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

Functional Programming – ST 2024
Reading Guide 04: core.async[1]

Timeline: This unit should be completed by 13.05.2024.

1 Material

- Rick Hickey: Clojure core.async `https://www.youtube.com/watch?v=yJxFPoxqzWE` (Talk containing the motivation, backgrounds, application areas and implementation ideas.)
- Timothy Baldridge: core.async `https://www.youtube.com/watch?v=enwIIGzhahw` (or work through the REPL session presented in the video): `https://github.com/halgari/clojure-conj-2013-core.async-examples/blob/master/src/clojure_conj_talk/core.clj`
- Clojure for the Brave and True, Chapter 11: Mastering Concurrent Processes with core.async (This is an alternative to the material above. It provides another perspective on core.async. Further, a hot dog vending machine is created, which definitely is a plus.)
- Rich Hickey: Implementation details of core.async Channels `https://github.com/matthiasn/talk-transcripts/blob/master/Hickey_Rich/ImplementationDetails.md` (transcript), `https://www.youtube.com/watch?v=hMEX6lfBeRM` (video) (Though we do not care about most implementation details too much, the video provides information about channels and interaction with them. The first 16 minutes are the most relevant.)
- Blogpost announcing core.async: `https://clojure.org/news/2013/06/28/clojure-clore-async-channels` (A summary to wrap up.)

Useful resources:

- GitHub/Source Repo: `https://github.com/clojure/core.async`
- API docs: `https://clojure.github.io/core.async/`

2 Learning Outcomes

After completing this unit you should be able to

- explain and implement communication using core.async channels.
- name and explain different buffer strategies and the rationale for buffers.

[1] This reading guide was originally created by Florian Mager and was kindly made available under a CC BY-SA 4.0 licence.

- describe the anatomy of a channel.
- explain what happens when a channel is closed.
- decide when to use the core.async library.

3 API

You should be able to use the following API:

- Creating channels: `(chan)` / `(chan buf-or-n)`
- Creating buffers: `(buffer n)` / `(dropping-buffer n)` / `(sliding-buffer n)`
- Creating processes: `(thread & body)` / `(go & body)`
- Put: `(>! port val)` / `(>!! port val)`
- Take: `(<! port val)` / `(<!! port val)`
- Alt: `(alt! & clauses)` / `(alt!! & clauses)`
- Alts:`(alts! & ports & {:as opts})` / `(alts!! & ports & {:as opts})`
- Timeout channel: `(timeout msecs)`
- Closing channels: `(close! chan)`