Automatically Fixing Data Races in Parallel Java Programs Using Privatization

Loránd Szakács, “Politehnica” University of Timișoara
ADVISOR: Danny Dig

Background
- We now live in the **multicore** era!
- Existing software needs to be refactored to benefit from multiple processors.
- The changes needed to ensure thread safety are **nontrivial**.

Goals
- Build an automated analysis and refactoring tool that can fix known races using data privatization.
- Compare the solutions offered by our tool with the ones implemented by software developers of open-source projects.

Fundamental Questions/Challenges
- Can we trace the origin of objects involved in races using only **static analysis**?
- Can we build an algorithm that can safely determine when and how to apply data privatization?

Research Plan

1. Analyze open-source projects to understand parallelization challenges
   - Ex: Jmol, VASSAL, Art of Illusion

2. Build an analysis that can automatically infer the necessary transformations.

3. Interface with a refactoring engine that can perform the changes.

Research Results
- Developed a static analysis that can trace the origin of the objects involved in races and that outputs a privatization plan:

  ```
  int x;
  public void parallelOperation() {
    x = rand();
    foo(x);
  }
  
  ThreadLocal<Integer> x;
  public void parallelOperation() {
    x.set(rand());
    foo(x.get());
  }
  
  ThreadLocal ensures the uniqueness of each object/thread.
  ```

Related Work/Interaction with Other Projects
- We built the analysis on top of the WALA static analysis library.