Vérification formelle en Coq de la chaîne des sémantiques pour la compilation d'Esterel

Lionel Rieg

Yale University

Collège de France
28 Mars 2018

Esterel parmi les langages synchrones

Esterel

- Langage synchrone flot de donnée
  - Synchrone : le temps est divisé en instants
    $\rightsquigarrow$ le calcul dans un instant ne prend pas de temps
  - Flot de données : déplacement des données
- Orienté contrôle, avec une saveur impérative

  au contraire de Lustre
- Exemples filés dans la suite
  - Sémantique : règle if-then
  - Programme : ABRO

Objectif : prouver le schéma de compilation d'Esterel

- Vérifier formellement la compilation vers les circuits
  - Basé sur le livre web de Gérard Berry
    [The Constructive Semantics of Pure Esterel] cf. page du cours
  - Compilation modulaire
  - Même principe que CompCert
    $\rightsquigarrow$ la sémantique est ~~raffinée~~ préservée par la compilation
- Coq
  - Assistant de preuves interactif
  - preuve = programme — Curry-Howard
  - 4 couleurs, Feit-Thomson, CompCert
- Restrictions
  - Compilation vers des circuits
  - Pas de données : Esterel Pur v.5
  - Pas de réincarnation — pour plus tard . . .

Syntaxe d'Esterel (instructions)

$p, q :=$

`nothing`	
`pause`	
`emit` s	
`exit` T^k	k est l'indice
`trap` T `in` p `end`	
`if` s `then` p `else` q `end`	$s\,?\,p\,,\,q$
`suspend` p `when` s	
p `;` q	
p `||` q	
`loop` p `end`	
`signal` s `in` p `end`	$p \backslash s$

\+ constructions dérivées (macros)

Quelques macros en Esterel

```
halt :=
  loop
    pause
  end
```

```
awimm s :=    await immediate
  trap T in
    loop
      if s then
        exit T²
      else
        pause
      end
    end
  end
```

```
abort p when s :=
  trap T in
      pause ;
      loop
        if s then
          exit T²
        else
          pause
        end
      end
    ||
      suspend p when s ;
      exit T²
  end
```

“Hello world !” en Esterel : ABRO

Spécification :

- Dès que les signaux A et B sont reçus, on émet O
- Remise à zéro lorsque R est reçu

```
halt              := loop pause end
awimm s           := trap T in
                         loop if s then exit T2 else pause end end
abort p when s    := trap T in
                         loop (if s then exit T2 else pause end) end
                       ||
                         (p ; exit T2)
```

“Hello world !” en Esterel : ABRO

Spécification :

- Dès que les signaux *A* et *B* sont reçus, on émet *O*
- Remise à zéro lorsque *R* est reçu

```
(awimm A || awimm B);
emit O;
halt
```

```
halt              := loop pause end
awimm s           := trap T in
                         loop if s then exit T2 else pause end end
abort p when s    := trap T in
                         loop (if s then exit T2 else pause end) end
                       ||
                         (p ; exit T2)
```

“Hello world !” en Esterel : ABRO

Spécification :

- Dès que les signaux *A* et *B* sont reçus, on émet *O*
- Remise à zéro lorsque *R* est reçu

```
abort
    (awimm A || awimm B);
    emit O;
    halt
when R
```

```
halt              := loop pause end
awimm s           := trap T in
                         loop if s then exit T2 else pause end end
abort p when s    := trap T in
                         loop (if s then exit T2 else pause end) end
                       ||
                         (p ; exit T2)
```

“Hello world !” en Esterel : ABRO

Spécification :

- Dès que les signaux *A* et *B* sont reçus, on émet *O*
- Remise à zéro lorsque *R* est reçu

```
loop
    abort
        (awimm A || awimm B);
        emit O;
        halt
    when R
end
```

```
halt            := loop pause end
awimm s         := trap T in
                       loop if s then exit T^2 else pause end end
abort p when s  := trap T in
                       loop (if s then exit T^2 else pause end) end
                   ||
                       (p ; exit T^2)
```

Quel type de sémantiques ?

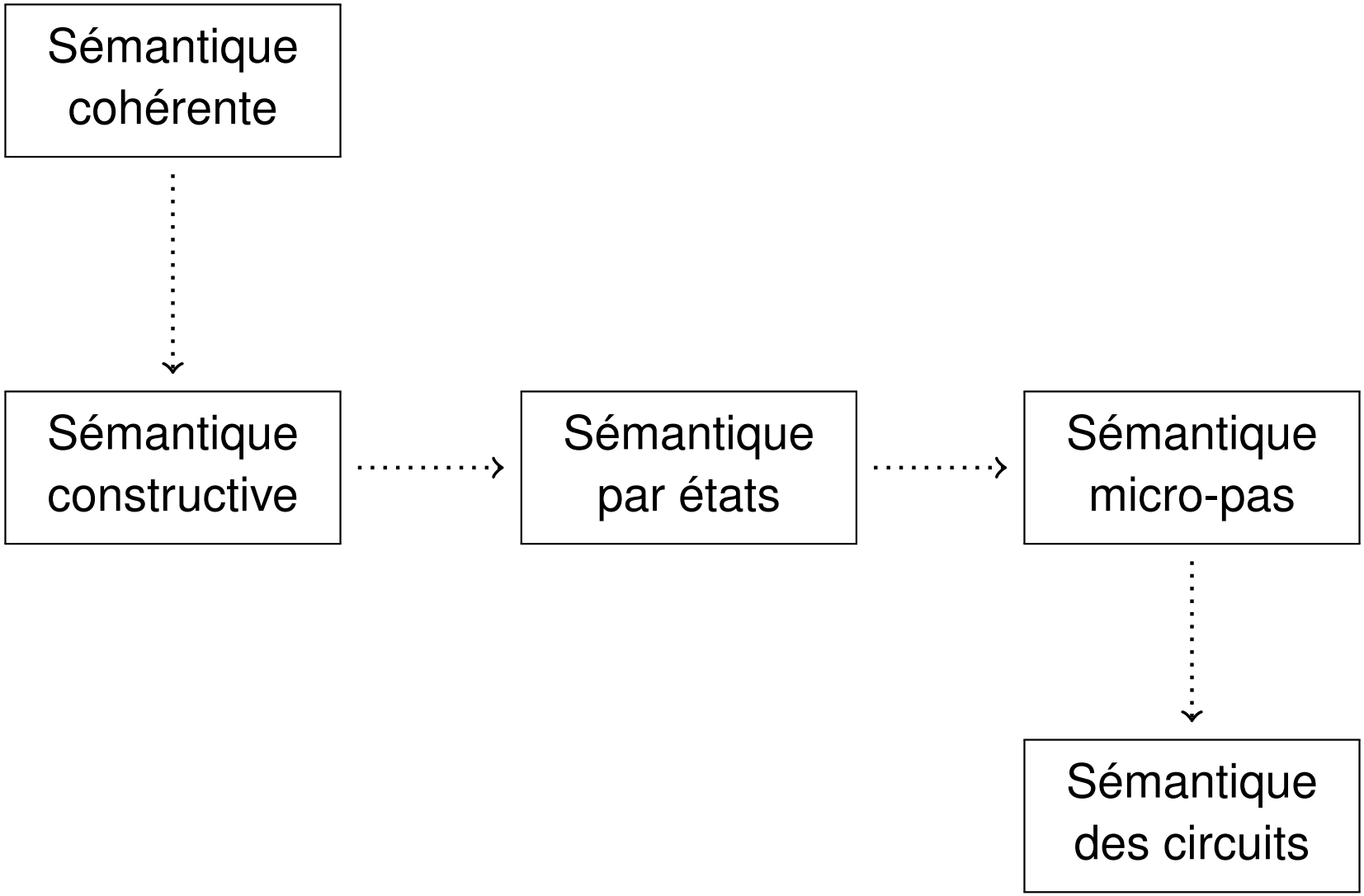
Sémantique Opérationnelle Structurelle (SOS)

- Définition mathématique par réécriture
  - Opérationnelle : une transition = un instant
  - Structurelle : selon la structure du programme
- Forme des règles : $p \xrightarrow[E]{E', k} p'$
  - Entrées E
  - Sorties E'
  - Un code de retour k 0 = fini, 1 = en pause, 2+ = exceptions

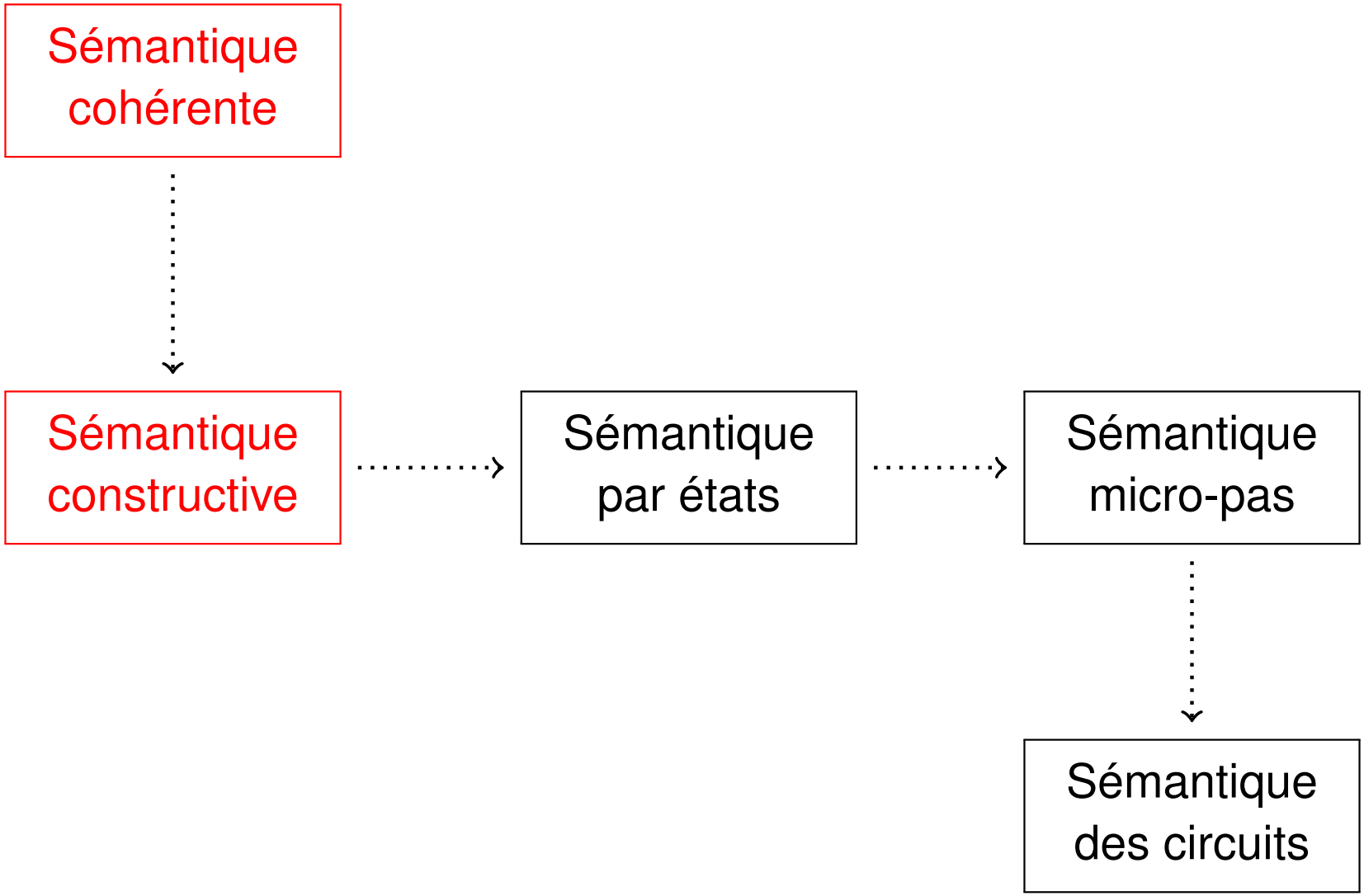
Quelques remarques :

- E et E' associe à chaque signal son statut :
  présent ($+$) absent ($-$) inconnu ($\perp$)

Chaîne des sémantique d'Esterel



Chaîne des sémantique d'Esterel



Sémantique cohérente

Loi de cohérence : un signal est présent ssi il est émis

$$\text{if-then}\ \frac{s^+ \in E \qquad p \xrightarrow[E]{E',k} p'}{s\,?\,p\,,\,q \xrightarrow[E]{E',k} p'} \qquad \frac{s^- \in E \qquad q \xrightarrow[E]{E',k} q'}{s\,?\,p\,,\,q \xrightarrow[E]{E',k} q'}\ \text{if-else}$$

$$\text{sig}+\ \frac{p \xrightarrow[E\cup\{s^+\}]{E',k} p' \qquad s^+ \in E'}{p\backslash s \xrightarrow[E]{E'\backslash\{s\},k} p\backslash s'} \qquad \frac{p \xrightarrow[E\cup\{s^-\}]{E',k} p' \qquad s^- \in E'}{p\backslash s \xrightarrow[E]{E'\backslash\{s\},k} p\backslash s'}\ \text{sig}-$$

- Que peut-on en dire ?
  - Non-déterministe
  - Mal justifiée

corrigé par Tardieu

Sémantique cohérente

Loi de cohérence : un signal est présent ssi il est émis

$$\text{if-then}\ \frac{s^+ \in E \qquad p \xrightarrow[E]{E',k} p'}{s\,?\,p\,,\,q \xrightarrow[E]{E',k} p'} \qquad \frac{s^- \in E \qquad q \xrightarrow[E]{E',k} q'}{s\,?\,p\,,\,q \xrightarrow[E]{E',k} q'}\ \text{if-else}$$

$$\text{sig}+\ \frac{p \xrightarrow[E\cup\{s^+\}]{E',k} p' \qquad s^+ \in E'}{p\backslash s \xrightarrow[E]{E'\backslash\{s\},k} p\backslash s'} \qquad \frac{p \xrightarrow[E\cup\{s^-\}]{E',k} p' \qquad s^- \in E'}{p\backslash s \xrightarrow[E]{E'\backslash\{s\},k} p\backslash s'}\ \text{sig}-$$

- Que peut-on en dire ?
  - Non-déterministe
  - Mal justifiée

corrigé par Tardieu

Sémantique constructive

- Élimine les problèmes de causalité & le non-determinisme

  interdit “`if s then emit s else nothing end`”

- Se rapproche des circuits : [Berry, Mendler, Shiple]

  propagation électrique = logique intuitionniste

Sémantique constructive

- Élimine les problèmes de causalité & le non-determinisme

  interdit “`if s then emit s else nothing end`”

- Se rapproche des circuits : [Berry, Mendler, Shiple]

  propagation électrique = logique intuitionniste

- Can/Must pour les signaux locaux
  - sig+ si s doit être émis
  - sig– si s ne peut pas être émis

$$\text{sig+}\ \frac{s \in \mathit{Must}_s(p, E \cup \{s^\perp\}) \qquad p \xhookrightarrow[E \cup \{s^+\}]{E', k} p'}{p \backslash s \xhookrightarrow[E]{E' \backslash \{s\}, k} p \backslash s'}$$

$$\text{sig-}\ \frac{s \notin \mathit{Can}^+_s(p, E \cup \{s^\perp\}) \qquad p \xhookrightarrow[E \cup \{s^-\}]{E', k} p'}{p \backslash s \xhookrightarrow[E]{E' \backslash \{s\}, k} p \backslash s'}$$

L'exécution d'ABRO

```
loop
    abort
        (awimm A || awimm B);
        emit O;
        halt
    when R
end
```

L'exécution d'ABRO

```
loop
    abort
        (awimm A || awimm B);
        emit O;
        halt
    when R
end
```

$\{B\}$

L'exécution d'ABRO

```
abort
    (awimm A || awimm B);
    emit O;
    halt
when R;
loop
    abort
        (awimm A || awimm B);
        emit O;
        halt
    when R
end
```

$\{B\}$

L'exécution d'ABRO

```
abort
    (awimm A || nothing);
    emit O;
    halt
when R;
loop
    abort
        (awimm A || awimm B);
        emit O;
        halt
    when R
end
```

$\{B\}$

L'exécution d'ABRO

```
abort
    (awimm A || nothing);
    emit O;
    halt
when R;
loop
    abort
        (awimm A || awimm B);
        emit O;
        halt
    when R
end
```

$\{B\} \Longrightarrow \{A,$

L'exécution d'ABRO

```
abort
    (nothing || nothing);
    emit O;
    halt
when R;
loop
    abort
        (awimm A || awimm B);
        emit O;
        halt
    when R
end
```

$\{B\} \Longrightarrow \{A,$

L'exécution d'ABRO

```
abort
    emit O;
    halt
when R;
loop
    abort
        (awimm A || awimm B);
        emit O;
        halt
    when R
end
```

$\{B\} \Longrightarrow \{A,$

L'exécution d'ABRO

```
abort
    halt
when R;
loop
    abort
        (awimm A || awimm B);
        emit O;
        halt
    when R
end
```

$\{B\} \Longrightarrow \{A, O\}$

L'exécution d'ABRO

```
abort
    halt
when R;
loop
    abort
        (awimm A || awimm B);
        emit O;
        halt
    when R
end
```

$\{B\} \Longrightarrow \{A, O\} \Longrightarrow \{B\}$

L'exécution d'ABRO

```
abort
    halt
when R;
loop
    abort
        (awimm A || awimm B);
        emit O;
        halt
    when R
end
```

$\{B\} \Longrightarrow \{A, O\} \Longrightarrow \{B\} \Longrightarrow \{R\}$

L'exécution d'ABRO

```
loop
    abort
        (awimm A || awimm B);
        emit O;
        halt
    when R
end
```

$\{B\} \Longrightarrow \{A, O\} \Longrightarrow \{B\} \Longrightarrow \{R\}$

L'exécution d'ABRO

```
abort
    (awimm A || awimm B);
    emit O;
    halt
when R;
loop
    abort
        (awimm A || awimm B);
        emit O;
        halt
    when R
end
```

$$\{B\} \Longrightarrow \{A, O\} \Longrightarrow \{B\} \Longrightarrow \{R\}$$

L'exécution d'ABRO

```
abort
    (awimm A || awimm B);
    emit O;
    halt
when R;
loop
    abort
        (awimm A || awimm B);
        emit O;
        halt
    when R
end
```

$$\{B\} \Longrightarrow \{A, O\} \Longrightarrow \{B\} \Longrightarrow \{R\} \Longrightarrow \{A, B,$$

L'exécution d'ABRO

```
abort
    halt
when R;
loop
    abort
        (awimm A || awimm B);
        emit O;
        halt
    when R
end
```

$$\{B\} \Longrightarrow \{A, O\} \Longrightarrow \{B\} \Longrightarrow \{R\} \Longrightarrow \{A, B, O\}$$

Que peut-on dire de la sémantique constructive ?

- Préserve les signaux déclarés

- Si E est total (pas de $\perp$), E' l'est aussi
- Si le code de retour n'est pas 1, le résidu est `nothing`

- Déterministe

- Proverbe Esterel : « If must, do. If no can, no do. »

Que peut-on dire de la sémantique constructive ?

- Préserve les signaux déclarés

$$p \xhookrightarrow[E]{E', k} p' \implies \mathrm{dom}(E) = \mathrm{dom}(E')$$

- Si E est total (pas de $\bot$), E' l'est aussi
- Si le code de retour n'est pas 1, le résidu est `nothing`

- Déterministe

- Proverbe Esterel : « If must, do. If no can, no do. »

Que peut-on dire de la sémantique constructive ?

- Préserve les signaux déclarés

$$p \xhookrightarrow[E]{E',k} p' \implies \mathrm{dom}(E) = \mathrm{dom}(E')$$

- Si E est total (pas de $\bot$), E' l'est aussi
- Si le code de retour n'est pas 1, le résidu est `nothing`

$$p \xhookrightarrow[E]{E',k} p' \implies k \neq 1 \implies p' = \texttt{nothing}$$

réciproque fausse : `pause`

- Déterministe

- Proverbe Esterel : « If must, do. If no can, no do. »

Que peut-on dire de la sémantique constructive ?

- Préserve les signaux déclarés

$$p \xhookrightarrow[E]{E',k} p' \implies \mathrm{dom}(E) = \mathrm{dom}(E')$$

- Si E est total (pas de $\bot$), E' l'est aussi
- Si le code de retour n'est pas 1, le résidu est `nothing`

$$p \xhookrightarrow[E]{E',k} p' \implies k \neq 1 \implies p' = \texttt{nothing}$$

réciproque fausse : `pause`

- Déterministe

$$p \xhookrightarrow[E]{E'_1,k_1} p'_1 \implies p \xhookrightarrow[E]{E'_2,k_2} p'_2 \implies p'_1 = p'_2 \wedge \ldots$$

- Proverbe Esterel : « If must, do. If no can, no do. »

Que peut-on dire de la sémantique constructive ?

- Préserve les signaux déclarés

$$p \xhookrightarrow[E]{E',k} p' \implies \mathrm{dom}(E) = \mathrm{dom}(E')$$

- Si E est total (pas de $\bot$), E' l'est aussi
- Si le code de retour n'est pas 1, le résidu est `nothing`

$$p \xhookrightarrow[E]{E',k} p' \implies k \neq 1 \implies p' = \texttt{nothing}$$

réciproque fausse : `pause`

- Déterministe

$$p \xhookrightarrow[E]{E'_1,k_1} p'_1 \implies p \xhookrightarrow[E]{E'_2,k_2} p'_2 \implies p'_1 = p'_2 \wedge \ldots$$

- Proverbe Esterel : « If must, do. If no can, no do. »
  - $s \in \mathit{Must}_s(p,E) \implies p \xhookrightarrow[E]{E',k} p' \implies s^+ \in E'$
  - $s \notin \mathit{Can}^+_s(p,E) \implies p \xhookrightarrow[E]{E',k} p' \implies s^- \in E'$

Que peut-on dire de la sémantique constructive ?

- Préserve les signaux déclarés

$$p \xhookrightarrow[E]{E',k} p' \implies \mathrm{dom}(E) = \mathrm{dom}(E')$$

- Si E est total (pas de $\bot$), E' l'est aussi
- Si le code de retour n'est pas 1, le résidu est `nothing`

$$p \xhookrightarrow[E]{E',k} p' \implies k \neq 1 \implies p' = \texttt{nothing}$$

réciproque fausse : `pause`

- Déterministe

$$p \xhookrightarrow[E]{E'_1,k_1} p'_1 \implies p \xhookrightarrow[E]{E'_2,k_2} p'_2 \implies p'_1 = p'_2 \wedge \ldots$$

- Proverbe Esterel : « If must, do. If no can, no do. »
  - $p \xhookrightarrow[E]{E',k} p' \implies (s^+ \in E' \iff s \in \mathit{Must}_s(p,E))$
  - $p \xhookrightarrow[E]{E',k} p' \implies (s^- \in E' \iff s \notin \mathit{Can}^+_s(p,E))$

Que peut-on dire de la sémantique constructive ?

- Préserve les signaux déclarés

$$p \xhookrightarrow[E]{E',k} p' \implies \mathrm{dom}(E) = \mathrm{dom}(E')$$

- Si E est total (pas de $\bot$), E' l'est aussi
- Si le code de retour n'est pas 1, le résidu est `nothing`

$$p \xhookrightarrow[E]{E',k} p' \implies k \neq 1 \implies p' = \texttt{nothing}$$

réciproque fausse : `pause`

- Déterministe $\quad p \xhookrightarrow[E]{E'_1,k_1} p'_1 \implies p \xhookrightarrow[E]{E'_2,k_2} p'_2 \implies p'_1 = p'_2 \wedge \ldots$
- Proverbe Esterel : « If must, do. If no can, no do. »
  - $p \xhookrightarrow[E]{E',k} p' \implies (s^+ \in E' \iff s \in \mathit{Must}_s(p,E))$
  - $p \xhookrightarrow[E]{E',k} p' \implies (s^- \in E' \iff s \notin \mathit{Can}^+_s(p,E))$
  - $p \xhookrightarrow[E]{E',k} p' \implies \mathit{Must}(p,E) = \mathit{Can}^+(p,E)$

Que peut-on dire de la sémantique constructive ?

- Préserve les signaux déclarés

$$p \xhookrightarrow[E]{E', k} p' \implies \mathrm{dom}(E) = \mathrm{dom}(E')$$

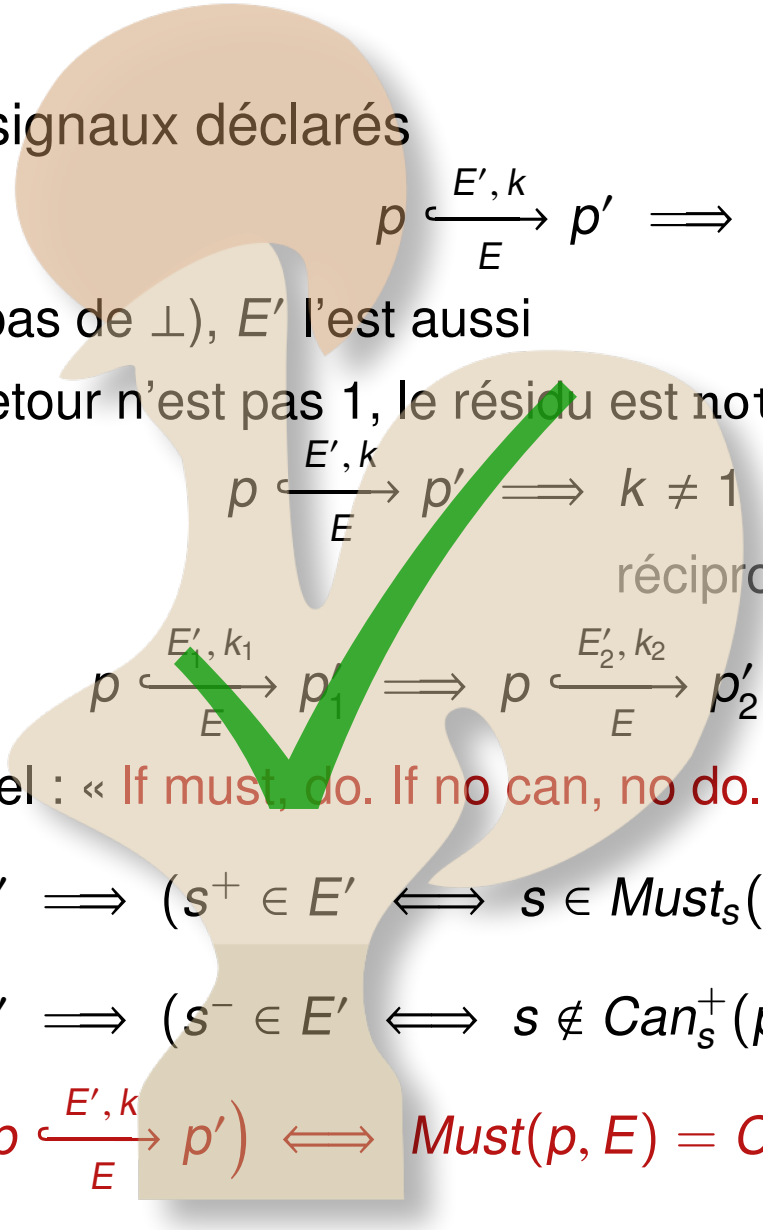
- Si E est total (pas de $\bot$), E' l'est aussi
- Si le code de retour n'est pas 1, le résidu est `nothing`

$$p \xhookrightarrow[E]{E', k} p' \implies k \neq 1 \implies p' = \texttt{nothing}$$

réciproque fausse : `pause`

- Déterministe $\quad p \xhookrightarrow[E]{E'_1, k_1} p'_1 \implies p \xhookrightarrow[E]{E'_2, k_2} p'_2 \implies p'_1 = p'_2 \wedge \ldots$
- Proverbe Esterel : « If must, do. If no can, no do. »
  - $p \xhookrightarrow[E]{E', k} p' \implies (s^+ \in E' \iff s \in \mathit{Must}_s(p, E))$
  - $p \xhookrightarrow[E]{E', k} p' \implies (s^- \in E' \iff s \notin \mathit{Can}^+_s(p, E))$
  - $\left(\exists E'\ k\ p', p \xhookrightarrow[E]{E', k} p'\right) \iff \mathit{Must}(p, E) = \mathit{Can}^+(p, E)$

Que peut-on dire de la sémantique constructive ?



- Préserve les signaux déclarés

$$p \xhookrightarrow[E]{E',k} p' \implies \mathrm{dom}(E) = \mathrm{dom}(E')$$

- Si E est total (pas de $\bot$), E' l'est aussi
- Si le code de retour n'est pas 1, le résidu est `nothing`

$$p \xhookrightarrow[E]{E',k} p' \implies k \neq 1 \implies p' = \texttt{nothing}$$

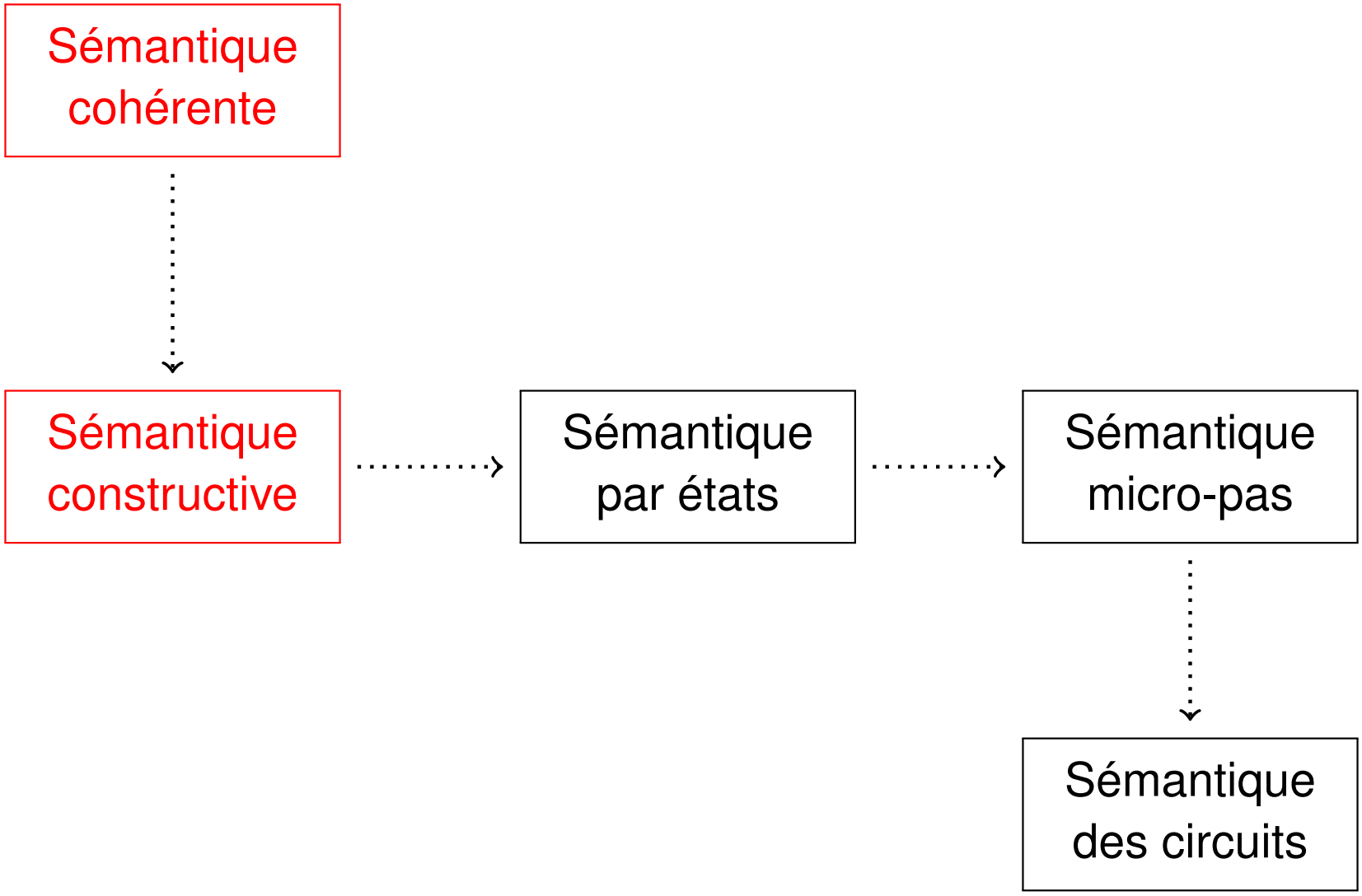
réciproque fausse : `pause`

- Déterministe

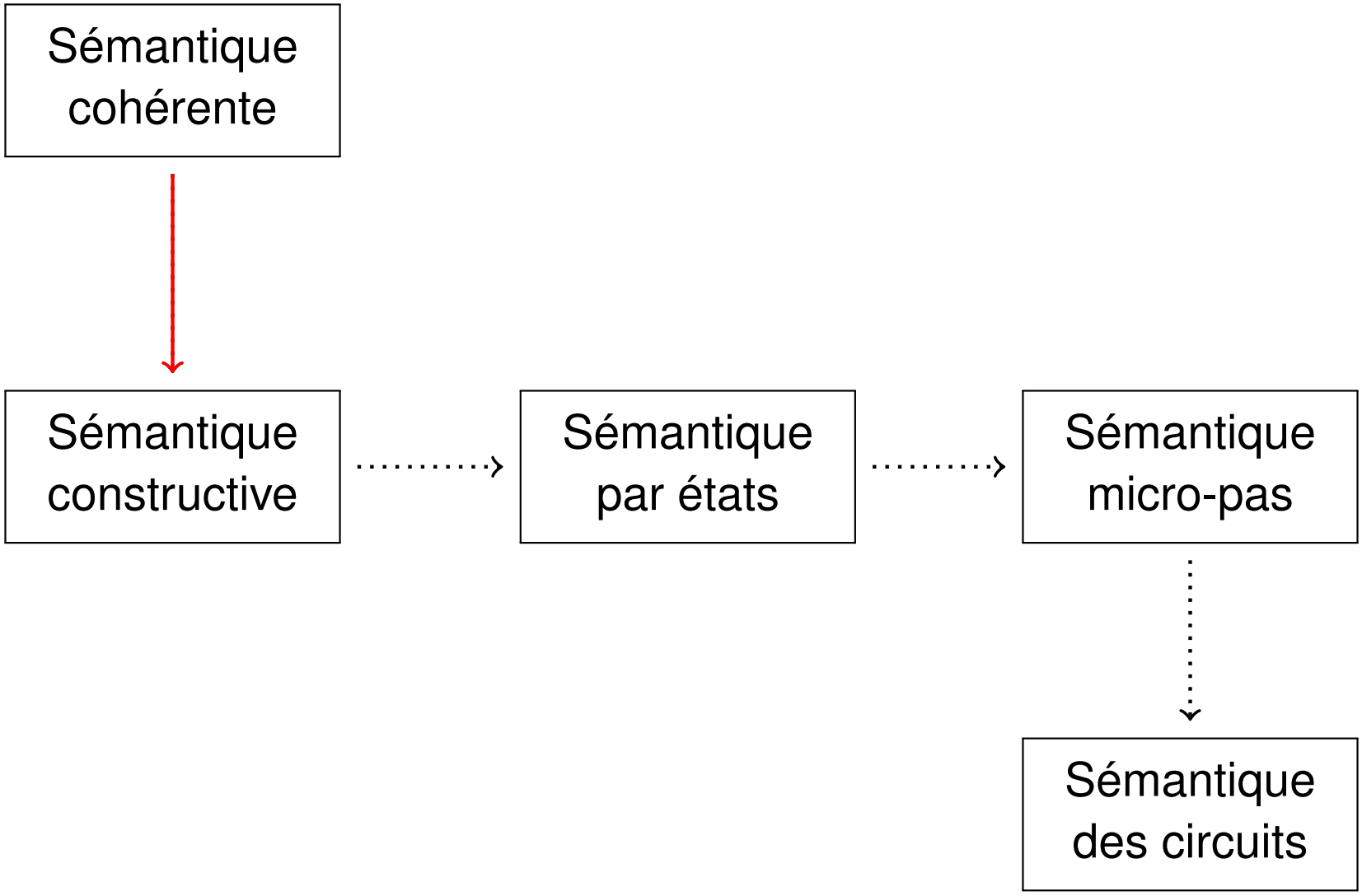
$$p \xhookrightarrow[E]{E'_1,k_1} p'_1 \implies p \xhookrightarrow[E]{E'_2,k_2} p'_2 \implies p'_1 = p'_2 \wedge \ldots$$

- Proverbe Esterel : « If must, do. If no can, no do. »
  - $p \xhookrightarrow[E]{E',k} p' \implies (s^+ \in E' \iff s \in \mathit{Must}_s(p,E))$
  - $p \xhookrightarrow[E]{E',k} p' \implies (s^- \in E' \iff s \notin \mathit{Can}^+_s(p,E))$
  - $\left(\exists E'\ k\ p', p \xhookrightarrow[E]{E',k} p'\right) \iff \mathit{Must}(p,E) = \mathit{Can}^+(p,E)$

Chaîne des sémantique d'Esterel



Chaîne des sémantique d'Esterel



Relation cohérente/constructive

Objectif : Montrer $p \xrightarrow[E]{E', k} p' \implies p \xrightarrow[E]{E', k} p'$

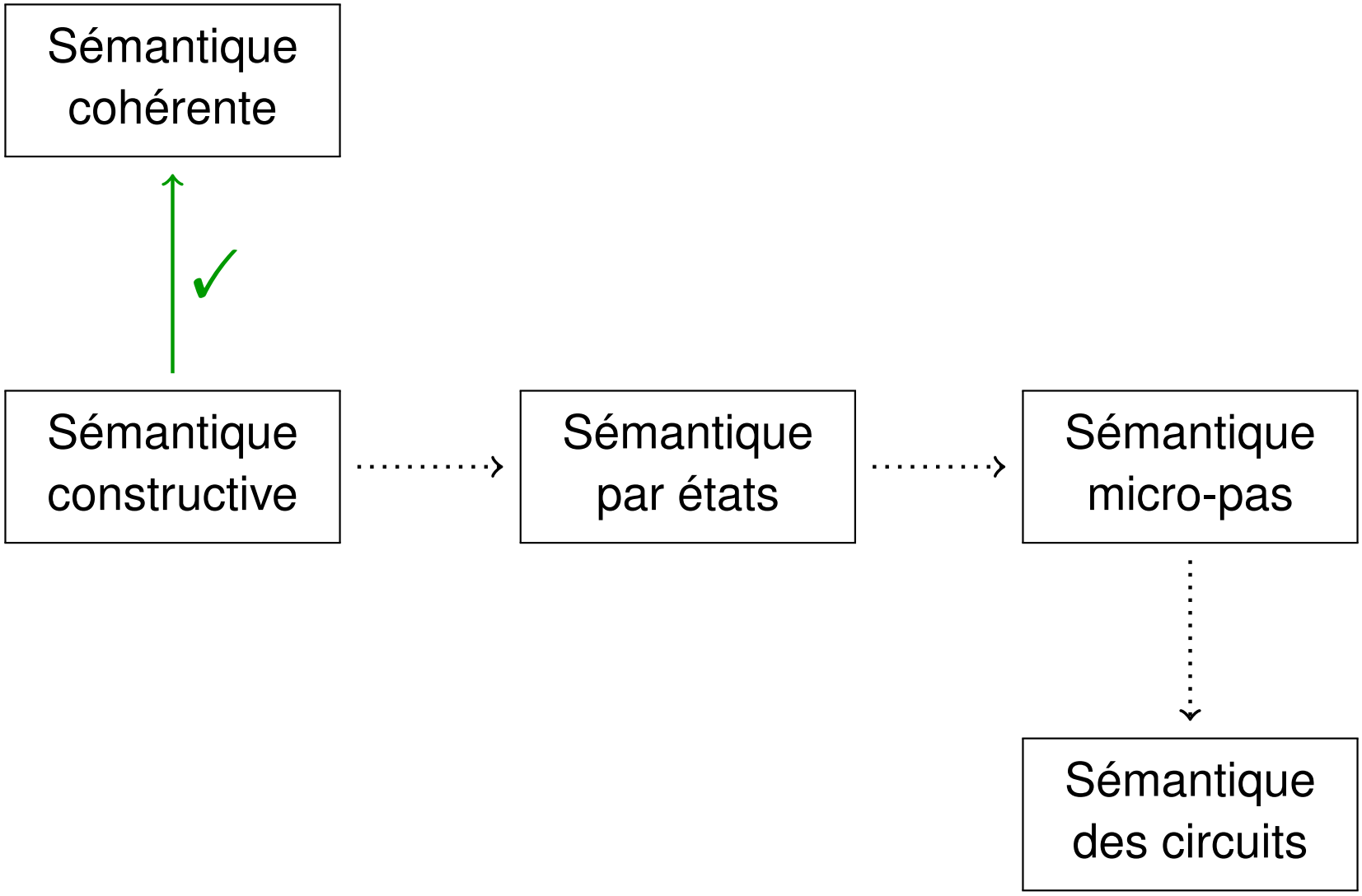
La réciproque est fausse.

Preuve facile :

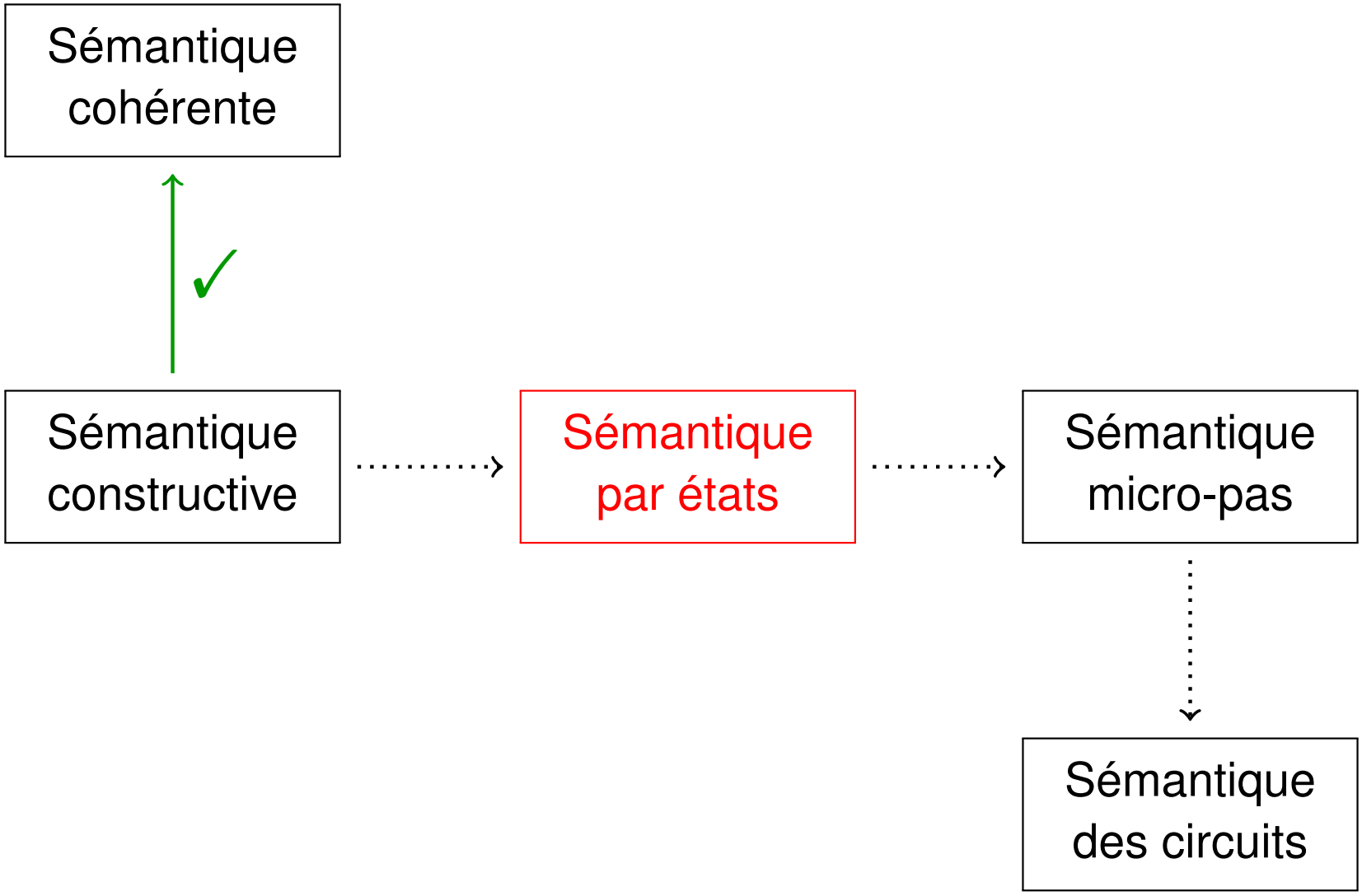
- Tous les cas sont identiques sauf sig$+/-$.
- Pour sig$+/-$, on utilise « If must, do. If no can, no do. »
  $\rightsquigarrow$ la difficulté est cachée dans Must/Can

$$E_1 \subseteq E_2 \implies Must(p, E_1) \subseteq Must(p, E_2)$$
$$E_1 \subseteq E_2 \implies Can^+(p, E_1) \supseteq Can^+(p, E_2)$$
$$Must(p, E) \subseteq Can^+(p, E)$$

Chaîne des sémantique d'Esterel



Chaîne des sémantique d'Esterel



Pourquoi une sémantique par états ?

Problème : la sémantique constructive réécrit les programmes

- les parties déjà exécutées sont effacées
- elle duplique les boucles $\texttt{loop p end} \equiv \texttt{p}\,;\,\texttt{loop p end}$

$\rightsquigarrow$ peu réaliste pour un circuit

Pourquoi une sémantique par états ?

Problème : la sémantique constructive réécrit les programmes

- les parties déjà exécutées sont effacées
- elle duplique les boucles `loop p end` ≡ `p ; loop p end`

⇝ peu réaliste pour un circuit

Sémantique par états : étape intermédiaire vers les circuits

- Le programme sous-jacent ne change pas
  ≠ sémantiques précédentes
- Des annotations $\widehat{\ }$ indiquent où l'on est à chaque instant
  ⇝ parallélisme = plusieurs annotations à la fois
- Plus proche de la sémantique des circuits
  ⇝ pause activé = registre activé
- Deux ensembles de règles

  start : programme → terme — Go

  resume : état → terme — Resume

Programmes et états

État = programme actif à la fin d'un instant

$\widehat{p}, \widehat{q} :=$

- `nothing`
- `pause`
- `emit` s
- `exit` T^k
- `trap` T `in` p `end`
- `if` s `then` p `else` q `end`
- `suspend` p `when` s
- $p\,;q$
- p `||` q
- `loop` p `end`
- `signal` s `in` p `end`

Programmes et états

État = programme actif à la fin d'un instant

$\widehat{p}, \widehat{q} :=$

`pause`
`trap` T `in` p `end`
`if` s `then` p `else` q `end`
`suspend` p `when` s
$p\,;q$
p `||` q
`loop` p `end`
`signal` s `in` p `end`

Programmes et états

État = programme actif à la fin d'un instant

$\widehat{p}, \widehat{q} :=$

$\widehat{\texttt{pause}}$	pause activé
`trap` T `in` p `end`	
`if` s `then` p `else` q `end`	
`suspend` p `when` s	
$p\,;q$	
p `||` q	
`loop` p `end`	
`signal` s `in` p `end`	

Programmes et états

État = programme actif à la fin d'un instant

$\widehat{p}, \widehat{q} :=$

$\widehat{\texttt{pause}}$	pause activé
$\texttt{trap T in } \widehat{p} \texttt{ end}$	
$\texttt{if s then } \widehat{p} \texttt{ else q end} \quad \mid \quad \texttt{if s then p else } \widehat{q} \texttt{ end}$	
$\texttt{suspend } \widehat{p} \texttt{ when s}$	
$\widehat{p}\,;q \quad \mid \quad p\,;\widehat{q}$	
$\widehat{p} \;\texttt{||}\; \widehat{q} \quad \mid \quad \widehat{p} \;\texttt{||}\; q \quad \mid \quad p \;\texttt{||}\; \widehat{q}$	
$\texttt{loop } \widehat{p} \texttt{ end}$	
$\texttt{signal } s \texttt{ in } \widehat{p} \texttt{ end}$	

Programmes et états

État = programme actif à la fin d'un instant

$\widehat{p}, \widehat{q} :=$

$\widehat{\texttt{pause}}$	pause activé
$\texttt{trap T in } \widehat{p} \texttt{ end}$	
$\texttt{if s then } \widehat{p} \texttt{ else q end} \quad \mid \quad \texttt{if s then p else } \widehat{q} \texttt{ end}$	
$\texttt{suspend } \widehat{p} \texttt{ when s}$	
$\widehat{p}\,;q \quad \mid \quad p\,;\widehat{q}$	
$\widehat{p} \;\|\|\; \widehat{q} \quad \mid \quad \widehat{p} \;\|\|\; q \quad \mid \quad p \;\|\|\; \widehat{q}$	
$\texttt{loop } \widehat{p} \texttt{ end}$	
$\texttt{signal } s \texttt{ in } \widehat{p} \texttt{ end}$	

Terme $\overline{p}$:= état (actif) $\widehat{p}$ | programme (inerte) p

Constructive vs par état : la règle if-then

- Sémantique constructive

$$\text{if-then}\ \frac{s^+ \in E \qquad p \xhookrightarrow[E]{E',k} p'}{s\,?\,p\,,\,q \xhookrightarrow[E]{E',k} p'}$$

- Sémantique par états : deux règles

Règle start

$$\frac{s^+ \in E \qquad p \xrightarrow[E]{E',k}\!\!\!\!\rightarrow_s \overline{p'}}{s\,?\,p\,,\,q \xrightarrow[E]{E',k}\!\!\!\!\rightarrow_s s\,?\,\overline{p'}\,,\,q}$$

Règle resume

$$\frac{\widehat{p} \xrightarrow[E]{E',k}\!\!\!\!\rightarrow_r \overline{p'}}{s\,?\,\widehat{p}\,,\,q \xrightarrow[E]{E',k}\!\!\!\!\rightarrow_r s\,?\,\overline{p'}\,,\,q}$$

Le retour d'ABRO

Sémantique constructive

```
loop
    abort
        (awimm A || awimm B);
        emit O;
        halt
    when R
end
```

Sémantique par états

```
loop
    abort
        (awimm A || awimm B);
        emit O;
        halt
    when R
end
```

Le retour d'ABRO

Sémantique constructive

```
loop
    abort
        (awimm A || awimm B);
        emit O;
        halt
    when R
end
```

$\{B\}$

Sémantique par états

```
loop
    abort
        (awimm A || awimm B);
        emit O;
        halt
    when R
end
```

Le retour d'ABRO

Sémantique constructive

```
abort
    (awimm A || nothing);
    emit O;
    halt
when R;
loop
    abort
        (awimm A || awimm B);
        emit O;
        halt
    when R
end
```

$\{B\}$

Sémantique par états

```
loop
    abort
        (awimm A || awimm B);
        emit O;
        halt
    when R
end
```

Le retour d'ABRO

Sémantique constructive

```
abort
    (awimm A || nothing);
    emit O;
    halt
when R;
loop
    abort
        (awimm A || awimm B);
        emit O;
        halt
    when R
end
```

$$\{B\} \Longrightarrow \{A, O\}$$

Sémantique par états

```
loop
    abort
        (awimm A || awimm B);
        emit O;
        halt
    when R
end
```

Le retour d'ABRO

Sémantique constructive

```
abort
    halt
when R;
loop
    abort
        (awimm A || awimm B);
        emit O;
        halt
    when R
end
```

$$\{B\} \Longrightarrow \{A, O\}$$

Sémantique par états

```
loop
    abort
        (awimm A || awimm B);
        emit O;
        halt
    when R
end
```

Le retour d'ABRO

Sémantique constructive

```
abort
    halt
when R;
loop
    abort
        (awimm A || awimm B);
        emit O;
        halt
    when R
end
```

$$\{B\} \Longrightarrow \{A, O\} \Longrightarrow \{B\}$$

Sémantique par états

```
loop
    abort
        (awimm A || awimm B);
        emit O;
        halt
    when R
end
```

Le retour d'ABRO

Sémantique constructive

```
loop
    abort
        (awimm A || awimm B);
        emit O;
        halt
    when R
end
```

Sémantique par états

```
loop
    abort
        (awimm A || awimm B);
        emit O;
        halt
    when R
end
```

$$\{B\} \Longrightarrow \{A, O\} \Longrightarrow \{B\} \Longrightarrow \{R\}$$

Le retour d'ABRO

Sémantique constructive

```
abort
    (awimm A || awimm B);
    emit O;
    halt
when R;
loop
    abort
        (awimm A || awimm B);
        emit O;
        halt
    when R
end
```

Sémantique par états

```
loop
    abort
        (awimm A || awimm B);
        emit O;
        halt
    when R
end
```

$$\{B\} \Longrightarrow \{A, O\} \Longrightarrow \{B\} \Longrightarrow \{R\}$$

Le retour d'ABRO

Sémantique constructive

```
abort
    halt
when R;
loop
    abort
        (awimm A || awimm B);
        emit O;
        halt
    when R
end
```

Sémantique par états

```
loop
    abort
        (awimm A || awimm B);
        emit O;
        halt
    when R
end
```

$$\{B\} \Longrightarrow \{A, O\} \Longrightarrow \{B\} \Longrightarrow \{R\} \Longrightarrow \{A, B, O\}$$

Que peut-on dire de la sémantique par états ?

- Préserve les signaux déclarés
- Préserve le programme sous-jacent
- Si E est total (pas de $\bot$), E' l'est aussi
- Le code de retour n'est pas 1 ssi le résidu est inerte
- Déterministe :
- Must/Can peuvent s'étendre à $\widehat{p}$: sMust/sCan

Que peut-on dire de la sémantique par états ?

- Préserve les signaux déclarés

$$\overline{p} \xrightarrow[E]{E',k}\!\!\!\!\!\rightarrow\!\!\!\!\!\rightarrow_{s/r} \overline{p'} \implies \mathrm{dom}(E) = \mathrm{dom}(E')$$

- Préserve le programme sous-jacent

- Si E est total (pas de $\bot$), E' l'est aussi
- Le code de retour n'est pas 1 ssi le résidu est inerte

- Déterministe :

- Must/Can peuvent s'étendre à $\widehat{p}$: sMust/sCan

Que peut-on dire de la sémantique par états ?

- Préserve les signaux déclarés

$$\overline{p} \xrightarrow[E]{E',k} \!\!\!\!\!\rightarrow_{s/r} \overline{p'} \implies \mathrm{dom}(E) = \mathrm{dom}(E')$$

- Préserve le programme sous-jacent

$$\overline{p} \xrightarrow[E]{E',k} \!\!\!\!\!\rightarrow_{s/r} \overline{p'} \implies \mathit{base}(\overline{p}) = \mathit{base}(\overline{p}')$$

- Si E est total (pas de $\bot$), E' l'est aussi
- Le code de retour n'est pas 1 ssi le résidu est inerte

- Déterministe :

- Must/Can peuvent s'étendre à $\widehat{p}$: sMust/sCan

Que peut-on dire de la sémantique par états ?

- Préserve les signaux déclarés

$$\overline{p} \xrightarrow[E]{E',k} \!\!\!\!\rightarrow_{s/r} \overline{p'} \implies \mathrm{dom}(E) = \mathrm{dom}(E')$$

- Préserve le programme sous-jacent

$$\overline{p} \xrightarrow[E]{E',k} \!\!\!\!\rightarrow_{s/r} \overline{p'} \implies base(\overline{p}) = base(\overline{p}')$$

- Si E est total (pas de $\bot$), E' l'est aussi
- Le code de retour n'est pas 1 ssi le résidu est inerte

$$\overline{p} \xrightarrow[E]{E',k} \!\!\!\!\rightarrow_{s/r} \overline{p'} \implies (k \neq 1 \iff \overline{p'} = base(\overline{p}))$$

- Déterministe :

- Must/Can peuvent s'étendre à $\widehat{p}$: sMust/sCan

Que peut-on dire de la sémantique par états ?

- Préserve les signaux déclarés

$$\overline{p} \xrightarrow[E]{E',k}\!\!\!\!\!\rightarrow\!\!\!\!\!\rightarrow_{s/r} \overline{p'} \implies \mathrm{dom}(E) = \mathrm{dom}(E')$$

- Préserve le programme sous-jacent

$$\overline{p} \xrightarrow[E]{E',k}\!\!\!\!\!\rightarrow\!\!\!\!\!\rightarrow_{s/r} \overline{p'} \implies \mathit{base}(\overline{p}) = \mathit{base}(\overline{p}')$$

- Si E est total (pas de $\bot$), E' l'est aussi
- Le code de retour n'est pas 1 ssi le résidu est inerte

$$\overline{p} \xrightarrow[E]{E',k}\!\!\!\!\!\rightarrow\!\!\!\!\!\rightarrow_{s/r} \overline{p'} \implies (k \neq 1 \iff \overline{p'} = \mathit{base}(\overline{p}))$$

- Déterministe : $\overline{p} \xrightarrow[E]{E',k}\!\!\!\!\!\rightarrow\!\!\!\!\!\rightarrow_{s/r} \overline{p'_1} \implies \overline{p} \xrightarrow[E]{E',k}\!\!\!\!\!\rightarrow\!\!\!\!\!\rightarrow_{s/r} \overline{p'_2} \implies \ldots$
- Must/Can peuvent s'étendre à $\widehat{p}$: sMust/sCan

Que peut-on dire de la sémantique par états ?

- Préserve les signaux déclarés

$$\overline{p} \xrightarrow[E]{E',k} \!\!\!\!\rightarrow_{s/r} \overline{p'} \implies \mathrm{dom}(E) = \mathrm{dom}(E')$$

- Préserve le programme sous-jacent

$$\overline{p} \xrightarrow[E]{E',k} \!\!\!\!\rightarrow_{s/r} \overline{p'} \implies \mathit{base}(\overline{p}) = \mathit{base}(\overline{p}')$$

- Si E est total (pas de $\bot$), E' l'est aussi
- Le code de retour n'est pas 1 ssi le résidu est inerte

$$\overline{p} \xrightarrow[E]{E',k} \!\!\!\!\rightarrow_{s/r} \overline{p'} \implies (k \neq 1 \iff \overline{p'} = \mathit{base}(\overline{p}))$$

- Déterministe : $\overline{p} \xrightarrow[E]{E',k} \!\!\!\!\rightarrow_{s/r} \overline{p'_1} \implies \overline{p} \xrightarrow[E]{E',k} \!\!\!\!\rightarrow_{s/r} \overline{p'_2} \implies \ldots$
- Must/Can peuvent s'étendre à $\widehat{p}$: sMust/sCan

  avec $\mathit{sMust}(\widehat{p}, E) = \mathit{Must}(\mathcal{E}(\widehat{p}), E)$

  $\mathit{sCan}^+(\widehat{p}, E) = \mathit{Can}^+(\mathcal{E}(\widehat{p}), E)$

  $\mathcal{E}(\widehat{p})$ = expansion de p ce qui reste à exécuter

  $\rightsquigarrow$ On récupère tous les résultats de Must/Can

Que peut-on dire de la sémantique par états ?

- Préserve les signaux déclarés

$$\overline{p} \xrightarrow[E]{E',k} \twoheadrightarrow_{s/r} \overline{p'} \implies \mathrm{dom}(E) = \mathrm{dom}(E')$$

- Préserve le programme sous-jacent

$$\overline{p} \xrightarrow[E]{E',k} \twoheadrightarrow_{s/r} \overline{p'} \implies base(\overline{p}) = base(\overline{p'})$$

- Si E est total (pas de $\perp$), E' l'est aussi
- Le code de retour n'est pas 1 ssi le résidu est inerte

$$\overline{p} \xrightarrow{E',k} \twoheadrightarrow_{s/r} \overline{p'} \implies (k \neq 1 \iff \overline{p'} = base(\overline{p}))$$

- Déterministe :

$$\overline{p} \xrightarrow[E]{E',k} \twoheadrightarrow_{s/r} \overline{p'_1} \implies \overline{p} \xrightarrow[E]{E',k} \twoheadrightarrow_{s/r} \overline{p'_2} \implies \dots$$

- Must/Can peuvent s'étendre à $\widehat{p}$: sMust/sCan

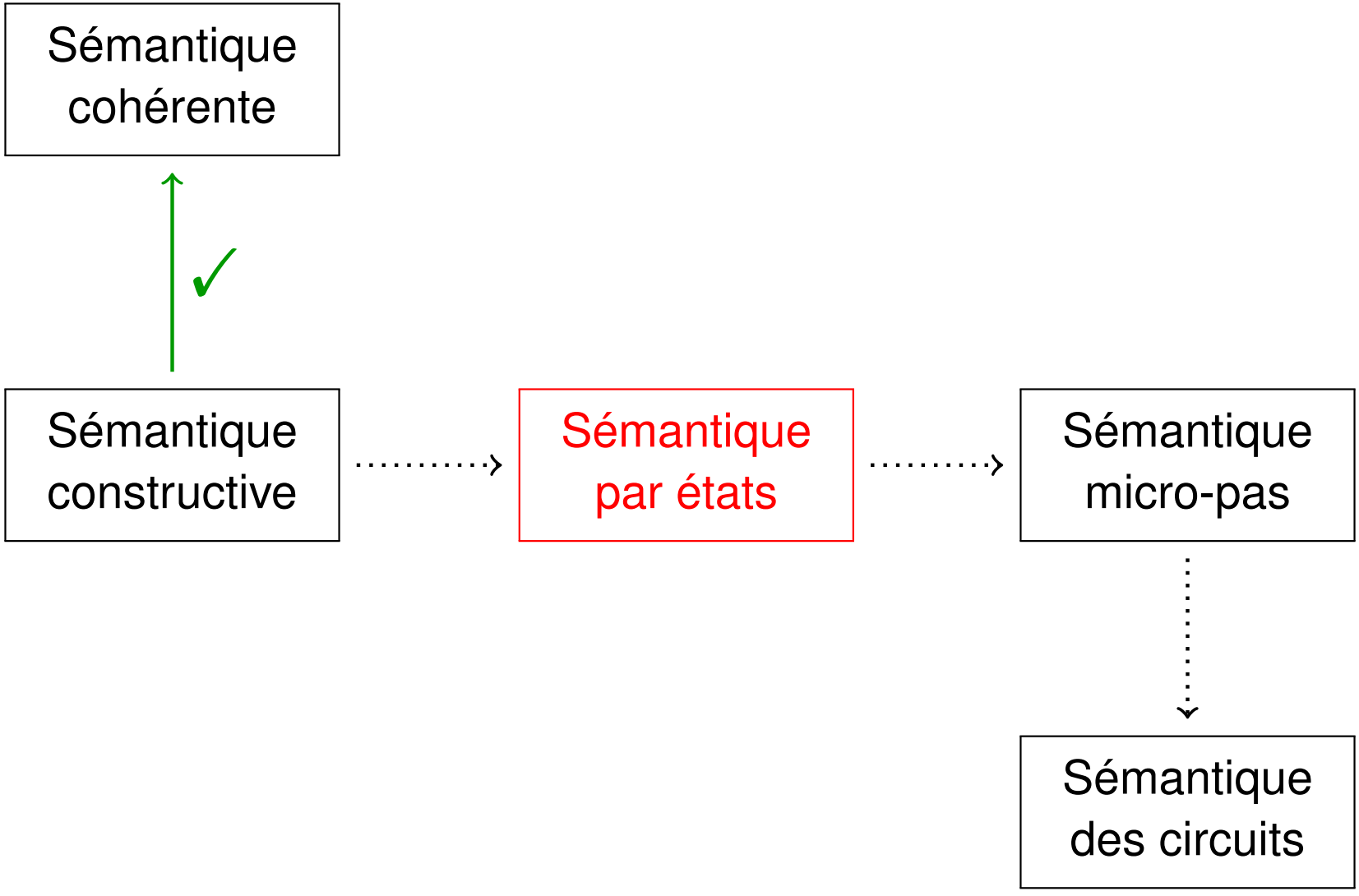
  avec $sMust(\widehat{p}, E) = Must(\mathcal{E}(\widehat{p}), E)$

  $sCan^{+}(\widehat{p}, E) = Can^{+}(\mathcal{E}(\widehat{p}), E)$

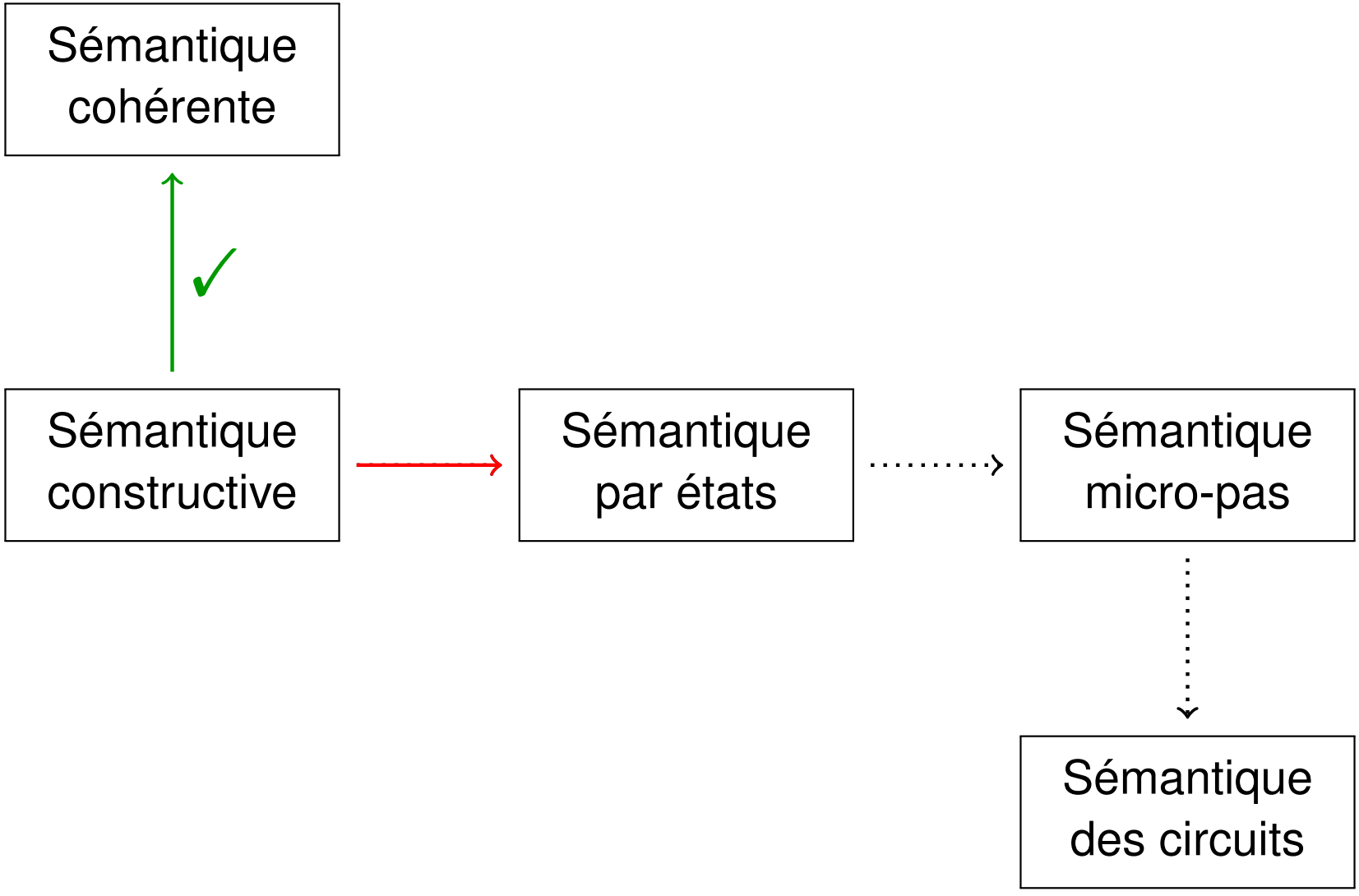
  $\mathcal{E}(\widehat{p})$ = expansion de p ce qui reste à exécuter

  $\rightsquigarrow$ On récupère tous les résultats de Must/Can

Chaîne des sémantique d'Esterel



Chaîne des sémantique d'Esterel



Équivalence constructive/par états

Objectif : Montrer $p \xrightarrow[E]{E',k}\!\!\!\!\twoheadrightarrow_s \overline{p}' \iff p \xhookrightarrow[E]{E',k} \mathcal{E}(\overline{p}')$

$\widehat{p} \xrightarrow[E]{E',k}\!\!\!\!\twoheadrightarrow_r \overline{p}' \iff \mathcal{E}(\widehat{p}) \xhookrightarrow[E]{E',k} \mathcal{E}(\overline{p'})$

Équivalence constructive/par états

Objectif : Montrer $p \xrightarrow[E]{E',k}\!\!\!\!\!\rightarrow_s \overline{p}' \iff p \xhookrightarrow[E]{E',k} \mathcal{E}(\overline{p}')$

$\widehat{p} \xrightarrow[E]{E',k}\!\!\!\!\!\rightarrow_r \overline{p}' \iff \mathcal{E}(\widehat{p}) \xhookrightarrow[E]{E',k} \mathcal{E}(\overline{p}')$

$\overline{p}$ et $\mathcal{E}(\overline{p})$ n'ont pas toujours la même structure

`loop p end`, `suspend p when s`

Équivalence constructive/par états

Objectif : Montrer $p \xrightarrow[E]{E',k}\!\!\!\!\twoheadrightarrow_s \overline{p}' \iff p \xhookrightarrow[E]{E',k} \mathcal{E}(\overline{p}')$

$\widehat{p} \xrightarrow[E]{E',k}\!\!\!\!\twoheadrightarrow_r \overline{p}' \iff \mathcal{E}(\widehat{p}) \xhookrightarrow[E]{E',k} \mathcal{E}(\overline{p}')$

$\overline{p}$ et $\mathcal{E}(\overline{p})$ n'ont pas toujours la même structure

`loop` p `end`, `suspend` p `when` s

Solution : bisimilarité $p \equiv q$ — p et q se comportent pareil

$\rightsquigarrow$ exemples : $p \equiv p$; `nothing` $\equiv$ `nothing` || p

`loop` p `end` $\equiv p$; `loop` p `end` $\equiv$ `loop` p ; p `end`

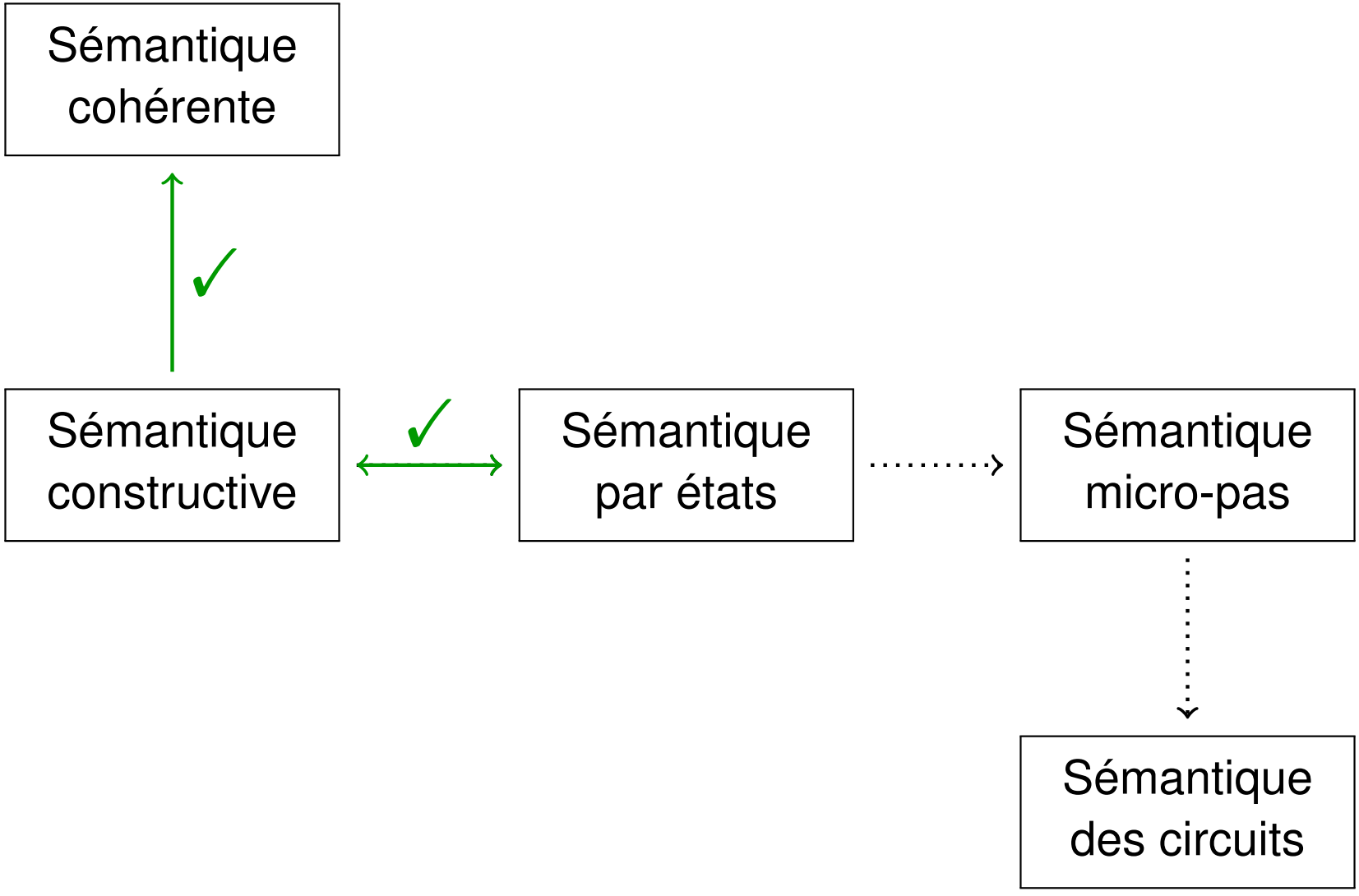
On démontre à la place

$$\widehat{p} \xrightarrow[E]{E',k}\!\!\!\!\twoheadrightarrow_r \overline{p'} \implies \exists q, \mathcal{E}(\widehat{p}) \xhookrightarrow[E]{E',k} q \wedge \mathcal{E}(\overline{p'}) \equiv q$$

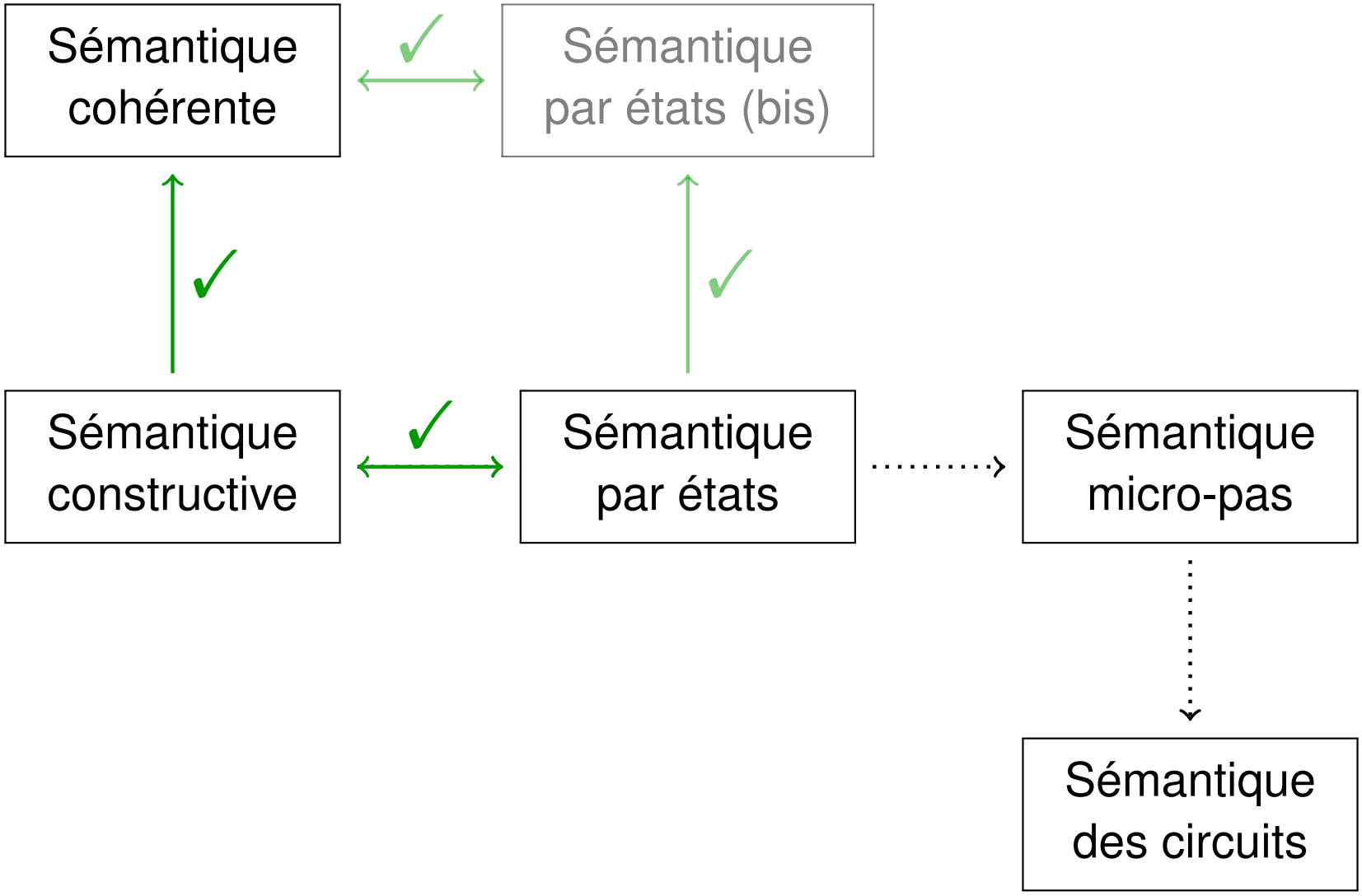
+ les 3 autres variantes

Le reste est facile (les règles sont les mêmes).

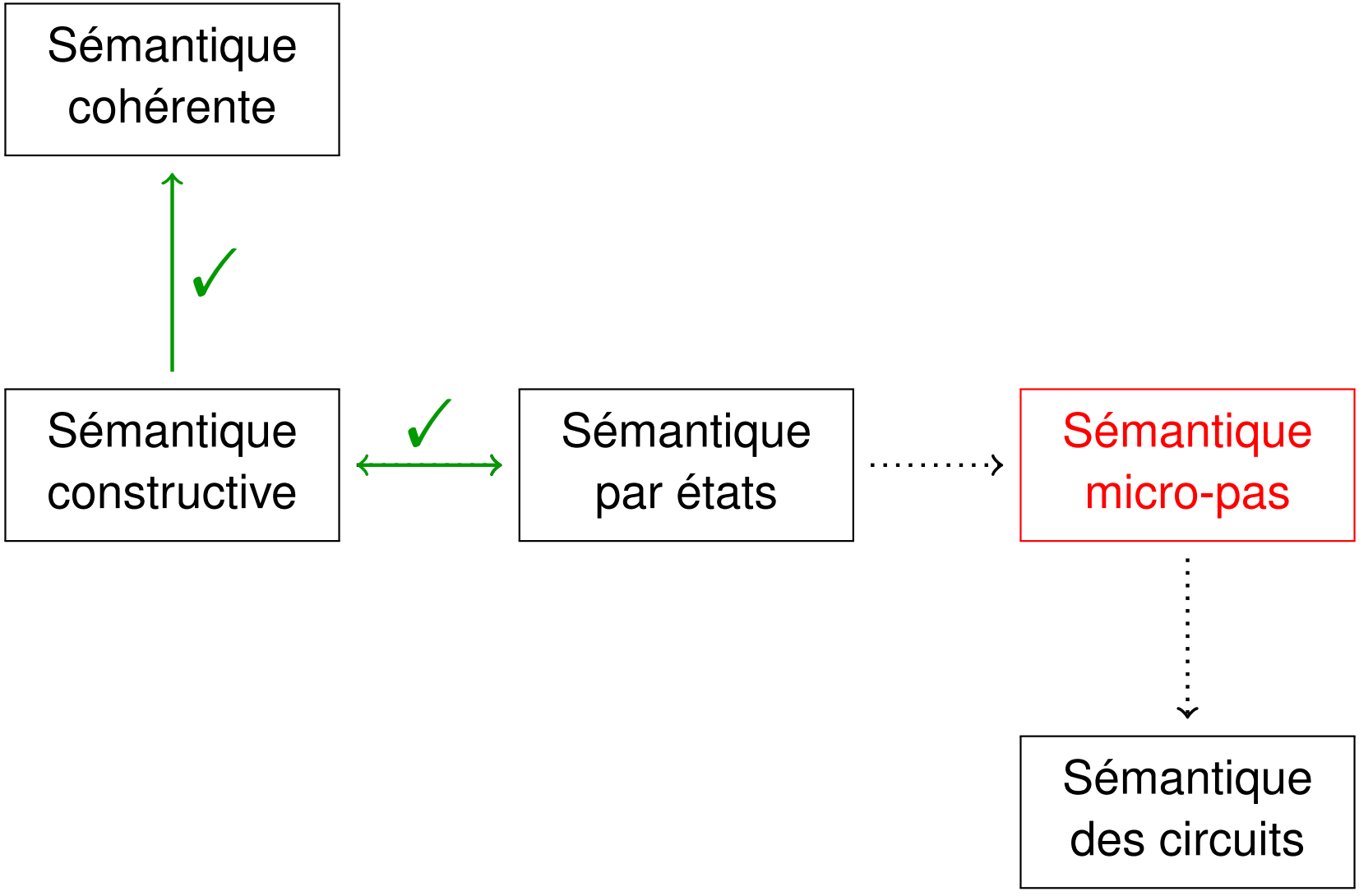
Chaîne des sémantique d'Esterel



Chaîne des sémantique d'Esterel



Chaîne des sémantique d'Esterel



Entrons dans l'instant : la sémantique micro-pas

Sémantique par états

- Correspondance parfaite avec les circuits entre les instants
- Calcul des signaux locaux en 2 étapes

Sémantique micro-pas

- Expliquer le calcul dans un instant
  $\rightsquigarrow$ être aussi précis que les portes logiques
- S'affranchir de Must/Can

Limitations

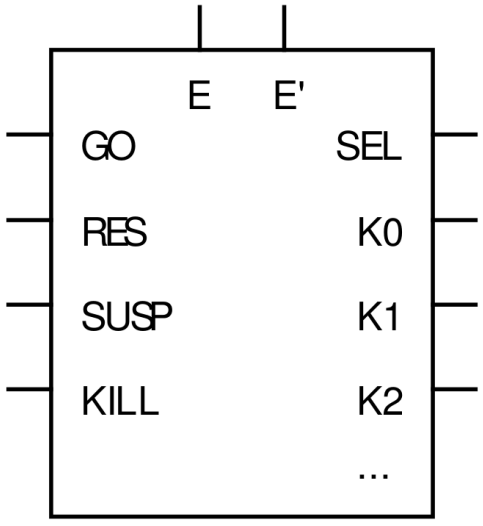
- Pas de fils pour les signaux, on garde E
  $\rightsquigarrow$ beaucoup moins de fils
- Synchroniseur comme une boîte noire
  $\rightsquigarrow$ on utilise sa spécification
- Pas de réincarnation car pas de boucle

Intuition de la sémantique micro-pas

Inspiration et idées

- Garder la structure des programmes — SOS
- Augmenter l'information — sémantique de Scott
- Se limiter à l'intérieur d'un instant
  $\rightsquigarrow$ on ne franchit jamais un `pause`
- Combiner avec la sémantique par états
  $\rightsquigarrow$ à chaque frontière d'instant, on repasse par un état
- Trois phases dans le circuit
  - **S**tart : transférer le contrôle en entrée
  - **C**ontext : exécuter les sous-circuits
  - **E**nd : calculer les codes de retours
- Forme générale d'un micro-état : $\Box\ p\ \circ$

Intuition pour les micro-états



D'après la traduction en circuits

Go/Resume transfèrent le contrôle

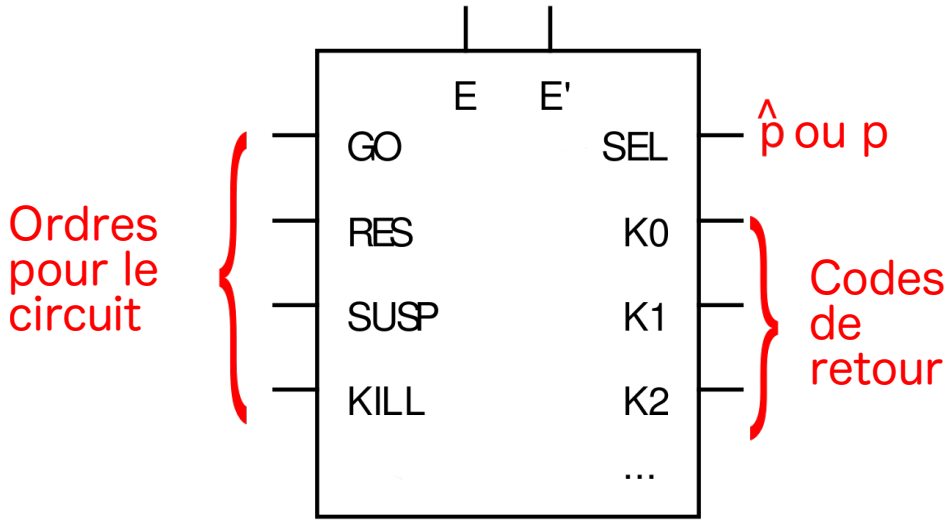
Suspend gèle le calcul

Kill empêche l'activation des `pauses`

Sel propage l'activation

Ki propagent la terminaison et les trappes

Intuition pour les micro-états



D'après la traduction en circuits

Go/Resume transfèrent le contrôle

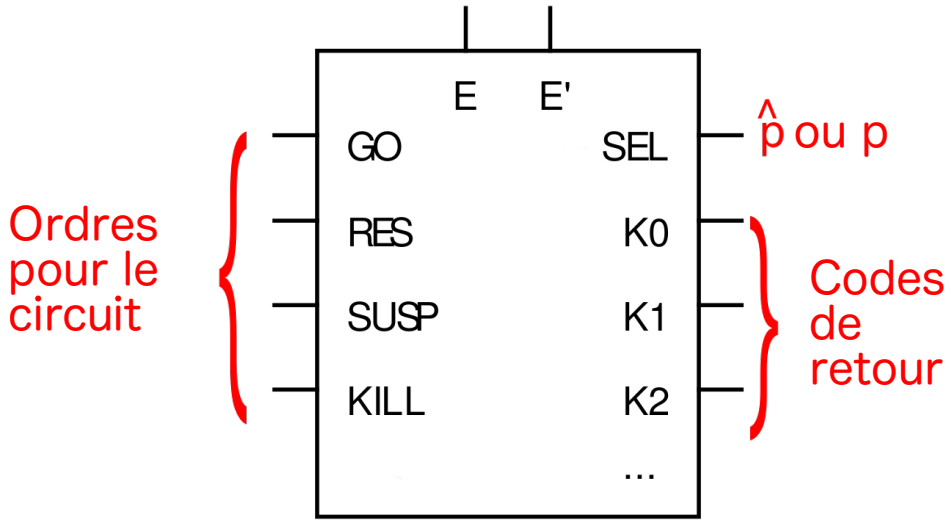
Suspend gèle le calcul

Kill empêche l'activation des `pauses`

Sel propage l'activation

Ki propagent la terminaison et les trappes

Intuition pour les micro-états



D'après la traduction en circuits

Go/Resume transfèrent le contrôle

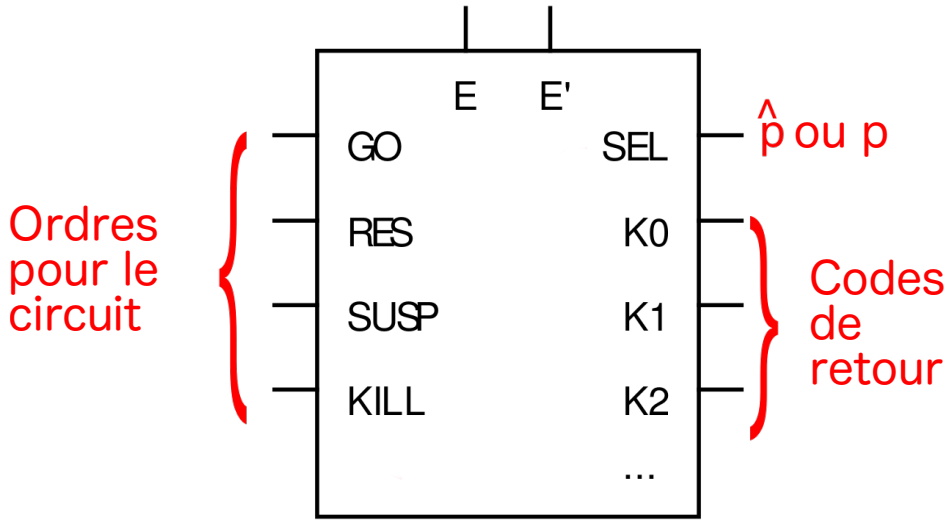
Suspend gèle le calcul

~~Kill~~ ~~empêche l'activation des pauses~~

~~Sel~~ ~~propage l'activation~~

Ki propagent la terminaison et les trappes

Intuition pour les micro-états



D'après la traduction en circuits

Go/Resume transfèrent le contrôle

Suspend gèle le calcul

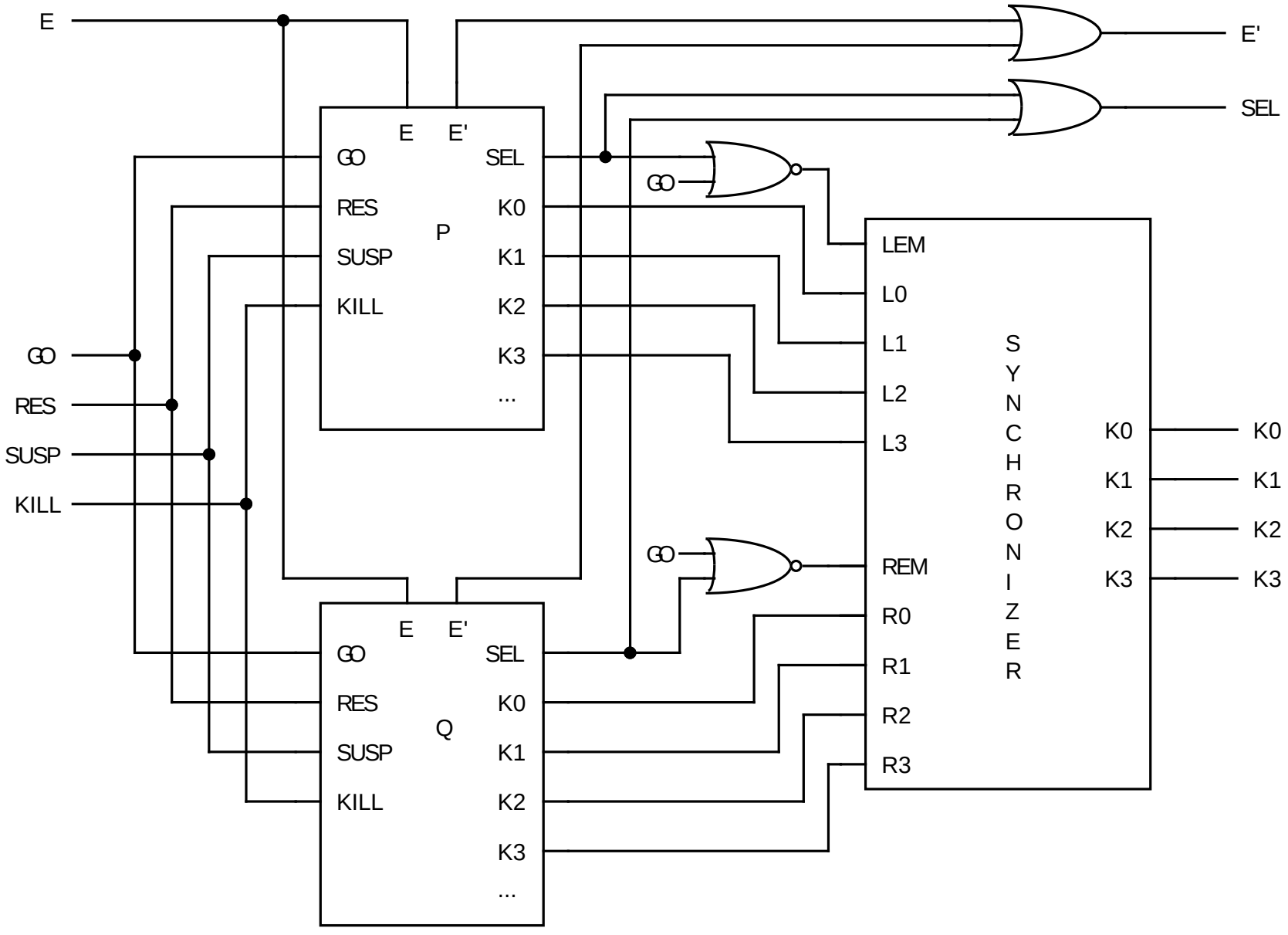
~~Kill~~ ~~empêche l'activation des pauses~~

~~Sel~~ ~~propage l'activation~~

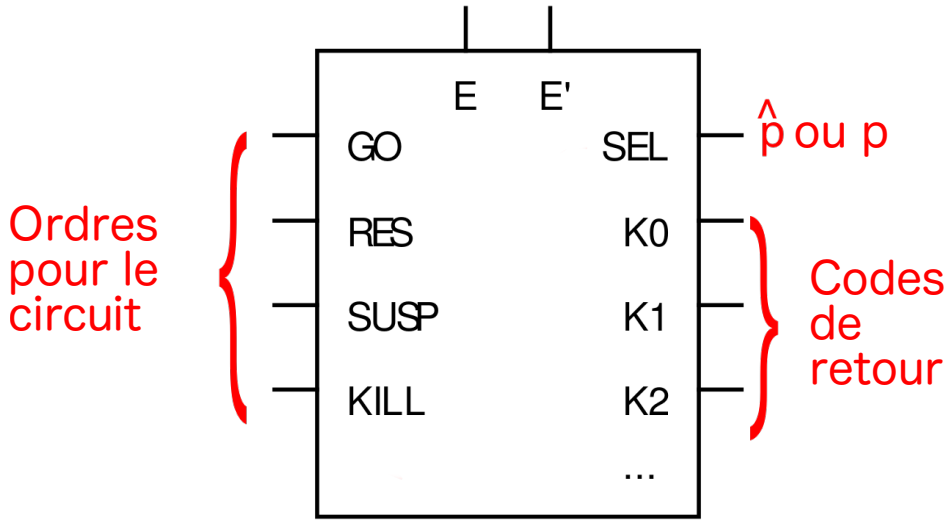
Ki propagent la terminaison et les trappes

Sel est utilisé dans le synchroniseur !

Le synchroniseur de *p* || *q*



Intuition pour les micro-états



D'après la traduction en circuits

Go/Resume transfèrent le contrôle

Suspend gèle le calcul

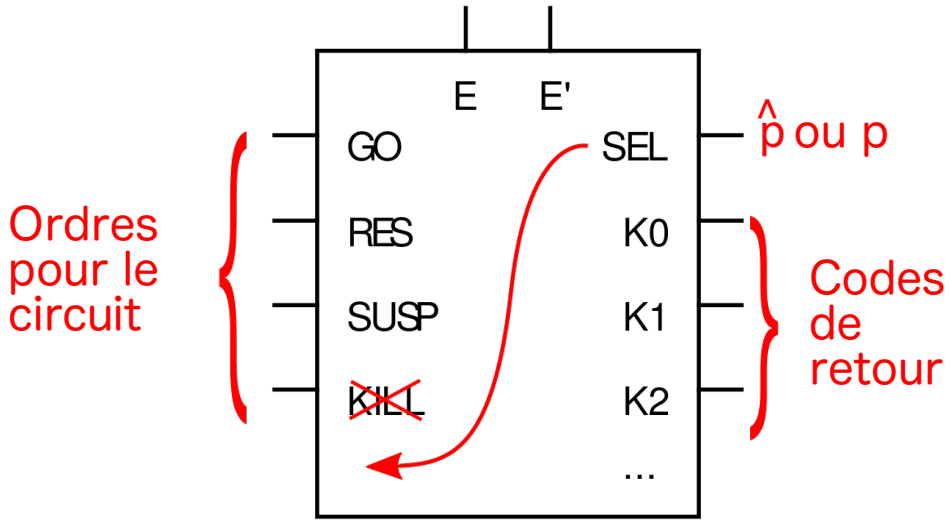
~~Kill~~ ~~empêche l'activation des pauses~~

~~Sel~~ ~~propage l'activation~~

Ki propagent la terminaison et les trappes

Sel est utilisé dans le synchroniseur !

Intuition pour les micro-états



D'après la traduction en circuits

Go/Resume transfèrent le contrôle

Suspend gèle le calcul

~~Kill empêche l'activation des pauses~~

~~Sel propage l'activation~~

Ki propagent la terminaison et les trappes

Sel est utilisé dans le synchroniseur !

Définition des micro-états

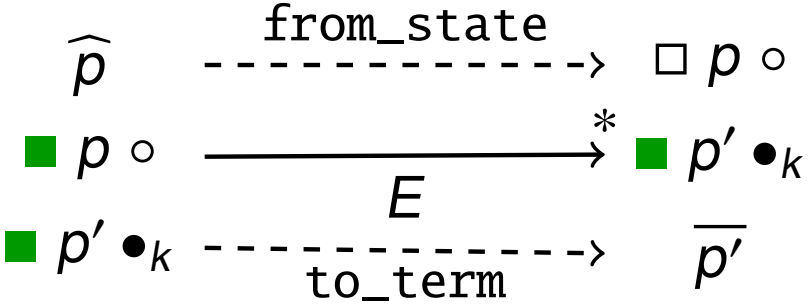
Micro-état : $\square\ p\ \circ$

- Entrées $\square$ — $+, -, \perp$
  - Go, Res, Susp
- Programme p
  - Inerte, tout se passe dans $\square$ et $\circ$
  - Structurel ($\square$ / $\circ$ dans les sous-circuits)
  - Sel fixé à la création
- Sorties $\circ$
  - Codage « 1-hot » en sortie — au plus un fil à 1
  - Si un fil à 1 : $\bullet_k$ — Must_K
  - Si aucun fil à 1 : $\circ_K$ — Can_K
- Ordre de Scott $\leqslant$ — mesure l'information
  - Entrées : $\square \leqslant \square$ par composantes
  - Sorties : $\circ_K \leqslant \circ_L := L \subseteq K$
    $\circ_K \leqslant \bullet_k := k \in K$
    $\bullet_k \leqslant \bullet_l := k = l$

Définition des micro-pas

Micro-pas : $\Box\, p \circ \xrightarrow[E]{} \Box\, p' \circ$

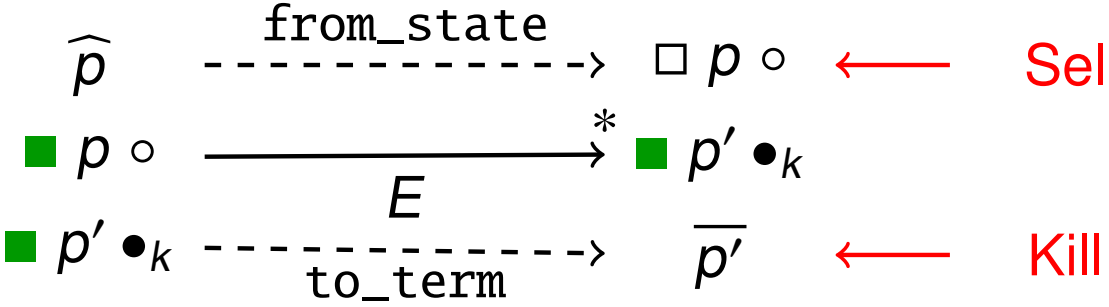
- Mettre à jour $\Box$ et $\circ$ jusqu'à atteindre l'info max
  - Entrées : Go, Res, Susp $\neq \bot$ faire disparaître $\bot$
  - Sorties : $\bullet_k$ ou $\circ_\varnothing$
- Trop petits pas pour être une transition d'état
  - Pas de code de retour
    $\rightsquigarrow$ dans les sorties $\circ$
  - Pas de signaux émis (gérés par les règles sig$+$/$-$)
    - Must/Can remplacé par lecture du micro-état
    - s émis ssi $\Box$ `emit` $s\, \bullet_0$
- Connexion avec la sémantique par état :



Définition des micro-pas

Micro-pas : $\Box\, p \circ \xrightarrow[E]{} \Box\, p' \circ$

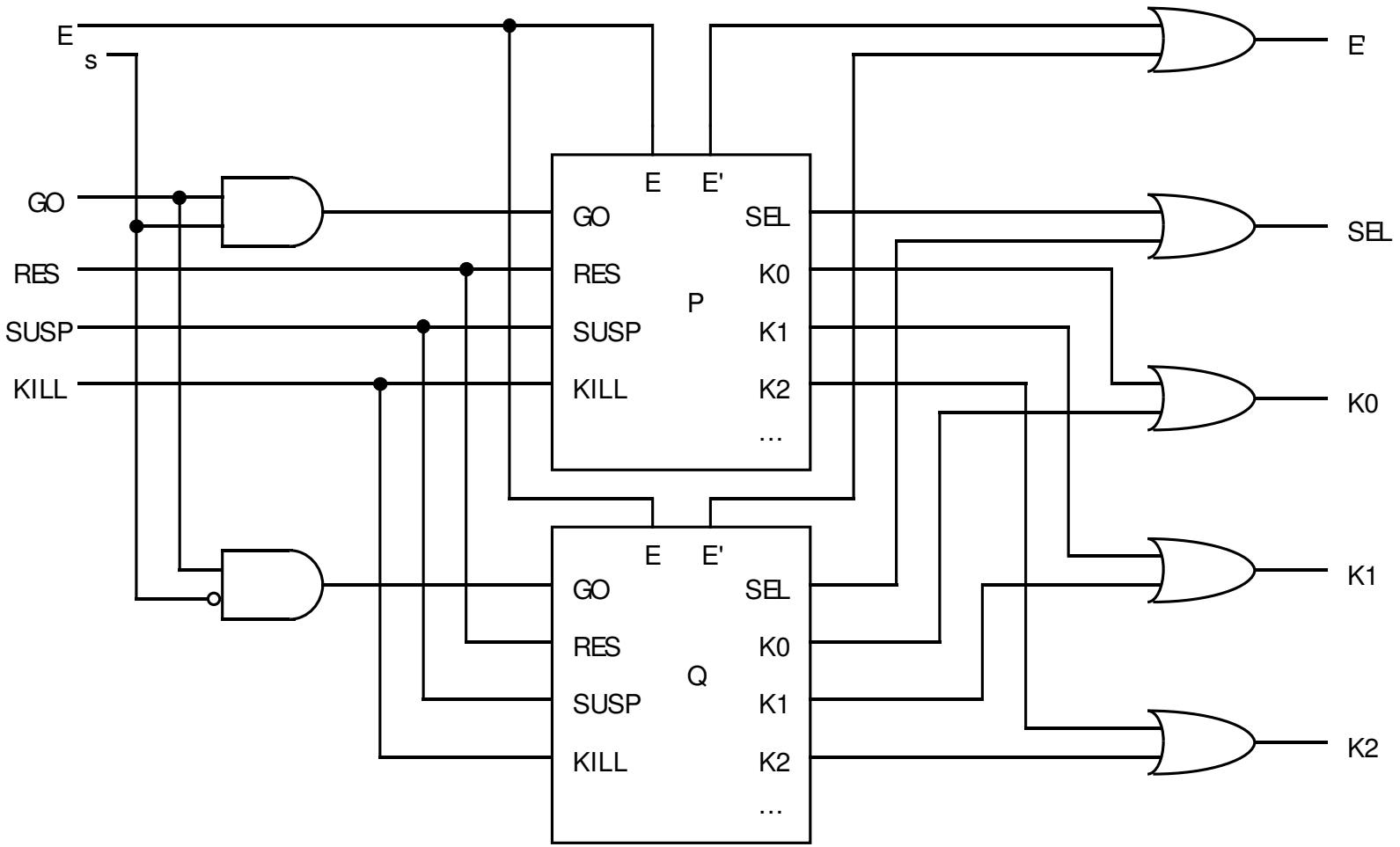
- Mettre à jour $\Box$ et $\circ$ jusqu'à atteindre l'info max
  - Entrées : Go, Res, Susp $\neq \bot$ faire disparaître $\bot$
  - Sorties : $\bullet_k$ ou $\circ_\varnothing$
- Trop petits pas pour être une transition d'état
  - Pas de code de retour
    $\rightsquigarrow$ dans les sorties $\circ$
  - Pas de signaux émis (gérés par les règles sig$+$/$-$)
    - Must/Can remplacé par lecture du micro-état
    - s émis ssi $\Box$ `emit` $s\, \bullet_0$
- Connexion avec la sémantique par état :



Règles micro-pas pour if-then

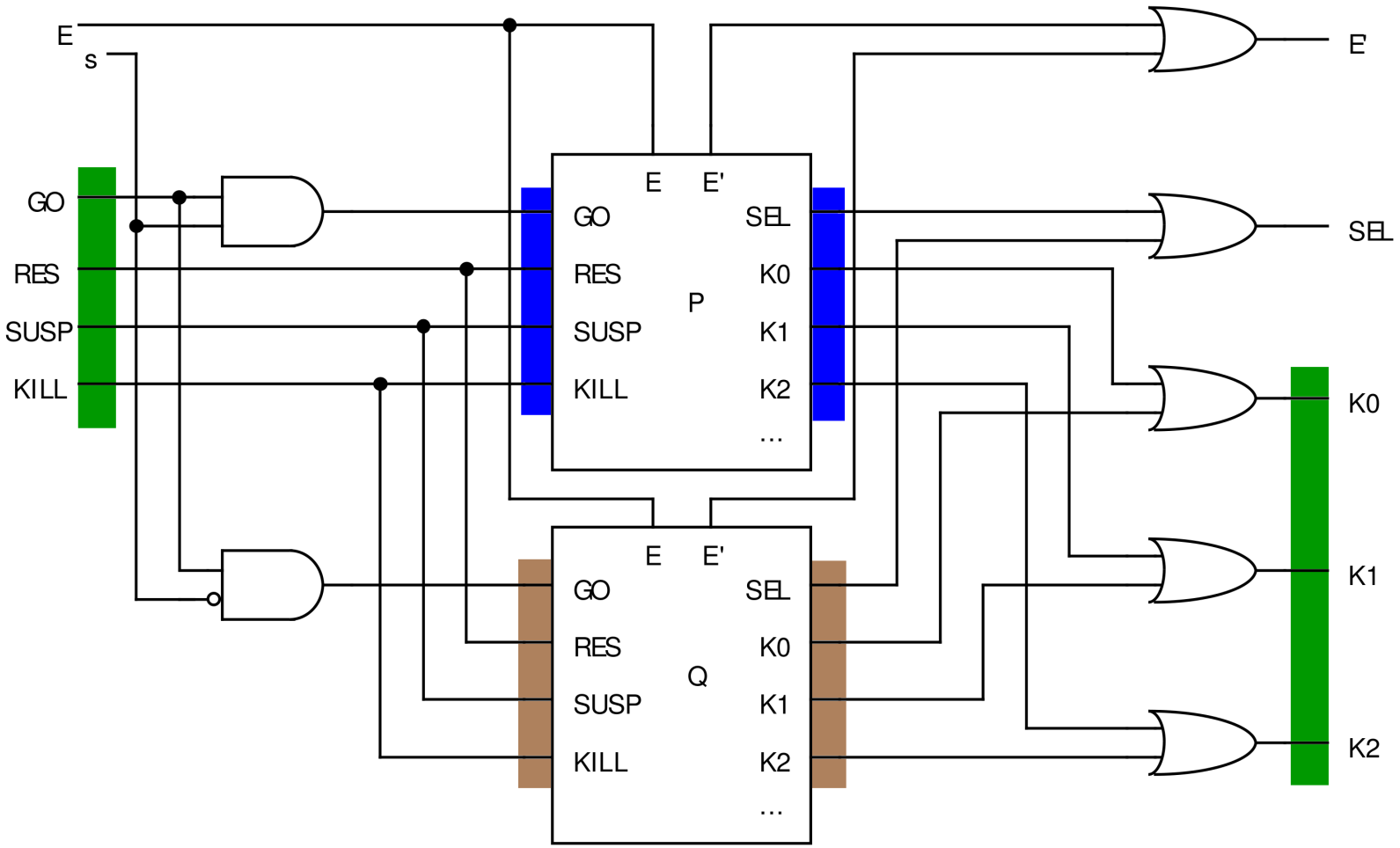
Traduction en circuit de `if` *s* `then` *p* `else` *q* `end`

`if` *s* `then` *P* `else` *Q* `end`



Traduction en circuit de `if` *s* `then` *p* `else` *q* `end`

`if` *s* `then` *P* `else` *Q* `end`



Règles micro-pas pour if-then

$\square$ = entrée $\qquad$ $\circ$ = sortie

$$\frac{s^b \in E \qquad (\mathrm{Go}\,\square) < (\mathrm{Go}\,\square) \wedge b \qquad \square = \square[\mathrm{Go} \leftarrow (\mathrm{Go}\,\square) \wedge b]}{\square(s\,?\,(\square p\circ)\,,\,(\square q\circ))\circ \xrightarrow[E]{E',k} \square(s\,?\,(\square p\circ)\,,\,(\square q\circ))\circ}$$

$$\frac{\square p\circ \xrightarrow[E]{E',k} \square p'\circ}{\square(s\,?\,(\square p\circ)\,,\,(\square q\circ))\circ \xrightarrow[E]{E',k} \square(s\,?\,(\square p'\circ)\,,\,(\square q\circ))\circ}$$

$$\frac{\circ < (\circ \vee \circ)}{\square(s\,?\,(\square p\circ)\,,\,(\square q\circ))\circ \xrightarrow[E]{E',k} \square(s\,?\,(\square p\circ)\,,\,(\square q\circ))(\circ \vee \circ)}$$

Premier instant d'ABRO en microsteps

```
loop
  abort
       awimm A
    ||
      awimm B
    ;
    (emit O ;
     halt)
  when R
end
```

$E_1 = \{ A^-, B^+, R^-, O^\perp \}$

Premier instant d'ABRO en microsteps

$\square$ loop
  $\square$ abort
    $\square \left\{ \; \square [\; (\square$ awimm $A \circ_{\{0,1\}})$
      ||
      $(\square$ awimm $B \circ_{\{0,1\}})$
    $]\circ_{\{0,1\}}$;
    $\square [\, (\square$ emit $O \circ_{\{0\}})$;
      $(\square$ halt $\circ_{\{1\}})$
    $] \circ_{\{1\}}$
    $\right\} \circ_{\{1\}}$
  when $R \circ_{\{0,1\}}$
end $\circ_{\{1\}}$

$E_1 = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^\perp \}$

```
loop
  abort
      awimm A
    ||
      awimm B
    ;
    (emit O ;
     halt)
  when R
end
```

Premier instant d'ABRO en microsteps

$\square$: Sel^{-}, Go^{+}

$\square$ loop

$\square$ abort

$\square \left\{ \right.$ $\square$ [($\square$ awimm A $\circ_{\{0,1\}}$)

||

($\square$ awimm B $\circ_{\{0,1\}}$)

]$\circ_{\{0,1\}}$;

$\square$ [($\square$ emit O $\circ_{\{0\}}$) ;

($\square$ halt $\circ_{\{1\}}$)

] $\circ_{\{1\}}$

$\left. \right\} \circ_{\{1\}}$

when R $\circ_{\{0,1\}}$

end $\circ_{\{1\}}$

```
loop
  abort
       awimm A
    ||
      awimm B
    ;
    (emit O ;
     halt)
  when R
end
```

$E_1 = \{ \mathrm{A}^{-}, \mathrm{B}^{+}, \mathrm{R}^{-}, \mathrm{O}^{\perp} \}$

Premier instant d'ABRO en microsteps

$\square$: Sel^{-}, Go^{+}

$\square$ loop

$\square$ abort

$\square \left\{ \; \square \left[\; (\square \; \texttt{awimm}\ A \circ_{\{0,1\}}) \right.\right.$

$\left. \left. \; || \; (\square \; \texttt{awimm}\ B \circ_{\{0,1\}}) \right] \circ_{\{0,1\}} ; \right.$

$\square \left[\; (\square \; \texttt{emit}\ O \circ_{\{0\}}) \; ; \; (\square \; \texttt{halt} \circ_{\{1\}}) \right] \circ_{\{1\}}$

$\left. \right\} \circ_{\{1\}}$

when $R \circ_{\{0,1\}}$

end $\circ_{\{1\}}$

$E_1 = \{ \mathrm{A}^{-}, \mathrm{B}^{+}, \mathrm{R}^{-}, \mathrm{O}^{\perp} \}$

```
loop
  abort
       awimm A
    ||
      awimm B
    ;
    (emit O ;
     halt)
  when R
end
```

Premier instant d'ABRO en microsteps

$\square$: Sel^{-}, Go^{+}

$\square$ loop

$\square$ abort

$\square$ $\{$ $\square$ [($\square$ awimm A $\circ_{\{0,1\}}$)

||

($\square$ awimm B $\circ_{\{0,1\}}$)

]$\circ_{\{0,1\}}$;

$\square$ [($\square$ emit O $\circ_{\{0\}}$) ;

($\square$ halt $\circ_{\{1\}}$)

] $\circ_{\{1\}}$

$\}$ $\circ_{\{1\}}$

when R $\circ_{\{0,1\}}$

end $\circ_{\{1\}}$

```
loop
  abort
       awimm A
    ||
      awimm B
    ;
    (emit O ;
     halt)
  when R
end
```

$E_1 = \{ \mathrm{A}^{-}, \mathrm{B}^{+}, \mathrm{R}^{-}, \mathrm{O}^{\perp} \}$

Premier instant d'ABRO en microsteps

$\square$: $\mathrm{Sel}^-, \mathrm{Go}^+$

$\square$ loop
  $\square$ abort
    $\square$ $\{$ $\square$ [($\square$ awimm A $\circ_{\{1\}}$)
      ||
      ($\square$ awimm B $\circ_{\{0,1\}}$)
      ]$\circ_{\{0,1\}}$;
      $\square$ [($\square$ emit O $\circ_{\{0\}}$) ;
      ($\square$ halt $\circ_{\{1\}}$)
      ] $\circ_{\{1\}}$
    $\}$ $\circ_{\{1\}}$
  when R $\circ_{\{0,1\}}$
end $\circ_{\{1\}}$

$E_1 = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^\perp \}$

```
loop
  abort
      awimm A
    ||
      awimm B
  ;
  (emit O ;
   halt)
  when R
end
```

Premier instant d'ABRO en microsteps

$\square$: Sel^{-}, Go^{+}

$\square$ loop
  $\square$ abort
    $\square$ $\{$ $\square$ [($\square$ awimm A $\circ_{\{1\}}$)
      ||
      ($\square$ awimm B $\circ_{\{0,1\}}$)
      ]$\circ_{\{0,1\}}$;
      $\square$ [($\square$ emit O $\circ_{\{0\}}$) ;
        ($\square$ halt $\circ_{\{1\}}$)
        ] $\circ_{\{1\}}$
    $\}$ $\circ_{\{1\}}$
  when R $\circ_{\{1\}}$
end $\circ_{\{1\}}$

```
loop
  abort
      awimm A
    ||
      awimm B
    ;
    (emit O ;
     halt)
  when R
end
```

$E_1 = \{ \mathrm{A}^{-}, \mathrm{B}^{+}, \mathrm{R}^{-}, \mathrm{O}^{\perp} \}$

Premier instant d'ABRO en microsteps

□ : $\mathrm{Sel}^{-}, \mathrm{Go}^{+}$

□ loop
  □ abort
    □ { □ [(□ awimm A $\circ_{\{1\}}$)
          ||
          (□ awimm B $\circ_{\{0,1\}}$)
        ]$\circ_{\{0,1\}}$;
        □ [(□ emit O $\circ_{\{0\}}$) ;
          (□ halt $\circ_{\{1\}}$)
        ] $\circ_{\{1\}}$
    } $\circ_{\{1\}}$
    when R $\circ_{\{1\}}$
  end $\circ_{\{1\}}$

```
loop
  abort
       awimm A
    ||
      awimm B
    ;
    (emit O ;
     halt)
  when R
end
```

$E_1 = \{ \mathrm{A}^{-}, \mathrm{B}^{+}, \mathrm{R}^{-}, \mathrm{O}^{\perp} \}$

Premier instant d'ABRO en microsteps

$\square$: Sel^{-}, Go^{+}

$\square$ loop
  $\square$ abort
    $\square$ $\{$ $\square$ [($\square$ awimm A $\circ_{\{1\}}$)
      ||
      ($\square$ awimm B $\circ_{\{0,1\}}$)
      ]$\circ_{\{0,1\}}$;
      $\square$ [($\square$ emit O $\circ_{\{0\}}$) ;
        ($\square$ halt $\circ_{\{1\}}$)
        ] $\circ_{\{1\}}$
    $\}$ $\circ_{\{1\}}$
  when R $\circ_{\{1\}}$
end $\circ_{\{1\}}$

```
loop
  abort
      awimm A
    ||
      awimm B
    ;
    (emit O ;
     halt)
  when R
end
```

$E_1 = \{ \mathrm{A}^{-}, \mathrm{B}^{+}, \mathrm{R}^{-}, \mathrm{O}^{\perp} \}$

Premier instant d'ABRO en microsteps

$\square$: Sel^{-}, Go^{+}

$\square$ loop
  $\square$ abort
    $\square$ $\{$ $\square$ [($\square$ awimm A $\bullet_1$)
      ||
      ($\square$ awimm B $\circ_{\{0,1\}}$)
      ]$\circ_{\{0,1\}}$;
      $\square$ [($\square$ emit O $\circ_{\{0\}}$) ;
        ($\square$ halt $\circ_{\{1\}}$)
        ] $\circ_{\{1\}}$
    $\}$ $\circ_{\{1\}}$
  when R $\circ_{\{1\}}$
end $\circ_{\{1\}}$

$E_1 = \{ \mathrm{A}^{-}, \mathrm{B}^{+}, \mathrm{R}^{-}, \mathrm{O}^{\perp} \}$

```
loop
  abort
      awimm A
    ||
      awimm B
    ;
    (emit O ;
     halt)
  when R
end
```

Premier instant d'ABRO en microsteps

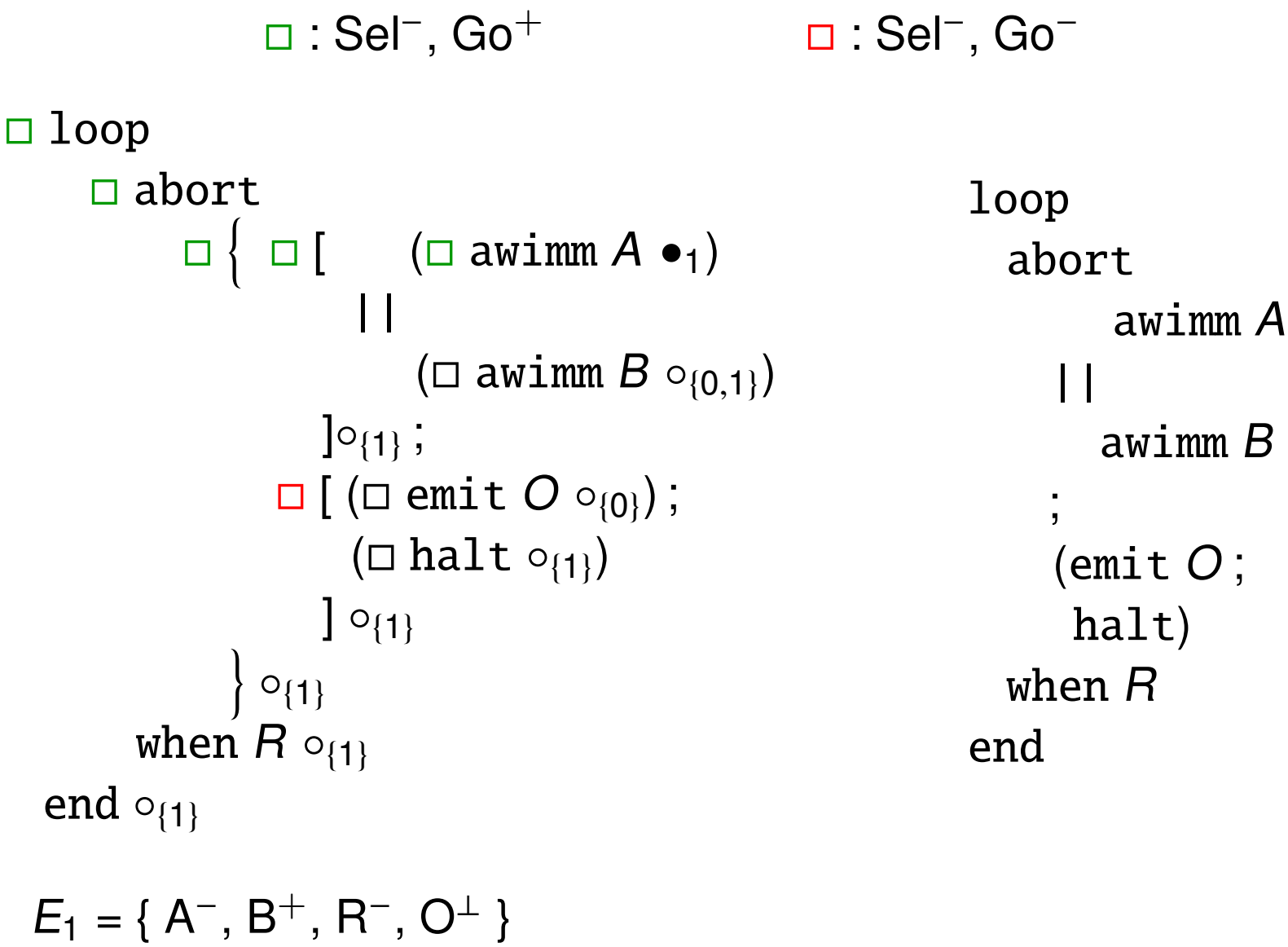
$\square$: Sel^{-}, Go^{+}

$\square$ loop
  $\square$ abort
    $\square$ { $\square$ [($\square$ awimm A $\bullet_1$)
      ||
      ($\square$ awimm B $\circ_{\{0,1\}}$)
      ]$\circ_{\{1\}}$;
      $\square$ [($\square$ emit O $\circ_{\{0\}}$) ;
        ($\square$ halt $\circ_{\{1\}}$)
        ] $\circ_{\{1\}}$
    } $\circ_{\{1\}}$
    when R $\circ_{\{1\}}$
  end $\circ_{\{1\}}$

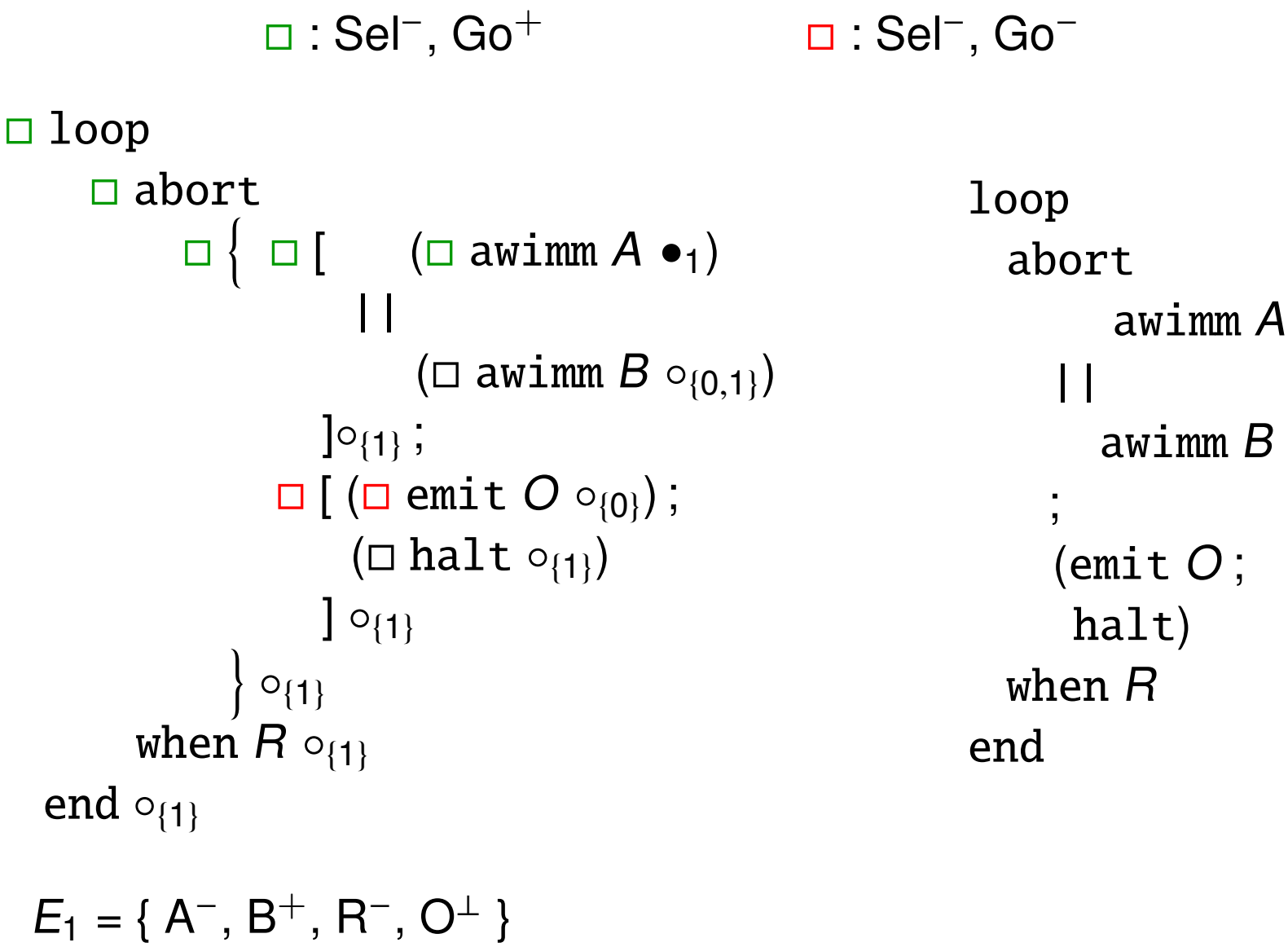
$E_1 = \{ \mathrm{A}^{-}, \mathrm{B}^{+}, \mathrm{R}^{-}, \mathrm{O}^{\perp} \}$

```
loop
  abort
      awimm A
    ||
      awimm B
    ;
    (emit O ;
     halt)
  when R
end
```

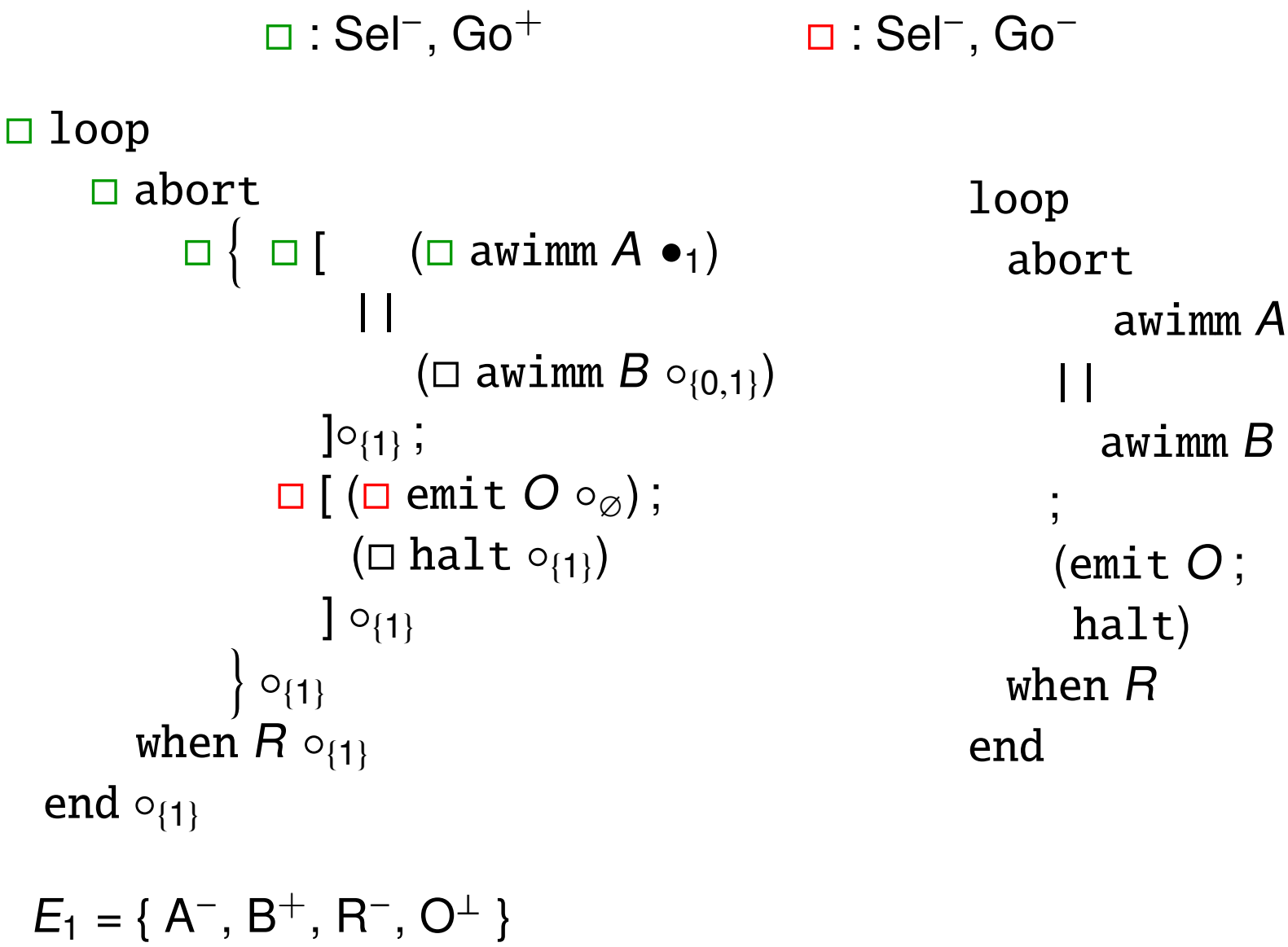
Premier instant d'ABRO en microsteps



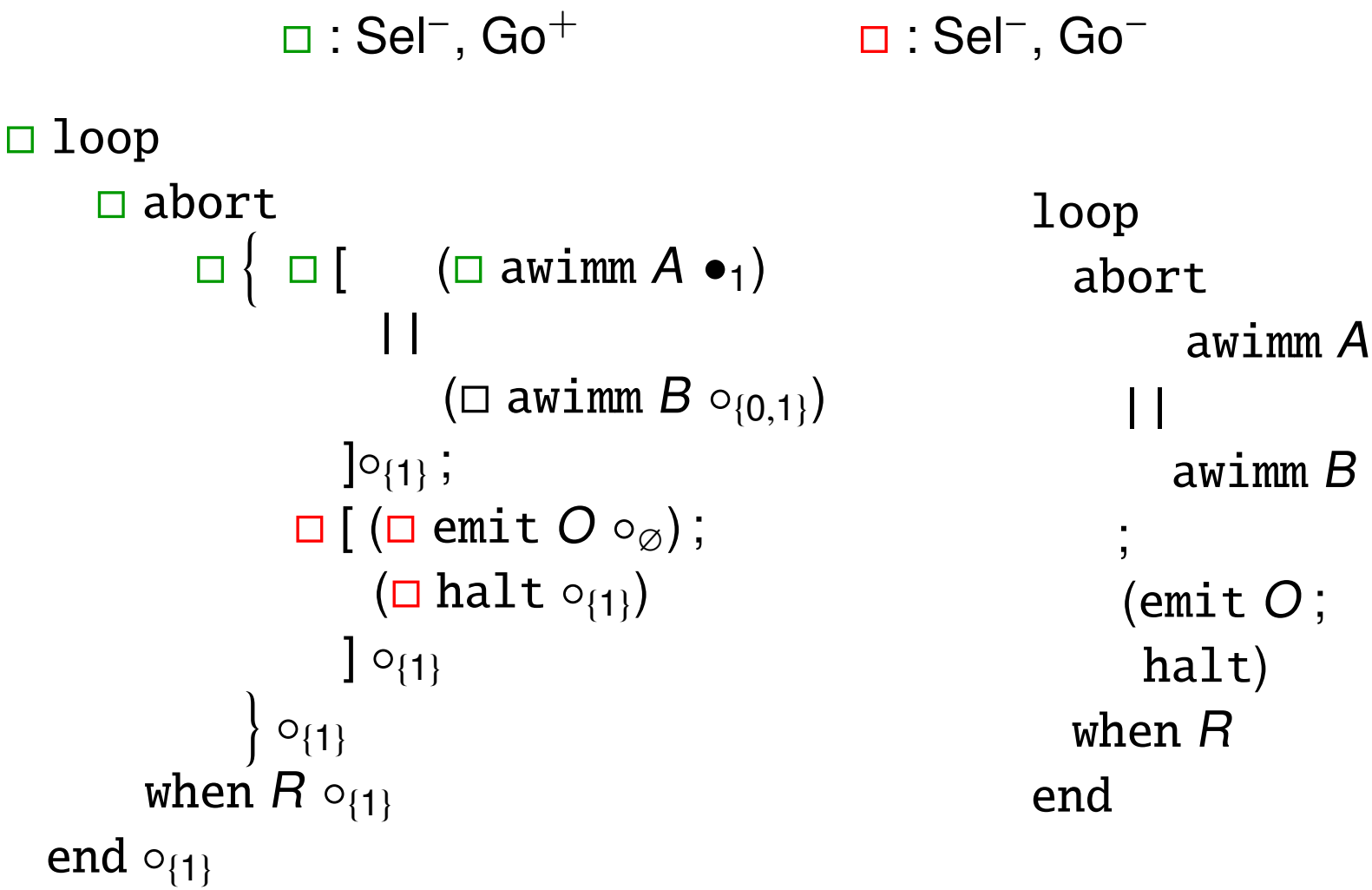
Premier instant d'ABRO en microsteps



Premier instant d'ABRO en microsteps

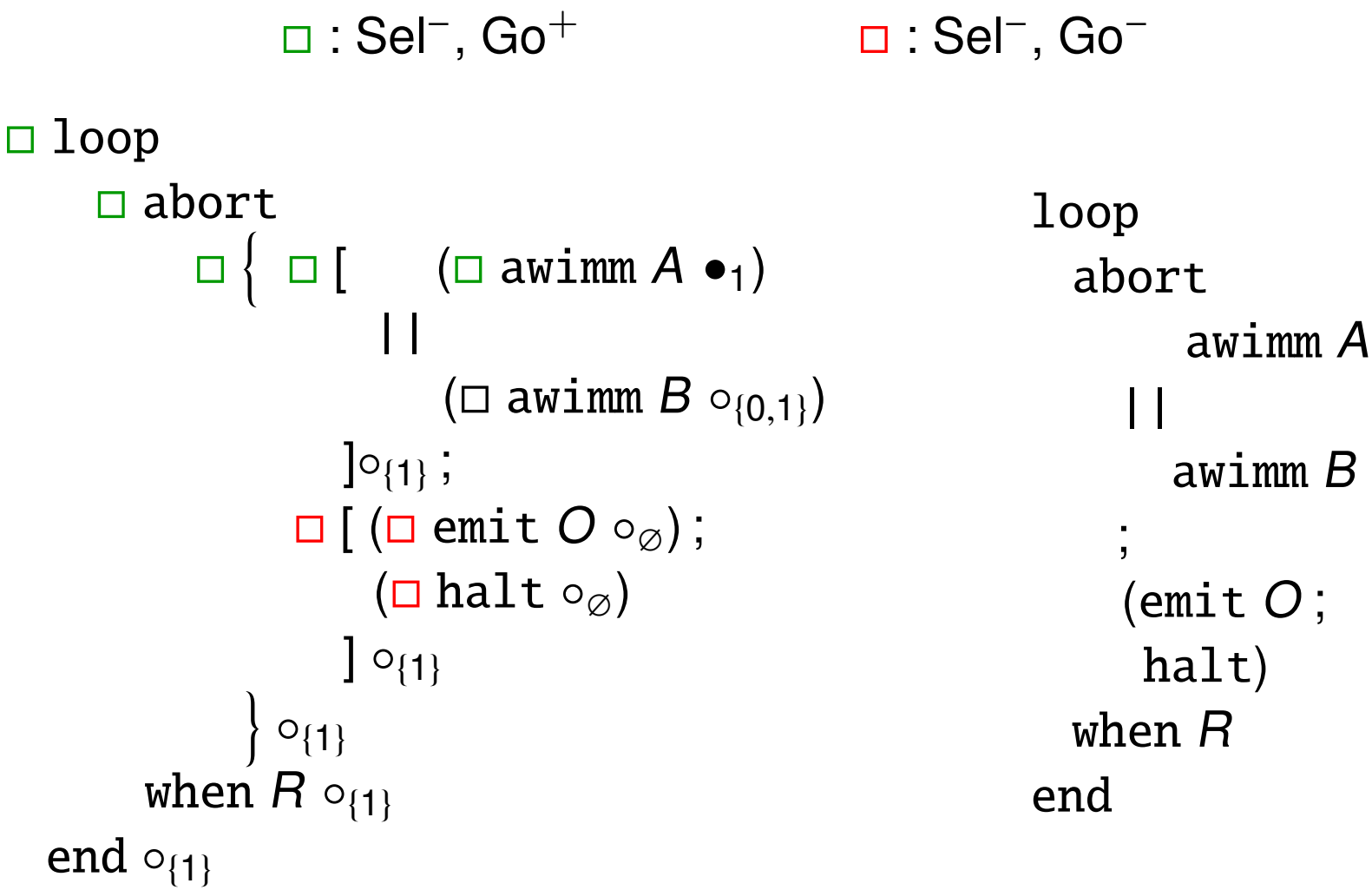


Premier instant d'ABRO en microsteps

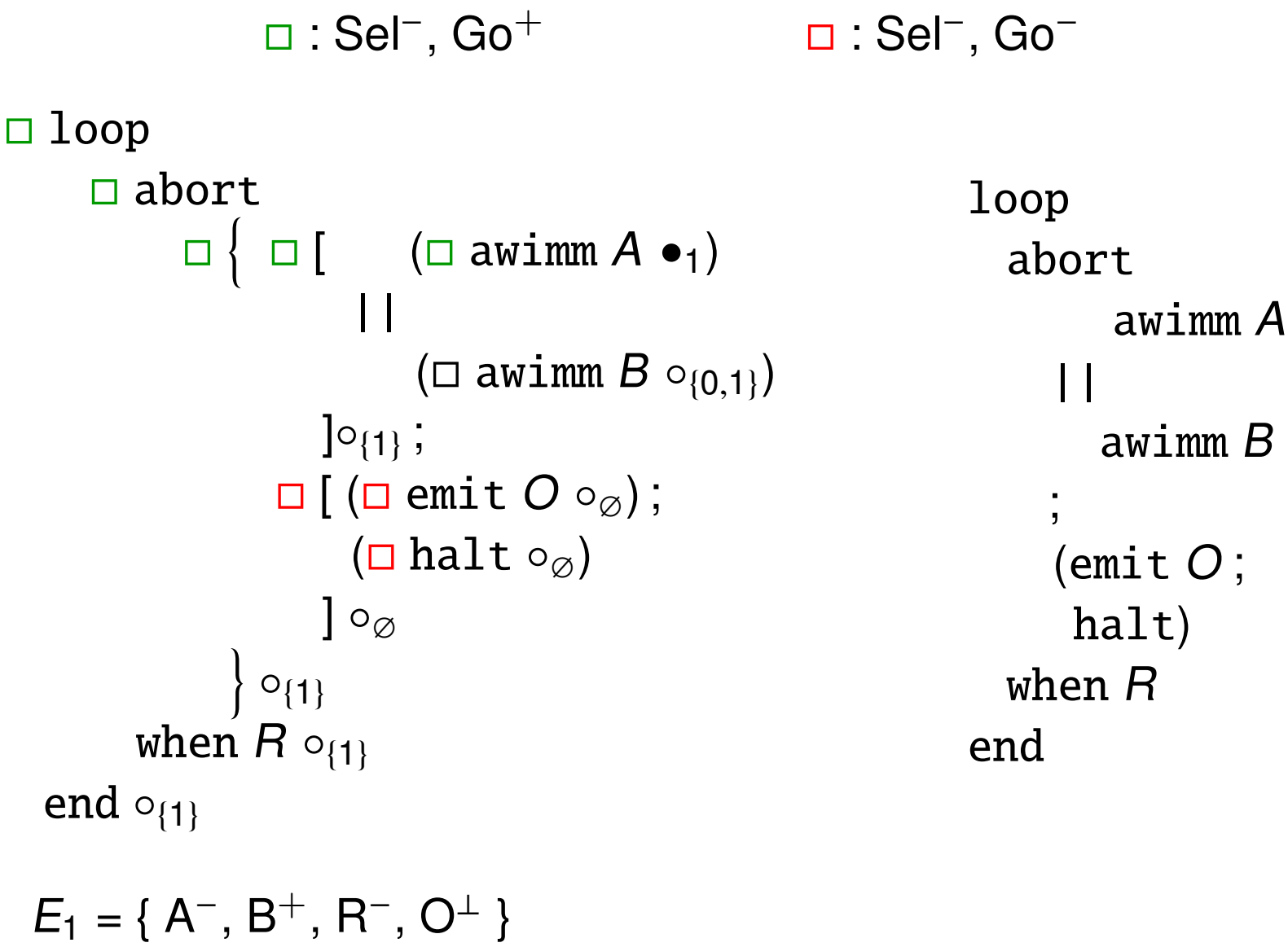


$E_1 = \{ A^-, B^+, R^-, O^\perp \}$

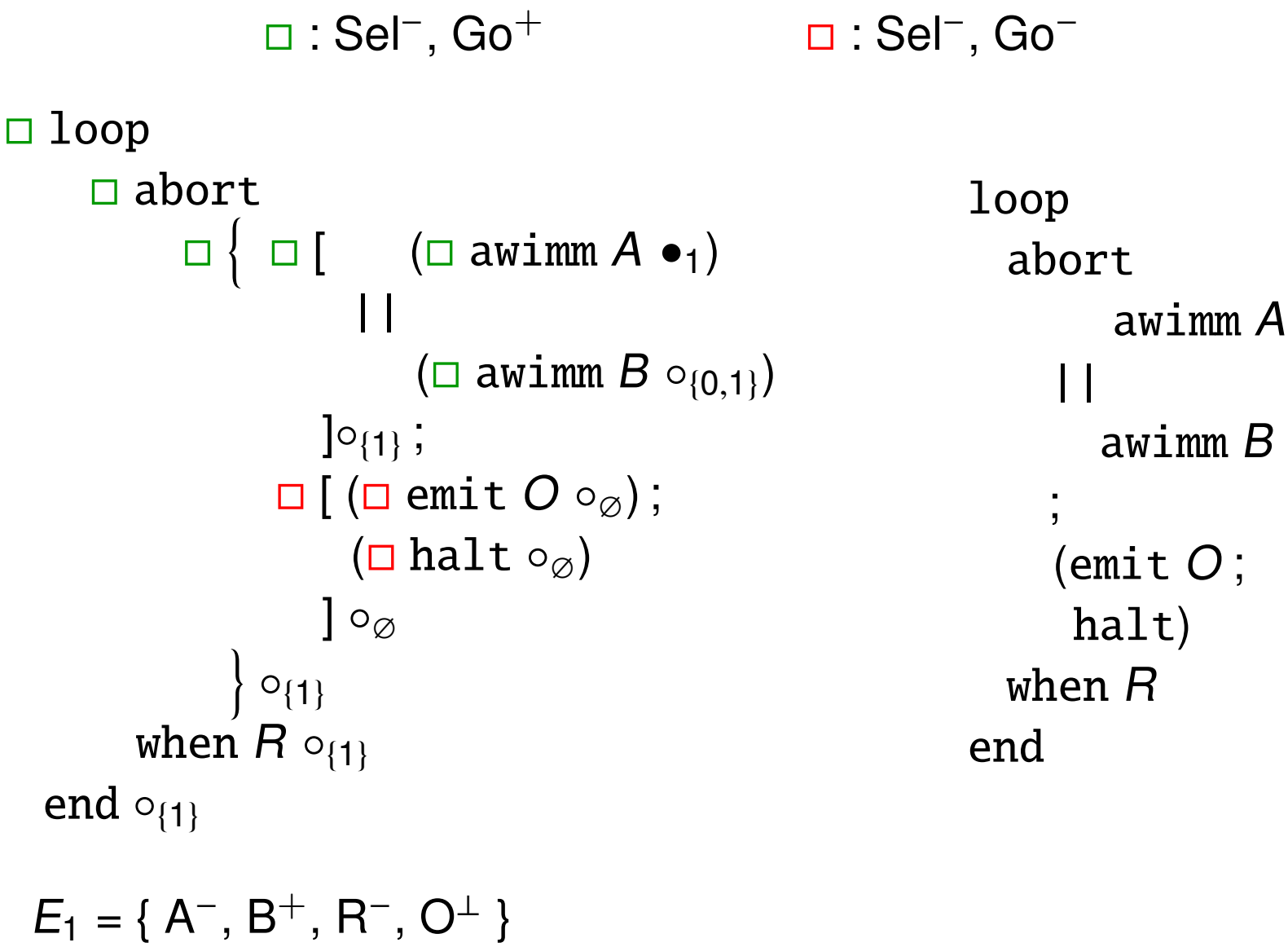
Premier instant d'ABRO en microsteps



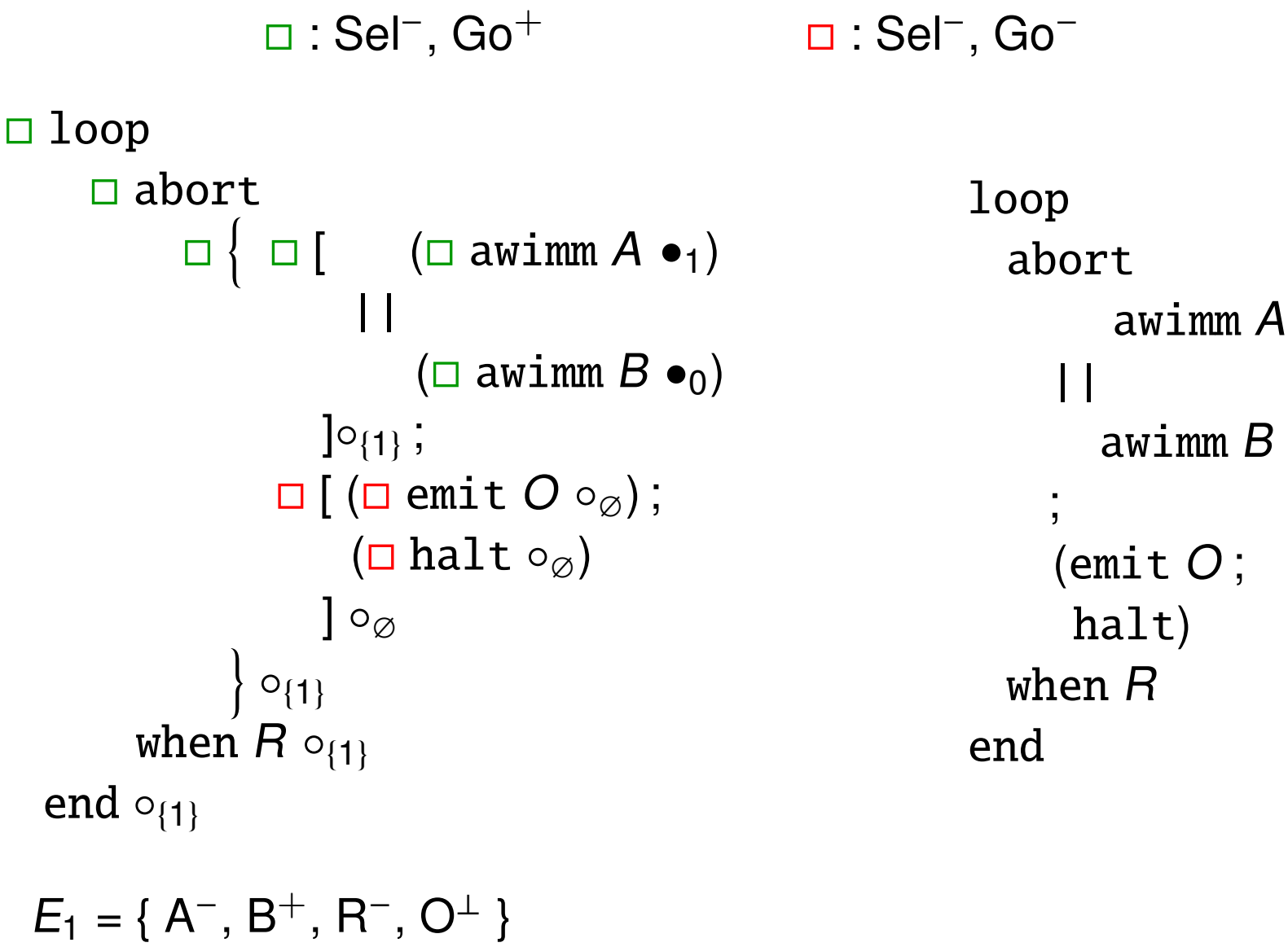
Premier instant d'ABRO en microsteps



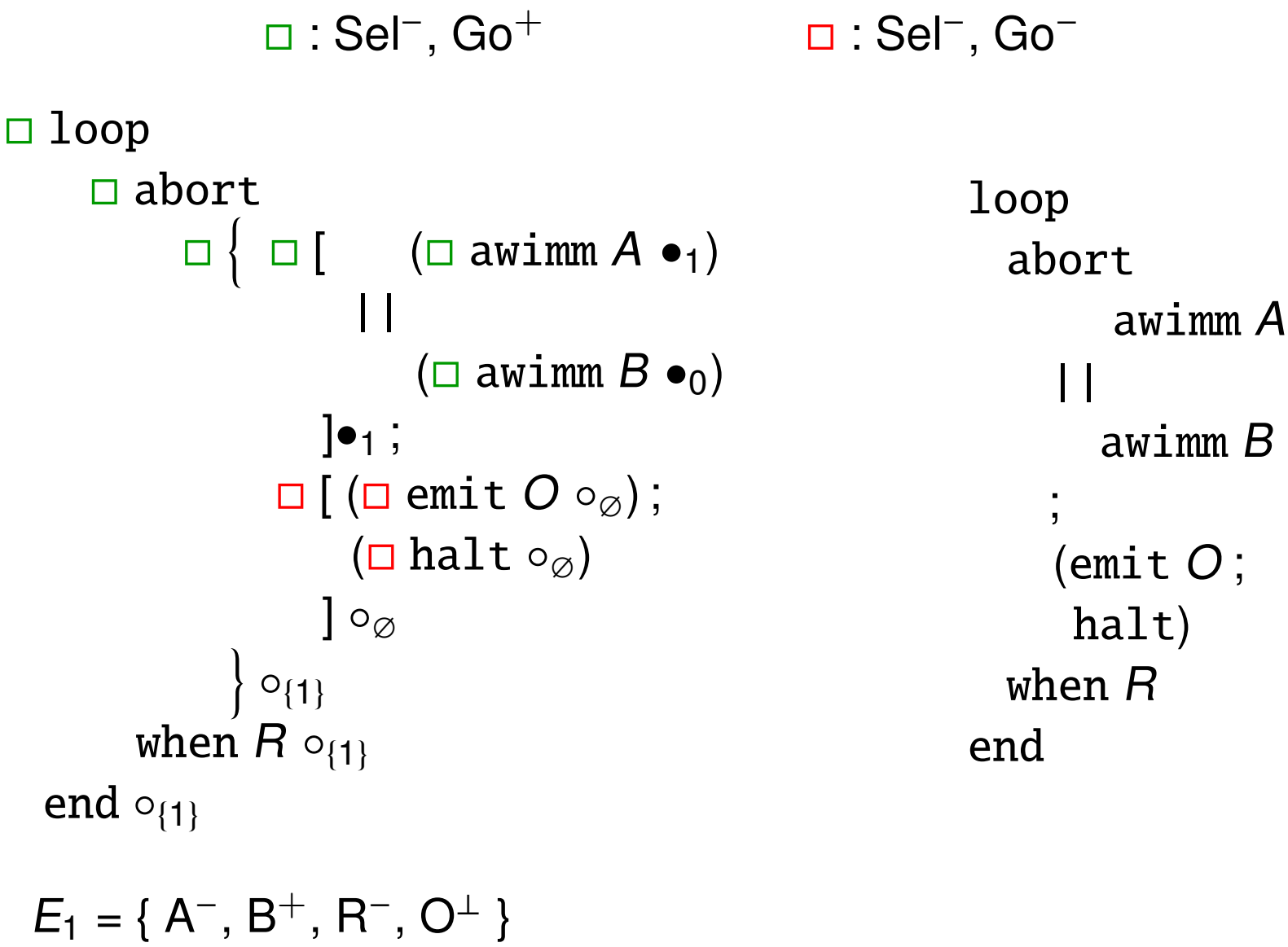
Premier instant d'ABRO en microsteps



Premier instant d'ABRO en microsteps



Premier instant d'ABRO en microsteps



Premier instant d'ABRO en microsteps

□ : $\mathrm{Sel}^-, \mathrm{Go}^+$

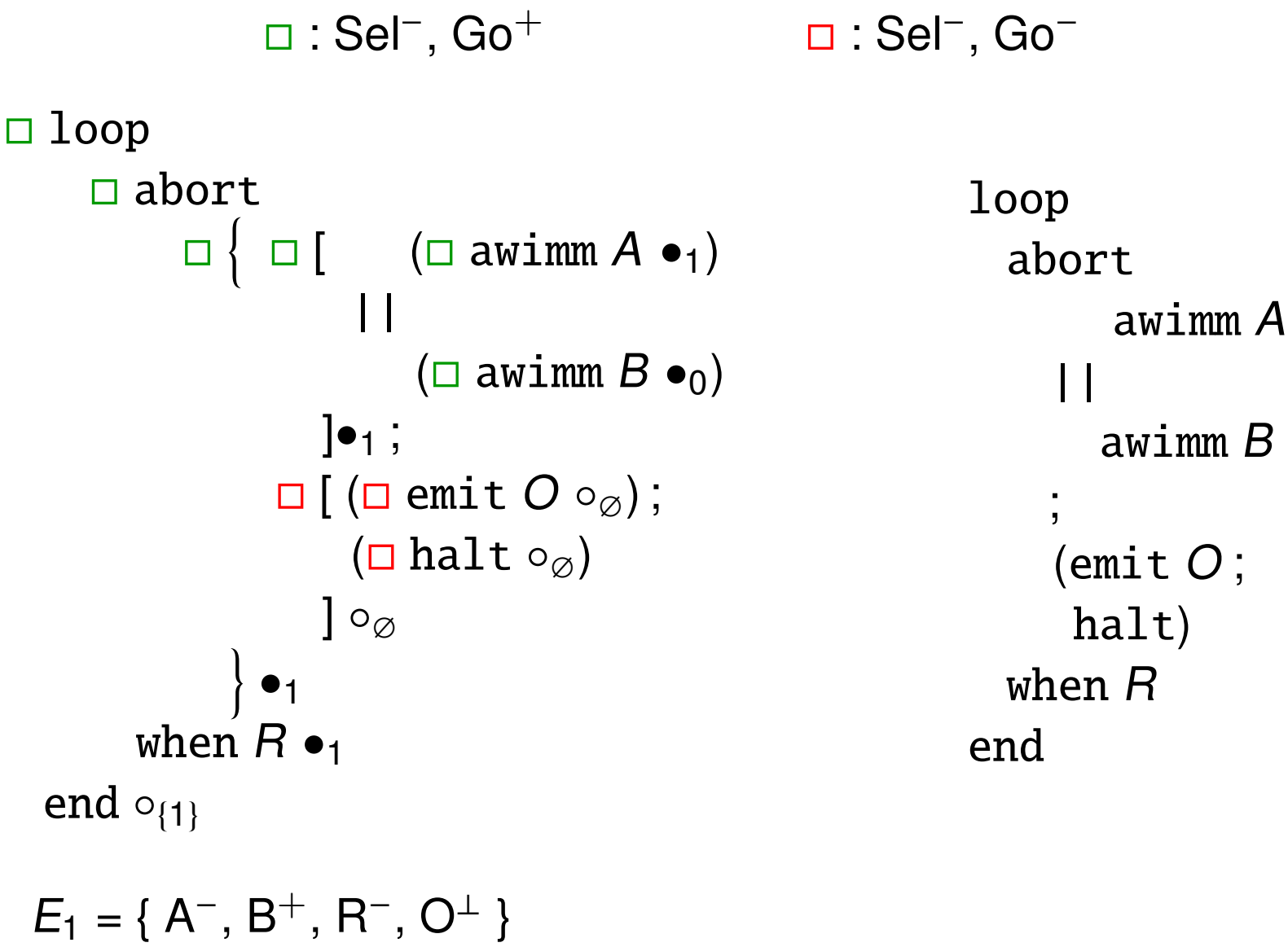
□ : $\mathrm{Sel}^-, \mathrm{Go}^-$

```
□ loop
    □ abort
        □ { □ [   (□ awimm A •1)
                ||
                  (□ awimm B •0)
              ]•1 ;
            □ [ (□ emit O ∘∅) ;
                (□ halt ∘∅)
              ] ∘∅
        } •1
      when R ∘{1}
  end ∘{1}
```

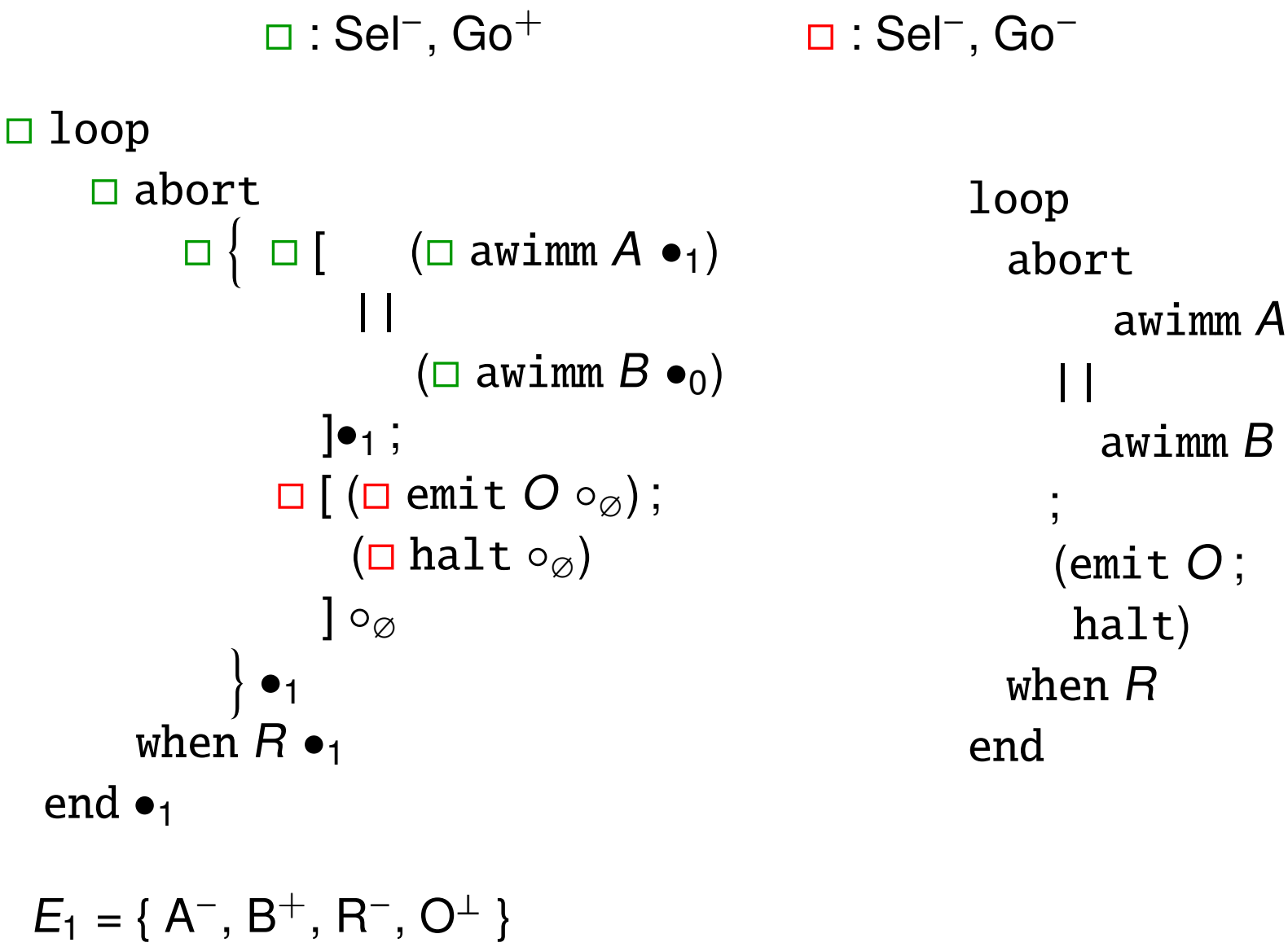
```
loop
  abort
       awimm A
    ||
      awimm B
  ;
  (emit O ;
   halt)
  when R
end
```

$E_1 = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^\perp \}$

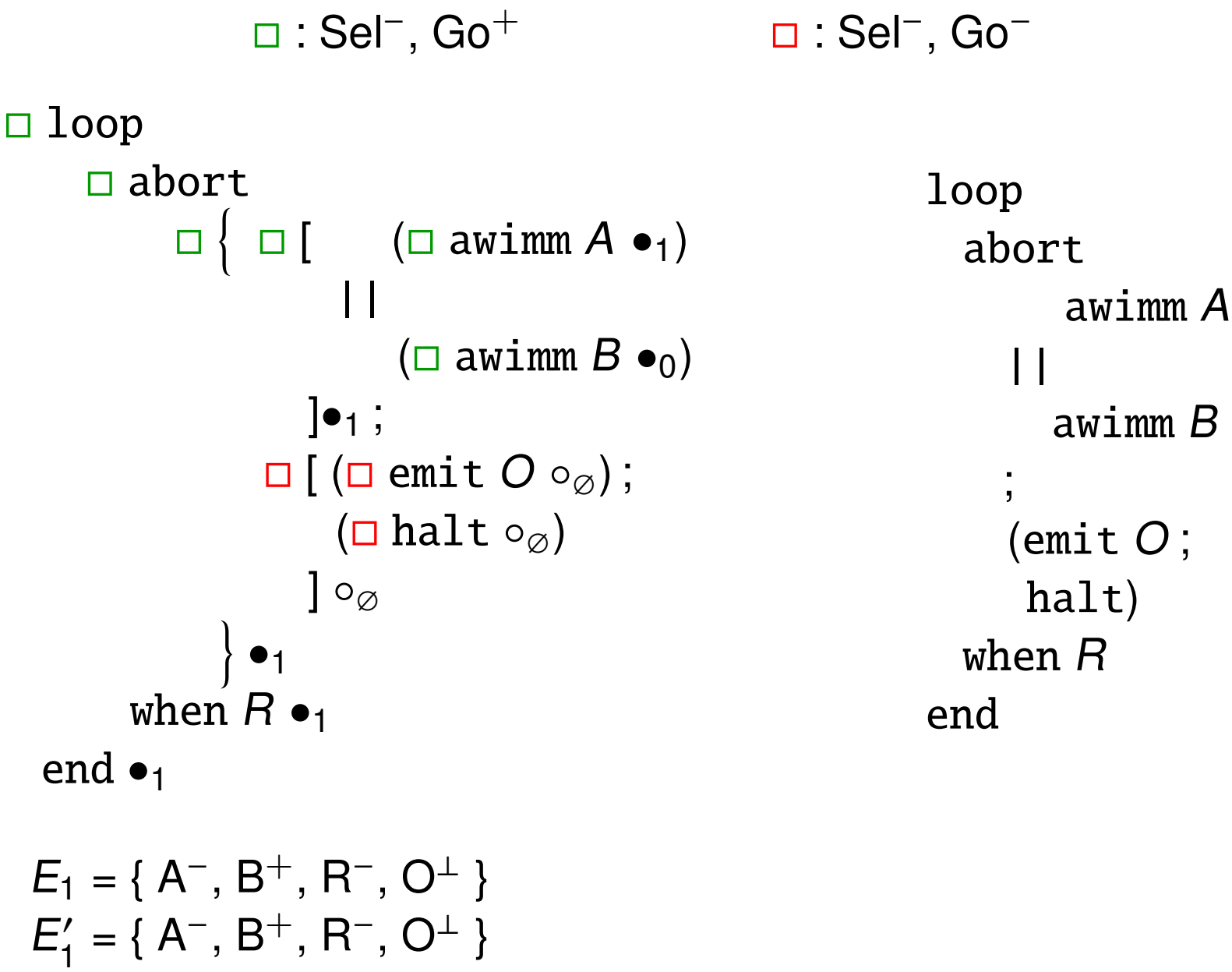
Premier instant d'ABRO en microsteps



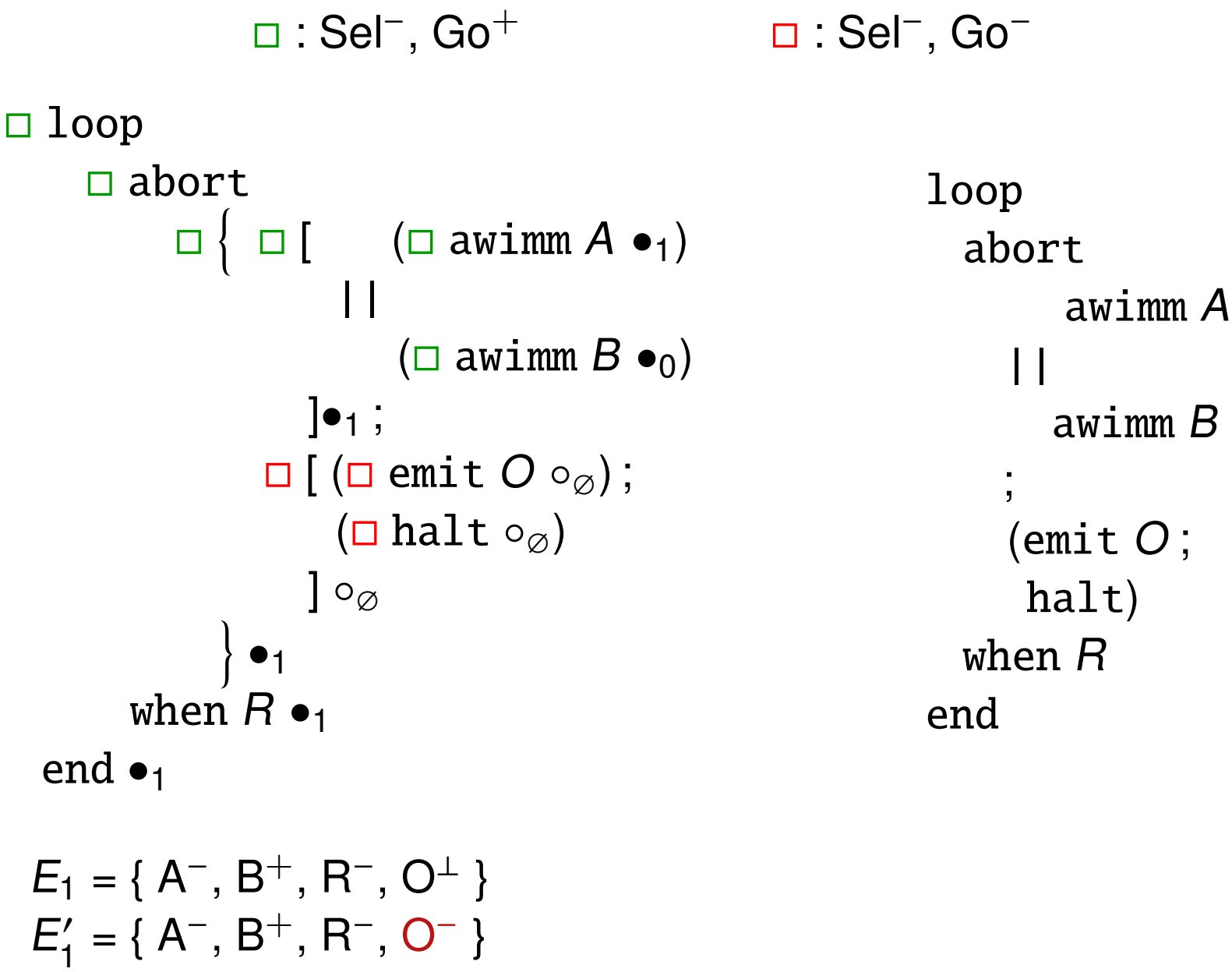
Premier instant d'ABRO en microsteps



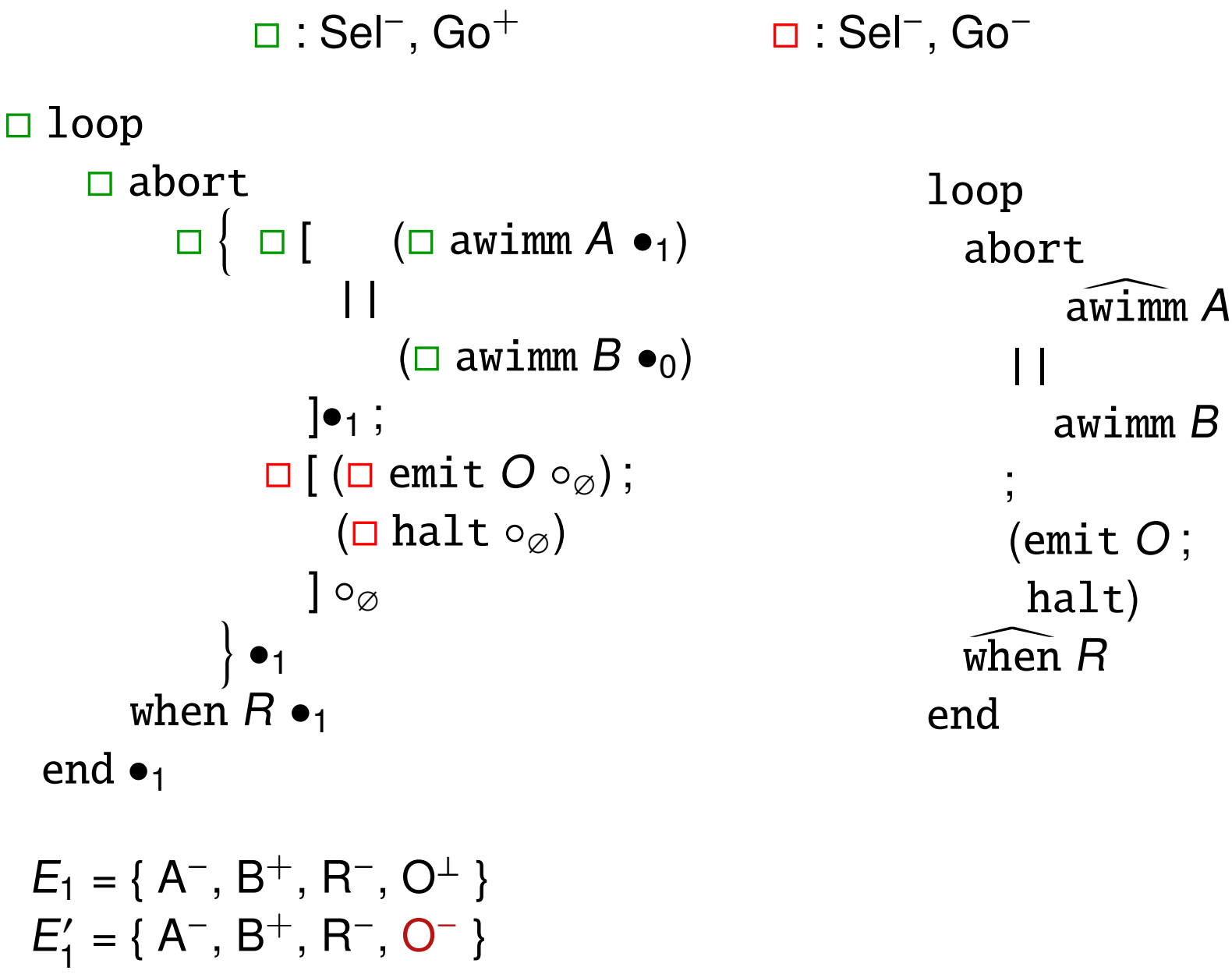
Premier instant d'ABRO en microsteps



Premier instant d'ABRO en microsteps



Premier instant d'ABRO en microsteps



Deuxième instant d'ABRO en microsteps

```
loop
  abort
       awimm A
    ||
      awimm B
    ;
    (emit O ;
     halt)
  when R
end
```

$$E_1 = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^\perp \} \qquad E_2 = \{ \mathrm{A}^+, \mathrm{B}^-, \mathrm{R}^-, \mathrm{O}^\perp \}$$
$$E_1' = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^- \} \quad \Longrightarrow$$

Deuxième instant d'ABRO en microsteps

```
loop
  abort
       awimm A
    ||
      awimm B
    ;
    (emit O ;
     halt)
  when R
end
```

$$E_1 = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^\perp \} \qquad E_2 = \{ \mathrm{A}^+, \mathrm{B}^-, \mathrm{R}^-, \mathrm{O}^\perp \}$$
$$E_1' = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^- \} \quad \Longrightarrow$$

Deuxième instant d'ABRO en microsteps

$\square$ loop
  $\square$ abort
    $\square \left\{ \; \square [\; (\square$ awimm $A \circ_{\{0,1\}})$
      ||
      $(\square$ awimm $B \circ_{\{0,1\}})$
    $]\circ_{\{0,1\}}$;
    $\square [\, (\square$ emit $O \circ_{\{0\}})$;
      $(\square$ halt $\circ_{\{1\}})$
    $] \circ_{\{1\}}$
    $\right\} \circ_{\{1\}}$
  when $R \circ_{\{0,1\}}$
end $\circ_{\{1\}}$

```
loop
  abort
      awimm A
    ||
      awimm B
    ;
    (emit O ;
     halt)
  when R
end
```

$E_1 = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^\perp \}$
$E_1' = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^- \}$
$\Longrightarrow$ $E_2 = \{ \mathrm{A}^+, \mathrm{B}^-, \mathrm{R}^-, \mathrm{O}^\perp \}$

Deuxième instant d'ABRO en microsteps

- : Sel^+, Go^-, Res^+
- loop

$$
\begin{array}{l}
\square\ \texttt{abort} \\
\quad \square \left\{ \begin{array}{l} \square\ [\quad (\square\ \texttt{awimm}\ A \circ_{\{0,1\}}) \\ \quad\ ||\ \\ \quad\quad (\square\ \texttt{awimm}\ B \circ_{\{0,1\}}) \\ \quad]\circ_{\{0,1\}}\ ; \\ \quad \square\ [\ (\square\ \texttt{emit}\ O \circ_{\{0\}})\ ; \\ \quad\quad (\square\ \texttt{halt} \circ_{\{1\}}) \\ \quad]\ \circ_{\{1\}} \end{array} \right\} \circ_{\{1\}} \\
\quad \texttt{when}\ R \circ_{\{0,1\}} \\
\texttt{end} \circ_{\{1\}}
\end{array}
$$

```
loop
  abort
      awimm A
    ||
      awimm B
    ;
    (emit O ;
     halt)
  when R
end
```

$$
\begin{array}{l} E_1 = \{\ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^\perp\ \} \\ E_1' = \{\ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^-\ \} \end{array} \Longrightarrow \quad E_2 = \{\ \mathrm{A}^+, \mathrm{B}^-, \mathrm{R}^-, \mathrm{O}^\perp\ \}
$$

Deuxième instant d'ABRO en microsteps

- : Sel^+, Go^-, Res^+
- loop
  - abort

$$\square \left\{ \square \left[\quad (\square\ \texttt{awimm}\ A \circ_{\{0,1\}}) \right. \right.$$
$$\left. \left. \quad || \quad (\square\ \texttt{awimm}\ B \circ_{\{0,1\}}) \right] \circ_{\{0,1\}} ; \right.$$
$$\left. \square \left[(\square\ \texttt{emit}\ O \circ_{\{0\}}) ; \right. \right.$$
$$\left. \left. \quad (\square\ \texttt{halt} \circ_{\{1\}}) \right] \circ_{\{1\}} \right\} \circ_{\{1\}}$$

    when R $\circ_{\{0,1\}}$

  end $\circ_{\{1\}}$

```
loop
  abort
      awimm A
    ||
      awimm B
    ;
    (emit O ;
     halt)
  when R
end
```

$$E_1 = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^\perp \} \qquad E_2 = \{ \mathrm{A}^+, \mathrm{B}^-, \mathrm{R}^-, \mathrm{O}^\perp \}$$
$$E_1' = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^- \} \implies$$

Deuxième instant d'ABRO en microsteps

- : Sel^+, Go^-, Res^+
- loop
  - abort
    - $\{$ $\square$ [($\square$ awimm A $\circ_{\{0,1\}}$) || ($\square$ awimm B $\circ_{\{0,1\}}$)]$\circ_{\{0,1\}}$; $\square$ [($\square$ emit O $\circ_{\{0\}}$) ; ($\square$ halt $\circ_{\{1\}}$)] $\circ_{\{1\}}$ $\}$ $\circ_{\{1\}}$
    - when R $\circ_{\{0,1\}}$

  end $\circ_{\{1\}}$

```
loop
  abort
      awimm A
    ||
      awimm B
    ;
    (emit O ;
     halt)
  when R
end
```

$E_1 = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^\perp \}$
$E_1' = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^- \}$
$\Longrightarrow$ $E_2 = \{ \mathrm{A}^+, \mathrm{B}^-, \mathrm{R}^-, \mathrm{O}^\perp \}$

Deuxième instant d'ABRO en microsteps

- : Sel^+, Go^-, Res^+
- loop
  - abort
    - $\{$ [$(\square$ awimm $A \circ_{\{0,1\}})$
      ||
      $(\square$ awimm $B \circ_{\{0,1\}})$
      $]\circ_{\{0,1\}}$;
      $\square$ [$(\square$ emit $O \circ_{\{0\}})$;
      $(\square$ halt $\circ_{\{1\}})$
      $] \circ_{\{1\}}$
      $\}\circ_{\{1\}}$
    - when $R \circ_{\{0,1\}}$

  end $\circ_{\{1\}}$

```
loop
  abort
      awimm A
    ||
      awimm B
    ;
    (emit O ;
     halt)
  when R
end
```

$E_1 = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^\perp \}$
$E_1' = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^- \}$
$\Longrightarrow$ $E_2 = \{ \mathrm{A}^+, \mathrm{B}^-, \mathrm{R}^-, \mathrm{O}^\perp \}$

Deuxième instant d'ABRO en microsteps

- : Sel^+, Go^-, Res^+
- loop
  - abort
    - $\{$ [(awimm A $\circ_{\{0,1\}}$)
      || ($\square$ awimm B $\circ_{\{0,1\}}$)
      ]$\circ_{\{0,1\}}$;
      $\square$ [($\square$ emit O $\circ_{\{0\}}$) ;
      ($\square$ halt $\circ_{\{1\}}$)
      ] $\circ_{\{1\}}$
      $\}$ $\circ_{\{1\}}$
    - when R $\circ_{\{0,1\}}$

  end $\circ_{\{1\}}$

```
loop
  abort
      awimm A
    ||
      awimm B
    ;
    (emit O ;
     halt)
  when R
end
```

$E_1 = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^\perp \}$
$E_1' = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^- \}$
$\Longrightarrow$ $E_2 = \{ \mathrm{A}^+, \mathrm{B}^-, \mathrm{R}^-, \mathrm{O}^\perp \}$

Deuxième instant d'ABRO en microsteps

■ : $\mathrm{Sel}^+, \mathrm{Go}^-, \mathrm{Res}^+$ □ : $\mathrm{Sel}^-, \mathrm{Go}^-, \mathrm{Res}^+$

■ loop

■ abort

■ { ■ [(■ awimm A $\circ_{\{0,1\}}$)
||
(□ awimm B $\circ_{\{0,1\}}$)
]$\circ_{\{0,1\}}$;
□ [(□ emit O $\circ_{\{0\}}$) ;
(□ halt $\circ_{\{1\}}$)
] $\circ_{\{1\}}$
} $\circ_{\{1\}}$

when R $\circ_{\{0,1\}}$

end $\circ_{\{1\}}$

```
loop
  abort
      awimm A
    ||
      awimm B
    ;
    (emit O ;
     halt)
  when R
end
```

$E_1 = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^\perp \}$
$E_1' = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^- \}$
$\Longrightarrow$ $E_2 = \{ \mathrm{A}^+, \mathrm{B}^-, \mathrm{R}^-, \mathrm{O}^\perp \}$

Deuxième instant d'ABRO en microsteps

■ : $\mathrm{Sel}^+, \mathrm{Go}^-, \mathrm{Res}^+$ □ : $\mathrm{Sel}^-, \mathrm{Go}^-, \mathrm{Res}^+$

■ loop

■ abort

■ { ■ [(■ awimm A $\bullet_0$)

||

(□ awimm B $\circ_{\{0,1\}}$)

]$\circ_{\{0,1\}}$;

□ [(□ emit O $\circ_{\{0\}}$) ;

(□ halt $\circ_{\{1\}}$)

] $\circ_{\{1\}}$

} $\circ_{\{1\}}$

when R $\circ_{\{0,1\}}$

end $\circ_{\{1\}}$

```
loop
  abort
       awimm A
    ||
      awimm B
    ;
    (emit O ;
     halt)
  when R
end
```

$E_1 = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^\perp \}$
$E_1' = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^- \}$
$\Longrightarrow$ $E_2 = \{ \mathrm{A}^+, \mathrm{B}^-, \mathrm{R}^-, \mathrm{O}^\perp \}$

Deuxième instant d'ABRO en microsteps

■ : $\mathrm{Sel}^+, \mathrm{Go}^-, \mathrm{Res}^+$ □ : $\mathrm{Sel}^-, \mathrm{Go}^-, \mathrm{Res}^+$

■ loop

■ abort

■ { ■ [(■ awimm A $\bullet_0$)

||

(□ awimm B $\circ_\varnothing$)

]$\circ_{\{0,1\}}$;

□ [(□ emit O $\circ_{\{0\}}$) ;

(□ halt $\circ_{\{1\}}$)

] $\circ_{\{1\}}$

} $\circ_{\{1\}}$

when R $\circ_{\{0,1\}}$

end $\circ_{\{1\}}$

```
loop
  abort
      awimm A
    ||
      awimm B
    ;
    (emit O ;
     halt)
  when R
end
```

$E_1 = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^\perp \}$
$E_1' = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^- \}$
$\Longrightarrow$ $E_2 = \{ \mathrm{A}^+, \mathrm{B}^-, \mathrm{R}^-, \mathrm{O}^\perp \}$

Deuxième instant d'ABRO en microsteps

■ : $\mathrm{Sel}^+, \mathrm{Go}^-, \mathrm{Res}^+$ □ : $\mathrm{Sel}^-, \mathrm{Go}^-, \mathrm{Res}^+$

■ loop

■ abort

■ { ■ [(■ awimm A $\bullet_0$)

||

(□ awimm B $\circ_\varnothing$)

]$\bullet_0$;

□ [(□ emit O $\circ_{\{0\}}$) ;

(□ halt $\circ_{\{1\}}$)

] $\circ_{\{1\}}$

} $\circ_{\{1\}}$

when R $\circ_{\{0,1\}}$

end $\circ_{\{1\}}$

```
loop
  abort
      awimm A
    ||
      awimm B
    ;
    (emit O ;
     halt)
  when R
end
```

$E_1 = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^\perp \}$
$E_1' = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^- \}$
$\Longrightarrow$ $E_2 = \{ \mathrm{A}^+, \mathrm{B}^-, \mathrm{R}^-, \mathrm{O}^\perp \}$

Deuxième instant d'ABRO en microsteps

■ : $\mathrm{Sel}^+, \mathrm{Go}^-, \mathrm{Res}^+$ □ : $\mathrm{Sel}^-, \mathrm{Go}^-, \mathrm{Res}^+$ □ : $\mathrm{Sel}^-, \mathrm{Go}^+, \mathrm{Res}^+$

■ loop

■ abort

■ { ■ [(■ awimm A $\bullet_0$)
|| (□ awimm B $\circ_\varnothing$)
] $\bullet_0$;
□ [(□ emit O $\circ_{\{0\}}$) ;
(□ halt $\circ_{\{1\}}$)
] $\circ_{\{1\}}$
} $\circ_{\{1\}}$

when R $\circ_{\{0,1\}}$

end $\circ_{\{1\}}$

```
loop
  abort
      awimm A
  ||
     awimm B
  ;
  (emit O ;
   halt)
  when R
end
```

$E_1 = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^\perp \}$
$E_1' = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^- \}$
$\Longrightarrow$ $E_2 = \{ \mathrm{A}^+, \mathrm{B}^-, \mathrm{R}^-, \mathrm{O}^\perp \}$

Deuxième instant d'ABRO en microsteps

■ : $\mathrm{Sel}^+, \mathrm{Go}^-, \mathrm{Res}^+$ □ : $\mathrm{Sel}^-, \mathrm{Go}^-, \mathrm{Res}^+$ □ : $\mathrm{Sel}^-, \mathrm{Go}^+, \mathrm{Res}^+$

■ loop

■ abort

■ { ■ [(■ awimm A $\bullet_0$)
|| (□ awimm B $\circ_\varnothing$)
] $\bullet_0$;
□ [(□ emit O $\circ_{\{0\}}$) ;
(□ halt $\circ_{\{1\}}$)
] $\circ_{\{1\}}$
} $\circ_{\{1\}}$

when R $\circ_{\{0,1\}}$

end $\circ_{\{1\}}$

```
loop
  abort
      awimm A
    ||
      awimm B
    ;
    (emit O ;
     halt)
  when R
end
```

$E_1 = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^\perp \}$
$E_1' = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^- \}$
$\Longrightarrow$ $E_2 = \{ \mathrm{A}^+, \mathrm{B}^-, \mathrm{R}^-, \mathrm{O}^\perp \}$

Deuxième instant d'ABRO en microsteps

■ : $\mathrm{Sel}^+, \mathrm{Go}^-, \mathrm{Res}^+$ □ : $\mathrm{Sel}^-, \mathrm{Go}^-, \mathrm{Res}^+$ □ : $\mathrm{Sel}^-, \mathrm{Go}^+, \mathrm{Res}^+$

■ loop

■ abort

■ { ■ [(■ awimm A $\bullet_0$)
|| (□ awimm B $\circ_\varnothing$)
] $\bullet_0$;
□ [(□ emit O $\bullet_0$) ;
(□ halt $\circ_{\{1\}}$)
] $\circ_{\{1\}}$
} $\circ_{\{1\}}$

when R $\circ_{\{0,1\}}$

end $\circ_{\{1\}}$

```
loop
  abort
      awimm A
  ||
    awimm B
  ;
  (emit O ;
   halt)
  when R
end
```

$E_1 = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^\perp \}$
$E_1' = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^- \}$
$\Longrightarrow$ $E_2 = \{ \mathrm{A}^+, \mathrm{B}^-, \mathrm{R}^-, \mathrm{O}^\perp \}$

Deuxième instant d'ABRO en microsteps

■ : $\mathrm{Sel}^+, \mathrm{Go}^-, \mathrm{Res}^+$ □ : $\mathrm{Sel}^-, \mathrm{Go}^-, \mathrm{Res}^+$ □ : $\mathrm{Sel}^-, \mathrm{Go}^+, \mathrm{Res}^+$

■ loop

■ abort

■ { ■ [(■ awimm A $\bullet_0$)
|| (□ awimm B $\circ_\varnothing$)
] $\bullet_0$;
□ [(□ emit O $\bullet_0$) ;
(□ halt $\circ_{\{1\}}$)
] $\circ_{\{1\}}$
} $\circ_{\{1\}}$

when R $\circ_{\{0,1\}}$

end $\circ_{\{1\}}$

```
loop
  abort
      awimm A
  ||
    awimm B
  ;
  (emit O ;
   halt)
  when R
end
```

$E_1 = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^\perp \}$
$E_1' = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^- \}$
$\Longrightarrow$ $E_2 = \{ \mathrm{A}^+, \mathrm{B}^-, \mathrm{R}^-, \mathrm{O}^\perp \}$

Deuxième instant d'ABRO en microsteps

■ : $\mathrm{Sel}^+, \mathrm{Go}^-, \mathrm{Res}^+$ □ : $\mathrm{Sel}^-, \mathrm{Go}^-, \mathrm{Res}^+$ □ : $\mathrm{Sel}^-, \mathrm{Go}^+, \mathrm{Res}^+$

■ loop

■ abort

■ { ■ [(■ awimm A $\bullet_0$)

|| (□ awimm B $\circ_\varnothing$)

]$\bullet_0$; □ [(□ emit O $\bullet_0$) ; (□ halt $\bullet_1$)] $\circ_{\{1\}}$ } $\circ_{\{1\}}$

when R $\circ_{\{0,1\}}$

end $\circ_{\{1\}}$

```
loop
  abort
      awimm A
    ||
      awimm B
    ;
    (emit O ;
     halt)
  when R
end
```

$E_1 = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^\perp \}$
$E_1' = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^- \}$ $\Longrightarrow$ $E_2 = \{ \mathrm{A}^+, \mathrm{B}^-, \mathrm{R}^-, \mathrm{O}^\perp \}$

Deuxième instant d'ABRO en microsteps

■ : $\mathrm{Sel}^+, \mathrm{Go}^-, \mathrm{Res}^+$ □ : $\mathrm{Sel}^-, \mathrm{Go}^-, \mathrm{Res}^+$ □ : $\mathrm{Sel}^-, \mathrm{Go}^+, \mathrm{Res}^+$

■ loop

■ abort

■ { ■ [(■ awimm A $\bullet_0$)

|| (□ awimm B $\circ_\varnothing$)

]$\bullet_0$; □ [(□ emit O $\bullet_0$) ; (□ halt $\bullet_1$)] $\bullet_1$ } $\circ_{\{1\}}$

when R $\circ_{\{0,1\}}$

end $\circ_{\{1\}}$

```
loop
  abort
      awimm A
    ||
      awimm B
    ;
    (emit O ;
     halt)
  when R
end
```

$E_1 = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^\perp \}$
$E_1' = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^- \}$
$\Longrightarrow$ $E_2 = \{ \mathrm{A}^+, \mathrm{B}^-, \mathrm{R}^-, \mathrm{O}^\perp \}$

Deuxième instant d'ABRO en microsteps

■ : $\mathrm{Sel}^+, \mathrm{Go}^-, \mathrm{Res}^+$ □ : $\mathrm{Sel}^-, \mathrm{Go}^-, \mathrm{Res}^+$ □ : $\mathrm{Sel}^-, \mathrm{Go}^+, \mathrm{Res}^+$

■ loop

■ abort

■ { ■ [(■ awimm A $\bullet_0$)

|| (□ awimm B $\circ_\varnothing$)

]$\bullet_0$;

□ [(□ emit O $\bullet_0$) ;

(□ halt $\bullet_1$)

] $\bullet_1$

} $\bullet_1$

when R $\circ_{\{0,1\}}$

end $\circ_{\{1\}}$

```
loop
  abort
      awimm A
    ||
      awimm B
    ;
    (emit O ;
     halt)
  when R
end
```

$E_1 = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^\perp \}$
$E_1' = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^- \}$ $\Longrightarrow$ $E_2 = \{ \mathrm{A}^+, \mathrm{B}^-, \mathrm{R}^-, \mathrm{O}^\perp \}$

Deuxième instant d'ABRO en microsteps

■ : $\mathrm{Sel}^+, \mathrm{Go}^-, \mathrm{Res}^+$ □ : $\mathrm{Sel}^-, \mathrm{Go}^-, \mathrm{Res}^+$ □ : $\mathrm{Sel}^-, \mathrm{Go}^+, \mathrm{Res}^+$

■ loop

■ abort

■ { ■ [(■ awimm A $\bullet_0$)

|| (□ awimm B $\circ_\varnothing$)

] $\bullet_0$;

□ [(□ emit O $\bullet_0$) ;

(□ halt $\bullet_1$)

] $\bullet_1$

} $\bullet_1$

when R $\bullet_1$

end $\circ_{\{1\}}$

```
loop
  abort
      awimm A
    ||
      awimm B
    ;
    (emit O ;
     halt)
  when R
end
```

$E_1 = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^\perp \}$
$E_1' = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^- \}$ $\Longrightarrow$ $E_2 = \{ \mathrm{A}^+, \mathrm{B}^-, \mathrm{R}^-, \mathrm{O}^\perp \}$

Deuxième instant d'ABRO en microsteps

■ : $\mathrm{Sel}^+, \mathrm{Go}^-, \mathrm{Res}^+$ □ : $\mathrm{Sel}^-, \mathrm{Go}^-, \mathrm{Res}^+$ □ : $\mathrm{Sel}^-, \mathrm{Go}^+, \mathrm{Res}^+$

■ loop

■ abort

■ { ■ [(■ awimm A $\bullet_0$)

|| (□ awimm B $\circ_\varnothing$)

]$\bullet_0$;

□ [(□ emit O $\bullet_0$) ;

(□ halt $\bullet_1$)

] $\bullet_1$

} $\bullet_1$

when R $\bullet_1$

end $\bullet_1$

```
loop
  abort
      awimm A
    ||
      awimm B
    ;
    (emit O ;
     halt)
  when R
end
```

$E_1 = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^\perp \}$
$E_1' = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^- \}$ $\Longrightarrow$ $E_2 = \{ \mathrm{A}^+, \mathrm{B}^-, \mathrm{R}^-, \mathrm{O}^\perp \}$

Deuxième instant d'ABRO en microsteps

■ : $\mathrm{Sel}^+, \mathrm{Go}^-, \mathrm{Res}^+$ □ : $\mathrm{Sel}^-, \mathrm{Go}^-, \mathrm{Res}^+$ □ : $\mathrm{Sel}^-, \mathrm{Go}^+, \mathrm{Res}^+$

■ loop

■ abort

■ { ■ [(■ awimm A $\bullet_0$)

|| (□ awimm B $\circ_\varnothing$)

]$\bullet_0$;

□ [(□ emit O $\bullet_0$) ;

(□ halt $\bullet_1$)

] $\bullet_1$

} $\bullet_1$

when R $\bullet_1$

end $\bullet_1$

```
loop
  abort
      awimm A
    ||
      awimm B
    ;
    (emit O ;
     halt)
  when R
end
```

$E_1 = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^\perp \}$
$E'_1 = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^- \}$
$\Longrightarrow$
$E_2 = \{ \mathrm{A}^+, \mathrm{B}^-, \mathrm{R}^-, \mathrm{O}^\perp \}$
$E'_2 = \{ \mathrm{A}^+, \mathrm{B}^-, \mathrm{R}^-, \mathrm{O}^\perp \}$

Deuxième instant d'ABRO en microsteps

■ : $\mathrm{Sel}^+, \mathrm{Go}^-, \mathrm{Res}^+$ □ : $\mathrm{Sel}^-, \mathrm{Go}^-, \mathrm{Res}^+$ □ : $\mathrm{Sel}^-, \mathrm{Go}^+, \mathrm{Res}^+$

■ loop

■ abort

■ { ■ [(■ awimm A $\bullet_0$)
|| (□ awimm B $\circ_\varnothing$)
] $\bullet_0$;
□ [(□ emit O $\bullet_0$) ;
(□ halt $\bullet_1$)
] $\bullet_1$
} $\bullet_1$

when R $\bullet_1$

end $\bullet_1$

```
loop
  abort
      awimm A
    ||
      awimm B
    ;
    (emit O ;
     halt)
  when R
end
```

$E_1 = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^\perp \}$
$E'_1 = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^- \}$
$\Longrightarrow$
$E_2 = \{ \mathrm{A}^+, \mathrm{B}^-, \mathrm{R}^-, \mathrm{O}^\perp \}$
$E'_2 = \{ \mathrm{A}^+, \mathrm{B}^-, \mathrm{R}^-, \mathrm{O}^+ \}$

Deuxième instant d'ABRO en microsteps

■ : $\mathrm{Sel}^+, \mathrm{Go}^-, \mathrm{Res}^+$ □ : $\mathrm{Sel}^-, \mathrm{Go}^-, \mathrm{Res}^+$ □ : $\mathrm{Sel}^-, \mathrm{Go}^+, \mathrm{Res}^+$

■ loop

■ abort

■ { ■ [(■ awimm A $\bullet_0$)

|| (□ awimm B $\circ_\varnothing$)

]$\bullet_0$; □ [(□ emit O $\bullet_0$) ; (□ halt $\bullet_1$)] $\bullet_1$ } $\bullet_1$

when R $\bullet_1$

end $\bullet_1$

loop

abort

awimm A

||

awimm B

;

(emit O ; $\widehat{\text{halt}}$)

$\widehat{\text{when } R}$

end

$E_1 = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^\perp \}$
$E_1' = \{ \mathrm{A}^-, \mathrm{B}^+, \mathrm{R}^-, \mathrm{O}^- \}$
$\Longrightarrow$
$E_2 = \{ \mathrm{A}^+, \mathrm{B}^-, \mathrm{R}^-, \mathrm{O}^\perp \}$
$E_2' = \{ \mathrm{A}^+, \mathrm{B}^-, \mathrm{R}^-, \mathrm{O}^+ \}$

Que peut-on dire de la sémantique micro-pas ?

- Préserve les signaux déclarés
- Préserve le programme sous-jacent
- Si E est total (pas de $\perp$), E' l'est aussi
- Le code de retour n'est pas 1 ssi le résidu est inerte
- Déterministe
- Must/Can peuvent s'étendre : *sMust*, *sCan*

Que peut-on dire de la sémantique micro-pas ?

- ~~Préserve les signaux déclarés~~ plus de E'
- Préserve le programme sous-jacent
- Si E est total (pas de $\perp$), E' l'est aussi
- Le code de retour n'est pas 1 ssi le résidu est inerte
- Déterministe
- Must/Can peuvent s'étendre : *sMust*, *sCan*

Que peut-on dire de la sémantique micro-pas ?

- Préserve les signaux déclarés

$$p \xrightarrow[E]{} p' \implies \mathrm{dom}(\texttt{to\_event}\, p') = \mathrm{dom}\, E$$

- Préserve le programme sous-jacent

- Si E est total (pas de $\bot$), E' l'est aussi
- Le code de retour n'est pas 1 ssi le résidu est inerte
- Déterministe
- Must/Can peuvent s'étendre : *sMust*, *sCan*

Que peut-on dire de la sémantique micro-pas ?

- Préserve les signaux déclarés

$$p \xrightarrow[E]{} p' \implies \mathrm{dom}(\texttt{to\_event}\, p') = \mathrm{dom}\, E$$

- Préserve le programme sous-jacent

$$p \xrightarrow[E]{} p' \implies \mathit{base}(p) = \mathit{base}(p')$$

- Si E est total (pas de $\bot$), E' l'est aussi
- Le code de retour n'est pas 1 ssi le résidu est inerte
- Déterministe
- Must/Can peuvent s'étendre : *sMust*, *sCan*

Que peut-on dire de la sémantique micro-pas ?

- Préserve les signaux déclarés

$$p \xrightarrow[E]{} p' \implies \mathrm{dom}(\texttt{to\_event}\ p') = \mathrm{dom}\ E$$

- Préserve le programme sous-jacent

$$p \xrightarrow[E]{} p' \implies \mathit{base}(p) = \mathit{base}(p')$$
$$\mathit{base}(\texttt{from\_cmd}\ p) = p$$
$$\mathit{base}(\texttt{from\_state}\ \widehat{p}\,) = \mathit{base}(\widehat{p}\,)$$
$$\mathit{base}(\texttt{to\_term}\ p) = \mathit{base}(p)$$

- Si E est total (pas de $\bot$), E' l'est aussi
- Le code de retour n'est pas 1 ssi le résidu est inerte
- Déterministe
- Must/Can peuvent s'étendre : *sMust*, *sCan*

Que peut-on dire de la sémantique micro-pas ?

- Préserve les signaux déclarés

$$p \xrightarrow[E]{} p' \implies \mathrm{dom}(\texttt{to\_event}\ p') = \mathrm{dom}\ E$$

- Préserve le programme sous-jacent

$$p \xrightarrow[E]{} p' \implies \mathit{base}(p) = \mathit{base}(p')$$
$$\mathit{base}(\texttt{from\_cmd}\ p) = p$$
$$\mathit{base}(\texttt{from\_state}\ \widehat{p}\,) = \mathit{base}(\widehat{p}\,)$$
$$\mathit{base}(\texttt{to\_term}\ p) = \mathit{base}(p)$$

- ~~Si E est total (pas de $\bot$), E' l'est aussi~~
- Le code de retour n'est pas 1 ssi le résidu est inerte
- Déterministe
- Must/Can peuvent s'étendre : *sMust*, *sCan*

Que peut-on dire de la sémantique micro-pas ?

- Préserve les signaux déclarés

$$p \xrightarrow[E]{} p' \implies \mathrm{dom}(\texttt{to\_event}\ p') = \mathrm{dom}\ E$$

- Préserve le programme sous-jacent

$$p \xrightarrow[E]{} p' \implies \mathit{base}(p) = \mathit{base}(p')$$
$$\mathit{base}(\texttt{from\_cmd}\ p) = p$$
$$\mathit{base}(\texttt{from\_state}\ \widehat{p}\,) = \mathit{base}(\widehat{p}\,)$$
$$\mathit{base}(\texttt{to\_term}\ p) = \mathit{base}(p)$$

- ~~Si E est total (pas de $\bot$), E' l'est aussi~~
- ~~Le code de retour n'est pas 1 ssi le résidu est inerte~~
- Déterministe
- Must/Can peuvent s'étendre : *sMust*, *sCan*

Que peut-on dire de la sémantique micro-pas ?

- Préserve les signaux déclarés

$$p \xrightarrow[E]{} p' \implies \mathrm{dom}(\texttt{to\_event}\ p') = \mathrm{dom}\ E$$

- Préserve le programme sous-jacent

$$p \xrightarrow[E]{} p' \implies \mathit{base}(p) = \mathit{base}(p')$$
$$\mathit{base}(\texttt{from\_cmd}\ p) = p$$
$$\mathit{base}(\texttt{from\_state}\ \widehat{p}\,) = \mathit{base}(\widehat{p}\,)$$
$$\mathit{base}(\texttt{to\_term}\ p) = \mathit{base}(p)$$

- ~~Si E est total (pas de $\bot$), E' l'est aussi~~
- ~~Le code de retour n'est pas 1 ssi le résidu est inerte~~
- ~~Déterministe~~ Non-déterministe
- Must/Can peuvent s'étendre : *sMust*, *sCan*

Que peut-on dire de la sémantique micro-pas ?

- Préserve les signaux déclarés

$$p \xrightarrow[E]{} p' \implies \mathrm{dom}(\texttt{to\_event}\ p') = \mathrm{dom}\ E$$

- Préserve le programme sous-jacent

$$p \xrightarrow[E]{} p' \implies \mathit{base}(p) = \mathit{base}(p')$$
$$\mathit{base}(\texttt{from\_cmd}\ p) = p$$
$$\mathit{base}(\texttt{from\_state}\ \widehat{p}\,) = \mathit{base}(\widehat{p}\,)$$
$$\mathit{base}(\texttt{to\_term}\ p) = \mathit{base}(p)$$

- ~~Si E est total (pas de $\bot$), E' l'est aussi~~
- ~~Le code de retour n'est pas 1 ssi le résidu est inerte~~
- ~~Déterministe~~ Non-déterministe
- ~~Must/Can peuvent s'étendre : *sMust*, *sCan*~~

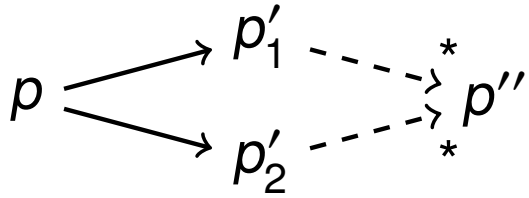
Que peut-on dire de la sémantique micro-pas ? (2)

- Non déterministe mais confluente
- Les entrées ne changent pas
- Une seule valeur change à la fois
- Augmente l'information
- Compatible avec l'augmentation d'information

- On ne crée pas de 1
- ...

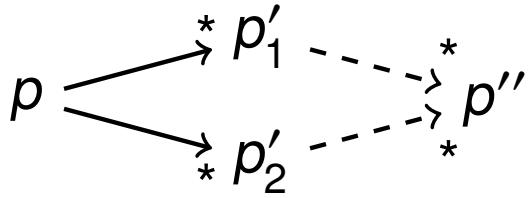
Que peut-on dire de la sémantique micro-pas ? (2)

- Non déterministe mais confluente



- Les entrées ne changent pas
- Une seule valeur change à la fois
- Augmente l'information
- Compatible avec l'augmentation d'information
- On ne crée pas de 1
- ...

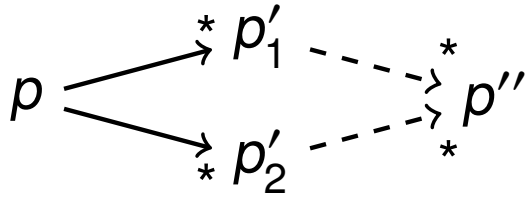
Que peut-on dire de la sémantique micro-pas ? (2)



- Non déterministe mais confluente
- Les entrées ne changent pas
- Une seule valeur change à la fois
- Augmente l'information
- Compatible avec l'augmentation d'information
- On ne crée pas de 1
- ...

Que peut-on dire de la sémantique micro-pas ? (2)

- Non déterministe mais confluente



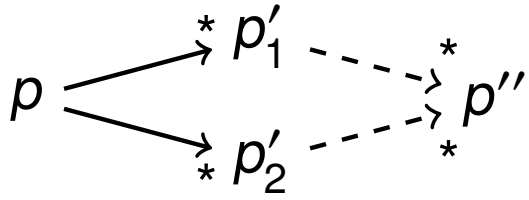
- Les entrées ne changent pas

$$\square\, p \circ \xrightarrow[E]{} \square\, p' \circ \implies \square = \square$$

- Une seule valeur change à la fois
- Augmente l'information
- Compatible avec l'augmentation d'information
- On ne crée pas de 1
- ...

Que peut-on dire de la sémantique micro-pas ? (2)

- Non déterministe mais confluente



- Les entrées ne changent pas

$$\square\, p \circ \xrightarrow[E]{} \square\, p' \circ \implies \square = \square$$

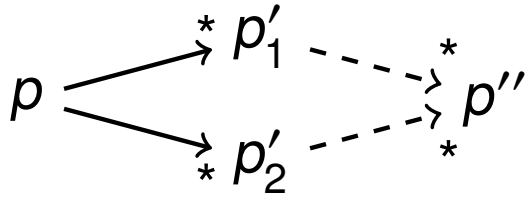
- Une seule valeur change à la fois

$$\square\, p \circ \xrightarrow[E]{} \square\, p' \circ \implies \circ = \circ \vee p = p'$$

- Augmente l'information
- Compatible avec l'augmentation d'information
- On ne crée pas de 1
- ...

Que peut-on dire de la sémantique micro-pas ? (2)

- Non déterministe mais confluente



- Les entrées ne changent pas

$$\square\, p \circ \xrightarrow[E]{} \square\, p' \circ \implies \square = \square$$

- Une seule valeur change à la fois

$$\square\, p \circ \xrightarrow[E]{} \square\, p' \circ \implies \circ = \circ \vee p = p'$$

- Augmente l'information

$$p \xrightarrow[E]{} p' \implies p < p'$$

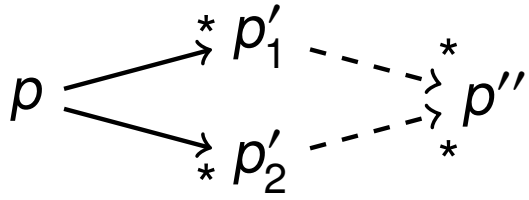
- Compatible avec l'augmentation d'information

- On ne crée pas de 1

- ...

Que peut-on dire de la sémantique micro-pas ? (2)

- Non déterministe mais confluente



- Les entrées ne changent pas

$$\square\, p \circ \xrightarrow[E]{} \square\, p' \circ \implies \square = \square$$

- Une seule valeur change à la fois

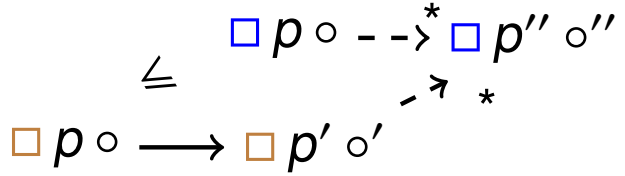
$$\square\, p \circ \xrightarrow[E]{} \square\, p' \circ \implies \circ = \circ \vee p = p'$$

- Augmente l'information

$$p \xrightarrow[E]{} p' \implies p < p'$$

- Compatible avec l'augmentation d'information

$$p \xrightarrow[E]{} p' \implies E \leqslant E' \implies p \xrightarrow[E']{} p'$$

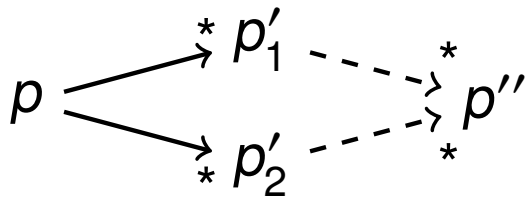


- On ne crée pas de 1

- ...

Que peut-on dire de la sémantique micro-pas ? (2)

- Non déterministe mais confluente



- Les entrées ne changent pas

$$\square\, p \circ \xrightarrow[E]{} \square\, p' \circ \implies \square = \square$$

- Une seule valeur change à la fois

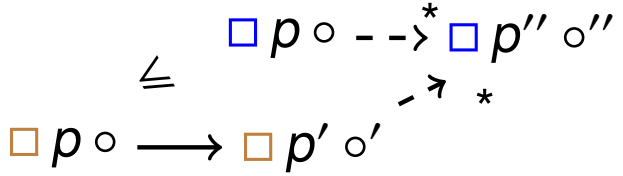
$$\square\, p \circ \xrightarrow[E]{} \square\, p' \circ \implies \circ = \circ \vee p = p'$$

- Augmente l'information

$$p \xrightarrow[E]{} p' \implies p < p'$$

- Compatible avec l'augmentation d'information

$$p \xrightarrow[E]{} p' \implies E \leqslant E' \implies p \xrightarrow[E']{} p'$$



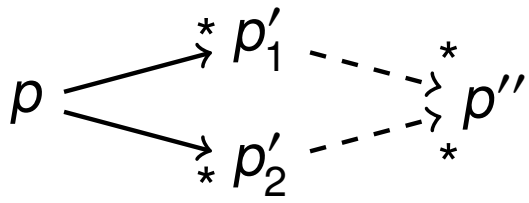
- On ne crée pas de 1

$$\square\, p \circ \xrightarrow[E]{} \square\, p \circ_{\varnothing}$$

- ...

Que peut-on dire de la sémantique micro-pas ? (2)

- Non déterministe mais confluente



- Les entrées ne changent pas

$$\square\, p\, \circ \xrightarrow[E]{} \square\, p'\, \circ \implies \square = \square$$

- Une seule valeur change à la fois

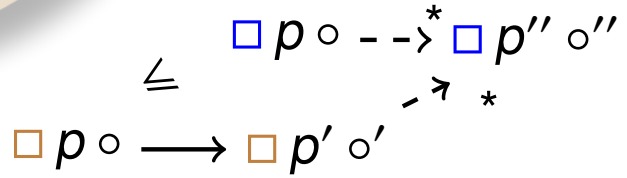
$$\square\, p\, \circ \xrightarrow[E]{} \square\, p'\, \circ \implies \circ = \circ \vee p = p'$$

- Augmente l'information

$$p \xrightarrow[E]{} p' \implies p < p'$$

- Compatible avec l'augmentation d'information

$$p \xrightarrow[E]{} p' \implies E \leqslant E' \implies p \xrightarrow[E']{} p'$$

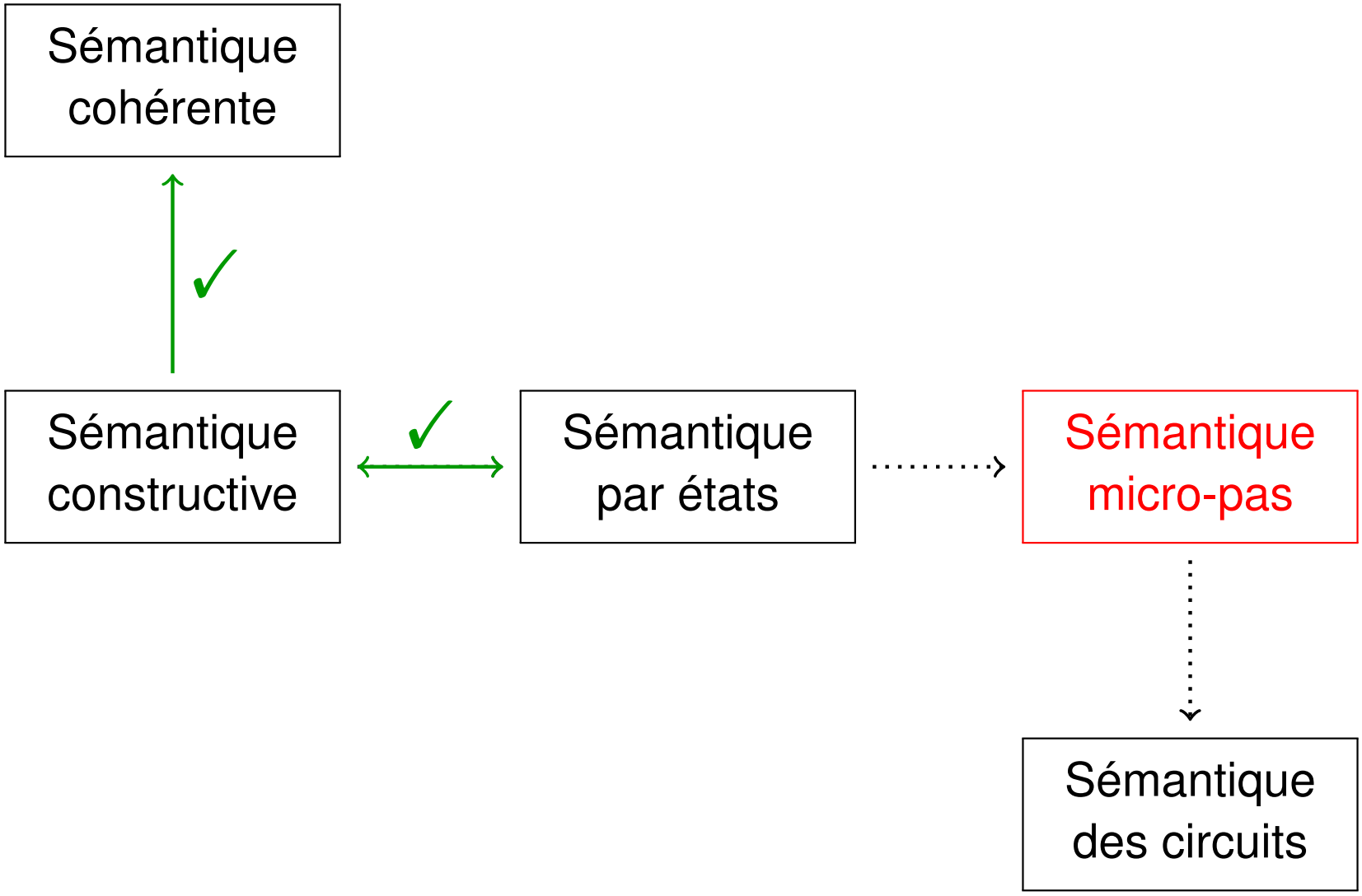


- On ne crée pas de 1

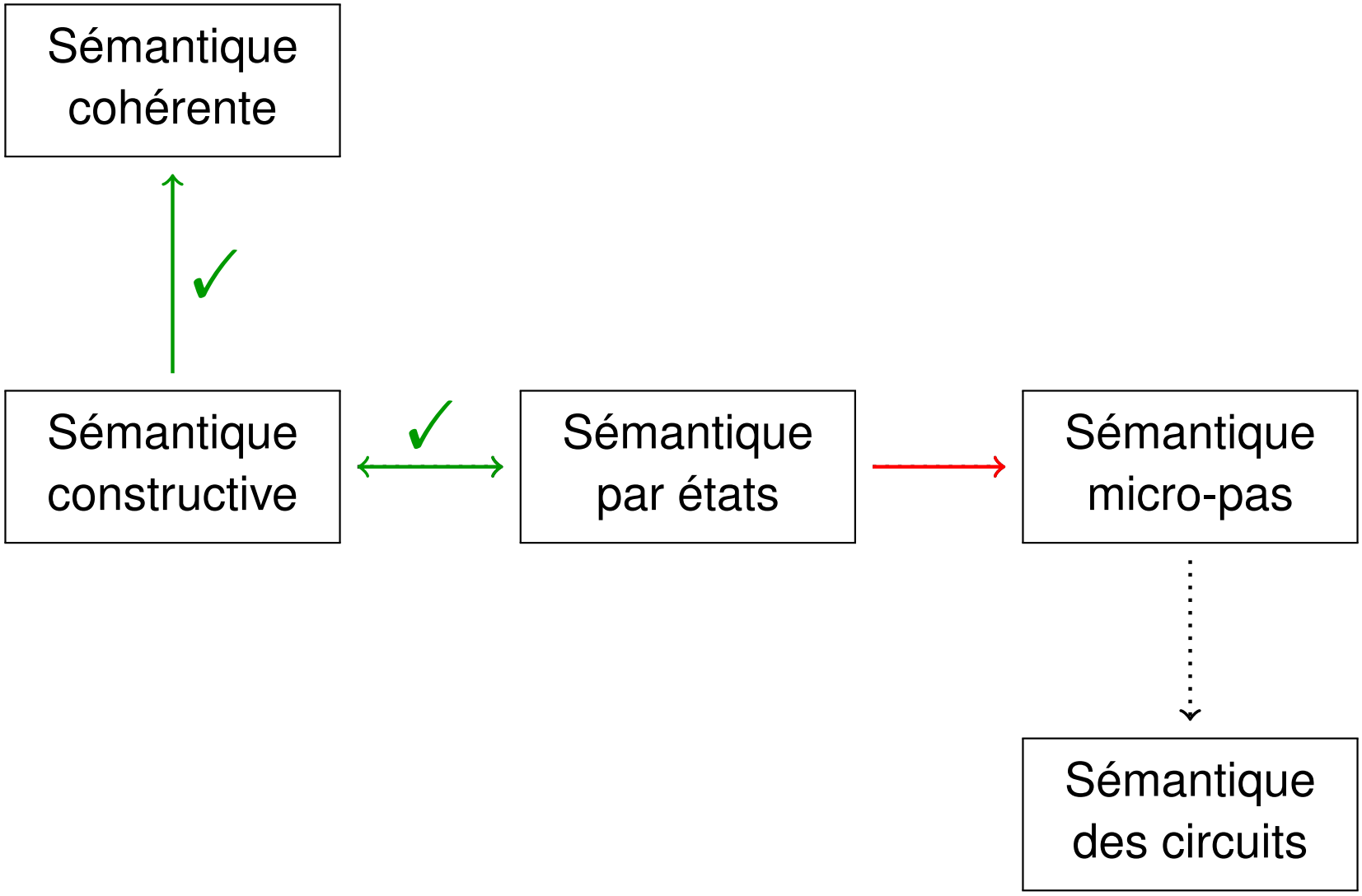
$$\square\, p\, \circ \xrightarrow[E]{} \square\, p \circ_{\varnothing}$$

- ...

Chaîne des sémantique d'Esterel



Chaîne des sémantique d'Esterel



Équivalence états/micro-états : sens direct

Objectif : Montrer

$$p \xrightarrow[E]{E',k}\!\!\!\!\twoheadrightarrow_s \overline{p'} \implies \exists p'',\ \square\, p \circ \xrightarrow[E]{}{}^{*} \square\, p'' \bullet_k$$
$$\wedge\ \texttt{to\_term}\ p'' = \overline{p'} \ \wedge\ \texttt{to\_event}\ p'' = E'$$

$$\widehat{p} \xrightarrow[E]{E',k}\!\!\!\!\twoheadrightarrow_r \overline{p'} \implies \exists p'',\ \blacksquare\, p \circ \xrightarrow[E]{}{}^{*} \blacksquare\, p'' \bullet_k \ \wedge\ \ldots$$

Équivalence états/micro-états : sens direct

Objectif : Montrer

$$p \xrightarrow[E]{E',k}\!\!\!\!\twoheadrightarrow_s \overline{p'} \implies \exists p'', \Box\, p \circ \underset{E}{\longrightarrow}^* \Box\, p'' \bullet_k$$
$$\wedge\ \texttt{to\_term}\ p'' = \overline{p'} \ \wedge\ \texttt{to\_event}\ p'' = E'$$

$$\widehat{p} \xrightarrow[E]{E',k}\!\!\!\!\twoheadrightarrow_r \overline{p'} \implies \exists p'', \blacksquare\, p \circ \underset{E}{\longrightarrow}^* \blacksquare\, p'' \bullet_k \wedge \ldots$$

- C'est le sens facile !
  $\rightsquigarrow$ on peut choisir les micro-pas
- Difficulté : correspondance entre Must/Can et `to_event`
  - $s \in \mathit{Must}_s(p, E) \implies \exists p', \Box\, p \circ \underset{E}{\longrightarrow}^* p' \wedge s^+ \in \texttt{to\_event}\ p'$
  - $s \notin \mathit{Can}^+_s(p, E) \implies \exists p', \Box\, p \circ \underset{E}{\longrightarrow}^* p' \wedge s^- \in \texttt{to\_event}\ p'$
  - $s \notin \mathit{Can}^\perp_s(p, E) \implies \exists p', \Box\, p \circ \underset{E}{\longrightarrow}^* p' \wedge s^- \in \texttt{to\_event}\ p'$

Équivalence états/micro-états : sens direct

Objectif : Montrer

$$p \xrightarrow[E]{E',k}\!\!\!\!\twoheadrightarrow_s \overline{p'} \implies \exists p'', \Box\, p \circ \underset{E}{\longrightarrow}^* \Box\, p'' \bullet_k$$
$$\wedge\ \texttt{to\_term}\ p'' = \overline{p'} \ \wedge\ \texttt{to\_event}\ p'' = E'$$

$$\widehat{p} \xrightarrow[E]{E',k}\!\!\!\!\twoheadrightarrow_r \overline{p'} \implies \exists p'', \blacksquare\, p \circ \underset{E}{\longrightarrow}^* \blacksquare\, p'' \bullet_k \wedge \ldots$$

- C'est le sens facile !
  $\rightsquigarrow$ on peut choisir les micro-pas
- Difficulté : correspondance entre Must/Can et `to_event`
  - $s \in \mathit{Must}_s(p, E) \implies \exists p', \Box\, p \circ \underset{E}{\longrightarrow}^* p' \wedge s^+ \in \texttt{to\_event}\ p'$
  - $s \notin \mathit{Can}^+_s(p, E) \implies \exists p', \Box\, p \circ \underset{E}{\longrightarrow}^* p' \wedge s^- \in \texttt{to\_event}\ p'$
  - $s \notin \mathit{Can}^\perp_s(p, E) \implies \exists p', \Box\, p \circ \underset{E}{\longrightarrow}^* p' \wedge s^- \in \texttt{to\_event}\ p'$
  - + 3 autres pour les codes de retour ...

Équivalence états/micro-états : sens direct

Objectif : Montrer

$$p \xrightarrow[E]{E',k}\!\!\!\!\twoheadrightarrow_s \overline{p'} \implies \exists p'', \Box\, p \circ \underset{E}{\longrightarrow}^* \Box\, p'' \bullet_k$$
$$\wedge\ \texttt{to\_term}\ p'' = \overline{p'} \ \wedge\ \texttt{to\_event}\ p'' = E'$$

$$\widehat{p} \xrightarrow[E]{E',k}\!\!\!\!\twoheadrightarrow_r \overline{p'} \implies \exists p'', \blacksquare\, p \circ \underset{E}{\longrightarrow}^* \blacksquare\, p'' \bullet_k \wedge \ldots$$

- C'est le sens facile !
  $\rightsquigarrow$ on peut choisir les micro-pas
- Difficulté : correspondance entre Must/Can et `to_event`
  - $s \in \mathit{Must}_s(p, E) \implies \exists p', \Box\, p \circ \underset{E}{\longrightarrow}^* p' \wedge s^+ \in \texttt{to\_event}\ p'$
  - $s \notin \mathit{Can}^+_s(p, E) \implies \exists p', \Box\, p \circ \underset{E}{\longrightarrow}^* p' \wedge s^- \in \texttt{to\_event}\ p'$
  - $s \notin \mathit{Can}^\perp_s(p, E) \implies \exists p', \Box\, p \circ \underset{E}{\longrightarrow}^* p' \wedge s^- \in \texttt{to\_event}\ p'$
  - + 3 autres pour les codes de retour ...
  - ... $\times 2$ pour $\widehat{p}$

Équivalence états/micro-états : sens direct

Objectif : Montrer

$$p \xrightarrow[E]{E',k}\!\!\!\!\twoheadrightarrow_s \overline{p'} \implies \exists p'', \Box\, p \circ \underset{E}{\longrightarrow}^* \Box\, p'' \bullet_k$$
$$\wedge\ \texttt{to\_term}\ p'' = \overline{p'} \ \wedge\ \texttt{to\_event}\ p'' = E'$$

$$\widehat{p} \xrightarrow[E]{E',k}\!\!\!\!\twoheadrightarrow_r \overline{p'} \implies \exists p'', \blacksquare\, p \circ \underset{E}{\longrightarrow}^* \blacksquare\, p'' \bullet_k \wedge \ldots$$

- C'est le sens facile !
  $\rightsquigarrow$ on peut choisir les micro-pas
- Difficulté : correspondance entre Must/Can et `to_event`
  - $s \in \mathit{Must}_s(p, E) \implies \exists p', \Box\, p \circ \underset{E}{\longrightarrow}^* p' \wedge s^+ \in \texttt{to\_event}\ p'$
  - $s \notin \mathit{Can}^+_s(p, E) \implies \exists p', \Box\, p \circ \underset{E}{\longrightarrow}^* p' \wedge s^- \in \texttt{to\_event}\ p'$
  - $s \notin \mathit{Can}^\perp_s(p, E) \implies \exists p', \Box\, p \circ \underset{E}{\longrightarrow}^* p' \wedge s^- \in \texttt{to\_event}\ p'$
  - + 3 autres pour les codes de retour ...
  - ... $\times 2$ pour $\widehat{p}$
- Et pour la réciproque ?

Équivalence états/micro-états : sens direct

Objectif : Montrer

$$p \xrightarrow[E]{E',k}\!\!\!\!\twoheadrightarrow_s \overline{p'} \implies \exists p'', \Box\, p \circ \underset{E}{\longrightarrow}^* \Box\, p'' \bullet_k$$
$$\wedge\ \texttt{to\_term}\ p'' = \overline{p'} \ \wedge\ \texttt{to\_event}\ p'' = E'$$

$$\widehat{p} \xrightarrow[E]{E',k}\!\!\!\!\twoheadrightarrow_r \overline{p'} \implies \exists p'', \blacksquare\, p \circ \underset{E}{\longrightarrow}^* \blacksquare\, p'' \bullet_k \wedge \ldots$$

- C'est le sens facile !
  $\rightsquigarrow$ on peut choisir les micro-pas
- Difficulté : correspondance entre Must/Can et `to_event`
  - $s \in \mathit{Must}_s(p, E) \implies \exists p', \Box\, p \circ \underset{E}{\longrightarrow}^* p' \wedge s^+ \in \texttt{to\_event}\ p'$
  - $s \notin \mathit{Can}^+_s(p, E) \implies \exists p', \Box\, p \circ \underset{E}{\longrightarrow}^* p' \wedge s^- \in \texttt{to\_event}\ p'$
  - $s \notin \mathit{Can}^\perp_s(p, E) \implies \exists p', \Box\, p \circ \underset{E}{\longrightarrow}^* p' \wedge s^- \in \texttt{to\_event}\ p'$
  - + 3 autres pour les codes de retour ...
  - ... $\times 2$ pour $\widehat{p}$
- Et pour la réciproque ? ça se corse **beaucoup** !

Équivalence états/micro-états : la réciproque

Problème : on ne sait rien d'un micro-état

- Est-ce que les $\square$ et $\circ$ sont cohérents ?
- Est-ce qu'on peut mener l'évaluation au bout ?

Équivalence états/micro-états : la réciproque

Problème : on ne sait rien d'un micro-état

- Est-ce que les $\square$ et $\circ$ sont cohérents ?
- Est-ce qu'on peut mener l'évaluation au bout ?

Solution : un invariant `valid_coloring` sur les micro-états

- Contraindre les $\square$ et $\circ$
- Assurer des invariants globaux de la traduction
  $\rightsquigarrow$ dans $p\,;q$ et $s\,?\,p\,,q$ on a $\mathrm{Sel}(p)\ \#\ \mathrm{Sel}(q)$

Invariant pour $s\ ?\ p\ ,\ q$

```
(* The signal s must exist in the environment *)
S.find s E = Some bo ->
(* The input satisfies its invariant *)
input_invariant gr ->
(* Two AND gates starting execution *)
AND (Go gr) bo (get_Go p) ->
AND (Go gr) (Bneg bo) (get_Go q) ->
(* The other input color are transmitted to both branches *)
Ble (get_Res p) (Res gr) ->
Ble (get_Res q) (Res gr) ->
Ble (get_Susp p) (Susp gr) ->
Ble (get_Susp q) (Susp gr) ->
(* An if-then-else is selected iff exactly one of its branches is *)
Sel gr = (get_Sel p || get_Sel q)%bool ->
get_Sel p && get_Sel q = false ->
(* Return code is union *)
out_le out ((get_output_color p) ∪ (get_output_color q)) ->
out_le ((get_output_color (from_cmd (base p)))
         ∪ (get_output_color (from_cmd (base q)))) out ->
(* Recursive calls *)
valid_coloring E p ->
valid_coloring E q ->
valid_coloring E (Colors gr (s ? p, q) out)
```

Équivalence états/micro-états : la réciproque

Problème : on ne sait rien d'un micro-état

- Est-ce que les $\square$ et $\circ$ sont cohérents ?
- Est-ce qu'on peut mener l'évaluation au bout ?

Solution : un invariant `valid_coloring` sur les micro-états

- Contraindre les $\square$ et $\circ$
- Assurer des invariants globaux de la traduction
  $\rightsquigarrow$ dans $p\,;q$ et $s\,?\,p\,,q$ on a $\mathrm{Sel}(p)\ \#\ \mathrm{Sel}(q)$

Équivalence états/micro-états : la réciproque

Problème : on ne sait rien d'un micro-état

- Est-ce que les $\square$ et $\circ$ sont cohérents ?
- Est-ce qu'on peut mener l'évaluation au bout ?

Solution : un invariant `valid_coloring` sur les micro-états

- Contraindre les $\square$ et $\circ$
- Assurer des invariants globaux de la traduction
  $\rightsquigarrow$ dans $p\,;q$ et $s\,?\,p\,,q$ on a $\mathrm{Sel}(p)\ \#\ \mathrm{Sel}(q)$

On démontre au final :

Si $\square\ (\texttt{from\_cmd}\ p) \leqslant p'$, $\mathrm{Total}(p')$ et $\texttt{valid\_coloring}\ E\ p'$,
alors $p \xrightarrow[E]{\texttt{to\_event}\ p', k(p')} \!\!\!\!\twoheadrightarrow_s\ \texttt{to\_term}\ p'$.

Équivalence états/micro-états : la réciproque

Problème : on ne sait rien d'un micro-état

- Est-ce que les $\square$ et $\circ$ sont cohérents ?
- Est-ce qu'on peut mener l'évaluation au bout ?

Solution : un invariant `valid_coloring` sur les micro-états

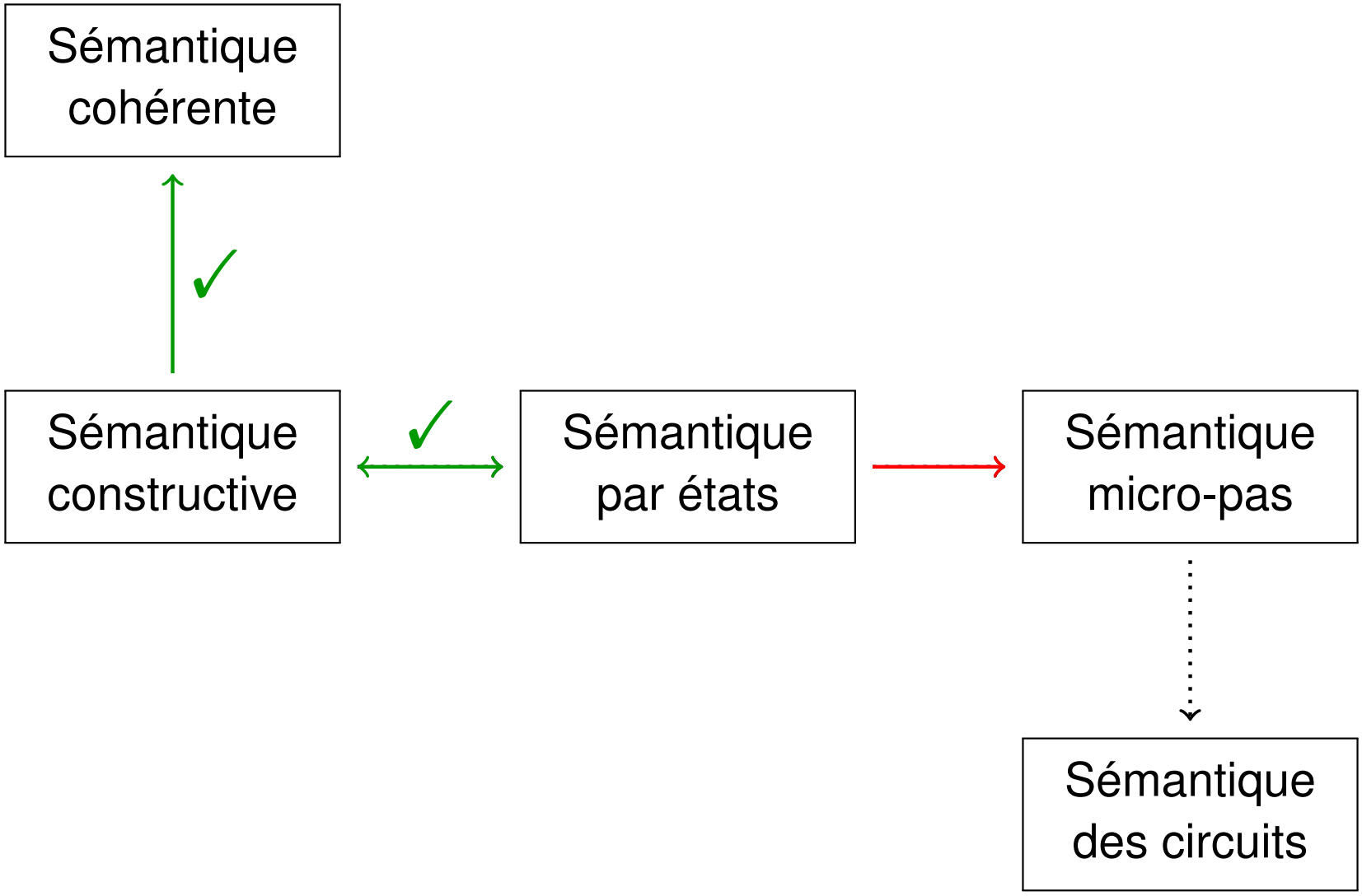
- Contraindre les $\square$ et $\circ$
- Assurer des invariants globaux de la traduction
  $\rightsquigarrow$ dans $p\,;q$ et $s\,?\,p\,,q$ on a $\mathrm{Sel}(p)\ \#\ \mathrm{Sel}(q)$

On démontre au final :

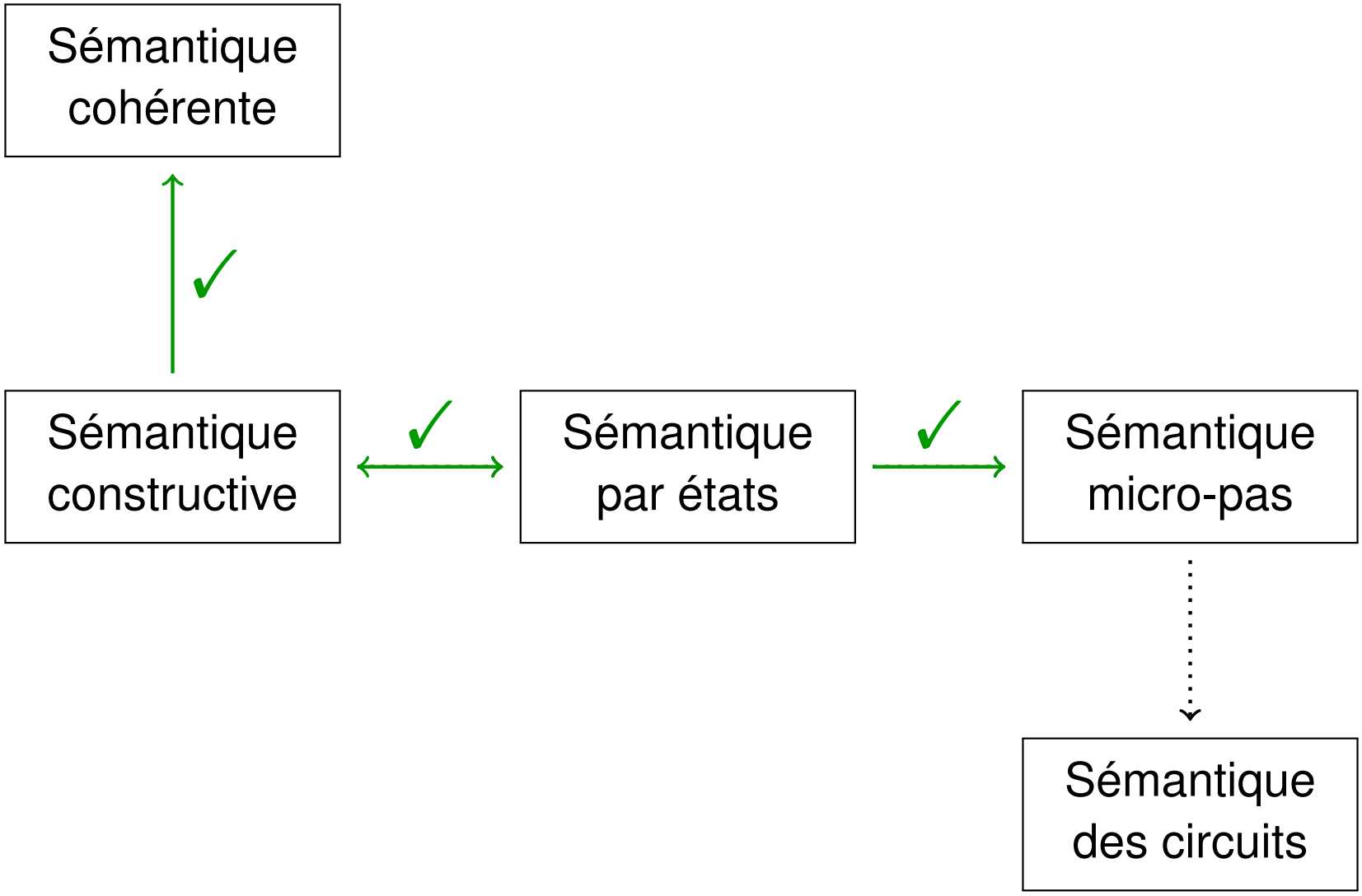
Si $\square\ (\texttt{from\_cmd}\ p) \leqslant p'$, $\mathrm{Total}(p')$ et $\texttt{valid\_coloring}\ E\ p'$,
alors $p \xrightarrow[E]{\texttt{to\_event}\ p', k(p')}\!\!\!\!\twoheadrightarrow_s\ \texttt{to\_term}\ p'$.

État actuel : réciproque du lien entre Must/Can et `to_event`

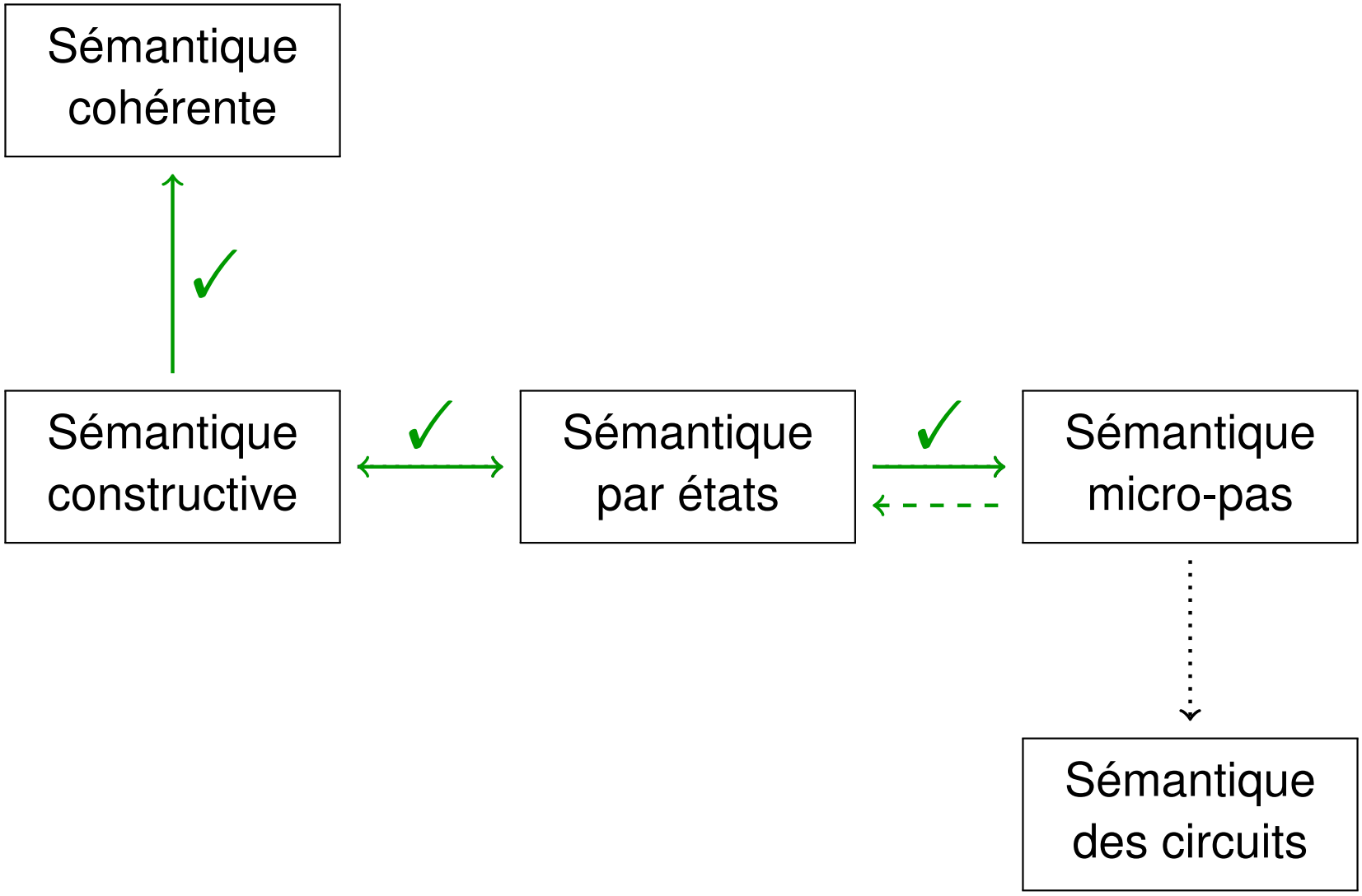
Chaîne des sémantique d'Esterel



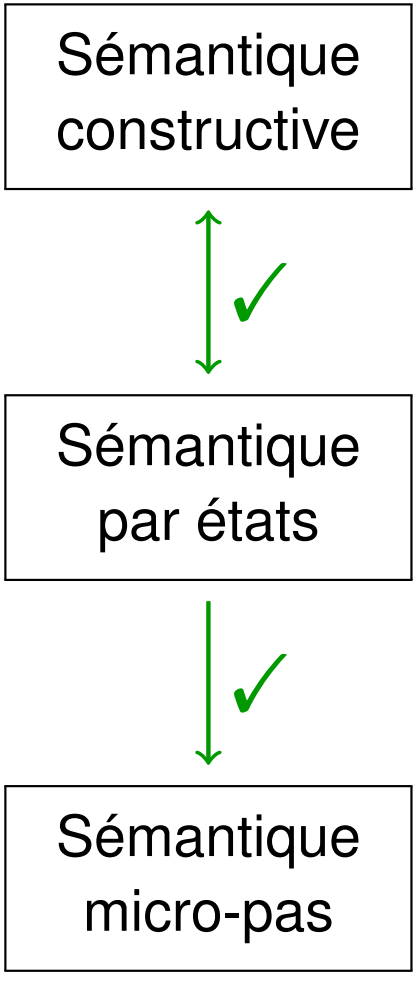
Chaîne des sémantique d'Esterel



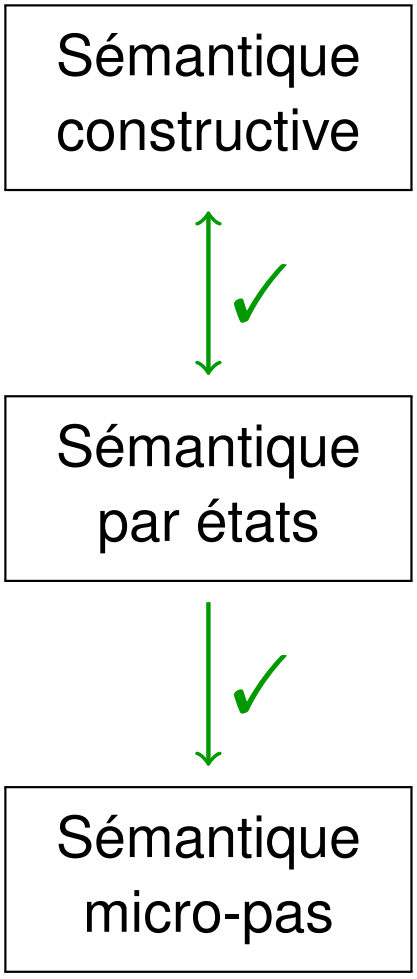
Chaîne des sémantique d'Esterel



Conclusion sur les sémantiques d'Esterel



Conclusion sur les sémantiques d'Esterel



- ⊕ sémantique usuelle pour un langage
- ⊕ peu de règles (14)
- ⊖ réécrit le programme

- ⊕ exécution par marquage
- ⊕ lien avec les états stables des circuits
- ⊖ deux types de règles (14 + 15)

- ⊕ sémantique locale bas niveau
- ⊕ très proche des circuits
- ⊕ Can/Must ont disparu
- ⊕ un seul type de règles
- ⊖ beaucoup de règles (52)
- ⊖ pas de boucle/réincarnation

Conclusion sur l'utilisation de Coq

Deux grandes phases

- Formaliser le livre web [The Constructive Semantics of Pure Esterel]
  - Facile : tout est clair
  - Il suffit de traduire dans Coq
- Passer aux micro-pas
  - Inventer puis ajuster une bonne sémantique
  - Processus itératif assez long 1 itération $\approx$ 1–2 mois

Intérêt de Coq

- Tout vérifier
- Comprendre les détails synchroniseur

Conclusion sur l'utilisation de Coq

Deux grandes phases

- Formaliser le livre web [The Constructive Semantics of Pure Esterel]
  - Facile : tout est clair
  - Il suffit de traduire dans Coq
- Passer aux micro-pas
  - Inventer puis ajuster une bonne sémantique
  - Processus itératif assez long 1 itération ≈ 1–2 mois

Intérêt de Coq

- Tout vérifier
- Comprendre les détails synchroniseur

The End