arXiv:2603.13443v1 [cs.SE] 13 Mar 2026

NormCode Canvas：通过基于案例的推理实现 LLM智能体工作流的可持续发展

Xin Guan[1[0000−0003−1670−4937]]、Yunshan Li[2,1],和 Ze Wang[3,1]

[1] Psylens AI，中国[2] 深圳大学，中国[3] 伦敦大学学院，英国
garguan2001@outlook.com

摘要。 我们介绍了 **NormCode Canvas**（v1.1.3），这是一个已部署的系统，在多步 LLM 工作流中以两个级别实现基于案例的推理。其基础是 NormCode [8], ，一种半形式化的规划语言；其经编译器验证的作用域规则确保每个执行检查点都是真正自包含的案例，从而消除了标准编排框架中使检索不可靠且故障难以定位的隐式共享状态。**1级** 将每个检查点视为一个具体案例（挂起的运行时）；Fork 实施检索并复用，值覆盖则实施修订，并自动传播过时边界。**2级** 将每个编译后的计划视为一个抽象案例；编译流水线本身就是一个 NormCode 计划，从而支持递归案例学习。由此产生三个结构性特征：**(C1)**直接检查点检查；**(C2)** 通过编译器生成的叙述进行执行前审查；**(C3)** 作用域限定的选择性重新执行。四个已部署计划构成了结构化证据：PPT生成通过商业 API 生成演示文稿，每张幻灯片耗时∼40秒；代码助手执行多步软件工程任务，最多涵盖十个推理周期；NC 编译将自然语言规范转换为可执行的 NormCode 计划；Canvas 助手在连接外部 AI 代码编辑器后，可自动执行计划调试。这些计划共同构成一个自我维持的生态系统，其中计划相互生成、调试和完善，从而在系统规模上实现累积式基于案例的学习。

关键词： 基于案例的推理 · 大语言模型工作流 · AI 工作流开发 · 工作流管理 · 智能体 · LLM 推理 · 智能体领域专用语言

1 引言

多步 LLM 工作流——其中每一步推理都会为下一步提供输入——已成为在实际应用中部署语言模型的标准架构。LangChain [7],LangGraph [13], AutoGen [23],和 PromptFlow [15] 等框架让构建此类工作流变得轻而易举。然而，要对其进行调试、恢复和系统性改进仍然很困难：状态通过对话历史、可变对象和特定于框架的控制逻辑向后续步骤传递，

因此，第 k 步的中间检查点可能会隐式依赖检查点本身未记录的先前上下文。故障恢复变得脆弱，重放难以令人信服，而调试则需要重建先前执行过程中的大部分内容。

工具可以增加持久化和追踪功能，但根本局限依然存在：规划、其中间语义以及有效继续执行的条件，分散在代码和提示词中，而未以一种统一且可解释的形式表达。

缺少的是一种工作流语言，借助它可以明确表示、检查、验证、执行和修订多步推理，同时兼具可供人类阅读、可由机器执行和结构上可强制执行的特性。

NormCode [8]是一种为此而设计的半形式化规划语言。其作用域规则保证每个执行检查点都是一个**自包含案例**，不存在隐式的上下文依赖（第 3 节）。这一保证带来三项实用能力：可直接检查任意流程索引处确切的作用域内输入（C1）；可在执行前对编译后的计划进行结构审查（C2）；以及可进行作用域限定的选择性重新执行（C3）。**NormCode Canvas (v1.1.3)** 是实现 C1–C3 的已部署系统，并呈现一种两级基于案例的推理架构：1级管理具体案例（各检查点处挂起的运行时）；2级管理抽象案例（编码可复用工作流模式的可执行计划）。我们不主张其在最终任务质量上普遍优越；我们的主张针对的是多步大语言模型工作流的可检查性、可修订性和有界重新执行能力。

与既有工作的关系。既有工作 [8]是一份语言规范：它定义了 NormCode 的语法、类型系统和作用域规则。既有工作回答了*NormCode* 是什么。本文则以四个已部署的工作流为基础，涵盖从 PPT 生成和代码辅助到自托管编译与交互式画布控制，回答*NormCode* 在实践中能实现什么。

贡献。(1) 提出对大语言模型工作流开发的两级基于案例的推理阐释（第 4–5 节）：将 1级案例视为挂起的运行时，将 2级案例视为可执行计划，并将编译流水线本身视为一个 2级案例。(2) 提出一种满足六项属性的作用域工作流语言（第 3 节）：可解释、可强制执行、可组合、可局部寻址、可泛化且可移植。(3) 部署一个实现三项结构性能力的系统（第 6–7 节）：直接检查检查点（C1）、执行前审查（C2）和选择性重新执行（C3）。(4) 提供来自四个生产工作流的结构化证据（第 7 节），其中包括一个由计划相互生成、调试和完善的生态系统。

2 背景

2.1 基于案例的推理与面向过程的基于案例推理

基于案例的推理 [1,12] 通过检索并改编过往案例来解决新问题。这个四阶段循环——**检索**、**重用**、**修订**、**保留** ——取

决于案例完整性：每个案例都必须是对过往解决方案状态的自包含记录，不依赖隐含上下文。

面向过程的基于案例推理（POCBR） [5,16]将这一理念扩展到**以流程为案例**：案例记录工作流轨迹，包括中间状态和数据流。多步 LLM 工作流是面向过程的基于案例推理的天然应用领域；其挑战在于确保其中间状态适合作为独立、完整的案例。

2.2 案例完整性与结构隔离

大多数大语言模型编排框架默认不满足案例完整性：步骤 k 的检查点可能包含从步骤 1、……、$k-1$累积的对话历史或提示上下文，使其成为共享状态的快照，而非独立、完整的案例。由此会产生三种后果：**检索不可靠**——在新上下文中从此类检查点继续执行会导致行为错误；**故障无法定位**——步骤 k 发生故障的原因可能隐藏在任何累积的隐式状态中；**案例不可复现**——加载同一检查点的两次执行可能产生不同的输出。

这些失败可通过**结构隔离**消除：如果每个步骤只能访问其自身作用域块中明确声明的数据，编译器在形式化时确定每个步骤的作用域，而编排器仅根据每个步骤声明的引用构造其输入，那么这种工作流语言就具备结构隔离。在这一保证下，步骤 k 存储的输出恰好包含下游步骤所需的内容；失败可以定位到出错步骤声明的输入；复用检查点也能重现原有的数据依赖行为。
NormCode [8] 正是围绕这一促成条件而设计的。

3 NormCode 作为一种工作流语言

NormCode 是一种用于构建推理计划，而非仅仅描述推理计划的表述语言。本节论证 NormCode 满足人工智能编排所需的实用工作流语言的六项属性。形式语言规范见 [8]。激活之前，NormCode 计划有三种形式：由人工编写的半形式化源文件（.ncds）、经过严格形式化的编译形式（.ncd），以及为执行前审查而生成的纯文本叙述（.ncn）。

3.1 实用工作流语言的六大特性

(P1) **可解释**：.ncds 格式读起来就像结构化的自然语言；形式化会生成 .ncn，这是一份用于执行前审查（C2）的纯文本叙述；概念名称是自然语言标识符，因此可以用任何语言编写计划。(P2) **可强制执行**：编译器会在首次运行前确定每一步的作用域；.ncn 是对实际形式化计划的确定性呈现，而非释义。(P3) **可组合**：三个符号和一条作用域规则即可表达嵌套循环、条件语句、并行推导和多智能体协作，适用于

任何规模的计划。(P4) **可局部寻址**：每一步都有一个点分隔流程索引；每一步都是自包含的，因此无需重建前序步骤，就能检查、修订或复用任意检查点。(P5) **可泛化**：在已部署的示例中，相同的三个基本元素支持多种工作流类别，包括 PPT 生成、代码辅助、自托管编译和交互式画布控制，且无需特定领域的语法。(P6) **可移植**：NormCode 计划是纯文本 .ncds，附带 concept_repo.json 和 inference_repo.json——可进行版本控制、共享，并可由任何兼容的编排器执行。

3.2 语法与作用域规则

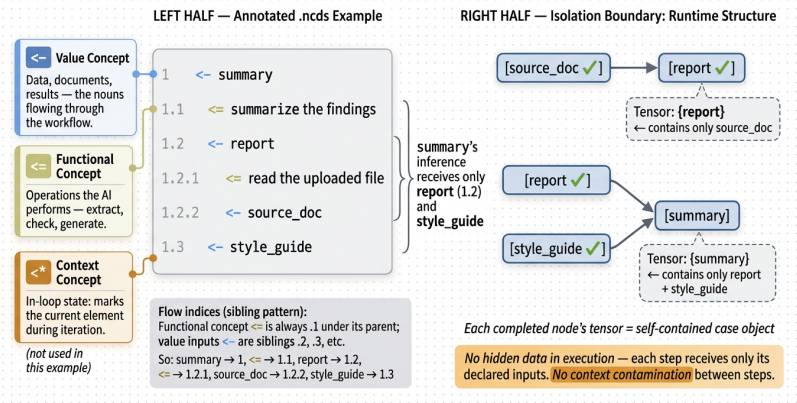

图1. NormCode 语法与作用域规则。左：带注释的 .ncds；右：运行时的隔离边界。每个节点的张量都是独立自包含的（C1）。

三个标记构成完整语法（图1； [8] 中的形式化规范）：<- 值， <= 函数， <* 上下文（循环当前元素）。这种语言是**半形式化的**：概念名称和描述可以是自由形式的自然语言字符串（例如 {document summary} 或 {分析}），而三个标记和缩进则受到严格约束，因此计划既便于领域专家阅读，也可由编排器编译，并且**完全语言无关**（第 7 节）。每个推理的输入，恰好是其直接外层代码块中声明的概念——不存在隐式共享。编译器在形式化阶段确定作用域；编排器在运行时强制执行。计划按从上到下的顺序读取（先读根概念），并由内向外执行（先解析最深层的依赖项）。流程索引（run_id、flow_index）为每个检查点提供稳定地址。**语义推理**会调用大语言模型（具有概率性且消耗词元）；**句法推理**则执行确定性数据操作（路由、分组、计时、循环），且不产生大语言模型调用成本——在生产计划中占节点总数的 60–70%。句法检查点完全可复现；语义检查点则具有有界随机性。概念数据以**引用**（命名轴张量）的形式存储；大型输入则以感知符号的形式传递——它们是紧凑的指针，仅在大语言模型调用时才转换为内容，从而避免上下文累积。详情

面板公开展示每个已完成步骤的精确有界输入，只需恒定的检查工作量即可直接查看（故障定位，C1）。

4 两级基于案例的推理：1级——具体案例管理

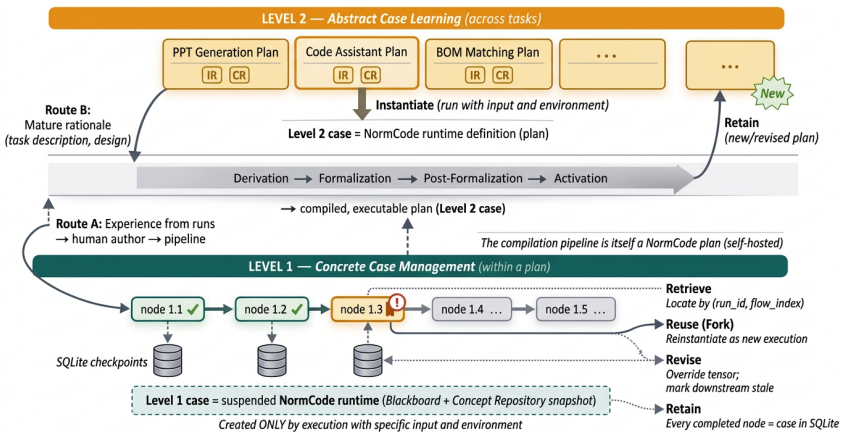

图2. 两级基于案例推理架构。2级（顶部）：抽象案例（计划）；1级（底部）：具体案例（挂起的运行时）。路径 A 将经验提炼为计划；路径 B 将计划实例化，以生成 1级案例。

4.1 1级案例定义与案例库

图 2 展示了两级架构。1级案例是一个**挂起的 NormCode 运行时**——在特定流程索引处捕获并持久化的完整执行环境。形式化地说，1级案例 $c = (r,\ f,\ \mathcal{B}_f,\ \mathcal{C}_f,\ \mathcal{E}?\ f)$，其中$r$ 是运行标识符；f 是最后一个已完成节点的流程索引；$\mathcal{B}_f$ 是**Blackboard 快照**：f 完成时的完整节点状态映射；$\mathcal{C}_f$ 是**概念存储库快照**：截至 f完成的所有节点的全部张量；$\mathcal{E}?\ f$ 是可选的环境快照，当 $\mathcal{C}_f$ 包含感知符号时存在。由于作用域规则阻止任何节点访问其声明区块之外的数据，$\mathcal{C}_f$ 是自包含的——不存在隐藏的上下文依赖。案例库 $\mathcal{CB}$ 是所有运行中的全部 1级案例的集合：

$$\mathcal{CB} = \{(r, f, \mathcal{B}_f, \mathcal{C}_f, \mathcal{E}_f^?) \mid r \in \text{Runs},\ f \in \text{FlowIndices}(r)\} \tag{1}$$

$\mathcal{CB}$ 会持久化存储在每个项目独立的 SQLite 数据库中；检查点面板是其可视化界面。

检索。检索会从 $\mathcal{CB}$ 中选取一个过往案例，作为新执行的起点。触发条件有两种：（1）一次运行在流程索引 f*处失败；（2）新任务与某次过往执行相似。通过张量检查器检查案例时，可在表格、列表或 JSON 视图中，打开任意流程索引处的 $\mathcal{C}_f$。由于 $\mathcal{C}_f$ 恰好包含

每个已完成步骤的有界输入，因此检查直接且耗时恒定——故障定位（C1）。

复用 — Fork。 Fork 操作实现复用：(1) 创建新的运行标识符r'；(2) 编排器从检索到的案例中加载（$\mathcal{B}_f$，$\mathcal{C}_f$）；(3) Blackboard 将流程索引 $\leq f$ 处的所有节点标记为已完成；(4) 从第一个待处理节点开始执行。不会重新执行上游节点。

修订 — 值覆盖 + 选择性重新运行。(1) 操作员检索流程索引 f处的一个案例；(2) 通过张量检查器的编辑模式覆盖 $\mathcal{C}_f$ 中的一个张量；(3) 编排器将所有下游节点标记为过期；(4) 恢复运行时，仅重新执行过期节点；(5) 修订后的运行会在每个已完成节点生成新案例。由于作用域规则保证，只有 f 下游的节点才可能依赖 f处的张量，因此过期节点集合可由数据依赖图精确确定——即选择性重新执行（C3）。无需手动指定。

保留。运行期间完成的每个节点都会自动向 $\mathcal{CB}$ 写入一个新案例。案例库会持续累积所有运行的 Blackboard 和 Concept Repository 快照。

5 两级基于案例的推理：2级——抽象案例学习

一级案例推理（第4节）在一个规划内部进行。二级案例推理则跨越多个计划进行：它将计划本身视为案例——即抽象工作流模式，这些模式为一类问题编码了可复用的解决方案结构。

5.1 什么是 Level 2 案例

Level 2 案例是一种**NormCode 运行时定义**：一种可执行计划，用于指定某类任务的推理结构。计划可以由人工编写、通过编译流水线（第5.3节）生成，或在人机协作的迭代过程中开发——其定义性特征在于，最终结果是一个完全可执行的 .ncds + 存储库制品，而非其生成途径。

定义。一个*Level 2* 案例 P 是一个三元组：

$$P = (\mathcal{IR},\ \mathcal{CR},\ \mathcal{V}) \tag{2}$$

其中，$\mathcal{IR}$ 是**推理库**：包含执行配置（范式分配、主体能力、感知规范）的完整推理图；$\mathcal{CR}$ 是**概念库**：包含概念定义、张量轴、形状和基底引用（指向输入资源的感知符号）；$\mathcal{V}$ 是**供给集**：不变的垂直资源——提示模板、脚本、范式文件和配置参数——在编译时绑定，并在此计划的所有运行中作为基底输入引用。$\mathcal{V}$ 正是抽象案例区别于纯推理图之处：它编码了操作如何执行，而不仅仅是操作的结构。

5.2 Level 2 基于案例的推理循环

检索：从业者阅读 .ncn 叙述，并判断现有计划（或其部分内容）是否适用于新的任务类别。**复用**：使用新的输入值实例化检索到的计划并运行；所有四个可投入生产的计划在每次运行中都以这种方式复用，各自产生一组新的 Level 1 案例。**修订**：当计划不适用于新任务时，设计者修改 .ncds 源文件并重新编译；通过重新审阅 .ncn，确认修订后的抽象案例无误，然后才生成任何具体案例。**保留**：修订后的计划或新编写的计划会加入案例库，案例库也随着使用不断扩充。

5.3 NormCode 生产的编译流水线

编译流水线（推导 → 形式化 → 后形式化 →激活）是一条标准化的提炼路径，用于从经验和意图中生成二级案例：推导 从自然语言中提取概念结构；形式化 分配流程索引并确定执行类型；后形式化 分配执行资源；激活 将抽象案例序列化为编排器就绪的 JSON。

NormCode 作为一种工作流语言所带来的一个显著结果是，**编译流水线本身就是一个 NormCode 计划**。该流水线是一个 2级案例（NC Compilations 计划），但它并不是一个通用编译器。它所生成的内容取决于任务的结构以及输入规范的质量。对于具有清晰顺序结构的计划，该流水线能够可靠地生成可执行计划。对于包含更丰富领域逻辑、更复杂工具交互或更开放式控制流的计划，手动编写或人工辅助迭代仍是切实可行的路径。NC Compilations .ncds 与其他生产计划一同展示于图 5 中。

这种递归结构有三个结果：(1)**验证**——流水线前后一致，能够编译自身；(2) **通过修订改进**——改进编译流水线，就是在元级修订抽象案例；(3) **递归保留**—— 每次改进都会丰富案例库。自动编译的范围和局限性将在第 8 节中讨论。

6 NormCode Canvas：已部署的系统

NormCode Canvas (v1.1.3) 是让同一环境能够支持两个层级的基于案例的推理的已部署系统（图 3）。工作流语言本身在结构上具备以下三项实用特性，Can- vas 则将这些特性付诸实现：(1) **(C1) 直接检查检查点**。 在任何断点处，步骤 k 的确切作用域限定输入都可直接检查，而非从日志中重建——因为作用域规则已在运行时强制限定这些输入。(2) **(C2) 执行前结构审查**。 在任何大语言模型调用之前，都可以查看 .ncn 叙述；领域专家可以在计算开始前验证并否决规划。与基于人类反馈的强化学习 [20] 或宪法式人工智能 [4], 不同，这种干预作用于规划结构层面。(3) **(C3) 作用域限定的选择性重新执行**。 修订案例

在流程索引 f 处会触发重新执行，但仅重新执行传递依赖的下游节点；所有上游结果均从缓存中复用。

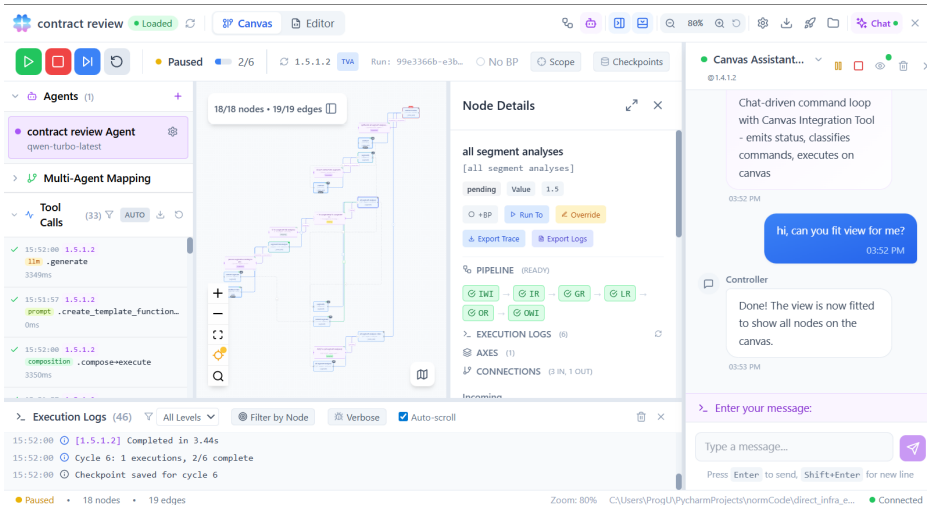

图 3. NormCode Canvas (v1.1.3)。图画布（中央）呈现完整的推理图；详情面板（右侧）用于检查任意节点的有界输入，从而实现故障定位（C1）。

6.1 二级界面：抽象案例管理

Canvas 中的 2级案例是一个**NormCode 项目**：一个包含 .ncds 源文件、已编译的 concept_repo.json 和 inference_repo.json、智能体配置及执行设置的目录。项目面板是打开、创建和切换项目的入口。首次运行前，可对 .ncn 叙述进行执行前审查（C2）：领域专家阅读以纯文本呈现的内容，核实推理结构是否符合意图，并在任何 LLM调用发生前批准或拒绝。Canvas 还内置了一个**Canvas 助手** NormCode 计划，可在应用内运行，将自然语言消息分派给 Canvas 集成功能（me.hands、me.mind、me.vision）；其逐次迭代检查点本身就是一级案例。运行 NC Compilations 计划时，每个编译阶段都是一个一级节点，因此可通过一级案例推理对编译流水线进行调试。

6.2 一级界面：具体案例管理

1级案例存储在**每个项目各自的 SQLite 数据库**中。每个已完成的节点都会自动写入检查点（run_id、flow_index、Blackboard 快照、概念存储库快照）——保留完全自动化。图

画布会在执行前呈现完整的推理图；每个节点都是潜在的检查点，并由其流程索引标识。**检索**：任何节点都可指定为断点；执行会暂停，详情面板会直接显示该节点完整的N维参考——实现故障定位（C1）。**修订**：操作员可在任意断点或已完成的案例处点击覆盖，编辑张量并恢复执行；只有下游过时节点会重新执行——实现选择性重新执行（C3）。**复用**：检查点面板可浏览活动项目中的所有运行；Fork 会将（$\mathcal{B}_f$，$\mathcal{C}_f$）载入新的运行 r'，将上游节点标记为已完成，并重新执行下游节点——支持故障恢复、A/B测试和迭代优化。

6.3 跟踪与调试界面

Canvas 提供三种互为补充的跟踪视图（图 4）。**以智能体为中心的跟踪** 记录每次工具调用及其结果——包括感知、提示组装、组合和大语言模型调用——并附有时间戳、流程索引和持续时间；展开条目即可查看完整的提示输入和模型的结构化输出（图 4a）。**以图为中心的数据跟踪** 按流程索引选取一段节点范围，并显示在这些节点之间流转的限定范围的参考数据（图 4b）。由于作用域规则保证输入自包含，此视图直接实现了检查点检查（C1）：操作员无需从日志中重建数据，即可查看确切的张量内容。**以编排为中心的跟踪** 记录每项调度决策、就绪检查、错误链和周期级细分（图 4c）——当语法或结构错误源自编排器内部而非模型输出时，操作员会查看此视图。实践中，首先检查智能体跟踪和数据跟踪，因为它们能解决最常见的问题；需要更细粒度的证据时，才会查阅编排跟踪。

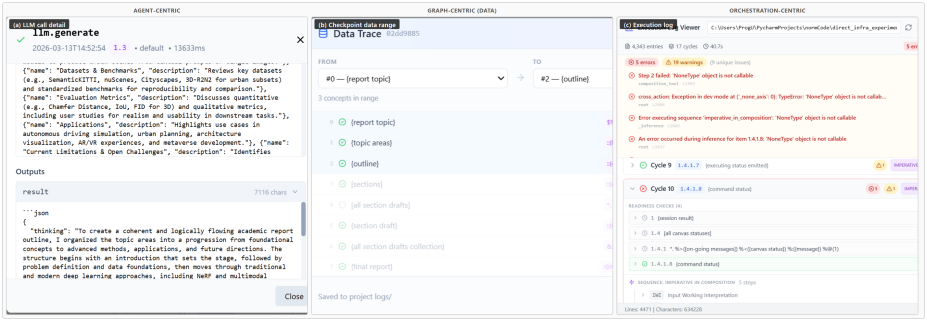

图 4. NormCode Canvas 中的三种跟踪视图。（a）以智能体为中心：包含提示输入和结构化输出的 LLM调用详细信息。（b）以图为中心：所选节点索引（C1）之间的限定范围的参考数据。（c）以编排为中心：包含错误链和周期级细分的执行日志。

6.4 架构

前端： React 18 搭配 TypeScript；使用 React Flow 实现图形可视化；Tail wind CSS。**后端：** FastAPI 搭配 Python 3.11；每个项目使用 SQLite 持久化 1级案例；使用 Python WebSockets 实时传输执行事件。**编排器：**通过 Waitlist 和 Blackboard 实现依赖驱动的调度；作用域规则通过构造来强制执行——每次推理只传递已声明的张量引用，绝不传递整个概念库。可注册使用不同大语言模型（qwen-plus、gpt-4o、claude-3-opus）的多个智能体；根据流程索引上的模式规则，将推理映射到相应智能体。

7 案例研究

我们使用四个已部署的 NormCode 计划作为支撑论文论点的**结构化证据**，而非将其作为衡量普遍输出质量的基准。它们共同涵盖了不同的工作流类别，并表明 C1–C3 在不同的控制流模式、人工审查节点和修订边界下均成立。在这些计划中，句法节点占推理总数的 60–70%。NC Compilations 已使用 qwen-plus 端到端执行，并为报告撰写、合同审查等计划生成了正确的存储库。随后，我们将在下文更详细地追踪其中两个计划。

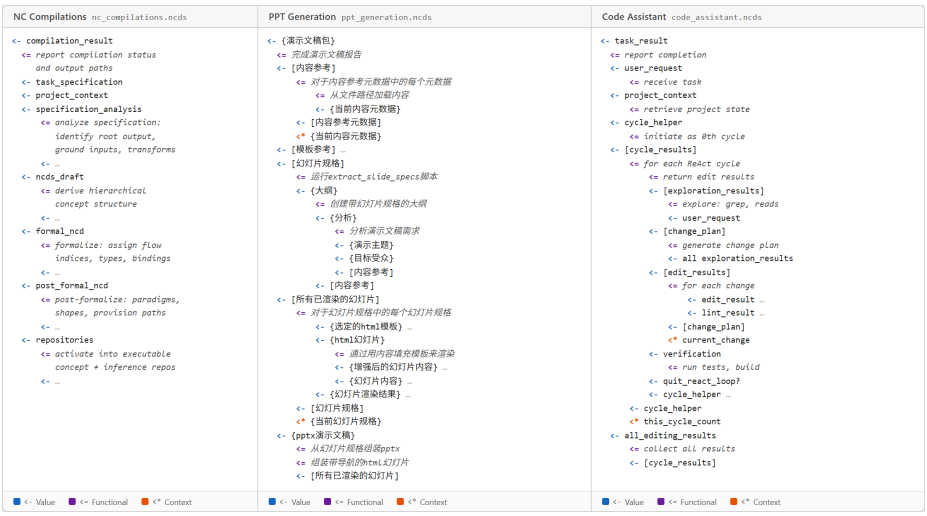

图 5. 三个生产级 NormCode 计划作为二级案例的核心摘录（`.ncds`）；实际生产计划规模更大、复杂度更高。左：NC Compilations——顺序编译流水线（第 5.3 节）。中：PPT生成——嵌套双输出循环（HTML + 每页 PPTX）。右：代码助手——ReAct 外循环 [25]，包含内部的探索、规划、编辑和验证阶段。 <- 值； <= 函数； <* 上下文。所有三个计划在开始运行之前，均可作为明确的计划供审阅（C2）。

7.1 案例研究1：PPTAgent——2级复用与1级积累

ppt_generation.ncds（图 5，中间）是一个 2级 案例，编码了一个五阶段流水线：(0) 加载参考文件，(1) 分析需求并生成幻灯片大纲，(2) 通过脚本提取逐页规格，(3) 逐页循环处理，根据同一份组件化内容 JSON 同时生成 HTML幻灯片和 PPTX规格，以及 (4) 组装最终演示文稿。该计划采用**基于组件的设计**，包含 6 种布局和 14 种组件类型，每页组合使用，因此一份结构化内容 JSON 即可驱动两个渲染器。

整个计划均以中文编写——每个概念名称（分析、大纲、幻灯片内容等）都是中文词元——展现了完全的语言无关性（图6）。该计划**部署为服务器端工作流**：客户端（网页向导、命令行界面或桌面图形界面）提交主题和参考文件，并实时查看每个步骤的完成情况；完成后，客户端会收到可下载的 .pptx 和 .html 输出。逐页计时显示，内容生成耗时 ∼4 秒，HTML渲染耗时 ∼12–17 秒，而加载、保存和收集节点的运行时间均不到 0.2 秒——这证实了在所有计划中，60–70% 的节点都是语法性的。



图6。PPT生成计划输出。(a) 一次中文运行生成的六张幻灯片；每张幻灯片都是一个1级检查点。(b) 客户端界面展示逐步进度以及可下载的 .pptx/.html 输出。

7.2 案例研究 2：CodeAssistant——作为 2级案例的 ReAct 规划

代码_assistant.ncds（图 5，右侧）是一个 2级案例，用于编码 ReAct循环软件工程工作流 [25]。每个周期中，探索 观察 （并行读取文件，零 LLM调用成本），规划 推理 （基于限定范围的探索结果进行一次 LLM调用），实施 执行 （进行精准编辑，并包含一个内部代码检查修复循环），验证则检查结果。这个案例最清楚地展示了作用域规则：实施接收的是变更计划，而不是来自探索的原始文件内容。这种分离让 C1 的证据尤为具体，因为操作员可以检查传递给每个阶段的确切输入，而不是合并后的对话历史。它也支持 C3：修订规划或实施检查点

只会重新运行依赖于该检查点的下游阶段。每次运行都会生成一个一级案例库，因此 ReAct循环的每次迭代都能在 Canvas 中检查和修订（第 6 节）。

表1总结了所有四个已部署计划及其撰写时的运行状态。

表1。四个已部署的 NormCode 计划：工作流类别、规模和运行状态。

Plan	工作流类别和规模	控制流模式	运行状态
PPT 生成 tion	幻灯片 生成 成； ∼30 推 论；6 种布局，14 组件类型	嵌套 双输出 loop （HTML + 每张幻灯片一个 PPTX）	运行于 qwen-turbo/qwen-plus com- 商业 API，成本几乎可以忽略； 可在 ∼40 秒内生成完整的演示文稿 （每张幻灯片）。
Code 助手 tant	软件 工程 工 作流； ∼40推理	ReAct 外层 loop 并行探索	最多执行 10 步的推理循环 用于执行复杂代码编辑任务的循环 ，包含范围明确的探索、规划、 实施和验证阶段。
NC 编译 版本	自托管 编译 ； ∼25 推断 结果	顺序 流水线 （推导 → 激活- 化）	With qwen-plus/GLM-4.7，生成 在 ∼15 步内生成可执行计划 用于报告撰写、论文起草， 以及可直接作为新 NormCode 计划运行的合同审查 直接作为新的 NormCode 计划。
Canvas 助手 tant	内置聊天控制器 ； ∼20 推理	交互式 命令分派 loop	可处理多种 Canvas 任务；连接到 AI 代码 编辑器（Cursor， Claude Code），启 用自动化计划调试， 代表用户接管画布界面。 代表用户。

7.3 生态系统现状

在基于案例的推理中，随着案例库不断扩充，单个案例的质量也会提升。NormCode 的两个层级都遵循同样的规律：每次运行都会新增一级案例，而每个 2级计划则发挥互补作用，彼此强化。NC Compilations 充当生态系统的计划生成器，可在 ∼15 步内将自然语言任务描述转换为可执行计划。代码助手则作为智能体探索器，通过范围明确的探索、规划、实施和验证阶段，执行多步骤软件工程任务；每个循环都会用可复用的中间状态丰富案例库。Canvas 助手则充当计划调试器：连接 Cursor[3] 或 Claude Code [2], 等外部 AI 代码编辑器后，它会接管画布，检查检查点、诊断故障并应用修复，从而为任何计划自动执行 Level 1 基于案例的推理循环。

这些角色构成一个闭环：NC Compilations 生成计划；这些计划在运行过程中积累一级案例；Canvas 助手调试失败的运行；代码助手扩展平台。每个计划的质量都随着生态系统积累的经验而提升，从而实现系统规模的累积学习。

8 讨论

语言是实现可持续改进的基底。C1、C2 和 C3 是工作流语言的结构性结果，而非设计选择

关于 Canvas 应用——并且每一项都直接促成了累积改进的一个维度。C1（故障定位）源于局部可寻址性：每个步骤都有精确的流程索引和明确定义的输入，因此每次运行都会产生具体的 1级案例，这些案例无需操作员干预即可立即用于检索、复用和修订。C2（执行前审查）源于可解释性 + 可执行性：.ncn 文件可供人类阅读，并且确切地 规定了实际执行的内容（图 7），从而能够在生成任何新的 1级案例之前，对 2级案例的修订进行审查。C3（选择性重新执行）源于局部可寻址性 + 可执行性：作用域规则决定哪些下游节点依赖于任何经过修订的步骤，而编排器会强制执行这一边界——因此，2级案例的修订能够精确传播，而 1级分支只会重新执行必要的部分。总体而言，C1–C3 在两个层级上实现了基于案例的推理循环：每次运行都会积累经验（1级），每次领域修订都会扩展抽象案例库（2级），而编译流水线——其本身就是一个 2级案例——也会通过同一循环递归改进 [5,17]。现有的大语言模型编排工具可能提供类似的检查点或图编写功能，但据我们所知，如果没有满足第 3 节所述属性的等效作用域中间表示，它们并不能通过构造保证 C1–C3。

因此，我们的主张涉及多步大语言模型工作流的可检查性、可修订性和有界重执行，而非最终任务质量的普遍优越性。

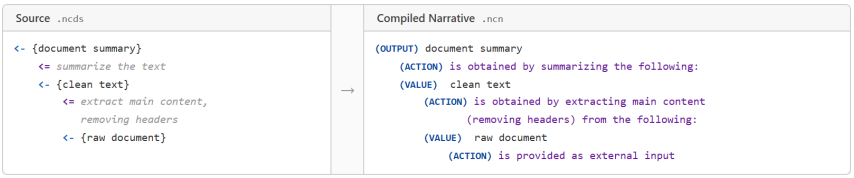

图 7. 由 .ncds 源文件编译而成的 .ncn 叙述。左：包含三个标记的源文件。右：用于执行前审查（C2）的确定性纯文本呈现。领域专家阅读该叙述，以验证规划逻辑是否符合意图，然后才进行任何大语言模型调用。

8.1 局限性

操作层面的局限性。 目前，案例检索需要操作员判断并选择合适的分支目标；学习得到的相似度度量可以降低这一门槛。案例库会无限增长——基于能力的剪枝是顺理成章的下一步。语义检查点（大语言模型输出）具有有界随机性：相同输入会产生相同的输出分布，而非固定值。包含感知符号的检查点张量在结构上是自包含的，但要复用这些检查点，必须能够访问其引用的外部资源；案例编码的是引用了什么，而非其内容。

编译适用范围。NC Compilations 能够可靠地为具有明确顺序结构的任务生成可执行计划（文档分析、报告生成、

合同审查）。对于领域逻辑复杂或与环境交互丰富的任务（例如 PPT 生成的双格式输出循环），则需要人工或代码助手编写；其界限在于任务复杂度，而非语言限制。

评估范围。全面的质量评估需要一个组合测试矩阵（$|L_2| \times |L_1|$ 个单元），其中结果质量还取决于大语言模型、输入领域和主观判断。案例研究（第7节）主张的是2级泛化能力——即结构属性在四个生产计划中均成立，且C1–C3（故障定位、执行前审查、选择性重新执行）均得以实现——而非绝对输出质量；后者会因领域、模型选择和主观标准而异。

9 相关工作

基于案例的推理与大语言模型。Aamodt和Plaza [1] 给出了经典的基于案例的推理循环；Bergmann[5] 和Minor等人 [16] 确立了面向过程的基于案例推理这一领域，其中案例用于捕获工作流知识。在面向过程的基于案例推理领域，Bergmann和Gil [6] 探讨了基于结构相似度的检索；Muller和Bergmann [18] 提出了部分工作流查询语言。Muller和Bergmann [17] 将抽象工作流模式视为案例对象——这是最接近NormCode 2级的先行工作。Rubin等人 [21] 检索少样本提示；Wilkerson和Leake [22]使用大语言模型实施基于案例的推理组件；Lenz等人 [14] 评估大语言模型作为相似度评估器。这些工作利用基于案例的推理改进单次大语言模型调用。Guo等人 [9]（DS-Agent）与本文最为直接相关：它通过基于案例的推理迭代修订多步骤数据科学计划。DS-Agent的案例是叙述性计划，不具备强制执行的保证；NormCode的检查点则由编译器和运行时在结构上隔离——这是实现可靠检索和修订的前提（第2节）。

大语言模型工作流工具。LangGraph、PromptFlow 和 LangFlow 提供检查点功能和基于图的创作；AutoGen [23]支持多智能体编排。它们都通过累积的对话历史或提示模板传递状态，无法保证结构隔离。因此，检查点包含隐式共享状态，故障无法定位，而且复用案例可能无法复现原始的数据依赖行为——这正是第 2 节所述的三个条件。

大语言模型智能体与 ReAct。 Yao等人 [25] 提出了 ReAct（观察–推理–行动）；MetaGPT [10] 将软件工程标准操作流程编码为提示序列；SWE-agent [24] 引入了智能体-计算机接口。已部署的智能体编码工具——Cursor [3], 、Claude Code [2], 和 OpenClaw [19] ——在生产规模上实施持久的观察–推理–行动循环。NormCode 将这些模式表达为编译后的计划：每次循环迭代都是一个具有流程索引和自动检查点的限定节点。在标准 ReAct 中，推理-行动链是隐式延续的，缺乏结构透明性；AgentStepper [11] 正是因此为其补充了交互式调试功能。NormCode 旨在从一开始就将这些属性结构化：每个智能体步骤都是一个 1级案例，可在故障定位（C1）下进行检查，并通过选择性重新执行（C3）进行修订。

致谢。 本工作在 Psylens AI（广州 Psylens 科技工作室）开展，其使命是构建可审计的 AI，以实现透明且可控的开发——让人类价值观能够被清晰表达、记录，并作为可持续的技术资产加以执行。更多信息请访问 https://psylensai.com。我们感谢 NormCode Canvas 用户社区提供的工作流反馈，这些反馈为案例研究提供了参考。

关于生成式人工智能的声明 在准备本作品期间，作者使用了人工智能辅助写作工具进行起草和编辑。使用这些工具后，作者根据需要审阅并编辑了内容，并对出版物的内容承担全部责任。

引用

1. Aamodt, A., Plaza, E.：基于案例的推理：基础问题、方法变体与系统方法。AI Communications**7**(1), 39–59 (1994). https: //doi.org/10 7104 2 3233 1994 . /AIC- - . Anthropic：Claude Code：终端中的智能体编程。软件（2025），https: // docs.anthropic.com/en/docs/claude-code 3. Anysphere：Cursor：AI 代码编辑器。软件（2024），https://www.cursor.com 4. Bai, Y., Jones, A., Ndousse, K., Askell, A., Chen, A., DasSarma, N., Drain, D., Fort, S., Ganguli, D., Henighan, T., Joseph, N., Kadavath, S., Kernion, J., Conerly, T., El-Showk, S., Elhage, N., Hatfield-Dodds, Z., Hernandez, D., Hume, T., Johnston, S., Kravec, S., Lovitt, L., Nanda, N., Olsson, C., Amodei, D., Brown, T., Clark, J., McCandlish, S., Olah, C., Mann, B., Kaplan, J.：宪法式人工智能：来自人工智能反馈的无害性。arXiv预印本 arXiv:2212.08073 (2022), https://arxiv.org/ abs/2212.08073 5. Bergmann, R.：《经验管理：基础、开发方法论与基于互联网的应用》，《计算机科学讲义》，第2432卷。Springer，柏林，海德堡（2002 4 6 10 1007 3 540 46237 ）。https://doi.org/ . / - - - . Bergmann, R., Gil, Y.：语义工作流的相似性评估与高效检索。Information Systems **40**, 115–127 (2014). https://doi.org/10.1016/ j.is.2012 004 7 11. . . Chase, H.：LangChain。GitHub代码库（2022），https://github.com/ langchain-ai/langchain 8. Guan, X., Li, Y., Wu, Z., Zhang, R.：NormCode：一种用于可审计人工智能规划的半形式语言（2026 10563 9 2512 ），https://arxiv.org/abs/ . . Guo, S., Deng, C., He, A., Huang, F., Tong, H., Huang, J.：DS-Agent：通过基于案例的推理赋能大语言模型，实现自动化数据科学。arXiv预印本 arXiv:2402 17453 10 17453 2024 2402 . ( )，https://arxiv.org/abs/ . . Hong, S., Zhuge, M., Chen, J., Zheng, X., Cheng, Y., Zhang, C., Wang, J., Wang, Z., Yau, S.K.S., Lin, Z., Zhou, L., Ran, C., Xiao, L., Wu, C., Schmidhuber, J.：MetaGPT：用于多智能体协作框架的元编程。载于：国际学习表征会议论文集（ICLR 2024）（ 2024），https://openreview.net/forum?id=VtmBAGCN7o 11. Hutter, R., Pradel, M.：AgentStepper：软件开发智能体的交互式调试。arXiv预印本 arXiv:2602.06593 (2026)，https://arxiv.org/abs/ 2602.06593

12. Kolodner, J.：《基于案例的推理》。Morgan Kaufmann，加利福尼亚州圣马特奥（1993）。https://doi.org/10 7 13 1016 2013 0 07426 . /C - - - . LangChain AI：LangGraph：以图的形式构建可靠的语言智能体。GitHub代码库（2024），https://github.com/langchain-ai/langgraph 14. Lenz, L., Hoffmann, M., Berg mann, R.：LLsiM：将大语言模型用作基于案例推理的相似度度量。载于：《基于案例的推理研究与发展》（ICCBR 2025）。《计算机科学讲义》，第15662卷，第126–141页。Springer，尚（2025）。https://doi.org/10.1007/ 978 3 031 63646 2_9 - - - - . 微软：PromptFlow：构建高质量的大语言模型应用。[15] GitHub代码库（2023），https://github.com/microsoft/promptflow 16. Minor, M., Montani, S., Recio-Garcia, J.A.：面向过程的基于案例推理。Information Systems **40**, 103–105 (2014). https://doi.org/10.1016/j.is.2013. 11.002 17. Muller, G., Bergmann, R.：广义工作流案例：学习基于工作流的案例调整。载于：第28届FLAIRS会议论文集（FL AIRS-28）。AAAI出版社（2015）18. Muller, G., Bergmann, R.：用于语义过程检索的基于工作流的查询语言。载于：LWA 2015研讨会论文集。CEUR研讨会论文集，第1458卷，第247–255页。CEUR-WS.org（2015），https://ceur-ws.org/ Vol-1458/ 19. OpenClaw：OpenClaw：开源智能体编程助手。软件（2026），https://github.com/openclaw-ai/openclaw 20. Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C.L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., Lowe, R.：通过人类反馈训练语言模型遵循指令。载于：神经信息处理系统进展（NeurIPS 2022）。第35卷，第27730–27744页。Curran Associates, Inc.（2022），https://proceedings.neurips.cc/paper_ files/paper/2022/hash/ b1efde53be364a73914f58805a001731-Abstract-Conference.html 21. Rubin, O., Herzig, J., Berant, J.：学习检索用于上下文学习的提示词。载于：2022年计算语言学协会北美分会会议：人类语言技术（NAACL-HLT 2022）论文集。第2655–2671页。计算语言学协会，华盛顿州西雅图（2022 191 22 10 18653 1 2022 ）。https://doi.org/ . /v / . naacl-main. . Wilkerson, J., Leake, D.B.：大语言模型作为基于案例的推理系统组件。载于：《基于案例的推理研究与发展》（ICCBR 2024）。《计算机科学讲义》，Springer，尚（2024）。https://doi. org/10 2 23 1007 978 3 031 63646 . / - - - - . Wu, Q., Bansal, G., Zhang, J., Wu, Y., Li, B., Zhu, E., Jiang, L., Zhang, X., Zhang, S., Liu, J., Awadallah, A.H., White, R.W., Burger, D., Wang, C.：AutoGen：通过多智能体对话赋能新一代大语言模型应用。arXiv预印本 arXiv:2308 08155 24 08155 2023 2308 . ( ), https://arxiv.org/abs/ . . Yang, J., Jimenez, C.E., Wettig, A., Lieret, K., Yao, S., Narasimhan, K., Press, O.：SWE-agent：智能体-计算机接口助力自动化软件工程。arXiv预印本 arXiv:2405 15793 25 15793 2024 2405 . ( )，https://arxiv.org/abs/ . . Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., Cao, Y.：ReAct：在语言模型中协同推理与行动。载于：国际学习表征会议论文集（ICLR 2023）（2023），https://openreview.net/forum?id=WE_vluYUL-X