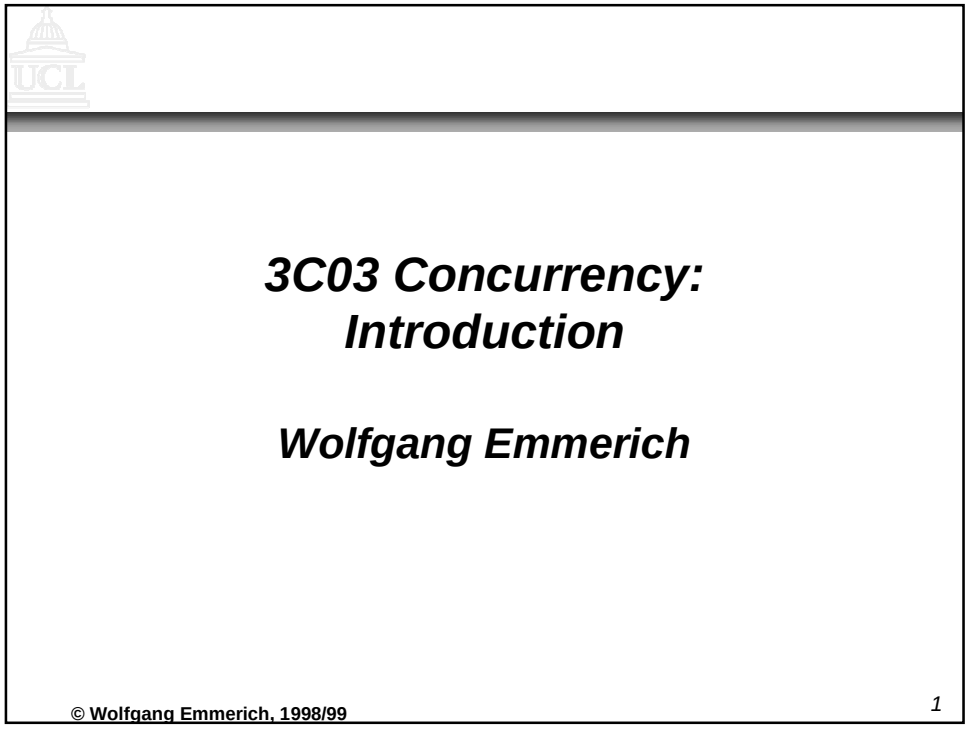
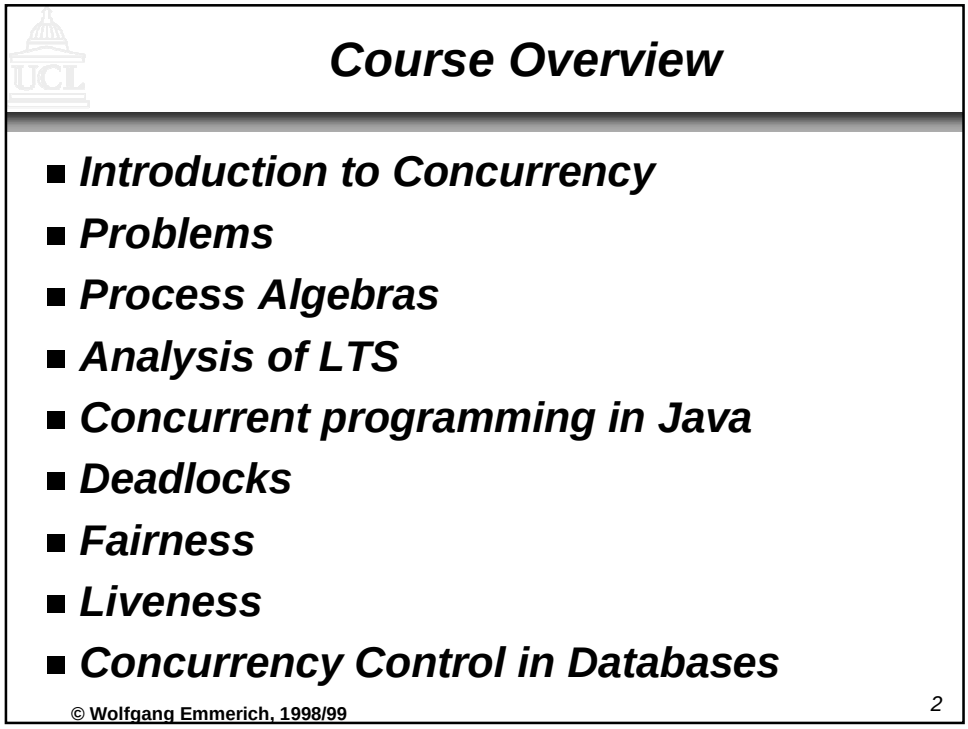


3C03 Concurrency: Introduction

Wolfgang Emmerich



Course Overview

- Introduction to Concurrency
- Problems
- Process Algebras
- Analysis of LTS
- Concurrent programming in Java
- Deadlocks
- Fairness
- Liveness
- Concurrency Control in Databases

No Phones

Communication

- *E-Mail to me*
  - *w.emmerich@cs.ucl.ac.uk*
- *E-Mail to rest of the course*
  - *3c03@cs.ucl.ac.uk*
- *If you have not already done so, subscribe to the e-mail list 3c03!*

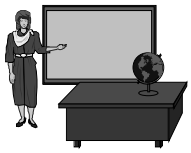
How to reach me?

Pearson Building, 105

www.cs.ucl.ac.uk/staff/w.emmerich

020 7679 4413

Organisation



- ***Lectures***
  - ***Three per week***
- ***Tutorials/Lab***
  - ***Monday 9-10***
- ***Reading***

Bibliography

- *J. Magee & J. Kramer. Concurrency - State Models and Java Programs. Wiley. 1999*
- *A. Burns & G. Davis. Concurrent Programming. Addison Wesley - International Computer Science Series 1993*
- *G.R. Andrews. Concurrent Programming: Principles and Practice. Benjamin/Cummings, 1991*
- *D. Lea. Concurrent Programming in Java™: Design Principles and Patterns. The Java Series, Addison-Wesley, 1996*
- *David Flanagan.Java in a Nutshell. O'Reilly & Associates Inc. 1996*

What are you going to learn?

- *Problems that occur when writing concurrent programs*
- *Formalisms to specify concurrency*
- *Analysis techniques to reason about correctness of specifications*
- *Implementation of concurrency in Java*
- *Practical experience (specification, analysis, implementation) in exercises and coursework*

Lecture Plan until Reading Week

1 Introduction
2 Modelling Processes
3 Modelling Concurrency in FSP
4 FSP Tutorial
5 LTSA Lab
6 Programming in Java
7 Concurrency in Java
8 Lab: Java Thread Programming
9 Mutual Exclusion
10 Lab: Synchronization in Java
11 Semaphores and Monitors
12 Conditional Synchronization
13 Fairness & Liveness
14 Safety
15 Tutorial: Model Checking

Why Concurrent Programming?

- Performance gain from multiprocessing hardware
  - (parallelism)
- Increased application throughput
  - (I/O call only blocks one thread)
- Increased application responsiveness
  - (high priority thread for user requests).
- More appropriate structure
  - (for programs which control multiple activities and handle multiple events)

Engineering of Concurrent Systems

- *Concurrency in safety-critical Systems*
  - *Therac-25 failed due to race conditions*
- *Concurrency in mission-critical Systems*
  - *Increasing amount of business applications uses concurrency*
- *Availability of concurrency in mainstream programming languages*
  - *e.g. Java and Ada-95*

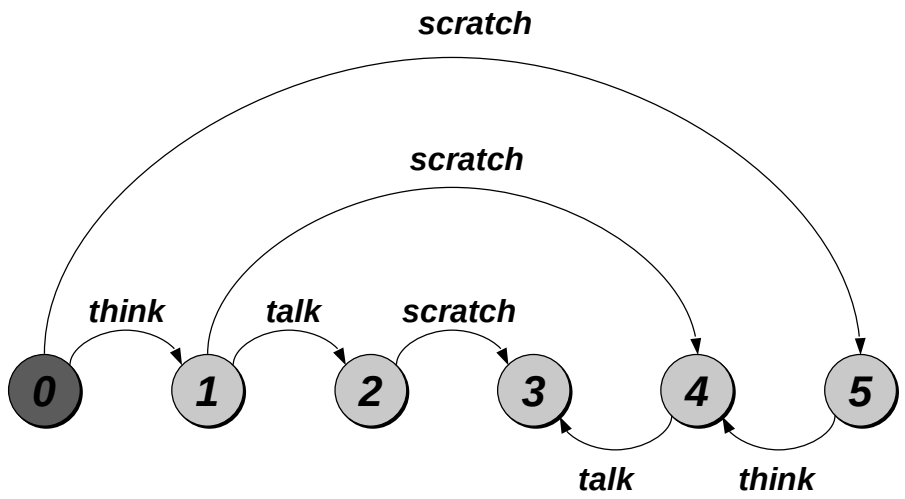
Modelling Concurrency

- *Analogy to Models in Engineering*
- *Modelling Concurrency*
  - *Process Algebras in FSP*
- *Analysis of Models*
  - *Using Labelled Transistion System Analysis*
- *Transformation of Models*
  - *into Java Implementations using Threads*

FSP Example

```
ITCH =              (scratch->STOP).
CONVERSE =          (think->talk->STOP).
||CONVERSE_ITCH =  (ITCH || CONVERSE).
```


LTS Example

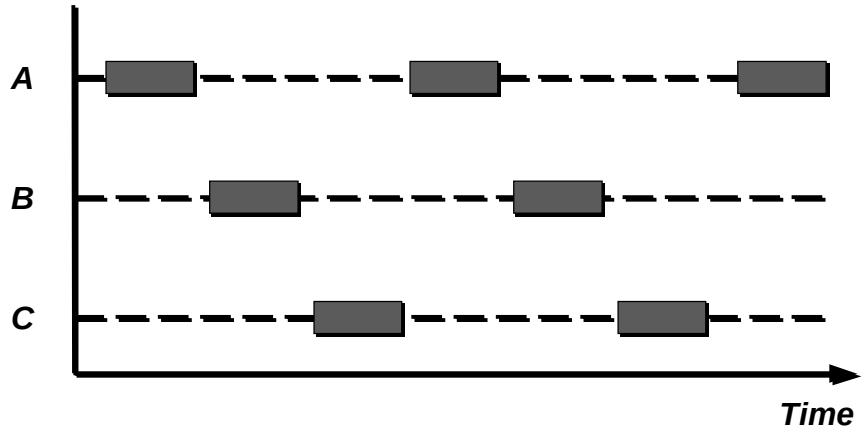


Definitions

- ***Parallelism***
  - *Physically simultaneous processing*
  - *Involves multiple PEs and/or independent device operations.*
- ***Concurrency***
  - *Logically simultaneous processing*
  - *Does not imply multiple processing elements (PEs).*
  - *Requires interleaved execution on single PE.*

Interleaved Model of Concurrency

- *Executing 3 processes on 1 processor:*



Summary

- *Motivation for concurrent programs*
- *Engineering approach to concurrency*
- *Finite State Processes*
- *Labelled Transition Systems*
- *Parallelism vs. concurrency*
- *Interleaved model of concurrency*
- *Next Lecture: modelling processes in FSP*