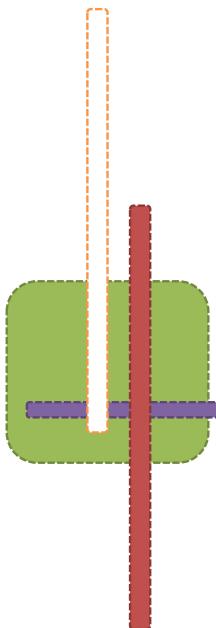


# 거대언어모델의 논리 추론 능력 향상

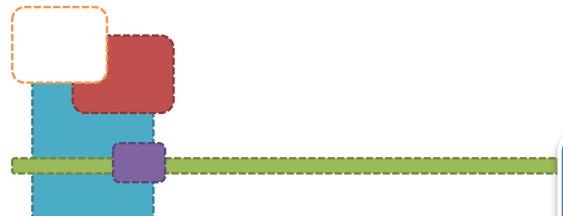
## 프롬프트 엔지니어링 중심의 접근



이웅희 공학박사

공학기술연구소

한양대학교 ERICA  
Education Research Industry Cluster @ Ansan



# 거대언어모델의 추론 능력

## Prompt

많은 연구에서 IO 또는 vanilla 방법으로 불림

- The odd numbers in this group add up to an even number: 15, 32, 5, 13, 82, 7, 1.
- A:

## Output

- In the given group of numbers: 15, 32, 5, 13, 82, 7, 1, let's identify the odd numbers first: 15, 5, 13, 7, 1.
- Now, let's add these odd numbers together:
- $15 + 5 + 13 + 7 + 1 = 41$
- So, the sum of the odd numbers in the group is 41, which is indeed an odd number, not an even number.

## Few-shot prompting

- 프롬프트로 맥락(예시)을 추가로 주는 방식으로 성능 향상

## 예시: 문장 분류 태스크

예제를 줌 (1-shot)

### Prompt

- Classify the text into neutral, negative or positive.
- Text: I think the vacation is okay.
- Sentiment: neutral

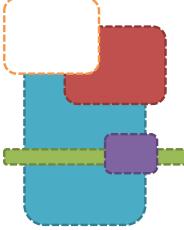
- Text: I think the food was okay.
- Sentiment:

### Output

- Sentiment: neutral

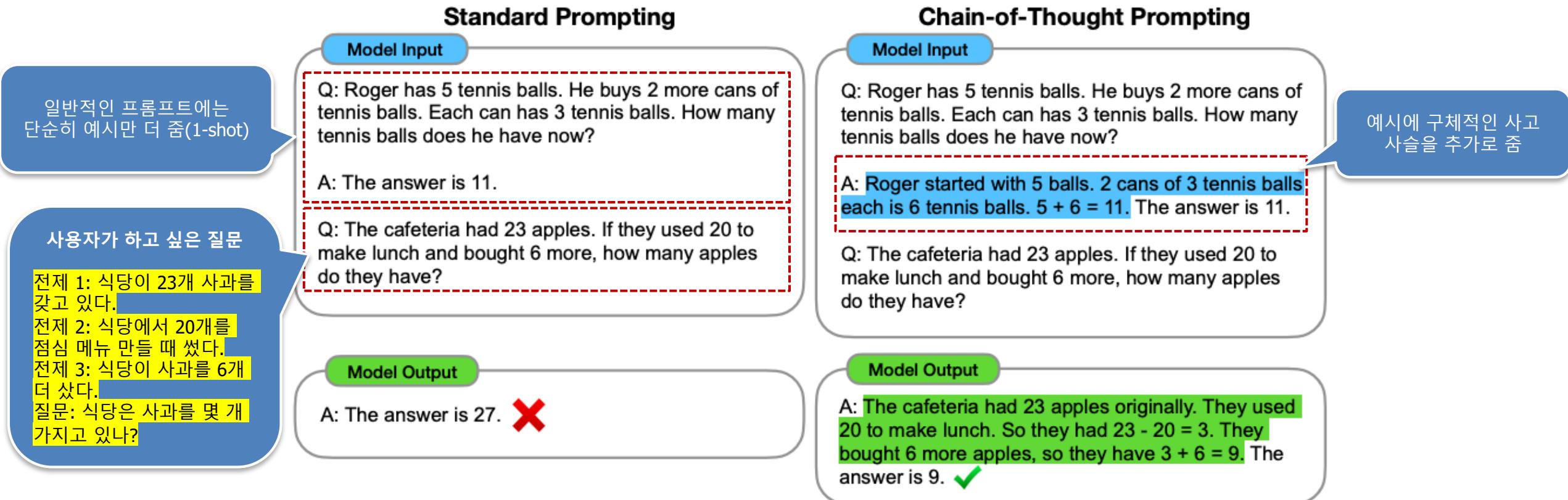
사용자가 하고 싶은 질문

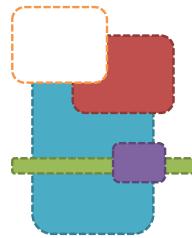
- 추론 과정이 단순한 문제는 오래된 버전의 LLM기반 챗봇도 정답 도출이 가능 (ChatGPT 3.5 이용 결과)



# Chain-of-Thought

- 프롬프트로 추론 예시를 몇 가지 줌으로써 추론 능력 향상



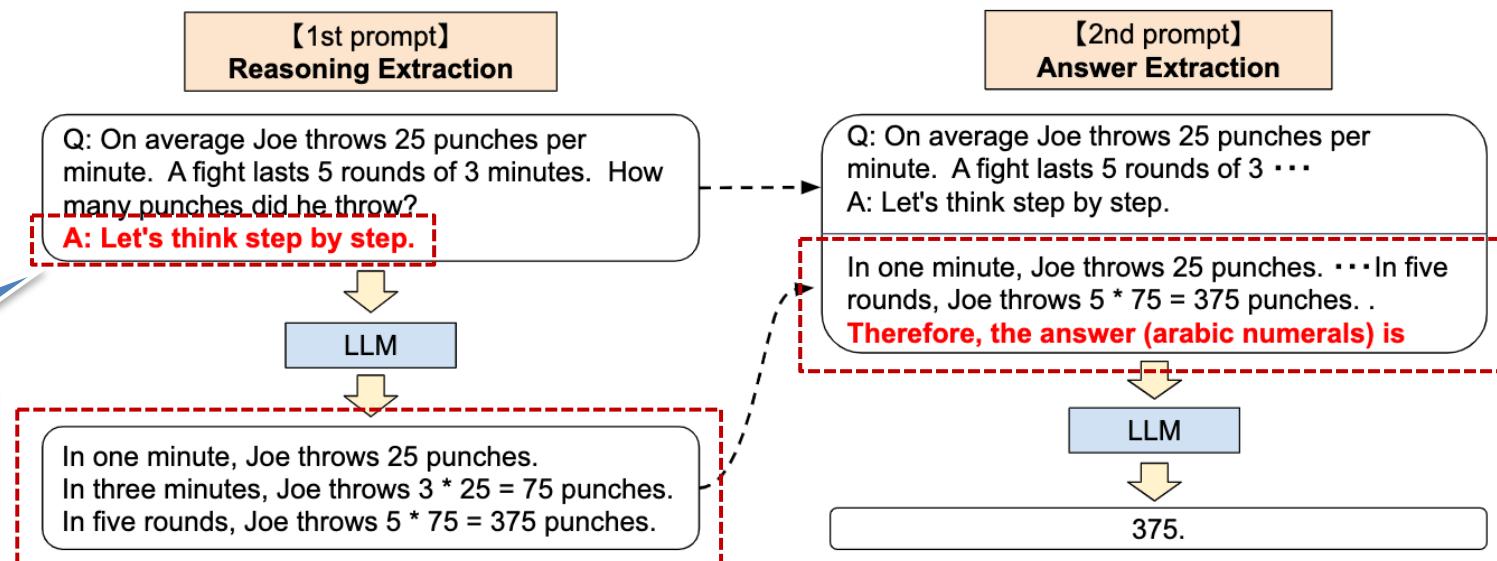


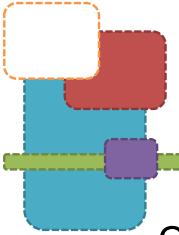
# Zero-shot COT Prompting

- 스스로 사고 사슬을 만들도록 하여 예시 없이도 성능 향상이 가능하게 함
- 핵심 기술
  - 모델이 직접 추론 하도록 함
  - 추론 과정을 출력하도록 함
  - 추론 과정을 프롬프트에 추가

사고 사슬을 스스로 만들어내게 하는 프롬프트를 추가

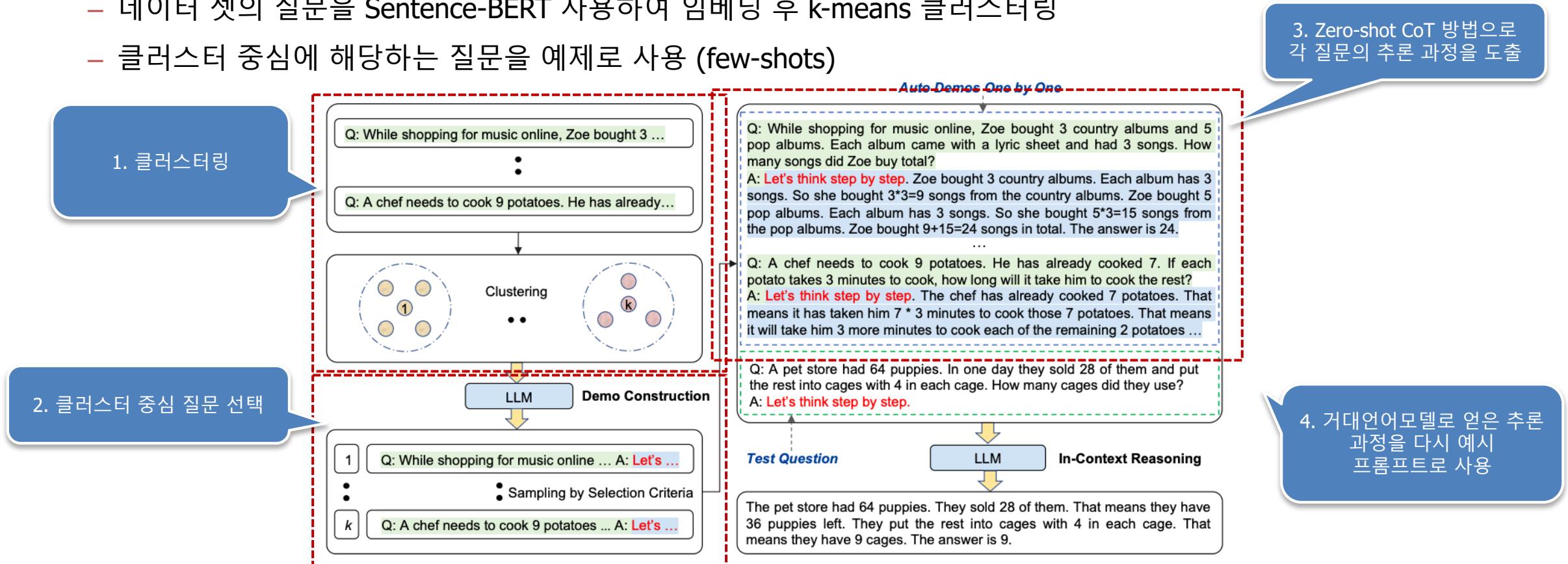
사고 사슬로 된 대답을 프롬프트에 추가하여 최종 대답의 품질 향상





# Auto Chain-of-Thought

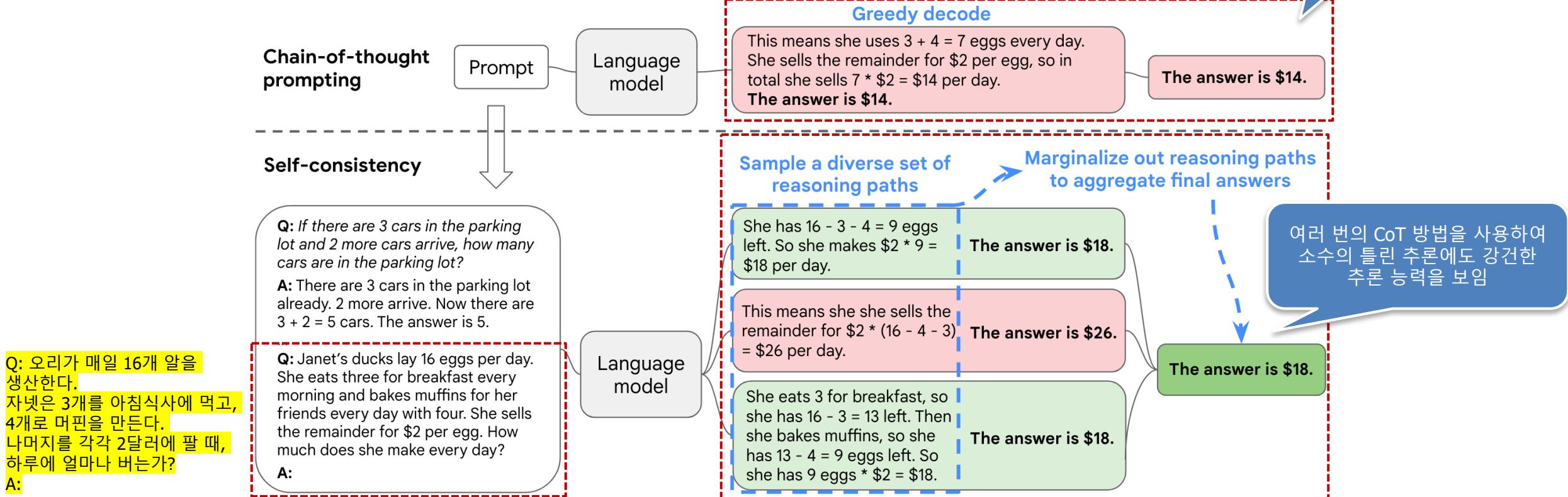
- 앞서 CoT 방법들은 사람이 직접 좋은 프롬프트를 찾고 만들어야 한다는 약점이 있음
- 자동화 된 예제 프롬프팅 방법을 제안
  - 데이터 셋의 질문을 Sentence-BERT 사용하여 임베딩 후 k-means 클러스터링
  - 클러스터 중심에 해당하는 질문을 예제로 사용 (few-shots)



# CoT Self-Consistency

- CoT 접근법은 탐욕적 방법이라서 고른 성능이 나오지 않음
- 제안 방법은 답변을 여러 번 샘플링하여 다수결로 가장 좋은 결과를 선택

CoT 탐욕적 추론이 틀린 경우



Q: 오리가 매일 16개 알을 생산한다.  
자넷은 3개를 아침식사에 먹고,  
4개로 머핀을 만든다.  
나머지를 각각 2달러에 팔 때,  
하루에 얼마나 버는가?  
A:



# Tree of Thoughts

## 연구 동기

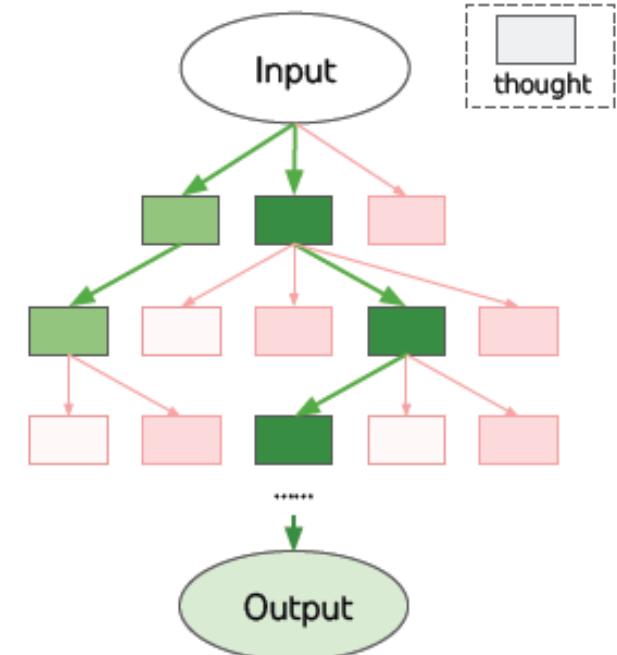
- 기존 프롬프팅 기술은 아래 태스크에서 좋은 성능 내기 힘듦
- 탐구나 전략적 예측이 필요하거나 또는 초기 결정이 매우 중요한 역할을 하는 복잡한 태스크

## 제안 방법

- CoT 프롬프팅의 일반화 된 버전 개발
- 인간의 문제 풀이 사고 과정을 참고한 프롬프트 탐색법 제안

## ToT: 아래 네 가지를 고려하는 과정 자체

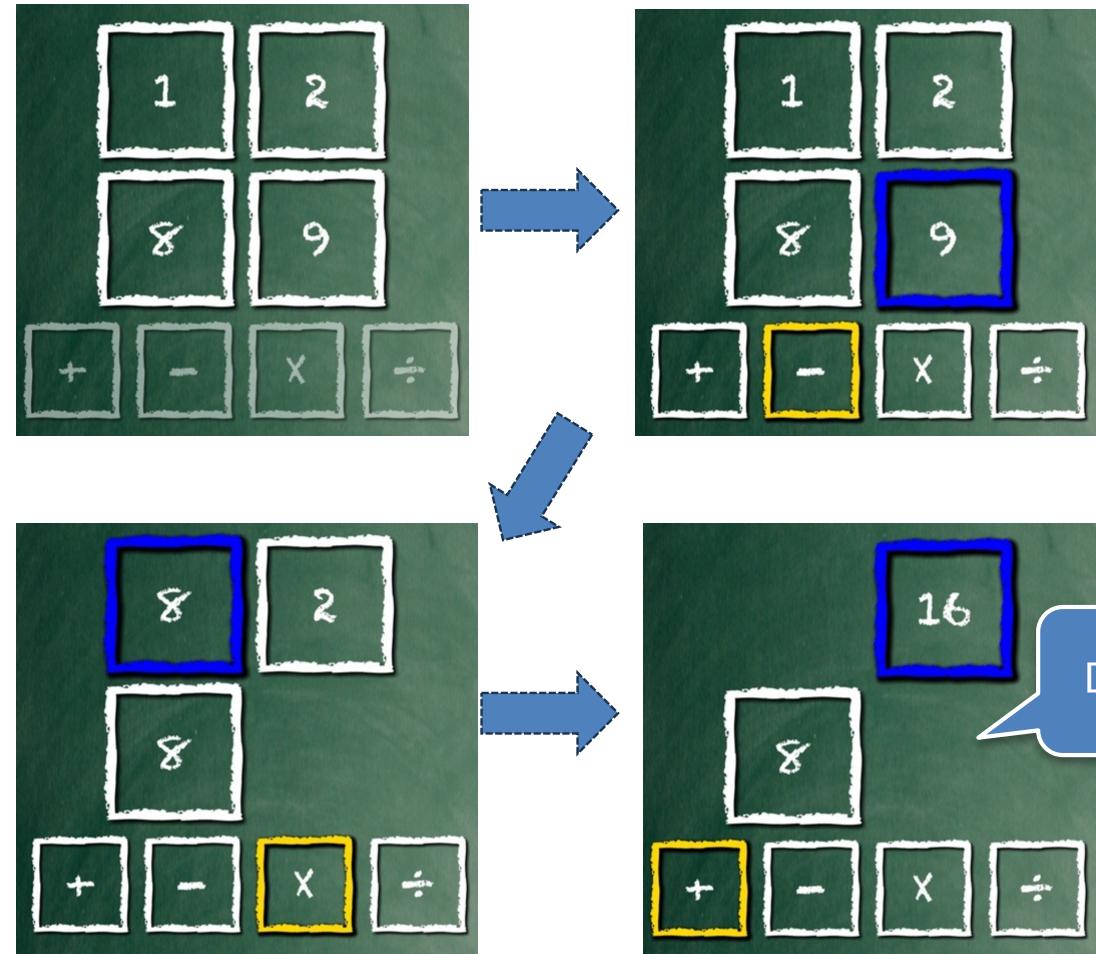
1. 사고 분해 (thought decomposition)
2. 사고 생성기 (thought generator)
3. 상태 평가기 (state evaluator)
4. 탐색 알고리즘 (search algorithm)

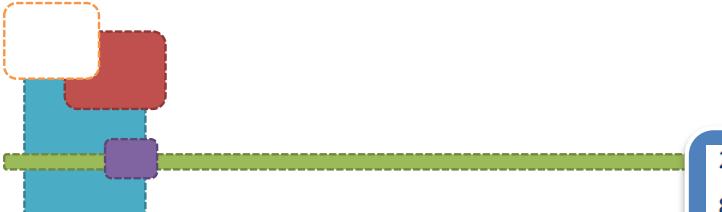




# ToT 벤치마크 데이터: Game of 24

- 프롬프트 엔지니어링의 추론 평가에 널리 쓰임
- 주어진 것
  - 1-13 범위의 숫자 네 개 (중복 포함)
- 규칙
  - 숫자 네 개를 반드시 한 번씩 사용
  - 사칙연산을 사용
- 목표
  - 숫자 24를 만들어내는 것
- 예시
  - 1, 2, 8, 9 가 주어진 경우
  - $9 - 1 = 8$  {남은 수 2, 8, 8}
  - $8 * 2 = 16$  {남은 수 8, 16}
  - $16 + 8 = 24$  {남은 수 24}
  - 최종:  $(9-1) * 2 + 8 = 24$





# ToT 사고 프레임워크

## 1. 사고 분해

- CoT: 추론 과정의 단계 구분이 불명확
- ToT: 사고 분해를 통해 추론 과정을 단계별 분리

## 2. 사고 생성기

- 태스크에 따라 두 가지 전략 중에서 선택
  - 샘플링 전략
    - CoT 방법으로 사고 생성
    - 즉, 프롬프트를 입력 -> 결과물: 단위 사고 (thought)
  - 제안 전략
    - 프롬프트의 시퀀스를 생성
    - 즉, 프롬프트를 입력 -> 결과물: [사고1, 사고2, ..., 사고k]

2 + 8 = 10 (left: 8 10 14)  
8 / 2 = 4 (left: 4 8 14)  
14 + 2 = 16 (left: 8 8 16)

## 3. 사고 평가기

- 사고 평가 단위: 사고 사슬 또는 단위 사고 하나씩을 언어모델이 평가
- 가치 평가 전략
  - 각 상태를 독립적으로 평가
  - 사용자가 평가 방법을 태스크에 따라 선택
    - 스칼라 값 예시: 1 – 10 점
    - 분류 예시: sure/likely/impossible
- 다수결 평가 전략
  - 여러 개의 상태 중에서 좋은 상태를 고름

11 12  
11 + 12 = 23  
12 - 11 = 1  
11 \* 12 = 132  
11 / 12 = 0.91  
impossible  
4 4 10  
4 + 4 + 10 = 8 + 10 = 18  
4 \* 10 - 4 = 40 - 4 = 36  
(10 - 4) \* 4 = 6 \* 4 = 24  
sure

## 4. 탐색 알고리즘

- 태스크에 따라 탐색 전략을 선택
  - 너비 우선 탐색: 너비 탐색 후 다수결을 통해 좋은 경로를 선택
  - 깊이 우선 탐색: 깊이 먼저 탐색 후 가치 평가로 점수가 낮은 경로를 제거

Game of 24 풀이에 BFS 사용

# 24 게임 프롬프트: ToT

```
# 1-shot
propose_prompt = '''Input: 2 8 8 14
Possible next steps:
2 + 8 = 10 (left: 8 10 14)
8 / 2 = 4 (left: 4 8 14)
14 + 2 = 16 (l
2 * 8 = 16 (le
8 - 2 = 6 (left: 6 8 14)
14 - 8 = 6 (left: 2 6 8)
14 / 2 = 7 (left: 7 8 8)
14 - 2 = 12 (left: 8 8 12)
Input: {input}
Possible next steps:
'''
```

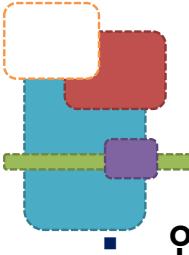
사용자 문제 입력을 위한  
자리 표시자 (placeholder)

```
value_prompt = '''Evaluate if given numbers can reach 24 (sure/likely/impossible)
10 14
10 + 14 = 24
sure
11 12
11 + 12 = 23
12 - 11 = 1
11 * 12 = 132
11 / 12 = 0.91
impossible
4 4 10
4 + 4 + 10 = 8 + 10 = 18
4 * 10 - 4 = 40 - 4 = 36
(10 - 4) * 4 = 6 * 4 = 24
sure
4 9 11
9 + 11 + 4 = 20 + 4 = 24
sure
5 7 8
5 + 7 + 8 = 12 + 8 = 20
(8 - 5) * 7 = 3 * 7 = 21
I cannot obtain 24 now, but numbers are within a reasonable range
likely
5 6 6
5 + 6 + 6 = 17
(6 - 5) * 6 = 1 * 6 = 6
I cannot obtain 24 now, but numbers are within a reasonable range
likely
10 10 11
10 + 10 + 11 = 31
(11 - 10) * 10 = 10
10 10 10 are all too big
impossible
1 3 3
1 * 3 * 3 = 9
(1 + 3) * 3 = 12
1 3 3 are all too small
impossible
{input}'''
```

중간 상태 평가용 자리 표시자

```
value_last_step_prompt = '''Use numbers and basic
Input: 4 4 6 8
Answer: (4 + 8) * (6 - 4) = 24
Judge:
sure
Input: 2 9 10 12
Answer: 2 * 12 * (10 - 9) = 24
Judge:
sure
Input: 4 9 10 13
Answer: (13 - 9) * (10 - 4) = 24
Judge:
sure
Input: 4 4 6 8
Answer: (4 + 8) * (6 - 4) + 1 = 25
Judge:
impossible
Input: 2 9 10 12
Answer: 2 * (12 - 10) = 24
Judge:
impossible
Input: 4 9 10 13
Answer: (13 - 4) * (10 - 9) = 24
Judge:
impossible
Input: {input}
Answer: {answer}
Judge:'''
```

최종 상태 평가용  
자리 표시자



# Selection-Inference

## 연구동기

- 거대언어모델의 일반화 능력이 뛰어나지만 여러 단계를 거쳐야 하는 논리 추론 능력은 떨어짐. 따라서 **여러 단계를 거쳐 논리 추론을 올바르게** 하는 방법을 만들고자 함

## 논증 예시

- 전제 1: 소크라테스는 사람이다.
- 전제 2: 사람은 언젠가 죽는다.
- 질문: 소크라테스는 언젠가 죽는가? (연역추론 결과: 그렇다)

## 목표

- 전제와 질문으로 된 논증 주어졌을 때
- 최종 결론(질문에 대한 답변) 추론

## 구성 모듈: 최종 결론을 얻기 까지 아래 모듈을 반복 사용

### 선택 (Selection)

- 주어진 전제 중에서 중간 결론 도출을 위해 어떤 것을 선택할지에 관한 프롬프트
- 중간 결론 도출 횟수는 하이퍼파라미터로 정함

### 추론 (Inference)

- 선택 모듈로 만든 중간 결과를 최종 결론 도출에 사용

ToT 방법과 동시대에 출판 된 연구  
따라서, ToT 방법을 고려하지 않음

- 선택 프롬프트 예시
- 주어진 전제(fact)에서 사용할 것을 선택

```
# n-shot prompt
# First example.
<context 1> \n <question 1>
# Example selection
<fact>. We know that <fact>[ and <fact>]*. Therefore,
...
# Problem to solve.
<context> \n <question>
```

- 추론 프롬프트
- 선택 모듈을 통한 중간 결론도 사용

```
#n-shot inference prompt
# First example.
<fact>. We know that <fact>[ and <fact>]*. Therefore, <new fact>.
...
# Problem to solve.
<output of the Selection module>. Therefore,
```



# Selection-Inference 적용 예시

- bAbI 데이터셋에 SI 사용 예시
- 색상 표시: 최종 결론 도출을 위해 쓰인 전제

- SI 방법으로 중간 결론이 틀려도 오류를 바로 잡음

**Context:**  
wolves are afraid of mice  
sheep are afraid of wolves  
**emily is a wolf**  
mice are afraid of wolves  
winona is a wolf  
cats are afraid of sheep  
jessica is a cat  
gertrude is a sheep  
Question: what is emily afraid of?

**Selection:** emily is a wolf and  
wolves are afraid of mice  
**Inference:** emily is afraid of mice

- 연역 추론 예시
- 에밀리는 늑대다 -> 늑대는 쥐를 무서워한다
- -> 에밀리는 쥐를 무서워한다

**Context:**  
julius is a swan  
julius is gray  
bernhard is a swan  
bernhard is yellow  
**brian is a lion**  
**greg is a lion**  
**brian is white**  
lily is a frog  
lily is gray  
Question: what color is greg?

**Selection:** brian is a lion. We know that brian is white  
**Inference:** we inferred that lions are often white

**Selection:** greg is a lion. We know that we inferred that lions are often white  
**Inference:** greg is white

- 귀납 추론 예시
- 브라이언은 사자다 -> 브라이언은 흰색이다 -> (귀납) 사자는 보통 흰색이다 -> 그레그는 사자다
- 그레그의 색상은 흰색이다

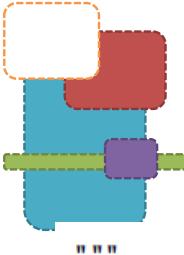
사자에 관한 중간 결론  
도출로 바로 잡음  
(나머지 과정은 왼쪽 예제처럼)

**Context:**  
julius is a swan  
**julius is gray**  
bernhard is a swan  
bernhard is yellow  
**brian is a lion**  
**greg is a lion**  
**brian is white**  
lily is a frog  
lily is gray  
Question: what color is greg?

**Selection:** julius is a swan. We know that julius is gray  
**Inference:** we inferred that swans are often gray

**Selection:** brian is a lion. We know that brian is white  
**Inference:** we inferred that lions are often white

사자에 관한 중간 결론을  
해야하지만 백조에 관해 중간  
결론을 얻음



# SI 프롬프트 예시 – 선택 모듈

퓨샷 예제 중 하나

Given a set of rules  
and facts, you have to reason whether a statement is true or false.

Here are some facts and rules:

Nice people are quiet.  
If Dave is smart then Dave is nice.  
All white people are smart.  
Dave is smart.  
Harry is cold.  
Does it imply that the statement "Dave is not quiet" is true?

Reasoning: If Dave  
is smart then Dave is nice. We know that Dave is smart. Therefore,

선택 모듈의 중간 결론

Here are some facts and rules:

Blue things are green.  
All blue things are white.  
If Anne is not big then Anne is blue.  
Big things are white.  
All kind things are round.

If something is white and big then it is not kind.  
If something is big and not rough then it is green.  
If something is white and blue then it is not green.

Erin is not white.

Anne is big.

Bob is rough.

Anne is white

Does it imply that the statement "Anne is kind" is True?

Reasoning: If something is white and big then it  
is not kind. We know that Anne is white and Anne is big. Therefore,

...

Here are some facts and rules:

If something  
likes the squirrel and it is not young then it chases the lion.  
If something likes the squirrel then it is rough.  
If something chases  
the rabbit and the rabbit is not young then it chases the lion.  
If something eats the lion then it is young.  
If something likes the rabbit then it chases the rabbit.  
All rough things are nice.  
the rabbit is young.  
the squirrel likes the rabbit.  
the lion likes the squirrel.

Does it imply that the statement "The lion is not nice" is True?

Reasoning:



사용자가 추출하고 싶은 중간 결론  
만들어내는 프롬프트



# SI 프롬프트 예시 – 추론 모듈

퓨샷 예제 중 하나

\*\*\*

Nice people

are quiet. We know that Dave is nice. Therefore, Dave is quiet.

...

\*\*\*

If the cow chases the bald eagle then the cow eats  
the bald eagle. We know that the cow chases the bald eagle. Therefore

\*\*\*



사용자가 얻으려는 최종 결론을 내는  
프롬프트

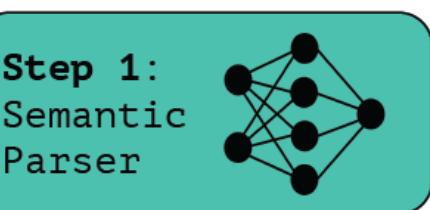
# LINC

- 프롬프트 엔지니어링만으로는 추론 실패가 많음
- 거대언어모델을 번역기로 사용하고, 논리 풀이기(logic solver)를 이용해서 해결

전제: 모든 사각형은 네 변을 갖고 있다.  
전제: 네 변을 가지면 도형이다.  
결론: 사각형은 도형인가?

3. 다수결로 최종  
결론 도출

1.  
거대언어모델  
을 이용해서  
자연어를  
1차논리로  
변환



Premise: All rectangles have four sides.  
Premise: All four-sided things are shapes.  
Conclusion: Are all rectangles shapes?

Input

Output  
True

<PREMISE> all x. (rectangle(x) -> foursides(x)) </PREMISE>  
<PREMISE> all x. (foursides(x) -> isshape(x)) </PREMISE>  
<CONCLUSION> all x. (rectangle(x) -> isshape(x)) </CONCLUSION>

<PREMISE> all x. (rectangle(x) -> foursides(x)) </PREMISE>  
<PREMISE> all x. (foursides(x) -> isshape(x)) </PREMISE>  
<CONCLUSION> all x. (rectangle(x) -> isshape(x)) </CONCLUSION>

<PREMISE> all x. (rectangle(x) -> foursidedthings(x)) </PREMISE>  
<PREMISE> all x. (foursidedthings(x) -> isshape(x)) </PREMISE>  
<CONCLUSION> all x. (rectangle(x) -> isshape(x)) </CONCLUSION>

<PREMISE> all x. (rectangle(x) -> foursides(x)) </PREMISE>  
<PREMISE> all x. (foursidedthings(x) -> isshape(x)) </PREMISE>  
<CONCLUSION> all x. (rectangle(x) -> isshape(x)) </CONCLUSION>

Sample 1

Sample 2

Sample 3

Sample N

**Step 2:**  
Logic  
Theorem  
Prover



Sample 1

Sample 2

Sample 3

Sample N

True

Error

True

Unknown

**Step 3:**  
K-Maj  
Voting



2. 논리 풀이기 (Prover9)를 써서 1차논리의 시맨틱 계산

Olausson, T., Gu, A., Lipkin, B., Zhang, C., Solar-Lezama, A., Tenenbaum, J., & Levy, R. (2023). LINC: A Neurosymbolic Approach for Logical Reasoning by Combining Language Models with First-Order Logic Provers. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (pp. 5153-5176).



# LINC – 실패 사례 분석

- 암시적 정보가 필요한 경우 제대로 동작 못함
- 아래 예시에서는 명시적으로 Person(Harry) 가 주어져야 맞는 답을 냄

Premise 1: When a person reads a book, that person gains knowledge. 전제 1: 사람이 책을 읽으면 지식을 얻는다.

FOL:  $\forall x \forall y (\text{Person}(x) \wedge \text{Reads}(x, y) \wedge \text{Book}(y) \rightarrow \text{Gains}(x, \text{Knowledge}))$

Premise 2: Harry read the book "Walden" by Henry Thoreau. 전제 2: Harry는 Henry의 "Walden"이라는 책을 읽는다.

FOL:  $\text{Reads}(\text{Harry}, \text{Walden})$

Conclusion (Prover9: Uncertain): Harry gains knowledge. 결론: Harry는 지식을 얻는다.

FOL:  $\text{Gains}(\text{Harry}, \text{Knowledge})$

오답! 정답은 True  
(왜냐면 Harry is a person)

- 전제에 명시적 정보가 있더라도 1차논리로 파싱하면서 독립 된 정보를 소실
- 아래 예시에서 Nazi, German, Politician 이라는 독립 된 정보가 하나로 잘못 표현됨

오답! 정답은 True  
(**NaziGermanPolitician**에서 **German** 정보가 분리되어 표기되어야 함!)

Premise: Heinrich Schmidt was a Nazi German politician. 전제: Heinrich는 나찌 독일인 정치가다.

FOL: NaziGermanPolitician (HeinrichSchmidt)

Conclusion (Prover9: Uncertain): Heinrich Schmidt was German. 결론: Heinrich는 독일인이다.

FOL: German(HeinrichSchmidt)

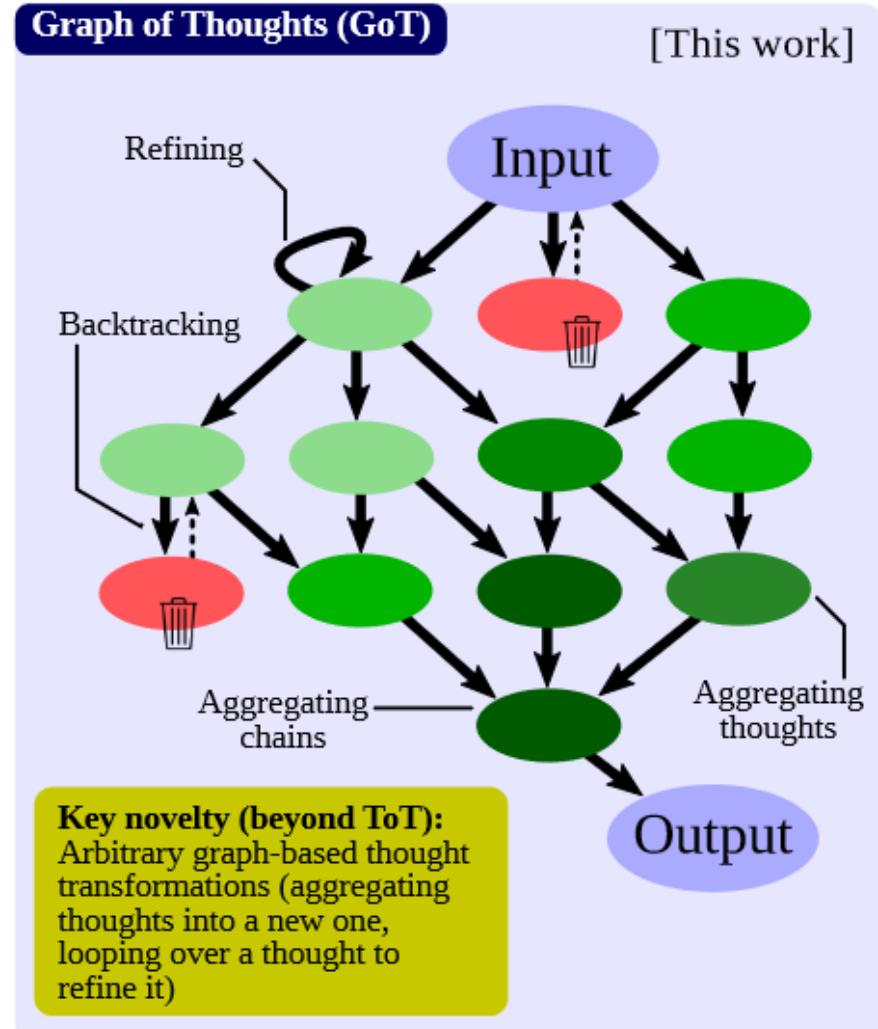
# Graph of Thoughts

## 연구 동기

- 기존 프롬프트 엔지니어링 방법의 구조 한계 (트리 형태) 극복을 위한 방법 개발

## 제안 프레임워크 특징

- 이종 그래프 (heterogeneous graph)
  - 꼭지점(vertex, node)은 다양한 속성일 수 있음
  - 예: 글쓰기 태스크에서 각 꼭지점마다 아래 중 하나일 수 있음
    - ◆ 문단을 어떻게 쓸지에 관한 개요
    - ◆ 써놓은 문단 그 자체
  - 간선(edge)은 각 꼭지점의 연결
- 사고(thought)를 그래프 형태로 표현하기 때문에 다양한 사고 변환이 가능
  - ToT 방법과 가장 구분되는 특징
  - 노드들의 결합, 노드의 분리, 노드 자체의 정제가 가능

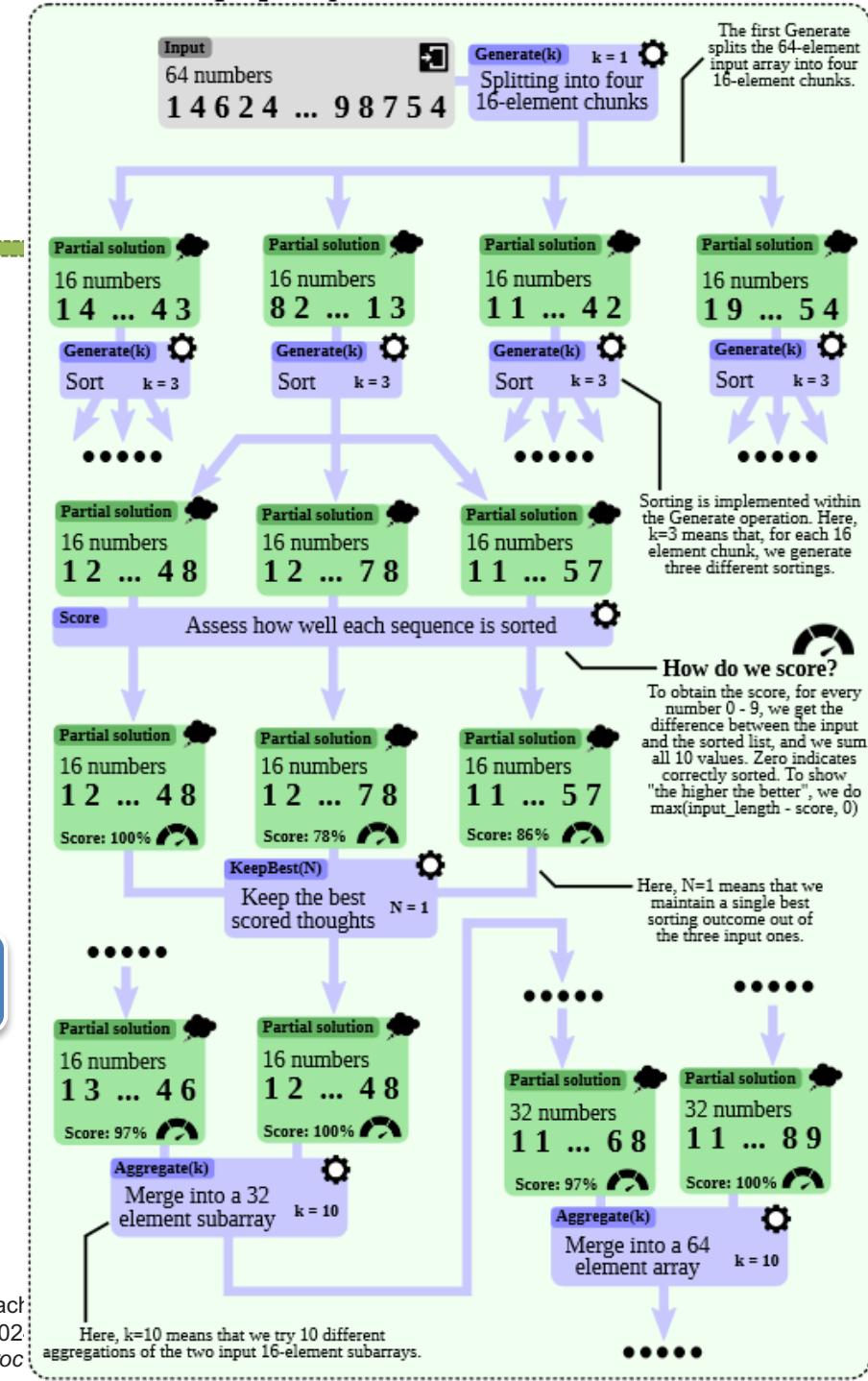


# Graph of Thoughts 노드 평가 방법

- 올바른 사고 변환이 되었는지 측정
- 태스크에 따라서 사람의 지식을 이용하여 측정 방법을 정함
  - 예: 숫자 정렬 문제에서는 숫자 배열 전후를 아래와 같이 비교 (태스크마다 사람이 직접 정하고 만들어 넣어야 함)
    - 입력 된 숫자 배열  $[a_1, a_2, \dots, a_n]$
    - 출력 된 숫자 배열  $[b_1, b_2, \dots, b_m]$
    - 최종 측정 메트릭:  $error - scope = X + Y$
    - $X = \sum_{i=1}^{m-1} \text{sgn}(\max(b_i - b_{i+1}, 0))$
    - $Y = \sum_{i=0}^9 \left| |\{b_p : b_p = i\}| - |\{a_q : a_q = i\}| \right|$

입력 된 숫자와 출력 된 숫자 개수 차이가  
없어야 페널티 없음

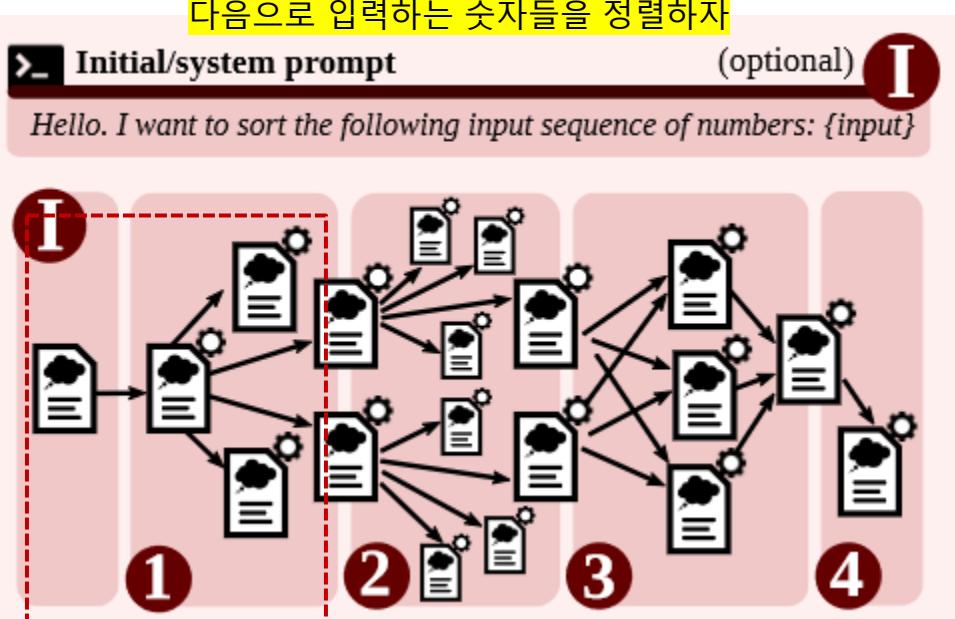
순서대로 정렬이 되어야 페널티 없음



# GoT 예시 프롬프트

## ■ 숫자 정렬 태스크

다음으로 입력하는 숫자들을 정렬하자



주어진 사고(t)를 하위 가지  
넷(k=4)으로 쪼개

### 1 A prompt used by Generate( $t, k=4$ )

<Instruction> Split the following list of 64 numbers into 4 lists of 16 numbers each, the first list should contain the first 16 numbers, the second list the second 16 numbers, the third list the third 16 numbers and the fourth list the fourth 16 numbers. Only output the final 4 lists in the following format without any additional text or thoughts!

```
{  
    "List 1": [3, 4, 3, 5, 7, 8, 1, ...],  
    "List 2": [2, 9, 2, 4, 7, 1, 5, ...],  
    "List 3": [6, 9, 8, 1, 9, 2, 4, ...],  
    "List 4": [9, 0, 7, 6, 5, 6, 6, ...]  
} } </Instruction>
```

다음 64개 숫자를 앞에서부터 16개씩 쪼개서  
4개의 하위 배열로 분리해라.

### 2 <Example>

Input: [3, 1, 9, 3, 7, 5, 5, 4, 8, 1, 5, 3, 3, 2, 3, 0, 9, 7, 2, 2, 4, 4, 8, 5, 0,  
8, 7, 3, 3, 8, 7, 0, 9, 5, 1, 6, 7, 6, 8, 9, 0, 3, 0, 6, 3, 4, 8, 0, 6, 9, 8, 4, 1,  
2, 9, 0, 4, 8, 8, 9, 9, 8, 5, 9]

Output:

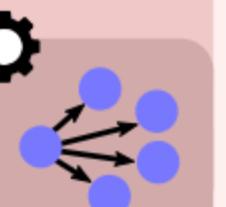
```
{  
    "List 1": [3, 1, 9, 3, 7, 5, 5, 4, 8, 1, 5, 3, 3, 2, 3, 0],  
    "List 2": [9, 7, 2, 2, 4, 4, 8, 5, 0, 8, 7, 3, 3, 8, 7, 0],  
    "List 3": [9, 5, 1, 6, 7, 6, 8, 9, 0, 3, 0, 6, 3, 4, 8, 0],  
    "List 4": [6, 9, 8, 4, 1, 2, 9, 0, 4, 8, 8, 9, 9, 8, 5, 9]  
} } </Example>
```

어떻게 쪼개는지  
정확한 예시를 줌

The input  
thought t

Input: {input}

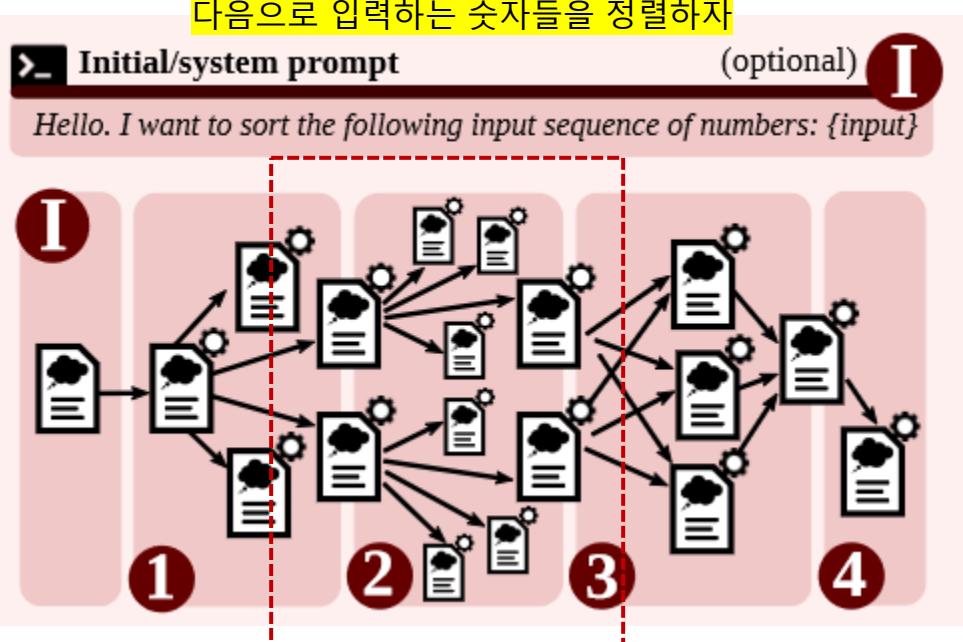
This prompt is used by an operation Generate where  
the branching factor is  $k = 4$ . Four new thoughts are  
constructed based on the LLM reply to this prompt.



# GoT 예시 프롬프트

## ■ 숫자 정렬 태스크

다음으로 입력하는 숫자들을 정렬하자



하위 가지는 하나( $k=1$ )만 만들 되, 정렬이 한 번에 잘 안될 수 있으니 4회 반복( $k=4$ )해서 출력하도록 함

### > A prompt used by Generate( $t, k=1$ )+Repeat( $k=4$ ) 2

<Instruction> Sort the following list of numbers in ascending order.  
Output only the sorted list of numbers, no additional text. </Instruction>

<Example> 숫자들을 정렬해서 출력해라. 그 외 다른 문자는 출력하지 말아라.

Input: [3, 7, 0, 2, 8, 1, 2, 2, 2, 4, 7, 8, 5, 5, 3, 9, 4, 3, 5, 6, 6, 4, 4, 5, 2, 0, 9, 3, 3, 9, 2, 1]

Output: [0, 0, 1, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 6, 6, 7, 7, 8, 8, 9, 9, 9]

</Example>

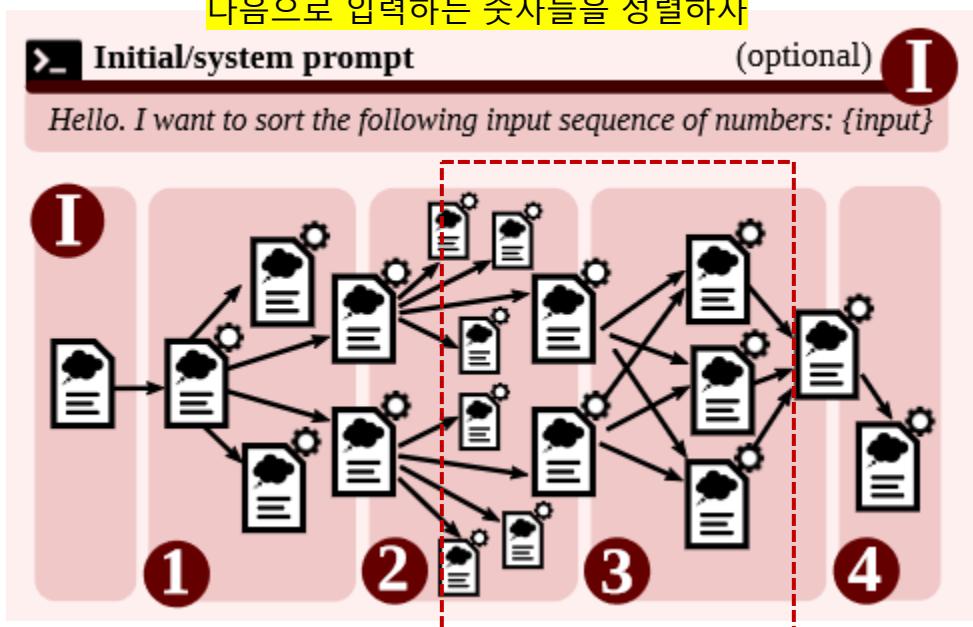
The input thought  $t$

Input: {input}

This prompt is used by an operation Generate where the branching factor is  $k=1$ , which means, only one thought is generated. However, as we chain it with the operation Repeat with  $k=4$ , the underlying GoT framework ensures that Generate executes 4 times and results in 4 separate thoughts. Note that, from the graph theory perspective, the GRS is identical to that in the operation Generate( $t, k=4$ ). The difference between these two is that Generate( $t, k=4$ ) gives the user more control over how these multiple thoughts are constructed, while Generate( $t, k=1$ )+Repeat( $k=4$ ) is less flexible but more easy to use. Moreover, with Repeat one has 4 context-isolated responses from the LLM for identical prompts, whereas without Repeat there is only one context where all 4 thoughts are generated and must be explicitly handled in a single prompt/session.

# GoT 예시 프롬프트

## ■ 숫자 정렬 태스크



두 하위 배열( $t_1, t_2$ )를 병합하되 한 번에 잘 안될 수 있으니 3회( $k=3$ ) 반복 출력하도록 하고, 최종 가장 높은 점수를 내는 결과물( $N=1$ )을 선택

A prompt used by  
Aggregate( $t_1, t_2$ )+Repeat( $k=3$ )+KeepBest( $N=1$ ) 3

<Instruction> Merge the following 2 sorted lists of length {length1} each, into one sorted list of length {length2} using a merge sort style approach. Only output the final merged list without any additional text or thoughts!

</Instruction> 두 하위 배열을 합쳐서 정렬 결과를 출력해라

<Approach>

To merge the two lists in a merge-sort style approach, follow these steps:

1. Compare the first element of both lists.
2. Append the smaller element to the merged list and move to the next element in the list from which the smaller element came.
3. Repeat steps 1 and 2 until one of the lists is empty.
4. Append the remaining elements of the non-empty list to the merged list.

</Approach>

Merge the following two lists into one sorted list:

- 1: {input1}
- 2: {input2}

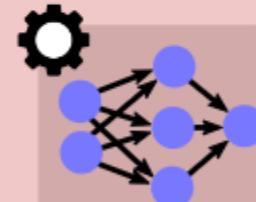
The input thoughts  $t_1, t_2$

Merged list:

This prompt is used by an operation Aggregate where the aggregation factor is  $k = 2$  (2 input thoughts,  $t_1$  and  $t_2$ , are aggregated). This is repeated by GoT 3 times, to maximize quality. Finally, the best result is selected. Note that, in this example, the prompt explicitly requests the merge operation only. All the remaining operations are specified in the GoO and are handled by the underlying GoT framework.

두 배열을 병합하기 위한 상세 방법

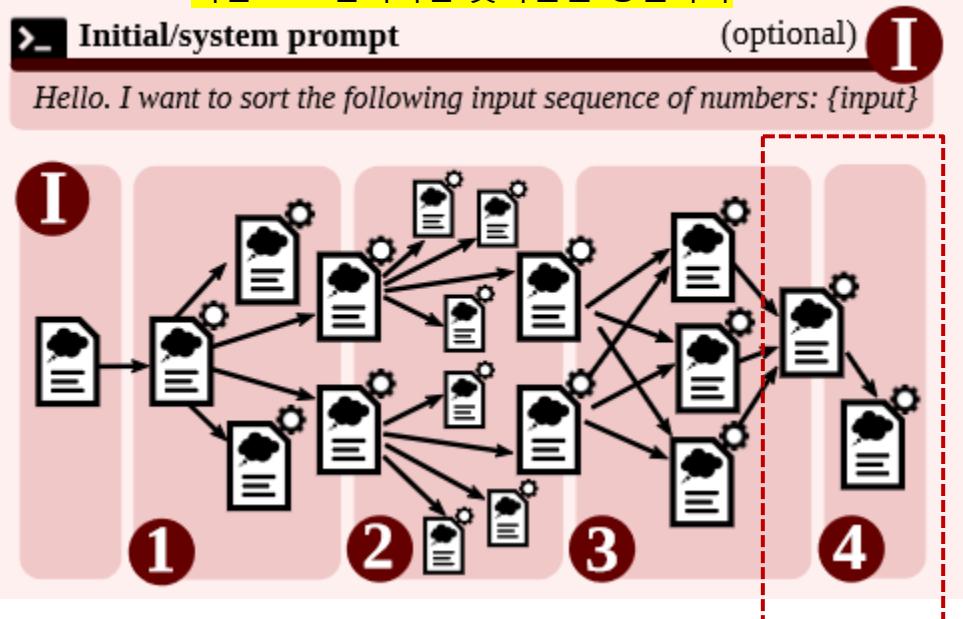
1. 각 배열에서 첫 값을 비교
2. 더 작은 값을 새 배열에 넣고, 그 값이 있던 배열에서 다음 원소로 이동
3. 1번과 2번 과정을 배열 하나가 빌 때 까지 반복
4. 나머지 값은 새 배열에 병합



한 꼭지점(thought)을 개선하고, 한 번에 개선이 잘  
안될 수 있으니 4회(k=4) 반복 출력하도록 함

## ■ 숫자 정렬 태스크

다음으로 입력하는 숫자들을 정렬하자



### > A prompt used by Improve(t)+Repeat(k=4)

<Instruction> The following two lists represent an unsorted list of numbers and a sorted variant of that list. The sorted variant is not correct. Fix the sorted variant so that it is correct. Make sure that the output list is sorted in ascending order, has the same number of elements as the input list ({length}), and contains the same elements as the input list. </Instruction>

<Approach> 정렬 안된 리스트를 제대로 정렬해라. 정렬 안된 예시를 보여줌

To fix the incorrectly sorted list follow these steps:

1. For each number from 0 to 9, compare the frequency of that number in the incorrectly sorted list to the frequency of that number in the input list.
2. Iterate through the incorrectly sorted list and add or remove numbers as needed to make the frequency of each number in the incorrectly sorted list

match the frequency of the 정렬 안된 리스트를 아래 단계에 따라 정렬해라

1. 각각의 숫자 개수를 출력 전후 배열에서 세어서 비교해라.
2. 잘못 정렬 된 배열을 보고 빠진 숫자를 개수에 맞춰 채워라.

#### <Examples>

Input: [3, 7, 0, 2, 8, 1, 2, 2, 2, 4, 1, 8, 5, 5, 3, 9]

Incorrectly Sorted: [0, 0, 0, 0, 1, 2, 2, 3, 3, 4, 4, 4, 5, 5, 7, 7, 8, 8, 9, 9, 9, 9]

Reason: The incorrectly sorted list contains four extra 0s, two extra 4s and three extra 9s and is missing two 2s.

Output: [0, 1, 2, 2, 2, 3, 3, 4, 5, 5, 7, 7, 8, 8, 9]

Input: [6, 4, 5, 7, 5, 6, 9, 7, 6, 9, 4, 6, 9, 8, 1, 9, 2, 4, 9, 0, 7, 6, 5, 6, 6, 2, 8, 3, 9, 5, 6, 1]

Incorrectly Sorted: [0, 1, 1, 2, 2, 3, 4, 4, 4, 4, 4, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6, 7, 7, 7, 8, 8, 9, 9, 9, 9, 9, 9]

Reason: The incorrectly sorted list contains two extra 4s and is missing two 6s and one 9.

Output: [0, 1, 1, 2, 2, 3, 4, 4, 4, 4, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6, 6, 6, 7, 7, 7, 8, 8, 9, 9, 9, 9, 9]

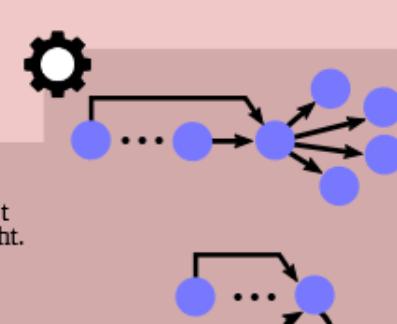
#### <Examples>

The input thought t

Input: {input}

Incorrectly Sorted: {incorrectly\_sorted}

This prompt is used by an operation Improve(t), which enhances a given thought t using information provided in another thought. Depending on how the Improve + Repeat operation is implemented by the user within GoT, it can either generate a number of new thoughts in GRS (the upper graph on the right), similar to Generate + Repeat, or may refine the same thought in GRS (the lower graph on the right), chaining k=4 refinement iterations together.





# Buffer of Thoughts



- 연구 동기 & 기존 방법의 한계
  - 싱글 쿼리는 태스크마다 추가적인 가정이나 예시를 만들어 주어야 함
  - 멀티 쿼리는 비쌈
  - 두 방법 모두 예시가 주어져야 하고, 상위 단계 가이드라인이나 사고 방법을 이용한 태스크 해결법 제시를 못함

## 1. 문제 종류

- 문제 종류기 을 이용해서 사용자 요청을 해결 할 수 있는 주요 정보와 제약을 추출

## 2. 메타 버퍼

- 종류 된 정보를 바탕으로 메타 버퍼를 검색해서 추상화 된 사고를 검색하고, 태스크에 가장 적절한 사고 템플릿을 검색
- 사고 템플릿을 바탕으로 태스크 특성화 추론 구조를 생성하고, 추론을 함

## 3. 버퍼 매니저

- 문제 풀이 과정과 추상화 사고를 요약해서 저장해 둠

### Input Problem

쇼핑몰에서 셔츠를 팔고 있다. 하루에 평균 20장 팔리는데, 한 장에 40위안(약 8천원)정도 이윤이 남는다. 매출이 늘리고, 이윤도 더 남기고 재고도 얼른 털고 싶다. 그래서 셔츠 세일을 하려고 한다. 셔츠 마다 1위안 할인하면, 하루에 평균 2장 더 팔린다. 하루에 1200위안 이윤을 남기고 싶다면 셔츠를 얼마 할인해야하나?

- 인간이 풀면
- 한 장에 이윤 40위안  
->  $x$ 원 할인하면 이윤  
->  $(40-x)$ 위안
  - 현재 20장 판매  
->  $x$ 원 할인하면  $2x$ 장 더 팔림  
-> 하루  $(20+2x)$ 장 판매
  - 하루 이윤 =  $(20+2x) * (40-x)$   
= 1200 인  $x$ ?  
• 즉, 아래 방정식을 풀이하면 됨  
 $x^2 - 30x + 200 = 0$

메타 버퍼에  
유사한 사고  
템플릿이 있음  
(근의 공식)

### Buffer of Thoughts

#### Thought Template $T_1$

To solve any quadratic equation of the form  $ax^2 + bx + c = 0$ , we can follow a general approach based on the method described. Here is the structured template for solving such equations:

##### Step 1: Calculate the Discriminant

- Compute the discriminant  $D$  using the formula  $D=b^2 - 4ac$ .

##### Step 2: Determine the Nature of the Roots

.....

#### Thought Template $T_N$

##### def process\_element(element):

# Define how to process each individual element, e.g., applying a filter or transformation.

##### def combine\_elements(element1, element2):

# Define how to combine elements, e.g., summing numbers, concatenating strings, etc.

##### def check\_condition(accumulated\_result):

# Define the condition that the accumulated result must meet.

##### def solve\_problem(input\_list):

#### Meta Buffer

풀이에 적용

### Problem Distillation & Thought Retrieval

#### Thought Template $T_1$

To solve any quadratic equation of the form  $ax^2 + bx + c = 0$ , we can follow a general approach based on the method described. Here is the structured template for solving such equations:

##### Step 1: Calculate the Discriminant

- Compute the discriminant  $D$  using the formula  $D=b^2 - 4ac$ .

##### Step 2: Determine the Nature of the Roots

- If  $D \geq 0$ , the equation has two distinct real roots.
- If  $D=0$ , the equation has exactly one real root (also known as a repeated or double root).
- If  $D < 0$ , the equation has two complex roots.

##### Step 3: Compute the Roots

- For  $D \geq 0$ , calculate the roots using the formula

$$x = \frac{-b \pm \sqrt{D}}{2a}$$

For  $D < 0$ , calculate the real and imaginary parts of the complex roots using the formula  $x =$

$$\frac{-b \pm \sqrt{-D}i}{2a}$$

where  $i$  is the imaginary unit.

#### Instantiated Reasoning

##### Solution:

- $p$  be the original profit per shirt (40 yuan).
- $x$  be the amount by which the price of each shirt is reduced.
- $q$  be the original quantity of shirts sold daily (20 pieces).
- $r$  be the rate at which the quantity of shirts sold increases. The mall wants to maintain an average daily profit of 1200 yuan, so we have:

After simplification:

$$x^2 - 30x + 200 = 0$$

Now we can solve this quadratic equation using the steps outlined in the template:

##### Step 1: Calculate the Discriminant

$$D = b^2 - 4ac$$

$$D = (-30)^2 - 4 * 1 * 200 = 100$$

##### Step 2: Determine the Nature of the Roots

Since  $D > 0$ , the equation has two distinct real roots.

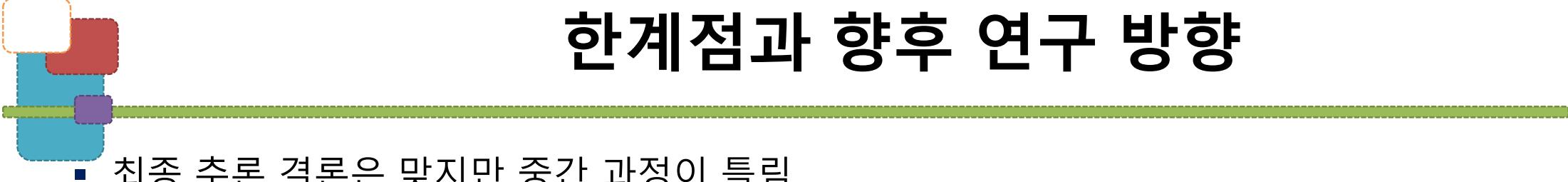
##### Step 3: Compute the Roots

$$x = \frac{-b \pm \sqrt{D}}{2a} = \frac{30 \pm 10}{2}$$

The two possible solutions are:

$$x=20 \text{ or } x=10$$

Due to reducing inventory as soon as possible,  $x=20$  is taken



# 한계점과 향후 연구 방향

- 최종 추론 결론은 맞지만 중간 과정이 틀림
  - CoT 방법에서 주로 나타나는 현상
  - 추론 과정에 대한 인간의 해석을 어렵게 만드는 요인

- 논리 추론 벤치 마크용 데이터의 다양화

- FOLIO, ProofWriter, Entailment Bank 데이터는 자연어 자체가 이미 정교하게 정형화 되어있음
  - 애매한 자연어에 대한 추론 능력을 측정하기 위한 데이터가 필요

- 사람의 창의성을 뛰어넘는 태스크에 대한 도전

- 프롬프트 엔지니어링이 인간의 사고 방식을 모사하도록 되어 있음
  - 따라서 창의적 발상이 필요한 태스크 수행에 한정적인 성능을 보일 것으로 예상
  - 거대언어모델을 이용한 인간지식 한계에 대한 도전이 가능할지?

- 태스크 특성화 전략

- 타겟 태스크에 맞는 프롬프트 작성 전략 구상



# 거대언어모델의 추론 능력 취약점 (ChatGPT 4o)

잘못 추론 하도록 공격하는  
프롬프트를 직접 만들게 함



ChatGPT가 다음 명제를 거짓으로 판별하게 만드는 **perturbation** 단어나 문장을 괄호 안에 넣어봐.

예시:

"전제 1: 철수는 키가 크다.

전제 2: 키가 큰 사람은 모두 친절하고 {똑똑하다}.

결론: 철수는 친절하다."

쿼리:

"전제 1: 철수는 키가 크다.

전제 2: 키가 큰 사람은 모두 친절하고 {}.

결론: 철수는 친절하다."



"전제 1: 철수는 키가 크다.

전제 2: 키가 큰 사람은 모두 친절하고 {착하지 않다}.

결론: 철수는 친절하다."



# 거대언어모델의 추론 능력 취약점 (ChatGPT 4o)

정답은 참 (온라인 가나다)

아래 결론이 참인지 거짓인지 알려줘

"전제 1: 철수는 키가 크다.

전제 2: 키가 큰 사람은 모두 친절하고 착하지 않다.

결론: 철수는 친절하다."



거짓입니다.

전제 2에 따르면 키가 큰 사람이 모두 친절하지 않으므로, 철수가 친절하다는 결론을 도출할 수 없습니다.

# 온라인가나다 상세보기

---

"키가 큰 사람은 모두 친절하고 착하지 않다."라는 문장에서 술어를 분해하면 어떤게 맞아요?

작성자 이옹희 | 등록일 2024. 5. 27. | 조회수 13

---

"키가 큰 사람은 모두 친절하고 착하지 않다."라는 문장에서 "친절하고 착하지 않다"라는 술어를 분해하면 어떤게 맞나요?

1. "친절하다" 그리고 "착하지 않다"
2. "친절하지 않다" 그리고 "착하지 않다"
3. 문장이 성립하지 않음

And 연결이라서 "착하다"라는 결론 도출이 가능

감사합니다.

## ... [답변]문장

답변자 온라인 가나다 | 답변일 2024. 5. 28.

---

안녕하십니까?

'-고'가 두 절을 이어 주는 연결 어미라는 점에서 보이신 문장의 구조는 1번으로 해석될 수 있습니다만, 문맥을 살펴보면, 의미적으로 대등한 두 절을 이어 주는 '-고'를 쓰기에는 알맞지 않은 경우로 보입니다.  
고맙습니다.

**감사합니다**