Summary

INF2140 Parallel Programming

May 9, 2012

Plan for today

- **Looking ahead**
  - Two guest lecturers with short presentations
- **Final exam**
  - format
  - examples
  - discussion
- **Overview**
  - main issues in the course
- **Questionnaire**
  - we would like some feedback on the course from you

Guest presentations

Peter Ölveczky

- Advanced modeling of concurrent and distributed systems
- INF3230 - Formal modeling and analysis of communicating systems

Volker Stolz

- Analysis which applies directly to concurrent Java programs
- Demo of the Java Pathfinder tool

Final Exam

Practical information

June 15, 09:00, 4 hours

What kinds of questions can we expect?

- Theory (Understanding of main concepts)
- Modelling
- Programming
- Applications

Approximate distribution of work load:

- Theory/Modelling/Programming: 50%?
- Applications: 50%?

Example 1: Theory (deadlock and safety)

- What are the four conditions for deadlock?
- Give a simple example of a deadlock!

Example 2: FSP (alphabets)

- What is the alphabet of a process?
- When do you need to hide actions?
- When do you need to extend an alphabet?
- When do you need to rename actions?

Example 3: FSP (properties)

- How do you include safety checks in FSP?
- What is required from a safety property in FSP?
- How do you specify liveness properties in FSP?
- What is fairness, and how is this treated in FSP?
- When should you use action priority in FSP?

Example 4: Modelling in FSP

Explain the difference between

```
a -> ( b -> P
     | c -> P)
```

and

```
( a -> b -> P
| a -> c -> P)
```

Example 5: Programming in Java

- How do you define a monitor in Java?
- How do you test a condition in Java?
- When should you use `notify` and
  when should you use `notifyAll`?
- Can you ensure that all threads waiting on a condition variable in a monitor will eventually get to execute?

Example 6: Application

Format: You get an informal description of a system.

For example:

a railroad crossing

You could be asked to

- model the system (or a part of it) in FSP
- formulate some safety and liveness properties
- discuss/analyze
- implement the system (or a part of it) in Java

- A **"mini-assignment"** in the exam

Overview of the course

- We look at the main topics covered in the lectures

Why is concurrency difficult to understand and analyse?

- For n processes with m atomic statements each, the formula for the number of possibly different executions is

$$\frac{(n * m)!}{m!^n}$$

- For 3 processes, each with given number of atomic operations:

process 1	process 2	process 3	number of executions
2	2	2	90
3	3	3	1680
4	4	4	34 650
5	5	5	756 756

Chapter 2: Processes and Threads

- FSP: Finite state processes to model processes as sequences of actions
- Basic FSP: choice, recursion, guards, indexing, alphabet
- Labelled transition systems (state machines) and LTSA
- Java implementation: program (sequential) processes as threads

Chapter 3: Concurrency

- Parallel composition of processes in FSP
- Interleaving:
  - n state machine || m state machine = n*m state machine
- Synchronization: shared actions in FSP
- Prefixing and renaming of actions in FSP
- Silent actions
- Structure diagrams

Chapter 4: Shared Objects & Mutual Exclusion

- How do threads interact in Java?
- Interference and mutual exclusion
- Java synchronized objects and statements
- Test processes and exhaustive search in LTSA

Chapter 5: Monitors & Condition Synchronization

- Guarded actions in FSP
- Condition synchronization in Java
- Java: `notify()` vs `notifyAll()`
- Semaphores in FSP and Java
- Binary semaphores
- Monitor invariants
- Bounded buffer
- "Nested monitor" problem, deadlock

Chapter 6: Deadlock

- 4 general conditions for deadlock
  - Serially reusable resources, incremental acquisition, no pre-emption, wait-for cycle
- Deadlock in FSP: no eligible action
- Deadlock in Java: blocked threads
- Dining philosophers
  - "wait-for cycle"
  - Breaking the cycle: even numbered philosophers get their left forks first, and odd their right first.
- Deadlock avoidance: to design systems in which deadlock cannot occur

Chapter 7: Safety & Liveness Properties

- Property: true for *every* possible execution
- Model safety: no reachable ERROR/STOP state
- Model progress: an action is eventually executed
- Fairness in FSP and in Java
- Terminal set of states
- Imitation of worst case scenario/high load by action priority
- Examples: Single Lane Bridge, Readers/Writers
- Safety properties

Chapter 8: Model-Based Design

- Start from
  - User requirements, goals scenarios, (external) properties
  - Structure diagrams,
  - Processes and their alphabet
  - (Internal) properties
  - Implementation from the verified model
- Cruise Control System

Chapter 9: Dynamic systems

- Arbitrary number of threads in Java
- Finite number of recursive processes in FSP
- Starvation, overtaking
- **Golf example**
  - Liveness violation: some players may wait forever
  - Solution: Bounded overtaking using tickets
- Java: slave thread
- Java's `join` method

Chapter 10: Message Passing

- Synchronous message passing - channel
- Selective receive in FSP and Java
- Asynchronous message passing - port
- Rendezvous
- Java Implementation: Entry
- Rendezvous vs Monitor Method Invocations

Chapter 11: Concurrent Software Architectures

Examples:

- Filter pipelines
- Supervisor and workers
  - Linda tuple space
- Announcer/listener model

Goal:

- General designs and general properties
- Abstraction

Chapter 12: Timed Systems

- Discrete time
- Ticks, Global clock
- Time consistency, "time-stop"
- Maximal progress in FSP, two-phase model in Java
- Output intervals, jitter, timeout in FSP
- Thread-based vs event-based in Java
- `sleep(ms)`, `wait(ms)`, timeout in Java

Next lecture: May 23

Plan for next lecture: Repetition of selected topics

- Which topics would you like to discuss in more detail?
- You may send us questions by email!
- Deadline for questions: May 16