what is offered here?

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

**Concurrent & Distributed Systems**

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
who are these people? – introductions

Uwe R. Zimmer & Charles Martin

Abigail (Abi) Thomas, Brent Schuetze, Calum Snowdon, Chinmay Garg, David (Dave) Quarel, Johannes (Johnny) Schmalz, Joshua (Josh) Gilbert, Michael Bennett, Peter Baker, Rohan McLure, Timothy (Tim) Lee, Tommy Lui, William (Bill) Cashman, Yaya Lu

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.

Text book for the course

[Ben-Ari06]
M. Ben-Ari
Principles of Concurrent and Distributed Programming

Lectures:
- 2 x 1.5 hours lectures per week … all the nice stuff
  Tuesday 14:00 (Cinema) & Wednesday 10:00 (Copeland Theatre)

Laboratories:
- 2 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 Friday, 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%)

Topics

Language refresher [3]
1. Concurrency [3]
5. Data Parallelism [1]
7. Safety and liveness [2]
8. Distributed systems [4]
9. Architectures [1]
<table>
<thead>
<tr>
<th>Topics</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concurrency</td>
<td>3</td>
</tr>
<tr>
<td>Mutual exclusion</td>
<td>2</td>
</tr>
<tr>
<td>Condition synchronization</td>
<td>4</td>
</tr>
<tr>
<td>Non-determinism in concurrent systems</td>
<td>2</td>
</tr>
<tr>
<td>Scheduling</td>
<td>2</td>
</tr>
<tr>
<td>Safety and liveness</td>
<td>3</td>
</tr>
<tr>
<td>Architectures for CDS</td>
<td>1</td>
</tr>
<tr>
<td>Distributed systems</td>
<td>7</td>
</tr>
</tbody>
</table>

**1. Concurrency**
- Forms of concurrency [1]
- Coupled dynamical systems [1]
- Models and terminology [1]
- Proofs in concurrent and distributed systems [1]
- Processes & threads [1]
  - Basic definitions
  - Process states
  - Implementations

**2. Mutual exclusion**
- by shared variables [1]
  - Failure possibilities
  - Dekker's algorithm
- by test-and-set hardware [0.5]
  - Minimal hardware support
- by semaphores [0.5]
  - Dijkstra definition
  - OS semaphores

**3. Condition synchronization**
- by semaphores [0.5]

**4. Non-determinism in concurrent systems**
- in concurrent systems [1]
  - Forms of non-determinism
  - Non-determinism in concurrent systems [1]
  - Is consistency/correctness plus non-determinism a contradiction?
- Select statements [1]
  - Forms of non-deterministic message reception

**5. Scheduling**

**6. Safety and liveness**

**7. Architectures for CDS**

**8. Distributed systems**
1. Concurrency [3]
5. Scheduling [2]

7. Architectures for CDS [1]
8. Distributed systems [7]
24 Lectures

1. Concurrency (3)
   1.1. Forms of concurrency (1)
   1.2. Models and terminology (1)
   1.3. Processes & threads (1)

2. Mutual exclusion (2)
   2.1. by shared variables (1)
   2.2. by test-and-set hardware support (0.5)
   2.3. by semaphores (0.5)

3. Condition synchronization (4)
   3.1. Shared memory synchronization (2)
   3.2. Message passing (2)

4. Non-determinism in concurrent systems (2)
   4.1. Correctness under non-determinism (1)
   4.2. Select statements (1)

5. Scheduling (2)
   5.1. Problem definition and design space (1)
   5.2. Basic scheduling methods (1)

6. Safety and liveness (3)
   6.1. Safety properties (1)
   6.2. Deadlocks: Cycles (1)
   6.3. Deadlocks: Circuits (1)
   6.4. Petri nets (1)
   6.5. Interleaved atomic operation (1)

7. Architectures for CDS (1)
   7.1. Hardware architectures (1)
   7.2. Software architectures (1)

8. Distributed systems (7)
   8.1. Networks (5)
   8.2. Global states (2)
   8.3. Distributed states (2)
   8.4. Distributed communication (1)

9. Distributed deadlock detection (1)
10. Forms of distribution/redundancy (1)
   10.1. computation (1)
   10.2. memory (1)
   10.3. operations (1)

Laboratories & Assignments

Laboratories

2. Concurrent programming (3)
   2.1. Synchronization
   2.2. Remote invocation
   2.3. Client-Server architectures

Assignments

2. Concurrent programming in multicore systems (15%)
   3.1. Multi-core process creation, termination
   3.2. Multi-core process communication

Examinations

1. Mid-term check (10%)
   2. Final exam (55%)

Marking

The final mark is based on the assignments (30%) plus the examinations (65%) plus the lab mark (5%).