STARS: Static Analysis-Guided Assertion Synthesis using Large Language Models

Jialun Cao[*], Haoyu Wang[$], Haoran Yan[$], Ming Wen[$], Michael Pradel[#]

[*] HKUST → Imperial College London, [$] HUST, [#] CISPA

Assertion Synthesis Problem

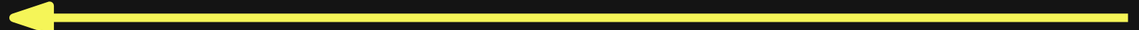
```
@Test
public void testGetTextElements() {
  Cell cell = new Cell(0, 0, 0, 0);
  TextElement tElement = new TextElement(0, 0, 0, 0,
    new PDType1Font(Standard14Fonts.FontName.HELVETICA_BOLD), 10, "test", 5);
  TextChunk tChunk = new TextChunk(tElement);
  List<TextChunk> tList = new ArrayList<>();
  tList.add(tChunk);
  cell.setTextElements(tList);

}
```

Given: Test prefix
(optionally, also focal method or class)

Assertion Synthesis Problem

```
@Test
public void testGetTextElements() {
  Cell cell = new Cell(0, 0, 0, 0);
  TextElement tElement = new TextElement(0, 0, 0, 0,
    new PDType1Font(Standard14Fonts.FontName.HELVETICA_BOLD), 10, "test", 5);
  TextChunk tChunk = new TextChunk(tElement);
  List<TextChunk> tList = new ArrayList<>();
  tList.add(tChunk);
  cell.setTextElements(tList);
  // ???????
}
```



Want: One or more assertions

Assertion Synthesis Problem

```
@Test
public void testGetTextElements() {
  Cell cell = new Cell(0, 0, 0, 0);
  TextElement tElement = new TextElement(0, 0, 0, 0,
    new PDType1Font(Standard14Fonts.FontName.HELVETICA_BOLD), 10, "test", 5);
  TextChunk tChunk = new TextChunk(tElement);
  List<TextChunk> tList = new ArrayList<>();
  tList.add(tChunk);
  cell.setTextElements(tList);
  assertEquals("test", cell.getTextElements().get(0).getText());
}
```

Ground truth (written by developer)

Assertion Synthesis Problem

```
@Test
public void testGetTextElements() {
  Cell cell = new Cell(0, 0, 0, 0);
  TextElement tElement = new TextElement(0, 0, 0, 0,
    new PDType1Font(Standard14Fonts.FontName.HELVETICA_BOLD), 10, "test", 5);
  TextChunk tChunk = new TextChunk(tElement);
  List<TextChunk> tList = new ArrayList<>();
  tList.add(tChunk);
  cell.setTextElements(tList);
  assertEquals("test", cell.getTextElements().get(0).getText());
}
```

How to guess this complex expression?

Ground truth (written by developer)

Assertion Synthesis Problem

```
@Test
public void testGetTextElements() {
  Cell cell = new Cell(0, 0, 0, 0);
  TextElement tElement = new TextElement(0, 0, 0, 0,
    new PDType1Font(Standard14Fonts.FontName.HELVETICA_BOLD), 10, "test", 5);
  TextChunk tChunk = new TextChunk(tElement);
  List<TextChunk> tList = new ArrayList<>();
  tList.add(tChunk);
  cell.setTextElements(tList);
  assertEquals("test", cell.getTextElements().get(0).getText());
}
```

How to guess this complex expression?

How to even know that this method exists?

Ground truth (written by developer)

Previous Approaches

- **Pre-LLM**
  - Static analysis + heuristics
  - Rule-based inference from documentation
  - Specialized neural models
- **Prompt LLM with**
  - Test prefix
  - Focal method or class
  - Documentation

Previous Approaches

- **Pre-LLM**
  - Static analysis + heuristics
  - Rule-based inference from documentation
  - Specialized neural models

Unrealistic or incorrect assertions

- **Prompt LLM with**
  - Test prefix
  - Focal method or class
  - Documentation

Previous Approaches

- **Pre-LLM**
  - Static analysis + heuristics
  - Rule-based inference from documentation
  - Specialized neural models

Unrealistic or incorrect assertions

- **Prompt LLM with**
  - Test prefix
  - Focal method or class
  - Documentation

Insufficient context limits success

Previous Approaches

- **Pre-LLM**
  - Static analysis + heuristics
  - Rule-based inference from documentation
  - Specialized neural models
- **Prompt LLM with**
  - Test prefix
  - Focal method or class
  - Documentation

Unrealistic or incorrect assertions

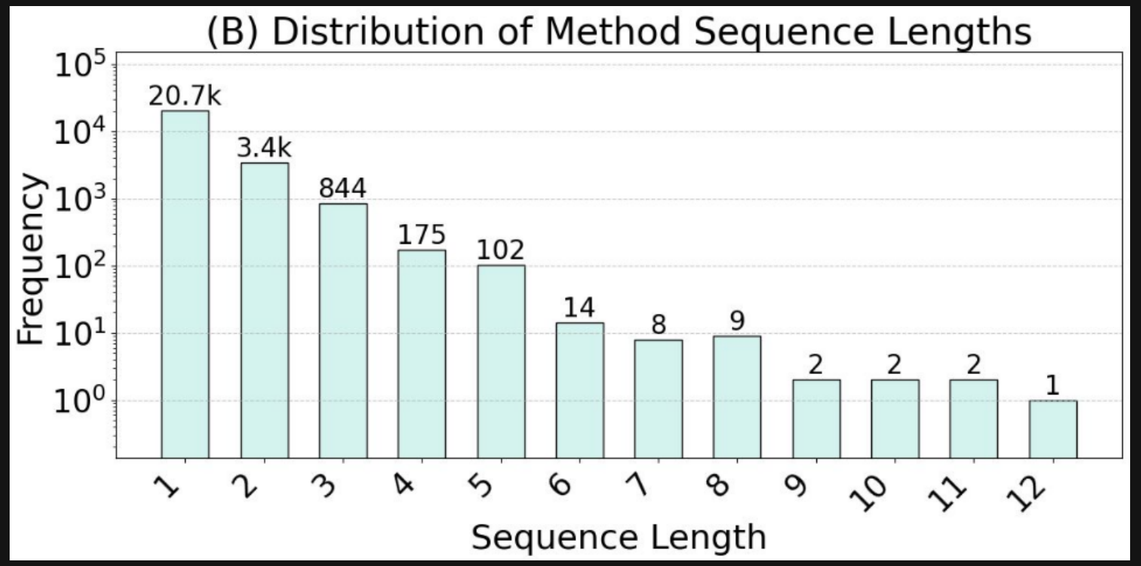
Insufficient context limits success

Huge search space

Characteristics of Real-World Assertions

Preliminary study of 32.7k assertions *

Assertions	Nb.	Ratio
With variables only	3.7k	11.3%
With method calls	25.3k	77.4%
incl. arguments	13.6k	41.5%
to JDK API	14.6k	44.5%

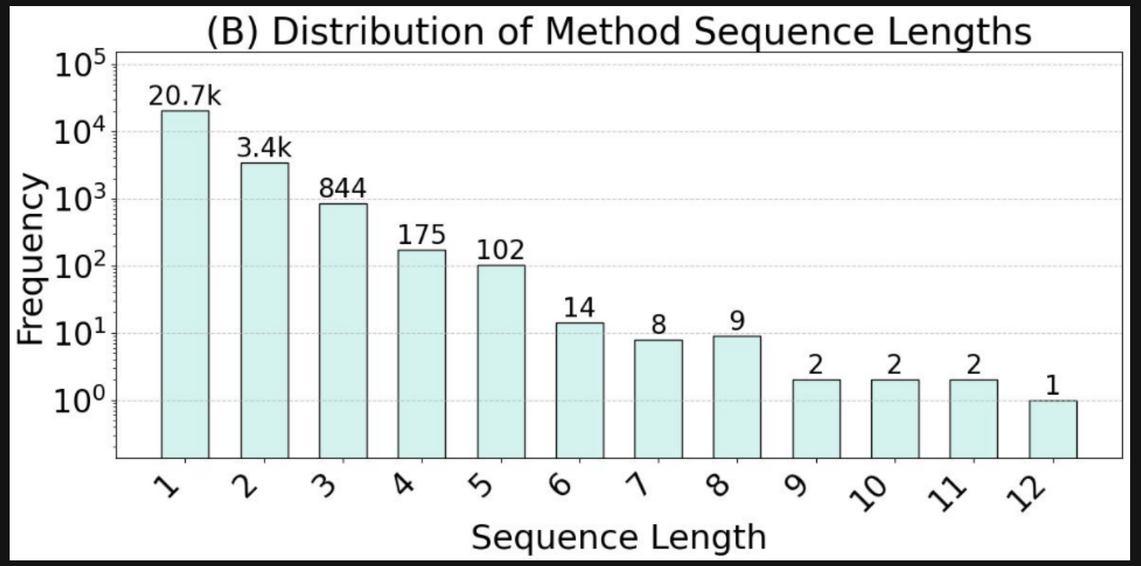


* Based on *Methods2Test* dataset, Tufano et al., 2020

Characteristics of Real-World Assertions

Preliminary study of 32.7k assertions *

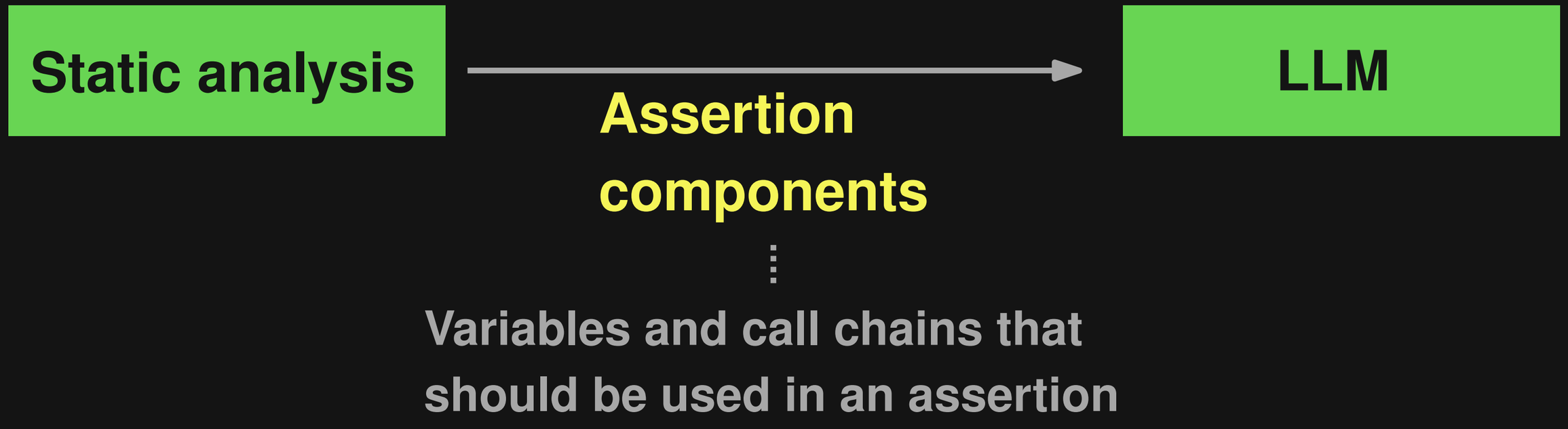
Assertions	Nb.	Ratio
With variables only	3.7k	11.3%
With method calls	25.3k	77.4%
incl. arguments	13.6k	41.5%
to JDK API	14.6k	44.5%



Many assertions: Almost impossible for an LLM to guess

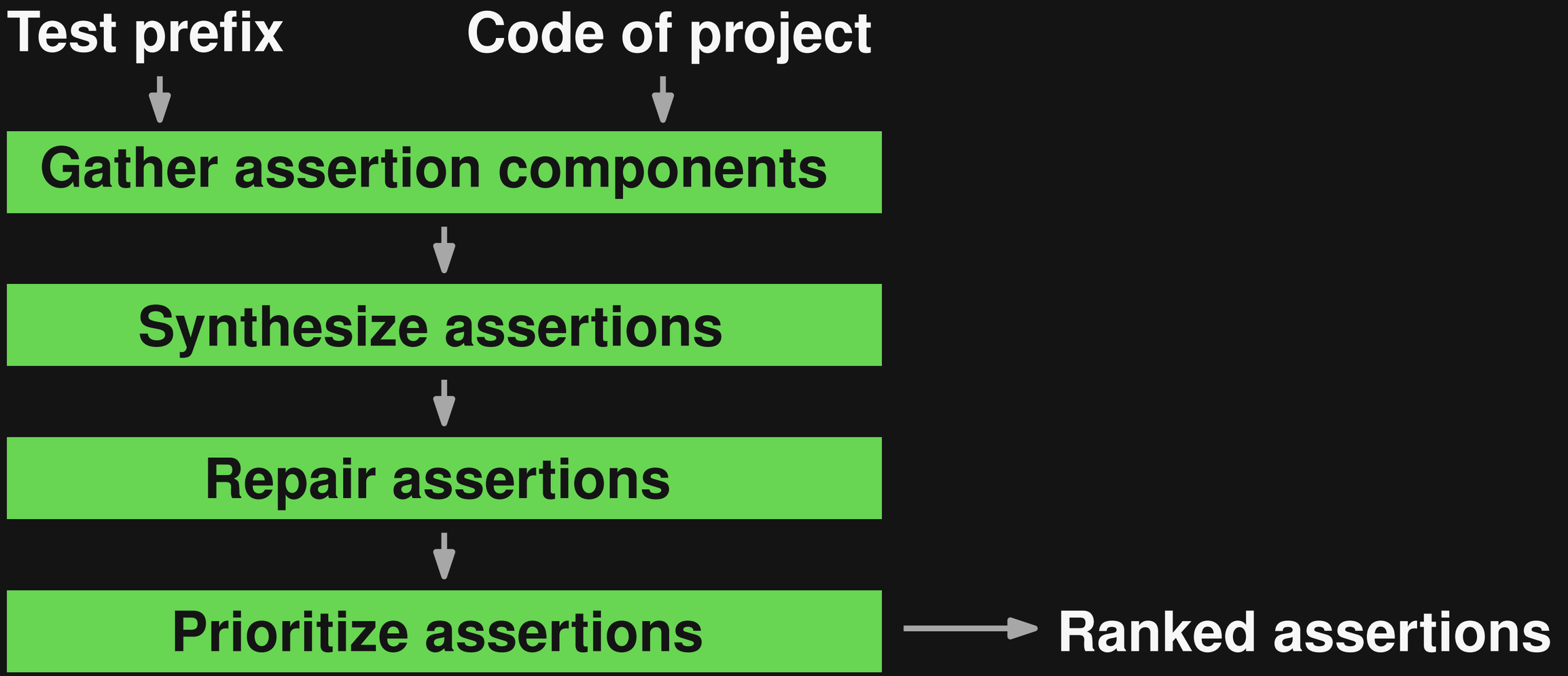
* Based on *Methods2Test* dataset, Tufano et al., 2020

Key Idea: Assertion Components



Reduces search space & provides missing context to LLM

Overview of STARS



Gathering Assertion Components

Three kinds of assertion components:

- **Variables** → `assertNotNull(x)`
- **Pairs of variables** → `assertEquals(expX, actX)`
- **Method call sequences** → `assertEquals("ab", x.getY().getZ())`

Gathering Assertion Components

Three kinds of assertion components:

- **Variables** → `assertNotNull(x)`
- **Pairs of variables** → `assertEquals(expX, actX)`
- **Method call sequences** → `assertEquals("ab", x.getY().getZ())`

Challenge: Naive approach yields huge set of components

Assertion Components: Variables

```
public class CellTest {
  private final Cell initialCell = ...;

  @Test
  public void testGetTextElements() {
    Cell cell = new Cell(0, 0, 0, 0);
    TextElement tElement = new TextElement(0, 0, 0, 0,
    new PDType1Font(Standard14Fonts.FontName.HELVETICA_BOLD), 10, "test", 5);
    TextChunk tChunk = new TextChunk(tElement);
    List<TextChunk> tList = new ArrayList<>();
    tList.add(tChunk);
    cell.setTextElements(tList);
    // assertion?
}}
```

Assertion Components: Variables

```
public class CellTest {
  private final Cell initialCell = ...;

  @Test
  public void testGetTextElements() {
    Cell cell = new Cell(0, 0, 0, 0);
    TextElement tElement = new TextElement(0, 0, 0, 0,
    new PDType1Font(Standard14Fonts.FontName.HELVETICA_BOLD), 10, "test", 5);
    TextChunk tChunk = new TextChunk(tElement);
    List<TextChunk> tList = new ArrayList<>();
    tList.add(tChunk);
    cell.setTextElements(tList);
    // assertion?
}}
```

Extract variables and their types

Assertion Components: Variables

```
public class CellTest {
  private final Cell initialCell = ...;

  @Test
  public void testGetTextElements() {
    Cell cell = new Cell(0, 0, 0, 0);
    TextElement tElement = new TextElement(0, 0, 0, 0,
    new PDType1Font(Standard14Fonts.FontName.HELVETICA_BOLD), 10, "test", 5);
    TextChunk tChunk = new TextChunk(tElement);
    List<TextChunk> tList = new ArrayList<>();
    tList.add(tChunk);
    cell.setTextElements(tList);
    // assertion?
}}
```

Extract variables and their types

Pair variables of same type

Assertion Components: Method Seqs.

```
@Test
public void testGetTextElements() {
  Cell cell = new Cell(0, 0, 0, 0);
  TextElement tElement = new TextElement(0, 0, 0, 0,
    new PDType1Font(Standard14Fonts.FontName.HELVETICA_BOLD), 10, "test", 5);
  TextChunk tChunk = new TextChunk(tElement);
  List<TextChunk> tList = new ArrayList<>();
  tList.add(tChunk);
  cell.setTextElements(tList);
  // assertion?
}
```

Assertion Components: Method Seqs.

Compute type-correct method calls chains

```
@Test
public void testGetTextElements() {
  Cell cell = new Cell(0, 0, 0, 0);
  TextElement tElement = new TextElement(0, 0, 0, 0,
    new PDType1Font(Standard14Fonts.FontName.HELVETICA_BOLD), 10, "test", 5);
  TextChunk tChunk = new TextChunk(tElement);
  List<TextChunk> tList = new ArrayList<>();
  tList.add(tChunk);
  cell.setTextElements(tList);
  // assertion?
}
```

```
cell.setSpanning(boolean)
cell.hasColor(Color)
cell.getTextElements()
```

Assertion Components: Method Seqs.

```
@Test
public void testGetTextElements() {
  Cell cell = new Cell(0, 0, 0, 0);
  TextElement tElement = new TextElement(0, 0, 0, 0,
    new PDType1Font(Standard14Fonts.FontName.HELVETICA_BOLD), 10, "test", 5);
  TextChunk tChunk = new TextChunk(tElement);
  List<TextChunk> tList = new ArrayList<>();
  tList.add(tChunk);
  cell.setTextElements(tList);
  // assertion?
}
```

Compute type-correct method calls chains

Skip methods with side effects

~~`cell.setSpanning(boolean)`~~

`cell.hasColor(Color)`

`cell.getTextElements()`

Assertion Components: Method Seqs.

```
@Test
public void testGetTextElements() {
  Cell cell = new Cell(0, 0, 0, 0);
  TextElement tElement = new TextElement(0, 0, 0, 0,
    new PDType1Font(Standard14Fonts.FontName.HELVETICA_BOLD), 10, "test", 5);
  TextChunk tChunk = new TextChunk(tElement);
  List<TextChunk> tList = new ArrayList<>();
  tList.add(tChunk);
  cell.setTextElements(tList);
  // assertion?
}
```

Compute type-correct method calls chains

Skip methods with side effects

~~cell.setSpanning(boolean)~~

~~cell.hasColor(Color)~~

cell.getTextElements()

Skip methods with unavailable parameters

Assertion Components: Method Seqs.

```
@Test
public void testGetTextElements() {
  Cell cell = new Cell(0, 0, 0, 0);
  TextElement tElement = new TextElement(0, 0, 0, 0,
    new PDType1Font(Standard14Fonts.FontName.HELVETICA_BOLD), 10, "test", 5);
  TextChunk tChunk = new TextChunk(tElement);
  List<TextChunk> tList = new ArrayList<>();
  tList.add(tChunk);
  cell.setTextElements(tList);
  // assertion?
}
```

Compute type-correct method calls chains

Skip methods with side effects

~~`cell.setSpanning(boolean)`~~

~~`cell.hasColor(Color)`~~

`cell.getTextElements()` → `.spliterator()`

`.get(int)`

`List`

Skip methods with unavailable parameters

Assertion Components: Method Seqs.

```
@Test
public void testGetTextElements() {
  Cell cell = new Cell(0, 0, 0, 0);
  TextElement tElement = new TextElement(0, 0, 0, 0,
    new PDType1Font(Standard14Fonts.FontName.HELVETICA_BOLD), 10, "test", 5);
  TextChunk tChunk = new TextChunk(tElement);
  List<TextChunk> tList = new ArrayList<>();
  tList.add(tChunk);
  cell.setTextElements(tList);
  // assertion?
}
```

Compute type-correct method calls chains

Skip methods with side effects

~~cell.setSpanning(boolean)~~

~~cell.hasColor(Color)~~

cell.getTextElements() → ~~.spliterator()~~

.get(int)

List

Focus on most frequent JDK methods

Skip methods with unavailable parameters

Assertion Components: Method Seqs.

```
@Test
public void testGetTextElements() {
  Cell cell = new Cell(0, 0, 0, 0);
  TextElement tElement = new TextElement(0, 0, 0, 0,
    new PDType1Font(Standard14Fonts.FontName.HELVETICA_BOLD), 10, "test", 5);
  TextChunk tChunk = new TextChunk(tElement);
  List<TextChunk> tList = new ArrayList<>();
  tList.add(tChunk);
  cell.setTextElements(tList);
  // assertion?
}
```

Compute type-correct method calls chains

Skip methods with side effects

~~cell.setSpanning(boolean)~~

~~cell.hasColor(Color)~~

cell.getTextElements() → ~~.spliterator()~~

.get(int) → .getText()

List

TextChunk

Focus on most frequent JDK methods

Skip methods with unavailable parameters

Assertion Synthesis

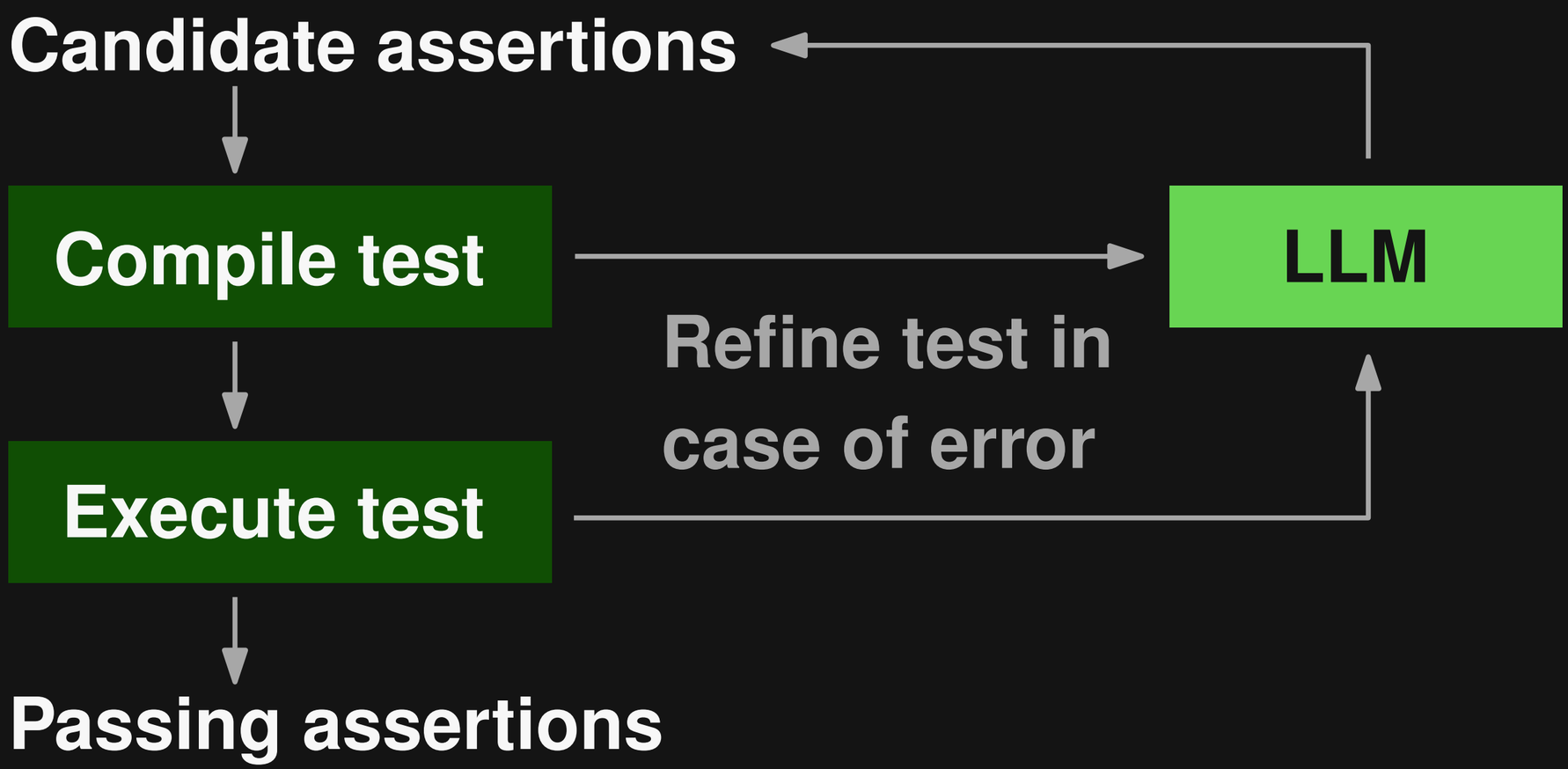
Test prefix

Assertion components:

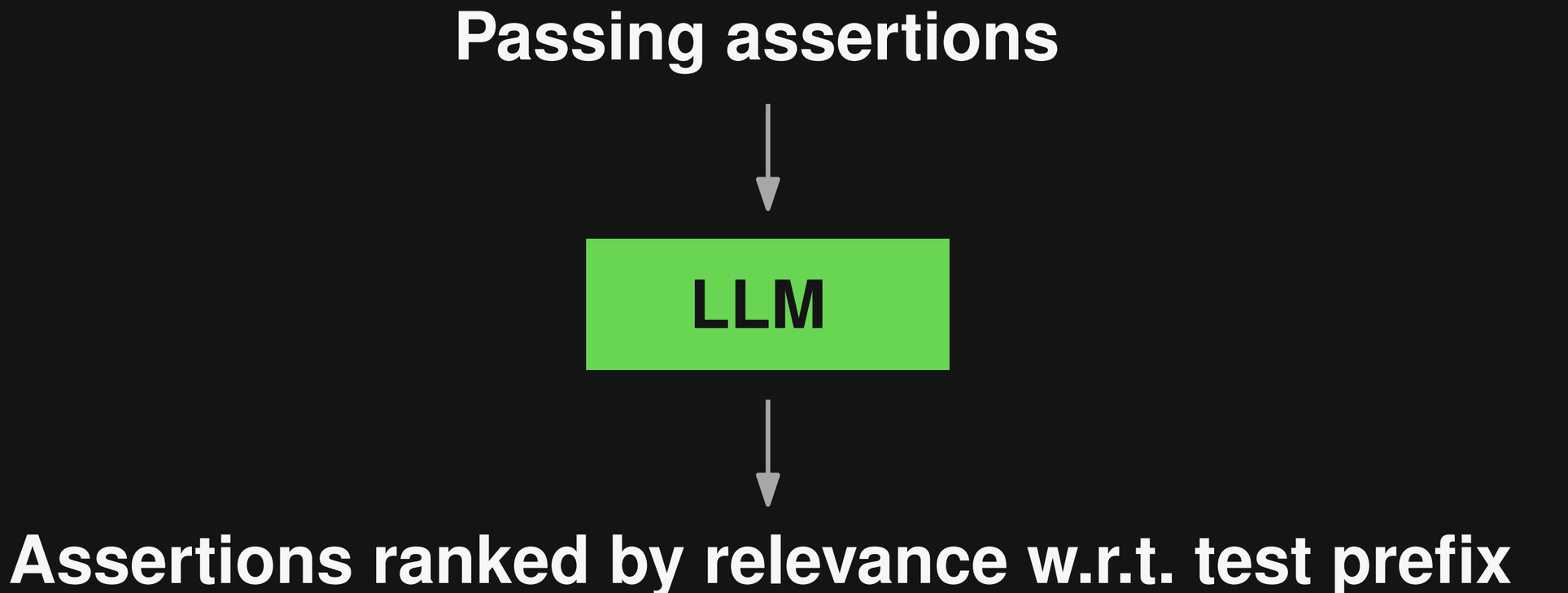
```
initialCell, cell
cell.getSize()
cell.getTextElement().get(_).getText()
tChunk.getText()
```

Most likely assertions built from components

Repairing Assertions



Assertion Prioritization



Evaluation

- **Experimental setup**
  - 500 test prefixes with 1,683 ground truth assertions *
  - LLMs: GPT-3.5-turbo, DeepSeek-V3, GPT-5-mini, GPT-5.4
- **Questions**
  - RQ1: Effectiveness (exact match, test strength, etc.)
  - RQ2: Comparison with ChatAssert baseline
  - RQ3: Ablation study
  - RQ4: Cost analysis

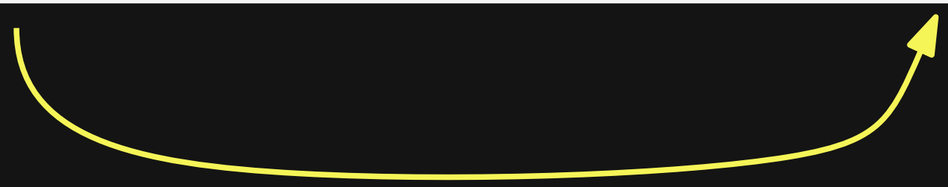
* Augmented version of dataset from *ChatAssert: LLM-Based Test Oracle Generation With External Tools Assistance*, Hayet et al., 2025

Comparison with Frontier LLM

Assertion Per Prefix	Direct Use (GPT-5.4)			STARS (GPT-5.4)		
	W/o Repair	W/ Repair		W/o Repair	W/ Repair	
	Match	Match	Strength	Match	Match	Strength
Top 1	103	103	61.21	183	293	65.8
Top 10	202	208	65.73	285	415	69.53
All	210	216	67.16	306	417	71.43

Comparison with Frontier LLM

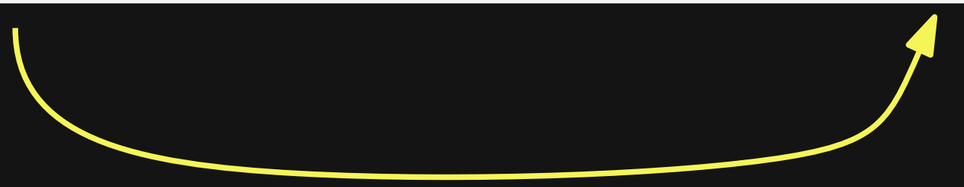
Assertion Per Prefix	Direct Use (GPT-5.4)			STARS (GPT-5.4)		
	W/o Repair	W/ Repair		W/o Repair	W/ Repair	
	Match	Match	Strength	Match	Match	Strength
Top 1	103	103	61.21	183	293	65.8
Top 10	202	208	65.73	285	415	69.53
All	210	216	67.16	306	417	71.43



Matches ground truth more often

Comparison with Frontier LLM

Assertion Per Prefix	Direct Use (GPT-5.4)			STARS (GPT-5.4)		
	W/o Repair	W/ Repair		W/o Repair	W/ Repair	
	Match	Match	Strength	Match	Match	Strength
Top 1	103	103	61.21	183	293	65.8
Top 10	202	208	65.73	285	415	69.53
All	210	216	67.16	306	417	71.43



Assertions kill more mutants

Comparison with Baseline Approach

		ChatAssert	STARS
Exact match	Top-1	119	146
Exact match	Top-10	209	260
Test strength	Top-1	36.9%	49.5%
Test strength	Top-10	45.1%	59.1%

Baseline:

ChatAssert: LLM-Based Test Oracle Generation With External Tools Assistance, Hayet et al., 2025

Comparison with Baseline Approach

		ChatAssert	STARS
Exact match	Top-1	119	146
Exact match	Top-10	209	260
Test strength	Top-1	36.9%	49.5%
Test strength	Top-10	45.1%	59.1%

More effective

(with only 75% token consumption)

Baseline:

ChatAssert: LLM-Based Test Oracle Generation With External Tools Assistance, Hayet et al., 2025

Conclusion

- **Assertion synthesis benefits from both ...**
  - Static analysis $\rightarrow$ Reduces search space
  - LLMs $\rightarrow$ Produces natural, likely correct assertions

More general neuro-symbolic pattern

- **Key idea:**
  Gather assertion components before querying the LLM
- **Increases exact-match rate & mutation score, while reducing token costs**