Executing one’s way out of the Chinese room

GI@ICSE2024

Shin Yoo | COINSE@KAIST | 2024.04.16

Who am I?
Shin Yoo

- Associate Prof@KAIST
- Leads Computational Intelligence for Software Engineering Group (https://coinse.github.io)
- COINSE focuses on: testing & debugging, AI4SE & SE4AI, SBSE (including GI 😎)

漢

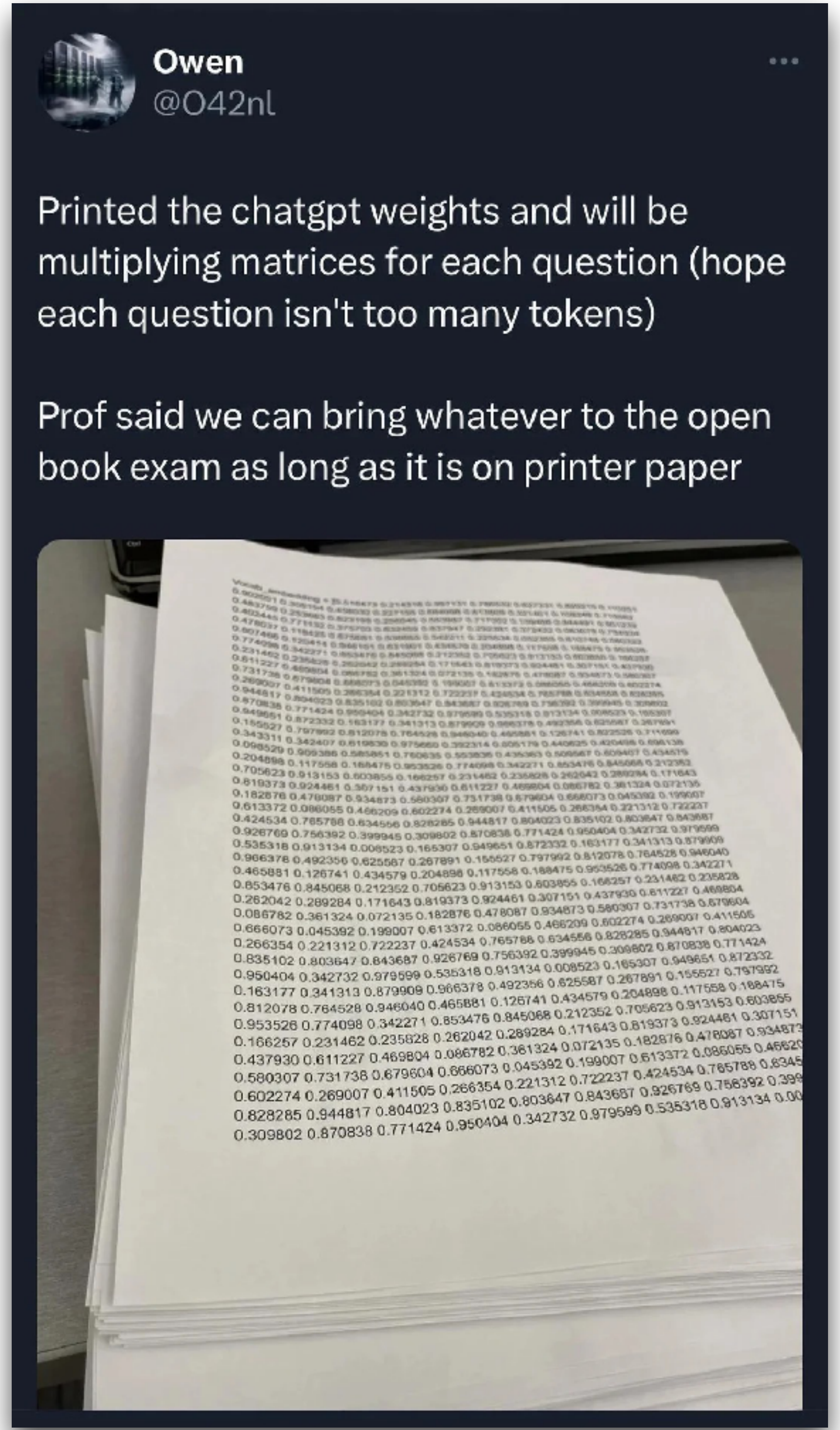
The Chinese Room

A Thought Experiment

John Searle, “Mind, Brains, and Programs” in 1980

- Suppose we have a computer program that behaves as if it understands Chinese language.
- You are in a closed room with the AI program source code.
- Someone passes a paper with Chinese characters written on it, into the room.
- You use the source code as instruction to generate the response to the input, and sends the response out of the room.
- Do you understand Chinese language, or not?

“And we’re talking about this because…”

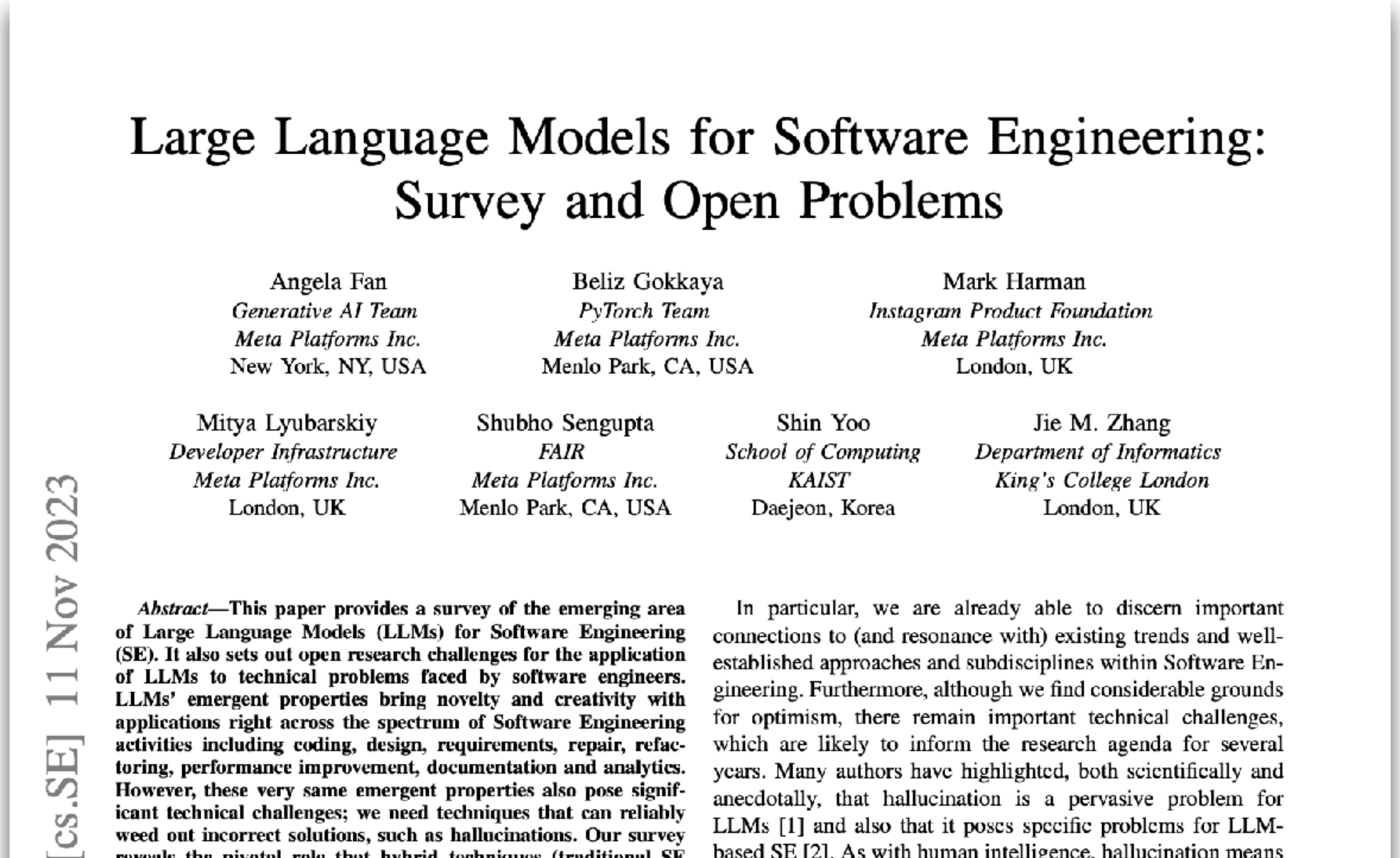


(Obviously we are all a bit like this now)

Landscape

Survey of the Explosion 💥

ICSE 2023 Future of SE Track (https://arxiv.org/abs/2310.03533)



[cs.SE] 11 Nov 2023

Large Language Models for Software Engineering: Survey and Open Problems

Angela Fan
Generative AI Team
Meta Platforms Inc.
New York, NY, USA

Beliz Gokkaya
PyTorch Team
Meta Platforms Inc.
Menlo Park, CA, USA

Mark Harman
Instagram Product Foundation
Meta Platforms Inc.
London, UK

Mitya Lyubarskiy
Developer Infrastructure
Meta Platforms Inc.
London, UK

Shubho Sengupta
FAIR
Meta Platforms Inc.
Menlo Park, CA, USA

Shin Yoo
School of Computing
KAIST
Daejeon, Korea

Jie M. Zhang
Department of Informatics
King's College London
London, UK

***Abstract*—This paper provides a survey of the emerging area of Large Language Models (LLMs) for Software Engineering (SE). It also sets out open research challenges for the application of LLMs to technical problems faced by software engineers. LLMs' emergent properties bring novelty and creativity with applications right across the spectrum of Software Engineering activities including coding, design, requirements, repair, refactoring, performance improvement, documentation and analytics. However, these very same emergent properties also pose significant technical challenges; we need techniques that can reliably weed out incorrect solutions, such as hallucinations. Our survey reveals the pivotal role that hybrid techniques (traditional SE**

In particular, we are already able to discern important connections to (and resonance with) existing trends and well-established approaches and subdisciplines within Software Engineering. Furthermore, although we find considerable grounds for optimism, there remain important technical challenges, which are likely to inform the research agenda for several years. Many authors have highlighted, both scientifically and anecdotally, that hallucination is a pervasive problem for LLMs [1] and also that it poses specific problems for LLM-based SE [2]. As with human intelligence, hallucination means

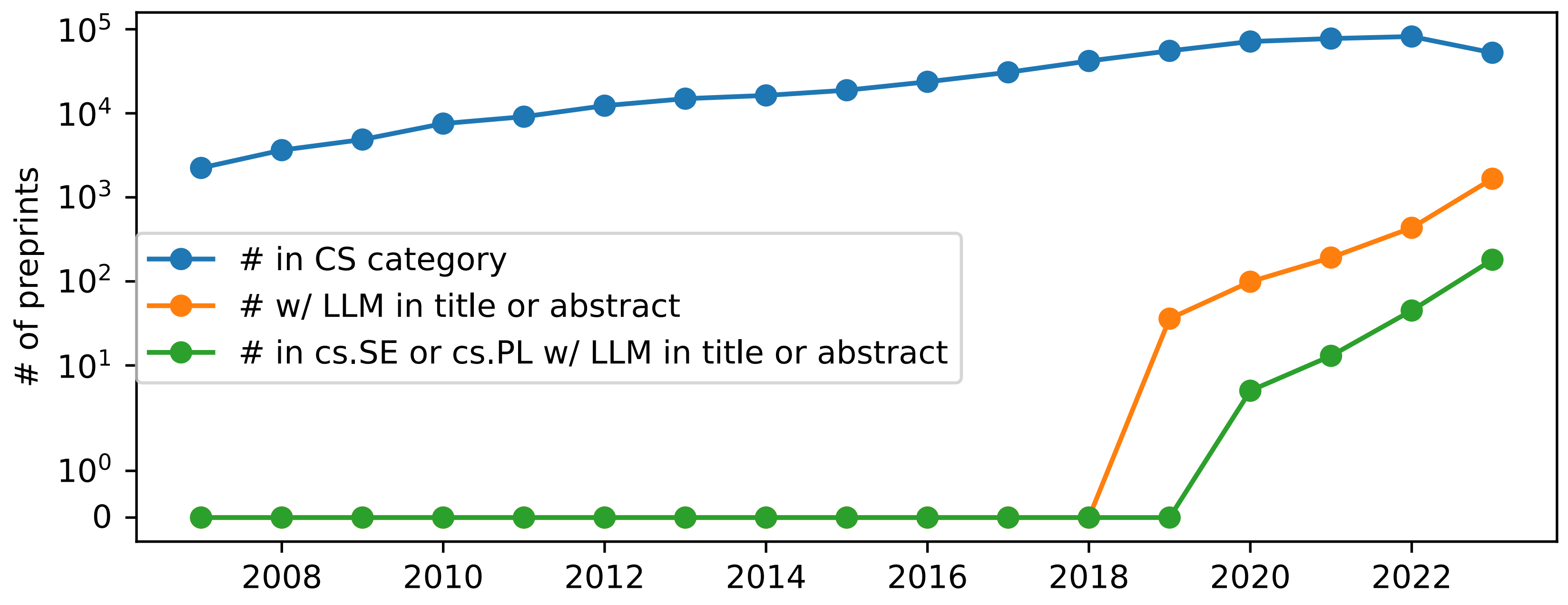


Fig. 2. Trends in number of arXiv preprints. The blue line denotes the number of preprints categorised under "CS". The orange line denotes the number of preprints in AI (cs.AI), Machine Learning (cs.LG), Neural and Evolutionary Computing (cs.NE), Software Engineering (cs.SE), and Programming Language (cs.PL) whose title or abstract contains either "Large Language Model", "LLM", or "GPT". The green line denotes the number of preprints in SE and PL categories whose title or abstract contains either "Large Language Model", "LLM", or "GPT"

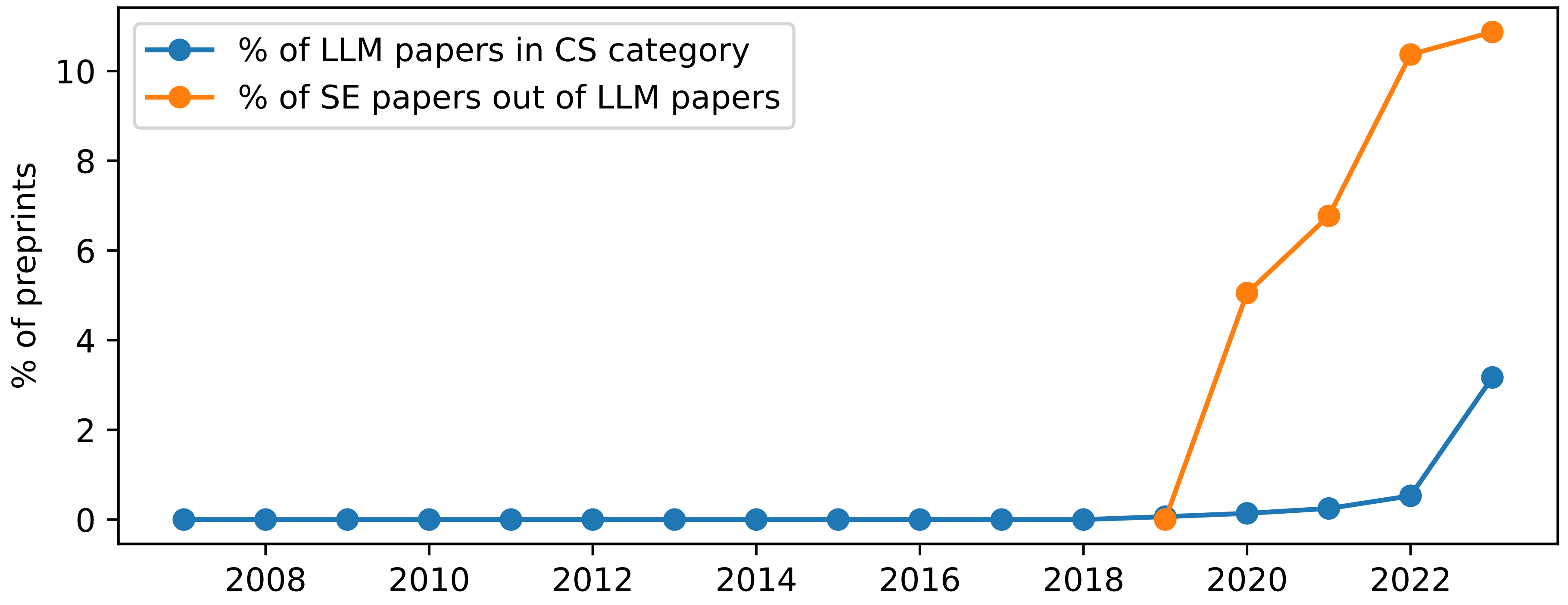


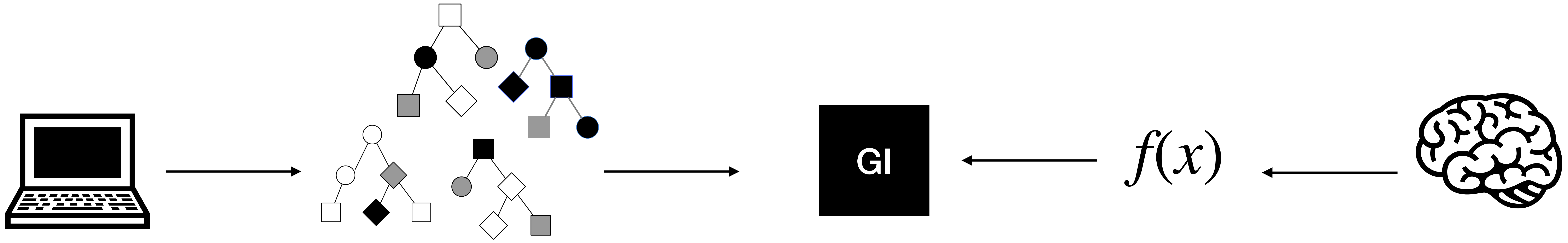
Fig. 3. Proportions of LLM papers and SE papers about LLMs. By "about LLMs", we mean that either the title or the abstract of a preprint contains "LLM", "Large Language Model", or "GPT". The blue line denotes the percentage of the number of preprints about LLMs out of the number of all preprints in the CS category. The orange line denotes the percentage of the number of preprints about LLMs in cs.SE and cs.PL categories out of all preprints about LLMs

https://arxiv.org/abs/2310.03533

“But why are LLMs <u>**so popular**</u> among SE researchers…?”

Correlation vs. Causation, or Syntax vs. Semantic

- MIP talk at ICSE 2019 captured this beautifully - *“It Does What You Say, Not What You Mean: Lessons from 10 Years of Program Repair”*
- Traditionally, computing the semantic has been either very difficult or infeasible; as it is well known to the GI community!



Candidate solutions, (randomly) generated via **syntactic** perturbations

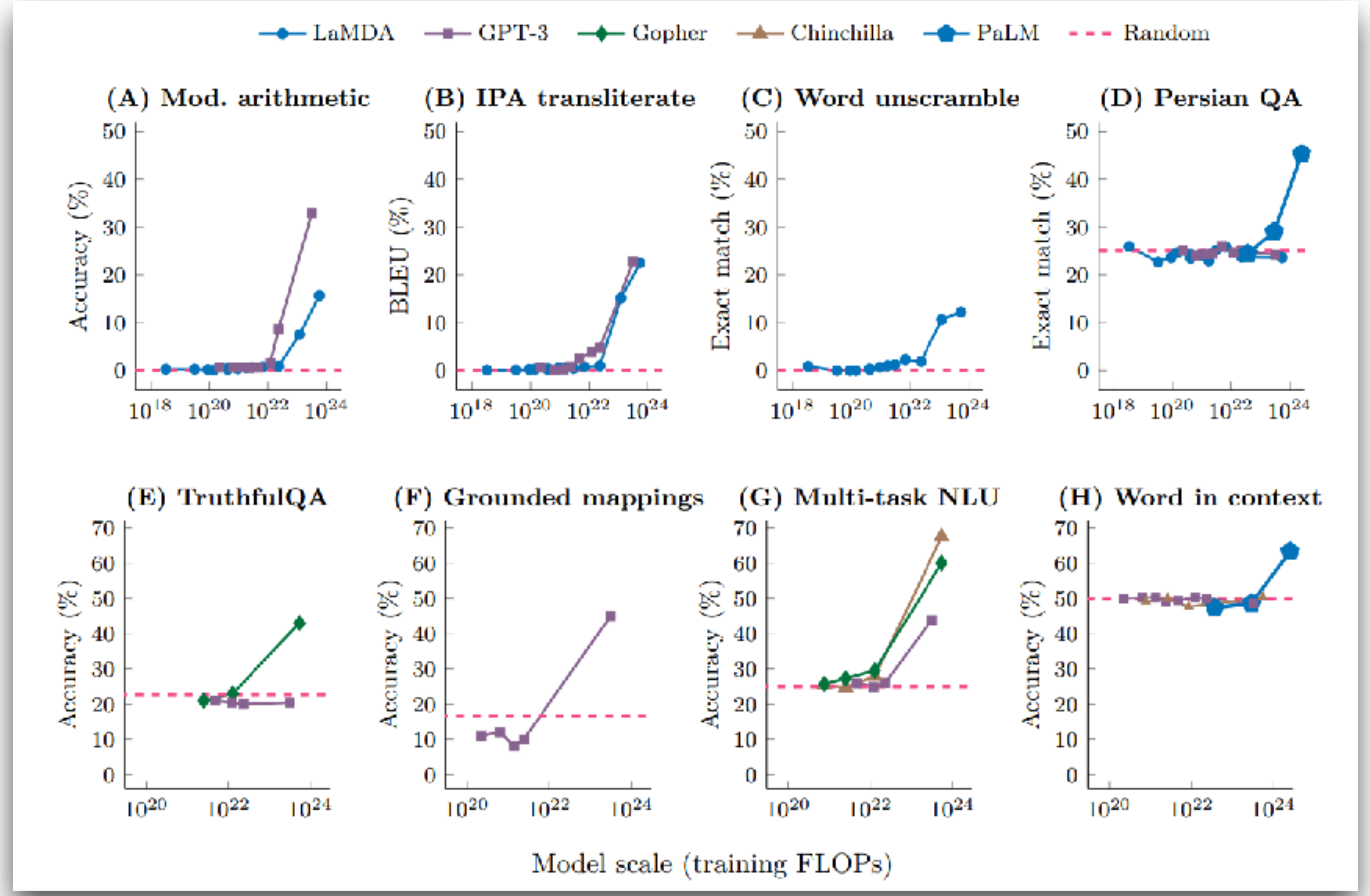
Semantic, captured (imperfectly) in fitness functions

Large Language Model
(really, a very large statistical language model)

- Mainly Transformer-based DNNs that are trained to be an auto-regressive language model, i.e., given a sequence of tokens, it repeatedly tries to predict the next token.
- The biggest hype in SE research right now with an **explosive** growth, because:
  - **Emergent behaviour** leading to very attractive properties such as in-context learning, Chain-of-Thoughts, or PAL
  - They **seem to** get the **semantics** of the code and **work across natural and programming language**

What is an Emergent Behavior?

- Above certain size, LLMs change their behavior in interesting ways
- The point of change in slope is referred to as “breaks”



Caballero et al., https://arxiv.org/abs/2210.14891

Chain-of-Thoughts

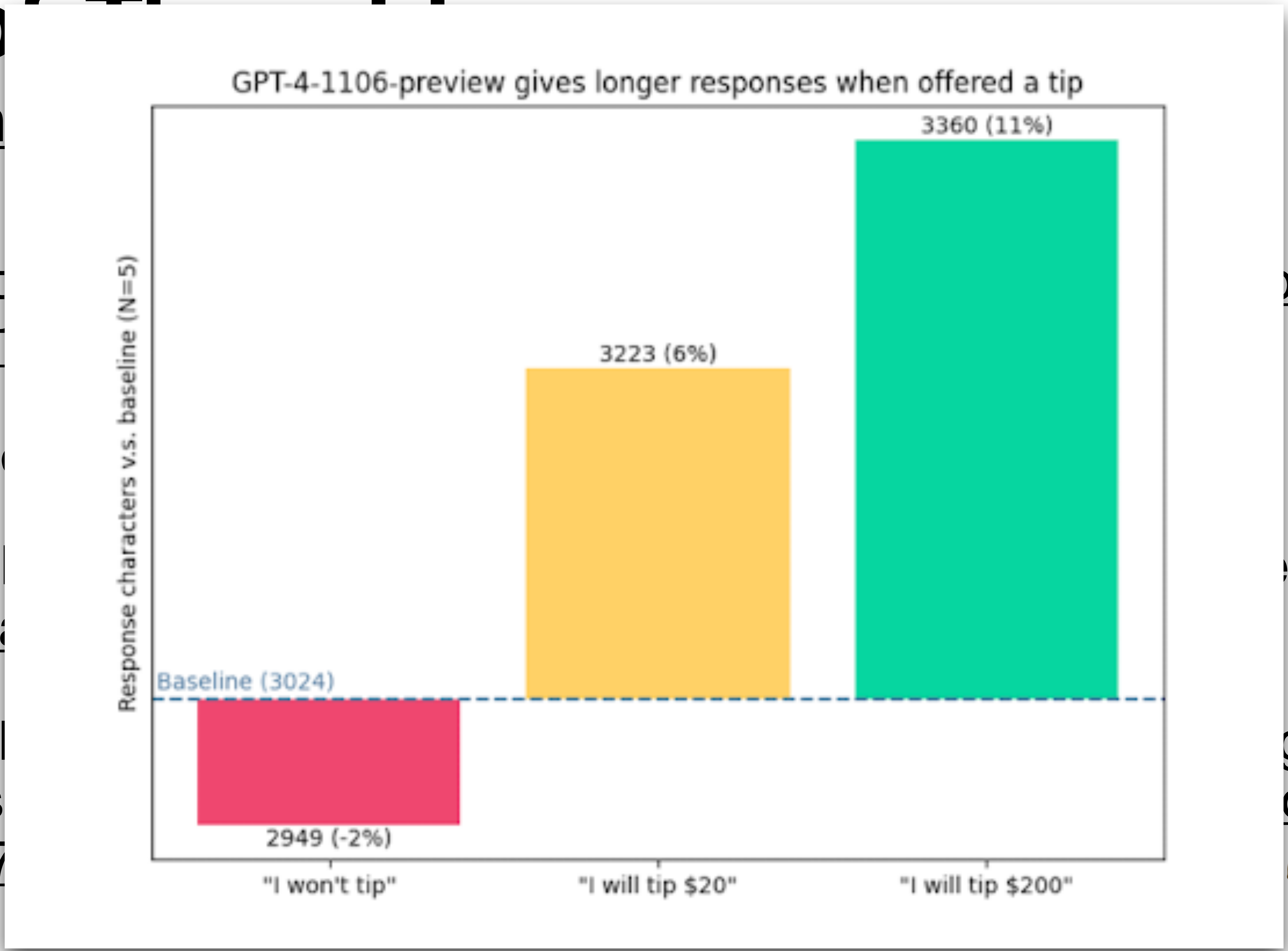
Wei et al., https://arxiv.org/abs/2201.11903

- Underneath, LLMs are doing autocompletion, not any other type of reasoning: they appear to be capable of rational inference because the corpus they are trained with includes traces of logical reasoning.
- So, **conditioning** the model (with the context) to be more precise about the reasoning steps can result in generation of more accurate reasoning steps.
    - Add “Let’s think in step by step” at the end of every prompt (https://arxiv.org/abs/2205.11916) 🙃 🫥 🫠

Chain-o

Wei et al., h

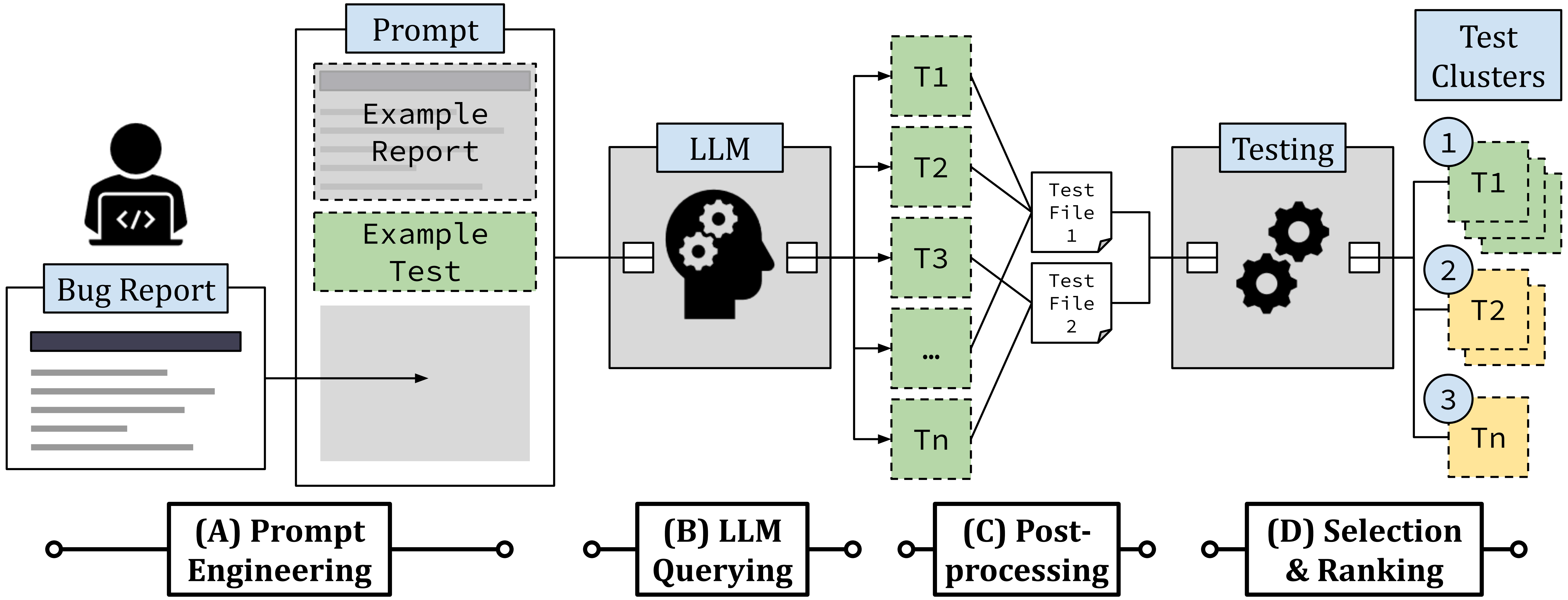
- Add "Let's t ps://arxiv.org/
  abs/2205.1
- We have ev
  - If you mal s (https://
    arxiv.org/a
  - Apparentl ge tip
    produces oogel/status/
    17307267 💰



“Okay, it talks like a human and can answer some questions. But why SE?”

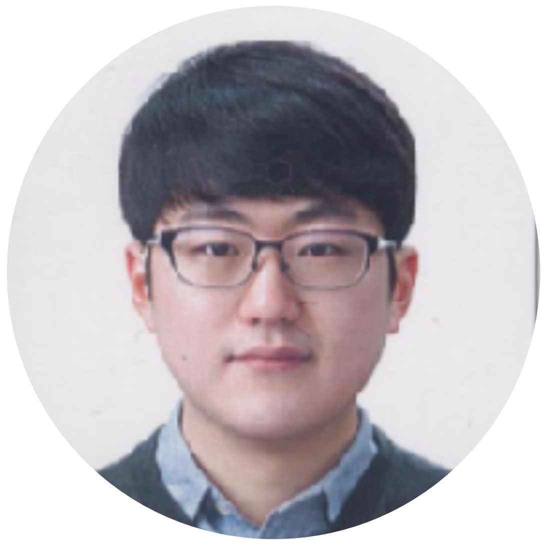
LLMs seemingly handle semantics across NL/PL barrier

LLM-based Bug Reproduction (Kang, Yoon & Yoo, ICSE 2023)



Title	**assertContainsIgnoringCase fails to compare i and I in tr_TR locale**
	See org.assertj.core.internal.Strings#assertContainsIgnoringCase [url] I would suggest adding [url] verification to just ban toLowerCase(), toUpperCase() and other unsafe methods: #2664

```java
public void testIssue952(){
  Locale locale = new Locale("tr", "TR");
  Locale.setDefault(locale);
  assertThat("I").as("Checking in tr_TR locale")
  .containsIgnoringCase("i");
}
```

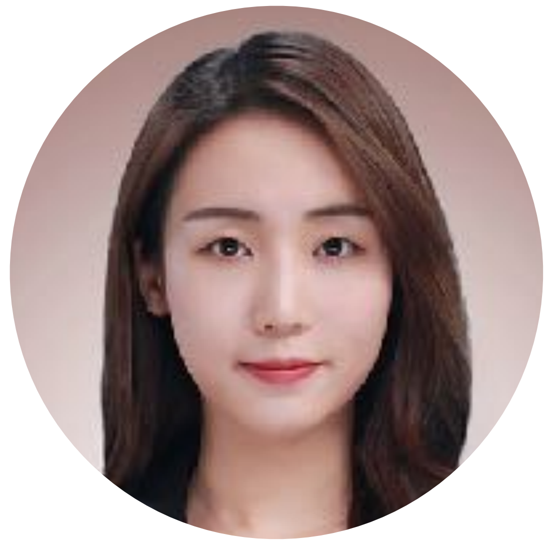


Sungmin Kang
(PhD Candidate)

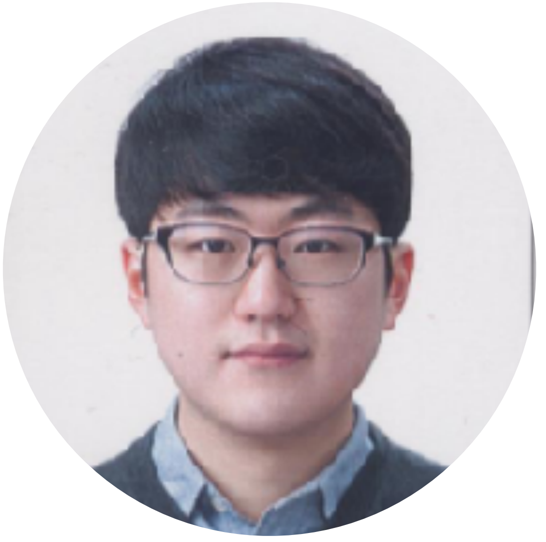
Juyeon Yoon
(PhD Candidate)

AutoFL: LLM based FL

Kang, An & Yoo (https://arxiv.org/abs/2308.05487)



Gabin An
(PhD Candidate)



Sungmin Kang
(PhD Candidate)

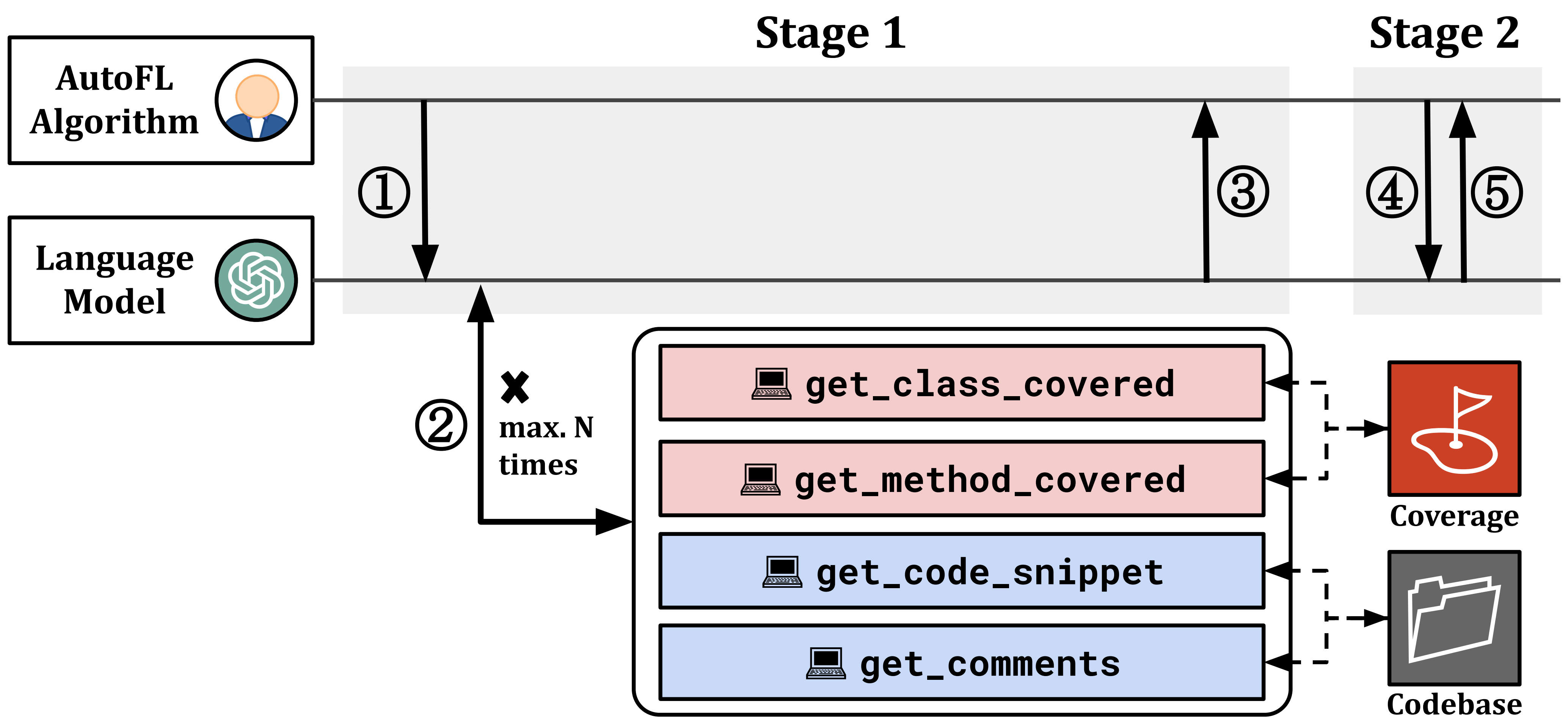


Figure 1: Diagram of AutoFL. Each arrow represents a prompt / response between components, with the circled numbers indicating the order of interactions. Function invocations are made at most N times, where N is a predetermined parameter of AutoFL.

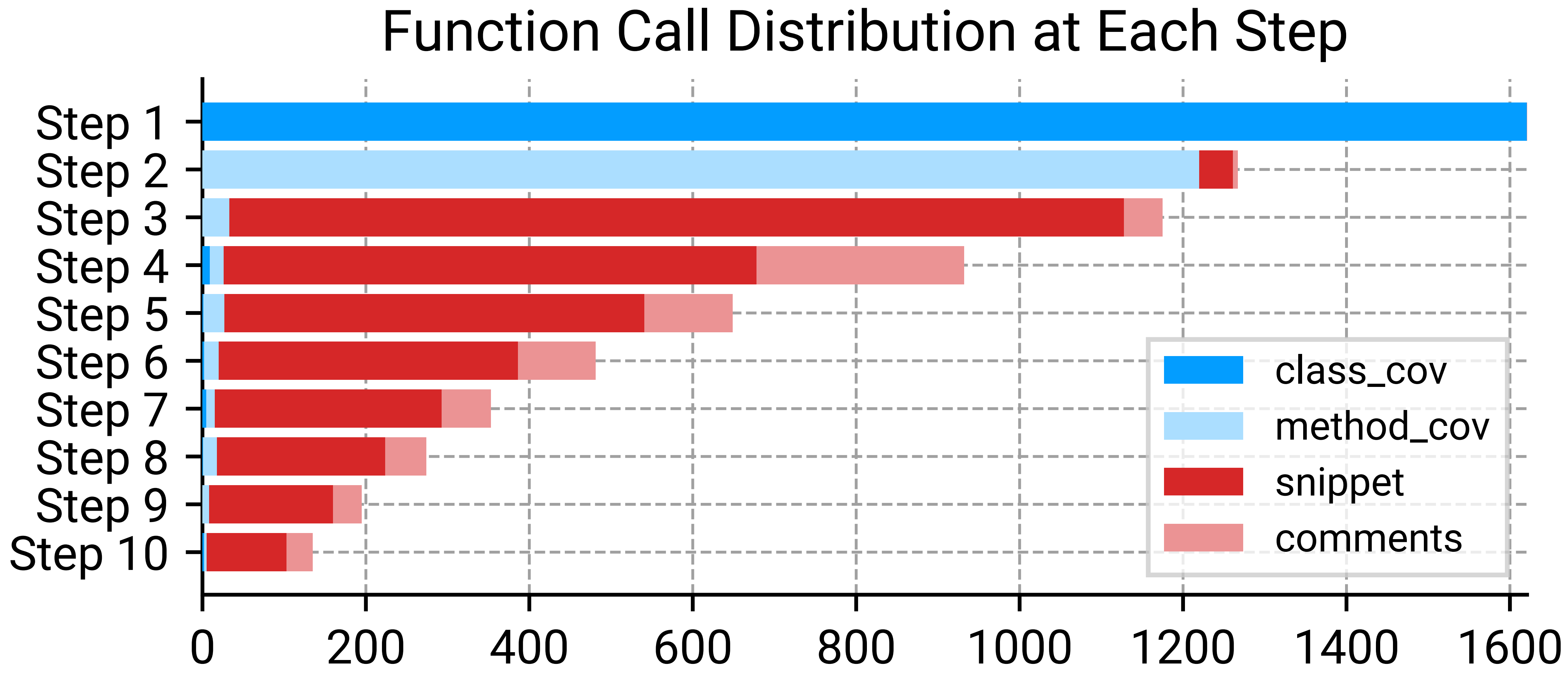
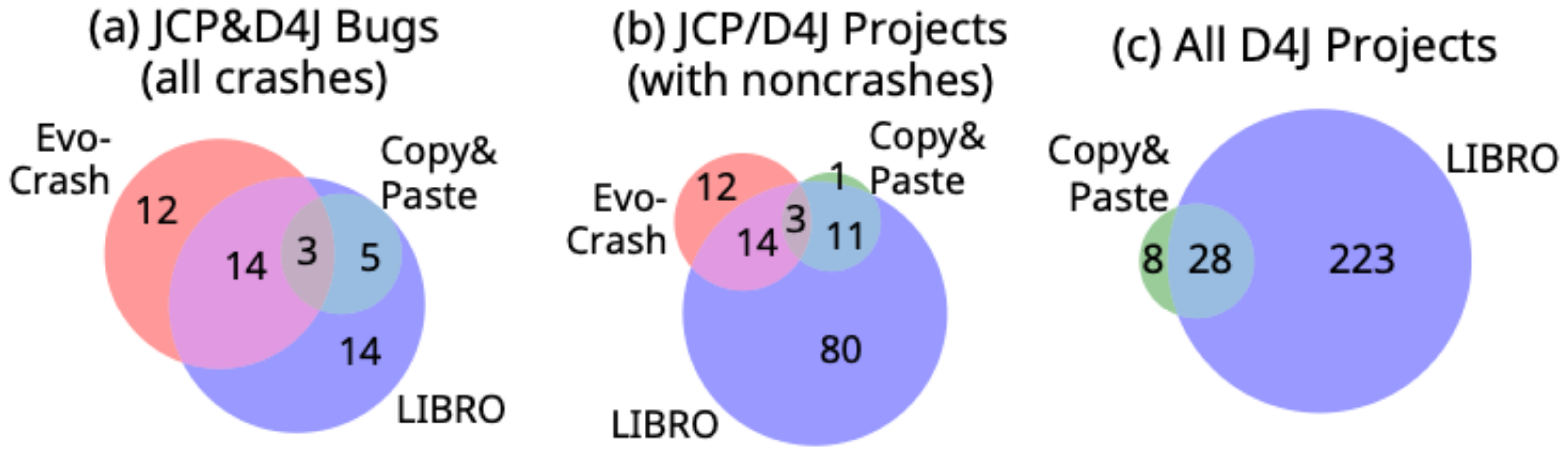


Figure 5: Function call frequency by step over all five runs of AutoFL. The total length at each step decreases as AutoFL can stop calling functions at any step; e.g. about 400 AutoFL processes stopped calling functions after the first step.



Family	Technique	acc@1	acc@3	acc@5
	Predicate Switching	42	99	121
	Stack Trace	57	108	130
	Slicing (frequency)	51	96	119
MBFL	MUSE	73	139	161
	Metallaxis	106	162	191
SBFL	Ochiai	122	192	**218**
	DStar	125	**195**	216
	SBFL-F	34	66	78
LLM-Based	LLM+Test	81	94	97
	AutoFL	**149**	180	194

Libro Reproduction Results
(against of 750 Bugs)

AutoFL Evaluation Metric
(against of 353 Bugs)

“Sounds like LLMs will solve all SE problems. Can we go home now?”

Hallucination 🌀

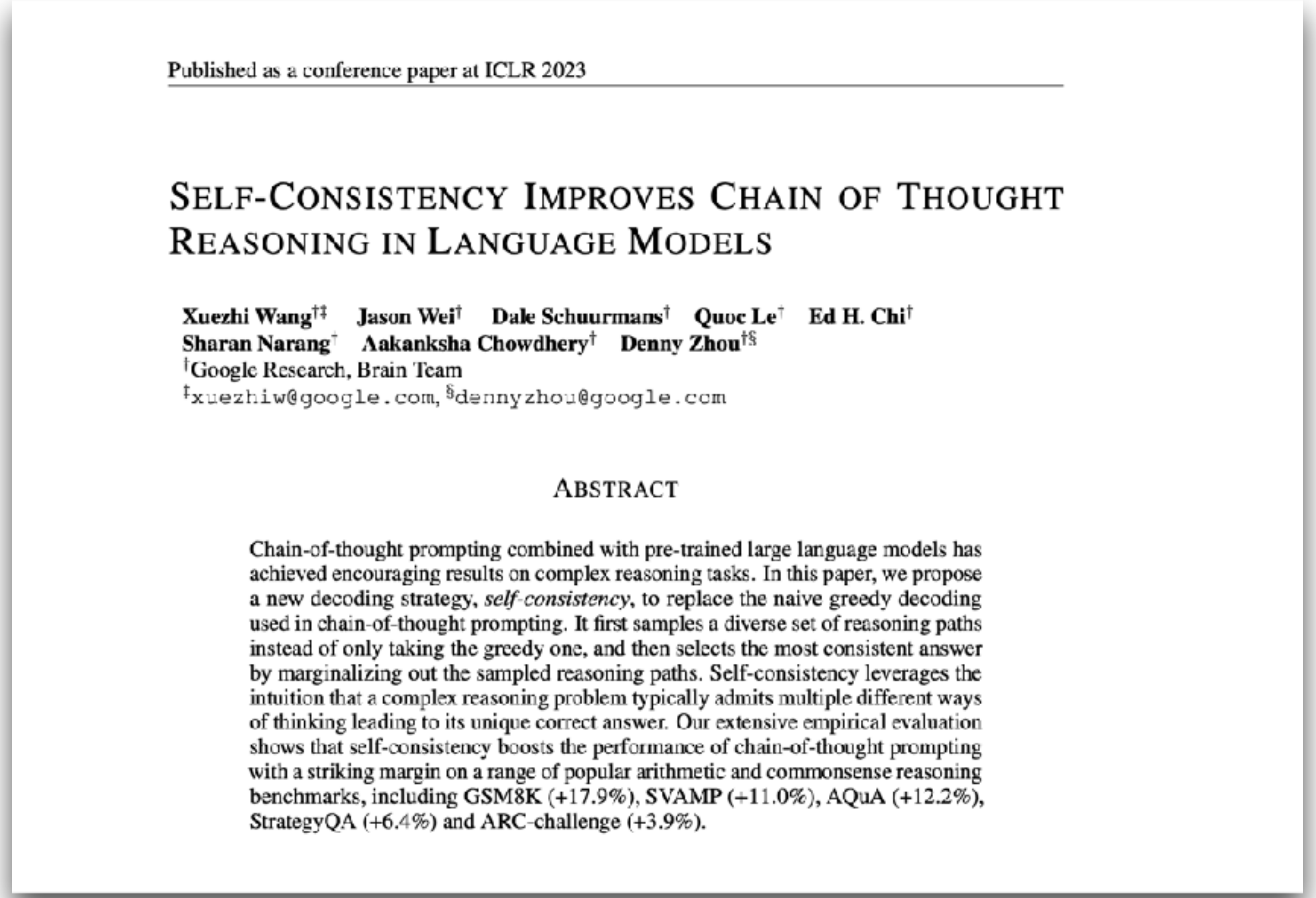
- LLM = (Statistical) Autocompletion = completion **not because it is the right choice, but because it is the most likely choice.**

- This will affect the accuracy of LLM outputs, to the extent that it fabricates incorrect/non-factual solutions and responses.

Self-Consistency

Wang et al., ICLR 2023

- When sampling answers from an LLM, take multiple answers with high temperature.
- If there is an answer that has the majority among the sampled answers, it is more likely to be the correct one.

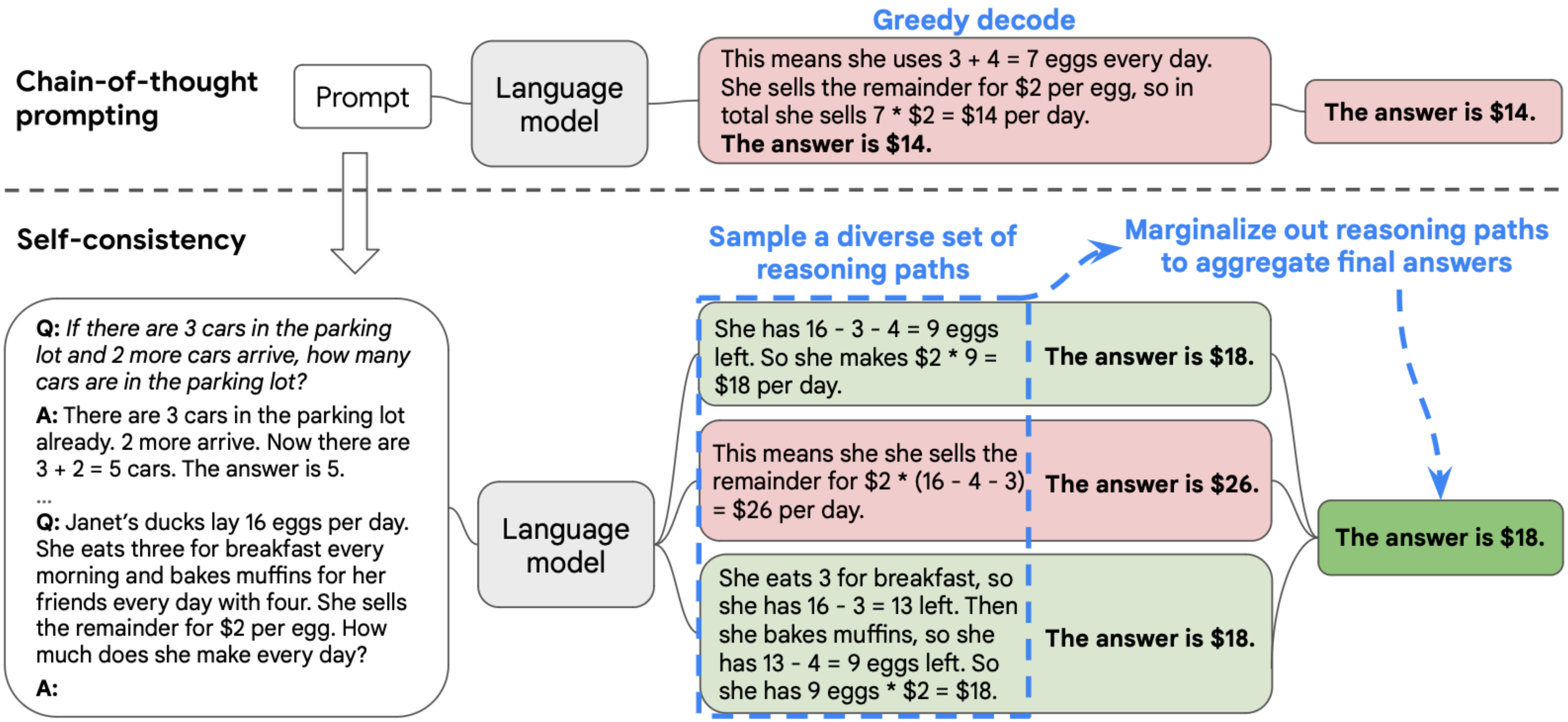

Published as a conference paper at ICLR 2023

SELF-CONSISTENCY IMPROVES CHAIN OF THOUGHT REASONING IN LANGUAGE MODELS

Xuezhi Wang[†‡] Jason Wei[†] Dale Schuurmans[†] Quoc Le[†] Ed H. Chi[†]
Sharan Narang[†] Aakanksha Chowdhery[†] Denny Zhou[†§]
[†]Google Research, Brain Team
[‡]xuezhiw@google.com, [§]dennyzhou@google.com

ABSTRACT

Chain-of-thought prompting combined with pre-trained large language models has achieved encouraging results on complex reasoning tasks. In this paper, we propose a new decoding strategy, *self-consistency*, to replace the naive greedy decoding used in chain-of-thought prompting. It first samples a diverse set of reasoning paths instead of only taking the greedy one, and then selects the most consistent answer by marginalizing out the sampled reasoning paths. Self-consistency leverages the intuition that a complex reasoning problem typically admits multiple different ways of thinking leading to its unique correct answer. Our extensive empirical evaluation shows that self-consistency boosts the performance of chain-of-thought prompting with a striking margin on a range of popular arithmetic and commonsense reasoning benchmarks, including GSM8K (+17.9%), SVAMP (+11.0%), AQuA (+12.2%), StrategyQA (+6.4%) and ARC-challenge (+3.9%).



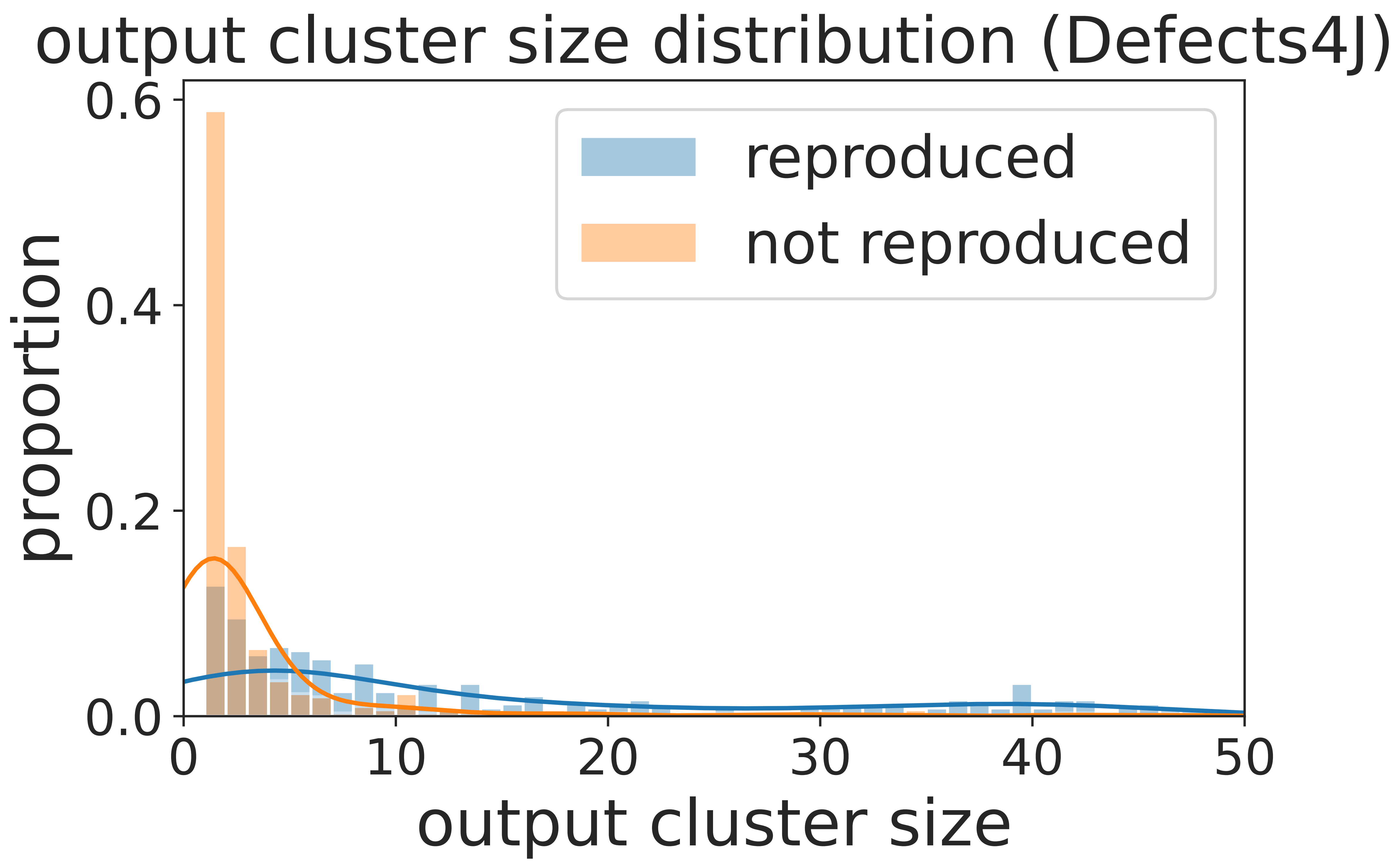
Wang et al., ICLR 2023

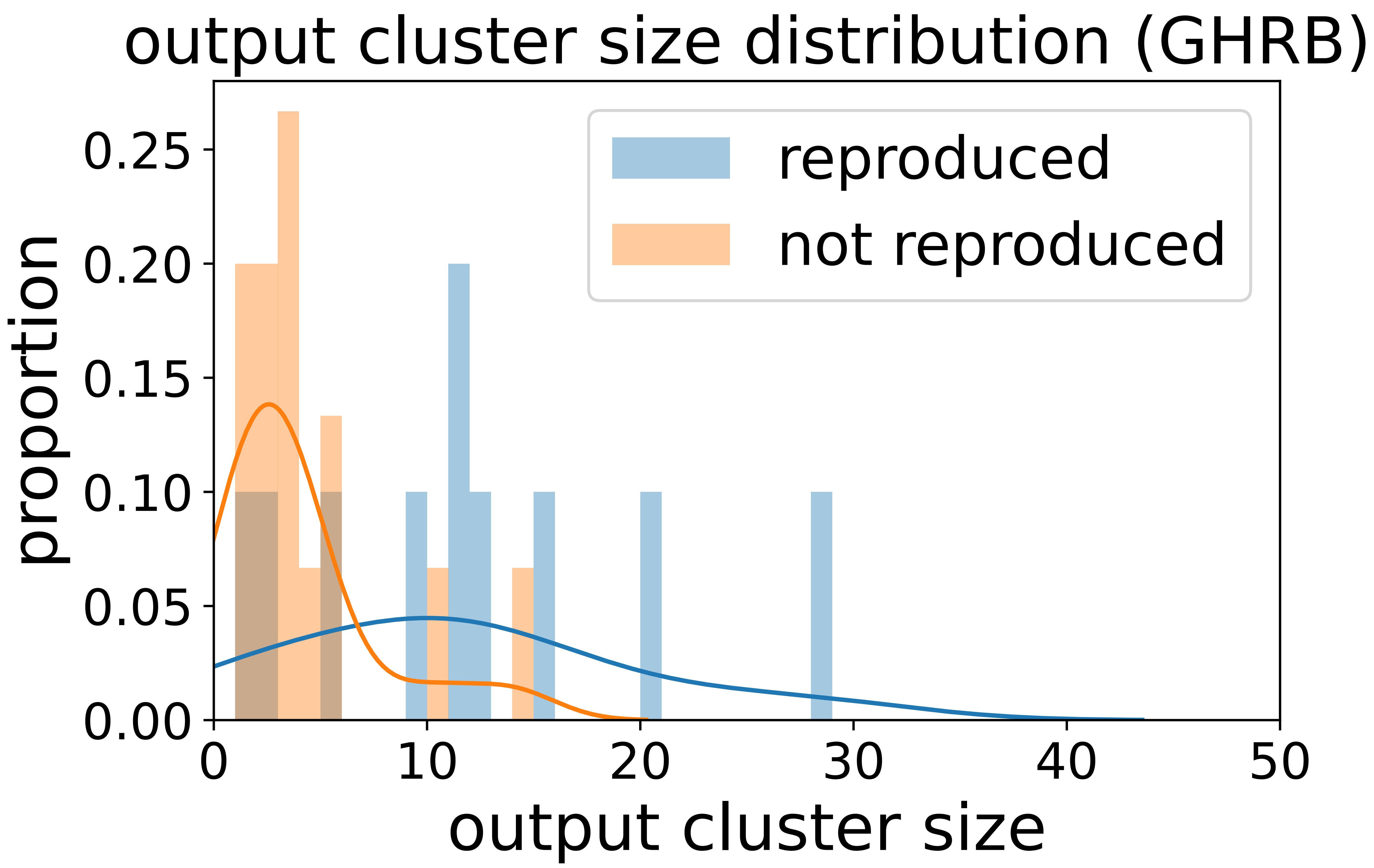
But… really? That simple…?

“the face of a man who is surprised that the answer was so simple.”

LLM-Based Bug Reproduction

Kang, Yoon, & Yoo, ICSE 2023

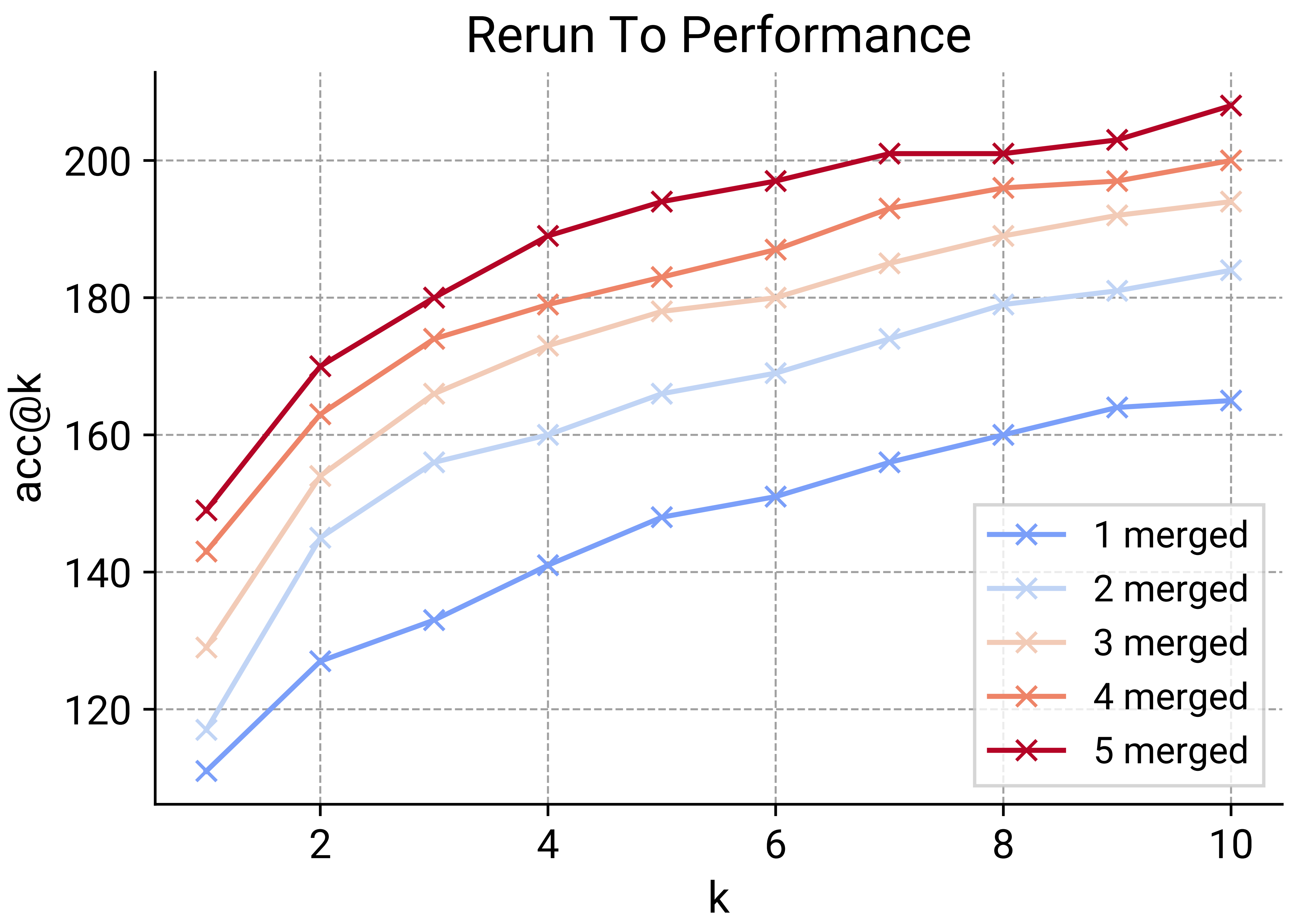




LLM-based Fault Localization

Kang, An & Yoo 2023, https://arxiv.org/abs/2308.05487

Family	Technique	acc@1	acc@3	acc@5
	Predicate Switching	42	99	121
	Stack Trace	57	108	130
	Slicing (frequency)	51	96	119
MBFL	MUSE	73	139	161
	Metallaxis	106	162	191
SBFL	Ochiai	122	192	**218**
	DStar	125	**195**	216
	SBFL-F	34	66	78
LLM-Based	LLM+Test	81	94	97
	AutoFL	**149**	180	194



So, self-consistency is everywhere

It works for code-related LLM tasks too!

- One of the easiest post-processing to improve LLM generations: no external dependencies (well, except the additional cost)
- Can we explain **why** this is the case?
- Can we **model** its behavior?
- Can we apply this to any **target**?

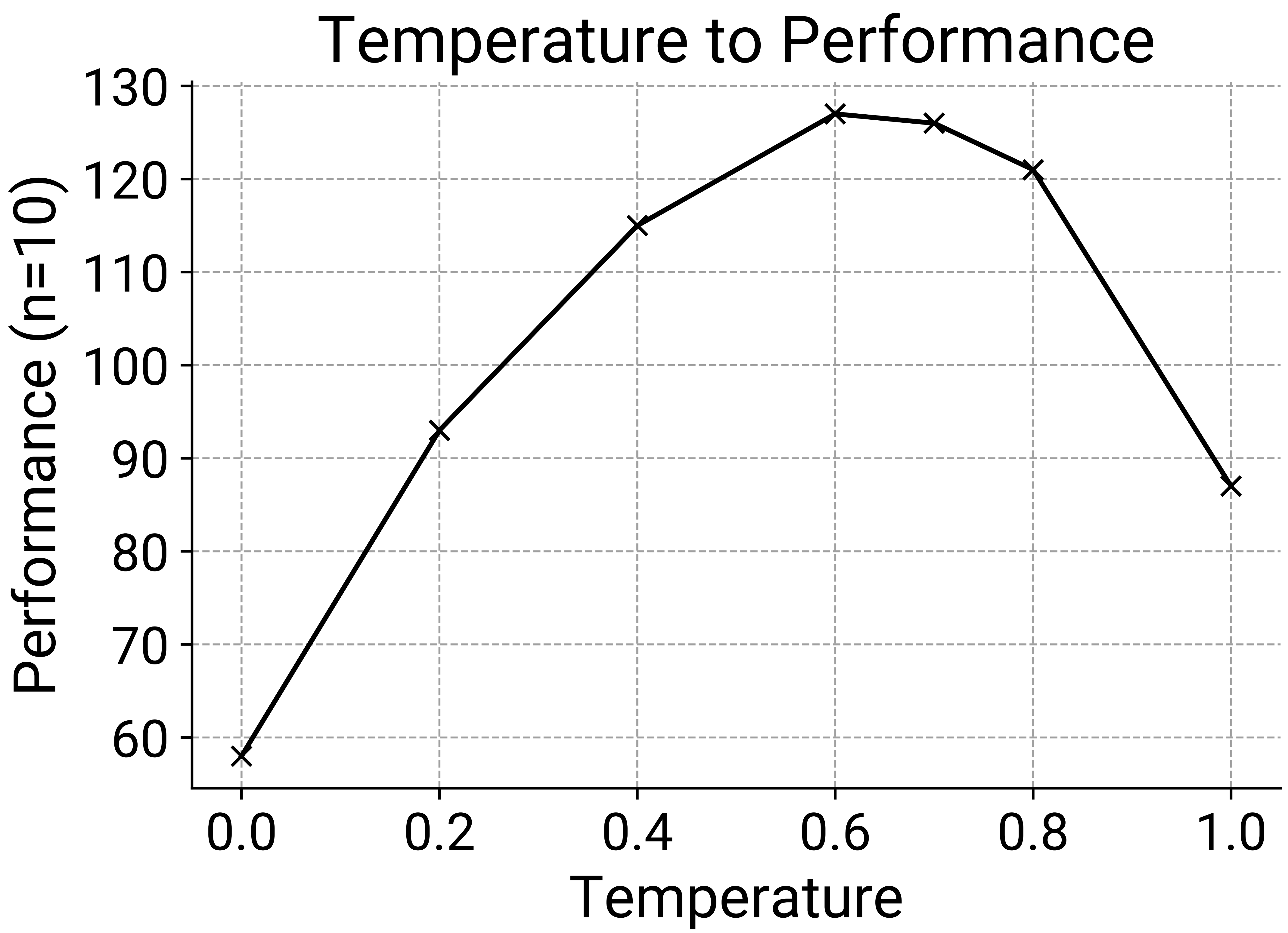
Why does this work?

- Wang et al.'s original intuition: "there are many reasoning paths to the correct solutions, but only one way to arrive at a specific incorrect solution"
- My first reaction: "surely there are infinite ways to arrive at a single incorrect solution!"
- My second reaction: "oh, it is probably assumed that the LLM is at least **trying**… that is, there are infinite total nonsense ways to arrive at a specific incorrect solution, but perhaps *fewer ways to move from the question to a specific incorrect solution while trying to appear plausible*"

LLM-Based Bug Reproduction

Kang et al., Under Review

- Empirical evidence for my second reaction…?
- Too high a temperature —> too random sequence sampling —> not really trying to make sense —> self-consistency seems to break down…



Déjà Vu from Self-Consistency, Part 1

N-version Programming

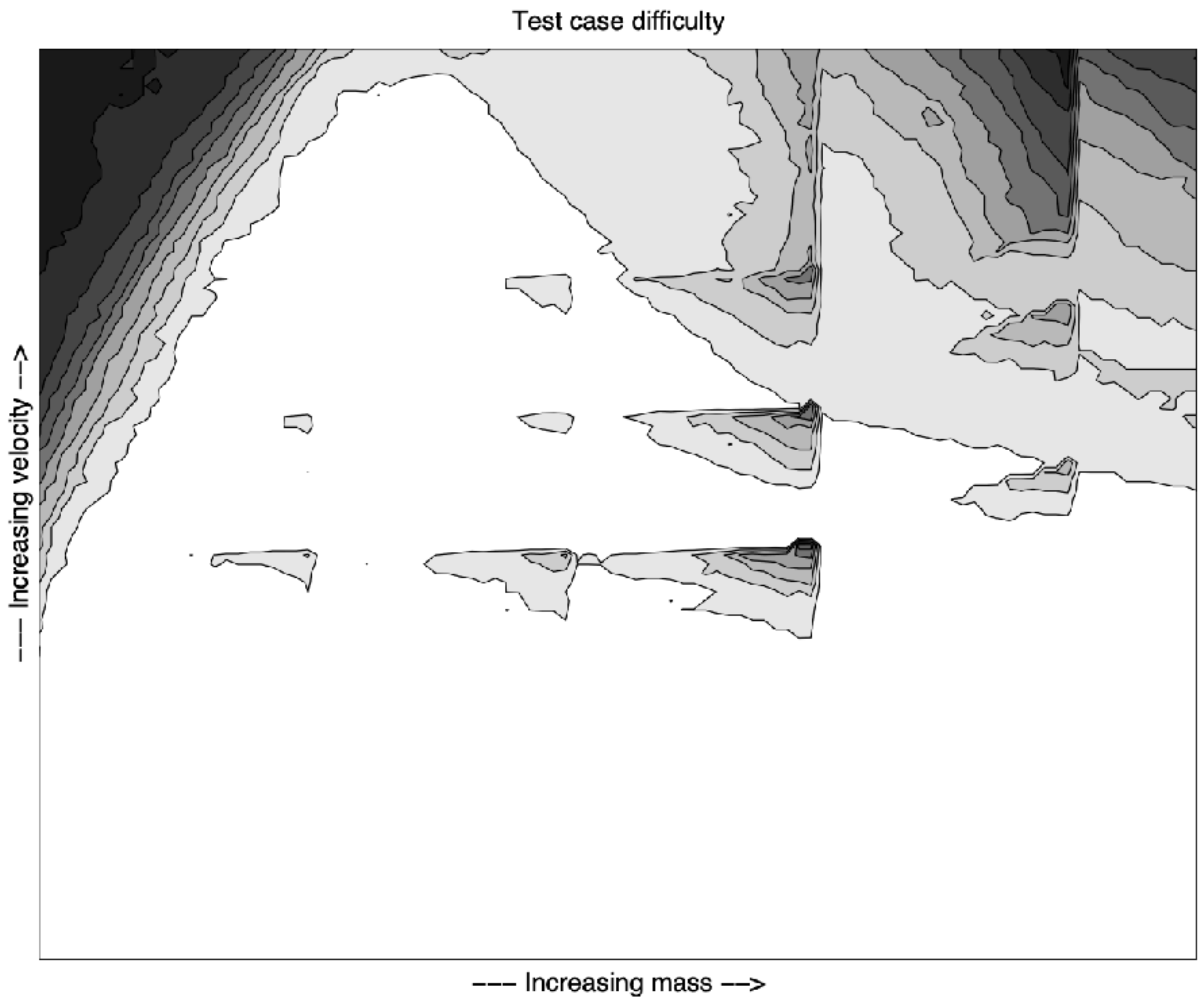
- For a mission-critical system, n-version programming is to make N independent teams to develop N different versions of the system, that are deployed in parallel. Any final decision is made by the majority voting among the N systems and their outputs.

- In some sense, N samples we take from an LLM is N different reasoning chains —> strongly reminiscent of N-version Programming

Déjà Vu from Self-Consistency, Part 1

N-version Programming

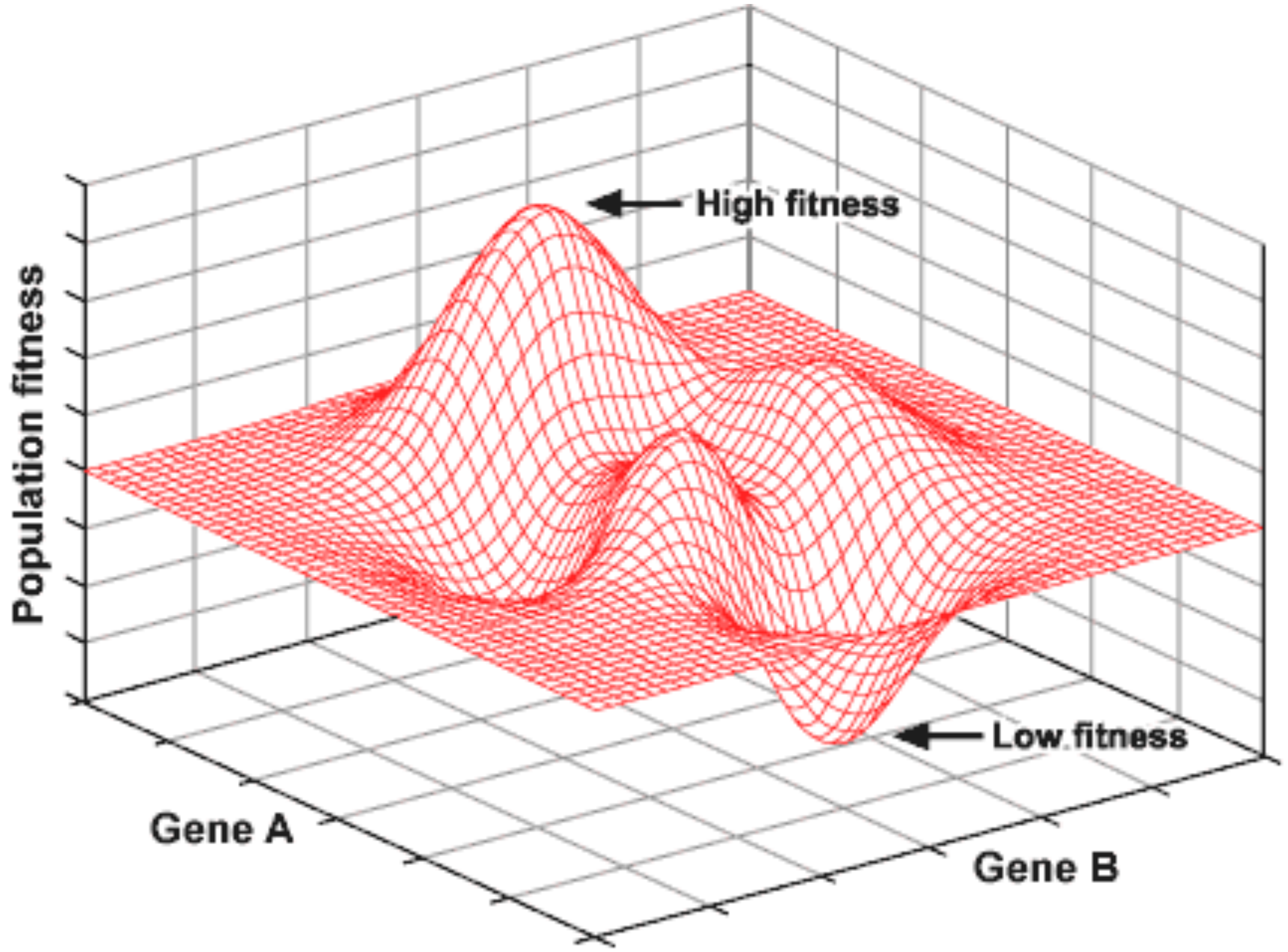
- Feldt, 1999 applied GP to generate 400 versions of Aircraft Braking controller systems.
- Figure shows rate of failure among 400 versions against different areas of input space (aircraft velocity and mass).
- Can self-consistency tell us where the **difficult** problems are?



Déjà Vu from Self-Consistency, Part 2

Fitness Landscape Analysis from Optimization Literature

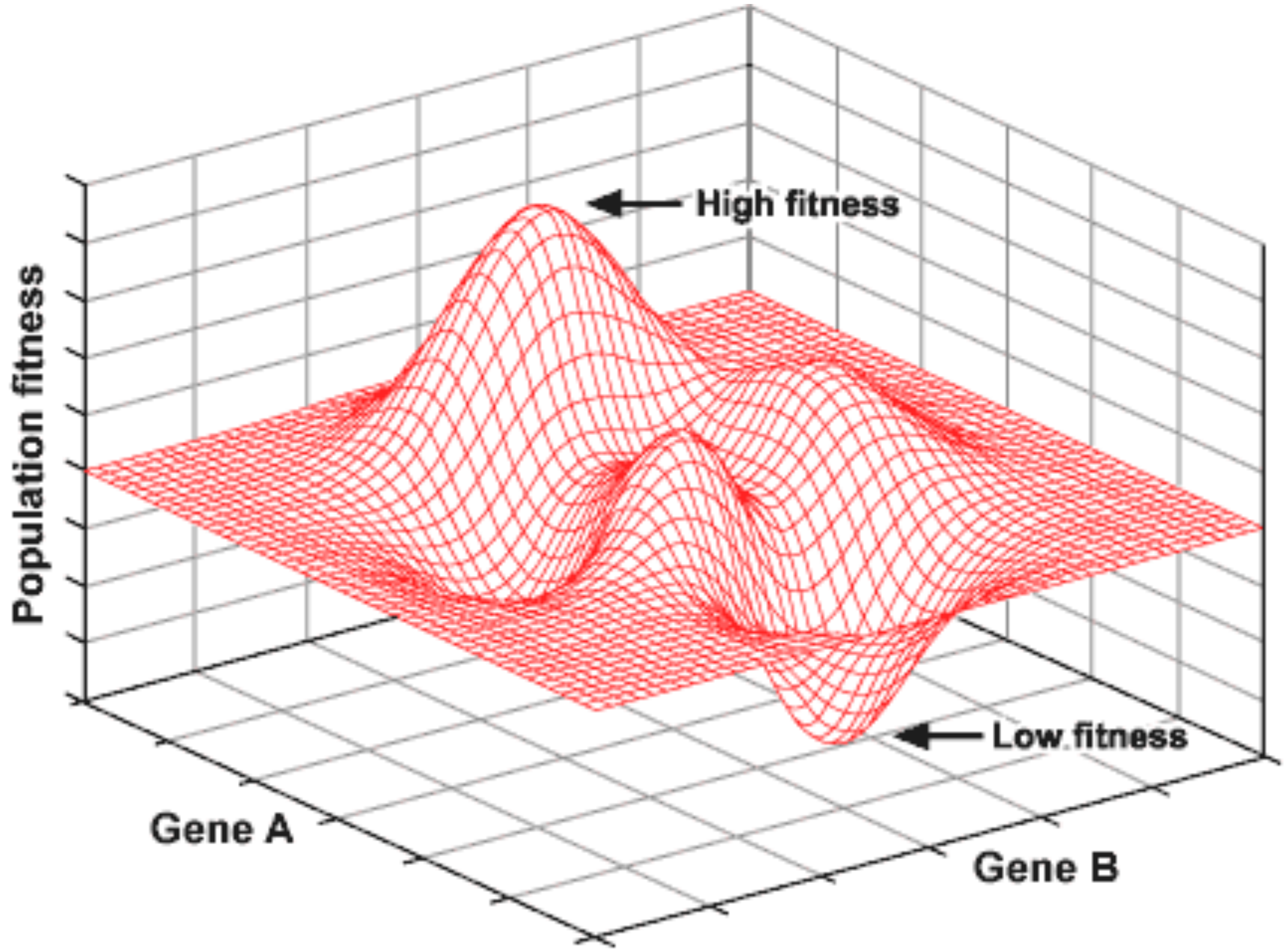
- Fitness Landscape = [solution space] × [fitness dimension]
- Optimisation is essentially climbing up hills to get higher fitness
- What if we see LLM-based solution generation as an optimisation process?
    - What would be the landscape that results in self-consistency?



Déjà Vu from Self-Consistency, Part 2

Fitness Landscape Analysis from Optimisation Literature

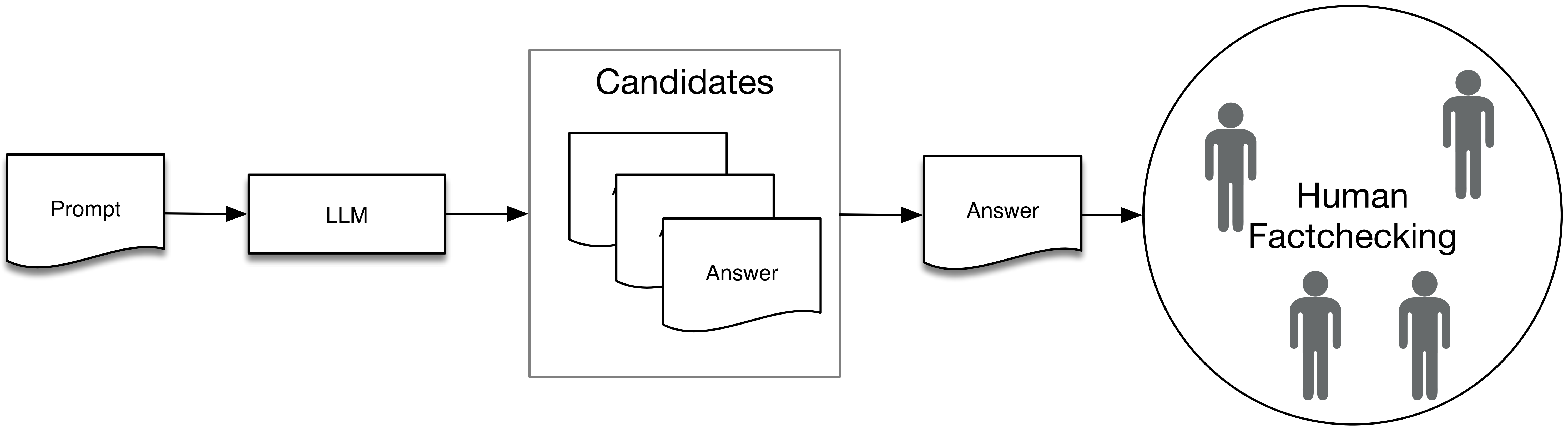
- With problems for which the self-consistency works, I hypothesise that:
  - The tallest hill is also the largest; there are multiple starting points and pathways to the top
  - Smaller hills (=incorrect solutions) have smaller base area, resulting in fewer pathways to their top



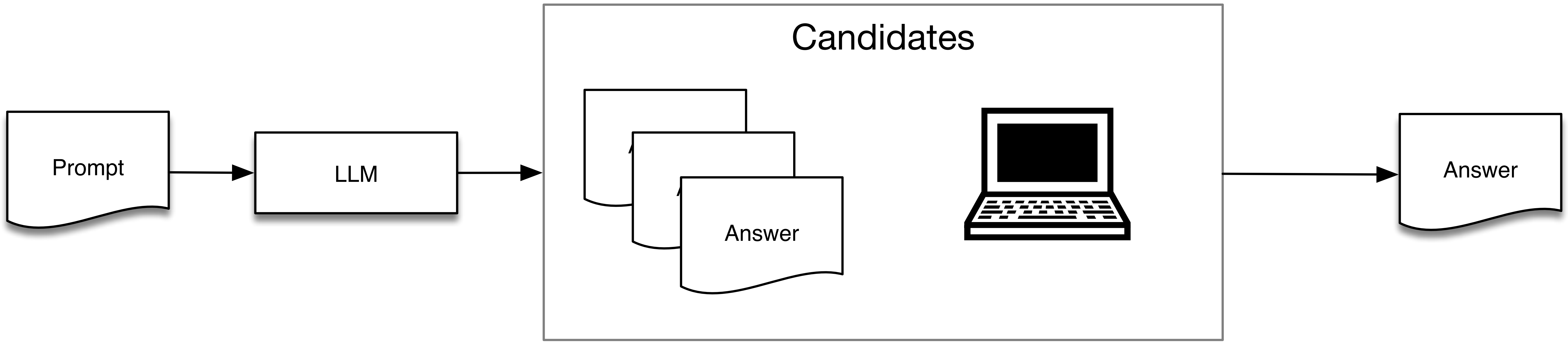
“Interesting. Where does the executability fits in?”

Code is a unique w.r.t. LLM because it executes.

NL + LLM Pipeline

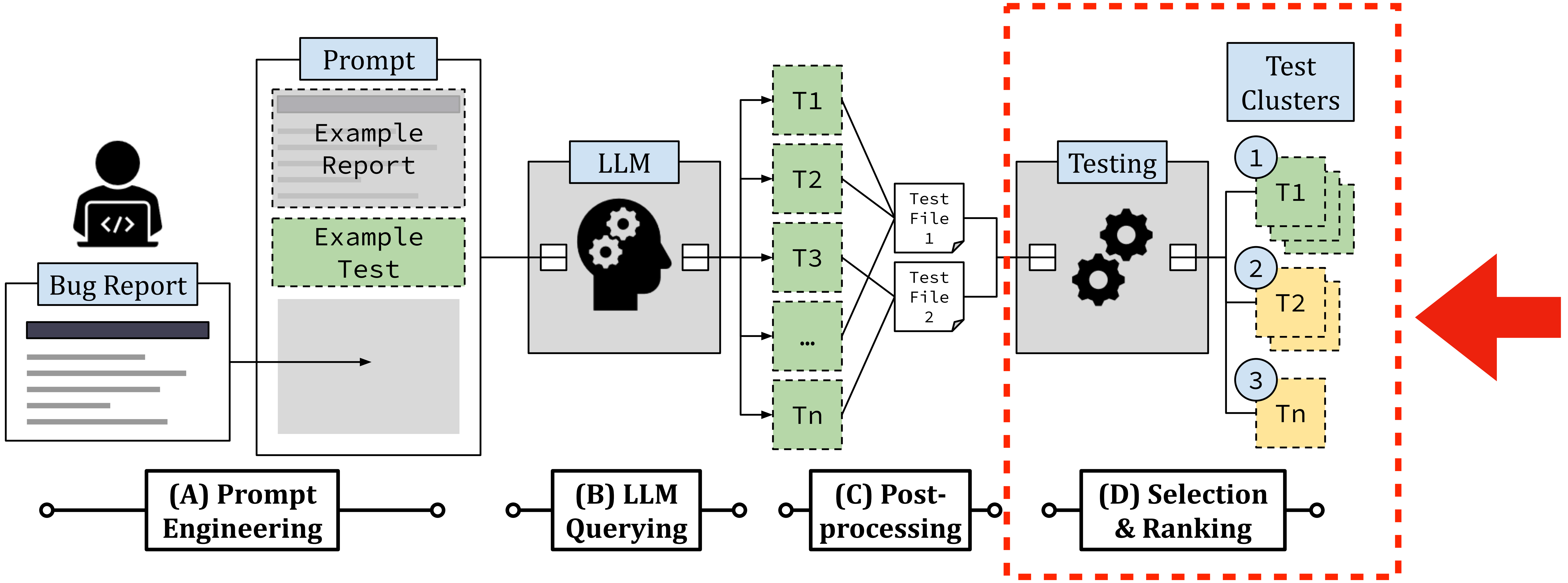


PL/NL + LLM Pipeline



Execution enabling self-consistency

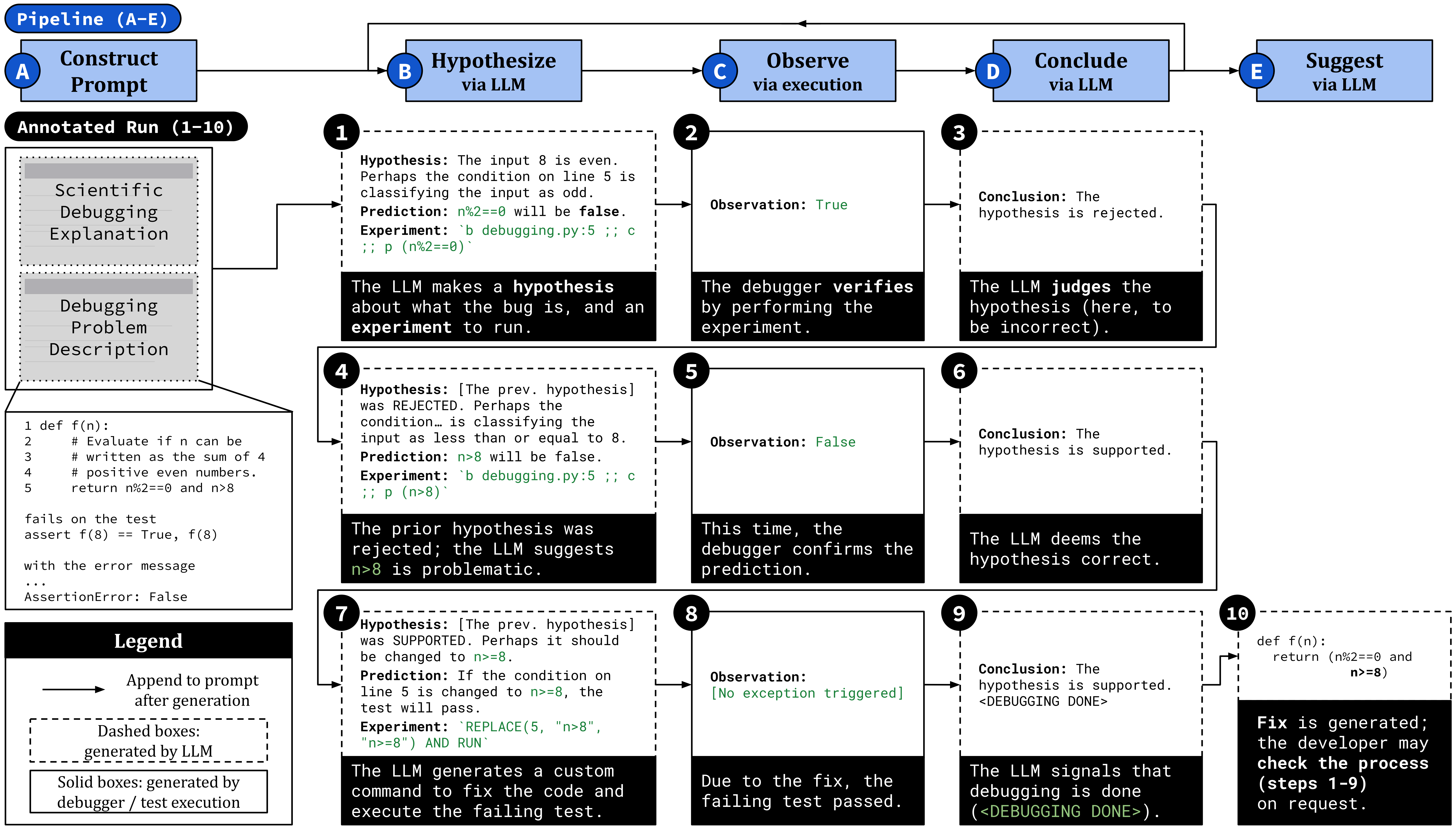
LLM-based Bug Reproduction (Kang, Yoon & Yoo, ICSE 2023)



- Any test that does not fail in the buggy version are filtered out
- Failure type and error messages are considered when clustering tests.

Execution enabling Chain-of-Thoughts

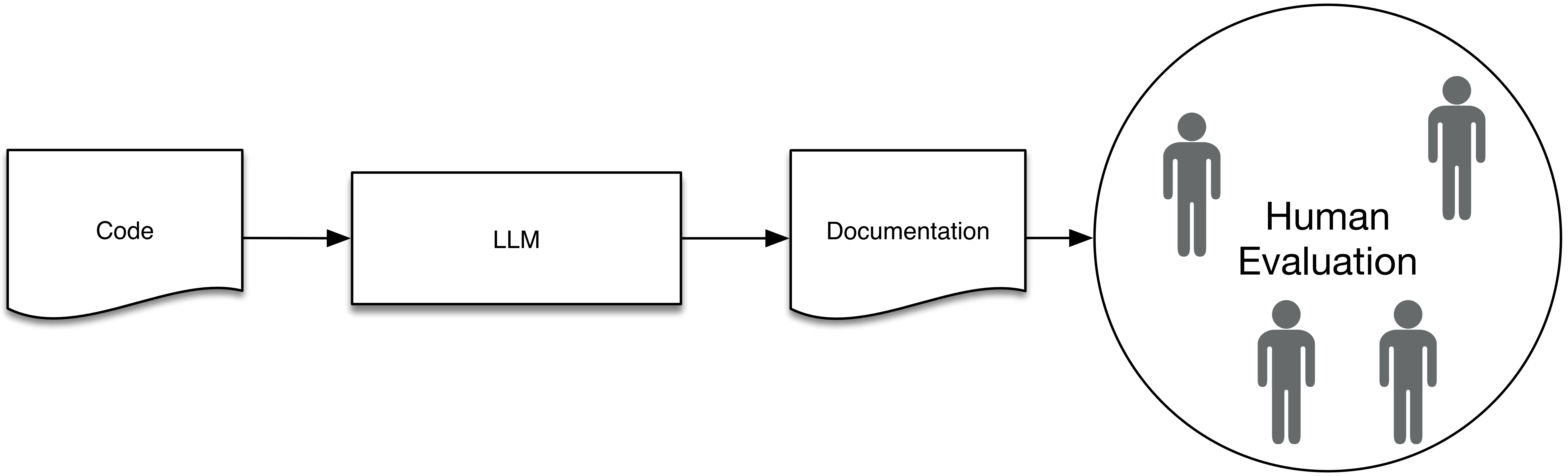
Automated Scientific Debugging, Kang et al., https://arxiv.org/abs/2304.02195

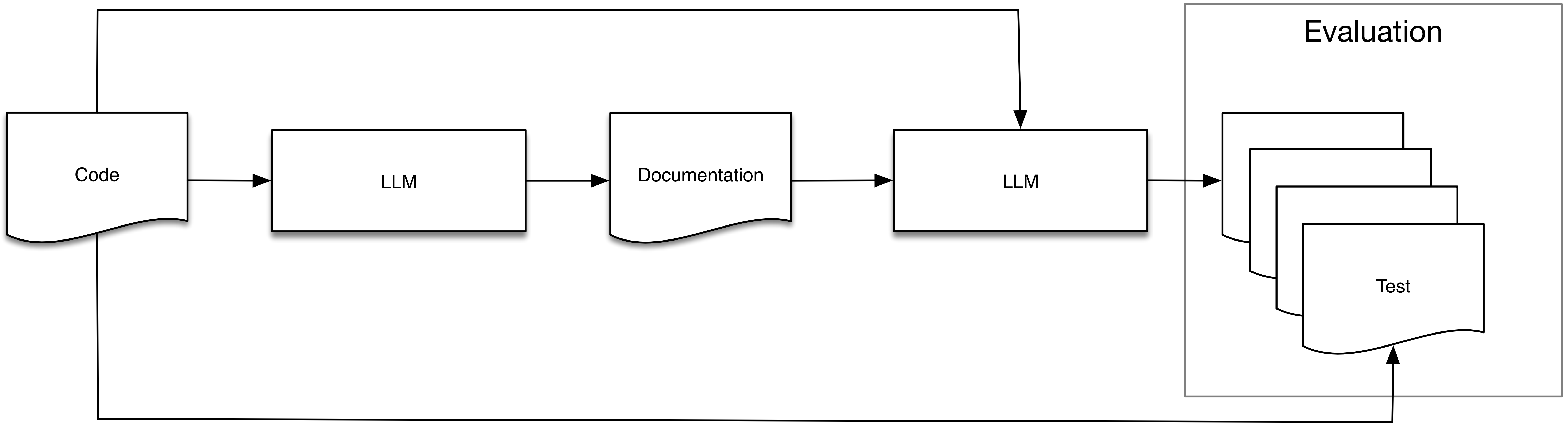


Sungmin Kang
(PhD Candidate)

Executing non-executables (?)
(secondary execution via LLMs)

Sungmin Kang
(PhD Candidate)

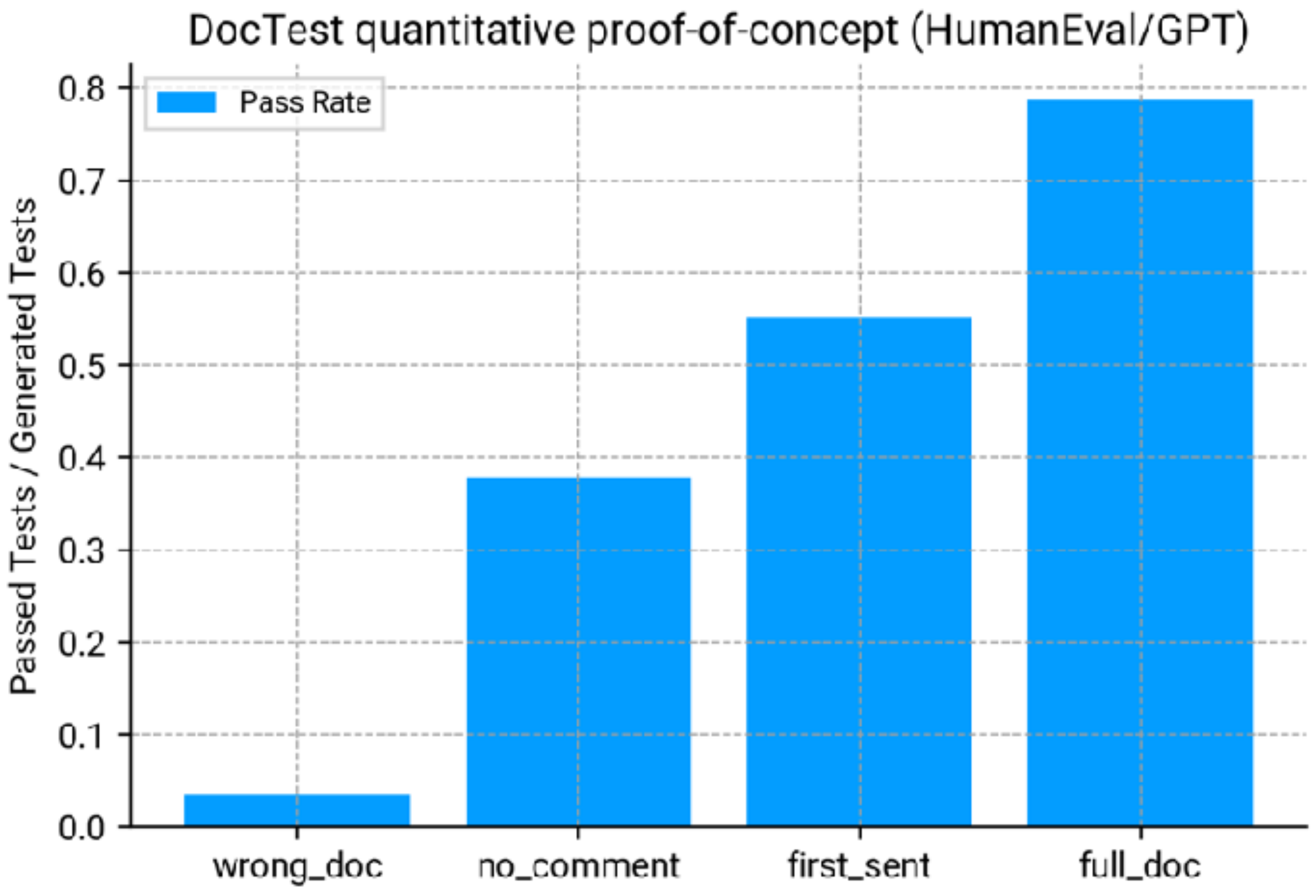


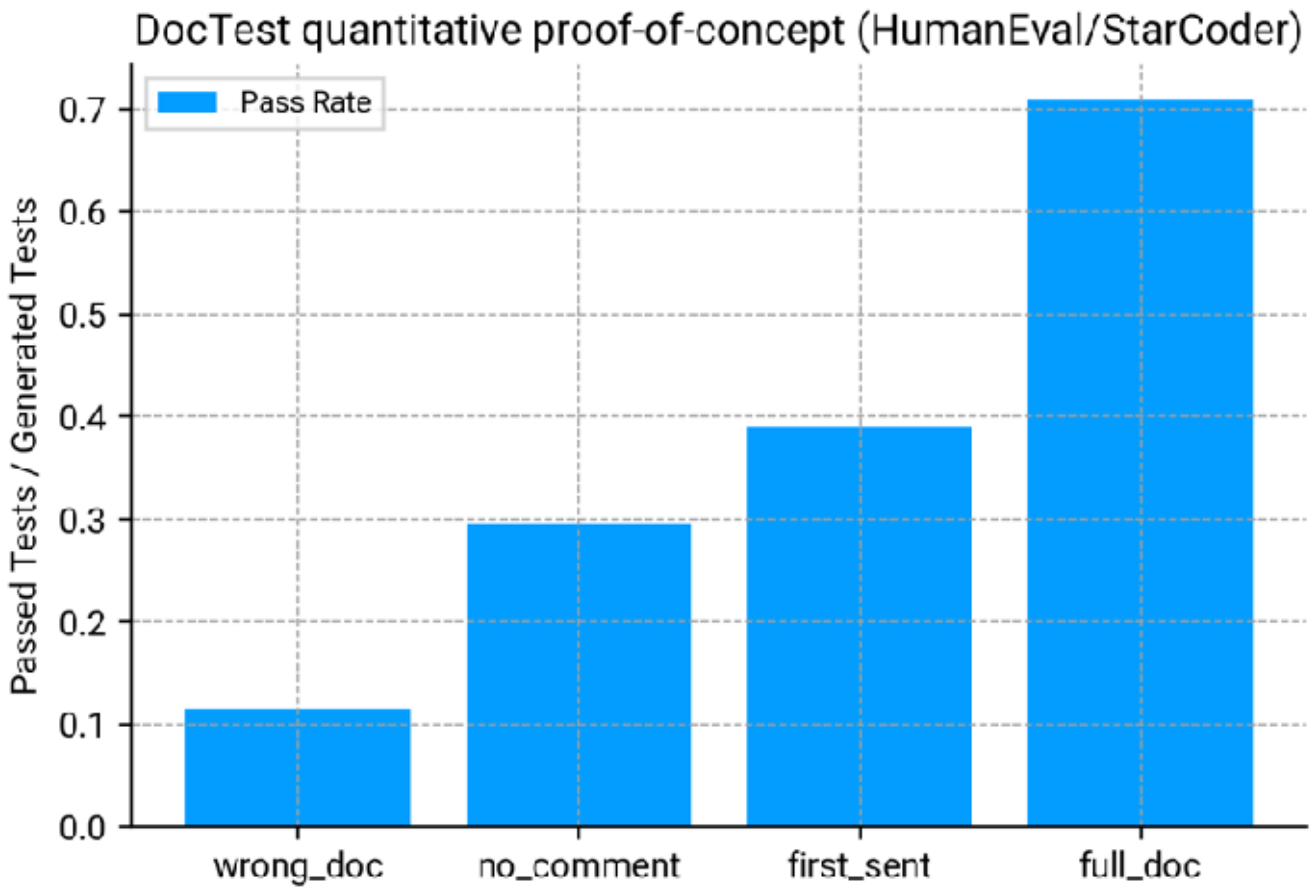


- How do we evaluate the quality of automatically generated documents?
- Derive executables from documents using LLMs, then exploit the executability!
  - (Yes, the derivation introduces imprecision & noise, but still…)

Power of secondary executability

An ongoing work





The better documents the secondary execution is based on, the higher the pass rate becomes.

What this means to GI community

- We need to re-think the semantic/syntactic boundary.
- Naively asking LLMs to do such and such will only go so far; especially if the scope is very narrow, e.g., rewriting a few lines of code.
  - Can LLMs do more structural changes? Refactoring?
- We have amassed a mountain of experience on how we can exploit executions to extract (semantic) interpretations of code and also to induce desirable (semantic) changes - use them well with LLMs!
  - Software testing, program analysis, GI applications…

A critical and essential perspective

LLMs are still autocompletion engines - does it speak Chinese? 🙂

- Do not be too easily persuaded into thinking that they can think :)
- Try to imagine whether the given task can be broken down to chunks of text generation (ideally text that it has seen during training)

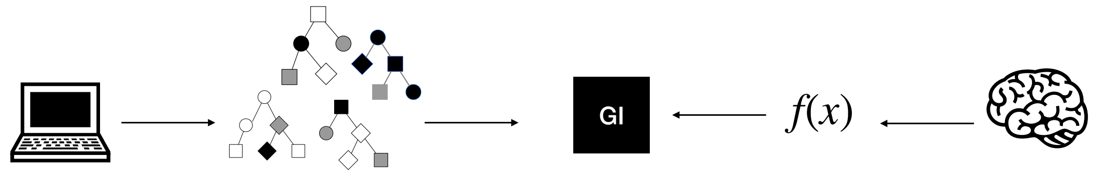
A Thought Experiment

John Searle, “Mind, Brains, and Programs” in 1980

- Suppose we have a computer program that behaves as if it understands Chinese language.
- You are in a closed room with the AI program source code.
- Someone passes a paper with Chinese characters written on it, into the room.
- You use the source code as instruction to generate the response to the input, and sends the response out of the room.
- Do you understand Chinese language, or not?

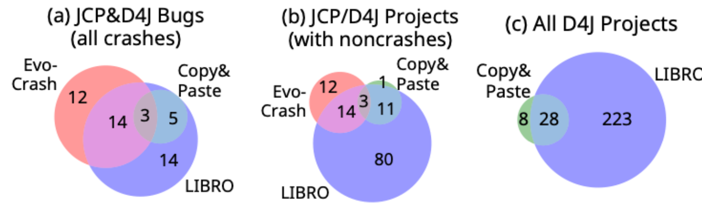
Correlation vs. Causation, or Syntax vs. Semantic

- MIP talk at ICSE 2019 captured this beautifully - *“It Does What You Say, Not What You Mean: Lessons from 10 Years of Program Repair”*
- Traditionally, computing the semantic has been either very difficult or infeasible; as it is well known to the GI community!



Candidate solutions, (randomly) generated via **syntactic** perturbations

Semantic, captured (imperfectly) in fitness functions



Family	Technique	acc@1	acc@3	acc@5
	Predicate Switching	42	99	121
	Stack Trace	57	108	130
	Slicing (frequency)	51	96	119
MBFL	MUSE	73	139	161
	Metallaxis	106	162	191
SBFL	Ochiai	122	192	**218**
	DStar	125	**195**	216
	SBFL-F	34	66	78
LLM-Based	LLM+Test	81	94	97
	AutoFL	**149**	180	194

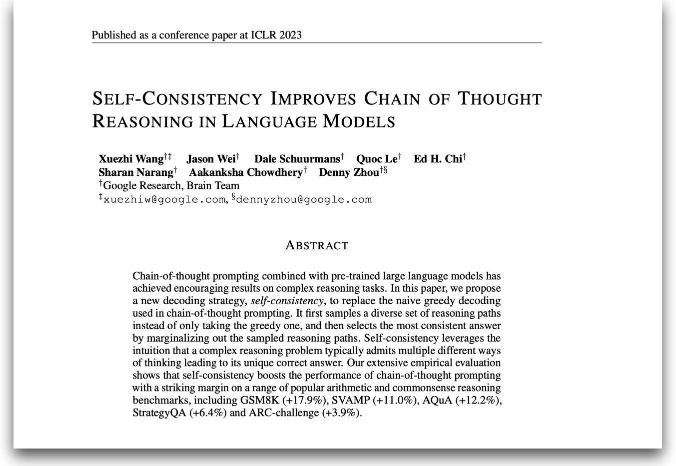
Libro Reproduction Results (against of 750 Bugs)

AutoFL Evaluation Metric (against of 353 Bugs)

Self-Consistency

Wang et al., ICLR 2023

- When sampling answers from an LLM, take multiple answers with high temperature.
- If there is an answer that has the majority among the sampled answers, it is more likely to be the correct one.



Published as a conference paper at ICLR 2023

SELF-CONSISTENCY IMPROVES CHAIN OF THOUGHT REASONING IN LANGUAGE MODELS

Xuezhi Wang[†‡] Jason Wei[†] Dale Schuurmans[†] Quoc Le[†] Ed H. Chi[†]
Sharan Narang[†] Aakanksha Chowdhery[†] Denny Zhou[†§]
[†]Google Research, Brain Team
[‡]xuezhiw@google.com, [§]dennyzhou@google.com

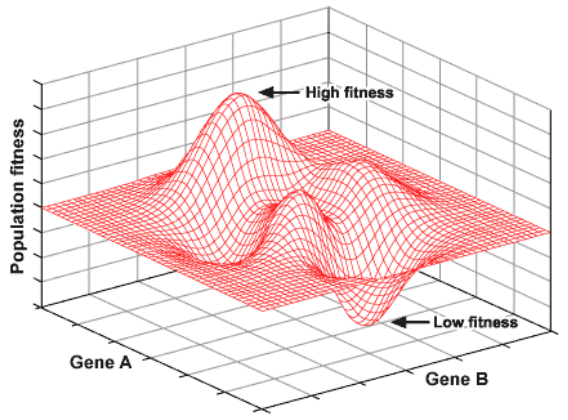
ABSTRACT

Chain-of-thought prompting combined with pre-trained large language models has achieved encouraging results on complex reasoning tasks. In this paper, we propose a new decoding strategy, *self-consistency*, to replace the naive greedy decoding used in chain-of-thought prompting. It first samples a diverse set of reasoning paths instead of only taking the greedy one, and then selects the most consistent answer by marginalizing out the sampled reasoning paths. Self-consistency leverages the intuition that a complex reasoning problem typically admits multiple different ways of thinking leading to its unique correct answer. Our extensive empirical evaluation shows that self-consistency boosts the performance of chain-of-thought prompting with a striking margin on a range of popular arithmetic and commonsense reasoning benchmarks, including GSM8K (+17.9%), SVAMP (+11.0%), AQuA (+12.2%), StrategyQA (+6.4%) and ARC-challenge (+3.9%).

Déjà Vu from Self-Consistency, Part 2

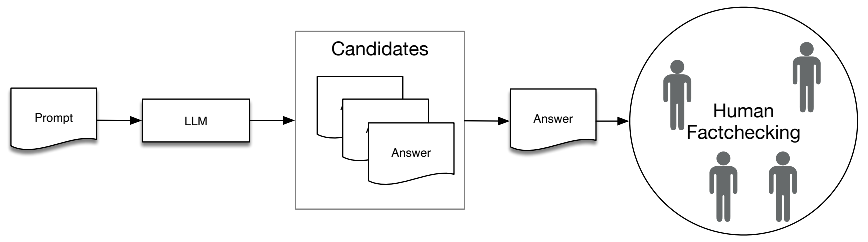
Fitness Landscape Analysis from Optimization Literature

- Fitness Landscape = [solution space] × [fitness dimension]
- Optimisation is essentially climbing up hills to get higher fitness
- What if we see LLM-based solution generation as an optimisation process?
  - What would be the landscape that results in self-consistency?

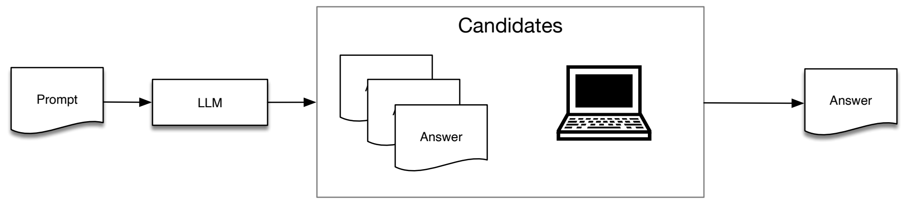


Code is a unique w.r.t. LLM because it executes.

NL + LLM Pipeline



PL/NL + LLM Pipeline



shin.yoo@kaist.ac.kr / https://coinse.github.io