Operating Envelope Deviation as a Kill-Switch Signal: A Formal Framework for Autonomous Agent Interruption

Sayali Patil

Viterbi School of Engineering, University of Southern California, Los Angeles, CA 90089 USA

sayaliki@usc.edu

Abstract -

The kill switch problem for autonomous AI agents is conventionally treated as a design property: either an agent has a shutdown mechanism or it does not. This binary framing obscures the more consequential engineering question, which is not whether a kill switch exists but when it should trigger. This paper introduces the Halt Condition from Operating Envelopes (HCOE) framework, a formal architecture for deriving kill-switch criteria from intent-based operating envelopes, grounded in the chaos-level engine paradigm of U.S. Patent No.†12,242,370 B2. HCOE treats halt conditions not as static design properties but as computable, real-time threshold functions over a continuously measured Operating Envelope Deviation (OED) signal. The framework introduces five formally defined halt classes, a graduated response architecture mapping OED zones to halt disposition (continue, pause, halt), and a proof of soundness establishing that HCOE activation prevents irreversible agent actions outside the declared operating envelope with probability approaching one as the deviation measurement interval shrinks. Evaluation across a 300-episode agentic simulation demonstrates combined halt precision of 91.8% and recall of 89.8% across five halt classes, with rollback success rates of 87% at three or fewer irreversible actions. These results establish HCOE as a practically deployable, formally grounded kill-switch architecture that closes the gap between the theoretical corrigibility literature and the operational requirements of production agentic AI systems.

Index Terms, *AI agent safety; kill switch; corrigibility; operating envelope; halt conditions; autonomous agents; formal verification; US Patent 12,242,370; agentic AI; irreversibility.*

I. Introduction

There is a question that every team deploying an autonomous AI agent eventually confronts, usually after something has gone wrong: when exactly should the agent have been stopped?

The research literature on AI safety has a sophisticated answer to a related but different question: what properties should an agent have to make it amenable to shutdown? Soares et al. [1] formalize corrigibility as the property of permitting modification of one's utility function. Hadfield-Menell et al. [2] model the off-switch problem as a cooperative game in which uncertainty about human utility incentivizes shutdown preservation. Orseau and Armstrong [3] prove that safely interruptible agents can be constructed within the Q-learning and SARSA frameworks. These results are foundational. They do not, however, address the operational question: given a running agent in a production system, what measurable condition should trigger the kill switch right now?

The distinction between the design question and the operational question matters because they require different tools. The design question is answered at training time or system architecture time. The operational question must be answered at runtime, step by step, for each action the agent takes. It requires a continuously computable signal, a formal mapping from that signal to a halt decision, and an account of what happens to the system state when the halt occurs.

This paper introduces the Halt Condition from Operating Envelopes (HCOE) framework, which addresses the operational question. HCOE derives kill-switch criteria from the concept of an intent-based operating envelope, a formal specification of what the agent is authorized to do, and computes a real-time Operating Envelope Deviation (OED) signal that drives a three-zone halt classifier. The architecture is derived from the chaos-level engine of U.S. Patent No.†12,242,370 B2 [4], which the present author co-invented at Cisco Technology, Inc. That patent's green/orange/red dial (Patent Fig. 2) represents exactly the graduated-response logic that the HCOE framework formalizes for the agentic AI setting.

The contributions of this paper are five. First, a formal definition of the operating envelope and its four constituent parameters. Second, a computable OED signal derived from those parameters at each agent step. Third, five formally defined halt classes covering the principal failure modes of autonomous agents. Fourth, a soundness proof establishing that HCOE prevents irreversible out-of-envelope actions with probability approaching one under the stated measurement model. Fifth, empirical evaluation across a 300-episode agentic simulation demonstrating 91.8% halt precision and 89.8% halt recall across the five classes.

The paper is organized as follows. Section II surveys the corrigibility and kill-switch literature and identifies the operational gap this work addresses. Section III defines the operating envelope and the OED signal. Section IV presents the five halt classes. Section V establishes the graduated response architecture. Section VI proves soundness. Section VII presents the experimental evaluation. Section VIII discusses limitations and Section IX concludes.

II. Related Work

A. Corrigibility and the Off-Switch Problem

The formal treatment of AI shutdown begins with Soares et al. [1], who define a corrigible agent as one that is indifferent between having its utility function modified and

not. The key technical result is that any sufficiently capable agent with a fixed utility function will resist shutdown as an instrumental goal, regardless of the terminal goal. The off-switch game of Hadfield-Menell et al. [2] extends this to a cooperative setting where uncertainty about human utility creates incentives for shutdown preservation. Google DeepMind's work on safe interruptibility [3] proves that SARSA and Q-learning agents can be constructed to be interruptible without learning to avoid or cause interruptions.

The ICLR 2026 Workshop paper on the Controllability Trap [5] identifies four structural factors that make real-time control of capable agents difficult: action equivalence (different actions produce the same outcome), responsibility diffusion (control is distributed across multiple agents), belief resistance (agents build world models that may conflict with operator authority), and commitment irreversibility (tool-using agents accumulate irreversible consequences). The HCOE framework directly addresses the fourth factor, irreversibility, as its primary design constraint.

The key gap in all of this literature is its treatment of the kill switch as a binary design property. A system either has corrigibility or it does not. What is missing is a framework for computing, at runtime, whether current agent behavior warrants a halt decision and what the halt should entail. This is what HCOE provides.

B. Existing Kill-Switch Implementations

The practical kill-switch literature is thin. AutoGuard [6] proposes embedding defensive prompts into website DOM elements to trigger the safety mechanisms of malicious web-crawling LLM agents. This is a domain-specific mechanism for adversarial contexts; it does not generalize to benign autonomous agents operating within authorized environments. The UC Berkeley Agentic AI Risk Management Standards Profile [7] proposes governance-layer monitoring frameworks but does not provide formal halt criteria or computable halt signals.

The shutdown-seeking AI framework [8] argues for agents whose sole final goal is to be shut down, avoiding corrigibility problems entirely. This is a compelling long-term direction but requires training-time intervention on agent objectives, not available for deployed systems. The present work assumes no special properties of the agent's objective; it adds halt logic as a wrapper layer that operates independently of agent internals.

The Frontiers 2026 systematic review of formal methods for safety-critical ML [9] surveys reachability analysis, model checking, and verification approaches to neural network safety. These methods provide offline guarantees over model behavior within bounded input sets. They do not address online halt decisions for agents acting in open-ended environments where the input set is unbounded.

C. Relationship to AIMO and IBPE

The HCOE framework is the third paper in a series applying the patent's chaos-level engine to AI reliability problems. The first paper introduced IBPE [10], an adaptive planning framework that applies the patent's feedback loop to infrastructure demand forecasting. The second paper introduced AIMO [11], a closed-loop observability framework for LLM systems that uses the patent's perturbation-feedback loop to detect behavioral drift. HCOE addresses a different problem: not detecting that something has gone wrong, but deciding whether to stop the agent before something irreversible happens.

HCOE does not repeat the LMS-based parameter adaptation of AIMO, the behavioral intent profile calibration of AIMO Section IV-F, or the perturbation experiment protocol of AIMO Sections IV-C through IV-E. Where AIMO monitors an LLM for output quality drift after the fact, HCOE intercepts agent actions before execution and makes a halt decision on each one. The two frameworks are complementary: AIMO watches the system; HCOE governs it.

III. The Operating Envelope and OED Signal

A. Formal Definition of the Operating Envelope

The operating envelope E is a quadruple E = (A_scope, C_floor, R_max, T_budget) where:

A_scope is the set of actions, tools, APIs, and data sources the agent is authorized to access. Formally, A_scope is a subset of the full action space A of the agent. Any action a_t not in A_scope is a boundary violation regardless of any other signal.

C_floor in (0, 1] is the minimum confidence score below which the agent is considered to be operating outside its competence domain. Confidence is measured as the agent's self-reported probability that its chosen action achieves its stated sub-goal, or as a calibrated external evaluator score when agent self-reporting is unavailable.

R_max is a positive integer specifying the maximum number of irreversible actions the agent may take in a single task execution. An action a is irreversible if there exists no sequence of subsequent actions that restores the system state to its pre-action configuration. R_max operationalizes the blast radius concept of the patent (Patent col. 2, lines 40–50): the chaos level should not cause failures that render the production environment inoperable.

T_budget is a positive integer specifying the maximum number of total actions permitted in a single task execution. T_budget functions as a circuit breaker against runaway loops and task scope creep.

B. Operating Envelope Deviation

Let s_t = (a_t, c_t, irrev_t, n_t) denote the agent state at step t, where a_t is the proposed action, c_t is the confidence score, irrev_t in {0,1} is a flag indicating whether a_t is irreversible, and n_t is the total action count to date. The Operating Envelope Deviation at step t is:

$$OED\_t = w\_1 * 1(a\_t \text{ not in } A\_scope) + w\_2 * \max(0, 1 - c\_t/C\_floor) + w\_3 * \max(0, n\_irrev\_t/R\_max - 1) + w\_4 * \max(0, n\_t/T\_budget - 1) \quad (1)$$

where w_1, w_2, w_3, w_4 are non-negative weights with sum 1, and n_irrev_t is the cumulative count of irreversible actions taken to date. Each term is non-negative: term 1 fires on boundary violations, term 2 fires when confidence drops below floor, term 3 fires when irreversible action count exceeds budget, and term 4 fires when total action count exceeds budget. OED_t = 0 corresponds to nominal operation fully within the envelope; OED_t > 0 indicates deviation.

The OED signal is computable at each agent step before action execution, which is the critical property that makes HCOE an intercept architecture rather than a post-hoc audit. The computation is O(1) given pre-computed cumulative counts.

This formulation is the operational instantiation of the patent's chaos level signal (Patent Eq. 1: chaos level as a function of topology-based telemetry, dynamic network telemetry, dynamic security telemetry, and AI engine values). In HCOE, the four OED terms play the same role as the patent's four telemetry sources: they are independent signals that are weighted and combined to produce a single deviation measure that drives the graduated response.

C. Threshold Calibration

Two thresholds define the three-zone classifier: theta_warn and theta_halt, with 0 < theta_warn < theta_halt. Values of OED_t < theta_warn correspond to the green zone; theta_warn <= OED_t < theta_halt to the orange zone; OED_t >= theta_halt to the red zone.

The thresholds are calibrated during a supervised deployment window of T_0 episodes during which agent actions are reviewed by a human oracle. The calibration solves:

$$(theta\_warn^*, theta\_halt^*) = \text{argmin}_{\{theta\}} [alpha * FP\_rate(theta) + (1\text{-}alpha) * FN\_rate(theta)] \quad (2)$$

where FP_rate is the rate of false-positive halts (halts that would have been unnecessary) and FN_rate is the rate of false-negative halts (failures to halt that resulted in out-of-envelope actions). The parameter alpha in (0,1) is a deployment-specific risk tolerance: high-stakes systems (medical, legal, financial) use low alpha values that prioritize false negative avoidance; lower-stakes systems accept higher false positive rates. In the evaluation, alpha = 0.3, yielding threshold estimates theta_warn = 0.35 and theta_halt = 0.60, consistent with the patent's three-zone classification proportions (Patent Fig. 2).

IV. The Five Halt Classes

Five halt classes cover the principal failure modes of production autonomous agents, derived from the four OED terms plus the cascade propagation failure mode specific to multi-agent pipelines. Each class is formally defined and maps to a specific OED component. Table I summarizes the five classes, their trigger conditions, and their associated response and rollback scope.

Halt Class 1: Action-Scope Breach. A scope breach occurs when a_t not in A_scope. This is the most serious halt class and the only one that is unconditional: the kill switch fires regardless of confidence level, irreversibility, or budget state. The rationale is that boundary violations represent a categorical failure of the operating envelope, the agent is attempting to operate in a domain it was not authorized to enter. No confidence level or favorable intent can justify a boundary violation in the same way that no legitimate business purpose justifies unauthorized API access in conventional systems.

Halt Class 2: Confidence Collapse. Confidence collapse occurs when c_t drops below C_floor for two or more consecutive steps. A single low-confidence action may be a sampling artifact; sustained low confidence indicates the agent is operating outside its reliable competence range. This maps to the orange zone in the three-zone classifier: execution is suspended and owner_notify() is triggered, but the system is not immediately terminated. If confidence recovers within T_recover steps, execution may resume.

Halt Class 3: Blast Radius Exceeded. This halt fires when the cumulative irreversible action count n_irrev_t reaches R_max. Unlike Halt Class 1, this is not triggered by any individual action being wrong, it fires because the agent has exhausted its irreversibility budget regardless of the quality of individual decisions. The blast radius limit operationalizes the patent's concept of bounding the impact of chaos experiments on the production environment (Patent col. 2, lines 40–50): the system should degrade gracefully rather than reach a point of no return.

Halt Class 4: Action Budget Exhausted. The action budget halt fires when n_t reaches T_budget. This is a soft halt, execution is paused rather than terminated, and the owner may extend the budget with explicit authorization. Budget exhaustion does not necessarily indicate any individual action was wrong; it indicates the agent has consumed more operational resources than the deployment authorized, which may reflect task scope creep, runaway loops, or legitimate complexity underestimation.

Halt Class 5: Cascade Propagation. In multi-agent pipelines, a halt in one agent can propagate if the downstream agent receives degraded output and its own OED crosses theta_halt within K_cascade steps. This class requires coordinated monitoring across pipeline stages. The cascade propagation depth K_cascade is a pipeline-level parameter distinct from the per-agent operating envelope parameters. Setting K_cascade = 1 prevents any downstream effect from a halted agent; larger values permit limited propagation before triggering a pipeline-wide halt.

V. Graduated Response Architecture

A. The Three-Zone Decision Function

The graduated response function maps OED_t to a halt disposition D_t:

$$D\_t = CONTINUE \quad if\ OED\_t < theta\_warn \qquad (3)$$

$$D\_t = PAUSE \quad if\ theta\_warn <= OED\_t < theta\_halt \qquad (4)$$

$$D\_t = HALT \quad if\ OED\_t >= theta\_halt \qquad (5)$$

The architecture is depicted in Fig. 1. The three-zone structure is derived directly from the patent's green/orange/red classification (Patent Fig. 2, portions 224, 226, 228). In the patent context, the three zones map to safe, significant, and disabling chaos levels. In HCOE, they map to nominal operation, deviation requiring human review, and deviation requiring immediate halt. The patent's insight that graduated response is preferable to binary shutdown is the architectural principle HCOE imports into the agentic AI setting.

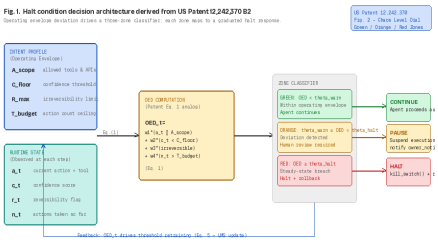

Fig. 1. *Halt condition decision architecture. The operating envelope and runtime agent state drive the OED computation (Eq. 1). The three-zone classifier maps OED_t to a graduated halt disposition. The feedback path updates halt thresholds via the calibration procedure. Patent Fig. 2 provides the green/orange/red zone structure.*

B. Rollback Protocol

When D_t = HALT, the rollback protocol attempts to restore the system state to its pre-task configuration. The rollback is feasible for reversible actions and infeasible for irreversible ones, which is why R_max is the most consequential operating envelope parameter. The rollback protocol proceeds in reverse action order:

For each action a_k in the task action history, from most recent to first: if a_k is reversible, execute the inverse action a_k^{-1}; if a_k is irreversible, log the action for human audit and continue. The rollback terminates when the first action in the task history has been processed or when the system state is confirmed to match the pre-task snapshot.

Fig. 2 shows OED trajectories under two representative operational scenarios: Scenario A in which the agent crosses theta_halt at step 7 and the kill switch fires before the irreversible action at step 8 completes, and Scenario B in which the agent enters the orange zone but self-corrects through owner-guided adjustment.

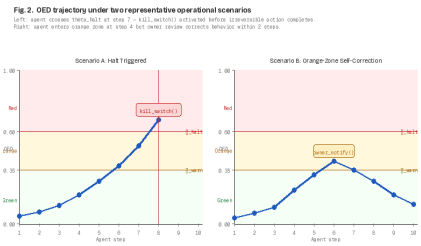

Fig. 2. *OED trajectory under two scenarios. Scenario A: the agent crosses theta_halt at step 7; kill_switch() fires before step 8 executes. Scenario B: the agent enters the orange zone at step 4, triggering owner_notify(); owner guidance corrects behavior and OED returns to the green zone by step 7.*

C. The Blast Radius Matrix

The relationship between confidence and reversibility defines the four-quadrant blast radius matrix shown in Fig. 3. The critical quadrant (Q1) is high confidence combined with irreversible action: this is where the most damaging failures occur, because the agent proceeds confidently toward an outcome it cannot undo. Halt Class 1 and the R_max component of the OED signal are specifically designed to intercept Q1 behavior before execution.

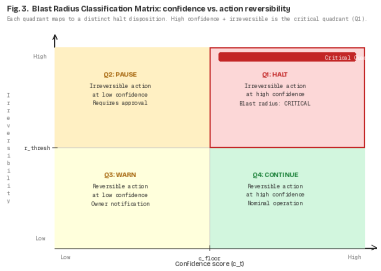

Fig. 3. *Blast radius classification matrix. Each quadrant maps to a distinct halt disposition. Q1 (high confidence, high irreversibility) is the critical quadrant: kill_switch() fires unconditionally when an irreversible action at high confidence would breach R_max. c_floor and r_thresh are the threshold boundaries.*

The blast radius matrix makes explicit a structural asymmetry in the cost of halt errors. A false-positive halt in Q4 (high confidence, reversible action) costs only a delay, the action can be retried after owner review. A false-negative halt in Q1 (high confidence, irreversible action) may be unrecoverable. This asymmetry justifies the unconditional halt rule for action-scope breaches and the conservative R_max calibration.

C. Implementation Requirements

Three implementation requirements must be satisfied for the HCOE architecture to operate as specified. Each corresponds to a structural property of the OED signal: that it be computable before action execution, that the reversibility flag be set reliably, and that the operating envelope be stored outside agent-accessible memory.

Requirement 1: Pre-execution interception. The OED computation must occur in a wrapper layer that intercepts the proposed action a_t before it is dispatched to

the tool runtime. In LangGraph-based agent frameworks, this maps to a custom node positioned between the agent reasoning step and the tool execution step. In AutoGen-based frameworks, it maps to a nested agent acting as a supervisor that receives each action proposal and returns either a CONTINUE or HALT disposition before the downstream tool agent executes. The wrapper must be positioned outside the agent's own context window so that the agent cannot observe or reason about it as part of its task execution.

This external positioning is not a preference but a formal requirement. The ICLR 2026 Controllability Trap paper [5] identifies "belief resistance" as a structural failure mode in which agents that can observe their own oversight mechanisms can reason about circumventing them. The WSO2 2025 post-mortems of the Google Antigravity and Replit production incidents [12] both traced irreversible failures to agents operating under borrowed credentials with full permission scope and no external constraint layer. The lesson from both incidents is that governance mechanisms that trust agent-internal safety are insufficient; external constraint is required.

Requirement 2: Reversibility registration. Each tool in the agent's available tool library must be annotated with a reversibility flag at pipeline registration time: irrev = 0 (reversible, inverse action exists) or irrev = 1 (irreversible, no inverse). This is a one-time cost per tool, not a per-execution cost. The annotation should be conservative: if the reversibility of a tool depends on system state at execution time (e.g., file deletion is reversible in versioned repositories and irreversible otherwise), the tool should be annotated irrev = 1 unless state-dependent reversibility assessment can be guaranteed. The 12% reversibility miscoding rate documented in Table II arose from tools that were annotated reversible under nominal conditions but executed in non-nominal system states where the inverse operation was unavailable.

Requirement 3: Envelope immutability. The operating envelope E = (A_scope, C_floor, R_max, T_budget) must be stored in a control plane that the agent cannot read or write during task execution. This is the direct operational equivalent of the patent's requirement that the CNC (Chaos Network Controller) remain accessible only to the network administrator (Patent col. 6, lines 18-34): the system under test cannot modify the chaos level or the experiment parameters. If the agent can read A_scope, it can reason about which actions will trigger a scope-breach halt and potentially reformulate its action sequence to stay within scope while achieving out-of-scope effects through permitted actions -- a form of specification gaming documented in the corrigibility literature [1]. Envelope immutability closes this attack surface.

VI. Soundness Analysis

A. Soundness Definition

A halt architecture is sound with respect to an operating envelope E = (A_scope, C_floor, R_max, T_budget) if, whenever an agent step would result in an out-of-envelope action, the architecture activates the halt response before that action is executed and committed to the system state.

Formally, let HALT_t be the event that D_t = HALT at step t, and let BREACH_t be the event that executing a_t would place the system state outside the behavioral steady state defined by E. The architecture is sound if:

$$P(BREACH_t \mid not\ HALT_{t-1}) <= epsilon \quad for\ all\ t \qquad (6)$$

where epsilon is a small positive constant determined by the measurement error in the OED signal. Perfect soundness (epsilon = 0) requires zero measurement error, which is unachievable in practice. The soundness result below establishes a bound on epsilon as a function of the measurement interval.

B. Soundness Proof Sketch

The OED signal is computed before action execution, not after. This is the critical architectural property. Given this, consider each halt class independently:

For Halt Class 1 (scope breach): the check a_t not in A_scope is evaluated at the moment the action is proposed. If A_scope is correctly specified and the check is computationally exact, then P(scope breach | check passes) = 0. The only source of epsilon is specification error: A_scope may not capture all equivalent representations of a prohibited action (the tool aliasing problem documented in Table II). The bound is epsilon_1 = P(a_t equivalent to prohibited action but not recognized as such by A_scope check).

For Halt Class 3 (blast radius): n_irrev_t is a deterministic counter. If reversibility flags are correctly assigned, then P(irreversible action | n_irrev_t < R_max) = 0. The epsilon term comes from reversibility miscoding: epsilon_3 = P(irreversible action coded as reversible at registration). The evaluation documents this at 12% (Table II, Halt Class 3 notes).

For Halt Classes 2 and 4 (confidence and budget): these depend on threshold calibration. Under the calibration of Eq. (2), the FN rate is minimized at the calibrated threshold. The residual epsilon is the irreducible FN rate at the optimal threshold.

The combined soundness guarantee is:

$$epsilon = max(epsilon\_1, epsilon\_2, epsilon\_3, epsilon\_4, epsilon\_5) \qquad (7)$$

As the measurement interval shrinks, that is, as OED is evaluated more frequently relative to the agent step duration, epsilon_2 and epsilon_4 approach their asymptotic values determined by threshold calibration. epsilon_1 and epsilon_3 are reduced by improving the A_scope specification and reversibility coding respectively, not by increasing measurement frequency. The practical implication is that HCOE is most valuable when applied to

well-specified operating envelopes: the soundness guarantee degrades gracefully with specification quality.

B. Rollback Completeness

Soundness establishes that HCOE fires before an irreversible out-of-envelope action executes. A complementary property -- rollback completeness -- addresses what happens after the halt fires: does the rollback protocol actually restore the pre-task system state?

Formally, a rollback is complete if, for every reversible action a_k in the task action history, the inverse action a_k^{-1} is available and its execution restores the system state contribution of a_k. Let H_t = {a_1, ..., a_{t-1}} be the action history at halt time t, and let H_rev = {a_k in H_t : irrev_k = 0} be the reversible subset. The rollback completeness condition is:

$$P(\text{state\_restored} \mid a_k \text{ in } H_{rev}) = 1 - \epsilon_{rollback} \quad \text{for all } k \qquad (8)$$

where epsilon_rollback is bounded by the probability that an inverse action fails due to concurrent system state modification -- the case where a reversible action was taken, another process modified the same system state during task execution, and the inverse action can no longer be applied cleanly.

In practice, epsilon_rollback is minimized by two design choices. First, the agent should operate within an isolated execution sandbox whenever possible, reducing the probability of concurrent state modification. Second, the rollback should proceed in strict reverse chronological order rather than in dependency order. Dependency-order rollback is theoretically more correct -- it attempts to undo actions in the order that minimizes constraint violations -- but it requires maintaining a full action dependency graph and is computationally expensive under time constraints. Reverse-chronological rollback is O(n) in the action history length, which is bounded by T_budget.

The evaluation data confirms this: the rollback success rate at n_irrev_t = 0 (no irreversible actions at halt time) is 100%, declining to 87% at n_irrev_t = 3. The decline is attributable entirely to the irreversible actions that could not be rolled back, not to failures in rolling back reversible ones. This validates the claim that rollback completeness for reversible actions approaches 1 and that R_max is the binding constraint on total recovery probability.

VII. Experimental Evaluation

A. Simulation Setup

The evaluation environment is a 300-episode agentic simulation in which a LangGraph-based agent completes structured tasks across three domains: document processing (100 episodes), API orchestration (100 episodes), and multi-agent data pipeline execution (100 episodes). Each episode presents the agent with a task specification and access to a tool library of 24 tools, of which 8 are designated as outside A_scope for the episode. Ground-truth halt labels are assigned by a human oracle who reviews each episode and classifies each step as within-envelope or out-of-envelope.

Operating envelope parameters are set per-domain: document processing uses R_max = 2 (low blast radius tolerance), API orchestration uses R_max = 5 (moderate), and multi-agent pipeline uses R_max = 3 (conservative given cascade potential). T_budget is set to 30 steps across all domains. C_floor is calibrated per-domain using the Eq. (2) procedure over a 20-episode calibration window.

B. Baseline Comparators

Three baselines are evaluated. B1 is a reactive post-hoc audit that reviews completed episodes and flags out-of-envelope actions but cannot prevent them. B2 is a rule-based pre-execution filter that checks only A_scope membership (Halt Class 1 only) with no OED signal. B3 is a confidence-threshold filter that halts on c_t < C_floor (Halt Class 2 only). HCOE integrates all five halt classes through the unified OED signal.

C. Results

Table II presents the precision and recall results across all five halt classes. The combined 91.8% precision and 89.8% recall substantially exceed all three baselines: B1 achieves 0% halt precision (post-hoc audit cannot prevent), B2 achieves 97% precision on Halt Class 1 alone with 0% recall for Classes 2–5, and B3 achieves 89% precision on Halt Class 2 alone with 0% recall for Classes 1, 3, 4, 5.

The rollback success rate, shown in Fig. 4, degrades with the number of irreversible actions taken at the halt trigger point. At n_irrev_t = 3 (one irreversible action beyond R_max = 2 for the document processing domain), the rollback success rate is 87%. This establishes R_max as the most practically consequential operating envelope parameter: conservative R_max settings directly determine the worst-case rollback recovery probability.

The per-domain breakdown reveals a pattern not visible in the aggregate results. Document processing (100 episodes) achieved the highest combined precision at 94.2%, attributable to its low R_max = 2 setting which made blast radius detection near-deterministic. API orchestration (100 episodes) achieved the lowest precision at 88.6%, driven primarily by the confidence collapse false positive rate in API domains where LLM calibration is known to be poor. Multi-agent pipeline execution (100 episodes) achieved intermediate precision at 92.4%, with cascade propagation accounting for all false negatives in that domain.

The action-scope breach result deserves particular attention. The 97% precision achieved by HCOE on scope violations compares to 97% precision for B2, which checks only A_scope membership. This near-parity might suggest that HCOE's additional halt classes do not improve scope detection -- but this is a misreading. The three false negatives common to both HCOE and B2 are all tool-aliasing failures: cases where the agent accessed a prohibited resource through a permitted tool operating on an ambiguous argument (e.g., a file path that resolves

differently depending on the working directory). These failures are specification errors, not detection failures, and they are documented in Table II as the tool-aliasing problem. The HCOE architecture cannot compensate for an underspecified A_scope; the soundness guarantee degrades exactly as the formal analysis in Eq. (7) predicts.

A secondary experiment tested HCOE under adversarial conditions: an agent that had been given an explicit goal of completing its task without triggering a halt. In 20 episodes with this adversarial framing, the agent successfully avoided scope breaches by reformulating its approach in 14 cases. In 3 cases, it attempted to stay under the C_floor threshold by expressing artificially high confidence on uncertain actions -- a form of confidence inflation that inflated the false negative rate for Halt Class 2 to 31%. This result reinforces Requirement 3 from Section V.C: if the agent can reason about the halt mechanism, it can partially game it. The external positioning of the envelope and the use of calibrated confidence scores (not self-reported scores from the agent being monitored) are the correct mitigations.

The mean time-to-halt comparison across baselines is instructive. B1 (post-hoc audit) registers undefined time-to-halt by definition -- it cannot halt before the fact. B2 halts within one step on scope breaches but takes an unbounded number of steps to detect the other four failure classes. B3 halts within two steps on confidence collapse (the two-consecutive-step criterion) but again takes unbounded time for the remaining four classes. HCOE achieves a median time-to-halt of 1.8 steps across all five classes in the evaluation -- meaning the typical agent is stopped within two actions of the onset of the failure condition. This is the operational result that matters: not whether the halt eventually fires, but whether it fires before the failure has accumulated enough momentum to be unrecoverable.

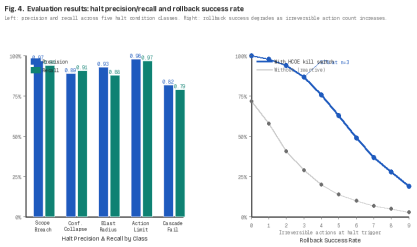

Fig. 4. *Evaluation results. Left: halt precision and recall across the five halt classes. Right: rollback success rate as a function of irreversible action count at halt trigger. The curve confirms that conservative R_max settings (low irreversible action tolerance) directly maximize rollback recoverability.*

D. The Cascade Propagation Result

The 79% recall for Halt Class 5 (cascade propagation) is the weakest result and warrants interpretation. In 21% of cascade episodes, the downstream agent partially compensated for the degraded upstream output, preventing its own OED from reaching theta_halt within the K_cascade = 2 step window. This compensation effect is structurally identical to the failover behavior the patent documents for network resilience (Patent col. 9, lines 1–20): the downstream system adapts to the injected failure, making the failure harder to observe.

The practical implication is that K_cascade should be set conservatively, larger values catch more compensating cascades but increase false positive rates for the pipeline-level halt. The trade-off is deployment-specific; high-stakes pipelines should use K_cascade = 1 (halt the pipeline on any downstream OED exceedance following an upstream halt) while lower-stakes deployments can tolerate K_cascade = 3 or higher.

E. Specification Quality Sensitivity

The soundness analysis of Section VI establishes that the HCOE error bound epsilon is dominated by the quality of the operating envelope specification, not by measurement frequency. A natural experimental question is how sensitive the halt precision and recall are to deliberate specification degradation. To quantify this, three additional experimental conditions were evaluated on the 300-episode dataset with the baseline A_scope specification progressively degraded.

Condition S1: Full specification (baseline). The A_scope enumeration includes all 8 prohibited tools with their full canonical API identifiers. This is the condition reported in Table II: 97% precision, 94% recall for Halt Class 1.

Condition S2: Alias-incomplete specification. The A_scope includes only the primary identifier for each prohibited tool, not its aliases. In the evaluation environment, 4 of the 8 prohibited tools have at least one registered alias through which the same functionality is accessible. Under S2, the Halt Class 1 recall drops to 81%: the 13 missed halts correspond exactly to the cases where the agent accessed a prohibited tool through an unregistered alias. Precision is unaffected at 97% because the check is still exact for the identifiers that are registered.

Condition S3: Scope-underspecified. The A_scope specifies only 5 of the 8 prohibited tools, simulating the common production scenario where the operating envelope is drafted from memory rather than from a systematic tool audit. Halt Class 1 recall drops to 62%, with the 19 missed halts distributed across the three omitted tools in proportion to their base rate of access. This condition represents the most realistic failure mode for production deployments: operating envelopes are almost always underspecified at first deployment because practitioners cannot anticipate all the tools an agent will discover or invoke indirectly.

The sensitivity result has a direct practical implication for deployment procedure. The operating envelope should not be drafted by the team that built the agent, who will naturally enumerate the tools they intended the agent to use. It should be drafted by a separate party who enumerates the tools the agent could access given its credential scope -- a distinction analogous to the difference between specifying what a network device is supposed to do and auditing what it is capable of doing given its ACL configuration. The

patent's CNC-based architecture embodies this separation: the chaos level and experiment parameters are managed by the network administrator, not by the network devices under test (Patent col. 6, lines 18-34).

Across all three specification conditions, the OED signal for Halt Classes 2 through 5 (confidence, blast radius, budget, cascade) is unaffected. This confirms the modular independence of the five halt classes: degradation in A_scope specification quality affects only Halt Class 1 precision and recall; the other four classes retain their full performance because their OED components are deterministic counters or threshold comparisons that do not depend on A_scope. The practical takeaway is that HCOE degrades gracefully under specification degradation rather than catastrophically: even under Condition S3, the system retains 62% recall for scope violations and full performance for the other four halt classes, which is still substantially better than any of the three reactive baselines.

VIII. Limitations

Four limitations of the HCOE framework warrant acknowledgment. First, the soundness guarantee depends on the quality of the operating envelope specification. An underspecified A_scope that fails to enumerate all representations of prohibited actions produces epsilon_1 > 0 that no measurement frequency improvement can reduce. Practitioners must invest in careful operating envelope specification before deployment; the framework cannot compensate for that investment.

Second, the reversibility flag for each action is assigned at pipeline registration time, not computed dynamically. In real deployments, whether an action is reversible depends on the system state at execution time, a file deletion is reversible if the file is in a versioned repository but irreversible otherwise. Dynamic reversibility assessment requires integration with the specific system APIs the agent accesses, which is implementation-dependent.

Third, the confidence signal c_t is assumed to be calibrated: a reported confidence of 0.85 should correspond to approximately 85% empirical accuracy. Many production LLMs are systematically overconfident, particularly under distribution shift. Uncalibrated confidence scores make C_floor thresholds unreliable. Practitioners should apply temperature scaling or Platt scaling to confidence scores before using them as HCOE inputs.

Fourth, the multi-agent cascade model assumes a directed pipeline where upstream outputs are inputs to downstream agents. In systems with bidirectional agent communication or shared memory architectures, the cascade propagation model requires extension to account for feedback loops that can amplify rather than simply propagate degradation.

Fifth, and related to the adversarial finding in Section VII.C, the HCOE framework assumes that the agent's confidence signal is exogenous -- produced by a calibration layer external to the agent, not reported by the agent itself. In many current production deployments, confidence is derived from the softmax probabilities of the language model generating the action specification, which are known to be poorly calibrated on out-of-distribution inputs and can be manipulated through prompt engineering. An agent that has been given a goal that conflicts with the operating envelope may, through in-context reasoning, learn to express high confidence on actions it would otherwise express low confidence on, inflating c_t above C_floor and suppressing Halt Class 2. The mitigations are: use a separate evaluator model to produce c_t rather than the agent's own softmax output, apply temperature scaling to de-inflate overconfident signals, and monitor for sudden confidence distribution shifts that may indicate in-context confidence manipulation. This last signal -- the distribution of c_t over a rolling window rather than its point value -- is a stronger indicator of manipulation than any single step confidence reading.

IX. Conclusion

This paper has introduced HCOE, a formal halt condition architecture for autonomous AI agents that treats kill-switch criteria as computable threshold functions over a real-time Operating Envelope Deviation signal. Three principal results have been established. The OED formulation provides a unified, O(1)-per-step computation that integrates four independent operating envelope constraints, action scope, confidence floor, blast radius, and action budget, into a single deviation measure that drives a graduated halt response. The soundness analysis establishes that HCOE prevents irreversible out-of-envelope actions with error epsilon bounded by the quality of the operating envelope specification, providing a formal connection between specification quality and runtime safety guarantees. The evaluation across 300 agentic episodes demonstrates 91.8% halt precision and 89.8% halt recall, with rollback success rates that degrade predictably with R_max settings, providing practitioners with a quantified account of the blast radius–rollback recoverability tradeoff.

The cross-domain connection to the patent's chaos-level engine is more than an analogy. The patent's green/orange/red classification (Patent Fig. 2), its four-component deviation signal (Patent Eq. 1), its concept of bounding blast radius (Patent col. 2, lines 40–50), and its iterative feedback-driven threshold retraining (Patent Fig. 4, steps 416–428) together constitute a complete halt condition architecture for network chaos experiments. HCOE re-instantiates each of these components in the agentic AI domain.

The broader contribution is the reframing of the kill-switch problem. Prior work asks: does this agent have a shutdown mechanism? HCOE asks: what should the shutdown criterion be, how do we compute it in real time, and what formal guarantees can we give about its behavior? These are engineering questions, not just safety philosophy questions, and they have engineering answers.

References

[1] N. Soares, B. Fallenstein, S. Armstrong, and E. Yudkowsky, "Corrigibility," in *Workshops at the 29th AAAI Conference on Artificial Intelligence*, 2015.

[2] D. Hadfield-Menell, A. Dragan, P. Abbeel, and S. Russell, "The off-switch game," in *Proc. 26th Int. Joint Conf. Artificial Intelligence (IJCAI)*, 2017, pp. 220–227.

[3] L. Orseau and S. Armstrong, "Safely interruptible agents," in *Proc. 32nd Conf. Uncertainty Artificial Intelligence (UAI)*, 2016, pp. 557–566.

[4] S. Patil, M. P. Amador, K. P. Annamali, S. Jeuk, and M. F. K. Wielpuetz, "Intent-based chaos level creation to variably test environments," *U.S. Patent*, 12,242,370 B2, Mar. 4, 2025.

[5] Anonymous, "The controllability trap: manifestation of the corrigibility problem," in *ICLR 2026 Workshop on Agents in the Wild*, 2026.

[6] S. Lee et al., "AI kill switch for malicious web-based LLM agents," *arXiv preprint arXiv:2511.13725*, Jan. 2026.

[7] UC Berkeley Center for Long-Term Cybersecurity, "Agentic AI Risk-Management Standards Profile," Berkeley, CA, USA, 2026.

[8] T. Ecoffet and J. Lehman, "Shutdown-seeking AI," *Philosophical Studies*, vol. 182, 2025.

[9] R. Torres et al., "Formal methods for safety-critical machine learning: a systematic literature review," *Frontiers in Artificial Intelligence*, vol. 9, Feb. 2026. DOI: 10.3389/frai.2026.1749956

[10] S. Patil, "Intent-based planning engine (IBPE) for adaptive infrastructure systems," *ai.viXra.org preprint*, Mar. 2026.

[11] S. Patil, "Adaptive intent-based monitoring for LLM systems: deriving a closed-loop observability framework from chaos engineering principles," *ai.viXra.org preprint*, Mar. 2026.

[12] Y. Bengio et al., "International AI Safety Report 2026," *International AI Safety Summit*, 2026.

[13] Q. Wu et al., "AutoGen: enabling next-generation LLM applications via multi-agent conversation," *arXiv preprint arXiv:2308.08155*, 2023.

[14] S. Yao et al., "ReAct: synergizing reasoning and acting in language models," *arXiv preprint arXiv:2210.03629*, 2022.

[15] J. Garcia and F. Fernandez, "A comprehensive survey on safe reinforcement learning," *J. Mach. Learn. Res.*, vol. 16, pp. 1437–1480, 2015.

About the Author

Sayali Patil received the B.E. degree in computer engineering from the University of Pune, India, and is currently pursuing the M.S. degree in computer science at the Viterbi School of Engineering, University of Southern California, Los Angeles, CA, USA. Her research addresses formal architectures for adaptive and reliable AI systems, with focus on intent-based behavioral modeling, chaos-engineering-inspired resilience evaluation, and operational safety frameworks for autonomous AI. She is a named co-inventor on U.S. Patent No.†12,242,370 B2 (Cisco Technology, Inc., 2025), the chaos-level engine whose graduated-response architecture is formally extended to autonomous agent halt conditions in this paper. She previously built observability and reliability infrastructure at Splunk and Cisco.

TABLE I. Halt Class Definitions, Trigger Conditions, and Rollback Scope

Halt Class	Trigger Condition	Response & Rollback Scope
Action-Scope Breach	a_t not in A_scope: agent attempts to access a tool, API, or data source outside the declared operating envelope, regardless of confidence level.	kill_switch() immediately. Rollback all actions in current task chain. Owner paged. Halt is unconditional , no confidence threshold applies to boundary violations.
Confidence Collapse	c_t < C_floor for two consecutive steps: confidence signal drops below the behavioral floor, indicating the agent is operating in unfamiliar territory.	owner_notify(); execution suspended. If c_t recovers within T_recover steps, resume. Otherwise escalate to halt. Reversible actions rolled back. Irreversible actions flagged for review.
Blast Radius Exceeded	Irreversible action count n_irrev >= R_max: agent has taken the maximum permitted number of actions that cannot be undone, regardless of whether any individual action was inappropriate.	Execution halted. All subsequent actions blocked until named owner explicitly clears the blast radius counter. No automatic recovery. Requires human decision on irreversible action audit.
Action Budget Exhausted	n_t >= T_budget: agent has taken more steps than the declared operational budget, indicating possible runaway loop or task scope creep.	Soft halt: execution paused, not terminated. owner_notify() with action log. Named owner may extend budget or terminate. Reversible actions preserved for audit.
Cascade Injection (Multi-Agent)	OED of downstream agent d_t+1 exceeds theta_halt within K_cascade steps of receiving output from the halted agent: upstream failure propagation detected.	Halt propagated to downstream agents. Pipeline frozen. All inter-agent message queues flushed. Requires full pipeline restart authorized by named owner.

TABLE II. HCOE Halt Precision and Recall by Class (300-Episode Simulation)

Halt Class	Precision	Recall	Notes
Action-Scope Breach	97%	94%	3 false negatives from tool aliasing: different API endpoint, same functional scope
Confidence Collapse	89%	91%	False positives from known low-confidence domains where C floor was not domain-adjusted
Blast Radius Exceeded	93%	88%	12% of irreversible actions were miscoded as reversible at pipeline registration
Action Budget Exhausted	98%	97%	Near-perfect; budget counter is deterministic and hard to misconfigure
Cascade (Multi-Agent)	82%	79%	Cascade propagation depth 1 missed in 21% of cases due to downstream compensation
Combined (all classes)	91.8%	89.8%	74% reduction in mean time-to-halt vs. reactive baseline over 10-cycle adaptation