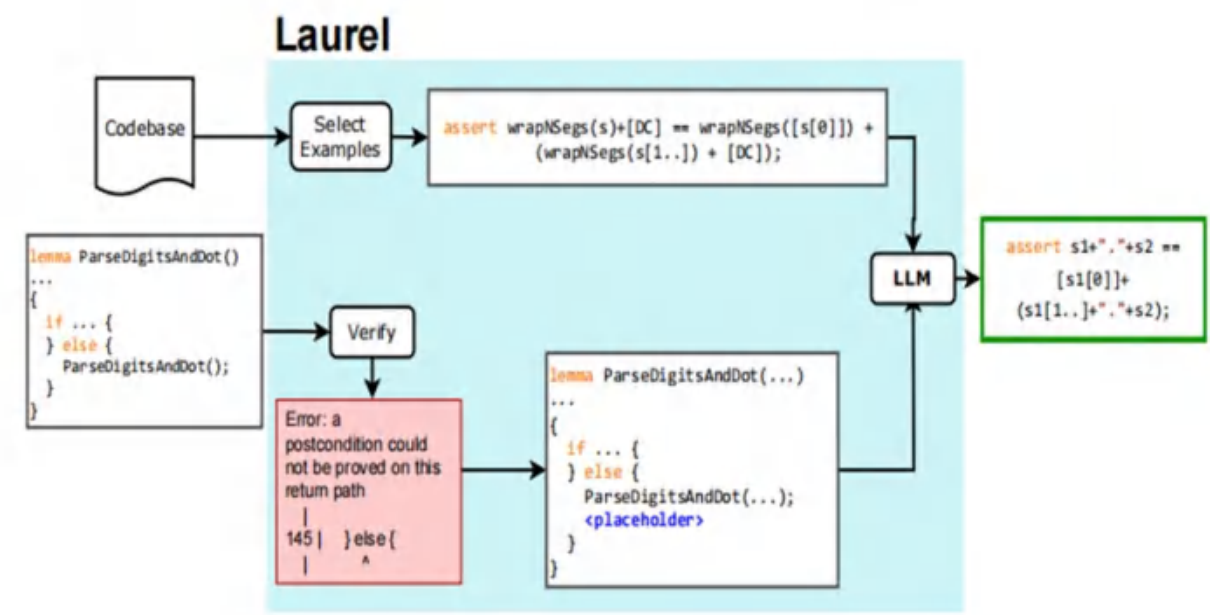
Laurel: Unblocking Automated Verification with Large Language Models

汇报人：吴文杰

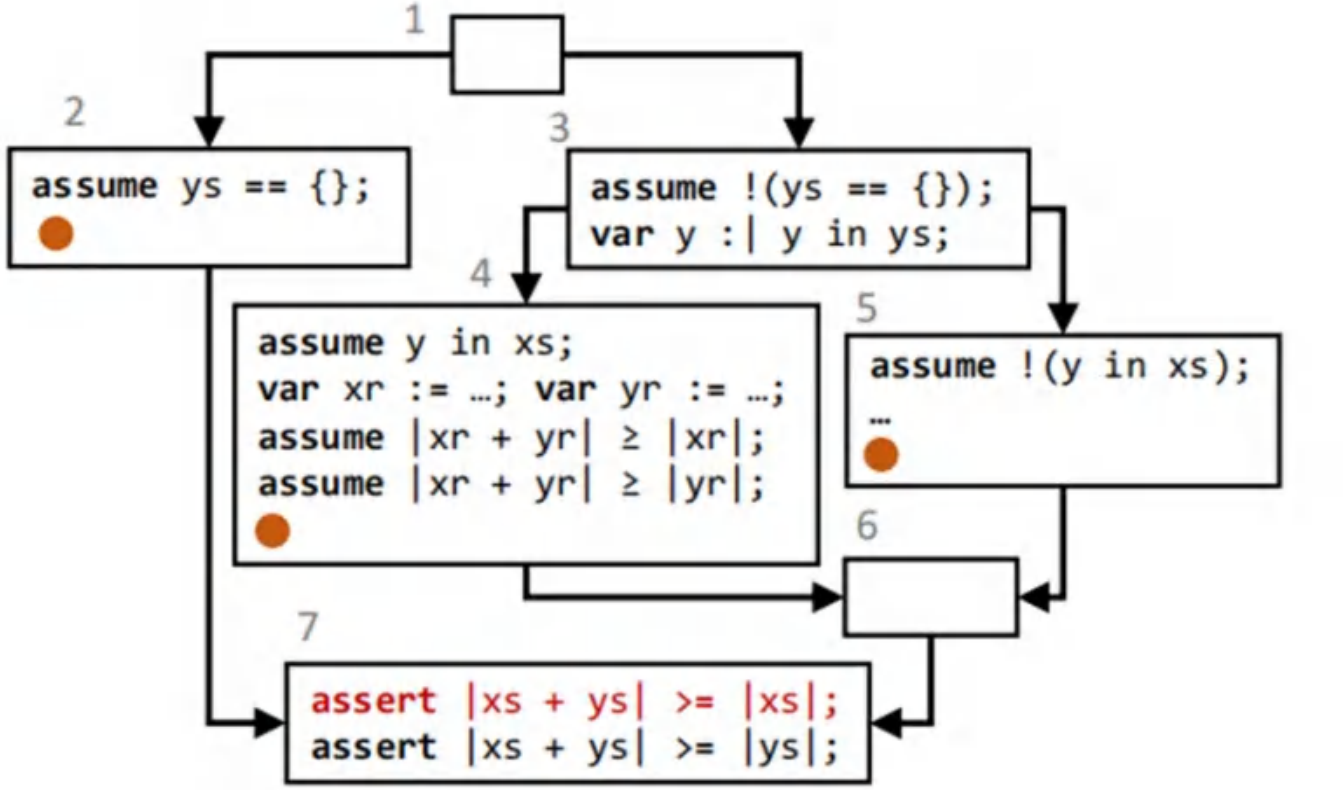
日期：2025-9-11

1.工作流框架



数据集构建（DafnyGym）： Dafny 标准库、Cedar 授权库和 DafnyVMC 蒙特卡洛算法库。

将目标 Dafny 程序转换为控制流图（CFG），把程序的分支、循环、函数调用等逻辑，拆解为节点（代表代码块）和边（代表控制流走向）的结构。



考虑从入口到违规点中，哪一条分支是验证失败的关键路径。

代码相似度算法

该算法的核心思路：不依赖复杂的语法解析，直接利用代码“行→标记→字符”的自然层级，捕捉结构相似性。

1. 令牌级（Token – Level）：统一标识符，简化对比

所有标识符（变量名、函数名、类名）视为等价令牌（比如变量 x 和 y，在令牌级别被视为同一类）；而关键字、运算符等非标识符令牌，仅与自身等价。

2. 行级（Line – Level）：对比行的令牌序列

将每行代码拆分为令牌序列后，用动态规划计算两行令牌序列的编辑距离（Edit Distance，衡量两个序列通过增 /删/ 改操作达到一致的成本）得到行相似度。

3. 证明级（Proof – Level）：对比引理 / 函数的行序列

一个代码单元（比如引理 lemma、函数 function）可视为行序列。以行相似度为基础，再次用**动态规划**计算两个代码单元的编辑距离，得到**证明相似度**。

文中在代码相似度计算中还提到了**神经嵌入**和 **TF-IDF**

神经嵌入是指将代码片段映射为向量，通过计算向量之间的距离（如余弦相似度），来衡量代码之间的相似性。

TF-IDF 由两个部分组成：词频（TF）和逆文档频率（IDF），在代码相似度计算中，可将代码中的元素（如标识符、关键字等）看作“词语”，代码文件看作“文档”，通过计算 TF-IDF 值来衡量不同代码文件中元素的重要性，进而评估代码文件之间的相似性，辅助筛选上下文示例等操作。

2.实验

以 DafnyGym 数据集的 143 个任务为测试对象，使用 Dafny 4.3.0 作为验证器，GPT-4o 作为 LLM，设置temperature为 1.0 增加生成多样性。每个任务允许 LLM 尝试 10 次，对比不同提示策略下的验证成功率。

RQ1) How effective are LLMs with a baseline prompt at generating assertions?

RQ2) Do assertion placeholders help LLMs generate Dafny assertions?

RQ3) Does selecting in-context examples via proof similarity help LLMs generate Dafny assertions?

研究问题	实验策略	核心参数	10 次尝试下成功率	关键结论

RQ1：LLMs 使用基线提示生成断言的有效性	Baseline	仅提供目标引理代码	6.2%	LLMs 难以定位问题，易在无关位置添加断言或修改前置 / 后置条件，基线策略效果极差
	Error	提供代码 + Dafny 初始错误消息	15%	错误消息可翻倍成功率，但 LLMs 仍难关联错误与代码位置，绝对成功率仍低
	Iterative	基线基础上，生成断言失败后反馈错误消息重试	29.6%	依赖增加尝试次数提升成功率，本质是扩大生成样本量，非提示策略本身优化
RQ2：断言占位符对 LLMs 生成 Dafny 断言的帮助	Placeholder	提供代码 + 断言占位符	34.4%	占位符利用 LLMs 代码填充优势，超半数断言首次尝试生成成功，大幅提升成功率；Cedar 因断言复杂，成功率仍最低

	Placeholder+Error	提供代码 + 占位符 + 错误消息	26.2%	错误消息对 Libraries 有负面影响，对 Cedar/DafnyVMC 无显著影响，易误导 LLM 或限制生成多样性，需更多尝试达最大成功率
RQ3：基于证明相似性选择上下文示例的作用	Placeholder	仅占位符（对照组）	34.4%	上下文示例对提升成功率至关重要，仅占位符效果有限
	Random	占位符 + 6 个随机代码库示例	51.7%	相同代码库的示例可提升成功率，证明上下文示例的基础价值
	Similarity	占位符 + 6 个基于证明相似性选择的示例	56.6%	相似性示例效果最优，与占位符互补；对复杂领域（Cedar）提升最显著
	Similarity-no-placeholder	无占位符 + 6 个相似性示例	46.8%	缺少占位符会削弱效果，证明占位符与上下文示例的协同作用

实验数据图

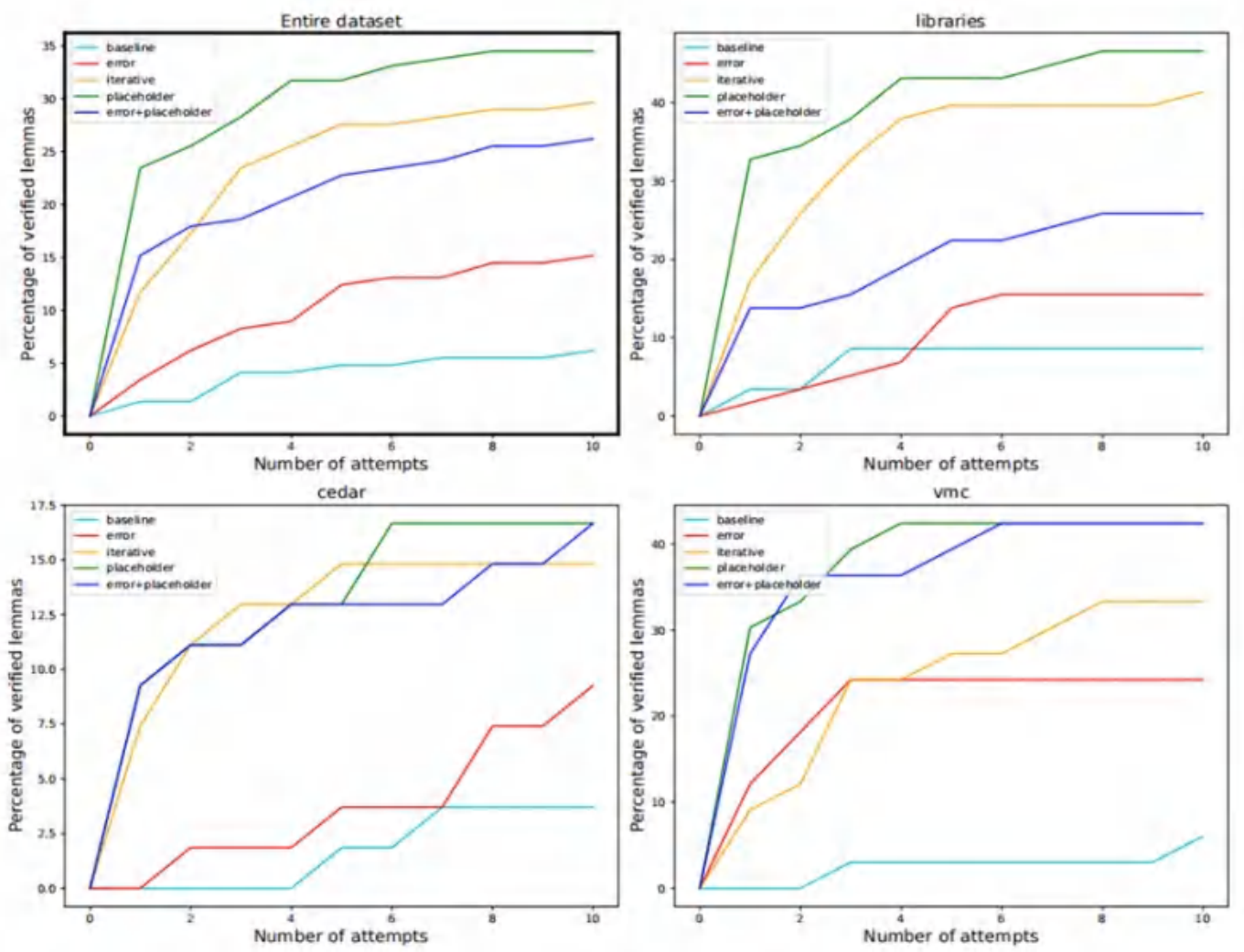


Fig. 9. Percentage of verified lemmas vs inference cost for the entire DafnyGym dataset, and by codebase, using placeholder.

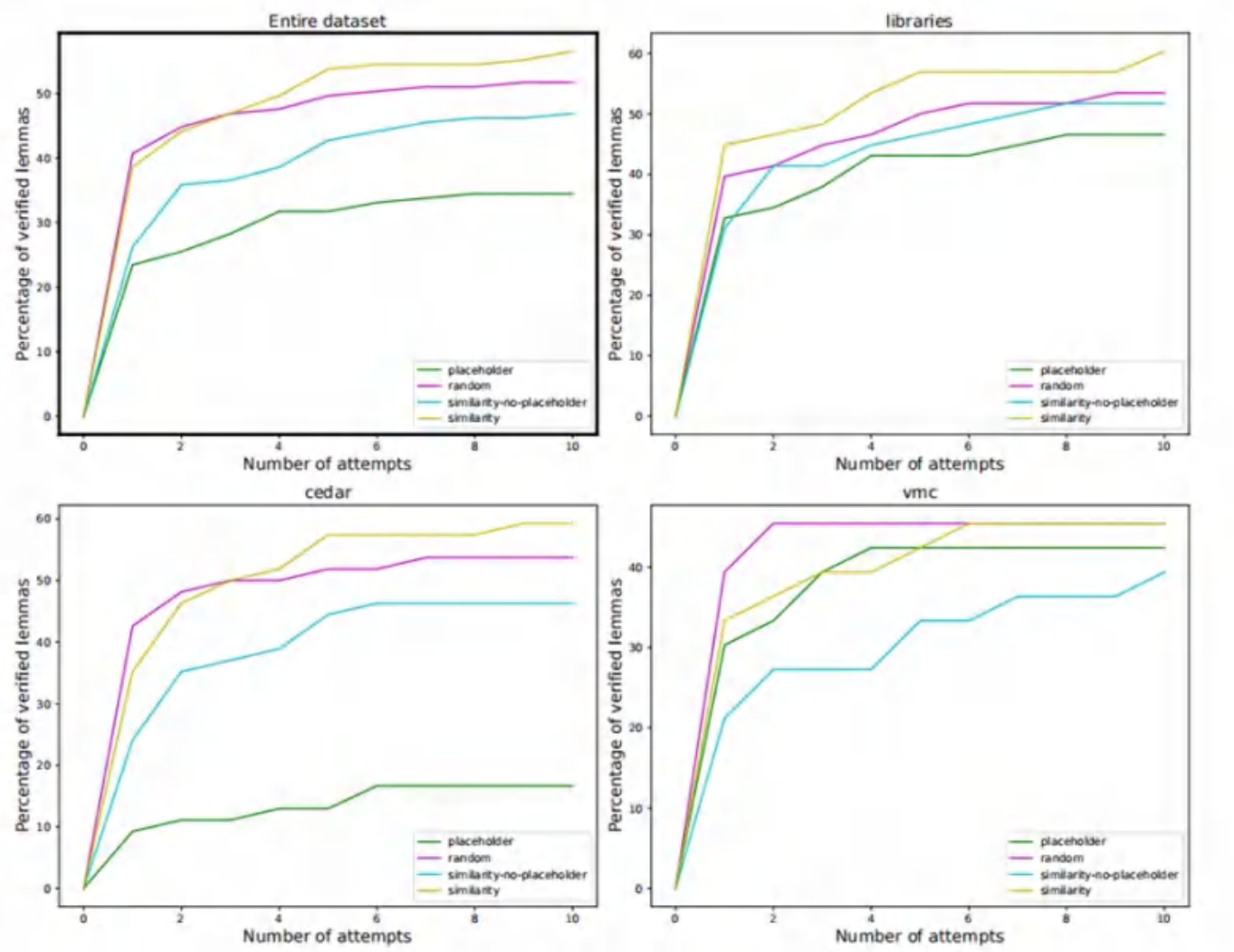


Fig. 11. Percentage of verified lemmas vs inference cost for the entire DafnyGym dataset, and by codebase, using similarity.

最主要的失败原因

1. **函数缺失上下文**

```
lemma SoundArith(op: BinaryOp, e1: Expr, e2: Expr, t: Type, effs:
    Effects)
  requires ...
  ensures ...
{
  assert TC.inferArith2(op,e1,e2,effs) = types.Ok(Type.Int);
  ...
}
```

TC.inferArith2是 Cedar 代码库中TC模块下的领域特定函数，作用是“推断二进制算术运算的结果类型”，其定义不在当前引理所在的文件中，且未在引理的依赖

> 中显式声明。

信息缺失：Laurel 未收集引理依赖的跨模块函数信息，导致 LLM 无法获知TC.inferArith2的存在；

示例局限性：相似示例仅提供TC.inferXX的命名前缀，无法覆盖函数后缀数字的领域规则（如1对应一元运算，2对应二进制运算），LLM 无法通过示例推断出后缀差异。

2.**函数过度使用干扰**

```
lemma LemmaFlattenConcat<T>(xs: seq<seq<T>>, ys: seq<seq<T>>)
  ensures Flatten(xs + ys) = Flatten(xs) + Flatten(ys)
  {
    if |xs| = 0 {
      assert xs + ys = ys;
    } else {
      ... // Flatten called another 6 times
    }
  }
```

引理体中Flatten函数（用于扁平化嵌套序列被过度使用—— 仅在引理的 else 分支中就被调用 6 次，加上前置条件、后置条件中的调用，总调用次数达 9 次。这种高频会导致LLM出现认知偏差，认为所有断言都必须包含Flatten函数，从而无法生成不依赖该函数的正确断言，最终验证失败。

针对之前提到的失败原因，我想到了以下优化的方向

1.扩展上下文收集范围，通过静态分析提取目标引理依赖的所有跨模块函数（如 `TC` 模块下的 `inferXX` 系列函数），将函数名列表加入 LLM 提示；

2.提示中加入断言无需包含所有函数的引导性说明（如 若基础 case 无需 `Flatten`，可生成不包含该函数的断言），帮助 LLM 跳出认知偏差。

思考

> **1.Laurel 仅支持单断言生成，而 LOOPY 通过 Houdini 算法支持多不变量子集筛选，为何 Laurel 不引入类似 Houdini 的符号机制解决多断言问题?**

> 2.LOOPY 支持 scalar、数组、递归、终止性四类任务，而 Laurel 仅聚焦断言生成，为何两者的任务覆盖范围有差异?
>
> 3.两者均依赖 LLM 与验证器的交互，但交互模式不同（Laurel：多轮生成→验证；LOOPY：生成→筛选→修复→验证）。能不能进行相互之间的借鉴呢