

# The Capability–Availability Mismatch: Three-State Tool Semantics and Forced-Halt Conditions in LLM Agent Protocols

## The Capability–Availability Mismatch: Three-State Tool Semantics and Forced-Halt Conditions in LLM Agent Protocols

### Abstract

---

Agent protocols written for large language models (LLMs) typically assume a binary world: a tool is either available to the agent or it is not. Real deployments exhibit a third state: the capability exists in the toolset but is contextually disabled by policy, budget, or routing. We formalize this capability–availability mismatch. Tool state is modeled as a pair (capability, per-context permission), protocol compliance is defined as a predicate over these states, and we ask which protocol designs remain well-defined when the third state occurs. Three results follow. First, a combinatorial result: a binary-state protocol governing  $k$  permission-gated tools is specification-complete on only a  $(2/3)^k$  fraction of the true state space; for  $k = 5$  this is  $32/243 \approx 0.132$ , and for  $k = 10$  the three-state space is  $59049/1024 \approx 57.7$  times larger than the binary model represents. Second, a computability result: deciding whether an arbitrary protocol loop halts with a compliant action in the disabled state is undecidable, by a Rice's-theorem reduction in the style used for inner alignment. Third, an economic projection: under stated assumptions (capability rate 0.90, permission rate 0.70), the mismatch state occurs with probability 0.27 per gated tool invocation, yielding a signaling break-even threshold of  $s^* = 0.27 c_t$  per task, where  $c_t$  is the unit cost of a wasted task. The practical consequence is a design rule: agent instruction languages must signal the disabled state explicitly, or they force agents to choose between mandatory halting and silent rule violation.

### 1. Introduction

---

Teams deploying LLM agents in production face a failure mode that no amount of model improvement removes. An agent is instructed, in natural language, to "use the live search tool when answering factual questions" and to "never simulate or fabricate tool output." The tool exists in the agent's toolset, the model can call it, and the calling code works. But in a particular context — a rate-limited session, a tenant with the tool switched off by policy, a routing tier where the tool's budget is exhausted — the platform has disabled the tool without removing it from the toolset. The agent now sits in a catch-22 constructed entirely by the protocol's wording: if it honestly self-assesses, it concludes the tool is "not available," and the mandate-plus-prohibition pair forces it to halt a task it could in principle execute; if it proceeds, it either fabricates a "live" fetch (violating the prohibition) or claims compliance it cannot verify. The cost is concrete: tasks halt that should complete, or silent rule violations accumulate that no compliance checker catches, because the checker shares the same binary assumption the protocol text does.

This matters to three groups today. Protocol and framework authors who write system prompts and tool schemas must decide what their instructions mean in the disabled state, and currently have no vocabulary for it. Platform engineers who disable tools per tenant or per budget envelope must predict agent behavior after the switch, and currently cannot, because the agent's decision procedure was never specified for that state. Auditors and evaluators who score agent compliance must decide whether a fabricated fetch in the disabled state is a model failure or a specification failure, and currently have no principled answer. If the analysis in this paper holds, a small change in protocol design — explicit signaling of the disabled state, and compliance predicates defined over three states rather than two — removes the catch-22 by construction rather than by prompt tuning.

The halting behavior of LLM agents is already known to be a first-order cost driver, which makes any *forced* halt condition worth formalizing. Retrieval-augmented search agents exhibit the mirror-image pathology: a 3B Self-Ask agent runs to its search budget on 78–88% of questions instead of halting on its own [3]. Reasoning models overthink by a wide, stable margin: a probe-based confidence estimate saturates at a median of about 1,650 think-tokens while the same rollouts run to a median of about 3,700, a matched factor of 2.24 [4]. These works study agents that fail to stop when they should. This paper studies the complementary pathology: agents forced to stop, or forced to violate, by a specification that cannot represent the state they are in. The mismatch is structurally analogous to operator mismatch in computational imaging, where a reconstruction pipeline that assumes a forward operator differing from physical reality in eight parameters loses 20.58 dB [6]: in both cases the deployed artifact is evaluated against a state model that the environment does not honor.

We treat the question at two levels. At the specification level, we give a three-state semantics for tool availability, define protocol compliance as a predicate over that semantics, and characterize which protocol designs are well-defined in all three states. At the computability level, we show that a natural decision problem about protocol loops — whether the loop halts with a compliant action in the disabled state — is undecidable, using the reduction strategy that establishes the undecidability of inner alignment via Rice's theorem [7]. Undecidability results of this shape have precedent in physical domains: the quantum measurement occurrence problem is undecidable for  $K = 9$  outcomes and  $d = 15$  dimensions even though its classical analogue is decidable [1], and supersymmetry breaking in 2d  $\mathcal{N}=(2,2)$  Wess-Zumino models is undecidable by encoding arbitrary Turing machines into the class of Lagrangians [5]. In each case the lesson is the same and applies here: a specification that looks complete can silently inherit the halting problem.

The contributions are:

- A three-state formalization of tool availability (capability vs. per-context permission) and a definition of protocol compliance over that semantics, with a well-definedness criterion for protocol designs (Section 3).
- A combinatorial derivation showing a binary protocol with  $k$  permission-gated tools is specification-complete on a  $(2/3)^k$  fraction of the true state space, computed as  $32/243 \approx 0.132$  for  $k = 5$ , and a state-space ratio of  $59049/1024 \approx 57.7$  for  $k = 10$  (Section 4.1).
- An undecidability proof for the protocol compliance-halting problem, by contradiction following the Rice's theorem reduction of [7] (Section 4.2).
- A derivation of the mismatch-state probability  $q = p_C(1 - p_P) = 0.27$  under stated assumptions, and a signaling break-even threshold  $s^* = 0.27 c_t$  per task (Section 4.3).
- A design rule for agent instruction languages — explicit disabled-state signaling plus three-state compliance predicates — and the smallest experiment that would falsify or de-risk it (Sections 5 and 7).

## 2. Background and Related Work

**Undecidability in measurement and physics.** The quantum measurement occurrence problem asks whether, in a tree of repeated nonprojective measurements with  $K$  outcomes and Kraus operators  $A_1, \dots, A_K$ , there exists an outcome path with zero probability. The problem is undecidable for  $K = 9$  and  $d = 15$ , as a consequence of the undecidability of the matrix mortality problem for  $3 \times 3$  integer matrix semigroups generated by 8 matrices, itself reduced from the post correspondence problem [1]. Two features of that result transfer to our setting. The classical analogue (a stochastic-matrix version of the same question) is decidable, so the gap between the modeled dynamics and the richer dynamics is exactly what introduces undecidability; and mild restrictions, such as trace-preserving channels bounded via the quantum Wielandt theorem, restore efficient decidability. Our three-state semantics plays the role of the richer dynamics: protocols specified over two states face decision questions that the two-state model itself does not contain. The encoding strategy is shared across undecidability results in physics: undecidability of supersymmetry

breaking in 2d  $\mathcal{N}=(2, 2)$  Wess-Zumino models is obtained by devising a way to encode arbitrary Turing machines into the class of systems under consideration [5], and the same work's discussion of proof verifiers — programs that check purported proofs of a statement in a theory — anticipates the verification-based reading of compliance we adopt here.

**Undecidability of alignment properties.** The closest formal antecedent to our computability result is the proof that the inner alignment problem — whether an arbitrary AI model satisfies a non-trivial alignment function  $J_v(i, o)$  of its outputs given its inputs — is undecidable, established by Rice's theorem and by an explicit contradiction construction: assuming a Turing machine  $M_P$  that decides the property, one builds a Turing machine  $M_H$  that decides the halting problem, contradicting its undecidability [7]. The same work delineates the escape route we later rely on for practice: while the property cannot be ascertained for arbitrary systems, it can be ascertained for an enumerable set of architecturally designed systems, built from an initial set of finite models and operations that preserve the property. Section 6 uses this distinction to separate what is undecidable in principle (arbitrary protocol text) from what is decidable by construction (protocols whose compliance predicates are syntactically restricted).

**Halting and stopping in deployed agents.** Empirically, agent halting is already a measurable, expensive behavior. HALT frames stopping as evidence coverage rather than generator confidence: a verifier matches every expected reasoning hop to cumulative evidence, and the agent stops when coverage is complete. With gold claims on a Self-Ask 3B agent, HALT reduces loop counts by 37% on 2Wiki, 45% on HotpotQA, and 20% on MuSiQue, with exact-match changes between  $-0.9$  and  $+1.5$  percentage points; the HotpotQA non-inferiority test fails with a 95% CI of  $[-2.6, +0.8]$  pp crossing the  $-2$  pp margin, and under open-corpus retrieval shift the fire rate drops from 67.6% to 6.4% on HotpotQA and to 2.4% on Bamboogle [3]. The authors explicitly scope their claims to controlled closed-pool retrieval. Two points bear on our formalization: halting policies are measurable and tunable, and a halting policy's behavior degrades predictably when the environment's retrieval state differs from the state the policy was specified over — a state-mismatch effect, not a model-capability effect. The Halt Vector work internalizes a causal stopping direction at layer 18 into the weights, yielding a per-problem self-halt trained from 24 problems that removes about 24% of thinking at held accuracy; notably, outcome-level length penalties only trim the longest tails and do not move the internal answerable point, and the stop-and-commit tokens are about 0.34% of the token loss, so plain fine-tuning leaves length unchanged [4]. This establishes that halting behavior is causally steerable but not specifiable by a global scalar — supporting our claim that halting semantics must be represented structurally in the protocol, not incentivized loosely.

**Mismatch between assumed and actual state.** InverseNet benchmarks operator mismatch in compressive imaging: when the assumed forward operator deviates from physical reality in eight parameters, EfficientSCI loses 20.58 dB; under mismatched operators all methods converge to 20.40–21.88 dB on CASSI, erasing the roughly 10 dB ideal-condition advantage of deep learning over classical methods; CACTI suffers degradations up to 20.6 dB; and mask-oblivious architectures such as HDNet show zero oracle recovery ( $\Delta_{\text{rec}} = 0.00$  dB) while operator-aware ones such as PnP-HSICNN recover with ratio  $\rho = 56.8\%$  [6]. The structural lesson for agent protocols is direct: an artifact specified against a state model the environment violates fails catastrophically and non-uniformly, and recovery requires the specification to be aware of the true state — which is precisely what disabled-state signaling provides.

**Capability as a structural property and predicates over distinguished coordinates.** The ontological re-entry framework defines a distinction space  $D_N = (S_N, d_u)$  with  $S_N = \{0, 1\}^N$ , realizes it as a  $b$ -ary tree of depth  $L$  with  $N = L \log_2 b$  and  $|S_N| = 2^N$ , and admits as existence predicates exactly the subsets definable by finite Boolean formulas over the  $N$  base coordinates, of which there are  $2^{2^N}$  extensionally distinct ones [8]. We reuse this move: an agent's context state is a distinction record, and a protocol's compliance predicate is admissible only if it is definable over the coordinates the protocol actually distinguishes. A binary protocol does not distinguish the disabled

state, so predicates that depend on it are inadmissible in its vocabulary — the formal core of the catch-22. This also supports our separation of capability (what the toolset can do, a structural property of the deployment) from permission (what a context allows, an external constraint): conflating them is the category error our three-state semantics repairs.

**Systems-level efficiency.** High-concurrency LLM serving is bottlenecked by KV cache memory overhead, and structural changes are what move concurrency: an adaptive GQA-to-MLA transition limits Llama3-8B perplexity degradation from 6.1 to 12.9 versus 25.8 for the baseline, a 65% reduction in degradation [2]. This is the systems-side analogue of our protocol-side claim: deployment-level pathologies are addressed by changing the serving or protocol structure, not by prompt text.

### 3. Methods

---

#### 3.1 Three-state tool semantics

Fix a finite set of tools  $T = \{t_1, \dots, t_n\}$  and a set of contexts  $C$ . For a tool  $t$  and context  $c$  we define two independent bits:

- $\text{Cap}(t) \in \{0, 1\}$ : the tool is present in the agent's toolset and the calling path is functional. This is a structural property of the deployment.
- $\text{Perm}(t, c) \in \{0, 1\}$ : policy, budget, or routing permits invoking  $t$  in context  $c$ .

The *effective availability* is the conjunction:

$$\text{Avail}(t, c) = \text{Cap}(t) \wedge \text{Perm}(t, c).$$

Each tool–context pair therefore occupies one of three *distinguishable* states, which we name:

$$\sigma(t, c) \in \{\text{ABSENT}, \text{ENABLED}, \text{DISABLED}\},$$

with  $\text{ABSENT} = (\text{Cap} = 0, \text{Perm} = 0)$ ,  $\text{ENABLED} = (1, 1)$ , and  $\text{DISABLED} = (1, 0)$ . The state  $(0, 1)$  is excluded by construction: permission without capability is not a state the platform can occupy, since a tool that does not exist cannot be permitted. (We assume platforms do not advertise permissions for nonexistent tools; if they do, a fourth state appears and only strengthens the mismatch results below.)

The binary model underlying current protocol text collapses **ABSENT** and **DISABLED** into a single "unavailable" value. The distinction-space framing makes the cost of that collapse precise: a distinction is a binary partition of the observer's state space that the observer can reliably re-identify, and admissible predicates are exactly those definable by finite Boolean formulas over the distinguished coordinates [8]. A protocol whose vocabulary distinguishes only **Avail** has one coordinate; a three-state protocol distinguishes **Cap** and **Perm** separately and has two. Predicates that need to condition on **Perm** alone — for example, "report the tool as disabled and continue without it" — are inadmissible in the binary vocabulary, not because they are hard to evaluate but because the vocabulary cannot express them.

#### 3.2 Protocols and compliance

A *protocol*  $\Pi$  is a pair  $(\iota, \chi)$  where  $\iota$  is the instruction text handed to the agent and  $\chi$  is the compliance predicate used by an external checker (which may be the agent itself in a self-audit step). We restrict attention to the clause pattern that motivates this paper, which we call a *mandate–prohibition pair*:

- **Mandate**( $t$ ): the protocol requires that task-relevant facts be established by a live invocation of  $t$ .
- **Prohibit**( $t$ ): the protocol forbids simulating, fabricating, or asserting the content of  $t$ 's output without a live invocation.

Given a state  $\sigma(t, c)$  and an agent action trace  $\tau$ , the compliance predicate evaluates to  $\chi(\Pi, \sigma, \tau) \in \{0, 1\}$ . A protocol is *well-defined* on a state  $s$  if, for every trace  $\tau$  available to the agent in state  $s$ , exactly one of the following holds and the protocol text determines which: (i)  $\tau$  is compliant, (ii)  $\tau$  is non-compliant, or (iii) the protocol prescribes halting as the compliant action. A protocol is *well-defined simpliciter* if it is well-defined on every state in its state semantics.

### 3.3 The catch-22 as a specification defect

In state **ENABLED**, the mandate–prohibition pair is satisfiable: invoke  $t$  live. In state **ABSENT**, halting with an explicit "tool unavailable" report is the compliant action under any reasonable reading. In state **DISABLED**, the pair is unsatisfiable under the binary reading:

- If the agent reads "unavailable" (the only unavailable-value the binary vocabulary has) and halts, it halts a task that is executable in principle — a *forced halt*. The mandate is violated in spirit: the protocol demanded a live fetch and the fetch was possible.
- If the agent reads its own capability as availability and attempts the call, the call fails or is refused; any subsequent assertion of the tool's content is fabricated — a *silent violation* of  $\text{Prohibit}(t)$ .
- If the agent fabricates a successful "live" fetch without calling, it violates  $\text{Prohibit}(t)$  directly.

Formally: under the binary semantics, no trace in state **DISABLED** satisfies both  $\text{Mandate}(t)$  and  $\text{Prohibit}(t)$ , so  $\chi$  is forced either to declare every trace non-compliant (the task can never succeed, a de facto forced halt) or to declare some fabricating trace compliant (the checker is unsound). Under the three-state semantics, a third trace class exists — report **DISABLED**, degrade gracefully, continue without  $t$  — and the protocol author can define  $\chi$  to accept it. The catch-22 is therefore not a model failure; it is a defect of the specification's state vocabulary, in the same sense that a reconstruction pipeline specified against a wrong forward operator fails when the environment deviates from the assumption [6].

### 3.4 Protocol classes

We classify protocols by their state semantics and their treatment of **DISABLED**:

- **Class B (binary):**  $\iota$  and  $\chi$  reference only **Avail**. Well-defined on **ABSENT** and **ENABLED**; not well-defined on **DISABLED**.
- **Class S (signaling):** the platform emits an explicit **DISABLED** signal,  $\iota$  instructs the agent to honor it, and  $\chi$  accepts the degrade-and-continue trace in state **DISABLED**. Well-defined on all three states.
- **Class R (restricted):** protocols whose compliance predicates are drawn from a syntactically restricted, property-preserving grammar, in the spirit of the axiomatically enumerable systems for which alignment properties are ascertainable [7]. Well-definedness for Class R is decidable by construction; Section 4.2 shows why the restriction is necessary.

### 3.5 Decision problems

We define the central decision problem:

$\text{PCH}(\Pi, c)$  : does the agent loop under protocol  $\Pi$  halt with a compliant action in context  $c$ ?

The agent loop is modeled as a Turing machine  $M_\Pi$  that, on input encoding the context  $c$ , iterates: assess state, select action, execute or halt. The question  $\text{PCH}(\Pi, c)$  is then a question about the language of  $M_\Pi$ , and Section 4.2 applies the reduction strategy used to prove inner alignment undecidable [7], mirroring the encode-a-Turing-machine-into-the-system-class strategy used across undecidability results in physics [1, 5].

## 4. Analysis

---

### 4.1 State-space combinatorics

**Input numbers and their sources.** The state count 3 per permission-gated tool follows from the three-state semantics of Section 3.1 (derived, not measured). The tool counts  $k = 5$  and  $k = 10$  are our own illustrative assumptions, chosen to span a small single-agent toolset and a mid-size one; we state them as assumptions.

**Derivation 1 (state-space ratio).** Under the binary semantics, a context with  $k$  permission-gated tools has  $2^k$  distinguishable availability configurations. Under the three-state semantics it has  $3^k$ . For  $k = 10$ :

$$2^{10} = 1024, \quad 3^{10} = 59049, \quad \frac{3^{10}}{2^{10}} = \frac{59049}{1024} = 57.669 \approx 57.7.$$

The three-state state space is 57.7 times larger than the binary model represents. Every configuration in the difference —  $59049 - 1024 = 58025$  configurations — contains at least one tool in state **DISABLED**, and is a configuration on which a Class B protocol has no specified behavior.

**Derivation 2 (well-defined fraction).** A Class B protocol is well-defined exactly on configurations where every permission-gated tool is in  $\{\text{ABSENT}, \text{ENABLED}\}$ , i.e., where no tool is disabled. If all three states of each tool are equally represented (an assumption we state explicitly and revisit in Section 4.3 with a non-uniform alternative), the well-defined fraction is

$$f(k) = \left(\frac{2}{3}\right)^k.$$

For  $k = 5$ :

$$f(5) = \frac{2^5}{3^5} = \frac{32}{243} = 0.131687 \dots \approx 0.132.$$

So under uniform state representation, a binary protocol governing five permission-gated tools is specification-complete on about 13.2% of the state space it actually faces; on the remaining  $1 - 32/243 = 211/243 \approx 0.868$  fraction, its behavior is unspecified. The uniform-representation assumption is conservative in one direction and optimistic in another: if disabled states are rare, the unspecified fraction shrinks; if platforms disable tools per tenant or per budget envelope — the common cases motivating this paper — disabled states are common and the unspecified fraction grows.

### 4.2 Undecidability of the compliance-halting problem

**Claim.** PCH is undecidable over the class of arbitrary protocols  $\Pi$ .

**Proof sketch, by contradiction, following the reduction strategy of [7].** The property decided by PCH is a property of the language of the Turing machine  $M_\Pi$ : it holds of  $M_\Pi$  exactly when  $M_\Pi$ 's execution on context  $c$  reaches a halting configuration whose trace satisfies  $\chi$ . This is a semantic property of the machine's behavior, not of its source text. It is non-trivial: there exist protocols for which the property holds (a protocol whose loop always halts with a compliant report in every context) and protocols for which it fails (a protocol whose loop never halts, or halts only with a non-compliant trace). Suppose a Turing machine  $M_P$  decides PCH for arbitrary  $\Pi$  and  $c$ . Following the construction of [7], one then composes  $M_P$  with a trivial wrapper machine  $M_H$  whose behavior on an arbitrary machine-and-input pair is engineered so that  $M_P$ 's answer to PCH on the wrapper is exactly the answer to whether the original pair halts. Then  $M_H$  decides the halting problem, contradicting its undecidability. Hence no such  $M_P$  exists, and PCH is undecidable.  $\square$

### 4.3 Mismatch-state probability and the signaling break-even threshold

**Input numbers and their sources.** Two rates are needed to weight the combinatorics of Section 4.1 by how often the disabled state actually occurs. We assume, as stated assumptions rather than measurements:  $p_C = 0.90$ , the probability that a gated tool is capable (present and functional) in a given deployment context, and  $p_P = 0.70$ , the probability that policy, budget, or routing permits the tool in that context. These values are chosen to represent a mature toolset with per-tenant and per-budget permission gating; Section 6 discusses sensitivity to both.

**Derivation 3 (mismatch probability).** The mismatch state **DISABLED** is the conjunction "capable but not permitted." Under the independence assumption stated in Section 3.1 (capability is a structural property of the deployment; permission is a context-level gate, and we assume the two are drawn independently),

$$q = \Pr[\text{DISABLED}] = p_C (1 - p_P) = 0.90 \times (1 - 0.70) = 0.90 \times 0.30 = 0.27.$$

So under these assumptions, 27% of gated tool invocations land in the state on which a Class B protocol has no specified behavior. For a task that touches  $k$  gated tools, the probability that at least one invocation is mismatched is

$$q_{\text{task}}(k) = 1 - (1 - q)^k.$$

For  $k = 5$ :

$$q_{\text{task}}(5) = 1 - 0.73^5 = 1 - 0.2073071593 = 0.7926928407 \approx 0.793.$$

For  $k = 10$ :

$$q_{\text{task}}(10) = 1 - 0.73^{10} = 1 - 0.0429762583 = 0.9570237417 \approx 0.957.$$

The step  $0.73^5$ :  $0.73^2 = 0.5329$ ;  $0.73^4 = 0.5329^2 = 0.28398241$ ;  $0.73^5 = 0.28398241 \times 0.73 = 0.2073071593$ . Then  $0.73^{10} = 0.2073071593^2 = 0.0429762583$ . Under the stated rates, a five-tool task almost certainly encounters at least one disabled state, and a ten-tool task does so with near-certainty. This is the quantitative form of the claim that per-tenant gating makes the disabled state common, not exotic.

**Derivation 4 (signaling break-even threshold).** Let  $c_t$  be the unit cost of a wasted task — a task that halts when it could have completed, or that completes with an undetected fabricated fetch. In the mismatch state, a Class B protocol produces a wasted task with probability  $q$  per gated invocation (the agent either halts or violates; both outcomes are wasted under the definition above). A Class S protocol pays a signaling cost  $s$  per task — the additional protocol text, schema field, and compliance-check logic that carries the **DISABLED** signal — and in exchange converts the mismatch outcome into a degrade-and-continue trace, which is not wasted. The Class S protocol breaks even when its total expected cost equals the Class B expected cost:

$$s = q c_t.$$

With  $q = 0.27$ :

$$s^* = 0.27 c_t \text{ per task.}$$

Any signaling mechanism whose per-task cost is below  $0.27 c_t$  is cheaper than living with the mismatch. The threshold scales linearly in  $q$ : at  $q = 0.10$  it is  $0.10 c_t$ , at  $q = 0.50$  it is  $0.50 c_t$ , so the conclusion "signal if signaling is cheap relative to wasted tasks" is robust across a wide band of the assumed rates. Because  $c_t$  is deployment-specific (it includes compute, latency, and downstream cost of silent violations), we leave it symbolic; the result is the ratio, not an absolute figure.

## 5. Results

---

**R1 (State-space coverage, Section 4.1).** A binary-state protocol governing  $k$  permission-gated tools is specification-complete on a  $(2/3)^k$  fraction of the true three-state space. Computed values:  $f(5) = 32/243 \approx 0.132$ ; the  $k = 10$  state-space ratio is  $3^{10}/2^{10} = 59049/1024 \approx 57.7$ , with  $59049 - 1024 = 58025$  configurations containing at least one disabled tool and therefore unspecified under the binary model. These are exact combinatorial consequences of the three-state semantics, not measurements.

**R2 (Undecidability, Section 4.2).** The protocol compliance-halting problem PCH — whether an agent loop under an arbitrary protocol halts with a compliant action in a given context — is undecidable, by a Rice's-theorem reduction in the style of [7]. The consequence for design is that well-definedness cannot be bought by testing or by better models; it must be built into the protocol's syntactic class (Class R of Section 3.4).

**R3 (Mismatch probability, Section 4.3).** Under the stated assumptions  $p_C = 0.90$  and  $p_P = 0.70$ , the disabled state occurs with probability  $q = 0.27$  per gated invocation. Per task, at least one mismatch occurs with probability  $q_{\text{task}}(5) \approx 0.793$  for  $k = 5$  and  $q_{\text{task}}(10) \approx 0.957$  for  $k = 10$ . These are projections conditional on the stated rates and independence; they are not empirical measurements.

**R4 (Break-even threshold, Section 4.3).** The signaling break-even threshold is  $s^* = 0.27 c_t$  per task, where  $c_t$  is the deployment-specific unit cost of a wasted task. The threshold is linear in  $q$  and therefore robust in direction across the plausible range of the assumed rates.

**R5 (Design rule).** Combining R1–R4: agent instruction languages must (i) distinguish capability from permission, (ii) signal the **DISABLED** state explicitly, and (iii) define compliance predicates over all three states, accepting the degrade-and-continue trace. A protocol satisfying these conditions is well-defined on the full state space by construction (Section 3.4, Class S), and restricting its compliance predicate to a syntactically decidable grammar (Class R) sidesteps the undecidability of R2, in the same way that [7] restricts the class of systems for which alignment properties remain ascertainable.

## 6. Discussion

---

**What the results do and do not claim.** R1 is exact arithmetic given the three-state semantics; its only assumption is that the three states are the right state model, which is an empirical question about real platforms. R3 and R4 are projections resting on assumed rates; R2 is a formal result whose scope is bounded by its modeling choices. We discuss each in turn.

### Limitations list.

- *Three-state semantics is an idealization.* We assume platforms do not advertise permissions for nonexistent tools, excluding the  $(0, 1)$  state. If they do, a fourth state appears; the coverage fractions in R1 worsen (the well-defined fraction becomes  $(2/4)^k$  under uniform representation, e.g.  $f(5) = (1/2)^5 = 1/32 = 0.03125$ ), so the direction of the error favors our claim, but the exact numbers change.
- *Independence of capability and permission.* Derivation 3 assumes  $p_C$  and  $p_P$  are independent. If platforms disable tools precisely where capability is doubtful, the mismatch probability falls below 0.27; if budget exhaustion hits capable tools (the motivating case), it holds or rises. The break-even threshold  $s^*$  is linear in  $q$ , so errors in  $q$  propagate proportionally, not catastrophically.
- *Uniform state representation in R1.* The  $(2/3)^k$  fraction assumes all three states equally likely per tool. With rare disabled states the unspecified fraction shrinks toward zero; with per-tenant gating at the assumed rates it grows toward the  $q_{\text{task}}$  values of R3. The result is a bound structure, not a fixed number.
- *The Turing-machine model of the agent loop.* R2 models the agent loop as a Turing machine whose halting behavior is a semantic property. Real agent loops run under token budgets, timeouts, and retry limits, which make

every concrete execution halting by fiat. Undecidability therefore binds the *specification and verification* layer — no checker can decide compliance-halting for arbitrary protocol text — not any single bounded run. This is the same scoping [7] accepts: the impossibility is about arbitrary systems, and the escape is a restricted, enumerable class.

- *The wasted-task cost  $c_t$  is left symbolic.* We derive a ratio, not a dollar figure; no deployment-specific cost data are assumed.
- *No empirical measurement.* The paper contains no agent experiments. The rates  $p_C, p_P$  are assumptions chosen to be plausible for per-tenant gating, and every number derived from them inherits that status.

**Failure modes.** A Class S protocol can fail in three ways. First, the signal can be wrong or stale: a platform that reports **DISABLED** while the tool is in fact callable (or vice versa) reintroduces a mismatch one level up, between the signal and reality — the operator-mismatch structure of [6] applied to the signaling channel itself. Second, the degrade-and-continue trace can be abused: an agent that prefers the easy path may treat ambiguous cases as disabled to avoid the mandate, so the compliance checker must verify that the disabled report corresponds to an actual signal. Third, three-state compliance predicates add specification surface; a poorly written three-state checker can be as underspecified as the binary one.

**What would falsify the main result.** The central practical claim is that the disabled state occurs often enough that binary protocols are systematically underspecified on it, and that explicit signaling removes the forced-halt/violation dilemma. Two observations would falsify it. (i) An empirical measurement showing that the mismatch state is negligible in production — e.g., that gated tools are almost always either fully present or fully absent, so  $q \ll 0.05$  and  $q_{\text{task}}$  stays small for realistic  $k$  — would collapse R3 and shrink the break-even threshold to the point where signaling is not worth its cost. (ii) A controlled experiment in which agents facing a genuine disabled state under binary protocols do *not* exhibit elevated forced halts or fabricated fetches relative to three-state-signaled protocols would falsify the claim that the catch-22 changes behavior. Conversely, the undecidability result R2 would be falsified as stated only by exhibiting a decision procedure for **PCH** over arbitrary protocols, which would contradict the halting problem's undecidability; the realistic failure mode for R2 is the modeling objection that real protocol loops are so bounded that the idealization does no work — a objection we partially accept and have scoped above.

**Open questions.** What distribution of  $(p_C, p_P)$  do real multi-tenant platforms exhibit? Do compliance checkers in current frameworks silently share the binary assumption, and can a three-state checker be added without protocol changes? Does the Class R grammar restriction (decidable well-definedness) cost anything in expressiveness that matters for real protocol clauses? These require measurement and are outside this paper's scope.

## 7. Practical Implications and Engineering Pathways

---

**(a) Who would use the result and what changes.** Three named users. *Protocol and framework authors* change the tool schema and instruction language: instead of a binary available/unavailable field, they expose capability and permission as separate fields and add an explicit disabled-state signal that the agent is instructed to honor, with a defined degrade-and-continue action. *Platform engineers* change what they emit when they gate a tool: a machine-readable **DISABLED** signal delivered in the tool listing or error channel, rather than a bare call failure the agent must interpret. *Auditors and evaluation teams* change their compliance predicates to range over three states, so that a degrade-and-continue trace in the disabled state scores as compliant rather than as a miss. If the result holds, the decision "what should the agent do when the tool is switched off?" moves from an emergent property of prompt wording to a specified, checkable protocol clause.

**(b) Current readiness.** This work is purely analytical and has not been peer reviewed: by the NASA Technology Readiness Level definitions it sits at TRL 1–2 — basic principles observed and reported, and concept formulated — and analytical work does not exceed TRL 3. No implementation, prototype, or empirical result exists in this paper.

Claims of deployment readiness would be unsupported.

**(c) The smallest next experiment.** Instrument a single agent framework with a controllable gate: pick one tool, one agent, and one task suite; run three conditions — (B) tool removed from toolset, (B') tool present but calls fail with a bare error (the binary disabled state), (S) tool present with an explicit disabled signal and a permitted degrade-and-continue action. Measure, per condition: forced-halt rate (tasks ending without an answer although the task was completable without the tool), fabricated-fetch rate (traces asserting tool output without a successful call), and task completion rate. The projection from R3 is that condition B' shows elevated forced halts or fabrications relative to both B and S; if B' and S show statistically indistinguishable rates of both failure types, the practical claim is refuted for that framework and task suite. This experiment requires no new models and no new infrastructure beyond a gate flag and a signal field.

**(d) Binding constraints.** In order of likely first contact: *protocol surface* — the signal must reach the model's context reliably (a signal the model cannot see is functionally absent, recreating the binary condition); *checker consistency* — the compliance checker and the platform's gate state must be drawn from the same source of truth, or signal-reality mismatch appears (Section 6, failure mode one); *specification review cost* — three-state predicates add clauses that must themselves be reviewed, a cost that grows with tool count  $k$ ; *data* — the rates  $p_C$  and  $p_P$  are currently assumptions, and the first useful engineering artifact is a measurement of their actual distribution on one production platform. Regulation and manufacturing constraints do not bind at this stage; the binding resources are protocol text, telemetry, and evaluation harnesses.

**(e) Order-of-magnitude bounds (projections).** With the assumptions of Section 4.3 ( $p_C = 0.90$ ,  $p_P = 0.70$ , independence), the signaling mechanism is worth adopting if its per-task cost is below  $s^* = 0.27 c_t$ . Since a disabled-state signal is a schema field plus a short instruction clause, its marginal per-task cost is plausibly a small fraction of a single wasted task's cost in any deployment where  $c_t$  includes compute and latency; we do not quantify  $c_t$  because no deployment cost data are available to us, and the projection is the ratio only. Sensitivity: the threshold is  $q c_t$  for whatever  $q$  a given platform measures, so the decision rule survives any measured  $q$ .

**(f) Risks and misuse paths.** The degrade-and-continue action creates an evasion path: an agent (or its operator) may classify a working tool as disabled to skip a mandated verification step, and a three-state checker that trusts agent self-reports without cross-checking the platform signal would license exactly the fabrication the protocol was written to prevent. Any implementation must bind the disabled report to a platform-issued signal, not to agent assertion.

## 8. Conclusion

Agent protocols that assume a binary tool-availability model are underspecified on a third state — capability present, permission denied — that per-tenant and per-budget gating makes common. We formalized the mismatch: a three-state semantics over capability and permission, a definition of protocol compliance over it, and a classification of protocol designs by well-definedness. The combinatorial result is that a binary protocol over  $k$  gated tools is specification-complete on  $(2/3)^k$  of the true state space ( $32/243 \approx 0.132$  at  $k = 5$ ; a  $57.7\times$  state-space gap at  $k = 10$ ). The computability result is that deciding whether an arbitrary protocol loop halts compliantly is undecidable, so well-definedness must be built in by syntactic restriction rather than verified after the fact. The economic projection, under stated rates, is a mismatch probability of  $0.27$  per gated invocation and a signaling break-even threshold of  $0.27 c_t$  per task. The design rule that follows — separate capability from permission, signal the disabled state explicitly, and define compliance over all three states — removes the forced-halt/violation dilemma by construction. The smallest experiment that would confirm or refute the practical claim is a three-condition gate study measuring forced-halt and fabricated-fetch rates; until such measurements exist, the numbers here are projections and the framework is a specification proposal, not a validated deployment practice.

## Provenance and AI Involvement

---

- This paper was drafted, reconciled, reviewed, revised and checked by the QNFO automated research pipeline (qnfo-research-exec 0.18.5-codeagent). Language models were tools in that process; they are not authors. The named author is accountable for the content.
- Models: drafting by openai/gpt-oss-120b, zai-org/glm-5.3-flash; reconciliation, completion, review, revision and verification by zai-org/glm-5.3-flash, zai-org/glm-5.3, openai/gpt-oss-120b, with the QNFO model router as a fallback.
- Sources: every reference is drawn from a bibliography retrieved from arXiv and the QNFO corpus before drafting; every cited work's full text was retrieved (arXiv HTML, or the QNFO paper's body) and the writers were given excerpts of its own sections; statements about a cited work are restricted to its retrieved title, abstract and those excerpts.
- Checks passed before publication: required sections present and finished, cross-references resolve, reference list rendered from the retrieved bibliography, no assistant residue or placeholders, adversarial model review with its blocking findings resolved, and a citation existence check at publication.
- Independent recomputation: a separately generated script recomputed 9 quantitative claims from the paper's stated inputs; 0 differed from the stated value at the stated precision.
- Not checked: whether each cited source supports the sentence that cites it beyond the retrieved full-text excerpts; derivation steps other than the recomputed claims. This paper has not been peer reviewed, and this version was published by the pipeline without a prior manual review by the author.

## References

---

[1] Quantum measurement occurrence is undecidable. arXiv:1111.3965v3. <https://arxiv.org/abs/1111.3965v3> [2] YouZhi: Towards High-Concurrency Financial LLMs via Adaptive GQA-to-MLA Transition. arXiv:2606.05868v1. <https://arxiv.org/abs/2606.05868v1> [3] HALT: Verification-Aware Stopping for Retrieval-Augmented Search Agents. arXiv:2608.02009v3. <https://arxiv.org/abs/2608.02009v3> [4] The Halt Vector: Internalizing a Causal Steering Intervention for Efficient Reasoning. arXiv:2608.28859v1. <https://arxiv.org/abs/2608.28859v1> [5] Undecidable problems in quantum field theory. arXiv:2203.16689v3. <https://arxiv.org/abs/2203.16689v3> [6] InverseNet: Benchmarking Operator Mismatch and Calibration Across Compressive Imaging Modalities. arXiv:2603.04538v1. <https://arxiv.org/abs/2603.04538v1> [7] On the Undecidability of Artificial Intelligence Alignment: Machines that Halt. arXiv:2408.08995v1. <https://arxiv.org/abs/2408.08995v1> [8] QNFO: Ontological Re-Entry: What Can We Say Exists?