PlanBench：一个用于评估大型语言模型规划和变化推理能力的可扩展基准

摘要

生成行动计划和推理变化一直被认为是智能体核心能力。因此，评估大型语言模型(LLMs)的规划和推理能力已成为研究的热门话题。然而，大多数关于 LLM 规划能力的说法都是基于常识任务的——在这种情况下，很难判断 LLM 是在进行规划还是仅仅从其庞大的世界知识中检索信息。迫切需要系统且可扩展的规划基准，这些基准具有足够的差异性来评估 LLM 是否具有天生的规划能力。受此启发，我们提出了 PlanBench，一个基于自动规划社区（特别是国际规划竞赛）中使用的领域类型的可扩展基准套件，用于测试大型语言模型在规划或推理行动和变化方面的能力。PlanBench 在任务领域和特定规划能力方面提供了足够的差异性。我们的研究还表明，在许多关键能力（包括计划生成）方面，即使使用最先进的模型，LLM 的性能也远远不足。因此，PlanBench 可以作为衡量 LLM 在规划和推理方面的进展的有用指标。

关键词： Large Language Models, Planning

1 引言

大型预训练语言模型的出现彻底改变了自然语言处理领域，并引起了公众的广泛关注。这些基于 Transformer 的大型语言模型 (LLM) 目前在许多标准 NLP 任务中提供了最先进的性能。大型语言模型 (LLM) 本质上是根据特定上下文预测句子中的下一个词，这些模型最初是为执行词序列完成任务而开发的。近年来，有轶事证据和说法表明它们拥有其他与序列完成无关的能力。这导致了突然爆发的研究，探究和研究它们的行[F]，就好像它们是人工生物一样（参见 [11]）。在本文中，我们特别关注一类研究工作，它们调查（并展示）大型语言模型的推理能力，包括常识推理 [4,18,20]、逻辑推理 [19]，甚至道德推理 [10]。这些工作在很大程度上表明，LLM 确实能够进行这种推理 [2,13,22]。

规划是人工智能领域中一个得到充分研究的推理任务。在最基本的形式中，规划涉及到制定一系列行动（策略），当执行这些行动时，将使智能体从某个初始状态转移到期望的世界状态。规划通常被主要研究为关于世界和奖励模型的推理问题。这些模型可以由人类指定，也可以由智能体通过与世界交互来学习。在本文中，我们想研究大型语言模型在涉及常识规划任务的动作和变化推理方面的能力。我们提出 PlanBench，一个基于自动规划社区中使用的领域（特别是在国际规划竞赛 (IPC) [9] 中）的可扩展基准套件，用于测试这一点。我们关注规划不仅是因为它是人类智能的核心方面 [17]，而且因为它也是自动代码生成等被认为是大型语言模型潜在应用的任务所必需的。

本文的贡献是评估规划的课程，其中我们确定了一组相关的但不同的任务，这些任务对于智能体成功执行规划和推理行动至关重要，并介绍了一个框架来开发旨在自动生成每个任务的可能查询的代码。我们使用两个 IPC 领域初始化 PlanBench：Blocksworld 和 Logistics。我们还提供了这些领域的混淆版本，混淆方式包括误导性词语或随机字母数字字符串。总的来说，我们提供了约 26250 个提示作为我们数据集的一部分。我们也正在积极地将其他 IPC 领域和任务添加到基准测试中。该基准的更新版本，包括工具、数据集以及用于重现本文提示和结果的脚本，可在 https://github.com/karthikv792/LLMs-Planning 找到。

2 相关工作

据我们所知，我们是第一个专门设计用于评估大型语言模型（LLM）新兴规划能力（如果有）的基准测试。我们的基准测试的初始版本在近一年前公开发布，目前正在被各种研究人员使用 [2]。我们已经从初始版本中显著扩展了基准测试的范围（增加了测试用例和领域）。但是，开发基准来评估 LLM 的新兴属性的想法本身并不新鲜。一些著名的现有推理基准包括 BIG-BENCH [19]、GSM8K [3]、AQUA [14]、SVAMP [15]、CommonsenseQA [20] 和 StrategyQA [4]。然而，这些任务很简单，涉及浅层推理，无法洞察它们的规划能力。由于大型语言模型 (LLM) 在此类任务中表现出色，人们对其规划能力的胜利论调越来越多，这在社区中也得到了呼应。

大型语言模型 (LLM) 与规划的交叉领域取得了重大进展，LLM 承担着各种角色 [12]。这些角色从生成计划 [7,21] 和启发式方法 [1,21] 到提取规划知识 [5] 不等。例如，在 Say-Can [1] 中，LLM 被用作评分模型，可以被视为为具身机器人可以执行的动作提供规划启发式方法。此外，LLM 也正在被用于利用来自用户或环境的反馈来提高其规划性能 [8,16,23]。我们的工作主要建立了一个评估框架，用于评估各种大型语言模型 (LLM) 的规划能力。PlanBench 包含多个任务，每个任务都旨在评估关于行动和变化的推理的某个方面。我们任务的提示以少样本示例的形式展示，其中我们提供一个实例和一个示例完成，然后要求对一个新实例进行完成。此外，我们使用领域描述来明确约束可能的行动。在许多日常场景中，我们经常被要求考虑不可预见的限制和约束。我们明确的领域描述使我们能够引入这些挑战，并迫使 LLM 超越仅仅重复他们在训练数据中可能遇到的关于该领域的可能信息。对提示条件的适应能力对于将通用系统定制到特定领域至关重要。

3 本文方法

3.1 背景

由于我们对研究基本规划问题感兴趣，因此我们希望关注最基础的规划形式化，即目标导向确定性规划问题。这类问题通常被称为经典规划问题。

经典规划问题可以用元组 $P = \langle D, I, G \rangle$ 进行数学表示。D 称为问题域，I 是初始状态，G 是目标规范。谓词上的所有可能的真值赋值定义了规划问题的状态空间。该领域再次由元组 $D = \langle F, O \rangle$ 定义。F 对应于一组流变，即用于定义状态空间的状态变量，每个流变对应于具有某些元数的谓词，而 A 对应于可以作为规划问题的一部分执行的动作集。每个动作 $a_i[V] \in A$（其中 a_i 是操作符标签，V 是操作符使用的变量，每个变量都可以映射到一个对

象)，可以进一步由两个部分定义：先决条件 $prec[V]$，它描述了何时可以执行动作；以及效果 $eff[V]$，它定义了执行动作时会发生什么。我们将假设 $prec[V]$ 由定义在变量 V 上的一组谓词组成。一个动作只有在满足其先决条件的情况下才可执行，即，先决条件中的谓词在给定状态下成立。动作 $f[V]$ 的影响由元组 $\langle add[V], del[V]\rangle$ 进一步定义，其中 $add[V]$ 或添加影响是动作将设置为真的谓词集，而 $del[V]$ 或删除影响是动作将设置为假的谓词集。如果我们用一个对象替换每个变量，则称一个动作是接地的，否则它被称为一个提升域模型（我们使用类似的约定来区分提升和接地谓词）。下面是来自一个名为 Blocksworld 的流行基准问题的动作片段，用 PDDL 语言描述。该动作对应于在该领域中将一个积木放在桌子上。

```
(:action put-down
  :parameters  (?ob)
  :precondition (holding ?ob)
  :effect (and (clear ?ob) (arm-empty) (on-table ?ob)
               (not (holding ?ob))))
```

在上面的代码片段中，参数行提供了可能的变量，在本例中为 $?ob$，它可以代表可能的块。前提条件说明，只有当你手持一个块时，你才能放下它（即谓词 $(holding\ ?ob)$ 对该块为真）。这些效果告诉你，执行动作“放下”后，积木将在桌子上并且是空的。你不会再拿着积木，手臂将被视为空闲。此外，动作可能与成本相关联，在这种情况下，人们也可以谈论最优计划，即，如果不存在比 π 成本更低的计划，则计划 π 被称为最优计划。上述描述呈现了一种较为简单的规划模型类别，可以从多个方面进行扩展，包括允许对象类型化（包括类型层次结构）、更复杂的先决条件和条件效应形式，更不用说支持更丰富的规划形式化类别。

3.2 评估架构

我们的基本测试框架包含两类组件：领域无关组件，作为框架的一部分提供；以及领域相关组件，需要为我们测试的每个新领域开发。

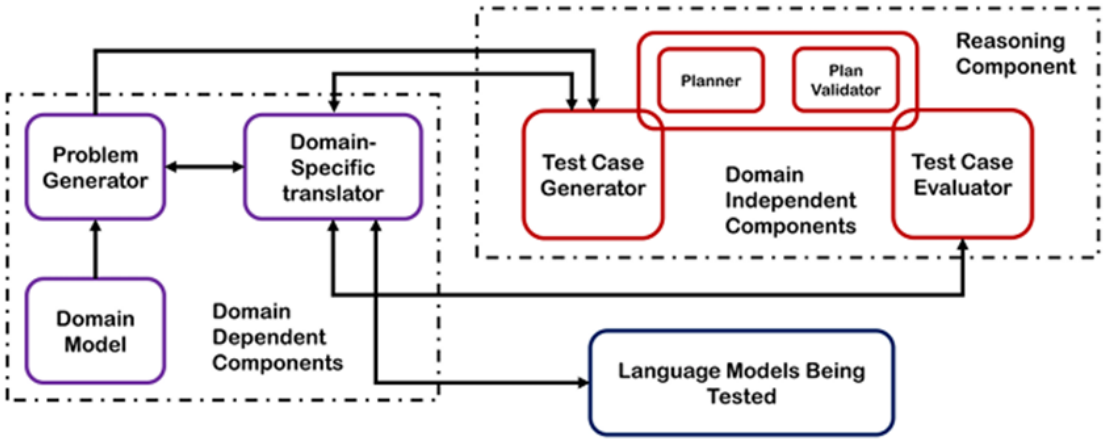

图 1. 整体测试框架的示意图。我们的系统由一个领域特定组件组成，该组件允许生成各种特定 PDDL 规划问题的实例，并实现 PDDL 与文本之间的转换。领域无关组件负责生成将输入到 LLM 中的测试实例，并验证 LLM 生成的输出。

领域独立组件：领域独立组件围绕一个规划器和一个计划验证组件构建，该组件处理各种规划问题，并制作与各种课程项目相对应的测试实例。该组件提供了验证 LLM 生成的解决

方案的机制。当前的方法将几乎完全基于符号模型（特别是使用 PDDL [20] 指定的模型）和与这些表示兼容的其他结构化输入进行操作。

该组件负责生成各种测试用例中产生的提示的内容，并验证 LLM 生成的输出。如前所述，该组件主要处理问题的形式化表示，因此它依赖于翻译组件将它生成的任何信息转换为自然语言，或将自然语言信息转换回形式化表示。对于每个测试用例，我们主要依赖一个与领域无关的规划器和一个计划验证器来生成相关信息或验证 LLM 提供的输出。对于某些测试用例，我们在设计提示时融入了特定领域的信息。在每种情况下，都有一个特定于测试用例的组件，该组件利用问题生成器组件提供的问题来制作特定的测试用例内容。

领域相关组件：领域相关组件由三个部分组成：领域模型、问题生成器和翻译器。提升后的领域文件描述了可用于解决任何给定规划问题的各种动作，可用于描述给定问题实例中可能存在的对象之间各种关系的各种谓词，以及可能成为给定问题一部分的各种类型的对象。领域模型被提升，因为它不涉及可能成为问题一部分的实际对象，而是独立于它可能影响的具体对象来定义操作。

问题生成器的作用是生成随机问题实例，这些实例包含各种对象、初始状态和目标。这些问题将成为生成我们将在整个框架中使用的各种测试用例的基础。我们希望在测试中使用的任何分布要求都可以内置到这个问题生成器中。

翻译器将符号模型信息转换为自然语言文本，反之亦然。将提示转换为自然语言将有利于基准测试的用户，他们可能希望参与其中以施加额外的约束或自己评估计划，因为这两项任务更自然地用自然语言完成。对于当前的测试平台（如下所述），我们开发了一种基于模板的机制来实现这一点。具体来说，我们为每个谓词和每个动作提供一个自然语言模板，并通过连接这些单独的字符串来形成状态和计划的文本。在将自然语言文本解析回结构化形式方面，我们感兴趣的特定任务是将 LLM 生成的计划转换回可以被计划验证工具（如 [6]）使用的计划形式。由于我们使用提示来塑造 LLM 的输出，因此我们要求计划中的每个动作都列在不同的行上。然后，我们可以通过使用基于模板的匹配或假设句子中的动词对应于动作，并且每个名词对应于构成动作参数的对象（然后将其映射到可能的动作）来解析动作的精确动作和参数。

4 复现细节

4.1 与已有开源代码对比

与开源代码相比，我在开源代码上新添加部分改变，包括域描述提示词的修改与优化、思维链提示示例的嵌入。

4.2 开源代码

原代码提供了对于 LLM 在规划任务下的评估测试功能，原论文对 GPT-4、和 text-davinci-002 模型进行基准测试，评估了这两模型在规划和推理行动及变化的方面的表现情况。并得到下面结果，图 2

Task	Instances correct	
	GPT-4	I-GPT3
Plan Generation We showcase an instance and the respective plan as an example and prompt the machine with a new instance.	206/600 (34.3%)	41/600 (6.8%)
Cost-Optimal Planning We showcase an instance, the respective optimal plan and the associated cost as an example and prompt the machine with a new instance.	198/600 (33%)	35/600 (5.8%)
Plan Verification We showcase three instances and three distinct plans (goal reaching, non goal-reaching and inexecutable) and present the respective validation and explanations. We then present a new instance and a plan and ask the machine for to verify and provide an explanation, if needed.	352/600 (58.6%)	72/600 (12%)
Reasoning About Plan Execution We showcase an instance, an action sequence and the corresponding resulting state after executing the action sequence as an example. We then provide an instance and an executable action sequence and ask the machine to provide the resulting state.	191/600 (31.8%)	4/600 (0.6%)
Replanning We showcase an instance, the respective plan and present an unexpected change of the state. We then also present a new plan from the changed state. Finally, for a new instance we repeat the same except we ask the machine for the new plan.	289/600 (48.1%)	40/600 (6.6%)
Plan Generalization We showcase an instance and the respective plan as an example and prompt the machine with a new instance. The plans for both the instances can be generated by a fixed program containing loops and conditionals.	141/500 (28.2%)	49/500 (9.8%)
Plan Reuse We showcase an instance and the respective plan as an example and prompt the machine with an instance which requires only a certain prefix of the plan provided in the example.	392/600 (65.3%)	102/600 (17%)
Robustness to Goal Reformulation (Shuffling goal predicates) We showcase an instance and the respective plan as an example and prompt the machine with the same instance but shuffle the ordering of the goals.	461/600 (76.8%)	467/600 (77.8%)
Robustness to Goal Reformulation (Full → Partial) We showcase an instance with a fully specified goal state and the respective plan as an example and prompt the machine with the same instance but provide a partially specified goal state.	522/600 (87%)	467/600 (77.8%)
Robustness to Goal Reformulation (Partial → Full) We showcase an instance with a partially specified goal state and the respective plan as an example and prompt the machine with the same instance but provide a fully specified goal state.	348/600 (58%)	363/600 (60.5%)

图 2. GPT-4 和 Instruct-GPT3 (text-davinci-002) 在 Blocksworld 领域上的 PlanBench 结果。高亮行中的任务对应于实际规划问题，而其他任务对应于更简单的辅助规划任务。

我就其规划生成的测试任务流程，对 gpt-3.5-turbo、gpt-4o-mini、gpt-3.5-turbo-instruct、gemini-1.0-pro 模型进行了相关测试，得到如下复现结果，图 3

图 3. gpt-3.5-turbo、gpt-4o-mini、gpt-3.5-turbo-instruct、gemini-1.0-pro 在 Blocksworld、logistics 领域上的 PlanBench 结果

由于实验条件限制，对于 gpt-4 测试的代价过于昂贵。故所进行的测试复现选用的都是廉价的模型。其中可以发现 Gemini 在任务生成方面效果稍微好些。

4.3 改进工作

总共进行了两项工作，包括提示词优化和思维链优化，指在测试这两种优化对于大语言模型的规划能力的提升情况。

4.3.1 实验一：提示词优化

我发现其测试提示词环境描述提示词过于繁琐 LLM 理解，所以对此，我对其 PDDL 环境域的描述文本进行修改，尝试通过提示词工程可以优化该测试的提示词输入，图 4，使得关注于测试 LLM 的规划能力，而非理解能力上来。

修改前的Stack a Block描述

I can only stack a block on top of another block if I am holding the block being stacked.
I can only stack a block on top of another block if the block onto which I am stacking the block is clear.
Once I put down or stack a block, my hand becomes empty.
Once you stack a block on top of a second block, the second block is no longer clear.

修改后的Stack a Block描述

Stack a Block
- Description: Place the block you are holding on top of another **clear** block.
- Conditions:
  - You are **holding** a block.
  - The target block you are stacking onto is **clear**.
- Post-Action Effects:
  - Your hand becomes empty.
  - The target block is now **not clear** (since another block is stacked on it).

图 4. 提示词优化示例

通过对 gpt-3.5-turbo 进行 PlanBench 测试, 得到下面结果，图 5

优化前 gpt-3.5-turbo

```
100%|██████████| 500/500 [00:31<00:00, 15.72it/s]
Total correct: 15
Total instances: 500
Accuracy: 0.03
```

优化后 gpt-3.5-turbo

```
100%|██████████| 500/500 [00:31<00:00, 15.75it/s]
Total correct: 19
Total instances: 500
Accuracy: 0.038
```

图 5. 提示词优化结果

4.3.2 实验二: 思维链优化

对于原来的提示，我接着做了添加思维链的尝试，用于测试思维链在提升规划能力上效果。思维链是指在提示示例中添加相关的思维推导过程来使得提高模型在相关任务上的表现。

思维链示例如下：

[STATEMENT]↓

As initial conditions I have that, the blue block is clear, the orange block is clear, the hand is empty, the orange block is on top of the yellow block, the yellow block is on top of the red block, the red block is on the table and the blue block is on the table.↓

My goal is to have that the red block is on top of the orange block, the blue block is on top of the red block and the yellow block is on top of the blue block.↓

My plan is as follows:↓

[PLAN]↓

1. - Reasoning: To move the yellow block, the orange block must be removed first.↓
   - Action: Unstack the orange block from on top of the yellow block.↓
2. - Reasoning: Placing the orange block on the table clears the way to handle the yellow block.↓
   - Action: Put down the orange block.↓
3. - Reasoning: To move the red block, the yellow block must be removed first.↓
   - Action: Unstack the yellow block from on top of the red block.↓
4. - Reasoning: Placing the yellow block on the table allows access to the red block.↓
   - Action: Put down the yellow block.↓
5. - Reasoning: To place the red block on the orange block as per the goal.↓
   - Action: Pick up the red block.↓
6. - Reasoning: This achieves part of the desired final arrangement.↓
   - Action: Stack the red block on top of the orange block.↓
7. - Reasoning: To place it on top of the red block as per the goal.↓
   - Action: Pick up the blue block.↓
8. - Reasoning: This sets up the final position for the yellow block.↓
   - Action: Stack the blue block on top of the red block.↓
9. - Reasoning: To complete the goal by placing it on top of the blue block.↓
   - Action: Pick up the yellow block.↓
10. - Reasoning: This completes the desired final arrangement.↓
    - Action: Stack the yellow block on top of the blue block.↓

[PLAN END]↵

对 gpt-3.5-turbo 进行测试，得到结果，图 6

图 6. 思维链优化结果

通过前面的尝试发现提示词工程可以提高 LLM 的规划能力，于是尝试添加其他表现更好的模型如 gemini-pro 进行实验验证，得到下面结果，图 7

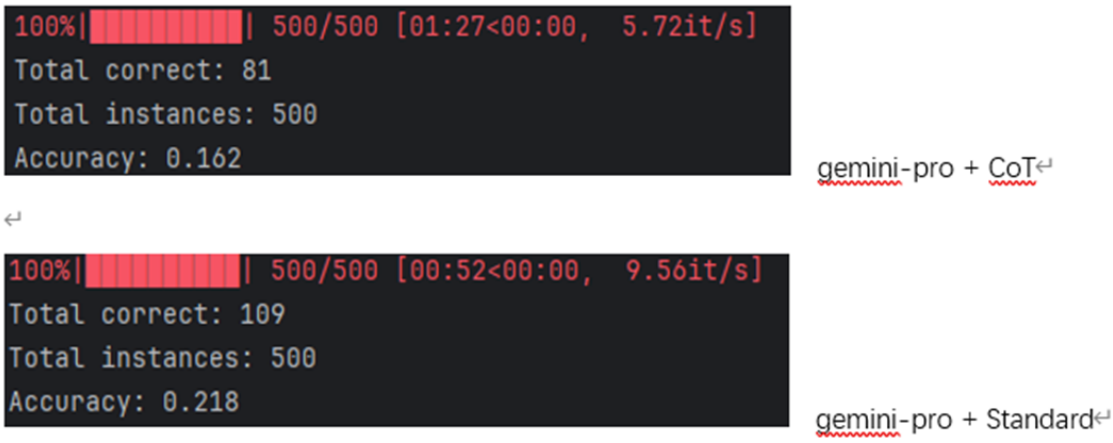

图 7. 思维链优化结果, gemini

在 gemini-pro 测试实验中思维链的效果低于标准的情况分析：添加思维链后输出结果中充斥着推理过程，这加大了从 LLM 输出中提取 plan 的难度。使得出现提出失败的案例，从而导致负优化效果。这也侧面说明了 PlanBench 对于从 LLM 输出的复杂文本提取规划动作方面存在困难。

5 实验结果分析

以下是我的实验结果表 1 ，可以发现在处理规划能力比较弱的模型上，可以通过提示词工程如思维链来提高其规划能力的表现，同时也发现了该 PlanBench 在应对 LLM 输出的复杂文本中提取规划动作方面上存在概率提取失败的缺陷。

		Time (ms)		
Model	Task	Standard ↑	Optimized ↑	CoT ↑
gpt-3.5-turbo	plan generation	15	19	25
gemini-pro		109	-	81

表 1. Comparison of time taken by different models for plan generation tasks.

6 总结与展望

本文对 PlanBench 进行了复现，并利用 PlanBench 测试了多个 LLM 模型，同时使用 PlanBench 测试了提示词工程对于 LLM 在规划任务上的提升效果，包括提示词优化、思维链优化。结果发现在处理规划能力比较弱的模型上，可以通过提示词工程来提高其规划能力的表现，同时也发现了 PlanBench 在应对从 LLM 输出的复杂文本中提取规划动作方面上存在可能提取失败的缺陷。

参考文献

[1] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[2] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

[3] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[4] Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361, 2021.

[5] Lin Guan, Karthik Valmeekam, Sarath Sreedharan, and Subbarao Kambhampati. Leveraging pre-trained large language models to construct and utilize world models for model-based task planning. *arXiv preprint arXiv:2305.14909*, 2023.

[6] Richard Howey, Derek Long, and Maria Fox. VAL: Automatic plan validation, continuous effects and mixed initiative planning using PDDL. In *16th IEEE International Conference on Tools with Artificial Intelligence*, pages 294–301. IEEE, 2004.

[7] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning*, pages 9118–9147. PMLR, 2022.

[8] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022.

[9] IPC. International planning competition. https://www.icaps-conference.org/competitions, 1998.

[10] Liwei Jiang, Jena D. Hwang, Chandrasekhar Bhagavatula, Ronan Le Bras, Maxwell Forbes, Jon Borchardt, Jenny Liang, Oren Etzioni, Maarten Sap, and Yejin Choi. Delphi: Towards Machine Ethics and Norms. *ArXiv*, abs/2110.07574, 2021.

[11] Subbarao Kambhampati. AI as (an Ersatz) Natural Science? https://cacm.acm.org/blogs/blog-cacm/261732-ai-as-an-ersatz-natural-science/fulltext, Jun 2022.

[12] Subbarao Kambhampati, Karthik Valmeekam, Matthew Marquez, and Lin Guan. On the role of large language models in planning, July 2023. Tutorial presented at the International Conference on Automated Planning and Scheduling (ICAPS), Prague. https://yochan-lab.github.io/tutorial/ICAPS-2023/.

[13] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large Language Models are Zero-Shot Reasoners. *arXiv preprint arXiv:2205.11916*, 2022.

[14] Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *arXiv preprint arXiv:1705.04146*, 2017.

[15] Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP Models really able to Solve Simple Math Word Problems? *arXiv preprint arXiv:2103.07191*, 2021.

[16] Shreyas Sundara Raman, Vanya Cohen, Eric Rosen, Ifrah Idrees, David Paulius, and Stefanie Tellex. Planning with large language models via corrective re-prompting. *arXiv preprint arXiv:2211.09935*, 2022.

[17] Stuart J Russell and Peter Norvig. *Artificial intelligence a modern approach*. London, 2010.

[18] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8732–8740, 2020.

[19] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*, 2022.

[20] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*, 2018.

[21] Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. On the planning abilities of large language models–a critical investigation. *arXiv preprint arXiv:2305.15771*, 2023.

[22] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

[23] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023.