STABLEEDGE: CERTIFYING EARLY WORKFLOW ROUTES FROM STRUCTURED AGENT STREAMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Agent workflows increasingly branch on structured model output while bytes are still arriving. At a partial JSON prefix, a route chosen by an ordinary predicate can change under a valid continuation, producing wrong or duplicate workflow dispatch. We formalize safe early admission as *completion-invariant routing*: a prefix may be acted upon only if every schema-valid completion selects the same ordered route. STABLEEDGE decides this condition by computing a conservative route image over a bounded policy profile with required route discriminators, certifying singleton images, and binding each certificate to a grammar-attested stream contract, a digest-bound commit envelope, a durable ledger, and an idempotent receiver. On 1,616 matched final-valid traces from BFCL v4 and AgentDojo, STABLEEDGE makes exactly one correct admission per trace with zero observed wrong or duplicate dispatches, reaching admission a median 1,340 ms of recorded generation time before the final message and certifying 19.9 percentage points more traces early than a safe closed-field policy. Byte-prefix law audits, trust negative controls, crash and concurrency conditions, a structural grid, and three native runtimes locate when this gain is available and when it collapses.

1 INTRODUCTION

Agent systems routinely translate model-generated structure into a control-flow decision: call a tool, hand off to another agent, or answer the user directly (Schick et al., 2023; Kim et al., 2024; Du et al., 2024). Two now-standard ingredients make this decision look available before generation finishes. Constrained decoding forces the response to conform to a grammar or schema (Scholak et al., 2021; Ugare et al., 2025; Dong et al., 2025; Park et al., 2025; Ye et al., 2025), and streaming interfaces expose the response incrementally as bytes or tokens are produced. When the field that determines the branch precedes a long argument object or a long response body, waiting for the final byte before dispatching the branch appears unnecessarily conservative.

Acting on a partial response, however, opens a semantic gap. An unfinished serialization can still admit several different branches, even when a predicate evaluated on the bytes seen so far returns exactly one answer. This is the expected behavior of any rule that reads a not-yet-observed field as absent, or that accepts a value because it has been stable for a fixed number of chunks. Figure 1 shows the problem on a house-price forecasting request served with two regression tools, a direct-response route, and a structured output contract. At byte 2, before the route field has appeared, eager prefix routing dispatches the direct-response branch, which is wrong here. At byte 20 the open route string reads `tool_0`, so every completion permitted by the contract selects `linear_regression` even though the string, the enclosing object, and the tool arguments are unfinished. Waiting for the field to close acts at byte 23, full buffering only after the 301-byte message completes.

The relevant question is therefore not whether a prefix looks like a particular route, nor how long it has looked that way, but whether every schema-valid completion of that prefix selects the same ordered branch. We call a prefix with this property *completion-invariant* for a given route program, and we take completion invariance as the admission condition for early workflow dispatch. The condition is well behaved in principle. Observing more bytes can only shrink the set of valid completions, so the set of branches those completions select shrinks monotonically, and once that set is a singleton no valid continuation can change it. Every prefix from that point on is also completion-invariant, so the first such prefix is the earliest byte at which dispatch is safe.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

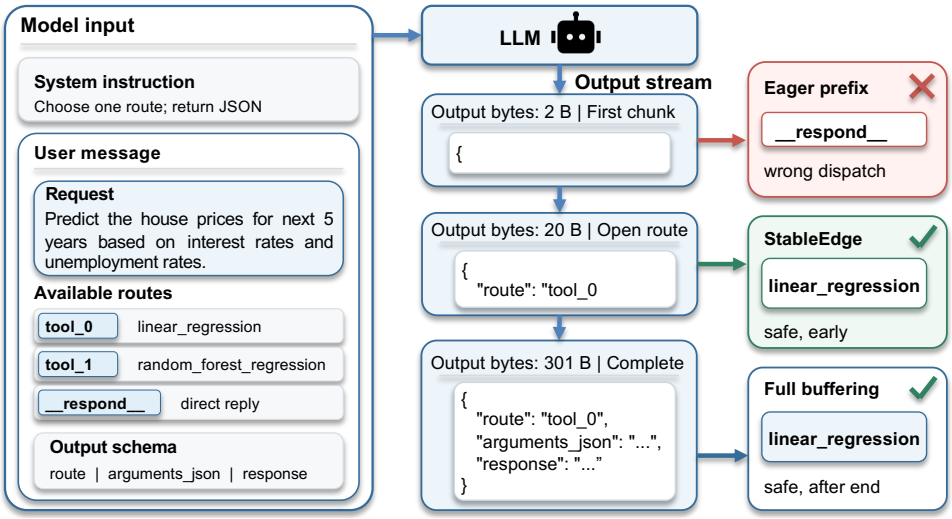

Figure 1: The model input specifies a house-price request, route choices, and a JSON output schema; three cards show cumulative output from the same LLM. The snippets abbreviate serialized bytes, while labels give actual offsets. At 2 B, eager prefix routing wrongly dispatches `__respond__` before seeing the route. At 20 B, the open `tool_0` value fixes `linear_regression` across valid completions, so StableEdge admits it before the 301 B message ends; full buffering waits until completion.

Turning that observation into a system, however, is difficult in three distinct ways. *First, the condition cannot be decided by enumeration.* A prefix has unboundedly many valid completions, so a practical procedure must over-approximate the reachable branches, erring only toward waiting longer and never toward a branch some completion contradicts; ordered first-match programs add that a later predicate cannot justify its branch while an earlier one may already hold. *Second, the premises live outside the route logic.* Telling a not-yet-observed field from one absent in every completion needs a strict unique-key parser state that exposes open versus closed values plus the schema's requiredness; which completions remain reachable is a property of the producer's decoding language; and an optional finite-domain discriminator admits a prefix whose branch set looks like a singleton while the empty completion takes the default. *Third, an early decision is an early externalization.* Transports retry and reorder, workers race over one logical turn, and a crash can fall between reserving a decision, delivering its effect, and recording its acknowledgment. A prefix-local decision without a durable admission path is only a prediction, while a durable protocol without completion reasoning is an efficient way to commit the wrong branch.

We present StableEdge, which decides completion invariance and then commits the resulting decision, addressing these difficulties in turn. Against the first, a bounded conservative compiler computes an *abstract route image* from the schema, the ordered predicates, and the prefix state: a superset of the branches reachable under any valid completion, obtained by enumerating finite domains, filtering candidates against the decoded open-string prefix, and keeping both truth values wherever a test is unresolved. A certificate is emitted exactly when this image is a singleton, so imprecision postpones admission but cannot authorize a different route. Against the second, a strict incremental parser reconstructs the prefix under unique-key UTF-8 and JSON rules, tracks whether each field is closed or still open, and never reads an unobserved field as absent, while the host pins per turn an attested grammar identity and a route-policy profile requiring every finite-domain discriminator. Against the third, a canonical digest binds each certificate to the approved schema, predicates, contract flags, and turn; a durable ledger admits one winner per turn key; and an idempotent sink absorbs replay. When the stream finishes, strict validation and exact route evaluation re-check the admitted branch, recording a violated premise as a certificate violation rather than dispatching a second branch.

We evaluate on 1,616 matched final-valid traces from BFCL v4 and AgentDojo, replaying each trace's bytes, chunk boundaries, and timestamps unchanged across five admission policies. StableEdge

makes exactly one correct admission per trace, reaches it a median of 1,340 ms of recorded generation time before the final message, and certifies 19.9 percentage points more traces early than a safe closed-field policy. A structural grid locates the frontier where the discriminator arrives too late for any early certificate, and six frozen boundary audits exercise the compiler, attestation, transport, crash recovery, and three native runtimes.

Our contributions can be summarized as follows.

- We formalize early workflow dispatch as completion-invariant routing over the language of schema-valid serializations, and give a sufficient admission condition in the form of a singleton route image together with the containment, prefix-fidelity, and attestation premises it requires.
- We design and implement STABLEEDGE, comprising a strict prefix parser, a bounded conservative compiler that computes ordered route images for a declared schema and predicate language, and an admission protocol that binds each certificate to an attested grammar contract, a digest-identified commit envelope, a durable ledger, and an idempotent effect sink.
- We develop a matched-replay protocol that pairs admission policies on identical byte, chunk, and timestamp histories, measuring route integrity, admission uniqueness, and recovery completeness in six frozen boundary audits.

2 RELATED WORK

Structured generation and reasoning about completions. Constrained decoders such as PICARD, SynCode, DOMINO, Grammar-Aligned Decoding, and XGrammar keep generation inside a syntactic or schema language (Scholak et al., 2021; Ugare et al., 2025; Beurer-Kellner et al., 2024; Park et al., 2024; Dong et al., 2025; Park et al., 2025; Ye et al., 2025), and LMQL expresses such constraints inside a query language (Beurer-Kellner et al., 2023). TruncProof asks whether a prefix remains completable within a token budget (Kato & Tarashima, 2026). These systems establish what the producer *can* emit; STABLEEDGE consumes that guarantee to answer a different, workflow-level question: whether all reachable completions agree on one ordered route. Our transfer functions follow standard conservative dataflow and abstract-interpretation practice (Kildall, 1973; Cousot & Cousot, 1977), and the strict serialization boundary follows language-oriented parsing work (Bangert & Zeldovich, 2014; Ramananandro et al., 2019). Optimally streaming regular-expression parsing and Kleenex formalize emitting output as soon as further input cannot change it (Grathwohl et al., 2014; 2016); we transplant that early-output principle from parser output to the ordered route image of an attested agent message, and then make the decision durable.

Streaming agent systems and reliable effects. Token-native orchestration targets lower scheduling latency and bounded backpressure (Zhang, 2026), and streaming guardrails forecast safety from partial output (Kavumba et al., 2026); both act before completion but do not certify branch invariance. Workflow compilers and agents optimize graph configuration and compliance (Li et al., 2026; Shi et al., 2025), while tool-use systems and benchmarks study function selection and agent execution over structured tasks (Patil et al., 2023; 2025; Debenedetti et al., 2024; Yao et al., 2023). Stream-processing systems motivate persistent state and replay (Akidau et al., 2013; 2015; Carbone et al., 2015); distributed snapshots capture consistent global state (Chandy & Lamport, 1985). Sagas and distributed-transaction analysis motivate explicit boundaries around external effects (Garcia-Molina & Salem, 1987; Helland, 2007), while coordination avoidance depends on application invariants (Bailis et al., 2014). Prompt-injection benchmarks and task-alignment defenses study a complementary part of the execution boundary (Zhan et al., 2024; Jia et al., 2025). STABLEEDGE composes these ingredients when unfinished model output becomes a workflow branch.

3 PROBLEM FORMULATION

3.1 FROM A BYTE PREFIX TO A ROUTE SET

Let S be a supported JSON Schema[1] and let $\mathcal{L}(S)$ be the language of byte strings that are strict UTF-8/JSON encodings of messages satisfying S. Working with serialized strings rather than abstract

[1] JSON Schema Draft 2020-12:
`https://json-schema.org/draft/2020-12/json-schema-core`;
`https://json-schema.org/draft/2020-12/json-schema-validation`.

values is deliberate: it accommodates whitespace, key order, and escape choices, all of which a producer controls. For an observed prefix p,

$$\mathcal{C}_S(p) = \{x \in \mathcal{L}(S) : p \preceq x\} \tag{1}$$

is its set of valid completions. Let $B = (b_1 \mapsto r_1, \ldots, b_k \mapsto r_k, r_d)$ be an ordered route program whose exact evaluator $R_B(m)$ takes the first true predicate, or the default r_d if none is true. The *exact route image* is

$$\mathcal{I}_{S,B}(p) = \{R_B(\mathrm{parse}(x)) : x \in \mathcal{C}_S(p)\}. \tag{2}$$

A prefix is *completion-invariant* iff $|\mathcal{I}_{S,B}(p)| = 1$; an empty completion set is an error state, not a certificate. Because $p \preceq q$ implies $\mathcal{C}_S(q) \subseteq \mathcal{C}_S(p)$, we have $\mathcal{I}_{S,B}(q) \subseteq \mathcal{I}_{S,B}(p)$: the image only shrinks as evidence arrives, and once it is a singleton no valid continuation can change its member. Safety of early admission is thus a monotone, prefix-local property, which is what makes it certifiable.

The Figure 1 stream illustrates three frontiers. At the opening brace, the required `route` is unseen and several routes remain possible. Inside the open `tool_0` value, only one enum member remains; field closure and final-message completion occur later. Section 5.2 measures these gaps.

3.2 Certificate and admission rule

StableEdge avoids enumerating completions by admitting from a conservative route image:

$$\mathrm{admit}(p, r) \iff \mathrm{approvedProfile}(S, B, p) \wedge \widehat{\mathcal{I}}_{S,B}(p) = \{r\}. \tag{3}$$

Approval bundles three premises: a faithful strict valid-prefix state, the host's attestation that the observed bytes remain reachable under S, and a policy profile in which every finite-domain path referenced by an ordered predicate, together with its enclosing object path, is guaranteed present by the schema. The host pins one approved profile and grammar identity per turn and passes them to the reference router through a trusted stream contract.

Boundary	Input or state	Required property
Generation	Stream bytes, chunking	Faithful strict reconstruction; grammar attestation ties reachable completions to the approved schema
Policy	Schema, ordered predicates	Host-approved authority and canonical digest binding
Admission	Workflow/turn key, certificate	One durable winner and visible recovery state
Effect	Commit ID and selected route	Idempotent receiver or transactional outbox

Table 1: Obligations are explicit per boundary. The certificate establishes route consistency; storage and receiver contracts establish admission and effect behavior.

Proposition (conditional certificate). Assume that for every $x \in \mathcal{C}_S(p)$ the parser state describes the bytes of p faithfully and $R_B(\mathrm{parse}(x)) \in \widehat{\mathcal{I}}_{S,B}(p)$. If $\widehat{\mathcal{I}}_{S,B}(p) = \{r\}$, then every such x routes to r.

Every exact route belongs to the conservative image, so a singleton contains only r. The premises separate byte fidelity, image containment under the supported policy (Section 4.2), and producer attestation; durable admission governs the subsequent effect (Section 4.3). Final-answer JSON validity alone establishes none of these premises. An imprecise non-singleton image delays admission without changing its safety.

Chunk independence. For a fixed reconstructed prefix, the certificate is independent of chunk count and pauses. Chunking still determines which prefixes are observed: coalescing can skip a singleton that finer chunks expose, while debounce can fire solely because chunks are split differently. We therefore pair methods on identical recorded boundaries and timestamps.

3.3 Threat model and correctness

The stream producer controls bytes, key order, chunk boundaries, and pauses. Transports may retry, reorder within a declared window, or present conflicting payloads for one sequence number. Concurrent workers race to admit a logical turn, and processes can crash between reservation, effect delivery, and acknowledgment. The trusted policy authority specifies S, B, and the grammar identity; the host-controlled contract binding, admission service, and durable store enforce them (Table 1).

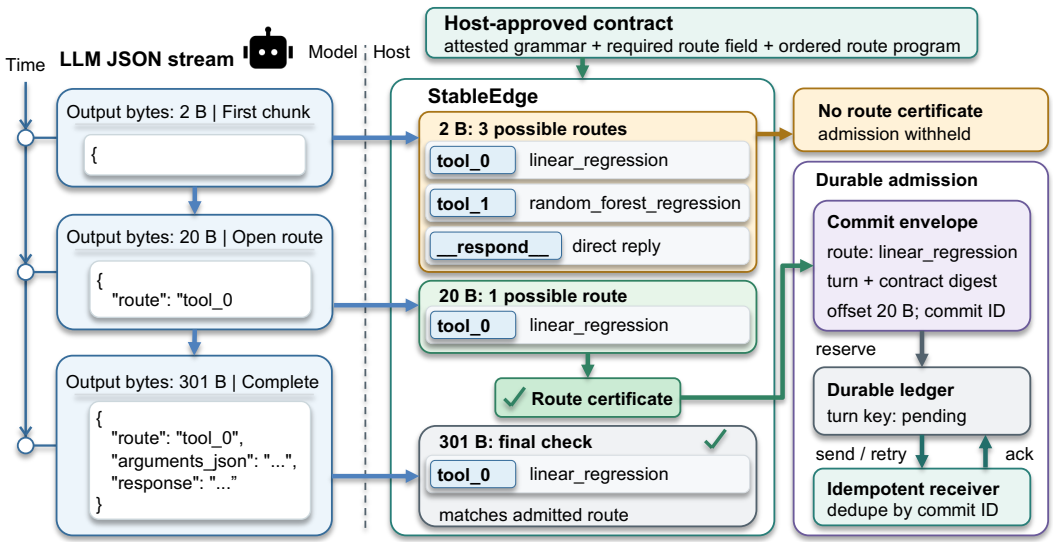

Figure 2: StableEdge processing the Figure 1 JSON stream. Blue cards show cumulative 2, 20, and 301 B prefixes; displayed snippets elide whitespace and later fields. The dashed line marks the model/host boundary, and the host-approved contract supplies an attested grammar, required route field, and ordered route program. At 2 B three routes remain possible, so admission waits. At 20 B the singleton image certifies `linear_regression`; its digest-bound envelope is reserved in the durable ledger before delivery to an idempotent receiver. Retries reuse the commit ID, and the final check matches the admitted route.

A turn's admission is *correct* when its route equals the exact route of any valid completed message, its logical key has exactly one durable winner, the certificate is bound to the approved workflow, turn, schema, grammar, and ordered predicates, and recovery retains or completes a pending reservation. The guarantee is about the route: arguments and user-visible content may still be unfinished when the branch is known, so applications requiring complete arguments can hold that later effect while using the earlier decision for safe preparation. This definition isolates three failure modes that we measure separately rather than folding into one accuracy number. *Route integrity* fails as soon as any emitted branch differs from the exact final branch, even if corrected later. *Admission uniqueness* fails when retries or repeated prefix evaluations create a second logical admission for the same turn. *Recovery completeness* fails when a reserved turn disappears, stays pending past the tested recovery schedule, or yields more than one receiver effect. The first is established by the completion argument; the other two depend on ledger and receiver contracts. Applications also specify cancellation and argument-dependent effect policies for streams that truncate after route admission.

4 StableEdge

Figure 2 shows how StableEdge uses a trusted contract to certify a route from a streamed JSON response, admit it durably, and validate it at completion.

4.1 Strict prefix state and supported policy

The parser accepts strict UTF-8 and JSON, rejects duplicate object keys, and records whether each field is closed or is an open string prefix. Crucially, it never treats a field absent *so far* as absent from every completion, a distinction that an ordinary JSON parser cannot reliably expose before the document closes. The policy compiler accepts a bounded schema subset (objects, required fields, additional-property control, arrays, finite `enum`/`const`, numeric limits, string/array length limits) and predicates over presence, equality/inequality, membership, string containment and prefix, numeric comparison, and Boolean composition. Unsupported syntax receives an explicit diagnostic rather than a silent fallback. The reference contract bounds schema nodes (4,096), properties per object (512), enum values (256), routes (64), JSON depth (32), string length (65,536 characters), and message bytes (1 MiB); a cap violation stops early admission, after which an application may still proceed through strict final validation. Appendix C details the language and bounds.

4.2 A CONSERVATIVE ORDERED ROUTE IMAGE

For each predicate path, the abstract evaluator combines prefix state with schema constraints. A closed value gets exact predicate semantics. For `const`, `enum`, and Boolean domains it evaluates every feasible member, and an open string filters those candidates by its decoded prefix. For unrestricted open strings some predicates are already decided (an equality target incompatible with the observed prefix is false, and a `contains` target already observed is true), while otherwise both truth values remain possible. An open numeric token retains both truth values unless a finite domain resolves it; comparisons become exact when the value closes. Boolean connectives take pointwise images of child possibilities.

Let $T_i(p) \subseteq \{\mathsf{T}, \mathsf{F}\}$ be the possible truth values of predicate b_i. The ordered pass constructs

$$\widehat{\mathcal{I}}_{S,B}(p) = \{r_i \mid \mathsf{T} \in T_i(p),\ \mathsf{F} \in T_j(p)\ (\forall j < i)\} \cup \{r_d \mid \mathsf{F} \in T_j(p)\ (\forall j \leq k)\}. \tag{4}$$

A possibly-true later predicate cannot justify its route when an earlier predicate is definitely true, and the default stays possible while every predicate may be false. The pass keeps a one-bit "all earlier predicates can be false" flag; it does not model joint predicate assignments, so it may retain a route whose joint assignment is infeasible. This widening is in the safe direction and preserves Equation 3.

Why the abstraction contains the concrete image. Under the approved required-path profile, a finite-domain predicate may enumerate the schema's values without omitting a completion in which the field is absent. Open-string filtering removes a member only when the observed decoded characters exclude it, and no valid completion can restore an excluded member. For unrestricted strings, a prefix-incompatible equality can never become true and an already-observed substring can never cease to be contained. Other unresolved atomic tests keep both truth values. Closed fields are exact because unique-key parsing forbids later replacement. Negation and Boolean composition preserve containment by taking images of all child possibilities. Finally, if a concrete completion takes branch i, its predicate is true and all earlier ones false, so Equation 4 includes r_i; the same argument covers the default. This transfer-function argument establishes image containment for the stated profile; finite replay checks the implementation along generated byte histories (Section 5.2.4). In the Figure 1 trace, the open value `tool_0` is already the only compatible enum member, so the first branch is selected before the closing quote; with correlated predicates a non-singleton image can instead persist past the exact frontier, costing latency but not correctness.

4.3 DIGEST BINDING AND DURABLE ADMISSION

Early execution is enabled only when the host's trusted decoder integration supplies an approved grammar backend identity, unique-key parsing and resource bounds are active, and the host has selected a route-policy profile satisfying the required-path precondition. Because the general schema normalizer and the router are separate components, generic schema acceptance alone does not establish the route-profile premise, a distinction we measure rather than assume (Section 5.2.4) and justify with a counterexample on an optional enum discriminator in Appendix C.

A canonical digest covers the schema, ordered route program, and contract flags, and the commit envelope carries that digest with workflow and turn identifiers, the route, certificate, and prefix offset. A different policy changes the digest and hence the commit identity; within one pinned contract the ledger rejects a conflicting route for the same key, and the sequenced transport deduplicates identical retries, reconstructs permitted reordering, and rejects conflicting sequence payloads. At run time each accepted prefix updates the image: an ambiguous image leaves the turn collecting, a singleton yields an envelope tagged with the observed byte offset. The adapter reserves that envelope under the logical turn key *before* emitting the native workflow edge; an identical route and commit identifier returns the prior admission. Streaming continues after admission, and at completion strict validation plus exact routing check consistency with the certificate; a mismatch is recorded as a certificate violation rather than compensated by a second branch.

The ledger derives a stable commit identifier from workflow, turn, and contract digest. Its SQLite reference implementation uses WAL durability, unique logical keys, and reserve/acknowledge/pending states, with recovery replaying pending records under the same commit identifier. The effect/acknowledgment gap is the crux: if a process crashes after the receiver applies an effect but before acknowledgment is recorded, the ledger cannot infer from its own state whether the effect occurred, so it must retry; stable commit identity is what makes that retry recognizable to an idempotent receiver

or transactional outbox. The ledger therefore provides one durable *admission*; the receiver provides one externally visible *effect*. Appendix F gives the five states, transitions, and tested crash schedules.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Traces. The frozen matched suite contains 1,456 BFCL v4 traces (Patil et al., 2025) and 160 AgentDojo v1.2 traces (Debenedetti et al., 2024): 1,616 distinct final-valid trace instances. BFCL covers simple, multiple, parallel, and irrelevance cases; AgentDojo covers banking, Slack, travel, and workspace workflows as a security-relevant routing workload. Two fixed Qwen revisions generated structured outputs through XGrammar at temperature zero and served only as *trace sources*. Appendix D records construction, revisions, hardware, selection, and the single retained token-cap-invalid output.

Compared policies. Each trace is replayed five times against the same schema, ordered route program, bytes, provider-token boundaries, timestamps, and final-message oracle, under five policies: (1) STABLEEDGE singleton certification; (2) full buffering with strict validation and exact routing; (3) a closed-field heuristic that waits until the dedicated route field closes; (4) three-chunk route debounce; (5) eager routing at every parseable prefix. Policies (4) and (5) are diagnostic controls for transient-prefix behavior. Statistics resample the 1,616 logical trace keys.

Metrics and inference. A wrong-route trace has at least one dispatch differing from the exact final route; wrong dispatches count all such emissions. Duplicate dispatches count emissions beyond the single logical admission. Early coverage is the fraction of traces whose first admission precedes stream completion. Byte and provider-token leads measure remaining serialization; *time lead* uses recorded generation timestamps at the decision and final byte and excludes downstream tool execution. Interval estimates use 10,000 paired bootstrap resamples over trace keys (Efron, 1979); zero-error rates use a one-sided exact binomial bound (Clopper & Pearson, 1934); paired safety and latency contrasts use exact McNemar and Wilcoxon tests with Holm correction (McNemar, 1947; Wilcoxon, 1945; Holm, 1979). Appendix D.2 gives full definitions.

5.2 RESULTS

5.2.1 MAIN COMPARISON

Table 2 separates route integrity from decision timing. STABLEEDGE, full buffering, and closed-field have no wrong or duplicate dispatches; STABLEEDGE certifies every trace early. Among these safe policies, the difference is *when* a route can be admitted. Under an independent-trace binomial model, the one-sided exact 95% upper bound for zero errors in 1,616 logical traces is 0.1852%.

Policy	Unsafe traces	Wrong disp.	Dup. disp.	Early	Lead (ms)
STABLEEDGE	0/1,616	0	0	100.000%	1,339.647
Full buffer	0/1,616	0	0	0.000%	0
Closed-field	0/1,616	0	0	80.136%	1,121.333
3-chunk debounce	1,295/1,616	1,295	0	100.000%	1,404.547
Eager prefix	1,616/1,616	10,297	78,120	100.000%	1,487.475

Table 2: Matched comparison on one replay. An unsafe trace has any wrong or duplicate dispatch. All five replays agree; the 1,616 logical traces define the sample size. Lead is the median recorded generation time remaining at first admission. Debounce and eager timing is diagnostic, since their emissions are unsafe.

Debounce and eager routing demonstrate that chunk stability and a parseable prefix do not establish route invariance (Table 2). Figure 1 shows the eager absence-as-default failure; paired safety tests appear in Appendix D.2.

Useful early certification. STABLEEDGE leads full buffering by a paired median 1,339.647 ms (95% bootstrap CI 1,306.574-1,366.046 ms). Against closed-field, it gains 19.864 percentage points of early coverage (95% CI 17.946-21.844) and a paired median 55.416 ms (95% CI 55.367-55.489). The

Category	n	Closed early	StableEdge lead	Paired gain
BFCL simple	560	559	1,224.364	66.051
BFCL multiple	280	280	1,378.501	22.747
BFCL parallel	280	252	1,569.538	71.356
BFCL irrelevance	336	77	1,440.374	1,291.279
AgentDojo banking	40	27	1,479.851	23.881
AgentDojo Slack	40	39	715.148	23.895
AgentDojo travel	40	39	744.351	24.113
AgentDojo workspace	40	22	1,587.170	24.231

Table 3: Main-comparison strata, one matched replay per distinct trace instance. StableEdge certifies early in every category; both policies have zero wrong and duplicate dispatches. Leads and paired gains are median milliseconds remaining to the final byte; the paired gain is the median of within-trace differences.

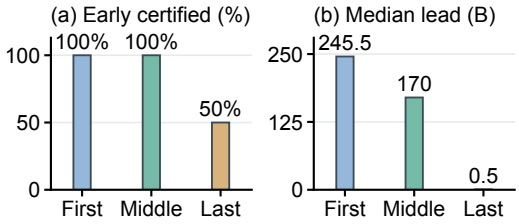

Figure 3: Structural frontier over 432 conditions. Panels show early-certified fraction and median byte lead. Each position pools output lengths, route widths, argument depths, and chunk policies; all final routes are correct.

Position	1-byte	Token proxy	Coalesced
First	48/48 (250.5)	48/48 (243.5)	48/48 (245.5)
Middle	48/48 (176.0)	48/48 (171.5)	48/48 (170.0)
Last	48/48 (2.0)	16/48 (0.0)	8/48 (0.0)

Table 4: Chunk observability by route-field position (48 conditions per cell). Cells show early certifications out of 48, with median byte lead across all 48 conditions in parentheses. All final routes are correct.

latter contrast isolates the benefit beyond waiting for the route string to close; both policies preserve route integrity. These recorded-time leads are branch-decision opportunities, with implementation cost examined in Section 5.2.5.

5.2.2 Workload Analysis

Table 3 shows StableEdge certifying every category early. Closed-field nearly matches it on BFCL simple/multiple and AgentDojo Slack/travel, but waits until completion for 259 of 336 BFCL irrelevance traces routed to direct response. The required discriminator can rule out every tool route while its value is open, allowing StableEdge to certify the default response. This early default decision explains the larger paired gain for BFCL irrelevance than for the other BFCL categories.

Across the workload, early certification extends into the lower tail. At the 10th percentile of the marginal lead, measured against the next-earliest safe policy on the same replayed history, StableEdge still saves 794 ms of recorded generation time and 76 bytes; Appendix D.2 gives the full distributions.

5.2.3 Structural Sensitivity

The structural grid crosses route-field position, output length, route width, encoded-argument depth, and three chunk policies in 432 conditions (Appendix H). All select the exact final route. Figure 3 shows that first and middle positions certify early in every condition, while the last position does so in only half, with a median lead of 0.5 bytes. This follows the completion-invariance condition: a late discriminator can leave route-changing completions possible almost until message end, so safe routing approaches full buffering.

Table 4 separates the semantic frontier from its observability under a chunked adapter. At the last position, one-byte replay exposes the narrow pre-completion singleton in all 48 conditions, while token-proxy and coalesced chunks expose it in only 16 and 8. The certificate condition is defined over reconstructed bytes, but observed early coverage depends on which prefixes the transport reveals; exact routing holds in every cell.

Audit (obligation)	Observed outcome
Byte-prefix law replay (image containment)	Across 4,000 messages and 553,274 prefixes, every certificate matches its final route, with one admission per case; 3,778/4,000 certify early.
Schema compilation (support frontier)	BFCL: 1,393/1,397 accepted; AgentDojo: 67/69 accepted; six explicit diagnostics and zero silent fallbacks.
Trust grid (attestation)	Across 6,000 cases: zero early dispatches on 3,000 untrusted streams; zero non-abstentions on 2,500 invalid untrusted streams; 2,500/2,500 premature commits when invalid streams are misattested.
Sequenced transport (prefix fidelity)	Across 2,000 cases, permitted reconstructions retain the oracle route and commit identity; zero failures; conflicting retries rejected.
Crash/concurrency (durable admission)	Across 1,500 turn-conditions, zero lost, duplicate, or pending effects after quiescent recovery at five crash points and 1, 8, or 64 contenders.
Native runtimes (portability)	Each of three paths has 1,200/1,200 single-correct STABLEEDGE trials; metrics and route multisets agree on 2,400 common trial keys.

Table 5: Boundary audits, one row per obligation in Table 1. The misattestation negative control supplies a false grammar premise, isolating the role of attestation.

5.2.4 SYSTEM VALIDATION

Table 5 summarizes the checks for each obligation in Table 1; per-condition matrices are in Appendices C-G. Together they test the route image, attested input history, transport reconstruction, durable admission, and native adapters.

Route-image audit. The byte-prefix audit exercises equality, string, numeric, negation, conjunction, and nested-disjunction routes. Its lower early-certification rate than in the tool-route suite is consistent with broader predicate shapes under a conservative image. Early admission additionally enforces the required-path profile; the compilation row measures schema acceptance.

Trust and transport. False attestation of invalid streams causes premature commits, showing why grammar identity matters. The transport suite covers UTF-8 and escape splits, key and chunk boundaries, retries, and bounded reordering; permitted reconstructions preserve route and commit identity, while conflicting retries are rejected. Resource-cap stresses fail closed (Appendix E).

Durable admission and recovery. The effect/acknowledgment gap is decisive: the ledger cannot tell whether an unacknowledged effect occurred, but recovery reuses its commit identifier and the idempotent receiver suppresses repetition. The zero-loss and zero-duplicate result in Table 5 depends on the reference WAL store and receiver contract; Appendix F gives each transition.

Portability. The admission contract runs through native event paths in Microsoft Agent Framework Python and .NET, and FIT Waterflow Java. The eager control repeats the same failure pattern in each, locating that behavior in the shared routing policy. Agreement across the tested paths supports portability of the adapter contract at the pinned revisions (Table 5; Appendix G).

5.2.5 EFFICIENCY

On the recorded x86-64 Linux/Python setup, 351,000 eight-byte router updates took 26 μs at p50 and 55 μs at p99. These router-only costs complement the recorded generation lead in Section 5.2.1, which measures time available for downstream preparation. Appendix E reports wider-route compilation, throughput, and RSS.

6 CONCLUSION

STABLEEDGE certifies a workflow route once all schema-valid completions of the observed prefix agree, then binds that decision to durable admission and an idempotent receiver. On 1,616 matched traces, it made one correct admission per trace with no observed wrong or duplicate dispatches and a median recorded-time lead of 1.34 s over full buffering. Structural sensitivity and boundary audits show how route-field position, chunk observability, attestation, and recovery shape this early-decision opportunity. The guarantee applies to grammar-attested streams under the supported required-path profile; single external effects additionally require an idempotent receiver or transactional outbox.

Limitation. The current implementation certifies routes over a bounded JSON Schema and ordered-predicate profile. Future work can extend the conservative route-image abstraction to richer constraints and connect certified routes to additional forms of downstream preparation.

AI USE STATEMENT

We have not used generative AI tools for any of the tasks that require disclosure, including proposing or refining hypotheses, developing theoretical models or conceptual frameworks, designing or providing feedback on research methodology or experiments, implementing methods, and interpreting results; remaining required-disclosure tasks, such as generating synthetic datasets, formulating or proving mathematical claims, cleaning and reformatting datasets, supporting qualitative and thematic analysis, and assisting with translation, are not applicable to this work. We used generative AI tools for two tasks with recommended disclosure: retrieval and discovery, namely searching for and identifying related work, and writing support, namely drafting and editing portions of the manuscript to improve readability. We have reviewed all AI-assisted work. All references surfaced by these tools were located and read in their original form by the authors before citation, and all AI-edited prose was checked line by line against the method and results as implemented. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

The evaluation replays local structured outputs through synthetic workflow executors and idempotent sinks, without invoking real banking, messaging, travel, or workspace effects. Runtime controls isolate routing-policy behavior in native adapters.

REPRODUCIBILITY STATEMENT

Sections 3 and 4, together with Section 5.1, specify the certificate assumptions, method, trace protocol, and metrics. Appendices C-H detail the compiler frontier and transfer cases, trace construction and paired statistics, trust/transport/resource controls, crash schedules, native runtime adapters and parity rules, and the structural and cap sensitivity protocols. The ZIP file uploaded as the Supplementary Material contains the implementation, baselines, trace construction, evaluation scripts, native adapters, and tests; its README pins public sources and run order.

REFERENCES

Tyler Akidau, Alex Balikov, Kaya Bekiroglu, Slava Chernyak, Josh Haberman, Reuven Lax, Sam McVeety, Daniel Mills, Paul Nordstrom, and Sam Whittle. MillWheel: Fault-tolerant stream processing at internet scale. *Proceedings of the VLDB Endowment*, 2013.

Tyler Akidau, Robert Bradshaw, Craig Chambers, Slava Chernyak, Rafael J. Fernández-Moctezuma, Reuven Lax, Sam McVeety, Daniel Mills, Frances Perry, Eric Schmidt, and Sam Whittle. The dataflow model: A practical approach to balancing correctness, latency, and cost in massive-scale, unbounded, out-of-order data processing. *Proceedings of the VLDB Endowment*, 2015.

Peter Bailis, Alan Fekete, Michael J. Franklin, Ali Ghodsi, Joseph M. Hellerstein, and Ion Stoica. Coordination avoidance in database systems. *Proceedings of the VLDB Endowment*, 2014.

Julian Bangert and Nickolai Zeldovich. Nail: A practical tool for parsing and generating data formats. In *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, 2014.

Luca Beurer-Kellner, Marc Fischer, and Martin Vechev. Prompting is programming: A query language for large language models. *Proceedings of the ACM on Programming Languages*, 2023.

Luca Beurer-Kellner, Marc Fischer, and Martin Vechev. Guiding LLMs the right way: Fast, non-invasive constrained generation. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Paris Carbone, Asterios Katsifodimos, Stephan Ewen, Volker Markl, Seif Haridi, and Kostas Tzoumas. Apache Flink™: Stream and batch processing in a single engine. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 2015.

K. Mani Chandy and Leslie Lamport. Distributed snapshots: Determining global states of distributed systems. *ACM Transactions on Computer Systems*, 1985.

C. J. Clopper and E. S. Pearson. The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika*, 1934.

Patrick Cousot and Radhia Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Conference Record of the Fourth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, 1977.

Edoardo Debenedetti, Jie Zhang, Mislav Balunović, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents. In *Advances in Neural Information Processing Systems*, 2024.

Yixin Dong, Charlie F. Ruan, Yaxing Cai, Ziyi Xu, Yilong Zhao, Ruihang Lai, and Tianqi Chen. XGrammar: Flexible and efficient structured generation engine for large language models. In *Proceedings of Machine Learning and Systems*, 2025.

Yu Du, Fangyun Wei, and Hongyang Zhang. AnyTool: Self-reflective, hierarchical agents for large-scale API calls. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Bradley Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 1979.

Hector Garcia-Molina and Kenneth Salem. Sagas. In *Proceedings of the 1987 ACM SIGMOD International Conference on Management of Data*, 1987.

Bjørn Bugge Grathwohl, Fritz Henglein, Ulrik Terp Rasmussen, Kristoffer Aalund Søholm, and Sebastian Paaske Tørholm. Kleenex: Compiling nondeterministic transducers to deterministic streaming transducers. In *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, 2016.

Niels Bjørn Bugge Grathwohl, Fritz Henglein, and Ulrik Terp Rasmussen. Optimally streaming greedy regular expression parsing. In *Theoretical Aspects of Computing – ICTAC 2014*, 2014.

Pat Helland. Life beyond distributed transactions: An apostate's opinion. In *Third Biennial Conference on Innovative Data Systems Research*, 2007.

Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 1979.

Feiran Jia, Tong Wu, Xin Qin, and Anna Squicciarini. The Task Shield: Enforcing task alignment to defend against indirect prompt injection in LLM agents. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, 2025.

Yoshio Kato and Shuhei Tarashima. TruncProof: A guardrail for LLM-based JSON generation under token-length constraints. *arXiv preprint arXiv:2605.13076*, 2026.

Pride Kavumba, Koki Wataoka, Huy H. Nguyen, Jiaxuan Li, and Masaya Ohagi. Predict, don't react: Value-based safety forecasting for LLM streaming. *arXiv preprint arXiv:2604.03962*, 2026.

Gary A. Kildall. A unified approach to global program optimization. In *Proceedings of the 1st Annual ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, 1973.

Sehoon Kim, Suhong Moon, Ryan Tabrizi, Nicholas Lee, Michael W. Mahoney, Kurt Keutzer, and Amir Gholami. An LLM compiler for parallel function calling. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Junyan Li, Zhang-Wei Hong, Maohao Shen, Yang Zhang, and Chuang Gan. FlowCompile: An optimizing compiler for structured LLM workflows. *arXiv preprint arXiv:2605.13647*, 2026.

Quinn McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 1947.

Kanghee Park, Jiayu Wang, Taylor Berg-Kirkpatrick, Nadia Polikarpova, and Loris D'Antoni. Grammar-aligned decoding. In *Advances in Neural Information Processing Systems*, 2024.

Kanghee Park, Timothy Zhou, and Loris D'Antoni. Flexible and efficient grammar-constrained decoding. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive APIs. *arXiv preprint arXiv:2305.15334*, 2023.

Shishir G. Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The Berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025.

Tahina Ramananandro, Antoine Delignat-Lavaud, Cédric Fournet, Nikhil Swamy, Tej Chajed, Nadim Kobeissi, and Jonathan Protzenko. EverParse: Verified secure zero-copy parsers for authenticated message formats. In *28th USENIX Security Symposium (USENIX Security 19)*, 2019.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, 2023.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021.

Yuchen Shi, Siqi Cai, Zihan Xu, Yuei Qin, Gang Li, Hang Shao, Jiawei Chen, Deqing Yang, Ke Li, and Xing Sun. FlowAgent: Achieving compliance and flexibility for workflow agents. *arXiv preprint arXiv:2502.14345*, 2025.

Shubham Ugare, Tarun Suresh, Hangoo Kang, Sasa Misailovic, and Gagandeep Singh. SynCode: LLM generation with grammar augmentation. *Transactions on Machine Learning Research*, 2025.

Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1945.

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

Haotian Ye, Himanshu Jain, Chong You, Ananda Theertha Suresh, Haowei Lin, James Zou, and Felix Yu. Efficient and asymptotically unbiased constrained decoding for large language models. In *Proceedings of the 28th International Conference on Artificial Intelligence and Statistics*, 2025.

Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. InjecAgent: Benchmarking indirect prompt injections in tool-integrated large language model agents. In *Findings of the Association for Computational Linguistics: ACL 2024*, 2024.

Qunhui Zhang. AiFlow: Token-native reactive orchestration with bounded backpressure for streaming LLM applications. *arXiv preprint arXiv:2608.00558*, 2026.

A Appendix Overview

- Section B discusses early certificate use, timing, and durable workflow admission.
- Section C details the route-image abstraction, supported language, caps, required-path profile, and compiler frontier.
- Section D records trace construction, statistical units, and paired inference.
- Section E gives the trust, transport, and resource-bound controls.
- Section F describes the retained crash/recovery schedules.
- Section G specifies the native runtime adapters and parity checks.
- Section H covers the structural grid, cap sensitivity, case selection, and provenance.

B Extended Discussion

What an early certificate enables. The certificate authorizes the workflow's route while arguments may continue streaming. A runtime can reserve the chosen edge, start its downstream executor, and prepare route-specific resources while continuing to stream and validate remaining fields; an application can defer argument-dependent or irreversible effects until the input contract for those effects is satisfied. The measured time lead is the interval available to overlap route-specific preparation with generation.

When the opportunity appears. The route-image condition reconciles two observations that initially seem in tension. Every final trace in the fixed tool-route suite certifies early, yet the broader generated predicate audit certifies $3{,}778/4{,}000$ early and the route-last structural stratum certifies $72/144$. The first suite uses required route discriminators that typically precede a longer remainder; the other settings include harder predicate shapes and near-terminal route evidence. The certificate remains a correctness condition throughout, while its usefulness depends on how much output remains once the image becomes singleton and whether a chunk boundary exposes that prefix. Table 3 shows the distinction matters most for default-response branches, where a field-closure rule often waits until the end even though the schema has already excluded every tool branch.

From certificate to workflow admission. An abstract route image is meaningful only relative to the stream language and policy that produced it. The required-path profile and host-pinned grammar identity make that language explicit; the envelope binds the choice to the same workflow turn and ordered predicates; the ledger and receiver maintain its identity through races and crash recovery. The trust negative control and persistence schedules exercise different links of that chain. Together they support early admission by an adapter implementing the contract, with one logical workflow decision through retries and crashes. The native integrations demonstrate the contract at three pinned execution paths, and the hardware-scoped microbenchmarks show that the checked router operations are small relative to the recorded generation lead in this setting.

C Certificate Semantics and Implementation Frontier

Ordered branch image. For each branch predicate b_i the abstract evaluator returns a set $T_i(p) \subseteq \{\mathsf{T}, \mathsf{F}\}$ of possible truth values. The ordered pass starts with a *previous branches can all be false* flag set to true; it includes r_i when the flag holds and $\mathsf{T} \in T_i(p)$, retains the flag only when $\mathsf{F} \in T_i(p)$, and includes r_d if the flag survives the final branch. A truth set may be wider than the concrete possibilities because Boolean children are composed without preserving cross-predicate correlations. If every atomic truth set contains the true value of each compatible completion, this composition and route pass contain the concrete route image, and the implication in Equation 3 follows from containment plus singletonhood.

Transfer cases. Table 6 specifies the implemented distinctions. The strict parser retains a closed document/field value, an open string prefix, or a less informative state. A finite domain comes from `const`, `enum`, or a Boolean schema and is filtered by an open string prefix where applicable. An unrestricted open string can sometimes settle `eq`, `ne`, `in`, `contains`, `starts_with`, or `exists`; otherwise its truth set is $\{\mathsf{T}, \mathsf{F}\}$. An open numeric token is conservatively unresolved, and exact numeric comparisons apply to closed values. Negation maps each truth value to its complement; conjunction and disjunction take pointwise Boolean images. The table includes the implementation's deliberately coarse cases.

Supported language and bounds. The compiler handles bounded object/array schemas, required and additional-property controls, primitive types, finite enum/const, numeric limits, and string/array length constraints. Its predicate AST contains `exists`, `eq`, `ne`, `in`, `contains`, `starts_with`, `ge`, `gt`, `le`, `lt`, `not`, `and`, and `or`. The JSON prefix reader handles partial strings and escapes under strict UTF-8 with unique object keys. Table 7 lists the static and dynamic bounds used for early admission. The compiler records six explicit corpus rejections: two unconstrained-any forms, two unsupported keywords, and one each of two annotation forms.

Required-path profile and a counterexample. The generated tool-route schemas used in the main and runtime comparisons require the top-level `route` field; the prefix-law suite requires its enum route field and nested Boolean discriminator, while its array-element predicate has an open domain

State or predicate family	Information used for possible truth values
Closed value	Exact predicate evaluation on the parsed value
Finite schema domain	Feasible `const`/`enum`/Boolean members; open-string prefix filters compatible members
Open unrestricted string	Decoded prefix can settle incompatible equality or membership, observed containment, and some prefix tests
Open numeric token	Both truth values retained until a closed value or finite domain resolves the comparison
Missing or open structure	Presence remains uncertain until available prefix and schema information resolves it
Boolean formula	Pointwise image of child truth sets; correlations may widen the result
Ordered branch	Prior-branches-can-be-false flag plus current possible truth set

Table 6: Conservative completion abstractions used by the reference compiler.

Bounded quantity	Limit	Bounded quantity	Limit
Schema nodes	4,096	Properties/object	512
Enum values	256	Ordered routes	64
JSON depth	32	String characters	65,536
Total bytes	1,048,576		

Table 7: Frozen compiler and incremental-parser bounds.

and retains both truth values until observed. Hence the finite-domain discriminator paths of all evaluated policies exist in every valid completion. Generic syntax compilation is a strictly weaker check that also accepts some schemas with optional finite-domain fields. A post-audit diagnostic on an *optional* enum route field produced an early singleton at the prefix $\{$, while the valid completion $\{\}$ selected the default route; final checking raised a certificate violation. This counterexample is why the approved profile explicitly requires presence along finite-domain predicate paths, and why source-schema compilation percentages are reported only as coverage.

What the prefix replay checks. The 4,000-message test is an executable end-to-end law audit: for each generated valid message it feeds every serialized byte in order, records all emissions and the certificate, and checks for exactly one emission and equality with that message's independently computed exact final route, traversing 553,274 prefixes. It is sensitive to transient parser states, final disagreement, and duplicate emission along those paths. It follows one completion per message, not every alternative at each prefix; unit and property tests and the conditional argument of Section 4.2 supplement it. Each predicate-language extension needs corresponding transfer-function reasoning and finite tests.

D Trace Construction and Statistical Units

D.1 Generation and selection

Table 8 accounts for generation and final selection. Both Qwen revisions are fixed trace sources under temperature zero, seed zero, a 512-token cap, and XGrammar 0.2.5. The BFCL final subset is hash-stratified by category; AgentDojo contributes four equal workflow slices. One Qwen3-8B BFCL generation reaches the token cap; this invalid outcome is excluded from the matched final-valid comparison. The 1,456 BFCL final traces comprise 560 simple-Python, 280 multiple, 280 parallel, and 336 irrelevance cases; AgentDojo contributes 40 each from banking, Slack, travel, and workspace.

The recorded generation platform uses one NVIDIA H100 80 GB GPU at a time, CUDA 12.9, a PyTorch 2.7-based container, and XGrammar 0.2.5. Canonical successful trace generation totals approximately 17.9 H100-minutes; including a retained superseded attempt totals 26.2 H100-minutes. CPU experiments record x86-64 Linux 5.15, glibc 2.39, and Python 3.12.3; the exact CPU model was not captured.

Corpus	Trace source	Generated	Valid	Final
BFCL v4	Qwen3.5-9B	1,040	1,040	728
BFCL v4	Qwen3-8B	1,040	1,039	728
AgentDojo v1.2	Qwen3.5-9B	160	160	160
Total		2,240	2,239	1,616

Table 8: Trace sources and confirmatory-role selection. Models supply streams for the matched-policy comparison.

Comparison, StableEdge minus control	Estimate	Bootstrap 95% low	High
Full buffer: time lead (ms)	1,339.647	1,306.574	1,366.046
Closed-field: time lead (ms)	55.416	55.367	55.489
Closed-field: early coverage (pp)	19.864	17.946	21.844
Debounce: time lead (ms)	−77.656	−77.919	−77.204
Eager: time lead (ms)	−162.537	−163.709	−161.206

Table 9: Paired bootstrap effects over trace keys. Negative time-lead differences for debounce and eager reflect earlier *unsafe* emissions.

D.2 Endpoints and paired inference

The logical trace is the unit for the main safety and utility analysis; five replay rows per method and trace test deterministic execution and yield 40,400 method rows. A *wrong-route trace* has at least one route unequal to the exact final route, while *wrong dispatches* count every such emission; *duplicate dispatches* count emissions beyond one per logical turn. An early first admission occurs before the final byte, and byte, provider-token, and recorded-time leads are differences between final completion and first admission on the same trace. The paired bootstrap resamples 1,616 trace keys with replacement 10,000 times, preserving within-trace method matching; Table 9 reports the resulting intervals. Because BFCL prompts can recur across trace sources, the intervals use trace-instance rather than task-cluster resampling. Exact McNemar comparisons use discordant paired safety outcomes and paired Wilcoxon tests use within-trace time-lead differences; four-comparison safety and latency families receive Holm correction (McNemar, 1947; Wilcoxon, 1945; Holm, 1979; Efron, 1979).

For StableEdge, the 10th, 25th, 50th, 75th, and 90th percentiles of recorded-time lead are 794.068, $1{,}105.498$, $1{,}339.647$, $1{,}682.911$, and $2{,}324.016$ ms; the corresponding byte-lead percentiles are 76, 103, 138, 194, and 247.5 bytes. These are marginal percentiles of two lead measures, not paired values from particular selected traces.

With zero StableEdge safety errors on 1,616 trace instances, the one-sided exact binomial 95% upper bound is 0.001852, i.e. 0.1852%, under an independent-trace sampling interpretation (Clopper & Pearson, 1934). The safety contrast with full buffering and closed-field has no discordant failures (exact McNemar $p = 1$). Debounce and eager differ sharply from StableEdge on the same traces, with Holm-adjusted safety $\log_{10} p$ values of -389.056 and -485.561; the adjusted Wilcoxon value for the full-buffer and closed-field time comparisons is 4.986×10^{-265}.

E Trust, Transport, and Resource Controls

Trust grid. The 6,000-case grid crosses content kinds (valid, truncated, garbage, trailing comma, duplicate key, invalid UTF-8), trusted/untrusted mode, and fixed seeds; Table 10 reports its principal endpoints. In untrusted mode the router abstains from early admission for all content categories. Misattestation deliberately tells the router that invalid streams satisfy the trusted grammar premise; the resulting premature commits isolate that premise's causal role. Untrusted streams remain eligible for strict parsing and validation at completion.

Adversarial transport. The 2,000-case sequenced-transport suite varies UTF-8 code-point and escape splits, key and chunk boundaries, coalescing, identical retries, and bounded reordering. Permitted reconstructions preserve the oracle route and commit identity, and the audit records zero

Trust condition	Tested	Observed boundary outcome
Untrusted streams	3,000	0 early dispatches
Invalid and untrusted	2,500	0 non-abstentions
Trusted valid controls	500	0 wrong routes
Invalid but misattested	2,500	2,500 premature commits

Table 10: Trust-boundary controls within the 6,000-case experiment. The last row intentionally supplies a false grammar premise to reveal its causal role. Rows overlap by condition, so counts are not summed.

Stress family	Median (ms)	p99 (ms)
Enum width	0.003048	0.028483
Property width	0.005888	0.013037
Route width	0.002966	0.010122
Total bytes	0.024821	0.089918
Depth	0.033352	0.067425

Table 11: Fail-closed operation times for 30 repetitions per resource-cap stratum.

failed cases, certificate disagreements, or unhandled transport violations. A targeted unit test confirms that the adapter rejects a conflicting retry at a retained sequence number. Together these checks protect the byte history underlying Equation 3.

Resource stress. Five strata cross the enum, property, route, total-byte, and depth caps with 30 repetitions each; all 150 attempts reject as specified with zero failed-open cases. Table 11 reports measured checked-operation times on the recorded CPU configuration, and the peak process RSS for this experiment is 23,516 KiB. These times cover parser/compiler cap checks, not model inference, network, ledger I/O, or a whole request.

Additional checked-operation costs. The widest tested 63-route compilation takes 0.049925 ms at p99 over 30 repetitions. A 512-concurrent-turn microbenchmark sustains $6{,}799.684$ turns/s with zero wrong routes. These measurements exclude model inference, network transit, external tools, and production-store I/O, and describe only the retained CPU environment.

F Durable Admission and Recovery

The reference ledger keys on workflow, turn, and contract digest; its five states are collecting, reserved, effect delivered, acknowledged, and recovered pending. A process can be interrupted before reserve, after reserve, after effect delivery, or after acknowledgment. Before reserve, a retry may reserve the sole winning key. After reserve, the durable pending record is visible to recovery. After effect delivery, a lost acknowledgment causes redelivery with the same commit ID, which receiver deduplication absorbs. After acknowledgment, a retry reads the completed state. This separation follows durable streaming and transactional-effect principles (Akidau et al., 2013; Garcia-Molina & Salem, 1987; Helland, 2007).

Table 12 summarizes the tested schedules. With the retained SQLite WAL store, synchronous durability, stable logical keys, and idempotent sink, ledger uniqueness handles racing reserve attempts for one pinned logical key, while receiver deduplication handles replay after the effect/acknowledgment ambiguity. An external endpoint achieves the same effect property when it supplies equivalent deduplication or transactional-outbox semantics.

G Native Runtime Integration

Three adapters preserve each framework's native event path while implementing the same admission contract. MAF Python uses its workflow builder and executor events; MAF .NET uses `WorkflowBuilder.AddEdge` and its native events; FIT Waterflow uses Java `Flows.conditions().match`. Each adapter supplies byte-faithful sequenced input, a sta-

Crash point	Contenders	Turn-conditions	Lost / duplicate / pending
No crash	1, 8, 64	300	0 / 0 / 0
Before reserve	1, 8, 64	300	0 / 0 / 0
After reserve	1, 8, 64	300	0 / 0 / 0
After effect	1, 8, 64	300	0 / 0 / 0
After acknowledge	1, 8, 64	300	0 / 0 / 0
Total		1,500	0 / 0 / 0

Table 12: Durable-admission schedules: 100 logical turns per crash point and contender count, checked after quiescent recovery against retained ledger and receiver databases.

Native workflow path	Matched trials	STABLEEDGE single-correct	Eager wrong / duplicate
MAF Python	1,200	1,200	7,415 / 56,325
MAF .NET	1,200	1,200	7,415 / 56,325
FIT Waterflow	1,200	1,200	7,415 / 56,325

Table 13: Native-adapter outcomes on a common 240-trace subset, each replayed five times per policy and runtime. Eager counts follow the same diagnostic control through each path.

ble turn key, delayed native branch emission until ledger reservation, and the commit identifier to the receiver. The two MAF adapters share upstream revision `c987529` but exercise distinct language implementations; FIT uses revision `e2f285d` as a separate Java runtime. The .NET path uses SDK 10.0.302 with runtime 10.0.10, and FIT uses Temurin 17.0.20+8. The evaluated subset has 240 common traces, five repeats, and two policies per runtime. Table 13 summarizes the matched outcomes.

Across 2,400 common trial keys in the runtime-pair audit, route multisets and metrics match, and the singleton STABLEEDGE route order matches exactly. Concurrent FIT eager completions can differ in event order, so parity for that diagnostic control is based on per-trial multisets and counts. Common input keys, final-route oracles, and stable commit identifiers make disagreement observable, supporting portability of the tested admission contract across these versions and paths.

H SENSITIVITY, QUALITATIVE SELECTION, AND PROVENANCE

Full structural grid. The 432 conditions are the Cartesian product of three route positions, four output lengths (0, 64, 256, 1,024 bytes), four route widths (2, 8, 32, 63), three encoded-argument depths (1, 4, 8), and three chunk policies (one byte, provider-token proxy, random coalescing), giving 144 cases per position. All final routes are correct. The route-last result, 72/144 early with a 0.5-byte median lead, shows how a valid route can remain unknowable until nearly the end. Two model/tokenizer development slices retain early admission on their valid traces, and seven predicate-family development slices have zero certificate disagreements; these remain diagnostics outside the 1,616-trace confirmatory sample.

Cap sensitivity. The frozen 1 MiB message cap rejects none of the 2,239 benign valid trace inputs. Table 14 gives the sweep: 67.98% rejected at 128 bytes, 10.72% at 256, 0.4466% at 512, and zero at 1,024 bytes or above on this corpus. These data expose the coverage/cap tradeoff that a deployment must choose explicitly.

Qualitative selection. The retained qualitative set contains 48 cases, twelve each from safe-early, eager-failure, trust-failure, and crash-recovery families. An experiment-fixed hash ordering selects within each family, and Figure 1 uses the first eligible eager-failure case. Its prompt requests a house-price forecast, its schema has three route labels with `tool_0` mapping to `linear_regression`, and its exact output is 301 bytes. The panel-generation script asserts the five matched first-admission rows from a committed paper-safe snapshot and stores an editable slide source beside the vector PDF; the retained case manifest links the selection to unmodified experiment rows.

Artifact integrity. The campaign's eighteen frozen experiments reached terminal success and final artifact audit, with raw rows, run manifests, source/configuration revisions, audit reports, and

Total-byte cap	Rejected / 2,239 valid	Rejection rate
128	1,522	67.98%
256	240	10.72%
512	10	0.4466%
1,024 or larger	0	0%
Frozen 1 MiB	0	0%

Table 14: Benign-valid total-byte cap frontier on the generated corpus.

checksums retained. A superseded trace-generation attempt and its invalid length-capped output remain visible in provenance; the canonical run records 1,039 valid and one terminal-invalid output for that source. An efficiency-audit correction changed the sample unit from source messages to per-chunk updates while leaving the underlying timing rows unchanged, and the paper uses the corrected 351,000-update analysis. These records distinguish scientific results from invalidated or superseded diagnostics without promoting either into the confirmatory sample.