TIE: An Interactive Visualization of Thread Interleavings

Gowritharan Maheswara ● Jeremy S. Bradbury ● Christopher Collins
Faculty of Science ● University of Ontario Institute of Technology ● Oshawa, Ontario, Canada
gowritharan.maheswara@mycampus.uoit.ca, {jeremy.bradbury, christopher.collins}@uoit.ca

UOIT
CHALLENGE INNOVATE CONNECT

Motivation

- Testing concurrent programs has become a major challenge due to the many different ways threads can interleave.
- One solution is to use tools, such as NASA's Java PathFinder (JPF), to explore the thread interleaving space.
  - This tool's limitation is that the data is generally in the form of bulk text logs, which provide little support for common analysis tasks, such as finding common and rare error states.

Research Goal:

To provide a solution to the analytic limitations of JPF by developing an interactive visualization tool, TIE, which integrates with JPF to enhance concurrency debugging.

Model Checking Tools

- Model Checking
  - Given a model of a system, a model checker will determine if the model satisfies a specific requirement.
- NASA's Java PathFinder (JPF) [1,2]
  - An explicit state model checker for Java bytecode programs that focuses on finding concurrency related defects such as deadlocks and data race conditions.
  - JPF determines if a program has a concurrency bug by searching the entire state space (i.e., all possible thread interleavings) of a concurrent program.

Thread Interleaving Explorer Visualization

Each column represents a specific schedule that leads to a concurrency bug. Each block within a column represents a transition in the schedule and the color of the block corresponds to a specific thread.

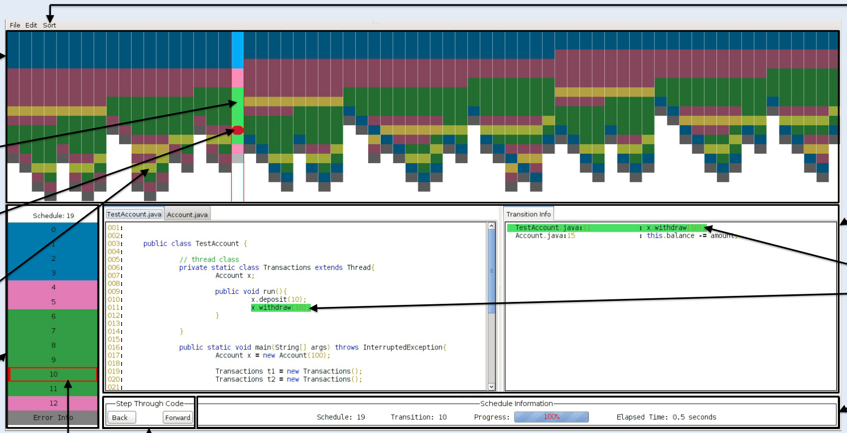
A selected schedule is rendered with brighter variants of the thread hues.

Within the selected schedule, the specific transition that is selected is identified by a red oval.

Gold highlights [optional] reveal all other transitions that execute the same source code as the selected transition.

Specific details of a particular thread interleaving schedule is presented.

When a transition is selected it is highlighted with a red border.



```
    public class TestAccount {

            // thread class
            private static class Transactions extends Thread{
                    Account x;

                    public void run(){
                            x.deposit(10);
                            x.withdraw(10);
                    }

            }

            public static void main(String[] args) throws InterruptedException{
                    Account x = new Account(100);

                    Transactions t1 = new Transactions();
                    Transactions t2 = new Transactions();
```

```
TestAccount.java:11          : x.withdraw(10);
Account.java:15              : this.balance -= amount;
```

```
Schedule: 19      Transition: 10      Progress: 100%      Elapsed Time: 0.5 seconds
```

The "Sort" function sorts the schedules in ascending or descending order based on the total number of transitions, the type of bug caught by the error schedule, or based on schedule similarity.

Source code from the concurrent program under test [left]. The JPF raw data output for the selected transition within a particular thread interleaving [right].

When a transition is selected (indicated by a red border), the corresponding source code and JPF output are highlighted.

A summary of the current JPF execution.

Control buttons support stepping sequential through the selected interleaving schedule.

Conclusions & Future Work

- The TIE Visualization allows a software developer to see the patterns of thread interleavings and explore error states resulting from concurrency bugs. The visualization is closely integrated with JPF to allow for visualization of outputs while model checking is in progress.
- Future research plans include:
  - Implement visualizations to represent overall statistical error data of the JPF execution
  - Implement additional visualization techniques to represent sorted thread interleaving data
  - Study and refine TIE techniques through deployment to the JPF Community

References

[1] K. Havelund and T. Pressburger. Model checking Java programs using Java PathFinder. *Int. J. on Software Tools for Technology Transfer (STTT)*, 2(4), 2000.

[2] Java Path Finder website, *2010.* http://babelfish.arc.nasa.gov/trac/jpf/

SQR GROUP
vialab visualization for information analysis

Funding provided by: NSERC CRSNG