トップダウン手続きを応用した LLM Agent のプランニングの試み

村上夏輝　加賀屋智之　黄瀬輝
パナソニック　コネクト株式会社
{murakami.natsuki,kagaya.tomoyuki,kinose.akira}@jp.panasonic.com

概要

LLM の著しい成果により，LLM を用いたエージェント (LLM Agent) 研究が近年増加しており，エージェントに重要なプランニングに LLM が用いられる研究が特に多い．本研究は，認知心理学や論理学から着想を得て，人間の問題解決能力を模倣するフレームワークを提案する．具体的には，複雑なゴールに対して，ルールに沿って，下位の簡単なサブゴールに木構造的に分解していく．評価実験の結果，既存手法と比較して提案手法によるプランニングのステップ数が少なくなることを示した．

1　はじめに

知的エージェント (以降，エージェントと呼ぶ) は人工汎用知能を達成するための有望なアプローチとして認識されており [1]，ソフトウェア領域だけでなく，Embodied AI としてロボティクスやナビゲーションの分野での応用が期待されている [2]．従来のアプローチでは，シンボリックアプローチ，強化学習や模倣学習を用いたアプローチが多く利用されていたが，大規模言語モデル（LLM) の著しい成果により，LLM を用いたアプローチ（以降，LLM Agent と呼ぶ）が近年増加している．

LLM Agent では，特にエージェントの能力の一つであるプランニングに LLM が利用されることが多い [3, 4, 5]．プランニングとは，与えられたゴールに対してそれを達成するためにとるべき行動の一連の流れを考えることを指す．例えば，「歯を磨く」というゴールには「洗面台へ向かう，歯磨き粉を探す，歯磨き粉をもつ，歯ブラシを探す，歯ブラシを持つ」というプランニングが必要である．もし，「歯磨き粉を探す」段階で失敗したら，「歯磨き粉を買いに行く」という新しいゴールを考える，もしくは「歯ブラシを持つ」までスキップするなど，推論や意思決定も必要となる．プランニングは，観測情報や環境をもとに，複雑な意味理解および，構造理解，推論，意思決定のプロセスを必要とし，エージェントの中心的能力と言われている．

LLM はプランニング能力を向上させる一方で，その推論過程には課題が残されている．LLM の一般的な推論は，現在の状態から始め目標に向かって段階的に推論を積み重ねていく「ボトムアップ手続き（forward reasoning)」である．この手法は，図 1 の (a) のように，解くタスクが複雑になるほど探索空間が膨大になり，効率が悪いと言われている [6]．加えて，探索空間が膨大になることで，推論途中で矛盾やゴールを見失うことがある．また，LLM の確率論的推論は物事を構造的に捉えているとは言えず，プランニングの説明性や解釈性には課題がある．これらの課題により，LLM agent のプランニングは，人間のような柔軟で効率的な問題解決能力には至っていない．特に，実世界での応用において重要となる，状況に応じた適切なプランの選択や，プランニングの説明性の欠如は，社会実装における大きな課題となっている．

本研究は，[7, 8] の認知心理学や論理学の証明手法から着想を得て，人間の問題解決能力を模倣するプランニングフレームワークを提案する．具体的には，複雑なゴールに対して，ルールに沿って，下位の簡単なサブゴールに木構造的に分解していく．それらのサブゴールが現状から実現可能かどうかを判定していくことで，現状を利用したプランニングを行うことができ，効率性を向上させる．

2　関連研究

2.1　木構造を用いたプランニング

本研究は，プランニングを構造的に捉える手法として木構造を利用している．木構造を用いたプラン

This work is licensed by the author(s) under CC BY 4.0
(https://creativecommons.org/licenses/by/4.0/).

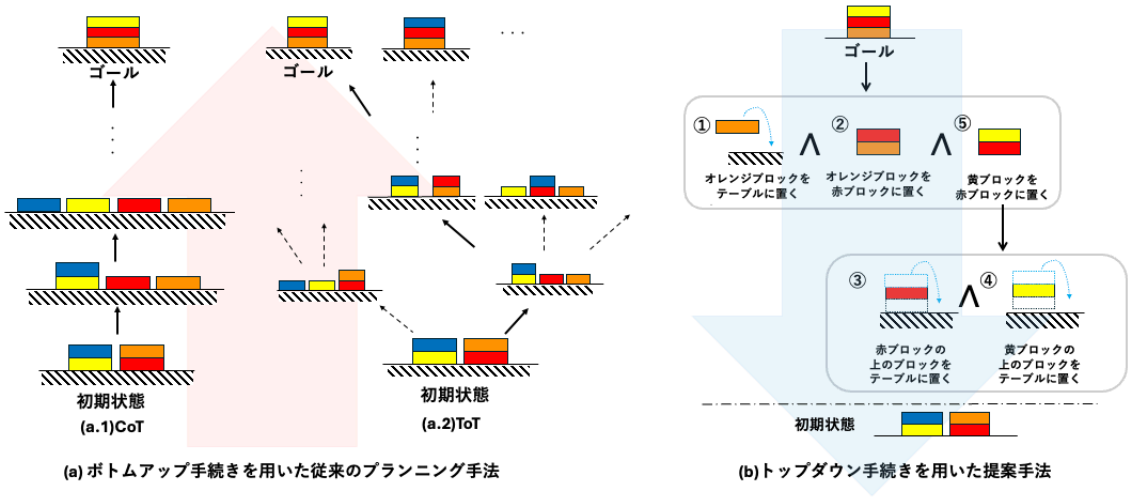

図 1　従来のプランニング手法と提案手法の比較
（a）従来のボトムアップ手法：初期状態から始めて可能な行動を次々と探索していくため，探索空間が増大する．（b）提案するトップダウン手法：目標状態を階層的に分解し，実行可能な基本行動まで段階的に具体化することで，探索空間を効率的に絞り込む．

ニングは，複数の結果の枝を探索する手法として長年研究されてきた．LLM Agent のプランニング研究でも，その古典的なアプローチを活用しようという研究が出てきている．Tree of Thought (ToT)[9] は LLM の推論に木構造を用いて，複数の選択肢を考慮する手法を提案した．RAP[10] は，ToT と同じようにヒューリスティックに基づく探索を適用しているが，ワールドモデルを導入した推論により，プランニングを行っている．LATS[11] は ToT や RAP のような木構造探索に，外部フィードバックを導入する手法を提案した．[12] によると，ToT や RAP のような木構造を用いた手法はマルチプラン選択に分類され，複数の可能なプランを並行して検討し，その中から最適なものを選択するという特徴を持つ．この特徴により，正解が一意でないクリエイティブなタスクのプランニングに特に適していると言える．

2.2　タスク分解のプランニング

複雑なタスクを単純化することは，人間の顕著な能力であり，これは一つのタスクをより単純なサブタスクに分割することに現れている [8]．この考えは LLM の推論にも応用されており，Chain-of-Thought（CoT）[13, 14] は，推論の際に推論のステップを踏む手段を提案した．ReAct[15] は推論とプランニングを分離し，交互に行うことで，プランニング能力に大幅な改善をもたらした．

これらの手法は，ボトムアップ手続きでの探索を採用しており，自由で柔軟な推論を得意とする．その一方で，タスクが複雑になると探索空間が大きくなってしまったり，動的な環境の情報を利用する効率的な推論ができない課題がある．

3　手法

本研究は，[7, 8] などの認知心理学や論理学の推論手法から着想を得ている．[7, 8] では，人間は複雑な問いに直面した場合，それをより簡単な問いに分解する傾向があると述べている．論理学では，推論手法にトップダウン手続きというものがある．トップダウン手続きは，終式から証明を始め，その式に至る推論規則を探していくことで始式に至る手法である．トップダウン手続きは，ボトムアップ手続きよりも，特定の問題を解決する手法として最適で，探索空間が小さくなるため効率的であると言われており [6]，多段階推論など複雑な推論タスクで利用されている [16, 17, 18]．本研究はこれらから着想を得て，複雑なゴールを簡単な行動に分解していくプランニング手法を提案する．

本研究が構築する推論の木 T は $T = (N, E)$ として定義される．N はノードの集合，E はエッジの集合である．各ノード $n \in N$ は，プランニングに必要な 4 つの要素から構成され，$n = (action, status, reason, rank)$ である．$status$ は現状から実現可能かどうかの状態，$reason$ は $status$ と判定した理由，$rank$ は同じ親ノードを持つ同じ深さのノード間での実行すべき順序である．入力として，ゴール $Goal$ と初期状態 $Initial\_condition$

This work is licensed by the author(s) under CC BY 4.0
(https://creativecommons.org/licenses/by/4.0/).

を渡す．出力としてはプランニングのパスである *path_list* が期待される．以下，アルゴリズムについて詳しく説明していく．

3.1 アクションの分解

親ノードの *action* は子ノードの *action* を満たせば到達するゴールである．本手法では，このゴールに対して，必要条件となるアクションへの分解を LLM が行う．以下，実験で利用した blockworld データセット [19] を用いた図 1 の (b) の例を考える．

Initial_condition：「青ブロック，オレンジブロックは空である．オレンジブロックは赤ブロックの上にあり，青ブロックは黄色ブロックの上にある．」(図 1 の (b) の初期状態)

Goal(= *action* $\in n_0$)：「黄色，赤，オレンジのブロックがある．一番上は黄色ブロック，２番目は赤ブロックで一番下はオレンジブロックである．」(図 1 の (b) のゴール)

図 1 の (b) の一段目の枠内のように，この *Goal* から *action* のリストは，土台を作るために「オレンジブロックをテーブルに置く」，「２番目は赤ブロックで一番下はオレンジブロック」を満たすために「赤ブロックをオレンジブロックに置く」，「一番上は黄色ブロック，２番目は赤ブロック」を満たすために「黄色ブロックを赤ブロックに置く」に分解できる．上記の例のようにサブゴールを達成するための必要なアクションのリストに分解される．

3.2 アクションの実行可能性

3.2.1 実行順の順位づけ

分解された *action* の実行すべき順を LLM が出力し，順位をつける．上記の *action* のリストでは，“オレンジブロックをテーブルに置く” が一番先に実行され，その後に “赤ブロックをオレンジブロックに置く”，“黄色ブロックを赤ブロックに置く” が続くべきである．よって順位のリストは $\{3, 2, 1\}$ となる．

3.2.2 実行可能性の判断

action が現状から実行可能かどうかを LLM が出力する *Initial_condition* からは，上記の *action* のリストの “オレンジブロックをテーブルに置く” のみが実行可能である．よって，実行可能性のリストは { 実行不可能, 実行不可能, 実行可能 } である．実行可能と判断された *action* は，*path_list* に追加する．なお，すでに実行済みで，実行する必要のない *action* も実行可能と判断する．

3.2.3 実行後の状況を予測

実行可能と判断された *action* を実行した場合，状況がどう変化するかを LLM が出力する．実行可能と判断された “オレンジブロックをテーブルに置く” を実行すると状況は図 2 の①実行後の状況のように，「青ブロック，オレンジブロック，赤ブロックはクリアである．青ブロックは黄色ブロックの上にある．オレンジブロック，赤ブロックはテーブルの上にある．」となる．

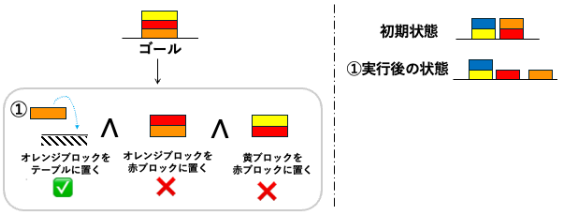

図 2 実行可能アクションを実行した後の path の状況

3.2.4 実行可能性の更新

状況が変わったことにより，実行可能性が変化した *action* の更新をする必要がある．“オレンジブロックをテーブルに置く” を実行し，状況が変化したことで，実行不可能だった “赤ブロックをオレンジブロックに置く” が実行可能となる．以上のように，実行可能の *action* の実行により，実行可能性が変化することがある．よって，実行不可能と判断された *action* が変わらなくなるまで，実行不可能と判断された *action* に対して3.2.2 〜 B を繰り返す．更新が止まった際の *action* の *status*, *reason* と，順位づけた *rank* を木 T のノード n に格納する．*status* が実行不可能なノードを親ノードとして，3.1〜3.2 を繰り返す．

3.3 木構築の終了

3.2.4 で *action* をノード n_t に格納した後，*status* が全て実行可能だった場合，親ノードの *status* を実行可能に更新する．更新したのちに，n_{t-1} も全て実行可能だった場合，その親ノード n_{t-2} の *status* を実行可能に更新する，と繰り返し，根ノードの *status* が実行可能と判断されることで T の構築が終了する．実行可能と判断された順の *action* が *path_list*

This work is licensed by the author(s) under CC BY 4.0
(https://creativecommons.org/licenses/by/4.0/).

で出力される．

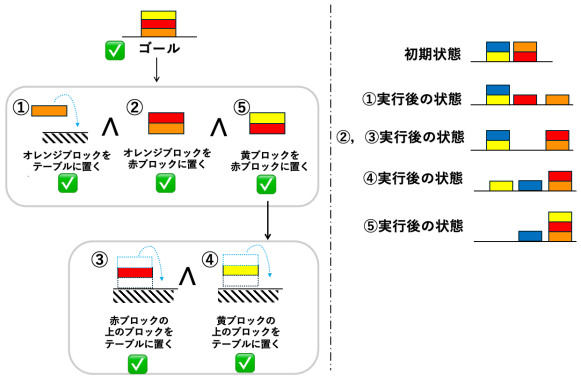

図 3　木の構築の終了

4　評価実験・考察

4.1　実験設定、データセット

データセット　実験の評価にはプランニングのベンチマークである PlanBench[19] の BlockWorld を使用した．BlockWorld は様々な色のブロックを自然言語で指示された通りにテーブルの上に積み上げるタスクである．BlockWorld は（1）自然言語で記述された複雑なゴールの理解（2）ブロックの空間的な依存関係の把握（3）ブロックの移動順序（プランニング）の最適化を評価することができる．

本研究では，データセットの “query” から，*Initial_condition* と *Goal* を抽出し，日本語に翻訳した．また，難しいタスクで評価するために，日本語に翻訳した *Goal* が句読点を 2 回以上含むものを難しいタスクとし，利用した．難しいタスクは 件がデータセットに含まれ，その中からランダムに抽出した 30 件を用いて評価実験を行った．

実験設定　BlockWorld データセットにおけるプランニングでは，取れるアクションは，“Pick up ”, “Unstack ”, “Put down ”, “Stack ” であるが，本研究では，“[blockA] を [blockB] に置く”，“[block] をテーブルに置く”，“[block] の上のブロックをテーブルに置く” に限る．これは，実行する際には “Pick up”,“Unstack ” が “置く” 動作の一連の流れであると捉え，複雑なゴールを簡単なサブタスクに分解するプランニング手法である本提案手法を利用した後のコード実行での手続きだと考えたからである．

提案手法で用いる LLM は全て Opan AI 社の GPT-4o モデル [20] を利用した．ベンチマークとしては提案手法と同じサブタスク分解の手法である CoT[14] と比較を行った．モデルは提案手法と同じ，GPT-4o モデルを利用した．

出力されたプランニング結果に対する評価は人手で行った．評価の際のルールは，A を参照されたい．正答率はプラン通りにブロックを動かし，初期状態からゴールに辿り着いた割合であり，平均ステップ数は提案手法，既存手法ともに正答したタスクのステップ数の平均である．最短ステップ数は，人手でアノテーションした，人が考える最も効率的なプランニングのステップである．各々の手法で正答したタスクとそのタスクの最短ステップ数との差を出している．

4.2　実験結果

実験結果は表 1 に示す．正答数は CoT に及ばなかったものの，平均ステップ数と最短ステップ数との差は減少しており，プランニングの効率が上がったと言える．

表 1　提案手法の評価実験結果

スコア	提案手法	CoT[14]
正答率 (件数)	56.7% (17 件)	**66.7%** (20 件)
平均ステップ数	**4.4**	4.9
最短ステップとの差	**+0.47**	+1.65

4.3　考察

提案手法がプランニングを失敗したタスクを解析したところ，分解と実行可能性の判断で失敗していることがわかった．先行研究である [16] でも分解に関して課題として挙げており，LLM は言語指示から構造的に分解することはできていない可能性がある．本研究は，言語指示から空間的な構造関係への分解することを必要としており，この観点での LLM の評価を今後の課題とする．エラー分析の詳細は B に示す．

5　終わりに

本研究では，複雑なゴールに対して，下位の簡単なサブゴールに木構造を用いて分解していく．それらのサブゴールが現状から実現可能かを判定していくことで，現状を利用したプランニングを行え，効率性を向上させるプランニング手法を提案した．結果として，正答数は既存手法に及ばなかったものの，ステップ数では大幅な減少が見られ，効率的なプランニングが行えたと言える．今後はロボティクス分野等での応用に向けて，本手法を改良していく．

This work is licensed by the author(s) under CC BY 4.0
(https://creativecommons.org/licenses/by/4.0/).

参考文献

[1] Lei Wang, Chengbang Ma, Xueyang Feng, Zeyu Zhang, Hao ran Yang, Jingsen Zhang, Zhi-Yang Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Ji rong Wen. A survey on large language model based autonomous agents. **ArXiv**, Vol. abs/2308.11432, , 2023.

[2] Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. A comprehensive overview of large language models, 2024.

[3] Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. Llm+p: Empowering large language models with optimal planning proficiency, 2023.

[4] Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: Generating situated robot task plans using large language models, 2022.

[5] Zihao Wang, Shaofei Cai, Guanzhou Chen, Anji Liu, Xiaojian Ma, and Yitao Liang. Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents, 2024.

[6] Fei Yu, Hongbo Zhang, and Benyou Wang. Natural language reasoning, a survey. **ACM Computing Surveys**, 2023.

[7] Herbert A Simon and Allen Newell. Human problem solving: The state of the theory in 1970. **American psychologist**, Vol. 26, No. 2, p. 145, 1971.

[8] Susan F Chipman, Jan Maarten Schraagen, and Valerie L Shalin. Introduction to cognitive task analysis. In **Cognitive task analysis**, pp. 3–23. Psychology Press, 2000.

[9] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models, 2023.

[10] Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with language model is planning with world model, 2023.

[11] Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning acting and planning in language models. **ArXiv**, Vol. abs/2310.04406, , 2023.

[12] Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Ruiming Tang, and Enhong Chen. Understanding the planning of llm agents: A survey, 2024.

[13] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023.

[14] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners, 2023.

[15] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023.

[16] Shangzi Xue, Zhenya Huang, Jiayu Liu, Xin Lin, Yuting Ning, Binbin Jin, Xin Li, and Qi Liu. Decompose, analyze and rethink: Solving intricate problems with human-like reasoning cycle. In **The Thirty-eighth Annual Conference on Neural Information Processing Systems**, 2024.

[17] Oyvind Tafjord, Bhavana Dalvi Mishra, and Peter Clark. Entailer: Answering questions with faithful and truthful chains of reasoning. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, **Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing**, pp. 2078–2093, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics.

[18] Yan Junbing, Chengyu Wang, Taolin Zhang, Xiaofeng He, Jun Huang, and Wei Zhang. From complex to simple: Unraveling the cognitive tree for reasoning with small language models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, **Findings of the Association for Computational Linguistics: EMNLP 2023**, pp. 12413–12425, Singapore, December 2023. Association for Computational Linguistics.

[19] Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change, 2023.

[20] OpenAI, :, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark ..., and Yury Malkov. Gpt-4o system card, 2024.

This work is licensed by the author(s) under CC BY 4.0
(https://creativecommons.org/licenses/by/4.0/).

A　評価のルール

- ブロックは重なっているのであれば，複数個動かしても良い．(現状を利用したプランニングを行うため，状態を保持した方が効率がいい場合がある)
- “[block] の上のブロックをテーブルに置く” は [block] の上のブロック全てを, 状態を保持したままテーブルに置く．
- “[blockA] を [blockB] に置く” 際は [blockB] の上に何もない（空である）時にしか実行できない．

B　提案手法の実験のエラー分析

提案手法がプランニングを失敗したものを解析した結果，実行可能性の判断ミスで失敗した件数と分解ミスで失敗した件数は同数であった．

先行研究である [16] でも分解に関しては課題として挙げており，[16] ではアノテーションした分解データセットを構築し，タスクに類似したものをプロンプトに補填する形で精度向上を行っている．本研究では，LLM が言語指示から構造的に分解できるのかという分析を今後の課題とする．実行可能性の判断ミスに関して,，本研究では実行可能性を判断させる “状況” を *initial_condition* のような自然言語の文の状態で LLM に渡している．また，で述べるように，アクション実行後の状況を LLM に予測させている．どちらが実行可能性の判断のミスの起因かを引き続き分析するとともに，自然言語の文の状態のままで “状況” を受け渡しするのではなく，整理した状態での受け渡しをする，画像情報を用いて受け渡しをするなどの手法の検討を行う．

表 2　エラー分析

エラーの種類	件数
実行可能性の判断ミス	6
分解ミス	6
順位付ミス	1
合計	13

C　効率的なプランニングの例

提案手法が効率的なプランニングを行うことができた例を示す．

initial_condition: 青いブロックは空いています．黄色いブロックは空いています．手は空です．青いブロックはオレンジのブロックの上にあります．オレンジのブロックは赤いブロックの上にあります．赤いブロックはテーブルの上にあり，黄色いブロックもテーブルの上にあります．

goal: 赤いブロックが青いブロックの上にあり，青いブロックが黄色いブロックの上にあり，黄色いブロックがオレンジ色のブロックの上にあります．

提案手法

アクション: 青いブロックを黄色いブロックに置く．
アクション: 赤いブロックの上のブロックをテーブルに置く
アクション: 赤いブロックを青いブロックに置く．
アクション: 黄色ブロックをオレンジブロックに置く．

cot

1. 青いブロックをテーブルに置く．
2. オレンジのブロックをテーブルに置く．
3. 赤いブロックを青いブロックの上に置く．
4. 青いブロックを黄色いブロックの上に置く．
5. 赤いブロックを青いブロックの上に戻す．
6. 黄色いブロックをオレンジ色のブロックの上に置く．

上記の例は，青いブロック，黄色いブロックともに空であるため，提案手法のように “青いブロックを黄色いブロックに置く” というアクションを取ることが最短であるが，CoT は “青いブロックをテーブルに置く” という余分なステップが提案されている．

This work is licensed by the author(s) under CC BY 4.0
(https://creativecommons.org/licenses/by/4.0/).