Lab 3: Data Race Detection and Repair

Instructor: Vivek Sarkar

1 Update your HJ Installation

Please update your HJ installation to make sure that you have the latest updates and bug fixes.

1. Download the jar file for DrHJ from `http://www.cs.rice.edu/~vs3/downloads/hj/drhj.jar`
2. A link to the above jar file can be obtained by following these links from the course web page: "HJ Info" → "HJ Download and Setup", and then searching for "Download the jar file corresponding to DrHJ"

NOTE #1 (Optional): If you prefer to use a command-line interface instead of DrHJ to compile and run HJ programs, you can down load an HJ installation from the "HJ Download and Setup" page listed above by searching for "Download the zip file containing the HJ package" and then following the subsequent instructions. The command-line interface only works on Unix-based systems (like CLEAR, Sugar, Mac OS), and not on Windows. In contrast, DrHJ runs on both Unix-based systems and also on many Windows installations. (For Windows, you should download a standard full JDK from Oracle to maximize the chances of DrHj working on your system.)

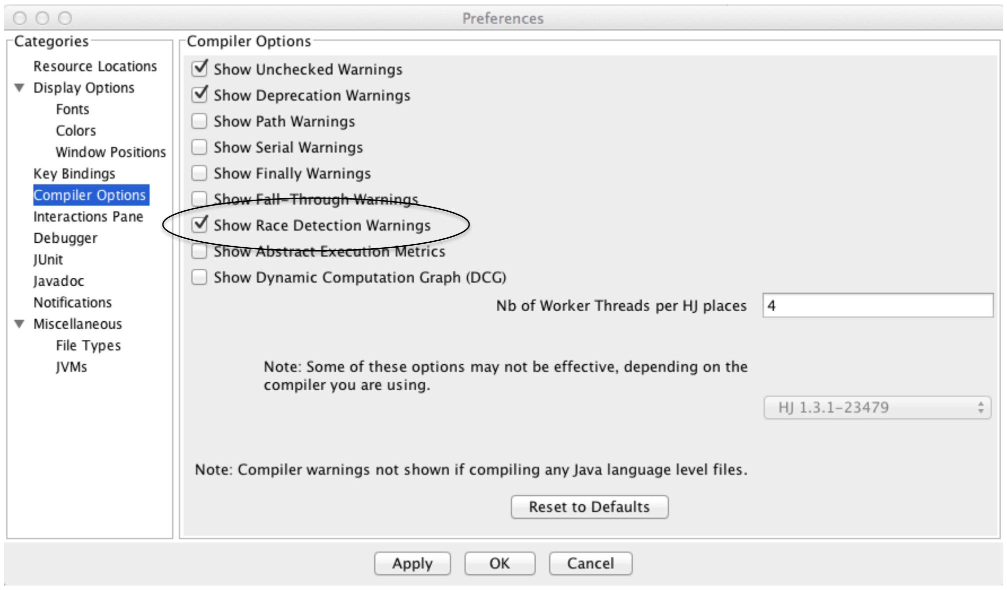
2 Example HJ Program for Data Race Detection

The HJ compiler supports a "-racedet" option to enable automatic instrumentation of the HJ program to check for data races at runtime. This capability can be enabled by selecting the "Show Race Detection Warnings" option in DrHJ's Compiler Option preferences (see Figure 1), or (if you're not using DrHJ) by typing the following command on the command line: *hjc -racedet ArraySum1*. After this compilation, the HJ program can be launched as usual. Instead of a normal parallel execution, the instrumented version runs on a single processor core (single worker) and reasons about all possible interfering accesses that are candidates for data races in this sequential execution. If a race is found, a pair of interfering accesses is printed with source code locations for the accesses.

1. Download the `RacyArraySum1.hj` file from the Code Examples column for Lab 3 in the course web page, `https://wiki.rice.edu/confluence/display/PARPROG/COMP322`.
2. Compile this HJ program with the appropriate option for enabling datarace detection, as described above.
3. Now, run the program, and observe from the output the exact location in the program where the data race exists. You can use this information to correct the errors in the program.

3 Lab 3 Assignment

1. Repeat the following steps for each of the remaining files in the Code Examples column for Lab 3 in the course web page, `https://wiki.rice.edu/confluence/display/PARPROG/COMP322`.
2. Download the file.
3. Compile the program with datarace detection enabled.



Compiler Options

- [✓] Show Unchecked Warnings
- [✓] Show Deprecation Warnings
- [ ] Show Path Warnings
- [ ] Show Serial Warnings
- [ ] Show Finally Warnings
- [ ] Show Fall-Through Warnings
- [✓] Show Race Detection Warnings
- [ ] Show Abstract Execution Metrics
- [ ] Show Dynamic Computation Graph (DCG)

Nb of Worker Threads per HJ places: 4

Note: Some of these options may not be effective, depending on the compiler you are using.

HJ 1.3.1-23479

Note: Compiler warnings not shown if compiling any Java language level files.

Figure 1: Selection of "Show Abstract Execution Metrics" in DrHJ's Compiler Option preferences

4. Run the program, and observe the source code location for conflicting accesses.
5. Edit the program to fix the data race bug and get the correct output. (You can make a copy of the program and comment out all occurrences of "async" to obtain a correct sequential program, if you'd like to use it as baseline to understand what the correct version of the parallel program is supposed to compute.) Repeat steps 3, 4, 5 for the current parallel program until no data race is reported.
6. The only edits permitted in step 5 are:
   (a) Insertion of finish keywords.
   (b) Replacement of normal integer add operations by AtomicInteger operations (Lecture 6).
7. Remember that you can generate the computation graph if it helps you gain a better understanding of the parallel program. Do not select both the race detection and computation graph options at the same time. Also, choose a smaller input size to ensure that the graph is small enough to understand.