
Une Introduction aux algèbres de processus



Elie Najm

elie.najm@telecom-paristech.fr

Introduction

- Fondements des « algèbres de processus » :
R. Milner (1980): *A Calculus of Communicating Systems* - Springer-Verlag.
- Principes :
 - le « processus » est l'unité de **comportement**. Une spécification est une composition de processus,
 - le **comportement** d'un processus se manifeste par les **actions successives** que ce processus peut exécuter.
 - Les **actions** sont soit **internes**, soit des **interactions** avec l'environnement,
 - une algèbre de processus est définie par ses **opérateurs de composition des comportements**,
 - parmi les opérateurs usuels : la **mise en parallèle**, la **préemption**, le **choix**, la **synchronisation**, ...
 - le **texte source** d'un processus représente son **état** : **l'exécution** d'une action par un processus se traduit par modifie (**réécrit**) le **texte** de ce processus.
- **CCS +** : CCS avec variables et valeurs et opérateur de préemption.

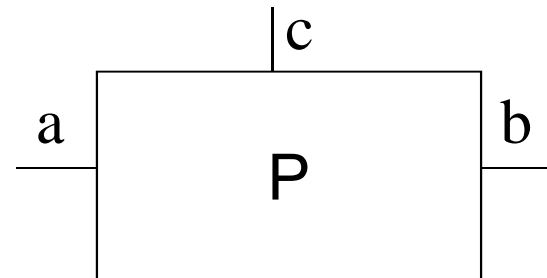
Éléments de base

- L'unité dont on décrit le comportement est le *processus*
- Un *processus* est doté d'un ensemble de portes à travers desquels il interagit avec son environnement
- Pour un *processus* P , muni des portes a , b et c , et ayant le comportement C , on écrira :

Process

$P[a, b, c] := C$

Endproc



- C est une Expression de Comportement (EdC) – voir suite ...

Syntaxe des expressions de comportements (EdC) -

- Un parallèle avec la syntaxe de l'arithmétique :
- Les opérateurs $+$, $*$, $-$, $/$, permettent de générer des expressions arithmétiques composites à partir d'expressions plus simples. Exemple :

Ea

12-4

Ec

13+2

Ea*Ec

(12-4)*(13+2)

- De même, en CCS+, des opérateurs permettent la composition d'Expressions de Comportements (EdC) en partant d'autres Expressions de Comportement plus simples.

Syntaxe des Expressions de Comportements

Les composantes syntaxiques des Expressions Arithmétiques :

- Les opérateurs binaires : $+$, $*$, $-$, $/$. Exemples : $4+2$, $17-3$
- Les opérateurs unaires : $-$, $1/$. Exemples : -130 , $1/117$
- Les opérateurs nullaires, çad, les nombres : 4 , 12 , ...

Les composantes syntaxiques des Expressions de Comportement :

- Les opérateurs nullaires : **STOP**
- Les opérateurs unaires : $a ; C$, $[x] \rightarrow C$, $C \setminus a_1, \dots, a_n$
- Les opérateurs binaires : $C_1 (+) C_2$, $C_1 ||| C_2$, $C_1 [>] C_2$

—— CCS+ par les exemples ——

```
PROCESS ne_fait_rien :=  
    STOP  
ENDPROC
```

CCS+ par les exemples

```
PROCESS Buffer_Puit [ a ] :=  
    a?x:nat ; STOP  
  
ENDPROC
```



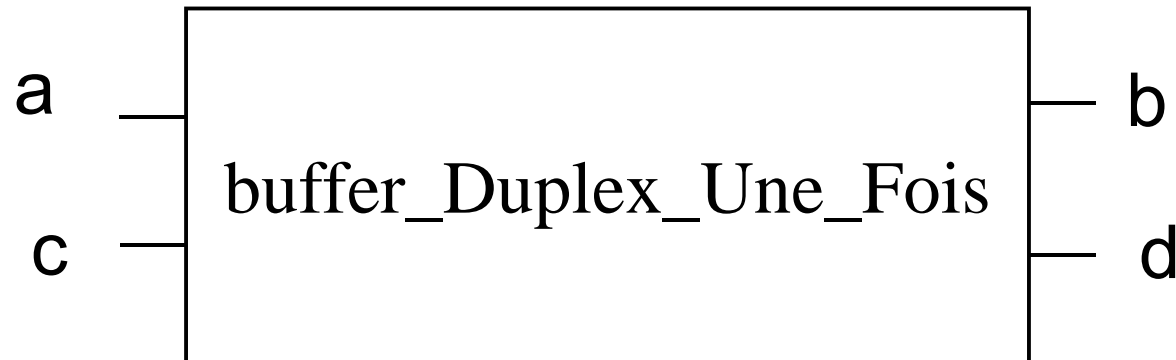
CCS+ par les exemples

```
PROCESS Buffer_Une_Fois [ a, b ] :=  
    a?x:nat ; b!x ; STOP  
  
ENDPROC
```



CCS+ par les exemples

PROCESS buffer_Duplex_Une_Fois [a, b, c, d] :=



ENDPROC

CCS+ par les exemples

PROCESS buffer_Duplex_Une_Fois [a,b,c,d] :=

```
( a?x ; ( c?y ; ( b!x ; d!y ; STOP )  
          (+)  
          ( d!y ; b!x; STOP )  
        )  
      )  
    (+)  
    ( b!x ; c?y ; d!y ; STOP )  
  )  
(+)  
( c?x ; ( a?y ; ( d!x ; b!y ; STOP )  
          (+)  
          ( b!y ; d!x; STOP )  
        )  
      )  
    (+)  
    ( d!x ; a?y ; b!y ; STOP )  
  )  
)
```

ENDPROC

CCS+ par les exemples

PROCESS buffer_Duplex_Une_Fois [a,b,c,d] :=

```
( a?x ; ( c?y ; ( ( b!x ; d!y ; STOP )  
          (+)  
          ( d!y ; b!x; STOP )  
        )  
      )  
    (+)  
    ( b!x ; c?y ; d!y ; STOP )  
  )  
  (+)  
  ( c?x ; ( a?y ; ( ( d!x ; b!y ; STOP )  
                  (+)  
                  ( b!y ; d!x; STOP )  
                )  
        )  
      (+)  
      ( d!x ; a?y ; b!y ; STOP )  
    )  
  )
```

ENDPROC

CCS+ par les exemples

PROCESS buffer_Duplex_Une_Fois [a,b,c,d] :=

```
( a?x ; ( c?y ; ( ( b!x ; d!y ; STOP )  
          (+)  
          ( d!y ; b!x ; STOP )  
        )  
      )  
    (+)  
    ( c?x ; ( a?y ; ( ( d!x ; b!y ; STOP )  
                    (+)  
                    ( b!y ; d!x ; STOP )  
                  )  
          (+)  
          ( d!x ; a?y ; b!y ; STOP )  
        )  
      )  
    )
```

ENDPROC

CCS+ par les exemples

PROCESS buffer_Duplex_Une_Fois [a,b,c,d] :=

```
( a?x ; ( c?y ; ( b!x ; d!y ; STOP )  
          (+)  
          ( d!y ; b!x ; STOP )  
        )  
      (+)  
      ( b!x ; c?y ; d!y ; STOP )  
    )  
  (+)  
  ( c?x ; ( a?y ; ( d!x ; b!y ; STOP )  
          (+)  
          ( b!y ; d!x ; STOP )  
        )  
      (+)  
      ( d!x ; a?y ; b!y ; STOP )  
    )  
  )
```

ENDPROC

CCS+ par les exemples

PROCESS buffer_Duplex_Une_Fois [a,b,c,d] :=

```
( a?x ; ( c?y ; ( b!x ; d!y ; STOP )  
          (+)  
          ( d!y ; b!x ; STOP )  
        )  
      )  
  (+)  
  ( c?x ; ( a?y ; ( d!x ; b!y ; STOP )  
            (+)  
            ( b!y ; d!x ; STOP )  
          )  
        )  
      (+)  
      ( d!x ; a?y ; b!y ; STOP )  
    )  
  )
```

ENDPROC

CCS+ par les exemples

PROCESS buffer_Duplex_Une_Fois [a,b,c,d] :=

```
( c?y ; ( ( b!x ; d!y ; STOP )  
          (+)  
          ( d!y ; b!x; STOP )  
        )  
      )  
      (+)  
      ( b!x ; c?y ; d!y ; STOP )
```

ENDPROC

CCS+ par les exemples

PROCESS buffer_Duplex_Une_Fois [a,b,c,d] :=

```
( c?y ; (( b!x ; d!y ; STOP )  
      (+)  
      ( d!y ; b!x; STOP )  
      )  
  )  
(+)  
( b!x ; c?y ; d!y ; STOP )
```

ENDPROC

CCS+ par les exemples

PROCESS buffer_Duplex_Une_Fois [a,b,c,d] :=

```
c?y ; (( b!x ; d!y ; STOP )  
      (+)  
      ( d!y ; b!x; STOP )  
      )
```

ENDPROC

CCS+ par les exemples

PROCESS buffer_Duplex_Une_Fois [a,b,c,d] :=

```
(( b!x ; d!y ; STOP )  
(+)  
( d!y ; b!x; STOP )  
)
```

ENDPROC

CCS+ par les exemples

PROCESS buffer_Duplex_Une_Fois [a,b,c,d] :=

```
( ( b!x ; d!y ; STOP )  
  (+)  
  ( d!y ; b!x ; STOP )  
 )
```

ENDPROC

CCS+ par les exemples

PROCESS buffer_Duplex_Une_Fois [a,b,c,d] :=

```
((b!x d!y ; STOP )  
(+  
(d!y ; b!x; STOP )  
)
```

ENDPROC

CCS+ par les exemples

PROCESS buffer_Duplex_Une_Fois [a,b,c,d] :=

b!x ; d!y ; **STOP**)

ENDPROC

CCS+ par les exemples

PROCESS buffer_Duplex_Une_Fois [a,b,c,d] :=

...; d!y; **STOP**)

ENDPROC

CCS+ par les exemples

PROCESS buffer_Duplex_Une_Fois [a,b,c,d] :=

; **STOP**)

ENDPROC

CCS+ par les exemples : la mise en parallèle

PROCESS buffer_Duplex_Une_Fois [a, c, b, d] :=

(a?x ; b!x ; **STOP**)

|||

(c?y ; d!y ; **STOP**)

ENDPROC

CCS+ par les exemples : la mise en parallèle

PROCESS buffer_Duplex_Une_Fois [a, c, b, d] :=

(a?x ; b!x ; **STOP**)

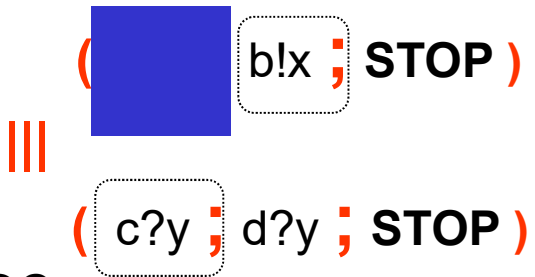
|||

(c?y ; d?y ; **STOP**)

ENDPROC

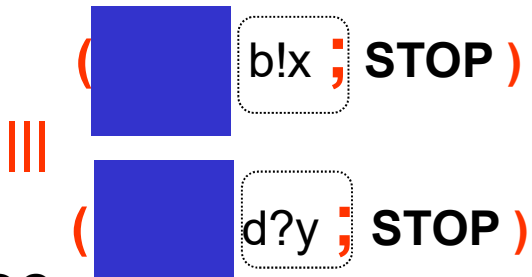
CCS+ par les exemples : la mise en parallèle

PROCESS buffer_Duplex_Une_Fois [a, c, b, d] :=


ENDPROC

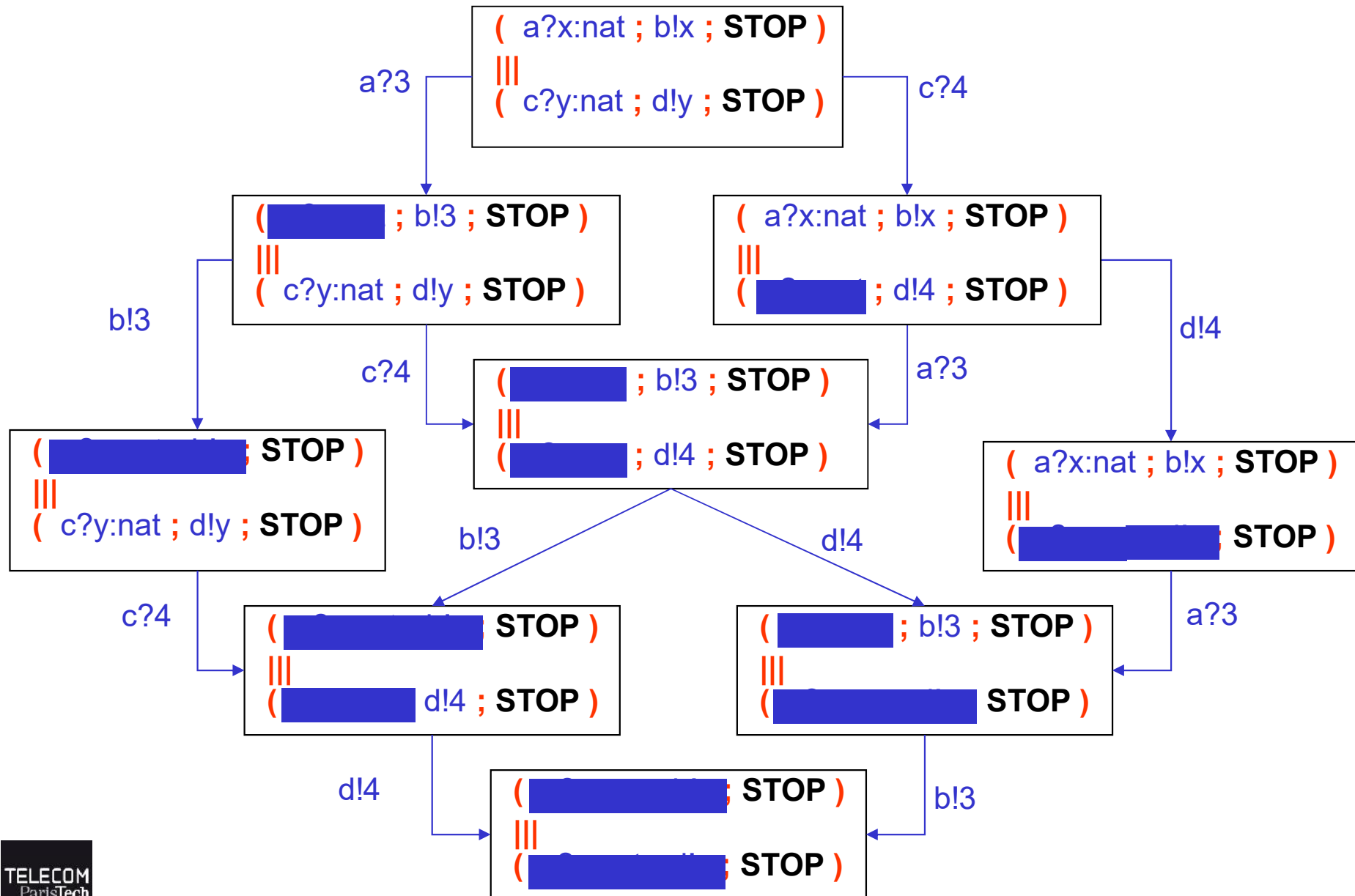
CCS+ par les exemples : la mise en parallèle

PROCESS buffer_Duplex_Une_Fois [a, c, b, d] :=


ENDPROC

CCS+ par les exemples : la mise en parallèle

-



CCS+ par les exemples : l'instantiation — —

PROCESS Buffer_Duplex_Une_Fois [a, b, c, d] :=

Buffer_Une_Fois [a, b]

|||

Buffer_Une_Fois [c, d]

WHERE

PROCESS Buffer_Une_Fois [e , f] :=

e?x:nat ; f!x ; **STOP**

ENDPROC

ENDPROC

CCS+ par les exemples : l'instantiation — —

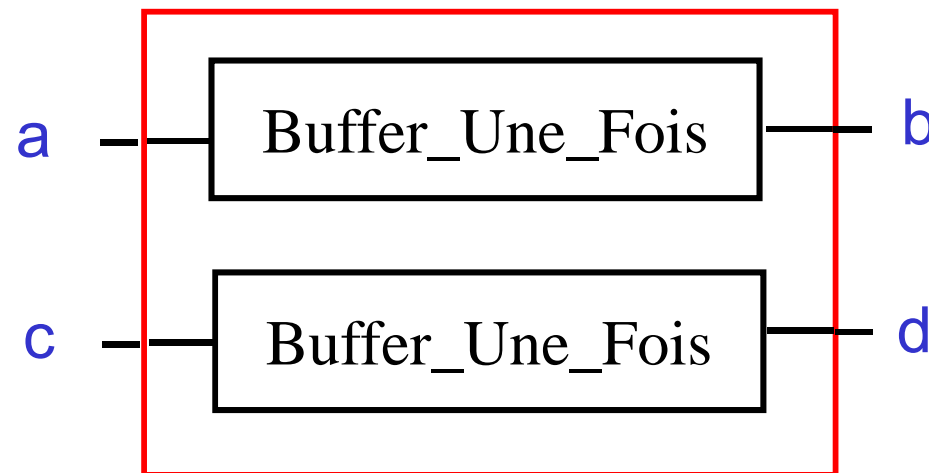
PROCESS Buffer_Duplex_Une_Fois [a, b, c, d] :=

Buffer_Une_Fois [a, b]

|||

Buffer_Une_Fois [c, d]

ENDPROC

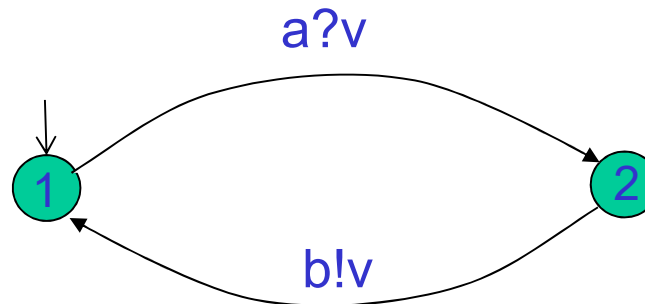
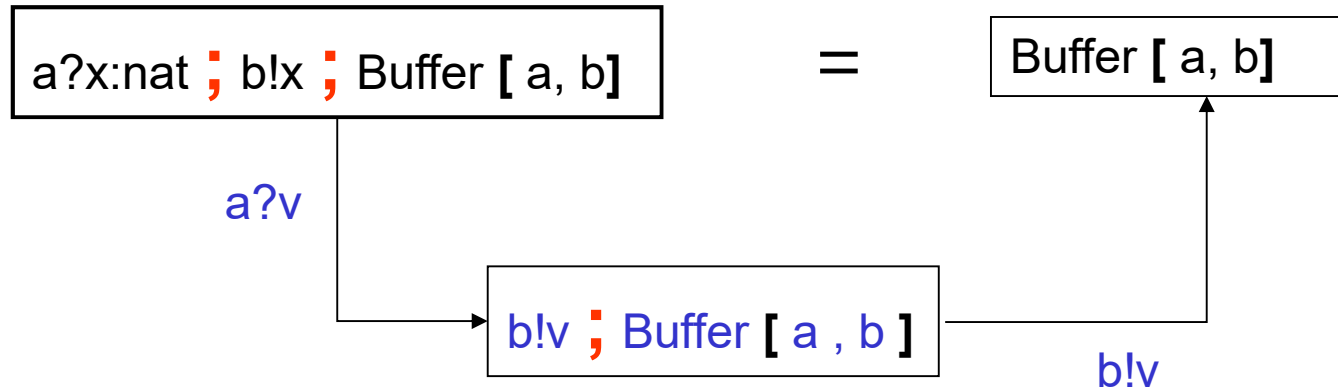


— CCS+ par les exemples : la récursion —

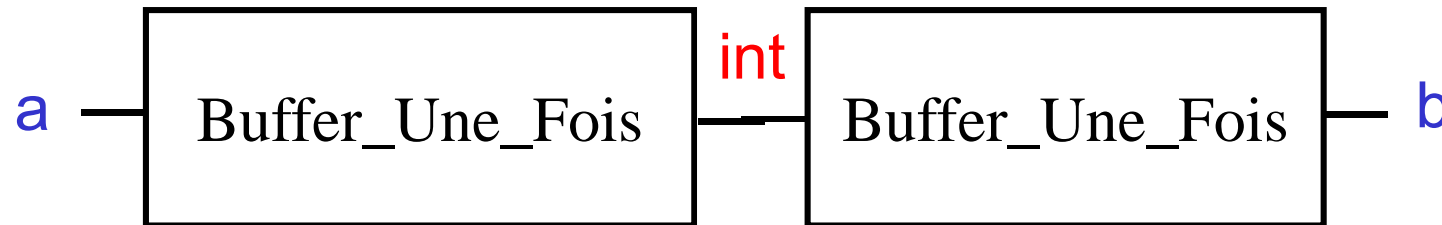
PROCESS Buffer [a , b] :=

$a?x:\text{nat} ; b!x ; \text{Buffer [a , b]}$

ENDPROC



CCS+ par les exemples : la synchronisation

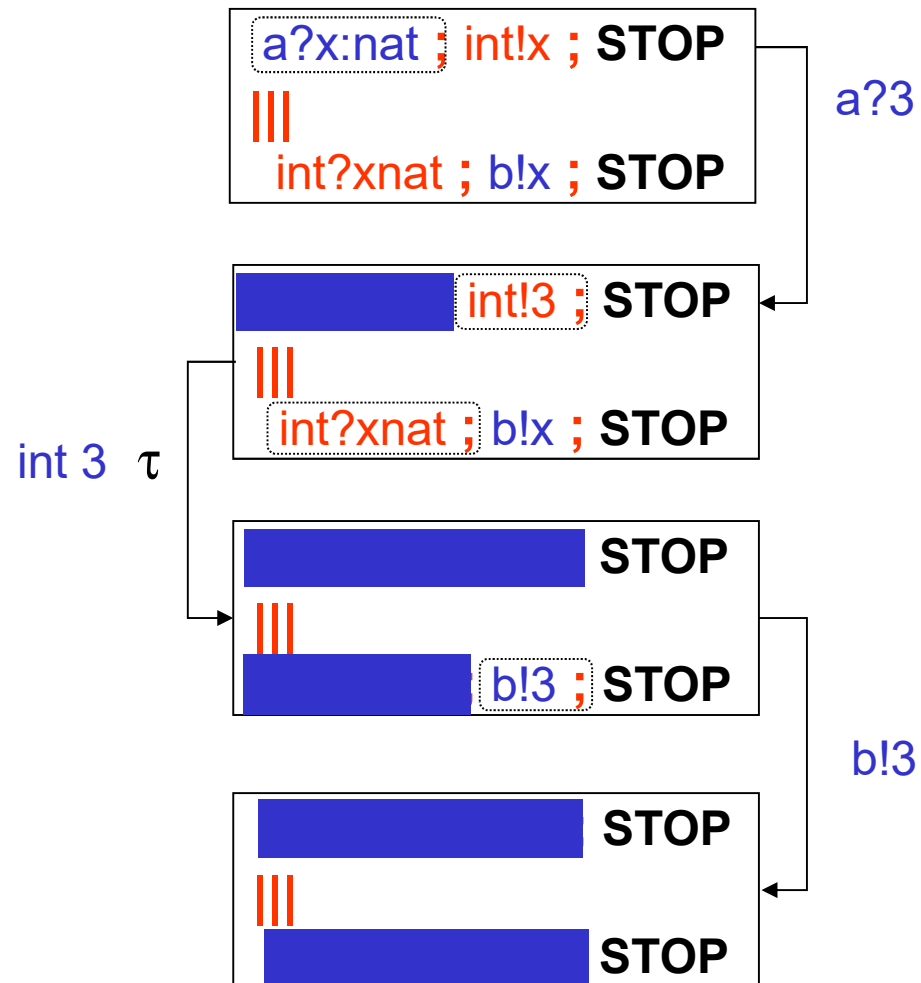


```
PROCESS Buffer_Une_Fois [ e, f ] := e?x:nat ; f!x ; STOP ENDPROC
```

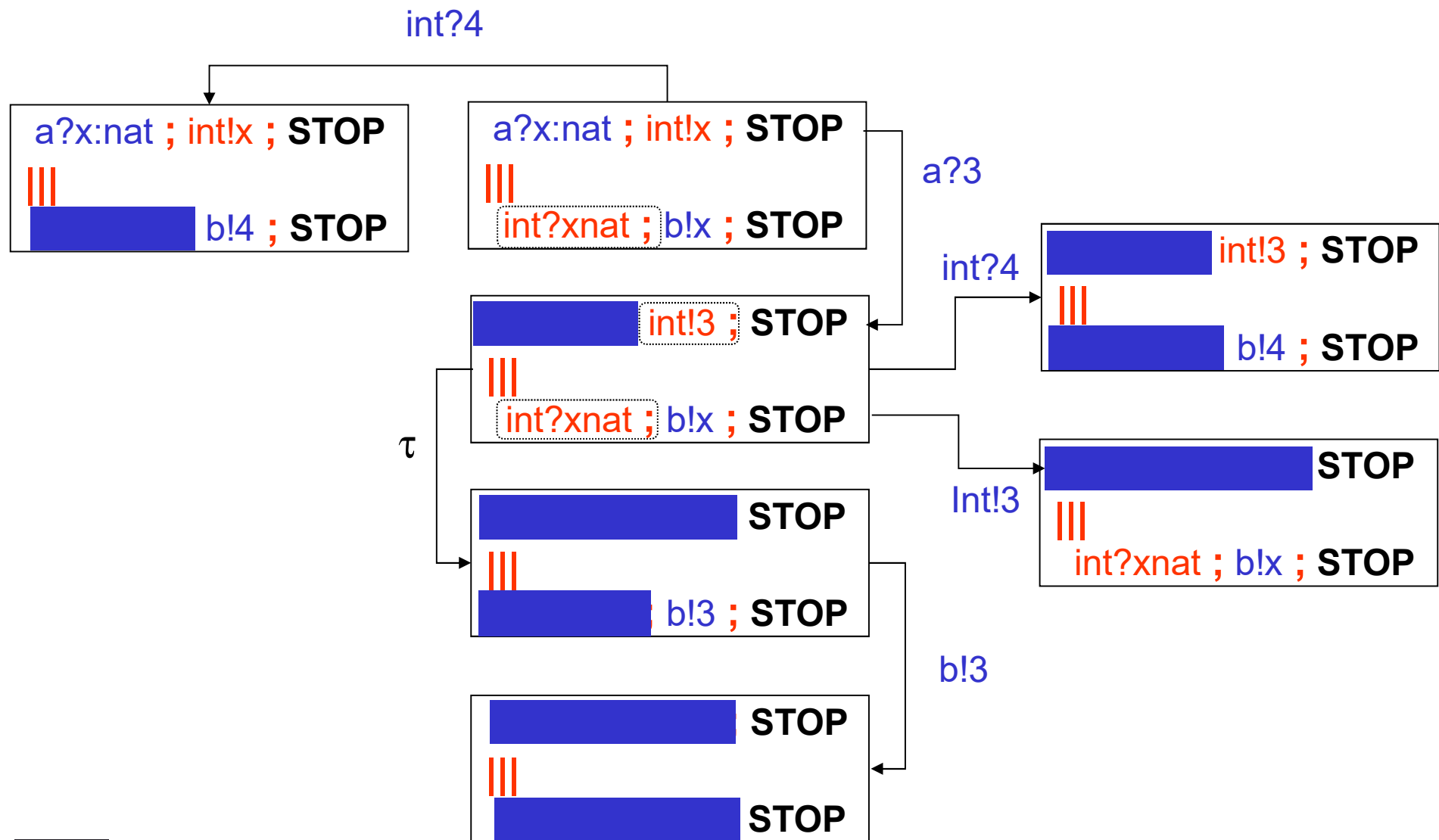
```
PROCESS Buffer_2_Places [ a, int, b ] :=  
Buffer_Une_Fois [ a, int ]  
|||  
Buffer_Une_Fois [ int, b ]  
  
ENDPROC
```

```
PROCESS Buffer_2_Places [ a, int, b ] :=  
a?x:nat ; int!x ; STOP  
|||  
int?x:nat ; b!x ; STOP  
  
ENDPROC
```


CCS+ par les exemples : la synchronisation



CCS+ par les exemples : la synchronisation



CCS+ par les exemples : la synchronisation

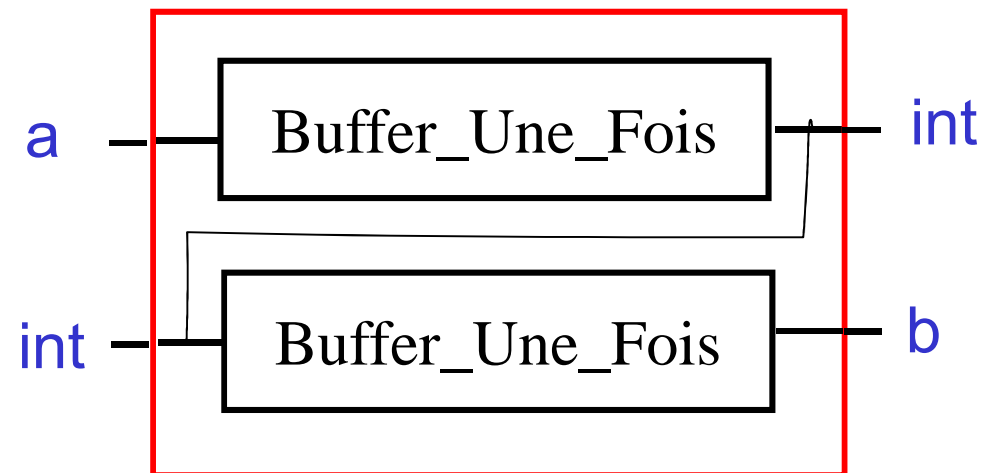
PROCESS Buffer_2_Places [*a*, int, *b*] :=

Buffer_Une_Fois [*a*, int]

|||

Buffer_Une_Fois [int, *b*]

ENDPROC



CCS+ par les exemples : la restriction — —

PROCESS Buffer_2_Places_bis [*a*, *b*] :=

(

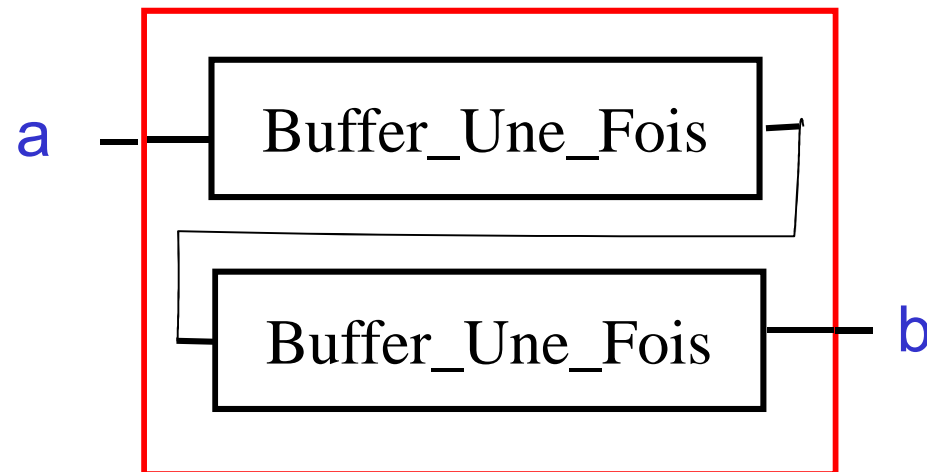
Buffer_Une_Fois [*a*, int]

|||

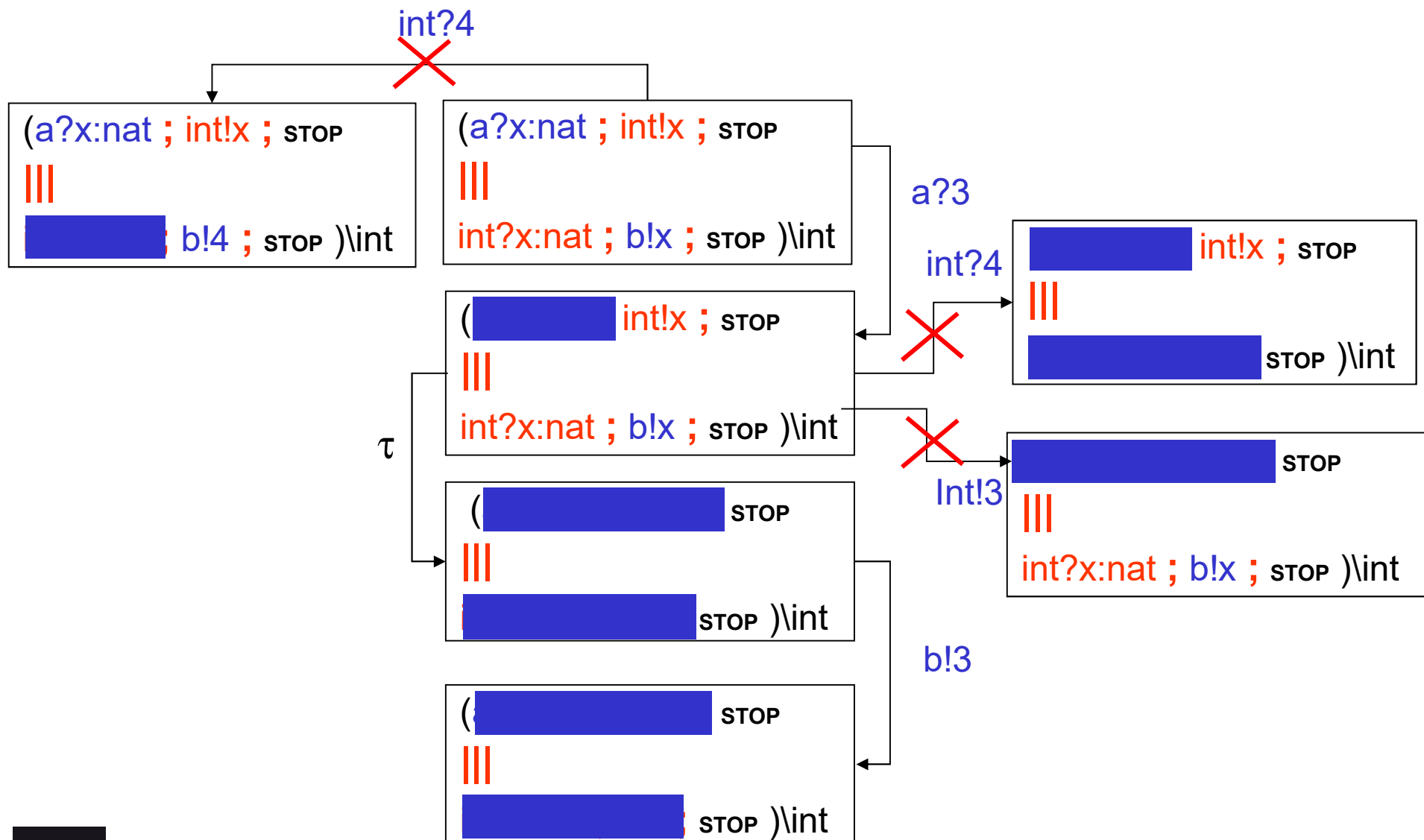
Buffer_Une_Fois [int, *b*]

) \ *int*

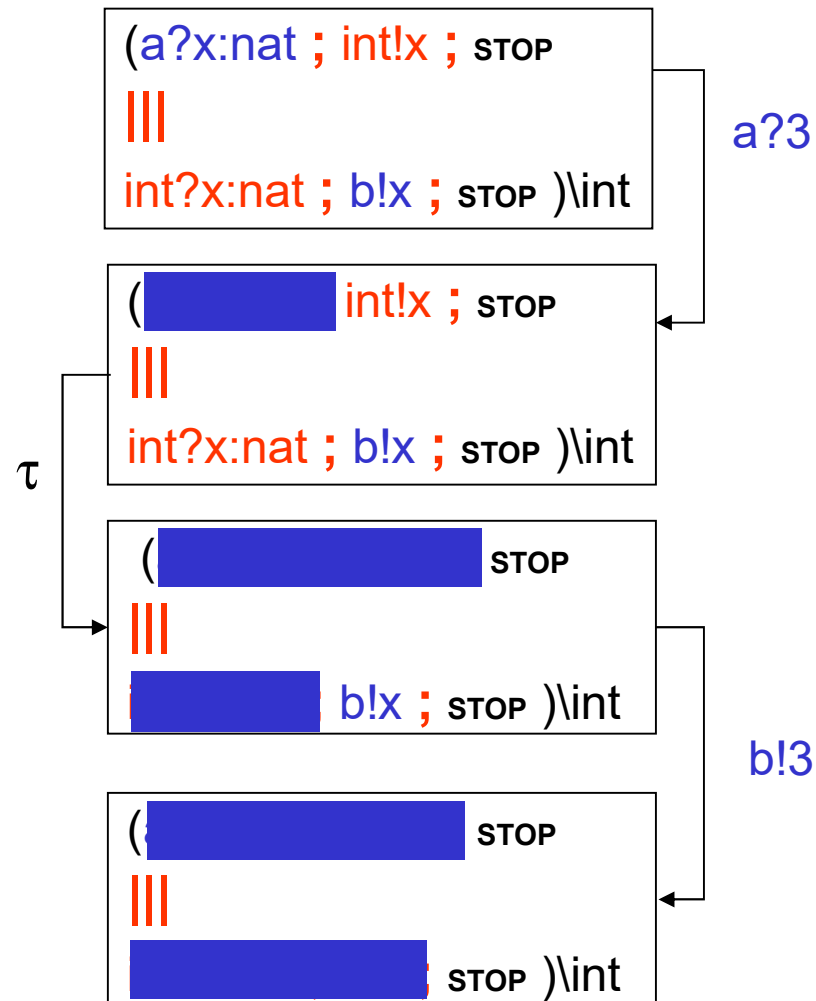
ENDPROC



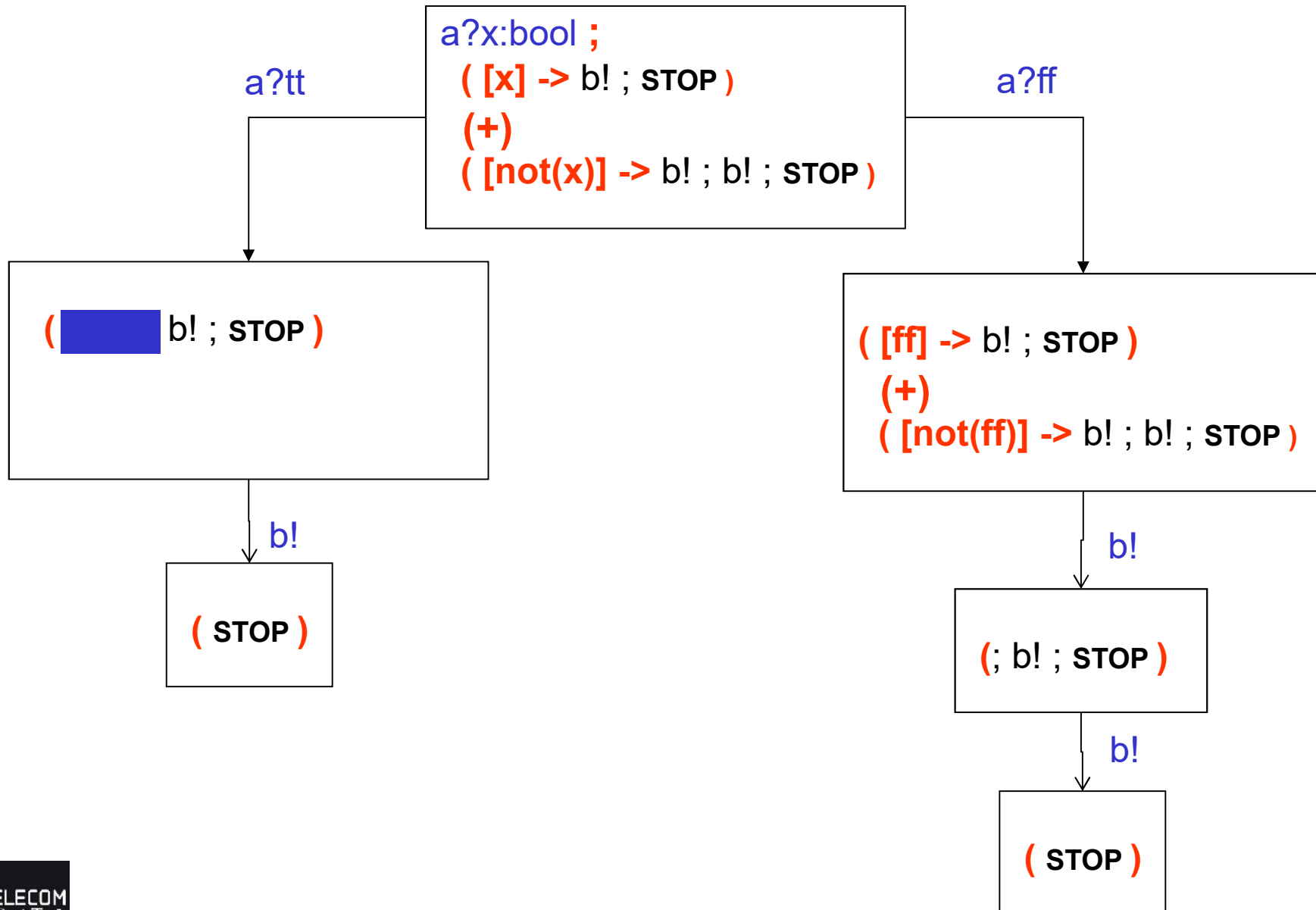
CCS+ par les exemples : la synchronisation



CCS+ par les exemples : la synchronisation



— CCS+ par les exemples : la garde —



CCS+ par les exemples : la préemption — —

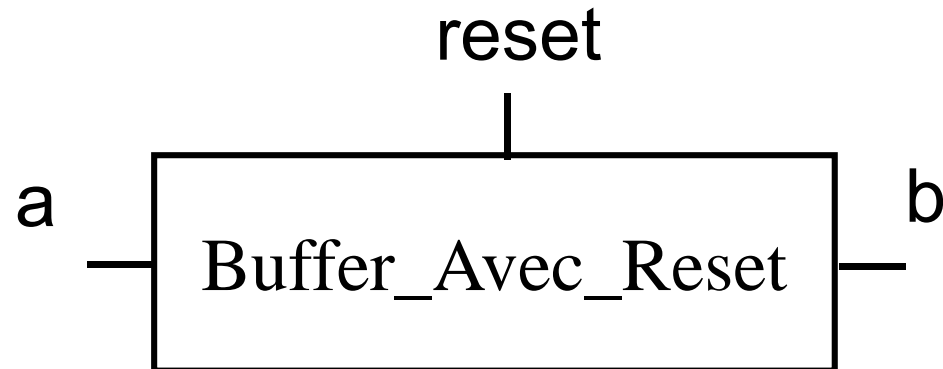
PROCESS Buffer_Avec_Reset [a, b, reset] :=

(a? ; b! ; **STOP**)

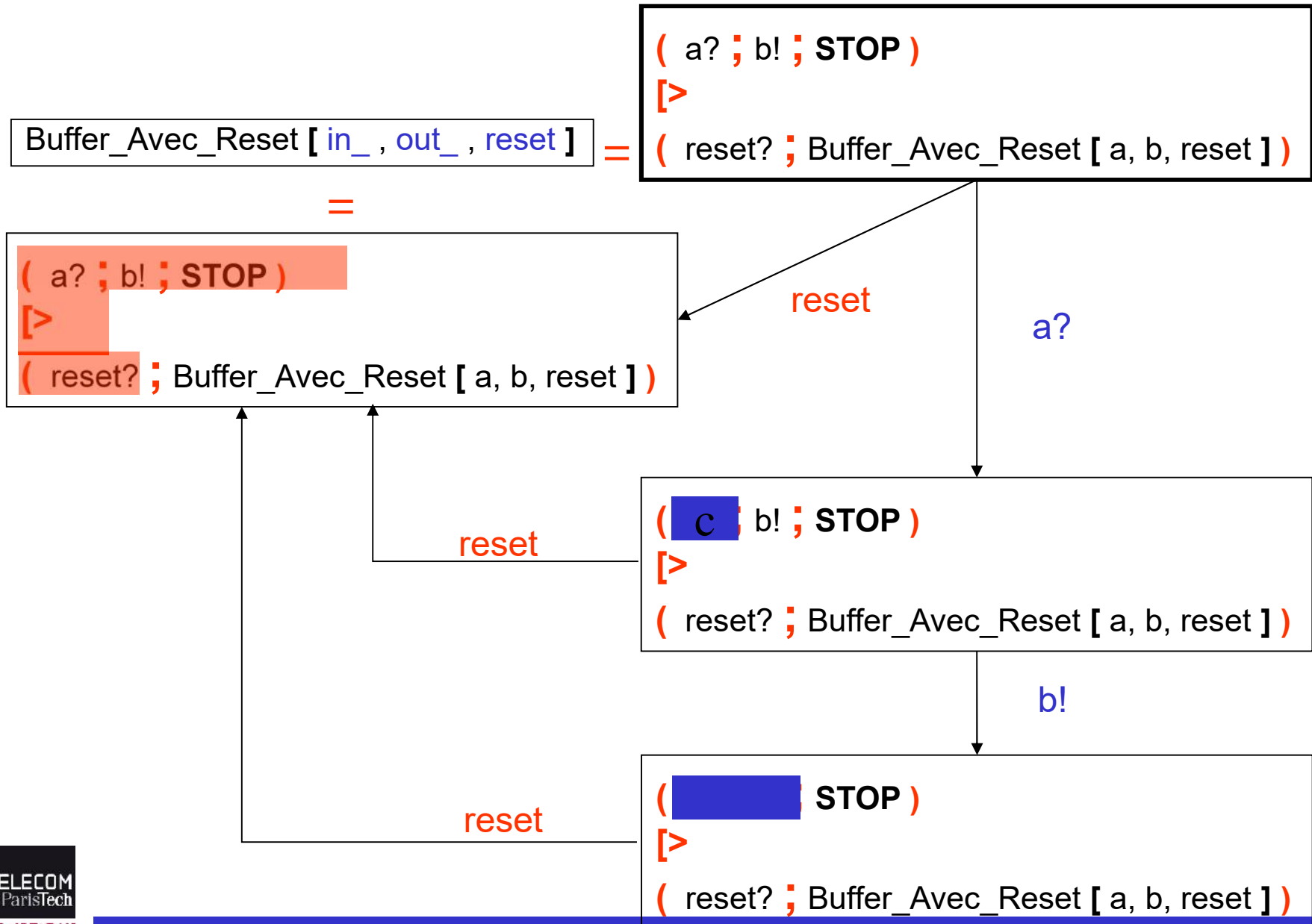
[>

(reset? ; Buffer_Avec_Reset [a, b, reset])

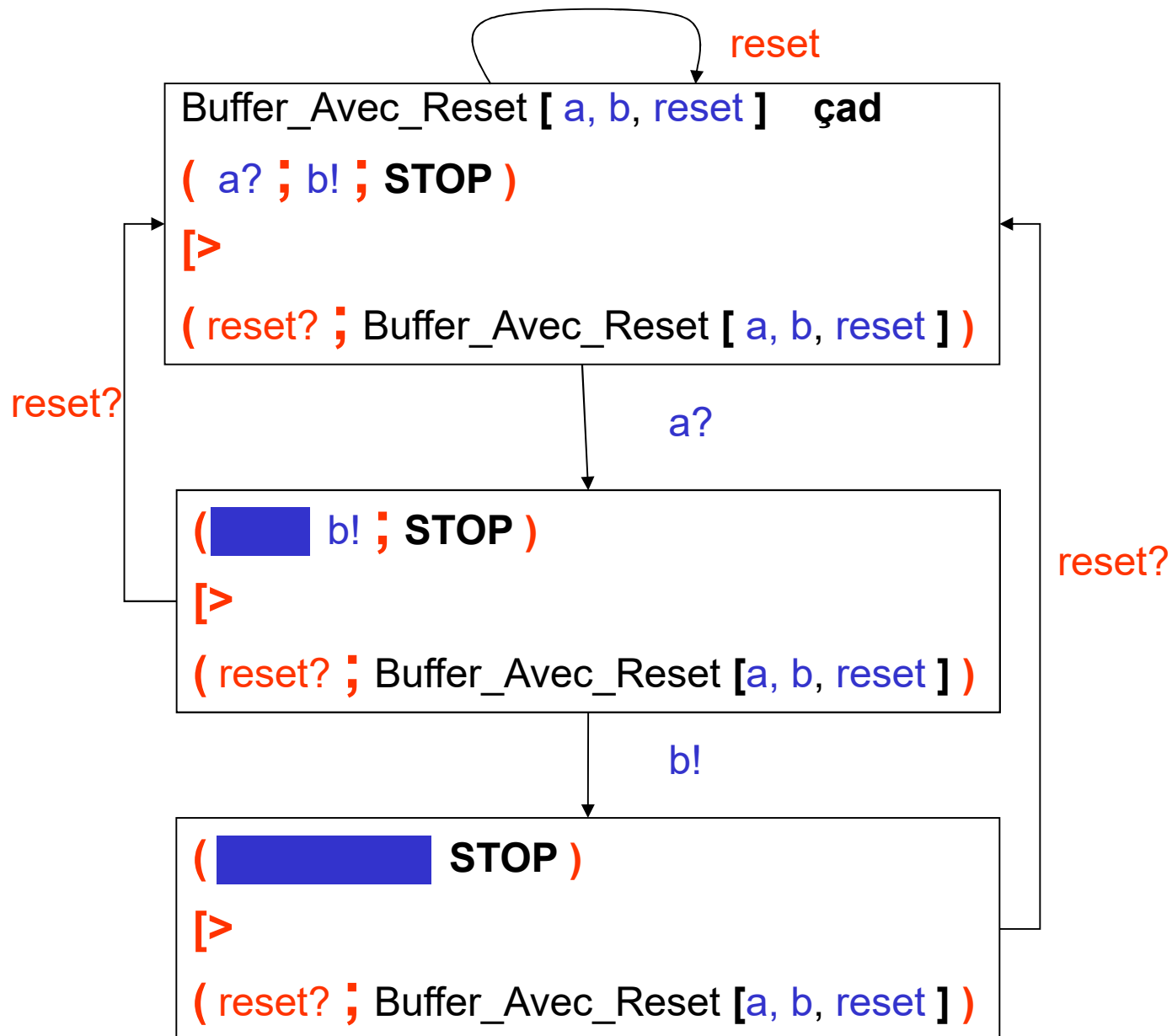
ENDPROC



CCS+ par les exemples : la préemption — —



CCS+ par les exemples : la préemption — —



CCS+ par les exemples : la préemption — —

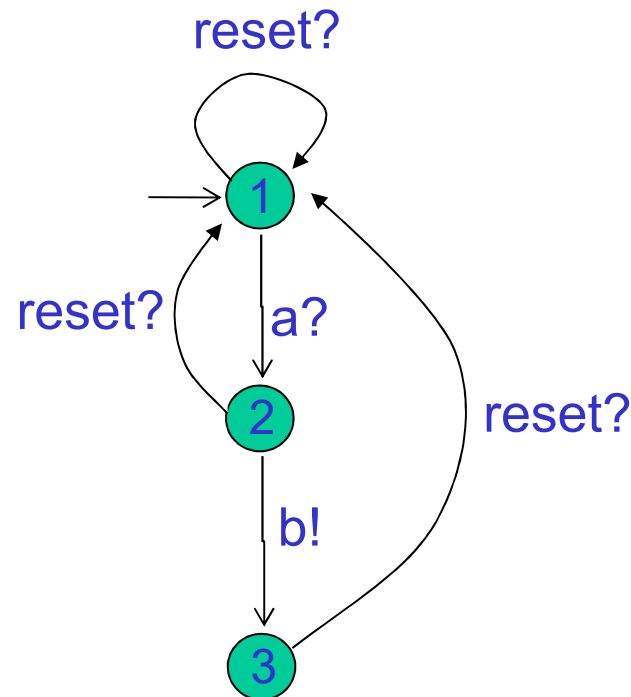
PROCESS Buffer_Avec_Reset [a, b, reset] :=

(a? ; b! ; **STOP**)

[>

(reset? ; Buffer_Avec_Reset [a, b, reset])

ENDPROC



Syntaxe des expressions de comportement CCS+

PP : ensemble des ports : $\{a, a', a_i, b, \dots\}$

Var : ensemble des variables :
 $\{x, x', x_i, y, \dots\}$

Val : ensemble des valeurs : $\{v, v', v_i, \dots\}$

Typ: ensemble des types : $\{T, T', T_i, \dots\}$

Exp : ensemble des expressions valuées :
 $\{E, E', E_i, F, \dots\}$

NP : ensemble des noms de process
 $\{P, P', P_i, \dots\}$

EdC : Expressions de Comportement CCS+
 $\{C, C', C_i, \dots\}$

$C ::= \text{STOP}$

| $a?x:T ; C$

| $b!E ; C$

| $\tau ; C$

| $C_1 (+) C_2$

| $C_1 ||| C_2$

| $C \setminus a_1, \dots, a_n$

| $C_1 [> C_2$

| $[E] \rightarrow C$

| $P[a_1, \dots, a_n](E_1, \dots, E_k)$

— Système de Transitions Etiqueté —

Un Système de transitions étiqueté S est un quadruplet : $S = \langle q_0, Q, A_\tau, T \rangle$ avec :

- Q un ensemble d'états : $q, q', q_0, q_1, \dots, p, p', p_0, p_1, \dots$
- q_0 un état particulier, l'état initial
- A_τ un ensemble d'actions.
Les symboles $\alpha, \alpha', \alpha_0, \alpha_1, \dots, \beta, \beta', \beta_0, \beta_1, \dots$ représentent des éléments de A_τ
- τ est une action particulière de A_τ : l'action interne
- $A = A_\tau - \{\tau\}$. A est l'ensemble des actions observables : $e, e', e_0, e_1, \dots, f, f', f_0, f_1, \dots$
- T un ensemble de transitions. $T \subseteq Q \times A_\tau \times Q$
- Le fait $(p, \alpha, p') \in T$ peut aussi s'écrire $p \xrightarrow{\alpha} p'$

Sémantique Opérationnelle Structurale de CCS+

- Sémantique donnée par traduction des termes CCS+ en Systèmes de Transitions étiquetés
- Traduction définie par un ensemble de règles dites de Sémantique Opérationnelle Structurale (SOS).
- Ces règles ont le format : (*Cond* / *Conc*)
- A chaque opérateur est associé un ensemble de règles qui définissent son comportement

une règle de l'opérateur *op* $\frac{\text{Conditions sur } C_1 \text{ et/ou } C_2}{\text{Une transition de } C_1 \text{ } op \text{ } C_2}$

- Une règle définit dans sa partie *Cond* des conditions sur C_1 et/ou C_2 permettant de déduire qu'une transition de $C_1 \text{ } op \text{ } C_2$, présentée dans la partie *Conc*, est tirable.

Systeme de Transitions Etiquet  associ    CCS+ -

Les Systeme de transitions  tiquet  associ s aux process CCS+ sont des quadruplets :

$$S = \langle q_0, Q, A_{CCS+}, T \rangle \text{ o  :}$$

- Q l'ensemble des Expressions de Comportement de CCS+ : $\{C, C', C_i, \dots\}$
- q_0 un terme particulier, qui correspond au terme initial.
- A_{CCS+} l'ensemble des actions de CCS+. Une action CCS+ a l'une des 5 formes suivantes (o  a est un port et v est une valeur) : $\tau, a?, a!, a?v, a!v$
Les symboles $\alpha, \alpha', \alpha_0, \alpha_1, \dots, \beta, \beta', \beta_0, \beta_1, \dots$ seront utilis s pour d noter des actions
- τ est une action particuli re de A_{CCS+} : l'action interne
- $A = A_{CCS+} - \{\tau\}$. A est l'ensemble des actions observables : $e, e', e_0, e_1, \dots, f, \dots$
- T un ensemble de transitions. $T \subseteq CCS+ \times A_{CCS+} \times CCS+$

Règles SOS de CCS+

STOP : aucune règle car **STOP** ne possède aucune transition

$$\tau ; C \quad \frac{-}{\tau ; C \xrightarrow{\tau} C}$$

$$a? ; C \quad \frac{-}{a? ; C \xrightarrow{a?} C}$$

$$a! ; C \quad \frac{-}{a! ; C \xrightarrow{a!} C}$$

$$a? x:T; C \quad \frac{v \in \text{Dom}(T)}{a? x:T ; C \xrightarrow{a?v} C[v/x]}$$

$$a!E; C \quad \frac{v = \text{Val}(E)}{a!E ; C \xrightarrow{a!v} C}$$

Règles SOS de CCS+

—

$C_1 (+) C_2$

$$\frac{C_1 \xrightarrow{\alpha} C'}{C_1 (+) C_2 \xrightarrow{\alpha} C'}$$

$$\frac{C_2 \xrightarrow{\alpha} C'}{C_1 (+) C_2 \xrightarrow{\alpha} C'}$$

$C_1 ||| C_2$

$$\frac{C_1 \xrightarrow{\alpha} C'}{C_1 ||| C_2 \xrightarrow{\alpha} C' ||| C_2}$$

$$\frac{C_2 \xrightarrow{\alpha} C'}{C_1 ||| C_2 \xrightarrow{\alpha} C_1 ||| C'}$$

$$\frac{C_1 \xrightarrow{\alpha_1} C_1' \quad C_2 \xrightarrow{\alpha_2} C_2' \quad \alpha_2 \neq \tau, \quad \overline{\alpha_2} = \alpha_1}{C_1 ||| C_2 \xrightarrow{\tau} C_1' ||| C_2'}$$

où, la fonction $\overline{\cdot}$ est définie par : $\overline{a?} = a!$, $\overline{a?v} = a!v$, $\overline{\overline{\alpha}} = \alpha$

Règles SOS de CCS+

$C \setminus a_1, \dots, a_n$

$$\frac{C \xrightarrow{\alpha} C' \quad \alpha = \tau \text{ ou } \text{port}(\alpha) \notin \{a_1, \dots, a_n\}}{C \setminus a_1, \dots, a_n \xrightarrow{\alpha} C' \setminus a_1, \dots, a_n}$$

$C_1 [> C_2$

$$\frac{C_1 \xrightarrow{\alpha} C'}{C_1 [> C_2 \xrightarrow{\alpha} C' [> C_2} \quad \frac{C_2 \xrightarrow{\alpha} C'}{C_1 [> C_2 \xrightarrow{\alpha} C'}$$

$[E] \rightarrow C$

$$\frac{C \xrightarrow{\alpha} C' \quad \text{Val}(E) = \text{true}}{[E] \rightarrow C \xrightarrow{\alpha} C'}$$

Règles SOS de CCS+

$P[a_1, \dots, a_n](E_1, \dots, E_k)$

$$\frac{C[a_1/b_1, \dots, a_n/b_n, E_1/x_1, \dots, E_k/x_k] \xrightarrow{\alpha} C'}{P[a_1, \dots, a_n](E_1, \dots, E_k) \xrightarrow{\alpha} C'}$$

Process

$P[b_1, \dots, b_n](x_1:T_1, \dots, x_k:T_k) := C$

Endproc