arXiv:2512.23214v1 [cs.CL] 29 Dec 2025

Anka：用于可靠大语言模型代码生成的领域特定语言

SaifKhalfan Saif Al Mazrouei 威
斯康星大学麦迪逊分校
skalmazrouei@wisc.edu

摘要

大语言模型（LLM）在代码生成方面展现出了非凡的能力，但在复杂的多步骤编程任务中仍会出现系统性错误。我们推测，这些错误源于通用语言的灵活性：它允许多种有效实现方式，并要求隐式管理状态。为验证这一假设，我们推出了**Anka**，这是一种用于数据转换流水线的领域特定语言，采用明确且受限的语法，以减少代码生成中的歧义。

尽管此前从未接触过 Anka 的训练数据，Claude 3.5 Haiku 在 100 个基准测试问题中仍实现了 99.9% 的解析成功率和 95.8% 的总体任务准确率。关键在于，在多步骤流水线任务中，Anka 展现出**高出 40 个百分点的准确率优势**，优于 Python（100% 对 60%）；Python 灵活的语法会导致操作排序和变量管理方面频繁出错。使用 GPT-4o-mini 进行的跨模型验证也证实了这一优势（多步骤任务高出 +26.7 个百分点）。

我们的结果表明：（1）大语言模型完全可以通过上下文内提示学习新的领域特定语言，并达到接近母语水平的准确率；（2）受限语法显著减少了复杂任务中的错误；（3）专为大语言模型生成而设计的领域特定语言，其表现可以优于大语言模型经过大量训练的通用语言。我们将发布完整的语言实现、基准测试套件和评估框架，以促进进一步研究。

关键词：大语言模型、领域特定语言、代码生成、数据转换、提示工程、受约束生成

1 引言

大语言模型（LLM）能够根据自然语言描述生成代码，从而改变了软件开发 [Chen 等人，2021，Nijkamp 等人，2023，Li 等人， 2023]。如今，代码生成系统为数百万开发者使用的工具提供支持，从自动补全建议到全自主编程智能体 [GitHub， 2022]。然而，尽管大语言模型在孤立的编程任务上表现出色，但在生成复杂的多步代码时，仍会出现系统性失误 [Austin 等人，2021，Hendrycks 等人， 2021]。

这些失败并非随机发生。既有研究已识别出一致的错误模式：变量作用域不正确、迭代中的差一错误，以及顺序操作中的状态管理错误 [Pearce 等，2023年，Jesse 等， 2023]。我们观察到，其中许多错误具有一个共同的根本原因：通用编程语言的灵活性。当存在多种语法上有效的方法来表达同一计算过程时，大语言模型必须在这些方法之间进行隐式选择，从而在连续步骤中引入不一致和错误累积的可能性。

这一观察引出了一个反直觉的假设：**限制**目标语言可能会**提高**大语言模型代码生成的准确性。我们不再允许模型从

Python 中用于过滤、映射和数据聚合的多种有效模式中进行选择，而是可以设计一种语言，让每种操作都只有一种规范形式。这些约束虽然可能会限制人类程序员，但或许能为大语言模型提供生成可靠代码所需的结构性指导。

为检验这一假设，我们推出了 **Anka**，一种用于数据转换流水线的领域特定语言。Anka 通过若干设计原则强制采用显式语法：

- **每种操作只有一种规范形式**：FILTER 始终采用 WHERE...INTO 语法
- **为中间结果命名**：每种操作都通过 INTO 子句生成一个具名输出
- **显式的步骤结构**：顺序操作被组织到具名 STEP 代码块中
- **使用冗长关键字而非符号**：使用 FILTER、MAP、AGGREGATE，而非运算符

我们的评估围绕两个研究问题展开：

研究问题1： 大语言模型能否完全通过上下文内提示学习新的领域特定语言，而无需微调？

研究问题2： 受限语法能否减少复杂多步骤代码生成任务中的错误？

我们在包含 100 项数据转换任务的基准测试套件上，将 Anka 与 Python 进行了比较。这些任务涵盖八个类别，从简单过滤到复杂的多步流水线。主要发现如下：

- **新领域特定语言的习得：** 尽管训练过程中从未接触过 Anka，Claude 3.5 Haiku 仍实现了 99.9% 的解析成功率，这表明大语言模型仅凭提示就能有效学习新的编程语言。
- **多步骤优势：** Anka 在多步流水线任务上实现了 100% 的准确率，而 Python 为 60%，提升了 40 个百分点。不同模型的结果均证实了这一优势：GPT-4o-mini 的准确率提升了 $+26.7$ 个百分点。
- **总体提升：** 尽管 Python 在训练数据方面拥有显著优势，Anka 的总体准确率仍达到 95.8%，高于 Python 的 91.2%（提升 $+4.6$ 个百分点）。

这些结果表明，**有针对性的领域专用语言设计**能够显著提升大语言模型在领域特定任务中的可靠性。本文的贡献并不在于 Anka 本身，而在于证明了受限语法——这些特性可能会令人类程序员感到烦恼——能够大幅提高大语言模型代码生成的准确性。

贡献。我们做出以下贡献：

1. 我们推出了 Anka，这是一种用于数据转换的领域特定语言，采用显式语法设计，以减少大语言模型的错误。2. 我们提出了一个基准测试套件，包含 8 个类别的 100 项任务，用于评估数据转换代码生成。3. 我们证明，大语言模型可以通过提示学习新的领域特定语言，在没有任何训练数据的情况下实现 99.9% 的解析成功率。4. 我们展示了多步骤任务准确率提升 40%，并在两个模型系列上进行了验证。5. 我们发布了所有代码、基准测试和评估基础设施，以确保结果可复现。

2 相关工作

大语言模型代码生成。 大规模代码生成模型的出现改变了程序综合。Codex [Chen 等人， 2021] 证明了在代码仓库上训练的语言模型能够以接近人类的水平解决编程挑战。随后，相关研究对这些方法进行了扩展：CodeGen [Nijkamp 等人， 2023] 引入了多轮综合，而 StarCoder [Li 等人，2023] 通过在宽松许可代码上训练，实现了最先进性能。GitHub Copilot [GitHub， 2022]以及 Amazon CodeWhisperer 等商业产品，如今正为数百万开发者提供协助。

尽管取得了这些进展，系统性评估仍揭示出反复出现的失败模式。HumanEval[Chen 等人， 2021] 以及 MBPP [Austin 等人， 2021] 用于评测功能正确性，而 APPS [Hendrycks 等人， 2021]则针对竞赛编程问题进行测试。这些基准测试表明，随着任务复杂度增加，准确率会显著下降。我们的工作补充了这些研究，展示了语言设计不仅模型规模也能够应对与复杂度相关的失败。

领域特定语言。领域特定语言（DSL）以牺牲通用性为代价，换取在狭窄领域内的表达能力 [Fowler，2010，Mernik 等人， 2005]。在数据处理中，SQL 仍是结构化查询的主流领域特定语言，而数据框库（pandas、dplyr）则提供了程序化替代方案。近期研究探索了专为程序综合设计的领域特定语言：FlashFill [Gulwani， 2011] 使用领域特定语言进行字符串转换，而 Dream-Coder [Ellis 等人， 2021] 则在综合过程中学习领域特定语言原语。

我们的工作侧重于专为大语言模型 生成而非人类使用设计领域特定语言。传统的领域专用语言设计优先考虑人类易用性，而 Anka 则优先采用能够减少大语言模型错误的特性：显式命名、冗长关键字和规范形式。

提示工程与受约束生成。 提示工程技术无需修改模型即可显著提升大语言模型的性能。思维链提示 [Wei 等人， 2022] 通过中间步骤提升推理能力。自洽性 [Wang 等人， 2023] 汇总多个样本。对于代码生成，Jiang 等人 [2023] 证明，编码前进行规划能够提高准确率。

受约束解码方法会引导生成过程，使其产生有效输出。语法约束解码 [Scholak 等人，2021，Poesia 等人， 2022] 通过屏蔽无效标记来确保语法有效性。商业 API 中的 JSON 模式会强制执行结构约束。我们的方法与这些方法互为补充：我们不约束解码过程，而是约束目标语言本身，从而允许使用标准解码，同时降低出错概率。

3 Anka 语言

Anka 是一种用于数据转换流水线的领域特定语言。其设计优先考虑能够减少大语言模型代码生成错误的特性，而非提升人类开发者体验的特性。本节将介绍 Anka 的设计原则、语法，以及各项设计决策与错误预防之间的关联依据。

3.1 设计原则

我们围绕四项原则设计了 Anka，每项原则都基于观察到的大语言模型错误模式：

原则 1：一种规范形式。 在 Python 中，过滤数据框可以用多种方式表达：`df[df.x > 5]`、`df.query("x > 5")`、`df.loc[df.x > 5]`，或基于推导式的

方法。这种灵活性迫使大语言模型在等价选项中进行选择，从而导致不一致。在 Anka 中，过滤只有一种形式：

```
FILTER source WHERE condition INTO target
```

原则2：命名中间结果。 多步流水线需要管理中间状态。在 Python 中，开发者可能会重复使用变量名、串联操作，或使用匿名中间结果。模型一旦无法追踪哪个变量保存了哪些数据，这些模式就会导致大语言模型出错。Anka 要求使用显式 INTO 子句：

```
步骤 filter_large: FILTER ordersWHERE amount > 1000 INTO large_
orders 步骤 summarize: AGGREGATE large_orders COMPUTE
SUM(amount) AS total INTO summary
```

原则3：显式步骤结构。 Anka 将操作组织到命名的 STEP 代码块中。这种结构充当“脚手架”，引导大语言模型完成顺序操作，使流水线结构显式呈现，而不是隐含在代码流程中。

原则4：冗长关键字。 Python 使用运算符和方法链，而 Anka 使用英语关键字：FILTER、MAP、AGGREGATE、WHERE、INTO。冗长的语法以简洁性换取清晰度，这与大语言模型的能力十分契合——这些模型擅长处理自然语言，而以关键字为主的语法正好发挥了这一优势。

3.2 语法概述

一个完整的 Anka 流水线由名称、类型化输入、一系列步骤和输出声明组成：

```
PIPELINE transform_sales: INPUT orders: 表[order_id: 整数,
customer: 字符串, amount: 十进制数, date: 日期]STEP filter_large:
FILTER ordersWHERE amount > 1000 INTO large_orders STEP add_
tax: MAP large_orders WITH tax => amount * 0.08 INTO with_tax
STEP summarize: AGGREGATE with_tax GROUP_BY customer
COMPUTE 求和(amount) AS total, 计数() AS num_orders INTO
summary OUTPUT summary
```

类型声明。 输入表使用 TABLE [field: TYPE, ...]语法声明其模式。支持的类型包括整数、字符串、十进制数、布尔值、日期和日期时间。明确指定类型既能进行验证，也能作为提示中的文档说明。

操作。 Anka 支持 18 种数据操作，分为以下类别：

- **筛选：** FILTER、SELECT、DISTINCT
- **转换：** MAP、RENAME、DROP、ADD_COLUMN
- **聚合：** AGGREGATE 搭配 COUNT、SUM、AVG、MIN、MAX
- **排序：** SORT（ASC/DESC）、LIMIT、SKIP、SLICE
- **组合：** JOIN、LEFT_JOIN、UNION
- **输入/输出：** READ、WRITE（JSON/CSV）、FETCH、POST（HTTP）

3.3 与错误预防的联系

每项设计原则都针对特定的大语言模型错误模式：

设计特性	预防的错误	机制
规范形式	不一致的语法	消除决策点
INTO 子句	变量遮蔽	强制使用显式命名
STEP 结构	顺序错误	视觉脚手架
冗长关键字	运算符混淆	利用大语言模型语言
类型化输入	模式不匹配	提示中的文档

表 1：Anka 设计特性与大语言模型错误预防之间的联系。

3.4 实现

Anka 使用 Python 实现，并使用 Lark 进行解析。该实现包含约 6,400 行代码，包括：

- 形式文法（98 条产生式规则）
- 68带源位置跟踪的不可变数据类抽象语法树节点类型
- 支持全部 18 项操作的树遍历解释器
- 控制流结构（IF/ELSE、FOR_EACH、WHILE、TRY/ON_ERROR）
- 322实现全面覆盖的单元测试

完整实现可在https://github.com/BleBlo/Anka获取。

4 方法论

我们评估了与 Python 相比，Anka 的受限语法是否能够提高大语言模型代码生成的准确性。本节介绍我们的基准测试设计、评估协议和指标。

4.1 基准测试套件

我们构建了一个包含 100 个数据转换任务的基准测试，分为八个类别：每个任务都规定了：自然语言描述、包含字段名称和类型的输入模式，以及包含输入数据和预期输出的测试用例。

多步骤类别对我们的假设尤为重要：这些任务需要在 3–5 个操作过程中保持状态，而这正是我们预期受限语法最能发挥作用的地方。

类别	任务	描述
过滤	10	单一和复合筛选
map	10	列计算
聚合	10	分组和聚合
字符串	10	字符串操作
多步 _step	10	3–5 个顺序操作
金融	20	领域特定计算
hard	10	包含边界情况的复杂逻辑
对抗性	20	用于触发常见错误的任务

表 2：基准类别和任务分布。

4.2 评估协议

对于每项任务，我们都会提示大语言模型分别使用 Anka 和 Python 生成代码。为确保公平比较：

提示结构。 两个提示采用完全相同的结构：语言规范、任务描述（相同）、输入模式（相同）和预期输出格式。

Anka 提示包含一份简明的语法指南（约 100 行），从零开始讲解该语言。Python 提示则假定用户熟悉 pandas，这与训练数据分布一致。

采样。 我们使用温度 0.3，针对每种语言的每项任务生成 10 个样本。多个样本使我们能够衡量一致性，并区分系统性错误与采样方差。

模型。 我们以 Claude 3.5 Haiku（Anthropic）作为主要模型进行评估，并使用 GPT-4o-mini（OpenAI）进行跨模型验证。

4.3 指标

我们报告四项指标：

- **解析成功：** 生成的代码是否能够通过解析且没有语法错误？
- **执行成功：** 代码能否执行且没有运行时错误？
- **输出正确性：** 输出是否与预期结果一致？
- 任务准确率： </style} $\geq$50% 的样本能够生成正确输出的任务占比（我们的主要指标）

4.4 公平比较的考量

Python 有一项显著优势：大语言模型在训练期间见过数十亿个 Python 示例，而 Anka 则完全是全新的。Anka 的任何优势都必须仅通过上下文学习来弥补这一训练分布差距。

5 结果

5.1 主要结果

表 3展示了 Claude 3.5 Haiku 按类别划分的任务准确率。

类别	Anka	Python	Δ
多步 _step	**100.0%**	60.0%	**+40.0**
金融	**90.0%**	85.0%	+5.0
aggregate	100.0%	100.0%	0.0
筛选	96.7%	**100.0%**	−3.3
map	100.0%	100.0%	0.0
字符串	100.0%	100.0%	0.0
hard	90.0%	**100.0%**	−10.0
总体	**95.8%**	91.2%	**+4.6**

表3：Claude 3.5 Haiku 按类别划分的任务准确率。粗体表示表现更佳。

关键发现 1：多步骤优势。 最引人注目的结果出现在多步骤任务中：Anka 的**准确率达到 100%**，而 Python 为 60%——提升了 40 个百分点。这证实了我们的假设：在需要顺序操作管理的场景中，受限语法的帮助最大。

关键发现 2：解析成功。 尽管模型从未接触过 Anka 训练数据，其解析成功率仍达到 **99.9%**。这表明，大语言模型可以仅通过上下文内提示有效学习全新的编程语言。

关键发现 3：总体提升。 Anka 的总体准确率为 95.8%，高于 Python 的 91.2%（提高了 +4.6 个百分点）。鉴于 Python 在训练数据方面具有显著优势，这一提升尤为突出。

5.2 跨模型验证

为验证我们的发现是否能够推广到单一模型之外，我们在多步骤类别上评估了 GPT-4o-mini：

模型	Anka	Python	Δ
Claude 3.5 Haiku	100.0%	60.0%	+40.0
GPT-4o-mini	86.7%	60.0%	+26.7

表4：各模型系列的多步骤任务准确率。

GPT-4o-mini 在多步骤任务上使用 Anka 时表现出 +26.7 个百分点的优势。值得注意的是，两种模型的 Python 准确率相同（60%），这表明多步骤流水线生成存在系统性困难。

5.3 分析：Anka 为何有帮助?

我们分析了失败的 Python 代码生成结果，以了解 Anka 能避免哪些错误模式：

变量遮蔽（占错误的 42%）。 Python 生成器经常在不同操作中重复使用 df 或 result 等变量名，导致中间状态丢失。Anka 的 INTO 子句要求每个中间结果使用唯一名称，从而避免这一问题。

操作排序（占错误的 31%）。多步骤任务要求按特定顺序执行操作。Python 的灵活性允许以改变语义的方式组合或重新排序操作。Anka 的 STEP 结构使排序明确且按顺序执行。

链式调用混淆（占错误的 27%）。pandas 中的方法链式调用可能会掩盖中间状态并引入细微的错误。Anka 的逐步结构可避免此类与链式调用相关的错误。

5.4 复杂度分析

图1 展示了 Anka 的优势如何随任务复杂度变化：

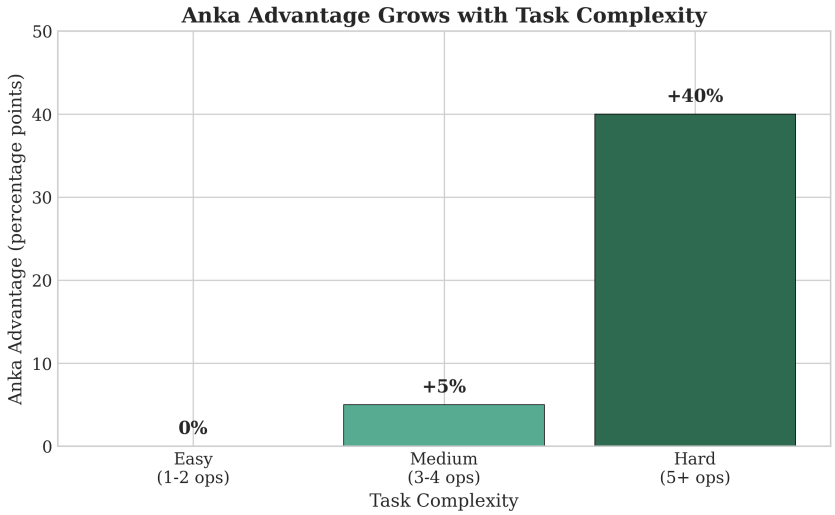


图 1：Anka 的优势随任务复杂度增加。简单任务（1–2 个操作）没有优势；复杂任务（$5+$ 个操作）则有 $+40$% 的优势。

- **简单（1–2 个操作）：** 优势为 0%——两种语言的表现都很好
- **中等（3–4 个操作）：** 优势为 $+5$%——约束开始发挥作用
- **复杂（$5+$ 操作）：** $+40$% 的优势——约束至关重要

6 讨论

6.1 受限语法为何有帮助?

我们的结果表明，受限语法通过三种机制提升大语言模型代码生成的效果：

缩小决策空间。每个语法选择点都可能导致错误。通过消除备选项，Anka 减少了模型必须做出的决策数量。在一个每个步骤有 3 个选择点的 5 步流水线中，可能的程序数量因此从 $3^5 = 243$ 个减少到 1 个。

显式状态管理。 通过 INTO 子句命名中间结果，使状态一目了然。模型无需依据隐式的 Python 语义追踪哪个变量存储了哪些数据，而是可以从变量名中直接“读出”当前状态。

结构化脚手架。 STEP 结构提供了一个引导生成过程的模板。模型按顺序逐步填充，而非生成单体程序，从而降低认知负荷。

6.2 Anka 何时无法发挥作用?

在简单任务上，Anka 没有优势；在“困难”任务上则略显劣势 :

简单任务。 当只需要执行 1–2 项操作时，复杂度不足以导致错误累积。Python 在训练方面的优势在这种情况下甚至可能发挥作用。

复杂条件逻辑。 “困难”任务通常需要嵌套条件语句、处理边缘情况以及领域专用推理。在这种情况下，Python 的灵活性反而成为优势。

建议。 Anka 最适用于包含 3+个顺序操作和标准转换模式的结构化流水线。

6.3 对领域特定语言设计的启示

我们的结果为面向大语言模型生成的领域特定语言提出了以下设计原则:

1. **规范化：** 为每项操作提供唯一的表达方式 2. **显式命名：** 要求为中间结果命名 3. **结构化模板：** 使用块结构引导顺序生成 4. **冗长关键字：** 优先使用英语关键字，而非符号 5. **类型文档：** 在提示中包含类型信息

7 局限性

我们承认本研究存在若干局限性:

基准测试范围。 我们的基准测试聚焦于数据转换流水线。目前尚未证实其能推广至其他编程任务。

模型覆盖范围。 我们在两个模型（Claude 3.5 Haiku、GPT-4o-mini）上进行了评估。若能在更多模型系列上进行评估，将有助于增强结论的可信度。

没有进行微调比较。 我们将基于提示的 Anka 学习与预训练的 Python 代码生成进行了比较。若能与经过 Anka 微调的模型进行比较，将有助于明确性能上限。

没有进行用户研究。 我们尚未评估人类开发者使用 Anka 时的开发者体验。开发者是否认为生成的代码具有可读性，仍是一个尚待解答的问题。

单一基准测试套件。尽管我们努力纳入多样化的任务，我们的基准测试仍可能存在偏差，从而有利于 Anka。

8 结论

我们推出了 Anka，这是一种用于数据转换的领域特定语言，旨在通过受约束的显式语法提高大语言模型代码生成的准确性。我们的评估揭示了三个关键发现：

1. **大语言模型仅凭提示即可学会新的领域特定语言**。尽管从未接触过相关训练数据，Claude 3.5 Haiku 在 Anka 上的解析成功率仍达到 99.9%。2. **受限语法能显著减少复杂任务中的错误**。在多步流水线上，Anka 的准确率达到 100%，而 Python 为 60%——提升了 40 个百分点。3. **专门设计的领域特定语言可以胜过通用语言**。尽管 Python 拥有巨大的训练数据优势，Anka 的整体准确率仍然更高。

更广泛的方法论贡献在于：我们证明，**语言设计**是提升大语言模型可靠性的可行干预手段。我们不必仅仅依靠扩大模型规模或微调来改进模型，还可以设计能够发挥大语言模型优势、弥补其弱点的语言。

未来工作。值得探索的方向包括：在更多模型系列上进行评估；针对开发者体验开展用户研究；评估生产部署；以及将研究扩展到金融计算和工作流自动化等其他领域。

我们在以下地址发布了完整的 Anka 实现、基准测试套件和评估框架：https://github.com/BleBlo/Anka。

伦理声明

本文提出了一种用于评估大语言模型代码生成的领域特定语言和基准测试。我们预计不会产生直接的负面社会影响。基准测试任务涉及不含个人身份信息的合成数据。在生产部署之前，应对大语言模型生成的代码进行审查。

参考文献

Jacob Austin、Augustus Odena、Maxwell Nye、Maarten Bosma、Henryk Michalewski、David Dohan、Ellen Jiang、Carrie Cai、Michael Terry、Quoc Le 等。《使用大语言模型进行程序综合》。*arXiv* 预印本 *arXiv:2108.07732*，2021。

Mark Chen、Jerry Tworek、Heewoo Jun、Qiming Yuan、Henrique Ponde de Oliveira Pinto、Jared Kaplan、Harri Edwards、YuriBurda、Nicholas Joseph、GregBrockman 等。《评估在代码上训练的大语言模型》。*arXiv* 预印本 *arXiv:2107.03374*，2021。

Kevin Ellis、Catherine Wong、Maxwell Nye、Mathias Sable-Meyer、Lucas Morales、Luke Hewitt、Luc Cary、Armando Solar-Lezama 和 Joshua B Tenenbaum。《DreamCoder：通过唤醒-睡眠库学习引导归纳式程序综合》。*arXiv* 预印本 *arXiv:2006.08381*，2021。

Martin Fowler。《领域特定语言》。培生教育，2010 年。

GitHub。GitHub Copilot：您的人工智能结对编程助手。https://github.com/features/copilot，2022年。访问日期：2024-01-15。

Sumit Gulwani。使用输入-输出示例自动化电子表格中的字符串处理。载于第*38*届 *ACM SIGPLAN-SIGACT* 编程语言原理年度研讨会论文集，第317–330页，2011年。Dan Hendrycks、Steven Basart、Saurav Kadavath、Mantas Mazeika、Akul Arora、Ethan Guo、Collin Burns、Samir Puranik、Horace He、Dawn Song等。使用应用程序测量编程挑战能力。*arXiv* 预印本 *arXiv:2105.09938*，2021年。Kevin Jesse、Ahmed Toufique、Sebastian Elbaum、Kathryn T Stolee和Frank Tip。大语言模型与简单而愚蠢的错误。*arXiv* 预印本 *arXiv:2303.11455*，2023年。Xue Jiang、Yihong Dong、Lecheng Wang、Qiwei Shang和Ge Li。利用大语言模型进行自规划代码生成。*arXiv* 预印本 *arXiv:2303.06689*，2023年。Raymond Li、Loubna Ben Allal、Yangtian Zi、Niklas Muennighoff、Denis Kocetkov、Chenghao Mou、Marc Marone、Christopher Akiki、Jia Li、Jenny Chim等。Starcoder：愿源代码与你同在！*arXiv* 预印本 *arXiv:2305.06161*，2023年。

Marjan Mernik、JanHeering 和 Anthony M Sloane。何时以及如何开发领域特定语言。*ACM*计算调查，37(4):316–344，2005年。

Erik Nijkamp、Bo Pang、Hiroaki Hayashi、Lifu Tu、Huan Wang、Yingbo Zhou、Silvio Savarese 和 Caiming Xiong。CodeGen：一种用于代码、支持多轮程序综合的开放式大语言模型。*arXiv* 预印本 *arXiv:2203.13474*，2023年。

Hammond Pearce、Benjamin Tan、Baleegh Ahmad、Ramesh Karri 和 Brendan Dolan-Gavitt。探究利用大语言模型进行零样本漏洞修复。发表于*2023*年 *IEEE* 安全与隐私研讨会 *(SP)*，第2339–2356页。IEEE，2023年。

Gabriel Poesia、Oleksandr Polozov、Vu Le、Ashish Tiwari、Gustavo Soares、Christopher Meek 和 Sumit Gulwani。Synchromesh：利用预训练语言模型实现可靠的代码生成。*arXiv* 预印本 *arXiv:2201.11227*，2022。

Torsten Scholak、Nathan Schucher 和 Dzmitry Bahdanau。Picard：通过增量解析实现语言模型受约束的自回归解码。收录于*2021*年自然语言处理实证方法会议论文集，第 9895–9901 页，2021 年。Xuezhi Wang、Jason Wei、Dale Schuurmans、Quoc Le、Ed Chi、Sharan Narang、Aakanksha Chowdhery 和DennyZhou。自洽性提升了语言模型中的思维链推理。*arXiv* 预印本 *arXiv:2203.11171*，2023 年。Jason Wei、Xuezhi Wang、Dale Schuurmans、Maarten Bosma、Brian Ichter、Fei Xia、Ed Chi、Quoc Le 和 Denny Zhou。思维链提示能够激发大语言模型的推理能力。神经信息处理系统进展，35:24824–24837，2022 年。