arXiv:2601.04688v1 [cs.CL] 8 Jan 2026

ToolGate：基于契约并经验证的大语言模型工具执行

Yanming Liu[1] **XinyuePeng**[2] **Jiannan Cao**[3] **XinyiWang SonghangDeng**
陈金涛[1] **JianweiYin**[1] **Xuhong Zhang**[1*]浙江大学 [2]东南大学[3]麻省理工学院
{oceann24, zhangxuhong, zjuyjw, chenjintao}@zju.edu.cn
xinyuepeng@seu.edu.cn, jiannan@mit.edu

摘要

借助外部工具增强的大语言模型（LLM）在复杂推理任务中展现出了卓越的能力。然而，现有框架高度依赖自然语言推理来判断何时可以调用工具，以及是否应采纳其结果，却缺乏逻辑安全性和可验证性的形式化保证。我们提出 **ToolGate**，这是一种前向执行框架，可为大语言模型工具调用提供逻辑安全保证和可验证的状态演化。ToolGate 将显式符号状态空间维护为类型化键值映射，在整个推理过程中表示可信世界信息。每个工具都被形式化为由前置条件和后置条件构成的霍尔式契约：前置条件通过检查当前状态是否满足所需条件来控制工具调用，后置条件则通过运行时验证，决定是否可以采纳工具结果以更新状态。我们的方法确保符号状态只通过经过验证的工具执行而演化，从而防止无效或幻觉结果破坏世界表示。实验验证表明，ToolGate 在保持复杂多步推理任务上竞争力的同时，显著提升了工具增强型大语言模型系统的可靠性和可验证性。本研究为构建更可信、更易调试的人工智能系统奠定了基础，这类系统能够将语言模型与外部工具相结合。

1 引言

大语言模型（LLM）在各种推理任务中取得了显著成功，尤其是在借助外部工具后，这些工具使其能够与现实世界交互（Yao 等人，2022；Brown 等人，2020；Chowdhery 等人，2022）。工具与大语言模型的集成为复杂的多步骤推理开辟了新的可能，使模型能够通过 API 调用检索信息、执行计算并采取行动（Qin 等人，2024）。然而，现有的大语言模型工具调用框架在很大程度上依赖自然语言推理来确定何时应调用工具，以及是否应信任其结果并将其纳入系统对世界的理解中（Yang 等人，2024a）。这种对隐式自然语言推理的依赖，使确保工具增强型大语言模型系统的逻辑安全性和可验证性面临挑战。

根本问题在于，工具调用和结果验证缺乏形式化保证。当前方法将工具调用视为黑箱过程，由大语言模型依据其内部推理自行决定，而没有明确机制来验证工具调用的前置条件是否得到满足，或工具输出是否符合预期的后置条件（Zhu et al.，2025；Shi etal.，2023）。这可能导致几个关键问题：工具调用时可能缺少必要参数或参数有误；无效结果可能被纳入推理过程；系统对世界状态的内部表示可能因幻觉或错误的工具输出而变得不一致或遭到破坏（Huang etal.，2025）。此外，随着可用工具数量增至数千个，高效检索和选择合适工具也变得愈发困难，需要采用超越简单关键词匹配的复杂检索机制（Xu etal.，2024）。近期方法仍缺乏统一框架，无法对何时可以安全调用工具以及何时可以信任其结果提供形式化保证。由于缺少明确的状态管理和基于契约的验证，错误可能沿推理链传播，导致复杂的多步工具调用场景中难以定位和调试故障。

*通讯作者。

为解决这些局限，我们提出**Tool-Gate**，这是一种面向前向执行的框架，可为大语言模型工具调用提供逻辑安全保证和可验证的状态演化。ToolGate 引入了明确的符号状态空间，其中维护一个类型化键值映射，在整个推理过程中表示可信的世界信息。每个工具都被形式化为一个霍尔式契约，其中前置条件用于控制工具调用，后置条件则用于判定工具结果能否提交以更新状态。ToolGate 将用于高效工具检索的嵌入语义搜索与霍尔契约逻辑检查相结合，确保符号状态仅通过经过验证的工具执行而演化，从而防止无效或幻觉生成的结果破坏世界表示。

我们的贡献。具体如下。

• 我们提出了 ToolGate，这是一种通过霍尔式契约对工具调用进行形式化的新框架，可为工具增强型大语言模型系统提供逻辑安全保证和可验证的状态演化。

• 我们引入了一个显式的符号状态空间，在推理过程中持续维护可信世界信息，从而能够对工具调用进行精确的前置条件和后置条件检查。

• 我们证明，基于契约的验证显著提高了工具增强型大语言模型系统的可靠性和可调试性，同时在复杂的多步推理任务上保持了有竞争力的性能。

2 相关工作

2.1 大语言模型的工具学习

外部工具与大语言模型的集成，已成为一项关键能力，使大语言模型推理得以从文本生成扩展到现实世界的交互。早期关于函数调用的研究，例如 OpenAI 的函数调用 API，使大语言模型能够通过结构化参数调用外部函数（Ouyang 等人，2022）。ReAct 框架将推理—行动范式形式化：大语言模型在推理步骤和工具调用之间明确交替，从而在复杂的多步推理任务中展现出更优表现（Yang 等人，2024a）。在此基础上，工具学习逐渐兴起成为一种有效范式，能够显著拓展大语言模型的能力（Schick 等人，2023；Qin 等，2023；Yu 等人，2025年）。早期研究提出，将大语言模型与外部工具（如程序执行器或搜索引擎）集成（Erdogan 等人，2024；Paranjape 等人，2023）。为全面衡量工具使用表现，研究人员推出了一系列基准测试，系统评估从 API 选择、参数生成质量到泛化能力等多个维度（Ye 等人，2025年；Patil 等人，2024；Du 等人，2024）。这些技术已扩展至图形用户界面智能体等多模态任务（Zhang 等人，2025a；Liu 等人，2025b）及专业领域（Su 等人，2025年）。近期，强化学习（RL）也被纳入该框架，以进一步优化工具学习表现（Qian 等人，2025年；Li 等人，2025年），并在信息检索和动态推理方面取得显著成果。这些进展表明，工具增强型大语言模型在开放领域通用推理方面展现出巨大潜力。

2.2 霍尔逻辑与形式化验证

霍尔逻辑（Hoare，1969）通过前置条件和后置条件，为推理程序正确性提供了一套形式化系统。近年来，形式化验证和霍尔逻辑日益被引入深度学习领域，用于描述和约束神经网络系统在不同输入和内部状态下可证明的行为（Corsi 等人，2021）。随着深度学习模型广泛应用于自动驾驶、机器人控制、医疗决策和工业系统等高风险、安全关键领域，确保模型输出不仅有效，而且可验证并符合预先定义的规范，已成为一个日益重要的问题（Meng 等人，2022；Swaroop 等人，2024）。在此背景下，霍尔逻辑提供的前置条件—后置条件框架用于规定神经网络在给定输入条件或一阶逻辑下必须满足的功能、安全或鲁棒性属性（Yang 等人，2024b；Han 等人，2024），并进一步与神经网络验证和大语言模型相结合，形成一种统一且严谨的推理与验证方法（Lee 等人，2025年；Grigorev 等人，2025年；Wang 等人，2019；Lin 等人，2024）。

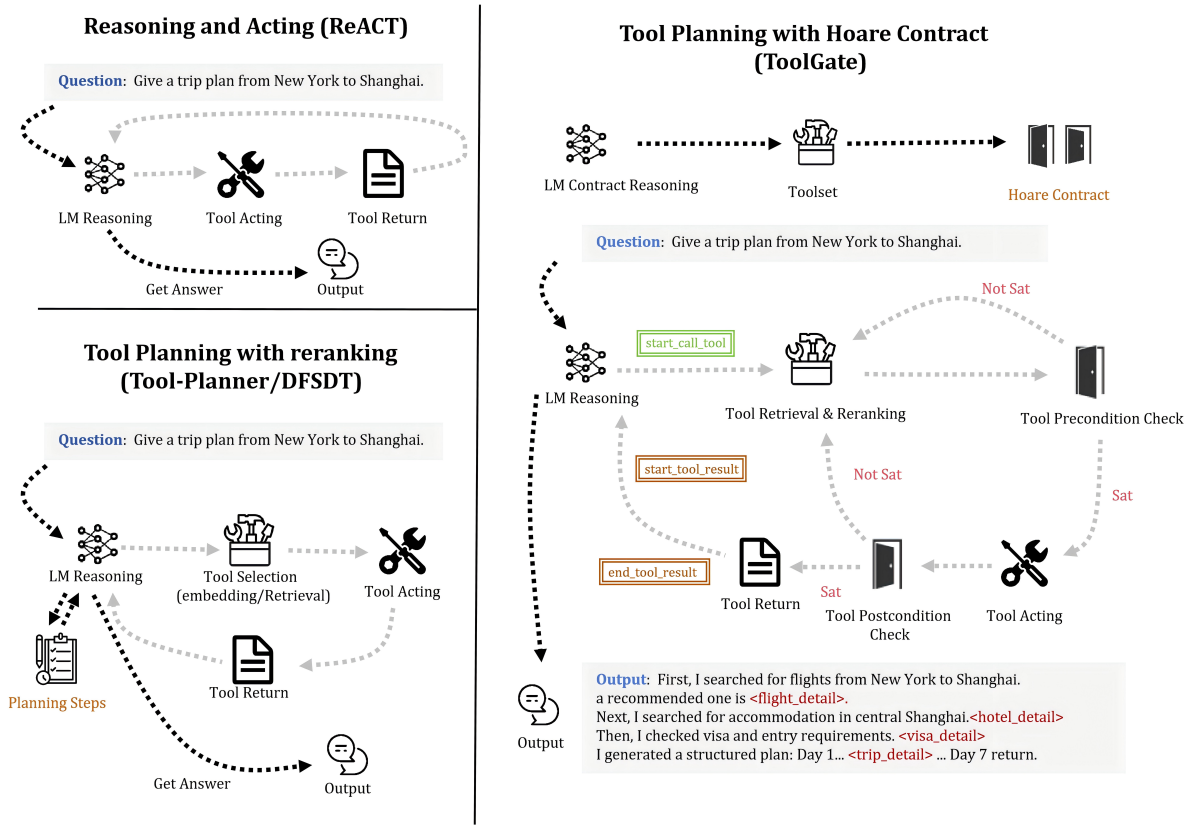


图1：ToolGate 框架概览。该框架基于霍尔逻辑，将工具调用过程形式化为一系列受约束的逻辑推理步骤，并持续维护可信状态 S，以验证工具调用的条件。

3 方法

3.1 问题设定与概述

工具学习赋予大语言模型规划、调用外部工具并围绕其进行推理的能力。然而，幻觉传播和不可靠的工具规划仍是主要瓶颈，常常导致结果不稳定且不可靠。为解决这些问题，我们提出了**ToolGate**，一个融合概率推理基础与逻辑可验证保证的框架。它由维护可信信息的类型化符号世界状态 S、工具的霍尔式逻辑契约 $\{P_t\}\, t\, \{Q_t\}$，以及一种由大语言模型驱动、但受霍尔逻辑约束的概率推理机制组成。

问题描述。 给定输入序列x和一组可用工具 $T = \{t_1, t_2, \ldots, t_n\}$，工具学习旨在通过以下方式生成答案：

$$y = \arg\max_{y_i} P(y_i \mid x,\ T_0 = \{t_{x_i}\}) \qquad (1)$$

其中，T_0 表示根据输入 x选择的工具及其对应的输出。

3.2 符号状态构建与工具契约

为确保工具执行并非仅由非结构化的自然语言记忆驱动，而是以可验证且可进行逻辑解释的世界模型为基础，我们首先构建类型化符号状态空间 Σ。我们维护一个可信的符号状态 $S \in \Sigma$，其中每个元素都表示为一个元组（k，v，σ），用于记录一个键及其值和对应的类型，即，

$$\Sigma = \{(k, v, \sigma)\} \qquad (2)$$

这种表示使系统能够以结构化、可检查的方式显式编码“当前已知的信息”。已验证实体、中间推理结果和经过验证的工具输出都会写入这一状态空间。为确保推理和工具执行过程中的逻辑一致性，我们还在 Σ 上定义了一组逻辑谓词，用于表达存在性约束、类型一致性和语义不变量；我们用 $S \models \varphi$ 表示给定的符号状态 S 满足逻辑条件 φ。

为防止模型随意调用工具，并减少幻觉或不受约束的执行行为，我们为每个工具 $t \in \mathcal{T}$ 指定一个霍尔风格的逻辑契约，其形式为

$$\{P_t\}\, t\, \{Q_t\} \tag{3}$$

前置条件 $P_t: \Sigma \to \{true, false\}$ spec-规定了工具能够合法调用所必须满足的最小状态要求，这意味着只有在满足 $S \models P_t$ 的情况下，工具才可执行。同时，后置条件 $Q_t: \Sigma \times R_t \to \{$true，false$\}$. 约束了运行时输出 r_t 的结构有效性、类型正确性和语义一致性，同时还定义了经过验证的结果如何更新系统状态。

3.3 工具调用与重排序

我们首先将模型的推理视为一个随时间进行的条件概率传播过程。在第 k 步，推理状态表示为

$$p(R_k \mid q, H, S_k) \tag{4}$$

其中，q 表示当前用户查询，H 代表稳定且外部可见的对话历史，S_k 表示此时的可信符号状态。我们将 R_k 定义为当前推理轨迹，其中记录了在第 k 步之前产生的中间推理内容、推理路径以及已注入的任何工具结果。在这一表述中，H 跟踪外部可观察的交互，而 R_k 则捕捉模型内部推理过程的演变，从而清楚地表明，在后续的工具选择和状态更新中，哪些信息源自用户，哪些源自内部推理。

接下来，我们将调用工具的选择转化为推理过程本身中的内生随机决策。在当前信息状态下，模型估计：

$$p(\text{<start_call_tool>} \mid q, H, S_k, R_k) \tag{5}$$

而这一概率直接决定模型是否生成 <开始_调用_工具>。一旦出现该标记，系统便进入工具选择与执行阶段；当后续拼接上 <开始_工具_结果>,<结束_工具_结果> 时，系统便退出工具阶段，恢复纯粹的自然语言回答。这种设计使工具使用取决于模型当下的不确定性和任务需求，而非僵化的手工设定的触发条件，从而能够更平滑地适应有时需要工具、有时不需要工具的场景。

基于当前查询 q、对话历史 H、符号状态 S_k 和推理轨迹 R_k，我们构建工具需求表示：

$$u_k = f(q, H, S_k, R_k) \tag{6}$$

它提供了对当前子问题的结构化描述，明确系统旨在实现的目标以及所期望的工具输出类型。随后，我们将 u_k 作为查询，从庞大的工具集 $\mathcal{T}$ 中进行检索，并结合向量嵌入提取排名前 K 的候选工具：

$$\mathcal{C}_k = \text{TopK-Retrieve}(u_k, \mathcal{T}) \tag{7}$$

这有效地缩小了工具空间，仅保留一个规模较小且高度相关的候选集。

对于候选集 $\mathcal{C}_k$，我们在 $\mathcal{C}_k$ 中应用重排序模型，生成经过优化的排序分布：

$$p_{\text{rank}}(t \mid u_k), \quad t \in \mathcal{C}_k \tag{8}$$

3.4 规划中的工具契约

对于每个候选工具 $t \in \mathcal{C}_k$，我们使用指示函数 $\mathbf{1}[S_k \models P_t]$，根据当前符号状态 S_k 判断其前置条件是否满足，并剔除所有先决条件未满足的工具。随后，我们仅针对前置条件成立的工具重新归一化排名分布，从而形成逻辑上有效的执行策略：

$$p^*(t \mid q, H, S_k, R_k) = \frac{p_{\text{rank}}(t \mid u_k) \cdot \mathbf{1}[S_k \models P_t]}{\sum_{t' \in \mathcal{C}_k} p_{\text{rank}}(t' \mid u_k) \cdot \mathbf{1}[S_k \models P_{t'}]} \tag{9}$$

这种过滤机制超越了简单的语义匹配，确立了形式化的执行可接受性；它要求当前状态 S_k 满足所选工具的最弱前置条件，记为 $S_k \models wp(t, P_t)$。通过将此类确定性约束嵌入概率采样过程，我们确保模型的轨迹始终处于有逻辑依据的解空间内，而非依赖不受约束的启发式转移。

我们将 $p^*(t)$ 视为受逻辑约束的策略分布，并从中采样：

$$t^* \sim p^*(t \mid q, H, S_k, R_k) \tag{10}$$

只要某个工具既合法又具有实际相关性，它自然仍有机会被探索，而其采样概率则反映其在上下文中的优先级。选定并调用最终工具 t^* 后，它会返回结果 r_t。在用该输出更新系统状态之前，我们引入一道安全门，即运行时合约验证流程，用于检查返回结果是否满足霍尔后置条件 Q_t。我们将其形式化为二元接受事件 $\mathcal{A}_t \in \{0,1\}$，其条件概率为 $p(\mathcal{A}_t = 1 \mid S_k,\ r_t,\ Q_t)$，并将其实现为一个具体的验证函数：

$$\mathcal{A}_t = \begin{cases} 1, & \text{if } (S_k, r_t) \models Q_t \wedge \mathrm{wf}(r_t), \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

通过这一步，每个工具输出都必须满足结构有效性、值域约束和格式要求，才能影响全局状态。只有在验证通过后，符号状态才会更新：

$$S_{k+1} = \begin{cases} \mathsf{Update}_t(S_k, r_k), & \mathcal{A}_t = 1, \\ S_k, & \mathcal{A}_t = 0, \end{cases} \tag{12}$$

随后，接受的结果会注入后续推理轨迹 R_{k+1}。如果验证失败，则会彻底丢弃该结果，防止受污染的输出继续传播，并提供一个明确的调试断点。

基于此，我们将经过验证的结果同时封装在 <开始_工具_结果> 和 <结束_工具_结果> 标签中，并注入后续推理轨迹 R_{k+1}，使后续自然语言推理和下一轮工具选择都能充分利用这条新获取的可信信息。

在此基础上，我们将整个系统视为一组随机轨迹 τ，每条轨迹都由 $(S_k,\ R_k)$、选定的工具 t_k 和接受事件 $\mathcal{A}_k$ 组成。系统会对所有可行的执行轨迹进行概率推理，最终输出 y 可通过轨迹级边缘化表示：

$$p(y \mid q, H) = \sum_{\tau} p\big(y \mid q, H, \tau\big)\, p\big(\tau \mid q, H\big) \tag{13}$$

其中，$p(\tau \mid q,\ H)$ 整合了上述讨论的所有组件：工具触发概率、需求抽象、检索与排序、合约过滤、受约束采样以及接受验证。

为确保霍尔合约不仅能规范局部行为，也能规范全局行为空间，我们施加严格的轨迹级约束：如果任何轨迹 τ 在任何步骤违反任何工具前置条件 P_t 或后置条件 Q_t，那么

$$p(\tau \mid q, H) = 0 \tag{14}$$

在这一表述下，推理和采样仅在遵循预定义契约的轨迹子空间内进行，从而为每次状态转换和工具执行提供形式逻辑上的依据。

4 实验设置

4.1 数据集。

我们使用 **ToolBench** (Qin 等人，2023）和 **MCP-Universe** （Luo 等人，2025年）作为我们的实验数据集。ToolBench 包含超过 16,000 个 API，并按照结构化工具类别进行组织，覆盖广泛的功能。上述设置共同评估局部工具调用能力和全局规划鲁棒性。

MCP-Universe 体现了更贴近现实的多工具环境。它汇集了来自真实世界系统的多种工具、插件和 API，涵盖信息检索、自动化、数据处理、系统操作和任务执行。我们使用 ToolBench 和 MCP-Universe 中选定的工具及其官方文档、规范和使用说明，提取结构化功能表示。更多数据集详情请参见附录 A。

4.2 评估指标

对于 ToolBench，我们采用 ToolEval（Qin 等人，2023）中的两项评估指标。第一项指标是**通过率**，即成功完成的任务所占比例，反映整体任务解决能力。第二项指标是**胜率**，我们将本框架生成的执行计划和结果与 Qwen-3235B-ReACT 生成的进行比较，并请大语言模型评审员判断哪种方案更优。如果我们的方法得出更好的方案，则记为胜；如果效果相当或更差，则记为平局或负。胜率因此同时衡量推理质量和执行优势。

对于 MCP-Universe，我们评估**成功率**和执行稳定性。MCP-Universe 中的许多任务涉及相对较少的工具调用步骤，但源于现实世界复杂系统。

表1：ToolBench 和 MCP-Universe 上的主要实验结果。我们报告了 ToolBench G1、G2 和 G3 任务的通过率（%）和胜率（%），以及 MCP-Universe 三个子任务的成功率（%）。

模型	方法	ToolBench						MCP-Universe		
		G1		G2		G3		位置导航	代码仓库管理	金融分析
		通过。	Win.	通过。	Win.	通过。	Win.			
Qwen-3-235B	ReACT	50.5	–	53.5	–	46.0	–	11.10	9.09	50.0
	DFSDT	57.0	53.8	61.5	67.5	48.8	56.8	11.10	12.12	50.0
	LATS	62.5	59.3	78.0	70.3	77.8	83.3	15.54	15.15	52.5
	ToolChain*	65.0	62.8	79.3	72.5	78.0	83.5	16.65	18.18	55.0
	工具规划器	60.3	58.0	70.5	68.8	65.5	72.3	13.32	12.12	52.5
	ToolGate	**68.3**	**65.5**	**82.5**	**78.0**	**81.0**	**82.3**	**18.87**	**21.21**	**60.0**
DeepSeek V3.2	ReACT	52.0	48.5	55.3	51.0	48.5	53.5	12.21	12.12	52.5
	DFSDT	58.5	55.0	63.0	69.3	50.3	58.8	13.32	15.15	55.0
	LATS	65.3	61.8	80.0	72.5	80.3	85.5	17.76	18.18	60.0
	ToolChain*	68.8	65.0	82.5	75.3	81.0	88.8	18.87	21.21	62.5
	工具规划器	62.5	60.3	73.8	70.0	68.0	75.5	15.54	15.15	57.5
	ToolGate	**72.0**	**70.3**	**85.5**	**80.0**	**85.3**	**81.3**	**22.20**	**24.24**	**67.5**
GPT-5.2	ReACT	63.5	62.8	65.0	63.2	58.3	59.5	18.87	24.24	65.0
	DFSDT	70.0	68.5	75.3	78.0	63.8	70.5	19.98	27.27	67.5
	LATS	80.3	78.8	88.5	85.3	85.0	90.8	28.86	36.36	82.5
	ToolChain*	82.8	80.0	90.5	88.3	88.5	92.5	29.97	39.39	85.0
	工具规划器	75.5	72.3	82.0	80.5	78.3	85.0	25.53	30.30	75.0
	ToolGate	**85.5**	**83.5**	**93.0**	**90.5**	**91.8**	**95.3**	**35.52**	**45.45**	**90.0**
Gemini 3 Pro	ReACT	60.0	60.5	63.8	57.0	55.5	56.5	16.65	21.21	62.5
	DFSDT	68.3	65.5	72.0	75.8	60.3	68.5	17.76	24.24	65.0
	LATS	78.5	75.3	85.8	82.0	82.5	88.3	26.64	33.33	77.5
	ToolChain*	80.0	78.5	88.3	85.0	85.8	90.0	27.75	36.36	80.0
	工具规划器	73.8	70.0	80.5	78.3	75.0	82.8	22.20	27.27	72.5
	ToolGate	**83.0**	**80.5**	**91.3**	**88.0**	**90.0**	**93.5**	**33.30**	**42.42**	**87.5**

4.3 基线方法

我们将框架与以下具有代表性的工具使用和规划基线方法进行比较：**Re-ACT**（Yao 等人，2022），**DFSDT**（Qin 等人，2023），**LATS**（Zhou 等人，2024），**ToolChain***（Zhuang 等人，2024），**Tool-Planner**（Liu 等人，2025c）。更多基线方法的详细信息见附录 B。

4.4 模型

我们在一系列大语言模型上评估了我们的框架，以验证其通用性和稳健性。专有模型包括 **Gemini 3Pro**(谷歌公司，2025年）、**GPT-5.2** (OpenAI，2025年）。开源模型包括 **DeepSeekV3.2** (Liu 等人，2025a）、**Qwen3-235B-A22B-Instruct-2507**(Yang 等人，2025年）。这些模型涵盖了异构的训练范式、推理能力和规模。尽管我们使用 **Qwen3-embedding-0.6B**和 **Qwen3-Reranker-0.6B** (Zhang 等人，2025b）进行工具嵌入和检索。

5 实验

5.1 主要结果

如表 1 所示，我们在 ToolBench（G1/G2/G3）和 MCP-Universe 上进行了全面评估。

对于 ToolBench。结果显示，ToolGate 在所有模型和所有评测基准测试中均取得了最佳或接近最佳的表现。在 ToolBench 上，ToolGate 在三个任务组中都显著提升了通过率和胜率。例如，在 GPT-5.2 下，ToolGate 在 G1/G2/G3 上分别达到 **85.5/ 83.5**、**93.0 / 90.5** 和**91.8 / 95.3**，胜率比最强基线 ToolChain* 高约 $4-6\%$。在 Qwen-3-235B、DeepSeek V3.2 和 Gemini 3 Pro 上也持续观察到类似提升，这表明 ToolGate **与模型无关**，能够在不同大语言模型骨干网络上稳定增强工具推理和执行能力。

针对 MCP-Universe。ToolGate 的优势更加显著，这凸显了其在长时程工具依赖和现实世界执行鲁棒性方面的表现。ToolGate 在位置导航和代码仓库管理上较 ToolChain* 提升了 $3-7\%$，并在金融分析上取得了当前最先进的表现。值得注意的是，GPT-5.2 搭配 ToolGate 在代码仓库管理上的得分达到 **45.45**，在金融分析上的得分达到**90.0**，大幅领先所有竞争系统。这些结果表明，ToolGate 不仅提高了任务成功率，还显著增强了执行复杂工具链时的稳定性和鲁棒性。

表 2：霍尔逻辑验证模块的综合消融研究。我们将完整的 ToolGate 架构与以下变体进行比较：**不** $\{P\}$ **检查**（跳过前置条件验证）和**不** $\{Q\}$ **检查**（跳过后置条件断言）。**MCP-平均值** 表示 MCP子任务的平均成功率。

模型	方法	ToolBench						MCP-Universe			
		G1		G2		G3		Loc.	仓库。	Fin.	MCP-平均值
		Pass.	Win.	通过。	Win.	通过。	Win.				
DeepSeek V3.2	ReACT	52.0	48.5	55.3	51.0	48.5	53.5	12.2	12.1	52.5	25.6
	DFSDT	58.5	55.0	63.0	69.3	50.3	58.8	13.3	15.2	55.0	27.8
	ToolChain*	68.8	65.0	82.5	75.3	81.0	88.8	18.9	21.2	62.5	34.2
	不含霍尔的 ToolGate	57.2	53.8	61.5	67.5	49.8	57.5	12.8	14.5	54.2	27.2
	– 无 $\{P\}$ **检查**	67.8	66.5	79.5	77.2	78.4	82.8	19.8	21.5	62.2	34.5
	– 无 $\{Q\}$**检查**	63.2	61.0	71.5	72.8	70.8	73.5	16.2	18.2	58.2	30.9
	ToolGate（完整）	**72.0**	**70.3**	**85.5**	**80.0**	**85.3**	**81.3**	**22.2**	**24.2**	**67.5**	**38.0**
GPT-5.2	ReACT	63.5	62.8	65.0	63.2	58.3	59.5	18.9	24.2	65.0	36.0
	DFSDT	70.0	68.5	75.3	78.0	63.8	70.5	20.0	27.3	67.5	38.3
	ToolChain*	82.8	80.0	90.5	88.3	88.5	92.5	30.0	39.4	85.0	51.5
	不含霍尔的 ToolGate	69.2	67.5	74.5	76.8	62.5	69.2	19.5	26.8	66.5	37.6
	– 无 $\{P\}$ **检查**	81.2	79.5	89.2	88.0	86.4	90.5	31.0	41.5	85.0	52.5
	– 无 $\{Q\}$**检查**	75.5	74.0	82.8	83.5	79.2	82.0	25.5	33.5	79.6	46.2
	ToolGate（完整版）	**85.5**	**83.5**	**93.0**	**90.5**	**91.8**	**95.3**	**35.5**	**45.5**	**90.0**	**57.0**

针对多工具指令任务。实验结果表明，这些提升并非仅仅源于更强的启发式方法或更积极的探索，而主要得益于 ToolGate 基于霍尔逻辑的形式化约束机制。在推理过程中，系统会显式维护一个可信状态集 S，并为每次工具调用构建一个霍尔三元组$\{P\}\ C\ \{Q\}$，强制验证前置条件和后置条件。这使 ToolGate 能够显著减少 G2 和 G3 等复杂 ToolBench 任务中的错误累积，从而取得更高且更稳定的胜率；与此同时，在 MCP-Universe 中，它能有效缓解长时程推理漂移，在多工具依赖和现实世界执行约束下持续提升性能。

5.2 消融研究

为评估 ToolGate 对其形式化验证机制的结构性依赖，我们在 DeepSeek V3.2 和 GPT-5.2 上开展了系统的消融研究。我们专门剥离了霍尔逻辑模块，以观察其对工具使用效能的影响。如表 2所示，结果揭示了一项关键发现：移除形式化验证层后，性能会略低于标准 DFSDT 基线。

例如，对于 GPT-5.2，未采用霍尔过滤的 ToolGate 的 MCP-平均值成功率为 37.6%，略低于 38.3% DFSDT 所达到的数值。DeepSeek V3.2 也呈现出相同趋势。这表明，如果没有霍尔逻辑所提供的剪枝能力，ToolGate 的底层搜索架构会比传统的深度优先搜索策略效率更低。

结果表明，霍尔逻辑验证是 ToolGate 搜索效率的关键因素。霍尔逻辑框架的两个基本组成部分——前置条件 $\{P\}$和后置条件 $\{Q\}$ ——在引导工具调用决策方面发挥着互补作用。实证结果表明，缺少 $\{Q\}$检查的影响远大于缺少 $\{P\}$ 检查。例如，在 GPT-5.2 上，移除 $\{Q\}$ 检查后，MCP 平均成功率下降了 10.8%；相比之下，省略 $\{P\}$ 检查仅导致 4.5% 的降幅。ToolGate 完整版实施了严格的 $\{P\}C\{Q\}$ 逻辑闭环，确保搜索过程中的每一步在形式上都有效，在实质上也都有效。ToolGate 完整模型与其消融版本之间的性能差距证实，形式逻辑是提升可靠性和任务成功率的主要驱动力。

5.3 工具推理效率

为评估 ToolGate 的搜索效率，我们关注完成任务所需的平均工具调用步骤数。这一指标可作为

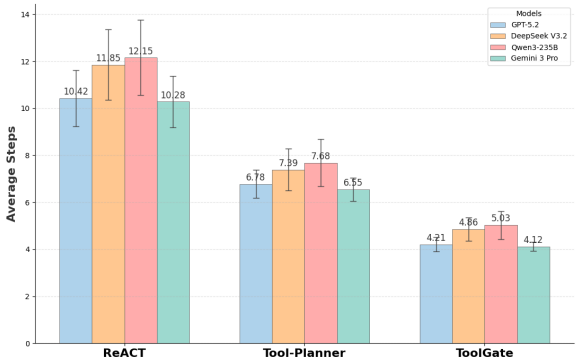


图2：Tool-Bench 中平均工具调用步骤的比较。

模型在复杂状态空间中进行导航且避免冗余探索的能力的替代衡量指标。

如图2所示，ToolGate 在 GPT-5.2 和 DeepSeek V3.2 两种骨干模型上始终实现最简洁的工具调用轨迹。具体而言，在使用 GPT-5.2 时，ToolGate 将平均调用步骤从 6.78 降至 4.21，效率提升了 37.9% 。相比之下，ReACT 和工具规划器等传统方法由于缺乏环境感知，往往陷入“试错”循环，而 ToolGate 则维持了接近理论最优路径的轨迹。

ToolGate 的效率主要归功于霍尔逻辑验证模块。在 Tool-Bench 庞大的状态空间中，工具前置条件与环境状态之间的逻辑冲突十分常见。不同于基于概率启发式探索分支的工具规划器，ToolGate 应用形式化约束来剪枝搜索树。通过在执行前验证工具调用的可行性，该系统能够有效压缩搜索空间，消除那些注定会在逻辑上失败的分支。

5.4 拒绝分布细粒度分析

为进一步探究 ToolGate 的内部决策机制，我们对评估期间的所有工具调用尝试进行了全面追踪。结果表明，在 **MCP-Universe** 等复杂度较高的基准测试中，形式化验证层会拦截约占工具调用请求总数**29.4%**的请求。基于对这些拒绝的细粒度分析，我们将发现归纳为三个关键方面：

通过 $\{P\}$ 前置条件检查主要过滤**参数幻觉实现静态剪枝**

表 3：霍尔组件中逻辑拒绝的细粒度分析。比率以 MCP-Universe 基准测试中的工具调用总数为基准进行归一化。

验证阶段	具体错误子类别	绝对比率（%）
前置条件 $\{P\}$	数值/实体幻觉	8.4%
	模式与格式违规	5.1%
	状态依赖缺失	4.1%
	小计 $\{P\}$ 拒绝	**17.6%**
后置条件 $\{Q\}$	空/空值	6.3%
	语义约束不匹配	3.7%
	状态更新不一致	1.8%
	小计 $\{Q\}$ 拒绝	**11.8%**
总计	**综合拒绝率**	**29.4%**

以及**状态依赖违规**。通过在执行前拦截无效 ID 和顺序错误的调用，$\{P\}$ 显著降低了计算开销，并防止搜索树扩展到无效分支中。

通过 $\{Q\}$ 实现动态纠正。后置条件断言能够捕捉标准模型容易遗漏的复杂失败。通过强制语义对齐和状态一致性，$\{Q\}$ 能够识别逻辑上空洞的步骤，并立即触发回溯，从而防止级联错误。

虽然 $\{P\}$ 通过静态剪枝 17.6% 的无效路径来优化**效率**，但 $\{Q\}$ 通过动态纠正剩余 11.8% 的逻辑偏移来确保**任务成功**。二者共同构成一个逻辑闭环，将智能体锚定在正确的语义轨迹上。

6 结论

本文介绍了 ToolGate，这是一种全面的方法，用于评估大语言模型（LLM）在工具使用场景中的关键决策与把关能力。ToolGate 将评估重点从单纯的执行成功转向对何时调用或拒绝使用工具这一细致决策的评估，揭示了当前最先进模型普遍存在过度依赖的倾向，尤其是在面对模糊、未经授权或高风险指令时。这些发现凸显了在功能熟练度与稳健决策框架之间取得平衡的迫切必要性。最终，ToolGate 既提供了诊断工具，也提供了基础框架，用于开发更加安全、可靠且自主的、面向现实世界应用的人工智能代理。

局限性

尽管 ToolGate 作出了诸多贡献，但仍应承认其存在若干局限。首先，尽管该基准测试涵盖了多种场景，但目前的范围主要局限于基于文本的交互和结构化数据交互，多模态工具以及长链条、多步骤的协作任务仍有待未来扩展。其次，评估环境总体上是静态的，可能无法充分捕捉现实世界 API动态的复杂性，例如网络延迟、速率限制或不断变化的数据状态，这些因素可能会干扰实时决策。此外，我们的评估指标仍主要是定量的；未来需要开发更加细粒度的定性评估，以衡量模型的解释性推理能力，以及其主动向用户索取缺失信息的能力。最后，基于提示的偏见仍然存在这一潜在问题，因为针对特定模型优化的策略可能无法完美泛化到整个开源大语言模型领域。

伦理考量

ToolGate 旨在为大语言模型推理中的正式、可验证且负责任的工具使用提供通用框架。所有实验均在公开可用的基准测试或开源工具环境中开展；在整个研究过程中，我们不会收集、访问或使用任何私人信息或个人身份信息。实验中调用的工具要么是模拟环境，要么是有适当使用许可且公开记录的 API。

我们的框架不会生成、存储或推断敏感个人属性，也不会针对任何特定人口群体。相反，ToolGate 专注于通过强制执行逻辑约束和可验证的执行条件，提升基于模型的工具调用的可靠性、可解释性和安全性。在评估过程中，我们严格遵守已发布的大语言模型、数据集、API 和基准测试平台所附带的许可证和使用条款。

此外，我们强调，ToolGate 的设计旨在增强 AI 行为的可信度，而非绕过防护措施或启用有害自动化。在实际系统中部署时，该方法可与额外的安全过滤器、审计流程和访问控制机制集成。我们认为，这有助于推动构建透明、可控且符合伦理的 AI 工具使用系统。

参考文献

Tom B. Brown、Benjamin Mann、Nick Ryder、Melanie Subbiah、Jared Kaplan、Prafulla Dhariwal、Arvind Neelakantan、Pranav Shyam、Girish Sastry、Amanda Askell、Sandhini Agarwal、Ariel Herbert-Voss、Gretchen Krueger、Tom Henighan、Rewon Child、Aditya Ramesh、Daniel M. Ziegler、Jeff Wu、Clemens Winter、Christopher Hesse、Mark Chen、Eric Sigler、Ma teusz Litwin、Scott Gray、Benjamin Chess、Jack Clark、Christopher Berner、Sam McCandlish、Alec Radford、Ilya Sutskever 和 Dario Amodei。2020。《语言模型是少样本学习者》。*ArXiv*，abs/2005.14165。

Aakanksha Chowdhery、Sharan Narang、Jacob Devlin、Maarten Bosma、Gaurav Mishra、Adam Roberts、Paul Barham、Hyung Won Chung、Charles Sutton、Sebastian Gehrmann、Parker Schuh、Kensen Shi、Sasha Tsvyashchenko、Joshua Maynez、Abhishek Rao、Parker Barnes、Yi Tay、Noam M. Shazeer、Vin- odkumar Prabhakaran、Emily Reif、Nan Du、Ben Hutchinson、Reiner Pope、James Bradbury、Jacob Austin、Michael I sard、Guy Gur-Ari、Pengcheng Yin、Toju Duke、Anselm Levskaya、Sanjay Ghemawat、Sunipa Dev、Henryk Michalewski、Xavier García、Vedant Misra、Kevin Robinson、Liam Fedus、Denny Zhou、Daphne Ippolito、David Luan、Hyeontaek Lim、Barret Zoph、Alexander Spiridonov、Ryan Sepassi、David Dohan、Shivani Agra wal、Mark Omernick、An- drew M. Dai、Thanumalayan Sankaranarayana Pillai、Marie Pellat、Aitor Lewkowycz、Erica Moreira、Re- won Child、Oleksandr Polozov、Katherine Lee、Zong- wei Zhou、Xuezhi Wang、Brennan Saeta、Mark Díaz、Orhan Firat、Michele Catasta、Jason Wei、Kathleen S. Meier-Hellstern、Douglas Eck、Jeff Dean、Slav Petrov 和 Noah Fiedel。2022 年。《PaLM：借助路径扩展语言建模》。*arXiv*，abs/2204.02311。

Davide Corsi、Enrico Marchesini 和 Alessandro Farinelli。2021。深度强化学习中安全关键任务的神经网络形式化验证。载于人工智能中的不确定性，第 333–343 页。PMLR。

Yu Du、Fangyun Wei 和 Hongyang Zhang。2024。Any-tool：面向大规模 API 调用的自我反思型分层智能体。载于第四十一届国际机器学习会议。

Lutfi Eren Erdogan、Nicholas Lee、Siddharth Jha、Se-hoon Kim、Ryan Tabrizi、Suhong Moon、Coleman Richard Charles Hooper、Gopala Anumanchipalli、Kurt Keutzer 和 Amir Gholami。2024。TinyAgent：边缘端的函数调用。载于《*2024* 年自然语言处理经验方法会议论文集：系统演示》，第 80–88 页。

谷歌公司。2025年。智能新时代：Gemini 3。https://blog.google/products/gemini/gemini-3/。访问日期：2025-11-18。

Danil S Grigorev、Alexey K Kovalev 和 Aleksandr I Panov。2025年。Verifyllm：基于大语言模型的机器人执行前任务计划验证。*arXiv* 预印本 *arXiv:2507.05118*。

Simeng Han、Hailey Schoelkopf、Yilun Zhao、Zhent- ing Qi、Martin Riddell、Wenfei Zhou、James Coady、David Peng、Yujie Qiao、Luke Benson 等。2024。Folio：基于一阶逻辑的自然语言推理。载于*2024*年自然语言处理实证方法会议论文集，第22017–22031 页。

Charles Antony Richard Hoare。1969。计算机编程的公理基础。《*ACM* 通讯》，12(10)：576–580。

Yue Huang、Chujie Gao、Siyuan Wu、Haoran Wang、Xiangqi Wang、Yujun Zhou、Yanbo Wang、Jiayi Ye、Jiawen Shi、Qihui Zhang 等。2025年。关于生成式基础模型可信度：指南、评估与观点。*arXiv* 预印本 *arXiv:2502.14296*。

Christine P. Lee、David Porfirio、Xinyu Jessica Wang、Kevin Chenkai Zhao 和 Bilge Mutlu。2025年。Veri-plan：将形式化验证和大语言模型融入终端用户规划。发表于*2025*年 *CHI* 人机交互系统因素会议论文集，CHI '25，美国纽约州纽约市。美国计算机协会。Chengpeng Li、Zhengyang Tang、Ziniu Li、Mingfeng Xue、Keqin Bao、Tian Ding、Ruoyu Sun、Benyou Wang、Xiang Wang、Junyang Lin 等。2025年。教语言模型使用工具进行推理。发表于第三十九届神经信息处理系统年会。Xiaohan Lin、Qingxing Cao、Yinya Huang、Haiming Wang、Jianqiao Lu、Zhengying Liu、Linqi Song 和 Xiaodan Liang。2024。FVEL：通过定理证明构建结合大语言模型的交互式形式化验证环境。神经信息处理系统进展，37：54932–54946。Aixin Liu、Aoxue Mei、Bangcai Lin、Bing Xue、Bingxuan Wang、Bingzheng Xu、Bochao Wu、Bowei Zhang、Chaofan Lin、Chen Dong 等。2025a。DeepSeek-v3.2：推动开放式大语言模型的前沿。*arXiv* 预印本 *arXiv:2512.02556*。Guangyi Liu、Pengxiang Zhao、Yaozhen Liang、Liang Liu、Yaxuan Guo、Han Xiao、Weifeng Lin、Yuxiang Chai、Yue Han、Shuai Ren 等。2025b。手机自动化中的大语言模型驱动图形用户界面智能体：进展与前景综述。*arXiv* 预印本 *arXiv:2504.19838*。Yanming Liu、Xinyue Peng、Jiannan Cao、Shi Bo、Yuwei Zhang、Xuhong Zhang、Sheng Cheng、Xun Wang、Jianwei Yin 和 Tianyu Du。2025c。工具规划器：利用多个工具之间的集群进行任务规划。发表于第十三届国际学习表征会议。

Ziyang Luo、Zhiqi Shen、Wenzhuo Yang、Zirui Zhao、Prathyusha Jwalapuram、Amrita Saha、Doyen Sahoo、SilvioSavarese、CaimingXiong 和 Junnan Li。2025年。MCP-universe：使用真实世界的模型上下文协议服务器对大型语言模型进行基准测试。载于智能体环境扩展研讨会。

Mark Huasong Meng、Guangdong Bai、Sin Gee Teo、Zhe Hou、Yan Xiao、Yun Lin 和 Jin Song Dong。2022。深度神经网络的对抗鲁棒性：形式化验证视角综述。*IEEE*可靠与安全计算汇刊。

OpenAI。2025年。GPT-5 正式推出。https://openai.com/index/introducing-gpt-5/。访问日期：2025-08-14。

Long Ouyang、Jeff Wu、Xu Jiang、Diogo Almeida、Carroll L. Wainwright、Pamela Mishkin、Chong Zhang、Sandhini Agarwal、Katarina Slama、Alex Ray、John Schulman、Jacob Hilton、Fraser Kelton、Luke E. Miller、Maddie Simens、Amanda Askell、Peter Welinder、Paul Francis Christiano、Jan Leike 和 Ryan J. Lowe。2022。通过人类反馈训练语言模型，使其遵循指令。*arXiv*，abs/2203.02155。

Bhargavi Paranjape、Scott Lundberg、Sameer Singh、Hannaneh Hajishirzi、Luke Zettlemoyer 和 Marco Tulio Ribeiro。2023 年。Art：面向大语言模型的自动多步推理与工具使用。*arXiv* 预印本 *arXiv:2303.09014*。Shishir G Patil、Tianjun Z hang、Xin Wang 和 Joseph E Gonzalez。2024年。Gorilla：连接海量 API 的大语言模型。神经信息处理系统进展，37：126544–126565。Cheng Qian、Emre Can Acikgoz、Qi He、Hongru WANG、Xiusi Chen、Dilek Hakkani-Tür、Gokhan Tur 和 Heng Ji。2025年。ToolRL：奖励是工具学习所需的一切。发表于第三十九届神经信息处理系统年会。Yujia Qin、Shengding Hu、Yankai Lin、Weize Chen、Ning Ding、Ganqu Cui、Zheni Zeng、Xuanhe Zhou、Yufei Huang、Chaojun Xiao 等。2024年。基于基础模型的工具学习。*ACM* 计算调查，57(4)：1–40。Yujia Qin、Shihao Liang、Yining Ye、Kunlun Zhu、Lan Yan、Yaxi Lu、Yankai Lin、Xin Cong、Xiangru Tang、Bill Qian 等。2023年。ToolLLM：帮助大语言模型掌握 16000+ 现实世界 API。发表于第十二届国际学习表征会议。Timo Schick、Jane Dwivedi-Yu、Roberto Dessì、Roberta Raileanu、Maria Lomeli、Eric Hambro、Luke Zettlemoyer、Nicola Cancedda 和 Thomas Scialom。2023年。Toolformer：语言模型可以自学

使用工具。神经信息处理系统进展，36：68539–68551。

Freda Shi、Xinyun Chen、Kanishka Misra、Nathan Scales、David Dohan、Ed H Chi、Nathanael Schärli 和 Denny Zhou。2023。大语言模型很容易受到无关上下文的干扰。载于国际机器学习会议，第 31210–31227 页。PMLR。

Zhaochen Su、Linjie Li、Mingyang Song、Yunzhuo Hao、Zhengyuan Yang、Jun Zhang、Guanjie Chen、Jiawei Gu、Juntao Li、Xiaoye Qu 等。2025年。Open-thinkimg：通过视觉工具强化学习，学习借助图像进行思考。*arXiv* 预印本 *arXiv:2505.08617*。

Anand Swaroop、Abhishek Singh、Girish Chandra、Shiv Prakash、Sohan Kumar Yadav、Tiansheng Yang 和 Rajkumar Singh Rathore。2024。一篇关于用于验证和优化的形式化方法与深度学习的全面综述。载于*2024* 年决策辅助科学与应用国际会议（*DASA*），第 1–6 页。IEEE。

Po-Wei Wang、Priya Donti、Bryan Wilder 和 Zico Kolter。2019。Satnet：利用可微 SAT 求解器连接深度学习与逻辑推理。载于国际机器学习会议，第 6545–6554 页。PMLR。

Qiancheng Xu、Yongqi Li、Heming Xia 和 Wenjie Li。2024。利用大语言模型的迭代反馈增强工具检索。载于计算语言学协会年会论文集：*EMNLP 2024*，第 9609–9619 页。

An Yang、Anfeng Li、Baosong Yang、Beichen Zhang、Binyuan Hui、Bo Zheng、Bowen Yu、Chang Gao、Chengen Huang、Chenxu Lv 等。2025年。通义千问3 技术报告。*arXiv* 预印本 *arXiv:2505.09388*。

Hsiu-Wei Yang、Abhinav Agrawal、Pavlos Fragkogian-nis 和 Shubham Nitin Mulay。2024a。AI 模型能否欣赏文档美学？关于易读性和版面质量与预测置信度之间关系的探索。*ArXiv*，abs/2403.18183。

Yu'an Yang、Siheng Xiong、Ali Payani、Ehsan Shareghi 和 Faramarz Fekri。2024b。Harnessing the power of large language models for natural language to fi rst-order logic translation（利用大语言模型的力量，将自然语言翻译为一阶逻辑）。载于计算语言学协会第*62*届年会论文集（第*1*卷：长篇论文），第 6942–6959 页。

Shunyu Yao、Jeffrey Zhao、Dian Yu、Nan Du、Izhak Shafran、Karthik R Narasimhan 和 Yuan Cao。2022。React：让语言模型中的推理与行动相辅相成。载于第十一届国际学习表征会议。

Junjie Ye、Guanyu Li、Songyang Gao、Caishuang Huang、Yilong Wu、Sixian Li、Xiaoran Fan、Shihan Dou、Tao Ji、Qi Zhang等。2025年。Tooleyes：面向大型语言模型在真实场景中的工具学习能力进行细粒度评估。收录于第*31*届国际计算语言学会议论文集，第156–187页。

Yuanqing Yu、Zhefan Wang、Weizhi Ma、Shuai Wang、Chuhan Wu、Zhiqiang Guo 和 Min Zhang。2025年。Steptool：通过步骤粒度强化学习提升大语言模型的多步工具使用。载于第*34*届美国计算机协会国际信息与知识管理会议论文集，CIKM '25，第 3952–3962 页，纽约，美国。美国计算机协会。

Chaoyun Zhang、Shilin He、Liqun Li、Si Qin、Yu Kang、Qingwei Lin、Saravan Rajmohan 和 Dong mei Zhang。2025a。API智能体与图形用户界面智能体：分歧与趋同。载于*ICML 2025*计算机使用智能体研讨会。

Yanzhao Zhang、Mingxin Li、Dingkun Long、XinZhang、Huan Lin、Baosong Yang、Pengjun Xie、An Yang、Dayiheng Liu、Junyang Lin等。2025b。通义千问3嵌入：通过基础模型推进文本嵌入与重排序。*arXiv*预印本 *arXiv:2506.05176*。

Andy Zhou、Kai Yan、Michal Shlapentokh-Rothman、Haohan Wang 和 Yu-Xiong Wang。2024。语言智能体树搜索统一了语言模型中的推理、行动与规划。载于国际机器学习会议，第 62138–62160 页。PMLR。

Xiaochen Zhu、Caiqi Zhang、Tom Stafford、Nigel Collier 和 Andreas Vlachos。2025年。大语言模型中的一致性。载于计算语言学协会第*63*届年会论文集（第*1*卷：长篇论文），第3854–3872页。

Yuchen Zhuang、Xiang Chen、Tong Yu、Saayan Mitra、Victor Bursztyn、Ryan A Rossi、Somdeb Sarkhel 和 Chao Zhang。2024。ToolChain*：利用 A*搜索实现大语言模型中的高效动作空间导航。载于第十二届国际学习表征会议。

A 数据集详述

A.1 ToolBench 数据集

ToolBench (Qin 等人，2023）是 ToolLLM 框架提出的大规模指令微调与评估数据集，旨在系统评估和增强大语言模型在真实环境中的工具选择、参数规划和可执行 API 调用能力。在构建过程中，ToolBench 首先从 RapidAPI Hub 收集了 **16,464 个真实 RESTful API**，涵盖 **49 个功能类别**（例如天气、社交媒体、电子商务和地图服务），并提取包括 API 名称、文档、参数模式在内的结构化元数据，

以及使用示例。基于这些 API，利用大语言模型自动生成自然语言任务指令，并采用基于深度优先搜索的决策树（DFSDT）来发现可行的**工具调用轨迹**作为解决方案路径。

在规模方面，ToolBench 在大型 API 空间中提供超过**126K 个指令—解决方案路径对**，并设计了多个训练/测试划分，用于测试模型在面对未见过的指令、工具，乃至工具类别时的泛化能力。此外，ToolBench 还包含一个“经典任务集”，涵盖**8 个具有代表性的工具环境**，例如 OpenWeather、VirtualHome 和 WebShop。每个环境包含约 100 条经人工验证的任务指令和 7–15 个工具接口，可用于开展更细致的消融实验和比较研究。

在评估方面，ToolBench 集成了 ToolEval 框架，对生成的 API 调用序列进行执行级评估。常见指标包括**通过率**（任务完成率）和**胜率**。一些研究还进一步采用“Plan.EM”和“Act.EM”，以区分规划质量与执行质量。凭借贴近实际且规模庞大的 API 空间，ToolBench 已成为后续一系列工具使用研究广泛采用的基准测试和数据源。

A.2 MCP-Universe 基准测试

MCP-Universe（Luo 等人，2025年）是一个用于评估大语言模型在**模型上下文协议（MCP）**范式下能力的综合基准测试，重点考察模型通过交互完成复杂任务的能力，交互对象包括真实的MCP 服务器。与离线工具使用数据集不同，MCP-Universe 直接连接到真实运行的 MCP 服务，强调长时程交互、未知工具发现，以及在动态环境中的稳健执行。

MCP-Universe 涵盖**6 个核心任务领域**和**11 个不同的 MCP 服务器**，包括位置导航、代码仓库管理、金融分析、3D 设计、浏览器自动化和网页搜索。该基准测试总共包含**231 个任务实例**，并基于环境与工具的不同组合形成多种基准测试配置。此外，该基准测试定义了**84 个独特的评估器**，用于覆盖不同的评估维度，例如结构正确性、逻辑合理性，以及与动态数据源的一致性。

从任务分布来看，该基准测试在保持合理平衡的同时，力求具有代表性：网页搜索任务约占 23.8%（55 项任务），位置导航占 19.5%（45 项任务），金融分析占 17.3%（40 项任务），浏览器自动化占 16.9%（39 项任务），代码仓库管理占 14.3%（33 项任务），3D 设计占 8.2%（19 项任务）。任务通常要求智能体经过多轮操作，与多个 MCP 工具交互，以完成路线规划、代码仓库操作、投资组合分析或自动化浏览器操作等复杂目标。

MCP-Universe 进一步区分三**类执行级评估器**:

(1)格式评估器，检查模型输出是否遵循 MCP 调用规范；

(2) 静态评估器，验证时不变任务的正确性；

(3) 动态评估器，查询实时数据源，为时间敏感任务构建真实值，例如金融价格或导航。

B 基线方法详述

我们将本方法与多种最先进且全面的基线方法进行比较，涵盖以下基准测试设置:

ReACT (Yao 等人，2022），该方法交替进行推理思维与执行行动，在语言推理和工具调用之间形成线性交互过程。它是工具增强型大语言模型最广泛使用的基线方法之一。

DFSDT (Qinet al.，2023），该方法采用深度优先搜索机制来探索推理—工具轨迹。每当模型走入错误路径时，DFSDT 就会将完整的失败历史呈现给模型，从而支持重新规划并最大限度地扩展探索空间。

LATS (Zhou 等人，2024），该方法利用前瞻式树搜索扩展多个候选工具序列，并评估其预期效果，在复杂的多步骤场景中展现出强大的规划能力。

ToolChain* (Zhuang 等人，2024），它明确构建工具链来建模多步依赖关系，并引导大语言模型完成序列化工具执行。尽管它增强了多工具任务的结构化推理能力，但其有效性仍主要依赖大语言模型驱动的自然语言推理，而非形式化执行约束。

工具规划器(Liu等人，2025年c），它引入明确的外部规划模块来控制工具序列生成，结合检索、候选项筛选和结构化规划策略，以提升全局执行连贯性和决策可靠性。

C 环境

所有实验均在统一的工具执行环境中进行。ToolBench API 被视为可调用的功能节点，而 MCP- Universe 工具则在官方沙箱中执行，并提供真实的执行反馈，包括逼真的执行延迟、工具失败信号以及随状态变化的输出。所有模型均通过其官方 API 访问，解码气温固定为 0.2，以尽可能减少推理和工具规划行为中的随机性。对于每项基准测试，所有系统均采用相同的任务指令、工具描述、执行限制和终止条件。

对于检索模块，我们采用基于嵌入的检索框架来构建工具语义表示。具体而言，我们使用 Qwen3-embedding-0.6b (Zhang 等人，2025年b) 模型，将工具描述、功能语义、参数规范和使用文档编码为稠密向量。在推理过程中，中间工具需求表示会被编码到相同的嵌入空间中，并通过余弦相似度检索 Top-K 个候选工具。除非另有说明，默认设置为 $K = 10$。检索完成后，我们会应用重排序模型，在缩小后的候选集中提升工具选择准确率。我们采用基于 Qwen3-Reranker-0.6B (Zhang 等人，2025年b) 构建的轻量级基于大语言模型的重排序器，它会综合考虑当前推理上下文、符号状态和候选工具语义，以评估其上下文适用性。

D 详细拒绝分析

D.1 前置条件 $\{P\}$ 验证

前置条件检查占所有拒绝的**17.6%**，发挥着“静态防火墙”的作用，在无效行动执行前将其拦截。

- **参数幻觉（8.4%）：** 这是最常见的错误类型。面对庞大的工具集时，大语言模型往往凭直觉生成参数，例如臆造文件 ID 或目录路径，而不是基于检索结果生成。通过强制验证符号链接，$\{P\}$ 会在执行前拦截此类请求，从而显著降低计算开销和令牌消耗。

- **状态依赖违规（4.1%）：** 模型偶尔会绕过必要的操作序列，例如在未先获取所需权限或句柄的情况下尝试修改文件。$\{P\}$ 会施加严格的逻辑和时间约束，确保每次调用都以有效的环境状态为前提。

D.2 后置条件 $\{Q\}$断言

尽管执行后断言触发的频率较低（**11.8%**），但它们能够处理复杂的逻辑错误，而传统搜索模型（如 DFSDT）通常无法检测到这些错误。

- **静默失败（6.3%）：** 在复杂任务中，API 往往会返回成功状态码（例如 HTTP 200），但响应内容却为空或毫无意义（例如 results:[]）。若没有 $\{Q\}$ 验证，智能体可能会将其误认为任务取得了进展，进而沿着徒劳无功的搜索路径继续前进。$\{Q\}$ 断言要求检查结果是否非空，从而识别这些“逻辑空洞”并立即触发回溯。

- **语义与状态对齐（5.5%）：** 这包括语义约束不匹配（3.7%）和状态更新不一致（1.8%）。这证实，$\{Q\}$能够捕捉工具输出与用户意图之间的细微偏差，确保搜索轨迹始终沿着正确的语义路径前进。

D.3 协同效应与逻辑闭合

我们的分析表明，$\{P\}$ 和 $\{Q\}$ 构成了稳健的逻辑闭环。$\{P\}$ 的高拒绝率（**17.6%**）主要通过剪除明显的错误分支来提升**搜索效率**，从而节省令牌和时间。相反，$\{Q\}$ 的精确度驱动拦截（**11.8%**）是**任务成功**的主要决定因素。通过识别技术上成功但逻辑上存在缺陷的步骤，$\{Q\}$ 可防止级联错误的累积成为未经验证的智能体系统的主要瓶颈。

D.4 提示模板

工具调用流程

大语言模型推理的系统提示词
你是一位乐于助人的助手，可以使用工具回答用户的问题 .

你可以使用一组工具，以及用于跟踪已验证事实的符号状态。

重要控制标记：
- 需要调用工具时，请输出： `<start_call_tool>`- 提供工具结果后，结果将以以下形式包裹：`<start_tool_result>...</end_tool_result>`- 使用完工具后，请输出： `<end_call_tool>`

状态信息：
当前符号状态包含已验证事实。使用这些信息来：1. 检查是否有足够的信息直接回答 2. 确定缺少哪些信息，以及需要通过工具检索哪些信息 3. 了解根据当前状态可以调用哪些工具

工具调用过程：
1. 思考你需要哪些信息 2. 输出 `<start_call_tool>` ，后面跟上对所需信息的简要描述 3. 等待工具结果 4. 根据新信息继续推理 5. 如有需要，重复上述步骤；否则提供最终答案

重要提示：工具失败时——继续尝试！ - 如果工具调用失败，不要立即放弃。可以考虑：* 尝试能提供类似信息的其他工具 * 使用不同参数再次尝试同一工具 * 尝试其他方法或搜索策略 * 思考还有哪些信息来源可能有帮助 - 只有在确信自己已经：* 成功检索到所需信息，或 * 已用尽所有合理的工具选项，并能根据现有信息给出有帮助的回答 时，才提供最终答案 - 不要仅仅因为一个工具调用失败就过早结束推理 - 坚持尝试并灵活寻找替代方案

输出格式：
- 如果可以直接回答：直接提供答案，不要使用 `<start_call_tool>`- 如果需要使用工具：输出 `<start_call_tool>`，然后说明要使用什么工具及其推理过程 - 如果工具失败：考虑替代方案，并使用`<start_call_tool>`重试

工具调用流程

用户消息格式示例
当前状态： 查询：在 YouTube 上查找一个关于机器学习的教程视频
主题：机器学习 平台：YouTube 内容_类型：教程视频

工具结果：（首次
调用时为空）

用户查询： 在 YouTube 上找一个关于机器学习的教程视频

你的任务：
请逐步思考。如果需要使用工具，请输出 `<start_call_tool>`
随后描述你的需求。
如果工具失败，　　请考虑替代方案并尝试其他工具
然后再放弃。
只有在你确信已经掌握足够的
信息或已经穷尽所有合理选项后，才提供最终答案。

E 规划流程

算法1形式化描述了我们的前向执行框架，该框架将契约验证整合到大语言模型的推理循环中。该流程首先根据用户查询和上下文初始化环境状态 S_0。在每一步 k，大语言模型充当控制器，决定是通过回答来结束任务，还是调用外部工具。

算法 1 带契约验证的正向执行

要求： 查询 q，上下文 H，最大迭代次数 $K_{\max}$，工具集 $\mathcal{T}$
确保： 最终答案 a 或失败信号

$S_0 \leftarrow$ 初始化状态(q, H), $\mathcal{T}_{\text{failed}} \leftarrow \emptyset$
对于 $k = 0$ **到** $K_{\max} - 1$ **执行**
  执行$_k \leftarrow$ 大语言模型_推理($q, H,$ 摘要(S_k))　▷ 生成推理步骤
  如果执行$_k$ 是答案(a)**那么**
    返回 a　▷ 任务已成功完成
  结束如果
  如果act$_k$ 是 调用工具(描述) **那么**
    $\mathcal{T}_{\text{cand}} \leftarrow$ 重排序(检索(q, S_k))　▷ 前-N 个候选工具
    是_已更新 $\leftarrow$ 假
    对于 $t \in \mathcal{T}_{\text{cand}} \setminus \mathcal{T}_{\text{failed}}$ **执行**　▷ 遍历有效候选项
      如果 $S_k \not\models \phi_{pre}\ (t)$ **则**　▷ 检查前置条件契约
        继续
      结束如果
      $\theta_t \leftarrow$ 生成参数(t, S_k)　▷ 合成参数
      $r_t \leftarrow$ 执行(t, θ_t)　▷ 工具调用
      如果 验证(S_k, r_t, $\phi_{post}(t)$) **则**　▷ 验证后置条件契约
        $S_{k+1} \leftarrow$ 更新(S_k, r_t)　▷ 提交状态转换
        是_已更新 $\leftarrow$True
        中断
      else
        $\mathcal{T}_{\text{failed}} \leftarrow \mathcal{T}_{\text{failed}} \cup \{t\}$　▷ 将工具标记为不可靠
      结束条件判断
    结束循环
    如果未 is_已更新**则**
      返回 失败　▷ 没有可满足契约的工具
    结束 if
  结束 if
结束 for
返回超时

F 示例

工具调用流程

工具调用示例 I

- **查询：** “帮我在 YouTube 上找一个关于机器学习的教程视频”
- **步骤 0：初始状态**

  S_0 ={ "query": "帮我找一个关于机器学习的教程视频", "topic": "机器学习", " platform": "YouTube", "content_type": "教程视频" }

- **步骤 1：大语言模型推理**

  对话历史：[]
  大语言模型回复：
  “我需要在 YouTube 上搜索机器学习教程视频。
  让我使用 YouTube 搜索工具查找相关视频。

  <开始 _call_工具>
  我需要在 YouTube 上搜索机器学习教程视频
  <end_call_工具>"

  行动：开始 _call_tool

- **步骤 1：工具检索**

  检索到的候选项（重排序后的前 5 个）：
  1. 简单_YouTube_搜索_搜索（得分：0.92）
  2. YouTube_视频_信息（评分：0.85）
  3. YouTube_频道_搜索（评分：0.78）
  4. 视频_搜索_API（评分：0.72）
  5. 内容_搜索（评分：0.68）

- **步骤 1：前置条件检查**

  检查 Simple_YouTube_搜索_搜索：
  前置条件：存在（查询）
  状态 S_0: {"query": "帮我找一个教程视频.....", ...}
  检查：exists(query) = true

- **步骤 1：工具执行**

  工具：Simple_YouTube_搜索_搜索
  参数：{"query": “机器学习教程”}

工具配置：{ "category": "数据", "api_name": "搜索", "tool_input": { "query": "机器学习教程", "type": "", "safesearch": false } } 工具结果：{ "results": [{ "title": "初学者机器学习教程", "url": "https://youtube.com/watch?v=abc123", "channel": {"name": "ML Course", "id": "UC123"}, "views": 1500000, "duration_formatted": "45:30" }, { "title": "完整机器学习课程——全套教程", "url": "https://youtube.com/watch?v=def456", "channel": {"name": "AI Academy", "id": "UC456"}, "views": 800000, "duration_formatted": "12:30:00" }]}

• **步骤 1：后置条件验证**后置条件 Q_t：结构检查：has_field("results") && is_list(r_t.results) 语义检查：对于 r_t.results 中的所有视频：has_field(video, "title") && has_field(video, "url") 验证结果：成功 - 结构检查：存在 "results" 字段，且其为列表 - 语义检查：所有视频都有 "title" 和 "url" 字段

• **步骤 1：状态更新**S_1 =更新_t(S_0, r_t) = { "query": "在 YouTube 上为我查找一个关于机器学习的教程视频

", "topic": "机器学习", "platform": "YouTube", "content_类型": "教程视频", "youtube_results": [

{"title": "初学者机器学习教程", ...}, {"title": "CompleteML课程——完整教程", ...}],"result_count": 19 }

- **步骤2：大语言模型推理（含工具结果）**对话历史：[{"role": "user", "content": "帮我找一个教程视频……"}, {"role": "assistant", "content": "我需要搜索……\n<start_call_tool>...<end_call_tool>"}, {"role": "user", "content": "<start_tool_result>\n{\"results\": [...]}\n<end_tool_result>"}]大语言模型回复："太好了！我在 YouTube 上找到了几部机器学习教程视频。根据 YouTube 的搜索结果，我为你找到了几部机器学习教程视频：1.《初学者机器学习教程》——150 万次观看，45:30 频道：ML Course 网址：https://youtube.com/watch?v=abc123 2.《完整机器学习课程——全套教程》——80 万次观看，12:30:00 频道：AI Academy 网址：https://youtube.com/watch?v=def456 这些教程涵盖了机器学习的基础概念和实际应用。" 行动：回答

- **最终状态**

S_最终 = { "query": "查找有关 YouTube 上机器学习的教程视频", "topic": "机器学习", "platform": "YouTube", "content_type": "教程视频", "youtube_results": [...],"result_count": 19 }

工具调用流程

后置条件失败时的工具调用流程

- **查询：** “今天纽约的天气怎么样？”
- **步骤 0：初始状态**

```
S_0 ={
    "查询":   "今天纽约的天气怎么样？",
    "位置":       "纽约"      ,
    "日期":  "今天"
}
```

- **步骤 1：大语言模型推理**

对话历史： []
大语言模型响应：
“我需要获取纽约当前的天气信息。
我来使用天气工具获取这些数据。

<开始 _call_工具>
获取纽约天气信息
<end_call_工具>"

行动：开始 _call_tool

- **步骤 1：工具检索**

检索到的候选项（重排序后的前 5 名）：
1. 天气_API_获取当前天气（得分：0.91）
2. OpenWeatherMap_当前天气（得分：0.88）
3. 天气服务_获取天气预报（分数：0.82）
4. 气候数据_获取天气（分数：0.75）
5. 天气信息_查询（分数：0.71）

- **步骤 1：前置条件检查（工具 1）**

正在检查天气_API_获取当前天气：
  前置条件：存在（位置）
  状态 S_0: {"位置": "纽约" , ...}
  检查：存在(位置) = 真

- **步骤 1：工具执行（工具 1）**

工具：天气 _API_获取当前天气
参数：{"location": "纽约"}
工具配置：{

"类别": "天气", "api_name": "获取当前天气", "tool_input":{"位置": "纽约"} } 工具结果: { "状态": "成功", "数据": { "城市":"纽约", "气温": 72, " 天气状况 ": " 晴朗 " } }

• **步骤 1：后置条件验证（工具1）**后置条件 Q_t：结构检查：has_field("气温") && has_field("天气状况") && has_field("湿度") 语义检查：气温为数值 && 天气状况为字符串 验证结果：失败 - 结构检查：缺少必需字段 "湿度" - 语义检查：气温和天气状况均有效 错误：工具结果不包含必需字段 "湿度"

• **步骤1：工具标记为失败**失败的工具：{Weather_API_获取当前天气} 状态保持不变：S_0（无更新）

• **步骤 1：尝试下一个工具 - 前置条件检查（工具2）**检查 OpenWeatherMap_当前天气：前置条件：exists(位置) 状态 S_0: {"位置": "纽约", ...} 检查：exists(位置) = 为真

• **步骤 1：工具执行（Tool2）**工具：OpenWeatherMap_当前天气 参数： {" location": "纽约"} 工具配置： {

"category": "天气", "api_name": "当前天气", "tool_input":{"location": "纽约"} } 工具结果： { "temperature": 72, "condition": "晴朗", "humidity": 65, "wind_speed": 10, "pressure": 1013, "location": "纽约, NY" }

• **步骤1：后置条件验证（工具2）**后置条件 Q_t：结构：包含_字段("temperature") && 包含_字段("condition") && 包含_字段("humidity") 语义：temperature 为数字 && condition 为字符串 && humidity 为数字 验证结果：成功 - 结构检查：所有必需字段均存在 - 语义检查：所有字段的类型均正确

• **步骤1：状态更新**S_1 =更新_t(S_0, r_t) = { "query": "今天纽约的天气怎么样？", "location": "纽约", "date": "今天", "temperature": 72, "condition": "晴朗", "humidity": 65, "wind_speed": 10, "pressure": 1013 }

• **步骤2：大语言模型推理（包含工具结果）**

对话历史： [{"role": "user", "content": "今天纽约的天气怎么样？"}, {"role": "assistant", "content": "我需要获取当前天气……\n<start_call_tool>...<end_call_tool>"},

{"role": "user"，"content": "<start_工具_result>\n{\"气温\": 72 ，
\"天气状况\": \"晴朗\", ...}\n<end_工具_result>"}
]

大语言模型响应：
“纽约今天天气晴朗，气温为 72°F。
湿度为 65%，风速为 10 mph，大气压
为 1013 hPa。 今天适合进行户外活动。”

行动：回答

- **最终状态**

S_最终 = {
"查询": "今天纽约的天气怎么样？",
"位置": "纽约" ,
"日期": "今天",
"气温": 72 ,
"天气状况": "晴朗",
"湿度": 65,
"风_速": 10 ,
“气压”：1013
}

G 使用霍尔逻辑与一阶契约进行形式化推导

在本附录中，我们给出若干具有代表性的推导，明确阐述 ToolGate 的逻辑基础。我们将单步工具执行、轨迹级安全性和不变量形式化为基于符号状态空间 Σ 及执行轨迹的一阶逻辑（FOL）公式和霍尔式证明义务。

G.1 符号与基本设定

我们回顾一下，可信符号状态是一个类型化键值映射 $S \in \Sigma$，其中

$$\Sigma \;=\; \{(k, v, \sigma)\}.$$

我们用 $S \models \varphi$ 表示一阶状态公式 φ 在 S中为真。工具 t 关联着一个霍尔式契约

$$\{P_t\}\; t\; \{Q_t\},$$

其中 $P_t(S)$ 是一个状态谓词（前置条件），而 $Q_t(S,\ r_t)$ 是关于前置状态S 和运行时结果 r_t的后置条件谓词:

$$Q_t : \Sigma \times R_t \to \{\text{true}, \text{false}\}.$$

我们将 $(S,\ r_t) \models Q_t$ 简写为 $Q_t(S,\ r_t) = 1$。

为了将逻辑验证与状态构造解耦，我们引入一个确定性状态更新算子

$$\mathsf{Update}_t : \Sigma \times R_t \to \Sigma,$$

它规定了将有效结果 r_t 整合到 S 后所生成的新可信符号状态。

我们称工具 t的（理想）运行时执行器为一个（可能是部分的）函数

$$\mathsf{Exec}(t, S) = r_t$$

该函数在状态 S下调用 t时返回运行时结果 r_t。

G.2 单步霍尔式推导

我们首先阐明在我们的设定中，单次工具调用的标准霍尔式证明义务。

单步可靠性义务。 若以下一阶逻辑公式成立，则契约 $\{P_t\}t\{Q_t\}$ 相对于 Exec 和 Update$_t$ 是可靠的:

$$\forall S, r_t.\ \Big(S \models P_t \ \wedge\ r_t = \mathsf{Exec}(t, S) \ \wedge\ Q_t(S, r_t)\Big) \ \Rightarrow\ \mathsf{GoodState}\big(\mathsf{Update}_t(S, r_t)\big), \tag{15}$$

其中，良好状态表示更新后的状态类型正确且保持一致（例如，满足键唯一性和类型健全性等全局不变量）。

单个 ToolGate 步骤的推理规则。 我们可以用以下霍尔式推导规则来表示 ToolGate 对单次工具调用的操作步骤:

$$\frac{S \models P_t \quad r_t = \mathsf{Exec}(t, S) \quad Q_t(S, r_t) \quad \mathsf{Inv}(S) \Rightarrow \mathsf{Inv}\big(\mathsf{Update}_t(S, r_t)\big)}{\{\, P_t(S) \wedge \mathsf{Inv}(S) \,\}\; t\; \{\, \mathsf{Inv}(S') \wedge Q_t(S, r_t) \wedge S' = \mathsf{Update}_t(S, r_t) \,\}}\ \textsc{Tool-Step}$$

其中，Inv 是任意选定的状态不变量（例如，S 仅包含已验证的工具结果）。

在小步转移形式中，单个 ToolGate 步骤可写为

$$\langle S_k, R_k \rangle \ \xrightarrow{t, r_t}\ \langle S_{k+1}, R_{k+1} \rangle$$

证明树如下:

$$\frac{S_k \models P_t \quad r_t = \mathsf{Exec}(t, S_k) \quad Q_t(S_k, r_t) \quad S_{k+1} = \mathsf{Update}_t(S_k, r_t) \quad R_{k+1} = R_k \cdot \langle t, r_t \rangle}{\langle S_k, R_k \rangle \xrightarrow{t, r_t} \langle S_{k+1}, R_{k+1} \rangle}\ \textsc{Tool-Exec}$$

G.3 前置条件过滤作为最弱前置条件

ToolGate 中的工具选择受前置条件 P_t 的约束。我们可以用最弱前置条件来表述这一点。令 $\mathsf{wp}(t,\ \Phi)$ 表示工具 t 相对于期望的后状态公式 $\Phi(S')$ 的最弱前置条件。那么：

$$\mathsf{wp}(t,\Phi)(S) \ \triangleq\ \exists r_t.\ \Big(S \models P_t \wedge r_t = \mathsf{Exec}(t,S) \wedge Q_t(S,r_t) \wedge \Phi\big(\mathsf{Update}_t(S,r_t)\big)\Big).$$

特别是，要求 t 在 S 中可执行，对应于

$$S \models \mathsf{wp}(t,\top) \iff \exists r_t.\ S \models P_t \wedge r_t = \mathsf{Exec}(t,S) \wedge Q_t(S,r_t).$$

那么，ToolGate 的前置条件过滤器可以表示为：

$$\forall t \in C_k.\ \mathsf{Admissible}(t,S_k) \ \triangleq\ S_k \models \mathsf{wp}(t,\top). \tag{16}$$

G.4 后置条件作为接受事件

ToolGate 中的运行时接受谓词 A_t 定义如下：

$$A_t(S_k,r_t) \ =\ \big(Q_t(S_k,r_t) \wedge \mathsf{wf}(r_t)\big),$$

其中，wf 表示 r_t 的结构和格式良构性。

我们将状态更新规则定义为

$$S_{k+1} = \begin{cases} \mathsf{Update}_t(S_k,r_t) & \text{if } A_t(S_k,r_t)=1, \\ S_k & \text{otherwise.} \end{cases}$$

该规则可以用以下霍尔三元组表示：

$$\{\, S_k \models P_t \,\}\ t\ \{\, A_t(S_k,r_t)=1 \Rightarrow \big(S_{k+1} = \mathsf{Update}_t(S_k,r_t) \wedge Q_t(S_k,r_t)\big) \,\}. \tag{17}$$

等价地，用一阶逻辑表示：

$$\begin{aligned} \forall S_k, r_t, S_{k+1}.\ & S_k \models P_t \ \wedge\ r_t = \mathsf{Exec}(t,S_k) \ \wedge\ A_t(S_k,r_t)=1 \\ & \Rightarrow\ \big(S_{k+1} = \mathsf{Update}_t(S_k,r_t) \wedge Q_t(S_k,r_t) \wedge \mathsf{GoodState}(S_{k+1})\big). \end{aligned} \tag{18}$$

G.5 轨迹级安全性推导

完整的 ToolGate 执行会产生一条轨迹

$$\tau = \big((S_0,R_0),(t_0,r_0,A_0),\ldots,(S_n,R_n)\big).$$

逐步安全性。当且仅当以下条件成立时，我们称步骤 k 是安全的：

$$\begin{aligned} \mathsf{SafeStep}_k(\tau) \triangleq \Big(& S_k \models P_{t_k} \ \wedge\ r_k = \mathsf{Exec}(t_k,S_k) \ \wedge\ A_{t_k}(S_k,r_k)=1 \\ & \Rightarrow \big(Q_{t_k}(S_k,r_k) \wedge S_{k+1} = \mathsf{Update}_{t_k}(S_k,r_k)\big)\Big). \end{aligned} \tag{19}$$

全局安全性。轨迹级安全性如下：

$$\mathsf{Safe}(\tau) \ \triangleq\ \bigwedge_{k=0}^{n-1} \mathsf{SafeStep}_k(\tau). \tag{20}$$

可靠性定理（概要）。 如果所有工具契约均可靠（公式 15），且初始状态 S_0 满足全局不变量 Inv，那么每条可达的 ToolGate 轨迹都是安全的：

$$\forall \tau.\ \mathsf{Reach}(q, H, \tau) \wedge S_0 \models \mathsf{Inv} \ \Rightarrow\ \mathsf{Safe}(\tau) \wedge \bigwedge_{k=0}^{n} \mathsf{Inv}(S_k). \tag{21}$$

这可以通过对 k 使用 TOOL-STEP 规则进行归纳证明：

$$\frac{\mathsf{Inv}(S_0) \quad \forall k.\ \mathsf{SafeStep}_k(\tau) \wedge \mathsf{Inv}(S_k) \Rightarrow \mathsf{SafeStep}_{k+1}(\tau) \wedge \mathsf{Inv}(S_{k+1})}{\forall k.\ \mathsf{Reach}_k(q, H, \tau) \Rightarrow \mathsf{SafeStep}_k(\tau) \wedge \mathsf{Inv}(S_k)}\ \textsc{Induction}$$

G.6 具体工具的契约实例化

举例来说，考虑一个（简化的）代码仓库管理工具 ListFiles，其契约如下：

$$\{P_{\mathsf{list}}\}\ \mathsf{ListFiles}\ \{Q_{\mathsf{list}}\}.$$

设符号状态包含一个表示当前工作目录的键“cwd”，以及一个用于表示符号文件系统抽象的键“fs”。我们实例化：

$$P_{\mathsf{list}}(S) \triangleq \exists d.\ \big(d = S[\mathsf{cwd}] \wedge d \in \mathsf{Dom}(S[\mathtt{fs}])\big), \tag{22}$$

$$Q_{\mathsf{list}}(S, r) \triangleq \exists d, L.\ d = S[\mathsf{cwd}] \wedge L = \mathsf{LookupDir}(S[\mathtt{fs}], d) \wedge r = L, \tag{23}$$

并将相应的状态更新算子定义为

$$\mathsf{Update}_{\mathsf{list}}(S, r)\ =\ S \cup \{(\mathtt{last_ls}, r, \mathsf{ListType})\}.$$

该工具对应的霍尔三元组为：

$$\{\, P_{\mathsf{list}}(S) \,\}\ \mathsf{ListFiles}\ \left\{\, Q_{\mathsf{list}}(S, r) \wedge S' = \mathsf{Update}_{\mathsf{list}}(S, r) \wedge \mathsf{Inv}(S') \,\right\}.$$

安全调用的一阶逻辑推导。假设我们处于步骤 k，状态为 S_k，且

$$S_k \models P_{\mathsf{list}}.$$

具体调用为：

$$r_k = \mathsf{Exec}(\mathsf{ListFiles}, S_k).$$

后置条件检查与接受给出：

$$\begin{aligned}(S_k, r_k) \models Q_{\mathsf{list}} \ \Rightarrow\ \exists d, L.\ &\big(d = S_k[\mathsf{cwd}]\big) \\ &\wedge \big(L = \mathsf{LookupDir}(S_k[\mathtt{fs}], d)\big) \\ &\wedge \big(r_k = L\big), \\ A_{\mathsf{list}}(S_k, r_k) = 1 \ \Rightarrow S_{k+1} = \mathsf{Update}_{\mathsf{list}}(S_k, r_k) &= S_k \cup \{(\mathtt{last_ls}, r_k, \mathsf{ListType})\},\end{aligned} \tag{24}$$

由此可共同推得，S_{k+1} 是 S_k的良构扩展。

综合以上内容，TOOL-EXEC 规则可具体表述为：

$$\frac{\begin{array}{l}S_k \models P_{\mathsf{list}} \quad r_k = \mathsf{Exec}(\mathsf{ListFiles}, S_k) \\ (S_k, r_k) \models Q_{\mathsf{list}} \quad S_{k+1} = \mathsf{Update}_{\mathsf{list}}(S_k, r_k) \quad R_{k+1} = R_k \cdot \langle \mathsf{ListFiles}, r_k \rangle\end{array}}{\langle S_k, R_k \rangle \xrightarrow{\mathsf{ListFiles},\, r_k} \langle S_{k+1}, R_{k+1} \rangle}\ \textsc{Tool-Exec-List}$$

G.7 契约约束的工具选择策略

最后，我们将概率排序分布与逻辑过滤结合起来。设 $\mathsf{rank}(t \mid u_k)$ 表示给定需求表示 u_k 后，候选工具的（归一化）排序分数。

我们定义契约约束策略：

$$\pi(t \mid q, H, S_k, R_k) \triangleq \frac{\mathsf{rank}(t \mid u_k) \cdot \mathbf{1}[S_k \models P_t]}{\sum_{t' \in C_k} \mathsf{rank}(t' \mid u_k) \cdot \mathbf{1}[S_k \models P_{t'}]}.$$

随后按以下方式采样轨迹 τ：

$$p(\tau \mid q, H) = \prod_{k=0}^{n-1} \Big(p(\langle S_k, R_k \rangle) \cdot p(\texttt{<start_call_tool>} \mid q, H, S_k, R_k) \cdot \pi(t_k \mid q, H, S_k, R_k) \cdot p(r_k = \mathsf{Exec}(t_k, S_k)) \cdot p(A_{t_k}(S_k, r_k) = 1 \mid S_k, r_k) \Big), \tag{25}$$

同时满足全局约束：任何违反 P_t 或 Q_t 的情况，其概率均为零：

$$\exists k.\ \neg \mathsf{SafeStep}_k(\tau) \Rightarrow p(\tau \mid q, H) = 0.$$

这种显式分解使概率推理与逻辑契约之间的交互在形式上清晰可见且可验证。