A Mechanized Theory of Communication Analysis in CML

March 3, 2019

Concurrent ML

- extension of Standard ML
- concurrency and synchronization
- synchronized communication over channels: send event, receive event
- composition of events: choose event, wrap event ...

Concurrent ML

```
type thread_id
val spawn : (unit -> unit) -> thread_id

type 'a chan
val channel : unit -> 'a chan

type 'a event
val sync: 'a event -> 'a

val recvEvt: 'a chan -> 'a event
val sendEvt: 'a channel * 'a -> unit event

val send: 'a chan * 'a -> unit
fun send (ch, v) = sync (sendEvt (ch, v))

val recv: 'a chan -> 'a
fun recv ch = sync (recvEvt ch)
```

Concurrent ML

```
structure Serv : SERV =
struct
  datatype serv = S of (int * int chan)
      chan

  fun make () =
  let
    val reqCh = channel ()
    fun loop state =
    let
      val (v, replCh) = recv reqCh
      val () = send (replCh, state)
    in
      loop v
    end
    val () = spawn (fn () => loop 0)
  in
    S reqCh
  end
```

```
  fun call (server, v) =
  let
    val S reqCh = server
    val replCh = channel ()
    val () = send (reqCh, (v, replCh))
  in
    recv replCh
  end
end

signature SERV =
sig
  type serv
  val make : unit -> serv
  val call : serv * int -> int
end
```