Testing Concurrent Programs

- Concurrent programs becoming more important
  - Computers become faster by adding more processor cores
  - To benefit from new hardware, programs have to concurrently use more than one processor core

- Unit testing is effective for single-threaded programs
  - Current approaches fail for concurrent programs

- Thread switching is non-deterministic and machine-specific
  - Success of a unit test does not imply correct behavior under all possible schedules and on all machines
  - Most programs are concurrent:
    - GUI: separate thread for display
    - Multi-core: programs must be concurrent to benefit
  - Current tools not effective or easy to use on large projects

- Concutest: A Framework for Testing Concurrent Programs
  - Concurrency-aware extension of JUnit (ConcJUnit)
  - Lightweight checking of concurrency invariants (ThreadCheck)
  - Logging of method execution to simplify and decouple unit tests for reactive programs
  - Execution with short delays inserted at critical places to test different execution schedules

- DrJava Case Study for ConcJUnit and ThreadCheck
  - 900 unit tests in DrJava code base
  - 20 previously unknown problems detected by Concutest
  - 1% slowdown

- Website: www.concutest.org

Multi-Stage Programming

- Program abstractions (e.g. recursion) without performance overhead
  - Abstractions make programs easier to understand
  - Staging moves abstractions out of the runtime into a code generation step
  - Result: Code written using abstractions (e.g. `power`) is optimized for special cases (e.g. `square`)
  - Killer example: Interpreters become compilers

```java
// unstaged power function in Java
double power(double x, int n) {
    if (n==0) return 1.0; // overhead: if and comparison
    else return x * power(x, n-1); // overhead: function call
}
```

```java
// staged power function in Java Mint, runs 9x faster than unstaged
Code<Double> power(Code<Double> x, int n) {
    if (n==0) return <| 1.0 |>; // overhead of abstractions removed in generated code:
    else return <| `x * `power(x, n-1) |>; // overhead of abstractions removed in generated code:
}
```

### Benchmark Table

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>speedup</th>
<th>unstaged µs</th>
<th>staged µs</th>
</tr>
</thead>
<tbody>
<tr>
<td>power</td>
<td>9.2</td>
<td>0.060</td>
<td>0.0065</td>
</tr>
<tr>
<td>fib</td>
<td>8.8</td>
<td>0.058</td>
<td>0.0065</td>
</tr>
<tr>
<td>mmult</td>
<td>4.7</td>
<td>13</td>
<td>2.7</td>
</tr>
<tr>
<td>eval-fact</td>
<td>20</td>
<td>0.83</td>
<td>0.042</td>
</tr>
<tr>
<td>eval-fib</td>
<td>24</td>
<td>18</td>
<td>0.73</td>
</tr>
<tr>
<td>av-mmult</td>
<td>65</td>
<td>20</td>
<td>0.30</td>
</tr>
<tr>
<td>av-mtrans</td>
<td>14</td>
<td>1.0</td>
<td>0.071</td>
</tr>
<tr>
<td>serialize</td>
<td>26</td>
<td>1.5</td>
<td>0.057</td>
</tr>
</tbody>
</table>

- DrJava first released in January 2002
  - Began working on DrJava in 2006, now one of two principal developers
  - 230,000 downloads in first 5 years, 870,000 downloads in 5 years since
  - Recently surpassed a million downloads

- DrJava implemented many useful features
  - Predictive input dialog ("Go to File", "Complete Word under Cursor")
  - Clipboard history
  - Multiple underlined searches ("Find All")
  - Detachable tabbed panes and debugger window
  - Persistent breakpoints and bookmarks

- Use DrJava as tool to make research accessible to students
  - Integrated Concutest
  - Integrated JavaMint
  - Integrated other Rice research projects (NextGen, Habanero Java)

- Website: www.drjava.org

Computer Science Education

- Designed assignments and class projects for programming classes
  - Marine Biology Simulation (OOP: object-oriented programming)
  - Design Patterns for Parsing (OOP)
  - Programming for Change (OOP, agile development)
  - Bounded Buffer, Readers/Writers Locking (concurrent programming)
  - Working on tutorial for Java Mint...

- Developed syllabus as instructor
  - Principles of Object-Oriented Programming II (COMP 202)
  - Production Programming: Concurrent Programming and DrJava (COMP 402)