Concurrency: Theory, Languages and Programming

– From CCS to PiLib –

Session 5 – November 20, 2002

Martin Odersky

EPFL-LAMP

Pilib

Pilib is a library, which allows one to use CCS primitives in a Scala program.

CCS constructs are modelled as Scala functions.

Their implementation is based on Java's threads.

Pilib's functions are implemented in two modules:

- *concurrency* for general thread management.
- *pilib* for CCS actions and sums.

An Example

Here is a two-place buffer implementation using Pilib.

```
import concurrency;   // make available Pilib functions
import pilib;         // without qualification.

module bufferExample with
  def Buffer[a](in : Chan[a], out : Chan[a]): Unit =
    def B0 : Unit =    val x = in.read; B1(x)
    def B1(x : a): Unit = choice
        out(x)    (B0)
        in    (y      B2(x, y))

    def B2(x : a, y : a): Unit =    out.write(x); B1(y)
    B0  // initial state
```

Explanations

Chan is the type of CCS names (or: channels).

Chan takes a *type parameter* *a*, which determines the type of values that can be read from and written to the channel.

The *Buffer* process is modelled by a recrusive Scala function, nested functions *B0*, *B1*, *B2*.

Each nested function represents a buffer state (0 = empty, 1 = half full, 2 = full).

A Buffer Client

```
val random = new java.util.Random ();

def Producer (n : Int, l : Chan [String ] ): Unit =
   sleep (1      random.nextInt (1000 ) );
   l.write (”object ”      n );
   System.out.println (”Producer gave ”      n );
   Producer (n      1, l )


def Consumer (r : Chan [String ] ): Unit =
   sleep (1      random.nextInt (1000 ) );
   val a = r.read;
   System.out.println (”Consummer took ”      a );
   Consumer (r )
```

```
def main(args : Array[String]): Unit =
  val in = new Chan[String];
  val out = new Chan[String];
  spawn    Producer(0, in)   Consumer(out)   Buffer(in, out)
```

Covered CCS Syntax

Action prefix		receive along
		send along
Guarded process		
Process	$\sum$	summation
		composition
		restriction
		agent
Agent definition		
Term		

From CCS to Pilib

Guarded process

* (   )

( ) * (   )

Process

choice (+ ... +   )

spawn   ...

{ val   = new Chan[T];   }

Agent definition

def   : Unit =