Analysing LLM Inference Chains

What can we tell about LLM as a mode of computation?

Shin Yoo | COINSE@KAIST | KAIST GSSW Colloquim, 17 March 2025

Who am I?

Shin Yoo

- Professor | School of Computing
- Leads Computational Intelligence for Software Engineering Group (https://coinse.github.io)
- Search Based Software Engineering, Automated Debugging, Software Testing…

Let’s go back to 2012

Hindle et al., ICSE 2012

- One of my favourite papers: On Naturalness of Software (https://dl.acm.org/doi/10.5555/2337223.2337322)
- “Programming languages, in theory, are complex, flexible and powerful, but the programs that real people actually write are mostly simple and rather repetitive, and thus they have usefully predictable statistical properties that can be captured in statistical language models and leveraged for software engineering tasks.”
- But what is “naturalness”?

What is “natural” about language?

- Natural language refers to ordinary languages that occur naturally in human community “by process of use, repetition, and change without conscious planning of premeditation” (Wikipedia)
- From the statistical point of view, it means that most of our utterances are simple, repetitive, and therefore predictable.
    - Surely this is how we all learn language.

John: Hi, nice to meet you. How are you?

Mary: I’m ____, _____ ___. ___ ___?

a) fine, thank you. And you?

b) okay, I guess. But why?

What about code?

- It is not “natural”, in the sense that we have artificially created the grammar for programming languages.
  - Programming languages do evolve, but how?
    - Intentionally? New grammars, language consortiums, etc…
    - Gradually? Languages do affect each other, a newer and more popular style eventually gets accepted, etc…

Python: for _ __ _____ …

a) i in range

b) (int i = 0;

Java: for _ ___ _ _ _ _ …

a) i in range

b) (int i = 0;

Statistical Language Model

- Given a set of tokens, $\mathcal{T}$, a set of possible utterances, $\mathcal{T}^*$, and a set of actual utterances, $\mathcal{S} \subset \mathcal{T}^*$, a language model is a probability distribution p over utterances $s \in \mathcal{S}$, i.e., $\forall s \in \mathcal{S}[0 < p(s) < 1 \wedge \sum_{s \in \mathcal{S}} p(s) = 1$

- An utterance (or a sentence) is a sequence of tokens (or words). Suppose we have N tokens, $a_1, a_2, \ldots, a_N$ that consist s. What is $p(s)$?

  - $p(s) = p(a_1)p(a_2 \mid a_1)p(a_3 \mid a_1 \,.\, a_2)p(a_4 \mid a_1, a_2, a_3)\ldots p(a_N \mid a_1 \ldots a_{N-1})$

  - But these conditional probabilities are hard to calculate: the only feasible approach would be count each utterance that qualifies, but $\mathcal{S}$ is too big, let alone $\mathcal{T}^*$.

N-Grams

- Assumes Markov property, i.e., the next token is influenced only by those came immediately before (say, within the window of n tokens)!
  - $p(a_i \mid a_1 \ldots a_{i-1}) \simeq p(a_i \mid a_{i-3} a_{i-2} a_{i-1})$
- This is now much more tractable:
  - $p(a_i \mid a_{i-3} a_{i-2} a_{i-1}) = \frac{count(a_{i-3}, a_{i-2}, a_{i-1}, a_i)}{count(a_{i-3}, a_{i-2}, a_{i-1}, *)}$
- Given some context, we can now compute the probability of the candidate token that comes next. In other words, we can predict the next token!

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:
  - They **seem to** get the semantics of the code and **work across natural and programming language**
  - **Emergent behavior** leading to very attractive properties such as in-context learning, Chain-of-Thoughts, or PAL

Survey of the Explosion

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

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

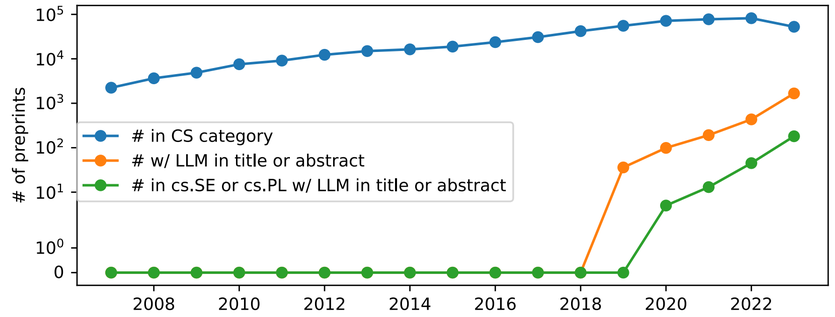


Fig. 2. Trends in number of arXiv preprints. The blue line denotes the number of preprints categorised under "CS". The orange line denotes the number of preprints in AI (cs.AI), Machine Learning (cs.LG), Neural and Evolutionary Computing (cs.NE), Software Engineering (cs.SE), and Programming Language (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"

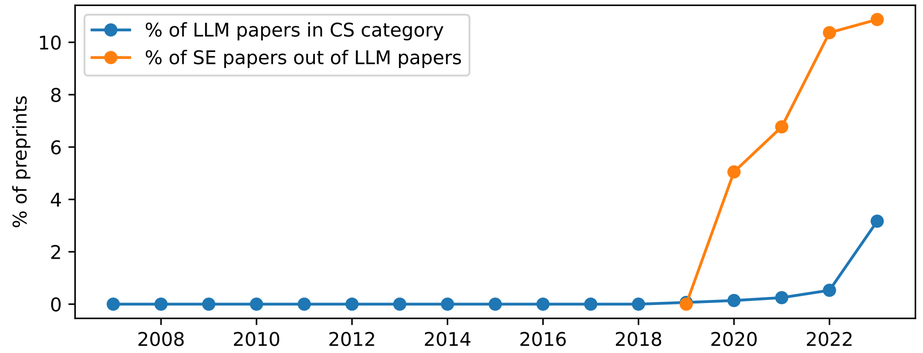
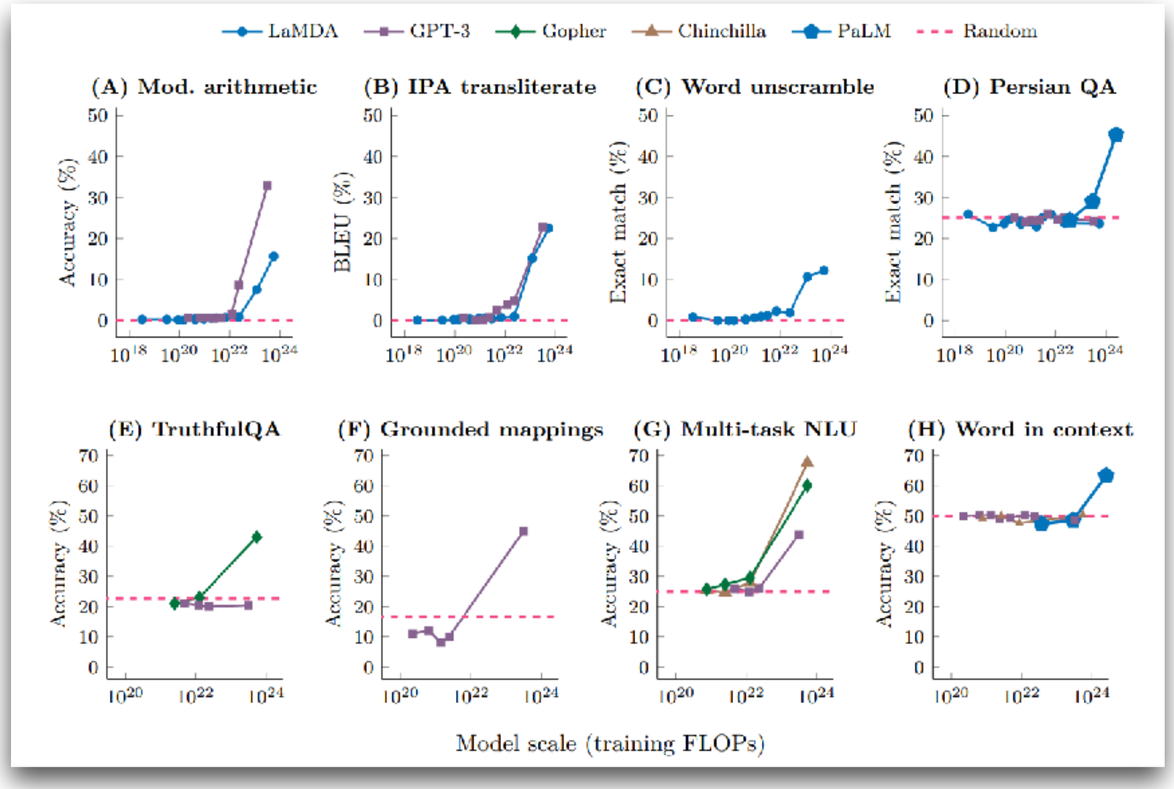


Fig. 3. Proportions of LLM papers and SE papers about LLMs. By "about LLMs", we mean that either the title or the abstract of a preprint contains "LLM", "Large Language Model", or "GPT". The blue line denotes the percentage of the number of preprints about LLMs out of the number of all preprints in the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

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

Do we expect LLMs to replace developers?

- I am cautiously skeptical about this.
- At its core, it is still a statistical language model, i.e., it predicts the most natural utterances.
- Whether this is a sufficient infrastructure for general intelligence, no one knows.
- But we can nonetheless harness the statistical nature of LLMs in a productive way in the context of software engineering :)

In-context Learning

- Previously, getting a model for a specific task involved either dedicated model + training, or at least general pre-trained model + fine-tuning
- Above certain size, LLMs show the ability to perform in-context learning, i.e., they learn as part of their context (i.e., preceding tokens), leading to **prompt engineering**:
  - Few-shot learning: the context explains the problem, and gives a few examples of question-answer. LLMs can now answer an un-seen question.
  - Zero-shot learning: the context explains the problem as well as how it can be solved. LLMs can now answer an un-seen problem.

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 include 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

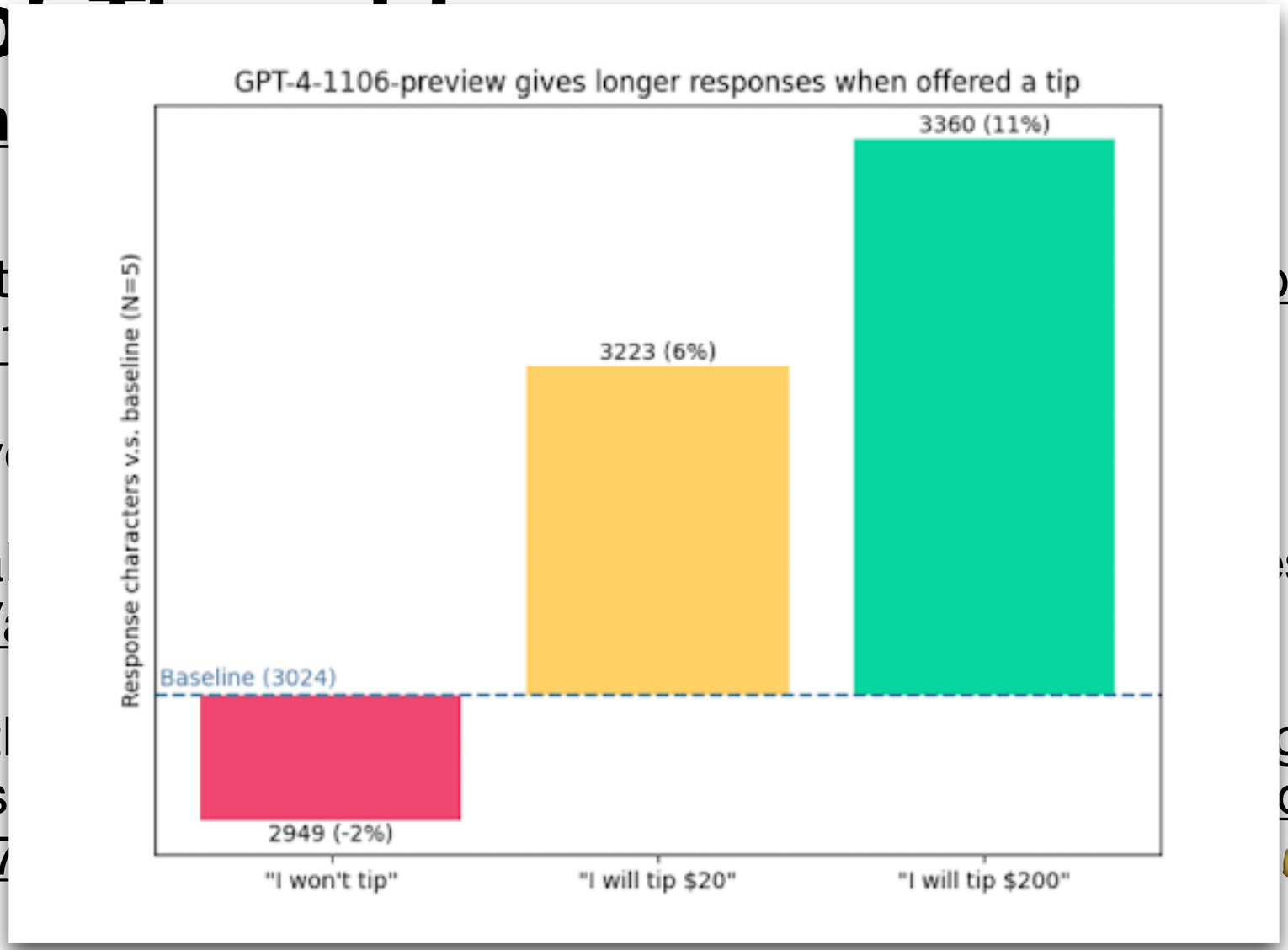
Wei et al., h

- Add “Let’s t
- We have ev
  - If you ma
  - Apparentl

ps://arxiv.org/abs/2205.11

es (https://arxiv.org/a

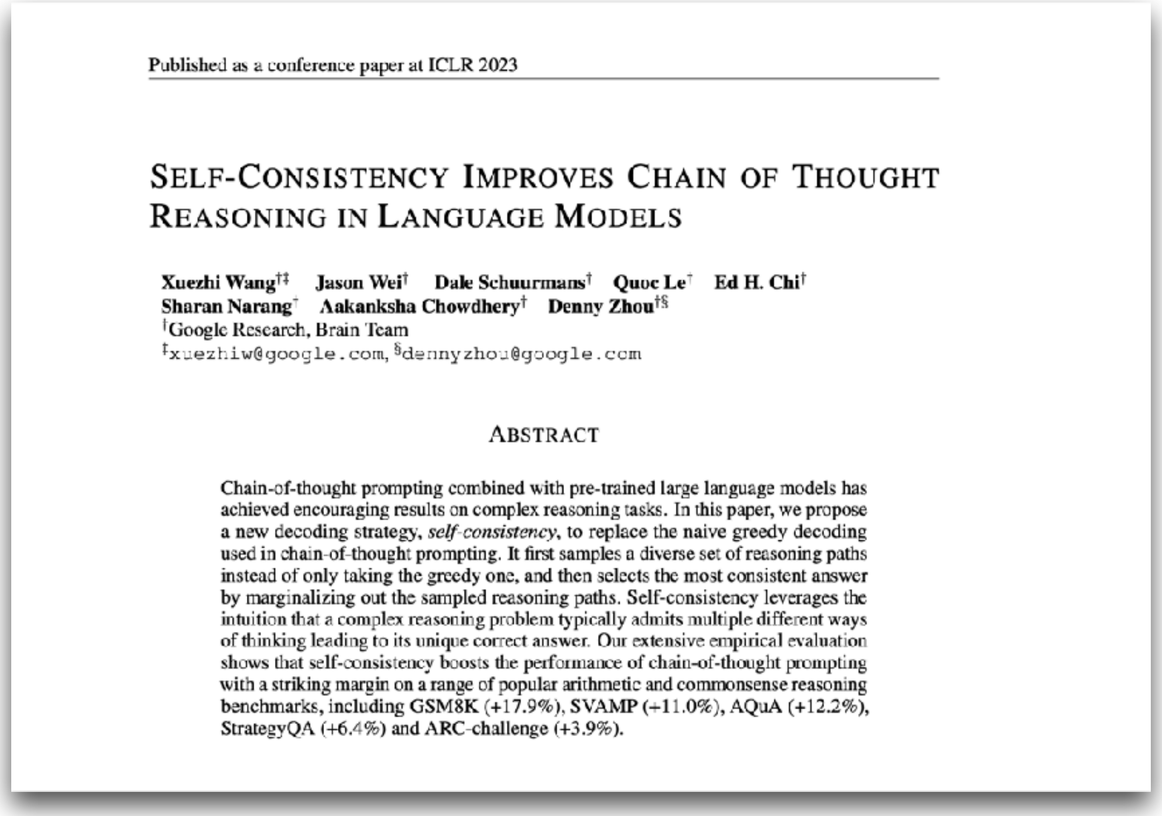
ge tip produces oogel/status/1730726



Self-Consistency

Wang et al., ICLR 2023 (https://arxiv.org/abs/2203.11171)

- 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.



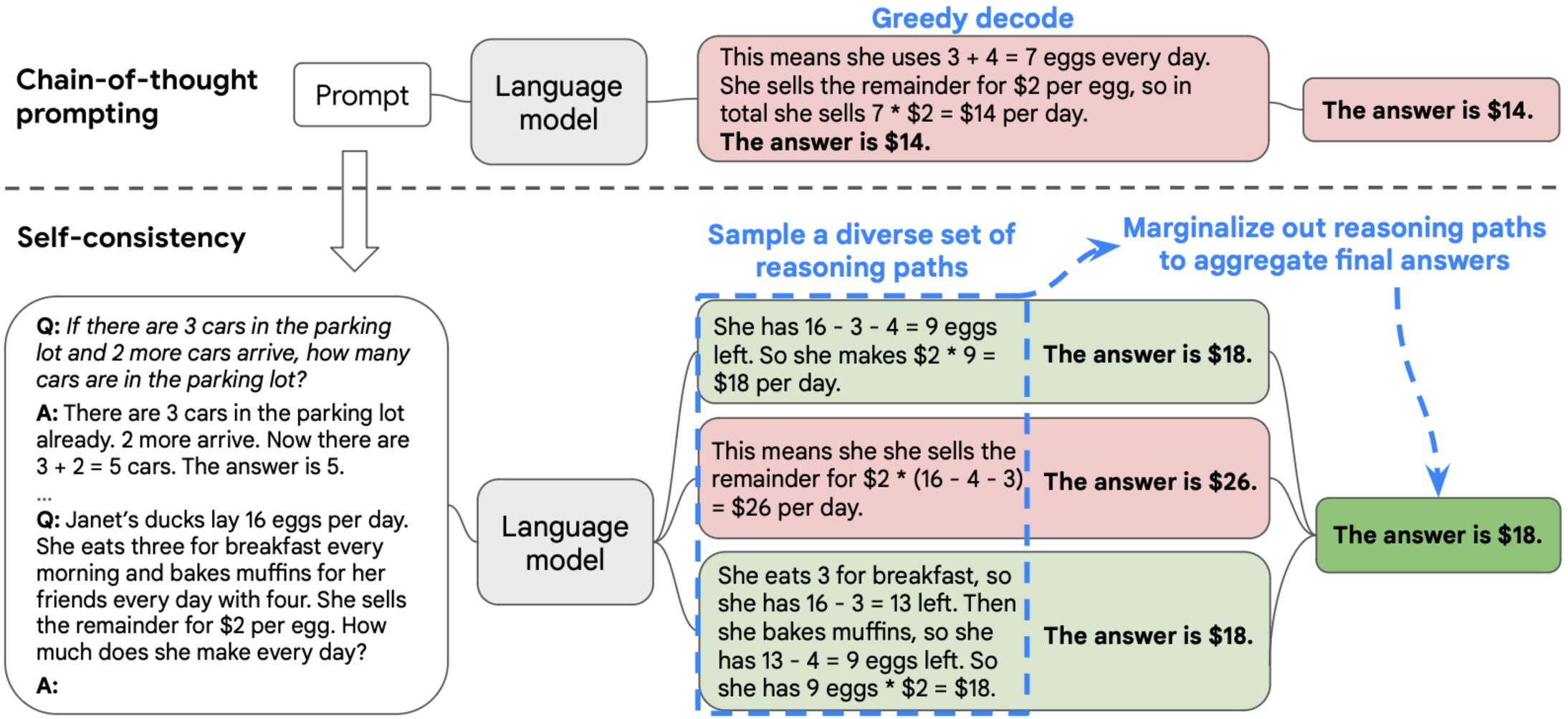
Published as a conference paper at ICLR 2023

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%).

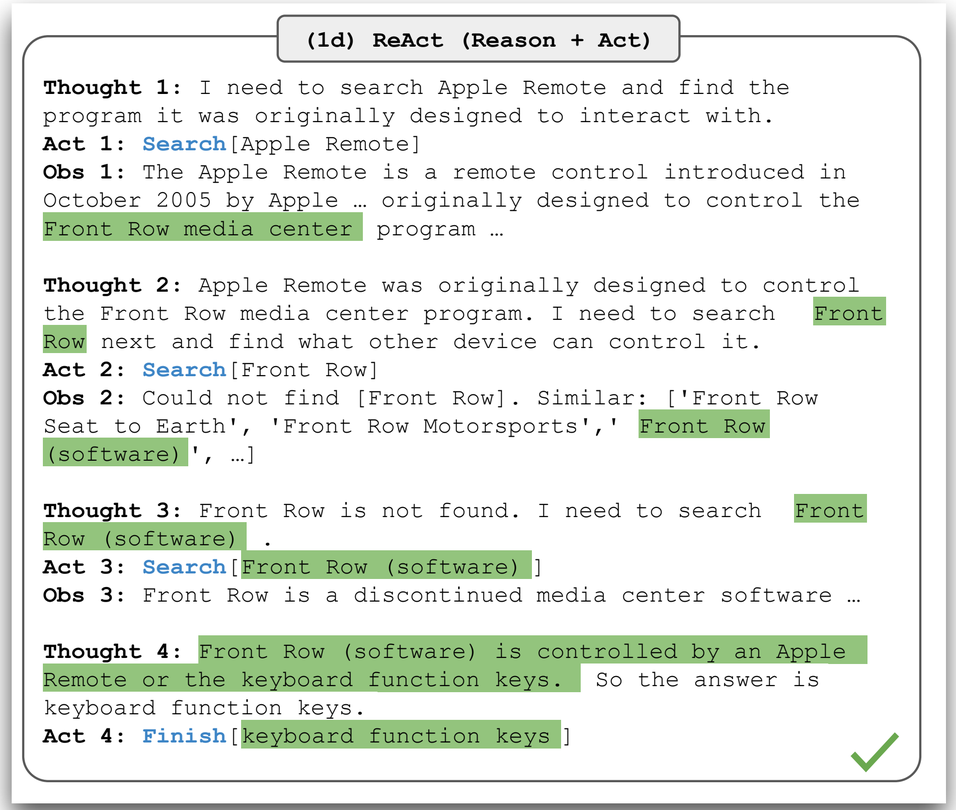


Wang et al., ICLR 2023

ReAct

Yao et al., ICLR 2023 (https://arxiv.org/abs/2210.03629)

- What if we need external information for the in-context learning? In other words, can LLMs be given tools?
- Remember that this is still autocompletion:
  - LLMs can be taught to signal the need to invoke tools
  - Whenever LLMs need a tool invocation, we can do it ourselves and paste the outcome back into the context



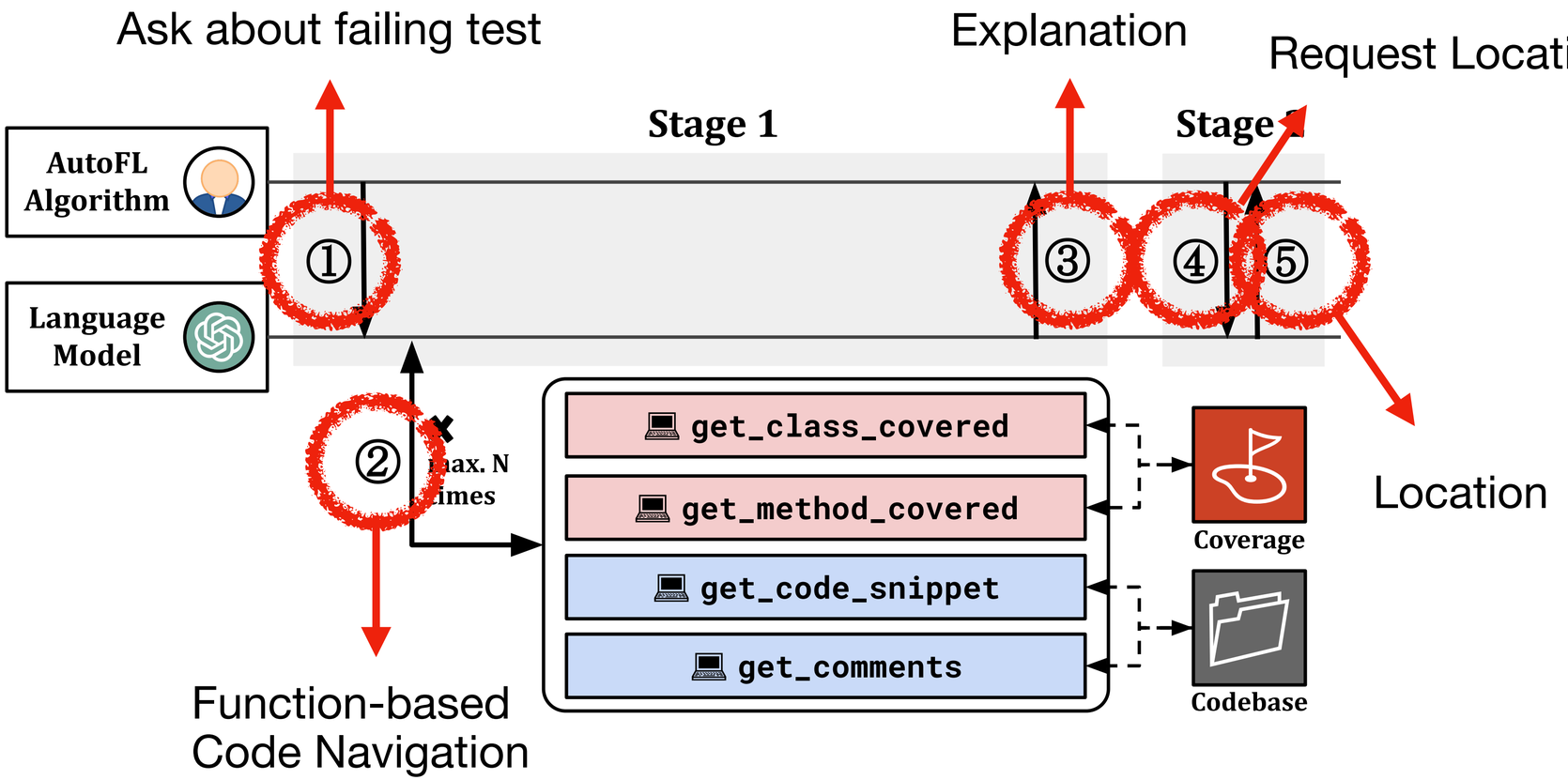
ReAct: Synergizing Reasoning and Acting in Language Models, Yao et al., ICLR 2023
https://arxiv.org/abs/2210.03629

LLM-based Fault Localization

Kang et al. FSE 2024 (https://arxiv.org/abs/2308.05487)

Dr. Gabin An
(ROKU Korea)

Dr. Sungmin Kang
(NUS)



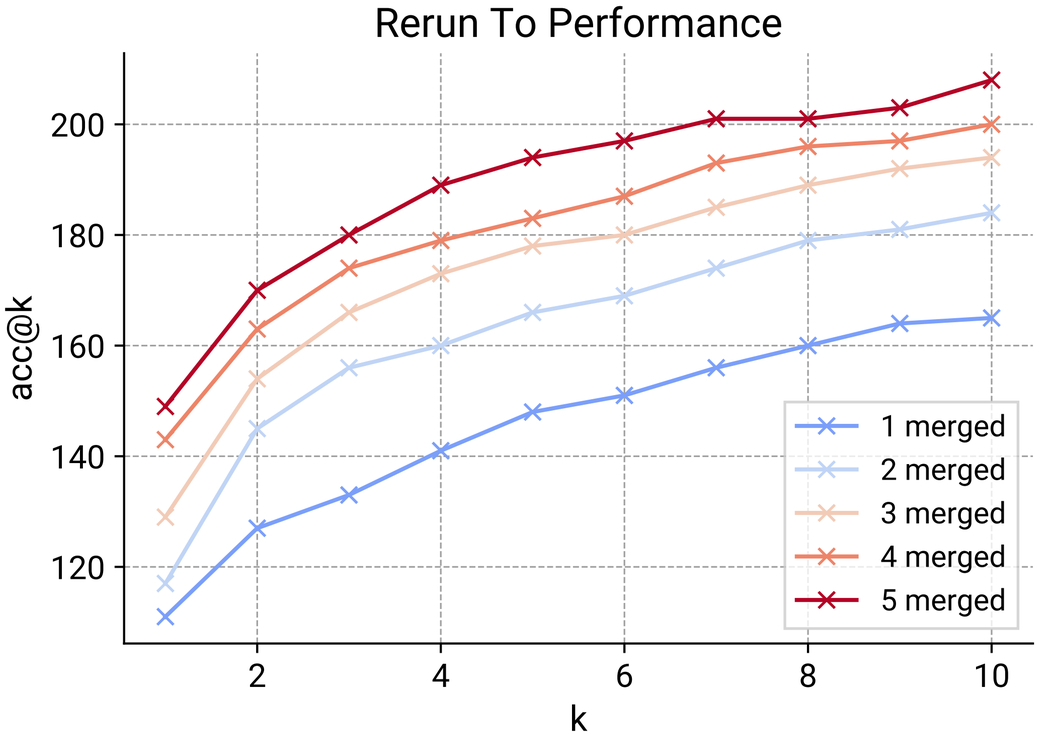
LLM-based Fault Localization

Kang et al. FSE 2024 (https://arxiv.org/abs/2308.05487)

Dr. Gabin An
(ROKU Korea)

Dr. Sungmin Kang
(NUS)

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



Autonomous GUI Testing for Android

Yoon et al., ICST 2024 (https://arxiv.org/abs/2311.08649)

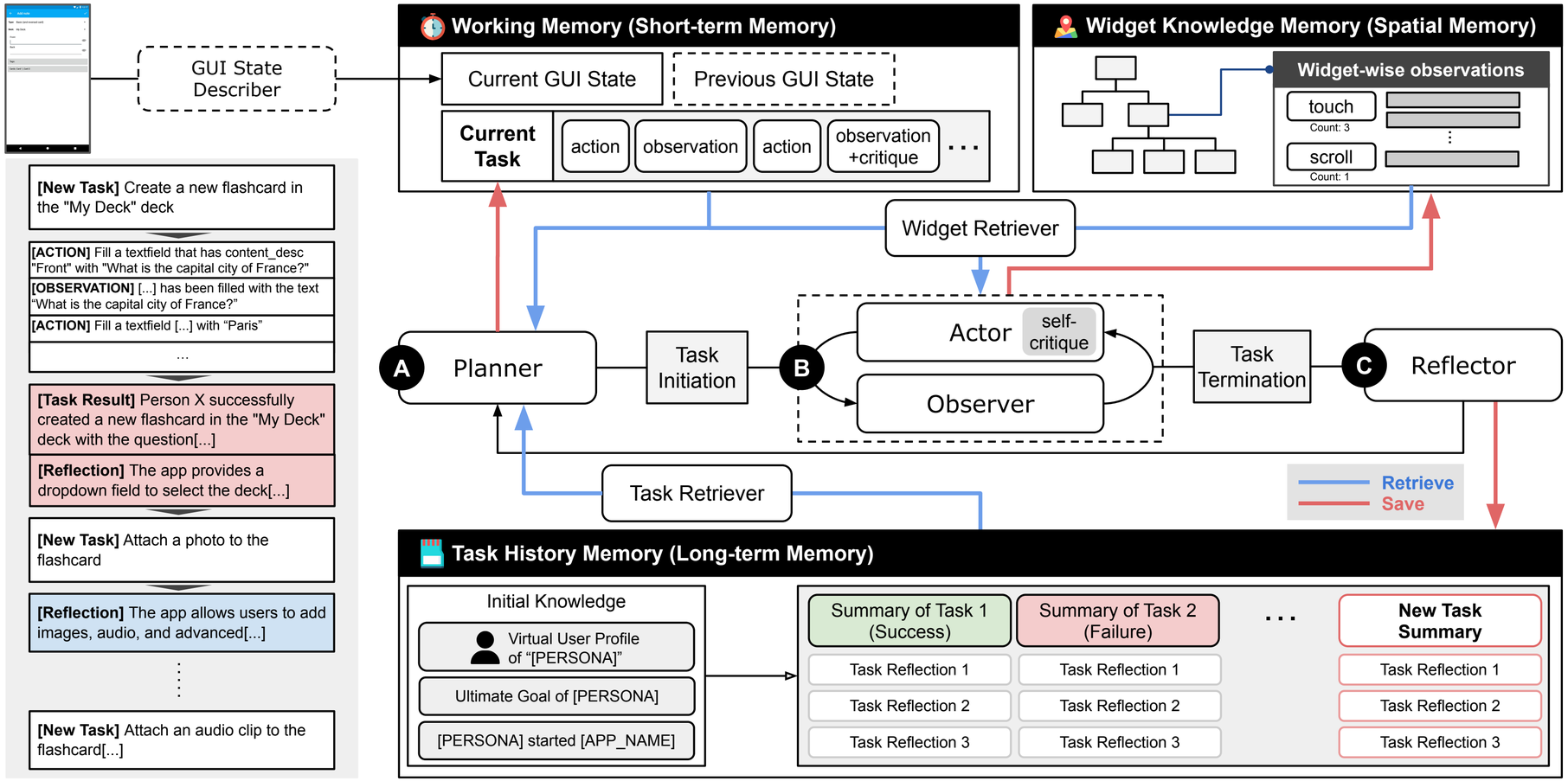


Fig. 1. Overview of DroidAgent with a task example.

Juyeon Yoon
(PhD Candidate)

Prof. Robert Feldt
(Chalmers)

Agents* are the future…?

(*: and not just advancing foundational models)

- Andrew Ng thinks that agents are the future (https://www.youtube.com/watch?v=KrRD7r7y7NY) 🤓
- Future SW = Agents driven by LLMs
  - Much more than prompt engineering: tool usage, design patterns, multiple agents collaborating internally, etc…

Analyzing the future SW :)

- The traditional program analysis:
  - Static analysis: requires clearly defined semantics + source code
  - Dynamic analysis: requires well formatted input spec + model of execution (coverage, dataflow, etc)
- Multi-Agent LLM System:
  - Static analysis: semantic behavior exist but no definition + no source code
  - Dynamic analysis: input is free form + no model of execution

Moirai

Three sisters of fate

At the beginning, there are inputs.

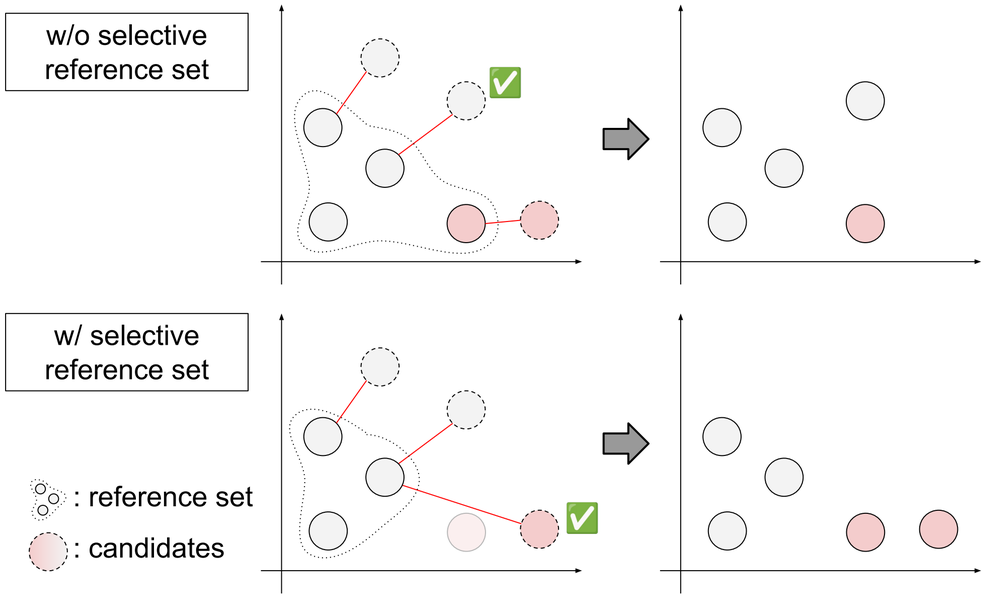
(Even more important than in traditional SW because…)

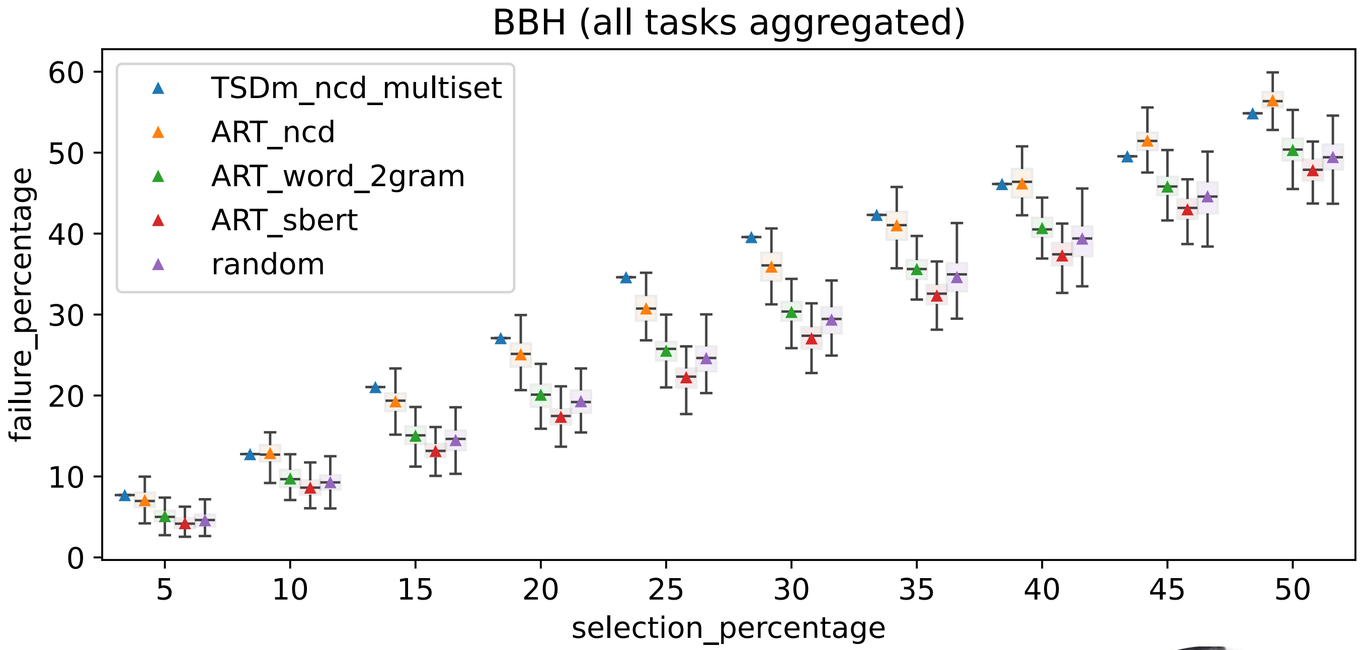
How do we develop and test these agents?
(given that we are dependent on foundational models)

- **BET**: we will all be doing TDD (for now no other ways of anticipating behavior of foundation models)
  - Step 1: design agent architecture, and fill in prompts
  - Step 2: build a reference input set
  - Step 3: iterate over step 1 & 2 until we reach a satisfactory fixpoint
- Question: how do we choose the NEXT test input?
  - Answer: measure similarity to the reference (=working) input set

Diversity-based Testing of LLM SW Systems

To appear at NEXTA@ICST 2025 (https://arxiv.org/abs/2501.13480)





Juyeon Yoon
(PhD Candidate)

Then the inputs initiates the agent behavior.

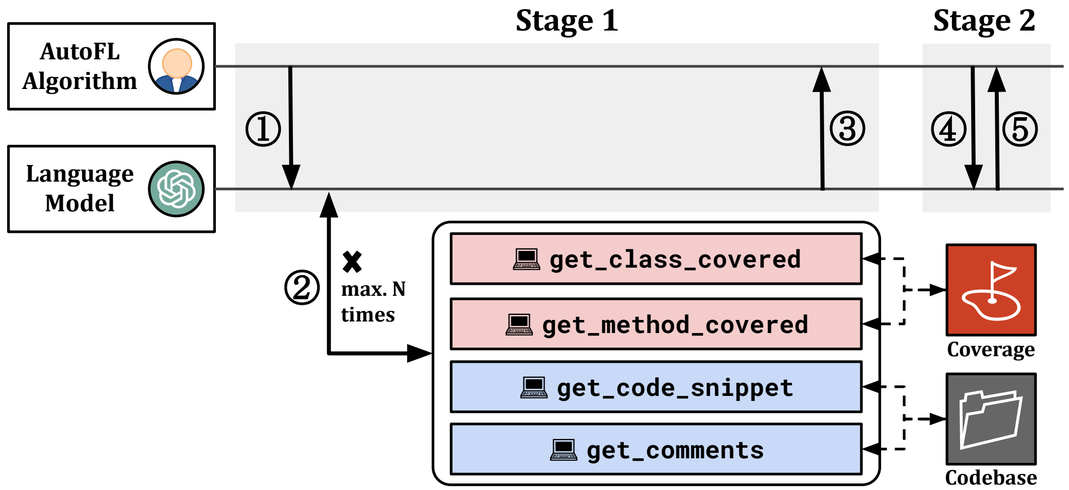
(How do we capture and represent the LLM agent execution?)

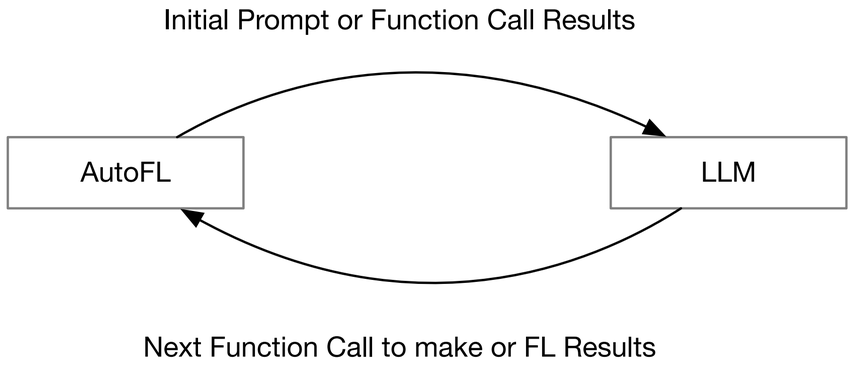
How do we represent LLM agents as a system?

- No taxation without representation, but also **no analysis without representation** 😎
- However, an execution of LLM agents has the following restrictions:
  - Important decision points are made by black boxes (LLMs)
  - A single run may not represent the true capability of the system
  - Input/output are unstructured

Let’s consider AutoFL as an example

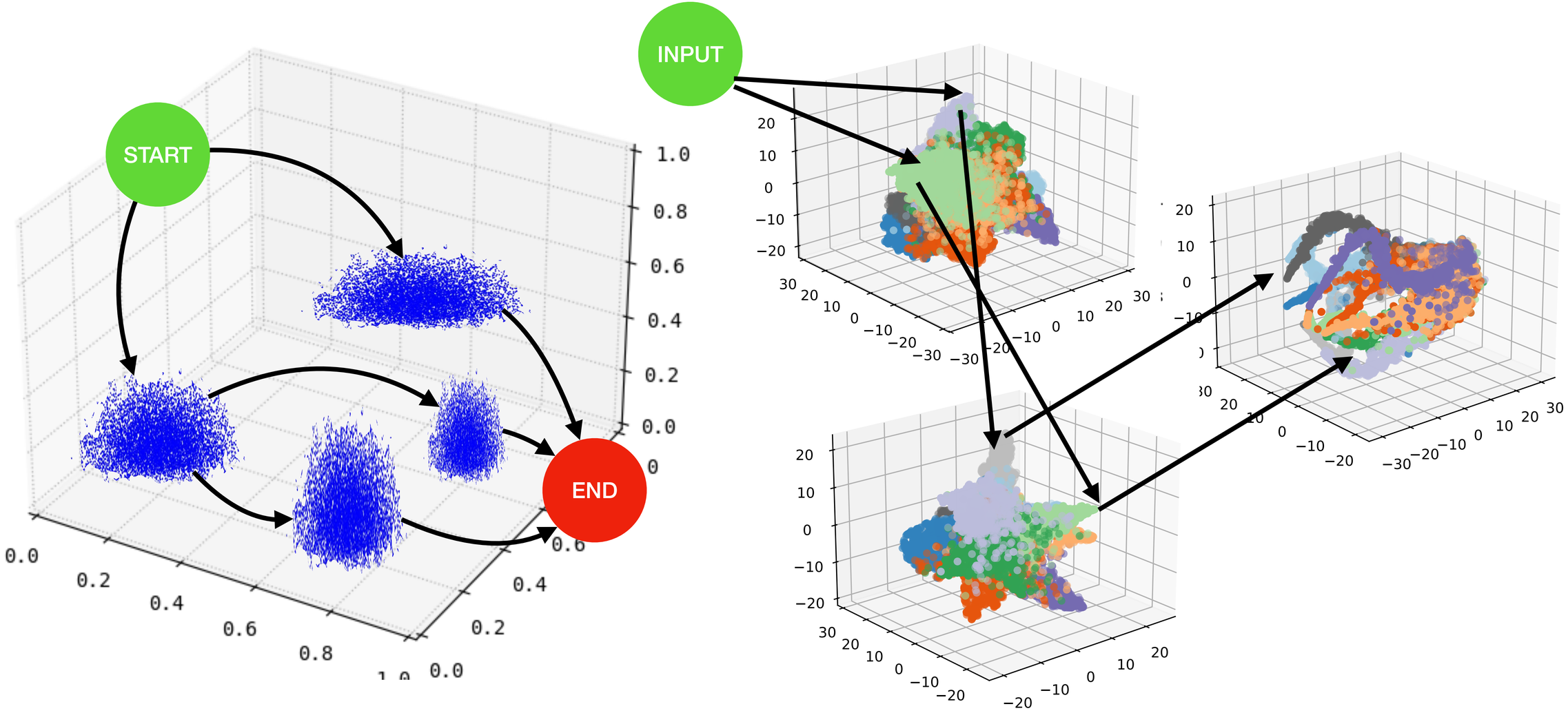
- One inference run of AutoFL is essentially a sequence of tool uses (i.e., function calls)
- After a fixed number of tool uses, it is expected to make a response
- This gives us some sense of control flow; however, traditional control flow does not reveal much.





~~Control~~ ~~Data~~ Semantic Flow

Graph Representation for Executions of LLM Agents (under review)



“Okay, but what use is it?”

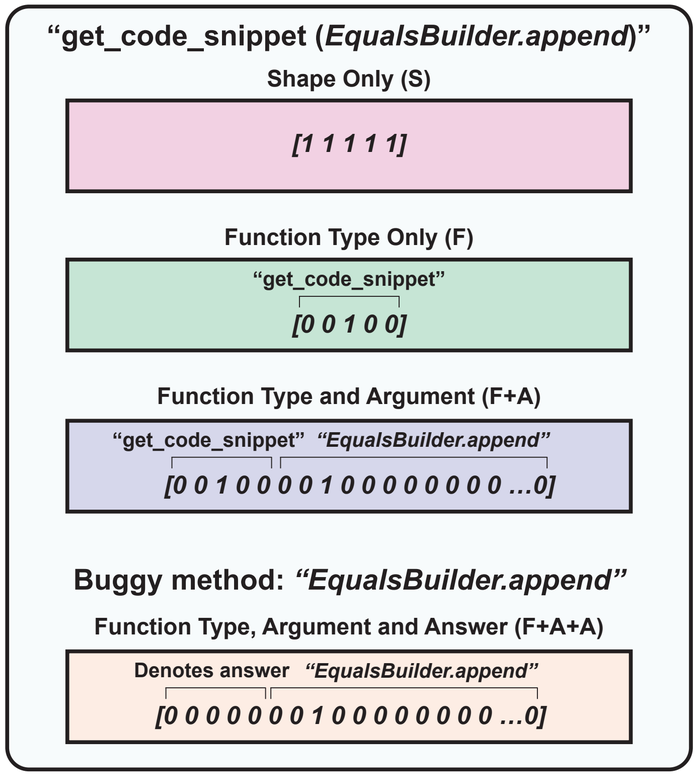
Wang's Assumption about Self Consistency

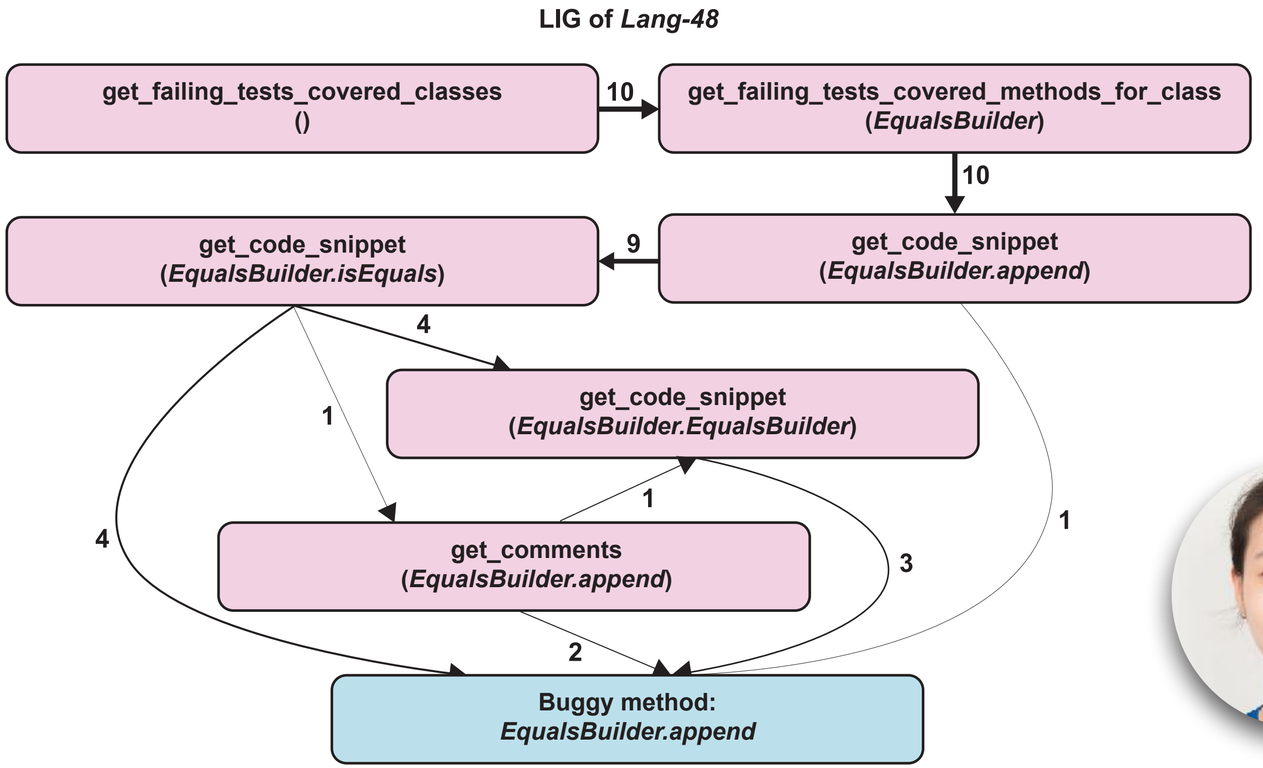
- Wang et al.'s original intuition: "there are many reasoning paths to the correct solutions, but only one way to arrive at a specific incorrect solution"
- My first reaction: "surely there are infinite ways to arrive at a single incorrect solution!"
- My second reaction: "oh, it is probably assumed 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*"

A concrete example for AutoFL

(https://arxiv.org/abs/2412.08281)

Lachesis: Predicting LLM Inference Accuracy using Structural Properties of Reasoning Paths





Naryeong Kim
(MSc Candidate)

Evaluating Wang et al. Hypothesis

Can we predict whether an LLM inference run is correct?

Naryeong Kim
(MSc Candidate)

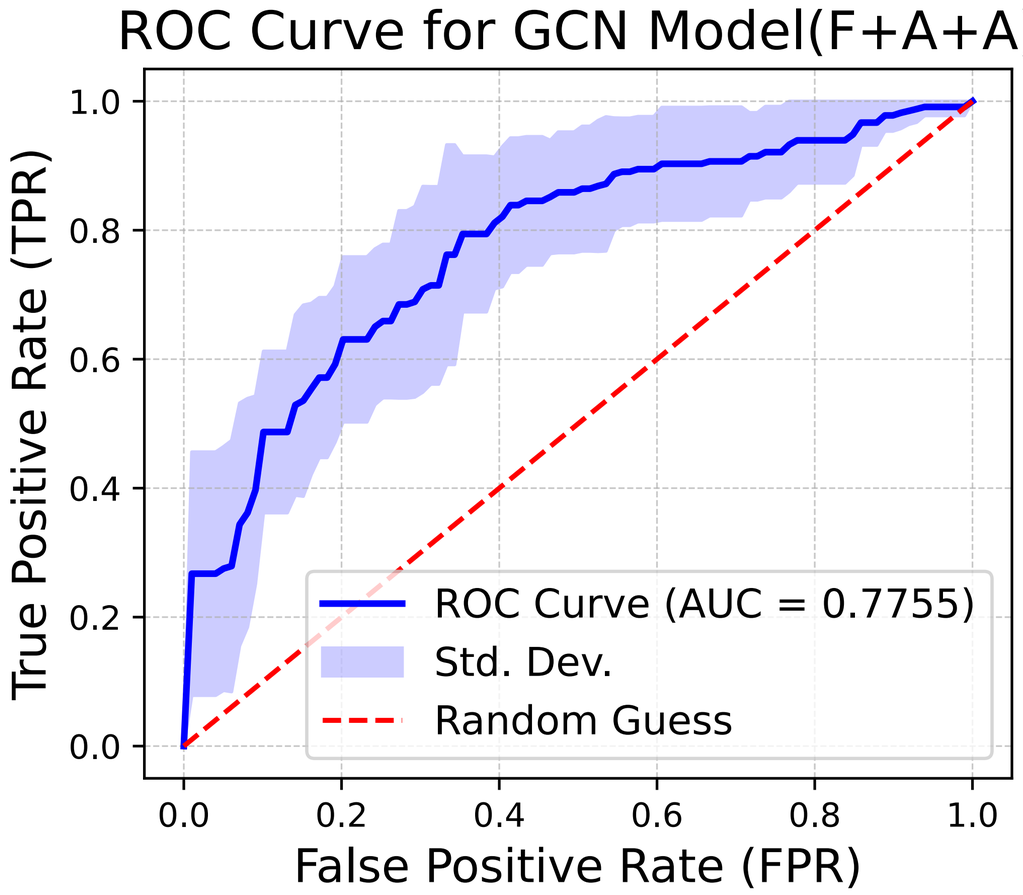


TABLE II: Performance of Prediction Models and Baselines

Method		Accuracy	ROC-AUC	Precision	Recall
LSTM	F	0.7063	0.7356	0.7570	0.8302
	F+A	0.7149	0.6870	0.7662	0.8268
	F+A+A	0.7191	0.7557	0.7711	0.8272
GCN	S	0.6900	0.7791	0.7323	**0.9182**
	F	0.7235	0.7866	0.7751	0.8524
	F+A	0.7454	0.7723	0.8022	0.8332
	F+A+A	0.7454	0.7755	0.8136	0.8172
AutoFL Conf.		**0.7610**	**0.8193**	**0.8173**	0.8306
Baseline		0.6732	-	0.6732	1.0000

The final fate cuts the thread.

(What does it mean to terminate an LLM inference?)

Cost-Effectiveness Analysis of LLM Agents

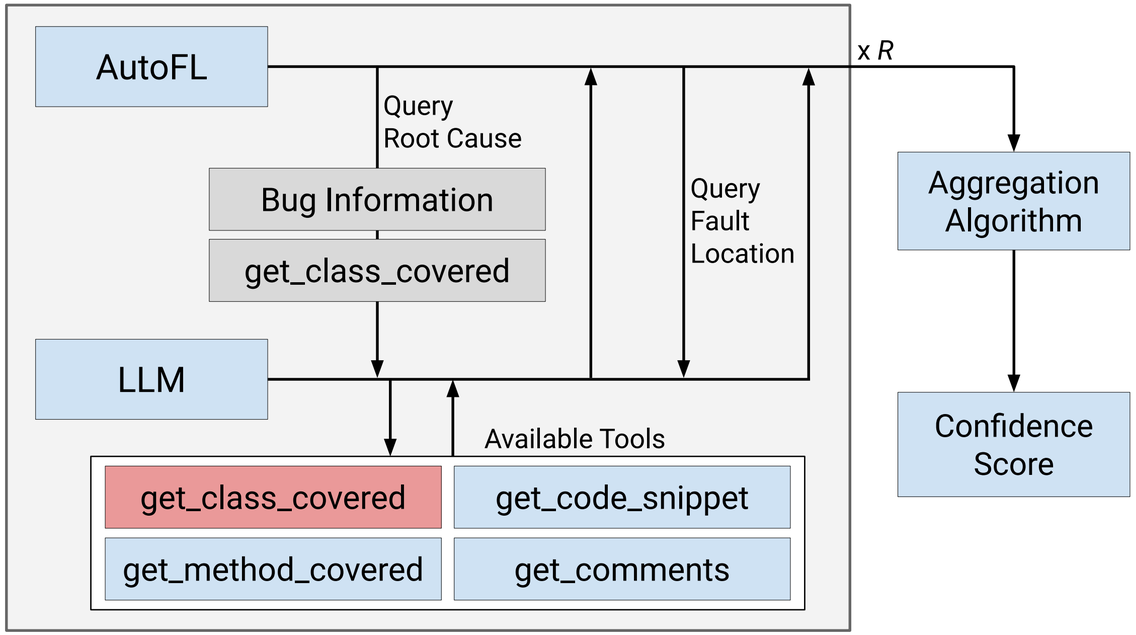
- Self-consistency improves performance, but also increases the cost.
- Dependence on closed-source LLM can be a deal-breaker.
- What is the cost-benefit trade-off between more powerful/expensive/ resource-hungry large models and weaker/free/lightweight open source models?

COSMOS: Collection of SLMs

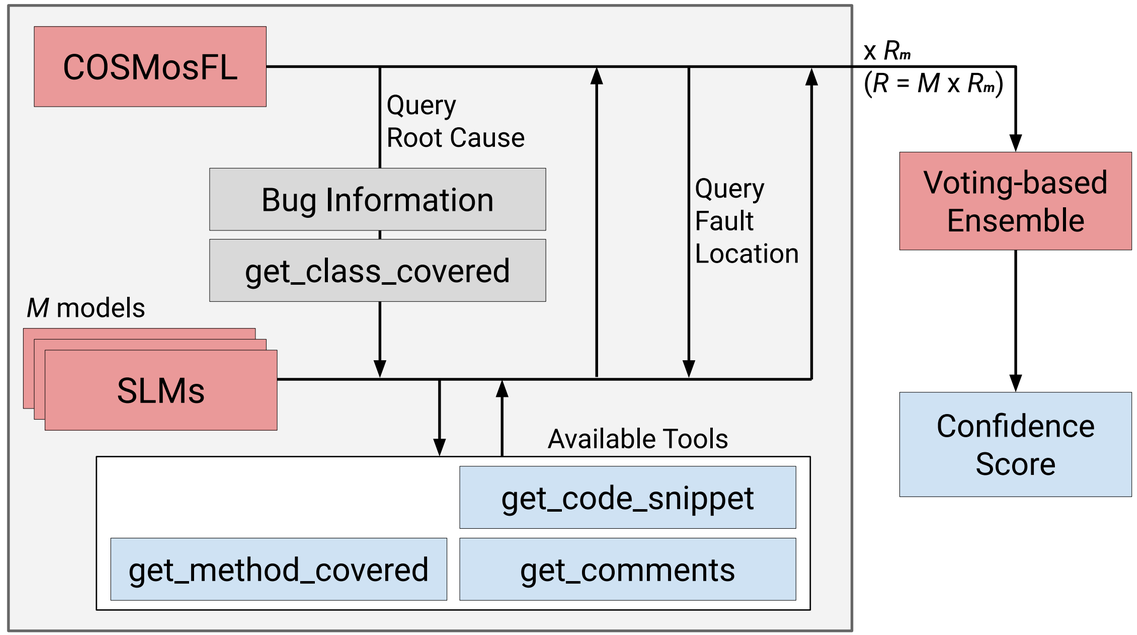
LLM4Code 2025 (https://arxiv.org/abs/2502.02908)

Hyunjoon Cho
(MSc Candidate)

(a) AutoFL



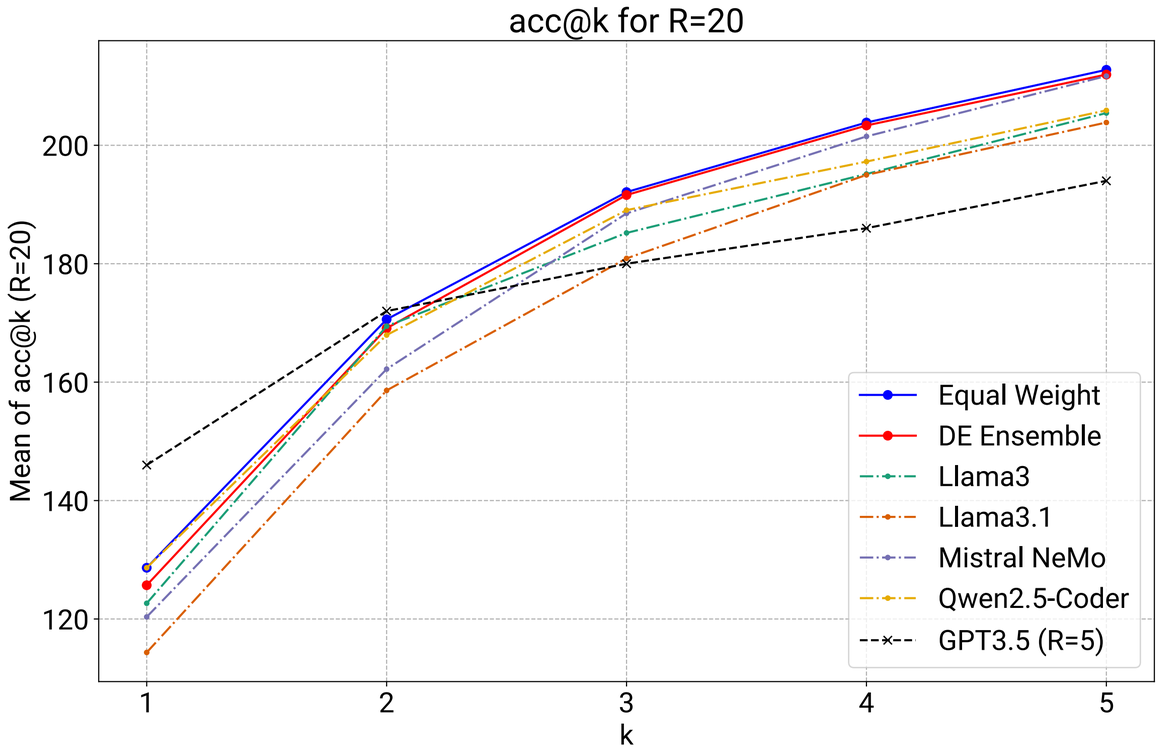
(b) COSMosFL

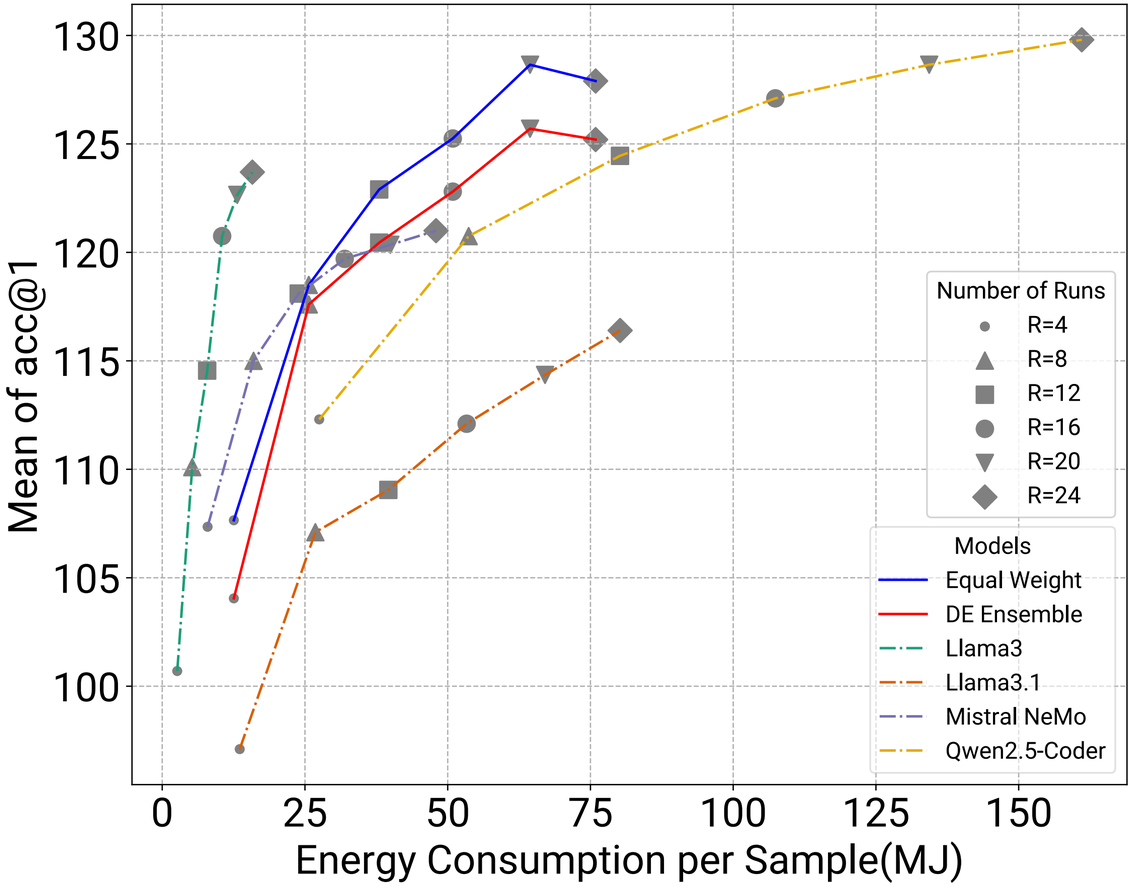


COSMOS w/ AutoFL

LLM4Code 2025 (https://arxiv.org/abs/2502.02908)

Hyunjoon Cho
(MSc Candidate)



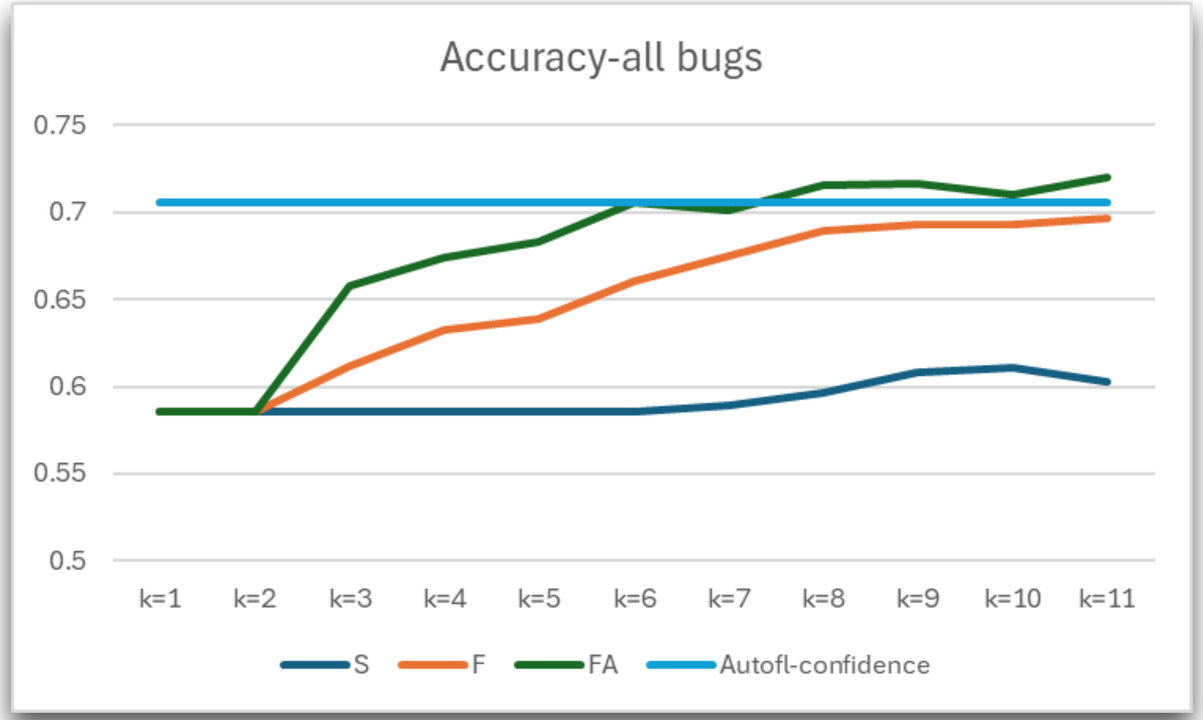


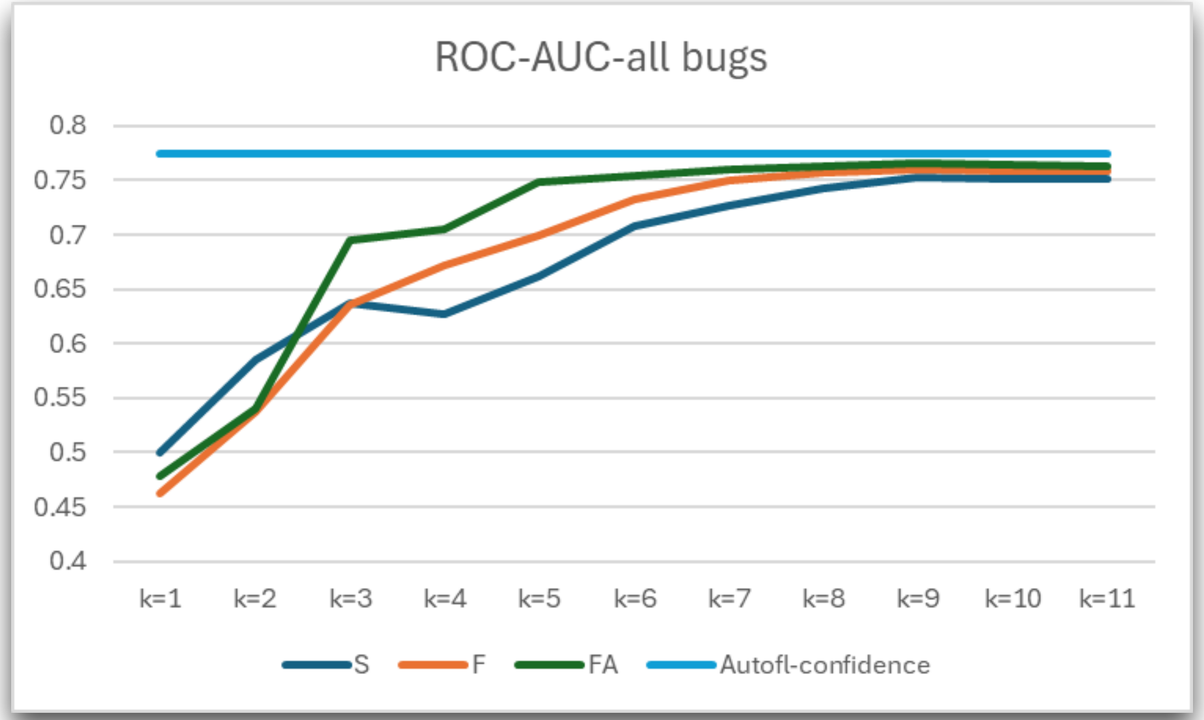
Can we get simpler than ensembles?

- Can we simply decide whether to use a local LLM, or to invoke a remote, closed-source LLM?
- In other words, is the current problem easy enough for the lighter model to solve, or hard enough to involve the heavier model?
- Voting-based aggregation allows us to mix-up inferences from different LLMs!
- Lachesis/Atropos can play an important role :)

Early Termination Early Results

After half the steps, it is reasonably accurate.





Conclusion

- There is a new type of software system emerging: a hybrid between semantic reasoning of LLMs and tools and actions of traditional SW.
- Developing these systems will require a very different approach from the established SE practices.
- We cannot rely on foundational models indefinitely getting better - we need to formulate ways to analyze the behavior of new systems, and eventually to refine their design and optimize their performance.