Handling Flaky Regression Tests in CPAchecker

Philipp Wendler

Joint work with Stefan Winter

LMU Munich, Germany

2023-09-11
@ CPA'23

Background

- CPAchecker relies heavily on integration tests to find regressions
- Large number of tests:
  - Dozens of test suites
  - With 100 - 55 000 tests each
- *Regression*: change in test result if new result $\neq$ "correct" (also includes changes between different bad results)
- Mail sent to developers for each test-suite execution with > 1 regression

Flaky Tests

Test result also changes due to reasons
not caused by changes in CPAchecker (*flakiness*):

- Non-deterministic behavior
- Hardware timing
- Random crashes

Often unavoidable in practice:

- Caused by external libraries or environment
- Conceptually inherent nondeterminism

Real problem:

- Some test suites always produce regression reports, but real regressions rare

Current Handling of Flaky Tests

Main strategy:

- Exclude flaky tests
- Manually identified

Plus a simplistic heuristic for flaky timeouts.

Current Handling of Flaky Tests

Main strategy:

- Exclude flaky tests
- Manually identified

Plus a simplistic heuristic for flaky timeouts.

Insufficient:

- Still many regression reports
- No defined rule what counts as “flaky”
- Manual effort (∼100 commits dealing with this)
- Excluded tests could still be useful
- Exclusion list outdated

Re-Runs

State of the art: re-run tests n times

- Expensive
- Hides newly introduced flakiness
- May not catch rare flakiness
- Increases time until developers get results

Inuitive Insight for Solution

Given a change of a test result, is it flaky?

- Assumption: Flakiness is probabilistic,
  i.e., flaky results independent and with certain probability
  (like a series of coin throws, dice rolls, etc.)
- Probability for long sequences of same flaky result low
- Real regressions with different behavior

Inuitive Insight for Solution

Given a change of a test result, is it flaky?

- Assumption: Flakiness is probabilistic,
  i.e., flaky results independent and with certain probability
  (like a series of coin throws, dice rolls, etc.)
- Probability for long sequences of same flaky result low
- Real regressions with different behavior

$\Rightarrow$ Check if result occurs more often in short sequences

RLE-based Identification of Flaky Test Results

1. Compute Run-Length Encoding (RLE) of test-result history
    - Lossless data compression format
    - Also used in time series analysis
    - `aaaabb` → `4a2b`
2. Count $(result, length)$ occurrences
3. Check for inverse correlation between length and count

Likely flaky if statistically significant inverse correlation is found.

Example: Task `sine_3.yml` in nightly-induction

Run-Length Encoding:

seq. length	*result*
94	false
1	EXCEPTION
40	false
1	TIMEOUT
4	false
2	TIMEOUT
4	false
1	TIMEOUT
1	false
2	TIMEOUT
4	false
2	TIMEOUT
2	false
1	TIMEOUT
4	false

seq. length	*result*
1	TIMEOUT
3	false
1	TIMEOUT
19	false
3	TIMEOUT
3	false
1	TIMEOUT
1	false
1	TIMEOUT
7	false
2	TIMEOUT
2	false
1	TIMEOUT
1	false
3	TIMEOUT
	...

Example: Task `sine_3.yml` in nightly-induction

Summary of sequences, sorted by $(result, length)$:

result	*length*	*count*
false	1	22
false	2	4
false	3	5
false	4	6
false	5	2
false	6	2
false	7	2
false	9	1
false	19	1
false	40	1
false	94	1

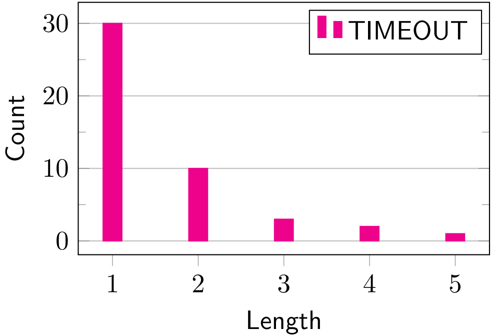
result	*length*	*count*
EXCEPTION	1	1
TIMEOUT	1	30
TIMEOUT	2	10
TIMEOUT	3	3
TIMEOUT	4	2
TIMEOUT	5	1

Example: Task `sine_3.yml` in nightly-induction

Summary of sequences, sorted by $(result, length)$:

result	*length*	*count*
false	1	22
false	2	4
false	3	5
false	4	6
false	5	2
false	6	2
false	7	2
false	9	1
false	19	1
false	40	1
false	94	1

result	*length*	*count*
EXCEPTION	1	1
TIMEOUT	1	30
TIMEOUT	2	10
TIMEOUT	3	3
TIMEOUT	4	2
TIMEOUT	5	1

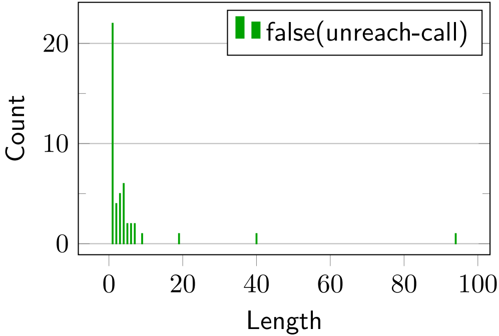


Example: Task `sine_3.yml` in nightly-induction

Summary of sequences, sorted by $(result, length)$:

result	*length*	*count*
false	1	22
false	2	4
false	3	5
false	4	6
false	5	2
false	6	2
false	7	2
false	9	1
false	19	1
false	40	1
false	94	1

result	*length*	*count*
EXCEPTION	1	1
TIMEOUT	1	30
TIMEOUT	2	10
TIMEOUT	3	3
TIMEOUT	4	2
TIMEOUT	5	1



Plan for CPAchecker

1. Implementation on top of BenchExec
2. Add column with likelihood of regression/flakiness to test-result tables
3. Use as heuristic for deciding whether to send regression mail
4. Remove existing naive heuristic

Feedback welcome!

Side Note

CPAchecker is a not-so-small and active software project, with > 15 years of history.

- Potential as case study in SE research!
- Example: years of test data available

Contact us!

Conclusion

- Better heuristic for flakiness in regression tests hopefully coming soon!
- Promising preliminary results
- Large data set for (flaky) tests