# Chain of Code:
# Reasoning with a Language Model-Augmented Code Emulator

**Chengshu Li, Jacky Liang, Andy Zeng, Xinyun Chen, Karol Hausman, Dorsa Sadigh, Sergey Levine, Li Fei-Fei, Fei Xia†, Brian Ichter†**

**chengshu@stanford.edu, xiafei@google.com, brian@physicalintelligence.company**

**ICML 2024 Oral**

Google DeepMind $\pi$ Stanford Berkeley UNIVERSITY OF CALIFORNIA

# Large Language Models

## General reasoning capabilities


Figure adopted from Bommasani, Rishi, et al. "On the opportunities and risks of foundation models." arXiv preprint arXiv:2108.07258 (2021).

# LLM code-driven reasoning

## Precise computation

What is the area of a triangle with a base of pi and a height of e? Give your final answer in 10 decimal points.

Analyzed

python — Always show details — Copy code

```python
import math

# Given values
base = math.pi
height = math.e

# Area of the triangle
area = 0.5 * base * height
area = round(area, 10)

area
```

Result

```
4.2698671113
```

The area of a triangle with a base of π and a height of e is approximately 4.2698671113 square units. [>_]

Results from ChatGPT 4o, July 2024

# LLM code-driven reasoning

## Compositionality

Consider two triangles formed by (0, 0), (2, 5), (3, 4) and (1, 2), (4, 10), (3, 7) respectively. Which triangle has a larger area? Feel free to write python code to solve the problem if necessary.

### Self-define new functions

Analyzed

python — Always show details — Copy code

```python
import numpy as np

# Function to calculate area of triangle using vertices (x1, y1), (x2, y2), (x3, y3
def triangle_area(x1, y1, x2, y2, x3, y3):
    return abs(x1*(y2 - y3) + x2*(y3 - y1) + x3*(y1 - y2)) / 2

# Coordinates of the first triangle
triangle1_coords = [(0, 0), (2, 5), (3, 4)]

# Coordinates of the second triangle
triangle2_coords = [(1, 2), (4, 10), (3, 7)]

# Calculate the areas of the triangles
area_triangle1 = triangle_area(*triangle1_coords[0], *triangle1_coords[1], *triang
area_triangle2 = triangle_area(*triangle2_coords[0], *triangle2_coords[1], *triang

area_triangle1, area_triangle2
```

Result

```
(3.5, 0.5)
```

The area of the first triangle is 3.5 square units, while the area of the second triangle is 0.5 square units. Therefore, the first triangle has a larger area.

### Leverage existing libraries

Analyzed

python — Always show details — Copy code

```python
from sympy import Polygon, Point

# Define the points of the triangles
triangle1 = Polygon(Point(0, 0), Point(2, 5), Point(3, 4))
triangle2 = Polygon(Point(1, 2), Point(4, 10), Point(3, 7))

# Calculate the areas of the triangles
area1 = triangle1.area
area2 = triangle2.area

area1, area2
```

Result

```
(-7/2, -1/2)
```

The calculated areas are given as negative due to the order of points provided. The absolute values represent the actual areas.

- Area of Triangle 1: $|-\frac{7}{2}| = 3.5$
- Area of Triangle 2: $|-\frac{1}{2}| = 0.5$

Thus, Triangle 1 (formed by points (0, 0), (2, 5), and (3, 4)) has a larger area of 3.5 square units compared to Triangle 2's area of 0.5 square units.

Results from ChatGPT 4o, July 2024

# LLM code-driven reasoning

## Limitations

Only applies to algorithmic / mathematical tasks

**How many times do you detect sarcasm in the paragraph?**

**“Oh, isn't it delightful how AI advances so fast? One minute, you're deep in your Ph.D. studies, the next, everything you know is outdated. With ICML spewing out 3000 new papers a year, it's like trying to drink from a firehose. What should I do…?”**

```
num_sarcasm = 0
for sentence in paragraph:
    num_sarcasm += detect_sarcasm(sentence)
return num_sarcasm

def detect_sarcasm(sentence):
    🤷
```

# LLM code-driven reasoning

## Ideals

Only applies to algorithmic / mathematical tasks

**How many times do you detect sarcasm in the paragraph?**

**“Oh, isn't it delightful how AI advances so fast? One minute, you're deep in your Ph.D. studies, the next, everything you know is outdated. With ICML spewing out 3000 new papers a year, it's like trying to drink from a firehose. What should I do…?”**

```
num_sarcasm = 0
for sentence in paragraph:
    num_sarcasm += detect_sarcasm(sentence)
return num_sarcasm

def detect_sarcasm(sentence):
```

```
>>> detect_sarcasm(“Oh, isn't it delightful how AI advances so fast?”)
True
>>> detect_sarcasm(“What should I do…?”)
False
```

# Related Work

## Chain of Thought [1]

```
Q: Roger has 5 balls.  He buys
2 more packs, each with 3.  How
many balls does he have now?

Roger starts with 5 balls.
2 packs of 3 balls is 6.
5 + 6 = 11.

A: 11
```

## Program of Thoughts [2]

```
Q: Roger has 5 balls.  He buys
2 more packs, each with 3.  How
many balls does he have now?

num_balls = 5
num_balls += 2 * 3
answer = num_balls

A: 11
```

## ScratchPad [3]

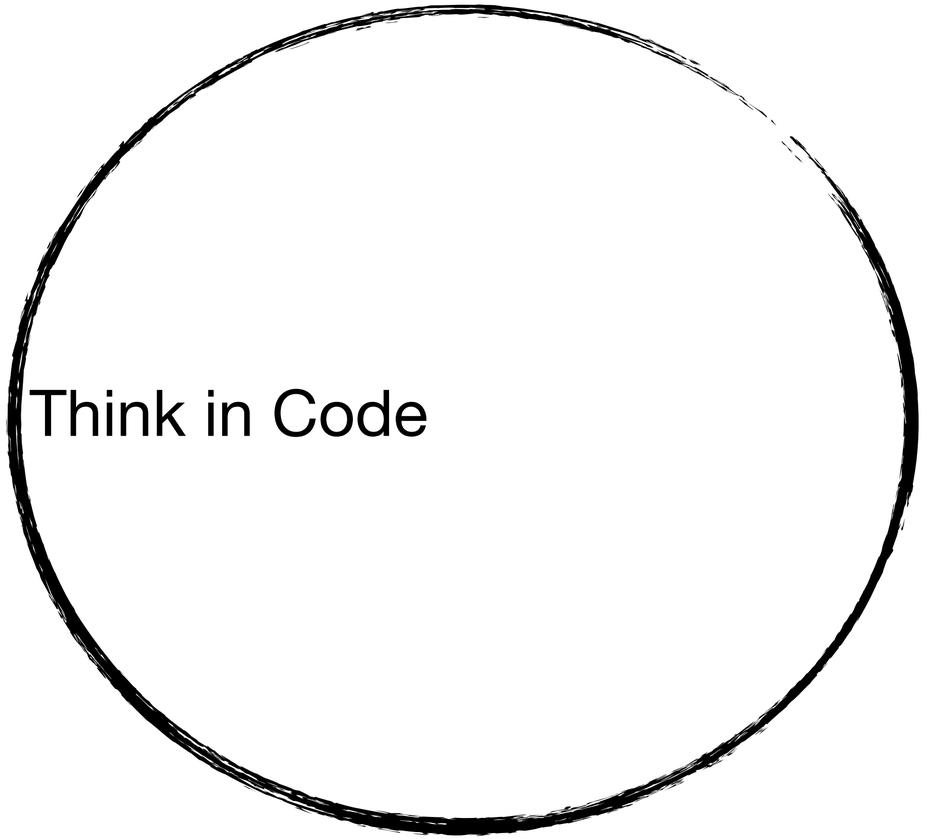
```
Q: Roger has 5 balls.  He buys 2 more packs,
each with 3.  How many balls does he have now?

num_balls = 5   state:  {num_balls = 5}
num_balls += 2 * 3   state:  {num_balls = 11}
answer = num_balls   state:  {answer = 11}

A: 11
```

Blue highlight indicates LM generation.
Red highlight indicates LM generated code being executed by an interpreter
Purple highlight indicates an LMulator (LM code emulator) simulating the code via a program state in green.


[1] Wei, Jason, et al. "Chain-of-thought prompting elicits reasoning in large language models." Advances in neural information processing systems 35 (2022): 24824-24837.
[2] Chen, Wenhu, et al. "Program of Thoughts Prompting: Disentangling Computation from Reasoning for Numerical Reasoning Tasks". *Transactions on Machine Learning Research* (2023).
[3] Nye, Maxwell, et al. "Show your work: Scratchpads for intermediate computation with language models." arXiv preprint arXiv:2112.00114 (2021).

# Key Insight

## Chain of Code = Think in Code + LMulators




Code provides a useful syntactic structure to reason through complex problems in a rigorous and scalable way.

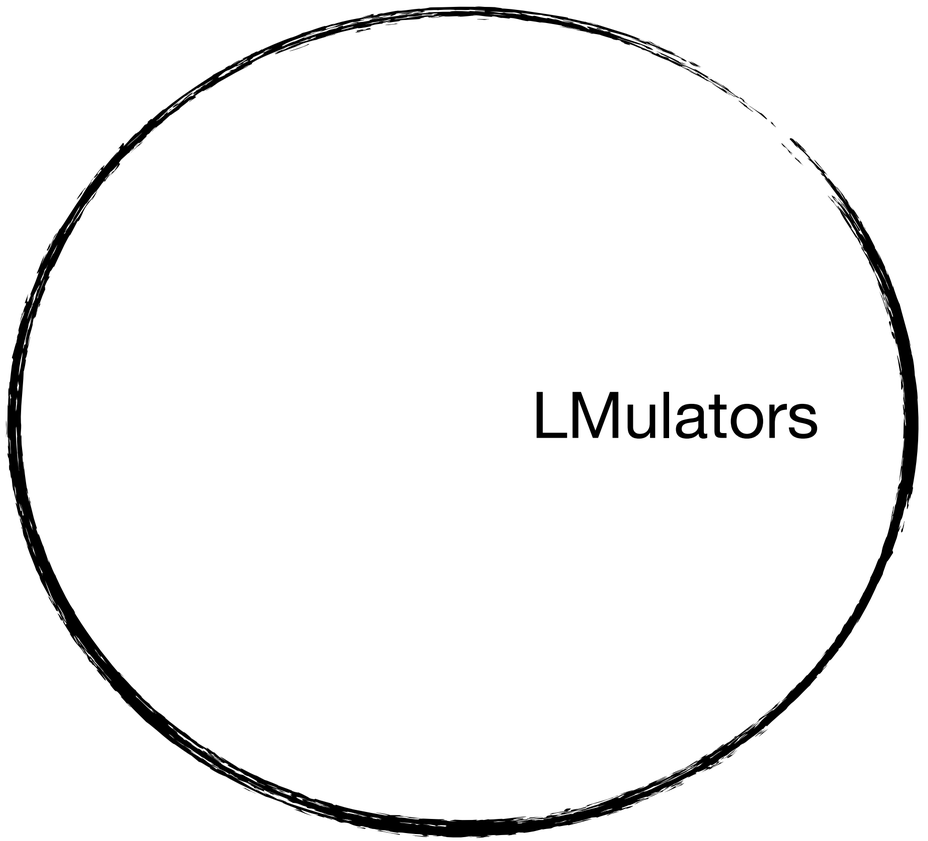
- Variables
- Control statements (e.g. if/else, for/while)
- Algorithms (e.g. search, sorting)
- Subproblem decomposition

# Key Insight

## Chain of Code = Think in Code + LMulators

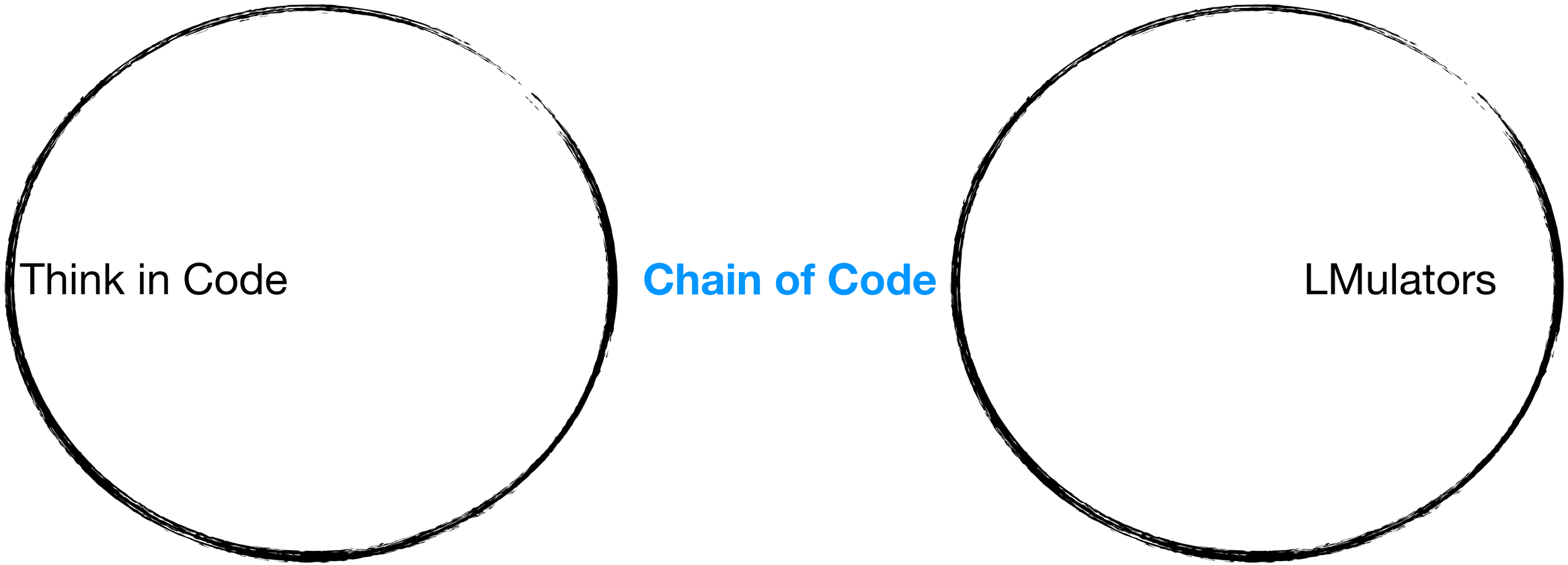
Language model code emulators (LMulators) can simulate code execution when the code interpreter fails to execute.

- Maintain and update program state directly
- Handle semantic subproblems (e.g. commonsense, linguistic, etc)



# Key Insight

## Chain of Code = Think in Code + LMulators



# Motivating Example

### Direct answer only

Q: How many countries have I been to? I've been to Mumbai, London, Washington, Grand Canyon, ...

**A: 32 (20%, ✗), 29 (10%, ✗), 52 (10%, ✓), ...**

### Chain of Thought

Q: Let's think step by step. How many countries have I been to? I've been to Mumbai, London, ...

We'll group by countries and count:
1. India: Mumbai, Delhi, Agra
2. UK: London, Dover, Edinburgh, Skye
3. USA: Washington, Grand Canyon, ...

**A: 61 (20%, ✗), 60 (20%, ✗), 52 (10%, ✓), ...**

### Chain of Code

Q: How many countries have I been to? I've been to Mumbai, London, Washington, Grand Canyon, Baltimore, ...

```
1    places, countries = ["Mumbai", ...], set()
2    for place in places:
3      country = get_country(place)
4      countries.add(country)
5    answer = len(countries)
```

**A:**

Blue highlight indicates LM generation.
Red highlight indicates LM generated code being executed by an interpreter.
Purple highlight indicates an LMulator (LM code emulator) simulating the code via a program state in green.

# Motivating Example

Chain of Code

Q: How many countries have I been to? I've been to Mumbai, London, Washington, Grand Canyon, Baltimore, ...

```
1    places, countries = ["Mumbai", ...], set()
2    for place in places:
3      country = get_country(place)
4      countries.add(country)
5    answer = len(countries)
```

**A:**

Blue highlight indicates LM generation.
Red highlight indicates LM generated code being executed by an interpreter.
Purple highlight indicates an LMulator (LM code emulator) simulating the code via a program state in green.

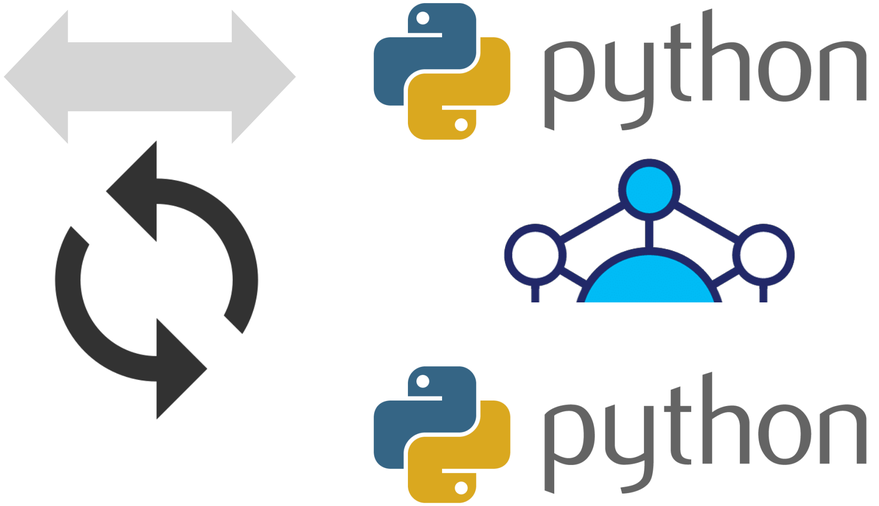
# Method

**Chain of Code**

Q: How many countries have I been to? I've been to Mumbai, London, Washington, Grand Canyon, Baltimore, ...

```
places, countries = ["Mumbai", ...], set()
delta state: {places = ['Mumbai', ...], countries = set()}

for place in places:
delta state: {place = 'Mumbai'}

    country = get_country(place)
delta state: {country = 'India')}

    countries.add(country)
delta state: {countries = {'India'}}

answer = len(countries)    delta state: {answer = 52}
```

**A: 52 (100%, ✓)**




Blue highlight indicates LM generation.
Red highlight indicates LM generated code being executed by an interpreter.
Purple highlight indicates an LMulator (LM code emulator) simulating the code via a program state in green.

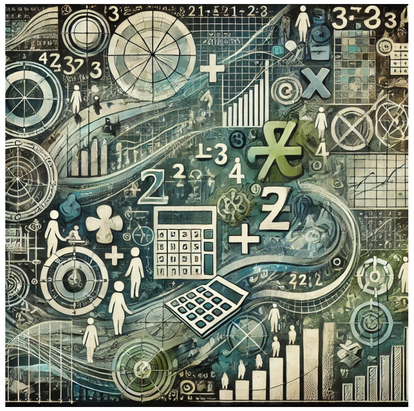
# Experimental Evaluation

## Dataset: BIG-Bench Hard [1]

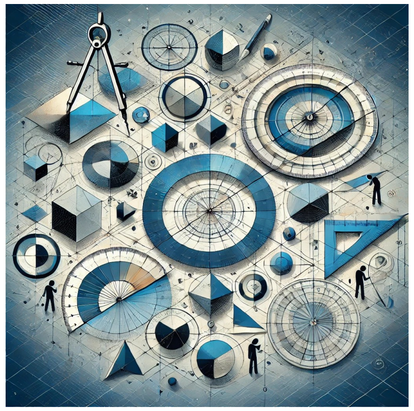
- Subset of the 23 most challenging tasks from BIG-Bench [2]
- Across diverse problem domains

Commonsense

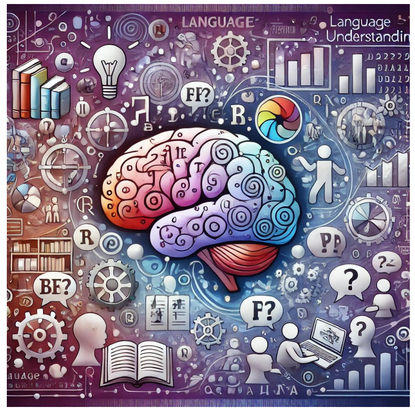

Logic



Arithmetic



Geometry



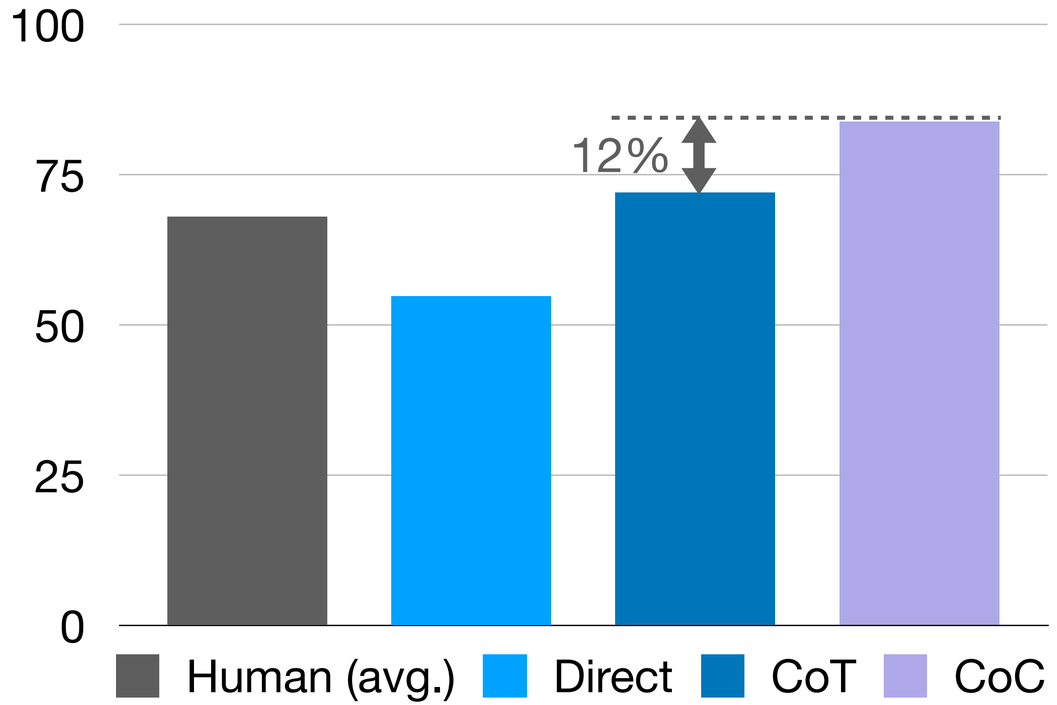

Language

[1] Suzgun, Mirac, et al. "Challenging big-bench tasks and whether chain-of-thought can solve them." Findings of the Association for Computational Linguistics (2023).
[2] Srivastava, Aarohi, et al. "Beyond the imitation game: Quantifying and extrapolating the capabilities of language models." Transactions on Machine Learning Research (2022).
Images on this slide are generated by DALL·E

# How well does CoC perform across a variety of tasks?

## Achieved sizable improvements over baselines and a new SOTA

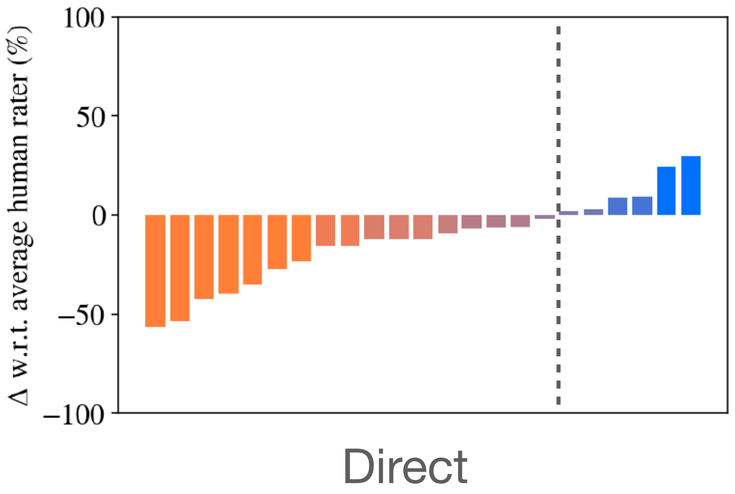
Chain of Code achieves **84**%, a gain of **12**% over Chain of Thought and a new **state of the art**.



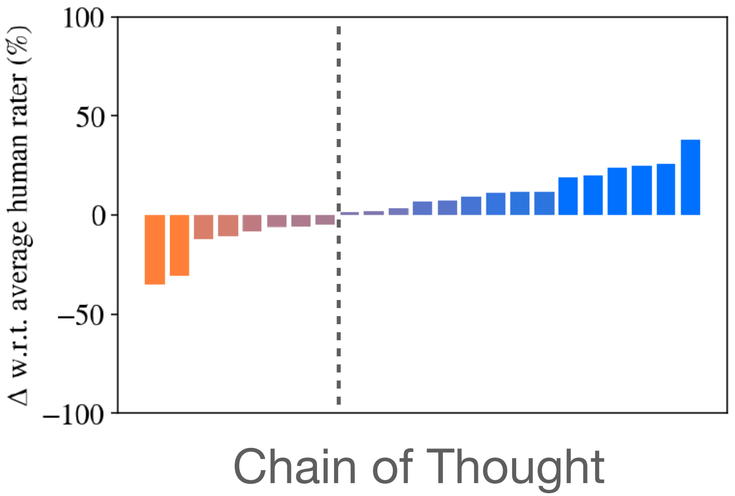
# How well does CoC perform across a variety of tasks?

## Outperforms humans for a vast majority of the tasks

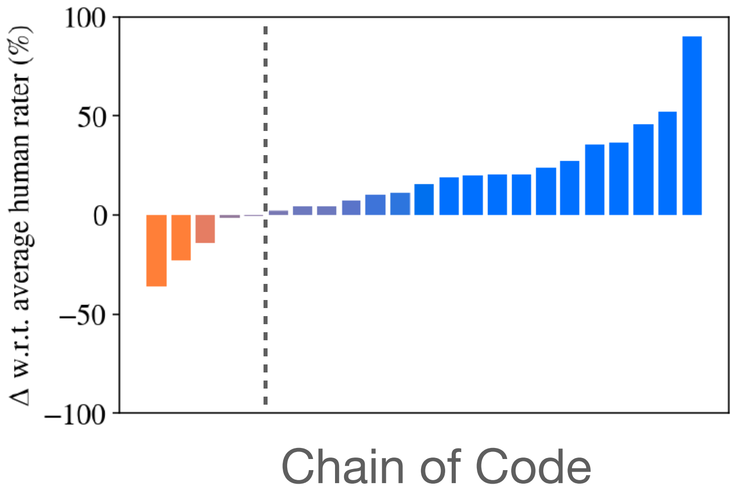
Chain of Code outperforms the **average human raters** in **18** out of 23 tasks.




Direct




Chain of Thought

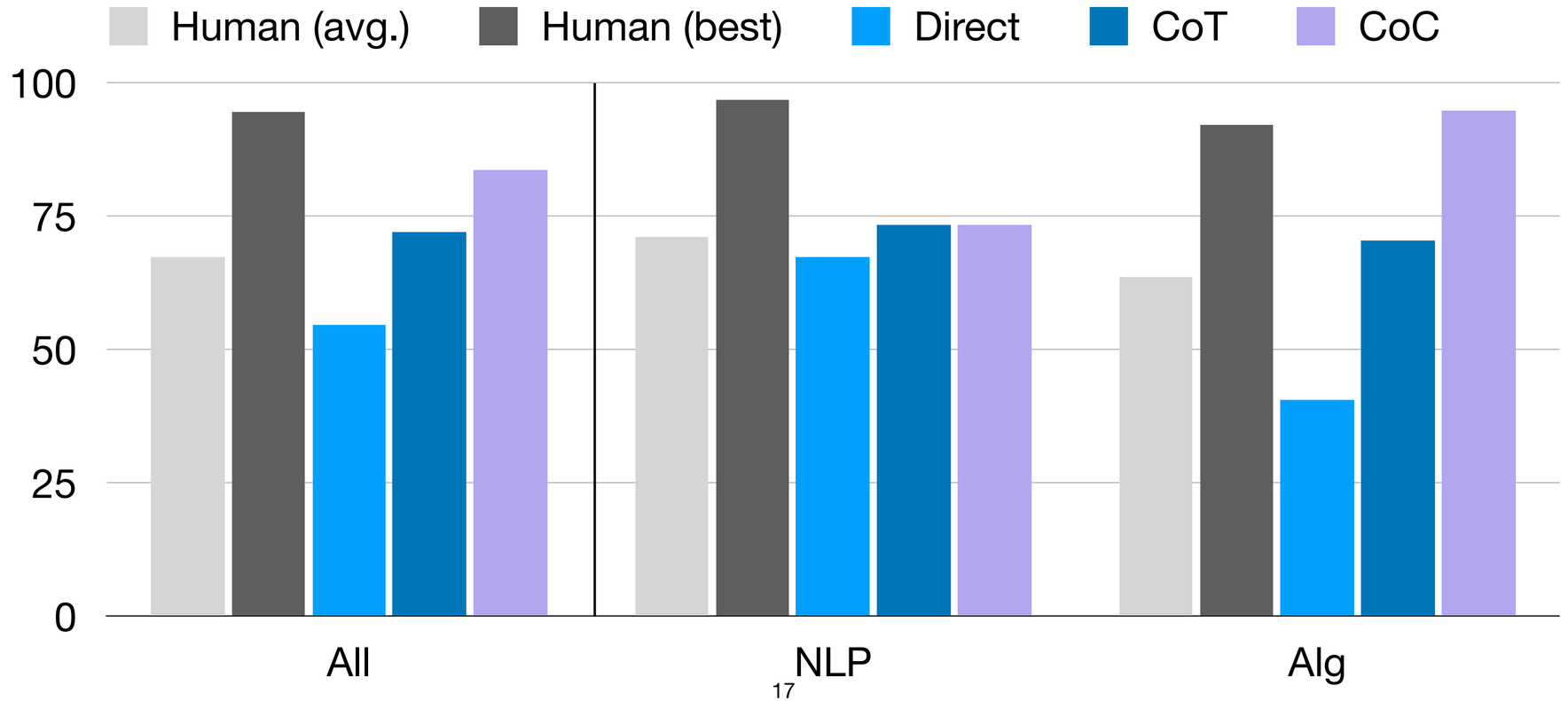



Chain of Code

# Which types of problems does CoC perform best?

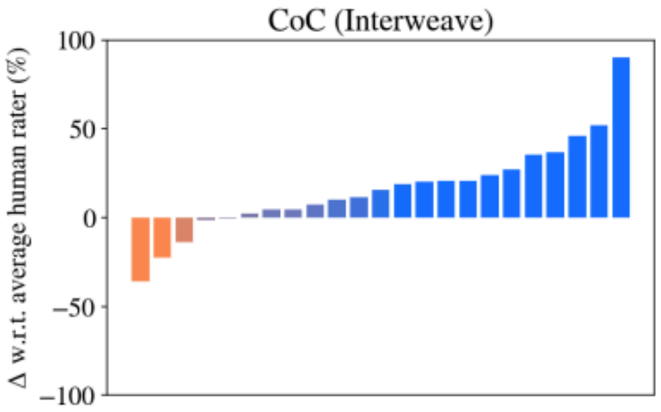
## Compare performance on the NLP and Algorithmic subset

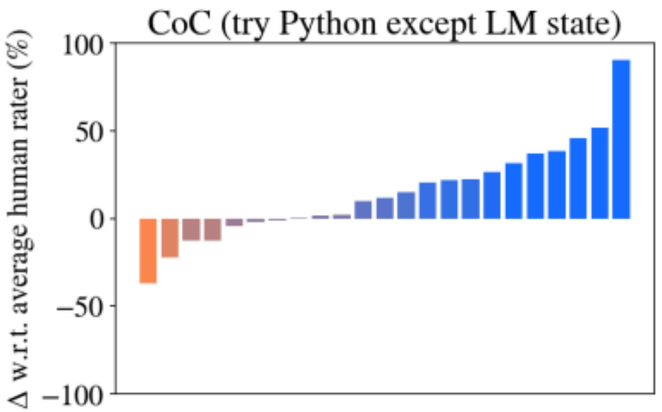
Chain of Code performs on par with Chain of Thought for the NLP subset, and **outperforms even the best human raters** for the algorithmic subset.

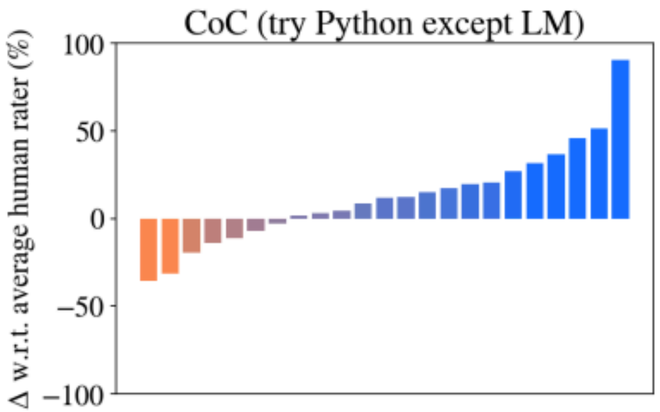


# How does each aspect of CoC affect performance?

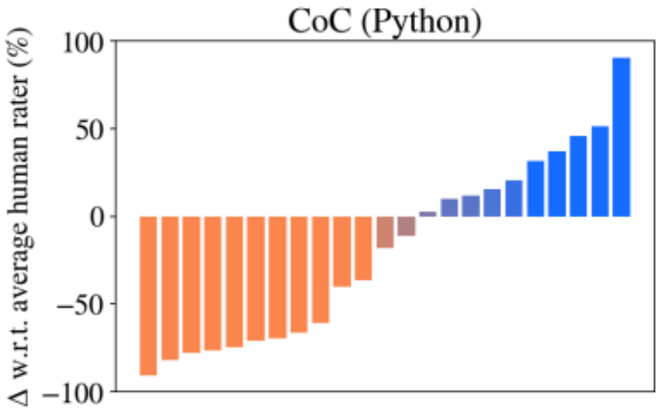
## Interweaving execution (full method) achieves the best results

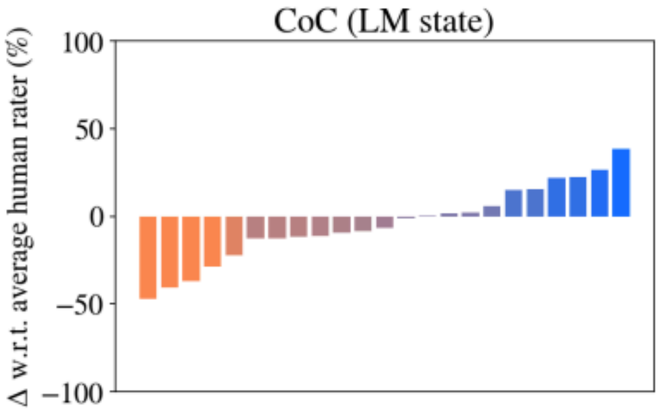


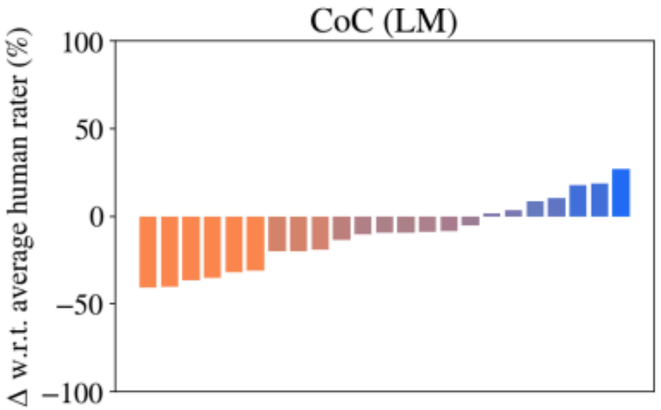




Similar to Program of Thoughts [1]



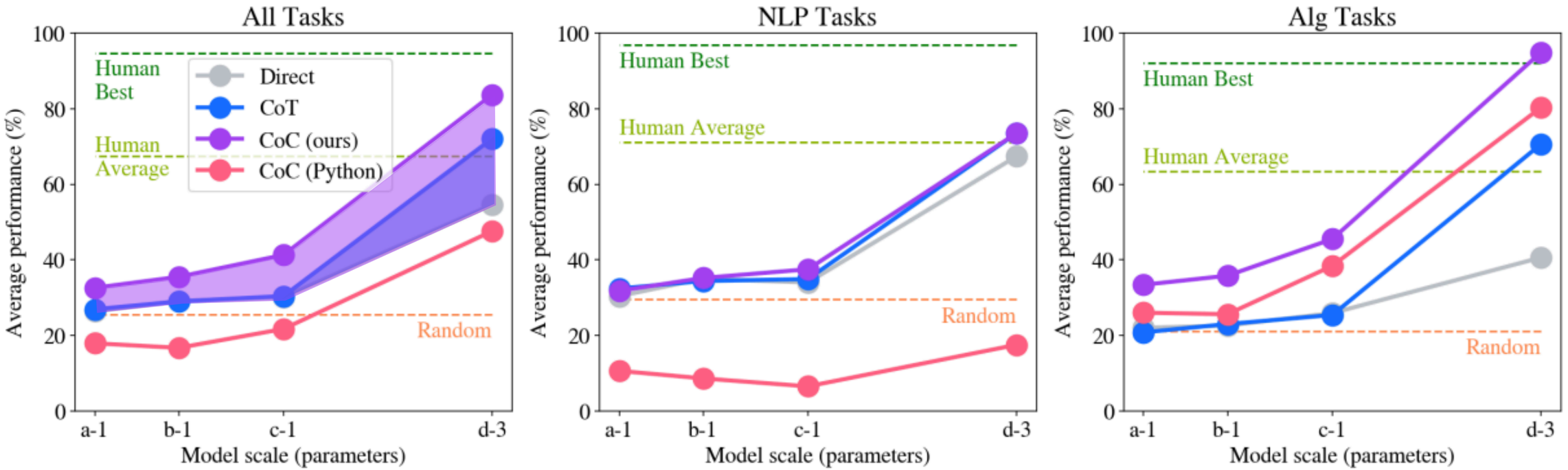

Similar to ScratchPad [2]




[1] Chen, Wenhu, et al. "Program of Thoughts Prompting: Disentangling Computation from Reasoning for Numerical Reasoning Tasks". *Transactions on Machine Learning Research* (2023).
[2] Nye, Maxwell, et al. "Show your work: Scratchpads for intermediate computation with language models." arXiv preprint arXiv:2112.00114 (2021).

# How does CoC scale with model size?

## Unlike CoT, CoC brings benefits even for smaller-sized models



# Can CoC be applied beyond language reasoning tasks?

## Robotics applications

Chain of Code is well fit for solving robotics tasks because

- They require both **semantic** and **algorithmic** reasoning.
- They involve interfacing with other APIs **through code** (e.g., control or perception APIs) and with users **through natural language**.

# Can CoC be applied beyond language reasoning tasks?

## Robotics applications

We evaluated Chain of Code on **seven** different tabletop robot manipulation tasks and observed very promising results!

- A prompt with a single example that informs the expected format
- Our method shows **strong generalization** capability

# Generalize to New Objects

## Sort the objects on the table into the compost bin and the recycle bin.

# Generalize to New Languages

**Prepare 西红柿炒蛋 (stir-fried tomato and eggs) in the pot.**

# Generalize to New Task Domains

## My steak is too bland. Can you help?

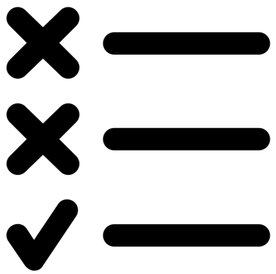
# Additional Results and Analysis

**Come to our poster #2809!**

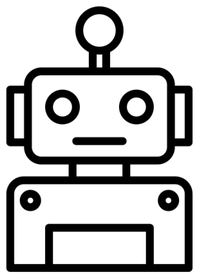
Qualitative Results

Instruction-tuned Models

Robustness



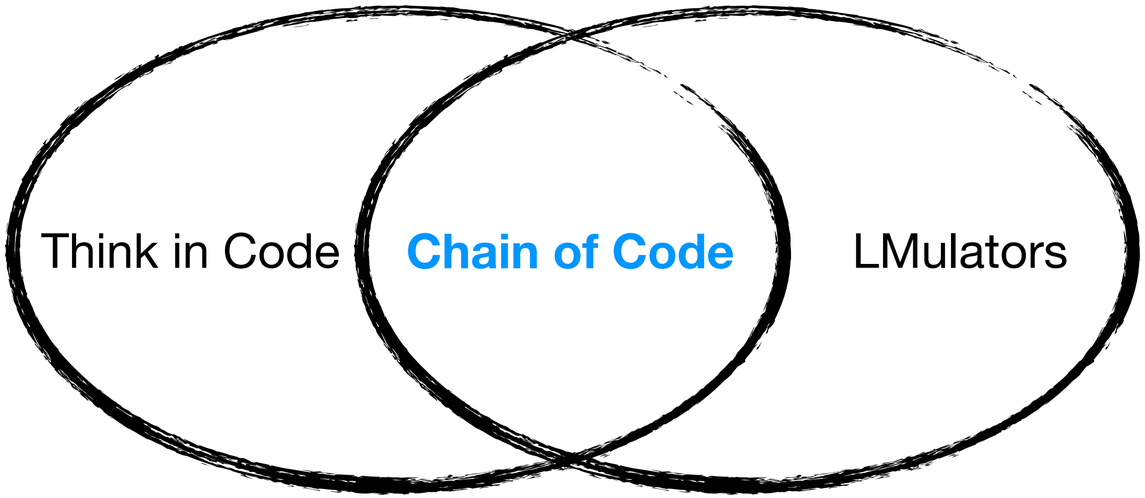
Cross-task Prompting



Robot Experiment Details

# Chain of Code:

## Reasoning with a Language Model-Augmented Code Emulator



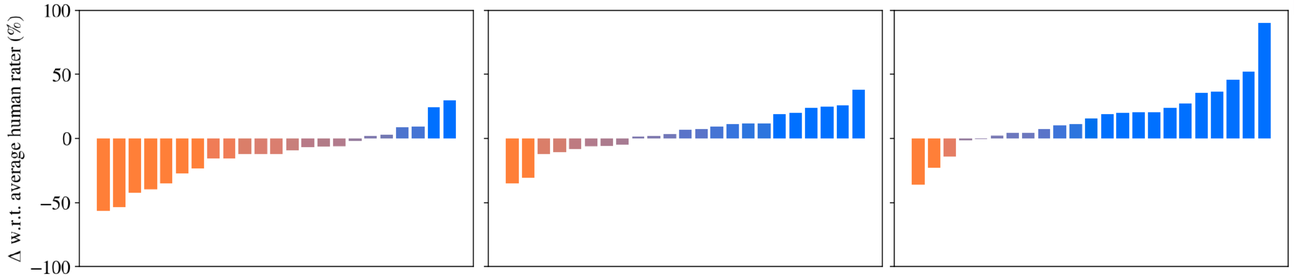

## Method

Chain of Code

```
Q: How many countries have I been to?  I've been to Mumbai,
London, Washington, Grand Canyon, Baltimore, ...
1    places, countries = ["Mumbai", ...], set()
2    for place in places:
3      country = get_country(place)
4      countries.add(country)
5    answer = len(countries)
A:
```

Blue highlight indicates LM generation.
Red highlight indicates LM generated code being executed by an interpreter.
Purple highlight indicates an LMulator (LM code emulator) simulating the code via a program state in green.

13




Poster at Hall C 4-9 **#2809**
Wed 11:30 a.m. - 1 p.m.
chain-of-code.github.io