arXiv:2601.01199v1 [cs.SE] 3 Jan 2026

溯因式氛围编程（扩展摘要）

Logan Murphy、Aren A. Babikian 和 Marsha Chechik

加拿大多伦多多伦多大学

摘要。 当人工智能模型生成软件制品（即“氛围编程”）时，人类工程师有责任对其进行确认。理想情况下，这种确认应通过创建正确性的形式化证明来完成。然而，在许多现实场景的氛围编程中，这并不可行，尤其是当人工智能生成制品的需求难以形式化时。本文扩展摘要介绍了一项正在进行的工作，旨在提取可分析的半形式化论证，以说明氛围编程产物是否充分。我们的框架不直接判定正确性，而是生成一组条件，在这些条件下，生成的代码可被视为充分。我们介绍了实现该框架的当前工作，以及预期的研究机会。

引言。 基于生成式人工智能模型（“编码代理”）的编程工具要实现可扩展性，核心障碍之一是必须信任它们生成的产物 [4]。提高信任度的一种方式是要求工具在提供代码的同时，给出解释，说明为何该代码能够充分解决给定的问题。这种解释可以通过多种方式生成；我们来看看两种极端情况。在我们所说的朴素方法中，解释以非结构化的自然语言给出。另一种是我们所说的演绎式方法，解释以正确性的形式化证明给出。

如图1所示，我们可以从两个维度比较这些方法的优缺点。第一个维度（横轴）考察解释本身如何得到验证。采用朴素方法时，负责工程师必须手动审查自然语言解释。当然，这很容易出错，尤其是在审查者有动机接受该解释的情况下——例如，拒绝该解释可能意味着他们得承担一些编程工作。另一方面，采用演绎方法生成的解释可以自动进行正确性检查。第二个维度（纵轴）反映了该方法的适用性，即该方法可应用于多少种编程场景。朴素方法具有广泛适用性（也就是说，它几乎适用于任何用例），而演绎方法则要求程序的语义和其正确性规约都能够被形式化地解释。

动机示例。 假设 Alice 正在开发一款软件工具，以支持一家金融公司的反洗钱（AML）流程，自动标记呈现某些风险因素的账户。Alice 希望一个编程智能体起草一个程序，该程序接收客户

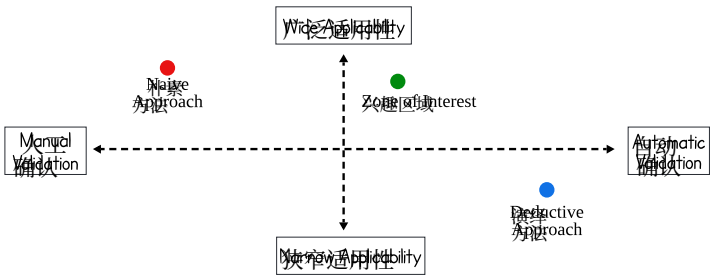

图1。 比较为氛围编程产物生成充分性解释的不同方法。我们的方案 溯因式氛围编程旨在进入“兴趣区”。

账户的一些信息（交易历史、账户年龄、是否存在先前警报等），并在适当时将该账户标记为可疑。为此，她编写了一段提示词，列出代码的以下需求等内容：

(1) 代码应返回一个元组 $\langle s,\ d,\ r\rangle$，其中 s 是一个整数评分，表示识别出的账户总体风险； $d \in \{flag,\ ok,\ review\}$ 是一个决策，用于判断该账户应被标记、忽略还是进行人工审核； r 是一个理由列表，解释账户的 s 值是如何计算得出的。(2) 计算 s和 d 所涉及的程序逻辑应符合反洗钱的一般优先事项和最佳实践。(3) r 中提供的解释应当是非推测性的、非指控性的。

我们将上述三点视为 Alice 希望生成的代码的“规范”，并强调，这一规范难以被彻底形式化，尤其是第 (2) 点和第 (3) 点。因此，如果 Alice 从大语言模型获得了一些声称符合该规范的代码，她很可能无法通过（传统的）正确性证明对其进行验证。附录A给出了一个根据 Alice 的提示词进行氛围编程生成的解决方案示例。

溯因式氛围编程。 我们提出溯因式氛围编程，作为对上述朴素方法和演绎方法的替代方案。“溯因式”这一限定词指的是通过假设生成 进行推理的过程，这一过程由 C.S.皮尔士 [7]阐述。我们的方法建立在这样一个基本假设之上：生成解释的智能体完全不可信，也就是说，它实际上并不了解任何有关现实世界、程序或工程师需求的有用事实。智能体所能做的，至多是提出假设，而这些事实可能为真，也可能为假。溯因式氛围编程鼓励智能体通过生成假设 来解释给定代码为何足以解决用户的问题；如果这些假设为真，就能说明该代码确实足以解决问题。

在我们对溯因式氛围编程的实现中，这些解释（我们称之为论证）以结构化、分层且半形式化的方式呈现

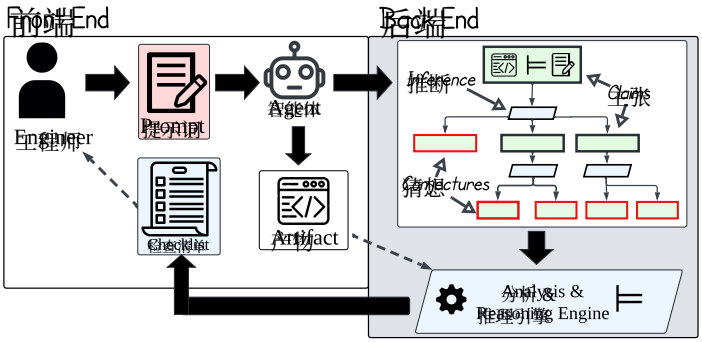

图 2. “溯因式氛围编程”工作流程 .

论证（如图 2 右侧所示）。我们的论证设计受到结构化保证案例的启发，这类案例以目标结构化表示法（GSN） [3]等语言表达。从结构上看，这类论证呈有根树的形式，其顶点主要是主张，即智能体假定为真、且有助于解释氛围编程程序充分性的命题。在我们的方案中，论证中的主张可以是形式化的（例如，一阶理论中可解释的句子），也可以是半形式化的（例如，包含一些未解释的原子命题或谓词）。树的根节点是一个主张 C，断言氛围编程产物是给定问题的充分解决方案。随后，该主张 C 被分解为一组子主张 $\{C_1, \ldots, C_n\}$，表示一个假定的推断 $(\bigwedge C_i) \implies C$。主张向子主张的这种分解会迭代进行，直到某个主张断言的内容应当由可观察的证据支持。这类主张（称为猜想）构成树的叶节点。因此，在构建论证时，智能体会迭代分解主张，生成合理的假设作为子主张，并确保由此产生的推断应当成立；这一过程持续进行，直到所有剩余的猜想在智能体看来都足够精确，可以直接验证，例如通过静态分析或求助于某个外部信息来源。

生成理由后，会从两个方面对其进行分析：**(i)** 检查每个推断 $(\bigwedge C_i) \implies C$ 是否有效（例如，使用 SMT 求解器），以及 **(ii)** 尝试验证每个可由某种特定判定过程可靠验证的猜想（例如，借助程序分析工具）。任何未能成功完成的任务，以及任何没有可用判定过程的猜想，都会以检查清单的形式返回给用户，列出需要离线调查的事实。按照构造，只要清单中的所有事项都确定成立，我们就能证明代码相对于原始规范是充分的。当然，清单中的某些事项可能带有主观成分；在这种情况下，用户需要判断自己对其真实性的信心或怀疑程度。但由于

清单中的事项直接提取自理由，因此用户可以追踪任何疑虑对论证其余部分的影响。

示例（续）。继续以 Alice 和反洗钱程序为例，假设我们要求智能体按照上述方式，为其充分性提供理由。附录 B 收录了一个具有代表性的示例，我们在此仅着重介绍其中的几个方面。首先，从断言代码充分性的根主张 C_R 开始，该理由首先提出对规范进行一种“浅层”形式化，其中需求 (1) 被形式化（因为它只考虑程序输出的结构），而需求 (2) 和 (3) 则用未解释谓词表示；随后利用这种形式化定义一个子主张 C_0，并引入推断 $C_0 \implies C_R$。由于 C_R 未经解释，这一推断会被列入检查清单，也就是说，Alice 需要确保该形式化充分体现了她对规范的意图。主张 C_0 本身被分解为三个子主张 $\{C_1, C_2, C_3\}$，分别断言三项（半）形式化需求中的一项。由于 C_0 仅仅是这三项需求的合取，因此所得推断在形式上有效，无须 Alice 审查。

主张 C_1 描述了程序输出的结构，在本例中，可以通过源代码对其进行直接的静态验证。主张 C_2关注程序逻辑与反洗钱实践的一致性，该主张本身被分解为四个子主张，每个子主张都代表程序逻辑的一个方面，而智能体 推测该方面体现了与反洗钱实践的一致性。由此分解得出的推断——Alice 需要对其进行审查——实际上断言，这些逻辑方面足以证明其与反洗钱实践的一致性。根据智能体的判断，这些子主张要么作为待验证的猜想保留（例如：“将风险因素映射到风险等级的过程受到反洗钱最佳实践和指南的指导”），要么进一步分解为更多子主张。主张 C_3断言，程序生成的文本说明是非推测性的且为非指控性的，因此被分解为两个子主张：其中一个识别程序输出中可能包含的所有字符串短语，另一个断言这些具体字符串中没有任何一个包含推测性或指控性语言。这种分解在逻辑上是有效的，并且相对于程序源代码，所提取字符串的完整性可以通过静态方式验证，因此 Alice 只需审查每个具体字符串是否恰当。

迈向实现。为了实现我们的提案，我们需要：（i）一种适合编码智能体生成的论证的格式；（ii）支持自动分析这些论证；以及（iii）促使智能体生成有用论证的有效策略。针对（i）和（ii），我们目前正在开发将论证嵌入 Lean 证明助手的支持，并以我们先前关于结构化保证案例的引导式形式化工作为基础 [10]。选择 Lean 是因为它具备丰富的元编程功能，使我们能够灵活地实现针对论证的原生分析和外部分析。

关于（iii），我们正在同时研究各种策略和技术，以促进严谨论证的创建，例如，开发可先验形式化的部分模板。

研究机会。 我们认为，作为一种通用策略，溯因式氛围编程提出了若干值得确认的有趣研究问题，并为沿不同方向进行扩展提供了机会。可以通过多种指标评估该框架，例如：不具备形式化方法知识的工程师使用该框架的易用性、该框架在提升人们对人工智能生成的工程制品的信任方面的成效、该框架对不同氛围编程场景的适用性，以及应用该框架的成本（按词元数量或人工投入计算）。我们预计，可以通过多种策略，从这些维度以及其他维度提升该框架的成效，由此带来若干研究机会。尽管本文介绍的溯因式氛围编程仅包含一轮（产物 $\rightarrow$ 理由 $\rightarrow$ 检查清单），但我们认为，将其扩展为迭代式流程会进一步提升该框架的效益。例如，在形成初步理由后，工程师可能会希望提示模型驳回某些猜想，或重新表述、完善其他猜想。或者，智能体生成的某些猜想可能会帮助工程师了解如何完善整体规格说明。如何制定有效策略，利用迭代的用户反馈来引导或微调理由的形成过程，是一个非常有趣且可能具有重要意义的研究机会。最后，尽管我们当前的提案主要关注确认人工智能生成的代码，但我们认为，这种方法可以推广至其他工程制品的确认，例如规格说明、模型、测试场景等。

相关工作。 已有多种方法提案，旨在根据非形式化规格说明验证人工智能生成的制品，从而建立对人工智能系统的信任。提出的策略包括尝试直接从生成的制品中提取解释 [8],对给定的自然语言解释进行语义分析 [6],以及将用户规格说明自动形式化，以便进行有形式依据的评估 [1, 2]。我们提出的溯因式氛围编程结合了非形式化规格说明的分析与生成、规格说明的自动形式化，以及形式推理与分析。

人们参与人工智能生成的制品的验证与确认，是一个新兴的研究领域。Shankar 等人 [9] 将人工反馈融入对部分自然语言解释的处理，强调人的偏好而非形式结构。在形式化场景中， [5]将人工反馈融入形式推理以及氛围编程工作流。我们的方案有两个显著特点：关注论证依据的逻辑结构，并支持针对待评估产物提出形式与非形式断言。

结论。在这篇扩展摘要中，我们介绍了溯因式氛围编程，将其作为一种方法，用于为人工智能生成的软件制品的适用性提供严谨且可部分分析的论证。我们的方案采用结构化论证，支持形式语义与非形式语义相结合，并鼓励编码代理生成可独立验证的假设。我们认为，溯因式氛围编程为提升人工智能生成制品的可信度提供了一条很有前景的路径，尤其适用于无法进行形式化验证的情形。此外，我们认为，该方案为进一步研究和开发带来了若干有趣的机会。

参考文献

1. Chen, B., Babikian, A.A., Feng, S., Varró, D., Mussbacher, G.：基于一致数据和检查器生成的字符串需求满足性检查。在：2025 IEEE 第 33 届国际需求工程会议（RE）论文集。第 231–243 页（2025） 2. Chen, K., Yang, Y., Chen, B., López, J.A.H., Mussbacher, G., Varró, D.：利用大语言模型自动化领域建模：一项比较研究。在：2023 ACM/IEEE 第 26 届模型驱动工程语言与系统国际会议（MODELS）论文集。第 162–172 页。IEEE（2023） 3. Kelly, T., Weaver, R.：目标结构化标注——一种安全论证标注。在：2004 年可信赖系统与网络保障案例研讨会论文集。第 6 卷。Citeseer，普林斯顿，新泽西州（2004） 4. Lyu, M.R., Ray, B., Roychoudhury, A., Tan, S.H., Thongtanunam, P.：自动编程：大语言模型及其未来。ACM软件工程与方法学汇刊 **34**(5), 1–33（2025） 5. Mitchell, J., Shaaban, Y.：立场：氛围编程需要氛围推理：通过形式化验证改进氛围编程。在：第 1 届 ACM SIGPLAN 语言模型与编程语言国际研讨会论文集。第 84–90 页。LMPL ’25，计算机协会（2025） 6. Nguyen Tung, L., Cho, S., Du, X., Neelofar, N., Terragni, V., Ruberto, S., Aleti, A.：自动生成机器学习文本分类器的可信性预言机。ACM 软件工程会议论文集 **2**(FSE), 2382–2405（2025） 7. Niiniluoto, I.：为溯因辩护。《科学哲学》**66**(S3), S436–S451（1999） 8. Ruan, H., Zhang, Y., Roychoudhury, A.：SpecRover：通过大语言模型进行代码意图提取。在：IEEE/ACM 第 47 届软件工程国际会议论文集。第 963–974 页。ICSE ’25（2025） 9. Shankar, S., Zamfirescu-Pereira, J., Hartmann, B., Parameswaran, A., Arawjo, I.：谁来验证验证者？使大语言模型输出的大语言模型辅助评估与人类偏好保持一致。在：第 37 届 ACM 用户界面软件与技术年会论文集。第 1–14 页。UIST ’24，计算机协会（2024） 10. Viger, T., Murphy, L., Di Sandro, A., Menghi, C., Shahin, R., Chechik, M.：构建有效的基于模型的安全论证的首要方法。《软件与系统建模》**22**(5), 1473–1494（2023）

示例氛围编程程序

```
def 评估_可疑_活动（账户,                                风险_因素）:
    理由 = []
    评分 = 0.0

    LOW_RISK_权重 = 1.0
    MID_RISK_权重 = 3.0
    HIGH_RISK_权重 = 6.0

    可疑 _阈值 = 8.0
    模棱两可 _阈值 = 4.0

    MAX_缓解 = 4.0

    交易 =风险_因素 . get (" 交易 ")
    账户_年龄_天数 =风险_因素 . get (" 账户_年龄_天数 ")
    客户_资料 =风险_因素 . get (" 客户_资料 ")
    之前_警报 =风险_因素。获取（“之前_警报”）

    高_风险_司法管辖区 =集合（高_风险_国家（））

    高_风险_交易_计数 = 0
    对于交易在交易:
        如果交易。获取（“国家”）在高_风险_司法管辖区:
            高_风险_交易_计数 += 1

    如果高_风险_交易_数量 > 0:
        评分 += 高 _RISK_权重
        理由。追加(“涉及交易                                较高风险
            观察到的司法辖区”)

    大型_交易_计数 = 总和(1 对于 交易 在 交易
                          如果 交易 . 获取 (" 金额 ",  0) >= 100000)

    如果 大型_交易_计数 >= 2:
        评分 += 中 _RISK_权重
        理由。 追加 (“审查期间内记录了多笔高额交易
        。”)

    如果 账户_年龄_天 < 90:
        评分 += 中 _RISK_权重
        理由。 追加(“账户相对较新,                          这可能会
        限制活动模式的历史背景。)

    if 之前_警报 > 0:
        评分 += 中 _RISK_权重
        理由。追加(“之前存在以下监控警报:
        此账户。”)
```

```
    如果 长度 ( 交易 )  > 20:
        评分  += 低     _RISK_权重
        理由。 添加(" 交易量高于
        典型基线活动。")

    if  " 现金密集型 "          在客户_档案:
        评分  += 低     _RISK_权重
        理由。追加（" 客户档案的特征是
        与中度升高的反洗钱监控相关
        敏感度 .")

    缓解_评分   =缓解_千字节（账户,                          风险_因素 )
    缓解_抵消   =最小值（缓解_评分,                          最大值_缓解 )
    评分-=  缓解_抵消

    if  缓解_偏移量  > 0:
        理由。追加(“记录在案的情境因素可能
        解释观察到的某些活动。”)

    评分  =最大值( 评分     , 0.0)

    如果评分  >=  可疑            _阈值:
        决策   =" 标记 "
    否则如果 评分  >=  模棱两可    _阈值 :
        决策   =" 审核 "
    否则:
        决策   =        " 好 "

    返回 {
        " 评分 ": 四舍五入 (评分     , 2),
        " 决策 ":          决策       ,
        "理由       " :  理由
    }
```

B 示例论证

令 A 为程序的有效输入集合。令 f 表示论证所讨论的程序。下面概述我们贯穿全文示例的示例论证的逻辑结构。此处描述的论证及其所描述的程序，均使用 OpenAI GPT 5.2 免费版，通过若干次手动迭代生成。

根本主张 C_R**：**该程序符合给定的规范。**分解** **d by**: $\{C_0\}$
注意：此分解必须由用户验证。

主张 C_0**：** $\forall x \in A,\ R_1(f(x)) \wedge R_2(x,\ f(x)) \wedge R_3(f(x))$ 其中

– $R_1(v)$ 断言 $v = \langle s,\ d,\ r \rangle$ 对于某些 $s \in \mathbb{R}$、$d \in \{flag,\ ok,\ review\}$ 和$r \in List(String)$。– $R_2(x,\ v)$ 断言，输出 v 所描述的评分和决策，反映了针对输入 x 应用时的一般反洗钱指南和最佳实践。– $R_3(f(x))$ 断言，输出中包含的任何解释都不具有指控性，也不具有推测性。形式化地，$R_3(v) \equiv \forall s \in reasons(v)\ \neg Accusatory(s) \wedge \neg Speculative(s)$，其中 $Accusatory(s)$ 和 $Speculative(s)$ 在形式上不作解释。

按 $\{C_1, C_2, C_3\}$**N 分解**
ote: This d分解 ition is l逻辑上有效。

猜想 C_1: $\forall x \in A,\ R_1(f(x))$**注意：**此猜想可通过对程序进行静态分析来验证。

声明 C_2: $\forall x \in A,\ R_2(x,\ f(x))$**按以下内容分解**: $\{C_4, C_5, C_6, C_7\}$
注意：此分解必须由用户验证。

猜想 C_4：每个风险因素都恰好映射到三个风险等级之一 $\{low, mid, high\}$

猜想 C_5：风险因素到风险层级的映射参考了反洗钱最佳实践和指南

猜想 C_6: $\forall a,\ b \in A.\ score(f(a)) \leq score(f(b)) \rightarrow decision(f(a)) \leq decision(f(b))$

注：这断言风险决策相对于评分是单调的，排序为 $ok < review < flag$。这可以通过静态分析进行验证。

主张 C_7：与风险层级和决策阈值相关的权重是适当的。

分解者： $\{C_8, C_9, C_{10}\}$**N**
ote经用户验证，直观而言，这b一推断断言， C_8、 C_9 和 C_{10} 构成了充分的需求集合，足以确保与风险层级相关的权重以及决策阈值设置恰当。

猜想 C_8：中风险层级因素的重要性是低风险层级因素的三倍，而高风险层级因素的重要性是中风险层级因素的两倍。

主张 C_9 ：风险因素权重（$low \mapsto 1.0$， $mid \mapsto$3。0，$high \mapsto 6.0$）、决策阈值（$review \mapsto 4.0$， $flag \mapsto 8.0$）和最大缓解因子（-4.0）的比例恰当，且符合最佳反洗钱实践。

主张 C_{10} ：知识库（“mitigation_kb”）为给定账户和风险因素提供适当的缓解评分。

主张 C_3： $\forall x \in A,\ R_3(f(x))$
等价地： $\forall x \in A,\ \forall s \in reasons(f(x))\ \neg Accusatory(s) \wedge \neg Speculative(s)$**分解依据**：
$\{C_{11}, C_{12}\}$
注意：此分解在逻辑上有效。

猜想 C_{11}： $\forall x \in A,\ \forall s,\ s \in reasons(f(x)) \implies s \in \{s_1, \ldots, s_7\}$，其中$s_1$ = “观察到涉及较高风险司法管辖区的交易”s_2 = “审查期内记录了多笔大额交易” ...s_7 = “记录在案的情境因素或可解释观察到的部分活动”

注意：此猜想可以通过静态验证。

猜想 C_{12}： $\forall s \in \{s_1, ..., s_7\},\ \neg Accusatory(s) \wedge \neg Speculative(s)$