Master-Seminar: Verification of Concurrent Programs

Corneliu Popeea

Technische Universität München, Germany

27 January 2012

Why concurrent programs?

- **Important**
- **Complex**
- **Buggy**
  - Apache web server, MySQL, Mozilla suite (“Learning from mistakes: A comprehensive study on real world concurrency bug characteristics” - ASPLOS 2008)

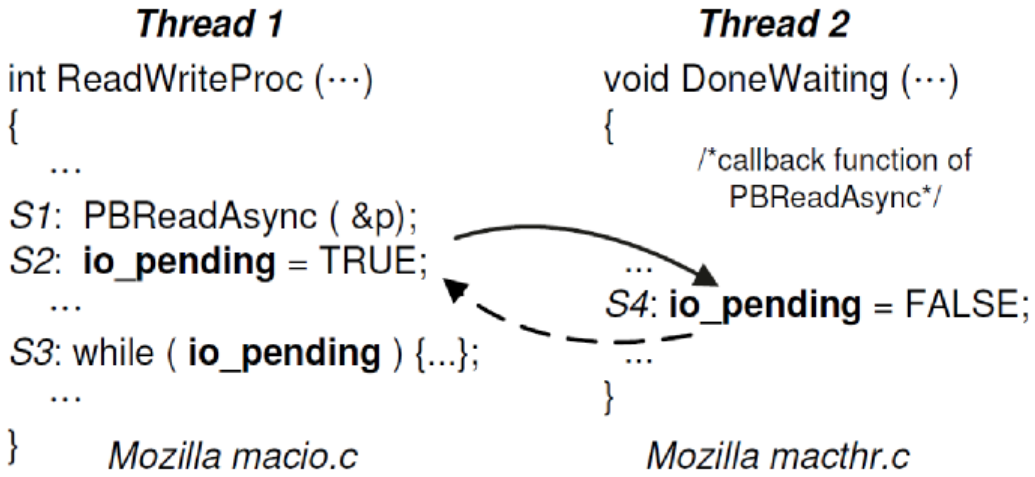
Why concurrent programs?

- **Important**
- **Complex**
- **Buggy**
  - Apache web server, MySQL, Mozilla suite (“Learning from mistakes: A comprehensive study on real world concurrency bug characteristics” - ASPLOS 2008)

Why concurrent programs?

- **Important**
- **Complex**
- **Buggy**
  - Apache web server, MySQL, Mozilla suite (“Learning from mistakes: A comprehensive study on real world concurrency bug characteristics” - ASPLOS 2008)

A buggy example



- An order violation bug from Mozilla. The program fails to enforce the programmer's intention: thread 2 is expected to write `io_pending` to be `FALSE` some time after thread 1 initializes it to `TRUE` .

Demo

- **Model checking:** use the tool Threader
- **Input:** file.c (program + property)
- **Output:**
  - 'Program is correct'
  - 'Feasible counterexample'
  - time-out / memory error / other error

Some challenges

- Scalability: analyzing all thread interleavings is prohibitively expensive
- Heap analysis
- Recursive procedures
- Various properties to check: termination, determinism

A list of topics

1. KISS: Keep it simple and sequential - PLDI 2004
   - **Algorithm:** program transformation from a concurrent to a sequential program that simulates a large subset of the behaviors of the concurrent program
   - **Model checking:** use the SLAM tool
   - **Examples:** detect race conditions in Windows device drivers

A list of topics

2. Asserting and checking determinism for multithreaded programs - FSE 2009

- **Specification:** regions of parallel programs that should behave deterministically
- **Algorithm:** automated directed testing
- **Examples:** Java benchmarks, Parallel Java library

A list of topics

3. Using Promela and Spin to verify parallel algorithms - LWN.net 2007

- **Specification:** synchronization algorithms written in the Promela language
- **Model checking:** use the Spin tool
- **Example:**
  - RCU (read-copy-update) is a synchronization mechanism with extremely low overhead, an alternative to the “readers-writers” mutual exclusion protocol
  - 2000 uses of RCU in the Linux kernel (as of 2008)

A list of topics

4. Verifying SystemC: a software model checking approach - FMCAD 2010

- **Application:** SystemC is becoming a de-facto standard for the development of embedded systems
- **Algorithm:** program transformation from SystemC programs to C programs with a non-preemptive scheduler
- **Model checking:** with explicit support for the scheduler
- **Evaluation:** designs such as token-ring protocols

A list of topics

5. A marriage of rely/guarantee and separation logic - CONCUR 2007
   - **Logic:** RGSep, a logic to reason about heap
   - **Example:** a linked-list implementation of a set

6. Proving that non-blocking algorithms don't block - POPL 2009
   - **Specification:** liveness properties
   - **Algorithm:** automates the application of RGSep
   - **Example:** a non-blocking stack implementation

A list of topics

5. A marriage of rely/guarantee and separation logic - CONCUR 2007
   - **Logic:** RGSep, a logic to reason about heap
   - **Example:** a linked-list implementation of a set

6. Proving that non-blocking algorithms don't block - POPL 2009
   - **Specification:** liveness properties
   - **Algorithm:** automates the application of RGSep
   - **Example:** a non-blocking stack implementation

A list of topics

7. Efficient algorithms for pre* and post* - POPL 2000
   - **Algorithm:** compute sets of reachable states
   - **Examples:** recursive procedures and fork-join synchronization

8. Proofs of networks of processes - TOSE 1981
   - **Logic:** for reasoning about processes that communicate via message-passing
   - **Examples:** computing odd primes, computing factorial numbers

A list of topics

7. Efficient algorithms for pre* and post* - POPL 2000
   - **Algorithm:** compute sets of reachable states
   - **Examples:** recursive procedures and fork-join synchronization

8. Proofs of networks of processes - TOSE 1981
   - **Logic:** for reasoning about processes that communicate via message-passing
   - **Examples:** computing odd primes, computing factorial numbers

A list of topics

9. A study of real-world bugs - ASPLOS 2008
   - **Application:** bugs in open-source software, e.g., Apache web server, Mozilla, MySQL, OpenOffice
   - **Study:** classification of concurrency bugs, how many threads/variables are involved, how were these bugs fixed

10. Scalable synchronous queues - PPoPP 2006
    - **Novel data structure:** a scalable synchronous queue
    - **Evaluation:** outperforms the SynchrnousQueue from Java SE 5.0

A list of topics

9. A study of real-world bugs - ASPLOS 2008
   - **Application:** bugs in open-source software, e.g., Apache web server, Mozilla, MySQL, OpenOffice
   - **Study:** classification of concurrency bugs, how many threads/variables are involved, how were these bugs fixed

10. Scalable synchronous queues - PPoPP 2006
    - **Novel data structure:** a scalable synchronous queue
    - **Evaluation:** outperforms the SynchrnousQueue from Java SE 5.0

Administrative issues

Dates

- **Vorbesprechung:** now
- **Topic assignment:** now or by email
- **First meeting with the supervisor:** first week of May at the latest
- **First version of the slides:** first week of June
- **Final version of slides and summary:** one week before the presentation
- **Presentation:** date to be decided

Administrative issues

Supervision

- Corneliu Popeea: homepage
- Andrey Rybalchenko: homepage
- Alexander Malkis: homepage

Administrative issues

Grading

- Preparation phase
- Writing a summary (4-5 pages)
- Giving a talk (40 minutes)
- Active participation during the talks

Questions ?