사고의 환상: 문제 복잡성의 관점에서 추론 모델의 강점과 한계를 이해하는

parshin shojaee*† iman Mirzadeh* keivan alizadeh maxwell horton samy bengio Mehrdad farjtabar

애플

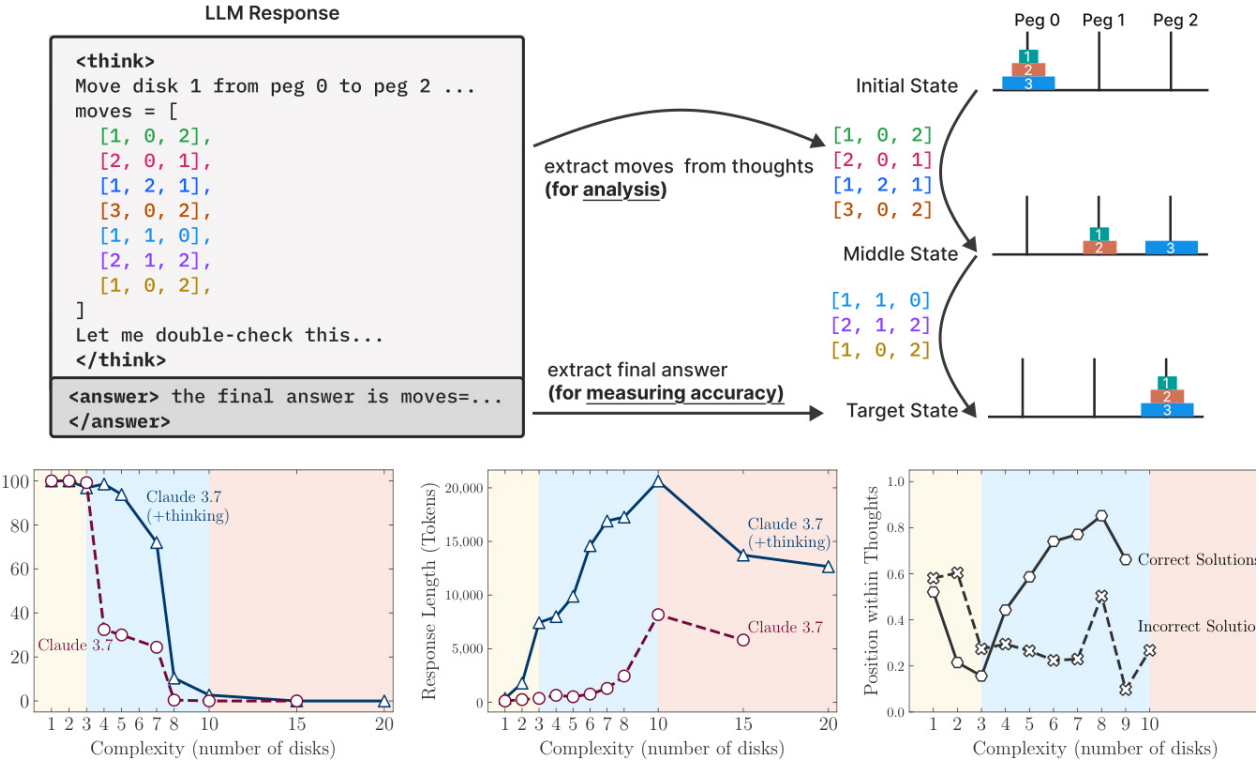
압축

최근 세대의 한계 언어 모델(frontier language models)은 대규모 추론 모델(large reasoning models, LRMs)을 도입하여 세부적인 사고 과정을 생성하고 답변을 제공합니다. 이러한 모델은 추론 벤치마크(braining benchmarks)에서 성능 향상을 보여주지만, 기본 능력, 확장 특성 및 한계는 여전히 충분히 이해되지 않고 있습니다. 현재 평가는 주로 기존의 수학적 및 코딩 벤치마크(benchmarks)에 초점을 맞추며, 최종 답변 정확도에 중점을 둡니다. 그러나 이러한 평가 패러다임은 데이터 손상으로 종종 고통받고, 추론 흔적의 구조와 품질에 대한 통찰력을 제공하지 않습니다. 이 연구에서는 합성적 복잡성의 정밀한 조작을 가능하게 하는 통제 가능한 퍼즐 환경(puzzle environments)의 도움을 받아 이러한 격차를 체계적으로 조사합니다. 이 설정은 최종 답변뿐만 아니라 내부 추론 흔적(brain traces)의 분석을 가능하게 하여 LRMs가 어떻게 "생각하는"지를 이해할 수 있게 합니다. 다양한 퍼즐에 대한 광범위한 실험을 통해, 우리는 한계 렌치마크(frontier language models)가 특정 복잡성을 넘어서는 완전한 정확도 붕괴에 직면한다는 것을 보여줍니다. 더욱이, 이들은 적절한 토큰 예산을 가지고 있음에도 불구하고 추론 노력이 문제 복잡성을 최대로 증가하고 감소합니다. LRMs와 표준 렌치마크(large reasoning language models)의 대조 counterparts를 동등한 추론 계산(equivalent inference compute) 하에 비교함으로써, 우리는 세 가지 성능 체계를 식별합니다: (1) 저 복잡성 작업(low complexity tasks)에서 표준 모델이 LRMs를 놀라울 정도로 능가하는 성능, (2) 중 복잡성 작업(medium-complexity tasks)에서 LRMs의 추가적인 사고가 이점을 보여주는 성능, (3) 고 복잡성 작업(high-complexity tasks)에서 두 모델이 완전한 붕괴를 경험하는 성능. LRMs가 정확한 계산에 한계가 있음을 발견했습니다: 이들은 명시적인 알고리즘을 사용하지 않고 퍼즐 전반에 걸쳐 일관되게 일관되게 추론하지 않기 때문입니다. 또한, 추론 흔

1 도입

대형 언어 모델(large language models, LLMs)은 최근 추론 작업에 명시적으로 설계된 특수화된 변형을 포함하도록 진화했습니다. 이러한 모델은 OpenAI의 o1/o3 [1, 2], DeepSeek-R1 [3],

Claude 3.7 Sonnet Thinking [4], Gemini Thinking [5]와 같은 대형 추론 모델(large reasoning models, LRMs)입니다. 이러한 모델은 자기 반영을 통한 긴 추론 체인(CoT)과 같은 "생각" 메커니즘으로 특징지어지며, 다양한 추론 벤치마크에서 유망한 결과를 보여주었습니다. 이러한 모델의 출현은 LLM 시스템이 복잡한 추론 및 문제 해결 작업을 접근하는 패러다임 전환을 제안하며, 일부 연구자들은 이를 보다 일반적인 인공지능 능력으로의 중요한 단계로 제안합니다.



그래프1: 상단: 우리의 설정은 최종 답변과 중간 추론 추적을 모두 검증할 수 있게 하여 모델 사고 행동의 세부 분석을 가능하게 합니다. 그래프1: 상단 왼쪽과 중간: 낮은 복잡성에서는 비생각 모델(non-thinking models)이 더 정확하고 토큰 효율적입니다. 복잡성이 증가함에 따라 추론 모델(reasoning models)은 능가하지만 더 많은 토큰이 필요합니다. 둘 다 중요한 경계를 넘어 더 짧은 추적을 통해 붕괴될 때까지 말입니다. 그래프1: 상단 오른쪽: 올바르게 해결된 경우, Claude 3.7 Thinking은 낮은 복잡성에서는 초기에 답을 찾고, 높은 복잡성에서는 후반에 답을 찾을 경향이 있습니다. 실패한 경우에서는 종종 초기 잘못된 답변에 고정되어 나머지 토큰 예산을 낭비합니다. 두 경우 모두 추론 과정에서 비효율성을 나타냅니다.

이러한 주장과 성능 발전에도 불구하고, LRM의 근본적인 이점과 한계는 충분히 이해되지 않고 있습니다. 여전히 중요한 질문이 남아 있습니다: 이러한 모델은 일반화 가능한 추론을 할 수 있는 능력이 있는 것일까요, 아니면 다양한 형태의 패턴 매칭(pattern matching)을 활용하고 있는 것일까요 [6]? 그들의 성능은 문제의 복잡성이 증가함에 따라 어떻게 확장될까요? 동일한 추론 토큰 계산을 제공받은 경우 비생각 표준 LLM과 비교할 수 있는 방법은 무엇일까요? 가장 중요한 것은 현재의 추론 접근 방식의 고유한 한계와 더 견고한 추론 능력으로 발전하기 위해 필요한 개선 방법이 무엇일까요?

이러한 질문을 조사하는 체계적인 분석의 부족은 현재 평가 패러다임의 한계에 의해 있다고 믿습니다. 기존의 평가는 주로 기존의 수학 및 코드링 벤치마크에 초점을 맞추고 있으며, 이는 가치가 있지만 데이터 오염 문제를 겪어 다양한 환경과 복잡성에 걸쳐 통제된 실험 조건을 허용하지 않습니다. 더욱이, 이러한 평가는 추론 흔적의 구조와 품질에 대한 통찰력을 제공하지 않습니다. 이러한 모델의 추론 행동을 보다 엄격하게 이해하기 위해서는 통제된 실험을 가능하게 하는 환경이 필요합니다.

이 연구에서는 문제의 관점에서 한계 LRM의 추론 메커니즘을 탐구합니다.

복잡성입니다. 표준 벤치마크(for example, 수학 문제) 대신, 우리는 제어 가능한 퍼즐을 채택합니다.
복잡성을 체계적으로 변환하여 퍼즐 요소를 조정하면서
핵심 논리—그리고 해결책과 내부 추론을 모두 조사합니다. (상단 1자). 이러한 퍼즐은: (1)의
복잡성에 대한 세밀한 제어; (2) 기존 벤치마크에서 흔히 발생하는 오염을 피하는 것;
(3) 알고리즘적 추론을 강조하며 명시적으로 제공된 규칙만을 요구하며, (4) 지원
정밀하고 시뮬레이터 기반 평가를 통해 정확한 해결책 검사와 세부적인 실패 분석을 가능하게 합니다.
우리의 경험적 조사는 현재 언어 추론 모델에 대한 여러 주요 발견을 밝혀냅니다.
(LRMs): 먼저, 강화를 통해 학습된 정교한 자기 반영 메커니즘에도 불구하고
학습을 통해 이러한 모델은 계획 작업을 위한 일반화 가능한 문제 해결 능력을 개발하지 못합니다.
특정 복잡성 경계를 넘어 성능이 제로로 붕괴하는 경우도 있습니다. 두 번째로, 우리의 비교는
LRM과 표준 LLM 사이에서 동등한 추론 계산을 통해 세 가지 이유를 밝혀냈습니다.
ing 시스템(Fig. 1, 하단). 더 단순하고 낮은 구성의 문제에 대해 표준 LLM은
문제의 복잡성이 약간 증가함에 따라, 사고 모델은
그러나 문제가 더 긴 구성 깊이로 높은 복잡성을 달성할 때,
두 모델 유형 모두 완전한 성능 붕괴를 경험합니다. 특히,
이 붕괴 시점에서 LRM은 추론 노력을 줄이기 시작합니다. 이는 추론 시간 토큰으로 측정됩니다.
문제의 복잡성이 증가함에도 불구하고 생성 길이 제한(generation length limits)을 능가하는 경향이 있습니다.
아래 중간). 이는 LRM의 추론에서 기본적인 추론 시간 확장 제한을 시사합니다.
마지막으로, 중간 추론 추적 또는
생각은 복잡성 의존적 패턴을 드러냅니다. 단순한 문제에서 추론 모델은 종종
올바른 해결책을 초기에, 그러나 비효율적으로 잘못된 대안을 탐구하는 것을 계속해야 합니다.
중간 복잡도에서는 광범위한 탐색 후만 올바른 솔루션이 나타나게 됩니다.
특정 복잡성 경계를 넘어서는 모델은 완전히 잘못된 경로를 발견하지 못합니다.
해결책(Fig. 1, 오른쪽 하단). 이는 LRM가 제한된 자기 수정 능력을 가지고 있음을 나타냅니다.

이는 가치가 있지만 근본적인 비효율성과 명확한 확장 한계를 드러냅니다.
이러한 결과는 기존의 LRM의 강점과 한계를 강조하며, 질문을 제기합니다.
이러한 시스템의 설계에 중요한 영향을 미치는 추론의 특성에 대해 설명하고
우리의 주요 기여는 다음과 같습니다:
• 기존의 수학 벤치마크에서 LRM의 현재 평가 패러다임에 대해 의문을 제기하고, 문제 복잡성에 대한 제어 가능한 실험을 가능하게 하는 알고리즘 퍼즐 환경을 활용하여 제어된 실험 기반을 설계합니다.
• 최첨단 LRM(e.g. o3-mini, DeepSeek-R1, Claude-3.7-Sonnet-Thinking)은 여전히 일반화 가능한 문제 해결 능력을 개발하지 못하며, 정확도는 다양한 환경에서 특정 복잡성을 넘어 0으로 붕괴합니다.
• 우리는 문제의 복잡성에 대한 LRM의 추론 노력에서 확장 한계가 존재한다는 것을 발견했습니다. 이는 복잡도 지점 후 사고 토큰의 직관에 반하는 감소 경향으로 입증되었습니다.
우리는 최종 정확도에 기반하여 현재 평가 패러다임에 의문을 제기하고, 결정론적 퍼즐 시뮬레이터(deterministic puzzle simulators)의 도움을 받아 사고 추적의 중간 해결에 대한 평가를 확장합니다. 우리의 분석은 문제 복잡도가 증가함에 따라 잘못된 문제와 비교하여 잘못된 문제의 후속 위치에서 올바른 솔루션이 체계적으로 나타나며, 이는 LRM 내의 자기 수정 메커니즘에 대한 정량적인 통찰력을 제공합니다.
• 우리는 명시적인 알고리즘의 혜택을 받지 못하는 LRMs와 퍼즐 유형에 걸쳐 일관되지 않은 추론을 포함한 정확한 계산을 수행하는 능력에 대한 놀라운 한계를 발견했습니다.

2개의 관련 작업

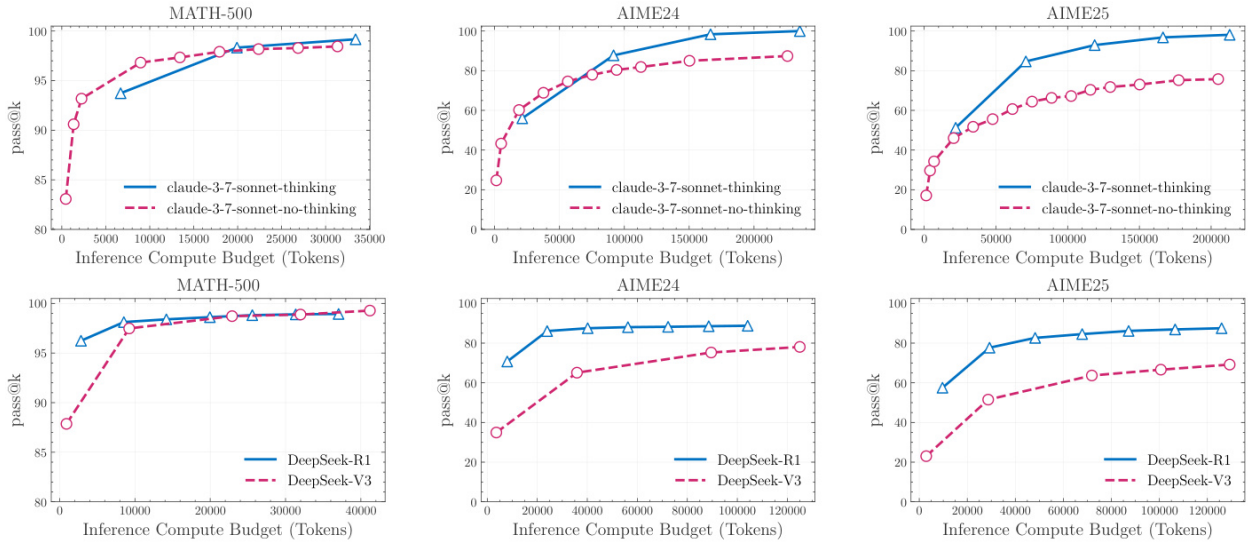
언어 모델에서의 추론(reasoning in language models) 대형 언어 모델(large language models, LLMs)은 방대한 양의 훈련 데이터를 사용하여 여러 비용이 많이 드는 훈련 단계를 겪습니다. 이러한 LLMs는 강력한 압축 능력을 갖춘 유망한 언어 이해를 보여주지만, 그들의 지능과 추론 능력은 여전히 과학적 논쟁의 중요한 주제로 남아 있습니다 [7, 8]. LLMs의 초기 반복 [9, 10, 11]은 추론 벤치마크에서 낮은 성능을 보였습니다 [12, 13, 14, 6]. 이러한 결함을 해결하기 위해, 여러 접근법이 탐구되었으며, 그들 중 공통 주제는 훈련 데이터와 테스트 시간 계산을 모두 "분산"하는 것으로 나타났습니다. 예를 들어, 생각의 체인(chain of Thought, CoT) 생성 [15, 16, 17, 18]과 최종 답변 전에 자기 검증 [19, 20, 21]을 통합하는 것이 모델 성능을 향상시키는 것으로 나타났습니다. 그러나 고품질의 확장 가능한 CoT 데이터를 얻는 것은 그 부족으로 인해 상당히 비싸습니다. 다른 연구 라인은 지도 학습(supervised learning) 또는 강화 학습(reinforcement learning)을 통해 모델을 더 효과적으로 생각하도록 가르치기(supervised learning)로 지도 데이터 부족을 보상하는 데 중점을 둡니다 [22, 23, 24, 25, 26, 27]. 이러한 개선의 주목할 만한 오픈 소스 예는 DeepseekR1 [3]로, 검증 가능한 보상을 가진 RL을 적용하면 모델 성능을 크게 향상시킬 수 있음을 보여주었으며, 이는 OpenAI의 o1 [2]와 같은 폐쇄 모델

과 일치하여 Gemini flash thinking [5], Claude 3.7 Sonnet thinking [4], etc.와 같은 새로운 세대의 언어 모델로 불리우는 대형 추론 모델(large reasoning models, LRMs)로 불리웁니다.

대형 추론 모델(large reasoning models)을 이해하는 것(understanding large reasoning models) 최근 연구에서는 추론 행동의 다양한 측면을 탐구했습니다: 대형 추론 모델(large reasoning models)은 연구자들이 말하는 과도한 사고 현상(overthinking phenomenon)을 통해 생각 추적과 최종 답변 간의 불일치(disrepancy between thought traces and final answers) [28, 29]와 효율성 문제(efficiency concerns)와 같은 발생하는 행동을 보여주었습니다. 여기서 모델은 해를 찾았을 때에도 명확하고 불필요한 출력을 생성하여 상당한 추론 계산 오버헤드를 생성합니다. 이 연구에서는 작업 복잡성(task complexity)을 넘어 모델이 얼마나 생각하는지를 체계적으로 분석합니다. 최근 Ballon et al. [34]는 새로운 LRMs에서 사고가 수학 문제에서 증가할 때 정확도가 일반적으로 감소함을 보여주었으며, 반면에 통제된 퍼즐 환경(controlled puzzle environment)에서 어려움이 특정 수준을 넘어 모델이 더 적게 생각하기 시작할 때, 사고와 작업 복잡성(task complexity)의 반대의 연관성이 약간까지 발생하는 것을 관찰했습니다. Yue et al. [35]는 강화 학습이 새로운 추론 패턴을 이끌어내는지, 추론 모델 vs 비합리 모델의 pass@k가 동일한 포인트로 수렴하는지를 묻었습니다. 또한, MATH-500에서는 pass@k가 추론 모델 vs 비합리 모델에 가깝다는 것을 관찰했지만, 퍼즐의 중간 및 높은 복잡성(medium and high complexity) 하에서 서로 다른 패턴을 관찰하였으며, 이는 일반 평가에서 사용되는 정규 수학 벤치마크에서 쉽게 관찰할 수 없습니다.

제어 가능한 평가 환경(controllable evaluation environments) 이전 연구들이 언어 모델의 추론 능력을 평가하기 위해 수학적 문제에 초점을 맞추었을 때와 달리, 이 연구는 제어 가능한 퍼즐 환경(controllable puzzle environments)을 도입합니다. 이러한 환경은 문제의 복잡성을 정밀하게 조작하면서도 일관된 논리적 과정을 유지하여 추론 패턴과 한계에 대한 더 엄격한 분석을 가능하게 합니다. 제어 가능한 환경(controllable environments)은 문헌에서 드니다 [12, 36, 37]. 그러나 우리의 주요 목표는 새로운 벤치마크를 제안하는 것이 아닙니다. 대신, 우리는 이러한 벤치마크를 언어 모델의 추론 능력을 이해하기 위한 실험 설계 도구로 사용합니다. Valmeekam et al. [38]의 밀접하게 관련된 연구는 o1-모델이 이전 모델에 비해 상당한 성능 향상을 보여주는 것을 보여줍니다. 우리의 연구는 사고/비상식 모델(thinking/non-thinking models)의 쌍을 조사하는 것과 같은 추가적인 통찰력을 제공합니다 (e.g., DeepSeek-R1/V3, Claude 3.7 Sonnet thinking/nonthinking). 또한, 우리는 LRM의 추론 추적을 더 깊이 연구하여 다양한 복잡도 수준에서 다양한 행동을 드러냅니다.

전반적으로, 최근 LRMs의 유망한 결과는 다음과 같은 중요한 질문을 제기합니다: 이전에 보고된 LLMs의 한계가 얼마나 개선되었을까요? 이 연구에서는 이러한 LRMs의 성능을 단순히 측정하는 것 이상으로 넘어가며, 이러한 LRMs가 다양한 복잡성의 문제를 얼마나 잘 해결하는지를 분석하고 추론 과정의 속성을 조사합니다.

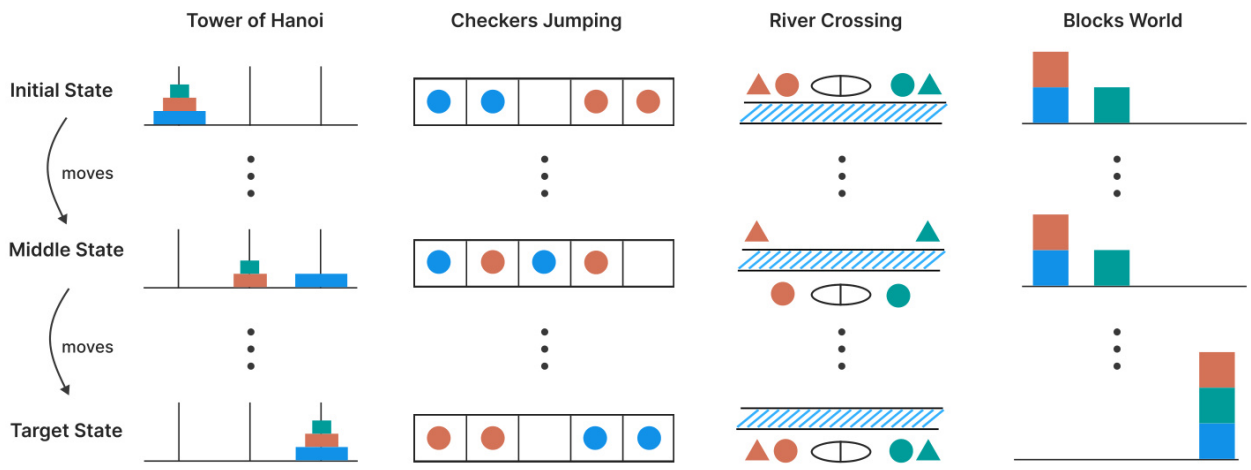


그래프 2: 수학 벤치마크(mathematical benchmarks) 전반에 걸쳐 생각 모델과 비 생각 모델의 비교 분석은 불일치한 성능 패턴을 나타냅니다. MATH-500 데이터셋(dataset)의 결과는 두 유형의 모델 간 비교 가능한 성능을 보여주지만, 생각 모델은 AIME24 및 AIME25 벤치마크에서 우수한 성능을 보여줍니다. 또한, AIME24에서 AIME25로의 성능 저하가 관찰된 것은 이러한 벤치마크가 데이터 손상 문제에 취약하다는 것을 강조합니다.

3 수학과 퍼즐 환경

현재, 최근 RL 기반 사고 모델(RL-based thinking models)에서 관찰된 성능 향상이 기존 수학적 벤치마크 데이터의 노출 증가, 사고 토큰에 할당된 더 큰 추론 계산, 또는 RL 기반 훈련에 의해 개발된 추론 능력으로 인해 발생하는 것일까요? 최근 연구 [35, 39]에서는 기존 수학 벤치마크(RL-based thinking models)와 RL 기반 사고 모델(RL-based thinking models)의 상한 능력(pass@k)과 비생각 표준 LLM 대사자(non-thinking standard LLM counterparts)를 비교하여 이 질문을 탐구했습니다. 연구자들은 동등한 추론 토큰 예산 하에서 비생각 LLM(non-thinking LLMs)이 MATH500 [40] 및 AIME24 [41]와 같은 벤치마크에서 사고 모델(thinking models)과 비교할 수 있는 성능을 마침내 달성할 수 있음을 보여주었습니다. 또한, Claude-3.7-Sonnet(vs. without thinking)와 DeepSeek(R1 vs. V3)와 같은 한계 LRM의 비교 분석을 수행했습니다. 우리의 결과(Fig. 2)는 MATH500 데이터셋(dataset)에서 비생각 모델(thinking models)의 pass@k 성능이 동일한 추론 토큰 예산을 제공했을 때 비생각 모델(non-thinking counterparts)과 비교할 수 있음을 확인합니다. 그러나, 우리는 AIME24 벤치마크에서 이 성능 격차가 AIME25에서 더욱 넓어지는 것을 관찰했습니다. 이러한 넓어지는 격차는 해석적 도전을 제시합니다. 이는 (1) 복잡성이 증가하여 더 정교한 추론 프로세스를 요구하여 더 복잡한 문제에 대한 사고 모델의 진정한 이점을 드러낼 수 있게 하며, (2) 새로운 벤치마크(particularly AIME25)에서 데이터 공해를 줄이기 때문입니다. 흥미롭게도, AIME25에서의 인간 성능은 AIME24보다 실제로 AIME24보다 더 높았으며, AIME25가 더 복잡할 수 있음을 시사합니다. 그러나 모델은

AIME24보다 AIME25에서 더 나은 성능을 발휘하여 한계 LRM의 훈련 중 데이터 공해를 시사할 수 있습니다.



그래프 3: 네 개의 퍼즐 환경의 설명 columns는 퍼즐의 초기 상태(상단)에서 중간 상태(간단)에 이르기까지 목표 상태(저단)로의 진전을 나타냅니다: 하노이의 탑(disc transfer across pegs), 체커스 ジャンピング(checkers jumping), 리버 크로싱(river crossing), 블록 월드(blocks world, 병합 재구성)

3.1 퍼즐 환경

우리는 구성 깊이, 계획 복잡성, 분포 설정을 포함한 네 가지 제어 가능한 퍼즐에서 LRM 추론을 평가합니다. 퍼즐은 아래 정의되며, 그래프 3에서 설명됩니다.

하노이의 탑(Tower of Hanoi)은 다양한 크기의 세 개의 페그와 n 디스크를 크기 순서에 따라 첫 페그에 쌓인 퍼즐입니다. 목표는 첫 페그에서 세 번째 페그로 모든 디스크를 전이하는 것입니다. 유효한 이동은 한 번에 한 개의 디스크만 이동하고, 페그에서 위쪽 디스크만 취하며, 작은 디스크의 위에는 더 큰 디스크를 절대로 놓지 않는 것을 포함합니다. 이 작업의 난이도는 초기 디스크 수에 의해 제어될 수 있으며, n 초기 디스크를 사용한 최소 이동 수는 $2^n - 1$일 것입니다. 그러나 이 작업에서는 최종 해의 최적성에 대한 평가를 하지 않고, 각 이동의 올바름과 목표 상태에 도달하는 것을 측정합니다.

체커 ジャンピング(Checker Jumping)은 빨간 체커(red checkers), 파란 체커(blue checkers), 그리고 선에서 단일 빈 공간을 정렬하는 일차원 퍼즐입니다. 목표는 모든 빨간 체커(red checkers), 파란 체커(blue checkers)의 위치를 교환하여 초기 설정을 효과적으로 반영하는 것입니다. 유효한 이동에는 체커를 인접한 빈 공간으로 슬라이딩하거나 반대 색의 정확히 한 체커를 넘어 빈 공간으로 착륙하는 것이 포함됩니다. 체커는 퍼즐 프로세스에서 역으로 이동할 수 없습니다. 이 작업의 복잡성은 체커의 수에 의해 제어될 수 있습니다: $2n$의 체커를 사용하면 최소한의 이동 수는 $(n+1)^2 - 1$일 것입니다. 강 건너(River Crossing)는 배를 사용하여 강을 건너야 하는

n 배우와 그에 대응하는 n 에이전트를 포함하는 제약 만족도 계획 퍼즐입니다. 목표는 모든 $2n$ 개인을 왼쪽 연안에서 오른쪽 연안으로 이동시키는 것입니다. 배는 최대 k 개인을 수송할 수 있으며 비활성으로 이동할 수 없습니다. 유효하지 않은 상황은 각 에이전트가 경쟁하는 에이전트로부터 고객을 보호해야 하기 때문에, 에이전트가 자신의 에이전트가 없는 다른 에이전트의 존재에 있을 때 발생합니다. 이 작업의 복잡성은 또한 에이전트/에이전트 쌍이 존재하는 수에 의해 제어될 수 있습니다. $n = 2, n = 3$ 쌍을 위해, 우리는 $k = 2$의 배 용량을 사용하고, 더 많은 수의 쌍을 위해 $k = 3$를 사용합니다.

블록 월드(Blocks World)는 초기 구성에서 특정 목표 구성으로 블록을 재구성하는 블록 スタキング 퍼즐입니다. 목표는 이 변환을 위해 필요한 최소한의 이동 수를 찾는 것입니다. 유효한 이동은 모든 スタ킹의 최상위 블록에 제한되며, 이는 빈 スタ킹이나 다른 블럭의 위에 배치될 수 있습니다. 이 작업의 복잡성은 현재 존재하는 블록 수에 의해 제어될 수 있습니다.

하노이 슈커 점프 블록 월드 리버크로싱
100F00 100 100F 100 4
80 클 lthide 37 onet 80 80 80 +
60 1 60 lthnde 3 oet 60 Claude 3.7 Sonnet Clthink 3,7y onet (+침묵)
40 40 40 òo. LA
20 claud 3ine. -. △ 20 0lio C 0 20 C 20Cla 3oo △ △ 12345 6 7 8 10 15 20 12345 67 8 10 15 20 2 10 20 30 40 50 2345678 10 15 20 복잡성(discs의 수가) 복잡성(Checkers의 수가) 복잡성(블럭의 수가) 복잡성(인원의 수가)
100F 100FA 100F 100 +
80 DeepSeek-R1 80 809 80 40$ DeepSeek-R1 60 DeepSeek-R1 60 DeepSeek-R1
20 O 20DepSeek-V3 20 20Deepeek-V3 DeepSeek-V3 DeepSeekv3 0 ..... 0 0 A 0000-00-0 △ o9...0...0....1.1..lo 0 0 O1O-O1O 15 20 12345 678 10 15 20 2 10 20 30 40 50 2345 678 10 15 20 복잡성(discs의 수가) 복잡성(Checkers의 수가) 복잡성(블럭의 수가) 복잡성(인원의 수가)

4번의 실험과 결과

4.1 실험적 설정

대부분의 실험은 추론 모델(reasoning models)과 Claude 3.7 Sonnet(thoughting/non-thinking) 및 DeepSeek-R1/V3와 같은 비생각 모델(non-thinking counterparts)에서 수행됩니다. 이러한 모델은 OpenAI의 o- 시리즈(o-series)와 달리 생각 토큰(thoughting tokens)에 접근할 수 있기 때문에 선택되었습니다. 최종 정확성에만 집중한 실험에서는 o- 시리즈 모델(o-series models)에서의 결과를 보고합니다. Claude 3.7 Sonnet 모델(Claude 3.7 Sonnet models)에서는 최대 토큰 예산(64k)을 허용합니다. 마찬가지로, 지역 서버의 DeepSeek-R1/V3 모델(DeepSeek-R1/V3

models)에서는 최대 길이가 최대 64k 토큰(64k tokens)으로 허용됩니다. 각 퍼즐 인스턴스(puzzle instances)에 대해 25개의 샘플을 생성하고 각각의 모델의 평균 성능을 보고합니다. 실험 설정과 결과에 대한 포괄적인 세부 사항은 부재에 있습니다.

4.2 복잡성은 추론에 어떤 영향을 미칠까요?

4.2.1 복잡성의 세 가지 경계

2차 그래프의 관찰에 의해 동기를 부여하여 문제 복잡성(problem complexity)이 추론 행동에 미치는 영향을 체계적으로 조사했습니다. 우리는 제어된 퍼즐 환경에서 사고 및 비 사고 모델 쌍을 비교하는 실험을 수행했습니다. 우리의 분석은 동일한 모델 백본을 가진 일치하는 LLM 쌍, 특히 Claude-3.7-Sonnet (w. vs. w/o thinking) 및 DeepSeek (R1 vs. V3)에 초점을 맞추었습니다. 각 퍼즐에서는 문제 크기 N(disc count, checker count, block count, or crossing elements)를 조작하여 복잡성을 변경합니다.

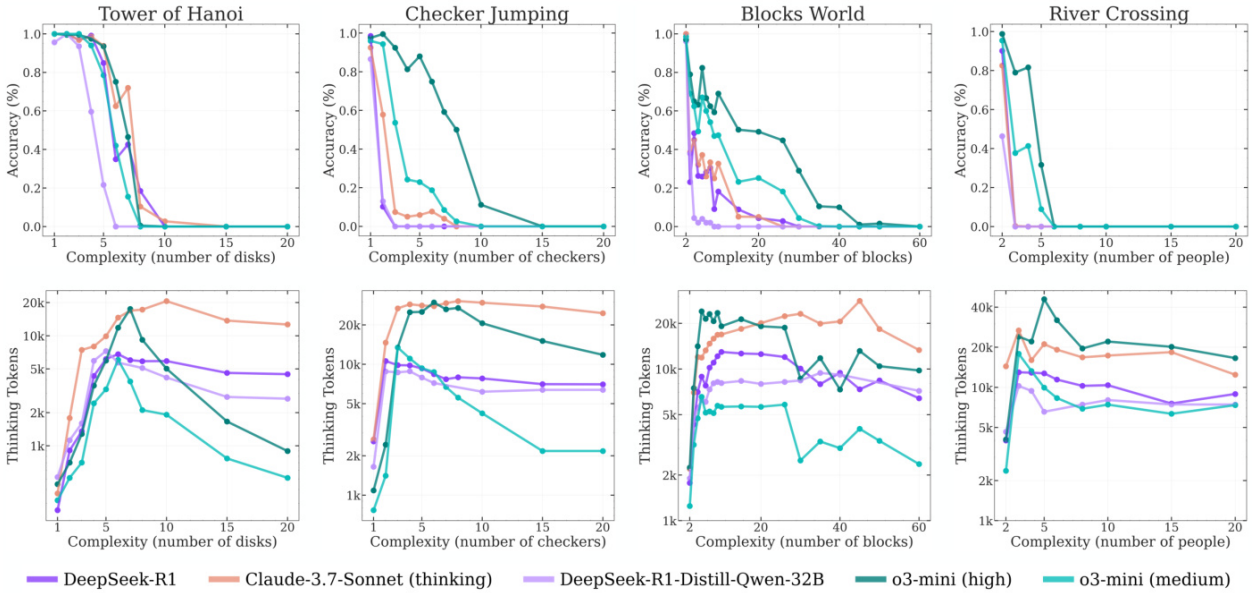
그래프.4는 모든 퍼즐 환경에서 문제 복잡성의 함수로 두 모델 유형의 정확성을 제시합니다. 이를 완충하여, 그래프.5는 이러한 모델 쌍의 상한 성능 능력(pass@k)을 동등한 추론 토큰 계산(모든 퍼즐에 걸친 평균) 하에서 나타냅니다. 이는 수학적 벤치마크(그래프.2)에서 통제된 퍼즐 환경으로의 이전 분석을 확장합니다. 이러한 두 그래프의 결과는 수학적 관찰과 달리 복잡성 측면에서 이러한 모델의 행동에 세 가지 체계가 존재함을 보여줍니다. 문제 복잡성이 낮은 첫번째 체계에서는 비생각 모델이 더 토큰 효율적인 추론을 가진 사고 모델과 비교할 수 있는 성능을 얻을 수 있음을 관찰합니다.

100 oo0oopoopo △ 100 100 80 △ Clande 317 oneet 80 Clande 37 onet 80
20 2 S α .0:0:0 0 0AO111b16i2O1ORO4O:O:O:OO 0 40000 80000 120000 0 40000 80000120000160000 0 80000 160000 240000 예산(기호) 계산 예산(기호) 계산 예산(기호) 계산
100F 100 100 s1rlrld 80 X DeepSeek-R1 80 DeepSeek-R1 80
d 0 0 5 60r$ A 2 O S 6f0 DeepSeek-V3 200 20 DeepSeek-V3 20 DeepSeek-R1 DeepSeek-V3 0OroioRrirlo 20.....0 0 0 20000 40000 60000 0 25000 50000 75000 100000 0 40000 80000 120000 예산(기호) 계산 예산(기호) 계산 예산(기호) 계산 예산(기호)

중간 복잡도(medium complexity)를 가진 두 번째 체계에서는 긴 사슬(chain-of-thought)을 생성할 수 있는 추론 모델의 장점이 나타나기 시작하며, 모델 쌍 간의 성능 격차가 증가합니다. 가장 흥미로운 체계는 문제 복잡도가 더 높고 두 모델의 성능이 제로로 붕괴된 세 번째 체계입니다. 결과는 생각 모델이 이러한 붕괴를 지연할 때, 비생각 모델이 궁극적으로 비생각 모델과 같은 근본적인 한계를 겪는다는 것을 보여줍니다.

4.2.2 추론 모델의 붕괴

다음으로, 사고 토큰(thinking tokens)을 장착한 다양한 전문 추론 모델이 증가하는 문제 복잡성에 어떻게 반응하는지를 조사합니다. 우리의 실험은 o3-mini(평균 및 높은 구성), DeepSeek-R1, DeepSeek-R1-Qwen-32B, Claude-3.7- Sonnet(생각)의 다섯 가지 최첨단 사고 모델(state-of-the-art thinking models)을 평가합니다. 그림 6은 다양한 복잡도 수준에서 이러한 모델의 정확도(상위)와 사고 토큰 사용(bottom) 측면에서의 성능을 보여줍니다. 결과는 모든 추론 모델이 복잡성에 대해 유사한 패턴을 보인다는 것을 보여줍니다: 문제 복잡도가 증가함에 따라 정확도는 점진적으로 감소하여 모델별 복잡도 경계를 넘어 완전한 붕괴(제로 정확도)에 도달합니다. 추론적 사고 토큰 계산(inference thinking token compute)의 분석은 또한 이러한 모델이 학습한 사고 토큰 할당(thinking token allocation)에서 흥미로운 패턴을 보여줍니다. 추론 모델이 처음에는 문제 복잡성과 비례하여 사고 토큰(thinking tokens)을 비례적으로 증가시키는 것을 관찰합니다. 그러나 정확도 붕괴 지점(precision collapse point)과 밀접하게 일치하는 중요한 경계에 도달했을 때, 모델은 증가하는 문제 복잡에도 불구하고 추론 노력을 반관되게 줄이기 시작합니다. 이 현상은 o3-mini 변형에서 가장 많이 나타나고, Claude-3.7-Sonnet(생각) 모델에서 더 덜 심각합니다. 특히, 충분한 추론 예산이 있는 생성 경계에서 잘 작동하지만, 문제가 복잡해지면서 추가적인 추론 계산(inference compute)을 활용하지 못합니다. 이러한 행동은 문제 복잡성에 대한 현재 추론 모델의 사고 능력에 대한 근본적인 확장 제한을 시사합니다.

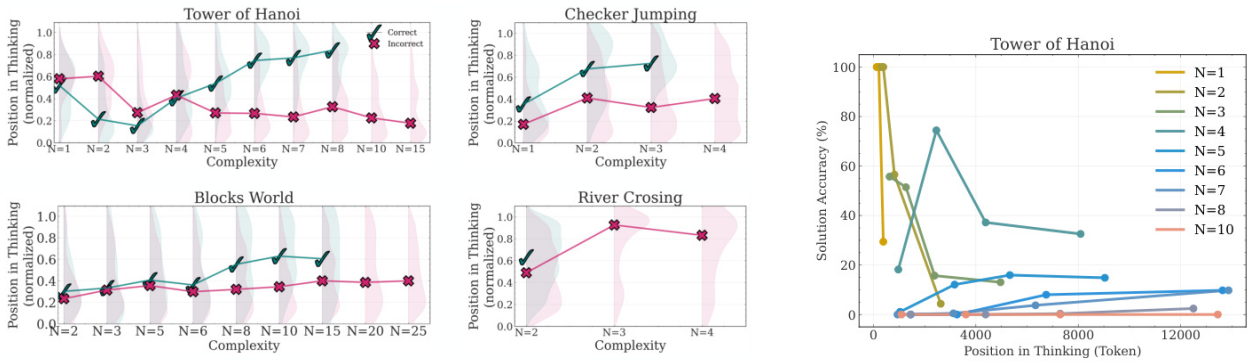


그래프 6: 퍼즐 환경에서 추론 모델의 정확성과 사고 토큰 대 문제 복잡성(problem complexity) 추론 모델의 복잡성이 증가함에 따라, 복잡성이 증가함에 따라 추론 모델은 처음에는 더 많은 토큰을 사용하지만, 정확도는 점차 감소하여 추론이 붕괴하는 중요한 시점에 도달합니다. performance drops sharply and reasoning effort decreases.

4.3 추론 모델의 생각 내에서 무슨 일이 일어난 것일까요?

추론 모델의 사고 과정에 대한 더 깊은 통찰을 얻기 위해, 우리는 그들의 추론 추적(reduction traces)에 대한 세밀한 분석을 수행했습니다. 1차 그래프에서 보여드린 바와 같이, 퍼즐 환경(puzzle environments)의 설정은 최종 답변을 넘어 이러한 모델이 생성한 추론 추적(thoughts)에 대한 더 자세한 통찰을 얻을 수 있게 합니다. 우리는 퍼즐 시뮬레이터(puzzle simulators)의 도움을 받아 모델의 생각 내에서 탐구된 중간 솔루션(intermediate solutions)을 추출하고 분석합니다. 우리의 조사는 이러한 중간 솔루션의 패턴과 특성, 추론 과정의 순차적 위치에 대한 올바른ness, 그리고 이러한 패턴이 증가하는 문제 복잡성과 함께 어떻게 진화하는지를 조사합니다. 이 분석을 위해, 우리는 퍼즐 세트 전반에 걸쳐 Claude-3.7-Sonnet-Thinking에 의해 생성된 추론 추적(reduction traces)에 중점을 둡니다. traces 내에서 식별된 각각의 중간 솔루션에 대해 (1) 추론 추적(reduction trace)의 상대적 위치(상태를 총 생각 길이로 정규화), (2) 우리의 퍼즐 시뮬레이터(puzzle simulators)가 검증한 올바른ness, (3) 해당 문제의 복잡성을 기록했습니다. 이는 추론 과정 전반에 걸쳐 솔루션 개발의 진행과 정확성을 특징짓을 수 있게 합니다.

그래프 7a(Figure 7a)는 모든 퍼즐 환경에서 생각 내 중간 솔루션의 위치, 그 올바함, 문제 복잡성과 사이의 관계를 보여줍니다. 추론 추적(researching traces) 분석은 또한 앞서 논의된 복잡성의 세 가지 Regime를 추가로 검증합니다. 단순한 문제에서는 추론 모델(researching models)이 초기에 올바른 솔루션을 찾는 경우가 많지만, 그 후에는 올바른 솔루션을 탐색을 계속합니다. 불일치한 솔루션의 분포(red)는 올바른 솔루션(green)과 비교하여 생각의 종단으로 더 upward로 이동합니다. 이 현상은 문헌에서 과도한 생각(overthinking)이라고 불리며, 계산의 낭비로 이어집니다. 문제는 비교적 더 복잡해지면서, 이 트렌드는 역전이 됩니다: 모델은 먼저 불일치한 솔루션을 탐색하고, 주로 이후에 생각에서 올바른 솔루션을 얻습니다. 이번에는 불일치한 솔루션의 분포(red)는 올바른 솔루션(green)과 비교하여 더 하향식으로 이동합니다. 마지막으로, 더 높은 복잡성으로 구성된 문제에서는 붕괴가 나타나서 모델이 생각 내에서 올바른 솔루션을 생성하지 못한다는 것을 의미합니다.



(a) (b)

그래프 7: 왼쪽과 중간: 다양한 복잡도 수준의 네 개의 퍼즐에서 추론 추적 추적 내 중간 해의 위치와 올바성. ✓는 올바른 해를 나타내고, x는 불일치한 해를 나타내며, 분포 밀도는 어둠으로 나타냅니다. 오른쪽: 다양한 복잡도 수준에서 하노이 타워를 위한 사고의 위치 대 해결 정확

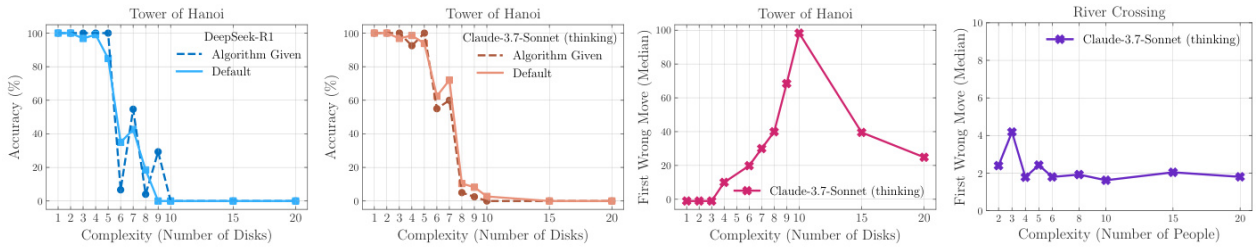
도. 간단한 문제(N=1-3)에서는 시간이 지남에 따라 초기 정확도가 감소하는 것을, 중간 문제(N=4-7)에서는 지속적인 추론과 함께 약간의 정확도 개선을 나타내며, 복잡한 문제(N $\geq$ 8)는 일관되게 제로 근사 정확도를 나타내며, 완전한 추론 실패를 나타냅니다.

7b 그래프는 하노이 탑(Tower of Hanoi) 환경에서 생각의 순차적 세그먼트(bins) 내에서 해결 확률의 상호 보완적 분석을 제시합니다. 더 간단한 문제(smaller N)에서는 해결 확률이 생각의 진행에 따라 감소하거나 진동하는 경향이 있으며, 이는 과도한 생각 현상의 추가적인 증거를 제공합니다. 그러나 이러한 경향은 더 복잡한 문제에서 변화하며, 해결 확률은 생각의 진행에 따라 일정한 한계까지 증가합니다. 이러한 복잡성 한계를 넘어서는 "붕괴 모드(collapse mode)"에서는 정확도가 제로입니다.

4.4 공개 질문: 추론 모델의 혼란스러운 행동

이 부분에서는 정확한 문제 해결 단계를 실행하는 데 있어 추론 모델의 한계에 대한 놀라운 결과를 제시하고, 이동 횟수에 따라 모델의 다양한 행동을 보여줍니다.

8a와 8b의 그래프에서 입증된 바와 같이, 하노이 탑 환경(Tower of Hanoi environment)에서 모델이 특정 단계만 실행해야 할 때에도 성능이 향상되지 않고, 관찰된 붕괴는 여전히 대략 동일한 시점에서 발생합니다. 이는 given 알고리즘을 단순히 실행하는 것보다 훨씬 더 많은 계산(search and verification, 어.g.)을 필요로 하기 때문에 주목할 만한 일입니다. 이는 검증과 문제 해결을 위한 다음 논리적 단계에서 추론 모델의 한계를 더욱 강조하며, 이러한 모델의 상징적 조작 능력을 이해하기 위한 추가 연구가 필요하다는 것을 시사합니다 [44, 6]. 또한, 8c와 8d의 그래프에서 Claude 3.7 Sonnet 사고 모델과 매우 다른 행동을 관찰합니다. 하노이 탑 환경(Tower of Hanoi environment)에서 모델의 제안된 솔루션에서의 첫 오류는 종종 훨씬 후반에 발생하며, 예를 들어 (N=10)에 대해 100번의 이동 정도입니다. 이는 River Crossing 환경(River Crossing environment)에서 모델이 4번의 이동 전까지 유효한 솔루션을 생성할 수 있는 경우에 해당합니다. 이 모델은 31번의 이동이 필요한 하노이 탑(Tower of Hanoi)을 해결할 때도 거의 완벽한 정확도를 달성하지만, 11번의 이동의 솔루션을 가진 (N=3)의 River Crossing 퍼즐을 해결할 때는 실패합니다. 이는 N $>$ 2의 River Crossing 예제가 웹상에서 부족하다는 것을 시사하며, 이는 훈련 중 LRM이 이러한 인스턴스를 자주 만나거나 기억하지 못했을 수 있음을 의미합니다.



(a) (b) (c) (d)

그래프 8: (a) & (b) 프롬프트에서 솔루션 알고리즘을 제공했지만, 실행 실패는 유사한 시점에서 발생하여 논리적 단계 실행에서 추론 모델의 한계를 강조합니다. (c) & (d) 특히, 클로드 3.7 ソネット 모델(Claude 3.7 Sonnet model)은 하노이 타워(Tower of Hanoi)에서 초기 오류와 비교하여 훨씬 더 긴 오류 없는 시나리오를 보여줍니다.

5단 결론

이 논문에서 우리는 제어 가능한 퍼즐 환경을 사용하여 문제 복잡성 관점에서 한계 큰 추론 모델(large reasoning models, LRMs)을 체계적으로 검토합니다. 우리의 발견은 현재 모델의 근본적인 한계를 드러냅니다: 정교한 자기 반사 메커니즘에도 불구하고, 이러한 모델은 특정 복잡성 경계를 넘어 일반화 가능한 추론 능력을 개발하지 못합니다. 우리는 세 가지 다른 추론 방식을 식별했습니다: 표준 LLMs는 낮은 복잡성에서는 LRMs를 능가하며, LRMs는 중간 복잡성에서는 뛰어난 성능을 발휘하며, 높은 복잡성에서는 모두 붕괴합니다. 특히 우려되는 것은 문제가 중요한 복잡성을 접근할 때 추론 노력의 반intuitive 감소로 인해 LRMs의 내재된 계산 확장 한계가 나타났음을 시사합니다. 추론 추적 추적의 세부 분석은 더 많은 복잡성 의존적 추론 패턴을 노출시켰으며, 단순한 문제에서 비효율적인 "overthinking"을 넘어 복잡한 문제에서 완전한 실패를 달성하는 것으로 나타났습니다. 이러한 통찰은 LRM 능력에 대한 기존의 가정을 도전하고, 현재 접근 방식이 일반화 가능한 추론에 대한 근본적인 장애물에 직면할 수 있음을 시사합니다. 마침내, 우리는 LRMs에 대해 몇 가지 놀라운 결과를 제시하여 미래 연구에 대한 여러 질문을 열어주었습니다. 특히, 정확한 계산을 수행하는 데 있어 한계를 관찰했습니다. 예를 들어,Hanoi Tower의 해결 알고리즘을 모델에게 제공했을 때, 이 퍼즐에서의 성능이 개선되지 않았습니다. 또한, 모델의 첫 실패 이동을 조사한 결과는 놀라운 행동을 나타냈습니다. 예를 들어,Hanoi Tower의 해결에서 최대 100번의 올바른 동작을 수행할 수 있었지만, River Crossing 퍼즐에서 5번 이상의 올바른 동작을 제공하지 못했습니다. 우리는 이러한 결과가 이러한 시스템의 추론 능력을 위한 미래 조사를 위한 길을 열 수 있다고 믿습니다.

제한

우리는 우리의 연구가 한계를 가지고 있음을 인정합니다. 퍼즐 환경(puzzle environments)은 문제 복잡성에 대한 세밀한 제어를 통해 제어된 실험을 가능하게 하면서도, 추론 작업의 좁은 조

각을 나타냅니다. 또한 현실 세계나 지식 집약적 추론 문제의 다양성을 포착하지 않을 수 있습니다. 대부분의 실험이 폐쇄 경계 LRMs에 대한 블랙박스 API 접근에 의존한다는 것은 주목할 만한 사실입니다. 이는 내부 상태나 구조적 구성 요소를 분석하는 능력을 제한합니다. 또한, 결정론적 퍼즐 시뮬레이터(deterministic puzzle simulators)의 사용은 추론이 단계별로 완벽하게 검증될 수 있다고 가정합니다. 그러나, 더 덜 구조화된 도메인에서는 이러한 정밀한 검증이 실현 불가능할 수 있어 이 분석의 전이 가능성을 다른 더 일반화 가능한 추론으로 제한합니다.

인정

저자들은 Scott Hoang, Yichen Jiang, Minsik Cho, Mohammad Sekhavat, David Harrison, Mohammadreza Armandpour, Devi Krishna에 대한 가치 있는 피드백과 지원에 대해 감사합니다.

참조

[1] Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 시스템 카드. arXiv preprint arXiv:2412.16720, 2024.
[2] OpenAI. openai o1. Jan 2024. [3] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: 강화 학습을 통해 llms의 추론 능력을 촉진하는. arXiv preprint arXiv:2501.12948, 2025.
[4] 인문학적. 클로이드 3.7 소나트. 2월 2025. [5] 구글. 게미니 플라스틱 사고. 구글 AI 블로그, 1월 2025. [6] 세이드 이만 미즈라데, 케이반 알리자데, 후만 샤로키, 오토닐 투셀, 샘이 벤지오, 메헤르다드 파라지타바. GSM-상징적: 대형 언어 모델에서 수학적 추론의 한계를 이해하는. 학습 표현에 대한 열 treizinth 국제 회의, 2025.
[7] 프랑카오 콜레(Francois Chollet), 마이크 노프(Mike Knoop), 그래포리 컴라드(G Gregory Kamradt), 브라이언 랜더스(Brian Landers), 헨리 핑카드(Henry Pinkard). arc-agi-2: 경계 ai 추론 시스템의 새로운 도전. arXiv preprint arXiv:2505.11831, 2025. [8] 게리 마쿠스(Gary Marcus). 지난 3개월, 특히 심층 탐색 시대, "심층 학습이 장벽에 부딪히고 있다"는 것을 다섯 가지 방법으로 입증했습니다. 마쿠스(Markus) AI(Substack), 2월 2025. 블로깅 post. [9] 마라이 아빈(Marah I Abdin), 샘 아데 제코브스(Sam Ade Jacobs), 어마르 아흐메드 아완(Ammar Ahmad Awan), 조티 아네야(Joti Aneja), 아흐메드 아와달라(Ahmed Awadallah), 해니 아와달라(Hany Awadalla), Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat S. Behl, et. al. Phi-3 technical report: A highly capable language model locally on your phone. CoRR, abs/2404.14219, 2024.
[10] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Molnár, Guillaume Lample, Lucile

Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. CoRR, abs/ 2310.06825, 2023.

[11] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, et al. 라마 3 개의 모델 무리. CoRR, abs/2407.21783, 2024.

[12] 노하 지리, 시밍 루, 멜라니 스클러, 샤이랑 로렌 리, 리바이 존, bill yuchen 린, 션 웰크, peter west, 찰라 바가와투라, 론난 le bras, jen d. hwang, somya sanyal, 샤이랑 렌, ellyson elthinger, zaid harchaoui, yedin choi. belief and fate: limits of transformers on compositionality. in Alice oh, tristan naumann, amir globerson, kate saenko, moraitz hardt, and sergei levine, editors, advances in neural information processing systems 36: annual conference on neural information processing systems 2023, neurips 2023, new orleans, l a, usa, december 10 - 16, 2023, 2023.

[13] R. Thomas McCoy, Shunyu Yao, Dan Friedman, Matthew Hardy, and Thomas L. Griffiths. Embers of autoregression: Understanding large language models through the problem they are trained to solve, 2023.

[14] Marianna Nezhurina, Lucia Cipolina-Kun, Mehdi Cherti, and Jenia Jitsev. Alice in wonderland: Simple tasks showing complete reasoning breakdown in state-of-the-art large language models. arXiv preprint arXiv:2406.02061, 2024.

[15] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, Denny Zhou. 사고의 순환 유도(chain-of-thought prompting)는 대규모 언어 모델에서 추론을 유도합니다. Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, A. Oh, 편집자, 신경 정보 처리 시스템의 발전 35: 신경 정보 처리 시스템 연간 회의 2022, NeurIPS 2022, 뉴올리언즈, LA, USA, 11월 28 - 12월 9, 2022, 2022.

[16] 메헤란 Kazemi, Najoung Kim, Deepti Bhatia, Xin Xu, and Deepak Ramachandran. Lambada: 자연어의 자동 추론을 위한 역방향 추론. arXiv preprint arXiv:2212.13894, 2022.

[17] Hattie Zhou, Azade Nova, Hugo Larochelle, Aaron Courville, Behnam Neyshabur, and

Hanie Sedghi. 컨텍스트 학습을 통한 알고리즘 추론을 가르치는 방법. arXiv preprint arXiv: 2211.09066, 2022.
[18] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 대형 언어 모델(large language models)은 제로샷 추론기(zero-shot reasoners)입니다. 신경 정보 처리 시스템의 발전, 35:22199–22213, 2022.
[19] yixuan weng, minjun zhu, fei Xia, bin li, shizhu he, shengping liu, bin sun, kang liu, jun 샤오. 대형 언어 모델(large language models)은 자기 검증을 가진 더 나은 추론자입니다. Houda Bouamor, Juan Pino, and Kalika Bali, editors, Findings of the Association for Computational Linguistics: EMNLP 2023, pages 2550–2575, 싱가폴, 2023. Association for Computational Linguistics.
[20] Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. 단계 인식 검증기(step-aware verifier)를 사용하여 언어 모델을 더 나은 추론기로 만듭니다. 컴퓨팅 언어학 협회 61회 연간 회의 Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 5315–5333, 2023.
[21] Eric Zhao, Pranjal Awasthi, and Sreenivas Gollapudi. 샘플링, 검토 및 스케일: 스케일링 검증을 통해 효과적인 추론 시간 탐색. arXiv preprint arXiv:2502.01839, 2025.
[22] Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. STar: 추론과 연결된 추론. Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, Advances in Neural Information Processing Systems, 2022.
[23] Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, Vaishnavh Nagarajan. 강연 전에 생각을 해 보세요: language models with pause tokens. in The Twelfth International Conference on Learning Representations, 2024.
[24] David Herel and Tomas Mikolov. 언어 모델링을 위한 사고 토큰. ArXiv, abs/2405.08644, 2024.
[25] 지hong shao, peiyi wang, runxin xu qyao qhu, junxiao song, ingsuan chang, y.k. li, y. wu, and daia guo. deepseekmath: pushing the limits of mathematical reasoning in open language models, 2024.
[26] Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordoni, Siva Reddy, Aaron Courville, Nicolas Le Roux. Vineppo: refined credit assignment through unlocking rl potential for llm reasoning, 2024.
[27] Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tülu 3: 개방형 언어 모델 훈련 후 한계를 밀어내는 것. ArXiv, abs/2411.15124, 2024.
[28] Yanda Chen, Joe Benton, Ansh Radhakrishnan, Jonathan Uesato, Carson Denison, John Schulman, Arushi Somani, Peter Hase, Misha Wagner, Fabien Roger, et al. 추론 모델은 항상 자

기 생각을 말하지 않습니다. arXiv preprint arXiv:2505.05410, 2025.
[29] Dacheng Li, Shiyi Cao, Tyler Griggs, Shu Liu, Xiangxi Mo, Eric Tang, Sumanth Hegde, Kourosh Hakhamaneshi, Shishir G Patil, Matei Zaharia, et al. 시연에서 쉽게 추론하는 것을 학습할 수 있으며, 만족은 중요하지 않습니다! arXiv preprint arXiv:2502.07374, 2025.
[30] Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, et al. o1과 유사한 llms의 과도한 생각에 대해 2+3=?에 대해 그렇게 많이 생각하지 않으십니까. arXiv preprint arXiv:2412.21187, 2024.
[31] Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Hanjie Chen, Xia Hu, et al. 과도한 생각을 멈추세요: 대형 언어 모델의 효율적인 추론에 대한 설문 조사. arXiv preprint arXiv:2503.16419, 2025.
[32] 사라 베라 마르야비안토(Sara Vera Marjanović), 아키일 패텔(Arkil Patel), 바이부 아드라카(Vaibhav Adlakha), 밀라드 아가조하리(Milad Aghajohari), 파리사드 베한마가더(Parishad BehnamGhader), 메하르 Bhatia(Mehar Bhatia), 아디티 쿼드널(Aditi Khandelwal), 오스틴 크로프트(Austin Kraft), 벤노 크로제(Benno Krojer), 싱한 루(Xing Han Lù), 기타. 심세크-러1 사고론(Deepseek-r1 thoughtology): llm 추론에 대해 ＜. arXiv preprint arXiv:2504.07128, 2025.
[33] Yuxiao Qu, Matthew YR Yang, Amrith Setlur, Lewis Tunstall, Edward Emanuel Beeching, Ruslan Salakhutdinov, Aviral Kumar. 메타 강화 미세 조정을 통해 테스트 시간 계산을 최적화. arXiv preprint arXiv:2503.07572, 2025.
[34] 마르테 발론(Marthe Ballon), 안드레스 알가파(Andres Algaba), 빈센트 기니스(Vincent Ginis). 대형 언어 모델(large language models)에서 추론과 성능 간의 관계-o3(mini)는 더 많이 생각하고, 더 이상은 생각하지 않습니다. arXiv preprint arXiv:2502.15631, 2025.
[35] Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, Gao Huang. 강화학습이 기본 모델을 넘어서는 llms에서 추론 능력을 촉진하는가? arXiv preprint arXiv: 2504.13837, 2025.
[36] Benjamin Estermann, Luca A. Lanzendörfer, Yannick Niedermayr, and Roger Wattenhofer. Puzzles: A benchmark for neural algorithmic reasoning, 2024.
[37] Karthik Valmeekam, Alberto Olmo Hernandez, Sarath Sreedharan, and Subbarao Kambhampati. 대형 언어 모델은 여전히 계획을 세울 수 없습니다. CoRR, abs/2206.10498, 2022.
[38] Karthik Valmeekam, Kaya Stechly, and Subbarao Kambhampati. llms still can’t plan; can lrms? a preliminary evaluation of openai’s o1 on planbench. 2024.
[39] 벤지 마, 싱xuan 헤, 찰리 셀, 테일러 그리스, 세와운 मिन, 마티 자카라야. 추론 모델은 생각 없이 효과적일 수 있습니다. arXiv preprint arXiv:2504.09858, 2025.
[40] 헌터 라이트먼, 비니트 코사라ju, 유라 버다, 해리 에드워드스, 보운 베이커, 테디 리, 존 레이테, 존 슐만, 이일리아 사트스키버, 칼 Cobbe. 단계별로 검증합시다. arXiv 사전 출력 arXiv: 2305.20050, 2023.
[41] Mathematical Association of America. American invitational mathematics examination

(aime). https://maa.org/math-competitions/ american-invitational-mathematics-examination-aime, 2025. accessed: 2025-05-15.
[42] Art of Problem Solving. Amc historical results - aime i (2024년 2월 1일). https://artofproblemsolving.com/wiki/index.php/AMC_historical_results#AIME_ I_.282024년 2월 1.2C_2024.29, accessed: 2025-05-15.
[43] 문제 해결(art of problem solving) Amc historical results – aime i (2025.02.06). https://artofproblemsolving.com/wiki/index.php/AMC_historical_results#AIME_ I_. 282025.02.6.2C_2025.29. accessed: 2025-05-15.
[44] 게리 F 마커스. 대수적 사고: 연결론과 인지 과학의 통합. MIT press, 2003.
[45] Saul Amarel. On representations of problems of reasoning about actions. In Readings in artificial intelligence, pages 2–22. Elsevier, 1981.
[46] Günter Rote. 밤에 다리를 건너는 것. EATCS Bulletin, 78:241, 2002.

부재

이 부근에서는 실험 설정 명세, 추가 결과, 확장 분석 등 주요 텍스트를 보완하는 세부 사항을 제공합니다.

A.1 퍼즐 환경 명세와 설계에 대한 세부 사항-문제 설명, 프롬프트 설계, 시뮬레이터를 포함한 네 가지 퍼즐 환경의 포괄적인 설명 A.1.1 하노이 탑 A.1.2 슈커 ジャン핑 A.1.3 강 건너 A.1.4 블록 월드
A.2 구현 세부 사항(implementation details)- 실험 설정 세부 사항, 모델 구성, 추출 파이프라인 세부 사항, 그리고 주어진 알고리즘 실행 실험.
A.3 계산 복잡성에 대한 세부 사항 A.3.1 구성 깊이 특성화 A.3.2 성능 대 구성 깊이
A.4 추가 결과와 분석-모든 모델과 퍼즐 환경에서 추론 노력 패턴을 포함하는 확장된 분석과 세부적인 실패 분석

A.1 퍼즐 환경 특성 및 설계에 대한 세부 사항

A.1.1 하노이의탑

문제 설명(problem description). 하노이 탑(Tower of Hanoi)은 추론 모델에서 순차적 추론 및 계획 능력을 평가하는 데 있어 뛰어난 문제로 작용하는 고전적인 재귀 퍼즐(recursive puzzle)입니다. 이 퍼즐은 세 개의 페그(left-right로 레이블이 0, 1, 2인)과 다양한 크기의 N 디스크로 구성되어 있으며, 각 디스크는 1(최소)에서 N(최대)로 독특하게 numérot됩니다. 초기 설정에서는 모든 N 디스크가 가장 왼쪽 페그(peg 0)에 축소되어 있으며, 가장 큰 디스크는 아래에 있으

며, 가장 작은 디스크는 위쪽에 있습니다. 나머지 두 개의 페그 (1과 2)는 처음엔 비어 있습니다. 목표는 모든 디스크를 peg 0에서 peg 2로 전이하여 동일한 크기 순서를 유지하는 것입니다. 이 퍼즐은 세 가지 근본적인 제약 조건으로 지배됩니다: (1) 단일 디스크 이동(single disk movement): 한 번에 단일 디스크만 이동할 수 있으며, (2) 상단 디스크 접근(top disk access): 어떤 페그에서 가장 위쪽 디스크만 이동할 수 있으며, (3) 크기 순서 제약 조건(size ordering constraint): 더 큰 디스크는 더 작은 디스크의 꼭대기에 절대 놓을 수 없습니다. 이 퍼즐은 모델의 추론 및 계획 능력을 평가하는 데 있어 모델이 문제를 하위 문제로 분할(recursive thinking), 여러 상태와 디스크 위치를 동시에 추적(working memory management), 운동 규칙과 제약 조건을 준수하면서 계획(제약 만족)을 수행하는(constraint satisfaction), 최종 목표를 달성하기 위한 올바른 작업 순서를 결정하는(sequential planning)과 같은 주요 인지 요구 사항을 보여줄 수 있도록 요구하기 때문입니다.

N 디스커로 하노이 탑 재귀 퍼즐을 해결하기 위해 필요한 최소 움직임 수는 $2^N - 1$로, 이는 지수적으로 확장되는 문제입니다. 이 속성은 문제 크기를 초기 디스커 수와 조정하여 세밀한 난이도 제어를 가능하게 합니다. 그러나 우리의 평가 프레임워크에서는 최적성보다는 해결책 정확도에 중점을 둡니다. 성공 기준으로 각 움직임의 유효성과 모델이 목표 상태에 도달할 수 있는 능력을 평가합니다.

프롬프트 설계(prompt design) 시스템 프롬프트(prompt)는 퍼즐 설정을 설명하는 명확한 문제 문장으로 시작합니다. 이는 모든 디스크를 세 번째 페그로 전이하는 운동 규칙과 목표를 명시적으로 명시합니다. 이해를 촉진하기 위해 프롬프트(prompt)는 예시 시연과 중요한 형식 및 추론 기대를 포함합니다.

시스템 프롬프트(System Prompt) - 하노이의 탑

여러분은 유용한 보조자입니다. 저를 위해 이 퍼즐을 해결해주세요. 첫 퍼그에 다양한 크기의 세 개의 페그와 n 디스크가 쌓여 있습니다. 디스크는 1(최소)에서 n(최대)로 셀 수 있습니다. 이 퍼즐에서 디스크가 이동하는 것은 다음과 같습니다:

1. 단지 한 번에 한 개의 디스크만 이동할 수 있습니다.
2. 각 움직임은 한 개의 스택에서 상위 디스크를 가져와 다른 스택의 위에 놓는 것으로 이루어집니다.
3. 더 큰 디스크는 더 작은 디스크 위에 놓지 않을 수 있습니다. 목표는 전체 스택을 세 번째 페그로 이동시키는 것입니다.
   예를 들어: 3개의 디스크가 1(최소), 2(최대), 3(최대)로 나열되어 있다면, 초기 상태는 [[3, 2, 1], [], []]이며, 해답은 다음과 같습니다:

$\Sigma =$ [[1, 0, 2], [2, 0, 1], [1, 2, 1], [3, 0, 2], [1, 1, 0], [2, 1, 2], [1, 0, 2]]

이는 다음과 같습니다. 디스크 1을 peg 0에서 peg 2로 이동하고, 디스크 2를 peg 0에서 peg 1로 이동하는 등입니다.

• 사고 과정에서 잠재적 솔루션을 탐구할 때, 대응하는 완전한 이동 목록을 항상 포함하세요.
• 위치는 0 지표로 표시되어 있습니다. 가장 왼쪽의 픽은 0입니다.
• 최종 답변에 움직임의 완전한 목록을 형식으로 포함하도록 하세요: moves $\sigma = \sigma$ [[disc id, from peg, to peg], ...]

시스템 프롬프트 후의 사용자 프롬프트는 특정 퍼즐 인스턴스를 제시하며, 현재의 설정은 페그 전반에 걸쳐 디스크의 분포를 보여주고, 목표 설정은 목표 상태를 지정합니다.

N 디스크(N Disks)의 사용자 prompt 템플릿 - 하노이 타워 (Tower of Hanoi)

초기 설정을 가진 다양한 크기의 $103 디스크와 퍼즐을 가지고 있습니다.

• 페그 0: $103(bottom), . . . 2, 1(top)
• 봉투1: (empty)
• 봉투 2: (empty)

목표 설정:

• 피그 0: (empty)
피그 1(empty)
• 픽 2: N(bottom), . . . 2, 1(top)

규칙:

• 한 번에 단 한 개의 디스크만 이동할 수 있습니다.
• 어떤 스택에서 위의 디스크만 이동할 수 있습니다.
• 더 큰 디스크는 더 작은 디스크 위에 놓지 않을 수 있습니다.

초기 구성을 목표 구성으로 변환하기 위해 이동의 순서를 찾으세요.

시뮬레이터. 우리의 평가 프레임워크는 각 퍼즐에 대해 개별 퍼즐 시뮬레이터(puzzle simulators)를 사용하여 LRM에서 얻은 솔루션의 엄격하고 일관된 평가를 보장합니다. 하노이 탑 시뮬레이터(Tower of Hanoi simulator)는 세 개의 페그(pegs) 사이의 디스크 구성을 추적하고 퍼즐의 기본 제약에 반하여 제안된 각 이동을 검증하는 정성 있는 환경으로 설계되었습니다. 시뮬레이터 아키텍처는 상태 관리(state management), 이동 검증(move validation), 솔루션

검증(solution verification) 사이의 명확한 분리를 가진 모듈형 설계 패턴을 따르며, 이 시뮬레이터에는 현재의 디스크 구성을 추적하고 퍼즐의 기본 제약을 강화하는 퍼즐 클래스(puzzle class)가 있습니다. 또한, 퍼즐 설정에서 각 이동을 실행하고 네 층 Validation(four-layer validation)을 수행하는 방법이 있습니다: 페그 경계 조건(peg boundary conditions) (0-2)을 확인하고, 소스 페그(source pegs)가 디스크를 포함하고, 특정 디스크가 가장 위대하다는 것을 확인하고, 더 큰 디스크가 더 작은 데 놓여지지 않도록 하는 크기 순서 제약을 강화합니다. 성공적인 Validation 후, 방법은 디스크 전이(disc transfer)를 실행하고 게임 상태(game state)를 업데이트합니다. 그 후, 완전한 솔루션 검증(solution validation)은 이동 목록(move lists)을 순차적으로 처리하고 목표 상태 달성(goal state)을 검증함으로써 처리됩니다.

A.1.2 슈커 ジャンピング

문제 설명. 쿼터 ジャン핑(Checker Jumping)은 순차적 추론, 계획 및 규칙 이해 능력을 테스트하기 위해 설계된 일차원 제약 만족 쿼터(constraint-satisfaction puzzle)입니다. 이 쿼터는 빨간 쿼터(red checkers, 'R'), 파란 쿼터(blue checkers, 'B')의 선형 배치와 단일 빈 공간 $(\underline{?_})$로 구성됩니다. 표준 설정에서는 N 빨간 쿼터(red checkers)가 왼쪽에 위치하고, 중간에 빈 공간이 있고, N 파란 쿼터(blue checkers)가 오른쪽에 있으며, 길이는 $2N+1$인 선형 보드를 형성합니다. 목표는 모든 빨간 쿼터(red checkers)와 파란 쿼터(blue checkers)의 위치를 교환하여 초기 설정을 효과적으로 반영하는 것으로, 빨간 쿼터(red checkers)는 오른쪽에, 파란 쿼터(blue checkers)는 왼쪽에 end up. 이 쿼터의 운동은 두 가지 기본 규칙에 의해 지배됩니다: (1) 슬라이드 운동(slide movement): 쿼터가 인접한 빈 공간으로 스लाइ드할 수 있으며, (2) 점프 운동(jump movement): 쿼터는 정확히 반대 색의 한 쿼터를 능가하여 빈 공간으로 착륙할 수 있습니다. 따라서 쿼터는 출발 측면으로 역으로 이동할 수 없습니다. 빨간 쿼터(red checkers)는 오직 오른쪽으로 이동할 수 있으며, 파란 쿼터(blue checkers)는 초기 설정에서 오직 왼쪽으로 이동할 수 있습니다. 이 쿼터는 인지적 도전을 나타내며, 추론 모델을 위한 대단한 테스트 기반으로 작용합니다. 예를 들어, 모델은 공간적 추론(쿼터의 위치와 가능한 이동을 추적하는 것, 제약 만족(motion rules during puzzle), 회색 계획(lookahead planning, လက်ရှိ 이동이 목표로의 미래 가능성에 어떻게 영향을 미치는지를 예측하는 것) 및 상태 공간 탐색(state-space exploration, possible move sequences through searching to find

셰커 ジャン핑 퍼즐(Checker Jumping puzzle)의 난이도는 셰커 수를 기준으로 합니다. 각 색의 N 셰커를 사용하면 최소 해결책은 $(N+1)^2-1$ 움직임이 필요하며, 이는 문제 크기와 해결책 복잡성의 사분적 관계를 생성합니다. 평가 프레임워크에서 우리는 주로 최적성보다는 해결책의 정확성에 초점을 맞추며, 각 움직임을 퍼즐 제약 조건에 대해 평가하고 최종 상태가 목표 구성과 일치함을 확인합니다. 이 접근 방식은 해결 과정 동안 발생할 수 있는 추론 실패와 제약 위반을 정확하게 식별할 수 있게 합니다.

프롬프트 설계(prompt design) 시스템 프롬프트는 퍼즐 설정과 이동 규칙을 설명하는 명확한 문제 문장으로 시작합니다. 이는 목표를 명시하고, 이동이 어떻게 표현되어야 하는지를 설명하기 위해 작은 보드 구성을 갖춘 구체적인 예시를 제공합니다.

시스템 프롬프트(System Prompt) - 점검자 ジャンピング(Checker Jumping)

"당신은 유용한 보조원으로, 저를 위해 이 퍼즐을 해주십시오."
1차원 보드에는 빨간 셰커('R'), 파란 셰커(blue checkers) $(^{7}\mathrm{B}')$, 그리고 하나의 빈 공간 $(\underline{?}_)$가 있습니다. 셰커는 다음과 같이 이동할 수 있습니다:

1. 주변 빈 공간으로 진전하는 것, 또는
2. 반대 색의 정확히 한 슈커를 넘어 빈 공간에 착륙합니다.

목표는 모든 빨간색과 파란색 쿼커의 위치를 교환하여 초기 상태를 효과적으로 반영하는 것입니다.

예: 초기 상태가 $[\mathrm{'R'}, ', ', \mathrm{'B'}]$라면, 목표는 ['B', '_', 'R']를 달성하는 것입니다. 여러분의 해결책은 각 이동이 [checker_color, position_from, position_to]로 표현되는 이동의 목록일 것입니다. 예를 들어:

이동 [['R', 0, 1], ['B', 2, 0], ['R', 1, 2]]

이는 이렇습니다. 빨간 점검자를 0 position에서 1 position으로 이동하고, 파란 점검자를 2 position에서 0 position으로 이동하는 등입니다.

요구 사항:

• 사고 과정에서 잠재적 솔루션을 탐구할 때, 대응하는 완전한 이동 목록을 항상 포함하세요.
• 위치는 0 지표로 표시되어 있습니다. 가장 왼쪽 위치는 0입니다.
• 최종 답변에 최종 해결책을 위한 이동의 완전한 목록을 형식으로 포함하도록 하세요: moves $\sigma = \sigma$ [[checker_color, position_from, position_to], ...]

사용자 프롬프트는 초기 보드 설정과 목표 상태를 가진 특정 퍼즐 인스턴스를 제시합니다.

$21.\#\#\#\#\#\#\#\#\#\#103$의 빨간 셰커 (\mathrm { ^ { 3 } R ^ { 3 } }), $\#\#\#\#\#\#\#\#103(\#\#\#\#$, 그 간의 빈 공간 (\' _ { - } ^ { \phantom { \dagger } })가 선으로 배치됩니다.

초기 보드: R R ... R _ B B ... B 목표 보드: B B ... B _ R R ... R 규칙:

• 점검자는 주변 빈 공간으로 슬라이드를 할 수 있습니다.
• 슈커(checker)는 반대 색의 정확히 한 슈커를 넘어 빈 공간에 착륙할 수 있습니다.
• 점검자는 역으로 이동할 수 없습니다.

초기 보드를 목표 보드로 변환하기 위해 최소한의 이동 순서를 찾으세요.

시뮬레이터. 우리의 평가 프레임워크는 셰커 ジャン핑 퍼즐 솔루션(Checker Jumping puzzle solutions)을 검증하기 위해 맞춤형 시뮬레이터(custom simulator)를 사용합니다. 시뮬레이터는 모든 퍼즐 제약 조건을 부과하면서 해결책 경로 전반에 걸쳐 상태의 진화를 추적하는 포괄적인 검증 시스템을 구현합니다. 셰커 ジャン핑 시뮬레이터(Checker Jumping simulator)는 모든 셰커의 위치와 빈 공간을 추적하는 상태로운 환경으로 설계되어, 주어진 해결책의 동작 규칙에 대해 각 동작을 검증합니다. 시뮬레이터는 초기 상태와 목표 상태가 모두 잘 형성되어 동일한 수의 붉은 색과 파란 셰커와 정확히 한개의 빈 공간을 포함하고 있음을 검증하는 것으로 시작합니다. 그 후 각 동작은 위치 경계를 검증하고, 소스에서 올바른 셰커 색을 확인하며, 목표 위치가 빈임을 보장하며, 이동 유형을 슬라이드(거리=1) 또는 점프(거리=2)로 검증합니다. 시뮬레이터는 역방향 제약을 부과하여 역전(red checkers move right, blue checkers move left)의 움직임을 방지하며, 중간 위치에 대조 색의 셰커의 존재를 확인하여 점프 동작을 검증합니다. 성공한 검증 후, 방법은 위치를 업데이트하고 소스를 제거하여 셰커 전이(Checker transfer)를 실행합니다. 그 후, 완전한 이동 시퀀스가 최종 목표 상태 검증을 통해 처리됩니다.

A.1.3 강 건너

문제 설명. 강 건전(River Crossing)은 다중 에이전트 koordination와 제약 관리를 테스트하는 제약 만족 계획 퍼즐입니다. 이 퍼즐은 계획 문헌에서 광범위하게 연구된 미션리 및 cannibals 문제와 브리지 및 토치 문제와 같은 고전적인 문제의 일반화입니다 [45, 46]. 강 건전 퍼즐은 N 배우(denoted by $a_1, a_2, ..., a_N$)와 해당 N 에이전트(denoted by A_1 , A_2 , ..., A_N)를 포함합니다. 초기 상태에서는 2 N

bank######.#####bank#########.#################################(1)######## : ##########

k›k(N \leq 3)#####2, #####(N \leq 5$)에 대해 3로 설정됩니다. (2) 비 Empty Boat 제약: 배는 비활성으로 이동할 수 없고 최소한 한 명의 사람이 배에 들어갈 수 있어야 합니다. (3) 안전 제약: 배우는 경쟁하는 에이전트로부터 고객을 보호해야만 하는 경우에 다른 에이전트의 주석에 있을 수 없습니다. 이 안전 제약은 은행과 배 모두에 적용됩니다. 이 퍼즐은 복잡한 계획 및 상태 추적을 필요로 하며, 참가자들은 안전 제약을 항상 유지하면서 건전을 조심스럽게 조정해야 합니다. 해결자는 안전하게 함께 이동할 수 있는 다양한 조합을 통해 추론해야 하며, 건전 후 배로 누가 다시 오를지를 결정하고, 어떤 제약을 위반하지 않고 모두를 오른쪽 bank으로 데려 올 수 있는 시퀀스를 전략적으로 계획해야 합니다. 이 작업의 복잡성은 배우-에이전트 쌍과 배 용량의 수를 조정하여 모델을 추론할 수 있는 확장 가능한 도전 과제를 생성합니다.

프롬프트 설계(Prompt Design) 시스템 프롬프트는 배우와 에이전트를 나타내는 표식을 도입하고, 배 이동의 목록으로 솔루션 형식을 설정하며, 형식을 입증하기 위한 간단한 예제를 제공합니다.

시스템 프롬프트(System Prompt) - 강 건너(River Crossing)

"당신은 유용한 보조원으로, 저를 위해 이 퍼즐을 해주십시오."
액터(actor)를 a_1, a_2, ..., 에이전트(agent)를 A_1, A_2, ...로 표현할 수 있습니다. 해결책은 각 이동이 배에 있는 사람들을 나타내는 배 이동 목록이어야 합니다. 예를 들어, 두 액터(actor)와 두 에이전트(agent)가 있다면, 이러면:
$\tau = \tau$ [[" A_2 ", "a_2 "], [" A_2 "], [" A_1", "A_2 "], [" A_1 "], [" A_1 ", "a_1 "]]
이는 첫 번째 움직임에서 A_2와 a_2가 왼쪽에서 오른쪽으로, 두 번째 움직임에서는 A_2가 오른쪽에서 왼쪽으로 이동하는 것을 나타냅니다.

요구 사항:

• 사고 과정에서 잠재적 솔루션을 탐구할 때, 항상 해당된 보트 이동의 완전한 목록을 포함하세요.
• 명단에는 의견이 없애져야 합니다.
• 최종 답변에 최종 해결책을 위한 이동의 완전한 목록을 포함하도록 보장하세요.

사용자 프롬프트(user prompt)는 특정 퍼즐 인스턴스를 나타내며, N 배우-에이전트 쌍, 보트 용량 k , 그리고 솔루션 전반에 걸쳐 유지되어야 할 안전 제약을 제시합니다.

N 쌍을 위한 사용자 프롬프트 템플릿 - 리버크로싱

$103 배우와 그들의 $103 에이전트는 한 번에 $123의 사람들만을 수용할 수 있는 배로 강을 건너고 싶어하지만, 각 에이전트가 상대방이 고객을 밀렵할 것인지 우려하기 때문에 다른 에이전트가 존재할 수 없는 경우에도 배를 타고 있을 수 없습니다. 처음에는 모든 배우와 에이전트가 배를 타고 강의 왼쪽에 있습니다. 어떻게 강을 건너야 할까요? (모두: 배는 비어있게 이동할 수 없습니다)

시뮬레이터. 우리의 평가 프레임워크는 리버 크로싱 퍼즐 추출 솔루션을 검증하기 위해 맞춤형 시뮬레이터(custom simulator)를 사용합니다. 시뮬레이터는 모든 개인(작가와 에이전트)의 상태와 배 위치를 추적하면서 모든 퍼즐 제약 조건을 강화합니다. 각 이동은 다중 단계 검증(multi-step validation)을 통해 수행됩니다: 배 용량 제한을 확인하고, 모든 승객이 배의 현재 측면에 있는지 확인하며, 이동 후 배와 각 은행에 있는 다른 에이전트의 presence cannot be in the presence of other agents without their own agent present, both on the boat and on each bank. 시

뮬레이터는 동적 배 위치를 관리하며, 각 크로싱 후 각 측면을 자동으로 전환하고, 각 이동 후 완전한 상태를 검증하여 양 은행에서 안전 위반이 발생하지 않도록 합니다. 그 후, $2N$의 모든 개인이 옳은 은행에 성공적으로 도달하는 완전한 크로싱 시퀀스를 검증합니다.

A.1.4 블록 월드

문제 설명. 블록 월드(Blocks World)는 LLMs의 계획 능력을 분석하기 위해 최근 연구된 고전적인 계획 퍼즐(planning puzzle)입니다 [37, 38]. 이 퍼즐은 초기 구성에서 특정 목표 구성으로 재배치해야 하는 여러 블록(A, B, C, 기타)의 스택을 포함합니다. 각 블록은 문자로 독특하게 식별되며, 목표 상태로 변환하기 위해 초기 상태를 위한 최소한의 이동 순서를 찾는 것이 목표입니다. 이 퍼즐은 두 가지 기본 제약 조건 하에서만 작동합니다: (1) 상단 블록 이동(Top Block Movement): 어떤 스택에서 가장 위의 블록만이 이동할 수 있으며, (2) 유효한 배치(Valid Placement): 하나의 블록은 빈 위치나 다른 블록의 위에만 배치될 수 있습니다. 이러한 제약 조건은 작업 순서가 중요한 계획 문제를 초래하며, 일부 구성에서는 아래의 블록을 잠재적으로 배치하여 나중에 접근할 수 있을 수 있습니다. 블록 월드(Blocks World)는 추론 모델에서 계획 능력을 평가하는 데 있어 좋은 테스트 기반으로 작용하며, 이는 전방적 사고와 상태 추적(state tracking)을 필요로 합니다. 최근 연구에서는 이 퍼즐을 단순화된 설정(3개에서 5개까지의 블록)을 포함하여 순차적 계획 작업에서 LLM 성능을 평가하는 다양한 구성으로 검토했습니다 [37, 38]. 모델은 복잡한 상태 변환을 유효한 순차적 이동으로 분해할 수 있는 능력을 입증해야 하고, 블록 간의 종속성을 추론해야 하며, 예를 들어, 복잡한 상태 변환을 접근하기 전에 저층 블록을 제거해야 하며, 비법적인 이동 없이 목표 상태로의 경로를 효율적으로 계획해야 합니다.

이 퍼즐의 난이도(difficulty)는 여러 매개변수를 조정하여 확장될 수 있습니다. 이는 블록의 수가, 스택의 수가, 초기 및 목표 구성의 복잡성 등입니다. 우리는 주로 블록 셀 N를 통해 복잡성을 제어하며 초기 및 목표 구성에서 명확한 구조적 패턴을 따르습니다. 우리의 실험 설계에서는 초기 구성이 두 스택 간의 N 블록을 알파벳 순서로 일관되게 분할하며, 세 번째 스택은 작업 공간으로 빈 상태로 구성됩니다. 목표 구성은 모든 블록을 최초 스택에 통합하여 두 초기 스택의 블록 간의 변화를 일관되게 수행하며, 기존 스택의 완전한 분해 및 재조립이 필요한 특정 위치를 요구합니다. 예를 들어, $N = 4$の場合 초기 상태는 두 스택 [["A", "B"], ["C", "D"], []] 사이에 블록을 나누고, 목표 상태 [["D", "B", "C", "A"], [], []]에는 두 스택의 간의 교차 블록이 필요하며, $N = 6$の場合 초기 상태 [["A", "B", "C"], ["D", "E", "F"], []]는 [["F", "C", "E", "B", "D", "A"], [], []]로 변환되어 복잡한 교차 패턴을 형성합니다. N 증가함에 따라 상태 공간은 점차적으로 증가하며, 최소 해결 길이는 N와 함께 약 선형적으로 증가합니다. N의 작은 값에 (2-7) 경우, 퍼즐은 기본 계획을 테스트하며, 중간 값(8-20)에는 더 긴 계획 지평을 가진 더 복잡한 추론을 요구하며, 큰 값($N > 20$)에는 긴 해결 경로에서 광범위한 임시 이동과 패턴 인식을 요구하여 순차적 추론 능력의 한계를 도전합니다.

프롬프트 설계(Prompt Design) 시스템 프롬프트는 블록 월드 퍼즐의 기본 규칙을 소개하고, 이동 표현 형식을 설정하며, 해결 구조를 입증하기 위한 간단한 예제를 제공합니다.

"당신은 유용한 보조원으로, 저를 위해 이 퍼즐을 해주십시오."
이 퍼즐에서는 블록의 스택이 존재하며, 목표는 이를 이동의 순서를 사용하여 목표 설정으로 재배열하는 것입니다. 여기서는 다음과 같이 할 수 있습니다: • 어떤 스택에서 가장 위의 블록만이 이동할 수 있습니다.
• 블록은 빈 위치에 두거나 다른 블록의 위에 두거나 할 수 있습니다.

예: 초기 상태 [["A", "B"], ["C"], []] 및 목표 상태 [["A"], ["B"], ["C"]]에서 해결책은 다음과 같습니다:

moves $\sigma = \sigma$ [["C", 1, 2], ["B", 0, 1]]

이는 이렇습니다. 블록 C를 1 개의 스터크에서 2 개의 스터크로 이동하고, 블록 B를 0 개의 스터크에서 1 개의 스터크로 이동하는 것을 의미합니다.

• 사고 과정에서 잠재적 솔루션을 탐구할 때, 대응하는 완전한 이동 목록을 항상 포함하세요.
• 최종 답변에 최종 해결책을 위한 이동의 완전한 목록을 형식으로 포함하도록 하세요: moves $\sigma = \sigma$ [[blok, from stack, to stack], ...]

사용자 프롬프트(user prompt)는 초기 설정과 목표 설정을 제공하여 특정 퍼즐 인스턴스를 제시하며, 동작 제약에 대해 모델에 명시적으로 상기합니다.

N 블록을 위한 사용자 prompt 템플릿 - BlocksWorld

$103 블록으로 구성된 퍼즐을 가지고 있습니다. 초기 상태:

확률 0: $blocks_0$ (상단) 확률 1: $blocks_1$ (상단) 확률 m: blocks_ \mathrm { m } \mathfrak { F }$ (상단)

목표 상태:

확률 0: $goal_blocks_0$ (상단) 확률 1: $goal_blocks_1$ (상단) 확률 m: $goal_blocks_m 9$ (상단)

초기 상태를 목표 상태로 변환하기 위해 최소한의 이동 순서를 찾으세요. 각 스터크의 최상위 블록만이 이동할 수 있음을 기억하십시오.

시뮬레이터. 우리의 평가 프레임워크는 블록 월드 퍼즐 추출 솔루션(blocks world puzzle extracted solutions)을 검증하기 위해 맞춤형 시뮬레이터(custom simulator)를 사용합니다. 시뮬

레이터는 스태크 전반에 걸쳐 모든 블록의 상태를 관리하면서 퍼즐의 이동 제약을 강화합니다. 각 이동은 세 층의 검증을 통해 퍼즐 설정에서 실행됩니다: 스태크 지수가 경계 내에 있는지 확인하고, 소스 스태크가 블록을 포함하고, 명시된 블록이 스태크의 꼭대기에 있는지를 보장합니다(상위 블록만 이동 규칙을 강화합니다). 성공적인 검증 후, 블록 전이가 수행되며, 블록은 소스 스태크에서 포핑되어 목적지 스태크에 부착됩니다. 마지막으로, 블록 이동의 완전한 솔루션 시퀀스가 처리되어 결과적인 설정이 목표 목표 상태와 일치하는지를 검증합니다.

A.2 구현 세부 사항

구성 Our experiments primarily utilized reasoning models and their non-thinking counterparts to enable thorough analysis of the thinking process. We specifically selected Claude 3.7 Sonnet (thinking/non-thinking) and DeepSeek-R1/V3 due to their ability to provide access to thinking traces, a critical requirement for our analysis. For experiments focused solely on final accuracy metrics, we also included results from OpenAI's o3-mini models, as they lack access to thoughts. For Claude 3.7 Sonnet (thinking and non-thinking) models we used maximum generation budget of 64,000 tokens, accessed through the API interface. Temperature is set to 1.0 for all API rus (Claude-3.7-Sonnet and o3-mini runs). DeepSeek-R1, DeepSeek-V3, and DeepSeek-R1-Distill-Qern-32B 간의 실험은 지역 서버에서 수행되며, 최대 생성 길이는 64,000이고 온도는 1.0입니다. 모든 실험에서, 복잡성 수준(N 값)에 따라 퍼즐 인스턴스에 25개의 샘플을 생성하고, 모든 샘플에서 성능 평균을 보고했습니다.

해결책 추출 모델 응답(model responses)과 중간 추론 추적(thoughts)을 처리하기 위해 맞춤형 추출 파이프라인(custom extraction pipeline)이 개발되었습니다. 이 파이프라인은 여러 주요 구성 요소로 구성됩니다. 우리는 유연한 레ᄀᆞᆷ스 기반 추출기를 구현하여 최종 응답과 사고 추적 모두에서 잠재적인 해결책 시도를 식별했습니다. 추출 과정에서 정규 표현을 사용하여 해결책 패턴을 식별합니다. 특히, 사고 추적 내의 정확한 위치 추적을 위해, 우리는 모든 실험에서 토큰을 세는 모델과 동일한 토큰izer(cl100k_base)를 사용했습니다. 토큰 위치는 사고 길이에 대해 정규화되어 교차샘플 비교를 가능하게 했습니다. 마침내, 우리는 사고 추적 내에서 녹음된 해결책이 유일하고 복제 해결책(identical moves list)이 필터화되어 있음을 보장합니다. 복제 해결책의 경우, 분석을 위해 첫 번째 해결책만이 녹음됩니다.

해결책 평가 추출 후, 각 해결책 후보자는 조밀한 검증을 위해 퍼즐의 해당 시뮬레이터에 전달됩니다. 시뮬레이터는 동작의 목록으로 해결책을 취하고 퍼즐에 대한 평가를 수행합니다(각 퍼즐 시뮬레이터의 세부 사항을 확인하기 위해 A.1 앱을 점검하세요). 구성적 해결책의 각 동작은 이전의 동작과 퍼즐 규칙에 따라 순차적으로 실행됩니다. 그 후, 시퀀스 내 모든 동작에서 얻은 최종 상태는 퍼즐의 목표 상태와 비교되어 완전한 해결책 정확성을 결정합니다. 잘못된 해결책 경우, 퍼즐 시뮬레이터를 사용하여 이동 검증 중에 최초의 실패 동작과 실패 유형의 세부 사항도 수집됩니다.

처방된 단계의 실행 다양한 퍼즐에서 개방형 문제 해결(open-ended problem solving)을 수행할 뿐만 아니라, 명시적인 해결 알고리즘 지침을 처방된 단계로 제공함으로써 이러한 추론 모델의 행동이 어떻게 영향을 미칠지를 테스트하기 위한 집중 실험도 수행했습니다(Sec. 4.4).

우리는 처음부터 해결책을 찾는 것과 설계하는 것이 주어진 알고리즘의 단계를 따르는 것보다 모델에 대한 계산이 훨씬 더 많이 필요하다고 기대했습니다. 그러나 그래프 8a와 8b의 결과는 추론 모델의 행동이 크게 변하지 않고, 이 설정에서 이전과 대략 동일한 시점에서 붕괴가 여전히 발생한다는 것을 보여줍니다. 이러한 발견은 문제 해결 및 해결 전략 발견뿐만 아니라 생성된 추론 사슬 전반에 걸쳐 일관된 논리적 검증 및 단계 실행 제한에 대한 증거를 강화합니다.

예를 들어, 모델은 하노이 탑 퍼즐을 해결하는 완전한 재귀 알고리즘을 제공받고 있습니다. 이 알고리즘은 표준 문제 프롬프트에 스크래치패드를 추가하여 추론 행동에 미치는 영향을 테스트했습니다.

해노이 타워(Tower of Hanoi)에 대한 규정 알고리즘의 예

여기에 퍼즐을 해결하기 위한 재귀 알고리즘의 pseudocode가 있습니다:

ALGORITHM 해결(n, 소스, 목표, 보조, 이동) // n $\tau = \tau$ 이동할 디스크 수 // 소스 $\tau = \tau$ 시작 페그 (0, 1, or 2) // 목표 $\sigma = \sigma$ 목적지 페그 (0, 1, or 2) // 보조 $\tau = \tau$ 사용되지 않은 페그 (0, 1, or 2) // 이동 $\tau = \tau$ 이동 순서를 저장하기 위한 목록

n가 1을 곱하면 // 소스 페그 디스플레이에서 상위 디스플레이를 얻습니다 $\sigma = \sigma$ 소스 페그의 상위 디스플레이를 얻습니다 // 우리의 목록에 이동을 추가합니다: [disk_id, source, target] [디스플레이, 소스, 목표]를 추가하면 이동이 재현됩니다

끝났을 때

// 소스에서 보조 페그로 n-1 디스크를 이동시킵니다. $(\mathtt{n}-\mathtt{1}$, 소스, 보조, 목표, 이동)를 해결합니다.

// 네 번째 디스크를 소스에서 목표 디스크로 이동시킵니다 $\tau = \tau$ 소스 페그의 상단 디스크를 ADD [디스크, 소스, 목표]로 이동시킵니다

// 보조기에서 목표로 n-1 디스크를 이동시킵니다. 해결(n-1, 보조기, 목표, 소스, 이동) END ALGORITHM

peg 0에서 peg 2로 n 디스크를 이동시키는 전체 퍼즐을 해결하기 위해

1. 빈 목록을 시작하고, 이동합니다.
2. 해결(n, 0, 2, 1, 이동)을 실행

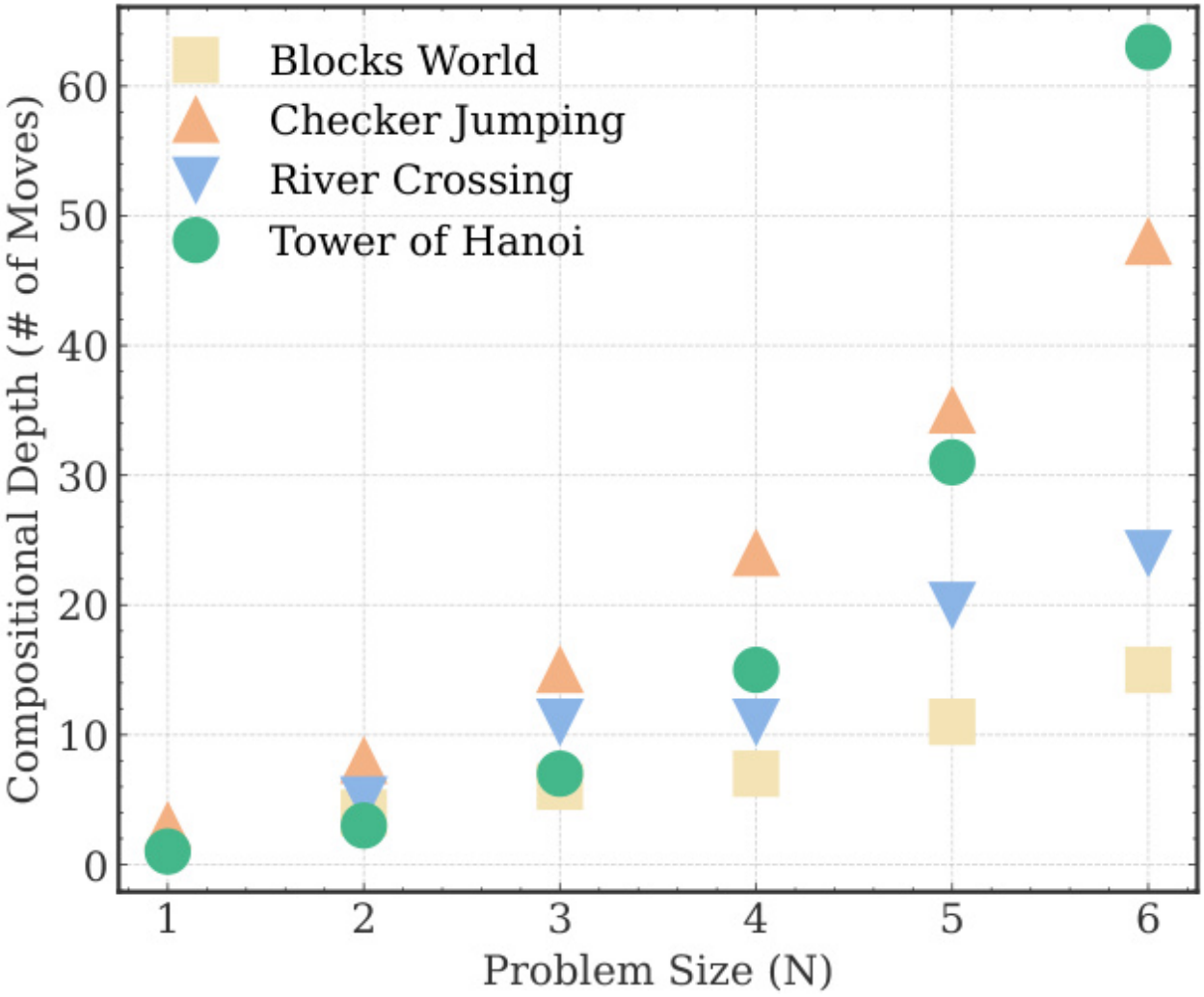
3. ’모브스’ 목록은 완전한 솔루션을 포함할 것입니다.

주의: 이 pseudocode를 실행할 때, 각 페그의 위에 있는 디스크를 추적하세요. 이동 목록의 디스크 ID는 실제 이동하는 디스크와 일치해야 합니다. 이 알고리즘을 스크래치패드로 사용할 수 있으며, 문제를 단계별로 해결할 수 있도록 도와줍니다.

A.3 계산 복잡성에 대한 세부 사항

A.3.1 구성적 깊이 특성화

구성 깊이(compositional depth)는 완전한 솔루션을 달성하기 위해 필요한 순차적 연산(sequential operations, moves)의 수를 나타냅니다. 9차 그래프는 이 깊이가 네 가지 플레이즈 환경에서 문제 크기(N)와 함께 어떻게 확장되는지를 보여줍니다. 각 플레이즈는 기하급수적 성장 패턴을 가지고 있으며, 이는 그 기저의 계산 복잡성을 반영합니다. 예를 들어, 하노이의 탑(Tower of Hanoi)은 기하급수적 성장을 $(2^N - 1)$로, 셰커 ジャン핑(Checker Jumping)은 사분적 확장($(N + 1)^2 - 1$)로 나타냅니다. 강 건너(River Crossing) 및 블록즈 월드(Blocks World) 플레이즈는 N로 더 균형 잡고, 거의 선형적인 성장을 나타냅니다. 이러한 다양한 구성 깊이 프로파일(compositional depth profiles)은 언어 추론 모델이 다양한 유형의 순차적 추론 문제를 처리하는 방식과 플레이즈를 해결하기 위해 필요한 구성 깊이(compositional depth)와 항상 그 정확도가 상관관계가 있는지를 평가할 수 있게 합니다. 이 분석에 대한 더 많은 세부 사항은 App. A.4.의 10차 그래프에서 제공합니다.



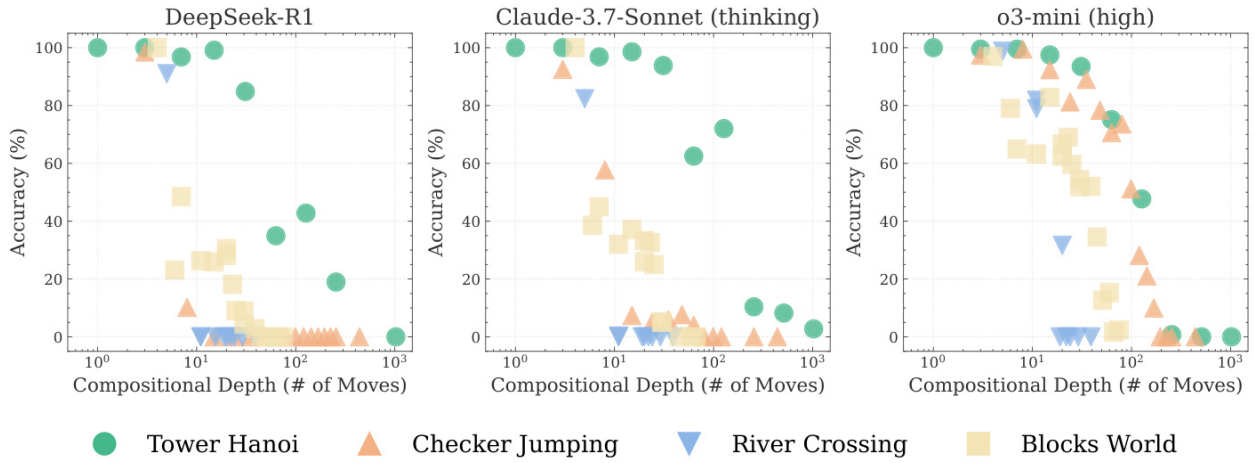
그래프 9: 네 개의 퍼즐 환경에 대해 다양한 문제 크기에서 구성 깊이(이해가 필요한 이동 수)를 나타냅니다.

A.3.2 성능 대 구성적 깊이

직관은 문제 복잡성과 모델 정확도 간의 부정적인 상관관계를 시사하지만, 우리의 분석은 구성 깊이와 LRM 성능 간의 보다 미묘한 관계를 드러냅니다. 그래프 10는 우리의 퍼즐 세트에서 이를 세 가지 최첨단 추론 모델(Claude-3.7-Sonnet w. thinking, DeepSeek-R1, and o3-mini)에서 입증합니다. 개별 퍼즐 유형 내에서, 우리는 구성 깊이가 증가함에 따라 모델 정확도가 일관되게 감소하는 기대 부정적 상관관계를 관찰합니다. 그러나, 다양한 퍼즐 유형에서 이러한 관계는 붕괴됩니다. 모델은 낮은 구성 깊이의 퍼즐을 겪을 수 있지만, 높은 구성 깊이의 다른 퍼즐에서 성공할 수 있습니다. 예를 들어, 모델은 tower of Hanoi 예제에서 $>50\%$의 정확도를 달성하는데, 이는 약 10^2 moves를 요구하지만, substantially lower compositional depth의 리버 크로싱 퍼즐에서 일관되게 실패합니다 ($\sim 10^1$ moves).

A.4 확장된 결과와 분석

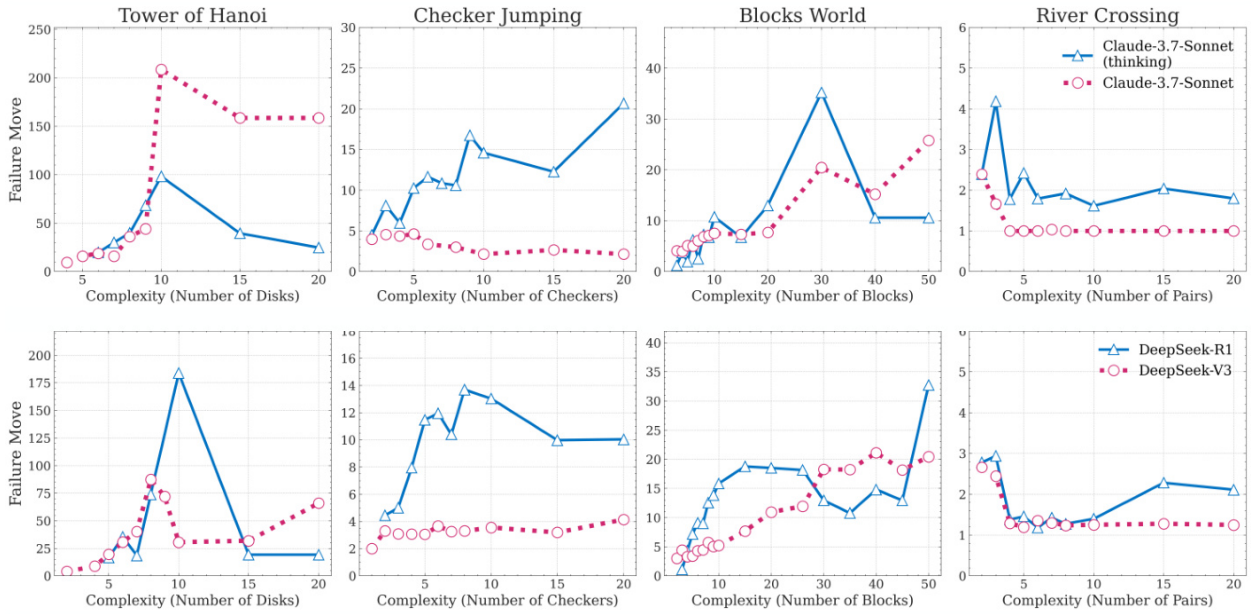
실패 분석(failure analysis) 구성 추론 단계 내에서 모델이 어디에서 실패하는지를 이해하는 것은 이진 성공 지표(binary success metrics)를 넘어서는 통찰력을 제공합니다. 정확성 평가는 전체 이동 시퀀스(move sequences)를 완벽하게 수행해야 하며, 단일 부정확한 이동은 실패를 초래합니다. 실패 패턴을 보다 세밀하게 조사하기 위해, 다양한 문제 복잡도 수준에서 모델이 처음 부정확한 동작을 할 때의 구성 깊이를 분석합니다.



그래프 10: 네 개의 퍼즐 환경에서 세 가지 LRM(DeepSeek-R1, Claude-3.7-Sonnet with thinking, and o3-mini)에 대한 정확도와 구성 깊이(작용 수 required)에 대한 확률

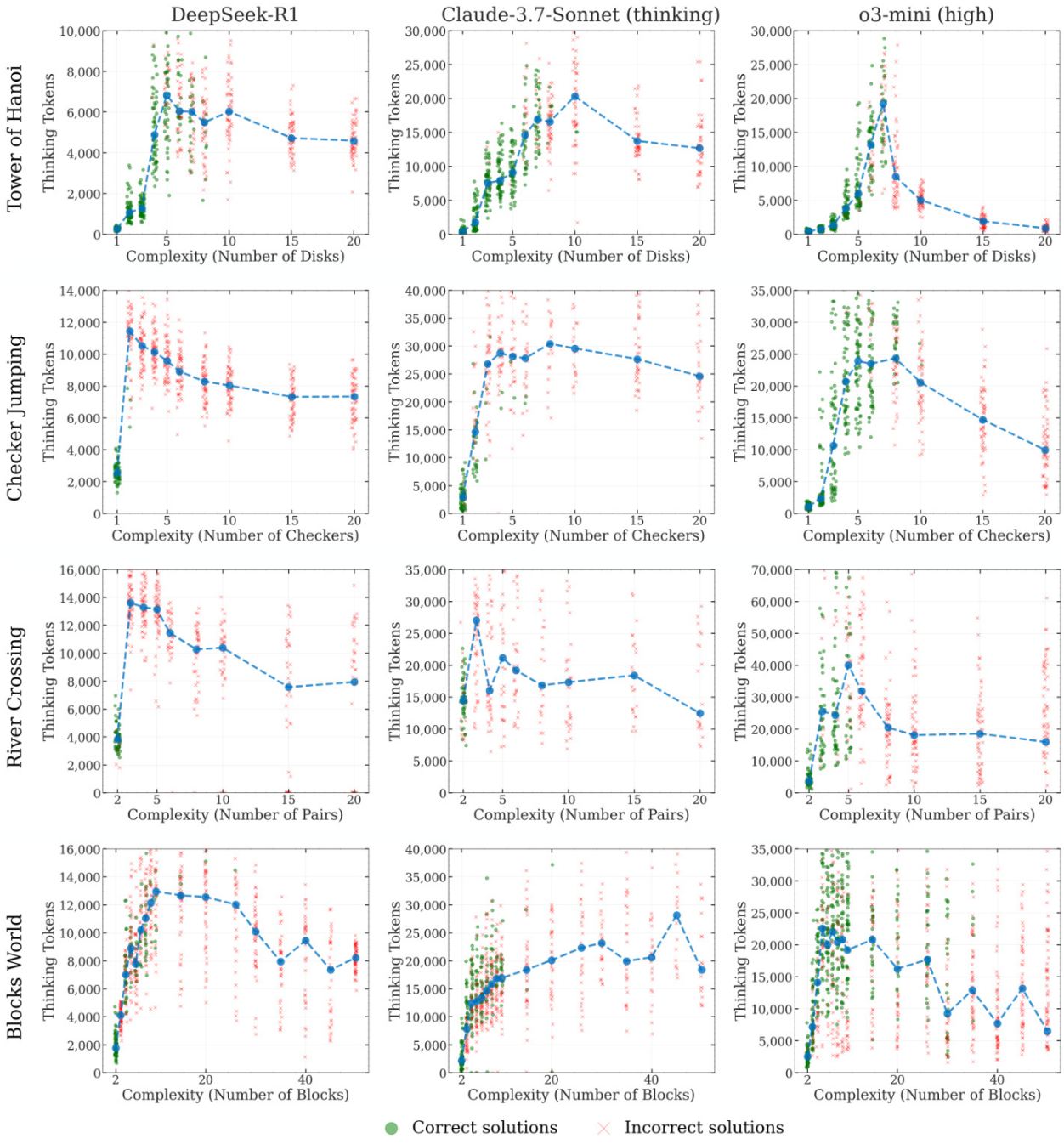
그래프 11는 해결 시퀀스 내에서 실패 이동 ID(failure move ID)와 문제 복잡성(N)을 나타냅니다. 상단 row는 Claude-3.7-Sonnet(Claude-3.7-Sonnet)을 사고 능력과 무언가와 비교하며, 하단 row는 DeepSeek-R1(생각)과 DeepSeek-V3(생각하지 않는)를 비교합니다. 이러한 비교는 LRM(large-resourced retrieval mechanisms, LRMs)의 사고 메커니즘이 퍼즐의 구성적 추론 작업에서 실패 패턴을 어떻게 영향을 미치는지를 보여줍니다. 우리의 분석에서 몇 가지 반관론적 패턴이 나타나고 있습니다. 먼저, 모델은 문제 복잡성(problem complexity)에 대한 비단조적 실패 행동을 보입니다. 이는 모델이 더 높은 N 값을 위해 해결 시퀀스에서 더 일찍 실패하는 경우에도 더 긴 전체 해결책이 필요합니다. 예를 들어, 하노이 타워(Tower of Hanoi)에서는 모델이 $N = 15$ 에 대해 50도 이하의 이동으로 실패하지만 $N = 8$ 에 대해 100개 이상의 이동으로 성공하는 경우가 있습니다. 이는 동일한 가족 내에서 사고 모델과 비생각 모델을 비교하여, 각 퍼즐 환경에 대한 문제 복잡성에 대해 집계된 실패 이동 위치의 밀도 분포를 보여줍니다. 이 그래프에 기반하여, 사고 모델(Claude-3.7-Sonnet(Claude-3.7-Sonnet with thinking and DeepSeek-)은 다양한 문제 스케일에서 학습 해결 전략을 적용하는 모델의 근본적인 불일치를 시사합니다. 또한, 두 모델 변형이 완전한 정확도 붕괴를 경험하는 고 복잡성 상태에서, 예를 들어, 하노이 타워(Tower of Hanoi)는 $N \geq 15$ 를, 블록즈 월드(Blocks World)는 $N \geq 40$ 를 보았으며, 비생각 모델은 때때로 해결 시퀀스에 더 깊이 있는 성능을 유지하고 생각 가능 변형보다 후의 이동에서 실패할 수 있습니다. 이는 LLM(large-resourced retrieval mechanisms, LLMs)의 구성적 추론 실패가 단순히 충분한

추론 노력 동역학(reasoning effort dynamics) Figure 13는 퍼즐 환경에서 추론적 사고 토큰(inference thinking tokens) 대 문제 복잡성(problem complexity)의 추론 노력(meaasured by inference thinking tokens)을 나타냅니다. 녹색 점들은 올바른 솔루션을 나타내고, 빨간 십자가는 올바른 솔루션을 나타내며, 파란 선은 다양한 퍼즐과 LRM에서 복잡성 수준(N)마다 평균적인 사고 토큰 사용을 추적합니다. 우리는 세 가지 추론 모델(DeepSeek-R1, Claude-3.7-Sonnet-thinking, o3-mini)에서 사고 토큰 사용(thought token usage), 즉 추론 노력(reasoning effort)이 문제 복잡성(problem complexity)과 함께 처음에 확장되다가 모델별 한계(model-specific threshold)에 도달한 후 반감각적으로 감소하는 일관된 패턴을 관찰합니다. 이는 특정 복잡성 한계(complexity thresholds)를 넘어 모델이 문제를 해결하지 못할 때 LRM 사고 과정에서 흥미롭고 근본적인 확장 제한을 시사합니다. 이는 더 어려운 문제에 직면하고 문맥 및 생성 한계(context and generation limits)를 상당히 능가하더라도 반감각적으로 추론 계산을 줄입니다.



그래프 11: 퍼즐 환경에서 사고 및 비 사고 모델에 대한 문제 복잡성(N) 대 최초 실패 이동 비교. 상단: 클로드-3.7-소네트 비교; 하단: 딥시크-R1 vs 딥시크-V3

하노이 슈커 점프 블록스월드 리버크로싱
0.01 50.0 100.0 150.0 200.0 0.0.0 10.0 15.0 20.0 10.0 15.0 20.0 0.0.c 5.0 7.5 10.0 실패 이동 실패 이동 실패 이동 실패 이동 0.35심밀 탐색-R1: 9.9
0.010 N 50.0 100.0 150.0 200.0.09.0 5.0 10.0 15.0 20.0 10.0 5.0 7.5 10.0 실패 이동 실패 이동 실패 이동 실패



그래프 13: 네 가지 퍼즐 환경에서 세 가지 LRM(DeepSeek-R1, Claude-3.7-Sonnet with thinking, and o3-mini)에 대한 추론 노력(추론적 사고 토큰으로 측정된)과 문제 복잡성(N)에 대한 세부 결과입니다.