

Morescient Generative AI for Software Engineering



Marcus Kessel <marcus.kessel@uni-mannheim.de> and Colin Atkinson



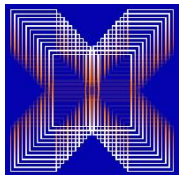
Aiware '24 <[arXiv:2406.04710](https://arxiv.org/abs/2406.04710)>

Generative AI for Software Engineering Tasks

- (Code) LLMs trained on massive amounts of open source code
- Tasks like code generation, test generation and many others ...



ChatGPT



Codex

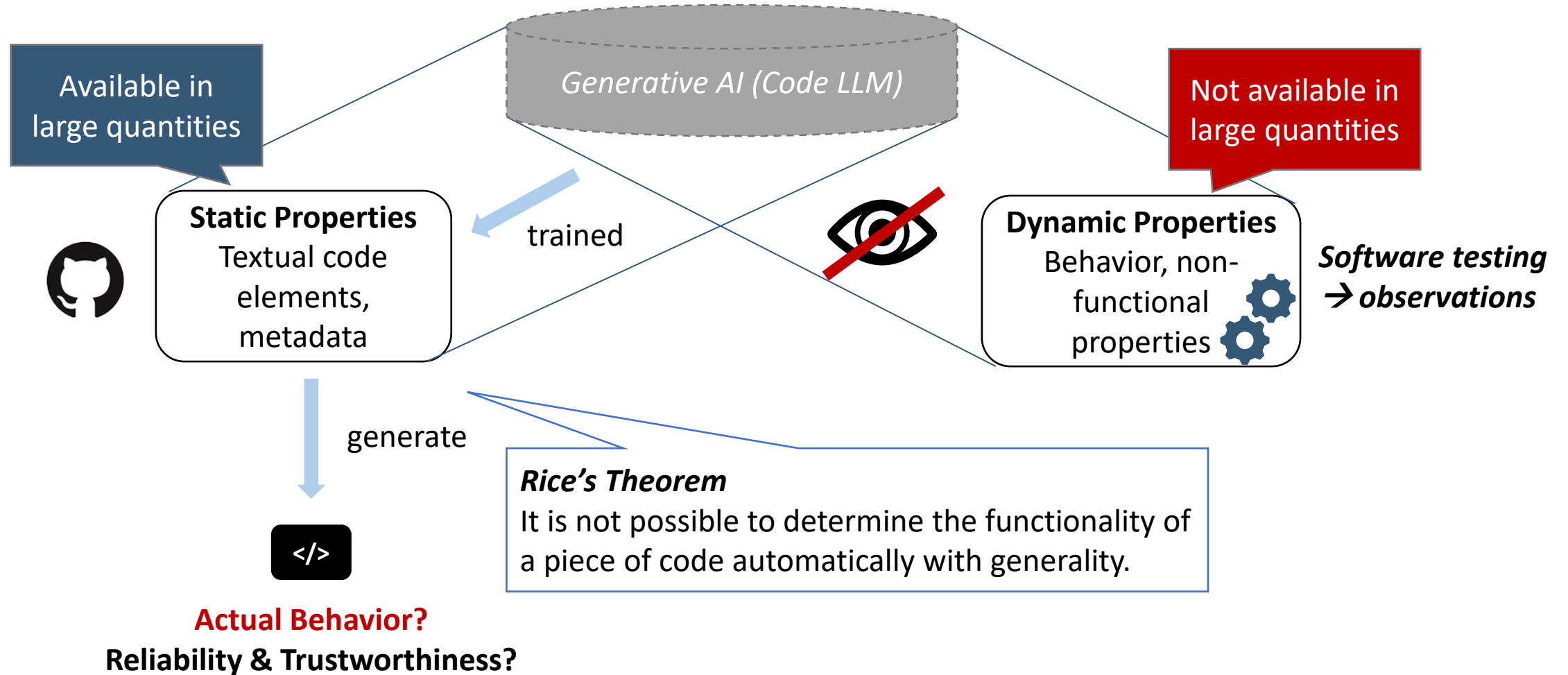


... many more (Huggingface, EvalPlus Leaderboard etc.)

```
1 import datetime
2
3 def parse_expenses(expenses_string):
4     """Parse the list of expenses and return the list of triples (date, value, currency).
5     Ignore lines starting with #.
6     Parse the date using datetime.
7     Example expenses_string:
8     2016-01-02 -34.01 USD
9     2016-01-03 2.50 DKK
10    2016-01-03 -2.72 EUR
11    ---
12    expenses = []
13    for line in expenses_string.splitlines():
14        if line.startswith("#"):
15            continue
16        date, value, currency = line.split(" ")
17        expenses.append((datetime.datetime.strptime(date, "%Y-%m-%d"),
18                        float(value),
19                        currency))
20    return expenses
```

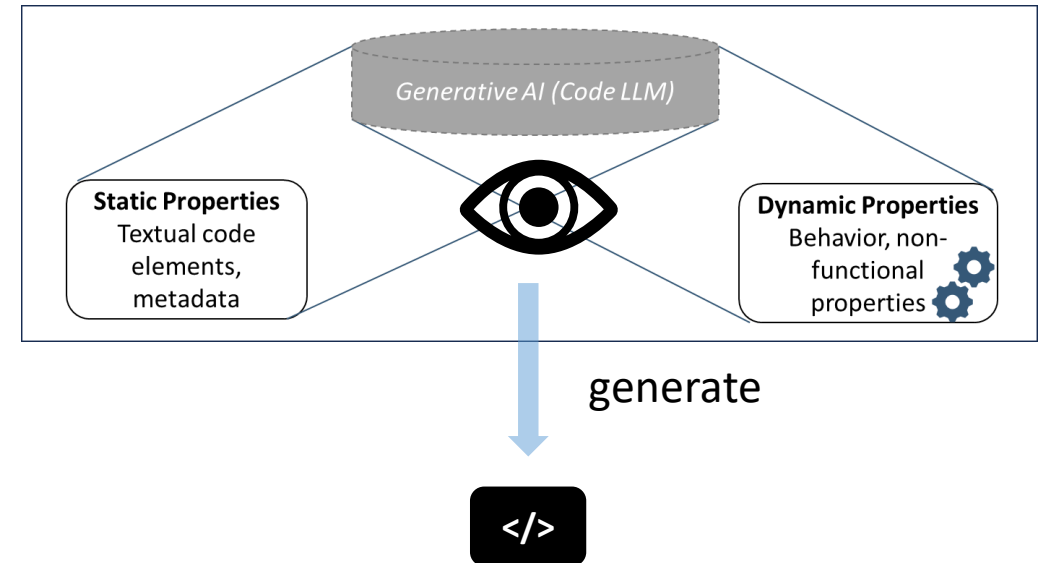
```
1 import static org.junit.Assert.*;
2 import org.junit.Test;
3
4 public class IsPrimeTest {
5
6     // Math.isPrime(int) returns whether the given number is prime or not
7     @Test
8     public void testIsPrime() {
9         assertTrue(Math.isPrime(2));
10        assertTrue(Math.isPrime(3));
11        assertTrue(Math.isPrime(5));
12        assertTrue(Math.isPrime(7));
13        assertTrue(Math.isPrime(11));
14        assertTrue(Math.isPrime(13));
15        assertTrue(Math.isPrime(17);
16        assertTrue(Math.isPrime(19);
17        assertTrue(Math.isPrime(23);
18        assertTrue(Math.isPrime(29);
19    }
20 }
```

Training – Blindness to “De Facto” Behavior



→ Dilemma: Production gains vs V&V effort

- “Morescient Generative AI”
 - more = behavior
 - scire = to know
 - Idea: Complement syntactic data with behavioral data
 - Similar to synergistic combination of static + dynamic program analysis
 - Practitioners & researchers can enhance and assess code LLMs
 - Morescient analysis services (building SE tools)
 - Test-driven software experimentation
 - Morescient data sets (training)
- Increased utility (quality)



Towards Generation of Large Behavioral Data

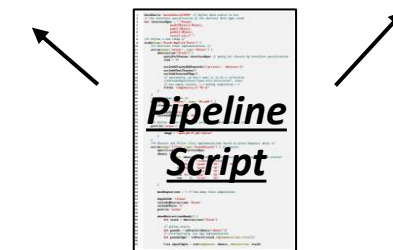
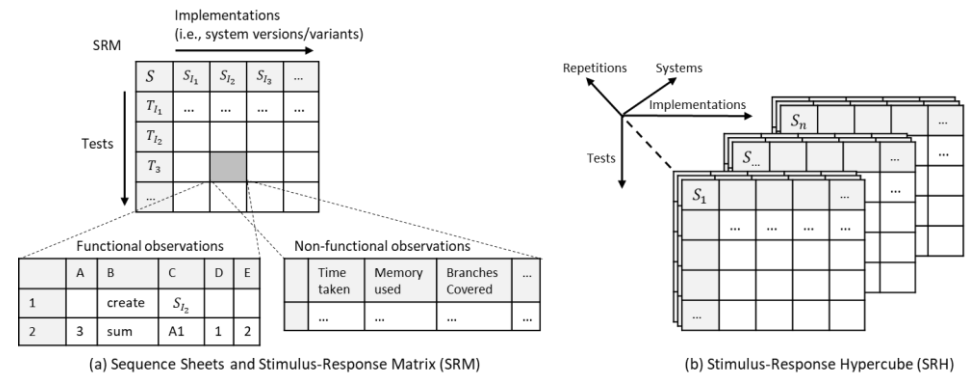
- Large-scale Software Observatory like LASSO [1] required
 - provides simple, transparent, domain-specific languages and data structures to create, analyze and store behavioral data at ultra-large scales (i.e. observations)

Data structures (tabular)

- Sequence Sheets
- Stimulus Response Matrices (SRMs)
- Stimulus Response Hypercubes (SRHs)

Languages

- LASSO Scripting Language (LSL)
- Sequence Sheet Notation (SSN)



[1] <https://softwareobservatorium.github.io/>

Future Directions and Challenges



- Short-term collection
 - (Automated) test-driven assessment/enhancement of code LLMs
 - Retrieval augment generation (external fact checking)
 - Prompting (domain knowledge)
 - Tool support: Analysis services (test generation, oracle recommendation ...)
- Long-term collection
 - Training: Large, open data sets as training data complementary to code
- Challenges
 - Scalability
 - Limitations of software testing (dynamic analysis)
 - ...

Thank you!

Resources

- Kessel, Marcus, and Colin Atkinson. "Morescient GAI for Software Engineering." arXiv preprint arXiv:2406.04710 (2024).
- Kessel, Marcus, and Colin Atkinson. "Promoting open science in test-driven software experiments." Journal of Systems and Software 212 (2024):111971. <https://doi.org/10.1016/j.jss.2024.111971>
- LASSO Platform, <https://softwareobservatorium.github.io/>