基于大语言模型的神经符号机器人动作规划框架

Alessio Capitanelli 和 Fulvio Mastrogiovan ni

热那亚大学信息学、生物工程、机器人学与系统工程系，Via OperaPia 13, 16145，意大利热那亚 alessio.capitanelli@dibris.unige.it, fulvio.mastrogiovanni@unige.it

arXiv:2303.00438v3 [cs.AI] 4 Jun 2024

摘要。 符号任务规划因其易于理解且易于部署在工程化机器人架构中，成为实现机器人自主性的一种广泛应用的方法。然而，在真实世界中高度动态的人机协作场景下，符号任务规划技术难以扩展，因为在动作效果可能不会立即显现的规划领域中，其性能不佳；或者当机器人工作空间的情况发生变化、需要频繁重新规划时，也会面临困难。长期来看，规划方案的有效性、计划长度和规划时间都可能妨碍机器人的效率，并对整体人机交互的流畅性产生负面影响。我们提出了一个框架，称为 Teriyaki，旨在弥合符号任务规划与机器学习方法之间的差距。其核心思路是训练大语言模型（LLM），即 GPT-3，使其成为兼容规划领域定义语言（PDDL）的神经符号任务规划器，并利用其生成能力克服符号任务规划器固有的一些局限。潜在优势包括：（i）随着规划领域复杂度提高，可扩展性更强，因为大语言模型的响应时间随输入和输出的总长度线性增长，而符号任务规划器的响应时间则呈超线性增长；（ii）能够逐动作合成规划方案，而非端到端生成，并在每个动作生成后立即提供执行，而不必等待完整规划方案生成，从而实现并发规划与执行。过去一年中，研究界投入了大量精力评估大语言模型的整体认知能力，并取得了不同程度的成效。与此不同，我们利用 Teriyaki，旨在特定规划领域中实现整体规划性能与传统规划器相当，同时发挥大语言模型在其他指标上的能力，尤其是其短期和中期生成能力，这些能力可用于构建前瞻预测规划模型。

这些能力用于构建前瞻预测规划模型。选定领域的初步结果表明，我们的方法能够：（i）解决测试数据集中 1000 个样本中 95.5% 的问题；（ii）生成比传统符号规划器短 13.5% 的规划方案；（iii）将规划方案可用前的平均总等待时间最多缩短 61.4%。

关键词： 人工智能、生成式、神经符号、大语言模型、任务规划、PDDL、人机交互、GPT

1 引言

如今，我们正见证着这样一类场景的出现：机器人被期望以更高的自主性、灵活性和鲁棒性开展工作，尤其是在需要与人类共同完成，或与人类密切协调的任务中。事实上，在人机协作任务中，人类可以只是机器人行为的受益者，也可以作为队友，在机器人动作展开的过程中实际参与其中。

这种情况不仅存在于辅助机器人和服务机器人领域，在受工业4.0范式启发的环境中也变得尤为重要；人们认为，该范式将深刻塑造下一代制造业运营的性质 [Heyer, 2010]。这些场景要求机器人展现出相当动态且可靠的能力，以支持与人类操作员开展有目的的协作：尽管机器人的自主性可能处于连续变化的范围内，并且还取决于当前所执行的具体操作，但人类操作员可以监督机器人动作，并在必要时进行干预。因此，机器人不仅必须根据与人类操作员的交互可靠地运行，还必须应对极有可能发生的意外事件，尤其是在待完成的工作结构性较弱且涉及难以预测的人类行为时 [Darvish等人, 2018]。这一范式转变要求机器人持续验证或重新规划其未来动作，理想情况下能够预测不断变化的条件或人类不可预见行为所产生的结果，至少也能对此作出反应。因此，规划性能对于确保机器人高效运行以及实现有效的人机协作至关重要。

在具有此类要求的场景中，传统的符号任务规划器存在局限，影响了其效能。在当前实践中，规划器使用搜索算法探索建模环境可能处于的各种状态，直到找到一系列规划算子（即动作），使状态之间的转换最终到达与期望目标情境相符的状态。

环境可以对其作出假设，直至找到一系列连接不同状态的规划算子（即动作），并最终到达与期望目标情境相符的状态。在最简单的形式下，规划引擎假设状态和动作以谓词表示，也就是说，谓词是用于建模现实某些方面的片段，而这些方面则通过（可能已实例化且带类型的）变量表示。如果采用封闭世界假设，即除非另有说明，否则谓词的真值均假定为假，那么每个中间状态所包含的具体谓词仅取决于生成该状态的动作所产生的效果。因此，规划过程可视为一种生成式马尔可夫决策过程，其基于由（已实例化且带类型的）变量、谓词及其在动作中的组织方式构成的符号基底。平均而言，随着规划过程中涉及的符号数量增加，搜索算法往往会在时间和空间复杂度上呈现组合爆炸 [Garrett等人，2020]。启发式方法旨在通过牺牲最优性能和算法完备性，来缓解这种不利的计算复杂度。尽管有各种可用的启发式方法，且它们往往使规划过程具有满意解导向的性质，但总体计算复杂度仍使频繁重新规划的代价极高，尤其是考虑到符号任务规划器合成的规划方案是原子实体：要么完整生成，要么无法使用。尽管文献中已提出分层任务规划方法，借此可以快速生成高层（通用）规划方案，并分批生成低层（具体）子计划，但基本问题依然存在：还必须谨慎设计哪些部分可以在高层合成，以及哪些部分（即机器人实际可执行的动作）应以子计划的形式提供[Mastrogiovanni 等人，2004，Murali 等人，2020，Darvish等人， 2021]。

当人类操作员与机器人协作完成任务时，他们可以实现高水平的协调，使各自的动作在精准高效的时机下同步配合 [Carf´ı 等人， 2019]。根据 Hoffman [2019], 的说法，这一特性称为 流畅性 。研究表明，人类完成一项动作后，由队友的动作所导致的延迟，与主观流畅性感知高度相关。显然，前述规划时间的组合爆炸会对人机协作的质量产生负面影响，尤其是其流畅性。有人认为，可以通过以下方式提升人机协作中的流畅性：

引入直观的沟通渠道，让人们清楚地了解机器人接下来将执行什么动作 Macci´o 等人， [2022]。事实上，这类方法在协作过程中有效；而在需要重新规划时，则必须暂停协作过程。

本文介绍了一个名为 Teriyaki[1], 的框架，用于训练和调用大语言模型（ LLMs） [Hatcher 和 Yu， 2018] 使其充当任务规划器。Teriyaki 基于两项概念原则，这些原则既体现在整体架构设计中，也体现在实际应用方面。第一项原则与以下事实有关：Teriyaki 是更通用的程序化工作流的一种可能实现，该工作流旨在生成数据驱动的领域特定规划器；第二项原则涉及一种工作流，其结构如下：首先，使用现有的基于 PDDL 的规划器生成领域特定的问题和规划方案数据集；然后，使用该数据集微调所选的大语言模型；最后，合成的规划器可替代用于生成该数据集的原始传统规划器。在任何传统的感知-规划-执行架构中，替换规划器都很简单；但也可以调整此类架构，以利用 Teriyaki 规划器相较于传统规划器的主要优势，即逐动作生成计划，从而实现并行规划与执行。

OpenAI 的 GPT-3 [Brown 等， 2020] 等大语言模型具有以下特点：(i) 计算复杂度随提示和补全的总长度线性增长；(ii) 在迭代预测序列中的下一组符号时，能够生成部分结果。我们认为，可以利用这两项特点来设计运行时间复杂度更低的任务规划器，并使其能够在完整计划综合完成之前提供下一步动作。这样，集成此类规划器的机器人架构便可在整个规划生成之前开始执行动作；这在人机协作场景中是一项非常理想的要求。如果机器人架构允许，这或许还能实现规划生成与执行的并发。此外，如果此类模型经过训练，能够使用规划领域定义语言（PDDL） [Aeronautiques 等，1998], 接收输入并返回输出，那么它将与机器人学界已广泛采用的现有符号任务规划软件框架完全兼容，例如 ROSPlan [Cashmore 等， 2015]。我们将 Teriyaki 背后的工作流程称为神经符号，因为它结合了基于神经网络的学习（子

[1] Teriyaki： https://github.com/alessiocpt/teriyaki

策略）与符号知识表示和推理（逻辑） [Garcez 和 Lamb， 2020]。与传统符号任务规划器不同，我们的方法依赖领域且需要训练，但使用传统任务规划器，便可轻松生成用于在给定领域上训练的大型数据集。

Teriyaki 已在两个规划领域中进行了评估和测试，具体针对人类操作员与机器人协作操作关节物体这一任务。这是一项具有挑战性的任务，文献中此前已有描述 [Capitanelli 等人，2018，Bertolucci 等人， 2019, 2021]。选择这两个领域，是因为它们给规划过程带来了挑战：(i) 对于符号任务规划器而言，操作关节物体的难度会随着其连杆、关节数量以及允许的角度配置数量增加而急剧上升；(ii) 对于大语言模型而言，这些领域包含条件效果，也就是说，某个动作可能会以无法从动作参数本身立即看出的方式改变表示的状态。例如，如果某个连杆绕关节旋转，那么链条中所有更下方的连杆也应相对于绝对参考系隐式地旋转。

训练过程采用了一个包含 9000 个问题-规划对的大型数据集；这些数据由现有的先进传统符号 PDDL 规划器 Probe [Lipovetzky 和 Geffner，2011]自动生成并求解。值得注意的是，该数据集以及用于生成更多数据对的代码均可在配套代码库中获取。我们使用此前未用于训练的 1000 对数据对对所得模型进行了严格测试，并根据有效规划方案的百分比、计划长度和规划时间进行评估。训练期间，我们收集了与规划有效性相关的数据。规划有效性定义为能够通过形式化证明其正确的规划方案所占的比例。我们通过这些数据考察规划有效性随训练样本数量增加而发生的变化，并评估相似规划领域之间是否可以进行迁移学习。

结果显示，Probe 与 Teriyaki 生成的求解器在规划有效性方面几乎完全相同。不过，在其中一个领域中，Teriyaki 的计划长度最多可比 Probe 缩短 13.5%。关于规划时间，值得注意的是，由于两种规划器运行所依托的计算架构不同，无法进行客观比较：Teriyaki 基于专有云服务 GPT-3，其响应时间在测试期间既无法保证，也无法预测；而 Probe 可以在完全开放且受控的工作站上运行。事实上，在当前的实验架构中，Probe 生成完整规划的速度仍然更快，也就是说，在需要 Teriyaki 端到端合成一份完整的 PDDL 兼容规划时，

然而，当规划器作为模块集成到机器人架构中时，使用 Probe 这类传统规划器会迫使我们采用纯粹的感知-规划-行动方法，即必须先完整合成规划，才能执行第一个动作；而 Teriyaki 则会在生成第一个动作后立即将其提供给执行。在我们的实验中，这利用了规划与执行的同步进行，使总体等待时间平均缩短了 61.4%。最后，我们还通过实验观察到，Teriyaki 求解器的扩展性取决于输入和输出长度，而非问题与领域的复杂程度。这表明，可能存在一些更复杂的领域，使 Teriyaki 能够胜过 Probe，乃至通常意义上的传统符号规划器。

总而言之，本文的主要贡献如下：(i) 我们设计、开发并向社区发布了 Teriyaki，这是一套用于训练和使用 GPT-3 来解决给定领域中兼容 PDDL 的问题的工作流，同时还发布了原始数据集和辅助代码；(ii) 与文献中的类似方法相比，尤其是 Plansformer 方法 [Pallagani 等人， 2023],Teriyaki 已在长度长 80% 且包含条件效果的规划方案上进行了测试；(iii) 我们证明，大语言模型在规划方案长度方面可以优于非最优规划器，例如 Probe [Lipovetzky 和 Geffner， 2011]；(iv) 我们提出利用大语言模型在生成规划方案时流式传输输出的能力，实现同步规划与执行，从而显著缩短执行开始前的等待时间，进而提升人机协作的流畅性。

尽管结果令人欣喜，我们仍想强调，目前不应将使用 Teriyaki 生成的求解器视为传统符号规划器的完整替代方案。相反，我们建议将其视为大语言模型规划能力的概念验证，以及随着技术成熟而可能应用于人机协作场景的探索。

2 相关工作

神经符号方法认为：（i）其知识以基于向量的表示形式编码，支持神经网络从数据中高效学习；（ii）通过查询和提取训练好的网络中的知识，可获得离散符号 [Garcez 和 Lamb,

2020]。长期以来，理论上认为，这类方法可以利用学习过程所诱导的推理机制，为推理中的组合爆炸问题提供一种替代方案。然而，直到最近，这些方法才开始受到关注，主要应用于自然语言处理领域 [Dale， 2021]。其中，最引人注目的是 GPT-3 及其后续版本 GPT-3.5 和 GPT-4。这些由 OpenAI 发布的大语言模型[2] 因能够生成由结构化句子构成、复杂程度达到人类水平的文本而闻名 [Brown 等， 2020]。其他流行的模型包括 LLama 2 [Touvron 等， 2023]、LaMDA [Thoppilan 等， 2022]、PALM [Chowdhery 等， 2022]、Megatron-Turing NLG [Smith 等， 2022] 以及 BLOOM [Scao 等， 2022]。这些模型的一些重要应用包括文本摘要生成、补全、情感分析，以及与我们的应用尤其相关的代码生成 [Chen 等， 2021]。

大语言模型是深度生成式模型 [Oussidi 和 Elhassouny， 2018]，最具前景的应用之一；也就是说，它们是一类无监督深度学习算法，能够捕捉可能构成某类数据生成基础的概率分布。一旦完成估计，便可以生成与该概率分布相兼容的合成数据。特别是，大语言模型能够估计一个句子的概率，该句子的概率表示为在其之前出现的符号条件下每个离散符号概率的乘积。例如，给定几个作为提示的词语，它们可以轻松建议最可能的句子补全方式。通常，这类符号被称为词元，表示较短的字符序列。

实现大语言模型的常见方式包括循环神经网络 [Mikolov 等人、 2011]，长短期记忆（LSTM）网络 [Sundermeyer 等人，以及2012] 近来出现的 Transformer [Vaswani 等人，GPT 模型是其中最著名的例子。Transformer 是基于编码器-解码器架构的模型，并采用自注意力 机制，使其能够对输入数据的各个部分赋予不同的权重。尽管就目前而言，基于Transformer的架构所生成答案的长期逻辑连贯性尚无保障，尤其是在处理较长的符号序列时，但这类模型展现出令人惊讶的能力，能够生成合理且连贯的指令列表，内容从烹饪食谱到可运行的代码都有，能力远超循环神经网络和LSTM网络。

[2] 网页：https://openai.com/blog/gpt-3-apps

自大语言模型普及以来，将大语言模型（LLM）集成到机器人应用中便受到了广泛关注，主要是将其作为提升机器人推理和规划能力的工具。PaLM-E [Driess 等人 2023]探索了将现实世界传感器数据融入语言模型的具身语言模型，为机器人领域的问题提供基础关联。这种方法将多模态输入用于训练，在序列机器人操作规划等任务中展现出良好前景，与我们提升机器人在动态场景中的自主性的目标一致。同样，GENOME [Chen 等人2023] 提出了一种利用大语言模型进行视觉推理的模型，重点在于模块生成与复用，以实现快速任务泛化；这一理念与我们的 Teriyaki 框架的目标相辅相成。

人工智能社区也对此广泛关注，重点在于评估大语言模型真正具备的通用逻辑能力，尤其是在零样本到少样本尝试中的表现。正如 Brown 等 [2020],所述，对于零样本方法，我们指的是直接向模型提出蕴含一定逻辑能力要求的问题；而单样本和少样本方法则会在请求中向模型提供数量非常有限的示例。在所有这些情况下，模型都只是接受提示，而不会使用特定示例进行进一步训练,这一过程如今通常被称为微调。Valmeekam 等人的工作 [2022]提出了一个用于评估少样本场景下基于大语言模型的规划和推理能力的基准。作者使用 PDDL，仅用于以自然语言自动生成模型此前从未见过的若干逻辑问题，并依据多项指标比较了主流大语言模型的表现。表现最佳的模型只能解决 500 个问题中的 25 个，即 5%。Wang 等人 [2023] 通过少样本学习和语法提示取得了更好的结果。在这种方法中，GPT-3.5 和 GPT-4 会根据测试输入预测一种专用语法，然后按照该语法本身的规则生成输出。除此之外，作者还测试了从流行但相对简单的 PDDL领域中提取的 PDDL 格式输入。尽管语法提示的优势显而易见，但从绝对结果来看，其表现取决于领域，结果不一，成功率介于 40% 和 100% 之间。LLM-Planner [Song 等人， 2023] 是另一种基于 GPT-3 的少样本方法，专门用于具身推理，即旨在应用于处理现实世界物理环境的智能体的技术。这种方法使用近似自然语言，其特点是基于智能体在每一步能够在环境中观察到的内容进行动态重新规划机制，而该智能体可能

观察到的内容。各类任务中的高层规划有效性介于 26% 和 51% 之间。Logeswaran 等人 [2022] 和 Wake 等人 [2023]也讨论了类似方法。Singh 等人 [2023] 则提出了 Progprompt，该方法利用 GPT-3 在代码生成方面的强大性能，通过提示模型生成可执行的 Python 代码来解决给定问题。Silver 等人 [2023] 进一步发展了这一概念，提出了一种在 PDDL领域中进行广义规划的方法，并加入了自动调试机制。

正如 Huang 等人所指出的，几乎所有这些方法的主要问题在于，大语言模型生成的规划方案往往无法准确映射到可行动作，因此大多数方法都试图通过巧妙的提示策略，提高实现预期目标状态的成功率。Silver 等人在尝试解决实际的 PDDL 问题时也面临类似挑战，并在输入和输出中都保持 PDDL 格式。作为替代方案，他们提出利用大语言模型生成的虽好但并不完美的输出，来辅助传统的（即基于启发式搜索的）基于 PDDL 的规划器。尽管这种方法缓解了前述问题，并且通常能将规划时间缩短 10% 以上，但它无法再逐个动作地生成规划方案，也就无法适用于人机协作场景。

因此，看来对于目前可用的大语言模型而言，若要可靠地解决任务规划问题，也就是说，达到与传统基于 PDDL 的规划器相当的成功率，只有通过适当的微调 过程才有可能。这种方法需要使用与当前规划领域相关的特定示例（即问题-计划对）对基础模型进行进一步训练。Pallagani 等人提出的 Plansformer 就是这类方法。 [2023]作者通过微调专门用于代码生成的大语言模型 CodeT5 [Wang 等人，2021], 训练了五种不同的模型，用以解决数量相同的简单且广为人知的基于 PDDL 的领域，例如汉诺塔；Plansformer 在其中实现了 97% 的有效规划方案。微调数据集通过 Fast-Downward 规划器 [Helmert，2006], 求解 18000 个随机生成的问题生成，随后将问题和规划方案都转换为更紧凑的形式。基于 PDDL 的问题还通过列出领域中的动作进行了扩充，以明确教会模型这些动作的前置条件和效果。尽管 Plansformer 展示了采用微调方法为不同领域训练求解器的可能性，但所考虑的规划领域相当简单，并未包含条件效果等高级规划概念，

例如。从更一般的意义上说，对这些结果的一种可能解释是：大语言模型学习某个符号在前序符号之后出现的概率这一能力，与传统基于 PDDL 的规划器所依赖的搜索启发式方法之间存在明显相似之处。因此，我们并非只是从模型中提取知识并利用其通用推理能力，而是在训练模型，使其近似生成数据集的规划器所采用的搜索启发式函数。

Teriyaki 采用了类似的方法，但有几个关键区别：（i）我们假设大语言模型能够处理未经修改的、基于 PDDL 的问题描述和规划描述，这也使该模型更易于在现有机器人架构中使用；（ii）为缩短提示长度（这是大语言模型的主要限制之一），我们选择省略部分信息，假设模型能够学会这些信息。例如，在微调期间，我们不会向 Teriyaki 明确提供动作前提条件和效果的相关知识；（iii）模型在一个包含非最优规划方案的数据集上进行训练，这使我们得以观察到，在选定领域中，就平均规划长度而言，大语言模型能够胜过其训练所依据的传统规划器；（iv）我们有意聚焦于复杂的规划领域，即需要使用高级规划构造（如条件动作）才能生成可证明有效的规划方案的领域；（v）我们提出在规划方案生成时使用 Teriyaki 对其进行流式传输，即逐个动作输出，从而在机器人任务中实现同步规划与执行，尤其是在人机协作任务中。

3 方法

3.1 规划领域、PDDL 版本与规划器

在我们的工作中，为了同时挑战所选用的传统PDDL规划器和 Teriyaki，我们选取了两个PDDL领域，以下分别称为 MACRO 和 NO-MACRO。这两个领域对关节物体上的操作动作进行建模，并已在此前的人机协作场景中展示。值得注意的是，在本文所述的工作中，我们采用了 Capitanelli 等人 [2018]讨论过的同一套机器人感知与控制架构。该架构包括基于视觉的感知模块，能够正确检测并识别关节物体的每个连杆及其位姿；这些模块的输出是一系列符号谓词，用于指定物体的整体构型



图 1. Baxter机器人在两个涉及关节物体操作的领域中执行动作。人类可以随时操作关节物体的关节，迫使机器人重新规划。

以用于规划和监控。同样，架构中还包括与操作相关的机器人控制方面，并提供了类似PDDL的符号动作与机器人运动之间适当的映射。图 1 展示了Baxter机器人执行来自这两个领域的一个规划，以及与人类进行的一种可能交互。这两个领域均采用 Capitanelli 等人 [2018], 所述的所谓绝对表示法，即关节物体中成对连杆之间的角度相对于绝对参考系表示。之所以如此选择，是因为采用绝对表示法的领域在规划领域中需要条件效果，也就是说，动作

可能会对未直接受到动作结果影响的关节角度产生隐式效果，因此，规划器需要在维护所有约束方面承受相当大的压力。事实上，如果由传统的符号动作规划器进行处理，就需要传播每个动作的（条件）效果，以修改未直接受到这些效果影响的状态变量值。同样，我们认为，对于 GPT-3 等大语言模型而言，条件效果可能会给模型带来压力，因为鉴于其底层的生成过程，模型应当更难以维持规划方案中的连贯性。

所选两个领域之间的主要区别在于，其中一个使用 PDDL宏 ，而另一个不使用。宏是有序的复合动作，将若干基本动作捆绑在一起；例如，使用一个抓取-旋转-释放动作代替三个原子 动作，即抓取、旋转和释放。在传统动作规划器中，使用宏是减少规划时间的有效方法，但代价是接受不太优的 规划方案。事实上，使用宏可能会导致规划器强制采用某个动作序列，也就是一个给定的宏，但代价是引入可能多余的动作。例如，如果目标状态假定关节物体处于某一给定配置，但机器人夹爪仍保持对物体的抓持，规划器可能会倾向于使用抓取-旋转-释放宏，随后再执行另一个抓取动作（部分抵消该宏的效果），而不是使用抓取和旋转这两个原子动作。由于预计使用宏能够显著缩短最终规划方案，宏也被认为有助于 GPT-3 的生成过程。

由于这两个领域在本质上相似，但会产生长度不同的规划方案，因此它们是检验基于 Teriyaki 的求解器如何随输出长度变化而扩展的理想对象。鉴于此处考虑的两个领域都使用了条件效果，我们采用了 PDDL 2.1 和一个兼容的基于 PDDL 的规划器，即 Probe [Lipovetzky 和 Geffner， 2011],这是 Capitanelli 等人先前工作中使用的规划器之一 [2018]。Probe 是一种满足性规划器，它通过贪心深度优先方法（即 $probe$）探索搜索空间，启发式地尝试生成单个动作序列。从统计上看，这种方法可以快速找到一个解，但该解不一定是最优解。Probe 可用作初始解，以此为基础寻找更优（更短）的规划方案。为此，可以分析 Probe 结果以寻找地标，也就是不同 Probe 都会到达的特定状态，因此这些状态是需要生成的重要状态。如果存在地标，就可以获得地标状态之间的子计划。选择 Probe 有两个

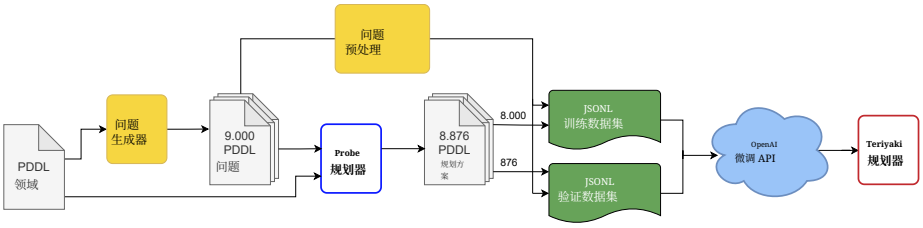

图 2. Teriyaki 微调过程示意图。黄色方框表示用于数据生成和处理的自定义代码，详见第 3.2 节。

原因。一方面，我们在先前工作中使用过它，因此能够直接比较其结果与 Teriyaki 生成的规划方案；另一方面，它的广泛应用有助于我们的工作更轻松地适配到不同场景中。事实上，选择 Probe 会影响结果的质量，因此应当指出，也可以采用更好的替代方案。

3.2 大语言模型的选择、数据集生成与组成

正如我们所预期的，本文采用了 GPT-3。它是一个闭源模型和商业产品，在本文撰写时只能通过 OpenAI 的 API 作为云服务访问。我们的结果并不局限于 GPT-3 的特定特性，原则上可以用任何可用的大语言模型复现。实际上，自本文介绍首次使用 GPT-3 进行实验以来，我们测试了几个可在本地运行的较小模型，例如 LLama 2 7B、13B 和 30B，但它们都无法可靠地解决所选两个领域中的问题，因此本文未报告相关结果。由于只能使用商业模型服务，我们选择 GPT-3，是因为与更新版本相比，它规模更大、速度更快。对于 GPT-3，可以通过提供一个结构化文件来进行训练，文件中的每一行都是一组提示-补全 对。我们假设，GPT-3 模型能够通过示例隐式学习通常在 PDDL 领域文件中定义的规则，例如允许（建模）的动作及其相应的前置条件和效果。因此，我们仅使用问题-规划 对进行训练，并将问题用作提示、将规划方案用作补全。需要指出的是，提示中只使用了部分 PDDL 问题内容，因为许多谓词实际上是非功能性的，也就是说，它们用于编码问题的一般属性，例如关节角配置之间的相对顺序。举例来说，关节处于 45度 时可以顺时针旋转至 60度，也可以逆时针旋转至 30度

每次以 15度 为增量。这些谓词在所有问题中都保持不变，也不会被动作效果修改。因此，我们假设模型能够像学习领域信息一样，从规划方案中编码的隐含信息中学习这些谓词。于是，我们将它们从提示中移除，以缩短提示长度；提示长度会影响响应时间，而且对于所选模型，每次查询的提示与补全总长度不得超过 2048 个词元，约相当于 8000 个字符。这样便可以生成更长的规划方案，避免截断并提升性能。

在进行实验时，OpenAI 的文档建议使用名为 davinci 的最大、最强模型来进行条件生成，即根据输入生成原创文本。针对这一用例，文档建议使用至少 500 个训练样本训练模型，并以 8000 个为目标。Teriyaki 的优势之一在于，我们可以使用随机生成的问题和基于 PDDL 的行动规划器，轻松生成大型训练数据集。因此，我们生成了 9000 个问题，其中 8000 个用于训练，1000 个用于验证。需要注意的是，在规划器执行的 9000 次规划尝试中，有 124 次失败，因此验证集减少为 876 个样本。最后，我们又添加了 1000 个问题-计划对作为测试集，验证了该测试集中的每个样本均可由 Probe 求解，并确保没有任何问题实例与训练集和验证集中已有的问题实例重复，从而避免数据集之间发生交叉污染。

下一步需要验证 Probe 生成的规划方案是否完整，以确保我们只使用正确的数据训练模型。为此，我们使用广为人知的 VAL 验证套件 [Howey 等人，2004]，该套件要求规划方案采用国际规划竞赛（IPC）标准格式[3]。由于 Probe 返回的规划方案不符合这一标准，我们编写了自定义代码，将规划文件重新格式化，使其与 VAL 兼容。这也有助于确保 Teriyaki 使用符合 IPC 标准的规划方案进行训练，从而在之后以同样的方式响应我们的查询，让我们能够轻松评估其规划有效性。使用 VAL 检查数据集后，所有规划方案都通过了测试，这意味着它们都达到了预期的目标状态。不过需要指出的是，Probe 此前在 9000 个问题中有 124 个完全未能生成规划方案，约占总数的 1.37%。

[3] 网页：https://ipc2023.github.io/

最后，我们以 JSONL 格式编制了训练数据集和验证数据集。数据集文件的每一行都由一个提示（即问题）和一个补全（即规划方案）组成。我们在每个提示和补全的末尾分别添加了相应的终止序列，即 \n\n###\n\n 和 END。顾名思义，终止序列会向网络标示消息的结束位置。这对补全尤其重要，因为在查询模型时，终止序列可用作停止条件。清单 1.1 给出了一行样例，为简洁明了起见，内容经过了编辑。图 2 概述了整个数据生成与数据集准备流程。

需要明确的是，我们的研究所使用的生成数据不应被视为带有「不够真实」这一负面含义的合成数据。鉴于我们的研究重点是符号任务规划，并假设机器人能够完美地进行感知和基础关联，那么从规划器的角度来看，这些数据与真实数据无从区分。因此，在本研究中，我们仅使用生成数据进行训练和分析，因为这些数据能够为规划模块提供同等的验证效果，同时不会引入感知和执行模块带来的差异和偏差。

```
{" 提示 ":
"(: init (angle_joint              angle315 joint1 )
( 角度_关节角度300 关节2)
( 角度_关节 角度285 关节3)
(位于中心关节2)
( 自由   gleft ) (自由        gright ))
(: 目标 (且
( 角度_关节 角度0 关节1)
( 角度_关节 角度300 关节2)
( 角度_关节         角度285 关节3 )))
\n\n ###\ n\n"   ,
" 补全 ":
"   0.00100:    ( 连杆 -到 - 中央 - 抓取            . . . )
0.00300:    ( 增加_角度_第一个_子项 _45              . . . )
0.00500:    ( 释放 - 连杆          . . . )
0.00700:    ( link -to - central - 抓取        . . . )
0.00900:     （减少_角度_第一个_子项 _45             . . . )
END "}
```

清单 1.1. 训练样本示例。

3.3 训练

关于训练过程，我们决定在 MACRO领域上进行完整的 8000 个样本训练，因为相关数据集中的规划方案平均较短，因此理论上更容易被 GPT-3 学习。我们将这个模型称为 Teriyaki-MACRO。

当我们训练大语言模型执行动作规划时，与生成过程相关的成本函数奖励的是语言连贯性，而非规划有效性。这意味着，我们假设语言连贯性可以作为逻辑连贯性的替代指标，因此必须假定，训练过程中的验证与最终模型的规划有效性有所不同。这里，“规划有效性”指规划方案中满足以下条件的比例：（i）形式上正确；（ii）能够达到期望的目标状态，或与之相容的状态。所谓“形式上正确”，是指每个动作的参数既符合规划领域给出的规范，也符合执行该动作时建模现实的状态；换言之，动作前提条件未遭违反。为了衡量规划有效性如何随所提供训练集的规模扩大而提高，我们决定分阶段进行训练。每个阶段都通过提供样本，将训练集总规模扩大一倍。从最低的 500 个样本开始，随后使用总计 1000、2000、4000 和 8000 个样本训练系统，并在每个阶段保存训练后模型的快照。

如上文所预期，训练过程选用的基础模型是 GPT-3，具体为 text-davinci-002。关于超参数，训练轮次设置为 2，而批次大小和学习率乘数则保留默认值。批次大小默认为训练集样本数量的 0.2%，默认学习率则根据最终批次大小，在 0.05 至 0.2 的范围内自动确定。我们还要强调，由于该模型实际上进行了五次微调，每次训练开始时都会将学习率重置为默认值，其效果类似于带热重启的梯度下降 [Loshchilov 和 Hutter, 2016]。我们确实提供了可选的验证集，并启用了 compute_classification_metrics 选项，以获取训练统计数据供进一步分析。训练时（01/12/2022），在单个规划领域上执行这一过程的总训练成本约为 250$。

3.4 迁移学习

对于 NO-MACRO 域，我们采用了相同的方法，但决定生成两个候选模型。第一个模型与之前一样，以 text-davinci-002 模型为起点进行训练；第二个模型则以此前获得的 MACRO 模型为起点进行训练。我们将这两个模型分别称为 Teriyaki-NO-MACRO（davinci）和 Teriyaki-NO-MACRO（MACRO）。其假设是，由于各领域共享许多概念，而 MACRO 模型已经针对这些概念进行了预训练，第二个候选模型应当能够以更少的训练样本达到更高的规划有效性。

第 4.2 节所示的结果证实了这一假设，因此，当 Teriyaki-NO-MACRO（MACRO）取得与其母模型相当的结果时，我们便中止了训练，并彻底弃用了 Teriyaki-NO-MACRO （ davinci ）。关于训练数据集，与清单 1.1 中的示例相比，唯一的区别是提示部分前面加上了 \n--NO-MACRO 标签。引入该标签是为了测试，只需在向系统发送的查询中添加该标签，模型是否就能同时解决 MACRO 和 NO-MACRO 领域的问题。不幸的是，NO-MACRO 模型失去了在 MACRO 领域解决问题的能力，这表明，若要生成能够解决多个领域问题的模型，训练时应包含所有感兴趣领域的示例。

3.5 调用大语言模型并流式传输规划

训练阶段结束后，可以通过 API 调用查询模型，方法是将描述机器人和关节物体初始状态与目标状态的 PDDL 谓词作为提示提供，就像使用传统的符号化兼容 PDDL 的动作规划器一样。可以配置多个参数，下面列出了其中会影响最终规划整体质量的参数。

– 温度是最重要的参数，因为它决定了回复（进而也决定了规划）的确定性。将其设为 0，相当于对下一个词元的概率执行 argmax 操作，并确保我们始终获得最可能的结果。虽然在常规文本生成中，适度的创造性是可取的，但我们强烈建议在生成规划时将其设为 0，尤其是在机器人动作规划涉及人机协作场景时。

– presence_penalty 和 frequency_penalty 的取值范围为 -2 到 2，可用于奖励或惩罚生成文本中的重复内容。我们发现，将前者设为 2 似乎能使规划有效性提高 $1-2\%$，反之亦然；但目前我们对这一效果的研究还不够充分，因此决定在测试中将两者的默认值都设为 0。

– stop 可用于将某个字符串设为停止条件，也就是说，当模型生成该字符串时，生成会立即终止。为与我们的训练数据集保持一致，我们将此值设为 END。– max_tokens 控制响应的最大长度，因此我们建议尽可能将其设高，以尽量降低规划被截断的可能性。由于每个模型的提示与补全总长度都有上限，而最坏情况下的提示长度取决于具体的规划领域，因此该值应根据具体情况逐一评估。就我们的情况而言，1900 似乎是确保稳定运行的最高值。

要流式传输规划，我们可以将参数 stream 设为 true。这样，模型一旦生成词元，就会立即开始返回，而不必等待整个词元序列生成完毕。这不会影响接收完整答案所需的时间，但会显著缩短接收第一个动作所需的时间，也就是响应时间。

3.6 同步规划与执行的架构

在传统的感知-规划-执行数据流中，系统按以下步骤运行：(i) 首先通过内生传感器和外生传感器收集有关机器人状态及其工作空间的数据；(ii) 将这些数据映射为符号，以表示当前状态；(iii) 生成从当前状态到达给定目标情境的端到端规划；(iv) 开始依次执行所有规划动作，并可能监控其结果；如遇到意外干扰、错误，或一般情况下预期状态与当前状态之间出现任何不一致，则停止执行。

借助 Teriyaki，并在 [Capitanelli 等人 2018],工作的基础上，可以如图 3 所示修改这一简单的数据流。虽然步骤 (i) 和步骤 (ii) 保持不变，但现在数据驱动合成规划器可以在模型生成一个动作后立即将其输出。只需将规划器生成的每一

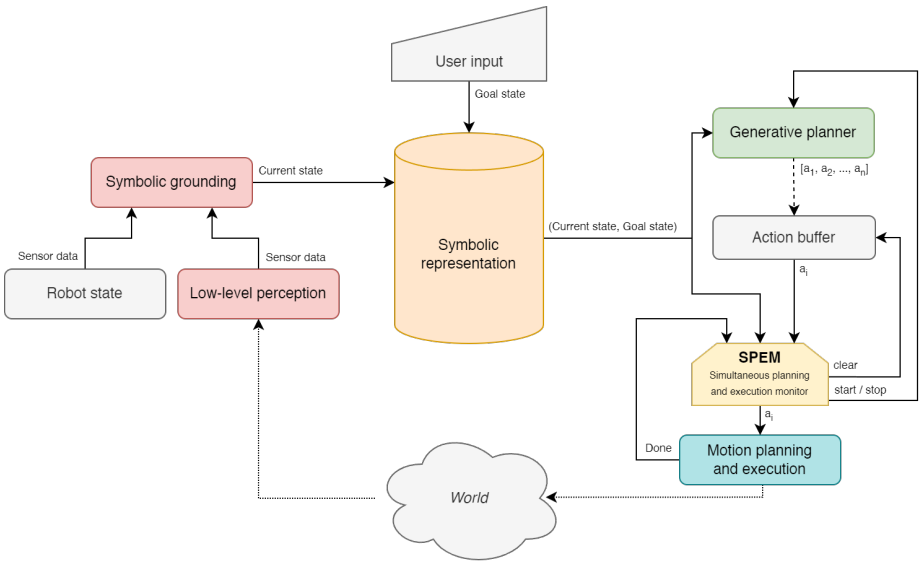

图3。 一种整合生成式 Teriyaki 规划器以实现同步规划与执行的可能架构。规划器下方的虚线表示动作逐个生成，然后缓存在动作缓冲区中等待执行。SPEM 模块会在提交每个动作执行之前，检查其前置条件是否得到满足。

行都视为一个动作，即可实现这一点。由于规划的生成通常无法暂停，动作会存储在先进先出缓冲区中。

为了监控每个动作的执行过程，我们引入了一个专用模块，称为同步规划与执行监控器（SPEM）。每个动作执行完毕后，SPEM 会验证两个条件：（a）检查总体目标是否在上一个动作执行期间发生变化，例如由于人的新动作而发生变化；（b）考虑最新的当前状态，并使用 VAL验证工具验证该状态是否符合下一个动作的前置条件。如果两个条件均满足，SPEM 就会将下一个动作提交给运动规划与执行；否则，它会重置规划器并清空动作缓冲区。如果发生这种情况，则可以结合最新的当前状态和目标状态重新开始规划。

条件 (b) 至关重要，因为规划器会根据首次初始化时可用的状态，继续逐个动作地生成规划。然而，当前状态会持续变化，既可能如预期般因动作执行而改变，也可能因无法预见的干扰和人类行为而发生意外变化——

在后一种情况下，下一个动作可能需要规划器认为已满足、但实际上已不再满足的前置条件。因此，SPEM 既能防止灾难性故障，也能从干扰中无缝恢复，或适应人类行为。考虑到通过 Teriyaki 工作流生成的规划器可以逐个动作地生成规划，这一恢复过程只需花费生成单个动作所需的时间。反之，在传统的感知-规划-执行数据流中，系统必须暂停执行，直到传统规划器生成完整规划为止。

需要说明两点。首先，如果我们每次只生成一个动作，然后在更新对世界的认知后，以相同的目标状态重新启动规划器，那么就不需要像 SPEM 这样的模块。遗憾的是，在我们的测试中，这种操作模式无法生成合理的规划方案，因为规划器会丢失先前已执行动作的上下文，进而很容易陷入循环，例如反复将关节物体围绕某个关节居中，然后又将其围绕另一个关节居中，如此往复。这种现象在概念上与著名的 Sussman 异常 [Sussman,1973],类似，而 SPEM 中编码的功能可以彻底解决这一问题。其次，从概念上说，这一工作流程仍然是传统的感知-规划-行动架构。事实上，将这种机器人控制架构并排与 Capitanelli 等人提出的架构进行比较， [2018], 机器人的实际行为难以区分，唯一的差异是每当需要重新规划时，等待时间有所缩短。因此，在第4.4节中，我们决定不展示执行用时，而是着重比较传统规划器与基于 Teriyaki 的规划器在执行前的等待时间。

4 结果

4.1 随着训练样本数量增加，词元与规划有效性之间的关系

在第3.3节中，我们指出使用语言连贯性作为逻辑连贯性的替代指标。图4比较了MACRO 模型的验证标记准确率和规划有效性随用于训练模型本身的样本数量的变化。一方面，验证标记准确率衡量的是预测下一个词元的准确率，也就是在我们的案例中，约为响应中的下4个字符，与

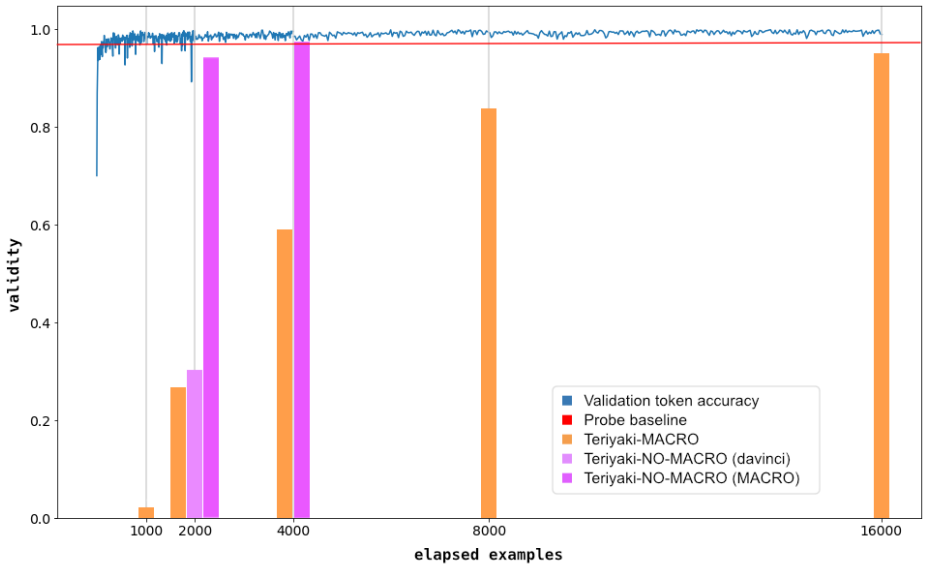

图 4. 随着训练示例数量增加，标记验证准确率和规划有效性的变化。蓝线表示学习过程中验证词元准确率的变化情况，该数据由 GPT-3 分类指标报告。柱形图表示 Teriyaki-MACRO、Teriyaki-NO-MACRO（davinci）和 Teriyaki-NO-MACRO（MACRO）模型的规划有效性。红线表示我们的基线，即由 Probe 求解的规划方案所占的百分比。

规划相对应，在验证集上进行评估。另一方面，这里所说的规划有效性，是指876个样本的验证集中同时形式上正确且能够达到所需目标状态的规划方案所占的百分比，也就是说，通过VAL验证工具测试的规划方案所占的比例。对于前者，我们从GPT-3自身在训练后报告的分类指标中获取数据；在信息可用的情况下，这些数据每8个训练步骤记录一次。对于后者，我们使用了模型在训练500、1000、2000、4000和8000个样本后保存的快照。这些快照按照第3.5节所述的条件，针对验证集中的全部876个问题进行规划。随后对得到的规划方案运行VAL，最终即可计算规划有效性。在图4中，规划有效性的变化用橙色柱表示，以突出显示该数值是在每个快照处测量的。还必须指出，参数elapsed_examples并不对应训练集中的唯一样本数量，而是乘以2进行了缩放，因为我们使用了两个训练轮次，因此每个样本都被使用了两次。

在图4中，橙色柱形表示 Teriyaki-MACRO 模型的验证准确率。正如预期，GPT-3 在最初的 500 个样本之后便达到了很高的验证标记准确率，这体现了该模型广为人知的少样本学习能力。然而，规划有效性的提升速度要慢得多。与少样本方法类似，大语言模型尚无法准确映射到可行动作，即使只错一个词元，也可能破坏原本在语言上连贯的规划。仅使用 500 个样本训练的模型生成规划方案时，一个非常常见的错误是忽略动作的条件效果。正如我们正确推测的那样，条件效果颇具挑战性，因为为了保持规划整体的语义连贯性，必须将其效果传播到规划后续步骤中。在这种情况下，动作本身正确，参数设置也基本合理，但由于忽略了某个关节的状态与模型预期不同（这是先前规划动作的间接效果所致），这些动作未满足必要的前置条件。最终，模型在使用 8000 个独有样本训练后，在验证集上的规划有效性达到很高的 95%。尽管如此，模型似乎并未对用于训练的问题过拟合，这一点可从第4.3节所示的测试集结果看出。因此，扩大训练数据集或增加训练轮次或许能进一步提高规划有效性。

规划有效性的结果与 Plansformer [Pallagani 等人， 2023], 取得的结果惊人地相似，但前者使用的训练集规模为 8000，而非 18000，训练轮次为 2，而非 3。值得注意的是，我们将 Plansformer 视为 Teriyaki 的一个可能基线，因为这两种方法在概念上有所关联，即都通过微调过程，利用特定示例来调整基础模型。还应指出，尽管目标相似，Plansformer 和 Teriyaki 旨在展示整体流程中的不同特点：Plansformer 专注于微调过程，旨在证明其能够适应不同的规划领域；而 Teriyaki 则着重展示如何通过数据驱动的基于大语言模型的规划器，处理现实世界机器人应用中出现的复杂规划领域，即高级规划概念和更长的问题-计划对。虽然我们的结果可能表明 GPT-3 在这项任务中的表现优于 CodeT5，但必须强调，基础模型不同、所选的类PDDL规划器不同、使用的 PDDL 构造不同，以及提示策略不同，因此需要进行远超本文范围的分析。

在这项实验中，我们还统计了因规划方案过长而失败的规划尝试次数。尽管与验证集规模相比，这一数字始终很小，但失败次数从第一个训练步骤的 24 次降至最终模型的 1 次，这进一步表明模型生成较短规划方案并避免不必要动作的能力有所提升。

4.2 迁移学习

如第 3.4 节所述，我们从两个基础候选模型入手，选定了适用于 NO-MACRO 规划领域的 Teriyaki 求解器，分别是 Teriyaki-NO-MACRO（davinci）和 Teriyaki-NO-MACRO（MACRO）。训练这两个模型时，我们按照第 3.4 节所述跟踪了规划有效性。图4 中的浅粉色和深粉色柱分别表示 Teriyaki-NO-MACRO（davinci）和 Teriyaki-NO-MACRO（MACRO）模型的规划有效性结果。在 1000 个样本后，Teriyaki-NO-MACRO（davinci）模型的规划有效性达到 32.5%，而 Teriyaki-NO-MACRO（MACRO）达到 95.2 %。鉴于这一结果，我们立即舍弃了前一个模型，并继续使用 2000 个样本训练后一个模型。此时，Teriyaki-NO-MACRO 模型的验证规划有效性达到 98.8%，超过了 MACRO 模型在 8000 个样本后达到的 95% 验证规划有效性。鉴于这一结果，我们决定不再继续训练；从现在起，当我们简称 Teriyaki-NO-MACRO 时，实际指的是 Teriyaki-NO-MACRO（MACRO）。

这项实验有两个值得注意的方面。

首先，在 1000 个样本时，Teriyaki-NO-MACRO（davinci）模型的规划有效性仍高于 Teriyaki-MACRO。这一结果或许表明，与我们最初的假设相反，GPT-3 更容易学习 NO-MACRO 模型，而非 MACRO 模型。还需要对此结果进行更深入的探究，不过它可能与规划领域中动作的数量和质量有关。

其次，Teriyaki 框架中 MACRO 与 NO-MACRO 域之间的迁移学习实验，凸显了其适应其他知识表示的能力。不过值得注意的是，这两个领域所代表的都是同一项本质任务，即对关节物体进行协作操作，并且拥有相近的动作集合和状态空间。因此，

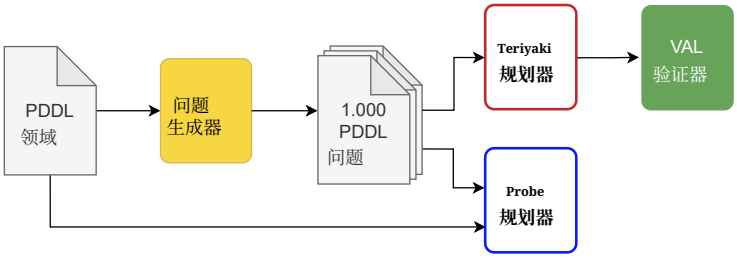

图 5. 第4.3节所述 Teriyaki 测试流程示意图。

在两个本质上不同的任务之间进行迁移学习的有效性，仍有待进一步探索。

4.3 求解器比较

为了避免交叉污染，我们在一个包含 1000 个此前未用于训练和验证的问题-计划对的测试集上，从规划有效性、计划长度和规划时间等方面测试了 Teriyaki-MACRO 和 Teriyaki-NO-MACRO 模型的性能，并将结果与传统动作规划器 Probe 的性能进行了比较。整个过程总结于图 5。

需要指出的是，由于运行 Teriyaki 和 Probe 所采用的计算工作流差异显著（前者基本上采用通过专有 API 访问的基于云的架构，后者则使用安装在本地计算机上的软件），因此无法在规划时间方面对二者进行公平比较。因此，本节中有关规划时间的结果仅用于为未来工作提供基线，并展示在本文考察的两个不同规划领域中，各 Teriyaki 规划器的规划时间如何呈现不同的扩展趋势。不过，Probe 仍然是一个重要的基线方法，因为它是用于获取 Teriyaki 训练数据的规划器；尽管 Probe 本身并非在计划质量方面的最优规划器，但 Teriyaki 在计划生成方面的整体性能理应与其结果相关。

Proberan 运行于 Ubuntu20.04.4 Windows LinuxSubsystem(WSL)环境中，部署在一台配备 8 核 Ryzen 3700X 3600 Mhz CPU 和 16GB@3600 Mhz RAM 的机器上。我们记录了规划器报告的规划时间，以及每个生成的规划方案。就规划有效性而言，

表1. 所测试模型及测试日期汇总

模型	基础模型 测试日期 开始 结束			
Teriyaki-MACRO	davinci	18/02/2023 11:17:56		13:47:41
Teriyaki-NO-MACRO	MACRO	2023年年19/02/2023 17:22:24		21:25:58

所有生成的规划方案均有效，但我们将规划器未能生成规划方案的情况视为失败。

对于 Teriyaki模型，我们使用第3.5节所述的设置向其发出提示，然后使用数据集生成阶段采用的同一 VAL验证工具，验证所得规划方案的有效性。由于 OpenAI API 未提供记录 API 调用确切执行时间的方法，因此我们通过禁用 GPT-3响应的流式传输选项，并记录每次 API 调用向客户端应用程序返回完整规划方案所耗费的时间，来测量 Teriyaki 求解器的规划时间。因此需要注意的是，在执行之前，每次调用在云端应用程序中排队了多长时间是无法确定的。我们认为排队的影响不可忽略：在 ChatGPT 正式发布后进行测试时，规划时间比此前记录的更长。ChatGPT 是 OpenAI 推出的另一款广受欢迎的 GPT-3 衍生产品，这种延长可能是由于服务器流量增加所致。为部分缓解这一影响，本文所述测试均在周末进行，并尽量安排在中欧时间上午。在表1中，我们还列出了每次测试会话的日期以及开始和结束时间，以供参考。

对于所有求解器和模型，计划长度均简单地计算为规划中的动作数量。表2在相应领域中对Teriyaki-MACRO和Teriyaki-NO-MACRO模型与Probe进行了比较，比较指标包括有效性（百分比）、平均计划长度、最长和平均规划时间，以及规划时间的标准差（以秒为单位）。

正如预期，在传统的感知-规划-行动工作流中，Probe 端到端生成规划方案的速度比经过训练的 Teriyaki 求解器更快。不过，Teriyaki 求解器仍然表现不俗。尽管 Teriyaki模型是在 Probe 生成的规划方案上训练的，但它们能够解决 Probe 无法处理的问题，甚至生成更短的规划方案。对于 Teriyaki-MACRO，计划长度的差异仅为 1.5%，但对于 Teriyaki-NO-MACRO，这一差异上升至 13.5%，后者通常会生成几乎长一倍的规划方案。这似乎表明，训练过程会奖励较短的补全，而且这种影响可能会

表2. Teriyaki模型与 Probe 在测试数据集各自领域中的比较

求解器	领域	准确率 [%]	步数 t	最大 [s] t	平均 [s]t	标准差 [s]
Teriyaki-MACRO	MACRO	95.5	10.953	54.32	8.99	4.77
Probe	MACRO	98.6	11.111	36.71	2.11	3.47
Teriyaki-NO-MACRO	NO-MACRO	94.0	19.158	54.71	14.61	7.16
Probe	NO-MACRO	98.6	22.137	43.77	2.79	3.30

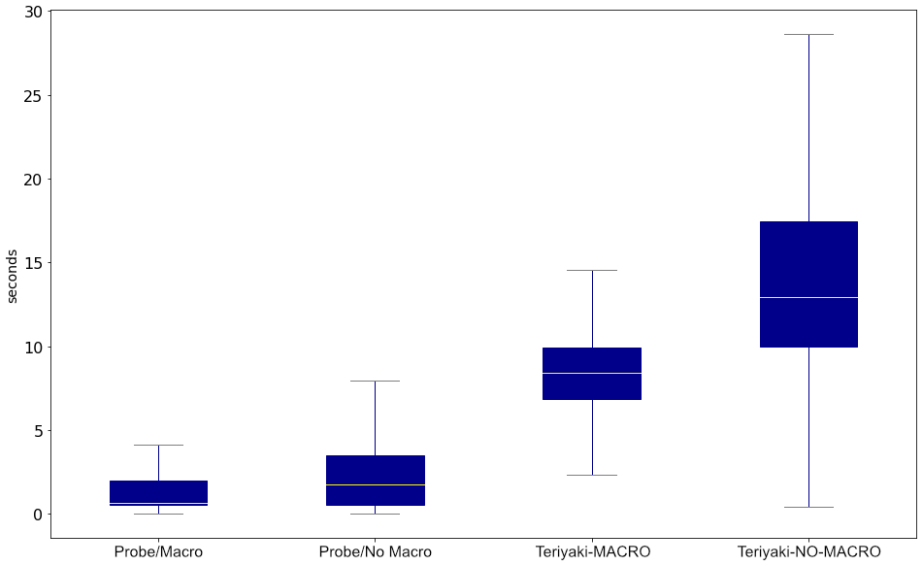

图 6. Teriyaki MACRO 和 NO-MACRO 模型在各自规划领域中与 Probe 的规划时间比较。

随着预期补全内容变长而变得更加明显。不过，这一现象仍需进行更系统的研究。

图6有助于更好地评估所提出的模型的计时性能，并将其与 Probe 生成的结果进行比较。可以观察到，正如假设的那样，Teriyaki模型确实会随着输入和输出总长度的增加而扩展。Teriyaki-NO-MACRO模型的规划时间约为Teriyaki-MACRO模型的两倍，这符合预期，因为前者的规划方案长度约为后者的两倍。与 Probe 的箱线图相比，图中与 Teriyaki模型相关的箱线图在两个

领域中都呈现出非常 distinct 的形状，这暗示规划时间几乎服从高斯分布。这与以下事实相一致：由随机初始化的问题生成的规划方案，其长度可以是任意的。

4.4 逐动作规划流式传输

使用大语言模型进行神经符号动作规划的一个主要优势是，规划方案可以在生成的同时进行流式传输。如果能够合理利用这一特性，就可以支持逐动作生成，而不必采用端到端的方法。在许多涉及使用机器人且要求具备灵活行为的应用中，尤其是在由于环境条件变化而预计需要频繁重新规划的情况下，通过并发进程（或进程集）实现同步规划与执行，可以大幅缩短等待规划方案可用并因此得以执行的等待时间。正如我们此前指出的，这可能与人机协作中感知到的交互流畅性形成良好的关联。

在第 3.6 节中，我们介绍了一种增强型感知-规划-行动架构，该架构集成了神经符号规划，以提高对干扰的反应能力和韧性，同时基本上能够产生与传统方法相同的规划方案输出。因此，本节将重点对等待时间进行定量比较。

为测试 Teriyaki 在这方面的性能，我们让 Teriyaki-MACRO 模型针对测试数据集进行规划，直到生成 1000 个动作。如第 3.2 节所述，我们生成的数据既能忠实地反映现实世界，又能支持大规模测试，同时避免受到机器人平台局限性所带来的偏差。

图 7 对比了 Teriyaki-MACRO 执行单个动作的耗时与 Probe 生成 1000 个完整规划方案的耗时。我们不仅发现平均等待时间减少了 61.2%，响应时间的标准差也从 3.47 降至仅 0.15。这相当于降低了 95.6%，使 Teriyaki 比传统的基于搜索的启发式求解器更具可预测性。

4.5 局限性与可扩展性

我们此前指出，尽管我们的结果相当有前景，因为它们让同步规划与执行成为可能，但

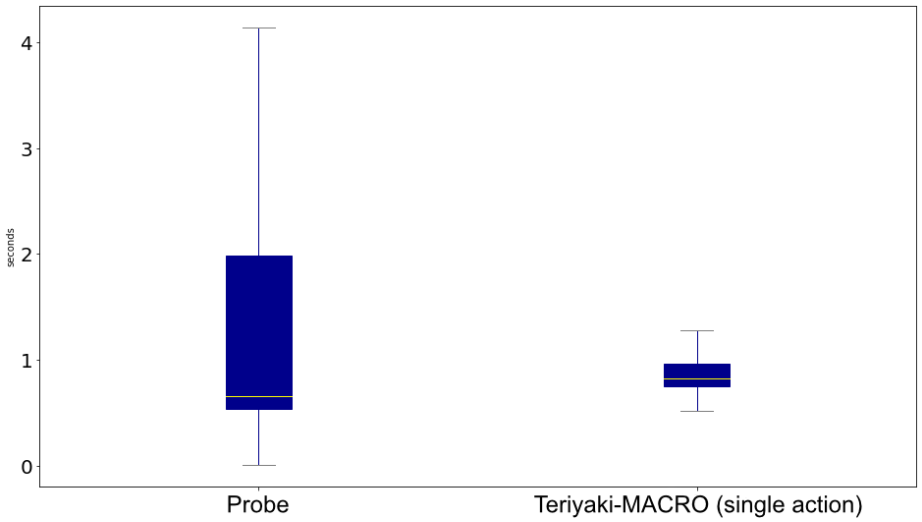

图 7. Teriyaki-MACRO 在 MACRO领域生成单个动作所需时间与 Probe 的比较。

在现阶段，Teriyaki 工作流所设想的基于数据的规划器合成还不能被视为传统动作规划的完整替代方案。事实上，要获得用于训练模型和合成求解器的问题-计划对数据，就需要传统的符号动作规划器。以下我们列出使用 Teriyaki 工作流目前面临的主要局限。

1. 与符号规划器不同，我们的方法具有特定领域性，需要进行训练。2. 在当前实现中，端到端规划时间比基线符号规划器 Probe 更长，至少 在使用 GPT-3 作为基础大语言模型来实现整体架构时是如此。3. 使用基于 Transformer的大语言模型时，问题—规划对的最大长度受模型上下文窗口的限制；对于 GPT-3，该窗口仅为2048个词元，即约8000个字符。4. 基于 Transformer的大语言模型在生成数学上准确的结果方面存在显著困难。因此，尽管我们纳入了条件效果等高级 PDDL 特性，但要将此方法扩展到其他相关能力（如数值规划和时间规划）可能较为困难。

尽管存在这些限制，但值得注意的是，在撰写本文时，已有许多不一定基于Transformer架构的大语言模型可供使用，它们或许能够帮助克服或减轻这些限制。例如，较新版本的 GPT（如 GPT-3.5-Turbo-16k 和 GPT-4）分别将上下文窗口扩展到 16.000 和 128.000 个词元，并且可能有效克服限制3），适用于所有实际的规划领域。更进一步，Mamba [Gu 和 Dao， 2023]等新型基于状态空间的架构可以用于实现完全不依赖上下文窗口的大语言模型，同时还能在长序列中展现更强的细节注意力，并实现更快的推理时间。

除了使用更早期的大语言模型外，进一步优化也能提升 Teriyaki 的有效性。具体而言，Teriyaki 中的规划时长受输入和输出的总长度影响，这意味着动作名称的长度会显著影响性能。如图6所示，采用简洁的动作命名规范，可以使 Teriyaki 的规划时间更接近 Probe 的表现。这凸显了在人类可读性与效率之间取得平衡的必要性。

关于可扩展性，仍有许多因素需要进一步探索。我们的研究以及 Pallagani 等人 [2023]开展的工作似乎都表明，足够大的模型和训练数据集，加上有效的训练，能够使大语言模型解决以 PDDL语法表示的通用规划领域，其中可能包含条件效果等某些高级元素。然而，目前的研究尚未确定规划领域的复杂程度与持续解决该领域问题所需的模型规模和训练数据集之间是否存在明确关联。据我们所知，主要挑战在于无法定量衡量规划领域的复杂性。至多只能根据两个规划领域的状态空间维度以及引发状态转移的动作数量进行比较。尽管如此，按照这些指标被认为更复杂 的领域，反而可能因其状态空间的其他特征以及规划器的具体情况而更容易解决。对于传统符号规划器，这取决于其启发式函数；而对于基于大语言模型的求解器，则取决于其底层神经网络架构。

鉴于本文讨论的良好性能扩展性，以及同步规划与执行所带来的可能性，我们认为基于大语言模型的神经符号

规划是一种很有前景、值得在未来进一步研究的方法。

5 结论

本文介绍了 Teriyaki，这是一种基于 GPT-3 生成神经符号、符合 PDDL 规范的规划器的框架。在当前实现中，Teriyaki 依赖一种实用且成本低廉的训练数据集生成流程，该流程基于使用现有的动作规划器，根据随机定义的输入生成规划方案。当前实现、用于训练和验证的数据集，以及用于从符合 PDDL 规范的规划器生成全部训练材料的辅助代码，均已开源。训练可以利用云端的高性能计算设备，生成的模型则可部署到任何采用标准 PDDL 兼容语法和接口的机器人软件架构中。

本文的主要贡献在于提供了实证证据，表明这一方法可以成为动作规划的可行解决方案，同时也对整体工作流程进行了具体的概念验证。具体而言，我们证明了：(i) 规划有效性与传统的先进水平 PDDL 兼容规划器相当；(ii) 平均计划长度最多缩短 13.5%；(iii) 在机器人必须频繁重新规划的场景中，可以更好地使用 Teriyaki；特别是由于它能够逐步生成规划方案，从而将规划的平均等待时间减少 61.2%，标准差减少 95.6%。

总体而言，这些结果暗示了一种情形：该方法在潜在支持的规划领域数量方面可以较好地扩展，尽管这仍是有待进一步研究的方面之一；同时，人机交互的流畅性也能从机器人动作规划能力所带来的更短反应时间中大幅受益。

利益冲突声明

作者声明，本研究是在不存在任何可能被视为潜在利益冲突的商业或财务关系的情况下开展的。

作者贡献

AC：概念构思、数据整理、软件、调查研究、撰写初稿。FM：监督、审阅与编辑。

致谢

我们衷心感谢哈德斯菲尔德大学的 Mauro Vallati 教授在任务规划方面给予的指导，并感谢他与我们合作定义了本研究所使用的 PDDL 领域。

数据可用性声明

本研究生成的代码和数据集可在以下代码仓库中找到：
https://github.com/alessiocpt/teriyaki。

参考文献

航空制造公司、Adele Howe、Craig Knoblock、ISI 的 Drew McDer- mott、Ashwin Ram、Manuela Veloso、Daniel Weld、SRI 的 David Wilkins、An- thony Barrett、Dave Christianson 等。PDDL——规划领域定义语言。技术报告，技术报告，1998年。Riccardo Bertolucci、Alessio Capitanelli、Marco Maratea、Fulvio Mastrogiovanni 和 Mauro Vallati。利用重力对三维关节物体进行操作的自动规划编码。载于人工智能**IA 2019*——人工智能进展：意大利人工智能协会第十八届国际会议，意大利伦德，*2019*年*11*月*19*日至*22*日，论文集 *18*，第135–150页。施普林格，2019年。Riccardo Bertolucci、Alessio Capitanelli、Carmine Dodaro、Nicola Leone、Marco Maratea、Fulvio Mastrogiovanni 和 Mauro Vallati。使用答案集编程通过双臂机器人操作关节物体。《逻辑编程理论与实践》，21(3):372–401，2021年。Tom Brown、Benjamin Mann、Nick Ryder、Melanie Subbiah、Jared D Ka- plan、Prafulla Dhariwal、Arvind Neelakantan、Pranav Shyam、Girish Sastry、Amanda Askell 等。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。Alessio Capitanelli、Marco Maratea、Fulvio Mastrogiovanni 和 Mauro Vallati。人机协作场景中的关节物体操作。《机器人与自主系统》，109:139–155，2018年。Alessandro Carf´ı、Francesco Foglino、Barbara Bruno 和 Fulvio Mastrogiovanni。用于人际交接的多传感器数据集。《数据简报》，22:119–117，2019年。Michael Cashmore、Maria Fox、Derek Long、Daniele Magazzeni、Bram Ridder、Arnau Carrera、Narcis Palomeras、Natalia Hurtos 和 Marc Carreras。Ros-plan：在机器人操作系统中进行规划。载于自动规划与调度国际会议论文集，第25卷，第333–341页，2015年。Mark Chen、Jerry Tworek、Heewoo Jun、Qiming Yuan、Henrique Ponde de Oliveira Pinto、Jared Kaplan、Harri Edwards、Yuri Burda、Nicholas Joseph、Greg Brockman 等。评估在代码上训练的大语言模型。*arXiv* 预印本 *arXiv:2107.03374*，2021年。

Zhenfang Chen、Rui Sun、Wenjun Liu、Yining Hong 和 Chuang Gan。GENOME：通过扩展和复用模块实现生成式神经符号视觉推理。*arXiv*预印本 *arXiv:2311.04901*，2023年。Aakanksha Chowdhery、Sharan Narang、Jacob Devlin、Maarten Bosma、Gaurav Mishra、Adam Roberts、Paul Barham、Hyung Won Chung、Charles Sutton、Sebastian Gehrmann 等。PALM：利用路径扩展语言建模。*arXiv*预印本 *arXiv:2204.02311*，2022年。Robert Dale。GPT-3：它有什么用？自然语言工程，27(1)：113–118，2021年。Kourosh Darvish、Francesco Wanderlingh、Barbara Bruno、Enrico Simetti、Fulvio Mastrogiovanni 和 Giuseppe Casalino。适用于辅助车间任务的灵活人机协作模型。机电一体化，51：97–115，2018年。Kourosh Darvish、Enrico Simetti、Fulvio Mastrogiovanni 和 Giuseppe Casalino。人机协作过程的分层架构。*IEEE*机器人学汇刊，37(2)：567–586，2021年。Danny Driess、Fei Xia、Mehdi SM Sajjadi、Corey Lynch、Aakanksha Chowdhery、Brian Ichter、Ayzaan Wahid、Jonathan Tompson、Quan Vuong、Tianhe Yu 等。PaLM-E：具身多模态语言模型。*arXiv*预印本 *arXiv:2303.03378*，2023年。Artur d'Avila Garcez 和 Luis C Lamb。神经符号人工智能：第三次浪潮。*arXiv*预印本 *arXiv:2012.05876*，2020年。C. R. Garrett、T. Lozano-Perez 和 L. P. Kaelbing。PDDLStream：将符号规划器与黑盒采样器相结合，实现乐观的自适应规划。载于第*30*届国际自动规划与调度会议（*ICAPS*）论文集，地球任意地点，2020年10月。Albert Gu 和 Tri Dao。Mamba：利用选择性状态空间进行线性时间序列建模。*arXiv*预印本 *arXiv:2312.00752*，2023年。William Grant Hatcher 和 Wei Yu。深度学习综述：平台、应用与新兴研究趋势。*IEEE Access*，6：24411–24432，2018年。Malte Helmert。Fast Downward规划系统。人工智能研究期刊，26：191–246，2006年。Clint Heyer。人机交互与未来工业机器人应用。载于*2010*年*IEEE/RSJ*智能机器人与系统国际会议，第4749–4754页。IEEE，2010年。Guy Hoffman。评估人机协作中的流畅性。*IEEE*人机系统汇刊，49(3)：209–218，2019年。

Richard Howey、Derek Long 和 Maria Fox。VAL：使用PDDL实现自动规划验证、连续效应和混合主动规划。载于第*16*届*IEEE*人工智能工具国际会议，第294–301页。IEEE，2004年。Wenlong Huang、Pieter Abbeel、Deepak Pathak 和 Igor Mordatch。将语言模型用作零样本规划器：为具身智能体提取可执行知识。载于国际机器学习会议，第9118–9147页。PMLR，2022年。Nir Lipovetzky 和 Hector Geffner。通过精心设计的探测寻找规划方案。载于国际自动规划与调度会议论文集，第21卷，第154–161页，2011年。Lajanugen Logeswaran、Yao Fu、Moontae Lee 和 Honglak Lee。利用语言模型进行少样本子目标规划。*arXiv*预印本 *arXiv:2205.14288*，2022年。Ilya Loshchilov 和 Frank Hutter。SGDR：带热重启的随机梯度下降。*arXiv*预印本 *arXiv:1608.03983*，2016年。Simone Macci´o、Alessandro Carf´ı 和 Fulvio Mastrogiovanni。服务移动机器人技术中的层次规划系统。载于*2022*年*IEEE*国际机器人与自动化会议，2022年。Fulvio Mastrogiovanni、Antonio Sgorbissa 和 Renato Zaccaria。服务移动机器人技术中的层次规划系统。载于第*8*届智能自主系统国际会议，2004年。ˇ Tom´aˇs Mikolov、Stefan Kombrink、Luk´aˇs Burget、Jan Cernock`y 和 Sanjeev Khudanpur。循环神经网络语言模型的扩展。载于*2011*年*IEEE*声学、语音与信号处理国际会议（*ICASSP*），第5528–5531页。IEEE，2011年。Prajval Kumar Murali、Kourosh Darvish 和 Fulvio Mastrogiovanni。在制造环境中部署和评估一种基于与或图的灵活人机协作模型。智能服务机器人，13：439–457，2020年。Achraf Oussidi 和 Azeddine Elhassouny。深度生成模型：综述。载于*2018*年智能系统与计算机视觉国际会议（*ISCV*），第1–8页。IEEE，2018年。Vishal Pallagani、Bharath Muppasani、Biplav Srivastava、Francesca Rossi、Lior Horesh、Keerthiram Murugesan、Andrea Loreggia、Francesco Fabiano、Rony Joseph、Yathin Kethepalli 等。Plansformer工具：演示如何使用Transformer生成符号规划方案。载于*IJCAI*，第2023卷，第7158–7162页。国际人工智能联合会议，2023年。

Teven Le Scao、Angela Fan、Christopher Akiki、Ellie Pavlick、Suzana Ili´c、Daniel Hesslow、Roman Castagn´e、Alexandra Sasha Luccioni、Fran¸cois Yvon、Matthias Gall´e 等人。BLOOM：一款拥有 1760 亿参数的开放获取多语言语言模型。*arXiv* 预印本 *arXiv:2211.05100*，2022年。Tom Silver、Varun Hariprasad、Reece S Shuttleworth、Nishanth Kumar、Tom´as Lozano-P´erez 和 Leslie Pack Kaelbling。使用预训练大语言模型进行 PDDL 规划。载于*NeurIPS 2022*决策基础模型研讨会，2022年。Tom Silver、Soham Dan、Kavitha Srinivas、Joshua B Tenenbaum、Leslie Pack Kaelbling 和 Michael Katz。使用预训练大语言模型在 PDDL领域中进行广义规划。*arXiv* 预印本 *arXiv:2305.11014*，2023年。Ishika Singh、Valts Blukis、Arsalan Mousavian、Ankit Goyal、Danfei Xu、Jonathan Tremblay、Dieter Fox、Jesse Thomason 和 Animesh Garg。Prog-prompt：使用大语言模型生成情境化机器人任务规划。载于*2023*年*IEEE*国际机器人与自动化会议（*ICRA*），第11523–11530页。IEEE，2023年。Shaden Smith、Mostofa Patwary、Brandon Norick、Patrick LeGresley、Samyam Rajbhandari、Jared Casper、Zhun Liu、Shrimai Prabhu moye、George Zerveas、Vijay Korthikanti 等人。使用 DeepSpeed 和 Megatron 训练 Megatron-Turing NLG 530B，这是一种大规模生成式语言模型。*arXiv* 预印本 *arXiv:2201.11990*，2022年。Chan Hee Song、Jiaman Wu、Clayton Washington、Brian M Sadler、Wei-Lun Chao 和 Yu Su。LLM-Planner：借助大语言模型为具身智能体进行少样本落地规划。载于*IEEE/CVF*国际计算机视觉会议论文集，第2998–3009页，2023年。Martin Sundermeyer、Ralf Schl¨uter 和 Hermann Ney。用于语言建模的 LSTM神经网络。载于国际语音通信协会第十三届年会，2012年。Gerald J Sussman。技能习得的计算模型。1973年。Romal Thoppilan、Daniel De Freitas、Jamie Hall、Noam Shazeer、Apoorv Kul-shreshtha、Heng-Tze Cheng、Alicia Jin、Taylor Bos、Leslie Baker、Yu Du 等人。LaMDA：面向对话应用的语言模型。*arXiv* 预印本 *arXiv:2201.08239*，2022年。Hugo Touvron、Louis Martin、Kevin Stone、Peter Albert、Amjad Almahairi、Yas mine Babaei、Nikolay Bashlykov、Soumya Batra、Prajjwal Bhargava、Shruti Bhosale 等人。Llama 2：开放基础模型与微调聊天模型。*arXiv* 预印本 *arXiv:2307.09288*，2023年。

Karthik Valmeekam、Alberto Olmo、Sarath Sreedharan 和 Subbarao Kamb-hampati。大语言模型仍然无法规划（针对大语言模型在规划和变化推理方面的基准测试）。*arXiv* 预印本 *arXiv:2206.10498*，2022年。Ashish Vaswani、Noam Shazeer、Niki Parmar、Jakob Uszkoreit、Llion Jones、Aidan N Gomez、 -Lukasz Kaiser 和 Illia Polosukhin。注意力机制就是你所需要的一切。神经信息处理系统进展，第30卷，2017年。Naoki Wake、Atsushi Kanehira、Kazuhiro Sasabuchi、Jun Takamatsu 和 Katsushi Ikeuchi。ChatGPT助力在各种环境中实现长步机器人控制：案例应用。*arXiv* 预印本 *arXiv:2304.03893*，2023年。Bailin Wang、Zi Wang、Xuezhi Wang、Yuan Cao、Rif A Saurous 和 Yoon Kim。利用大语言模型进行领域特定语言生成的语法提示。*arXiv* 预印本 *arXiv:2305.19234*，2023年。Yue Wang、Weishi Wang、Shafiq Joty 和 Steven CH Hoi。CodeT5：具备标识符感知能力的统一预训练编码器-解码器模型，用于代码理解与生成。*arXiv* 预印本 *arXiv:2109.00859*，2021年。