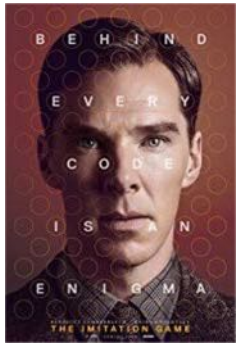
Diagrammes états-transitions algébriques

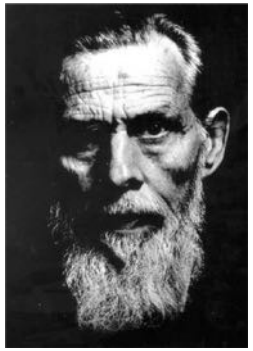
GDR GPL 2018, Grenoble, 12 juin 2018

Marc Frappier
http://info.usherbrooke.ca/mfrappier
GRIL – Groupe de recherche en ingénierie du logiciel
Département d'informatique
Faculté des sciences

UNIVERSITÉ DE SHERBROOKE

Automates : historique




- 1936 : Alan Turing†
  - *On Computable Numbers, with an Application to the Entscheidungsproblem*
    Proceedings of the London Mathematical Society
- 1943 : Warren McCulloch†, Walter Pitts†
  - *A logical calculus of the ideas immanent in nervous activity*
    Bulletin of Mathematical Biophysics
- 1951 : Stephen C. Kleene†
  - *Representation of events in nerve nets and finite automata*
    U.S. Air Force Project Rand Research Memorandum
- 1955 : George H. Mealy†
  - *A method for synthesising sequential circuits*
    Bell System Technical Journal.
- 1956 : Edward F. Moore†
  - Gedanken-experiments on sequential machines
    Annals of Mathematics Studies
- 1959 : Michael O. Rabin, Dana S. Scott
  - *Finite Automata and Their Decision Problems*
    IBM Journal

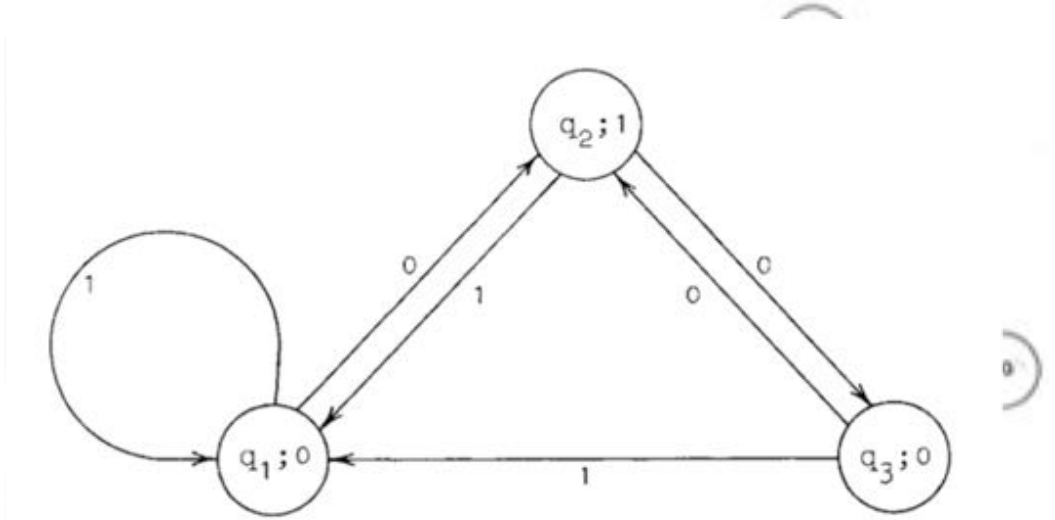


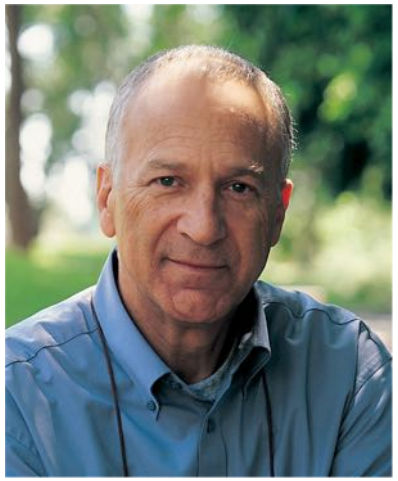
FIGURE 5. Transition Diagram of Machine C



1987 : Statecharts – David Harel

Statecharts: a visual formalism for complex systems

Science of Computer Programming



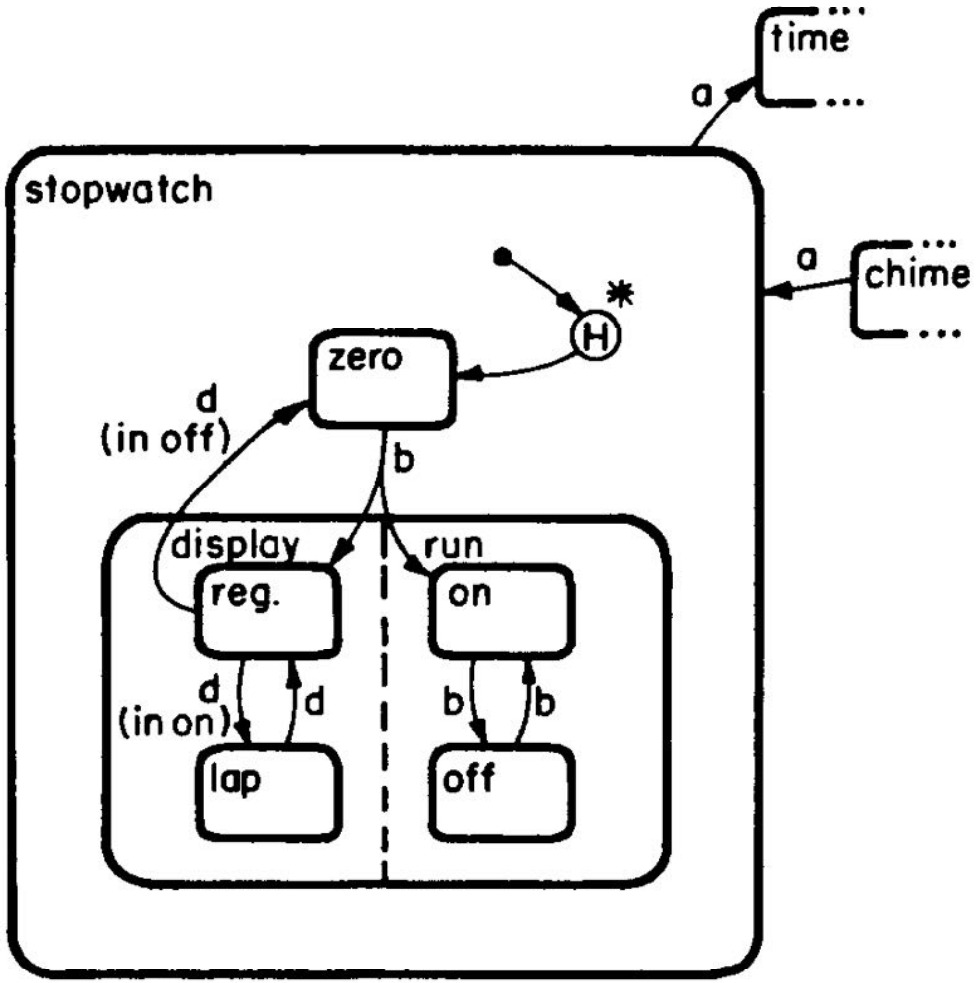
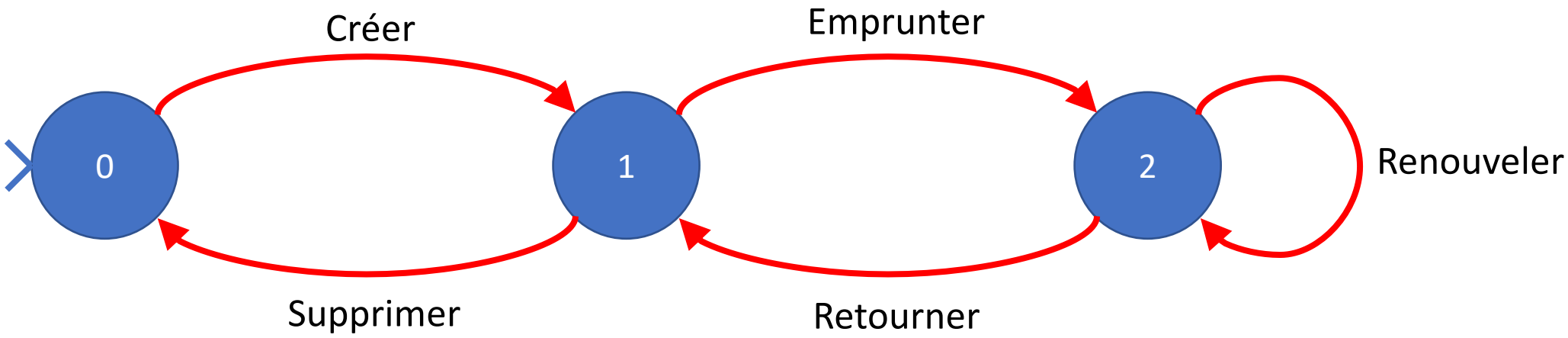
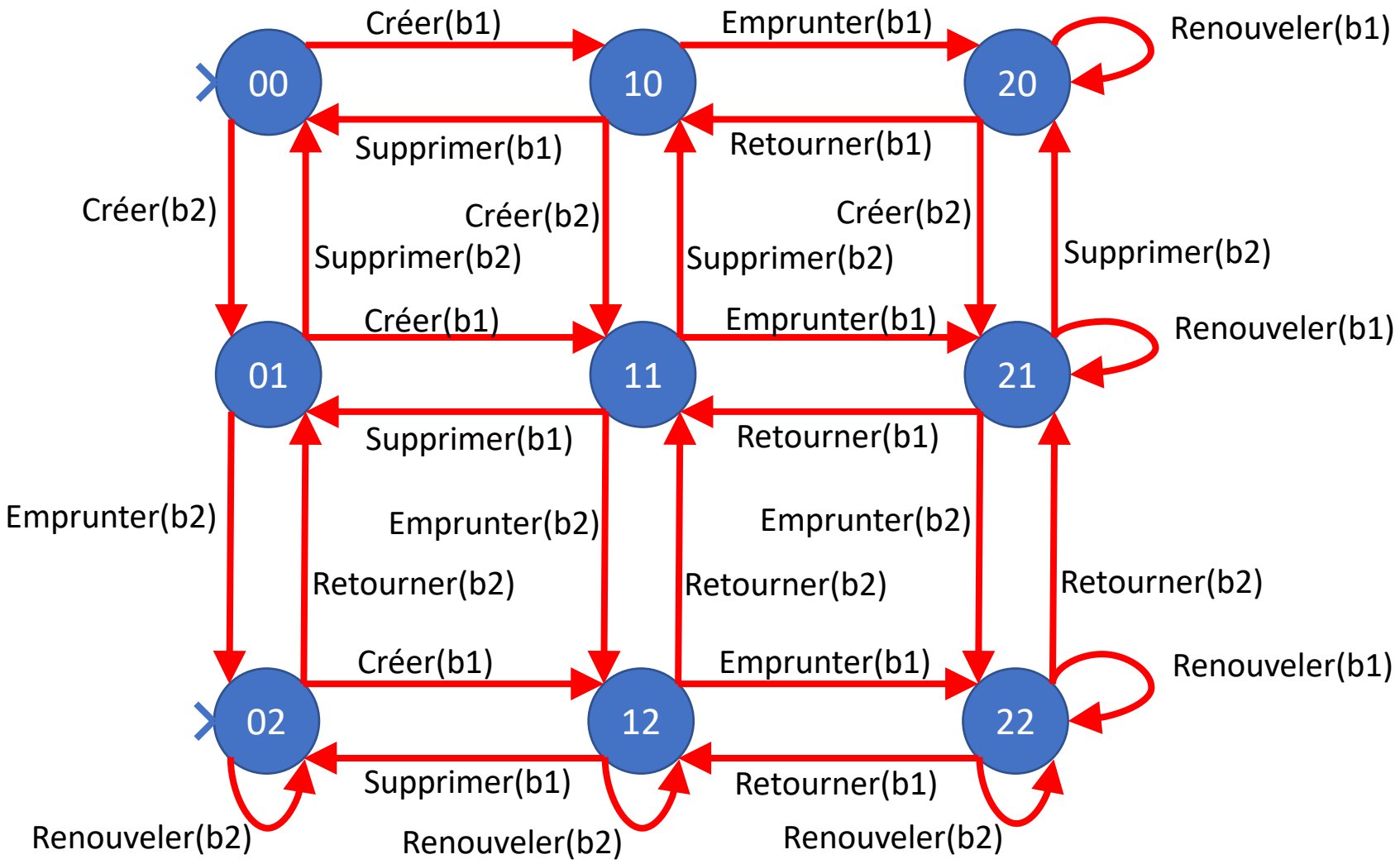


Fig. 25.

Automate : gérer un livre d'une bibliothèque



Automate : gérer 2 livres d'une bibliothèque



Produit d'automate

(aussi appelé entrelacement)

Statecharts

- AND state

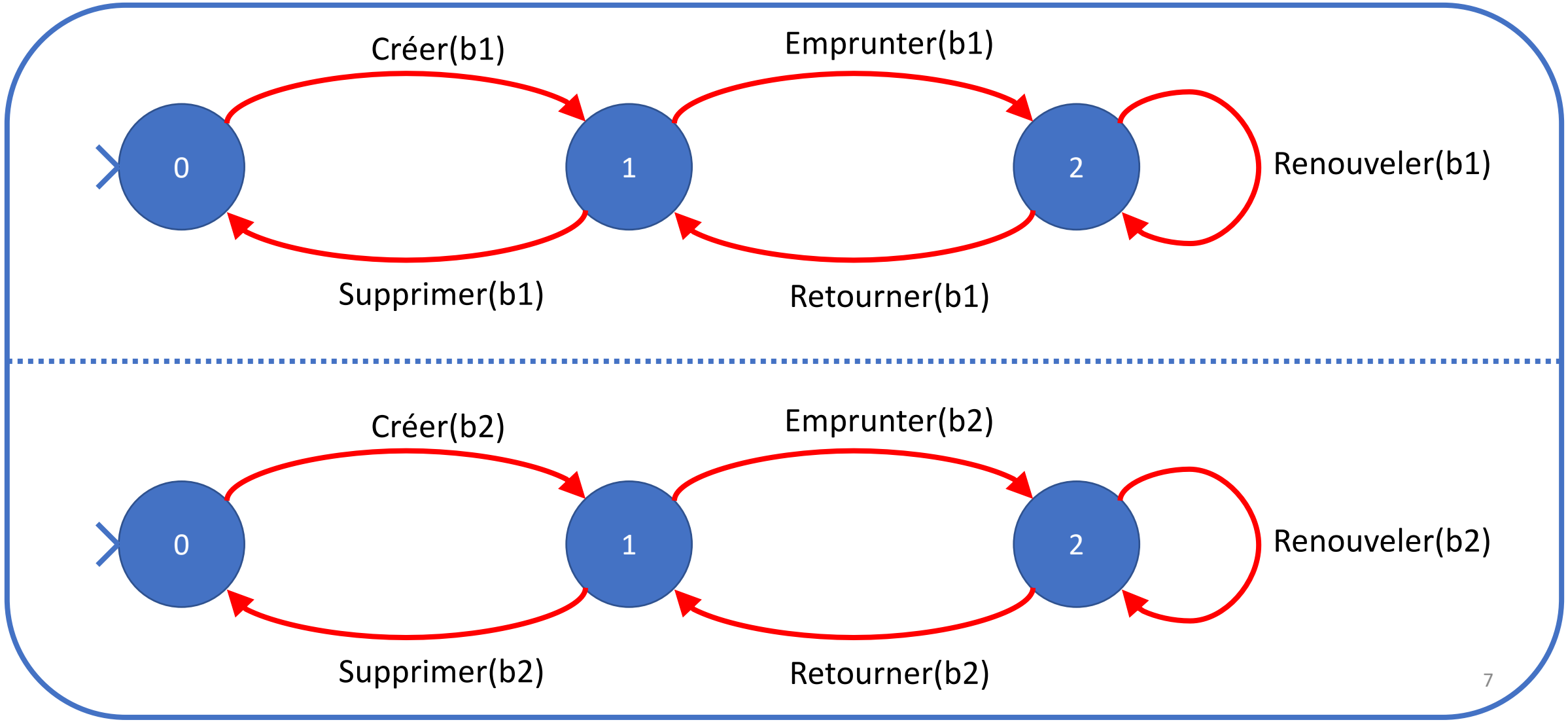
  =

  produit d’automate

  =

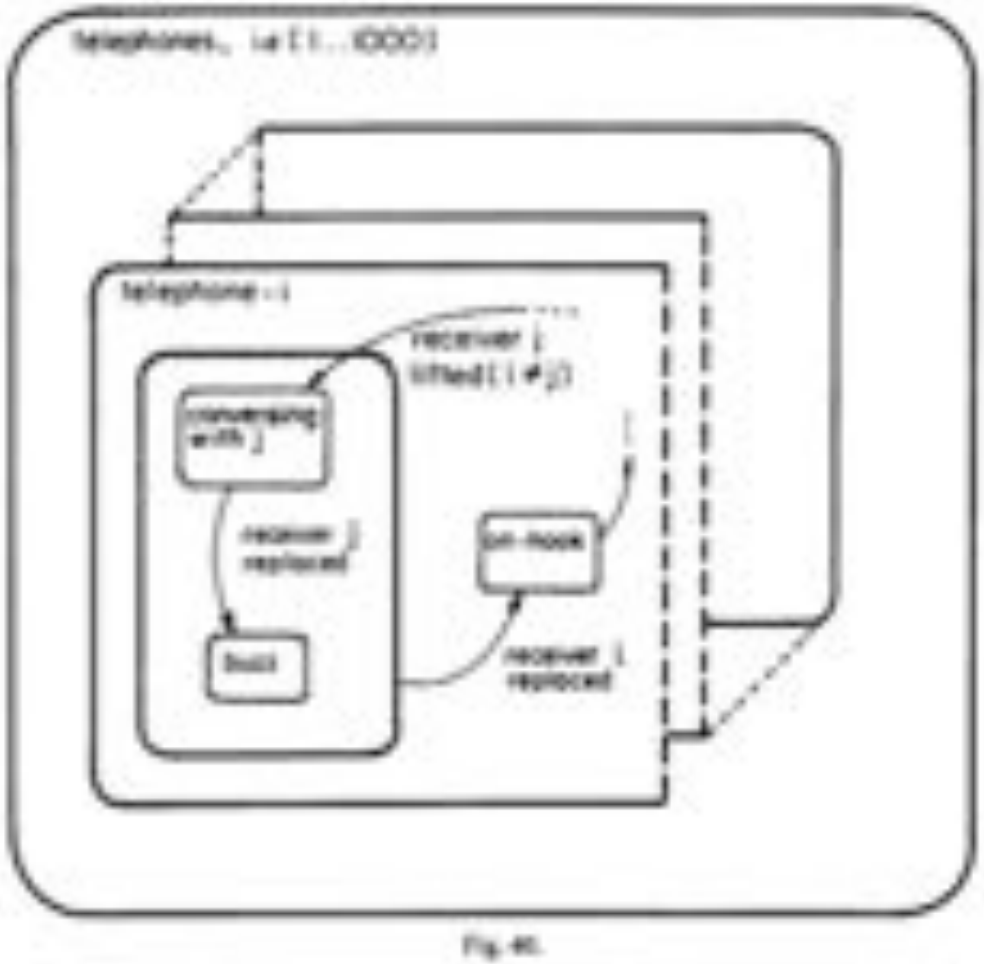
  entrelacement

Statecharts : gérer 2 livres



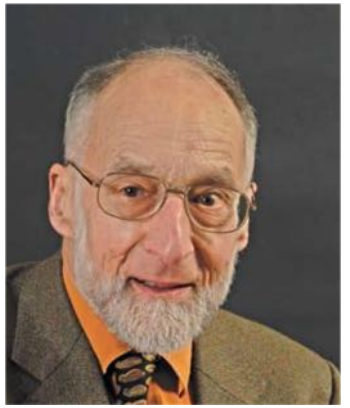
Statecharts : gérer *n* livres

- *Parameterized states*
  mentionnées dans SCP 1987
  comme une extension possible
- À ma connaissance
  - Pas supportée par UML
  - Pas supportée par les outils



Algèbre de processus (AP)

- Algèbre de processus
  - CSP : 1978 : C. A. R. (Tony) Hoare
    Communicating sequential processes
    Communications of the ACM
  - CCS : 1978 : Robin Milner †
    Algebras for communicating systems
    1er Colloque AFCET-SMF de mathématiques appliquées (Palaiseau - Paris)
  - SOS : 1980 : Gordon Plotkin
    An operational semantics for CSP
    Workshop on Logic of Programs



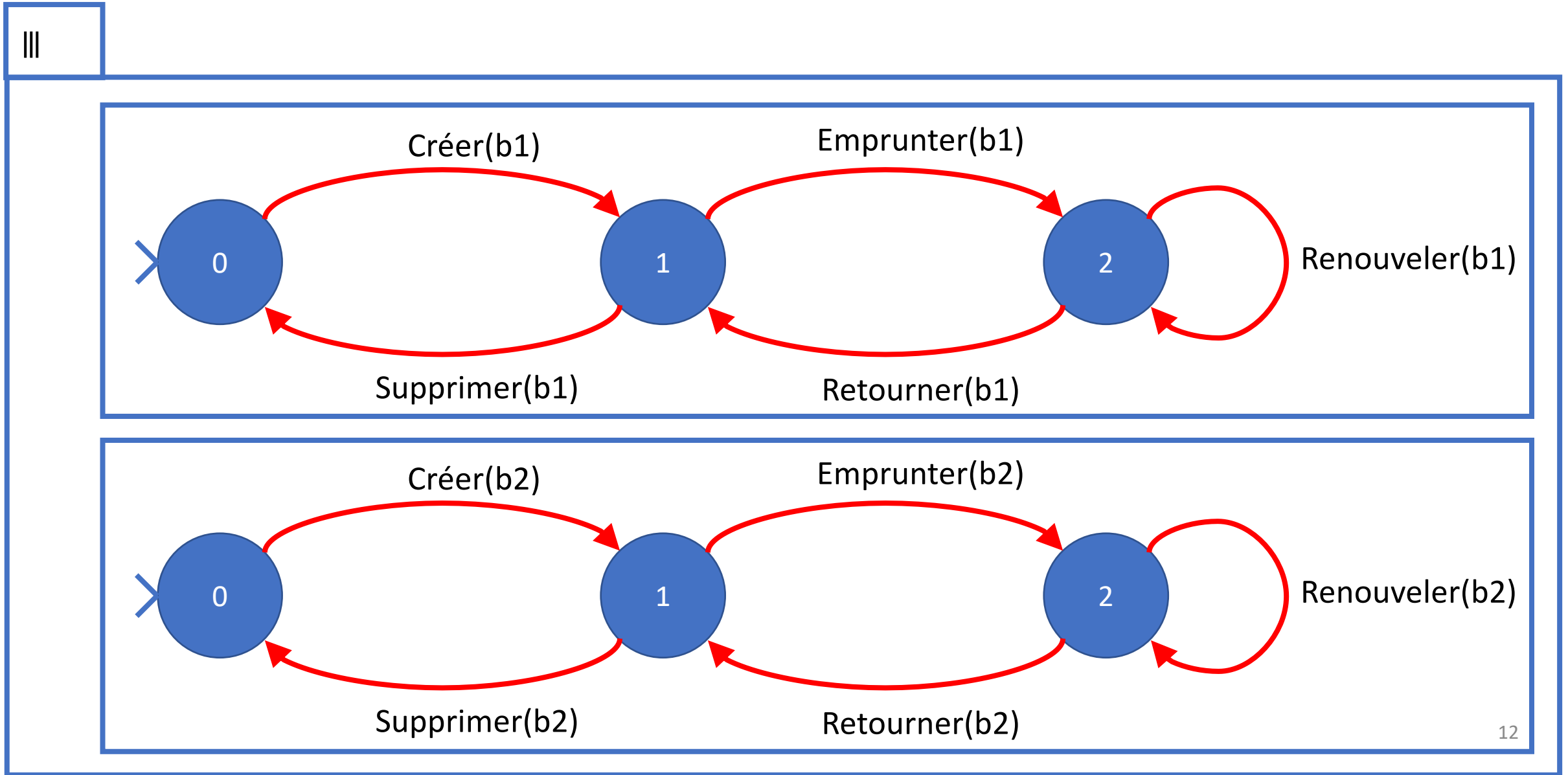
CSP – Quelques opérateurs (Hoare, Roscoe)

- SKIP — processus qui termine normalement
- STOP — processus qui ne fait rien (deadlock)
- a ➔ E — préfixe : exécute l'action a suivi du processus E
- E1 □ E2 — choix externe entre E1 et E2
- E1 ; E2 — composition séquentielle : exécution de E1 suivie de E2
- E1 ||| E2 — entrelacement de E1 et E2
- E1 || E2 — synchronisation de E1 et E2 sur les actions communes
- E1 |X| E2 — synchronisation de E1 et E2 sur les actions de X
  - E1 ||| E2 = E1 |∅| E2
  - E1 || E2 = E1 | α(E1) ∩ α(E1) | E2
- C & E — garde : exécute E si la condition C est vraie
- ||| x : T : E — entrelacement quantifié sur les valeurs de x dans T
- □x : T : E — choix externe quantifié sur les valeurs de x dans T

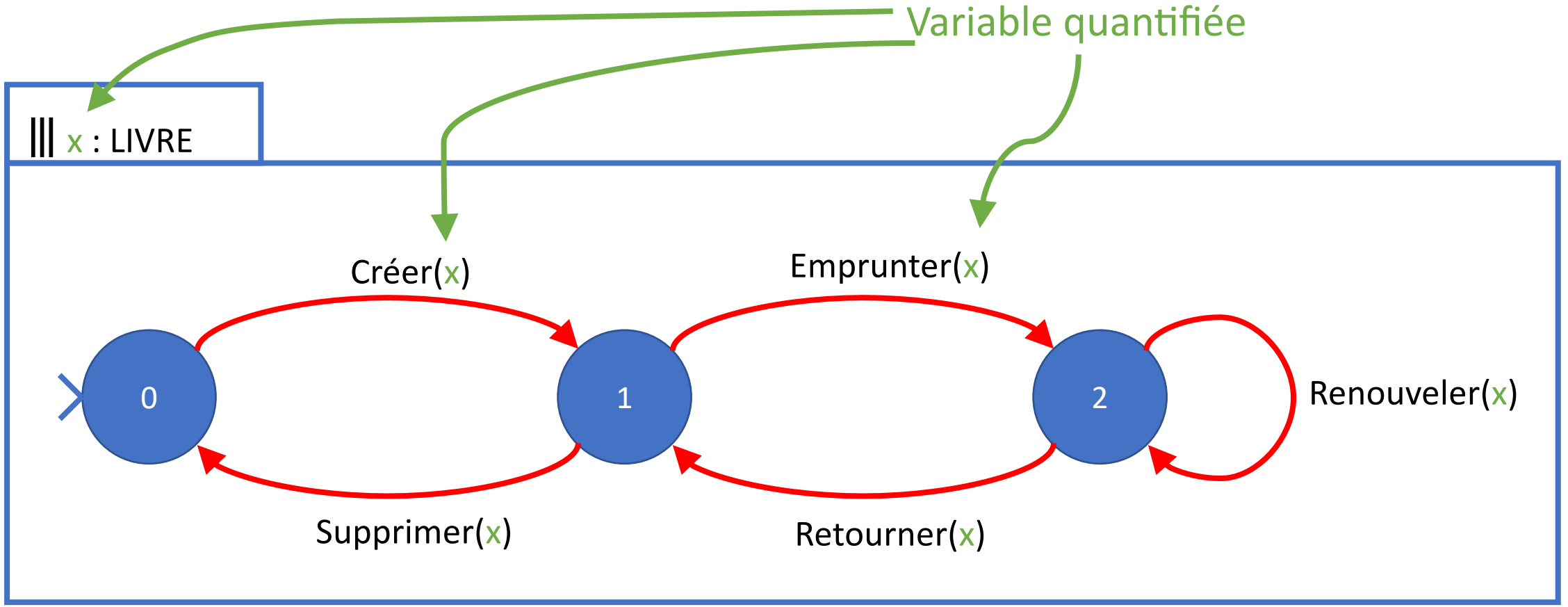
ASTD = Statecharts + AP

- Algebraic State-Transition Diagram (ASTD)
- Composer des Statecharts avec les opérateurs d'une AP
  - ASTD élémentaire = Statecharts
    - État complexe = ASTD
  - E1 | E2 choix externe entre E1 et E2
  - E1 ⇒ E2 composition séquentielle : exécution de E1 suivie de E2
  - E * fermeture de Kleene (itération arbitraire sur E)
  - E1 ||| E2 entrelacement de E1 et E2
  - E1 || E2 synchronisation de E1 et E2 sur les actions communes
  - E1 |X| E2 synchronisation de E1 et E2 sur les actions de X
  - C ⇒ E garde : exécute E si la condition C est vraie
  - ||| x : T : E entrelacement quantifié sur les valeurs de x dans T
  - |x : T : E choix externe quantifié sur les valeurs de x dans T

ASTD : gérer 2 livres d'une bibliothèque



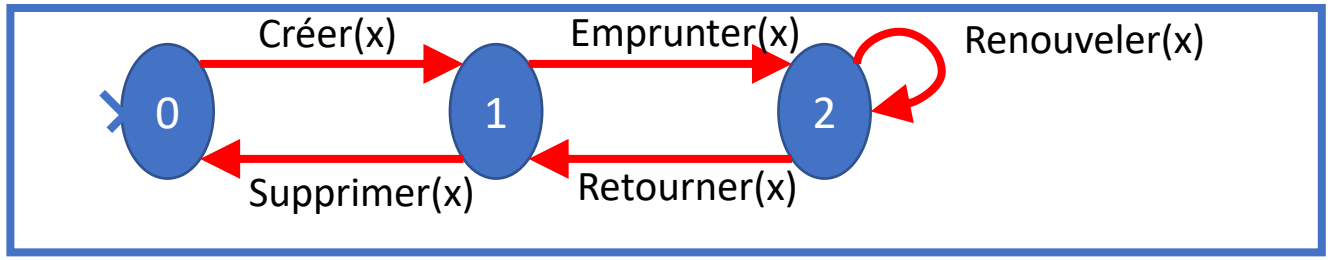
ASTD : Gérer *n* livres



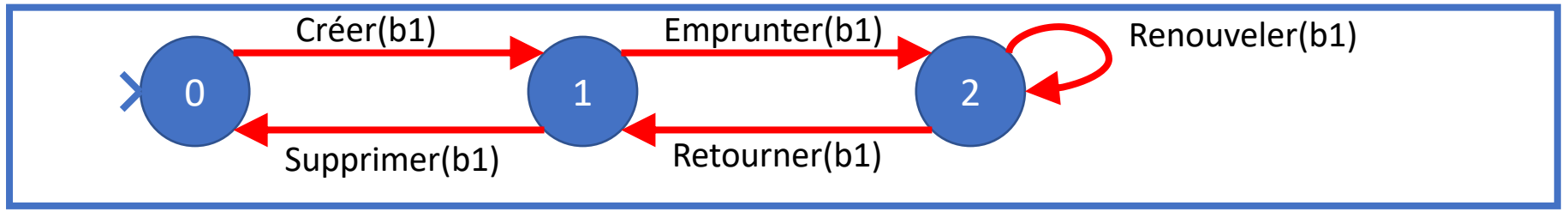
||| x : T : A Entrelacement d'instances de A indexée par x

Autant d'instances que de valeurs dans T

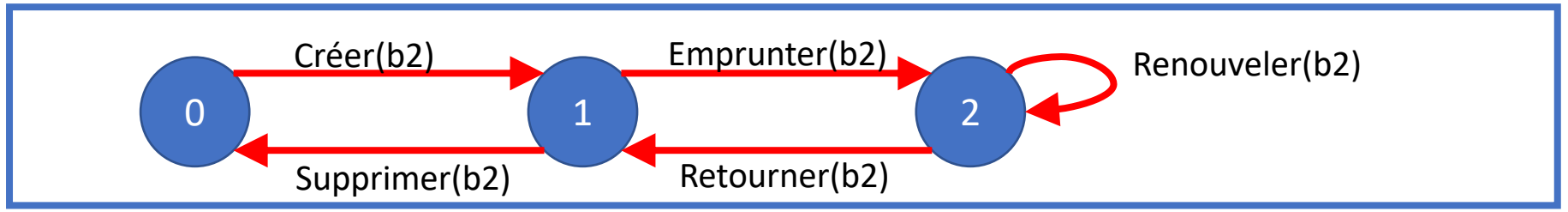
Ⅲ x : LIVRE



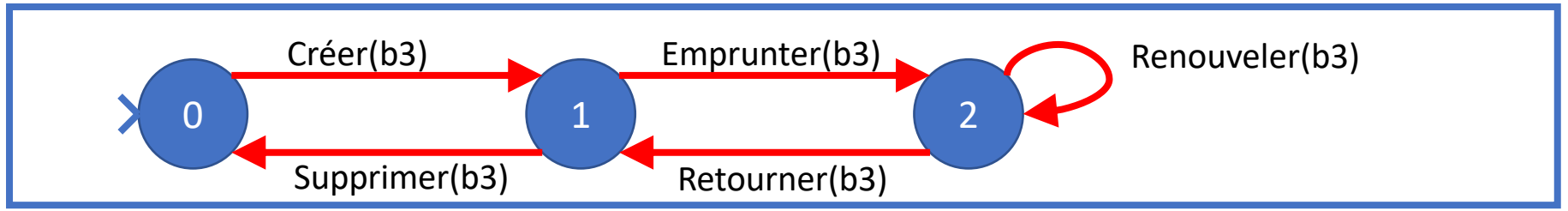
Ⅲ



x := b1



x := b2

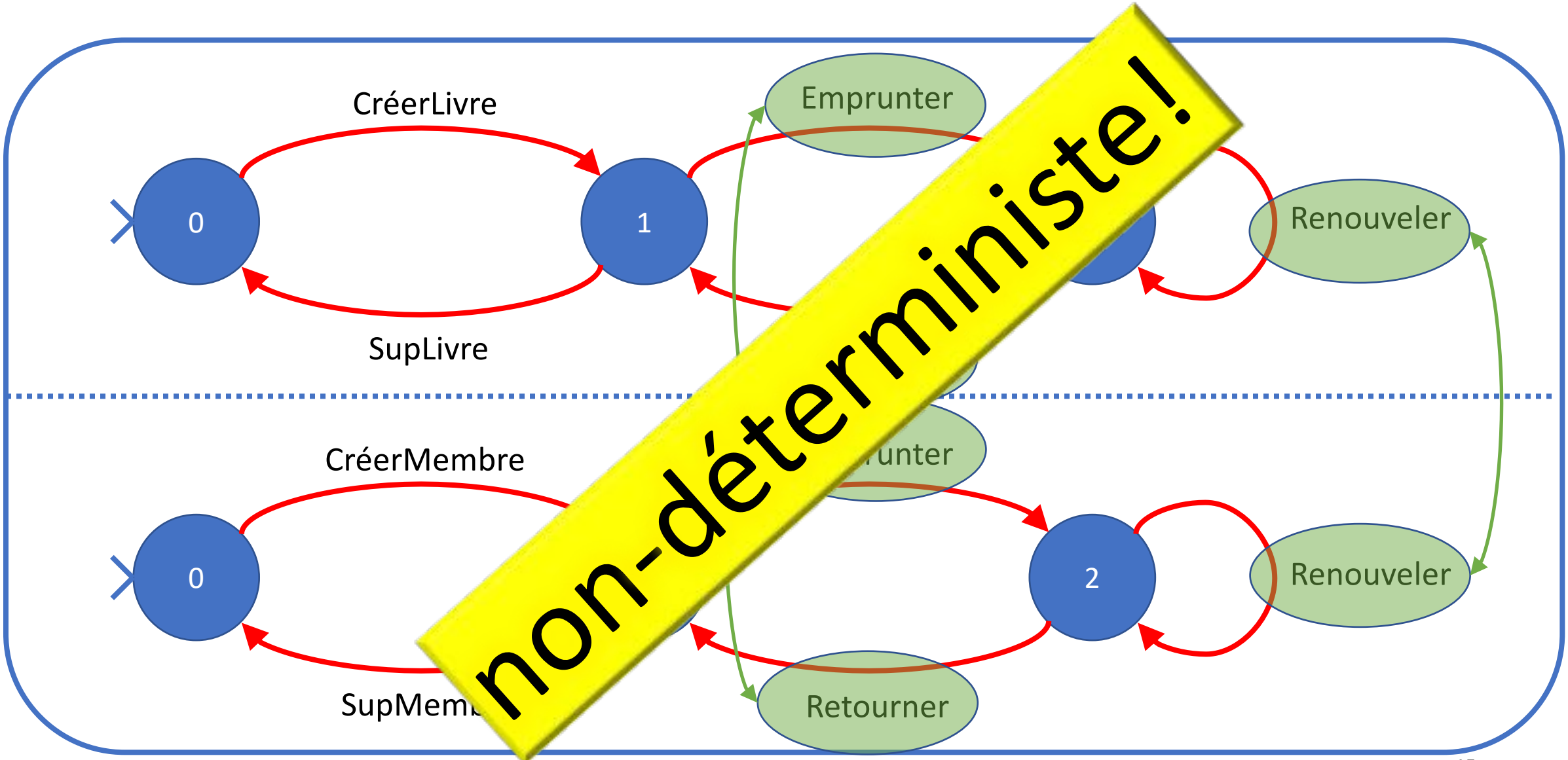


x := b3

• • •

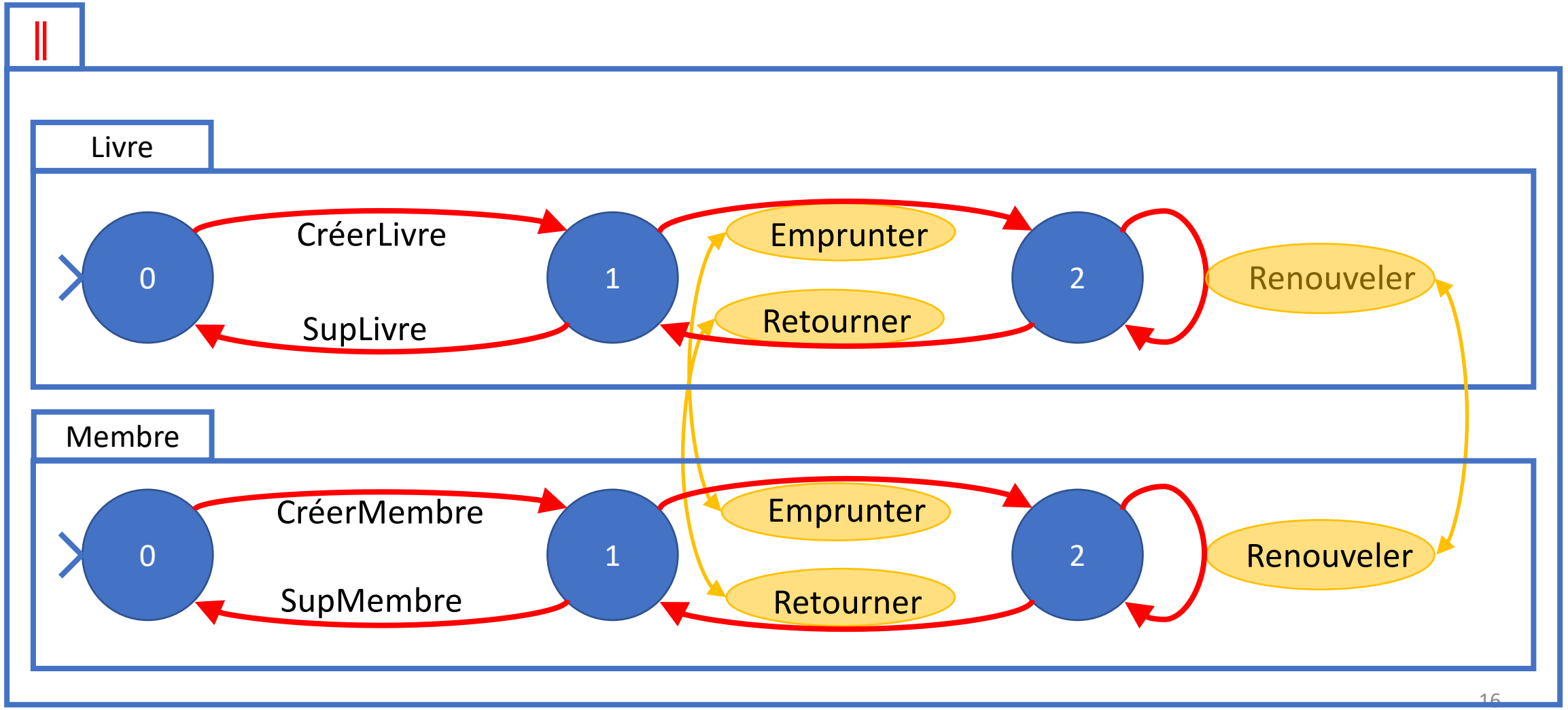
• • •

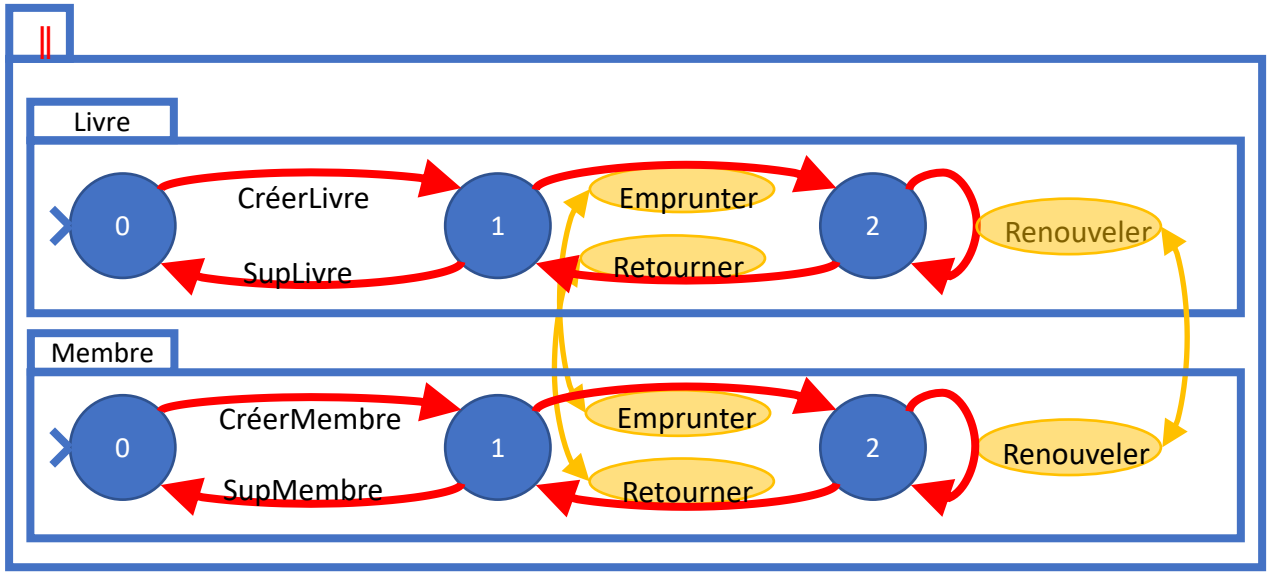
Statecharts : gérer un membre et un livre



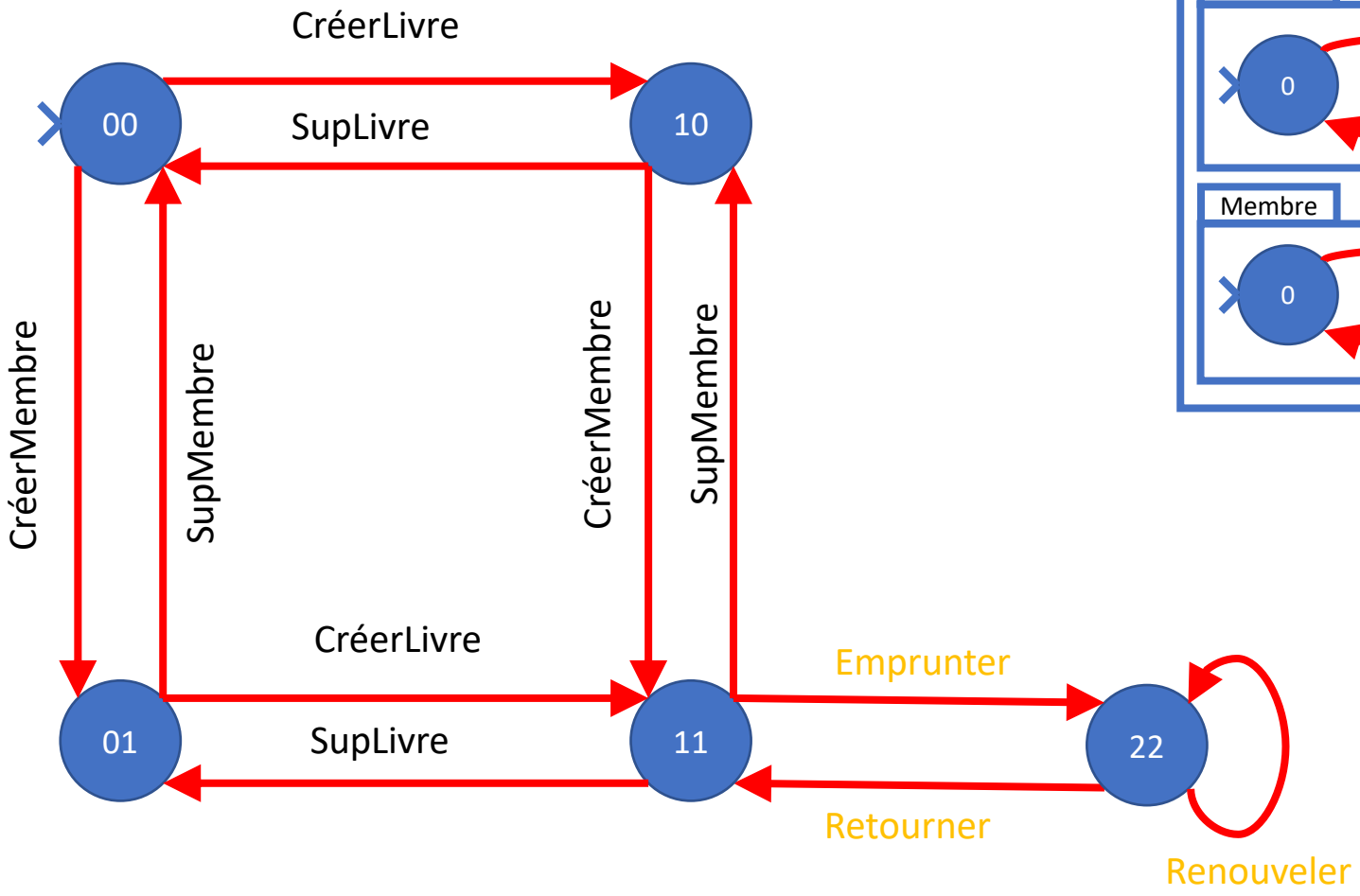
ASTD : Gérer 1 membre et 1 livre

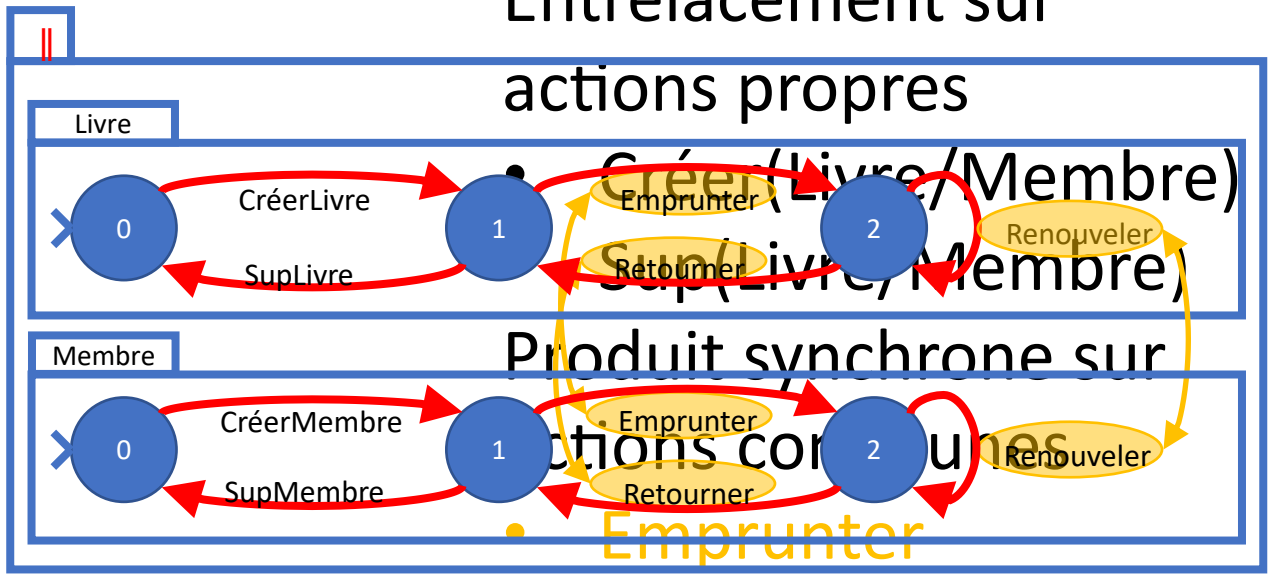
Synchronisation




Livre
0
CréerLivre
SupLivre
1
Emprunter
Retourner
2
Renouveler
Membre
0
CréerMembre
SupMembre
1
Emprunter
Retourner
2
Renouveler

ASTD : Sémantique de la synchronisation





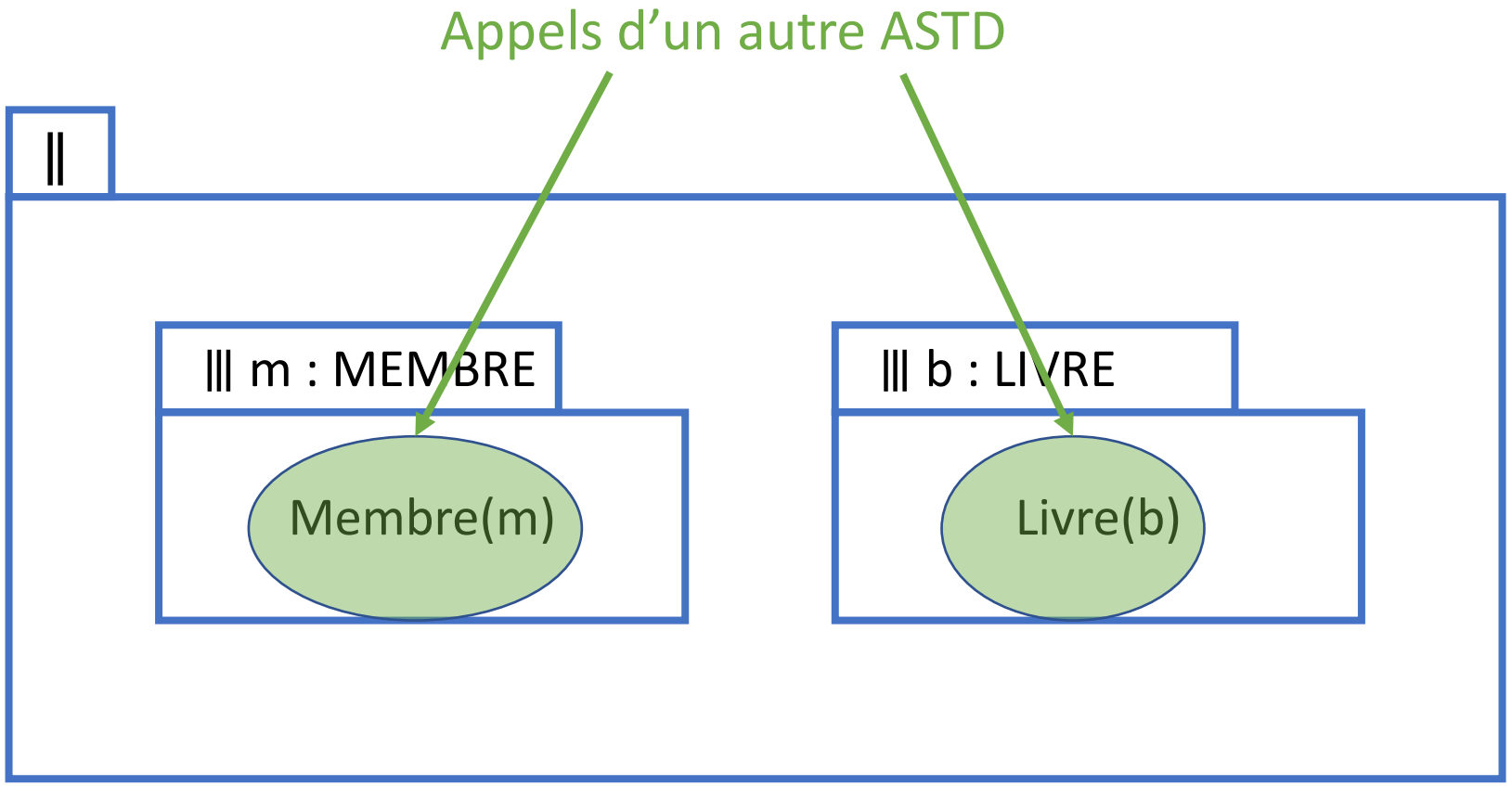
Entrelacement sur actions propres

- Créer(Livre/Membre)
- Sup(Livre/Membre)

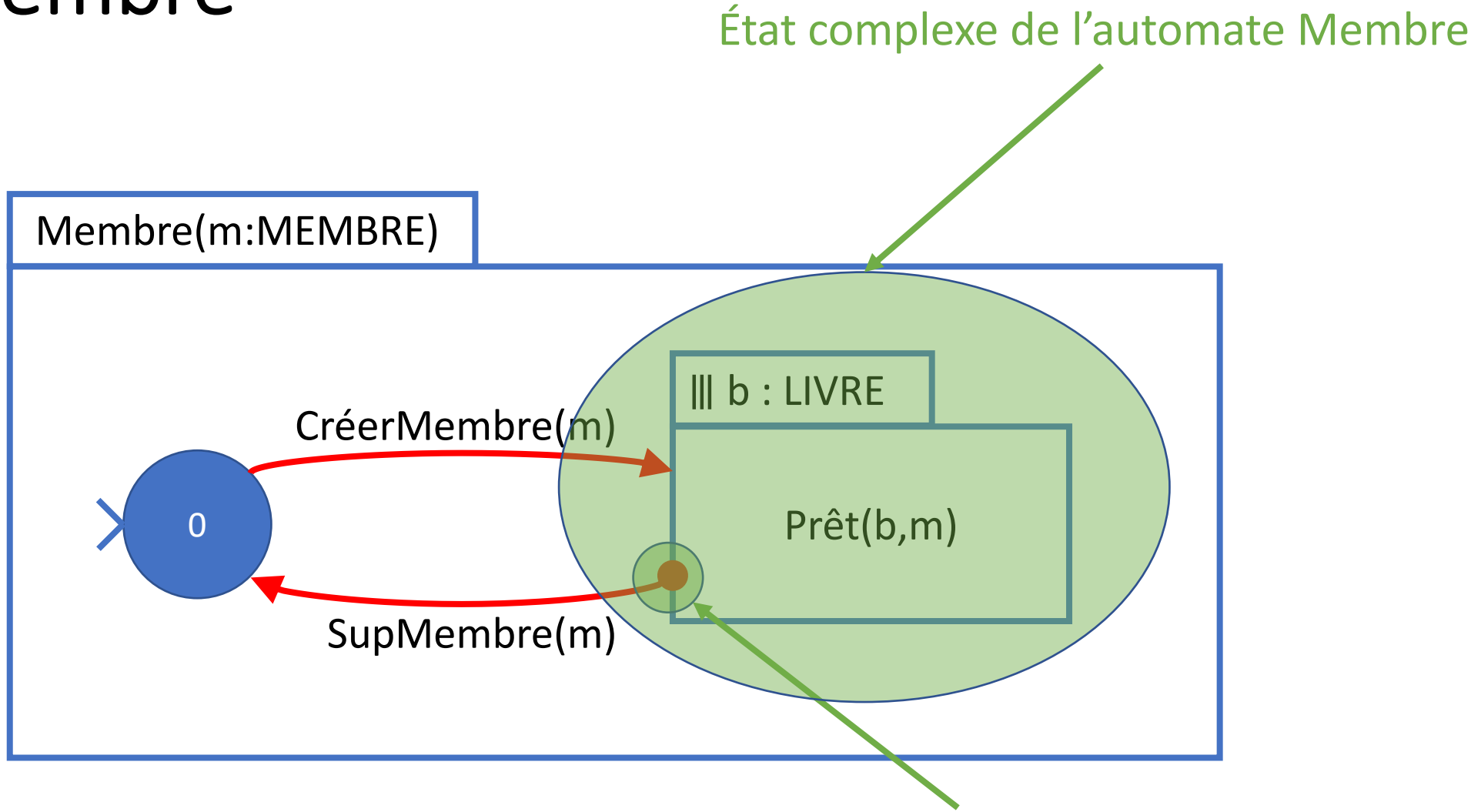
Produit synchrone sur actions communes

- Emprunter
- Renouveler
- Retourner

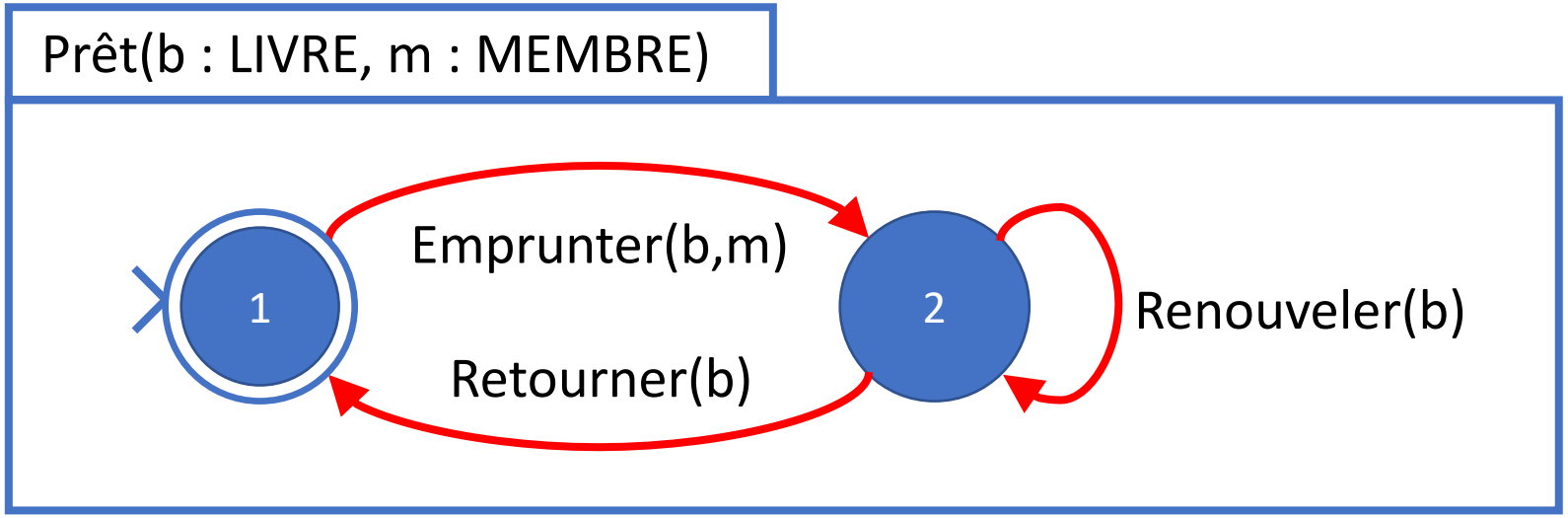
ASTD : Gérer n_1 membres et n_2 livres



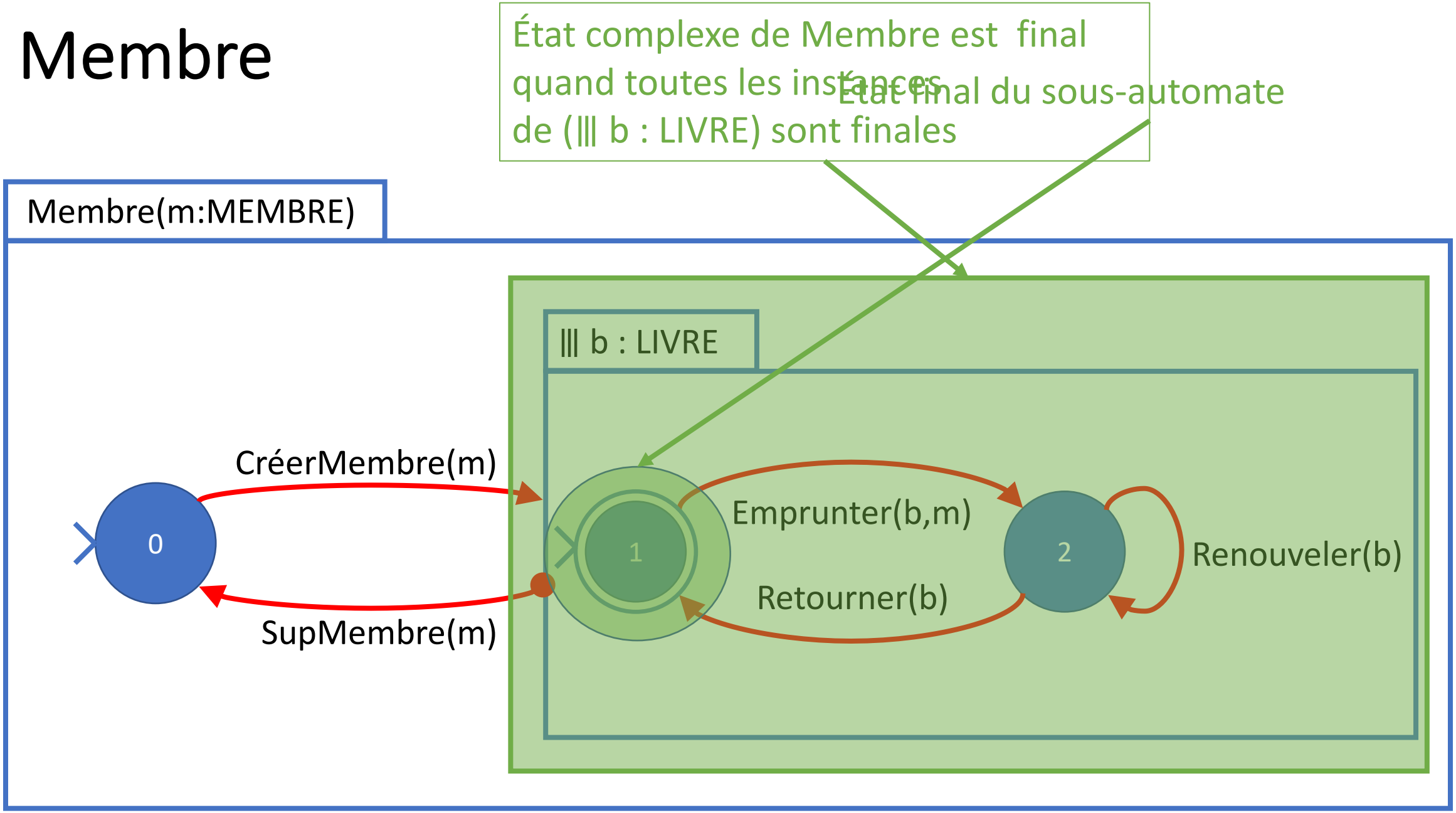
ASTD Membre


État complexe de l'automate Membre
Membre(m:MEMBRE)
||| b : LIVRE
CréerMembre(m)
0
Prêt(b,m)
SupMembre(m)
L'état source doit être dans un état final pour déclencher cette transition

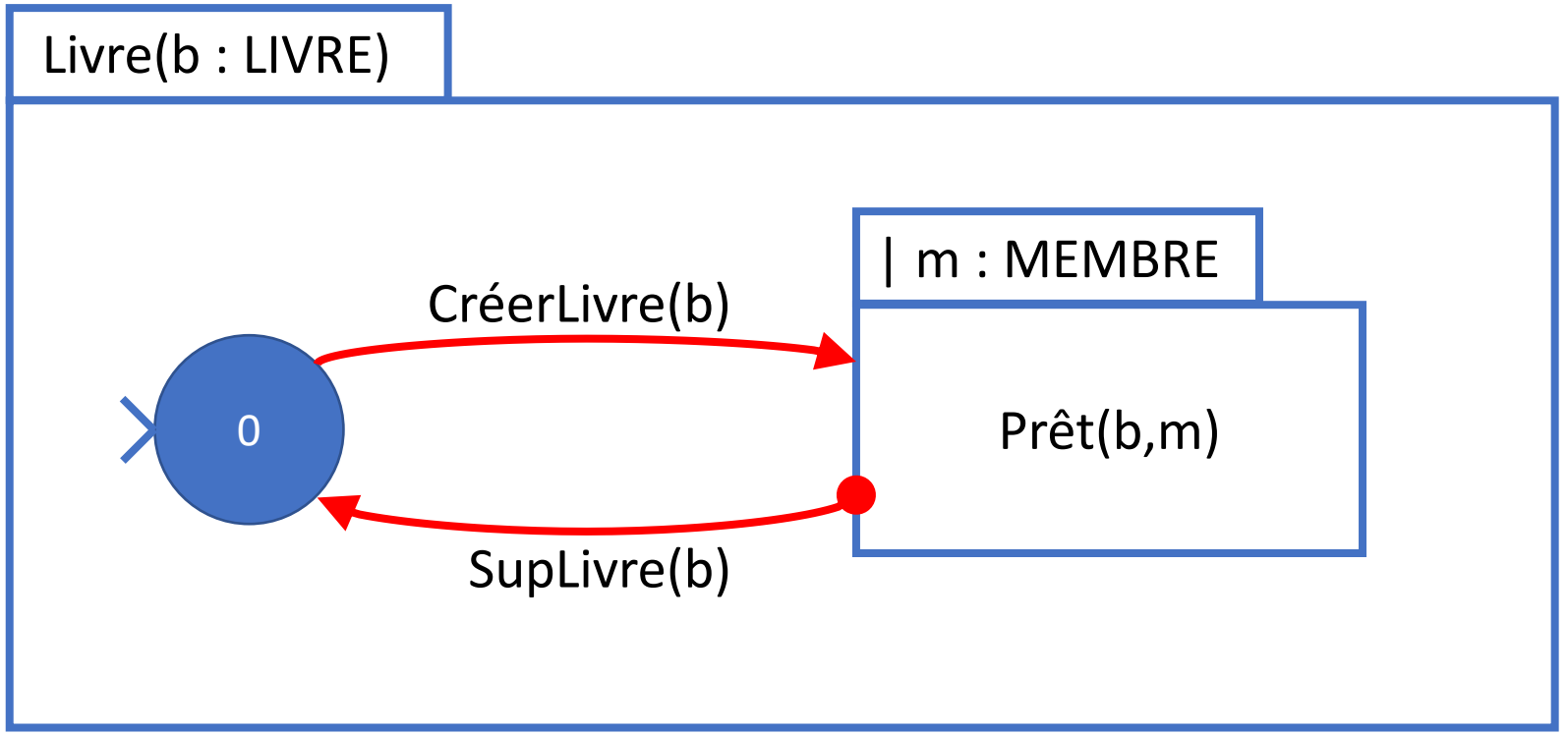
ASTD Prêt


Prêt(b : LIVRE, m : MEMBRE)
1
Emprunter(b,m)
Retourner(b)
2
Renouveler(b)

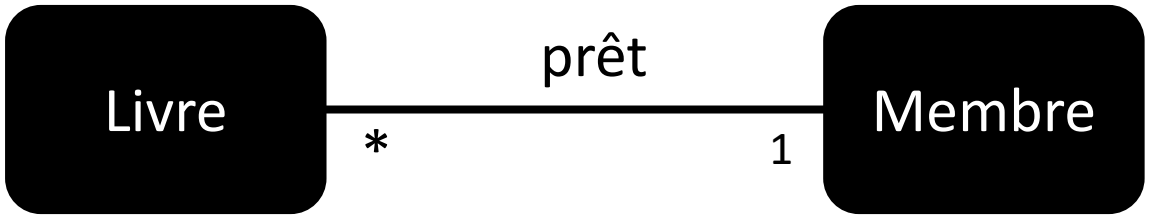
Membre


État complexe de Membre est final quand toutes les instances de (||| b : LIVRE) sont finales
État final du sous-automate
Membre(m:MEMBRE)
||| b : LIVRE
CréerMembre(m)
0
1
2
Emprunter(b,m)
Retourner(b)
Renouveler(b)
SupMembre(m)

Livre



Patron de conception : UML 2 ASTD

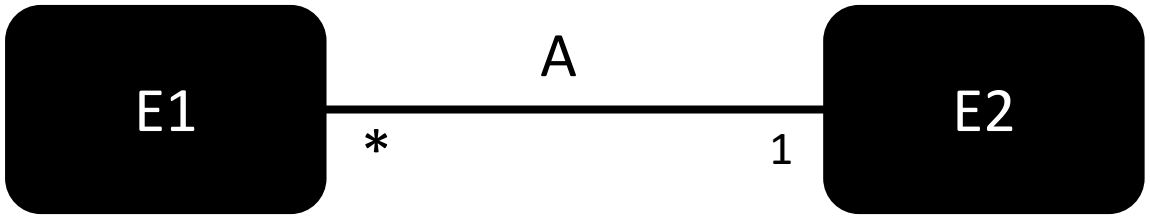



```
   ||| b : LIVRE : Livre(b)
||
   ||| m : MEMBRE : Membre(m)

Livre(b:LIVRE) =
   aut(|m : MEMBRE : Prêt(b,m))

Membre(m:MEMBRE) =
   aut(||| b : LIVRE : Prêt(b,m))
```

Patron de conception : UML 2 ASTD

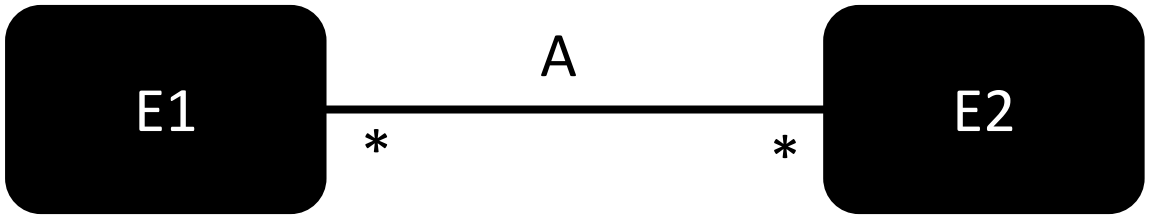



```
   ||| e1 : E! : E1(e1)
||
   ||| e2 : E2 : E2(e2)

E1(e1 : E!) =
   aut(|e2 : E2 : A(e1,e2))

E2(e2:E2) =
   aut(||| e1 : E1 : A(e1,e2))
```

Patron de conception : UML 2 ASTD

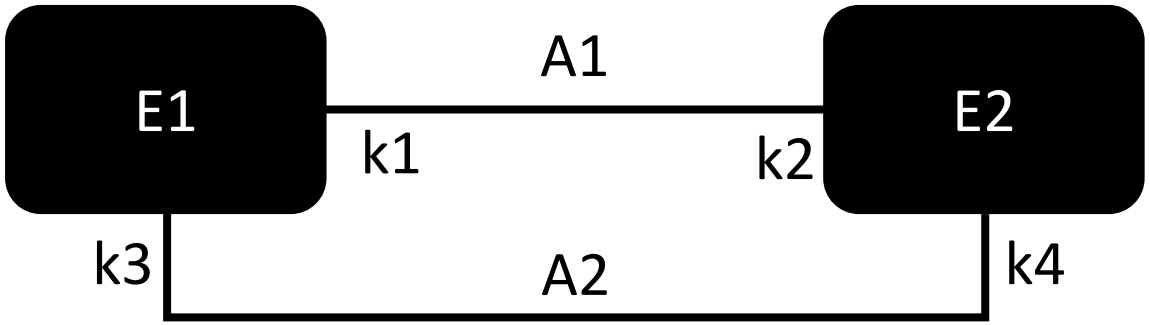



```
   ||| e1 : E! : E1(e1)
||
   ||| e2 : E2 : E2(e2)

E1(e1 : E!) =
   aut(||| e2 : E2 : A(e1,e2))

E2(e2:E2) =
   aut(||| e1 : E1 : A(e1,e2))
```

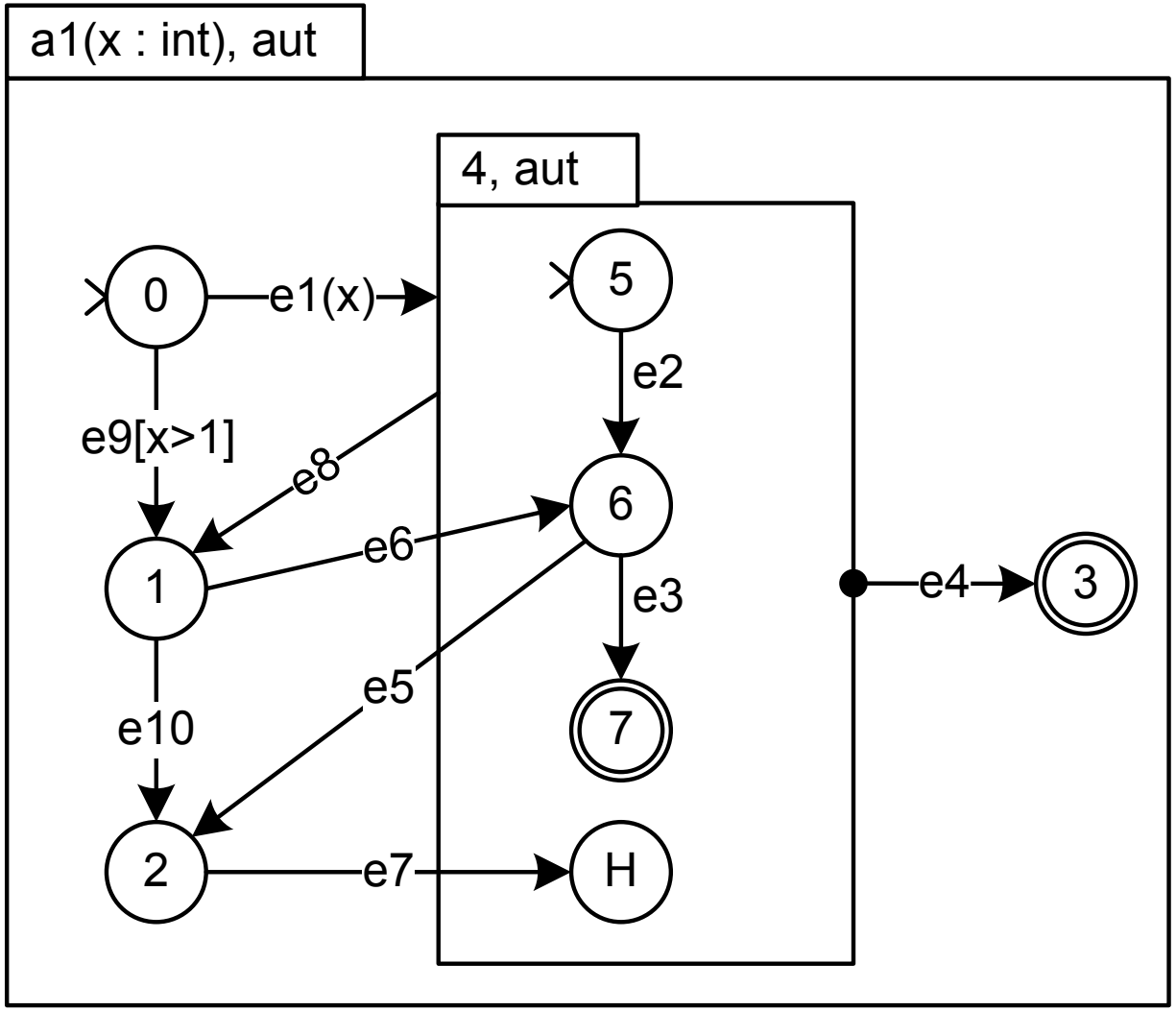
Patron de conception : UML 2 ASTD




```
E1(e1 : E1) =
  aut(
       k2 e2 : E2 : A1(e1,e2)
    ||
       k4 e2 : E2 : A2(e1,e2))

E2(e2 : E2) =
  aut(
       k1 e1 : E1 : A1(e1,e2)
    ||
       k3 e1 : E1 : A2(e1,e2))
```

Sémantique opérationnelle des ASTD



Sémantique opérationnelle des ASTD

- Init : ASTD ⟶ State
  - Donne l'état initial d'un ASTD
- Final : State ⟶ BOOL
  - Détermine si un état est final

State =
$$\langle \mathsf{aut}_\circ, n, h, s \rangle$$
$$\mid \langle |[]|, n, \Delta, l, r \rangle$$
$$\mid \langle |:_\circ, [\bot \mid v], [\bot \mid s] \rangle$$
$$\mid \langle |[]|:, n, x, T, \Delta, b \rangle$$
$$\mid \ \ldots$$

Sémantique et règles d'inférence

State $\times$ Event $\times$ State

$$s \xrightarrow{\sigma}_{\mathsf{a}} s' \qquad\qquad s \xrightarrow{\sigma,\Gamma}_{\mathsf{a}} s'$$

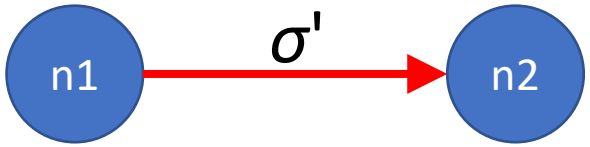
$$\text{env}\ \frac{s \xrightarrow{\sigma,([])} s'}{s \xrightarrow{\sigma} s'}$$

Sémantique automate

$$
\begin{aligned}
init((\mathsf{aut}, \ldots)) &\triangleq (\mathsf{aut}_\circ, n_0, h_{init}, init(\nu(n_0))) \\
h_{init} &\triangleq \{n \mapsto init(\nu(n)) \mid n \in N\} \\
final((\mathsf{aut}_\circ, n, h, s)) &\triangleq (n \in DF \wedge final(s)) \\
&\quad \vee \\
&\quad (n \in SF)
\end{aligned}
$$

$$
\begin{aligned}
\Psi \triangleq (\quad & (final? \Rightarrow final(s)) \\
\wedge \quad & g \\
\wedge \quad & \sigma' = \sigma \\
\wedge \quad & h' = h \lhd\!+ \{n_1 \mapsto s\} \\
) \quad & [\Gamma]
\end{aligned}
$$

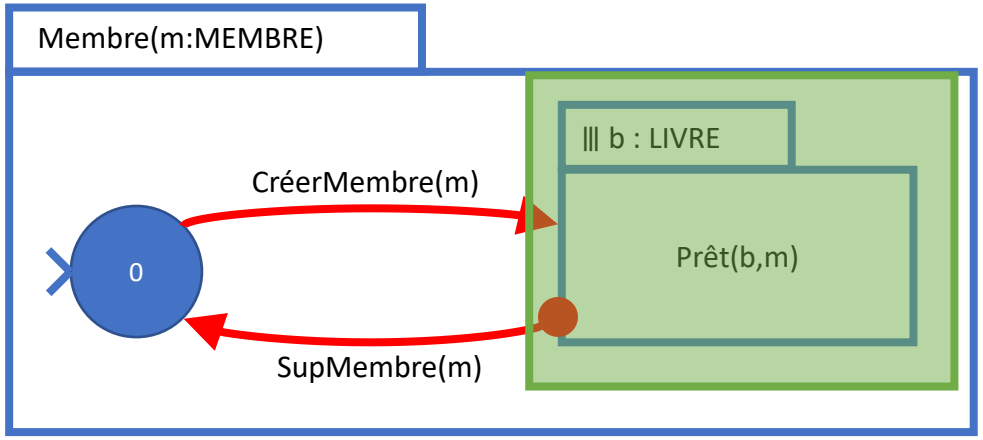
Transition entre états



$$
\mathsf{aut}_1 \ \frac{\delta((\mathsf{loc}, n_1, n_2), \sigma', g, final?) \qquad \Psi}{(\mathsf{aut}_\circ, n_1, h, s) \xrightarrow{\sigma,\Gamma} (\mathsf{aut}_\circ, n_2, h', init(\nu(n_2)))}
$$

Transition à l'intérieur d'un état complexe

$$
\mathsf{aut}_6 \ \frac{s \xrightarrow{\sigma,\Gamma}_{\nu(n)} s'}{(\mathsf{aut}_\circ, n, h, s) \xrightarrow{\sigma,\Gamma} (\mathsf{aut}_\circ, n, h, s')}
$$



Sémantique parallèle

$$
\begin{array}{rcl}
init((|[]|, n, \Delta, l, r))) & \triangleq & (|[]|_\circ, init(l), init(r)) \\
final((|[]|_\circ, s_l, s_r)) & \triangleq & final(s_l) \wedge final(s_r)
\end{array}
$$

$$
|[]|_1 \ \frac{\alpha(\sigma) \notin \Delta \qquad s_l \xrightarrow{\sigma,\Gamma}_l s'_l}{(|[]|_\circ, s_l, s_r) \xrightarrow{\sigma,\Gamma} (|[]|_\circ, s'_l, s_r)}
$$

$$
|[]|_2 \ \frac{\alpha(\sigma) \notin \Delta \qquad s_r \xrightarrow{\sigma,\Gamma}_r s'_r}{(|[]|_\circ, s_l, s_r) \xrightarrow{\sigma,\Gamma} (|[]|_\circ, s_l, s'_r)}
$$

$$
|[]|_3 \ \frac{\alpha(\sigma) \in \Delta \qquad s_l \xrightarrow{\sigma,\Gamma}_l s'_l \qquad s_r \xrightarrow{\sigma,\Gamma}_r s'_r}{(|[]|_\circ, s_l, s_r) \xrightarrow{\sigma,\Gamma} (|[]|_\circ, s'_l, s'_r)}
$$

Sémantique parallèle quantifié

$$init((|[]|:, n, x, T, \Delta, b)) \triangleq (|[]|:_\circ, T \times \{init(b)\})$$
$$final((|[]|:_\circ, f)) \triangleq \forall\, v : T \cdot final(f(v))$$

$$|[]|:_1 \ \frac{\alpha(\sigma) \notin \Delta \qquad f(v) \xrightarrow{\sigma, (\![x:=v]\!) \lhd \Gamma}_b s'}{(|[]|:_\circ, f) \xrightarrow{\sigma,\Gamma} (|[]|:_\circ, f \lhd\!+ \{v \mapsto s'\})}$$

$$|[]|:_2 \ \frac{\alpha(\sigma) \in \Delta \qquad \forall\, v : T \cdot f(v) \xrightarrow{\sigma, (\![x:=v]\!) \lhd \Gamma}_b f'(v)}{(|[]|:_\circ, f) \xrightarrow{\sigma,\Gamma} (|[]|:_\circ, f')}$$

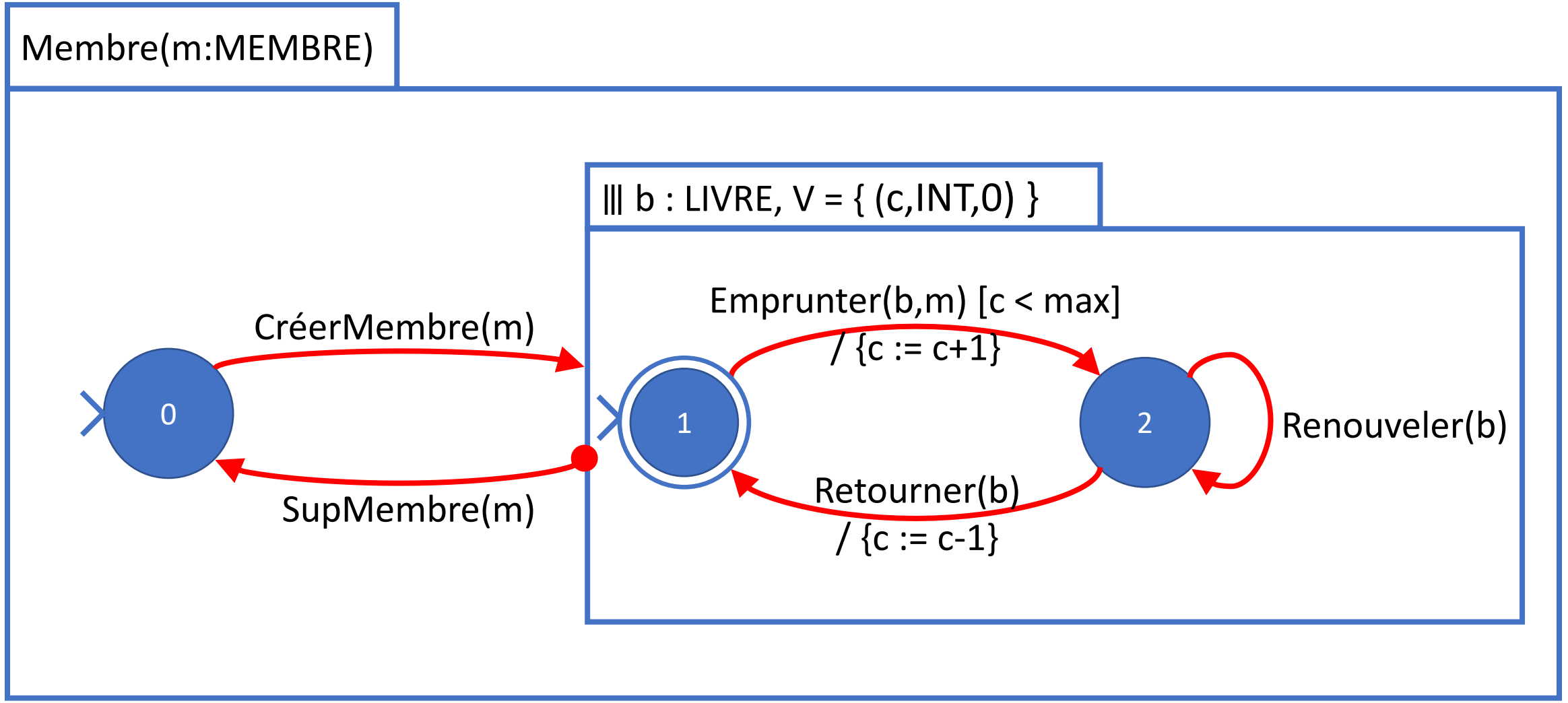
xASTD = ASTD + variables d'état

- Déclaration de variables dans un ASTD

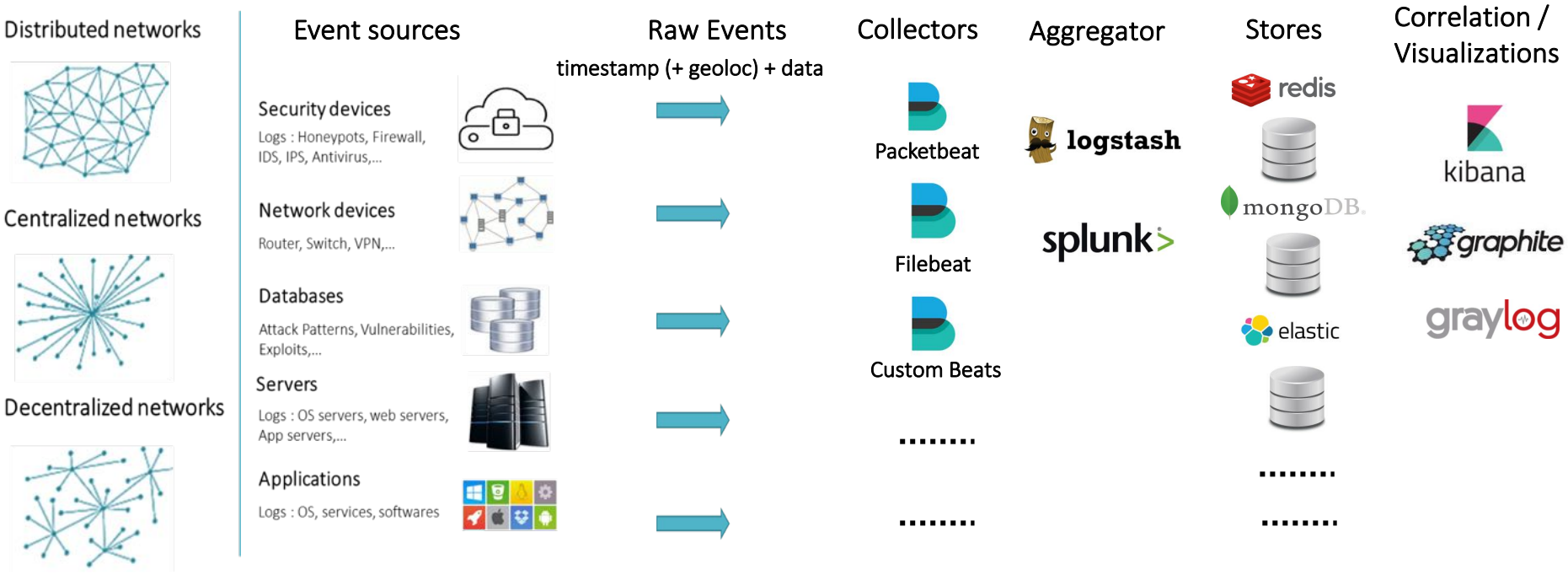
$$|||\ x : T\ ,\ V = \{\ (y,\ T,\ init)\ \}$$

- x : variable de quantification
  - en lecture seulement
  - portée = sous-ASTD
  - en lecture dans les gardes et les actions
- y : variable d'état
  - portée = sous-ASTD
  - modifiable dans les actions
  - en lecture dans les gardes

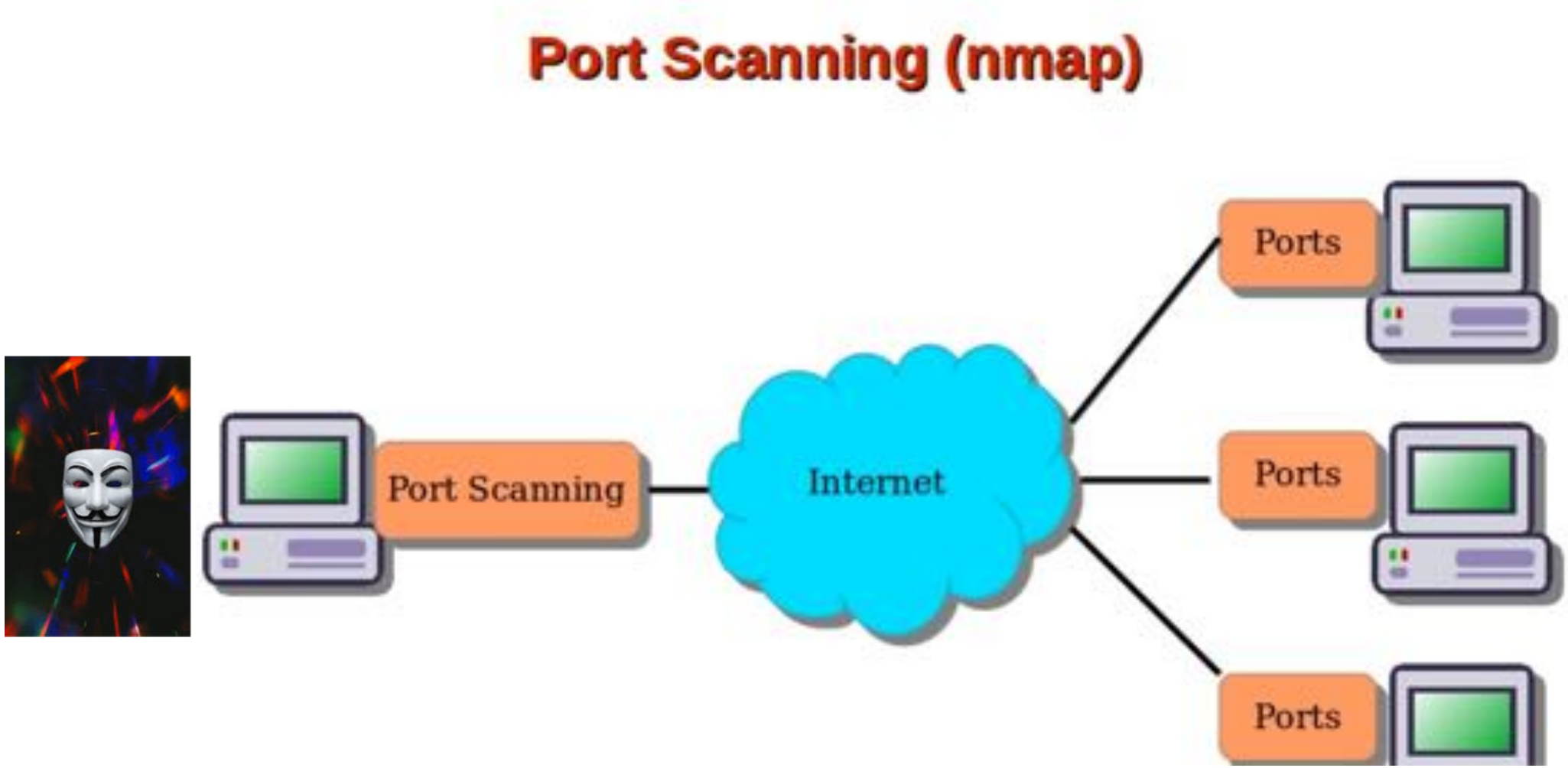
Compter le nombre de prêts



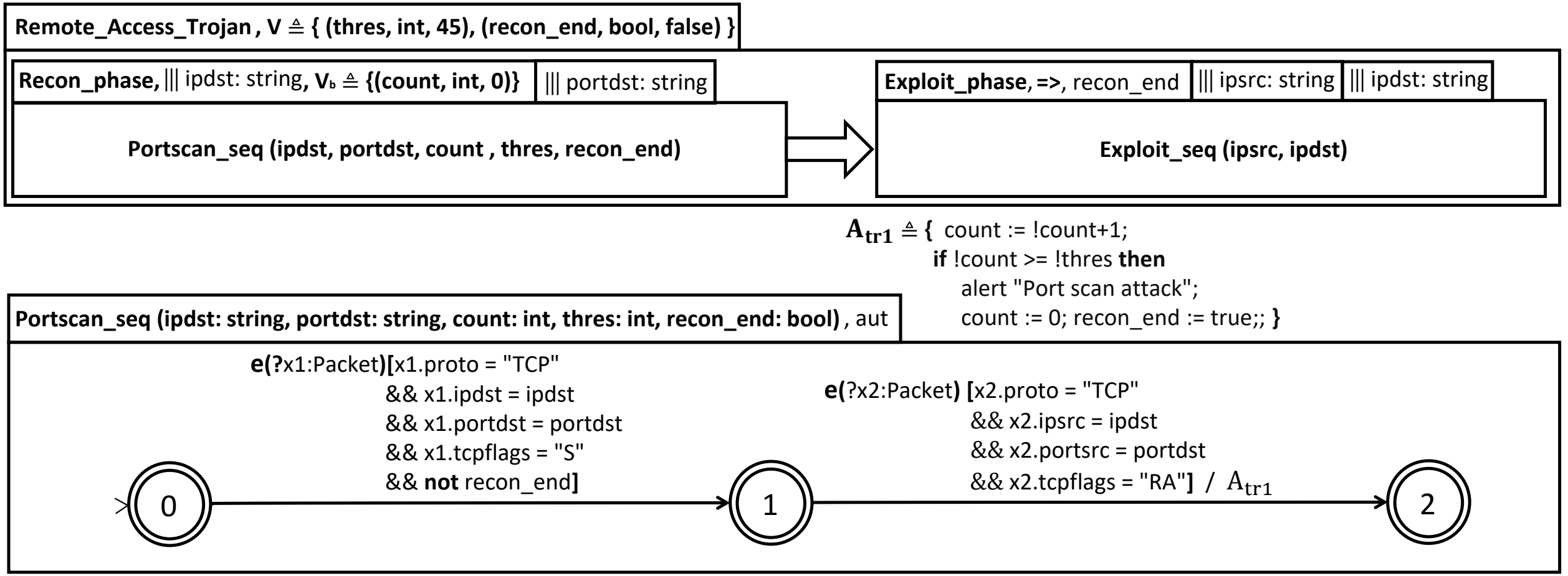
Détection d'intrusion



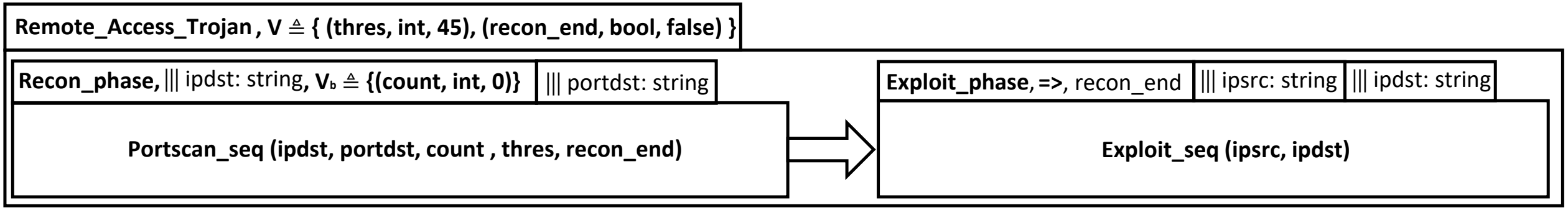
Spécification d’attaque



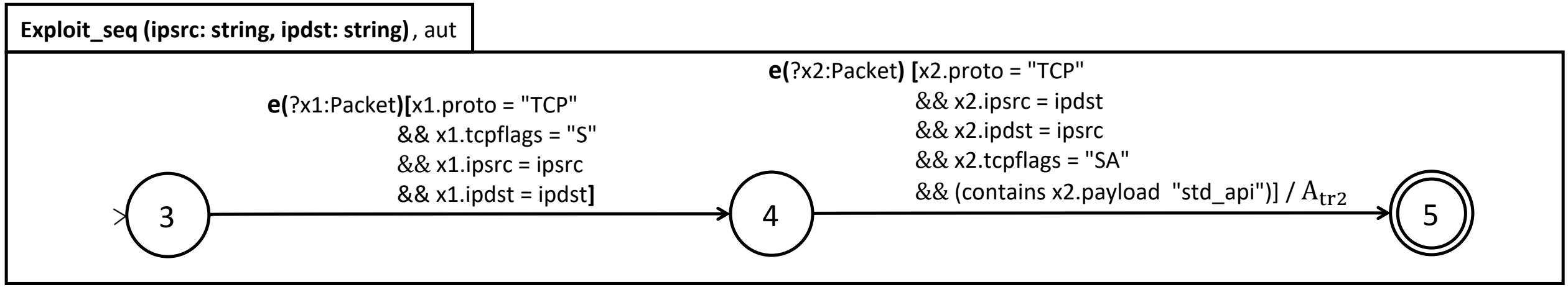
Exemple : détection d'attaques



Détection d’attaques (suite)



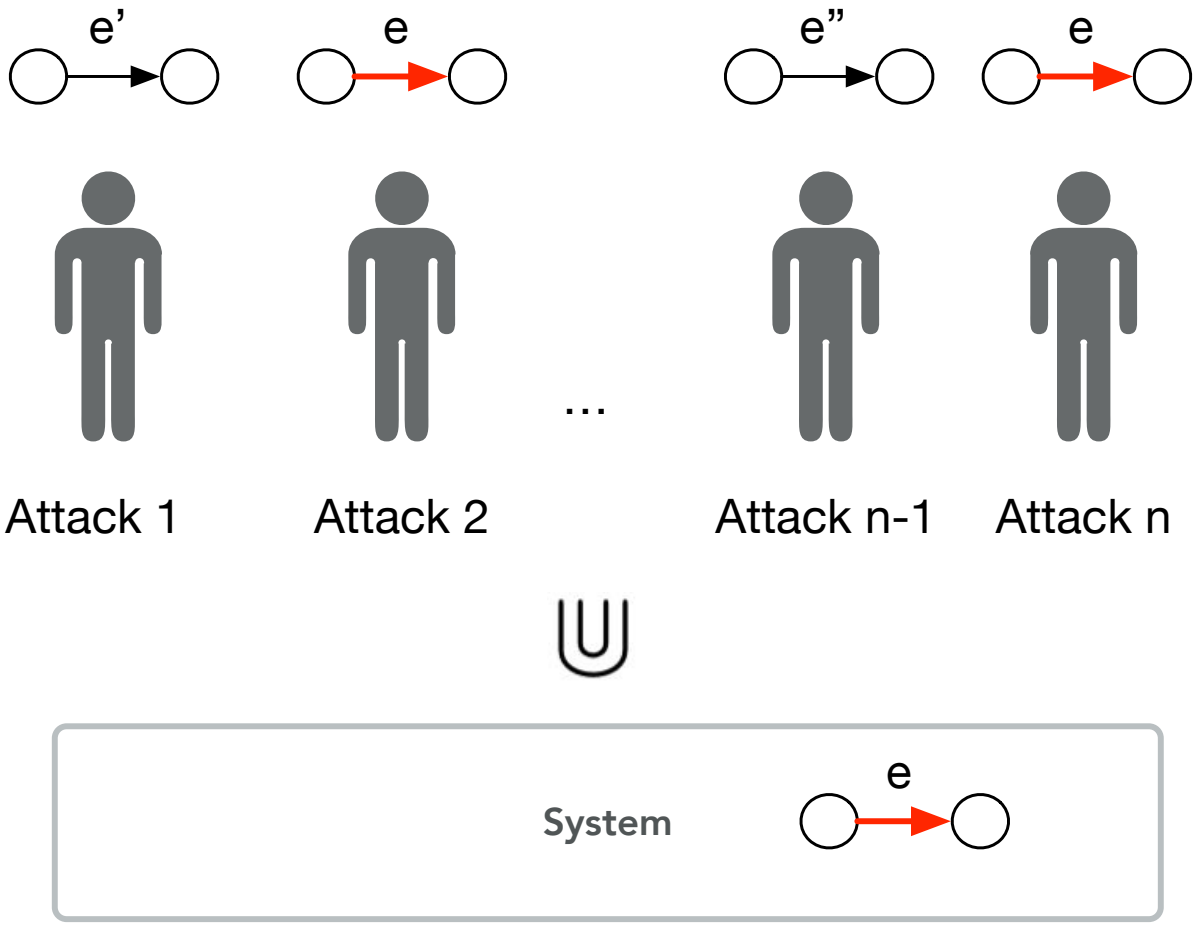
$\mathbf{A_{tr2}}$ ≜ { alert "Metasploit privilege escalation";; }



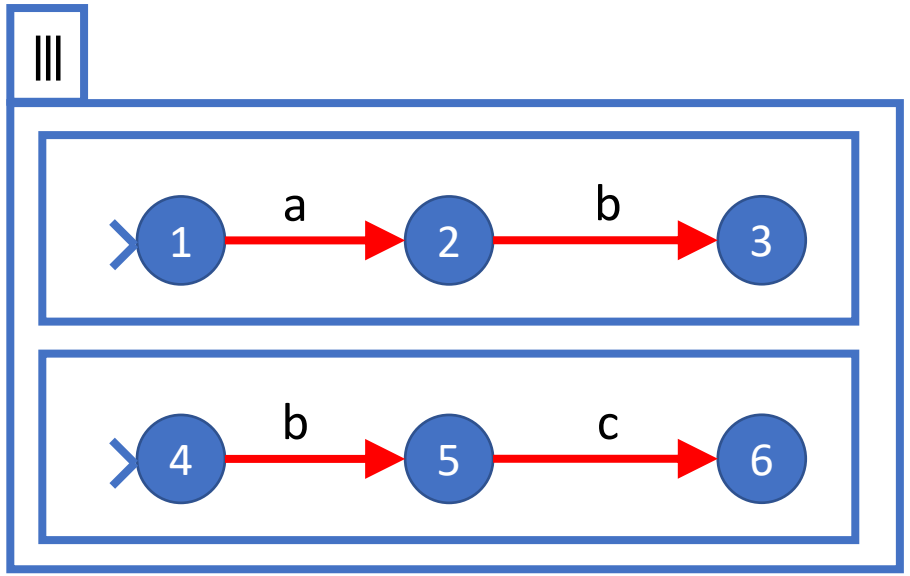
xASTD : Nouvel opérateur : flow ⋓

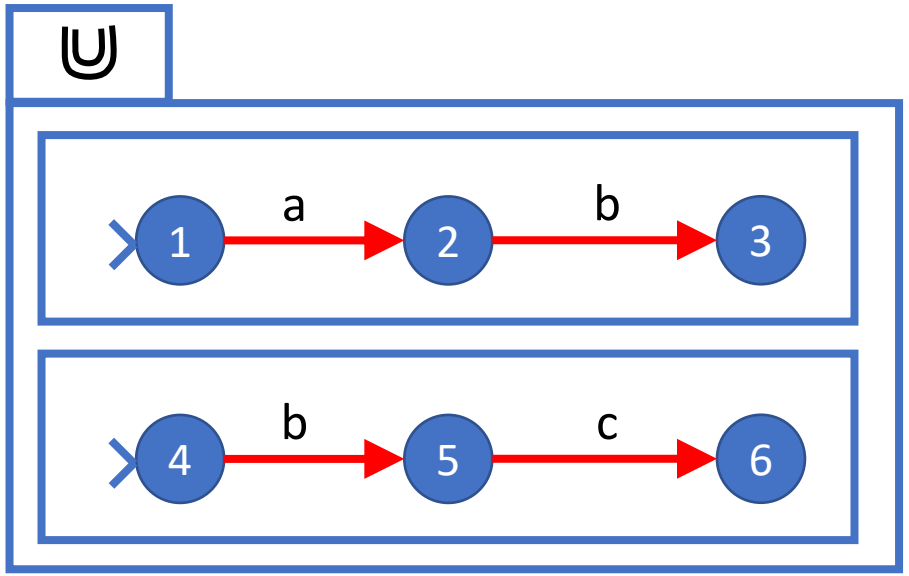
- Un événement peut appartenir à plusieurs specs d'attaque
- Soit S1, …, Sn des specs d'attaque
- S1 ||| … ||| Sn
  - une seule specs exécutera un événement e
  - souvent non-déterministe
- S1 || … || Sn : toutes les specs accepter e pour qu'il puisse s'exécuter
- Aucun des opérateurs n'est adéquat
- Nouvel opérateur flow
  - S1 ⋓ … … ⋓ Sn
  - chaque qui peut exécuter un événement l'exécute
  - déterministe

Flow



Flow ⋓





Peut refuser la trace abc
Non-déterminisme sur b

(1,4) –a-> (2,4) –b-> (3,4) –c-/>

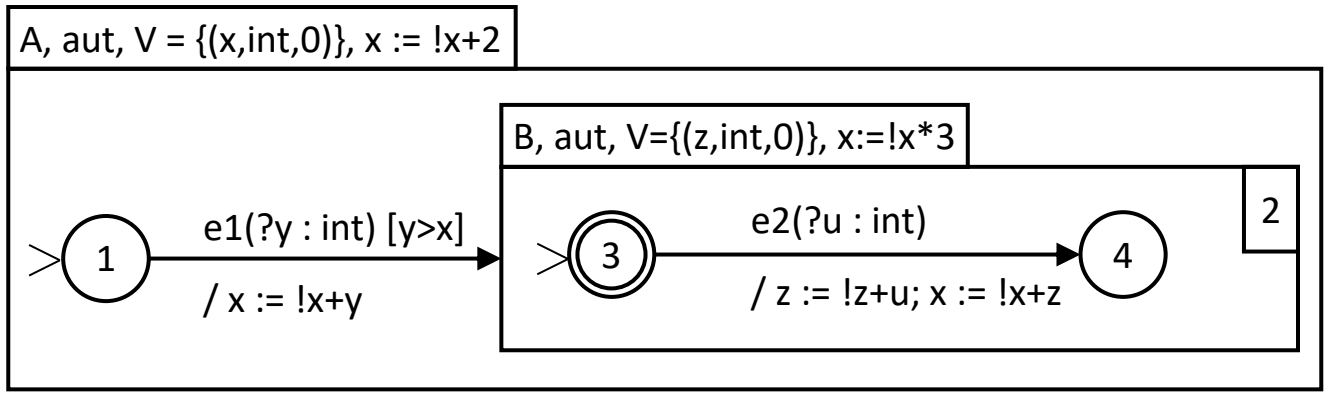
(1,4) –a-> (2,4) –b-> (2,5) –c-> (2,6)

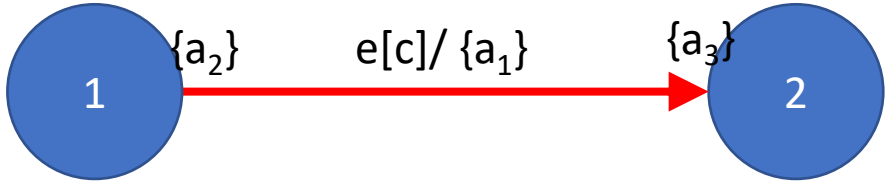
Accepte la trace abc
Déterministe

(1,4) –a-> (2,4) –b-> (3,5) –c-> (3,6)

xASTD : Autres caractéristiques

- Actions sur les ASTD
  - Factorisation du code commun à toutes les transitions de l'ASTD
- Actions sur les états
  - entry code, stay code, exit code





Sémantique de xASTD

$$\Omega_{loc} \triangleq \left\{ \begin{array}{c} \textbf{if } n_1 = n_2 \textbf{ then } A = A_{tr} \ ; \ a.\zeta(n_1).A_{stay} \ ; \ a.A_{astd} \\ \textbf{else } A = a.\zeta(n_1).A_{out} \ ; \ A_{tr} \ ; \ a.\zeta(n_2).A_{in} \ ; \ a.A_{astd} \textbf{ end} \\ E_g = E_e \lhd\!\!\!+ E \\ A(E_g, E'_g) \\ E'_e = E_e \lhd\!\!\!+ (V \lhd E'_g) \\ E' = V \lhd E'_g \\ h' = h \lhd\!\!\!+ \{n_1 \mapsto s\} \end{array} \right\}$$

$$\mathsf{aut}_1 \ \frac{a.\delta((\mathsf{loc}, n_1, n_2), \sigma', g, A_{tr}, final?) \quad \Psi \quad \Omega_{loc}}{(\mathsf{aut}_\circ, n_1, E, h, s) \xrightarrow{\sigma, E_e, E'_e}_a (\mathsf{aut}_\circ, n_2, E', h', init(a.\nu(n_2)))}$$

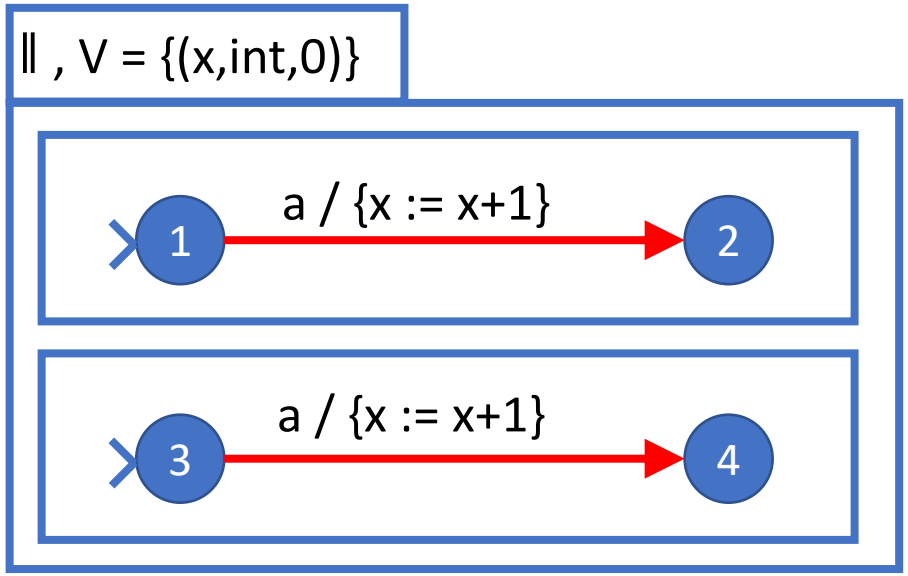
Sémantique des automates

$$\mathsf{aut}_6 \ \frac{s \xrightarrow{\sigma, E_g, E_g'''}_{a.\nu(n)} s' \qquad \Omega_{sub} \qquad \Theta}{(\mathsf{aut}_\circ, n, E, h, s) \xrightarrow{\sigma, E_e, E_e'}_{a} (\mathsf{aut}_\circ, n, E', h, s')}$$

$$\Omega_{sub} \triangleq a.\zeta(n).A_{stay}(E_g''', E_g'')$$

$$\Theta \triangleq \left(E_g = E_e \lhd E \quad a.A_{astd}(E_g'', E_g') \quad E_e' = E_e \lhd (V \lhd E_g') \quad E' = V \lhd E_g'\right)$$

Sémantique de la synchronisation



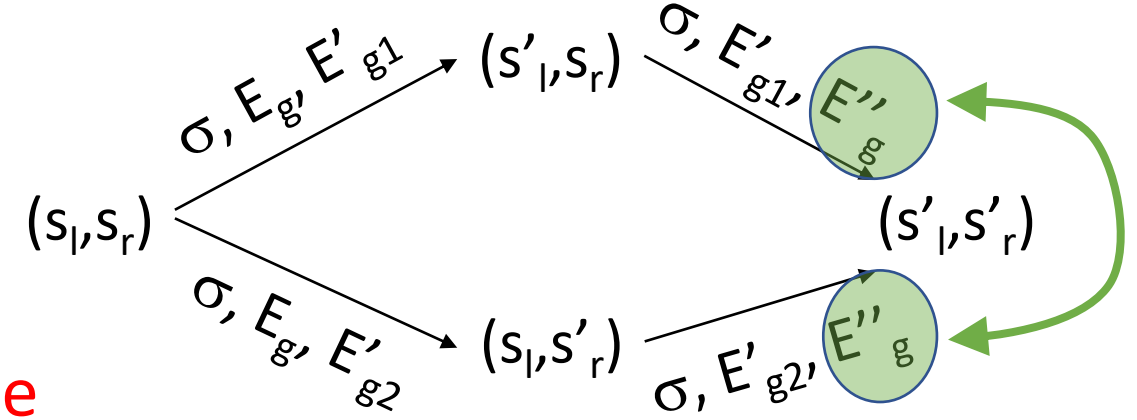
Valeur finale de x?

- x = 1?
- x = 2?

$$\Omega_{ilr} \triangleq \left(s_l \xrightarrow{\sigma, E_g, E'_{gl}}_{a.l} s'_l \qquad s_r \xrightarrow{\sigma, E_g, E'_{gr}}_{a.r} s'_r \right)$$

$$E'_{gl} = E'_{gr}$$

x = 1
exécution
Indépendante
en parallèle



x = 2
exécution
en séquence
ordre indépendant

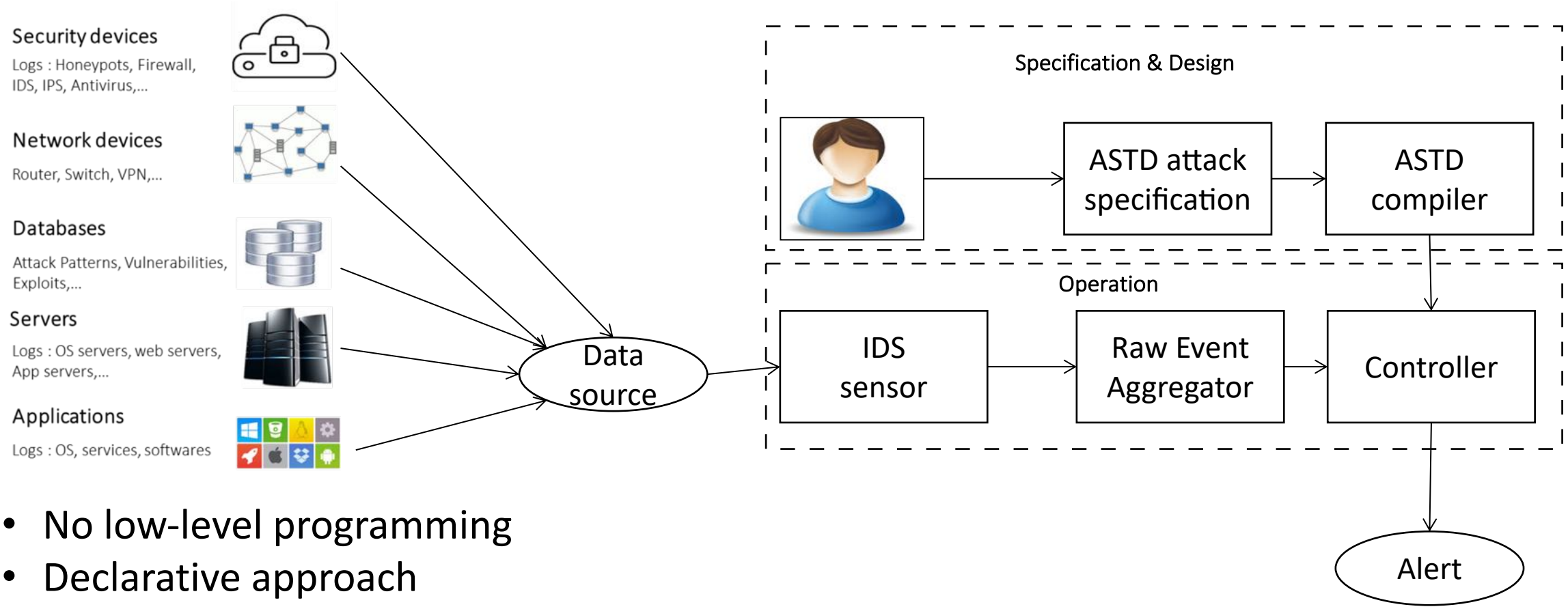
Sémantique de la synchronisation

$$\Omega_{lr} \triangleq \left(s_l \xrightarrow{\sigma, E_g, E'_{g1}}_{a.l} s'_l \qquad s_r \xrightarrow{\sigma, E'_{g1}, E''_g}_{a.r} s'_r \right)$$

$$\Omega_{rl} \triangleq \left(s_r \xrightarrow{\sigma, E_g, E'_{g2}}_{a.r} s'_r \qquad s_l \xrightarrow{\sigma, E'_{g2}, E''_g}_{a.l} s'_l \right)$$

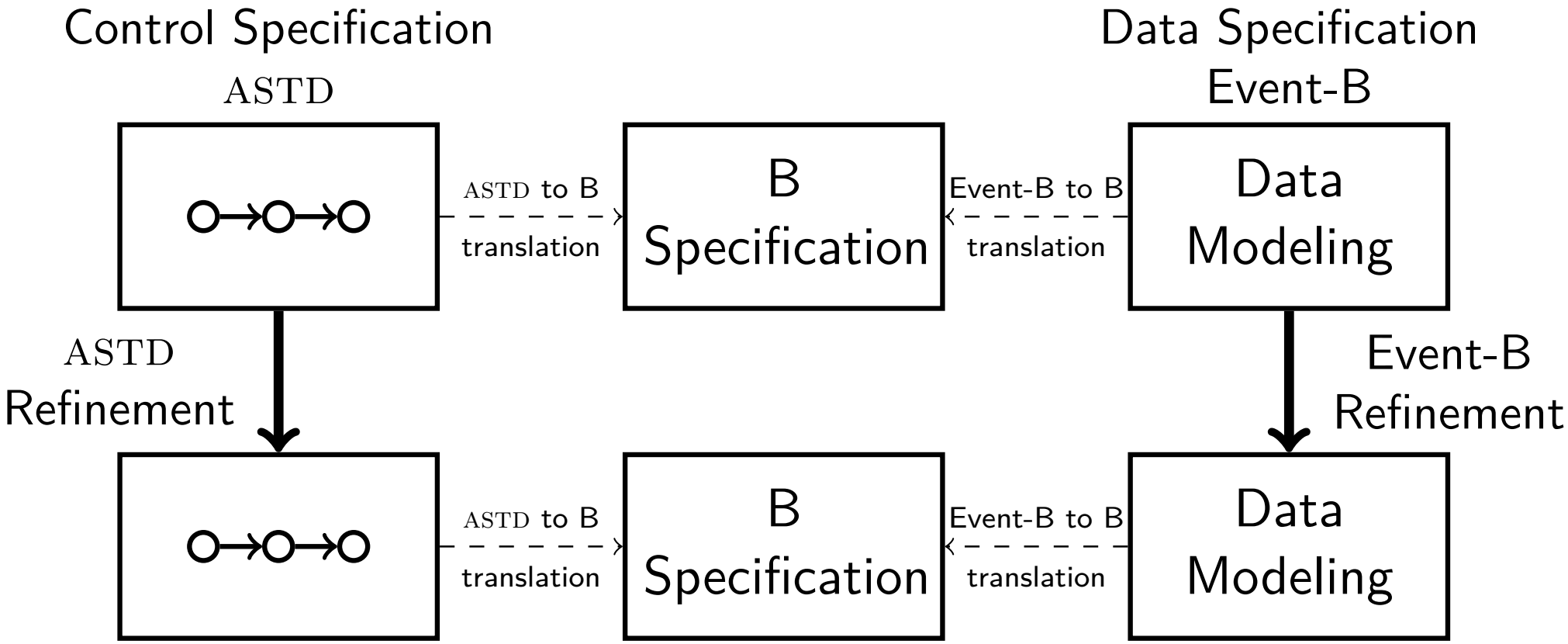
$$|[]|_3 \frac{\alpha(\sigma) \in \Delta \qquad \Omega_{lr} \qquad \Omega_{rl} \qquad \Theta}{(|[]|_\circ, E, s_l, s_r) \xrightarrow{\sigma, E_e, E'_e}_{a} (|[]|_\circ, E', s'_{l_1}, s'_{r_1})}$$

Approche pour la détection d'intrusion



- No low-level programming
- Declarative approach
- Abstraction, modularity, reuse
- Code generation

Combinaison ASTD-B (Fayolle)



Vérification ASTD paramétré - système infini (R. Chane-Yack-Fa)

- PASTD
- Pas un WSTS
  - Difficile de trouver un WQO
  - Bounded PASTD sont WSTS
- Accessibilité non décidable
- RMTS Rank Monotone Transition Systems
  - Algorithme pour le calcul d'une pred-base fini
  - PASTD sont RMTS

$$
\begin{array}{lll}
\mathbf{F} ::= \mathcal{A} & & (\text{automaton}) \\
\quad | \quad \mathbf{F} \,||_{\Delta}\, \mathbf{F} & & (\text{synchronization}) \\
\quad | \quad |_{x \in T} \mathbf{F} & & (\text{quantified choice}) \\
\quad | \quad |||_{x \in T} \mathbf{F} & & (\text{quantified interleaving})
\end{array}
$$

Outils

- iASTD – Interpréteur d’ASTD
- eASTD – éditeur d’ASTD
- ASTD2B – traducteur des ASTD en B
- ASTD 2 ProB – vérification d’un ASTD avec ProB

Remerciements

- Régine Laleau
- Richard St-Denis
- Benoit Fraikin
- Amel Mammar
- Frédéric Gervais
- Jérémy Milhau
- Thomas Fayolle
- Kevin Salabert
- Lionel Nganyewou Tidjon
- Jonathan Martineau
- Félix Vigneault
- Martin Fontaine
- Alexandre Guertin
- Michel Embe Jiague
- Alain Finkel
- Raphaël Chane-Yack-Fa
- Michael Leuschel