



C340 Concurrency: Introduction

Wolfgang Emmerich



Course Overview

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



How to reach me?



Pearson Building, 402



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



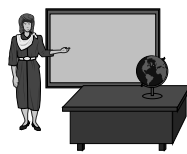
0171 504 4413

© Wolfgang Emmerich, 1998/99

3



Organisation



■ Lectures

- *Wed 1-2 (G22)*
- *Thu 3-4 (229)*
- *Fri 1-2 (G22)*



■ Tutorials/Labs



■ Reading

© Wolfgang Emmerich, 1998/99

4



Bibliography

- *J. Kramer & J. Magee. Concurrent Programming. Wiley. 1999 (to appear)*
- *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 | 9 Mutual Exclusion |
| 2 Modelling Processes | 10 Lab: Synchronization in Java |
| 3 Modelling Concurrency in FSP | 11 Semaphores and Monitors |
| 4 FSP Tutorial | 12 Conditional Synchronization |
| 5 LTSA Lab | 13 Fairness & Liveness |
| 6 Programming in Java | 14 Safety |
| 7 Concurrency in Java | 15 Tutorial: Model Checking |
| 8 Lab: Java Thread Programming | |

© Wolfgang Emmerich, 1998/99

7



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)

© Wolfgang Emmerich, 1998/99

8



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 Transition System Analysis*
- **Transformation of Models**
 - *into Java Implementations using Threads*

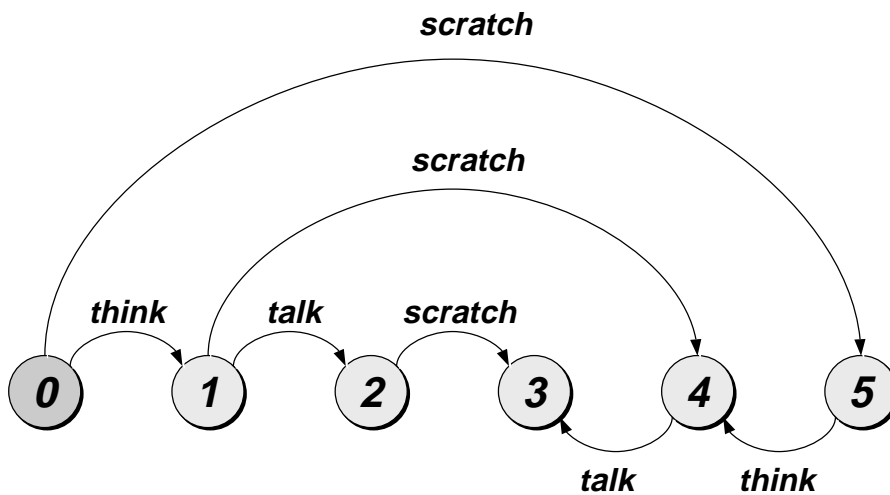


FSP Example

```
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.*

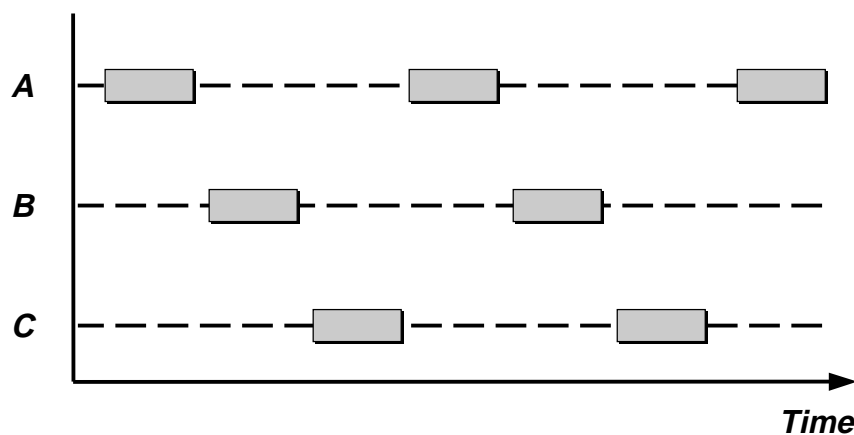
© Wolfgang Emmerich, 1998/99

13



Interleaved Model of Concurrency

■ Executing 3 processes on 1 processor:



© Wolfgang Emmerich, 1998/99

14



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*