Lecture 11: Message Passing 3

Adam Hawley

February 28, 2019

Contents

1 Erlang

- It is an **eager** functional language.
- Asynchronous message passing communication model
  - **Non-Blocking** send
  - Send never fails (even to non-existent ids)
  - Inter-process **buffers** are **unbounded** - conceptually
  - Sending order is mainitained at receiver
  - Receives from different senders, are read non-deterministically
- **Unlimited** message types
- **Direct asymmetric** naming via **process id** (pid)

1.1 Send !

Values of any type can be sent. It has Occam-like syntax (e.g `Pid4 ! {pos, 42}`).

1.2 `receive`

Each process has an unbounded queue for received messages. The arrival order is maintained for each sender but the buffer is interleaved between senders.

Removing messages from the mailbox is using pattern matching from the oldest entry and it is blocking if no match is available.

Receive so has ways of implementing a select from Ada or ALT from Occam and a guarded select:

```
% Ada Select or Occam ALT
receive
    pattern1 -> expression1;
    pattern2 -> expression2;
    pattern3 -> expression3;
end

% Guarded Select
receive
    pattern1 when guard1 -> expression1;
    pattern2 when guard2 -> expression2;
    pattern3 when guard3 -> expression3;
```