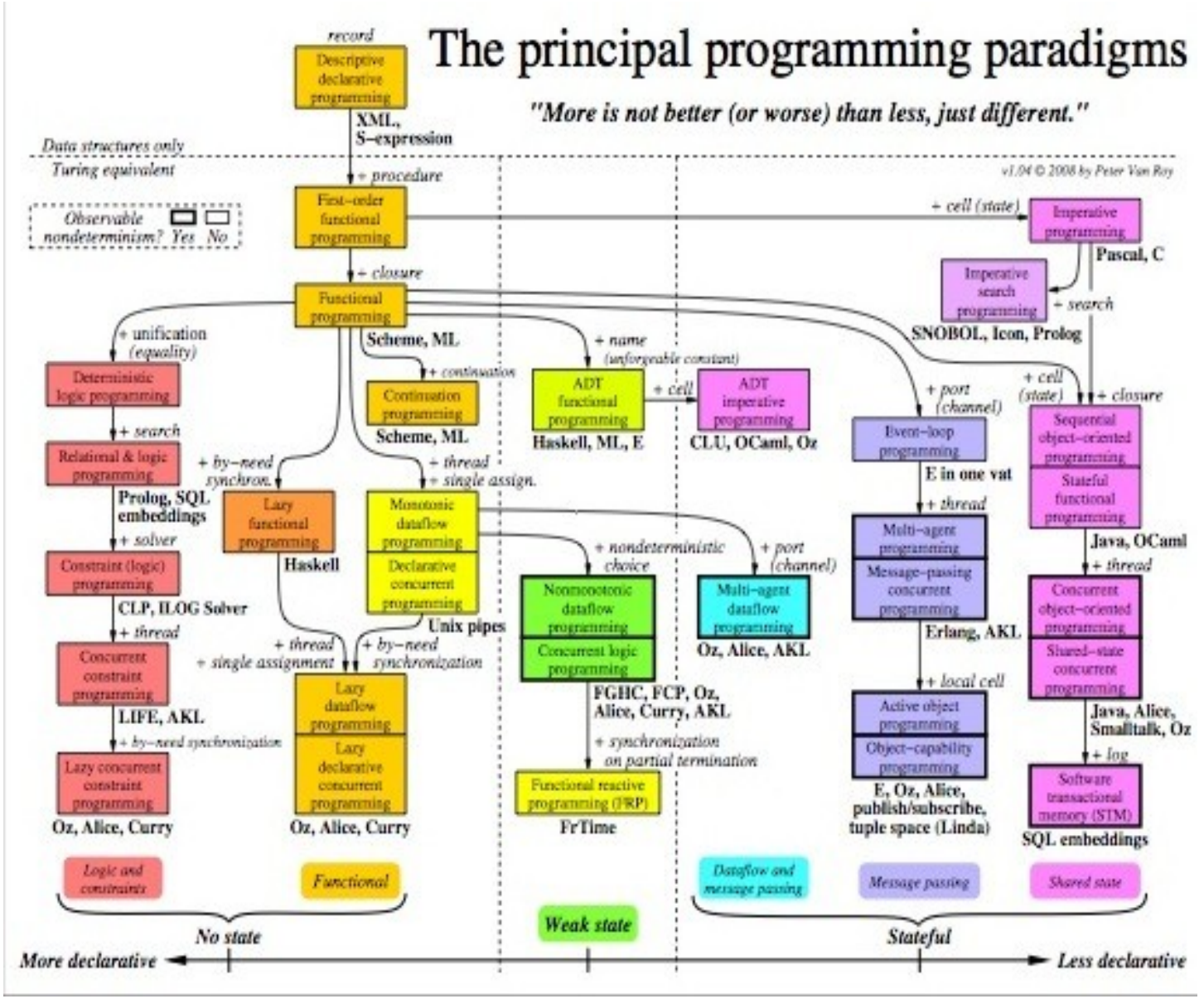
Programming Paradigms

- A programming paradigm is a fundamental style of computer programming.
- Paradigms differ in the concepts and abstractions used to represent the elements of a program (such as objects, functions, variables, constraints, etc.) and the steps that compose a computation (assignment, evaluation, continuations, data flows, etc.).

Programming Paradigms

- Different programming languages advocate different programming paradigms.
- A multi-paradigm programming language is a programming language that supports more than one programming paradigm.
- The design goal of such languages is to allow programmers to use the best tool for a job, admitting that no one paradigm solves all problems in the easiest or most efficient way.


The principal programming paradigms
"More is not better (or worse) than less, just different."
v1.04 © 2008 by Peter Van Roy
record
Descriptive declarative programming
XML, S-expression
Data structures only
Turing equivalent
Observable nondeterminism? Yes No
+ procedure
First-order functional programming
+ cell (state)
Imperative programming
Pascal, C
Imperative search programming
SNOBOL, Icon, Prolog
+ search
+ closure
Functional programming
Scheme, ML
+ 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
+ continuation
Continuation programming
Scheme, ML
+ by-need synchron.
Lazy functional programming
Haskell
+ thread
+ single assign.
Monotonic dataflow programming
Declarative concurrent programming
Unix pipes
+ 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
FGHC, FCP, Oz, Alice, Curry, AKL
+ synchronization on partial termination
Functional reactive programming (FRP)
FrTime
+ 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
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
Java, Alice, Smalltalk, Oz
+ log
Software transactional memory (STM)
SQL embeddings
Logic and constraints
Functional
Dataflow and message passing
Message passing
Shared state
No state
Weak state
Stateful
More declarative
Less declarative

Programming Paradigms

- There are many different programming paradigms – they can be broadly classified as:
  - The imperative paradigm
  - The functional paradigm
  - The logical paradigm
  - The object-oriented paradigm
- A programming paradigm can be defined in terms of an idea and a basic discipline.

http://people.cs.aau.dk/~normark/prog3-03/html/notes/paradigms_themes-paradigms.html

Imperative paradigm

- 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 commands
- Straightforward abstractions of the way a traditional Von Neumann computer works

Imperative paradigm

- 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, COBOL, 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"

Functional paradigm

- 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
  - Time only plays a minor role compared to the imperative paradigm

Functional paradigm

- Applicative
  - All computations are done by applying (calling) functions
- The natural abstraction is the function
  - 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

Logic paradigm

- 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
- Example : Prolog

Object-oriented paradigm

- 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

Object-oriented paradigm

- 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