

A Four-Mode Safety State Machine for Agentic AI Runtime Interruption

Subtitle: Exhaustive Safety Checks and Production Control-Plane Mapping

Status: Research Specification and Pre-Print Manuscript, v0.2.0

Date: 2026-05-05

Author: Reuben Bowlby (reuben@hummb1.io), HUMMBL Research Institute / HUMMBL, LLC

ORCID: 0009-0002-5620-1103

Companion Artifact: `kill_switch_model_check.py` (Exhaustive verification suite)

Abstract

Agentic AI systems need interruption mechanisms that are more expressive than binary on/off switches but still simple enough to audit. We present a four-mode safety state machine for agentic runtime interruption: `DISENGAGED`, `HALT_NONCRITICAL`, `HALT_ALL`, and `EMERGENCY`. The machine distinguishes normal operation, graceful degradation, full operational halt with safety tasks preserved, and human-reset-only emergency halt.

We specify the transition function and task-admission policy, verify three safety properties by exhaustive enumeration over 20 state/input pairs, and provide a model check whose transcript reports 20 generated states, 6 distinct states, complete graph depth 3, and zero errors. We then compose the state machine with adapter-level circuit breakers and compare the resulting control stack to production mechanisms including Kubernetes probes, Envoy circuit breakers, SageMaker deployment guardrails, and Vertex AI rolling deployments.

Our claim is scoped. The four-mode design is sufficient for the stated safety properties under a binary critical/non-critical task taxonomy, and it is minimal under an explicit pure-Mealy admission model where task admission depends only on state and task type. We do not claim that state-machine structure alone proves safe governance. Emergency irreversibility and auditability require supplementary identity, access-control, persistence, and append-only evidence mechanisms.

1. Problem

Interruptibility is often discussed as if it were a binary off switch. Production agentic systems do not fit that model. They coordinate heterogeneous tasks, external adapters, budget controls, safety monitors, and audit infrastructure. During an incident, it may be wrong to keep everything running, but it may also be wrong to stop audit logging, persistence, safety monitoring, or recovery workflows.

The design problem is therefore not “can the system be turned off?” The operational questions are:

1. Can ordinary work be paused without disabling safety infrastructure?
2. Can non-critical work be queued when recovery is plausible?
3. Can non-critical work be blocked when resumption would be unsafe?
4. Can all autonomous work be stopped in an emergency?
5. Can emergency exit be gated by a human or equivalent external authority?

This paper treats kill-switch design as a small, auditable state-machine problem rather than as a policy slogan.

2. Contribution

This manuscript makes four contributions:

1. It defines a four-mode state machine for agentic runtime interruption.

2. It states three formal safety properties: safe degradation, emergency override, and emergency irreversibility.
3. It verifies the stated machine by exhaustive Python enumeration and reachable-state checking over all state-input pairs.
4. It positions the design against production control mechanisms without claiming to replace them.

Explicit Non-Contributions:

1. We do not prove that four modes are universally necessary for every conceivable interruptibility design.
2. We do not prove that a state machine alone satisfies a complete governance framework.
3. We do not present independent, multi-organization production empirical telemetry.
4. We do not claim that placeholder production anecdotes establish safety effectiveness.

3. State Machine Formalization

Let the kill switch be a deterministic finite state machine:

$$K = (S, \Sigma, \delta, A, s_0)$$

where: - S is the finite set of operational modes, - Σ is the input alphabet, - $\delta : S \times \Sigma \rightarrow S \cup \{\perp\}$ is the deterministic transition function, - $A : S \times T \rightarrow \{\text{allow}, \text{queue}, \text{block}\}$ is the task-admission policy function, and - $s_0 \in S$ is the initial mode ($s_0 = \text{DISENGAGED}$).

The state set is:

$$S = \{\text{DISENGAGED}, \text{HALT_NONCRITICAL}, \text{HALT_ALL}, \text{EMERGENCY}\}$$

Mode	Meaning	Task Admission Policy
DISENGAGED	Normal operation	All tasks allowed.
HALT_NONCRITICAL	Graceful degradation	Critical tasks allowed; non-critical tasks queued.
HALT_ALL	Full operational halt	Critical tasks allowed; non-critical tasks blocked.
EMERGENCY	Human-reset-only halt	All autonomous tasks blocked.

The task taxonomy is partitioned into critical and non-critical subsets:

$$T = T_c \cup T_n, \quad T_c \cap T_n = \emptyset$$

where T_c denotes safety, logging, heartbeat, and audit tasks, and T_n denotes domain-level agent execution tasks.

The admission function $A(s, t)$ is defined:

$$A(s, t) = \begin{cases} \text{allow} & \text{if } s = \text{DISENGAGED} \\ \text{allow} & \text{if } s \in \{\text{HALT_NONCRITICAL}, \text{HALT_ALL}\} \wedge t \in T_c \\ \text{queue} & \text{if } s = \text{HALT_NONCRITICAL} \wedge t \in T_n \\ \text{block} & \text{if } s = \text{HALT_ALL} \wedge t \in T_n \\ \text{block} & \text{if } s = \text{EMERGENCY} \end{cases}$$

The input alphabet Σ consists of:

$$\Sigma = \{\text{engage}(\text{HALT_NONCRITICAL}), \text{engage}(\text{HALT_ALL}), \text{engage}(\text{EMERGENCY}), \text{disengage}, \text{human_reset}\}$$

Autonomous inputs comprise all transitions except `human_reset`. The input `human_reset` represents external, authenticated authority (such as an out-of-band operator, cryptographic hardware interlock, or governance quorum).

4. Safety Properties

P1: Safe Degradation

There exists a non-normal, non-emergency mode in which critical tasks continue and non-critical tasks do not execute:

$$\exists s \in S \setminus \{\text{DISENGAGED}, \text{EMERGENCY}\} : (\forall t_c \in T_c, A(s, t_c) = \text{allow}) \wedge (\forall t_n \in T_n, A(s, t_n) \neq \text{allow})$$

Operational meaning: The system can shed load or halt dangerous domain work without terminating audit, telemetry, or recovery mechanisms.

P2: Emergency Override

Emergency mode is reachable from every operational mode in at most one autonomous transition:

$$\forall s \in S, \quad \delta(s, \text{engage}(\text{EMERGENCY})) = \text{EMERGENCY}$$

Operational meaning: An emergency halt command immediately dominates all intermediate degradation or operational states.

P3: Emergency Irreversibility

No autonomous input can cause an exit from `EMERGENCY`:

$$\begin{aligned} \forall \sigma \in \Sigma_{\text{autonomous}}, \quad & \delta(\text{EMERGENCY}, \sigma) \in \{\text{EMERGENCY}, \perp\} \\ & \delta(\text{EMERGENCY}, \text{human_reset}) = \text{DISENGAGED} \end{aligned}$$

Operational meaning: Once an emergency halt is triggered, autonomous software cannot self-resume or bypass the shutdown. Only explicit, external human intervention can restore operational status.

5. Verified Results

Theorem 1 (Exhaustive Verification of the Four-Mode Machine)

The four-mode transition system satisfies Property 1 (Safe Degradation), Property 2 (Emergency Override), and Property 3 (Emergency Irreversibility) under complete state-input enumeration.

Verification Evidence: 1. Complete state-space enumeration over all $|S| \times |\Sigma| = 4 \times 5 = 20$ transition pairs via the companion verifier `kill_switch_model_check.py`. 2. Reachable-state model check generating 20 states, 6 distinct states, search depth 3, with zero state invariant violations.

```
=====
KILL SWITCH MODEL CHECK - Exhaustive Verification
=====
```

```
States: 4  Inputs: 5  Transitions: 20
```

```
[PASS] Transition completeness: all 20 (state, input) pairs defined
```

```
[PASS] P1 Safe Degradation: HALT_NONCRITICAL allows critical tasks and queues non-critical tasks
```

```
[PASS] P2 Emergency Override: EMERGENCY reachable from all states in <= 1 autonomous step
```

```
[PASS] P3 Emergency Irreversibility: no autonomous exit from EMERGENCY; human_reset reaches DISENGAGED
```

```
RESULT: ALL PROPERTIES VERIFIED
```

```
The four-mode kill switch satisfies P1, P2, P3.
```

```
Enumeration: 4 states x 5 inputs = 20 transition checks.
```

6. Scoped Minimality Proposition

Proposition 1 (Minimality Under Pure-Mealy Admission Semantics)

For deterministic state machines where task admission depends strictly on $(s, \text{task_type})$, no machine with fewer than four states can represent all four distinct admission regimes required by the operational control policy: 1. Normal operation (all tasks allowed), 2. Non-critical queueing with critical tasks preserved, 3. Non-critical blocking with critical tasks preserved, and 4. Global emergency halt (all tasks blocked).

Proof Sketch: - A 1-state machine cannot distinguish normal operation from any halt regime. - A 2-state machine distinguishes normal operation from full halt, but lacks an intermediate state for graceful degradation where critical tasks are preserved. - A 3-state machine provides normal operation, emergency halt, and one degraded state. Under pure-Mealy admission semantics, the admission policy for non-critical tasks in that degraded state must map to exactly one action: either `queue` or `block`. It cannot simultaneously support queueing under transient degradation and blocking under non-recoverable operational halts. - Therefore, at least 4 discrete states are strictly necessary. ■

7. Liveness and Recovery

Safety guarantees what bad states will not occur; liveness defines how the system recovers.

L1: Non-Emergency Recoverability

From `HALT_NONCRITICAL` and `HALT_ALL`, the normal operating state `DISENGAGED` is reachable via an authorized autonomous `disengage` command.

L2: Emergency Recoverability

From `EMERGENCY`, `DISENGAGED` is reachable if and only if authenticated `human_reset` is presented.

8. Circuit-Breaker Composition

The kill switch operates globally across the task scheduler. In contrast, circuit breakers operate locally at adapter, tool, and service-boundary call paths. The two mechanisms form a complementary, defense-in-depth control hierarchy.

Mechanism	Operational Scope	Granularity	Failure Action
Kill Switch	System-level task scheduler	Task class (T_c vs. T_n)	Queue, block, or emergency halt
Circuit Breaker	External adapter / I/O boundary	Dependency endpoint / call path	Trip open upon consecutive error thresholds

The composition rule guarantees that an outbound side effect executes if and only if both the global kill switch and the local circuit breaker permit it:

$$\text{CallAllowed}(t, \text{adapter}) \iff (A(s, t) = \text{allow}) \wedge (\text{CircuitBreakerStatus}(\text{adapter}) = \text{CLOSED})$$

When **EMERGENCY** is engaged, all local circuit breakers are forced open, preventing unmonitored sideband egress even if an anomalous agent thread bypasses task scheduling.

9. Production Control-Plane Comparison

System	Control Primitive	Architectural Similarity	Key Difference from Four-Mode Agent Kill Switch
Kubernetes Probes	Liveness, readiness, and startup probes	Health status dynamically routes traffic or triggers container restarts	Container lifecycle scope; lacks semantic agent task criticality distinction
Envoy Circuit Breakers	Outlier detection, max connections, pending requests	Fails closed at network service boundaries	Transport/RPC layer; does not model agent cognitive task intention
AWS SageMaker Guardrails	Canary, blue/green, linear traffic shifts with CloudWatch alarms	Automated rollback upon threshold breach	Deployment time; does not intercept runtime task cascades
GCP Vertex AI Deployments	Rolling model substitution and health gating	Manages staged traffic cutover	Infrastructure deployment scope rather than dynamic runtime orchestration
Four-Mode Kill Switch	Task-admission state machine	Deterministic gating of critical vs. non-critical workloads	First-class multi-agent runtime coordination with emergency human interlock

10. Governance & AI Safety Framework Alignment

The four-mode kill switch maps directly to established international AI safety guidelines:

Governance Objective	Kill Switch Contribution	Residual Host Requirement
Human Oversight (NIST AI RMF GOVERN)	EMERGENCY mode requires out-of-band human reset	Authenticated identity and quorum authorization protocol

Governance Objective	Kill Switch Contribution	Residual Host Requirement
Risk Response & Incident Handling	Graduated degradation prevents catastrophic all-or-nothing shutdowns	Defined operational escalation procedures and runbooks
Accountability & Traceability	State transitions emit structured governance events	Append-only tamper-evident event log (e.g. Cognitive Ledger)
Safety Testing & Assurance	Exhaustive state verification via automated model checker	Periodic regression testing under fault-injection scenarios

11. Limitations

1. **Binary Criticality Taxonomy:** The model assumes tasks are partitioned into T_c and T_n . Fine-grained task priority lattices will require multi-dimensional admission matrices.
2. **Host Environment Confinement:** Property 3 (Emergency Irreversibility) assumes the host operating system enforces process isolation; a rogue agent with root kernel access could tamper with in-memory state variables.
3. **Pure-Mealy Scoping:** The minimality proof applies strictly to state-admission functions without hidden runtime parameter state.

12. Conclusion

A four-mode safety state machine provides an optimal balance between formal auditability and operational resilience for multi-agent AI systems. By distinguishing normal execution, queueing degradation, non-critical halting, and human-gated emergency shutdown, autonomous fleets can safely shed non-critical load and isolate faults without severing the telemetry and audit infrastructure necessary to diagnose and recover from incidents.

References

1. Kubernetes Documentation. *Configure Liveness, Readiness and Startup Probes*. <https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-startup-probes/>
2. Envoy Project. *Circuit Breaking Architecture*. https://www.envoyproxy.io/docs/envoy/latest/intro/arch_overview/upstream/circuit_breaking
3. Amazon Web Services. *Amazon SageMaker Deployment Guardrails*. <https://docs.aws.amazon.com/sagemaker/latest/dg/clearly-defined-guardrails.html>
4. Google Cloud. *Vertex AI Rolling Deployments*. <https://docs.cloud.google.com/vertex-ai/docs/predictions/rolling-deployment>
5. National Institute of Standards and Technology (NIST). *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*. NIST AI 100-1, 2023. <https://doi.org/10.6028/NIST.AI.100-1>
6. Bowlby, R. *Typed Governance Tuples: Portable Interchange and Reference Mediation for Auditable Agentic Systems*. Zenodo, 2026. <https://doi.org/10.5281/zenodo.23001537>
7. Bowlby, R. *A Typed Delegation-Governance Tuple Profile for Multi-Agent Runtime Control*. Zenodo, 2026. <https://doi.org/10.5281/zenodo.23001327>