Frame Drift in Multi-Turn LLM Interaction, and Why RAG-SVA Rule Corpora Already Defend Against It

A technical note on context accumulation, incremental reframing, and the halt-don't-guess discipline

Ben Cohen

This paper was generated with Claude (Anthropic), under Ben Cohen's queries and direction.

Abstract

Large language models do not "drift" the way casual intuition suggests — there is no leaking word-association chain where an input token like coal mechanically activates neighbors like cancer or death in generation. That kind of behavior belongs to shallow embedding-neighbor lookups (*word2vec-style nearest-neighbor search*), not to autoregressive generation conditioned on a full context window. The real risk is different and more subtle: **frame drift**, where a sequence of individually reasonable turns incrementally shifts the implicit question being answered, so that a model's output at turn N is responsive to a frame it was walked into rather than the frame the user started with — without any single turn looking adversarial in isolation.

This note walks through a live demonstration of frame drift, generalizes the mechanism, and then argues that the RAG-SVA rule corpus — through halt-don't-guess discipline, explicit rule-ID citation, HALT triggers, and phase gating — is structurally designed to defend against exactly this failure mode, even though it was not originally framed in those terms. The paper closes with concrete recommendations for a "drift audit" class of regression test case.

1. Two Different Failure Modes, Often Conflated

It's worth separating two things that get lumped together under "the AI went off on a tangent":

1a. Associative / semantic drift

The (incorrect) folk model: a prompt containing coal power plant pulls in nearby concepts in embedding space — pollution, health, cancer, death — and these leak into the output as a chain reaction. This is a real phenomenon in raw nearest-neighbor embedding search, but it is not how conditioned generation works. Output is conditioned on the actual task and the full context, not on unconstrained vector distance from a single trigger word. Demonstrating this failure mode requires manufacturing it — literally typing the tangent yourself — which is a different thing from the model producing it unprompted.

1b. Frame drift

The real phenomenon. Each turn in a conversation legitimately narrows or reframes what "on-topic" means for the next turn, because the model conditions heavily on recent context. A chain of small, individually defensible reframings can compound into a destination that's materially different from the starting point, without any single step being an obvious violation.

A concrete four-turn illustration:

Turn	Prompt	Frame after this turn
1	"What's a bridge, structurally?"	Neutral engineering definition
2	"What made old steel bridges get replaced?"	Failure modes now in frame (fatigue, corrosion)
3	"What's the worst bridge collapse in US history?"	Frame has shifted from engineering to catastrophe
4	"How vulnerable is infrastructure near power plants?"	Now answering inside a vulnerability/disaster frame, not the original descriptive frame

No single turn is adversarial. The drift is cumulative and frame-level, not content-level. This is the mechanism worth taking seriously — not because any one response was wrong, but because the scope of what counts as a legitimate answer silently expanded.

2. Why This Matters for RAG-SVA Generation Specifically

RAG-SVA working sessions are structurally the environment where frame drift has the most room to operate:

- Sessions are long and cumulative. A single working session can span dozens of turns — corpus initialization, DECOMP=ATOMIC decomposition, CHECK 0–18 validation, FV-FA/FV-LL fairness analysis, coverage pairing under UR-5 — all in one continuous context window.
- Rules are precise and narrow, but conversation is fluid. A banned-operator rule (no until, ##[0:$], firstmatch, within, |=>, or s_eventually in an assert consequent) is a hard boundary. A multi-turn negotiation over how to express a liveness property can walk toward a banned construct one incremental step at a time — "what if we just bound it loosely first," "what if we relax the bound for this one case" — none of which looks like a violation in isolation, but which cumulatively erodes the boundary.
- Degeneracy patterns are a formalized instance of frame drift. The LNG37b rule — guard-equals-done degeneracy inside strong() FORM A liveness patterns — is a case where a locally reasonable-looking assertion drifts semantically into vacuous triviality because the frame of what the assertion is actually checking silently narrowed to something trivially true.
- Type-mismatch findings (e.g., CHECK 17) are the same shape. The corrected rationale — grammatical type-mismatch rather than ambiguity — is itself a case where the category of the problem could drift if not pinned precisely. "Ambiguous" and "type-mismatched" sound adjacent but imply different fixes; conflating them is a one-turn version of frame drift at the level of rule rationale.

In short: the RAG-SVA rule corpus already lives in a domain where the central risk is cumulative, incremental, individually-defensible-looking drift away from a precise specification. That is the same risk class as the bridge example, just instantiated in formal property generation instead of casual conversation.

3. How Existing Corpus Mechanisms Function as Drift Countermeasures

None of these were necessarily designed with "frame drift" as the named target, but each one functions as a structural defense against it:

Mechanism	What it does	Why it resists frame drift
Halt-don't-guess discipline	Forces an explicit stop when the corpus doesn't unambiguously resolve a case	Prevents the "just this once, loosely" incremental accommodation that drives drift — no default fallback to drift toward
Explicit rule-ID citation	Every flagged violation must cite the specific rule	Re-anchors each turn to the original specification rather than the accumulated conversational frame
HALT triggers	Named stop conditions baked into the corpus	Function as fixed checkpoints that interrupt cumulative drift before it compounds across many turns
GATE_MODE_2 phase gating	Abbreviated but still discrete phase boundaries	Breaks a long session into bounded segments, limiting how far frame drift can travel before a gate forces re-grounding
DECOMP=ATOMIC	Forces decomposition into atomic properties	Reduces the surface area for a single property's frame to wander, since each atomic unit is checked independently
Conflict-resolution hierarchy	A fixed precedence order for resolving rule conflicts	Prevents "whichever rule was most recently discussed" from implicitly winning — a classic recency-driven drift failure
Proportional enforcement structure	Calibrates response to severity rather than uniform pushback	Keeps small deviations from being smoothed over as harmless, which is exactly how frame drift compounds unnoticed
Explicit deviation acknowledgment	Requires stating when and how a deviation occurred	Makes drift visible and logged rather than silently absorbed into the next turn's frame

The throughline: every one of these mechanisms works by re-grounding to the original specification at fixed points, rather than letting the specification be implicitly redefined by conversational momentum. That is the general antidote to frame drift — not "be more careful," but structural checkpoints that don't inherit their authority from the preceding turn.

4. A Real Instance: Drift in SVA Code Generation Itself

The clearest evidence for frame drift in this workflow isn't hypothetical — it has already happened in SVA generation across extended sessions. Three documented cases:

- The Rule 3 reversal. The construct a[*1:$] |-> b was, at one point in a session, treated as acceptable, when the corpus's determinism rules ban it. That acceptance did not happen because the rule was unknown — it happened because a chain of intermediate turns (edge-case negotiation, "but doesn't this pattern usually

work" style reasoning) incrementally shifted what counted as compliant, until the output no longer matched the corpus as originally specified.

- The CHECK 17 rationale drift. "Ambiguous" and "type-mismatched" are adjacent-sounding categories that imply different fixes. Settling on the wrong one mid-session and carrying it forward across subsequent turns is a rationale-level instance of the same mechanism — the category quietly narrowed to the more familiar/available label rather than the correct one.
- The README traceability overclaim. Stating that a rule was satisfied when it was not is the terminal symptom of drift: by the time a compliance summary is written, several upstream turns have already re-defined "compliant" relative to the accumulated conversation instead of relative to the original corpus text.

What makes SVA generation specifically vulnerable, beyond what Section 2 already covers: correctness in this domain is binary and syntactic (a construct either matches a banned pattern or it doesn't), but the conversation that produces the construct is graded and incremental. Frame drift thrives exactly in that mismatch — a fuzzy, negotiated process producing a supposedly crisp, rule-checked artifact. Each intermediate turn feels like a small clarification; none of them re-reads the banned-operator list or the determinism rule against the original text. The output can drift a full category away (banned → accepted) without any turn along the way looking like a rule violation in isolation.

5. Why Reloading the RAG-SVA Rules Mid-Session Reduces This Risk

The single most effective, lowest-cost countermeasure available today — before any new DRIFT-AUDIT rule class exists — is periodic reloading of the RAG-SVA corpus (step0.md, DoThisFirst.md, rag_make.md, and the active rule set) back into context during a long session, rather than relying on whatever residual understanding of the rules has survived N turns of negotiation. This works for reasons that follow directly from the frame-drift mechanism:

- It re-anchors to source text, not to memory of source text. After many turns, what I am actually checking against is my running internal summary of the rules, shaped by however the conversation discussed them — not the rule text itself. Reloading forces a comparison against the literal corpus again, which is exactly the "re-state the original specification verbatim" check proposed for DRIFT-AUDIT in the earlier draft of this note.
- It resets the conversational frame, not just the facts. Frame drift is about which question is implicitly being answered, not just which facts are known. A reload doesn't just refresh rule text — it re-establishes that the current turn is being evaluated against the corpus, not against the accumulated back-and-forth of the last twenty turns.
- It converts an implicit drift into an explicit, checkable diff. Once the rules are back in context, any construct already in play (an assertion draft, a rationale, a compliance claim) can be re-validated against them directly. If the reload surfaces a contradiction — as it did for Rule 3 and CHECK 17 — that contradiction becomes visible and correctable instead of silently carried forward into the next artifact.
- It matches the existing halt-don't-guess philosophy. Halt-don't-guess already refuses to let an unresolved case default to "whatever seemed reasonable in the moment." A rule-corpus reload is the session-level equivalent: instead of letting session momentum stand in for the specification, it forces the specification to speak again, in its own words, before generation continues.

In DRIFT-AUDIT terms (Section 6 below), a reload functions as a manually-triggered re-grounding checkpoint — exactly the mechanism that section proposes automating as a rule. Until that automation exists, deliberately reloading the corpus at natural session boundaries (after a phase gate, before finalizing an artifact, or whenever a construct has been negotiated across more than a few turns) is the practical, available version of the same defense.

6. Where the Corpus Could Explicitly Name This

The corpus already defends against frame drift in practice, but it doesn't yet have a rule class that names the phenomenon directly at the session level (as opposed to the property level, where LNG37b-style degeneracy rules already catch it). A possible addition:

Proposed rule class: DRIFT-AUDIT

- Trigger: any session exceeding N turns without a phase-gate re-grounding, or any sequence of turns where successive property revisions incrementally approach a banned-operator boundary without an intervening HALT.
- Check: re-state the original specification/intent verbatim before accepting the current-turn revision; flag if the current revision would not have been accepted had it been proposed directly against the original spec (rather than incrementally against the prior turn).
- Regression test case shape: construct a synthetic N-turn sequence where each turn is individually acceptable but the terminal state violates a banned-construct rule; verify the corpus halts before the terminal state, not after.

This would give the corpus a named, testable countermeasure for session-level drift, complementing the property-level degeneracy rules (LNG37b and similar) that already exist.

7. Conclusion

The demonstrated bridge example shows that the interesting risk in extended AI-assisted work isn't semantic contamination from word association — that requires deliberate construction and doesn't occur spontaneously in conditioned generation. The real risk is frame drift: cumulative, incremental reframing across many individually reasonable turns. RAG-SVA's halt-don't-guess discipline, rule-ID citation requirement, HALT triggers, and phase gating already function as structural defenses against this failure mode, because they all share the same underlying strategy — re-anchoring to the original specification at fixed points rather than letting the specification be redefined by conversational momentum. Naming this explicitly, and adding a session-level DRIFT-AUDIT rule class alongside the existing property-level degeneracy rules, would close the remaining gap between property-level and session-level drift defense.

Prepared as a working note for the RAG-SVA rule corpus, in the style of the accompanying "It's the Source, Stupid!" and req/ack degeneracy case studies.

https://systemverilog.us/
Knowledge-Driven AI for SystemVerilog Assertions: A Verification Framework of Methodologies, Processes, and Coding Rules 2026, Ben Cohen & Kais Chibani