

missing_field_policy:
timestamp_policy:
identity_policy:
hash_policy:
privacy_filter:
conformance_tests:
max_supported_claim_level:
integrity_hash:

```

The default adapters are: *Git commit* to *item*, *parent*, *actor-process*, *input-output hash*, and *status*; *CI run* to *station*, *action type*, *resources*, *queue age*, *gate state*, and *evaluator identity*; *LLM call* to *substrate*, *prompt hash*, *tool-call count*, *token count*, and *output hash*; *tool call* to *permission tier*, *resource use*, *status*, and *incident link*; *review comment* to *audit event*, *label*, and *delayed validation*; *experiment run* to *resource vector*, *result hash*, *evaluator*, and *reproduction state*; *proof-checker run* to *checker identity*, *accepted or rejected status*, *dependency hashes*, and *replay tier*. An adapter can support only the strongest claim level listed in its conformance tests.

Definition 20 (Default adapter field map). *The default adapter map is:*

Source	Envelope fields	Sidecar fields	Max level
<i>Git commit or patch</i>	<i>event id, time, item, parent, station, actor, input hash, output hash, status</i>	<i>branch, diff summary, review link, dependency flag</i>	<i>controlled</i>
<i>CI or test run</i>	<i>event id, item, station, policy, action, status, resources, queue channel, queue age, hashes</i>	<i>evaluator id, failure class, gate state, incident id</i>	<i>audited</i>
<i>LLM or agent call</i>	<i>event id, item, policy, action, substrate, resources, input hash, output hash, status</i>	<i>prompt hash, tool manifest, permission tier, trace hash</i>	<i>controlled</i>
<i>Tool call</i>	<i>event id, item, action, resources, status, hashes, dependency flag</i>	<i>permission tier, environment manifest, incident id</i>	<i>controlled</i>
<i>Review or audit comment</i>	<i>event id, item, station, action, status, queue age, hashes</i>	<i>auditor id, label, assignment probability, reveal time</i>	<i>audited</i>
<i>Experiment run</i>	<i>event id, item, policy, substrate, status, resources, hashes</i>	<i>metric record, evaluator id, reproduction state, leakage check</i>	<i>prod. bridge</i>
<i>Proof-checker run</i>	<i>event id, item, checker policy, status, resources, hashes</i>	<i>checker id, dependency hashes, witness hash, replay tier</i>	<i>production</i>

An adapter that cannot produce the listed envelope fields may support only descriptive reporting. A source-specific adapter reaches the listed maximum only after conformance tests show stable field extraction, timestamp ordering, identity resolution, hash preservation, and missing-field behavior on synthetic and retrospective logs.

6 Layer 1: Online Service Control

Definition 21 (Service channel). *Each service channel j declares*

$$\Xi_j = (u_j, \Delta_j, A_j, S_j, Q_j, \text{Age}_j, \underline{\kappa}_j, \bar{\rho}_j, \bar{a}_j, o_j),$$

where u is unit, Δ control window, A arrivals, S completions, Q queue mass, Age age statistic, $\underline{\kappa}$ conservative service-rate lower bound, $\bar{\rho}$ maximum utilization, \bar{a} maximum age, and o overload action.

Definition 22 (Service vector). *The non-controller service state is*

$$\Sigma_t = (\Xi_t^{\text{val}}, \Xi_t^{\text{audit}}, \Xi_t^{\text{replay}}, \Xi_t^{\text{maint}}, \Xi_t^{\text{reg}}),$$

for validation, audit, replay refresh, maintenance, and registry checking or writing.

Definition 23 (Service unit ontology). *Every service unit records service unit identifier, channel, unit name, provider class, quality tier, deadline class, batching rule, sharing rule, preemption rule, parallelism limit, acceptance criterion, and failure charge. Service units of different quality tiers or deadline classes are not interchangeable unless a bridge rule maps one tier to another with a conservative mapping factor. Shared service may be counted once and allocated by a registered sharing rule; otherwise double counting is forbidden.*

Definition 24 (Service ledger). *Each service channel maintains an append-only ledger:*

```
service_ledger_event:
  service_event_id:
  channel:
  service_unit_id:
  window_id:
  unit:
  deadline_class:
  attempted:
  completed_accepted:
  completed_rejected:
  failed:
  cancelled:
  expired:
  held:
  unavailable_exposure:
  eligible_exposure:
  incident_id:
  contract_id:
  integrity_hash:
```

Completed accepted work can support capacity lower bounds. Failed, cancelled, expired, and held work remain liabilities until charged, resolved, or explicitly excluded by a downgrade. Unavailable exposure caused by incidents is recorded and charged; it is not silently removed from the denominator of production capacity.

Definition 25 (Service obligation). *Service capacity is consumed by obligations, not by informal intentions. Each obligation records:*

```
service_obligation:
  obligation_id:
  created_by_event_id:
  claim_id:
  channel:
  service_unit_id:
  deadline_class:
  required_quantity:
  reserved_quantity:
  completed_quantity:
  cancelled_quantity:
  expired_quantity:
```

```

held_quantity:
created_at:
due_at:
queue_discipline:
priority_class:
reservation_id:
status:
integrity_hash:

```

The status is one of *created*, *reserved*, *in_service*, *completed*, *held*, *cancelled*, *expired*, or *quarantined*. The service ledger records attempts and completions; the obligation ledger records outstanding certification load. A production or stronger claim cannot count an output whose material obligations are held, expired, or unreserved.

Definition 26 (Service-rate lower bound). For channel j and control time t , let $\mathcal{W}_j(t)$ be the contract-sealed rolling calibration window, $S_j(\mathcal{W})$ completed accepted service units, and $E_j(\mathcal{W})$ auditable eligible service exposure from the service ledger. The conservative lower service-rate bound is

$$\underline{\kappa}_{j,t} = \max \left\{ 0, \text{LCB}_{\alpha_j} \left(\frac{S_j(\mathcal{W}_j(t))}{E_j(\mathcal{W}_j(t))} \right) \right\}.$$

The window records minimum sample size, censoring rule, downtime accounting rule, incident accounting rule, and minimum service exposure. The default minimum is at least 30 completed accepted units and three control windows unless the contract chooses a stricter requirement or uses a zero lower bound. If the minimum observations are not met, censoring is unresolved, or the exposure denominator is not auditable, the channel is uncalibrated. A production or stronger claim that needs that channel may proceed only with a zero lower bound or a registered conservative default that still satisfies reservation.

Definition 27 (Service capacity envelope). For channel j , a capacity envelope contains three lower bounds:

$$K_{j,t}^{\text{env}} = (K_{j,t}^{\text{steady}}, K_{j,t}^{\text{burst}}, K_{j,t}^{\text{age}}).$$

K^{steady} is the lower bound on sustained accepted service across the sealed rolling window. K^{burst} is the lower bound available over the next control window after existing reservations. K^{age} is the maximum service mass that can be completed before the queue-age envelope is violated. A production or stronger claim must satisfy all three bounds for every material channel. If nonstationarity or serial dependence is detected, the checker uses a time-uniform, block, or worst-window lower bound rather than an independent-sample bound.

Protocol 6 (Operational capacity computation). For each channel and deadline class, the checker computes:

1. K^{steady} from the calibrated lower service-rate bound multiplied by eligible exposure in the sealed horizon, minus accepted reservations in the obligation ledger;
2. K^{burst} from the lower bound on accepted completions in the next control window, minus reserved obligations and unavailable exposure;
3. K^{age} by simulating service obligations under the declared queue discipline with conservative arrivals, held work, due times, and preemption rules, then taking the largest additional reservation that keeps every material queue age below \bar{a} .

If the declared queue discipline is not earliest-deadline-first, the simulation uses the declared discipline. If no discipline is declared, $K^{age} = 0$ for production or stronger claims.

Definition 28 (Service-load contract). Each action type a declares an \mathcal{F}_t -measurable expected service-load vector and worst-case service-load vector:

$$\ell_t(a) = (\ell_t^{val}, \ell_t^{audit}, \ell_t^{replay}, \ell_t^{maint}, \ell_t^{reg}), \quad \bar{\ell}_t(a) \geq \ell_t(a).$$

The contract records source, calibration window, state, last update time, observed violation count, calibration error, default upper bound, and violation penalty. It is estimated from historical events, static rules, or conservative defaults. Missing estimates force exploration status rather than production credit.

Definition 29 (Service-load contract schema). The machine-readable load contract is:

```
service_load_contract:
  action_type:
  substrate_scope:
  service_unit_id:
  channel:
  deadline_class:
  expected_load:
  upper_load:
  calibration_window:
  minimum_observations:
  contract_state:
  last_update_time:
  violation_count:
  calibration_error:
  default_upper_bound:
  reservation_rule:
  max_supported_claim_level:
  integrity_hash:
```

The same action type may have multiple contracts when substrate, station, deadline class, or service quality changes. Pooling across contracts requires an epoch bridge or a conservative maximum upper load.

Definition 30 (Service-contract calibration). For action type a and service channel j , let $O_t^j(a)$ be observed service load after completion and $\bar{\ell}_t^j(a)$ the reserved upper bound. A contract violation occurs when $O_t^j(a) > \bar{\ell}_t^j(a)$ after applying the declared measurement tolerance. Repeated violations quarantine the contract, replace it by the conservative default, and downgrade affected production claims until a new calibration window is completed.

Definition 31 (Service-contract state machine). Each action-channel contract is in exactly one state:

$$\{\text{draft}, \text{active}, \text{suspect}, \text{quarantined}, \text{recalibrated}\}.$$

Draft contracts may support exploration only, except when a conservative default upper bound is used and fully reserved. A draft contract becomes active after the declared minimum observations, service-rate calibration, and violation bound are satisfied. An active contract becomes suspect after a material violation, material missingness, substrate drift outside its bridge, or service-rate lower-bound failure. A suspect

contract becomes quarantined after repeated violations or one critical violation. A quarantined contract can become recalibrated only after a new calibration window, incident review, and bridge rule. Recalibrated contracts become active after a shadow window without material violation. The transition log is part of the claim dependency graph.

Definition 32 (Age-bucket service queue). *For service channel j and age bucket k ,*

$$Q_{j,k,t+1} = \text{age}(Q_{j,k,t}) + A_{j,k,t} - S_{j,k,t} - X_{j,k,t} + H_{j,k,t},$$

where A is new required service, S is completed service, X is cancelled or expired service, and H is held or returned work caused by incidents, failed checks, bridge requirements, or missing evidence.

Protocol 7 (Service reservation). *An action that creates certification load must reserve service capacity using a non-quarantined $\bar{\ell}_t(a)$ before the action is credited. If reservation fails, the action may run only as exploration or local operation; its outputs cannot increase production, transfer, certified-library, or reinvestment claims.*

Theorem 1 (Vector capacity ceiling). *Fix a horizon H and candidate set U . Suppose certifying item $x \in U$ as production or reusable library capability requires conservative service-load upper vector $\bar{\ell}_x \in \mathbb{R}_+^d$, and available accepted service capacity has conservative lower vector $\underline{K} \in \mathbb{R}_+^d$. Any policy, regardless of controller scale, can certify only a subset $A \subseteq U$ satisfying*

$$\sum_{x \in A} \bar{\ell}_x \leq \underline{K}$$

componentwise, except for claims explicitly downgraded to non-reusable or non-production status. A claim requiring certified mass beyond every feasible subset is impossible without increasing service capacity, reducing service load, narrowing the target set, or weakening the claim.

Proof. Every certified item consumes the service units required by its claim. Distinct service obligations cannot be charged to the same service unit unless a registered sharing rule permits it. The conservative upper vectors dominate required load, and the conservative lower vector is the accepted capacity available to the claim. Summing over certified items gives the componentwise bound. Controller scale may change which items are attempted, but it cannot process more accepted service than the service system supplies. \square

Theorem 2 (Window-wise capacity ceiling). *For control windows $h = 1, \dots, H$, let A_h be the set of items whose certification obligation is active in window h , let $\bar{\ell}_{x,h}$ be the conservative service-load upper vector charged in that window, let B_h be carried service backlog and held work, and let \underline{K}_h be the conservative lower service capacity. A production or stronger claim must satisfy*

$$\sum_{x \in A_h} \bar{\ell}_{x,h} + B_h \leq \underline{K}_h$$

componentwise in every material window after declared cancellations and expirations. Priority and preemption may choose which obligations receive service first, but unserved obligations remain as held work, backlog, expiration, or quarantine; they cannot disappear from the capacity account.

For strong claims, \underline{K}_h is the componentwise minimum of steady, burst, and age-envelope capacity after unit mapping and existing reservations.

Proof. Apply the vector capacity ceiling to each window, with incoming backlog treated as already admitted service obligation. Preemption changes timing and priority, not total unserved obligation. Therefore any window that violates the bound must either defer, cancel, expire, or quarantine some obligation. \square

Theorem 3 (Service overload implies throttling). *If for a declared horizon the conservative lower bound on completed service in channel j is less than the conservative upper bound on arrivals plus held work, then either age-bucket queue mass or maximum age must increase unless upstream admission is throttled, service capacity is increased, or service-load contracts are reduced.*

Proof. Sum the age-bucket service recursion over the horizon. Excess arrivals and held work not completed, cancelled, or expired remain in some bucket and then age into older buckets. Therefore mass or maximum age rises unless admission or capacity changes. \square

7 Layer 2: Certified Reusable Libraries

Definition 33 (Replay record). *For long-lived certified reuse, the minimal replay record is:*

```

replay_record:
  library_entry_id:
  interface_signature:
  contract_id:
  checker_id:
  trusted_base_id:
  replay_tier:
  replay_codec_id:
  witness_hash:
  protected_trace_hash:
  dependency_hashes:
  validation_event_id:
  maintenance_due_time:
  retention_class:
  integrity_hash:

```

Definition 34 (Replay tiers). *Replay evidence has four tiers:*

1. **Hash replay:** exact files, prompts, outputs, and witnesses are preserved by hash.
2. **Container replay:** execution environment and dependencies are preserved or rebuildable.
3. **Semantic replay:** independent checker or contract verifies the same property under substrate drift.
4. **Statistical replay:** a stochastic procedure reproduces a declared distributional guarantee under logged seeds, configurations, and confidence rules.

Definition 35 (Trusted base registry). *A trusted base registry contains checker, evaluator, contract, replay-codec, and registry-checker entries. Each entry records identity, build identifier, scope, owner, validation method, allowed substrates, revocation condition, bridge rule, and last audit time. A library entry cannot be admitted under a revoked or out-of-scope trusted-base entry.*

Definition 36 (Trusted-base audit and revocation). *The registry has a declared trust root, signing or approval rule, append-only audit log, registry checker, audit cadence, and emergency revocation path. Every registry mutation creates an event node. If a trusted-base entry is revoked, out of scope, unaudited beyond its cadence, or inconsistent with the registry checker, the registry emits an incident node. All claims and library entries reachable from that node are quarantined or narrowed by the incident scope rule until a bridge, replacement checker, or independent revalidation restores support.*

Definition 37 (Certified library entry). *A certified library entry is an ordinary artifact plus*

$$(s, c, k, \rho, r, w, \tau, d, \eta, \zeta),$$

where s is typed interface signature, c contract or acceptance condition, k checker identity, ρ replay codec, r replay tier, w protected witness or trace hash, τ validation event, d dependency set, η maintenance envelope, and ζ trusted-base registry entry.

Definition 38 (Library states). *Each library entry is in one of:*

{ candidate, experimental, admitted, promoted, due, downgraded, quarantined, retired }.

Admitted entries satisfy integrity, replay, contract, and checker requirements. Promoted entries additionally have measured positive marginal value. Only promoted entries count as reusable production capital.

Definition 39 (Library admission budget and retention). *For each scope, admitted but unpromoted entries consume an admission budget with units such as registry slots, maintenance service, replay refresh, retrieval surface, security review, and context burden. Each admitted entry records admission time, time-to-live, promotion deadline, maintenance due time, retirement rule, and owner. If the budget is exhausted, new entries can be candidates or experimental entries but cannot be admitted unless another entry is promoted, downgraded, retired, or assigned additional reserved service. If the promotion deadline passes without sufficient promotion evidence, the entry is downgraded or retired and cannot support reinvestment claims.*

Protocol 8 (Admission without value circularity). *An entry becomes admitted only if:*

1. *replay record exists and passes integrity checks;*
2. *trusted-base entries for checker, evaluator, contract, replay codec, and registry checker are active and in scope;*
3. *interface signature and contract are machine-checkable or independently auditable;*
4. *validation has passed under a registered evaluator or checker;*
5. *replay tier, replay codec, witness, and dependencies are protected and retrievable;*
6. *permission tier, security state, and maintenance envelope are declared;*
7. *admission budget, time-to-live, promotion deadline, owner, and retirement rule are declared and reserved.*

Expected future value is not an admission condition. Value is measured after admission through promotion evidence.

Definition 40 (Promotion tiers). *Library value evidence has four tiers:*

1. **Descriptive:** reuse and outcome association.
2. **Operational:** capped access rollout or access throttling with rollback.
3. **Causal:** randomized access, cluster rollout, factorial test, library ablation, or valid bridge design.
4. **Maintenance:** repeated positive return after retrieval, context, replay, security, conflict, and maintenance costs over multiple windows.

Definition 41 (Promotion attribution record). *Promotion evidence records dependency graph, co-use set, marginal ablation design or reason it is unavailable, interaction penalty, and service-load attribution. If multiple entries are used together and attribution cannot be separated, the cohort may be promoted but individual entries cannot receive independent reinvestment credit.*

Definition 42 (Promotion credit classes). *Promotion credit is classified as individual, cohort-bounded, or unattributed. Individual credit requires a valid marginal design or bridge for one entry. Cohort-bounded credit permits a named set of co-used entries to receive a capped shared credit with an interaction penalty. Unattributed credit records useful association but cannot support reinvestment. Reinvestment claims may count only individual credit and cohort-bounded credit after dilution by the cohort cap.*

Definition 43 (Library return). *For admitted or promoted cohort C ,*

$$\text{ROI}_C = \frac{N_C}{D_C}.$$

Here

$$\begin{aligned} N_C &= V_C^{\text{certified}} - Q_C^{\text{baseline}} - I_C^{\text{interference}} - E_C^{\text{erosion}}, \\ D_C &= C_C^{\text{create}} + C_C^{\text{validate}} + C_C^{\text{replay}} + C_C^{\text{registry}} \\ &\quad + C_C^{\text{retrieve}} + C_C^{\text{context}} + C_C^{\text{security}} + C_C^{\text{maintain}} + C_C^{\text{retire}}. \end{aligned}$$

The promotion tier used to estimate N_C is reported.

Definition 44 (Positive-value lineage). *Partition promoted entries into classes. Let M_{cd}^+ be the expected discounted number of future promoted class- d entries caused by one promoted class- c entry, counting only descendants with positive lower-bound return after replay, validation, registry, and maintenance costs. Experimental, merely admitted, downgraded, and quarantined entries are excluded.*

Definition 45 (Library reproduction number).

$$\mathcal{R}_A^+ = \rho(M^+).$$

Definition 46 (Reinvestment ledger). *Reinvestment credit is recorded separately from ordinary library return. A promoted entry can create reinvestment credit only through individual or cohort-bounded promotion credit supported by access rollout, ablation, time-lagged holdout, transfer test, or bridge design. The ledger records parent entry, child entry, delay, service charges, maintenance charges, attribution class, lower-bound return, and whether the edge is causal or descriptive. Descriptive edges may inform search and retirement but cannot increase reinvestment acceleration.*

Definition 47 (Signed lineage accounting). *Library lineage is signed. Positive edges record caused promoted descendants with positive lower-bound return. Negative edges record caused regressions, cannibalized discoveries, maintenance drag, security burden, context burden, evaluator overfitting, and retired descendants. The reinvestment matrix used for strong claims is*

$$M^{net} = M^+ - M^-,$$

where both matrices charge replay, validation, registry, and maintenance costs. Reinvestment acceleration may use only positive lower bounds on $\rho(M^{net})$ or on a declared finite-horizon descendant total.

Definition 48 (Finite-horizon reinvestment lower bound). *Because spectral-radius estimates are sensitive to sparse data, the primary reinvestment estimand is a finite-horizon lower bound. For horizon h and promoted class vector v_0 , define*

$$D_h^{net} = \sum_{k=1}^h \mathbf{1}^\top \text{LCB}\left((M^{net})^k v_0\right),$$

where the lower bound accounts for edge uncertainty, attribution dilution, negative lineage, maintenance drag, and cohort interaction penalties. A spectral-radius claim may be reported only when every material matrix block has enough audited edges to estimate a lower confidence bound and the same result is not contradicted by D_h^{net} . Otherwise, reinvestment claims use D_h^{net} and the declared horizon.

Protocol 9 (Negative lineage audit). *For every promoted entry or promoted cohort, a negative lineage audit is scheduled at admission, promotion, maintenance due time, incident time, and retirement consideration. The audit checks at least one of: access ablation, holdout cohort, no-library shadow cohort, regression scan, security review, context-burden measurement, maintenance-service accounting, and cannibalization test against baseline work. If no audit design is feasible, the entry may remain useful operationally but cannot support reinvestment claims above descriptive lineage.*

\mathcal{R}_A^+ is retrospective. It is descriptive unless every counted edge in M^+ is supported by the reinvestment ledger as causal or cohort-bounded with a declared dilution cap. Daily control uses library health, service queues, per-use telemetry, marginal return, ablation, replay refresh, and retirement.

8 Online Control Kernel

Definition 49 (Control state). *At control time t , the compact online state is*

$$X_t = (T_t, S_t, G_t, E_t, W_t, P_t, A_t, L_t, B_t, \Gamma_t, C_t, Q_t, \Sigma_t),$$

where T is telemetry coverage, S substrate drift, G gate health, E evaluator health, W aged work-in-process, P bottleneck pressure, A ordinary artifact health, L certified library health, B baseline health, Γ remaining exploration budget, C claim state, Q service queues, and Σ service state.

Definition 50 (State reducers). *The online state is not an independent data source. It is the deterministic reduction of append-only ledgers:*

```
state_reducer:
  telemetry_reducer(edge_events) -> T, S
  gate_reducer(edge_events, incidents, hard_ledgers) -> G
```

```

wip_reducer(edge_events, item_events) -> W
pressure_reducer(W, service_queues, failures, critical_path) -> P
artifact_reducer(edge_events, dependency_graph) -> A
library_reducer(replay_records, registry, lineage, maintenance) -> L
baseline_reducer(baseline_registry, assignments, bridges) -> B
exploration_reducer(actions, incidents, costs) -> Gamma
claim_reducer(contracts, checker_results, incidents) -> C
service_reducer(service_ledgers, service_obligations,
                load_contracts) -> Q, Sigma

```

Each reducer declares input ledgers, ordering key, conflict rule, correction rule, and output hash. Replaying the same event prefix under the same reducer registry must produce the same X_t and state hash. A claim above observable level must name the reducer registry hash.

Protocol 10 (Reference reducer semantics). The reference reducers use event time, then append order, then event identifier as the total ordering key. Corrections create compensating deltas rather than deleting prior events.

```

telemetry_reducer(events):
  for event in ordered(events):
    verify envelope hash and parent link
    mark required fields observed, missing, not_applicable, or estimated
    update coverage by claim_scope, station, and stratum
    update substrate drift from fingerprint changes without bridge
    emit T, S, output_hash

service_reducer(service_events, obligations, contracts):
  for obligation in ordered(obligations):
    add required_quantity to channel, unit, deadline, priority bucket
    subtract reserved, completed, cancelled, and expired quantities
    carry held_quantity into next age bucket
  for service_event in ordered(service_events):
    add accepted completions to calibration numerator
    add eligible and unavailable exposure to denominator records
    update violation counts against load contracts
  compute Q and Sigma by channel, unit, deadline class, and window
  emit Q, Sigma, output_hash

claim_reducer(contracts, checker_results, incidents):
  for claim in ordered(contracts):
    set requested level and sealed epoch
  for incident in ordered(incidents):
    mark dependency-reachable claims quarantined or narrowed
  for result in ordered(checker_results):
    set supported level to the most recent non-stale checker result
  emit C, output_hash

```

An implementation may optimize storage or indexing, but it must return the same outputs on the same ordered ledger prefix. Any nondeterministic reducer caps claims at descriptive reporting.

Definition 51 (Claim dependency graph). Each non-descriptive claim has a dependency graph whose nodes are events, items, evaluators, datasets, baselines, service contracts, trusted-base entries, library

entries, frontier sources, ledgers, and gates used by the claim. Edges are typed as direct use, validation dependence, replay dependence, attribution dependence, baseline dependence, frontier dependence, service dependence, or bridge dependence.

Definition 52 (Dependency-graph coverage). *A strong claim reports dependency-graph coverage: the fraction of contract-required nodes whose incoming and outgoing dependence edges have been recorded or independently bounded. Unknown dependence is represented by a typed conservative summary edge to the smallest auditable upstream boundary that contains every plausible source. If the summary edge reaches a hard stop, revoked trusted-base entry, quarantined service contract, leakage incident, or corrupted replay record, the claim is quarantined or narrowed. The default unknown-edge budget is zero for frontier and reinvestment claims and at most 1% of contract-required nodes for production claims. The burden of removing a conservative edge is an audit or bridge burden, not a post-hoc judgment.*

Definition 53 (Boundary certificate). *Any compressed dependency boundary records boundary identifier, included node types, excluded node types, upstream sources, downstream claims, owner, audit method, last audit time, and expansion rule. If a boundary certificate is missing, stale, inconsistent, or contradicted by an incident, the checker expands the boundary to the next enclosing audited boundary. If no enclosing boundary exists, unknown dependence reaches the entire claim scope.*

Definition 54 (Boundary hierarchy registry). *Boundary expansion is defined by an append-only hierarchy registry. Each boundary records parent boundary, child boundaries, root boundary, ownership, audit cadence, last audit result, allowed summary-edge types, expansion cost, and maximum supported claim level. A compressed dependency edge is valid only if every node type it hides is included by the boundary certificate and every excluded node type is outside the claim scope. If the hierarchy has multiple possible parents, the checker expands to the smallest audited ancestor that contains all plausible sources; if no such ancestor exists, it expands to the root scope and applies the unknown-dependency budget.*

Protocol 11 (Incident scope rule). *When a hard stop, leakage, revoked trusted-base entry, evaluator failure, corrupted replay record, or service-contract quarantine occurs, every claim reachable from the incident node through the dependency graph is quarantined or narrowed by the predeclared edge labels. The scope cannot be narrowed by post-hoc choice outside the graph.*

Definition 55 (Permitted actions). *The controller may choose only from the declared set $\mathcal{A}_t^{\text{perm}}$: observe, route, throttle, reserve, rollout, audit, validate, replay, refresh, quarantine, rollback, retire, bridge, and escalate. Each action type has preconditions, blast-radius limit, service-load contract, rollback rule, and claim consequence.*

Protocol 12 (Online intervention state machine). *For each claim contract and each small batch, the controller executes:*

```
observe:
  append edge events and service ledger events
update:
  replay reducers for telemetry, queues, gates, dependencies, baselines
gate:
  hard stop -> quarantine affected claim scope
  missing required identifier -> downgrade or quarantine
  evaluator below floor -> stop evaluator-dependent optimization
reserve:
```

<p> <i>require active or recalibrated service contracts</i> <i>require window-wise capacity for all certification obligations</i> <i>act:</i> <i>mandatory repair before optional intervention</i> <i>optional intervention only if exposure bound fits exploration budget</i> <i>account:</i> <i>charge WIP, service, audit, rollback, library, and incident costs</i> <i>decide:</i> <i>run reference checker</i> <i>escalate only to checker-supported level</i> <i>downgrade or quarantine on failure codes</i> <i>freeze only under frozen claim accounting</i> </p>
--

The first three blocks are Layer 0 compatible. The reserve block activates Layer 1. Replay and library accounting activate Layer 2 only for reusable certified capital.

Protocol 13 (Transition priority). When multiple triggers fire in the same control step, transitions are applied from highest to lowest priority:

1. hard stop;
2. integrity failure;
3. trusted-base revocation;
4. evaluator revocation;
5. security or permission failure;
6. service overload;
7. missing required field;
8. baseline or frontier debt;
9. optional action.

Higher-priority transitions can quarantine, freeze, or downgrade lower-priority actions before they are evaluated. Optional gain-seeking action is never executed in a step with an unresolved higher-priority failure.

Definition 56 (Action score). After hard constraints are satisfied, non-mandatory production candidate action a at station j receives priority score

$$\text{Score}_t(a, j) = \frac{\text{PressureRelief}_t(a, j) + \text{CertifiedGainLCB}_t(a, j)}{\epsilon + \text{ServiceCost}_t(a, j) + \text{RiskBound}_t(a, j)}.$$

All terms are normalized under a declared utility convention before the score is used, and the small $\epsilon > 0$ prevents division by zero. The score is a scheduling convention, not evidence of value and not admissible as support for a strong claim.

Protocol 14 (Action queues). *Actions are separated into repair, risk-reducing, exploration, and production queues. Repair and risk-reducing queues are ordered by transition priority, age, and blast-radius reduction. Exploration queues are ordered by exposure bound and information value, not certified gain. Only production queues may use the action score containing certified gain. This prevents immature evaluators or exploratory proxies from becoming optimization targets.*

Protocol 15 (Online kernel). *For every event or small batch:*

1. *append Layer 0 event and optional operational fields;*
2. *append replay record only for outputs proposed as reusable certified entries;*
3. *update X_t using only \mathcal{F}_t ;*
4. *contain hard stops, permission failures, leakage, security failures, integrity failures, and trusted-base revocations using the incident scope rule;*
5. *stop evaluator-dependent optimization when evaluator gates fail;*
6. *throttle, reserve, or recalibrate when service queues exceed envelopes or service-load contracts are violated;*
7. *choose mandatory repairs first, then choose remaining actions by priority score subject to service reservation and exploration budget;*
8. *after observation, update certified value, service queues, WIP, audit state, library state, baselines, and claim state;*
9. *escalate claim strength only when the required evidence layer has been satisfied.*

9 Ledgers, Gates, and Resources

Definition 57 (Ledger schema). *Each progress ledger ℓ declares*

$$\Lambda_\ell = (u_\ell, c_\ell, h_\ell, d_\ell, \delta_\ell, n_\ell, \xi_\ell),$$

where the fields denote unit, certifier, validity horizon, double-counting rule, decay rule, negative-credit rule, and uncertainty rule.

Definition 58 (Standard progress ledgers). *The default ledger set for AI R&D acceleration is:*

1. **Defect-adjusted accepted change:** *accepted code, proof, data, model, tool, or document changes after rollback, regression, security, and maintenance charges.*
2. **Validated experiment throughput:** *independently reproducible or audited experiment conclusions per constrained resource.*
3. **Time-to-target:** *calendar time and active service exposure required to reach a contract-sealed target.*
4. **Frontier coverage:** *certified mass on independently admitted task strata.*

5. **Reusable-capital net return:** promoted library value after creation, validation, replay, registry, retrieval, context, security, maintenance, and retirement costs.

A claim may add domain-specific ledgers, but a production or stronger claim reports why each default ledger is used, constrained, or inapplicable. A ledger marked inapplicable cannot later be used as supporting evidence for the same claim without a new sealed contract.

Definition 59 (Hard constrained ledgers). *Security, safety, permission, data-boundary, and severe regression ledgers are hard constrained by default. A hard constrained ledger cannot be offset by scalar gains in capability, throughput, coverage, or time-to-target. If a hard constraint fails, the affected claim scope is quarantined, narrowed, or downgraded before any scalar margin is computed.*

Definition 60 (Progress vector). *The evaluation layer maps raw observations, evaluator outputs, audits, and library checks to*

$$G_t(\pi) = (G_t^{cap}, G_t^{ctrl}, G_t^{infra}, G_t^{know}),$$

where capability, control, infrastructure, and knowledge ledgers are declared by scope through ledger schemas.

Definition 61 (Gate state). *Each event, item, library entry, and window has a gate state*

$$\{\text{pass, warn, quarantine, rollback, hard_stop}\}.$$

pass allows ordinary credit. *warn* allows credit after warning costs. *quarantine* withholds credit until independent resolution. *rollback* charges repair and reversal costs. *hard_stop* invalidates, stops, or narrows the affected claim scope before scalar value is computed.

Definition 62 (Resource vector). *The primary resource object is*

$$R_t = (R_t^{op}, R_t^{svc}),$$

where

$$\begin{aligned} R_t^{op} &= (R_t^{wall}, R_t^{compute}, R_t^{tokens}, R_t^{tools}, R_t^{human}, R_t^{money}, R_t^{external}, R_t^{instrument}), \\ R_t^{svc} &= (R_t^{validation}, R_t^{audit}, R_t^{replay}, R_t^{maintenance}, R_t^{registry}). \end{aligned}$$

Missing components are reported as missing, estimated, or out of scope. They are not silently priced as zero.

Definition 63 (Gate-adjusted margin). *For non-hard-stop states and declared ledger weights λ and resource prices p , define scalar margin against local reference rate τ as*

$$Z_t = \lambda^\top G_t - \tau p^\top R_t - D_t,$$

where D_t includes warning, rollback, incident, WIP, instrumentation, service, library, and artifact charges that apply to the claim. If *hard_stop* occurs, Z_t is not used for the affected primary claim.

Scalar margins are decision conveniences. Raw telemetry, vector resources, ledgers, gates, and service queues remain primary evidence.

10 Baselines, Target Frontier, and Coverage

Definition 64 (Baseline registry). *A baseline registry contains frozen, rolling, shadow, and external baselines. Each entry records policy identity, substrate fingerprint, service state, tools, permission tier,*

resource caps, update time, allowed information, task eligibility, assignment rule, contamination test, known limitations, and bridge status.

Definition 65 (Shadow baseline). *A shadow baseline is run on an assigned fraction of live or replayable tasks to monitor whether claimed gains survive ordinary task drift, substrate drift, and service-load changes. Its sampling rate may be small, but its assignment rule, eligibility set, and non-interference rule must be recorded. The default floor is the maximum of 5% of eligible task mass, 20 assigned units per active production window, and the sample size required by the sealed minimum detectable effect and variance bound, capped only by the eligible finite population. A contract may use a lower rate only by lowering the maximum supported claim level or by proving that safety or service constraints make the shadow design infeasible. A production or stronger claim with no shadow baseline must use a frozen or external baseline with a passed bridge and report baseline debt.*

Definition 66 (Baseline update contract). *Baseline changes are controlled by:*

```
baseline_update_contract:
  baseline_id:
  update_type:
  reason:
  sealed_before:
  allowed_information:
  excluded_information:
  assignment_rule:
  contamination_tests:
  bridge_design:
  maximum_update_frequency:
  max_supported_claim_level:
  integrity_hash:
```

If a rolling baseline learns from outcomes of the evaluated policy without a declared contamination test and bridge, it may remain an engineering reference but cannot support production or stronger claims.

Definition 67 (Bridge matrix). *For substrate, baseline, evaluator, or service states i and j , bridge entry M_{ij} records the estimated relation between outcomes under the two states, the design used to estimate it, the task strata covered, and uncertainty. Missing material entries are baseline debt.*

Definition 68 (Frontier source registry). *Each target-frontier source records source identity, quota, sealing time, admission authority, blinding rule, leakage screen, deduplication rule, update cadence, exclusion rule, and weight rule. Quotas and weights are frozen before outcome data from the evaluated policy are used. A task, rubric, evaluator, or value model produced by the evaluated policy may enter only escrow until independent admission, delayed transfer, or an external bridge breaks the dependency.*

Definition 69 (Frontier governance contract). *A frontier source supports frontier claims only through a governance contract:*

```
frontier_governance_contract:
  source_id:
  admission_authority:
  independence_rule:
  stakeholder_or_domain_scope:
  weight_authority:
```

```

weight_audit_rule:
leakage_screen:
deduplication_rule:
difficulty_strata:
quota_freeze_time:
update_cadence:
dispute_rule:
max_supported_claim_level:
integrity_hash:

```

Weights are admissible only if they are assigned before outcomes or by an authority blinded to evaluated-policy outcomes. If task importance, hardness, or weights are revised after observing outcomes, the checker treats the change as frontier debt and downgrades to production unless a bridge design covers the change.

Definition 70 (Frontier sampling frame). *A frontier claim declares a sampling frame with task source, eligibility rule, hardness proxy, stratum labels, quota, minimum task mass, minimum calendar duration, and leakage screen. The default minimum task mass is 50 independent task units or all eligible units in a sealed small population; the default minimum calendar duration is three active control windows. Hardness proxies are fixed before outcomes and are used only for stratification, not as value evidence. Task churn is reported as additions, removals, and reweighting; unexplained churn is baseline debt.*

Definition 71 (Frontier mass rule). *The minimum task mass for a frontier claim is the maximum of the default mass, the number required by the declared minimum detectable effect, the number required by the cluster or interference design, and the number required to cover each material stratum. If the eligible sealed population is smaller, the claim is finite-population only and cannot generalize beyond that population without transfer evidence. The minimum detectable effect, variance bound, cluster count, and coverage target are sealed before outcomes.*

Definition 72 (Target frontier update). *The practical target frontier is updated by*

$$F_{t+1} = \text{Update}(F_t, B_t^{\text{backlog}}, I_t^{\text{incident}}, U_t^{\text{user}}, H_t^{\text{holdout}}, T_t^{\text{transfer}}, E_t^{\text{escrow}}),$$

where backlog, incident stream, user or stakeholder tasks, sealed holdouts, transfer targets, and independently admitted escrow items are merged by the frontier source registry. AI-generated novelty proposals enter escrow first and cannot enter the primary frontier until independent admission or delayed transfer breaks self-selection.

Definition 73 (Practical frontier coverage). *For task universe \mathcal{U} , target frontier $F_t \subseteq \mathcal{U}$, certified set $C_t \subseteq \mathcal{U}$, and positive weights ν , coverage is*

$$\text{Cov}_t(C_t; \nu) = \frac{\sum_{x \in C_t \cap F_t} \nu(x)}{\sum_{x \in F_t} \nu(x)}.$$

Robust coverage is the infimum over a declared ambiguity class of weights.

AI-generated tasks, rubrics, benchmarks, evaluators, and value models cannot certify their parent policy unless independent validation, delayed transfer, or an external bridge breaks the dependency. Strata, quotas, and coverage weights prevent apparent acceleration from being created by shifting toward easier or more self-certifying work.

11 Work-in-Process and Delayed Validation

Definition 74 (Item state). *Each item is in one of:*

$$\{\text{created, active, submitted, certified, rejected, deferred, expired, quarantined}\}.$$

Primary value is credited only at certification or at a declared delayed-validation milestone.

Definition 75 (WIP conservation). *For stratum s , station j , and age bucket k ,*

$$W_{s,j,k,t+1} = \text{age}(W_{s,j,k,t}) + C_{s,j,k,t} - D_{s,j,k,t} - X_{s,j,k,t} - Q_{s,j,k,t},$$

where C is newly created work, D is certified or rejected departure, X is expiration, and Q is quarantine.

Definition 76 (Queue-specific liability). *For declared weights $a_{s,j,k} \geq 0$ and deferred-validation charge $d_s \geq 0$,*

$$L_t^{\text{wip}} = \sum_{s,j,k} a_{s,j,k} W_{s,j,k,t} + \sum_s d_s \text{Deferred}_{s,t} + \text{ExpiredCost}_t.$$

A production acceleration claim must show that gains are not explained by increasing this liability or by pushing validation past the horizon.

Definition 77 (Delayed audited label). *For item i , a delayed audited label record contains assignment probability p_i , assignment time, reveal time, auditor or evaluator identity, observed label Y_i , missingness state, and stratum. If p_i is unknown or unlogged, the label may be descriptive but cannot support a strong sequential or off-policy claim.*

Assumption 1 (Design-valid delayed labels). *For any delayed label used in a strong sequential, off-policy, audited, or production claim, the assignment probability is recorded exactly at assignment time, positivity holds with $p_i \geq p_{\min} > 0$ in the target stratum, outcomes are bounded or capped by a declared rule, adaptive sampling inputs are logged, and missingness or reveal-time censoring is handled by the gate rule. Labels that fail these conditions may be reported descriptively but cannot support inverse-propensity, doubly robust, or time-uniform claims.*

Definition 78 (Causal estimand contract). *Any causal, off-policy, or strong sequential estimate declares estimand, assignment unit, target population, exposure mapping, interference graph, calendar horizon, active-horizon rule, outcome cap, missing-label rule, and estimator. If interference can cross assignment units, the design uses cluster, switchback, stepped-wedge, or exposure-mapped analysis. Adaptive stopping cannot choose the estimand, horizon, or outcome cap after seeing outcomes; time-uniform tools may stop actions for safety but cannot create extra numerator credit.*

Definition 79 (Estimator admissibility profile). *Each estimator used for a strong claim declares required logs, required randomization or assignment record, positivity floor, outcome cap, clipping rule, variance cap, model-training data, cross-fitting or holdout rule, diagnostics, fallback estimator, maximum supported claim level on pass, and maximum supported claim level on fail. If diagnostics fail, the checker falls back to the strongest admissible conservative estimator and caps the supported level at the declared failure level. A doubly robust estimator may support a strong claim only when the propensity record is exact, the outcome model is trained without outcome leakage, and residual diagnostics or conservative bounds are recorded.*

Definition 80 (Estimator profile schema).

```
estimator_profile:
  estimator_id:
  estimand:
  required_logs:
  assignment_record:
  positivity_floor:
  outcome_cap:
  clipping_rule:
  variance_cap:
  training_data_exclusions:
  diagnostics:
  fallback_estimator:
  max_claim_level_on_pass:
  max_claim_level_on_fail:
  integrity_hash:
```

If either maximum level is absent, the checker assumes observable for failure and audited for pass, unless a stricter claim-level profile overrides this default.

Definition 81 (Logged-propensity estimator). For audited sample indicator A_i , assignment probability $p_i > 0$, and value label Y_i , the design-weighted finite-window total value estimate for a stratum is

$$\hat{V}_{HT} = \sum_i \frac{A_i Y_i}{p_i}.$$

When a valid outcome model $\hat{m}(x_i)$ is declared, a doubly robust variant may replace this with

$$\hat{V}_{DR} = \sum_i \left[\hat{m}(x_i) + \frac{A_i}{p_i} (Y_i - \hat{m}(x_i)) \right].$$

Mean value estimates divide the total estimate by the declared target mass or item count. Variance estimates use the logged sampling design or a conservative upper bound. Missing labels and reveal-time censoring are charged through the uncertainty rule or gate state.

Definition 82 (Hierarchical backoff). Let $\psi_0, \psi_1, \dots, \psi_m$ be observable task or interface labels from fine to coarse. When a fine cell lacks enough completed delayed labels, the claim backs off to the coarsest ancestor with sufficient audited support and charges a declared drift penalty.

12 Evaluator Audits

Definition 83 (Evaluator health). Evaluator health is

$$H_t^E = (g_t, b_t, c_t, s_t, \ell_t, d_t, m_t, u_t),$$

where g is known-good pass rate, b known-bad reject rate, c canary integrity, s shortcut-probe failure, ℓ leakage signal, d blinded-review disagreement, m missingness, and u unresolved incidents.

Definition 84 (Evaluator state machine). Each evaluator is in exactly one state:

$$\{\text{untrusted, monitored, audited, bridged, revoked}\}.$$

Untrusted evaluators can support descriptive observations only. Monitored evaluators can support controlled local claims if sentinel checks pass. Audited evaluators can support service-controlled and audited claims within their scope. Bridged evaluators can support production or stronger claims only for strata covered by a passed bridge audit. Revoked evaluators support no positive credit and create an incident node. State promotion requires logged canaries, known-good and known-bad checks, shortcut probes, leakage screens, missingness bounds, and audit-age compliance at the floor required by the claim contract.

Definition 85 (Evaluator promotion profile). *Evaluator promotion requires a sealed rotating sentinel pool, canary exposure budget, known-good and known-bad sample sources, shortcut probes, leakage probes, blinded disagreement rule, drift test, audit cadence, and revocation trigger. The sentinel pool is sampled without revealing labels to the optimized policy. If canary exposure exceeds its budget, if shortcut probes become predictable, or if drift exceeds the bridge scope, the evaluator returns to monitored or revoked state and dependent optimization stops.*

Definition 86 (Default evaluator floors). *Unless a stricter profile is sealed, evaluator promotion uses these floors. Monitored state requires known-good pass rate and known-bad rejection rate lower confidence bounds at least 0.90, canary exposure within budget, no unresolved leakage incident, and audit age within two active control windows. Audited state requires corresponding lower bounds at least 0.95, shortcut-probe failure rate upper confidence bound at most 0.05, blinded disagreement upper bound at most 0.10 on material strata, missingness below the telemetry floor, and audit age within one active control window. Bridged state requires audited state plus passed bridge diagnostics for every target stratum. Any unresolved evaluator incident sets the evaluator to revoked for affected strata until repair and bridge audit pass. These floors are defaults for claim checking, not claims that the evaluator measures true value without error.*

Definition 87 (Evaluator audit profile schema). *Each evaluator used above descriptive reporting declares:*

```
evaluator_audit_profile:
  evaluator_id:
  target_strata:
  alpha:
  known_good_min_count:
  known_bad_min_count:
  canary_min_count:
  shortcut_probe_min_count:
  blinded_review_min_count:
  lcb_method:
  ucb_method:
  canary_exposure_budget:
  leakage_probe_rule:
  drift_test:
  audit_cadence:
  max_supported_claim_level_on_pass:
  max_supported_claim_level_on_fail:
  integrity_hash:
```

The default α is 0.05 per active audit family after the declared multiple-testing correction. The default minimum count is 30 per material stratum for known-good, known-bad, shortcut, and blinded-review checks, or all sealed units in a smaller finite population. If minimum counts are not met, the evaluator may remain monitored for local control but cannot support production or stronger claims.

Definition 88 (Audit scheduler). *For evaluator exposure volume V_t , evaluator drift Δ_t , claim level r , incident indicator I_t , and audit queue pressure Q_t^{audit} , the minimum audit sample size request is*

$$n_t^{\text{audit}} \geq n_0 + \lceil \alpha_V V_t + \alpha_\Delta \Delta_t + \alpha_r r + \alpha_I I_t + \alpha_Q Q_t^{\text{audit}} \rceil.$$

If audit capacity cannot meet the request within the age envelope, affected optimization or claim escalation stops.

Protocol 16 (Evaluator stop rule). *If a canary is exposed, known-bad rejection fails materially, shortcut probes succeed, leakage is detected, blinded disagreement exceeds threshold, or audit queue age exceeds the claim envelope, optimization against that evaluator stops for affected strata. Existing evaluator-dependent claims are quarantined until the evaluator is replaced or repaired and a bridge audit is passed.*

Theorem 4 (Audit lower bound). *If a full audit gives simultaneous upper bounds on value-weighted false accepts, leakage, and missing certifiedness in all declared strata, then subtracting those bounds from observed accepted value gives a conservative lower bound on certified value with the declared coverage. False rejects are handled as opportunity loss or evaluator-health degradation, not as a subtraction from accepted certified value.*

Proof. On the simultaneous coverage event, unearned accepted value is no greater than the false-accept, leakage, and missing-certifiedness bounds. Subtracting those bounds gives a lower bound on certified accepted value. False rejects reduce discovered value but do not make accepted value too large. \square

13 Bottleneck Pressure and Sequential Control

Definition 89 (Station pressure). *For station j , define*

$$\begin{aligned} P_{j,t} = & \omega_q Q_{j,t} + \omega_w \text{Wait}_{j,t} + \omega_u U_{j,t} + \omega_r \text{Rework}_{j,t} \\ & + \omega_f F_{j,t} + \omega_i I_{j,t} + \omega_c \text{CPath}_{j,t} + \omega_s \text{SLoad}_{j,t}. \end{aligned}$$

The components are normalized queue length, waiting time, utilization, rework, failure rate, incident load, critical-path share, and service-load pressure.

Protocol 17 (Pressure trigger). *A reversible relief action may be tried when $P_{j,t}$ exceeds a declared threshold for K consecutive control windows, the station is not in cooldown, relevant gates pass, service reservations exist, and remaining exploration budget covers worst-case exposure. Action stops when pressure falls below the lower hysteresis threshold, certified value fails to improve, a rollback trigger fires, or an incident occurs.*

Definition 90 (Exploration budget). *An exploration budget Γ_t is the remaining allowed drawdown in gate-adjusted margin, resource waste, rollback burden, service burden, and blast-radius exposure over a horizon. It is charged before risky action by a conservative pre-action bound and updated after outcomes are observed.*

Definition 91 (Pre-action exposure bound). *For action a , let $D_t(a)$ be an \mathcal{F}_t -measurable upper bound on worst-case admissible loss before the next review point, including service load created by the action. The action is inadmissible if $D_t(a) > \Gamma_t$, if required service reservation fails, or if any mandatory gate fails.*

Assumption 2 (Sequential monitoring conditions). *For a monitored claim scope, increments of gate-adjusted margin are bounded or sub-exponential under declared constants, missing outcomes are handled by the gate rule, and the lower confidence sequence is time-uniform for the filtration generated by the online event stream. If the lower confidence sequence cannot be updated because required observations are missing, the action stream is throttled or the claim is frozen.*

Definition 92 (Sequential monitoring profile). *The monitoring assumptions are admissible only through a sealed profile:*

```
sequential_monitoring_profile:
  claim_id:
  increment_definition:
  increment_cap:
  tail_family:
  tail_diagnostic:
  missingness_rule:
  incident_handling_rule:
  confidence_sequence_method:
  alpha:
  review_interval:
  freeze_trigger:
  max_supported_claim_level_on_diagnostic_fail:
  integrity_hash:
```

If the increment cap is violated, the tail diagnostic fails, or missingness is above the declared bound, the checker freezes the monitored claim or falls back to the declared failure level. A production or stronger sequential claim cannot rely on an unprofiled sub-exponential assumption.

Protocol 18 (Frozen claim accounting). *When a claim is frozen, frozen duration, frozen outputs, and deferred labels created during the freeze do not enter the active-horizon numerator, duration threshold, or service-compliance count for a stronger claim. Restart requires either the same substrate, baseline, evaluator, service, and frontier state or a bridge record covering all material changes. A freeze prevents further downside exposure from the monitored claim; it is not evidence of improvement.*

Protocol 19 (Sequential admission rule). *An action may proceed only if one condition holds:*

1. *it is pure observation with negligible operational risk;*
2. *it satisfies epsilon dominance or constrained improvement on a rolling holdout;*
3. *it is reversible, passes screens, is out of cooldown, has a rollback trigger, has service reservation, and satisfies $D_t(a) \leq \Gamma_t$;*
4. *it has positive lower-bound incremental value under a Tier 2 design;*
5. *it repairs a gate, evaluator, security, replay, maintenance, or integrity failure without broadening the claim.*

Repeated adaptive changes require a rolling holdout, control chart, or time-uniform confidence tool.

Theorem 5 (Sequential drawdown guarantee). *Let $M_t = \sum_{i \leq t} Z_i$ be cumulative gate-adjusted margin for a declared claim scope over active, non-frozen periods. Under the sequential monitoring conditions, suppose $\text{LCB}_t(M_t)$ is a time-uniform lower confidence sequence with coverage $1 - \alpha$, non-observational actions are rejected whenever their pre-action exposure would make the post-action lower bound fall below $-\Gamma_0$, and missing required observations freeze or throttle the claim. Then, with probability at least $1 - \alpha$, monitored cumulative margin never falls below $-\Gamma_0$ over the active horizon.*

Proof. On the time-uniform coverage event, $M_t \geq \text{LCB}_t(M_t)$ for all active monitored t . The rejection, freeze, and throttle rules prevent active continuation when the lower bound would fall below $-\Gamma_0$. Frozen periods add no numerator or duration credit to the stronger claim. Hence $M_t \geq -\Gamma_0$ throughout active monitored periods on that event. \square

The theorem bounds online downside under declared assumptions. It does not prove improvement; improvement requires comparison, certification, and service-capacity evidence.

14 Improvement Criteria

Definition 93 (Claim-level uncertainty rules). *Allowed uncertainty rules are tied to claim strength. Observable and controlled local claims may use descriptive intervals or bootstrap checks. Service-controlled and audited claims require logged sampling rules, rolling holdouts, or conservative design-based bounds. Production and stronger claims require rolling holdouts, bridge designs, time-uniform confidence sequences, randomized designs, or declared conservative bounds justified by the service and task process. A claim cannot choose a weaker uncertainty rule after seeing the result.*

Definition 94 (Operational and causal acceleration claims). *Production claims are split into operational and causal forms. An operational production claim states that a sealed policy-substrate-service loop achieved certified progress under the logged conditions, after charges and baselines. It may use conservative design-based or bridge bounds. A causal production claim states that an intervention caused the improvement relative to a counterfactual policy. It requires randomized assignment, shadow assignment, switchback, stepped-wedge, cluster rollout, valid instrument, or a predeclared bridge design with logged assignment probabilities and interference handling. If a claim lacks such design evidence, the checker may support operational production but not causal production, transfer causality, or reinvestment causality.*

Definition 95 (Epsilon dominance). *For tolerance vector $\epsilon_R \geq 0$ and ledger tolerance $\epsilon_G \geq 0$, policy π epsilon-dominates baseline β over horizon H if, under an uncertainty rule allowed for the claim level, π is no worse than β by more than ϵ_G on every primary ledger, no worse by more than ϵ_R on constrained resources and service queues, has no worse gate profile, does not increase unresolved WIP or library liability, and strictly improves at least one primary ledger or constrained resource beyond tolerance.*

Definition 96 (Constrained improvement). *When dominance is too strict, a claim may fix resource caps, service envelopes, gate requirements, WIP limits, task strata, exploration budget, and library policy, then compare certified value within those constraints.*

Definition 97 (Optional scalar rate). *Given declared ledger weights λ and resource prices p , the scalar finite-horizon rate is*

$$\mu_{\pi,H}^{\lambda,p} = \frac{\mathbb{E}_{\pi}[\sum_{t \in H} \lambda^{\top} G_t - D_t]}{\mathbb{E}_{\pi}[\sum_{t \in H} p^{\top} R_t]}.$$

It is a decision convention, not the primary raw evidence.

Theorem 6 (Rate-improvement identity). *Let a baseline have expected scalar value $y_0 > 0$, expected scalar resource cost $r_0 > 0$, and rate $\tau = y_0/r_0$. Let an intervention change expected value and cost by Δy and Δr , with $r_0 + \Delta r > 0$. The intervention increases scalar rate exactly when*

$$\Delta y - \tau \Delta r > 0.$$

Proof. Subtract y_0/r_0 from $(y_0 + \Delta y)/(r_0 + \Delta r)$. The denominator is positive, so the sign is determined by $\Delta y - \tau \Delta r$. \square

15 Claim Ladder

Claim	Minimum evidence
Observable loop	Layer 0 integrity, coverage, and state reconstruction
Controlled local change	gates, WIP conservation, and bounded intervention class
Service-controlled improvement	Layer 1 service channels, active load contracts, reservations, and completed outcomes
Audited improvement	monitored or audited evaluator state, logged sampling, and relevant audit checks
Operational production acceleration	complete claim contract, repeated windows, shadow or bridged baseline, service envelopes, no unresolved hard stop
Causal production acceleration	operational production plus logged assignment, interference handling, and admissible causal design
Admitted-library capability	Layer 2 replay record, checker, contract, validation, and maintenance envelope
Promoted-library acceleration	admitted entries plus operational, causal, or maintenance promotion evidence
Transfer acceleration	operational or causal production acceleration plus bridged transfer-target results
Frontier acceleration	operational or causal production, or transfer acceleration, on sealed frontier sampling frames
Reinvestment acceleration	promoted entries show positive causal or cohort-bounded reinvestment ledger edges

Definition 98 (Dramatic acceleration profile). *A dramatic practical acceleration claim reports*

$$\Phi = (f, g_{rel}, g_{abs}, \Delta T, T, N, E, b, u, c, q, d, \sigma),$$

where f is claim form, g_{rel} is relative gain, g_{abs} absolute gain, ΔT time-to-target reduction, T calendar and active duration, N certified task mass, E service exposure, b transfer breadth, u promoted certified reuse, c implementation burden, q gate profile, d baseline debt, and σ service-capacity profile. A scalar gain is not sufficient for dramatic acceleration. The profile must show either independently admitted frontier coverage gain or promoted certified reuse that changes future verified production, with scalar gains reported as supporting evidence.

Definition 99 (Default dramatic profile). *Unless a stricter profile is declared before observation, a dramatic practical acceleration claim must satisfy all default conditions:*

1. *operational production acceleration or stronger for at least three consecutive active control windows, and causal production evidence if the dramatic claim is causal;*
2. *at least 50 independent certified task units or all eligible units in a sealed small population;*
3. *the same calendar horizon, active-horizon rule, and minimum service exposure are used for intervention and baseline comparison;*
4. *positive absolute gain after all gate, WIP, service, audit, library, maintenance, and instrumentation charges;*
5. *independently admitted frontier coverage gain, or promoted certified reuse with individual or cohort-bounded attribution and positive lower-bound future return;*
6. *scalar relative gain $g_{rel} \geq 1.5$ under a declared scalar convention, or epsilon dominance with at least one primary time-to-target, coverage, or constrained-resource improvement of at least 20%;*
7. *service queues remain inside envelopes and service-contract states are active or recalibrated throughout credited windows;*
8. *daily instrumentation burden remains below the default ceiling or the claim remains positive after charging the excess burden;*
9. *frozen periods, uncalibrated service channels, and missing material fields are excluded from the numerator and duration threshold;*
10. *the claim contains fresh-task survival, baseline-debt sensitivity, and service-capacity sensitivity.*

A dramatic claim must reduce time to a meaningful target or expand a verified frontier, maintain service queues inside envelopes, control implementation burden, address material baseline debt, show no hidden WIP transfer, and pass sensitivity checks over resource prices, ledger weights, task strata, baselines, service capacities, and library attribution. A local speedup alone is not dramatic acceleration.

16 Deployment Protocol

Protocol 20 (Minimal deployment). *Deploy LOSCR in seven stages:*

1. **Edge logging:** *record Layer 0 events for AI-assisted runs and relevant reference processes.*
2. **State reconstruction:** *compute telemetry coverage, substrate drift, gates, WIP, pressure, queue age, artifact use, and baseline coverage.*
3. **Low-risk control:** *enable observe, route, throttle, quarantine, rollback, retire, and bounded rollout actions with declared triggers.*
4. **Service control:** *define service channels, calibrated load contracts, reservations, age-bucket queues, and overload actions for validation, audit, replay, maintenance, and registry services.*
5. **Comparison registry:** *add frozen, rolling, shadow, and external baselines; maintain bridge matrix and baseline debt.*

6. **Certified library module:** register the trusted base, admit entries by replayable evidence, then promote them only by measured marginal value and attribution records.
7. **Strong claims:** add full audits, transfer tests, library ablations, Tier 2 causal designs, and sequential confidence tools.

The first three stages are already useful without strong claims: they reduce unobserved drift, reveal WIP buildup, expose bottlenecks, and prevent uncontrolled optimization against broken evaluators. Stage four adds service realism. Stage six is optional unless reusable certified capital is claimed.

Protocol 21 (Reference implementation contract). *A minimal implementation of LOSCR consists of:*

1. *append-only storage for edge envelopes, sidecars, service ledger events, incidents, contracts, checker results, and reducer outputs;*
2. *adapters for at least one work source, one evaluator or CI source, and one resource source;*
3. *deterministic reducers with replayable output hashes;*
4. *a checker implementing the reference checker order and failure-code taxonomy;*
5. *a dashboard or report showing current supported claim level, missing fields, oldest queue age, service reservation status, evaluator state, baseline debt, and active incidents;*
6. *conformance tests using synthetic logs for missing identifier, stale evaluator, service overload, baseline contamination, corrupted replay, and hard-stop reachability.*

An implementation that lacks reducers or checker conformance tests may use the terminology descriptively, but it cannot claim machine-checked production acceleration.

Protocol 22 (Daily minimal profile). *For ordinary AI-assisted development, the default lightweight profile records the event envelope through adapters and asks humans for only the fields that the adapters cannot infer: station, action type, raw status, dependency flag, reuse count, claim scope, and any exceptional incident label. It maintains five counters by station and stratum: completed items, unresolved WIP, maximum queue age, failed or rolled-back items, and missing required fields. It performs seven automatic interventions: quarantine hard stops, stop evaluator-dependent optimization on evaluator failure, throttle when maximum queue age exceeds envelope, downgrade claims when required fields are missing, reserve service before production credit, freeze rather than continue when confidence updates are impossible, and retire or downgrade admitted library entries past their deadline. This profile is sufficient for observable and controlled local claims and activates sidecar fields only when the claim contract, service contract, evaluator audit, baseline assignment, incident rule, or reusable-library claim requires them.*

Definition 100 (Canonical domain ledger maps). *The default domain maps from events to progress ledgers are:*

<i>Domain</i>	<i>Primary certified unit</i>	<i>Required charges</i>
<i>Software engineering</i>	<i>accepted change passing sealed tests, review, security gates, and regression checks</i>	<i>failed runs, review time, rollbacks, incidents, CI capacity, maintenance, dependency drift, and instrumentation</i>
<i>Formal proving</i>	<i>theorem, lemma, tactic, or proof artifact accepted by the declared checker under trusted dependencies</i>	<i>proof search time, checker time, dependency additions, replay refresh, broken library imports, and maintenance</i>
<i>ML research engineering</i>	<i>reproducible experiment conclusion, validated metric change, leakage-free dataset change, or accepted analysis artifact</i>	<i>failed runs, compute, evaluator audits, reproduction, leakage checks, reviewer time, storage, and stale-result retirement</i>

A domain may add units, but a production or stronger claim must explain every omitted required charge through a sealed inapplicability rule. A scalar ledger weight can aggregate these units only after the raw units and charges remain separately reported.

Protocol 23 (Empirical evaluation and stress-test program). *The theory is evaluated and stress-tested in five increasing stages:*

1. **Synthetic conformance:** *generate logs with known missing identifiers, stale evaluators, service overload, baseline contamination, corrupted replay, and incident reachability; require the checker to emit the expected status and supported level.*
2. **Retrospective replay:** *replay historical project logs using adapters; measure event reconstruction, reducer determinism, burden, missingness, and whether apparent gains are explained by WIP, service overload, evaluator drift, or baseline debt.*
3. **Live lightweight pilot:** *run the daily profile on ordinary work; report wall-time, token, compute, storage, and human-seconds-per-event overhead.*
4. **Controlled service pilot:** *activate service obligations, reservations, shadow baselines, evaluator floors, and bridge records; test whether throttling decisions predict queue-age and incident reduction.*
5. **Reusable-capital pilot:** *admit and promote a small library under replay and attribution rules; compare promoted, admitted-only, and no-library cohorts after maintenance and negative lineage charges.*

A claim that has passed only stages one to three may support lightweight operational control but not dramatic acceleration. Dramatic claims require at least one controlled service pilot and one reusable-capital or frontier-coverage pilot in the declared domain.

17 Experimental Programs

17.1 Software Engineering

Use private, rotating, or internally authored repository tasks pinned to commits. Layer 0 records the event envelope through Git, CI, review, LLM-call, and tool-call adapters. Raw state transitions, queue timestamps, evaluator identities, and incident links are sidecars activated by claim contracts and audits. Layer 2 is required only for reusable patches, repository maps, failing-test reducers, migration tools, or CI repair procedures claimed as certified library entries. Long-term maintenance tasks should include CI-style repeated commits, dependency drift, regression tracking, replay refresh, and service queue reporting.

17.2 Formal Theorem Proving

Use proof-checker certification. Core outcomes include proof accepted, proof rejected, proof length, tactic time, dependency additions, and verifier failures. Certified library entries include interface signatures for theorem families, replayable proof traces, checker identity, dependency hashes, and maintenance envelopes for library changes.

17.3 Machine-Learning Research Engineering

Use fixed-resource environments with contamination checks and independent reproduction when feasible. Core outcomes include experiment completion, metric result, compute use, failed runs, reproduction status, leakage checks, reviewer action, and artifact dependencies. Reusable entries include experiment templates, ablation schedulers, leakage detectors, result analyzers, compute allocators, and report checkers only after replay, validation, and maintenance costs are charged.

17.4 Evaluator Repair

Run rotating sentinel audits continuously. If known-good, known-bad, canary, shortcut, leakage, blinded-review, or audit-age checks fail, repair or quarantine the evaluator before optimizing against it. Evaluator repair is credited through improved evaluator health and downstream production rate, not through easier certification.

18 Falsification and Narrowing Conditions

Definition 101 (Falsification-to-checker map). *Every narrowing condition is represented by a checker failure family. The default maximum supported levels are:*

<i>Condition class</i>	<i>Failure family</i>	<i>Maximum supported level</i>
<i>absent checker, nondeterminism, unreplayable reducer</i>	<i>integrity or profile</i>	<i>descriptive only</i>
<i>missing identifier, broken hash, corrupted replay</i>	<i>integrity</i>	<i>none for affected scope until repaired</i>
<i>missing sidecar or nonmaterial ledger field</i>	<i>missing or ledger</i>	<i>strongest canonical profile not using the field</i>
<i>unbridged epoch, baseline, evaluator, frontier, substrate, or service change</i>	<i>epoch or baseline</i>	<i>descriptive comparison unless bridge passes</i>
<i>hard constraint, permission, security, leakage, or incident reachability</i>	<i>hard_constraint or dependency</i>	<i>quarantine for affected scope</i>
<i>service overload, unreserved obligations, or uncalibrated material service</i>	<i>service</i>	<i>controlled unless zero-bound reservation still passes</i>
<i>evaluator floor, sentinel, canary, drift, leakage, or audit-age failure</i>	<i>evaluator</i>	<i>controlled for unaffected strata, quarantine for affected strata</i>
<i>frontier admission, quota, blinding, deduplication, or weight failure</i>	<i>frontier</i>	<i>operational production at most</i>
<i>estimator, positivity, assignment, interference, or sequential diagnostic failure</i>	<i>estimator</i>	<i>estimator profile failure level</i>
<i>reinvestment attribution, negative lineage, or maintenance-drag failure</i>	<i>dependency or ledger</i>	<i>operational production at most</i>
<i>instrumentation burden above ceiling without charge</i>	<i>burden</i>	<i>operational production at most and no dramatic claim</i>

LOSCR should be rejected or narrowed for a declared scope if:

1. the contract validity checker is absent, non-deterministic, or unable to emit a supported claim level;
2. a required claim-contract field is absent, ambiguous, self-inconsistent, or sealed after outcome data are used;
3. a material contract epoch change is pooled without a bridge;
4. a claim-level profile is absent or inconsistent with the requested claim level;
5. default ledgers are omitted without a contract-level inapplicability rule;
6. hard constrained ledgers fail and scalar gains are used to offset them;
7. maintained baselines match or exceed the claimed gain with lower implementation or measurement cost;
8. gains vanish under fresh tasks, bridge trials, replay, transfer, shadow baselines, or holdout rotation;
9. resource-vector reporting shows hidden transfer to another scarce resource;
10. material Layer 0 fields are missing or estimated beyond declared tolerance for the claimed strata;
11. append-only telemetry is broken or corrections are used without dependency links;
12. service ledger denominators exclude incident or unavailable exposure without charging it;

13. service units, quality tiers, sharing rules, or preemption rules are inconsistent across capacity and load ledgers;
14. service-rate lower bounds are uncalibrated, unauditable, or inconsistent with completed service;
15. steady, burst, or age-envelope capacity is violated for a material service channel;
16. service-load contracts repeatedly violate observed loads or remain uncalibrated for material action types;
17. validation, audit, replay, maintenance, or registry queues exceed declared envelopes;
18. WIP aging, deferred validation, expired work, or carryover liability explains the apparent gain;
19. scalar gains depend on hard stops, leakage, permission failures, uncharged incidents, or rollbacks;
20. evaluator state is below the floor required by the claim level or bridge audit scope;
21. the trusted-base audit log, trust root, registry checker, or revocation cascade fails for a material dependency;
22. the dependency graph places a hard stop, trusted-base revocation, or corrupted replay record inside the claim scope, or dependency-graph coverage is below the declared threshold;
23. a compressed dependency boundary lacks a valid boundary certificate;
24. frontier source admission, quota freeze, blinding, deduplication, or leakage screening fails for material frontier strata;
25. frontier sampling frames change task mass, hardness strata, or weights after outcomes without a downgrade;
26. frontier task mass fails the default, minimum-detectable-effect, cluster, or coverage rule;
27. logged-propensity, positivity, bounded-outcome, interference, horizon, estimator admissibility, or adaptive-sampling records are absent for labels used in strong estimates;
28. sentinel or full-audit failures make the numerator uncertified;
29. evaluator sentinel rotation, canary exposure budget, leakage probes, or drift tests fail;
30. high-pressure stations repeatedly fail bounded relief tests;
31. admitted library entries exceed budget, pass promotion deadlines without evidence, or remain beyond time-to-live without reserved maintenance;
32. promoted library return is non-positive under operational, causal, or maintenance attribution;
33. promotion attribution cannot separate a claimed entry from co-used entries and the claim requires independent reinvestment credit;
34. reinvestment claims count descriptive lineage edges as causal capital;

35. negative lineage, maintenance drag, cannibalization, or security burden is omitted from reinvestment accounting;
36. library ablations show harmful interference;
37. claimed reinvestment counts experimental, merely admitted, downgraded, or quarantined entries as capital;
38. frozen periods or frozen outputs are counted toward stronger-claim duration or numerator without a bridge at restart;
39. daily instrumentation burden exceeds the ceiling and is not charged;
40. open-ended discoveries fail novelty, priority, utility, or replication checks;
41. gains are explained by unbridged substrate, service, baseline, task-stream, evaluator, team, or resource changes.

19 Limitations

LOSCR is strongest where runs, resources, outputs, tests, library entries, review outcomes, and incidents can be logged: software engineering, theorem proving, ML research engineering, simulation, evaluator construction, and internal tools. It is weaker where feedback is delayed for months, value is primarily institutional, or external-world experiments dominate cost.

The theory does not remove judgment. Task portfolios, ledger schemas, resource prices, tolerances, gate rules, exploration budgets, audit schedules, service envelopes, and dramatic thresholds remain choices. Its contribution is to make those choices explicit, lightweight at the base, dynamically checkable, and auditable at stronger claim levels.

The online kernel bounds and organizes intervention. It does not guarantee that intervention discovers the right action. Strong acceleration still requires competent task selection, useful library design, reliable evaluation, enough non-controller service capacity, and willingness to retire interventions that stop helping.

20 Conclusion

A scoped AI R&D loop supports a dramatic acceleration claim only when it turns AI-generated and human-generated work into verified reusable progress faster than it accumulates hidden resource costs, evaluator defects, unsafe artifacts, unfinished work, service overload, maintenance burden, and misleading claims. LOSCR provides a practical, model-independent control theory for that purpose. It starts with minimal edge telemetry, seals machine-readable claim contracts, checks the strongest currently supported claim level, adds calibrated service control when claims create validation or maintenance load, adds certified replay only when claiming reusable capital, reconstructs online state, reserves scarce service, controls intervention classes, audits evaluators in proportion to exposure, preserves WIP and service-age accounting, keeps baselines bridged, maintains a trusted base registry, admits certified library entries without value circularity, promotes them only by measured marginal value, invalidates claims through typed dependency graphs when incidents occur, and escalates claims through an evidence ladder. The operational instruction is compact: observe cheaply, update state continuously, reserve scarce service, intervene reversibly, certify what will be reused, measure before promotion, maintain what remains valuable, retire what stops helping, and reinvest only verified gains.

References

- [1] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv:2001.08361*, 2020. <https://arxiv.org/abs/2001.08361>
- [2] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models. *Advances in Neural Information Processing Systems*, 2022. <https://arxiv.org/abs/2203.15556>
- [3] Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. Mathematical discoveries from program search with large language models. *Nature*, 625:468–475, 2024. <https://www.nature.com/articles/s41586-023-06924-6>
- [4] Alexander Novikov, Ngan Vu, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. AlphaEvolve: A coding agent for scientific and algorithmic discovery. *arXiv:2506.13131*, 2025. <https://arxiv.org/abs/2506.13131>
- [5] Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The AI Scientist: Towards fully automated open-ended scientific discovery. *arXiv:2408.06292*, 2024. <https://arxiv.org/abs/2408.06292>
- [6] Chris Lu, Cong Lu, Robert Tjarko Lange, Yutaro Yamada, Shengran Hu, Jakob Foerster, David Ha, and Jeff Clune. Towards end-to-end automation of AI research. *Nature*, 651:914–919, 2026. <https://www.nature.com/articles/s41586-026-10265-5>
- [7] Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin Godel Machine: Open-ended evolution of self-improving agents. *arXiv:2505.22954*, 2025. <https://arxiv.org/abs/2505.22954>
- [8] Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Lilian Weng, and Aleksander Madry. MLE-bench: Evaluating machine learning agents on machine learning engineering. *arXiv:2410.07095*, 2024. <https://arxiv.org/abs/2410.07095>
- [9] Hjalmar Wijk, Tao Lin, Joel Becker, Sami Jawhar, Neev Parikh, Thomas Broadley, Lawrence Chan, Michael Chen, Josh Clymer, Jai Dhyani, Elena Elicheva, Katharyn Garcia, Brian Goodrich, Nikola Jurkovic, Megan Kinniment, Aron Lajko, Seraphina Nix, Lucas Sato, William Saunders, Maksym Taran, Ben West, and Elizabeth Barnes. RE-Bench: Evaluating frontier AI R&D capabilities of language model agents against human experts. *arXiv:2411.15114*, 2024. <https://arxiv.org/abs/2411.15114>

- [10] Giulio Starace, Oliver Jaffe, Dane Sherburn, James Aung, Jun Shern Chan, Leon Maksin, Rachel Dias, Evan Mays, Benjamin Kinsella, Wyatt Thompson, Johannes Heidecke, Mia Glaese, and Tejal Patwardhan. PaperBench: Evaluating AI’s ability to replicate AI research. *arXiv:2504.01848*, 2025. <https://arxiv.org/abs/2504.01848>
- [11] Hui Chen, Miao Xiong, Yujie Lu, Wei Han, Ailin Deng, Yufei He, Jiaying Wu, Yibo Li, Yue Liu, and Bryan Hooi. MLR-Bench: Evaluating AI agents on open-ended machine learning research. *arXiv:2505.19955*, 2025. <https://arxiv.org/abs/2505.19955>
- [12] Alisia Lupidi, Bhavul Gauri, Thomas Simon Foster, Bassel Al Omari, Despoina Magka, Alberto Pepe, Alexis Audran-Reiss, Muna Aghamelu, Nicolas Baldwin, Lucia Cipolina-Kun, Jean-Christophe Gagnon-Audet, Chee Hau Leow, Sandra Lefdal, Hossam Mossalam, Abhinav Moudgil, Saba Nazir, Emanuel Tewolde, Isabel Urrego, Jordi Armengol Estape, Amar Budhiraja, Gaurav Chaurasia, Abhishek Charnalia, Derek Dunfield, Karen Hambardzumyan, Daniel Izcovich, Martin Josifoski, Ishita Mediratta, Kelvin Niu, Parth Pathak, Michael Shvartsman, Edan Toledo, Anton Protopopov, Roberta Raileanu, Alexander Miller, Tatiana Shavrina, Jakob Foerster, and Yoram Bachrach. AIRS-Bench: A suite of tasks for frontier AI research science agents. *arXiv:2602.06855*, 2026. <https://arxiv.org/abs/2602.06855>
- [13] Jialong Chen, Xander Xu, Hu Wei, Chuan Chen, and Bing Zhao. SWE-CI: Evaluating agent capabilities in maintaining codebases via continuous integration. *arXiv:2603.03823*, 2026. <https://arxiv.org/abs/2603.03823>
- [14] Thomas Kwa, Ben West, Joel Becker, Amy Deng, Katharyn Garcia, Max Hasin, Sami Jawhar, Megan Kinniment, Nate Rush, Sydney Von Arx, Ryan Bloom, Thomas Broadley, Haoxing Du, Brian Goodrich, Nikola Jurkovic, Luke Harold Miles, Seraphina Nix, Tao Lin, Neev Parikh, David Rein, Lucas Jun Koba Sato, Hjalmar Wijk, Daniel M. Ziegler, Elizabeth Barnes, and Lawrence Chan. Measuring AI ability to complete long tasks. *arXiv:2503.14499*, 2025. <https://arxiv.org/abs/2503.14499>
- [15] Tingxu Han, Yi Zhang, Wei Song, Chunrong Fang, Zhenyu Chen, Youcheng Sun, and Lijie Hu. SWE-Skills-Bench: Do agent skills actually help in real-world software engineering? *arXiv:2603.15401*, 2026. <https://arxiv.org/abs/2603.15401>
- [16] OpenAI. Why SWE-bench Verified no longer measures frontier coding capabilities. Technical report, 2026. [OpenAI report page](#).
- [17] OpenAI. Measuring the performance of our models on real-world tasks. Technical report, 2025. [GDPval report page](#).
- [18] Jerzy Neyman. On the application of probability theory to agricultural experiments: Essay on principles. 1923. English translation in *Statistical Science*, 5(4):465–480, 1990.
- [19] Daniel G. Horvitz and Donovan J. Thompson. A generalization of sampling without replacement from a finite universe. *Journal of the American Statistical Association*, 47(260):663–685, 1952.
- [20] Donald B. Rubin. Estimating causal effects of treatments in randomized and nonrandomized studies. *Journal of Educational Psychology*, 66(5):688–701, 1974.

- [21] Miroslav Dudik, John Langford, and Lihong Li. Doubly robust policy evaluation and learning. *Proceedings of the 28th International Conference on Machine Learning*, 2011. <https://arxiv.org/abs/1103.4601>
- [22] Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society: Series B*, 57(1):289–300, 1995.
- [23] Steven R. Howard, Aaditya Ramdas, Jon McAuliffe, and Jasjeet Sekhon. Time-uniform, nonparametric, nonasymptotic confidence sequences. *The Annals of Statistics*, 49(2):1055–1080, 2021.
- [24] Ron Kohavi, Diane Tang, and Ya Xu. *Trustworthy Online Controlled Experiments: A Practical Guide to A/B Testing*. Cambridge University Press, 2020.
- [25] John D. C. Little. A proof for the queueing formula: $L = \lambda W$. *Operations Research*, 9(3):383–387, 1961.
- [26] Eliyahu M. Goldratt and Jeff Cox. *The Goal: A Process of Ongoing Improvement*. North River Press, 1984.
- [27] Abbas Kazerouni, Mohammad Ghavamzadeh, Yasin Abbasi-Yadkori, and Benjamin Van Roy. Conservative contextual linear bandits. *Advances in Neural Information Processing Systems*, 30, 2017. <https://papers.nips.cc/paper/6980-conservative-contextual-linear-bandits>
- [28] D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. Hidden technical debt in machine learning systems. *Advances in Neural Information Processing Systems*, 28, 2015. <https://papers.nips.cc/paper/5656-hidden-technical-debt-in-machine-learning-systems>
- [29] Charles A. E. Goodhart. Problems of monetary management: The U.K. experience. *Papers in Monetary Economics*, Reserve Bank of Australia, 1975.
- [30] Donald T. Campbell. Assessing the impact of planned social change. *Evaluation and Program Planning*, 2(1):67–90, 1979.
- [31] Marilyn Strathern. “Improving ratings”: Audit in the British university system. *European Review*, 5(3):305–321, 1997.