Once Bitten, Still Shy: Can We Prevent Cloud Systems from **Repeating** Their Mistakes?

Dimas Parikesit, Chang Lou

University of Virginia

HotNets 2025

UNIVERSITY of VIRGINIA

Frequent Updates in Cloud Systems Often Lead to Failures

40,000 commits/day [1]

80,000 commits/day [2]

A Google Cloud Outage Took Down Half The Internet On Thursday

By Jacob Siegal ▪ June 12, 2025 4:17 pm EST

Understanding the Microsoft Teams & Azure Disruptions

By Mike Hicks | February 2, 2024 | 14 min read

Amazon Web Services briefly hit by wide-ranging outage, impacting major websites

By Catherine Thorbecke, CNN

2 min read · Updated 6:56 PM EDT, Tue June 13, 2023

[1] Why Google stores billions of lines of code in a single repository
[2] The Telemetry Data Revolution @ Microsoft

Regression Failures

ZooKeeper / ZOOKEEPER-4773

Ephemeral node is not deleted when all followers are blocked with leader

ZooKeeper / ZOOKEEPER-2355

Ephemeral node is never deleted if follower fails while reading the proposal packet

ZooKeeper / ZOOKEEPER-4837

Network issue cau… session expiration

…started again, but its …r nodes

ZooKeeper / ZOO…

Ephemeral … partition wh…

Issues where violated semantics
have been fixed by developers but are
broken again due to changes in the source code.

ZooKeeper / ZOOKEEPER-4388

Recover from network partition, follower/observer ephemerals nodes is inconsistent with leader

ZooKeeper / ZOOKEEPER-3018

Ephemeral node not deleted after session is gone

ZooKeeper / ZOOKEEPER-2919

expired ephemeral node reappears after ZK leader change

ZooKeeper / ZOOKEEPER-2800

zookeeper ephemeral node not deleted after server restart and consistency is not hold

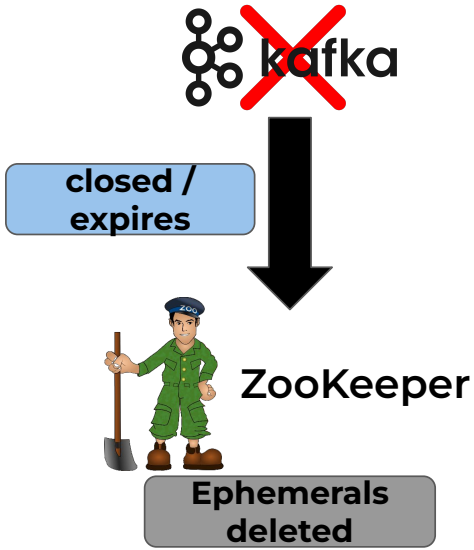
ZooKeeper / ZOOKEEPER-1496

Ephemeral node not getting cleared even after client has exited

Case Study: ZooKeeper Ephemerals Regression Failures

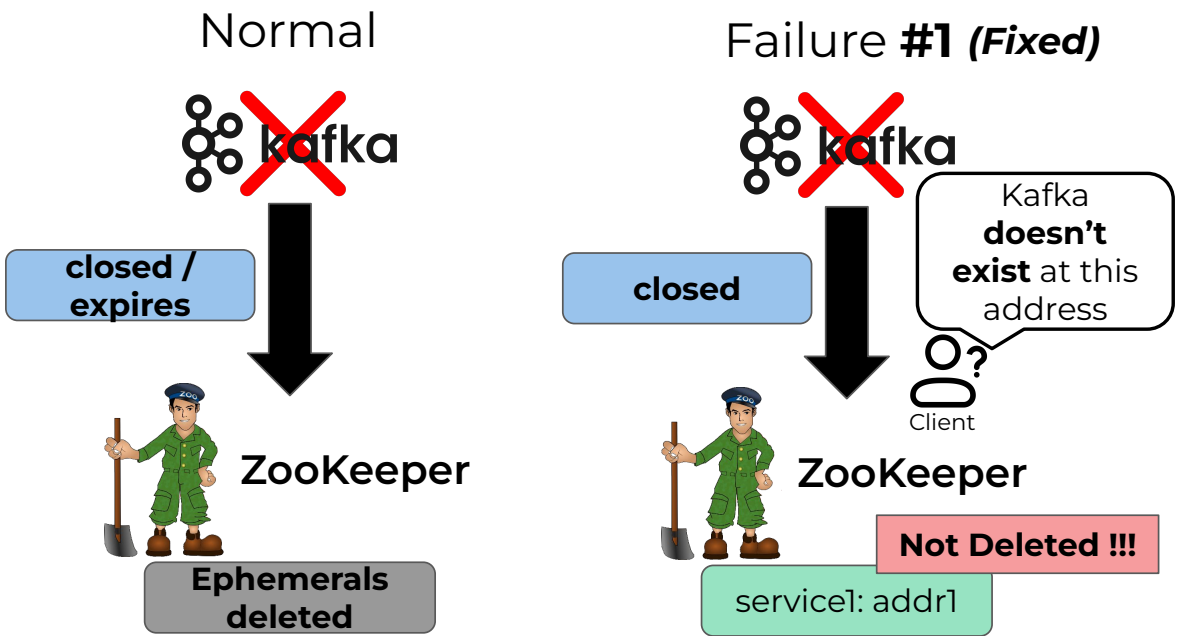
- ZooKeeper Ephemerals: **auto-deleted data records** upon session disconnection (explicitly closed / expires)
- This semantics is repeatedly violated throughout the development cycle
  - **46** violations over the past 14 years

Normal



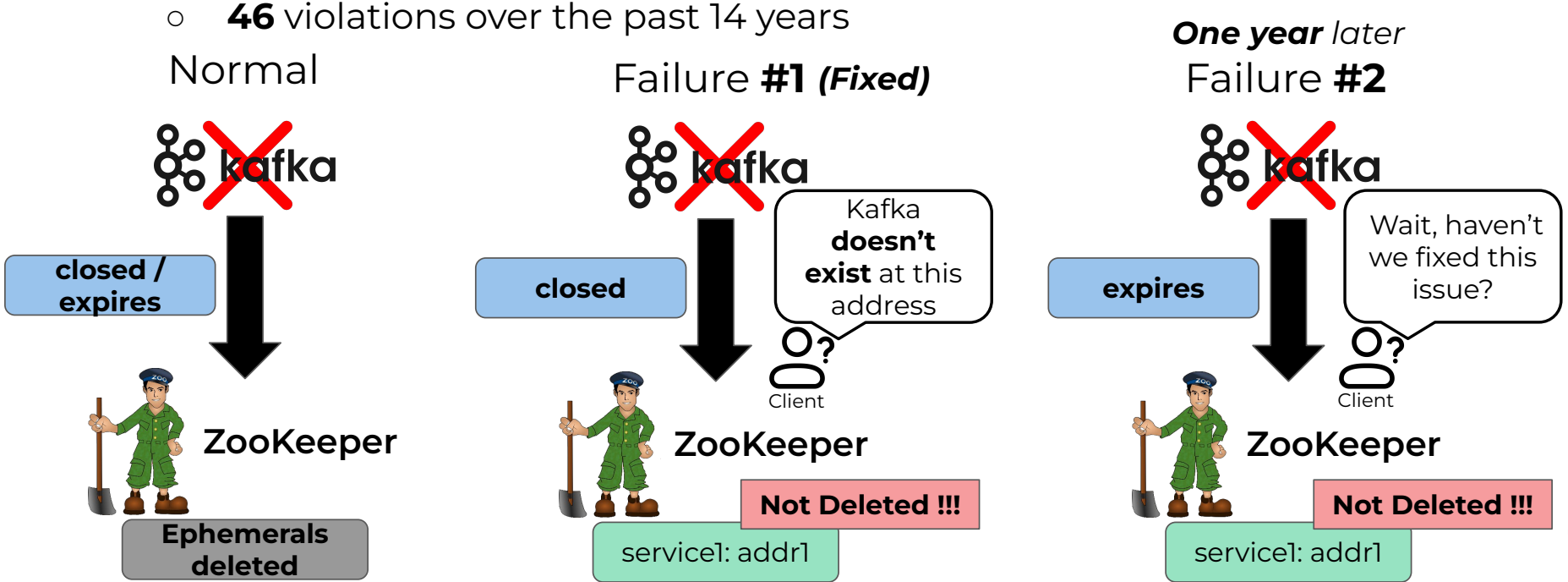
Case Study: ZooKeeper Ephemerals Regression Failures

- ZooKeeper Ephemerals: **auto-deleted data records** upon session disconnection (explicitly closed / expires)
- This semantics is repeatedly violated throughout the development cycle
  - **46** violations over the past 14 years



Case Study: ZooKeeper Ephemerals Regression Failures

- ZooKeeper Ephemerals: **auto-deleted data records** upon session disconnection (explicitly closed / expires)
- This semantics is repeatedly violated throughout the development cycle
  - **46** violations over the past 14 years



Case Study: ZooKeeper Ephemerals Regression Failures

- Rule: prohibits ephemeral creation when session is closing

Failure **#1** fix

```
// validate session before
// creating new ephemerals

protected void pRequest2Txn(...) {
  if (session == null
        || session.isClosing()){
    throw new Exception();
  }
  DataTree.createNode(...);
  ...
}
```

Failure **#2** fix

```
// ensuring expired session
// is not revived

public boolean touchSession(...) {
  if (session == null
        || session.isClosing()){
    return false;
  }
  ...
  return true;
}
if(touchSession()){
  DataTree.createNode(...);
}
```

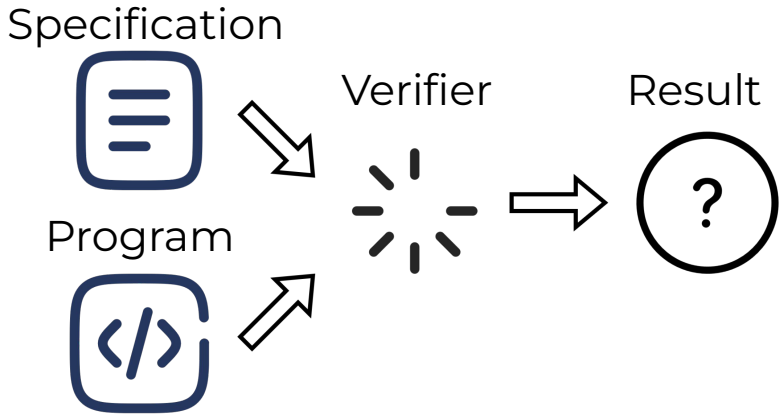
Failure #1 fix didn't add this check

Why Is It So Hard to Prevent Regression Failures?

Prior works rely on testing or formal verification

- Tests are specific to a workload
- Formal verification requires heavy specification and proof

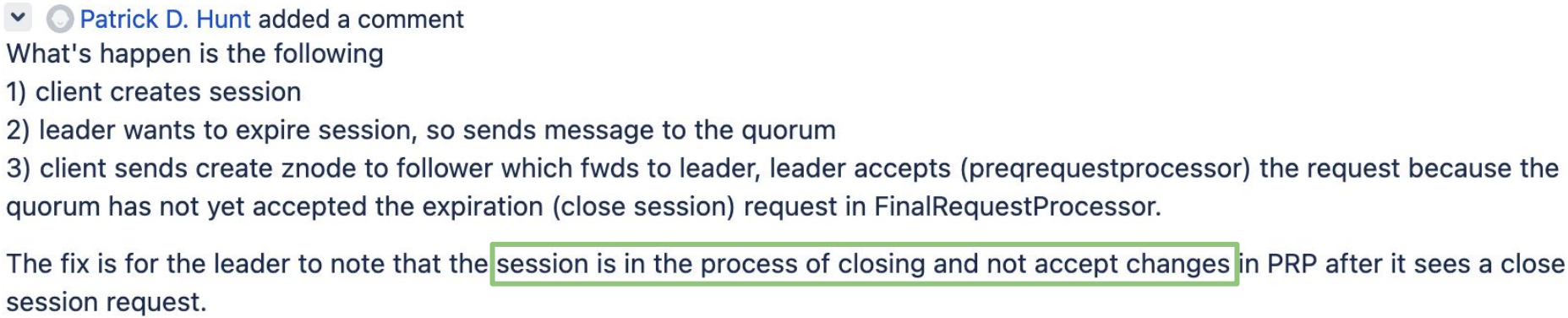
Rainmaker (NSDI ‘23) ADR (NSDI ‘25)
Legolas (NSDI ‘24)



IronFleet (SOSP ‘15) Verdi (PLDI ‘15)
Grove (SOSP ‘23)

A Hidden Opportunity in Post-Mortem

- After a failure occurs, developers learn semantic rules about the implementation-level correctness (“what must never happen again”)
- But these rules live only informally during discussions
- As a result, similar issues quietly reappear in future updates


Patrick D. Hunt added a comment

What's happen is the following

1) client creates session

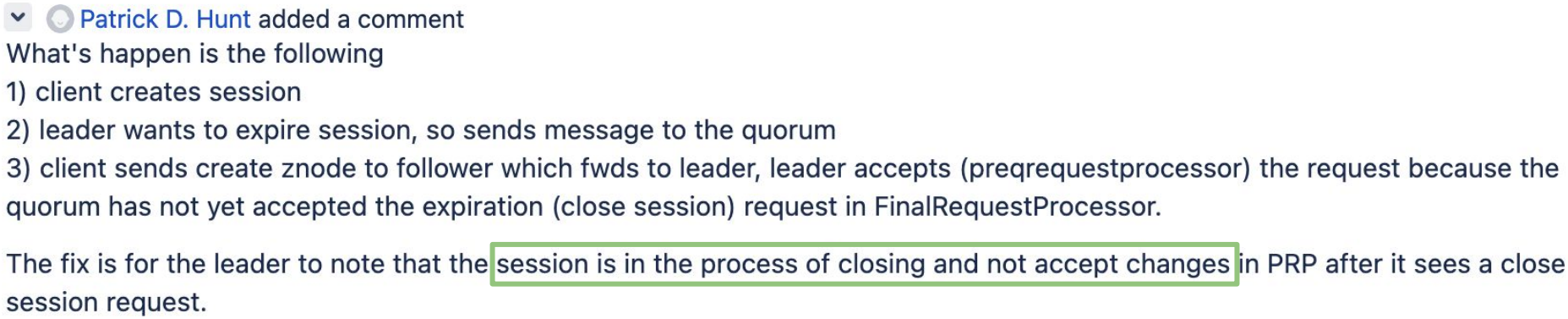
2) leader wants to expire session, so sends message to the quorum

3) client sends create znode to follower which fwds to leader, leader accepts (preqrequestprocessor) the request because the quorum has not yet accepted the expiration (close session) request in FinalRequestProcessor.

The fix is for the leader to note that the session is in the process of closing and not accept changes in PRP after it sees a close session request.

A Hidden Opportunity in Post-Mortem

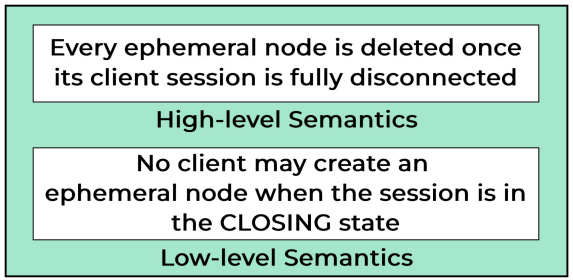
- After a failure occurs, developers learn semantic rules about the implementation-level correctness (“what must never happen again”)
- But these rules live only informally during discussions
- As a result, similar issues quietly reappear in future updates



How to automatically **extract** these semantics and systematically **check** it on the system?

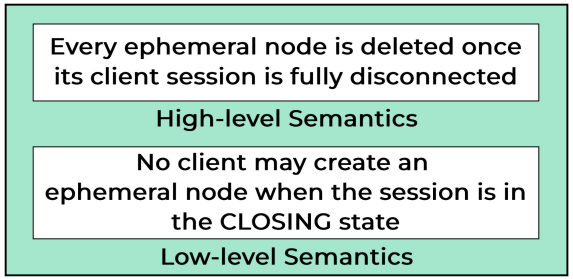
Abstraction: Low-level Semantics

- **High-level semantics (what we wish we could verify)**
  - Describe **system-wide** properties.
  - Great for reasoning, but hard to tie directly to complex, evolving implementations.
  - Enforcing them typically requires heavyweight models and full verification.



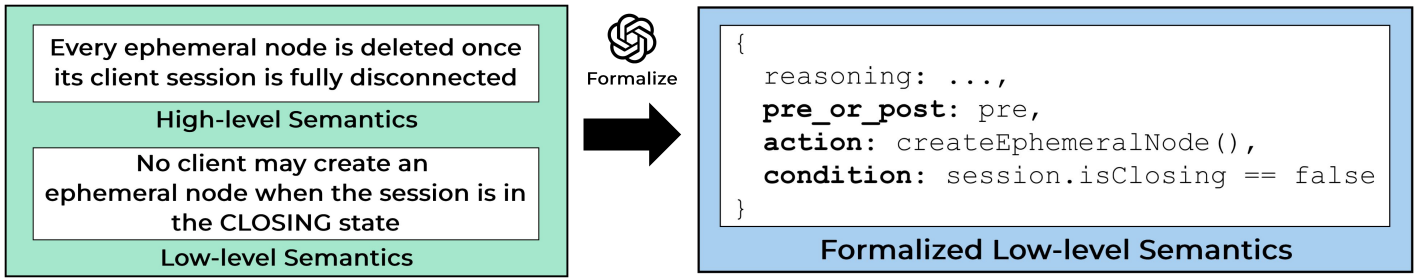
Abstraction: Low-level Semantics

- **High-level semantics (what we wish we could verify)**
  - Describe **system-wide** properties.
  - Great for reasoning, but hard to tie directly to complex, evolving implementations.
  - Enforcing them typically requires heavyweight models and full verification.
- **Low-level semantics (what we actually use)**
  - **Implementation-local** invariants over concrete state and control flow.
  - Attached to specific target statements and predicates over real variables.
  - Can be expressed as small safety contracts and checked automatically.



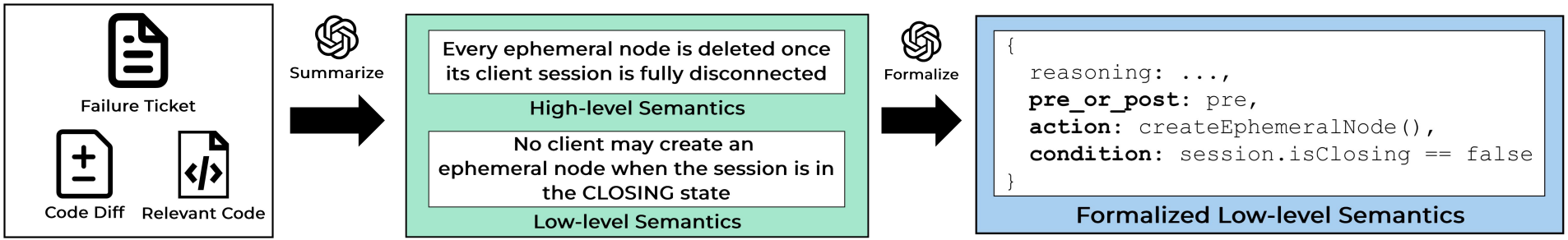
Abstraction: Low-level Semantics

- **action**: The concrete system event where the rule is enforced
- **condition**: A Boolean pre/post condition over state and control flow
- **pre_or_post**: Marks whether the condition must hold before the action (precondition) or after it (postcondition)
- **reasoning**: A brief natural-language explanation tying this rule used to help LLMs (and humans) infer and validate the semantics



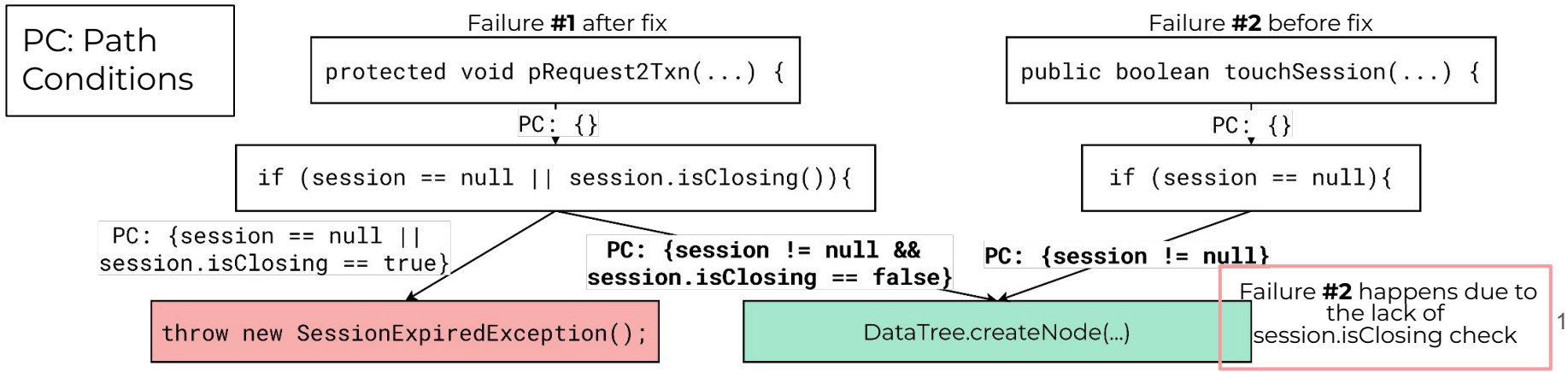
Step #1 How to Infer Low-level Semantics

- Input: Failure ticket, code diff, relevant code (full modified code files)
- LLM-guided inference to extract semantics (at high/low level)
- Translate into our low-level semantics schema (e.g., cond, action)
- Improve robustness
  - Force step-by-step reasoning instead of “just list rules”
  - Clarify what “low-level semantics” means, add examples, enforce JSON output, and use RAG to pull missing context



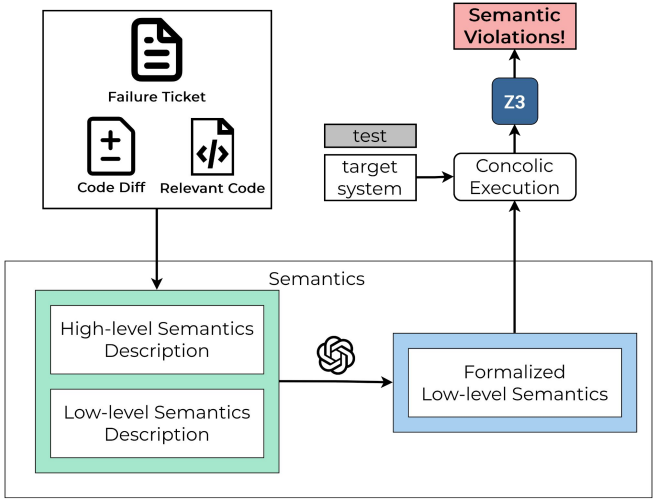
Step #2 How to Enforce Low-level Semantics

- Target only relevant paths
  - Call graph → collect all paths that reach the “action”of the semantics
- Run concolic execution on those paths
  - Use existing tests as concrete seeds; concolic engine explores nearby paths
- Check paths against the semantics with SMT solver (e.g., Z3)
  - Translate path conditions and the semantics into Z3 formulas
  - Path conditions is incomplete / contradicts the semantics → potential regression

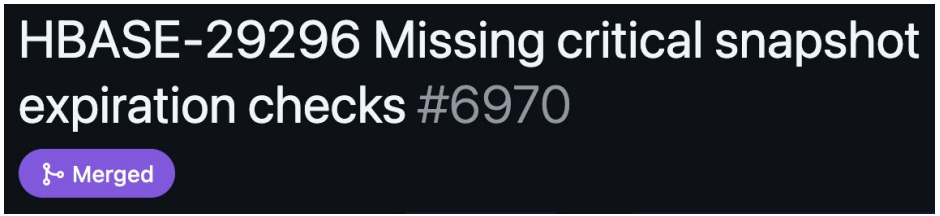


Preliminary Results

- Our prototype LISA currently supports ZooKeeper, HDFS, HBase
- For each system, LISA infers low-level semantics from past incidents
- Example (HBase): it infers that openSnapshot() has a precondition — the snapshot must not have expired



System Workflow

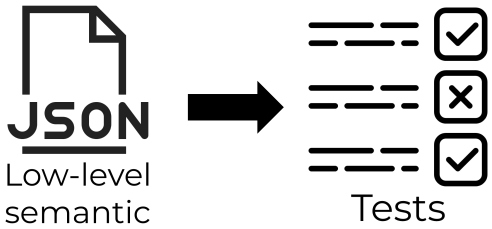


New regression bug found (and fixed)

Discussions

1. How to make LLM-generated semantics reliable?

Validation to mitigate indeterminism and hallucination



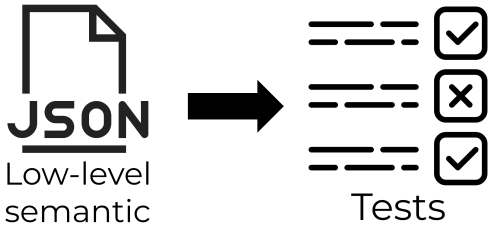
Discussions

1. How to make LLM-generated semantics reliable?

   Validation to mitigate indeterminism and hallucination

2. How to enable developers to proactively create low-level semantics?

   Prompt template based on developers' description



Discussions

1. How to make LLM-generated semantics reliable?

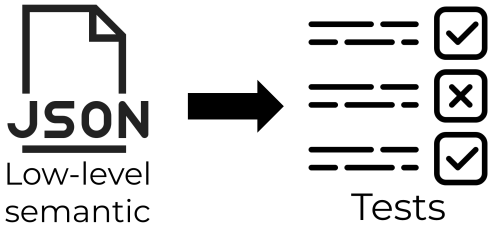
   Validation to mitigate indeterminism and hallucination

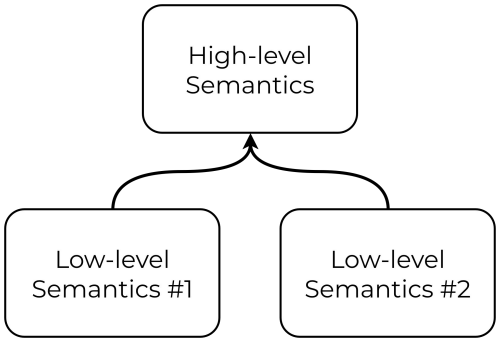
2. How to enable developers to proactively create low-level semantics?

   Prompt template based on developers' description

3. How to use low-level semantics to verify high-level system properties?

   Low-level semantics as **building blocks** for higher-level guarantee





Conclusion

- Cloud systems are updated frequently.
  - **Regression failures**: semantic violations reintroduced by updates.
  - Such failures cause service unavailability and wasted developer effort.
- The key to prevent regression failures lies in **low-level semantics**: implementation-centric rules embedded in postmortems and developer discussions.

LISA automatically **infers low-level semantics** from historical failures using **LLM** and uses **concolic execution** to verify the implementation against these inferred semantics.

dparikesit@virginia.edu