TOPIC #1: Concurrent Testing

- Eitan Farchi, Jan Fiedor, Tomas Vojnar, Michael Pradel, Severin Heiniger, Samira Tasharofi

When to stop?

- A concurrent program is run repeatedly – each run new interleavings are executing potentially resulting in exhibiting more problems. When should we stop rerunning the concurrent program

Current approaches

- Coverage concepts, shared lock coverage, lock contention coverage and invariants coverage, Two types - code coverage for which you know when 100% coverage is hit or not
- Based on probabilities you can know how long you need to run but it will only work with small spaces for which the interleaving space is known
- Stop when coverage is not advancing in repeated runs
- Relation between coverage saturation and defect finding capabilities is not clear and requires further research
- Use benchmarks to check the relation between stopping based on coverage saturation and defects found

Biggest problem

- How do we approximate the interleaving space?
- If the graph of the scheduler is know you can use local probabilities to increase the chances that desired interleavings are reached
- Are some of the parameters of the interleaving, e.g., depth, sufficient to estimate the time we need to run?
    - Combine static analysis with depth to approximate the interleaving space
    - Identify which branches is clearly impossible to get a good estimate of the interleaving space

A different approach

- data mining, clustering – classify the runs we have seen up to now and then identify when new interleavings are seen. Thus,, we need a distance notion between interleavings
- Use cut-off approaches? Ability to deduce about the program when executing with hundreds of threads from the same program executing with a few threads