Practical Information

TDA384/DIT391

Principles of Concurrent Programming

Nir Piterman and Gerardo Schneider

Chalmers University of Technology | University of Gothenburg

SP1 2023/2024

Canvas Room and Course Website

Make sure to regularly check the Canvas Room and Course Website:

Canvas	Announcements, discussion forum, videos https://chalmers.instructure.com/courses/25543
Website	Lectures, labs, exams, … http://www.cse.chalmers.se/edu/course/TDA384_LP1

These should be your primary sources of information about the course.

Discussion Forum

Use the Canvas discussion forum for questions and discussions of general interest to the course:

https://chalmers.instructure.com/courses/25543/discussion_topics

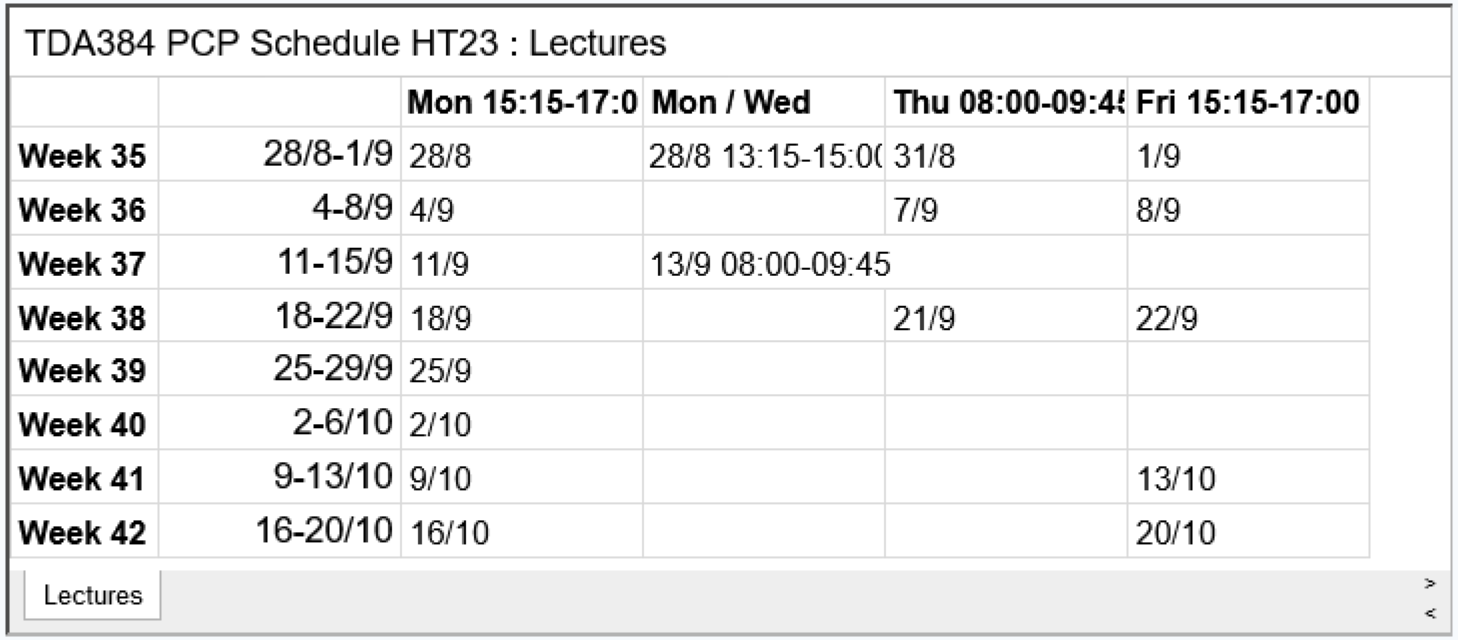
The forum URL is of course linked from the course website.

Do not share solutions to labs on Canvas

(or anywhere else) !!!

Lectures

- Check out TimeEdit.
- All lectures are given in HC2
  - Well done! You are here!



TDA384 PCP Schedule HT23 : Lectures

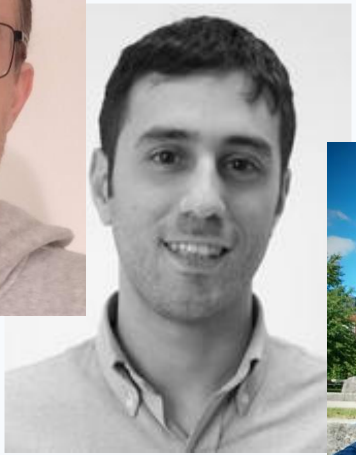
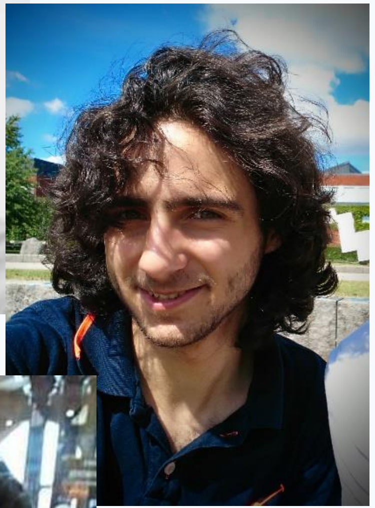
		Mon 15:15-17:0	Mon / Wed	Thu 08:00-09:4	Fri 15:15-17:00
Week 35	28/8-1/9	28/8	28/8 13:15-15:0	31/8	1/9
Week 36	4-8/9	4/9		7/9	8/9
Week 37	11-15/9	11/9	13/9 08:00-09:45		
Week 38	18-22/9	18/9		21/9	22/9
Week 39	25-29/9	25/9			
Week 40	2-6/10	2/10			
Week 41	9-13/10	9/10			13/10
Week 42	16-20/10	16/10			20/10

Lectures

Labs

- Mixing physical and online labs.
- Lab assistance requests –
  - Create a Zoom meeting w.o. password
  - Put support requests on Waglys
  - Name for support request (limited to 20 chars):
    - Zoom meeting ID (not link)
    - Add Chalmers ID (if possible)
- Demo signup –
  - A doodle with available slots will be posted on the appropriate lab page before each deadline
  - Create a Zoom meeting (w.o. password)
  - Register the day **before** the demos
  - Use group ID + Zoom meeting ID as name in the poll
  - Be on Zoom 5 minutes before your time and be ready to run the demo

The teaching team

Lecturer/Examiner

- Nir Piterman

Teaching assistants (TAs)

- William Hughes
- David Lidell
- Luca di Stefano
- Ghaith Abou Dan
- Ali Saaeddin

If you have questions

- Ask them during lectures and lab sessions.
- Post them on the discussion forum.
- Send an email to pcp-teachers@lists.chalmers.se (of personal nature)
- Book an appointment with the teacher or TAs (by email).

Protip: options 1 & 2 are quicker than options 3 & 4.

Student Representatives

Chalmers

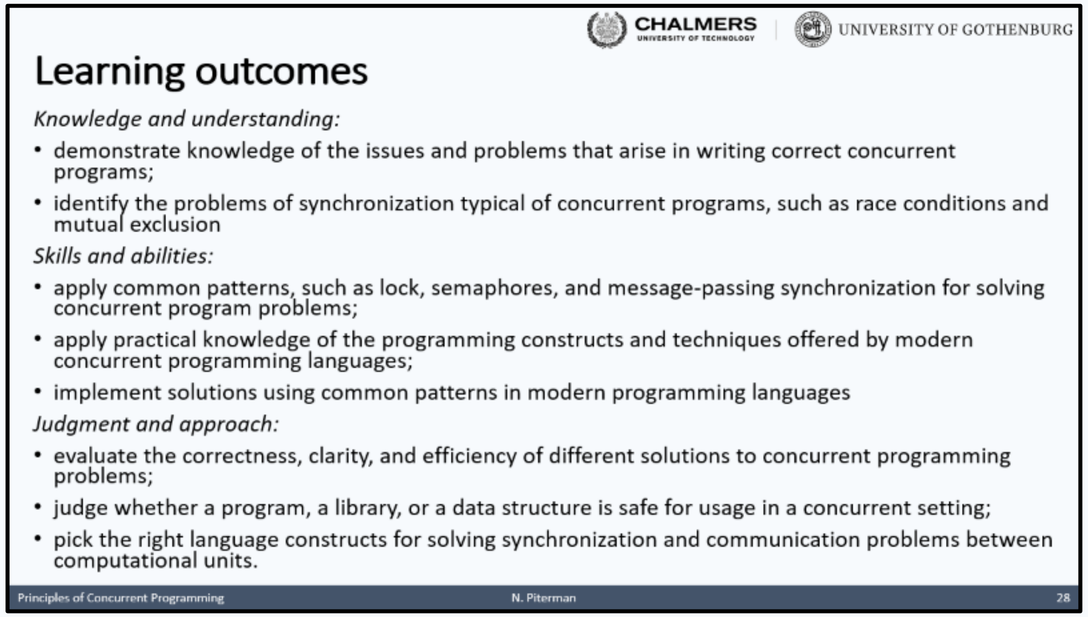
- Not communicated yet

GU

- You?

Main Learning Goals

- By the end of the course you should be able to
  - Understand the problems common to concurrent and parallel systems,
  - Demonstrate techniques and patterns to reason about and write correct and efficient concurrent programs,
  - Apply those techniques and patterns in modern programming languages.



Learning outcomes

Knowledge and understanding:

- demonstrate knowledge of the issues and problems that arise in writing correct concurrent programs;
- identify the problems of synchronization typical of concurrent programs, such as race conditions and mutual exclusion

Skills and abilities:

- apply common patterns, such as lock, semaphores, and message-passing synchronization for solving concurrent program problems;
- apply practical knowledge of the programming constructs and techniques offered by modern concurrent programming languages;
- implement solutions using common patterns in modern programming languages

Judgment and approach:

- evaluate the correctness, clarity, and efficiency of different solutions to concurrent programming problems;
- judge whether a program, a library, or a data structure is safe for usage in a concurrent setting;
- pick the right language constructs for solving synchronization and communication problems between computational units.

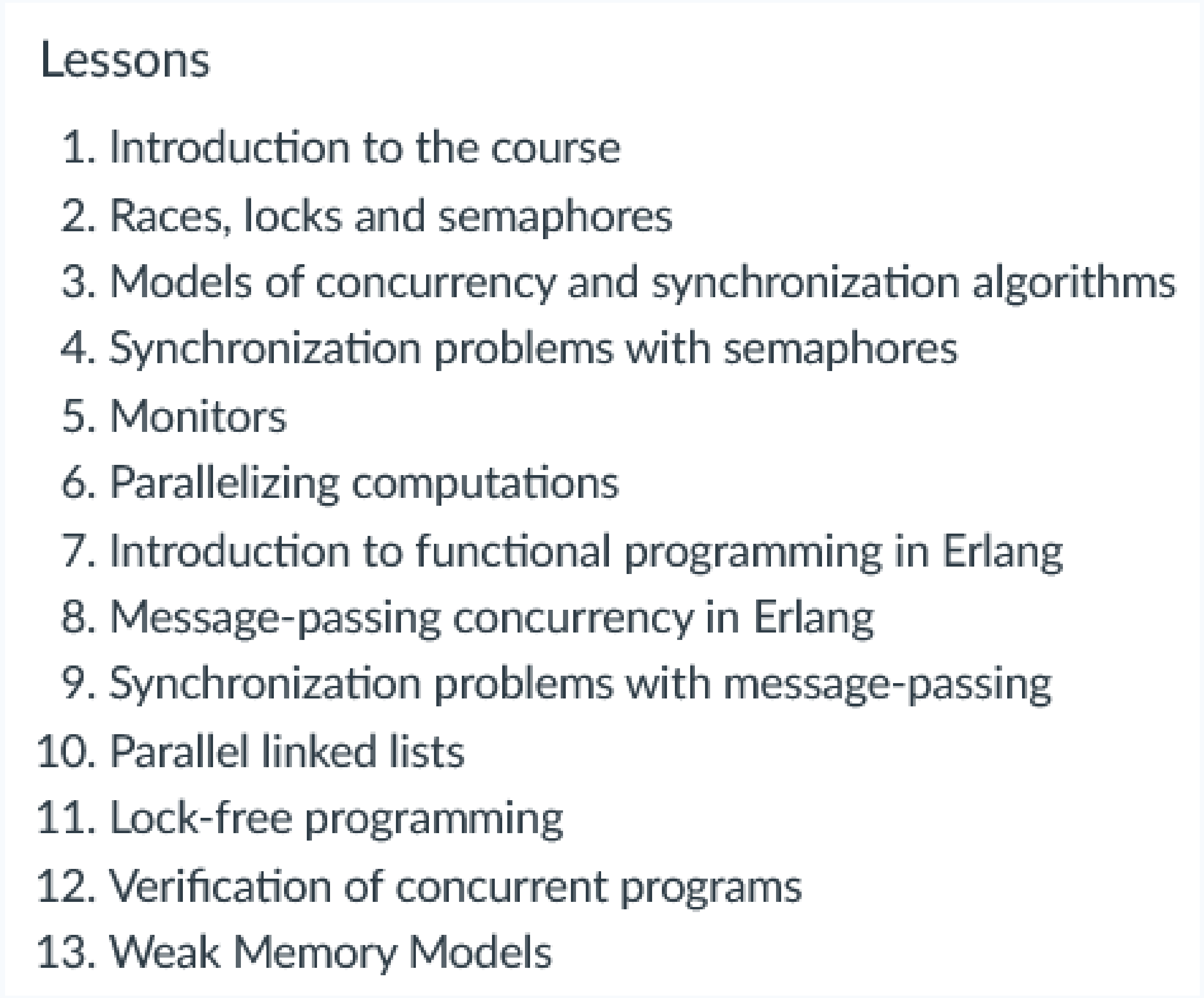
Principles of Concurrent Programming N. Piterman 28

CHALMERS UNIVERSITY OF TECHNOLOGY | UNIVERSITY OF GOTHENBURG

Overview of the Course

- Introduction to concurrency.
- Part 1. Classic, shared-memory concurrency in Java:
  - java threads,
  - locks, semaphores, and monitors,
  - Fork/join parallelism.
- Part 2. Message-passing concurrency:
  - Erlang and the actor model.
- Part 3. lock-free programming:
  - lock-free programming.

Lectures



Lessons

1. Introduction to the course
2. Races, locks and semaphores
3. Models of concurrency and synchronization algorithms
4. Synchronization problems with semaphores
5. Monitors
6. Parallelizing computations
7. Introduction to functional programming in Erlang
8. Message-passing concurrency in Erlang
9. Synchronization problems with message-passing
10. Parallel linked lists
11. Lock-free programming
12. Verification of concurrent programs
13. Weak Memory Models

- 12 lessons
- 2 Tutorials (Java and Erlang)
- Guest lecture?
- Revision
- Some lessons will take less/more time

Labs

There will be one preparation lab and three "real" labs – one for each part of the course:

1. Trainspotting (Java)

2. A-mazed (Java)

3. CCHAT (Erlang)

Descriptions of the labs, deadlines, and rules are on the website

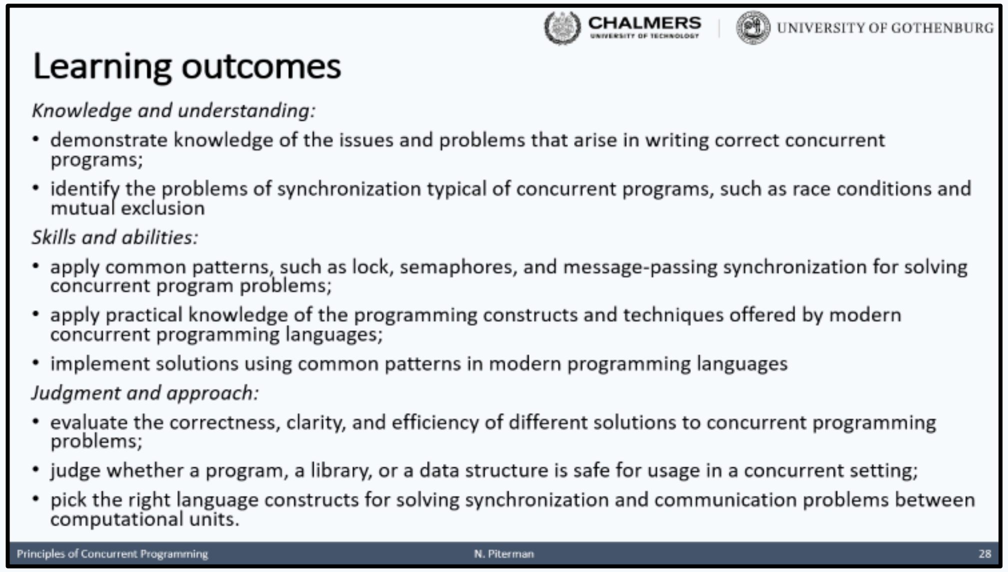
- **Lab 0**: Set up and register your group (2 students) in Fire
- Make sure to check the lab/room schedule on the website

Do not share solutions to labs on Canvas (or anywhere else) !!!

Tutorials

There will be 2 tutorials

1. Java tutorial (Thursday ( ))
2. Erlang tutorial (Sep 22nd ( ))



Learning outcomes

Knowledge and understanding:

- demonstrate knowledge of the issues and problems that arise in writing correct concurrent programs;
- identify the problems of synchronization typical of concurrent programs, such as race conditions and mutual exclusion

Skills and abilities:

- apply common patterns, such as lock, semaphores, and message-passing synchronization for solving concurrent program problems;
- apply practical knowledge of the programming constructs and techniques offered by modern concurrent programming languages;
- implement solutions using common patterns in modern programming languages

Judgment and approach:

- evaluate the correctness, clarity, and efficiency of different solutions to concurrent programming problems;
- judge whether a program, a library, or a data structure is safe for usage in a concurrent setting;
- pick the right language constructs for solving synchronization and communication problems between computational units.

Principles of Concurrent Programming N. Piterman 28

Learning outcomes

Knowledge and understanding:

- demonstrate knowledge of the issues and problems that arise in writing correct concurrent programs;
- identify the problems of synchronization typical of concurrent programs, such as race conditions and mutual exclusion

Skills and abilities:

- apply common patterns, such as lock, semaphores, and message-passing synchronization for solving concurrent program problems;
- apply practical knowledge of the programming constructs and techniques offered by modern concurrent programming languages;
- implement solutions using common patterns in modern programming languages

Judgment and approach:

- evaluate the correctness, clarity, and efficiency of different solutions to concurrent programming problems;
- judge whether a program, a library, or a data structure is safe for usage in a concurrent setting;
- pick the right language constructs for solving synchronization and communication problems between computational units.

Learning outcomes

Knowledge and understanding:

- demonstrate knowledge of the issues and problems that arise in writing correct concurrent programs;
- identify the probl s and mutual exclusion

Skills and abilities:

- apply common pa lving concurrent progr
- apply practical kn concurrent progr
- implement soluti

Judgment and appr

- evaluate the corr g problems;
- judge whether a program, a library, or a data structure is safe for usage in a concurrent setting;
- pick the right language constructs for solving synchronization and communication problems between computational units.

- Lesson 2
  - races
- Lesson 3
  - more interleaving and races
  - strong synchronization mechanisms – why?
- Lesson 4, Lesson 5, Lesson 9
  - challenges and their solution
- Lesson 6, Lesson 10, Lesson 11
  - high costs of synchronization and how to mitigate them
- Lesson 12
  - checking correctness
- Lesson 13
  - races again

Learning outcomes

Knowledge and understanding:

- demonstrate knowledge of the issues and problems that arise in writing correct concurrent programs;
- identify the problems of synchronization typical of concurrent programs, such as race conditions and mutual exclusion

Skills and abilities:

- apply common pattern
concurrent program pr
- apply practical knowle
concurrent programmi
- implement solutions u

Judgment and approach:

- evaluate the correctness, clarity, and efficiency of different solutions to concurrent programming problems;
- judge whether a program, a library, or a data structure is safe for usage in a concurrent setting;
- pick the right language constructs for solving synchronization and communication problems between computational units.

- Lesson 2
  - Races, critical sections
- Lesson 3
  - more interleaving and races
  - critical sections
- Lesson 4, Lesson 5, Lesson 9
  - challenges
- Lesson 13
  - races again

Learning outcomes

Knowledge and understanding:

- demonstrate knowledge of the issues and problems that arise in writing correct concurrent programs;
- identify the problems of synchronization typical of concurrent programs, such as race conditions and mutual exclusion

Skills and abilities:

- apply common patterns, such as lock, semaphores, and message-passing synchronization for solving concurrent program problems;
- apply practical knowle concurrent programmi
- implement solutions u

Judgment and approach:

- evaluate the correctne problems;
- judge whether a progr
- pick the right language computational units.

- Lesson 2, Lesson 3, Lesson 5
  - locks, semaphores, monitors
- Lesson 4, Lesson 5, Lesson 9
  - challenges and their solution
- Lesson 7, Lesson 9
  - message passing synchronization
- Lesson 6, Lesson 10, Lesson 11
  - high costs of synchronization and how to mitigate them
- Labs

Learning outcomes

Knowledge and understanding:

- demonstrate knowled
  programs;
- identify the problems d
  mutual exclusion

Skills and abilities:

- apply common patter g
  concurrent program problems;

- Lesson 2, Lesson 3
  - locks, semaphores, monitors
- Lesson 4, Lesson 5, Lesson 9
  - challenges and their solution
- Lesson 6, Lesson 10, Lesson 11
  - high costs of synchronization and how to mitigate them
- Labs

- apply practical knowledge of the programming constructs and techniques offered by modern concurrent programming languages;
- implement solutions using common patterns in modern programming languages

Judgment and approach:

- evaluate the correctness, clarity, and efficiency of different solutions to concurrent programming problems;
- judge whether a program, a library, or a data structure is safe for usage in a concurrent setting;
- pick the right language constructs for solving synchronization and communication problems between computational units.

Learning outcomes

Knowledge and understanding:

- demonstrate knowled
  programs;
- identify the problems d
  mutual exclusion

Skills and abilities:

- apply common patter g
  concurrent program p
- apply practical knowle
  concurrent programming languages;

- Lesson 2, Lesson 3
  - representation of concurrent programs
- Lesson 4, Lesson 5, Lesson 9
  - identify problems
- Lesson 6, Lesson 10, Lesson 11
  - work on efficiency
- Lesson 12
  - Verification
- Labs

- implement solutions using common patterns in modern programming languages

Judgment and approach:

- evaluate the correctness, clarity, and efficiency of different solutions to concurrent programming problems;
- judge whether a program, a library, or a data structure is safe for usage in a concurrent setting;
- pick the right language constructs for solving synchronization and communication problems between computational units.

Learning outcomes

Knowledge and understanding:

- demonstrate knowledge of the issues and problems that arise in writing correct concurrent programs;
- identify the problems of synchronization typical of concurrent programs, such as race conditions and mutual exclusion

Skills and abilities:

- apply common patter g
  concurrent program p
- apply practical knowl
  concurrent programm
- implement solutions

- Lesson 2, Lesson 3
  - different constructs of synchronization
- Lesson 4, Lesson 5, Lesson 9
  - patterns and applications of solutions
- Lesson 7, Lesson 8
  - message-passing concurrency
- Lesson 10, Lesson 11
  - alternative approaches to synchronization

Judgment and approach:

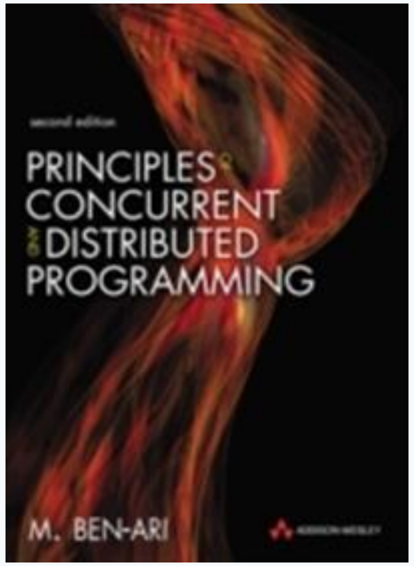
- evaluate the correctness, clarity, and efficiency of different solutions to concurrent programming problems;
- judge whether a program, a library, or a data structure is safe for usage in a concurrent setting;
- pick the right language constructs for solving synchronization and communication problems between computational units.

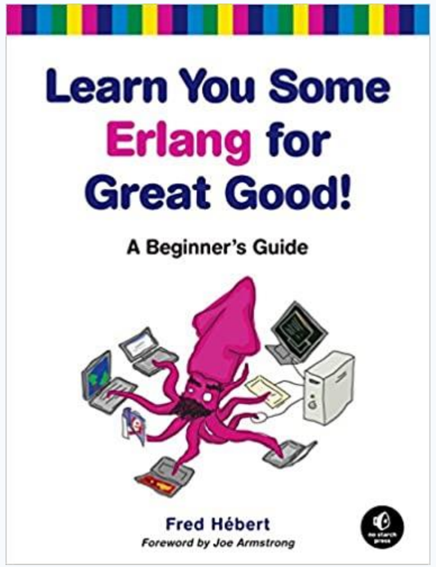
Slides and Reading Material

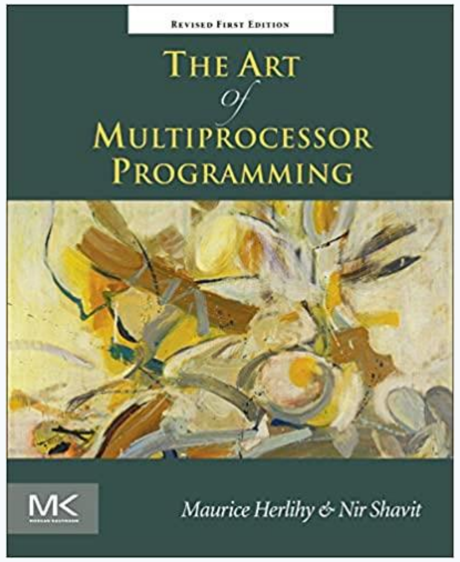
Lecture slides: will be on the website.

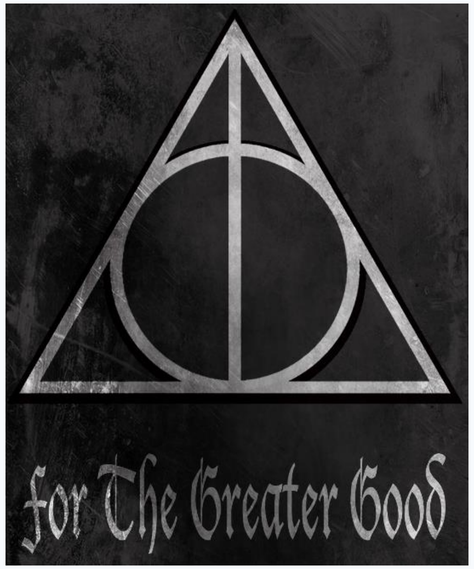
Books:

- Ben-Ari: Principles of concurrent and distributed programming, 2nd edition.
- Hébert: Learn you some Erlang for great good (free online),
- Herlihy & Shavit: The art of multiprocessor programming




Learn You Some Erlang for Great Good!
A Beginner's Guide
Fred Hébert
Foreword by Joe Armstrong


Revised First Edition
The Art of Multiprocessor Programming
MK
Maurice Herlihy & Nir Shavit



Exam

- Open-book exam:
  - max. 2 textbooks,
  - max. 4 two-sided A4 sheets of notes (printed or handwritten),
  - an English dictionary.
- All topics in the lectures can be examined (except guest lectures).
- See exams of previous years for examples (on the website).
- Exam dates:
  - **25 October 2023**
  - **?? March 2024**, **22 August 2024** (re-exam)
- Check the website for updates!
- Exam grading: see the course website.

Computing Resources

- Install Java and Erlang/OTP on your computers.
- Try out the examples presented in class; the complete examples will be available on the website for each lecture.
- Lab 1 (Trainspotting) requires a simulator, which runs on the lab computers (Unix/Linux workstations).
- See the course website for instructions on how to
  - use the lab computers, and
  - set up Java & Erlang/OTP on your own computers.

There’s a lab on Thursday – What’s the point of that?

- Complete the setup assignment!
- Create the groups!
- Setup the train system!
- Start playing with it with sequential programs:
  - Have only one train.
  - Start and stop.
  - Check distances and speeds.
  - When is a train on a switch?
  - Make plans.

Erlang, Erlang, Erlang, …

- Start early!
- Install the Erlang environment.
- Start the online tutorial.
- Attend the Erlang Tutorial!
- Especially if never done functional programming before.
- Compared to previous years:
  - Swapped labs 2 and 3 – more time to learn Erlang
  - First lab support on the Friday after demo of lab 2

Course Evaluation

- Please remember to fill in the course evaluation ("*kursvärdering*") when the time comes!
  - Important feedback for us
  - To know what can be improved as well as what is working well

Changes this year

- Extend Introductory Lecture
- Course slides with no animation for notes
- Introduce learning outcomes of lessons
- Discuss more labs
- Reorganize lessons so that Erlang can be moved to the end
- Reduce coding in exam