Why FP?

- FP has lots of methodological advantages:
  - Fewer errors
  - Better modularity
  - Higher-level abstractions
  - Shorter code
  - Increased developer productivity
- But these alone are not enough for mainstream adoption (after all FP has been around for 50 years)
- Need a catalyzer, something that sparks initial adoption until the other advantages become clear to everyone.

FP on the Rise

In Computer Science:

Compare #attendees of OOP conferences: OOPSLA, ECOOP
with FP conference: ICFP

2000 : ECOOP ~ 3 x ICFP

OOPSLA ~ 10 x ICFP

2018 : ICFP ~ 3 x ECOOP

OOPSLA stopped existing as independent conference

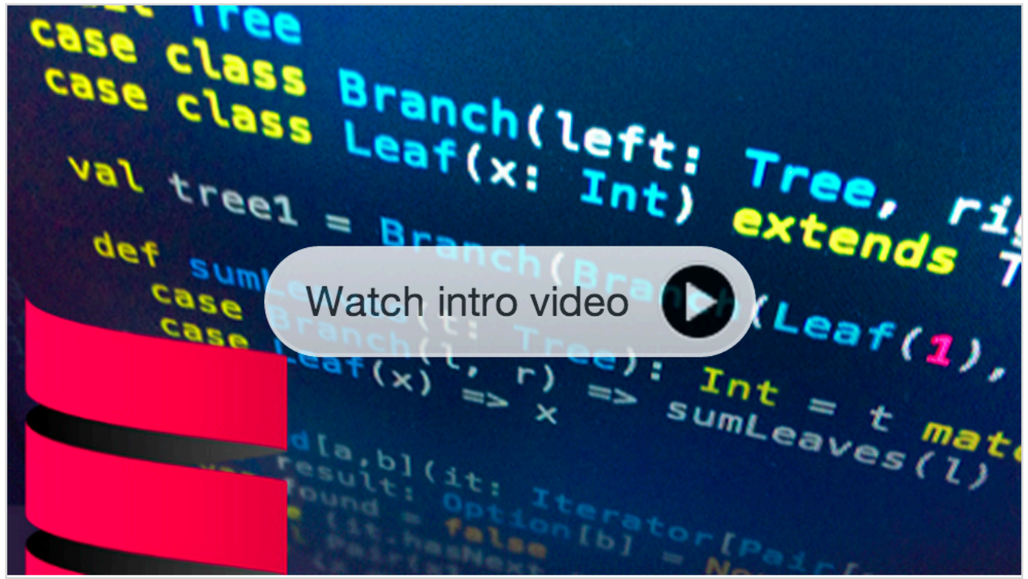
Developer interest: a million sign-ups for the courses of the Functional Programming in Scala specialization

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Functional Programming Principles in Scala

Martin Odersky

Learn about functional programming, and how it can be effectively combined with object-oriented programming. Gain practice in writing clean functional code, using the Scala programming language.



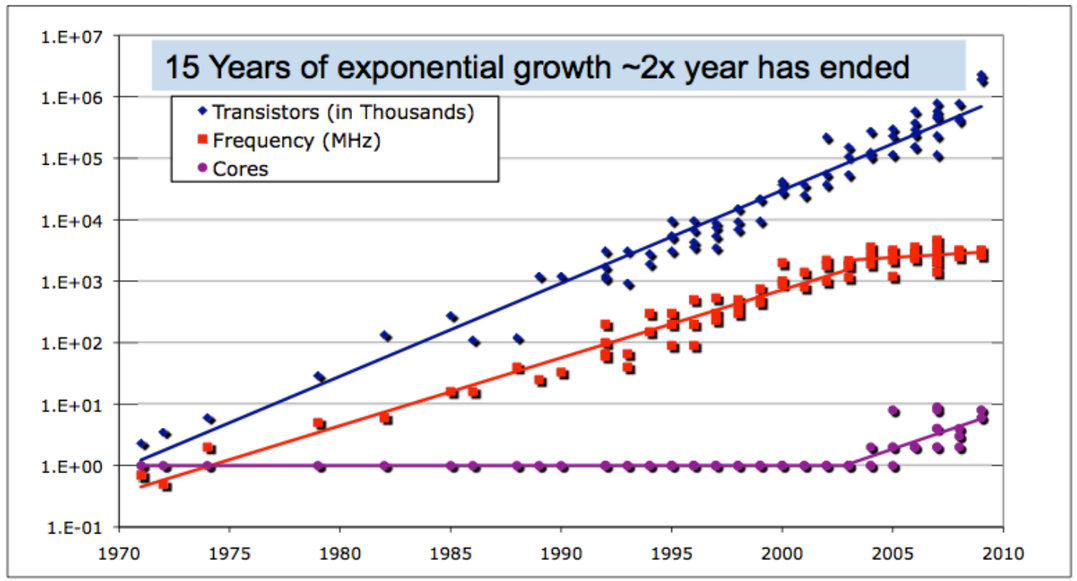
Session(s):

Mar 25th 2013 (7 weeks long) Go to class

Sep 18th 2012 (7 weeks long) View class archive

A Catalyzer

- Two forces driving software complexity:
  - Multicore (= parallel programming)
  - Cloud computing (= distributed programming)
- Current languages and frameworks have trouble keeping up (locks/threads don't scale)
- Need better tools with the right level of abstraction



Concurrency and Parallelism

Parallel programming	Execute programs faster on parallel hardware.
Concurrent programming	Manage concurrent execution threads explicitly.

Both are too hard!

The Root of The Problem

Non-determinism caused by *concurrent threads* accessing *shared mutable* state.

It helps to encapsulate state in actors or transactions, but the fundamental problem stays the same.

So,

non-determinism = *parallel processing* + *mutable state*

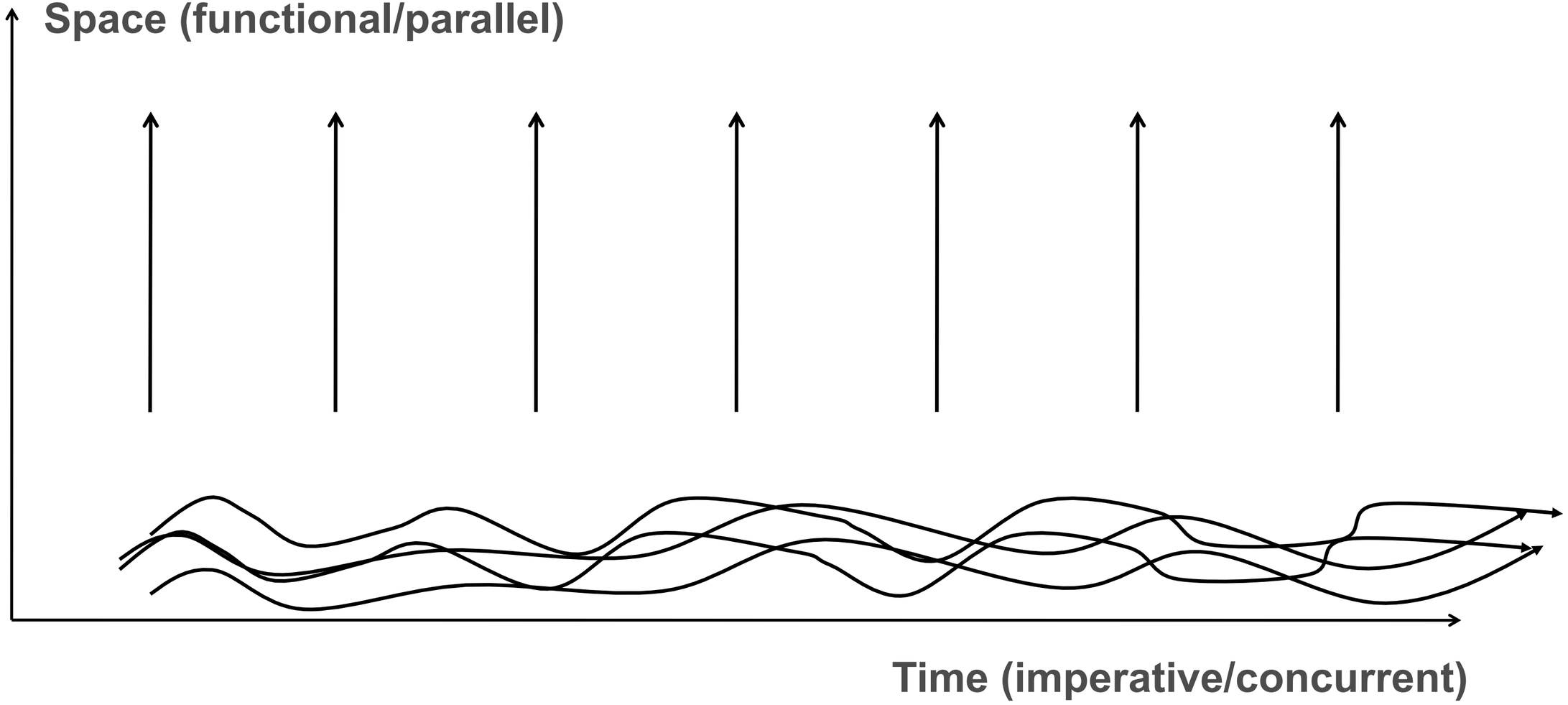
```
var x = 0
async { x = x + 1 }
async { x = x * 2 }

// can give 0, 1, 2
```

To get deterministic processing, avoid the mutable state!

Avoiding mutable state means programming *functionally*.

Space vs Time



The Essence of Functional Programming

Concentrate on *transformations* of *immutable* values

instead of *stepwise modifications* of *mutable* state.