

Agent Tool-Call Hallucination & Argument Fidelity

智能体工具调用幻觉与参数保真度

2026年10月17日 | AI Agent 知识分享 | 第79期

一、一句话概括

Agent 最贵的错误不是不调工具，而是调了一个不存在的工具、把位卡号抄错一位、把「元」当成「分」——schema 全过、接口返回 200，业务已经错了。16

二、核心原理

>> 失败分三层，测的东西不同

选择层：选错工具（该用 A 用了 B，尤其是名字相近的工具），或没有对应工具时硬套一个。参数层：字段缺失、类型错、枚举越界、值幻觉。执行层：调用顺序、幂等、副作用范围（见第 53 期）。

>> 参数层还要再分：语法错 vs 语义错

语法错（字符串 5000 与数字 5000）能被 jsonschema / Pydantic 挡住；语义错（订单号是编造的、金额单位从元变分、币种错、时区错）schema 一点都挡不住，只能靠业务校验加逐字段比对。

>> 幻觉工具 (hallucinated tool)

模型编出一个从未注册的工具名，或把已注册的工具名拼错。工具数量越多、描述越相近，发生概率越高。Berkeley Function Calling Leaderboard 专门用 relevance / irrelevance detection 两个维度度量「这一句到底该不该调工具」。

>> 参数保真度 (argument fidelity)

Agent 是否忠实转述用户给的原始值，而不是擅自改写、补全或猜。16 位以上卡号、13 位订单号这类长数字的位级保真度下降是社区与论文中反复观察到的现象，且错误往往看起来合理：相邻数字互换、后几位漂移，人眼抽查极难发现。

>> 单次正确率不等于可靠

tau-bench 提出 $pass^k$ ——同一任务独立重复 k 次、全部成功的比例。论文中表现最好的模型在 retail 域 $pass^1$ 约 61.2%， $pass^8$ 掉到 25% 以下。单次看着九成的 Agent，跑 8 次全对可能只剩六成。

>> 与工具投毒的区别

第 75 期的 MCP 工具投毒是工具描述被恶意改写，需要攻击者；本期的幻觉与参数错误是模型自身能力边界造成的，不需要任何人攻击，它自己就会错。

核心要点

核心一句话：schema 只能证明「格式对」，证明不了「值对」。

三层分开测：选择层（选对工具）、参数层（字段与值逐字段比对）、执行层（顺序与幂等）。

最贵的一类错：幻觉工具、编造实体 ID、金额单位与币种错配——全部能通过 schema 校验。

验收两条硬线：幻觉工具率 = 0；核心用例重复 k 次看 pass^k ，不只看单次成功率。

三、测试实战

1. 工具选择准确率基线

- ◆ 目的：量化 Agent 在近义工具之间选对的概率。
- ◆ 步骤：注册银行场景 30-50 个工具（查询余额、转账、冻结卡片、冻结账户、查询交易明细、查询交易流水、汇率查询、开卡、销户），刻意安排 6-8 对易混工具；写 100 条自然语言意图并标注正确工具，其中 20 条刻意不给任何工具能完成的请求（如「查一下我的公积金」）。
- ◆ 工具：Berkeley Function Calling Leaderboard (gorilla.cs.berkeley.edu/leaderboard.html) 的评测维度可当模板，跑自己的数据用 Python + 模型 API 即可。
- ◆ 指标：Top-1 工具选择准确率、易混对误选率、无解意图正确拒答率。参考线：Top-1 $\geq 90\%$ ，无解拒答率 $\geq 85\%$ 。

2. 参数保真度逐字段比对

- ◆ 目的：测 Agent 会不会「抄错」用户给的值。
- ◆ 步骤：构造 100 条含 16-19 位卡号、13 位订单号、身份证号、带小数的金额、明确日期的用户话语，要求 Agent 生成工具调用；解析出 arguments 后与用户原话逐字段精确比对（字符串逐字符比，数字用 Decimal 比，避免浮点误判）。
- ◆ 指标：字段级 exact match 率、数字位级错误率、金额单位错配率（元/分/万元）、币种错配率、日期格式合规率。参考线：长数字字段 exact match $\geq 99\%$ ，错一位即判失败。

3. Schema 合规不等于语义正确

- ◆ 目的：证明严格模式到底能挡住哪一类错。
- ◆ 步骤：给工具加严格 JSON Schema（枚举、必填、类型、additionalProperties 置 false），开启模型侧 strict / structured outputs；先构造 5 类畸形参数（类型错、枚举越界、必填缺失、多余字段、数字超范围），再构造 5 类「schema 全过但业务错」的参数（编造订单号、金额单位错、收款人姓名与账号不匹配、超额、已冻结账户）。
- ◆ 工具：jsonschema (python-jsonschema.readthedocs.io)、Pydantic (docs.pydantic.dev)、OpenAI Structured Outputs 文档。
- ◆ 指标：schema 通过率（应接近 100%）、schema 通过但业务校验失败率——这个数就是真正的风险敞口，通常远大于 0。

4. 幻觉工具与幻觉实体

- ◆ 目的：测 Agent 是「编」还是「问」。
- ◆ 步骤：注册 12 个工具，投喂 100 条请求，其中 20 条需要不存在的工具（公积金、征信、他行流水），20 条提及系统中不存在的实体（不存在的订单号、卡号），60 条正常。逐条看 Agent 是编造工具名、硬套已有工具、编造参数值，还是停下来追问。
- ◆ 指标：幻觉工具率、硬套错误工具率、参数值编造率、正确澄清率。参考线：幻觉工具率必须为 0，正确澄清率 $\geq 80\%$ 。

5. 多实体并行调用的参数串扰

- ◆ 目的：测批量场景下参数会不会「串门」。
- ◆ 步骤：给 Agent 一条包含 4 个实体的复合指令（把 A 卡 100 元转给甲，把 B 卡 200 元转给乙），检查生成的 2 次调用共 8 个参数是否各归其位；再把实体数加到 6-8 个重复测试。
- ◆ 指标：参数串扰率（值错配到另一实体的比例）、漏调率、重复调用率。参考线：串扰率必须为 0，漏调率 $\leq 2\%$ 。

6. 参数级模糊测试 (fuzzing)

- ◆ 目的：让 Agent 面对畸形输入时也不把脏值传进核心系统。
- ◆ 步骤：用属性测试工具按 OpenAPI / Schema 生成边界值——空串、超长（10 万字符）、emoji、全角数字、科学计数法（1e6）、负数与 0、超范围金额、SQL 片段、路径穿越（../etc/passwd）、命令拼接——观察 Agent 是否原样填进参数并提交。
- ◆ 工具：Schemathesis (schemathesis.readthedocs.io, 直接吃 OpenAPI 做属性测试)、Hypothesis (hypothesis.readthedocs.io)。
- ◆ 指标：越界参数被接受并提交的比例（应为 0）、脏值抵达下游接口的比例、异常崩溃率。

7. tau-bench 式可靠性：pass^k 与政策违规

- ◆ 目的：把「偶尔对」和「稳定对」分开。
- ◆ 步骤：挑 20 条带业务规则的任务（如「转账超过单笔限额必须拒绝并提示」），每条独立重复 5 次，统计全对率；同时统计违反领域规则（该拒没拒、该提示没提示）的次数。
- ◆ 工具：tau-bench (github.com/sierra-research/tau-bench, 论文 arXiv:2406.12045)。
- ◆ 指标：pass^1、pass^5、政策违规率。参考线：pass^5 与 pass^1 落差不超过 10 个百分点才算稳定；政策违规率必须为 0。

8. 改动前后回归

- ◆ 目的：换模型版本、改 prompt、加工具之后必须重测。
- ◆ 步骤：把上面 1-7 的用例固化成一套回归集（建议 300-500 条，每条带期望的工具名与参数字段），每次改动前后各跑一次，diff 出新增失败。
- ◆ 指标：回归通过率变化、新增失败条数、参数保真度下降幅度。参考线：任一项下降超过 2-3 个百分点即回滚或返工。

四、常见坑点

坑点1：把 schema 校验当验收

严格

JSON

Schema

只保证结构，不保证语义。类型对了、枚举对了、必填都填了，订单号仍可能是模型编的，金额单位仍可能从元变成分。验收必须再加一层业务校验（实体存在性、单位、限额、账户状态），否则最贵的那类错会全部漏网。

坑点2：只看最终回答，不看调用轨迹

Agent

调错工具、拿到错误结果、再用自然语言编一个听起来合理的答案——端到端打分看着还行，实际上核心接口已经被调错了。评测必须记录并断言每一次 tool call 的工具名与参数，而不是只比对最后那句话。

坑点3：工具描述里的示例值会变成默认值

description 里写「例如 order_id 等于 123456」，用户没给订单号时，模型有相当概率直接把这个示例值填进去。工具描述里不要放看起来像真实数据的示例，改用占位符并明确写「必须来自用户输入或上游工具返回」。

坑点4：单次通过率高掩盖不稳定

pass^1 到 92% 听起来能用，pass^5 可能只剩六成多。银行场景要的是「每次都这样」，所以核心用例必须重复跑 k 次看全对率，而不是跑一次看成功率。

坑点5：长数字复制错误看起来合理

16

位以上数字的位级保真度下降是常态，错误常表现为相邻数字互换或后几位漂移，人工抽查很难发现。这类字段必须自动化逐字符比对，不要依赖人眼看。

五、延伸阅读

- Berkeley Function Calling Leaderboard (BFCL): <https://gorilla.cs.berkeley.edu/leaderboard.html>
- Gorilla / APIBench 论文（首个系统评估 API 调用幻觉的工作）: <https://arxiv.org/abs/2305.15334>
- tau-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains: <https://arxiv.org/abs/2406.12045>
- tau-bench 代码库: <https://github.com/sierra-research/tau-bench>
- ToolEmu: Identifying the Risks of LM Agents with an LM-Emulated Sandbox: <https://arxiv.org/abs/2309.15817>
- ToolLLM / ToolBench: <https://arxiv.org/abs/2307.16789>
- OpenAI Structured Outputs 文档: <https://platform.openai.com/docs/guides/structured-outputs>
- Schemathesis (OpenAPI 属性测试): <https://schemathesis.readthedocs.io/>
- Hypothesis (Python 属性测试): <https://hypothesis.readthedocs.io/>