arXiv:2603.05728v1 [cs.LO] 5 Mar 2026

LTLGUARD：利用紧凑型语言模型和轻量级符号推理形式化 LTL 规范

Medina Andresel MEDINA.ANDRESEL@AIT.AC.AT
Cristinel Mateis CRISTINEL.MATEIS@AIT.AC.AT
Dejan Ni˘ckovi´c DEJAN.NICKOVIC@AIT.AC.AT
奥地利技术研究院，维也纳，奥地利

Spyridon Kounoupidis SKOUN@CSD.AUTH.GR
Panagiotis Katsaros KATSAROS@CSD.AUTH.GR
塞萨洛尼基亚里士多德大学，塞萨洛尼基，希腊

Stavros Tripakis STAVROS@NORTHEASTERN.EDU
东北大学，波士顿，马萨诸塞州，美国

摘要

由于自然语言（NL）存在歧义与表达差异，将非正式需求转化为形式化规格颇具挑战。当依赖紧凑型（小型和中型）语言模型时，这一挑战尤为突出，因为它们可能缺乏关于时序逻辑的扎实知识，因而难以生成语法有效且一致的形式化规格。在本工作中，我们致力于使资源高效的开放权重模型（参数量为4B–14B）能够根据非正式需求生成正确的线性时序逻辑（LTL）规范。我们提出LTLGUARD，这是一个模块化工具链，将受约束生成与形式一致性检查相结合，从而根据非正式输入生成无冲突的LTL规范。我们的方法将模型语言的生成能力与轻量级自动推理工具相结合，以迭代优化候选规范、理解冲突的来源，从而帮助消除不一致。我们展示了该方法的可用性与有效性，并对所得框架进行了定量评估。

1. 引言

形式化规范在广泛的验证活动中发挥着至关重要的作用。然而，将非正式需求转换为形式化规格仍是一项重大挑战，并被广泛认为是阻碍形式化方法在工业实践中得到更广泛应用的关键因素之一Davis 等人(2013年）。掌握这项需求工程工作需要具备特定技能，且学习曲线陡峭。根据定义，非正式需求也容易产生多种解读。以线性时序逻辑（LTL）这一规范形式体系为例Pnueli(1977）以及以下非正式需求：每个请求之后都会有一个授权。这一未充分明确的需求至少有三种不同的解读：每个请求之后的授权发生在（1）同一时刻、（2）下一时刻，或（3）严格的未来时刻。这个例子说明，需求形式化并不存在绝对真理——根据隐含假设和非正式需求的预期含义，每种解读都可能正确，也可能错误。

过去几十年来，为促进需求形式化，人们提出了多种方法 Buzhinsky (2019年)。提出的方法包括基于模式的规约 Dwyer 等人 (1998年)、Konrad 和 Cheng (2005年)、Giannakopoulou 等人 (2021）、样板Mokos 等人(2022年）、自然语言解析树 Yan 等人 (2015），以及统计和神经机器

翻译 He 等人 (2022）。尽管方法种类丰富，将非正式需求翻译为形式化规格的问题仍远未得到彻底解决。

大语言模型（LLM）的出现彻底改变了软件工程。大语言模型非常适合帮助工程师将非正式需求转化为形式化规范，因为它们能够理解自然语言上下文、识别领域特定模式并生成结构化输出。旗舰模型为需求形式化提供了令人印象深刻的基线。然而，在实际应用中采用这些模型仍存在多方面顾虑。首先，这些模型通常过于庞大，无法在本地部署；当必须将敏感需求发送到外部托管服务时，便会引发隐私问题。此外，它们体积庞大，能耗较高，成本与收益之间的权衡也值得商榷。最后，即使是旗舰大语言模型，也容易生成看似可信、但语义上具有误导性的幻觉Bai 等人(2024年)。专有大语言模型对解码和内部安全机制的控制有限，因此难以可靠地约束输出。相比之下，紧凑型开放权重模型可以部署在本地机器上，但在需求形式化等小众、逻辑密集型任务上的可靠性往往较低，也更容易产生缺乏依据或前后矛盾的输出（Ul Islam 等人，2025年）。

我们提出 LTLGuard，这是一个保护隐私的框架，利用紧凑型语言模型将非正式需求翻译为线性时序逻辑。这些模型不一定经过时序逻辑训练，其开箱即用的翻译准确率也无法媲美大型旗舰模型。我们不依赖微调等计算密集型方法，而是展示了轻量级技术（包括语法约束解码、检索增强少样本学习和一致性检查）如何大幅提升这些紧凑型模型的鲁棒性和翻译准确率。通过这种方式，我们证明了，只需以有原则的方式整合轻量级大语言模型提示与自动推理，无需专门训练或大规模模型适配，如今也能实现可靠且相互一致的形式化表示（Nestoridis 等人 (2026）。

2. 相关工作

除第1节提及的工作外，近期还有大量基于大语言模型的方法，可将非正式需求翻译为形式化规范。Fuggitti 和 Chakraborti(2023年)提出了 NL2LTL，通过将输入映射到预定义模式（如 DECLARE 模板），并使用过滤函数依据冲突或蕴含关系对候选公式排序，从而将自然语言翻译为线性时序逻辑。我们利用紧凑型语言模型，以更灵活地生成规范；同时借助基于自动推理的细化循环，帮助用户解决固定模式集合之外的语义不一致问题。Cosler 等人(2023年)提出了 nl2spec，这是一种交互式大语言模型框架，可将自然语言需求翻译为时序逻辑（包括线性时序逻辑），并特别强调人在回路消歧。其核心思想是将一项需求分解为细粒度子公式，从而在通过下游解析检查语法有效性的同时，实现迭代消歧。相比之下，我们的方法着重于语法约束生成，以及对规范集合进行自动一致性检查，以解决跨公式冲突，而非逐个公式进行语义调试。Kogler等人(2024年)通过基于自动机的流式后处理器，在词元生成过程中施加严格的语法约束，提高大语言模型生成规范的可靠性。其框架采用确定性有限自动机（DFA），依据 JSON 模式实时验证候选内容，确保输出符合形式化领域特定语言（DSL）。我们不将控制局限于词元级语法过滤，而是将自动语法纠正、迭代细化循环和自动推理相结合，以揭示翻译所得时序

逻辑规范中的语义问题和逻辑冲突。Xu等人(2024年)提出了一种交互式自然语言到LTL翻译方法，该方法利用基于嵌入的检索和动态提示演化，通过不断将用户修正后的翻译加入少样本示例，逐步提升每项需求的准确率，而无需微调。类似地，我们的方法也支持模块化示例检索和数据集增量扩充，但更侧重于语法约束生成和自动推理，以确保不同线性时序逻辑规范之间的一致性，而非仅关注单项翻译。Ma等人(2025年)提出了 REQ2LTL，这是一种两阶段框架，通过大语言模型将需求分解为分层中间表示，再将其翻译为语法正确的线性时序逻辑，从而减少端到端生成错误。我们采用紧凑型模型，目标是实现跨规范的一致性，而非仅确保每项需求的结构保真度。我们将语法约束解码、迭代修复和自动推理相结合，用于检查线性时序逻辑公式的一致性，并引导用户解决冲突。Ma等人报告了使用大型大语言模型后端取得的结果，但未公开完整实现和工业数据集，因此难以在我们的紧凑型语言模型设置下进行直接比较。Cohen等人(2025年)提出了 LLMon，这是一种交互式框架，可根据自然语言需求合成 LTL过去时监视器，生成多种语义解释（包括可执行监视器），并利用区分轨迹帮助用户在单个算子层面消除歧义。LLMon 在翻译过程的每个步骤中都只使用大语言模型，而我们的方法则着重于结合紧凑型模型与符号方法来处理多项需求。Shukla等人(2025年)提出了一种灰盒自然语言到时序逻辑翻译方法，该方法以结构化中间表示为条件输入大语言模型，并结合精选示例与微调来提高单公式翻译准确率，再通过下游验证工具评估正确性。相比之下，我们的工作不采用微调，而是着重于使用更轻量的方法提升翻译准确率。

3. 非正式需求形式化框架

本节介绍我们将需求形式化为 LTL 规范的框架。在第 3.1 节中，我们阐述要解决的问题以及适当解决方案应具备的属性。第 3.2 节概述我们的框架，第 3.3 节则详细介绍其组成模块。

3.1 问题定义与解决方案要求

我们要解决的需求形式化问题如下：给定一组以自然语言表述的功能需求，目标是推导出一组相应的 *LTL* 规范，忠实地表达预期语义，且彼此一致。

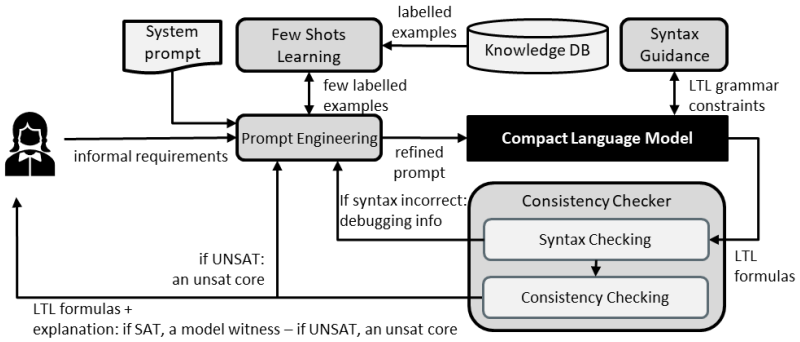
我们希望采用一种基于紧凑型语言模型的解决方案，该模型可在本地计算基础设施上运行，且无需专业知识或大规模模型调优。我们认为，正确性和鲁棒性是此类解决方案最重要的两项属性。

正确性。 我们从三个维度考量正确性。第一个维度是句法正确性：生成的输出必须是语法有效的线性时序逻辑公式。第二个维度是语义正确性：公式应保留非正式需求的预期语义。尽管语义正确性至关重要，但由于缺乏针对非正式到形式化翻译的权威真实值，实践中无法对其进行全面评估。第三个维度是歧义处理：解决方案应消除自然语言需求中的歧义表述；根据不同的解释，这些表述可能对应多个语义有效的形式化表示。

鲁棒性。 鲁棒性是指翻译对于输入的细微变化的稳定性。我们区分两种鲁棒性概念，详见第 4.3.2 节。

3.2. 框架概述

图 1 所示为所提出的方法。该工具链为需求形式化提供半自动化支持，同时让人类参与其中。作出这一选择的原因是，正确的形式化表示并不存在明确的真实值。工作流程从工程师编写的提示开始，请求将一组非正式需求转换为线性时序逻辑规范。单独的系统提示则为该任务提供必要的背景上下文。用户提示还通过检索增强少样本学习（RAFSL）模块检索到的额外自然语言-LTL 配对进一步充实。RAFSL 会从数据集中动态选择相关示例并将其纳入提示，从而为翻译器提供特定于任务的上下文。



随后，将充实后的提示提供给紧凑型语言模型。其输出受语法引导模块约束，该模块利用线性时序逻辑语法引导模型生成语法正确的翻译。随后对生成的候选公式进行检查

图 1：规范形式化方法概述。

由线性时序逻辑解析器检查，以确保语法有效性。如果解析失败，则将调试信息反馈给模型，以支持迭代优化，直至得到语法正确的公式。公式通过语法检查后，一致性检查器会验证其中是否存在相互冲突，这可能表明需求确实不一致，也可能表明其形式化存在错误。为帮助工程师了解问题根源，工具链会标出冲突需求，并提供一致性检查器计算出的相应不可满足核。对于不一致情况，系统会将这些诊断信息反馈给模型，尝试自动解决冲突。

3.3. 框架构成模块

3.3.1. 提示规范

使用语言模型确保自然语言需求得到正确且稳健翻译的第一步，是设计提示规范。它通过明确说明模型的角色、假设和预期输出结构，定义系统提示，以约束并引导大语言模型的行为，从而完成预期任务。提示规范通常是一个迭代过程，需要多次完善适用于特定任务的系统提示。我们最终确定了清单3.1所示的系统提示。它定义了模型的角色，提供了有关线性时序逻辑的语法指南，并规定了模型输出的格式。

列表 3.1：系统提示

你是形式化验证和时序逻辑方面的专家。你将收到文本形式的需求。你的任务是按照以下约定，将每项需求翻译成线性时序逻辑公式：- 时序算子：G = 始终，F = 最终，X = 下一个，U = 直到。- 原子命题必须是小写单词（例如 request、granted），不能是单个大写字母。- 严格且仅输出线性时序逻辑公式。不要包含任何推理、解释、步骤或自然语言。输出中只能包含公式，不能有其他内容。- 如果提供的文本完全与线性时序逻辑需求无关，只需输出“提供的文本与线性时序逻辑无关”。

3.3.2. 检索增强少样本学习（RAFSL）

与大型专有大语言模型相比，较小的大语言模型在小众任务中往往幻觉率更高、准确率更低（Ul Islam 等人，2025年）。在用于需求形式化时，由于预训练期间接触时序逻辑的机会有限，较小的大语言模型可能会生成语义或语法错误的公式。为弥补这些不足，我们采用检索增强少样本学习（RAFSL）方法，在推理时动态地向模型提供相关示例。

不同于常见的少样本学习技术（将预先设定的固定示例集直接添加到提示中）（Brown 等人，2020年；Dong 等人，2024年），我们的方法利用稠密检索，根据语义相似度为每条输入需求动态选择示例。给定一条自然语言需求作为输入，我们使用句子转换器将其编码为嵌入。随后，我们利用该嵌入搜索预先构建的自然语言-LTL 配对索引，其中每条自然语言需求都以与输入需求相同的方式进行了编码。我们根据与查询的余弦相似度，检索最相似的前 k 个示例，然后将这些示例添加到提示中，在大语言模型生成线性时序逻辑公式之前向其提供相关示例。

为构建自然语言-LTL 配对检索数据集，我们从Dwyer 等人 (1998年)中选取了一些时序属性模式。该文献汇集了常见的时序属性模式及其 LTL 形式化表示。我们仅选取对应的线性时序逻辑公式较为简洁的模式，以便公式易于理解。随后，我们从形式化验证文献中提取自然语言-LTL 配对，扩充了数据集。我们将自然语言需求及其对应线性时序逻辑公式中的所有原子命题抽象为通用占位符（原子 1、原子 2 等），使检索机制能够依据时序结构而非具体变量名进行匹配。为应对自然语言固有的歧义并提升鲁棒性，我们在数据集中加入了改述，确保无论需求的具体表述如何，我们的方法都能检索到相关示例。

3.3.3. 基于语法的引导

将基于文本的需求形式化为正确的线性时序逻辑公式时，使用大语言模型面临的一大挑战是确保输出符合线性时序逻辑输入语法、在语法上有效。为此，我们采用以下策略：(i) 在系统提示中加入语法并提供清晰的指示；(ii) 使用基于语法的解码工具；以及 (iii) 将 (i) 和 (ii) 结合起来。

我们使用基于语法的解码工具 SynCode（Ugare 等人（2025年）），这是一种先进方法。它基于给定的输入语法构建确定性有限自动机（DFA）查找机制，并采用掩码解码方法，引导大语言模型选择符合输入语法的下一个词元。具体做法是预先计算 DFA 掩码存储，这是一种从语法的 DFA 派生的查找表，用于将语法

状态映射到有效词元。这样，SynCode 就能快速过滤无效词元，从而实现高效的语法约束解码。尽管该方法在特定条件下具有可靠性和完备性，但并不总能保证结果完整且正确。

3.3.4. 使用语法检查与解释进行一致性检查

S

一致性检查器用于确定一个线性时序逻辑公式是否可满足，也就是说，是否至少存在一条执行轨迹满足该公式的所有时序约束。检查器通常会返回两种结果之一：SAT，表示公式一致，并附上该公式的一个模型；或 UNSAT，表示公式不一致，并附上一个不可满足核。能够证明线性时序逻辑公式可满足的模型是一条无限执行轨迹，公式在整条轨迹上始终成立。不可满足核是原公式的一个最小（或更小）子集，该子集本身不一致，因此可以解释为何完整公式无法满足。模型和不可满足核都能提供有价值的信息，向用户解释分析结果。这些信息还可用作反馈，以改进翻译过程。我们使用 BLACK Geatti 等人(2021) 作为线性时序逻辑的可满足性检查器。

4. 评估与实验结果

4.1. 研究问题

（研究问题 1）翻译准确率。LTLGUARD 将自然语言需求翻译为线性时序逻辑的准确率如何？各组件对此有何贡献？
理由：本研究问题旨在考察 LTLGUARD 将非正式需求与相应形式化规范衔接起来的能力，并量化 LTLGUARD 中各模块对翻译后 LTL 规范整体准确率的单独影响。指标：我们通过测量由带标签的自然语言需求—线性时序逻辑公式组成的基准测试集上的语法准确率和语义准确率，评估 LTLGUARD 的准确率。我们通过消融研究衡量各模块的贡献，即系统地禁用各个组件，并将规格准确率的变化与完整系统基线进行比较。

（研究问题 2）鲁棒性。LTLGUARD 对自然语言需求中的语言变化有多强的鲁棒性？
理由：评估鲁棒性有助于确定当自然语言需求存在语言变异性时，LTLGUARD 是否仍能保持稳定的性能。指标：通过比较同一需求的多个释义变体所生成的 LTL 规范的准确率来衡量鲁棒性。

（研究问题 3）一致性检查。 LTLGUARD 能在多大程度上检测自然语言需求中的逻辑不一致？

理由：该研究问题旨在考察 LTLGUARD 识别逻辑冲突的有效性；这些冲突可能源于需求本身不一致，也可能源于形式化过程中的错误。指标：我们通过一个小型基准测试来衡量 LTLGUARD 检测冲突的能力，该基准包含一致的需求集和故意设置为不一致的需求集。

4.2. 实验设置

实现。 LTLGUARD 使用 Python 3.10.18 实现，采用 SynCode 0.4.16、LangChain 1.0.7、Faiss-cpu 1.13.1 以及最新版本[1] 的 LTL 求解器 BLACK。为便于开展

1. 截至2026年2月。

消融研究并回答（研究问题 2），我们实现了框架的 7 个变体。表 1 展示了各变体中启用和禁用组件的组合。所有实验均在一台高性能计算集群上运行，该集群包含 3 个节点、64 个 CPU（Intel(R) Xeon(R) Silver，每个 CPU 有 16 个核心）和 6 个 GPU（2 块 NVIDIA TITAN RTX、4 块 NVIDIA GeForce RTX 2080）。

数据集。我们在实验中使用三个数据集：（i）消融研究（RQ1）使用一组精心整理的 70 个自然语言-LTL 配对，代表常见需求及其语言变体：它主要涵盖安全性规范与响应式规范，包括带有 **X**-嵌套的有界响应、互斥约束、多个性质的合取，以及少量额外算子（例如，**U**）；多个不同的自然语言释义映射到同一底层公式，使我们能够探究对措辞和命题命名的敏感性。（ii）为了与先前工作进行比较，我们使用 36 个实例的“困难”基准，其来自 `nl2spec` Cosler等人 （2023年）。（iii）最后，RAFSL 依赖于一个由提升的自然语言-线性时序逻辑示例对构成的知识库（第 3.3.2 节），并以两个版本使用：完整集（137 对）和去除重叠集（122 对），以量化与 `nl2spec` 基准的示例重叠效应。鲁棒性（研究问题 2）和一致性检查（研究问题 3）评估仅使用各自章节中引入的小型、明确列出的需求集。

4.3. 评估结果

4.3.1. (RQ1) 翻译准确率

翻译准确率与消融研究。 首个实验旨在评估我们的框架在提升以下方面的有效性：(i) 生成的线性时序逻辑公式的语法有效性；(ii) 相对于预期规范的语义正确性。我们针对 HuggingFace 上提供的四种紧凑型开放权重大语言模型开展消融研究：Qwen2.5-14B-InstructQwen团队 (2026a)、Mistral-Nemo-Instruct-12B Mistral AI (2026b)、Mistral-7B-InstructMistral AI (2026a) 和 Phi-3- Mini-4B-Instruct 微软(2026)。为 RAFSL 组件生成所需的句子嵌入时，我们使用 Qwen3-Embedding-0.6B Qwen团队 (2026b)。对于每种模型，我们评估了七种变体，这些变体逐步启用流水线中的各个组件。表 1 汇总了评估结果。

LTLGUARD 变体	**组件**				**QWEN2.5-14B**		**Mistral-Nemo-12B**		**Mistral-7B**		**Phi3-mini-4B**	
	G	**S**	**R**	**F**	**合成。**	**Sem.**	**Syn.**	**Sem.**	**合成。**	**语义。**	**合成。**	**Sem.**
V1：基础版（仅大语言模型 + 免G提示）	×	×	×	×	95.7	68.5	51.4	31.4	10.0	7.1	47.1	24.2
V2：提示语法	✓	×	×	×	57.1	34.2	17.1	5.7	5.7	1.4	35.7	11.4
V3：语法 +SynCode	✓	✓	×	×	97.1	50.0	85.7	27.1	15.7	5.7	55.7	15.7
V4：语法 +SynCode + RAFSL	✓	✓	✓	×	95.7	75.7	**92.8**	**67.1**	87.1	38.5	90.0	60.0
V5：语法 +SynCode +反馈	✓	✓	×	✓	**98.5**	52.8	85.7	28.5	42.8	7.1	65.7	13.0
V6：SynCode + RAFSL +反馈	×	✓	✓	✓	97.1	**78.6**	**92.8**	64.2	78.5	**40.0**	91.4	**64.2**
V7：完整系统	✓	✓	✓	✓	**98.5**	74.2	**92.8**	57.1	**92.8**	38.5	**92.8**	35.7

表 1：框架变体和消融结果。**G**：提示语法，**S**：SynCode 严格解码，**R**：RAFSL，**F**：解析器反馈（✓启用，× 禁用）。**Syn.**=% 语法有效的线性时序逻辑；**语义**。=% 在语义上与标签一致（接受逻辑等价的公式）。

我们观察到，启用流水线中的轻量级组件（不进行微调、不使用完整的检索增强生成，且仅使用有限数量的示例），能够显著提升所有受评估模型的输出质量。尤其是，先验线性时序逻辑知识较弱的模型提升最大。对于 Mistral-7B，语法有效性从 10.0%（V1）提升至 92.8%（V7），语义正确性（考虑等价性）从 7.1%（V1）提升至 40.0%（V6）/38.5%（V7）。类似地，Phi3-mini-4B 的语法有效性从 47.1% 提升至 92.8%，语义正确性则从 24.2% 提升至 64.2%

启用 V6 后，语义正确性达到 64.2%（该模型的最佳语义得分）。对于 Mistral-Nemo-12B，表现最佳的配置达到 92.8% 的语法有效性和 67.1% 的语义正确性（V4），而原始设置下这两项指标分别为 51.4% 和 31.4%。对于原始设置下表现已经很强的 Qwen2.5-14B，流水线仍能持续提升语义正确性（V6 中为 68.5% $\rightarrow$ 78.6%），同时保持近乎完美的语法有效性。总体而言，结合 SynCode 与相关示例检索的变体（V4、V6、V7）往往能带来最显著的语义提升，这表明当基础模型对时序逻辑的潜在知识有限时，提供有针对性的上下文尤其有益。

句子	预期	实际
系统最终应达到安全状态。	$\mathbf{G}\,\mathbf{F}\ safe$	$\mathbf{F}\ safe$
p 和 q 不应同时为真。	全局 $\neg(p \wedge q)$	$\neg(p \wedge q)$
在 p之后，始终 q。	$\mathbf{G}(p \rightarrow \mathbf{G}\ q)$	$\mathbf{G}(p \rightarrow \mathbf{X}\,\mathbf{G}\,q)$

表2：生成公式与预期公式不同的歧义需求。

表1中的语义正确性采用等价性感知标准进行计算：当生成公式与预期公式在逻辑上等价时，即计为正确。不过，自然语言歧义仍是一个实际挑战：在不同解读下，一项需求可能对应多个彼此不等价但都合理的形式化表示，如表2中我们实验的示例所示。

nl2spec “困难”基准。 我们在 Cosler等人 WEN2.5- 14B）提出的“困难”基准上评估了表现最佳的设置（V6 与 QCosler等人(2023年），他们报告了多种 nl2spec 设置的语义准确率（第 4.2节，表1），其中包括采用不同提示策略进行的初始（非交互式）翻译，以及通过用户引导的修正进行的交互式设置。他们的评估只有在结果符合专家的预期含义时，才将该实例计为正确；换言之，对于歧义自然语言输入，他们采用不容忍歧义的标准，不接受其他合理的解读。

数据集不规则性。 我们在困难数据集中发现一条需求，其英文难以理解（“a must always hold, but if is exceeds, it allow two timestamps to recover”），即使参考公式也无法推断其含义；因此，LTLGUARD的任何配置都无法复现预期标签。我们报告完整数据集上的结果，同时指出，对于任何翻译方法而言，这个实例很可能都构成不可避免的错误。

RAFSL 重叠与评估策略的作用。 我们发现，RAFSL数据集与nl2spec基准存在重叠：137个示例中有15个（11%）在模板结构上与36个困难实例中的15个（42%）相匹配（原子名称除外）。由于需求表述通常较为标准，且模式反复出现，这种重叠是意料之中的。为评估其影响，我们进行了两项实验：实验1（有重叠）使用完整的RAFSL数据集；实验2（无重叠）则在检索前移除重叠示例。我们采用两种策略评估语义正确性：S1（不容忍歧义）仅将与基准中的单一专家解读相符的预测计为正确（Cosler等人，2023年）；S2（兼容歧义）还接受合理的替代形式化表示。表3 报告了准确率，并将LTLGUARD与Cosler等人在同一基准上报告的系统进行比较。

解读。出现了几项趋势。首先，LTLGUARD 使用紧凑型模型，在这一颇具挑战性的基准测试中取得了出色表现；而 Cosler 等人（2023年）报告的 nl2spec 最高准确率，则是借助规模大得多的专有大语言模型（Codex 和 GPT-3.5）取得的，最佳情况下还使用了交互式用户调试。具体而言，在 RAFSL 重叠设置（实验 1）中，LTLGUARD 在不容忍歧义（S1）策略下的语义准确率至少达到 75.0%

设置（36 个困难实例）	后端	语法 (%)	语义 (%) S1	S2
NL2LTL Fuggitti 和 Chakraborti (2023年)	Rasa	–	2.7	–
T5 微调Hahn 等人(2022年)	T5	–	5.5	–
nl2spec初始Cosler等人(2023年)	Bloom	–	13.8	–
nl2spec初始Cosler等人(2023年)	Codex	–	44.4	–
nl2spec初始 +ID 示例 Cosler等人(2023年)	Codex	–	58.3	–
nl2spec交互式Cosler等人(2023年)	Codex	–	86.1	–
nl2spec初始Cosler等人(2023年)	GPT-3.5	–	33.3	–
nl2spec初始 +ID 示例 Cosler等人(2023年)	GPT-3.5	–	47.2	–
nl2spec交互式Cosler等人(2023年)	GPT-3.5	–	58.3	–
LTLGUARD：V6 + RAFSL 重叠（实验 1）	Qwen2.5-14B	**100.0**	**75.0**	**77.8**
LTLGUARD：V6 + RAFSL 无重叠（实验 2）	Qwen2.5-14B	**97.2**	**50.0**	**63.9**

表 3：在包含 36 个实例的 nl2spec “困难”基准测试上的结果。**语法**=% 的线性时序逻辑在语法上有效；在 **S1**（不容忍歧义；与专家对齐）和**S2**（兼容歧义）条件下，**语义**=% 的语义准确率。对于 nl2spec、NL2LTL 和 T5 行，我们复现了 Cosler等人的表 1 中的 S1 准确率；其中未报告语法和 S2，因而显示为“–”。

在兼容歧义策略（S2）下也达到至少 75.0%，同时保持完美的语法有效性，优于所有非交互式 nl2spec 变体，并接近表现最佳的交互式 nl2spec（使用 Codex，准确率为 86.1%）。这表明，即使不进行微调，精心设计提示结构、保证语法正确性并采用 RAFSP，也能大幅缩小与更大型大语言模型之间的差距。

其次，与其他基线的比较进一步凸显了 LTLGUARD 的有效性。NL2LTL（模式驱动翻译）Fuggitti 和 Chakraborti（2023年）在同一困难数据集上的准确率为 2.7%，而微调后的 T5 方法 Hahn 等人（2022年）（T5-base，2.2 亿参数）达到 5.5%。使用开放权重但规模非常大的模型（Bloom-176B）时，nl2spec 的准确率也仅为 13.8%。LTLGUARD 使用规模小得多的开放权重模型（Qwen2.5-14B），根据重叠情况和歧义处理方式，其语义准确率达到 50.0% 至 77.8%，这表明我们的工具链能够使中等规模模型在具有挑战性的自然语言到LTL的翻译任务中具备竞争力。

第三，移除重叠部分（实验 2）会使 S1语义准确率降至 50.0%，而句法准确率仍然很高（97.2%）。这表明，困难基准包含一些 RAFSL示例至关重要的模式；也就是说，仅靠提示和受约束解码，无法可靠地还原专家的预期解读。实验 2 中的 S1–S2差距（50.0% 对 63.9%）表明，在基准的不容忍歧义的评分下，许多“错误”反映的是合理的替代解读，而不是真正的翻译失败。因此，如果没有明确的歧义处理（例如提供多个候选形式化表示或标准公式），严格的专家对齐指标可能会低估其实际效用。

RAFSL 对歧义的示例说明。对于“每当 a 和 b 不成立时，c 最终成立。”，基准标签为 $\mathbf{G}(\neg(a \wedge b) \rightarrow \mathbf{F}\, c)$。我们认为该句存在歧义：复数形式“do not hold”可能表示 $\neg a \wedge \neg b$，但在实践中有时会被解读为 $\neg(a \wedge b)$。有重叠时，RAFSL 会检索到提升后的近重复对 ⟨“每当原子命题 1 和原子命题 2 不成立时，原子命题 3 最终成立。”、$\mathbf{G}(\neg(atom\ 1 \wedge atom\ 2) \rightarrow \mathbf{F}\, atom\bar{}\ 3)$⟩，而 LTL-GUARD 会匹配该标签；没有重叠时，它则会生成 $\mathbf{G}((\neg a \wedge \bar{}\neg b) \rightarrow \mathbf{F}\, c)$，$\bar{}$与另一种解读一致。此外，改变检索到的提升示例的存储标签，会系统性地引导生成（例如，对于“严格未来”的解读，生成 $\mathbf{G}(\neg(a \wedge b) \rightarrow \mathbf{X}\,\mathbf{F}\, c)$，或者生成

$\mathbf{G}(\neg(a \wedge b) \rightarrow (\neg(a \wedge b)\ \mathbf{U}\ c))$，要求在 $(a \wedge b)$ 再次成立之前满足 c），这说明 RAFSL 如何编码并传递特定领域的解读选择。

（RQ1）： LTLGUARD 将强大的语法保证与模块化检索机制相结合，从而提升了其在困难基准测试集上的语义准确率。与已存储示例的重叠程度会显著影响结果。对于现实需求的形式化，歧义感知生成至关重要。

4.3.2.（研究问题 2）鲁棒性评估

我们采用两种鲁棒性概念来评估自然语言到LTL的翻译的鲁棒性：（1）原子重命名鲁棒性（ARR）和（2）改述鲁棒性（RR）。ARR 的核心思想是：当英文短语的结构保持不变，只有映射到原子命题的词语发生变化时，所得的线性时序逻辑公式除原子命题重命名外应完全相同。例如，表4中 ARR 列下的所有需求，都应翻译为相同的线性时序逻辑公式模板。理想情况下，它们都应翻译为正确的请求-响应属性 $\mathbf{G}(p \rightarrow \mathbf{F}q)$，其中 p 和 q 会随每项需求而变化。但请注意，鲁棒性与正确性彼此正交。因此，即使某种翻译将所有这些需求都转换成形如 $\mathbf{GF}(p \wedge q)$ 的线性时序逻辑公式，虽然不正确，我们仍会认为它具有 ARR 鲁棒性。

ARR	**RR 组**	
1. 每个请求最终都会得到批准。 2. 每条消息最终都会送达。 3. 每次按钮按下最终都会得到处理。	G1	a. 每个请求最终都会得到批准。 b. 没有请求会一直未获批准。 c. 每个请求最终都会获批。
4. 每个订单最终都会发货。 5. 每个错误最终都会被记录。 6. 每项任务最终都会完成。	G2	d. 每条消息最终都会送达。 e. 没有任何消息会未送达。 f. 每条消息最终都会送达。

表4：鲁棒性评估数据集。

RR 的核心思想是：如果在不改变含义的情况下改写一条自然语言需求，那么所得线性时序逻辑公式也不应改变。例如，表4 中 G2组的 RR列下的所有需求都应翻译成同一个线性时序逻辑公式。理想情况下，该线性时序逻辑公式应是正确的翻译，$\mathbf{G}$(消息 $\rightarrow \mathbf{F}$已送达)。不过，我们仍将鲁棒性与正确性区分开来，并将不正确但一致的翻译视为 RR 鲁棒。

配置 (QWEN2.5-14B)	ARR	RR 第1组	RR 第2组
V1	3/6	**3/3**	2/3 †
V6	**6/6**	2/3	**2/3**
V7	4/6	**3/3**	1/3

表5：鲁棒性评估（†：该组的翻译稳健但不正确）。

我们评估了 LTLGUARD 的 V1、V6 和 V7 变体，使用 QWEN2.5-14B 评估 ARR 和 RR。对于 ARR，我们使用了表4 （ARR）中的需求。对于 RR，我们使用了表4 （RR，包含 G1 和 G2组）中的两组需求。结果汇总于表5。在 ARR 鲁棒性方面，V6 和 V7 相比 V1 均有所提升，其中 V6 达到了完美鲁棒性。

有趣的是，V1 将 6 条需求中的 3 条正确翻译为 $\mathbf{G}(p \rightarrow \mathbf{F}q)$ 形式，而将剩余 3 条需求翻译为同一种错误形式 $\mathbf{GF}q$，其中 q 是交付、发货或完- 成。V7 将 6 条需求中的 4 条（第 1、3、4 和 5 条）正确翻译为 $\mathbf{G}(p \rightarrow \mathbf{F}q)$ 形式，并将剩余 3 条需求翻译为同一种错误形式$\mathbf{GF}q$。V6 将全部 6 条需求正确翻译为 $\mathbf{G}(p \rightarrow \mathbf{F}q)$ 形式。

对于 RR 鲁棒性，原始变体 V1 在两组需求上都已经表现出较强的 RR 鲁棒性，而 V6 和 V7 都未能进一步提升这一表现。具体而言，V1 将 G1 的全部 3 条需求正确翻译为 **G(**请求 $\rightarrow$ **F**授予**)** 公式；V7 也是如此，但它将需求 b 翻译为 **G(**请求 $\rightarrow$ **F**已授予**)**（我们认为这是正确的）。V6 将 G1 的需求 a 和 c 翻译为正确公式，但将需求 b 翻译为错误公式**G(**请求 $\rightarrow$ **FG**已授予**)**。对于 G2，V1 将需求 d 和 f 翻译为同一种错误公式 **G(F**已交付**)**，并将需求 e 翻译为错误公式 **G(X**d $\rightarrow$ **F**d**)**。因此，V1 在 G2 上取得了 2/3 的 RR 鲁棒性，尽管其针对该组的所有翻译都是错误的（因此标记为 †）。V6 将需求 d 和 f 翻译为正确公式 **G(**消息 $\rightarrow$ **F**已交付**)**，并将需求 e 翻译为错误公式 **G(**已交付**)**。V7 将需求 f 翻译为正确公式，并分别将需求 d 和 e 翻译为错误公式 **G(F** 已交付**)** 和 **G(**已交付**)**。

（研究问题 2）： LTLGUARD（尤其是 V6）对需求的语言变体表现出整体稳健性，但要达到工业级应用水平，仍需进一步改进。

4.3.3.（研究问题 3）一致性检查

我们评估了一致性检查在检测需求中潜在不一致方面的效果，这些不一致可能由以下原因造成：(1) 自然语言需求中的冲突，或 (2) 翻译错误。我们使用 V6 和 V7 变体的 LTLGUARD 处理输入需求集 $R1$、$R2$ 和 $R3$，如表6（左侧）所示。

需求编号	需求
R1	每个请求都必须得到批准
R2	请求将不会得到批准
R3	将会提出某个请求

需求编号	要求
R2a	请求未获批准
R2b	不得批准任何请求
R2c	请求从未获准
R2d	请求不得获准
R2e	请求将永远不会获准

表 6：需求集合：（左）原始集合；（右）需求 R2 的变体

$R1$, $R2$ 和 $R3$ 被（联合）翻译为公式1中的线性时序逻辑公式。BLACK 判定该公式不可满足，从而检测出 $R1$, $R2$ 和 $R3$之间存在冲突。[2] 有趣的是，将 $R2$ 改为表 6（右侧）中看似等价的不同变体，会产生不同结果。将 $R1$ 和 $R3$ 与 $R2a$ 组合，会得到正确的线性时序逻辑公式（公式1）。将 $R1$和 $R3$与 $R2b$ 或 $R2d$ 组合，翻译器会生成公式2所示的线性时序逻辑公式，该公式同样不可满足。但此时的不一致源于 G(¬request) 与 F(request) 之间的冲突，这是因为 $R2b$ 或 $R2d$ 被翻译成了 G(¬request)，而不是 G(¬granted)。这表明，可满足性检查不仅有助于检测原始自然语言需求中的不一致，也有助于检测翻译错误。在 V6 下，将 $R1$ 和 $R3$ 与 $R2c$ 或 $R2e$ 组合，会为每项需求生成正确的独立线性时序逻辑公式；然而，这些公式被错误地以析取而非合取连接。相比之下，V7 会生成正确的联合公式（采用合取）。

$$\mathbf{G}(request \rightarrow \mathbf{F}\, granted) \ \wedge\ \mathbf{G}(\neg\, granted) \ \wedge\ \mathbf{F}(request) \tag{1}$$

$$\mathbf{G}(request \rightarrow \mathbf{F}\, granted) \ \wedge\ \mathbf{G}(\neg\, request) \ \wedge\ \mathbf{F}(request) \tag{2}$$

2. $R2$ 也可以译为 **G(**请求 $\rightarrow$ **G(**¬获准**))**。两种翻译都会导致不一致性。

（研究问题 3）：一致性检查在检测需求中的冲突方面发挥着关键作用，同时也能检测需求形式化中的错误。

5. 结论与未来工作

我们介绍了 LTLGUARD，这是一个使用紧凑型语言模型将非正式需求翻译为线性时序逻辑（LTL）规范的框架。我们展示了，将语言模型的生成能力与为 LTL 提供句法和语义指导的轻量级符号技术相结合后，该框架具备较强的鲁棒性和翻译准确率，其表现可与基于大型旗舰模型的方法相媲美。尽管如此，我们的结果表明，该方法在鲁棒性和整体准确率方面仍有很大的提升空间。

未来，我们计划采用更具互动性的方式，并对结果提供更准确的解释。我们将研究解决歧义需求翻译问题的方法。最后，我们将考虑表达能力更强的形式体系，包括带过去时算子的 LTL 和信号时序逻辑（STL）。最后，我们还将研究如何结合多种模态（例如，需求不仅由自然语言构成，也包含满足或违反这些需求的轨迹示例）。

致谢。本研究得到了以下资助：(i) 维也纳科学与技术基金（WWTF）资助，项目编号为 10.47379/ICT23-030；(ii) 欧盟资助。本文所表达的观点和意见仅代表作者本人，并不一定反映欧盟或欧洲卫生与数字执行局（HADEA）的观点。欧盟和资助机构均不对此承担责任。RobustifAI项目，编号 101212818；(iii) 美国国家科学基金会（NSF）形式化方法实地研究（FMitF）计划资助，项目编号为 2319500 和 2525087；以及 (iv) 欧盟 CYBERGUARD项目，项目编号为 101190281。

参考文献

白泽辰、王丕超、肖天俊、何通、韩宗博、张正和 Mike Zheng Shou。多模态大语言模型的幻觉：综述。*CoRR*，abs/2404.18930，2024年。doi：10.48550/ARXIV.2404.18930。网址：https://doi.org/10.48550/arXiv.2404.18930。

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, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever 和 Dario Amodei。《语言模型是少样本学习者》。*CoRR*，abs/2005.14165，2020年。网址：https://arxiv.org/abs/2005.14165。

Igor Buzhinsky。自然语言需求到时序逻辑的形式化：一项综述。载于*2019*年 *IEEE* 第*17*届工业信息学国际会议（*INDIN*），第1卷，第400–406页。IEEE，2019年。

Itay Cohen、Klaus Havelund、Doron Peled 和 Yoav Goldberg。《重新构建的力量：使用大语言模型合成运行时验证监视器》。载于 Bettina K¨onighofer 和 Hazem Torfah（编），运行时验证——第*25*届国际会议，*RV 2025*，奥地利格拉茨，*2025*年*9*月*15*日至*19*日，论文集，第16087卷，计算机科学讲义，第202–

212页。Springer，2025年。doi：10.1007/978-3-032-05435-7\ 12。网址 https://doi.org/10.1007/978-3-032-05435-7_12。

Matthias Cosler、Christopher Hahn、Daniel Mendoza、Frederik Schmitt 和 Caroline Trippel。《nl2spec：利用大语言模型交互式地将非结构化自然语言转换为时序逻辑》。载于 Constantin Enea 和 Akash Lal 编，计算机辅助验证——第 *35* 届国际会议 *CAV 2023*，巴黎，法国，*2023*年*7*月*17*日至*22*日，论文集第二部分，第 13965 卷，计算机科学讲义，第 383–396 页。Springer，2023年。doi：10.1007/ 978-3-031-37703-7\ 18。网址：https://doi.org/10.1007/978-3-031-37703-7_18。

Jennifer A. Davis、Matthew Clark、Darren Cofer、Aaron Fifarek、Jacob Hinchman、Jonathan Hoff- man、Brian Hulbert、Steven P. Miller 和 Lucas Wagner。《形式化方法在工业领域应用的障碍研究》。载于 Charles Pecheur 和 Michael Dierkes 编，工业关键系统的形式化方法，第 63–77 页，柏林、海德堡，2013年。Springer 柏林海德堡。国际标准书号 978-3-642-41010-9。

Qingxiu Dong、Lei Li、Damai Dai、Ce Zheng、Jingyuan Ma、Rui Li、Heming Xia、Jingjing Xu、Zhiyong Wu、Baobao Chang、Xu Sun、Lei Li 和 Zhifang Sui。《上下文学习综述》。载于 Yaser Al-Onaizan、Mohit Bansal 和 Yun-Nung Chen 编，《*2024*年自然语言处理实证方法会议论文集》，第 1107–1128 页，迈阿密，佛罗里达，美国，2024年11月。计算语言学协会。doi：10.18653/v1/2024.emnlp-main.64。网址：https://aclanthology.org/2024.emnlp-main.64/。

Matthew B Dwyer、George S Avrunin 和 James C Corbett。《有限状态验证的属性规约模式》。载于《软件实践中的形式化方法》第二届研讨会论文集，第 7–15 页，1998年。

Francesco Fuggitti 和 Tathagata Chakraborti。NL2LTL——一个用于将自然语言（NL）指令转换为线性时序逻辑（LTL）公式的 Python 软件包。载于 Brian Williams、Yiling Chen 和 Jennifer Neville 编，第三十七届 *AAAI* 人工智能大会（*AAAI 2023*）、第三十五届人工智能创新应用大会（*IAAI 2023*）、第十三届人工智能教育进展研讨会（*EAAI 2023*），美国华盛顿特区，*2023*年*2*月*7*日至*14*日，第 16428–16430 页。AAAI出版社，2023年。doi：10.1609/AAAI.V37I13.27068。网址https://doi.org/10.1609/aaai.v37i13.27068。

Luca Geatti、Nicola Gigante 和 Angelo Montanari。BLACK：一款快速、灵活且可靠的线性时序逻辑可满足性检查器。载于 Dario Della Monica、Gian Luca Pozzato 和 Enrico Scala 编，第三届人工智能与形式化验证、逻辑、自动机及综合研讨会论文集（由第十二届博弈、自动机、逻辑与形式化验证国际研讨会 *GandALF 2021* 主办），意大利帕多瓦，*2021*年*9*月*22*日，第 2987 卷，收录于*CEUR*研讨会论文集，第 7–12 页。CEUR-WS.org，2021年。网址https://ceur-ws.org/Vol-2987/paper2.pdf。

Dimitra Giannakopoulou、Thomas Pressburger、Anastasia Mavridou 与 Johann Schumann。结构化自然语言需求的自动化形式化。《信息与软件技术》，137：106590，2021年。

Christopher Hahn、Frederik Schmitt、Julia J. Tillman、Niklas Metzger、Julian Siber 与 Bernd Finkbeiner。基于自然语言的形式化规格。*CoRR*，abs/2206.01962，2022年。doi：10.48550/ARXIV.2206.01962。URLhttps://doi.org/10.48550/arXiv.2206.01962。

Jie He、Ezio Bartocci、Dejan Nickovic、Haris Isakovic 和 Radu Grosu。Deepstl——从英语需求到信号时序逻辑。载于 第 *44* 届 *IEEE/ACM* 国际软件工程会议（*ICSE 2022*），美国宾夕法尼亚州匹兹堡，*2022*年*5*月*25*日至*27*日，第 610–622 页。ACM，2022年。doi：10.1145/3510003.3510171。网址：https://doi.org/10.1145/3510003.3510171。

Philipp Kogler、Andreas Falkner 和 Simon Sperl。使用大语言模型可靠地生成形式化规格。载于 *SE 2024-Companion*，第 141–153 页。德国信息学会，2024年。

Sascha Konrad 和 Betty HC Cheng。《面向UML模型的自然语言属性自动化分析》。载于国际模型驱动工程语言与系统会议，第48–57页。Springer，2005年。

Zhi Ma、Cheng Wen、Zhexin Su、Xiao Liang、Cong Tian、ShengchaoQin 和 MengfeiYang。《弥合自然语言与形式化规范之间的鸿沟——通过大语言模型进行层次语义分解，将软件需求自动翻译为线性时序逻辑》。*arXiv*预印本 *arXiv:2512.17334*，2025年。

微软。Phi-3-mini-4k-instruct：<HuggingFace>模型卡。
https://huggingface.co/microsoft/Phi-3-mini-4k-instruct，2026年2月。访问日期：2026-02。

Mistral AI。Mistral-7B-Instruct-v0.1：<HuggingFace>模型卡。
https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.1，2026年2月。访问日期：2026-02。

Mistral AI. Mistral-Nemo-Instruct-2407：Hugging Face模型卡。
https://huggingface.co/mistralai/Mistral-Nemo-Instruct-2407，2026年2月。访问时间：2026年2月。

Konstantinos Mokos、Theodoros Nestoridis、Panagiotis Katsaros 和 Nick Bassiliades。自然语言系统需求的语义建模与分析。*IEEEAccess*，10:84094–84119，2022年。doi：10.1109/ACCESS.2022.3197281。URLhttps://doi.org/10.1109/ACCESS.2022.3197281。

Theodoros Nestoridis、Konstantinos Mokos 和 Panagiotis Katsaros。从自然语言需求规范到逻辑性质。载于 Panagiotis Katsaros（编），基于模型的安全与评估，第241–256页，沙姆，2026年。施普林格·自然瑞士。

Amir Pnueli. 程序时序逻辑。载于第*18*届计算机科学基础年度研讨会，普罗维登斯，罗得岛州，美国，*1977*年*10*月*31*日至*11*月*1*日，第46–57页。IEEE计算机学会，1977年。doi: 10.1109/SFCS.1977.32。URL https://doi.org/10.1109/SFCS.1977.32。

Qwen团队。Qwen2.5-14B-Instruct-1M：Hugging Face模型卡。
https://huggingface.co/Qwen/Qwen2.5-14B-Instruct-1M，2026年2月。访问日期：2026-02。

Qwen团队。Qwen3-Embedding-0.6B：Hugging Face模型卡。
https://huggingface.co/Qwen/Qwen3-Embedding-0.6B，2026年2月。访问日期：2026-02。

Eshita Shukla、Quinn Thibeault 与 Giulia Pedrielli。一种用于大语言模型引导的自然语言到时序逻辑自动翻译的灰盒方法。载于第*16*届 *ACM/IEEE* 网络物理系统国际会议论文集，*ICCPS 2025*（与 *CPS-IoT Week 2025* 同期），美国加利福尼亚州欧文，*2025*年*5*月*6*日至*9*日，第33:1–33:2页。ACM，2025年。doi: 10.1145/ 3716550.3725163。URLhttps://doi.org/10.1145/3716550.3725163。

Shubham Ugare、Tarun Suresh、Hangoo Kang、Sasa Misailovic 与 Gagandeep Singh。SynCode：基于语法增强的大语言模型生成。《机器学习研究汇刊》，2025年，2025年。

Saad Obaid Ul Islam、Anne Lauscher 和 Goran Glava˘s。大语言模型在不同语言中的幻觉程度如何？关于大语言模型幻觉的真实多语言估算。载于 Christos Christodoulopoulos、Tanmoy Chakraborty、Carolyn Rose 和 Violet Peng（编），*2025*年自然语言处理实证方法会议论文集，第 29077–29098 页，中国苏州，2025年11月。计算语言学协会。国际标准书号 979- 8-89176-332-6。doi：10.18653/v1/2025.emnlp-main.1481。网址：https://aclanthology.org/2025.emnlp-main.1481/。

Yilongfei Xu、Jincao Feng 和 Weikai Miao。从失败中学习：使用大语言模型将自然语言需求翻译为线性时序逻辑。载于 第*24*届 *IEEE* 国际软件质量、可靠性与安全会议，*QRS 2024*，剑桥，英国，*2024*年*7*月*1*日至*5*日，第 204–215 页。IEEE，2024年。doi：10.1109/QRS62785.2024.00029。网址 https://doi.org/10.1109/QRS62785.2024.00029。

Rongjie Yan、Chih-Hong Cheng 和 Yesheng Chai。自然语言规范的形式一致性检查。载于 *2015* 年欧洲设计、自动化与测试会议及展览（*DATE*），第 1677–1682 页。IEEE，2015 年。