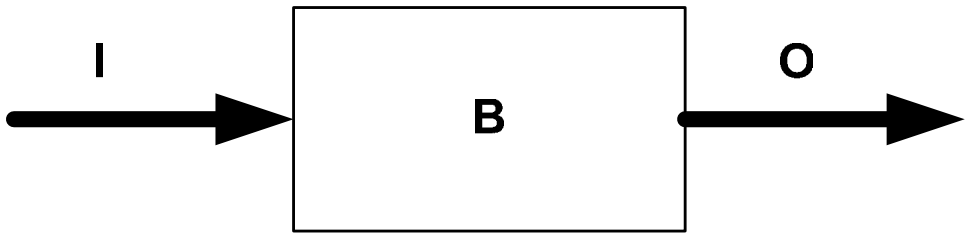
Logical semantics

Esterel v5

Comportement



$$B : I^* \rightarrow O^*$$

Réactivité: $\forall x \in I^* : B(x) \neq \varnothing$

Déterminisme: $B : I^* \rightarrow O^*$ est une fonction

Machines de Mealy

- Les ensembles I* et O* sont infinis
- Dans le cas des machines séquentielles on peut arriver à une description finie en introduisant la notion d'état:

Cas général

$$\delta : S \times I \rightarrow 2^S \quad \text{fonction état suivant}$$

$$\omega : S \times I \rightarrow 2^O \quad \text{fonction de sortie}$$

Cas particulier des machines complètement spécifiés

$$\delta : S \times I \rightarrow S \quad \text{fonction état suivant}$$

$$\omega : S \times I \rightarrow O \quad \text{fonction de sortie}$$

Sémantiques pour Esterel

- Adopter les machines de Mealy?
- Problèmes posés:
  - Problèmes liés au caractère non structuré des machines :

    Comment prendre en compte le parallélisme?

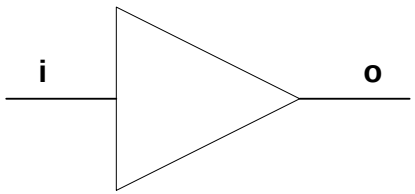
    Comment prendre en compte les préemptions?

    Comment prendre en compte la structiration du programme?
  - Problèmes liés aux hypothèses synchrones :

    Quels sont les programmes qui sont réactifs et déterministes?

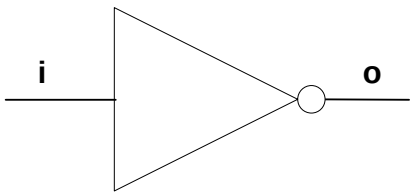
    Parmi les programmes précédents, quels sont ceux qui ont une interprétation « naturelle »?

Correction logique (1)



present i then emit o end

$o = i$ (buffer)



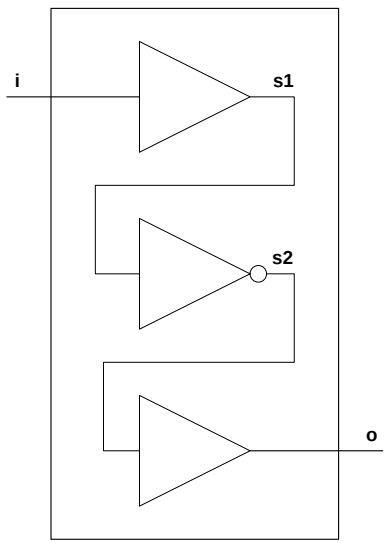
present i else emit o end

$o = i'$ (inverseur)

Ce sont deux systèmes réactifs et déterministes

Correction logique (2)

- Composition réactive et déterministe

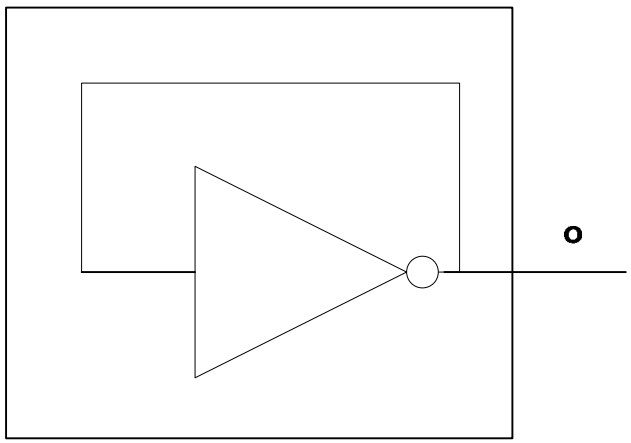


$$\left.\begin{array}{l} s1 = i \\ s2 = s1' \\ o = s2 \end{array}\right\} \Rightarrow o = i'$$

```
module P1:
  input i; output o;
  signal s1, s2 in
    present i then emit s1 end
   ||
    present s1 else emit s2 end
   ||
    present s2 then emit o end
  end signal
end module
```

Correction logique (3)

- Composition non réactive



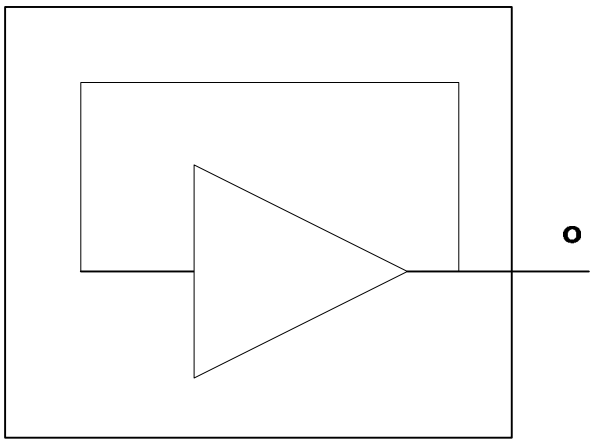

```
module P2:
   output o;
   present o else emit o end
end module
```

$$o = o'$$

Pas de solution logique

Correction logique (4)

- Composition non déterministe



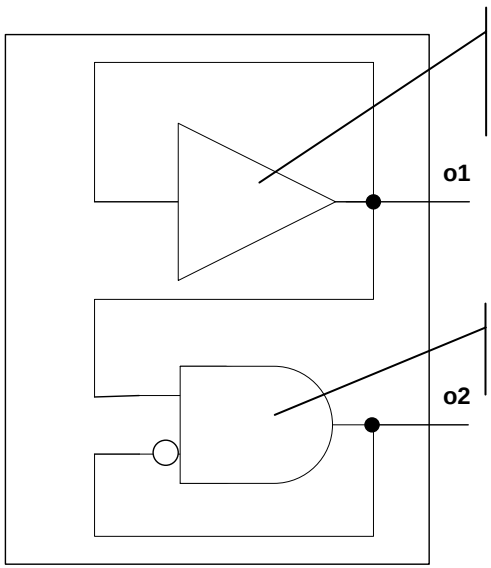

```
module P3:
   output o;
   present o then emit o end
end module
```

$$O = O$$

deux solutions logiques
o=0 ou o=1

Correction logique (5)

- Cas « étrange »



Non déterministe

Non réactif

```
module P6:
  output o1, o2;
  present o1 then emit o1 end
 ||
  present o1 then
    present o2 else emit o2 end
  end
end module
```

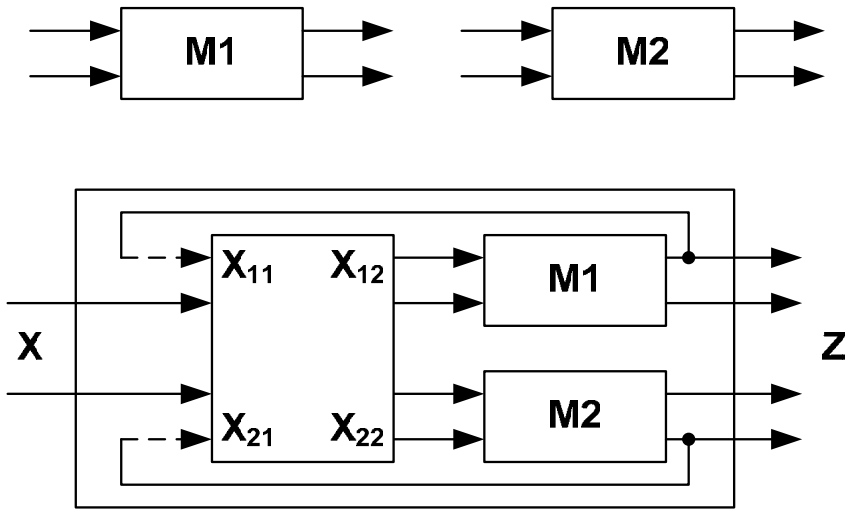
$$o1 = o1$$

$$o2 = o1 \bullet o2'$$

1 solution logique
o1=0 et o2=0

Correction logique (6)

- Généralisation



$$X_1^+ = C_1(X, Z_1, Z_2)$$

$$X_2^+ = C_2(X, Z_1, Z_2)$$

$$Z_1 = F_1(X_1^+) = F_1 \circ C_1(X, Z_1, Z_2)$$

$$Z_2 = F_2(X_2^+) = F_2 \circ C_2(X, Z_1, Z_2)$$

Calcul de point fixe:
0, 1, … solutions!

$$Z = F(X, Z)$$

Correction logique (6)

- Cohérence logique:

  Un signal S qui n'est pas un signal d'entrée, est présent à un instant si et seulement si `emit S` est exécuté dans cet instant.

- Un événement d'entrée pour un programme donné est une application de son ensemble de signaux d'entrée dans `{present, absent}`
- Réactivité logique pour un événement :

  Un programme est logiquement réactif pour un événement d'entrée si il existe au moins une solution telle que chaque signal respecte la loi de cohérence logique

- Déterminisme logique pour un événement :

  Un programme est logiquement déterministe pour un événement d'entrée si il existe au plus une solution telle que chaque signal respecte la loi de cohérence logique

- Correction logique pour un événement :

  A la fois réactif et déterministe

- Programme logiquement correct :

  Correct pour tous les événements d'entrée possibles

Correction logique (6)

- La correction logique d'un programme Esterel pur est décidable. (il suffit de faire les hypothèses sur l'état de présence de tous les signaux. Ceci est très lourd!)

```
module P7:
  output O;
  signal S in
    emit S;
    present O then
      present S then
        pause
      end present;
      emit O
    end present
  end signal
end module
```

S est present

Soit O est present
Soit O est absent

Si O est present, comme S est présent: on fait pause
On n'émet donc pas O: contradiction

Si O est absent, on n'a rien à faire.
Donc O non émis: cohérent

Solution logique unique: O absent

Correction logique (7)

```
module P8:
  input I;
  output O;
  trap T in
    present I else pause end;
    emit O
   ||
    present O then exit T end;
  end trap;
  emit O
end module
```

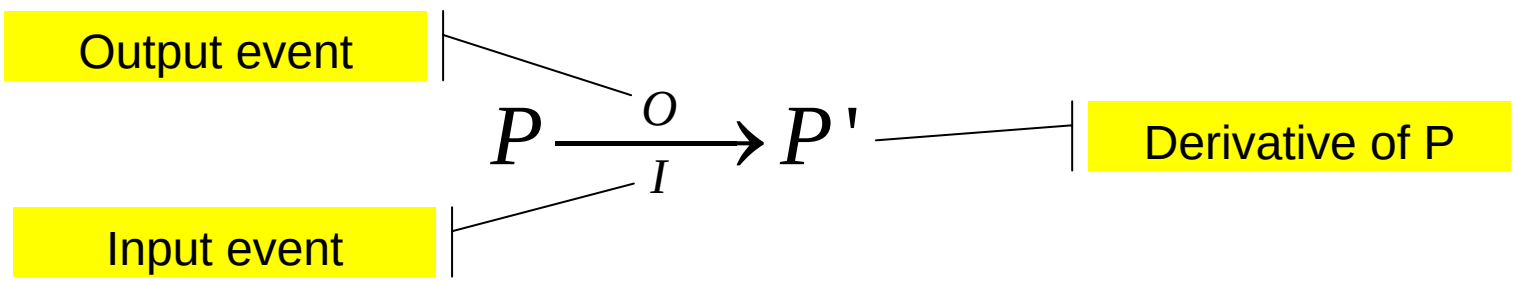
Exercice: montrer que
Si I présent: logiquement correct;
Si I absent: non déterministe

Semantics (1)

- The reaction of program *P* to the input sequence $I_1 \bullet I_2 \bullet \cdots \bullet I_n \bullet \cdots$ is the output sequence $O_1 \bullet O_2 \bullet \cdots \bullet O_n \bullet \cdots$ such that there exist $P_1, P_2, \cdots, P_n, \cdots$ derived programs

$$P = P_1 \xrightarrow[I_1]{O_1} P_2 \xrightarrow[I_2]{O_2} \cdots P_n \xrightarrow[I_n]{O_n} P_{n+1} \xrightarrow[I_{n+1}]{O_{n+1}} \cdots$$

The behavioral semantics formalizes a reaction of a program *P* as a behavioral transition of the form



Semantics (2)

- Coding states by program texts is standard in process calculi definitions based on Structural Operational Semantics (SOS) rules

$$P \rightarrow p$$

Program

process

Synchronous process calculus

0	**nothing**
1	**pause**
$!s$	**emit *s***
$s\,?\,p,q$	**present *s* then *p* else *q* end**
$s \supset p$	**suspend *p* when *s***
$p;q$	***p;q***
$p*$	**loop *p* end**
$p \mid q$	***p||q***
$\{p\}$	**trap *T* in *p* end**
$\uparrow p$	
k avec $k \geq 2$	**exit *T***
$p \setminus s$	**signal *s* in *p* end**

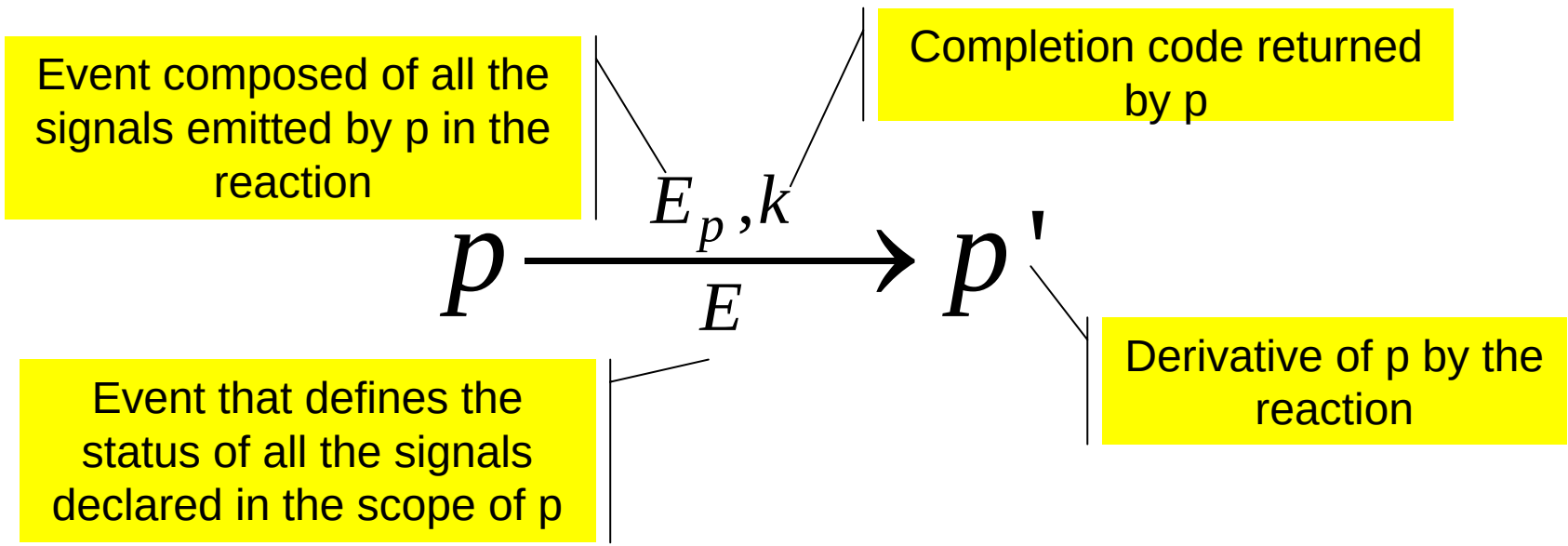
Nested Traps

```
trap U in
     trap T in
          nothing    0
     ||
          pause      1
     ||
          exit T     2
     ||
          exit U     3
     end trap
||
     exit U          2
end trap
```

$$\{\{0 \mid 1 \mid 2 \mid 3\} \mid 2\}$$

Semantics (3)

- Reactions are computed using an auxiliary statement transition relation, which has the following form



Semantics (4)

- Given a program *P* of body *p* and an input event *I*, the program transition of *P* is defined in terms of the statement transition of *p* in the following way:

$$P \xrightarrow[I]{O} P' \quad \text{iff} \quad \exists \mathrm{k}: p \xrightarrow[I \cup O]{O,k} p'$$

Logical coherence of the global event

Termination & Emit

$$k \xrightarrow[E]{\varnothing, k} 0 \quad (term)$$

k = 0: nothing
k = 1: pause
k > 1: exit

$$!S \xrightarrow[E]{\{S\}, 0} 0 \quad (emit)$$

Present

$$\frac{S \in E \quad p \xrightarrow[E]{E_p,k} p'}{S\,?\,p,q \xrightarrow[E]{E_p,k} p'} \quad (pres+)$$

$$\frac{S \notin E \quad q \xrightarrow[E]{E_q,k} q'}{S\,?\,p,q \xrightarrow[E]{E_p,k} q'} \quad (pres-)$$

Sequence

$$\frac{p \xrightarrow[E]{E_p,k} p' \qquad k \neq 0}{p;q \xrightarrow[E]{E_p,k} p';q} \; (seq1)$$

$$\frac{p \xrightarrow[E]{E_p,0} p' \qquad q \xrightarrow[E]{E_q,k} q'}{p;q \xrightarrow[E]{E_p \cup E_q,k} q'} \; (seq2)$$

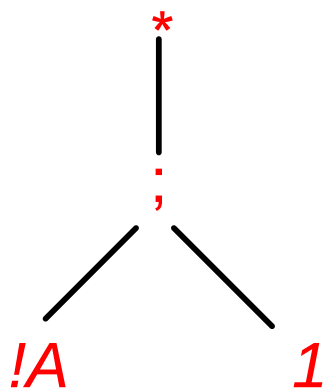
Parallel & Loop

$$\frac{p \xrightarrow[E]{E_p,k} p' \qquad q \xrightarrow[E]{E_q,l} q'}{p \mid q \xrightarrow[E]{E_p \cup E_q,\max(k,l)} p' \mid q'} \; (paral)$$

$$\frac{p \xrightarrow[E]{E_p,k} p' \qquad k \neq 0}{p* \xrightarrow[E]{E_p,k} p'; p*} \; (loop)$$

Deduction tree

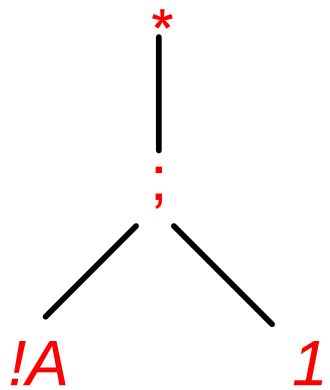
`loop emit A; pause end loop` $p = (!A;1)*$



$$p = (!A;1)* = p_1 * \xrightarrow[E]{?,?} ?$$

Deduction tree

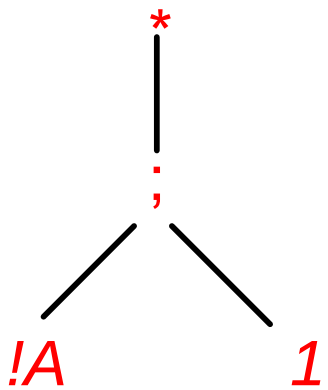
`loop emit A; pause end loop` $p = (!A;1)*$



$$\frac{\dfrac{}{p_1 = (!A;1) \xrightarrow[E]{?,?}}}{p = (!A;1)* = p_1 * \xrightarrow[E]{?,?}}$$

Deduction tree

`loop emit A; pause end loop` $p = (!A;1)*$

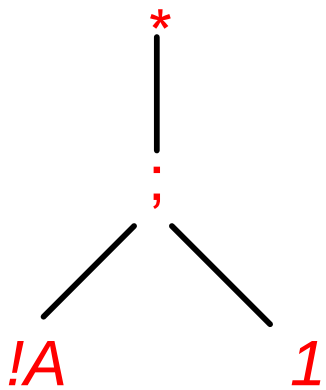


$$\frac{!A \xrightarrow[E]{\{A\},0} 0\ (\text{emit}) \quad 1 \xrightarrow[E]{\varnothing,1} 0\ (\text{term})}{p_1 = (!A;1) \xrightarrow[E]{?,?}}$$

$$p = (!A;1)* = p_1 * \xrightarrow[E]{?,?}$$

Deduction tree

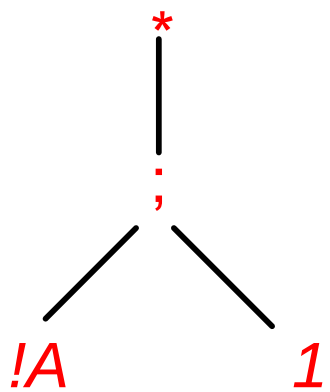
`loop emit A; pause end loop` $p = (!A;1)*$



$$\dfrac{\dfrac{!A \xrightarrow[E]{\{A\},0} 0 \ \text{(emit)} \quad 1 \xrightarrow[E]{\varnothing,1} 0 \ \text{(term)}}{p_1 = (!A;1) \xrightarrow[E]{\{A\},1} 0}\ \text{(seq2)}}{p = (!A;1)* = p_1 * \xrightarrow[E]{?,?}}$$

Deduction tree

`loop emit A; pause end loop` $p = (!A;1)^*$



$$\dfrac{\dfrac{!A \xrightarrow[E]{\{A\},0} 0 \ \text{(emit)} \quad 1 \xrightarrow[E]{\varnothing,1} 0 \ \text{(term)}}{p_1 = (!A;1) \xrightarrow[E]{\{A\},1} 0}\ \text{(seq2)}}{p = (!A;1)^* = p_1{}^* \xrightarrow[E]{\{A\},1} 0;\ p_1{}^* = p_1{}^* = p}\ \text{(loop)}$$

Suspension

$$\frac{p \xrightarrow[E]{E_p,k} p' \qquad k \neq 0}{s \supset p \xrightarrow[E]{E_p,k} s \widehat{\supset} p'} \; (susp1)$$

$$\frac{p \xrightarrow[E]{E_p,0} p'}{s \supset p \xrightarrow[E]{E_p,0} 0} \; (susp2)$$

Immediate Suspension

$$\left(s \widehat{\supset} p\right) \equiv s \supset \left(\left(s?1,0\right); p\right)$$

suspend *p* when immediate *s* =

```
suspend
      present s then pause else nothing end;
      p
when s
```

Trap

$$\frac{p \xrightarrow[E]{E_p,k} p' \quad (k=0) \vee (k=2)}{\{p\} \xrightarrow[E]{E_p,0} 0} \quad (trap1)$$

$$\frac{p \xrightarrow[E]{E_p,k} p' \quad (k=1) \vee (k>2)}{\{p\} \xrightarrow[E]{E_p,\downarrow k} \{p'\}} \quad (trap2)$$

Shift

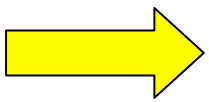
$$\frac{p \xrightarrow[E]{E_p,k} p'}{\uparrow p \xrightarrow[E]{E_p,\uparrow k} \uparrow p'}\ (shift)$$

$$\downarrow k = \begin{cases} 0 & \text{if } (k=0) \vee (k=2) \\ 1 & \text{if } k=1 \\ k-1 & \text{if } k>2 \end{cases}$$

$$\uparrow k = \begin{cases} k & \text{if } (k=0) \vee (k=1) \\ k+1 & \text{if otherwise} \end{cases}$$

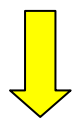
Example with trap (1)

```
p1::=
pause;
emit A
```

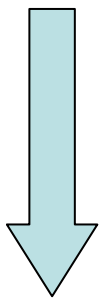


$p_1 = 1;!A$

```
p2::=
trap T in
    pause;
    exit T
end trap;
emit A
```



$p_2 = \{1;2\};!A$



$?\ p_1 \equiv p_2$

Example with trap (2)

First instant

$$\dfrac{1 \xrightarrow[E]{\varnothing,1} 0 \ \text{(term)}}{p_1 = (1;!A) \xrightarrow[E]{\varnothing,1} 0;!A = !A}\ \text{(seq1)}$$

$$\dfrac{\dfrac{\dfrac{1 \xrightarrow[E]{\varnothing,1} 0 \ \text{(term)}}{(1;2) \xrightarrow[E]{\varnothing,1} 0;2=2}\ \text{(seq1)}}{\{1;2\} \xrightarrow[E]{\varnothing,1} \{2\}}\ \text{(trap2)}}{p_2 = (\{1;2\};!A) \xrightarrow[E]{\varnothing,1} \{2\};!A}\ \text{(seq1)}$$

Second instant

$$!A \xrightarrow[E]{\{A\},0} 0 \ \text{(emit)}$$

$$\dfrac{\dfrac{2 \xrightarrow[E]{\varnothing,2} 0 \ \text{(term)}}{\{2\} \xrightarrow[E]{\varnothing,0} 0}\ \text{(trap1)} \qquad !A \xrightarrow[E]{\{A\},0} 0 \ \text{(emit)}}{(\{2\};!A) \xrightarrow[E]{\{A\},0} 0}\ \text{(seq2)}$$

Same behaviors: equivalent

Weak abort

```
weak abort p when s
≡
trap T in
     p; exit T
||
     loop
          pause;
          present s then
               exit T
          end present
     end loop
end trap
```

notation

$$s > p$$

$$\equiv$$

$$\{((\uparrow p; 2) \mid (1; s\,?\,2, 0)*\}$$

Behavior of the weak abort

First instant: either

$$\dfrac{\dfrac{\dfrac{p \xrightarrow[E]{E_p,0} 0}{\uparrow p \xrightarrow[E]{E_p,0} 0}\,(\text{shift}) \qquad 2 \xrightarrow[E]{\varnothing,2} 0\,(\text{term})}{(\uparrow p;2) \xrightarrow[E]{E_p,2} 0}\,(\text{seq2}) \qquad \dfrac{\dfrac{1 \xrightarrow[E]{\varnothing,1} 0\,(\text{term})}{(1;s?2,0) \xrightarrow[E]{\varnothing,1} 0;s?2,0=s?2,0}\,(\text{seq1})}{(1;s?2,0)^* \xrightarrow[E]{\varnothing,1} s?2,0;(1;s?2,0)^*}\,(\text{loop})}{\left((\uparrow p;2)\,|\,(1;s?2,0)^*\right) \xrightarrow[E]{E_p,2} 0\,|\,(s?2,0;(1;s?2,0)^*)}\,(\text{paral})}{\left\{(\uparrow p;2)\,|\,(1;s?2,0)^*\right\} \xrightarrow[E]{E_p,0} 0}\,(\text{trap 1})$$

or

$$\dfrac{\dfrac{\dfrac{p \xrightarrow[E]{E_p,k} p' \quad k\neq 0}{\uparrow p \xrightarrow[E]{E_p,\uparrow k} \uparrow p' \quad k\neq 0}\,(\text{shift}) \qquad 1 \xrightarrow[E]{\varnothing,1} 0\,(\text{term})}{(\uparrow p;2) \xrightarrow[E]{E_p,\uparrow k} (\uparrow p';2) \quad k\neq 0}\,(\text{seq1}) \qquad \dfrac{\dfrac{}{(1;s?2,0) \xrightarrow[E]{\varnothing,1} 0;s?2,0=s?2,0}\,(\text{seq1})}{(1;s?2,0)^* \xrightarrow[E]{\varnothing,1} s?2,0;(1;s?2,0)^*}\,(\text{loop})}{\left((\uparrow p;2)\,|\,(1;s?2,0)^*\right) \xrightarrow[E]{E_p,\max(\uparrow k,1)} (\uparrow p';2)\,|\,(s?2,0;(1;s?2,0)^*)}\,(\text{paral})}{\left\{(\uparrow p;2)\,|\,(1;s?2,0)^*\right\} \xrightarrow[E]{E_p,\max(\uparrow k,1)} \left\{(\uparrow p';2)\,|\,(s?2,0;(1;s?2,0)^*)\right\}}\,(\text{trap 2})$$

Homework exercise: compute the second instant reaction

Strong abort

$$s \gg p$$

notation

$$\equiv$$

$$s > (s \supset p)$$

behavior

$$\equiv$$

$$\{((s \supset\uparrow p);2)\,|\,(1;s\,?\,2,0)*\}$$

halt

Special case: await s ≡ [weak] abort pause* when s

$$s > 1* \equiv s \gg 1* \equiv \{(1;s\,?\,2,0)*\} \equiv 1;\{(s\,?\,2,1)*\}$$

Local Signals

$$\frac{p \xrightarrow[E\setminus\{s\}]{E_p,k} p' \quad s \notin E_p}{p \setminus s \xrightarrow[E]{E_p,k} p' \setminus s} \quad (sig-)$$

$$\frac{p \xrightarrow[E\cup\{s\}]{E_p,k} p' \quad s \in E_p}{p \setminus s \xrightarrow[E]{E_p\setminus\{s\},k} p' \setminus s} \quad (sig+)$$

Réactivité et déterminisme

- Un programme P est réactif pour un événement d'entrée I si il existe un événement de sortie O et un programme P' tel que $P \xrightarrow[I]{O} P'$
- Le programme est déterministe pour I s'il existe au plus un O et un P'.
- Le programme est logiquement correct pour I s'il est réactif et déterministe pour I.
- Le programme est logiquement correct si il est logiquement correct pour tous les événements d'entrée et si toutes ses dérivées sont logiquement correctes.
- Un programme P est fortement déterministe pour I s'il est réactif et déterministe pour I et si il existe une preuve unique de $P \xrightarrow[I]{O} P'$

Notion de sémantique constructive

```
present I then
   emit O
end present

module P9:
   output O;
   present O then nothing end;
   emit O
end module
```

Cause

Effet (conséquence)

Logiquement correct

Le test est évalué sur un état fixé ultérieurement

Conséquence?

Séquence!

Sémantique constructive (1)

- Respecte la sémantique du ;
- Ne fait pas d'hypothèses : elle se contente de propager des faits.
- Les faits sont ce que le programme doit faire (must) et ce qu'il ne peut pas faire (cannot).
- Se limiter aux solutions logiquement correctes qui respectent la causalité.

Sémantique constructive (2)

Raisonnements en analyse constructive.

ex: **`present S then p else q end`**

trois cas à envisager :

1. On sait que **S** est présent, alors le test doit se comporter comme **p** et ne peut pas se comporter comme **q**.
2. On sait que **S** est absent, alors le test doit se comporter comme **q** et ne peut pas se comporter comme **p**.
3. On ne sait pas encore si **S** est présent ou non, alors le test peut se comporter comme **p** ou **q**, par contre, on ne peut rien dire sur ce qui doit être.

Sémantique constructive (3)

si **i** est présent (Rq: **i** est un signal d'entrée)

```
module P1:
   input i; output o;
   signal s1, s2 in
     present i then emit s1 end
    ||
     present s1 else emit s2 end
    ||
     present s2 then emit o end
   end signal
 end module
```

Sémantique constructive (3)

si **i** est présent (Rq: **i** est un signal d'entrée)

```
module P1:
   input i+; output o;
   signal s1, s2 in
     present i+ then emit s1 end
    ||
     present s1 else emit s2 end
    ||
     present s2 then emit o end
   end signal
 end module
```

Sémantique constructive (3)

si **i** est présent (Rq: **i** est un signal d'entrée)

```
module P1:
   input i+; output o;
   signal s1, s2 in
     present i+ then emit s1 end
    ||
     present s1+ else emit s2 end
    ||
     present s2 then emit o end
   end signal
 end module
```

Sémantique constructive (3)

si **i** est présent (Rq: **i** est un signal d'entrée)

```
module P1:
   input i+; output o;
   signal s1, s2 in
     present i+ then emit s1 end
    ||
     present s1+ else emit s2 end
    ||
     present s2- then emit o end
   end signal
end module
```

O non émis, donc absent

Sémantique constructive (4)

si **i** est absent (Rq: **i** est un signal d'entrée)

```
module P1:
   input i; output o;
   signal s1, s2 in
     present i- then emit s1 end
    ||
     present s1- else emit s2 end
    ||
     present s2+ then emit o end
   end signal
 end module
```

O est émis, donc présent

Sémantique constructive (5)

```
module P7:
  output O;
  signal S in
    emit S;+
    present O then
      present S then
        pause
      end present;
      emit O
    end present
  end signal
end module
```

Sémantique constructive (5)

```
module P7:
  output O;
  signal S in
    emit S;+
    present O then
      present S+then pause
      else nothing
      end present;
      emit O
    end present
  end signal
end module
```

Sémantique constructive (5)

```
module P7:
  output O;
  signal S in
    emit S;+
    present O then
      present S+then pause
      else nothing
      end present;
      emit O -
    end present
  end signal
end module
```

Sémantique constructive (5)

```
module P7:
  output O;
  signal S in
    emit S;+
    present O- then
      present S+ then pause
      else nothing
      end present;
      emit O -
    end present
  end signal
end module
```

Sémantique constructive (5)

```
module P7:
  output O;
  signal S in
    emit S;+
    present O- then
      present S+ then pause
      else nothing
      end present;
      emit O-
    end present
  end signal
end module
```

Constructif :
O est absent

Sémantique constructive (6)

```
module P9:
   output O;
   present O then
      nothing
   else
      nothing
   end;
   emit O
end module
```

Statut inconnu

On n'a pas d'autre information sur ce qui doit se faire ou ne peut pas se faire

On ne peut pas aller plus loin.
Non constructif.