CS 6868: Concurrent Programming

01 Introduction

KC Sivaramakrishnan

Spring 2026, IIT Madras

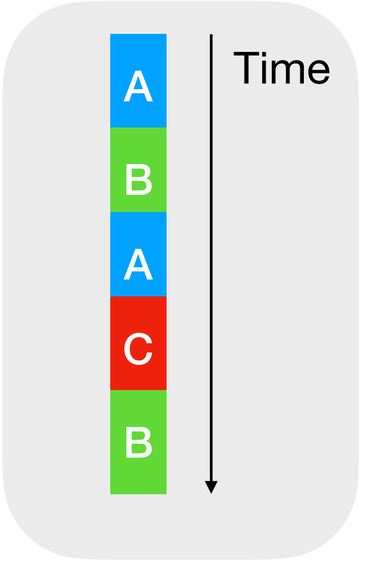
What is concurrency?

- We use concurrency in this course in the broad sense
  - Encompasses multiple definitions that people use in practice

What is concurrency?

- We use concurrency in this course in the broad sense
  - Encompasses multiple definitions that people use in practice
- **“Concurrency”**
  - ***Interleaved*** executions of tasks in time
  - Examples — Asynchronous I/O, GUI in JavaScript, interrupt-driven embedded systems, etc.
  - ***No need for multiple cores!***
  - ***Challenges*** — non-determinism, many different schedules complicate reasoning, and efficiency, etc.

Concurrency

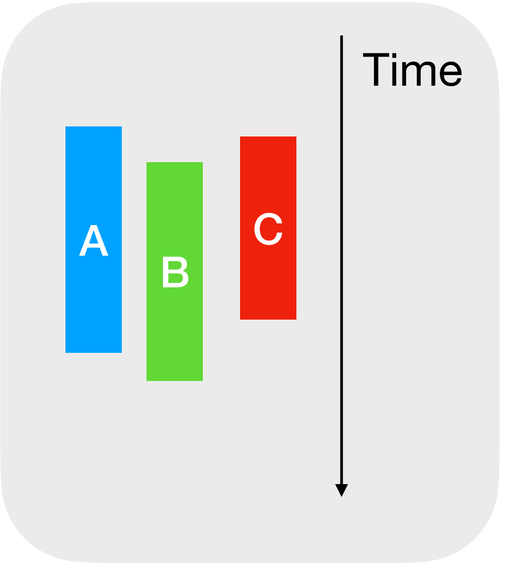


Interleaved execution

What is concurrency?

- We use concurrency in this course in the broad sense
  - Encompasses multiple definitions that people use in practice
- **Parallelism**
  - Multiple ***hardware execution units*** — Multicore CPUs, GPUs, parallelism across machines (such as supercomputers, GPU farms)
  - Goal is ***speed*** with correctness
  - ***Challenges*** — all the concurrency challenges + relaxed memory behaviours, performance scaling challenges, and debugging challenges.

Parallelism

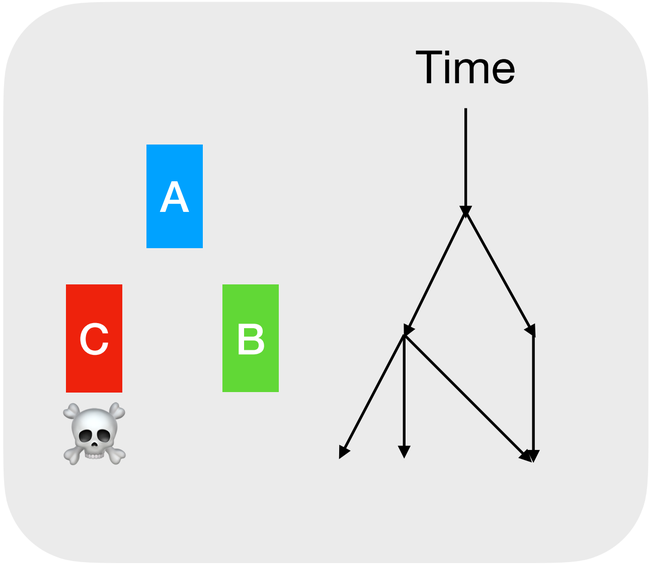


Simultaneous execution

What is concurrency?

- We use concurrency in this course in the broad sense
  - Encompasses multiple definitions that people use in practice
- **Distribution**
  - Execution is distributed across many ***loosely-coupled*** machines
    - There is often ***no shared global clock*** across computations
    - A subset of computations may ***crash/fail.***
  - **Goal** — performance, resilience in the presence of partial failures.
  - Examples — the Internet, blockchains, etc.
  - ***Challenges*** — All the problems of concurrency + failures + performance scaling + debugging

Distribution



Why this course?

- Concurrency is ***everywhere***
  - Embedded Systems, OS, Multicore Hardware, GPUs, Reactive Systems, Cloud, …

Why this course?

- Concurrency is ***everywhere***
  - Embedded Systems, OS, Multicore Hardware, GPUs, Reactive Systems, Cloud, …
- This course equips you with the skills to ***reason*** and ***program*** under concurrency.
  - **Models:** Histories, Linearizability, Memory Models, Data races
  - **Mechanisms:** Mutual Exclusion, Atomics, Lockfree Algorithms, Strong type systems
  - **Abstractions:** Concurrency Monads, Effect Handlers, Structured Concurrency
  - **Programming:** Implementing concurrency, safety, performance

Why this course?

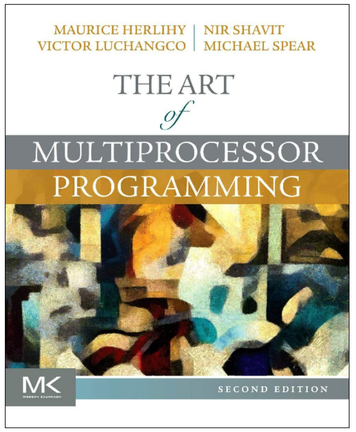
- Concurrency is ***everywhere***
  - Embedded Systems, OS, Multicore Hardware, GPUs, Reactive Systems, Cloud, …
- This course equips you with the skills to ***reason*** and ***program*** under concurrency.
  - **Models:** Histories, Linearizability, Memory Models, Data races
  - **Mechanisms:** Mutual Exclusion, Atomics, Lockfree Algorithms, Strong type systems
  - **Abstractions:** Concurrency Monads, Effect Handlers, Structured Concurrency
  - **Programming:** Implementing concurrency, safety, performance
- The course will not cover
  - Distribution
  - Performance optimisations

Course Outline

- **Parallelism**
  - Mutual Exclusion, Concurrent Objects, Relaxed Memory Models, Spin Locks, Contention, Blocking Synchronisation, Fine-grained Concurrent Data Structures

Course Outline

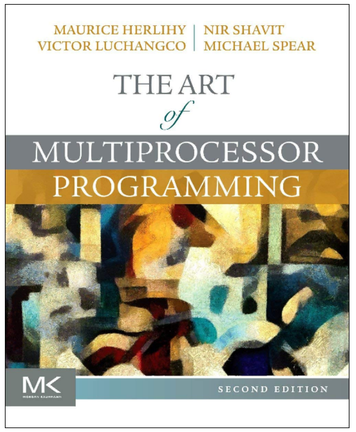
- **Parallelism**
  - Mutual Exclusion, Concurrent Objects, Relaxed Memory Models, Spin Locks, Contention, Blocking Synchronisation, Fine-grained Concurrent Data Structures



2nd Edition

Course Outline

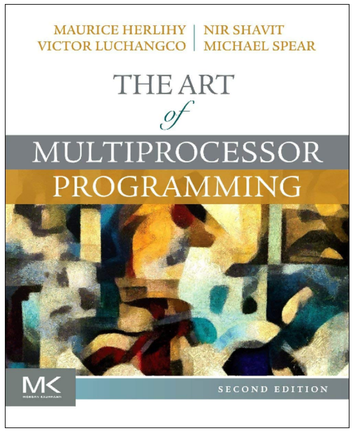
- **Parallelism**
  - Mutual Exclusion, Concurrent Objects, Relaxed Memory Models, Spin Locks, Contention, Blocking Synchronisation, Fine-grained Concurrent Data Structures
- **Concurrency**
  - Continuations, Concurrency Monads, Effect handlers, Schedulers, Concurrent data structures, Asynchronous I/O



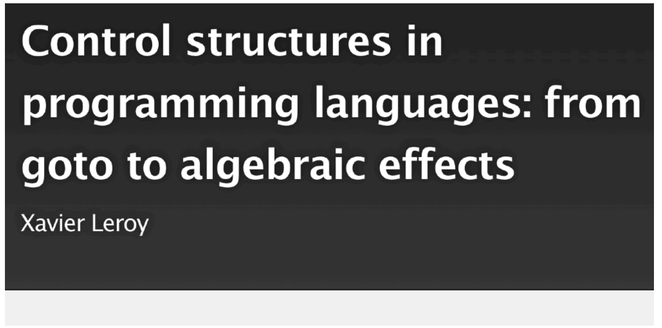
2nd Edition

Course Outline

- **Parallelism**
  - Mutual Exclusion, Concurrent Objects, Relaxed Memory Models, Spin Locks, Contention, Blocking Synchronisation, Fine-grained Concurrent Data Structures
- **Concurrency**
  - Continuations, Concurrency Monads, Effect handlers, Schedulers, Concurrent data structures, Asynchronous I/O



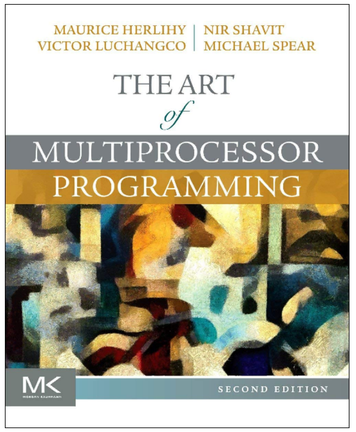
2nd Edition



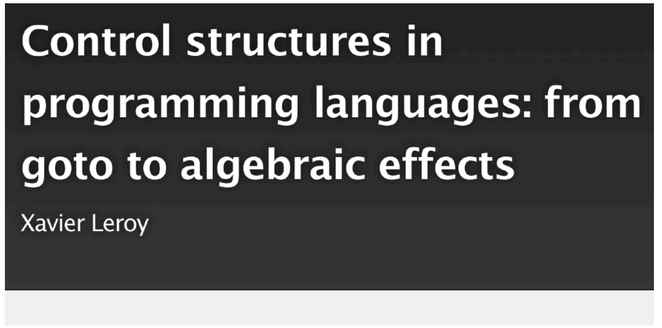
https://xavierleroy.org/control-structures/

Course Outline

- **Parallelism**
  - Mutual Exclusion, Concurrent Objects, Relaxed Memory Models, Spin Locks, Contention, Blocking Synchronisation, Fine-grained Concurrent Data Structures
- **Concurrency**
  - Continuations, Concurrency Monads, Effect handlers, Schedulers, Concurrent data structures, Asynchronous I/O
- **Safe Concurrent Programming**
  - Modes, DRF Parallel Programming



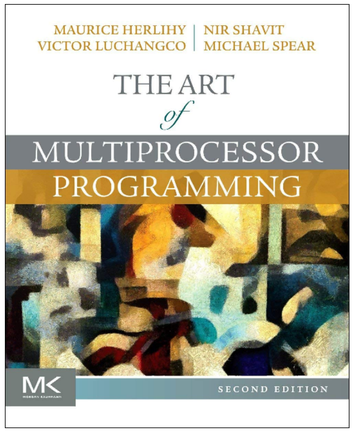
2nd Edition



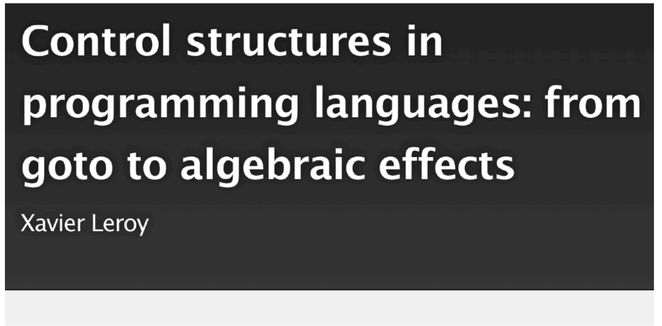
https://xavierleroy.org/control-structures/

Course Outline

- **Parallelism**
  - Mutual Exclusion, Concurrent Objects, Relaxed Memory Models, Spin Locks, Contention, Blocking Synchronisation, Fine-grained Concurrent Data Structures
- **Concurrency**
  - Continuations, Concurrency Monads, Effect handlers, Schedulers, Concurrent data structures, Asynchronous I/O
- **Safe Concurrent Programming**
  - Modes, DRF Parallel Programming
- **First iteration of the course**
  - Course content neither sound nor complete



2nd Edition



https://xavierleroy.org/control-structures/

Course Language

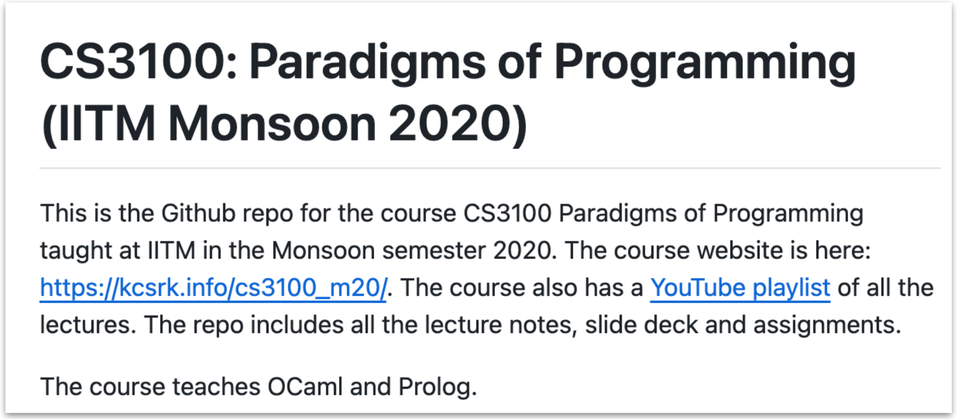
- ***OCaml*** will be the language of choice for the course

Course Language

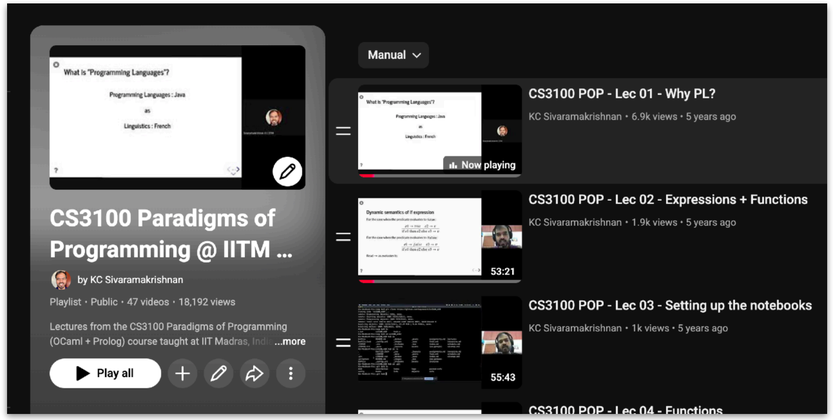
- ***OCaml*** will be the language of choice for the course
- **Strong Pre-requisite: CS3100 OCaml Parts**
  - *I will not cover OCaml basics in this course!*

Course Language

- ***OCaml*** will be the language of choice for the course
- **Strong Pre-requisite: CS3100 OCaml Parts**
  - *I will not cover OCaml basics in this course!*
- OCaml Resources — CS3100 @ IITM, CS3110 book from Cornell U

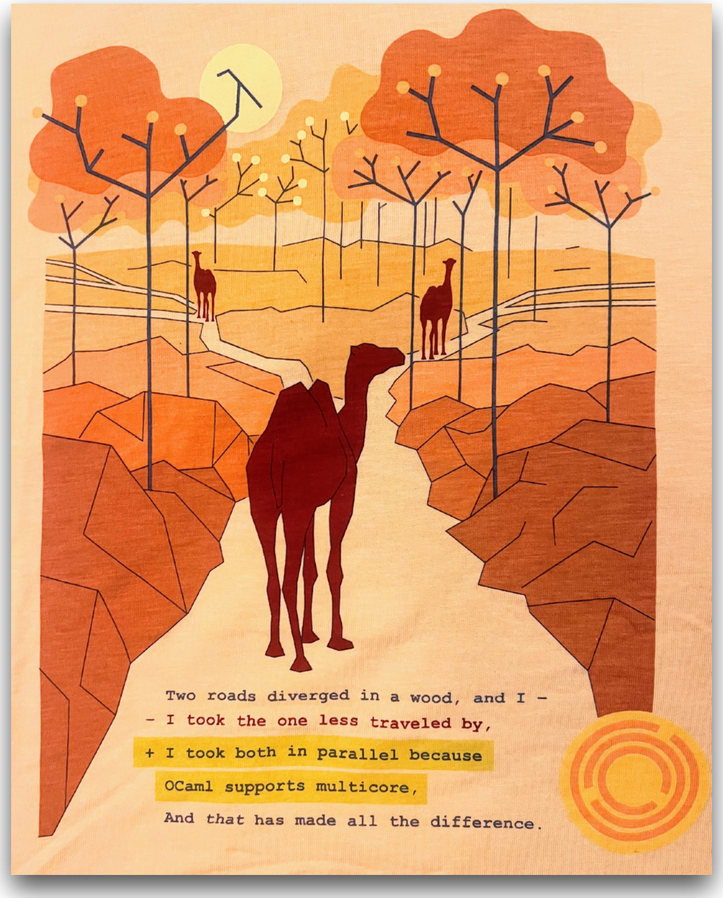


https://github.com/kayceesrk/cs3100_m20



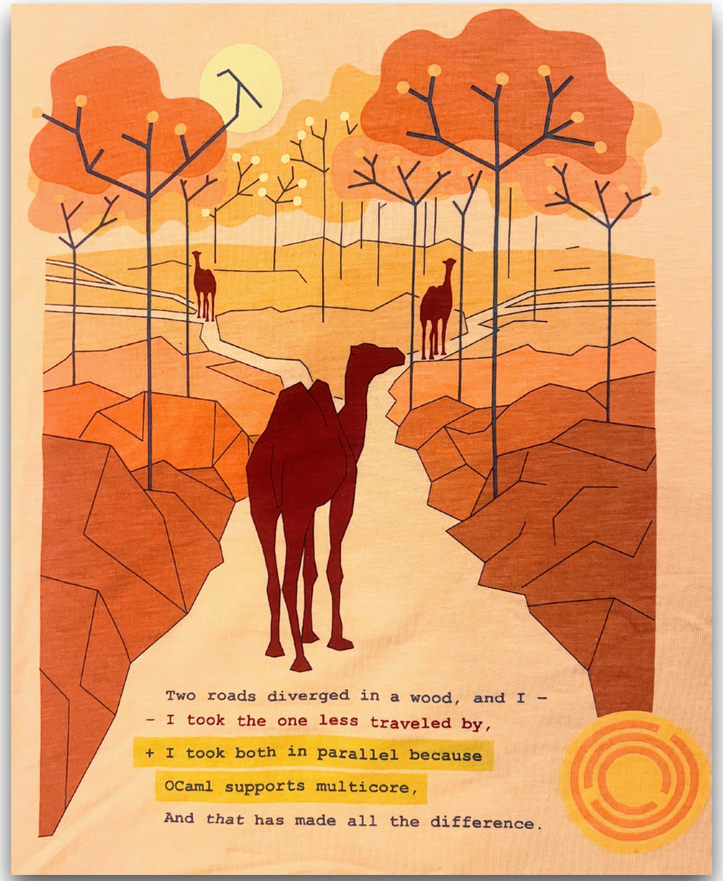
YouTube PlayList

Course Language



OCaml 5 — concurrency & parallelism

Course Language



OCaml 5 — concurrency & parallelism

OxCaml — safe, performance-oriented extension of OCaml

Evaluation Plan

- 6 in-class short quizzes **(20%)**
  - 15 minutes each, announced
  - 2 each — before quiz 1, before quiz 2, before end sem
  - Best 5/6

Evaluation Plan

- 6 in-class short quizzes **(20%)**
  - 15 minutes each, announced
  - 2 each — before quiz 1, before quiz 2, before end sem
  - Best 5/6
- Mid-term **(20%)**

Evaluation Plan

- 6 in-class short quizzes **(20%)**
  - 15 minutes each, announced
  - 2 each — before quiz 1, before quiz 2, before end sem
  - Best 5/6
- Mid-term **(20%)**
- End Sem **(20%)**

Evaluation Plan

- 6 in-class short quizzes **(20%)**
  - 15 minutes each, announced
  - 2 each — before quiz 1, before quiz 2, before end sem
  - Best 5/6
- Mid-term **(20%)**
- End Sem **(20%)**
- Programming assignments x 4 **(24%)**
  - Individual

Evaluation Plan

- 6 in-class short quizzes **(20%)**
  - 15 minutes each, announced
  - 2 each — before quiz 1, before quiz 2, before end sem
  - Best 5/6
- Mid-term **(20%)**
- End Sem **(20%)**
- Programming assignments x 4 **(24%)**
  - Individual

Don’t use LLM; they hinder learning

Evaluation Plan

- 6 in-class short quizzes **(20%)**
  - 15 minutes each, announced
  - 2 each — before quiz 1, before quiz 2, before end sem
  - Best 5/6
- Mid-term **(20%)**
- End Sem **(20%)**
- Programming assignments x 4 **(24%)**
  - Individual

Sign Honour Code;
Automated plagiarism check;
Institute policy on malpractice

Don't use LLM; they hinder learning

Evaluation Plan

- 6 in-class short quizzes **(20%)**
  - 15 minutes each, announced
  - 2 each — before quiz 1, before quiz 2, before end sem
  - Best 5/6
- Mid-term **(20%)**
- End Sem **(20%)**
- Programming assignments x 4 **(24%)**
  - Individual
- Research mini project **(16%)**
  - Groups of 2

Sign Honour Code;
Automated plagiarism check;
Institute policy on malpractice

Don't use LLM; they hinder learning

Evaluation Plan

- 6 in-class short quizzes **(20%)**
  - 15 minutes each, announced
  - 2 each — before quiz 1, before quiz 2, before end sem
  - Best 5/6
- Mid-term **(20%)**
- End Sem **(20%)**
- Programming assignments x 4 **(24%)**
  - Individual
- Research mini project **(16%)**
  - Groups of 2

Sign Honour Code;
Automated plagiarism check;
Institute policy on malpractice

Don't use LLM; they hinder learning

Use LLM; they make you productive*

Parallelism

Modern Multicore Processors