CptS 580 In Class

Note Title 1/22/2008

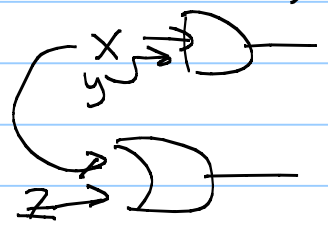
```
fun {GateMaker F}
   fun {$ Xs Ys}
      fun {Gateloop Xs Ys}
         case Xs#Ys of (X|Xr)#(Y|Yr) then
            {F X Y}|{Gateloop Xr Yr} end
         [] nil#nil then ... end
         [] nil#_
      end
      in thread {Gateloop Xs Ys} end
   end
end
```

uses instantiation

uses genericity

```
AndG = {GateMaker fun {$ X Y} X*Y end}
OrG  = {GateMaker                 X+Y-X*Y end}

AndG1 = {AndG Xs Ys}
OrG1  = {OrG  Xs Zs}
```



Joining threads – synchronizing on termination of threads {and extracting a result value}.

```
local X1 ... Xn in
   thread <S>1  X1 = unit  end
   thread <S>2  X2 = X1    end
                :
   thread <S>n  Xn = Xn-1  end
   {Wait Xn}
end
```

()

```
local X1 ... Xn in
   thread <S>1  X1 = unit  end
                :
   thread <S>n  Xn = unit  end
   {Wait X1} ...   {Wait Xn}
end
```

Is waiting necessary in the declarative model?

Coroutines

- a lot like a thread - it's a separate execution context (stack, pc ...) but switching betw. co-routines is controlled by coroutines themselves not by a scheduler.

CId = { Spawn P } — P: 0-arg. procedure; it does not run at this point - suspended.

{ Resume CId } — { P }

↑ caller suspends

order of execution

Python generators

Lazy evaluation (execution)

fun lazy {F A1 ... An} ... end (normal function Body)

- {F A1 ... An} creates a stopped execution (a bit like a coroutine) that gets activated when value of {F A1 ... An} is needed.

to implement: augment Kernel language with By-need triggers.

{ByNeed P Y} like thread {P Y} end except The thread is always executed, and the trigger may not be.

implement my {ByNeed <x> <y>} (<x>: code, <y>: variable that we trigger on.)

if <y> is determined (bound to a value)

Create a thread to evaluate {<x> <y>}

o.w. we add a trigger (x, y) [where x and y are the store variables associated to identifiers <x> <y> (in the current environment)] to the

new component of our OZ virtual machine.

trigger-store.

Trigger is activated when a need for y is detected:
- a thread suspends waiting for y to be bound
- a thread binds y.

(remove trigger from trigger store)
- evaluation of {<x> <y>} takes place in a new thread.

{ }

The amazing thing is that ByNeed preserves declarativeness.

□