Functional Programming with Lisp and Clojure

Yacin Nadji
ynadji@iit.edu
IIT-ACM
February 24th, 2009

Outline

- Lisp and Functional Programming
- Clojure - What makes it unique
  - Concurrency
  - Multimethods (Goodbye OOP!)
  - Missing Pieces
- Conclusion

What Is Lisp?

- Laundry list of features: first class functions, closures, OOP, multiple dispatch, (real) macros, anonymous functions
- Swallowed by newer languages
  - Ruby, C++0x, Python
- Prefix notation, syntax similar to AST

Lisp “Syntax”

- (function arg1 arg2 … argn)
- (+ 1 2 3 4)
- (+ 1 (* 3 4))

Clojure

- Lisp-1 compiles to JVM bytecode
- Basic data structures are immutable!
- Trivial Java interop
  - JOptionPane example

Clojure/Java

Java	Clojure	Syntactic Sugar
`new Widget("red")`	`(new Widget "red")`	`(Widget. "red")`
`Math.PI`	`(. Math PI)`	`Math/PI`
`System.currentTimeMillis()`	`(.currentTimeMillis System)`	`(System/currentTimeMillis)`
`rnd.nextInt()`	`(. rnd nextInt)`	`(.nextInt rnd)`
`object.method1().method2().method3()` `(.. object method1 method2 method3)`		

Datatypes

- Lists: `(1 2 3 4)`
- Vectors: `[1 2 3 4]`
- Maps: `{:key1 “value1”, :key2 “value2”}`
- Sets: `#{1 2 3 4}`
- Generalized insertion operator **`conj`**

Concurrency Rules!

- Software transactional memory (STM)
  - Synchronous shared "state"
- Agents
  - Asynchronous tasks
- Vars
  - Thread-local bindings

Software Transactional Memory (STM)

- Transactions -- **ACI** not **ACID**
  - Atomic -- duh
  - Consistent -- validation functions
  - Isolated -- (follows from atomic)
- `dosync` `alter` and `commute`

Agents

- Asynchronous commands
- Example
- `await`

Multimethods

- Achieves similar goal to polymorphism
- Example: an opinion generator based on known "good" and "bad" things

Missing Pieces

- No tail-call optimization in JVM!
  - `(loop [i 1] ... (recur new-i))`
- No built-in distributed model (a la Erlang)

Conclusion

- Lisp rules, Clojure rules more
  - Java improves = clojure improves
  - Crazy good interop w/ Java
  - Prettier than Lisp too!
- Use it!