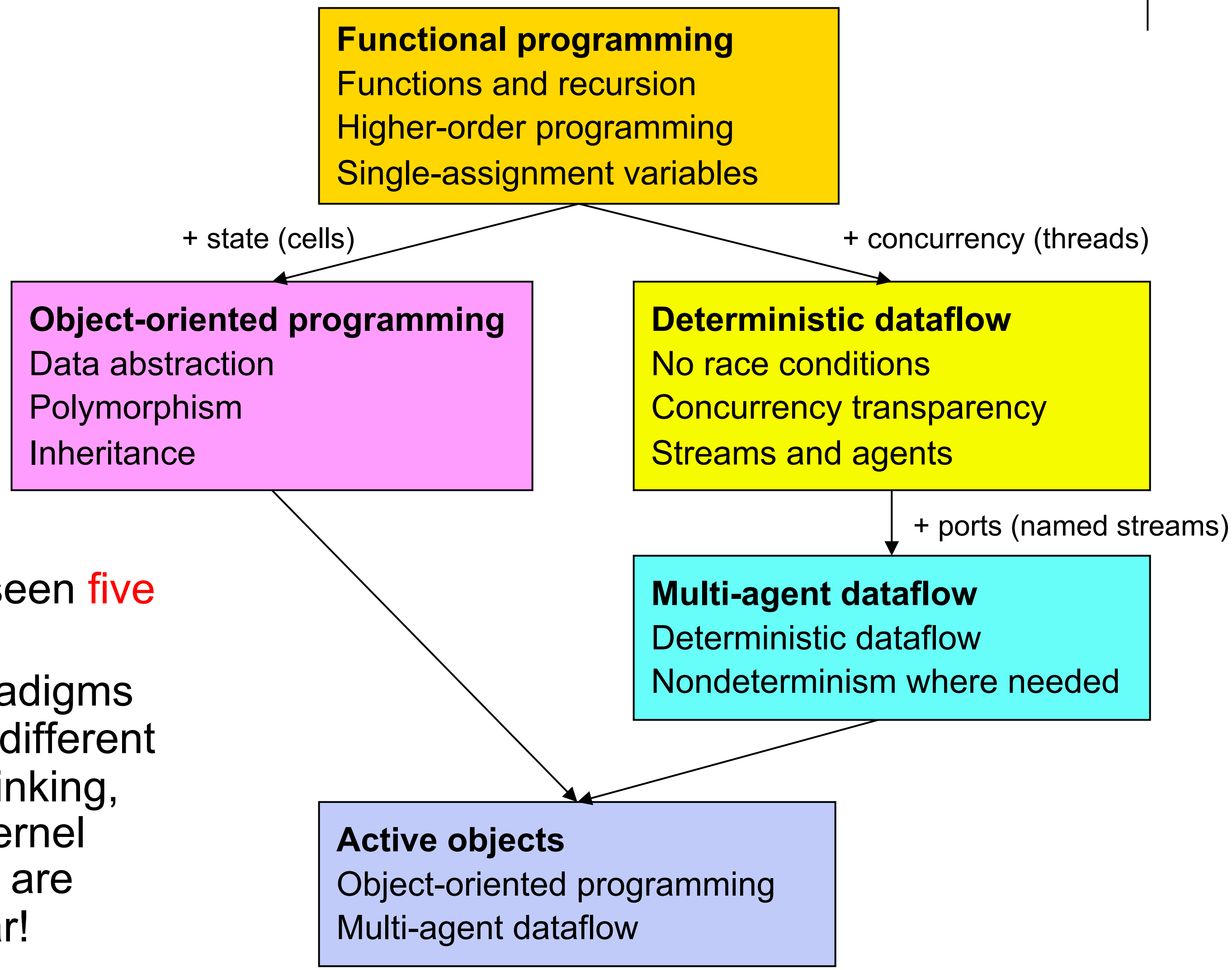
Paradigms of this course (1)

- We defined a paradigm as *an approach to programming a computer based on a coherent set of principles or a mathematical theory*
  - Different theories of computing result in different paradigms (λ calculus, π calculus, first-order logic, Hoare logic, ...)
  - Each theory highlights a different way of programming!
  - Programming is truly a new discipline that is not covered by traditional mathematical theories
- In this course we have covered five important paradigms
  - Why do we need more than one paradigm?
  - Because solving a problem is much easier when done in the right paradigm!

Paradigms of this course (2)



- We have seen five paradigms
- These paradigms have very different ways of thinking, but their kernel languages are very similar!

Kernel language of the functional paradigm

- <s> ::= **skip**
  | <s>$_1$ <s>$_2$
  | **local** <x> **in** <s> **end**
  | <x>$_1$=<x>$_2$
  | <x>=<v>
  | **if** <x> **then** <s>$_1$ **else** <s>$_2$ **end**
  | {<x> <y>$_1$ … <y>$_n$}
  | **case** <x> **of** <p> **then** <s>$_1$ **else** <s>$_2$ **end**
- <v> ::= <number> | <procedure> | <record>
- <number> ::= <int> | <float>
- <procedure> ::= **proc** {$ <x>$_1$ … <x>$_n$} <s> **end**
- <record>, <p> ::= <lit>(<f>$_1$:<x>$_1$ … <f>$_n$:<x>$_n$)

Kernel language of the object-oriented paradigm

- <s> ::= **skip**
  | <s>$_1$ <s>$_2$
  | **local** <x> **in** <s> **end**
  | <x>$_1$=<x>$_2$
  | <x>=<v>
  | **if** <x> **then** <s>$_1$ **else** <s>$_2$ **end**
  | {<x> <y>$_1$ … <y>$_n$}
  | **case** <x> **of** <p> **then** <s>$_1$ **else** <s>$_2$ **end**
  | {NewCell <x> <y>}
  | <y>:=<x>
  | <x>=@<y>
  | **try** <s>$_1$ **catch** <x> **then** <s>$_2$ **end**
  | **raise** <x> **end**

Functional paradigm

Extension with cells and exceptions

- <v> ::= <number> | <procedure> | <record>
- <number> ::= <int> | <float>
- <procedure> ::= **proc** {$ <x>$_1$ … <x>$_n$} <s> **end**
- <record>, <p> ::= <lit>(<f>$_1$:<x>$_1$ … <f>$_n$:<x>$_n$)

Kernel language of deterministic dataflow

- <s> ::= **skip**
  | <s>$_1$ <s>$_2$
  | **local** <x> **in** <s> **end**
  | <x>$_1$=<x>$_2$
  | <x>=<v>
  | **if** <x> **then** <s>$_1$ **else** <s>$_2$ **end**
  | {<x> <y>$_1$ … <y>$_n$}
  | **case** <x> **of** <p> **then** <s>$_1$ **else** <s>$_2$ **end**
  | **thread** <s> **end**

Functional paradigm

Extension with threads

- <v> ::= <number> | <procedure> | <record>
- <number> ::= <int> | <float>
- <procedure> ::= **proc** {$ <x>$_1$ … <x>$_n$} <s> **end**
- <record>, <p> ::= <lit>(<f>$_1$:<x>$_1$ … <f>$_n$:<x>$_n$)

Kernel language of multi-agent dataflow

- <s> ::= **skip**
  | <s>$_1$ <s>$_2$
  | **local** <x> **in** <s> **end**
  | <x>$_1$=<x>$_2$
  | <x>=<v>
  | **if** <x> **then** <s>$_1$ **else** <s>$_2$ **end**
  | {<x> <y>$_1$ … <y>$_n$}
  | **case** <x> **of** <p> **then** <s>$_1$ **else** <s>$_2$ **end**
  (Functional paradigms)
  | **thread** <s> **end**
  (Extension with threads)
  | {NewPort <x> <y>}
  | {Send <x> <y>}
  (Extension with ports)
- <v> ::= <number> | <procedure> | <record>
- <number> ::= <int> | <float>
- <procedure> ::= **proc** {$ <x>$_1$ … <x>$_n$} <s> **end**
- <record>, <p> ::= <lit>(<f>$_1$:<x>$_1$ … <f>$_n$:<x>$_n$)

Kernel language of active objects

- Active objects combine the abilities of object-oriented programming with multi-agent dataflow
  - The behavior of each active object is determined by a class, and active objects communicate by message passing
  - The kernel language is the union of the two kernel languages of these paradigms
  - I leave it as an exercise for you to write it down!