Heinrich-Heine-University Düsseldorf
Computer Science Department
Software Engineering and Programming Languages
Philipp Körner

Functional Programming – WT 2023 / 2024
Reading Guide 10: Time and Value (Praxis) — Atoms and Agents

Timeline: This unit should be completed by 18.12.2023.

Note: We will interleave a smaller unit next week so that there only a small workload until next year. The logically next unit will be on Refs, which are (by definition!) more complex.

1 Material

- Alternatives:
  -
    - 24_atoms.clj
    - 25_agents.clj
  - The Joy of Clojure, chapter 10.4 + 10.3
  - Clojure for the Brave and True, chapter 10 (covers most of refs as well): Object-Oriented Metaphysics, Clojure Metaphysics, Atoms, Watches and Validators
- Reference Pages:
  - Clojure Reference: Atoms `https://clojure.org/reference/atoms`
  - Clojure Reference: Agents `https://clojure.org/reference/agents`
- Learning Videos:
  - Atoms: `https://mediathek.hhu.de/watch/b97ca683-8682-41e1-a491-8e0168b09837`
  - Agents: `https://mediathek.hhu.de/watch/45050c42-1c2d-41ed-a7f1-66d3bd181c08`

2 Learning Outcomes

After completing this unit you should be able to

- describe and compare the semantics of atoms and agents constructs.
- explain the conflict behaviour of atoms and agents constructs.
- identify incorrect usage of atoms and agents constructs.
- identify use cases for atoms and agents.
- use atoms and agents in a consistent manner.

3 Highlights

- Atoms, agents
- Macros: `dosync`
- Functions: `swap!`, `reset!`, `deref`, `send`, `send-off`, `await`

4 Exercises

Exercise 10.1 (Minesweeper)
Implement Minesweeper (see `https://de.wikipedia.org/wiki/Minesweeper`, `https://www.youtube.com/watch?v=LHY8NKj3RKs`).

At least the following functions shall be available on the REPL:

- `(init! height width amount-of-mines)` should initialize the playing field with corresponding height, width and number of mines. Additionally `(init!)` should use 30 as default value for the height, 16 for the width and 99 for the number of mines. *Note:* the function `shuffle` could be useful.
- `(reveal! x y)` uncovers the field at coordinate (x,y). If a mine happens to be there, the game is lost. If there is no mine in the surrounding area of this field, all fields around it should be uncovered automatically. (and analogously for the neighbours of those fields, if there are also no mines in the surrounding area).
- `(flag! x y)` should mark the field at (x,y) as mine or remove this mark if it already was marked. Marked fields cannot be uncovered by `reveal!`.
- `(print-board!)` should output the board in an appropriate format.

Note: This requires the use of time management construct. Make sure that no inconsistent state can occur.

Note: Yes, it is somewhat ugly to extract the fields surrounding a given field.

Questions

If you have any questions, please contact Philipp Körner (`p.koerner@hhu.de`) or post it to the Rocket.Chat group.