Programming Paradigms

Lecture 13

Recap of Main concepts

- Programming paradigm (in this course)
  - A pattern that serves as a *school of thoughts* for programming of computers
- Programming technique
  - Related to an algorithmic idea for solving a particular class of problems
  - Examples: 'Divide and conquer' and 'program development by stepwise refinement'
- Programming style
  - The way we express ourselves in a computer program
  - Related to elegance or lack of elegance
- Programming culture
  - The totality of programming behavior, which often is tightly related to a family of programming languages
  - The sum of a main paradigm, programming styles, and certain programming techniques.

- Main programming paradigms
  - The imperative paradigm
  - The functional paradigm
  - The logical paradigm
  - The object-oriented paradigm
- Other possible programming paradigms
  - The visual paradigm
  - One of the parallel paradigms
  - The constraint based paradigm

Overview of the imperative paradigm

First **do this** and next **do that**

- Characteristics:
  - Discipline and idea
    - Digital hardware technology and the ideas of Von Neumann
  - Incremental *change of the program state* as a function of *time*.
  - Execution of computational steps in an order governed by *control structures*
    - We call the steps for *commands*
  - Straightforward abstractions of the way a traditional Von Neumann computer works
  - Similar to descriptions of everyday routines, such as food recipes and car repair
  - Typical commands offered by imperative languages
    - Assignment, IO, procedure calls
  - Language representatives
    - Fortran, Algol, Pascal, Basic, C
  - The natural abstraction is the procedure
    - Abstracts one or more actions to a procedure, which can be called as a single command.
    - "Procedural programming"

Overview of the functional paradigm

Evaluate an expression and use the resulting value for something

- Characteristics:
  - Discipline and idea
    - Mathematics and the theory of functions
  - The values produced are *non-mutable*
    - Impossible to change any constituent of a composite value
    - As a remedy, it is possible to make a revised copy of composite value
  - Atemporal
    - Abstracts a single expression to a function which can be evaluated as an expression
  - Functions are first class values
    - Functions are full-fledged data just like numbers, lists, ...
  - Fits well with computations driven by needs
    - Opens a new world of possibilities

Overview of the logic paradigm

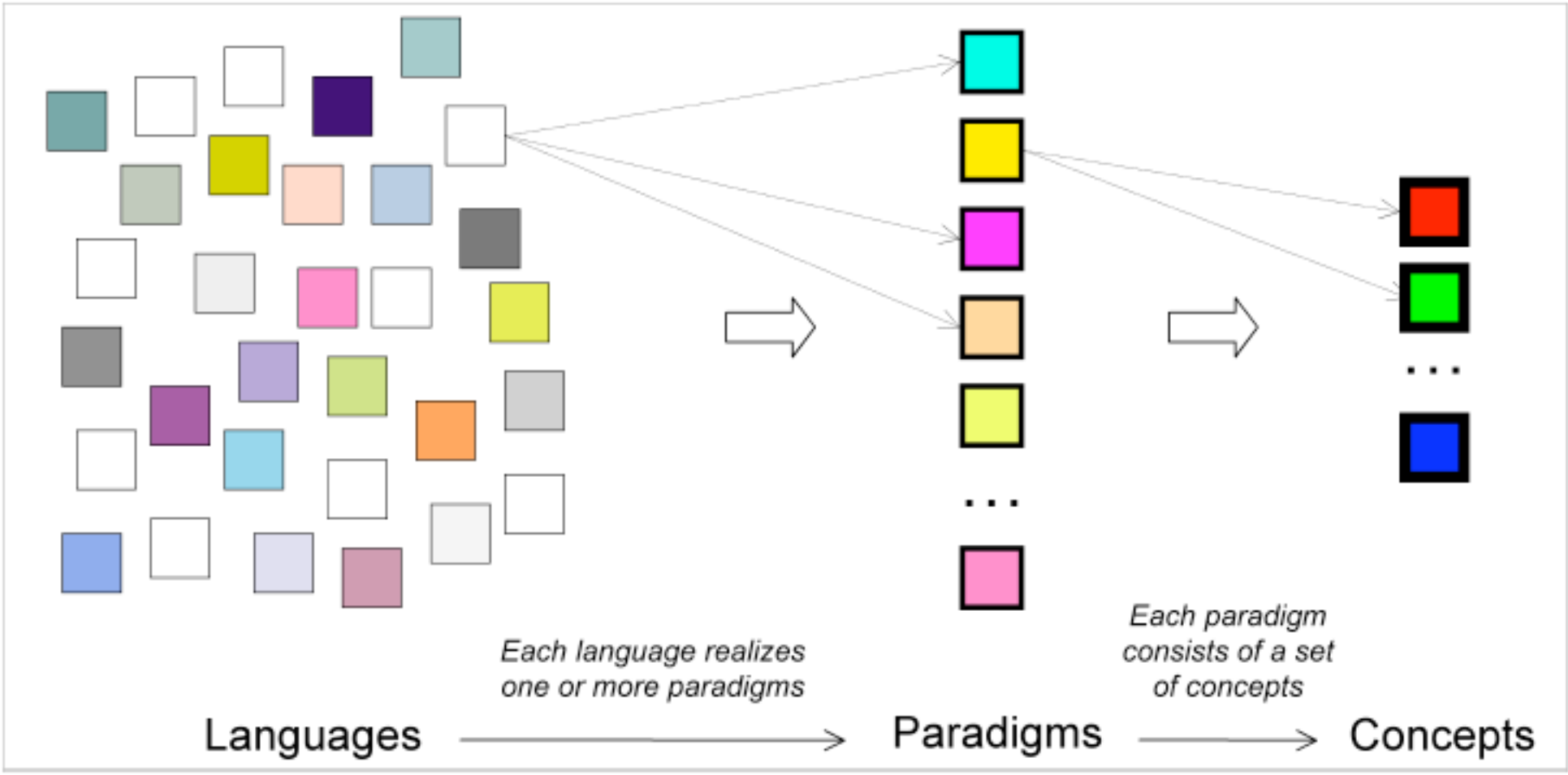
Answer a question via search for a solution

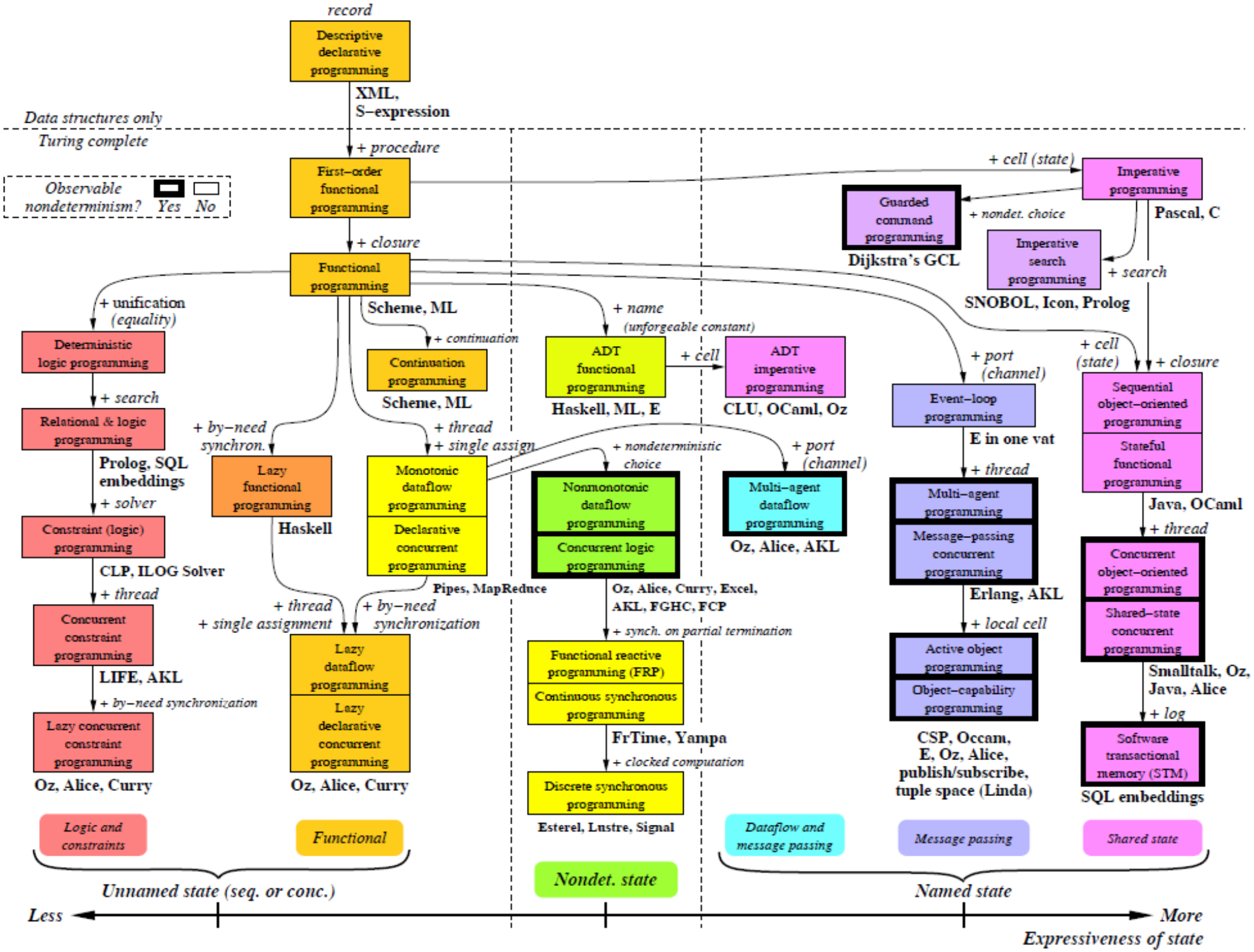
- Characteristics:
  - Discipline and idea
    - Automatic proofs within artificial intelligence
  - Based on axioms, inference rules, and queries.
  - Program execution becomes a systematic search in a set of facts, making use of a set of inference rules

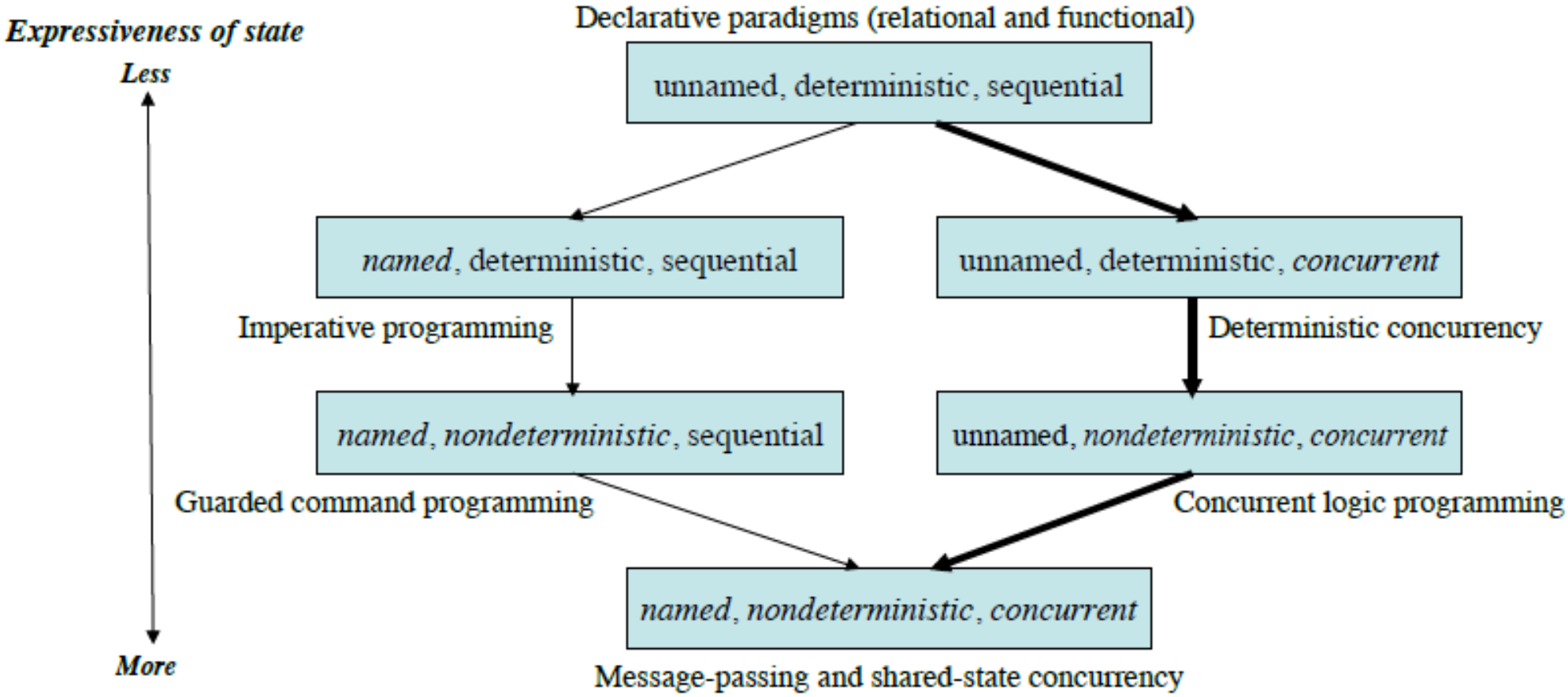
Overview of the object-oriented paradigm

Send messages between objects to simulate the temporal evolution of a set of real world phenomena

- Characteristics:
  - Discipline and idea
    - The theory of concepts, and models of human interaction with real world phenomena
  - Data as well as operations are encapsulated in objects
  - Information hiding is used to protect internal properties of an object
  - Objects interact by means of message passing
    - A metaphor for applying an operation on an object
  - In most object-oriented languages objects are grouped in classes
    - Objects in classes are similar enough to allow programming of the classes, as opposed to programming of the individual objects
    - Classes represent concepts whereas objects represent phenomena
  - Classes are organized in inheritance hierarchies
    - Provides for class extension or specialization


Each language realizes one or more paradigms
Each paradigm consists of a set of concepts
Languages
Paradigms
Concepts


record
Descriptive declarative programming
XML, S–expression
Data structures only
Turing complete
+ procedure
First–order functional programming
Observable nondeterminism? Yes No
+ closure
Functional programming
+ cell (state)
Imperative programming
Pascal, C
Guarded command programming
+ nondet. choice
Dijkstra's GCL
Imperative search programming
+ search
SNOBOL, Icon, Prolog
+ unification (equality)
Deterministic logic programming
+ search
Relational & logic programming
Prolog, SQL embeddings
+ solver
Constraint (logic) programming
CLP, ILOG Solver
+ thread
Concurrent constraint programming
LIFE, AKL
+ by–need synchronization
Lazy concurrent constraint programming
Oz, Alice, Curry
Scheme, ML
+ continuation
Continuation programming
Scheme, ML
+ by–need synchron.
Lazy functional programming
Haskell
+ thread
+ single assign.
Monotonic dataflow programming
Declarative concurrent programming
Pipes, MapRednce
+ thread
+ single assignment
+ by–need synchronization
Lazy dataflow programming
Lazy declarative concurrent programming
Oz, Alice, Curry
+ name (unforgeable constant)
ADT functional programming
Haskell, ML, E
+ cell
ADT imperative programming
CLU, OCaml, Oz
+ nondeterministic choice
Nonmonotonic dataflow programming
Concurrent logic programming
Oz, Alice, Curry, Excel, AKL, FGHC, FCP
+ synch. on partial termination
Functional reactive programming (FRP)
Continuous synchronous programming
FrTime, Yampa
+ clocked computation
Discrete synchronous programming
Esterel, Lustre, Signal
+ port (channel)
Multi–agent dataflow programming
Oz, Alice, AKL
+ port (channel)
Event–loop programming
E in one vat
+ thread
Multi–agent programming
Message–passing concurrent programming
Erlang, AKL
+ local cell
Active object programming
Object–capability programming
CSP, Occam, E, Oz, Alice, publish/subscribe, tuple space (Linda)
+ cell (state)
+ closure
Sequential object–oriented programming
Stateful functional programming
Java, OCaml
+ thread
Concurrent object–oriented programming
Shared–state concurrent programming
Smalltalk, Oz, Java, Alice
+ log
Software transactional memory (STM)
SQL embeddings
Logic and constraints
Functional
Dataflow and message passing
Message passing
Shared state
Unnamed state (seq. or conc.)
Nondet. state
Named state
Less
More
Expressiveness of state


Expressiveness of state
Less
More
Declarative paradigms (relational and functional)
unnamed, deterministic, sequential
named, deterministic, sequential
unnamed, deterministic, concurrent
Imperative programming
Deterministic concurrency
named, nondeterministic, sequential
unnamed, nondeterministic, concurrent
Guarded command programming
Concurrent logic programming
named, nondeterministic, concurrent
Message-passing and shared-state concurrency