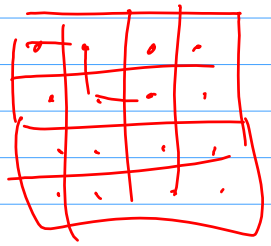
March 29, 2010 - part 2 - Concurrent ML.

- Synchronous Message Passing -
  Sender waits for a receiver to be ready to receive.
  - in Java: SynchronousQueue.

History:
- Anthony Hoare: Communicating Sequential Processes (CSP)
  - Ada
  - Occam



Robin Milner - Calculus of Communicating Systems
Pi - calculus

Concurrent ML – Standard ML + Sync. comm.

- channels – typed – carry values of only 1 type.

– first class values –

(α) channel

operations on channels

argument

result

void

send: (α channel * α) → unit

recv: (α channel) → α

channel type α Channel * α
value value (c, v)

Executing
Send (c, v)

waits until some other process executes
recv(c)
Then both functions return and the value
of the expression recv (c) is v.

Problems 1)
- process has to know what order messages arrive on different channels.
- deadlock possibilities: get it wrong and die

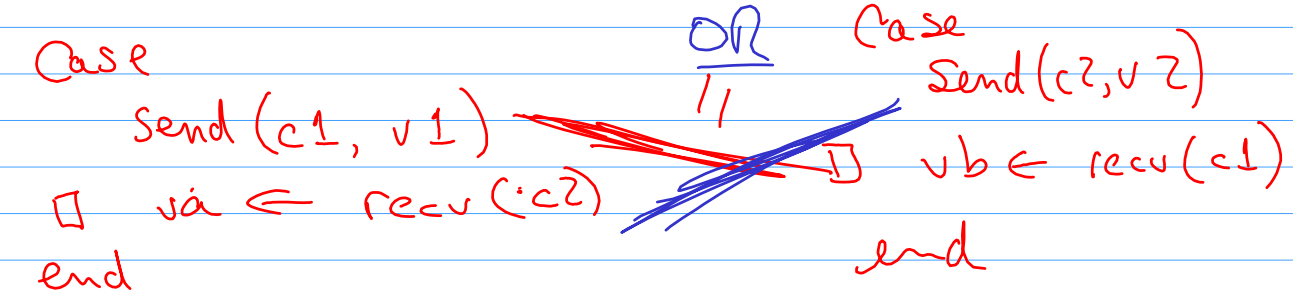
send (c1, v1)
v2 = recv(c2)

//

send(c2, va)
vb = recv(c1)

Solution is some kind of choice operation that lets a process be ready to communicate on multiple channels at one time

- Rendezvous — Ada, CSP, ..
- select



runtime system has to choose exactly one of the communications to occur

March 31, Concurrent ML continued

Concept: send event } value representing
receive event } willingness to communicate.

send recv. attempts to send / recv

SendEvt recvEvt : values repres

Send (c, v)

SE = sendEvt (c, v) — value

Sync (SE) ≡ Send (c, v)

RE = recvEvt (c)

select [se1, se2, ... re1, re2, ...]

send (c, v) ≡ select [sendEvt (c, v)]

se1 = sendEvt (c1, v1)

re1 = recvEvt (c2)

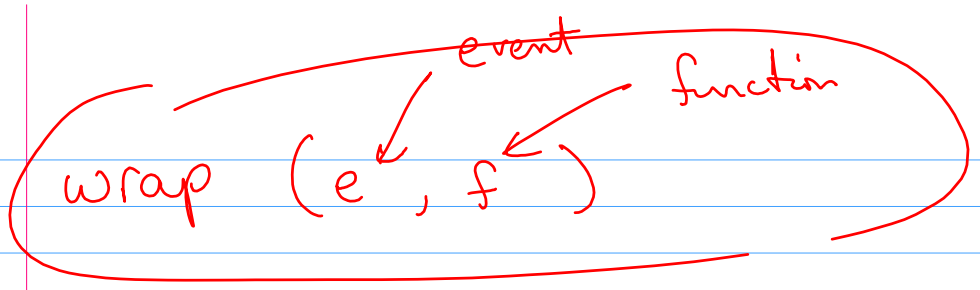
select [se1, re1]

se2 = sendEvt (c2, v2)

re2 = recvEvt (c1)

select [se2, re2]

- timeoutEvt (ms)
- alwaysEvt ()
- neverEvt ()



be willing to perform e and if it is performed return f (the value of performing e) as the result.

```
select [
    wrap (recvEvt (c), fn v => ... v ...),
    ⋮
]
```

Unix socket pgming

res = Select (infds , outfds, errfds)

↑ system call

list

res = error indication : -1 for error
n > 0 for number of ready descriptors

and infds, outfds, errfds are modified

if res > 0 {

loop through infds ;

for each fd find code needed to process that one

loop through outfds

$l \equiv$ list of floors at which the lift will stop.

t = target floor $\in l$

n = current floor

How long does it take to get from n to t.

timeToTarget (n, t, l) :

if l is empty $\Rightarrow$ error

if head$(l) \equiv t \Rightarrow |n - t|$

else $|n - \text{head}(l)| + 5 +$

timeToTarget (head(l), t, tail(l))

Grad Students Projects.

Brian –

Testing concurrent programs;

Relaying through sampling.

Context aware.

Monday of last week ~ 25 minutes presentation

10 minutes question

– Software TM – Coordinate week of April 5 Thursday office hours

---

Eric – discuss next week privately.

Thursday office hours