

Systems, Networks & Concurrency 2020



Uwe R. Zimmer - The Australian National University

Systems, Networks & Concurrency 2020



Uwe R. Zimmer - The Australian National University

Organization & Contents

what is offered here?



Fundamentals & Overview
as well as perspectives, paths, methods, implementations, and open questions

of/into/for/about

Concurrent & Distributed Systems

© 2020 Uwe R. Zimmer, The Australian National University

page 3 of 758 "Organization & Contents" up to page 179

Organization & Contents


who could be interested in this?

anybody who ...

- ... wants to work with **real-world** scale computer systems
- ... would like to learn how to **analyse and design operational and robust systems**
- ... would like to understand more about the existing trade-off between *theory, the real-world, traditions, and pragmatism* in computer science
- ... would like to understand why **concurrent systems** are an **essential basis** for most contemporary devices and systems

© 2020 Uwe R. Zimmer, The Australian National University


page 4 of 758 "Organization & Contents" up to page 179



Organization & Contents


who are these people? – introductions

Uwe R. Zimmer & Charles Martin
 Abigail (Abi) Thomas, Aditya Chilukuri,
 Brent Schuetze, Calum Snowdon, Chimmay Garg, Felix Friedlander
 Johannes (Johnny) Schmalz, Nicholas Philip Miehlsbradt,
 Tommy Liu, William (Will) Cashman & Yaya Lu



© 2020 Uwe R. Zimmer, The Australian National University

page 5 of 758 | "Organization & Contents" up to page 191



Organization & Contents

Text book for the course


[Ben-Ari06]
 M. Ben-Ari
Principles of Concurrent and Distributed Programming
 2006, second edition, Prentice-Hall, ISBN 0-13-711821-X

Many algorithms and concepts for the course are in there
 – **but not all!**

References for specific aspects of the course are provided during the course and are found on our web-site.

© 2020 Uwe R. Zimmer, The Australian National University

page 7 of 758 | "Organization & Contents" up to page 191



Organization & Contents

how will this all be done?

Lectures:

- 2x 1.5 hours lectures per week ... all the nice stuff Tuesday 12:00 & Friday 11:00 (all live on-line)

Laboratories:

- 3 hours per week ... all the rough and action stuff time slots: on our web-site -enrolment: <https://cs.anu.edu.au/streams/> (open since last Monday, more slots today)

Resources:

- Introduced in the lectures and collected on the course page: <https://cs.anu.edu.au/courses/comp2310/> ... as well as schedules, slides, sources, links to forums, etc. pp. ... keep an eye on this page!

Assessment (for discussion):

- Exam at the end of the course (50%) plus one hurdle lab in week 4 (5%) plus two assignments (15% + 15%) plus one mid-semester exam (15%)

© 2020 Uwe R. Zimmer, The Australian National University

page 6 of 758 | "Organization & Contents" up to page 191



Organization & Contents

Topics

Language refresher [3]

1. Concurrency [3]
2. Mutual exclusion [2]
3. Communication & Synchronization [4]
4. Non-determinism [2]
5. Data Parallelism [1]
6. Scheduling [2]
7. Safety and liveness [4]
8. Distributed systems [4]
9. Architectures [1]

© 2020 Uwe R. Zimmer, The Australian National University

page 8 of 758 | "Organization & Contents" up to page 191



Organization & Contents

Topics

1. **Concurrency** [3]
 - 1.1. Forms of concurrency [1]
 - Coupled dynamical systems
 - 1.2. Models and terminology [1]
 - Abstractions
 - Interleaving
 - Atomicity
 - Proofs in concurrent and distributed systems
 - 1.3. Processes & threads [1]
 - Basic definitions
 - Process states
 - Implementations
2. **Mutual exclusion** [2]
3. **Communication & Synchronization** [4]
4. **Non-determinism** [2]
5. **Data Parallelism** [1]
6. **Scheduling** [2]
7. **Safety and liveness** [4]
8. **Distributed systems** [4]
9. **Architectures** [1]



Organization & Contents

Topics

1. **Concurrency** [3]
2. **Mutual exclusion** [2]
 - 2.1. by shared variables [1]
 - Failure possibilities
 - Dekker's algorithm
 - 2.2. by test-and-set hardware support [0.5]
 - Minimal hardware support
 - 2.3. by semaphores [0.5]
 - Dijkstra definition
 - OS semaphores
3. **Communication & Synchronization** [4]
4. **Non-determinism** [2]
5. **Data Parallelism** [1]
6. **Scheduling** [2]
7. **Safety and liveness** [2]
8. **Distributed systems** [4]
9. **Architectures** [1]



Organization & Contents

Topics

1. **Concurrency** [3]
2. **Mutual exclusion** [2]
3. **Communication & Synchronization** [4]
 - 3.1. Shared memory synchronization [2]
 - Semaphores
 - Cond. variables
 - Conditional critical regions
 - Monitors
 - Protected objects
 - 3.2. Message passing [2]
 - Asynchronous / synchronous
 - Remote invocation / rendezvous
 - Message structure
 - Addressing
4. **Non-determinism** [2]
5. **Data Parallelism** [1]
6. **Scheduling** [2]
7. **Safety and liveness** [4]
8. **Distributed systems** [4]
9. **Architectures** [1]



Organization & Contents

Topics

1. **Concurrency** [3]
2. **Mutual exclusion** [2]
3. **Condition synchronization** [4]
4. **Non-determinism** [2]
 - 4.1. Correctness under non-determinism [1]
 - Forms of non-determinism
 - Non-determinism in concurrent/ distributed systems
 - Is consistency/correctness plus non-determinism a contradiction?
 - 4.2. Select statements [1]
 - Forms of non-deterministic message reception
5. **Data Parallelism** [1]
6. **Scheduling** [2]
7. **Safety and liveness** [2]
8. **Distributed systems** [4]
9. **Architectures** [1]



Organization & Contents

Topics

1. *Concurrency* [3]
2. *Mutual exclusion* [2]
3. *Condition synchronization* [4]
4. *Non-determinism* [2]
5. *Data Parallelism* [1]
- 5.1. Data-Parallelism
 - Vectorization
 - Reduction
 - General data-parallelism
- 5.2. Examples
 - Image processing
 - Cellular automata
6. *Scheduling* [2]
7. *Safety and liveness* [2]
8. *Distributed systems* [4]
9. *Architectures* [1]



Organization & Contents

Topics

1. *Concurrency* [3]
2. *Mutual exclusion* [2]
3. *Condition synchronization* [4]
4. *Non-determinism* [2]
5. *Data Parallelism* [1]
6. *Scheduling* [2]
- 6.1. Problem definition and design space [1]
 - Which problems are addressed / solved by scheduling?
- 6.2. Basic scheduling methods [1]
 - Assumptions for basic scheduling
 - Basic methods
7. *Safety and liveness* [2]
8. *Distributed systems* [4]
9. *Architectures* [1]



Organization & Contents

Topics

1. *Concurrency* [3]
2. *Mutual exclusion* [2]
3. *Condition synchronization* [4]
4. *Non-determinism* [2]
5. *Data Parallelism* [1]
6. *Scheduling* [2]
7. *Safety and liveness* [2]
- 7.1. Safety properties
 - Essential time-independent safety properties
- 7.2. Livelocks, fairness
 - Forms of livelocks
 - Classification of fairness
- 7.3. Deadlocks
 - Detection
 - Avoidance
 - Prevention (& recovery)
- 7.4. Failure modes
- 7.5. Idempotent & atomic operations
 - Definitions
8. *Distributed systems* [4]
9. *Architectures* [1]



Organization & Contents

Topics

1. *Concurrency* [3]
2. *Mutual exclusion* [2]
3. *Condition synchronization* [4]
4. *Non-determinism* [2]
5. *Data Parallelism* [1]
6. *Scheduling* [2]
7. *Safety and liveness* [3]
8. *Distributed systems* [4]
- 8.1. Networks [1]
 - OSI model
 - Network implementations
- 8.2. Global times [1]
 - Synchronized clocks
 - Logical clocks
- 8.3. Distributed states [1]
 - Consistency
 - Snapshots
 - Termination
- 8.4. Distributed communication [1]
 - Name spaces
 - Multi-casts
 - Elections
 - Network identification
- Dynamical groups
- 8.5. Distributed safety and liveness [1]
 - Distributed deadlock detection
- 8.6. Forms of distribution/redundancy [1]
 - computation
 - memory
 - operations
- 8.7. Transactions [2]
9. *Architectures* [1]



Organization & Contents

Topics

1. **Concurrency** [3]
 - 9.1. Hardware architecture
 - From switches to registers and addressers
 - CPU architecture
 - Hardware concurrency
2. **Mutual exclusion** [2]
3. **Condition synchronization** [4]
4. **Non-determinism** [2]
5. **Data Parallelism** [1]
6. **Scheduling** [2]
7. **Safety and liveness** [2]
8. **Distributed systems** [4]
9. **Architectures** [1]



Organization & Contents

24 Lectures

1. **Concurrency** [3]
 - 1.1. Forms of concurrency [1]
 - Compiled dynamical systems
 - 1.2. Abstraction and terminology [1]
 - Interleaving
 - Atomicity
 - Precedence in concurrent and distributed systems
 - 1.3. Processes & threads [1]
 - Basic definitions
 - Process states
 - Implementations
2. **Mutual exclusion** [2]
 - 2.1. by shared variables [1]
 - Failure possibilities
 - Dekker's algorithm
 - 2.2. by read-and-set hardware support [0.5]
 - Hardware support
 - 2.3. by semaphores [0.5]
 - Dijkstra definition
 - OS semaphores
3. **Communication & Synchronization** [4]
 - 3.1. Shared memory synchronization [2]
 - Cond. variables
 - 3.2. Examples
 - Image processing
 - Cellular automata
4. **Non-determinism** [2]
 - 4.1. Correctness under non-determinism [1]
 - Forms of non-determinism
 - Non-determinism in concurrent and distributed systems
 - Determinism vs. global non-determinism a contradiction?
 - 4.2. Select statements [1]
 - Forms of non-determinism
 - Message reception
5. **Data Parallelism** [1]
 - 5.1. Data Parallelism
 - Vectorization
 - General data-parallelism
 - 5.2. Examples
 - Image processing
 - Cellular automata
6. **Scheduling** [2]
 - 6.1. Problem definition and design space [1]
 - Which problem is to be addressed
 - Snapshots
 - Termination
 - 6.2. Basic scheduling methods [1]
 - Assumptions for basic scheduling
 - Basic methods
 - Multi-calls
7. **Safety and liveness** [2]
 - 7.1. Safety properties
 - Network identification
 - Distributed safety and liveness [1]
 - Distributed deadlock detection
 - 7.2. Liveness, fairness
 - Classification of fairness
 - 7.3. Deadlocks
 - Detection
 - Avoidance
 - 7.4. Failure modes (6. recovery)
 - Definitions
8. **Distributed systems** [4]
 - 8.1. Networks [1]
 - Network implementations
 - 8.2. Global times [1]
 - Synchronized clocks
9. **Architectures** [1]
 - 9.1. Hardware architecture
 - From switches to registers and addressers
 - Hardware concurrency
 - 9.2. Language architecture
 - Chapel
 - Occam
 - Actor
 - Ada
 - C++



Organization & Contents

Laboratories & Assignments

Laboratories [11]		Assignments [2]		Examinations [3]	
1. Structured Programming [2] <ul style="list-style-type: none"> • Program structures • Data structures 	5. Communicating Tasks [1] <ul style="list-style-type: none"> • Rendezvous 	1. Concurrent programming [15%] <ul style="list-style-type: none"> • Programming task involving: <ul style="list-style-type: none"> • Mutual exclusion • Synchronization • Message passing 	1. Hurdle check [5%] <ul style="list-style-type: none"> • Week 4 lab exam 	2. Mid-semester check [15%] <ul style="list-style-type: none"> • Exam of Selftest 	3. Final exam [50%] <ul style="list-style-type: none"> • Examining the complete course
2. Tasks [1] <ul style="list-style-type: none"> • Generics • Abstract types 	6. Distributing Server [1] <ul style="list-style-type: none"> • Entry families • Request facility 	2. Concurrent programming in multi-core systems [15%] <ul style="list-style-type: none"> • Multi-core process creation, termination • Multi-core process communication 	Marking		
3. Protection [1] <ul style="list-style-type: none"> • Memory based synchronization 	7. Implicit Concurrency [1] <ul style="list-style-type: none"> • Synchronization 	The final mark is based on the assignments (30%) plus the examinations (65%) plus the lab mark (5%).			
4. Task Lifetimes [1] <ul style="list-style-type: none"> • Creation • Termination 	8. Synchronized Data [1] <ul style="list-style-type: none"> • Synchronization 	10. Pipelines [1]			