Note du cours de "Concept de language informatique"

28 Février 2018
LINGI 1131
Valentin Haveaux

Exemple de fonction

Filter

```
%Filter
declare
fun{Filter L F}
  case L of H|T then
    if {F H} then H|{Filter T F}
    else {Filter T F}
    end
  [] nil then nil
  end
end

{Browse {Filter [1 2 3 4] fun{$ X} X mod 2 == 0 end}}

declare S1 S2 in
{Browse S1}
{Browse S2}

thread S2 = {Filter S1 fun{$ X} X mod 2 == 0 end} end

S1=1|2|3|4|5|_

S1=1|2|3|4|5|6|7|8|_
```

CFilter

```
%CFilter

declare
fun{CFilter L F}
  case L of H|T then
    if thread {F H} end then H|{CFilter T F}
```

```
    else {CFilter T F}
    end
  [] nil then nil
  end
end

declare S1 S2 in
{Browse S1}
{Browse S2}

thread S2 = {CFilter S1 fun{$ X} X mod 2 == 0 end} end
S1=1|2|3|_
```

Map

```
% Map
declare
fun{Map L F}
  case L of H|T then {F H}|{Map T F}
  [] nil then nil
  end
end

proc{Map L F R}
  case L of H|T then N K in
    R=N|K
    N={F H}
    {Map T F K}
  [] nil then R = nil
  end
end

declare S1 S2 in
{Browse S1}
{Browse S2}

thread S2={Map S1 fun{$ X} X*X end}end

S1=1|2|3|4|_

S1=1|2|3|4|5|5|6|7|_


declare S1 S2 in
{Browse S1}
{Browse S2}

thread S2={Map S1 fun {$ X} X*X end}end

S1=1|_|_|4|_
S1=1|2|3|4|_|6|7|_
```

CMap

```
declare
fun{CMap L F}
  case L of H|T then thread {F H}|{CMap T F}
  [] nil then nil
  end
end

declare S1 S2 in
{Browse S1}
{Browse S2}

thread S2={CMap S1 fun {$ X} X*X end}end
S1=1|_|_|4|_
S1=1|2|3|4|_|6|7|_
```

MyDelay

```
%MyDelay

declare
proc{MyDelay}
  {Delay {OS.rand} mod 5000}
end

declare S1 S2 in
{Browse S1}
{Browse S2}

thread S2={CMap S1 fun {$ X} {MyDelay} X*X end}end

S1=1|2|3|4|5|6|7|8|9|_
```

And

```
declare
fun{And A B}
  %if A==1 andthen B==1 then 1 else 0 end
  A*B
end

fun{AndLoop S1 S2}
  case S1|S2 of (A|T1)|(B|T2) then
```

```
    {And A B}|{AndLoop T1 T2}
  else nil
  end
end

declare S1 S2 S3 in
{Browse S1}
{Browse S2}
{Browse S3}

thread S3 = {AndLoop S1 S2} end

S1=0|1|0|1|_
S2=1|0|1|0|_
```

And Or Xor

```
declare
fun{GateMaker F}
  fun{$ S1 S2}
    fun{GateLoop S1 S2}
      case S1|S2 of (A|T1)|(B|T2) then
        {F A B}|{GateLoop T1 T2}
        else nil
        end
      end
    in
      thread {GateLoop S1 S2} end
  end
end

declare AndG OrG XorG in
AndG = {GateMaker fun{$ A B} A*B end}
OrG = {GateMaker fun{$ A B} A+B-A*B end} %if A ==1 orelse B==1 then 1 else 0 end}
XorG = {GateMaker fun{$ A B} A+B-2*A*B} %{Math.abs A-B}
                                        %(A+B) mod 2 if
                %(A==1 andthen B==0) orelse (A==0 andthen B==1) then 1 else 0 end}

declare S1 S2 S3 in
{Browse S1}
{Browse S2}
{Browse S3}

S3={AndG S1 S2}

S1=0|0|1|1|_
S2=1|1|0|0|_

declare S4 S5 in
{Browse S4}
{Browse S5}
```

```
S4={OrG S1 S1}
S5={XorG S1 S2}
```

FullAdder

Pour résoudre un schéma logique fait au tableau

> A AND B = AB
> A AND Ci = ACi
> B AND Ci = BCi
> AB OR ACi OR BCi = Co

> (A XOR B) XOR Ci = S

```
declare
proc{FullAdder X Y Ci S Co}
  A B C D E
in
  A={AndG X Y}
  B={AndG X Ci}
  C={AndG Y Ci}
  D={OrG A B}
  Co={OrG C D}

  E={XorG A Y}
  S={XorG E Ci}
end

declare X Y Ci S Co in
{Browse X}
{Browse Y}
{Browse Ci}
{Browse S}
{Browse Co}

{FullAdder X Y Ci S Co}

X=0|0|0|0|1|1|1|1|_
Y=0|0|1|1|0|0|1|1|_
Ci=0|1|0|1|0|1|0|1|_
```

NFullAdder

```
declare
proc{NFullAdder X Y Ci S Co}
```

```
  case X|Y of (A|T1)|(B|T2) then S1 S2 C2 in
    S=S1|S2
    {FullAdder A B Ci S1 C2}
    {NFullAdder T1 T2 C2 S2 Co}
  else
    S1=nil
    Co=Ci
  end
end

declare X0 X1 Y0 Y1 Y2 Ci S C0 in
{NfullAdder [X0 X1 X2] [Y0 Y1 Y2] Ci S Co}

{Browse X0}
{Browse X1}
{Browse X2}
{Browse Y0}
{Browse Y1}
{Browse Y2}
{Browse Ci}

{Browse S}
{Browse Co}

X0=1|0|_
X1=1|1|_
X2=0|1|_
Y0=1|1|_
Y1=1|0|_
Y2=0|1|_
Ci=0|1|_
```

WaitNeedEd

```
declare X in
{WaitNeedEd X}
X=100*100
{Browse X}

{Browse X+1}

declare
fun{F1 X} X*X end
{Browse {F1 100}}

declare
fun lazy {F1 X} X*X end

declare B in
B={F2 100}
{Browse B}
```

```
{Browse B+1}


declare
proc {F3 X R}
  thread {WaitNeedEd R} R=X*X end
end

declare
fun{Prod N}
  fun lazy {Prodn M}
    if M =< N then
      {MyDelay}
      M|{Prodn M+1}
      else
        nil
      end
    end
in
  {Prodn O}
end

declare P in
{Browse P}

P={Prod 10}

declare R in
{Browse R}

R={Map P fun{$ X} X+1 end}
```

BoundBuffer

```
declare
proc{BoundBuffer S1 S2 N}
  fun lazy {loop S1 End}
    case S1  of H|T then
      H|{Loop T thread End.2 end}
      end
    end
    End
  in
    thread End={List.drop S1 N} end
    S2={Loop S1 End}
  end

declare
fun lazy{Prod N}
  {Delay 2500}
  N|{Prod N+1}
end
```

```
declare
fun lazy {Cons S Acc}
  case S of H|T then Acc|{Cons T H+Acc} end
end

declare S1 S2 in
{Browse S1}
{Browse S2}

S1={Prod 1}

{BoundBuffer S1 S2 3}

S2 = {Cons S1 0}

{Browse S2.1}
{Browse S2.2.1}
{Browse S2.2.2.1}
```