arXiv:2303.04864v1 [cs.LO] 8 Mar 2023

nl2spec：使用大语言模型将非结构化自然语言交互式翻译为时序逻辑

Matthias Cosler[2], Christopher Hahn[1], Daniel Mendoza[1], Frederik Schmitt[2],和 Caroline Trippel[1]

[1] 斯坦福大学，美国加利福尼亚州斯坦福市 hahn@cs.stanford.edu、dmendo@stanford.edu、trippel@stanford.edu[2] CISPA 赫尔姆霍兹信息安全中心，德国萨尔布吕肯¨ matthias.cosler@cispa.de、frederik.schmitt@cispa.de

摘要。 在执行任何验证任务时，对所需系统需求进行严谨的形式化至关重要。这往往限制了验证技术的应用，因为编写形式化规范是一项容易出错且耗时的手工任务。为此，我们提出 nl2spec，这是一种应用大语言模型（LLM）从非结构化自然语言中推导形式化规范（以时序逻辑表示）的框架。具体而言，我们提出了一种新的方法，用于检测并解决自然语言系统需求中固有的歧义：利用大语言模型将形式化表示的子公式映射回输入中对应的自然语言片段。用户可以迭代地添加、删除和编辑这些子翻译，以修正错误的形式化表示；相比手动重新起草整个形式化表示，这种方式更加容易。该框架不依赖于特定的应用领域，并且可以扩展到类似的规范语言和新的神经模型。我们开展了一项用户研究，以获得具有挑战性的数据集，并利用该数据集对翻译质量进行实验。我们提供了一个开源实现，其中包括一个基于网页的前端。

1 引言

在执行任何与验证相关的任务时，例如模型检测 [8], 综合 [7],或运行时验证 [21], 对所需系统需求进行严谨的形式化至关重要。然而，编写形式化规范是一项容易出错且耗时的手工工作，通常只有该领域的专家才会承担。本文介绍了 nl2spec 这一框架及其配套的基于网页的工具，旨在简化并自动化形式化规范的编写（包括线性时序逻辑 [35]及类似的时序逻辑）。其核心贡献是一种新方法：利用大语言模型，将自然语言输入分解为子翻译。nl2spec 框架提供了一个交互式界面，用户可以添加、编辑和删除这些子翻译，而不必一次性处理整个形式化过程（类似工作中严重缺失这一功能，例如 [14,30]）。

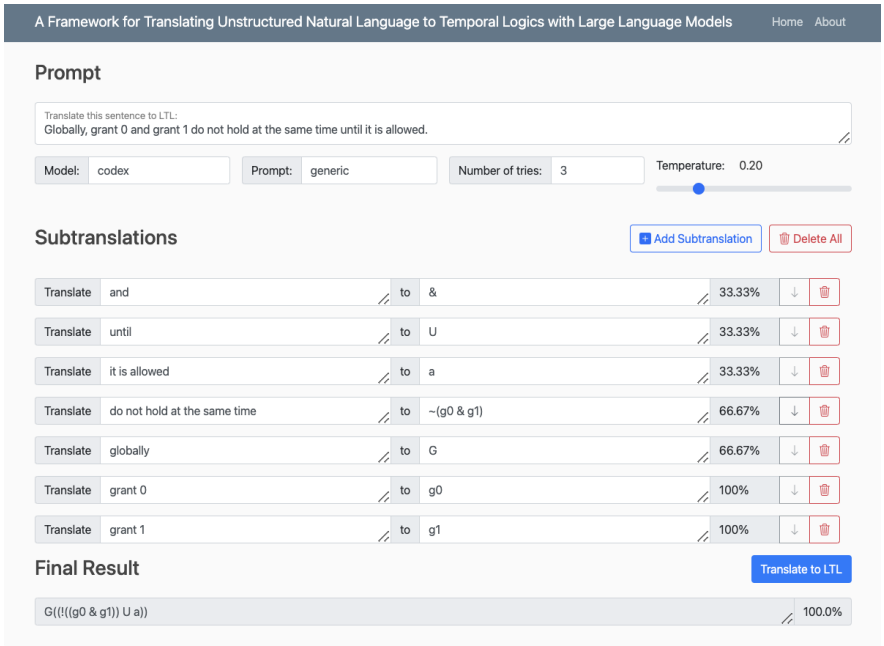

图 1：nl2spec 网页界面的屏幕截图。

图 1 展示了 nl2spec 的基于网页的前端。作为示例，我们考虑以下用自然语言表达的系统需求：“全局范围内，grant 0 和 grant 1 在获准之前不会同时成立。”该工具会自动将这一自然语言规范正确地翻译为线性时序逻辑公式 G((!((g0 & g1)) U a))。此外，该工具还会生成子翻译，例如配对项（“不会同时成立”，!(g0 & g1)），以帮助验证翻译是否正确。

不过，请看下面这个有歧义的例子：“a 一直成立，直到 b 成立，或者 a 始终成立”。要解决运算符优先级方面的歧义，需要人工监督。使用 nl2spec，只需通过显式括号添加或编辑子翻译即可轻松实现（更多详情和示例请参见第 4 节）。为了在基准数据集中捕捉这类（及其他类型的）歧义，我们开展了一项专家用户研究，专门征集将自然语言句子翻译为线性时序逻辑公式的挑战性案例。

nl2spec 设计的关键见解在于，翻译过程可以通过大语言模型自动分解为许多子翻译，而将其分解为子翻译后，用户便可通过交互式修改子翻译，轻松解决自然语言中的歧义和错误译文。nl2spec 的核心目标是尽可能减少人工监督，并提高其效率。为此，所有翻译都附有置信度分数；用户可以通过下拉菜单查看并选择子翻译的备选建议，也可以删除误导性的子翻译

，然后进入下一轮翻译。我们使用通过专家用户研究获得的基准数据集，评估所提出方法的端到端翻译准确率。

该框架不依赖于特定的机器学习模型或应用领域。我们将在第 3 节讨论该工具可能采用的参数化方式和输入。我们将在第 3.2 节更详细地讨论子翻译方法，并介绍一种交互式少样本提示方案，以便大语言模型生成这些子翻译。我们将讨论用户如何将该方案应用于各自的应用领域，以提高该框架的翻译质量。作为概念验证，我们在附录[3]中提供了额外的提示，包括一个用于 STL [31] 的提示。我们将在第 4 节中，利用通过专家用户研究获得的数据集，评估该工具解决错误形式化表示的有效性。我们将在第 5 节讨论该框架的局限性并总结全文。

2 背景与相关工作

2.1 自然 语言到线性时序 逻辑

线性时序逻辑（LTL） [35] 是许多实用规范语言的基础，例如 IEEE 属性规范语言（PSL） [23], 、信号时序逻辑（STL） [31],或 SystemVerilog断言（ SVA） [44]。通过聚焦于原型时序逻辑 LTL，我们使 nl2spec 框架能够扩展到特定应用领域中的规范语言。LTL 用时序算子 U（直到）和 X（下一个）扩展了命题逻辑。还有一些派生算子，例如 Fφ ≡真Uφ 和 G$\varphi \equiv \neg$F$\neg\varphi$。Fφ 表示 φ 将来最终成立，而 Gφ 表示 φ 始终成立。算子可以嵌套：例如，GFφ表示 φ必须无限次发生。LTL 规范描述系统随时间推移的行为及其与环境的交互。例如，给定进程 0、进程 1 和一个共享资源，公式 G($r_0 \rightarrow$ Fg_0) $\wedge$ G($r_1 \rightarrow$ Fg_1) $\wedge$ G$\neg$($g_0 \wedge g_1$)描述了：每当一个进程请求（r_i）访问共享资源时，它最终都会获得授权（g_i）。子公式 G$\neg$($g_0 \wedge g_1$)确保授予的授权彼此互斥。LTL 的形式语法和语义见附录 A。

早期将自然语言转换为时序逻辑的研究主要关注基于语法的方法，这类方法能够处理结构化自然语言 [25,18]。 [5]综述了深度学习兴起之前的早期研究。其他方法包括使用 SMT求解和语义解析的交互式方法 [16], ，以及具身机器人 [46] 和规划 [33]中的结构化时序方面。直到最近，神经网络才开始用于将文本转换为时序逻辑，例如，从头开始训练 STL 模型 [22], 、微调语言模型 [20], ，或采用单样本方式应用 GPT-3 [14,30], 由 [14] 输出一组受限的 Declare模板 [34] ，再将其转换为 LTLf 的一个片段 [10]。将自然语言转换为 LTL 一直是

[3] 该工具可在 GitHub 上获取：https://github.com/realChrisHahn2/nl2spec。

机器人学界尤其对此感兴趣（概述见 [17] ）；与我们的设定不同，该领域的数据集和应用领域都基于结构化自然语言。无论是否依赖结构化数据，先前所有工具都缺乏对自然语言固有歧义的检测和交互式消解能力，而这正是我们的框架的主要贡献。

2.2 大语言模型

大语言模型通常是由多达1760亿个参数组成的大型神经网络。它们在海量数据上进行预训练，例如“The Pile” [15]。大语言模型的例子包括 GPT [37] 和 BERT [11] 模型系列、开源模型（如 T5 [39] 和 Bloom [40], ）以及商业模型（如 Codex [6]）。大语言模型属于 Transformer [43], ，后者是自然语言处理领域最先进的神经网络架构。此外，Transformer 在应用于验证领域的经典问题（例如 [19,41,26,9]）、推理问题（例如 [28,51]），以及数学和形式化规范的自动形式化 [36]（例如 [50,20,22]）时，都展现出了卓越的性能。

在语言建模中，我们对文本中词元序列的概率进行建模 [42]。文本中词元的联合概率通常建模为 [40]:

$$p(x) = p(x_1, \ldots, x_T) = \prod_{t=1}^{T} p(x_t | x_{<t}) \ ,$$

其中， x 是词元序列， x_t 表示第 t 个词元，而 $x_{<t}$ 是位于 x_t之前的词元序列。我们将其称为自回归语言模型，它会迭代地预测下一个词元的概率。用于语言建模的神经网络方法已经取代了经典方法，例如 n-gram [42]。尤其是，Transformer [43] 被证明是撰写本文时最有效的架构 [37,2,24] （详情请参见附录 B）。

尽管针对特定翻译任务对神经模型进行微调仍是一种有效的方法，并且在将非结构化自然语言翻译为线性时序逻辑方面也已初步展现出泛化能力 [20], ，但在标注数据量有限的情况下，获得高性能的一种常用技术是所谓的“少样本提示” [4]。向语言模型提供任务的自然语言描述，通常还会附带一些示例，以展示输入-输出行为。本文提出的框架依赖于这一技术。我们将在第 3.2 节详细介绍所提出的少样本提示方案。

目前已在框架中实现并用于专家用户研究的模型是 Codex 和 Bloom，它们在测试中表现最佳。

Codex。 Codex [6] 是 GPT-3 的一种变体，初始版本规模最高可达 120 亿个参数，并针对代码进行了微调。GPT-3 最初版本本身的训练数据来自 Common Crawl 的多个变体、[4] Webtext-2 [38], 两个基于互联网的图书语料库以及维基百科 [4]。原始版本 Codex 的微调数据集是

[4] https://commoncrawl.org/

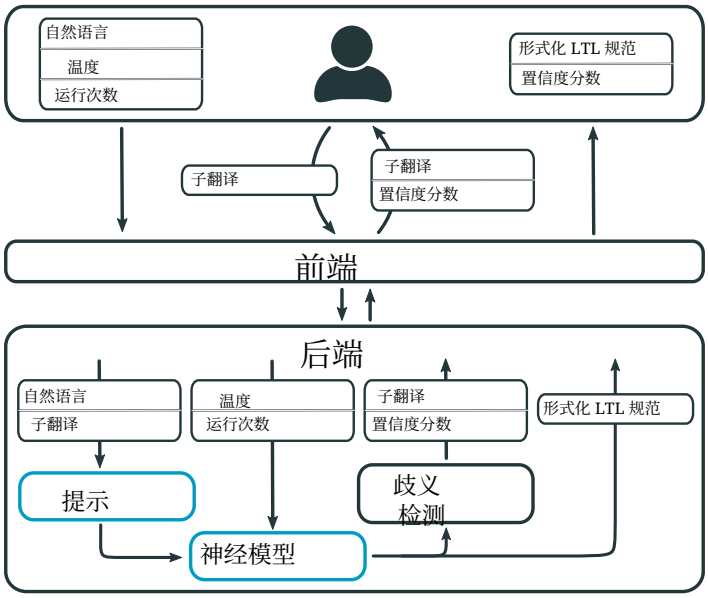

图 2：nl2spec 人在环路中的框架概览：高亮区域表示框架中可轻松扩展的部分。

于 2020年5月从托管在 GitHub 上的 5,400 万个公共软件代码库中收集的，使用了 159GB 的训练数据进行微调。在我们的实验中，我们使用了商业版 2022年 code-davinci-002，该版本可能更大（规模在 176[5]范围内），超过原始 Codex 模型。

Bloom. Bloom [40]是一个开源大语言模型系列，提供多种规模，参数量最高可达 176B，训练时使用了 46 种自然语言和 13 种编程语言。它基于 ROOTS 语料库 [27], 训练，该语料库由 498 个 hugging- face [49,29] 数据集组成，包含 1.61 太字节的文本。在我们的实验中，我们使用了运行于 huggingface 推理 API [6]上的 176B 版本。

3 nl2spec框架

3.1 概述

该框架采用标准的前后端实现。图2展示了 nl2spec 实现的概览。框架中可扩展以用于进一步研究或实际应用的部分已特别标出。该框架使用 Python 3 和 Flask [45], （一个轻量级 WSGI Web 应用框架）实现。在本文的实验中，我们使用 OpenAI

[5] https://blog.eleuther.ai/gpt3-model-sizes/ [6] https://huggingface.co/inference-api

minimal.txt

```
Translate the following natural language sentences into an LTL formula and explain your translation step by step. Remember that X means "next", U means "until", G means "globally", F means "finally", which means GF means "infinitely often". The formula should only contain atomic propositions or operators &, ~, ->, <->, X, U, G, F. Natural Language: Globally if a holds then c is true until b. Given translations: {} Explanation: "a holds" from the input translates to the atomic proposition a. "c is true until b" from the input translates to the subformula c U b. "if x then y" translates to an implication x -> y, so "if a holds then c is true until b" translates to an implication a -> c U b. "Globally" from the input translates to the temporal operator G. Explanation dictionary: {"a holds" : "a", "c is true until b" : "c U b", "if a holds then c is true until b" : "a -> c U b", "Globally" : "G"} So the final LTL translation is G a -> c U b.FINISH Natural Language: Every request r is eventually followed by a grant g. Given translations: {} Explanation: "Request r" from the input translates to the atomic proposition r and "grant g" translates to the atomic proposition g. "every" means at every point in time, i.e., globally, "never" means at no point in time, and "eventually" translates to the temporal operator F. "followed by" is the natural language representation of an implication. Explanation dictionary: {"Request r" : "r", "grant g" : "g", "every" : "G", "eventually": "F", "followed by" : "->"} So the final LTL translation is G r -> F g.FINISH
```

图 3：包含最少线性时序逻辑领域知识的提示。

库和 Hugging Face（Transformer）库 [48]。我们使用标准的线性时序逻辑解析器 [13]解析线性时序逻辑输出公式。该工具既可以作为命令行工具运行，也可以通过基于 Web 的前端运行。

前端负责与人在环路中的人员进行交互。界面分为三个视图：“提示”、“子翻译”和“最终结果”视图（见图 1）。该工具以自然语言句子、可选的子翻译、模型温度和运行次数作为输入，并输出子翻译、置信度分数、备选子翻译以及最终形式化结果。随后，前端支持交互式地选择、编辑、删除或添加子翻译。后端负责处理底层神经模型、生成提示以及消解歧义，即计算置信度分数（包括备选子翻译），并实现交互式少样本提示算法（参见第 3.2 节）。该框架旨在提供便捷的接口，以便实现新模型并编写特定领域提示。提示是一个 .txt 文件，可针对特定领域进行调整，以提高翻译质量（见附录 C）。然而，要应用该框架的方法，提示必须遵循我们的交互式提示方案，我们将在下一节介绍该方案。

3.2 交互式少样本提示

该方法的核心是将自然语言输入分解为若干子翻译。我们提出一种交互式提示方案，利用底层神经模型生成子翻译，并借助这些

算法 1： 交互式少样本提示算法

输入：自然语言 S、少样本提示 F、给定子翻译的集合 (s, φ)，以及语言模型 M
交互： 子翻译集合 (s, φ)、置信度分数 C
模型特定参数集合 P：例如，模型温度 t、数量运行 r
输出：线性时序逻辑公式 ψ，用于形式化 S
  $\psi, (s, \varphi) \quad C =$ 空
  当 用户不批准线性时序逻辑公式 ψ **时**
    交互式提示 $=$ 计算提示(S, F, (s, φ))
    $\psi, (s, \varphi) \quad C =$ 查询(M, P, 交互式提示)
    $(s, \varphi) =$ 用户交互($(s, \varphi) \quad C$)
  结束 while 循环
  返回 ψ

子翻译生成最终翻译。算法 1 展示了交互循环的高层概览。其主要思路是让人在环路中选择添加、编辑或删除子翻译，并将其作为提示中的“给定翻译”反馈给语言模型（见图 3）。使用此提示 F、模型特定参数 P以及循环中计算得出的交互式提示查询语言模型 M 后，模型会生成自然语言解释、子翻译词典和最终翻译。置信度分数是通过对 M 的多次查询进行投票计算得出的，并显示共识分数最高的（子）翻译；也可以显示备选翻译，并通过点击向下箭头进行交互式选择。

图 3 展示了一个通用提示，说明了我们的方法。该提示方案由三个部分组成：规范语言特定部分（第 $1-4$ 行）、少样本示例（第 $5-19$ 行），以及包含自然语言和子翻译输入的交互式提示（未显示，作为输入给出）。规范语言特定部分包含从“思维链”生成中借鉴的提示工程技巧，用于诱导大语言模型进行推理 [47]。然而，nl2spec 的关键在于少样本示例的设置。这个最小提示由两个少样本示例组成（第 $5-12$ 行和第 $12-19$ 行）。“FINISH”标记表示一个示例的结束，它是机器学习模型的停止标记。nl2spec 中的一个少样本示例由以下部分组成：自然语言输入（第 5 行）、给定翻译的词典，即子翻译（第 5 行）、用自然语言对翻译的解释（第 $6-10$ 行）、解释词典（用于总结子翻译），以及最终的线性时序逻辑公式。

该提示方案会从模型中引出子翻译，这些子翻译可作为对形式化过程的细粒度解释。需要注意的是，提示中提供的子翻译既不唯一，也不完整，但它们为语言模型生成正确的形式化结果提供了上下文。

4 评估

在本节中，我们在通过开展专家用户研究获得的数据集上评估我们的框架和提示方法。为展示该框架的普适性，我们使用了仅包含规范语言最少领域知识的最小提示（见图 3）。该提示是在开展专家用户研究之前有意编写的。我们将少样本示例限制为两个，甚至没有提供包含“给定翻译”的少样本示例。我们使用最小提示，旨在将评估重点放在交互式子翻译细化方法对于消除歧义和修复错误翻译的有效性上。在实践中，人们希望用能够尽可能准确地捕捉底层分布的领域特定示例替换这一最小提示。作为概念验证，我们将在附录 C 中对此进行详细说明。

4.1 研究设置

为了获得一个由非结构化自然语言及其线性时序逻辑形式化结果构成的基准数据集，我们请该领域的五位专家提供他们认为对神经机器翻译方法具有挑战性的示例。与遵循严格语法和句法结构的现有数据集不同，我们没有对研究参与者施加此类限制。每条自然语言规范限于一个句子和五个原子命题 a，b，c，d，e。请注意，nl2spec 不受特定原子命题集合的限制（参见图 1）。变量方案可通过初始子翻译指定。我们将在附录 E 中详细说明这一点。为确保实例互不重复，专家们在一份共享文档中协作，最终得到 36 个基准实例。我们为感兴趣的读者提供三个随机抽取的示例：

自然 l l$_{\text{语言}}$	S 线性时序逻辑	specification ψ
如果 b 成立，那么在下一步中，c 将一直成立，直到 a 成立；或者 c 始终成立。	b -> X ((c U a) | | G c)	
如果 b 在某个时刻成立，那么 a 必须在此之前的某个时刻成立。	(F b) -> (!b U (a & !b))	
以下命题中的一个将在所有时刻成立：a、b、c。	G(a | b |c)	

现有方法表现不佳（参见表1），这体现了该数据集的难度。

4.2 结果

我们使用最小提示评估了该方法（除非另有说明），运行次数设为三次，温度设为 0.2。

初始翻译质量。 我们分析了初始翻译的质量，即在任何人工交互之前获得的翻译。该实验表明，初始翻译具有较高质量，这对于确保高效的工作流程非常重要。我们将我们的方法与在结构化数据上对语言模型进行微调的方法 [20]，以及使用 GPT-3 或 Rasa [3] 将自然语言翻译为有限的 Declare模式集合的方法 [14] 进行了比较（该方法

即使将原子命题替换为其中使用的实体，也无法处理基准数据集中的大多数实例）。在我们的专家集上评估初始翻译准确率的结果如表1所示。

截至撰写本文时，在后端使用 Codex 在此任务上的表现优于 Bloom：采用最小提示时，能够正确翻译 44.4% 的实例。只有当译文与专家的预期含义一致时，我们才将该实例计为正确翻译；不接受对有歧义的输入的其他译法。除使用最小提示进行的实验外，我们还在用户研究完成后，随机从专家数据集中抽取四个示例，对增强提示进行了实验（其中 3 个示例此前尚未被解决，见附录 C）。使用该分布内提示（ID）时，该工具可翻译 21 个实例（抽取的四个示例仍保留在集合中），即正确率为 58.3%。

该实验表明：1）初始翻译质量较高，且比以往方法更能处理非结构化自然语言；2）仅从分布内抽取少样本示例，对该数据集的翻译质量提升不大；这使得 nl2spec 的关键贡献，即歧义检测和对错误形式化表示的轻松调试，具有价值。由于 nl2spec 不依赖于底层机器学习模型，我们预计随着更多微调模型的出现，未来其性能将进一步提升。

教师-学生实验。在这项实验中，我们使用 Codex 作为底层神经模型，生成一组初始子翻译。随后，我们以 Bloom 作为后端运行该工具，并将这些子翻译作为输入。Codex 最初能够解决、但 Bloom 无法解决的实例共有 11 个。在这些实例中，借助 Codex 提供的子翻译，Bloom 又成功解决了 4 个实例，即 36。4%。Bloom 借助 Codex 成功解决的四个实例是：“a 和 b 从不同时成立。”“每当 a 启用时，三步之后 b 就会启用。”“如果每个 a 最终都会跟在一个 b 后面，那么 c 就必须无限多次成立。”“以下 ap 中的一个将在所有实例中成立：a、b、c。”这说明我们的子翻译方法是有效的：提高子翻译的质量确实会提升最终形式化的质量。即使使用表现不佳的神经网络模型，这一点也成立。请注意，在这项实验中，无需人工监督便能提升形式化质量。

表1：基准数据集上的初始翻译准确率，其中 B 表示 Bloom，C 表示 Codex。

nl2ltl[14] `rasa`	T-5 [20] 微调后的	nl2spec+B 初始	nl2spec+C 初始	nl2spec+C 初始+ID	nl2spec+C 交互式
1/36（2.7%）	2/36（5.5%）	5/36（13.8%）	16/36（44.4%）	21/36（58.3%）	31/36（86.11%）

歧义检测。在基准集中 36 个实例中，至少有 9 个实例包含有歧义的自然语言。我们尤其观察到两类歧义：1）由自然语言的局限导致的歧义，例如运算符优先级；2）自然语言语义上的歧义。nl2spec 有助于消解这两类歧义。

我们数据集中的第一类歧义示例，是引言中提到的例子：“a holds until b holds or always a holds”，专家将其翻译为 (a U b) | G a。然而，运行该工具后，这个例子被翻译为 (a U (b | G(a)))（见附录F）。通过编辑“a holds until b holds”的子翻译，并添加显式括号将其改为 (a U b)，该工具便能按预期进行翻译。第二类歧义的示例是我们数据集中的以下实例：“Whenever a holds, b must hold in the next two steps.” 专家的预期含义是 G (a ->(b | X b))，而该工具将此句翻译为 G((a -> X(X(b))))。将“b must hold in the next two steps”的子翻译改为 b| X b 后，该工具便能按预期翻译输入（见附录F）。

修复错误翻译。 鉴于自然语言固有的歧义以及输入缺乏结构，不能指望该工具首次尝试就能始终正确完成翻译。然而，验证和调试子翻译明显比从头重新起草完整公式更加容易。在使用 Codex 和后端中的最小提示进行初次尝试时，数据集中的 20 个实例未能得到正确翻译（见表1）。通过最多三轮翻译循环（即添加、编辑和移除子翻译），我们成功为其中 15 个实例提取出了正确翻译；平均进行 1.86 轮翻译循环后，我们便能够获得正确结果。例如，考虑实例“每当 a 成立时，b 也成立”，该工具错误地将其翻译为 G(a & b)。将子翻译“b holds as well”修正为公式片段 -> b 后，该句便得到了符合预期的翻译（见附录G）。剩余五个包含高度复杂自然语言需求的实例，例如“a 发生后，b 不会再次发生”，则需要手动翻译（见附录D）。

总计，我们使用 nl2spec子翻译方法论，平均仅执行 1.4 个翻译循环，就正确翻译了 36 个实例中的 31 个，即 86.11%（见表1）。

5 结论

我们介绍了 nl2spec，这是一个将非结构化自然语言转换为时序逻辑的框架。这种方法的一个局限是，在推理时依赖计算资源。这是应用深度学习技术时普遍存在的局限。不过，商业模型和开源模型都提供了易于使用的应用程序编程接口。此外，初始翻译的质量可能会受到底层神经模型在预训练期间接触过的逻辑、代码或数学训练数据量的影响。

nl2spec 的核心是一种方法，可将自然语言输入分解为子翻译，即将公式片段映射到自然语言输入中的相关部分。我们提出了一种交互式提示方案，通过向大语言模型（LLM）查询子翻译，并实现了一个界面，供用户交互式地添加、编辑和删除子翻译，从而避免用户为了修正错误译文而手动重新起草整份形式化表达。我们开展了一项用户研究，结果表明，nl2spec 可用于高效地交互式形式化非结构化且含义模糊的自然语言。

致谢

感谢 OpenAI 提供 Codex 的学术访问权限，并感谢 Clark Barrett 对该工具早期版本提出宝贵意见。

参考文献

1. J.A.R.V.I.S. TSL/TLSF基准测试套件（2021），https://github.com/SYNTCOMP/ benchmarks/tree/master/tlsf/tsl_smart_home_jarvis 2. Al-Rfou, R., Choe, D., Constant, N., Guo, M., Jones, L.：具有更深自注意力的字符级语言建模。载于：AAAI人工智能会议论文集。第33卷，第3159–3166页（2019年） 3. Bocklisch, T., Faulkner, J., Pawlowski, N., Nichol, A.：Rasa：开源语言理解与对话管理。arXiv预印本 arXiv:1712.05181（2017年） 4. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Nee- lakantan, A., Shyam, P., Sastry, G., Askell, A., et al.：语言模型是少样本学习者。神经信息处理系统进展 **33**，1877–1901（2020年） 5. Brunello, A., Montanari, A., Reynolds, M.：从自然语言文本综合线性时序逻辑公式：研究现状与研究方向。载于：第26届国际时间表示与推理研讨会（TIME 2019）。Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik（2019年） 6. Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H.P.d.O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al.：评估基于代码训练的大语言模型。arXiv预印本 arXiv:2107.03374（2021年） 7. Church, A.：递归算术在电路综合问题中的应用。符号逻辑期刊 **28(4)**（1963年） 8. Clarke, E.M.：模型检测。载于：软件技术与理论计算机科学基础国际会议。第54–56页。施普林格（1997年） 9. Cosler, M., Schmitt, F., Hahn, C., Finkbeiner, B.：针对形式化规范的迭代式电路修复。载于：国际学习表征会议（即将出版）（2023年） 10. De Giacomo, G., Vardi, M.Y.：有限轨迹上的线性时序逻辑与线性动态逻辑。载于：IJCAI'13第23届国际人工智能联合会议论文集。第854–860页。计算机协会（2013年） 11. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.：Bert：用于语言理解的深度双向Transformer预训练。arXiv预印本 arXiv:1810.04805（2018年）

12. Donz´e, A.：论信号时序逻辑。载于：运行时验证：第4届国际会议，RV 2013，法国雷恩，2013年9月24日至27日。论文集4。第382–383页。施普林格（2013年） 13. Fuggitti, F.：LTLf2DFA。Zenodo（2019年3月）。https://doi.org/10.5281/ZENODO.3888410, https://zenodo.org/record/ 3888410 14. Fuggitti, F., Chakraborti, T.：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 15. Gao, L., Bider man, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., et al.：The Pile：一个用于语言建模、包含800GB多样化文本的数据集。arXiv预印本 arXiv:2101.00027（2020年） 16. Gavran, I., Darulova, E., Majumdar, R.：基于示例和自然语言的时序规范交互式综合。ACM编程语言学报 4(OOPSLA)，1–26（2020年） 17. Gopalan, N., Arumugam, D., Wong, L.L., Tellex, S.：非马尔可夫任务规约的序列到序列语言落地。载于：机器人学：科学与系统。第2018卷（2018年） 18. Grunske, L.：概率质量属性的规约模式。载于：2008 ACM/IEEE第30届国际软件工程会议。第31–40页。IEEE（2008年） 19. Hahn, C., Schmitt, F., Kreber, J.U., Rabe, M.N., Finkbeiner, B.：向神经网络教授时序逻辑。载于：国际学习表征会议（2021年） 20. Hahn, C., Schmitt, F., Tillman, J.J., Metzger, N., Siber, J., Finkbeiner, B.：源自自然语言的形式化规范。arXiv预印本 arXiv:2206.01962（2022年） 21. Havelund, K., Ro¸su, G.：使用Java PathExplorer监测Java程序。理论计算机科学电子笔记 55(2)，200–217（2001年） 22. He, J., Bartocci, E., Ni˘ckovi´c, D., Isakovic, H., Grosu, R.：DeepSTL：从英语需求到信号时序逻辑。载于：第44届国际软件工程会议论文集。第610–622页（2022年） 23. IEEE-Commission, et al.：IEEE属性规约语言标准（PSL）。IEEE标准1850-2005（2005年） 24. Kaplan, J., McCandlish, S., Henighan, T., Brown, T.B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., Amodei, D.：神经语言模型的缩放定律。arXiv预印本 arXiv:2001.08361（2020年） 25. Konrad, S., Cheng, B.H.：实时规约模式。载于：第27届国际软件工程会议论文集。第372–381页（2005年） 26. Kreber, J.U., Hahn, C.：利用Transformer生成对抗网络生成符号推理问题。arXiv预印本 arXiv:2110.10054（2021年） 27. Lauren¸con, H., Saulnier, L., Wang, T., Akiki, C., del Moral, A.V., Le Scao, T., Von Werra, L., Mou, C., Ponferrada, E.G., Nguyen, H., et al.：BigScience ROOTS语料库：一个1.6 TB的复合型多语言数据集。载于：第36届神经信息处理系统大会数据集与基准赛道（2022年） 28. Lewkowycz, A., Andreassen, A., Dohan, D., Dyer, E., Michalewski, H., Ramasesh, V., Slone, A., Anil, C., Schlag, I., Gutman-Solo, T., et al.：利用语言模型解决定量推理问题。arXiv预印本 arXiv:2206.14858（2022年） 29. Lhoest, Q., del Moral, A.V., Jernite, Y., Thakur, A., von Platen, P., Patil, S., Chaumond, J., Drame, M., Plu, J., Tunstall, L., et al.：Datasets：一个用于自然语言处理的社区库。arXiv预印本 arXiv:2109.02846（2021年）

30. Liu, J.X., Yang, Z., Schornstein, B., Liang, S., Idrees, I., Tellex, S., Shah, A.: Lang2ltl：使用大语言模型将自然语言命令转换为时序规范。载于：CoRL 2022 语言与机器人研讨会 31. Maler, O., Nickovic, D.: 连续信号时序性质的监测。载于：定时与容错系统的形式化技术、建模与分析：定时系统形式化建模与分析国际联合会议（FORMATS 2004）以及实时与容错系统形式化技术会议（FTRTFT 2004），法国格勒诺布尔，2004年9月22—24日。论文集。第152–166页。施普林格（2004）32. Mikolov, T., Chen, K., Corrado, G., Dean, J.: 向量空间中词表示的高效估计。arXiv 预印本 arXiv:1301.3781（2013）33. Patel, R., Pavlick, R., Tellex, S.: 学习将语言 grounding 到时序逻辑形式。NAACL（2019）34. Pesic, M., Van der Aalst, W.M.: 灵活业务流程管理的声明式方法。载于：业务流程管理研讨会：BPM 2006 国际研讨会，BPD、BPI、ENEI、GPWW、DPM、semantics4ws，奥地利维也纳，2006年9月4—7日。论文集 4。第169–180页。施普林格（2006）35. Pnueli, A.: 程序的时序逻辑。载于：第18届计算机科学基础年度研讨会（SFCS 1977）。第46–57页。IEEE（1977）36. Rabe, M.N., Szegedy, C.: 迈向自动数学家。载于：自动演绎国际会议。第25–37页。施普林格, Cham（2021）37. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., 等：通过生成式预训练提升语言理解（2018）38. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., 等：语言模型是无监督多任务学习器。OpenAI 博客 **1**(8), 9（2019）39. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J.: 探索采用统一文本到文本 Transformer 的迁移学习极限。机器学习研究期刊 **21**(140), 1–67（2020），http://jmlr.org/papers/v21/20-074.html 40. Scao, T.L., Fan, A., Akiki, C., Pavlick, E., Ili´c, S., Hesslow, D., Castagn´e, R., Luccioni, A.S., Yvon, F., Gall´e, M., 等：Bloom：拥有1760亿参数的开放获取多语言语言模型。arXiv 预印本 arXiv:2211.05100（2022）41. Schmitt, F., Hahn, C., Rabe, M.N., Finkbeiner, B.: 基于规约模式的神经电路综合。神经信息处理系统进展 **34**, 15408–15420（2021）42. Shannon, C.E.: 通信数学理论。《贝尔系统技术期刊》**27**(3), 379–423（1948）43. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: 注意力就是你所需要的一切。神经信息处理系统进展 **30** （2017）44. Vijayaraghavan, S., Ramanathan, M.: SystemVerilog 断言实用指南。施普林格科学与商业媒体（2005）45. Vyshnavi, V.R., Malik, A.: 使用 Python 和 Flask 进行 Web 开发的高效方法。《国际近期研究与应用期刊》**6**(2), 16–19（2019）46. Wang, C., Ross, C., Kuo, Y.L., Katz, B., Barbu, A.: 学习用于具身机器人的自然语言到 LTL 可执行语义解析器。arXiv 预印本 arXiv:2008.03277（2020）47. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Chi, E., Le, Q., Zhou, D.: 思维链提示能够激发大语言模型中的推理。arXiv 预印本 arXiv:2201.11903（2022）

48. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., 等：Huggingface 的 Transformers：先进的自然语言处理。arXiv 预印本 arXiv:1910.03771（2019）49. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., 等：Transformer：先进的自然语言处理。载于：2020年自然语言处理实证方法会议论文集：系统演示。第38–45页（2020）50. Wu, Y., Jiang, A.Q., Li, W., Rabe, M.N., Staats, C., Jamnik, M., Szegedy, C.: 使用大语言模型进行自动形式化。arXiv 预印本 arXiv:2205.12615（2022）51. Zelikman, E., Wu, Y., Goodman, N.D.: Star：通过推理自举推理。arXiv 预印本 arXiv:2203.14465（2022）

线性时序逻辑（LTL）

本节将向感兴趣的读者介绍线性时序逻辑（LTL）的形式语法和语义。形式化地，LTL 的语法如下：

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \bigcirc\varphi \mid \varphi \,\mathcal{U}\, \varphi \ ,$$

其中，$p \in AP$ 是一个原子命题。我们定义轨迹集合 $TR :=(2^{AP})^{\omega}$。我们使用以下记号来处理轨迹：设 $t \in TR$ 是一条轨迹，$i \in \mathbb{N}$ 是一个自然数。用 $t[i]$ 表示轨迹在第 i 个位置的命题集合。因此，$t[0]$ 表示该轨迹的起始元素。设 $j \in \mathbb{N}$ 和 $j \geq i$。那么，$t[i,j]$ 表示序列 $t[i]\ t[i+1] \ldots t[j-1]\ t[j]$，而 $t[i,\infty]$ 表示从位置 i开始的 t 的无限后缀。

设 $p \in AP$ 和 $t \in TR$。线性时序逻辑公式的语义被定义为满足以下条件的最小关系 $\vDash$：

$t \vDash p$ 当且仅当 $p \in t[0]t \vDash \neg\varphi$ 当且仅当 $t \nvDash \varphi t \vDash \varphi_1 \wedge \varphi_2$ 当且仅当 $t \vDash \varphi_1$ 且 $t \vDash \varphi_2 t \vDash \bigcirc\varphi$ 当且仅当 $t[1,\infty] \vDash \varphi t \vDash \varphi_1 \,\mathcal{U}\, \varphi_2$ 当且仅当存在 $i \geq 0$： $t[i,\infty] \vDash \varphi_2$且对于所有 $0 \leq j < i$： $t[j,\infty] \vDash \varphi_1$.

B Transformer架构

大语言模型底层的神经网络基于 Transformer架构的各种变体。基础版 Transformer 遵循基本的编码器-解码器结构。编码器一次性为输入序列 $x =$（$x0$、.....、x_n）中的每个输入嵌入x_i 构造一个隐藏嵌入 z_i 。嵌入是将纯输入（例如单词或字符）映射到高维向量的过程；相应的学习算法和工具包已经存在，例如 word2vec [32]。给定编码器的输出 $z =$（z_0、.....、z_k），解码器会自回归地生成一系列输出嵌入 $y =$（y_0、.....、y_m）。由于 Transformer架构既不包含循环结构，也不包含卷积，因此会在输入和输出嵌入中加入位置编码，以区分不同的顺序。不同的大语言模型会采用 Transformer架构的不同变体，例如 GPT 模型系列中的仅解码器模型 [37], ，或 BERT 模型系列中的仅编码器模型 [11]。

与其维护隐藏状态（例如在循环神经网络架构中），不如使用自注意力机制，使神经网络能够将其他重要输入元素的隐藏嵌入纳入当前所考察元素的隐藏嵌入中。对于每个输入嵌入 x_i，我们通过将其与权重矩阵 W_k、W_v 和 W_q 相乘，计算出 1）查询向量 q_i、2）键向量 k_i，以及 3）值向量 v_i ；这些权重矩阵是在训练过程中学习得到的

自注意力机制的核心思想是，为每一对 $(x_i,\ x_j)$ 计算一个分数，以表示在计算 x_i 的嵌入时，序列中的哪些位置应被赋予最高关注度。我们可以使用矩阵运算 [43] 一次性计算所有嵌入。设 $Q,\ K,\ V$ 为将由所有 x_i 组成的输入向量 X 分别与权重矩阵 W_k、W_v 和 W_q 相乘后得到的矩阵：

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \ .$$

C 提示

图4展示了在完成专家用户研究后扩充的提示。LTL 的初始教程（第 $1-5$ 行）有意保持与最小提示相同，以展示子翻译的影响。提示设计仍有很大的优化空间，但这超出了本文的讨论范围。例如，文中没有解释时序算子 FG 的组合，导致专家用户数据集中的翻译失败。

作为概念验证，我们另外提供两个构造提示的通用示例。一个基于近期智能家居应用案例研究 [1]（见图6），另一个是信号时序逻辑（STL）提示 [31]，它是 LTL 在连续信号上的一种变体（见图5）。这两个提示都将纳入该工具相应的开源发布中。

D 剩余实例

以下报告了我们使用 nl2spec 和最小提示进行 ≤ 3 次翻译尝试后仍无法翻译的实例。这些实例尤其困难，因为其撰写者使用了“释放”等高级运算符，或将时序含义隐藏在自然语言中：

自然 ll 语言	S 线性时序逻辑 specification ψ
一旦 a 发生，b 就不会再发生。	G (a ->X G ! b)
a 释放 b	(b U (b & ! a)) |G b
a 每第五步成立。	a & G (a ->X ! a & X X ! a & X X X ! a & X X X X ! a & X X X X X a)
a 必须始终成立，但如果它超出，它允许两个时间戳恢复。	! G (! (a & X a))
非 a 最多在两个时间戳上成立	! G (! (a & X a))

E 变量

使用大语言模型的一个优势是，它们能够适应自然语言中的模式。变量的翻译可以直接提供或编辑

为子翻译。例如，考虑以下自然语言输入：“全局地，当进程 0 终止时，进程 1 将在下一步启动”。在翻译中添加子翻译“进程 0 终止”作为 t p0，模型便会自动将“进程 1 将启动”对应的变量调整为 s p1（见图 7）。

F 歧义消解

图 8 和图 9 展示了 nl2spec 前端在消解示例中的歧义时的屏幕截图。

G 修复错误翻译

图 10 展示了 nl2spec 前端在编辑子翻译以调试错误翻译时的屏幕截图。

indistribution.txt

1 将以下自然语言句子翻译成一个线性时序逻辑公式 2，并逐步解释你的翻译过程。请记住，X 表示 3“下一步”，U 表示“直到”，G 表示“全局”，F 表示“最终”， 4 其中 GF 表示“无限多次”。公式应仅 5 包含原子命题或运算符 &、 ~、->、 <->、X、U、G、F。 6 自然语言：每次 a 事件发生后，最终都会发生 ae。 7 已给出的翻译：{} 8 解释：“每次”在输入句子中指的是时序 9 运算符“G”，意味着输入后续部分必须 10 在每个时间点都成立，即全局成立。“最终”在 11 输入句子中对应时序运算符“F”。 12 “a 后接 ae”表示“a”成立后，“e”也必须成立， 13 即翻译为蕴含。因此，“a 最终 14 后接 ae”翻译为“a->Fe”。解释词典： 15 {"每次":"G","最终":"F","a":"a","e":"e","a后 16 接ae":"a ->e","a最终后接ae":"a->Fe"} 17 因此，最终的线性时序逻辑翻译为 Ga->Fe。FINISH 18 自然语言：a 和 b 从不同时发生，但其中一个 19 在每个时间步都成立。已给出的翻译：{} 20 解释：输入中的“a 和 b”翻译为 21 原子命题 a、b 的合取，即 22 翻译为“a&b”。输入中的“a 和 b 从不发生” 23 翻译为这样的时序行为：在所有位置上，即 24 全局地，a 和 b 都不成立，即“G~a&b”。输入 25 还要求“其中一个在每个时间步都成立”， 26 这意味着 a 或 b 全局成立，即翻译 27 为“Ga b”。解释词典：{"a和b":"a&b", 28 "a和b从不发生":"G~a&b","其中一个在每个 29 时间步都成立":"Ga b"} 因此，最终的线性时序逻辑翻译为 30 G~a & b &Ga b。FINISH 自然语言：a 只有在 31 b 先发生后才能发生。已给出的翻译：{}解释：32 输入中的 “ 如果 b 先发生 ” 表示直到某个时刻 33 b 会发生，即翻译为 “Ub” ； 而 “a 只有在……时才能发生” 34 表示不允许 a 成立，即 35 翻译为“~a”。两者结合起来，这个句子表示 36 某件事在某个事件发生之前不应发生，在本例中， 37 b 成立之前不允许 a 成立。解释词典： 38 {"如果b先发生":"Ub","a只有在……时才能发生":"~a"} 因此， 39 最终的线性时序逻辑翻译为 ~aUb。FINISH 自然语言：a 成立 40 直到 b 成立，或者 a 始终成立。已给出的翻译：{}解释： 41 输入中的“a 成立直到 b 成立”翻译为时序 42 模态 “a Ub”，而输入中的“或者”翻译为 43 运算符“·”。输入中的“a 始终成立”表示全局 44 a 成立，即翻译为“Ga”。解释词典： 45 {"a成立直到b成立":"aUb","或者":"·","a始终成立": 46 "G a"} 因此，最终的线性时序逻辑翻译为 aUb Ga。FINISH

图4：用户研究结束后，通过从数据集中随机抽取四个示例设计的提示。

stl.txt

```
Translate the following natural langauge sentence into an STL formula and explain your
translation step by step. Let a and b be variables. Remember that U[a,b] means

"until the interval a and b"  "F [a,b]" means eventually in the interval between ,,,
a and b  and always is "G[a,b]". Additionally
                                                STL consists of predicates. ,,,,
Assume signals x1[t]  x2[t]  . . .   xn[t]  then atomic predicates are of the form:
fx1[t]  . . .   xn[t] > 0. The STL formula should only contain atomic propositions  ,,,

boolean operators &  ~  ->  <-> and temporal operators U[a,b]  G[a,b]  F[a,b].      ,,,,,,
Natural Language: The signal is never above 3.5. Given translations: {}
Explanation: "The signal" from the input translates to the variable "x[t]"
                                                                                "above 3.5" ,
from the input translates to "> 3.5"

                                        "never above 3.5" thus means that the signal ,,
should never be above 3.5  i.e.
                                    always under 3.5. Explanation dictionary:
{"The signal": "x[t]", "above 3.5": "> 3.5", "never above 3.5":
"G < 3.5"} So the final STL translation is G x[t] < 3.5.
Natural Langauge: Between 2s and 6s the signal is between -2 and 2. Given
temporal operator "G[2,6]" and "the signal is between -2 and 2" translates to the
predicate "absx[t] < 2". Explanation dictionary: {"Between 2s and 6s": "G[2,6]",
translations: {} Explanation: "Between 2s and 6s" from the input translates to the
"the signal is between -2 and 2":  absx[t] < 2"}.
So the final STL translation is G[2,6] absx[t] < 2.
```

图5：信号时序逻辑（STL）的提示示例。少样本示例取自一次演讲 [12]。

smart.txt

```
Translate the following natural language sentences into an LTL formula and explain your
translation step by step. Remember that X means "next"  U means "until"                ,,
                                                                    G means
"globally"  F means "finally"                                                ,,

                                which means GF means "infinitely often". The formula ,,,,,,,
should only contain atomic propositions or operators  &  ~  ->  <->  X  U  G
                                                                                F.
Given translations: {} Explanation: "The coffee machine is ready" from the input
Natural Language: The coffee machine is ready from the input room.
translates to "c" and "always" translates to "G". "Somebody is at the room" from the
input translates to "r". Explanation dictionary: {"The coffee machine is ready" :

Given translations: {} Explanation: "Lights are on" from the input translates to ,,
"l" and "somebody" from the input translates to "r". So the final LTL translation is
"G and "somebody" is in the room" : "r", "only if" : "<->"}
So the final LTL translation is G l <-> r.FINISH
```

图6：智能家居案例研究中的两个少样本示例的提示示例 [1]。

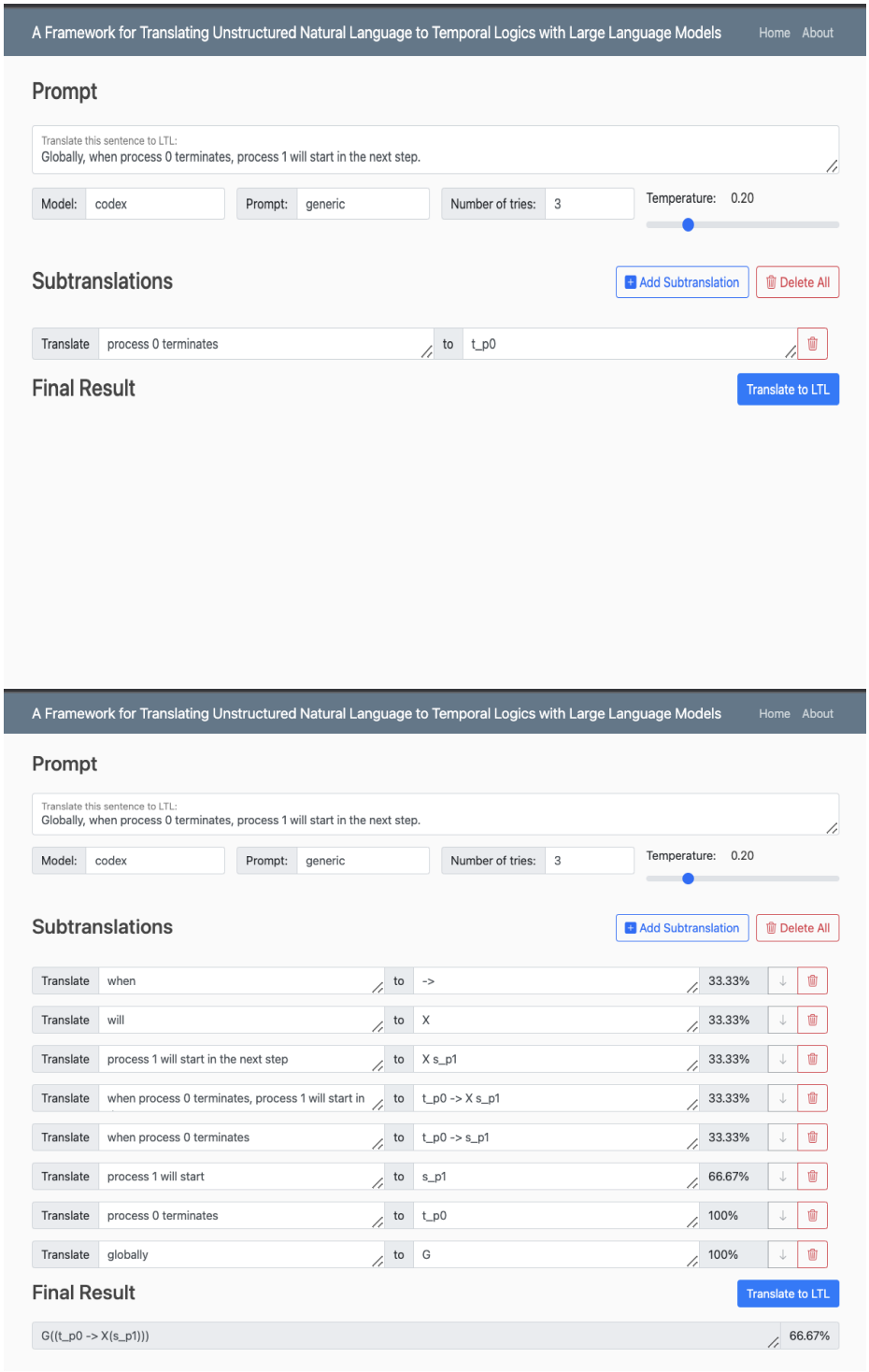

图7：为变量翻译提供子翻译。

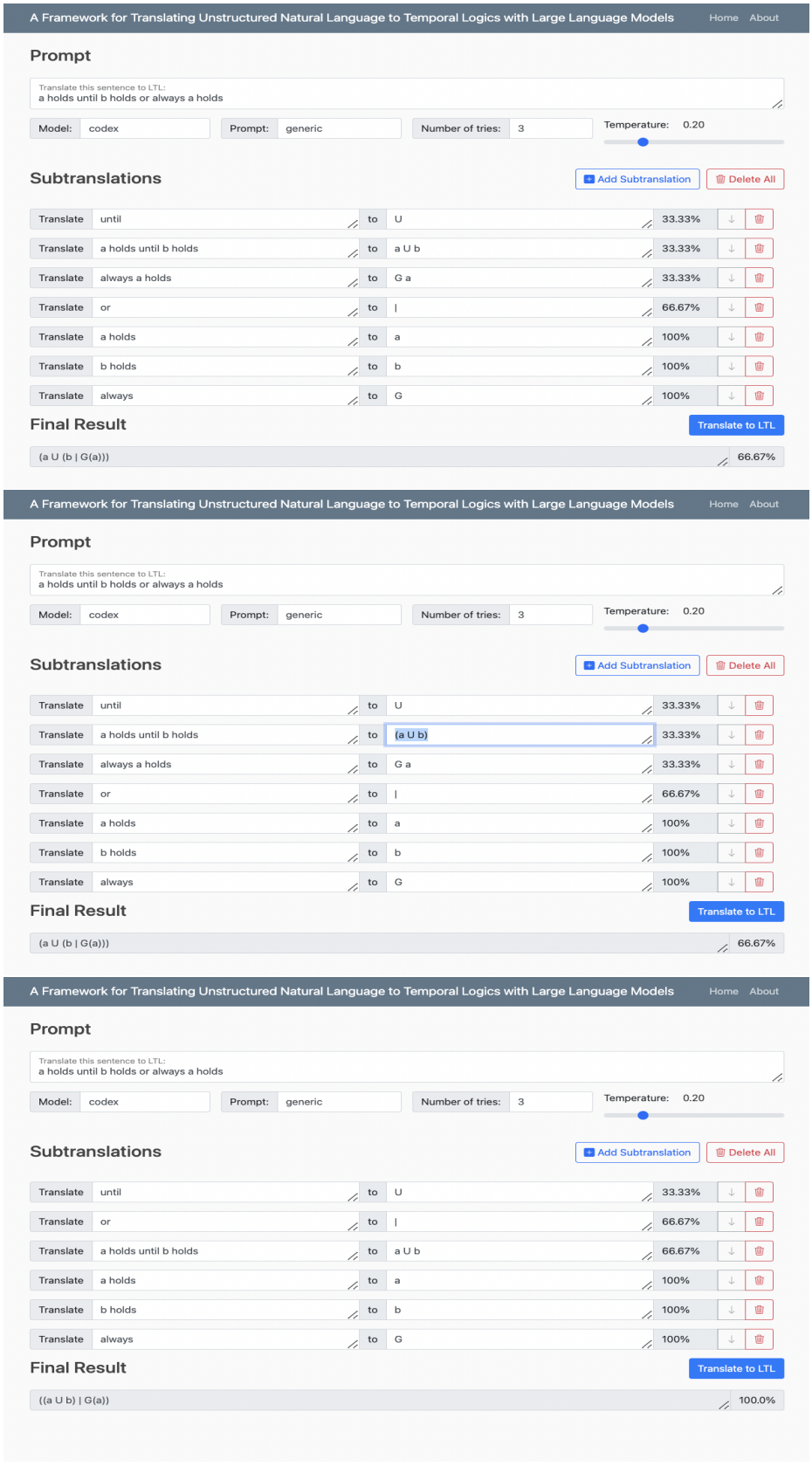

图 8：通过修正带有括号的子翻译，确保运算符优先级。

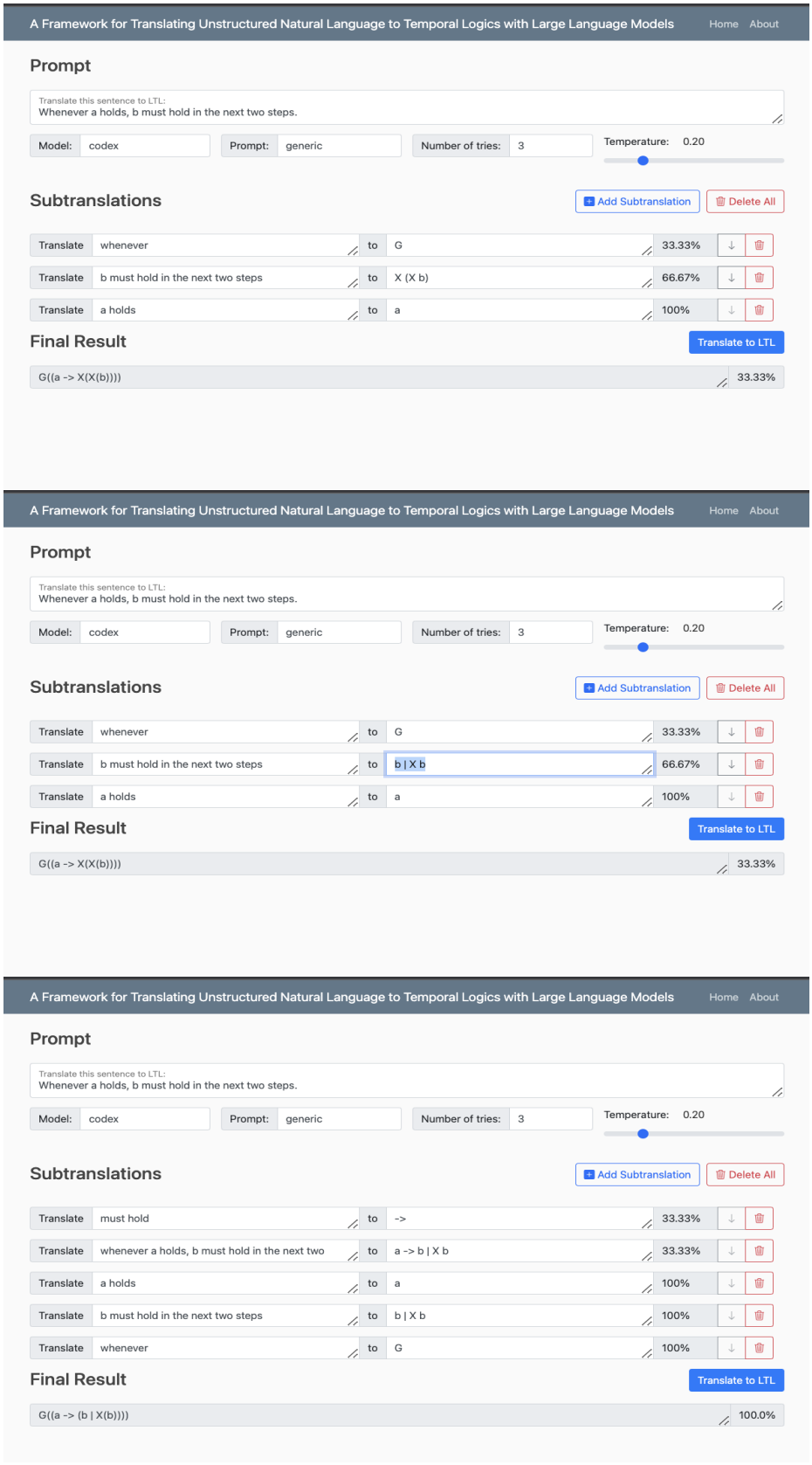

图9：编辑子翻译，以体现输入的含义。

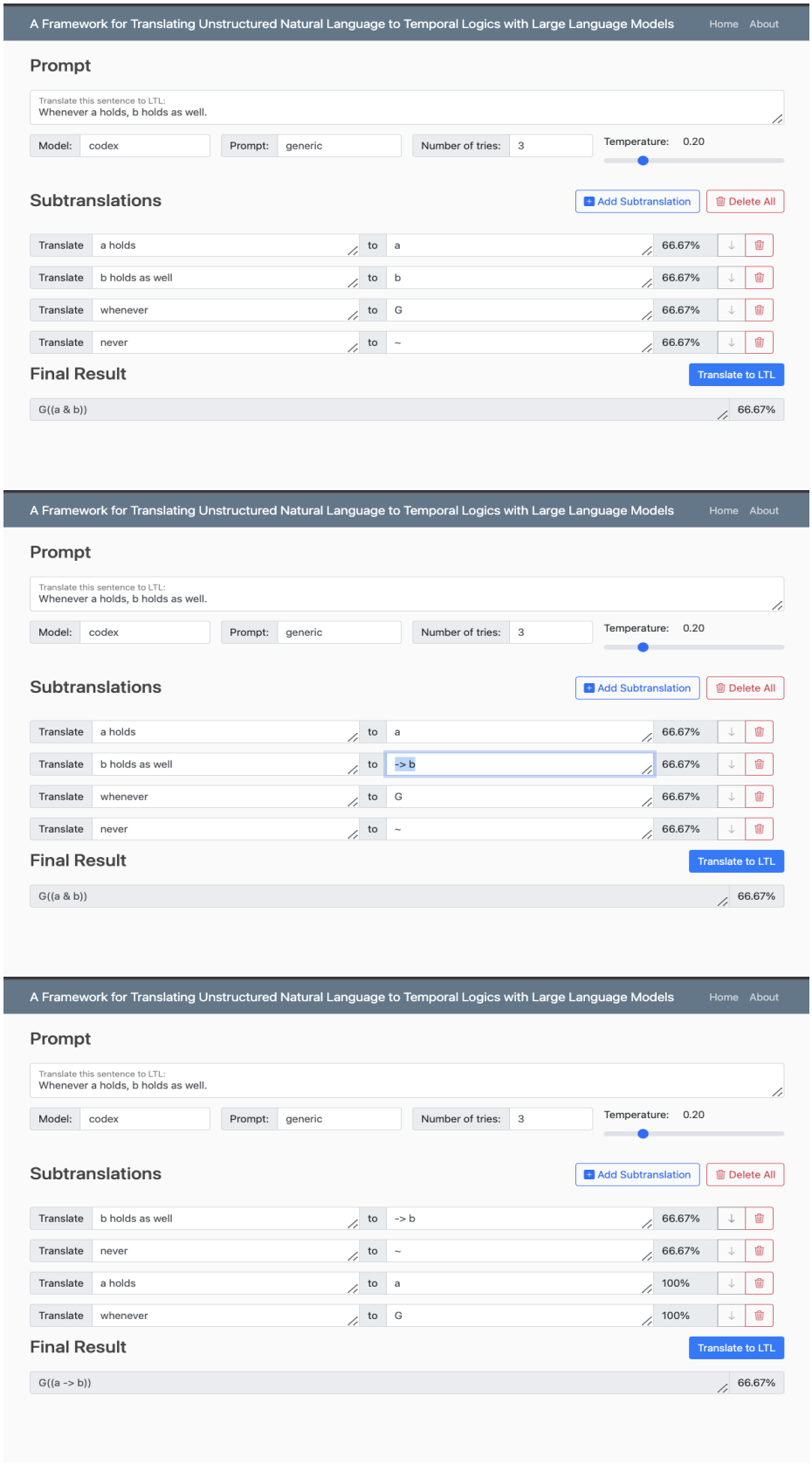

图 10：编辑子翻译以体现输入的含义。