# AgentGuard

Formally Verified Resource Coordination for Safe Multi-Agent LLM Systems

Yi-De Wu · Formace Lab · 2026-03

# Problem: Scheduling Bugs in Agent Runtimes

**Context**: LLM agents evolved from single-turn chat to persistent, multi-step services sharing tools and credentials

**Empirical study**: Analyzed OpenClaw (227K stars) + NanoClaw (14K stars)

> Found **43 concurrency bugs**, all falling into 3 classes with the same root cause:
> **Lack of a formally verified state machine**

| Bug Class | Count | Root Cause |
|---|---|---|
| A: Duplicate Execution | 8 | Poll loop re-enqueues tasks during long runs |
| B: Idle-Container Deadlock | 22 | Single `active` boolean conflates processing/idle |
| C: TOCTOU Race | 13 | Dedup check covers pending only, not running |

# Bug A: Duplicate Execution (8 instances)

**Scenario**: 30-min task in progress; 60s poll loop re-enqueues the same task

```
Task X enqueued -> running (30 min)
          ^ 60s poll: "X not completed yet" -> enqueue again!
```

**Impact**: Same email sent twice, same cron job executed twice

**AgentGuard solution**: `SkillExclusivity` invariant

- Each skill can only be held by one session at a time
- TLA+ `AtomicSchedule` transition eliminates duplicate resource claims
- **TLC verifies >1M states, 0 violations**

# Bug B: Deadlock (22 instances)

**Scenario**: Session crashes while holding a skill lock -> all subsequent tasks wait forever

```
Session A: acquire(fileio) -> crash! -> lock orphaned forever
Session B: acquire(fileio) -> wait... wait... wait...
```

**Impact**: NanoClaw #293 -- scheduled tasks deferred indefinitely, sessions permanently hung

**AgentGuard solution**: Stale lock timeout + Preempt action

- 30-second auto-release of orphaned locks by default
- `Preempt` counts toward retry limit (TLC discovered: without this, tasks loop infinitely)
- Llama experiment: LLM crash + corrupt PNG -> lock correctly reclaimed

# Bug C: TOCTOU Race (13 instances)

**Scenario**: Dedup check only inspects `pending[]`; after task enters `running`, check passes -> duplicate

```
t0: task X -> pending (dedup OK)
t1: task X -> running (removed from pending)
t2: task X -> pending again (dedup check: "X not in pending" -> passes!)
```

**AgentGuard solution**: `AtomicSchedule` + `NoBypassSafety` invariant

- Schedule is an atomic transition -- no intermediate state exists

**Evaluation**: 100 programmatic trials

| | Without AgentGuard | With AgentGuard |
|---|---|---|
| Race occurrence | **100%** | **0%** (95% CI [0%, 3.7%]) |

# Solution: TLA+ Verified Passive Guardrail

**AgentGuard = passive resource guard; does not execute tools, only manages locks**

**TLA+ Specification**

- 960 lines of TLA+ spec
- 6 states x 8 transitions
- **8 safety invariants**
- **2 liveness properties**
- TLC: **>1M states**, all pass

**Implementation**

- ~4,700 LOC Rust
- **146 tests** (unit/integration/scenario)
- 500K property-based transitions
- **Zero invariant violations**
- ~1ms per acquire/release

**Architecture**: Claude Code -> Unix socket -> AgentGuard daemon (FIFO queue + locks + invariant checker) -> Resources

# Baseline: AIOS (COLM 2025)

**Closest prior work**: AIOS -- LLM Agent Operating System

| Feature | AIOS | AgentGuard |
|---|---|---|
| Conflict detection | Hashmap (ad-hoc Python) | TLA+ verified state machine |
| FIFO fairness | No (re-queue unordered) | Yes, FIFO starvation-free |
| Multi-resource atomic | No (sequential acquire -> deadlock risk) | Yes, DeadlockFreedom invariant |
| Stale lock recovery | No (manual intervention) | Yes, 30s auto-release |
| Verification | None | >1M states, 8 invariants |

**Quantitative comparison** (100 trials, generous AIOS -- added caller-side retry):

| Scenario | No coordination | AIOS (retry) | AgentGuard |
|---|---|---|---|
| Bug A: Duplicate | 100% fail | 0% fail | 0% fail |
| Bug B: Recovery | N/A | **deadlock** | **recovered** (2s) |
| Bug C: TOCTOU | 100% fail | 0% fail | 0% fail |

**Key difference**: Bug B -- after session crash, AIOS deadlocks permanently; AgentGuard

# Deliberate Non-Goals

## Intentional scope limits (paper Discussion section)

| Non-goal | Why it is safe |
|---|---|
| Does not validate tool input / file content | Separation of concerns: coordination != LLM output |
| Does not prevent tool crashes | Stale lock timeout bounds blast radius |
| Does not execute tools | Pure passive guardrail, 0.7ms overhead |
| Does not manage MCP tools | Outside hook pipeline; can be proxied |
| Does not guarantee multi-node | Single-machine SQLite serializable txn; future work |

**Fail-open design**: When daemon is unreachable, tool calls proceed normally (exit 0)

-> Safety degrades gracefully without blocking work

# Reproducibility -- 5 Minutes to Reproduce

```
git clone https://github.com/yiidtw/agentguard && cd agentguard/software
cargo test                                          # 146 tests
bash appendix/demo-bug-a.sh                         # Reproduce Bug A
bash appendix/demo-bug-b.sh                         # Reproduce Bug B
bash appendix/demo-bug-c.sh --trials 100            # Reproduce Bug C + statistics
```

| Artifact | Command |
|---|---|
| Throughput benchmark | `cargo bench` |
| TLC verification | `cd spec && tlc AgentGuard.tla` (see `spec/README.md`) |
| All 43 bug links | `data/bugs.csv` |

**Live demo available**: `cargo run -- demo --port 3170`

# Summary

**43 real bugs -> 3 classes -> 960-line TLA+ spec -> >1M states verified**

**146 tests, 500K transitions, zero violations**

**vs. AIOS: +formal spec, +fairness, +atomicity, +recovery**

Three-layer verification = core methodological contribution:

1. **Empirical bug taxonomy** -- 43 bugs -> 3 classes
2. **Exhaustive model checking** -- TLA+, >1M states
3. **Conformance testing** -- 146 tests, 500K transitions

github.com/yiidtw/agentguard