Théorie et pratique de la concurrence

TD1 Composition parallèle

Lundi 20·09·2021

Exercice 1 · Caisses

On considère un système de prix dans une caisse de supermarché· Le système permet de scanner les codes "barre" de produits et d'imprimer leurs prix· Le système est composé de trois composantes fonctionnant en parallèle :

1· **Le processus de lecture de code** est un processus itératif ayant deux actions : **Scanner_code** qui permet de lire un code barre, puis **Emission_code** qui émet le code lu·

2· **Le processus de conversion de prix** est un processus itératif ayant deux actions : **Recevoir_code** lui permettant de recevoir un code, puis **Emission_prix** qui émet le prix correspondant au code reçu·

3· **Le processus d'impression de prix** est un processus itératif ayant deux actions : **Recevoir_prix** lui permettant de recevoir un prix, puis **Imprimer_prix** permettant d'imprimer le prix reçu·

On fait abstraction des valeurs des codes et des prix (c'est-à-dire que l'on ne distinguera pas entre les actions selon les valeurs de leurs paramètres)·

1 Donner les systèmes de transitions modélisant chacun des processus décrits ci-dessus.

Composition parallèle

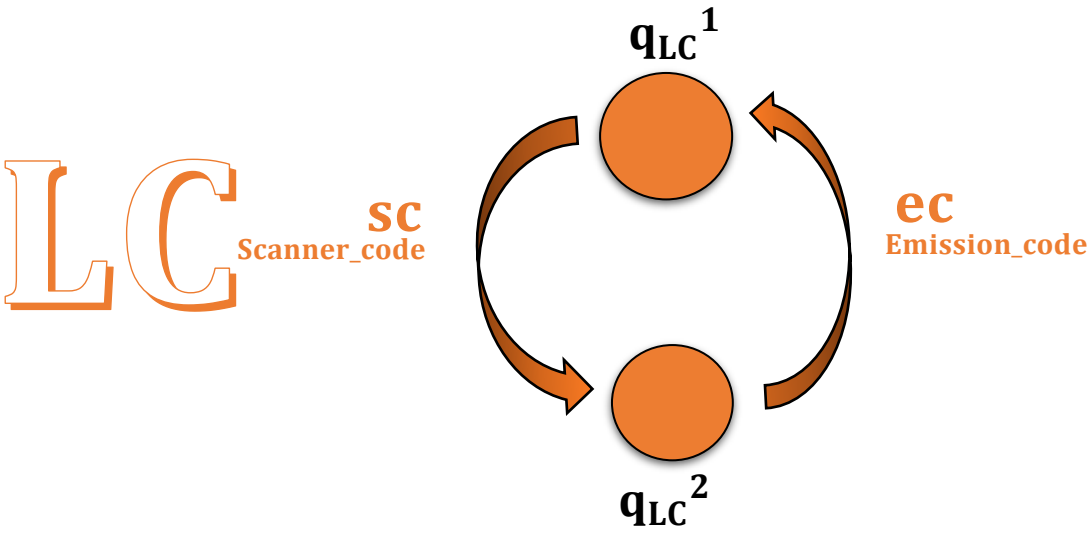
Etats — Actions — Relations de transitions

$$S = (Q, A, \delta) \qquad S' = (Q', A', \delta')$$

$$\delta \subseteq Q \times A \times Q \qquad \delta' \subseteq Q' \times A \times Q'$$

1. Le processus de Lecture de Code

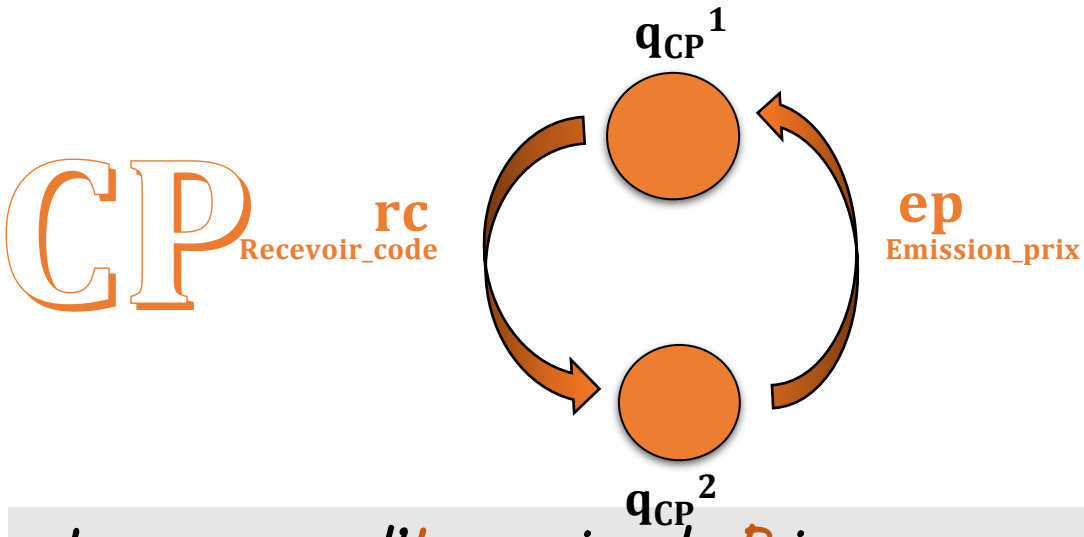
$$LC = (Q_{LC} = \{q_{LC}{}^1, q_{LC}{}^2\}, \quad A_{LC} = \{sc, ec\}, \quad S_{LC} = \{(q_{LC}{}^1, sc, q_{LC}{}^2), (q_{LC}{}^2, ec, q_{LC}{}^1)\})$$



2. Le processus de Conversion de Prix

$$CP = (Q_{CP} = \{q_{CP}{}^1, q_{CP}{}^2\}, \quad A_{CP} = \{rc, ep\}, \quad S_{CP} = \{(q_{CP}{}^1, rc, q_{CP}{}^2), (q_{CP}{}^2, ep, q_{CP}{}^1)\})$$

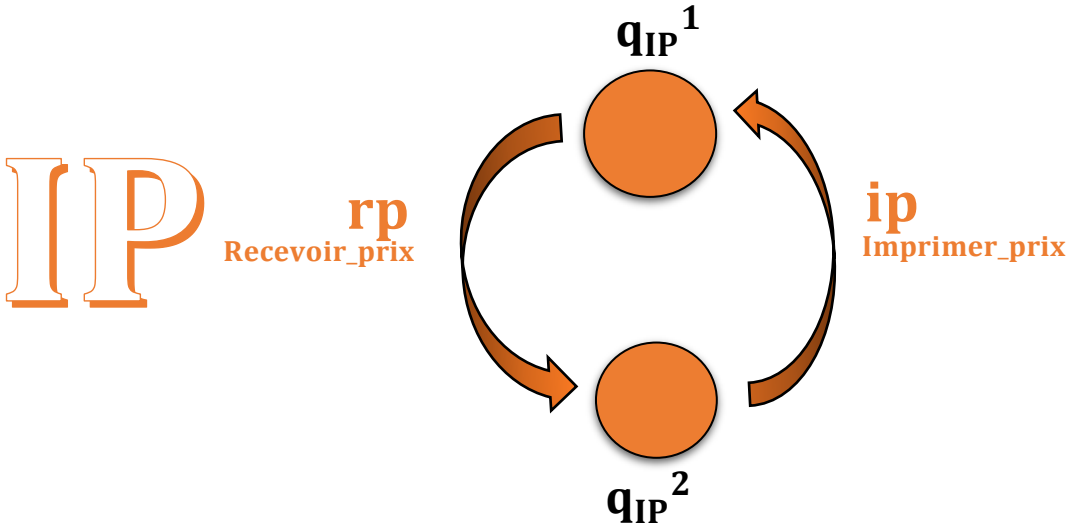
Recevoir code — Emission prix



3. Le processus d'Impression de Prix

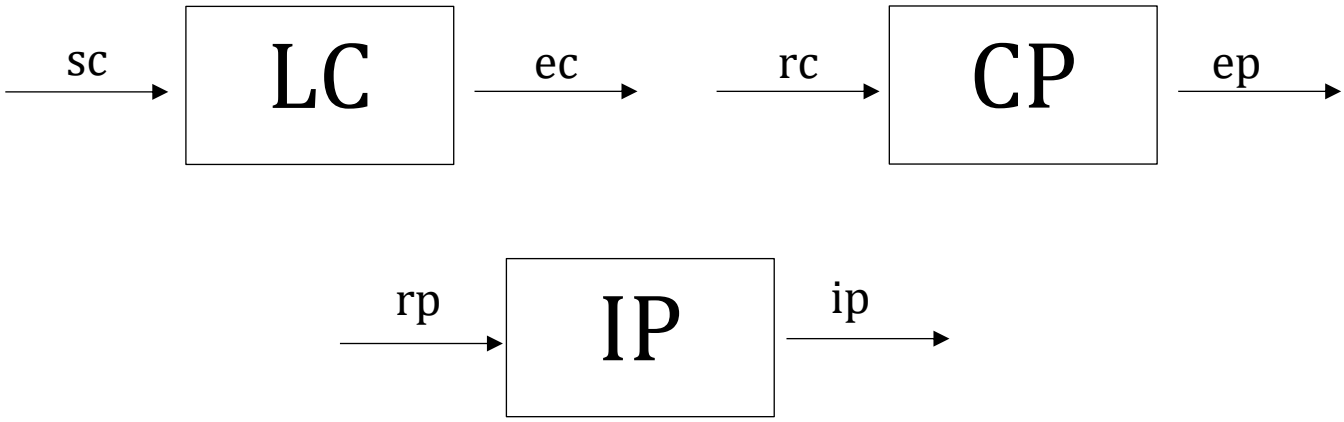
$$IP = (Q_{IP} = \{q_{IP}^1, q_{IP}^2\},\quad A_{IP} = \{rp, ip\},\quad S_{IP} = \{(q_{IP}^1, rp, q_{IP}^2), (q_{IP}^2, ip, q_{IP}^1)\})$$

Recevoir prix ← rp — ip → *Imprimer prix*

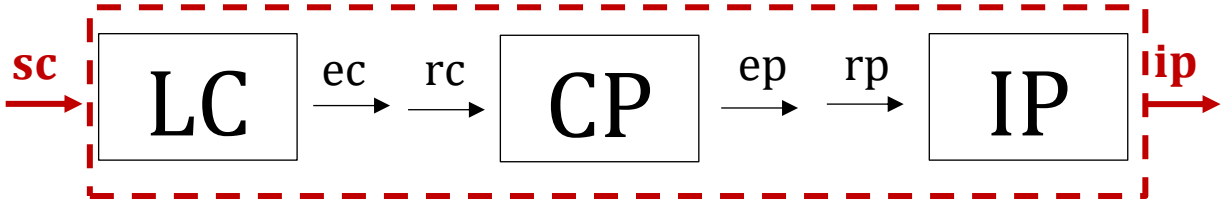


2 Donner le modèle pour le système de saisie des prix basé sur une composition parallèle des trois processus ci-dessus. (Préciser les actions de synchronisation.)

On veut un système qui scan et qui imprime. Si on met des choses en parallèle, faut voir si faut synchroniser des choses.



On va arriver à une boite comme ça :



Faut définir le système **LC || CP || IP**

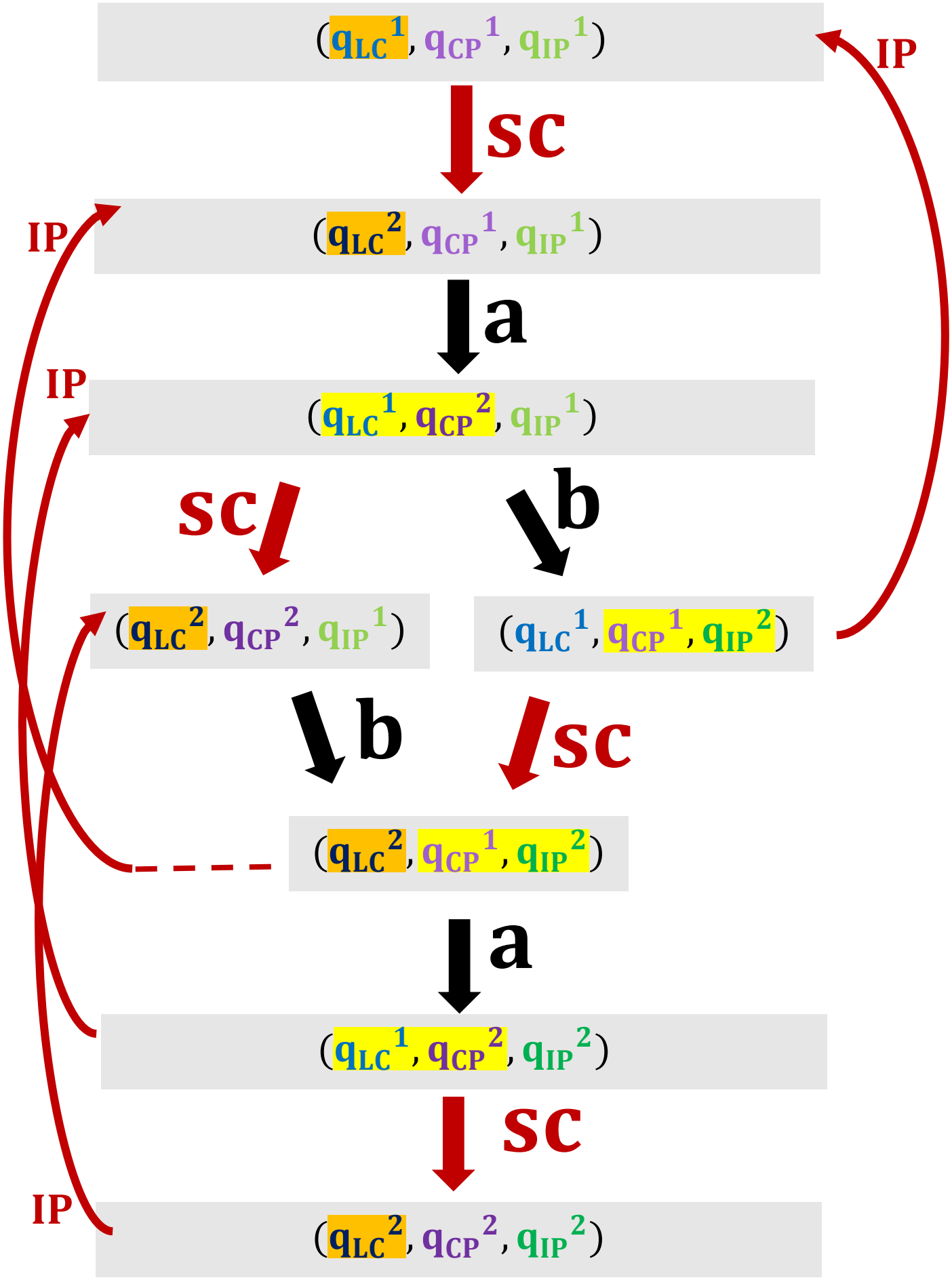
Et de manière plus précise :

On substitue (remplace) **a** *a* **ec**

$$LC[^{a}/_{ec}] \underbrace{||}_{\{a\}} CP[^{a}/_{rc}, {}^{b}/_{ep}] \underbrace{||}_{\{b\}} IP[^{b}/_{rp}]$$

Une fois qu'on a ça, faut déclarer que **a** *et* **b** *sont des actions de synchronisation.*

Faut utiliser les définitions formelles de composition parallèle vu en CM.



$2^3 = 8$ donc peut pas y avoir plus que 8 états.

3 -> car le buffer a 3 places.

Une règle

$$q_1 \xrightarrow{a}_{\delta} q_1' \quad , \qquad a \notin \text{Sync}$$

$$(q_1, q_2) \xrightarrow{a}_{\delta_{||}} (q_1', q_2)$$

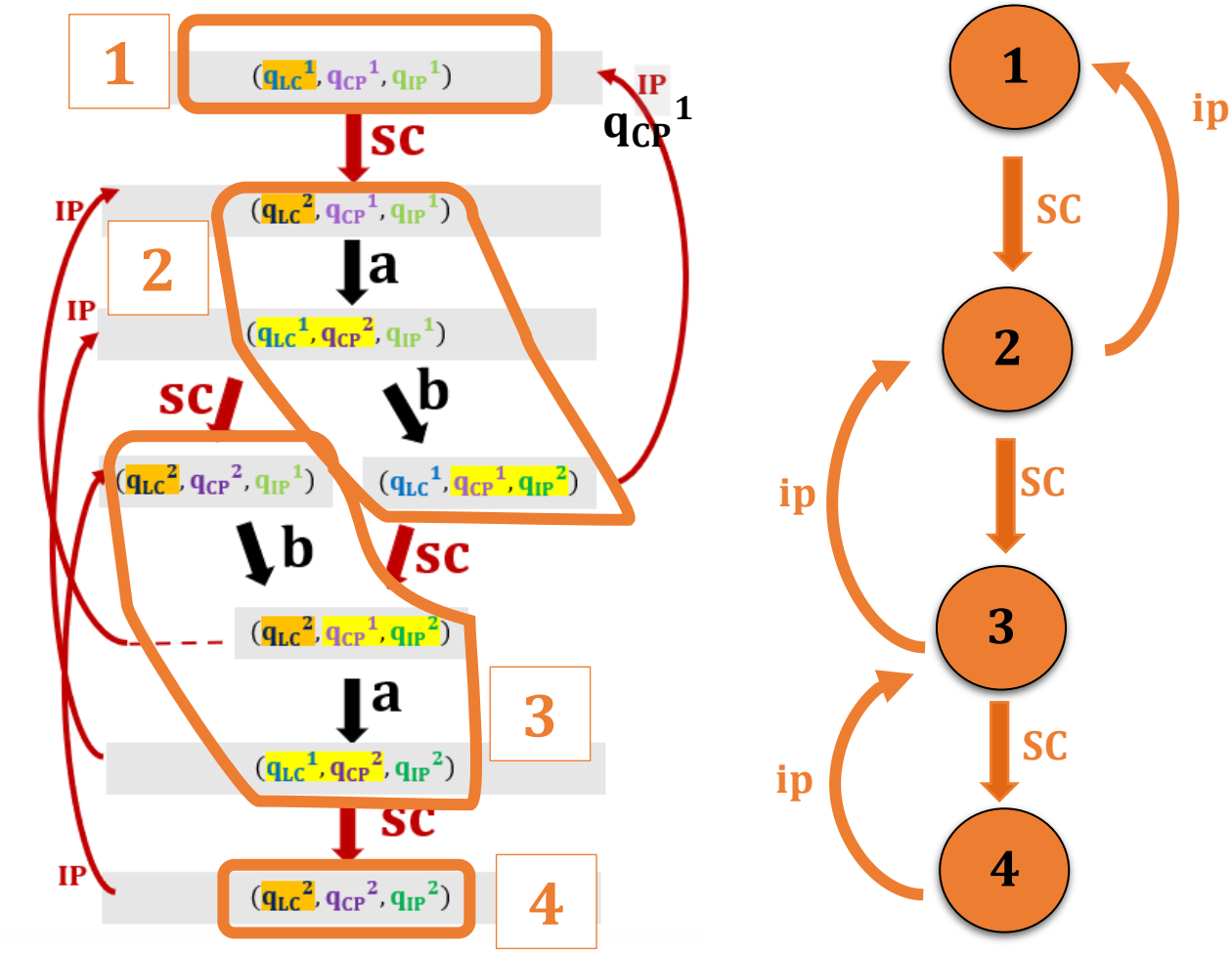
Pas de changement d'état

Une machine peut bouger avec une action asynchrone -> elle bouge **seul**.

- Les actions asynchrones, on peut toujours les faire.
- Les actions synchrones : que lorsque tous les « gens » concernés veule la faire.

---

4 Donner le système de transitions modélisant le comportement visible du système de saisie. Combien d'objets peuvent être scannés avant qu'un prix soit imprimé ?



On peut scanner 3 objets.

Exercice 2 · Exclusion mutuelle

On veut formaliser le problème de l'exclusion mutuelle· Le système est composé de trois processus· Deux processus qui utilisent une ressourcer, et un processus qui gère la ressource·

- **Les processus qui utilisent la ressource ont les transitions suivantes :**
  - **Demander_res** qui demande la ressource en question, et
  - **Rendre_res** qui rend la ressource après son utilisation·

- **Le processus représentant la gestion de la ressource a quatre transitions :**
  - **Accepter_proc1** et **Accepter_proc2** qui donnent accès à la ressource, quand elle est disponible, à chaque processus selon le cas, et
  - **Recevoir_proc1** et **Recevoir_proc2** qui reprend la ressource de chaque processus (selon le cas) et la rend disponible·

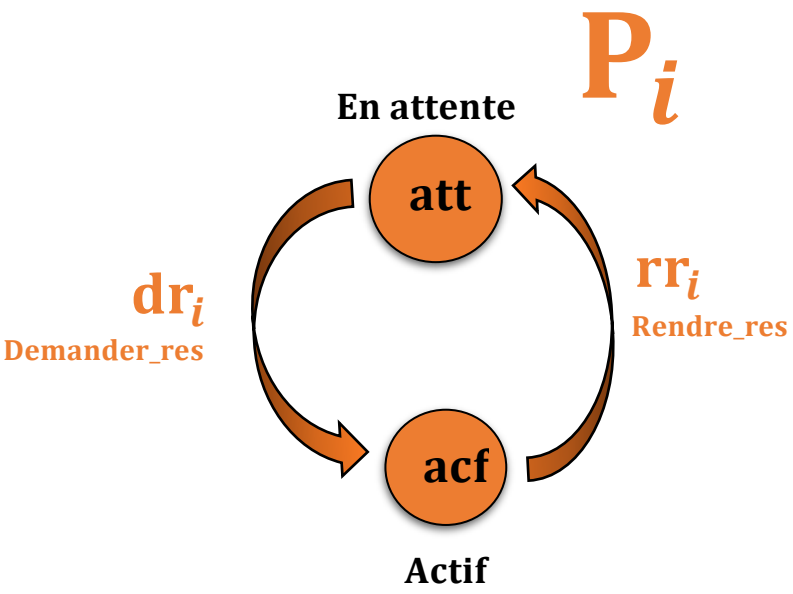
On a des processus qui doive se battre pour une certaine ressource· Il passe par une certaine entité qui doit gérés la ressource· Là on regarde ce problème de manière abstraite, on va donc essayer de modéliser le problème dans le cadre qu'on a vu·

<u>Question 4</u> – faut vérifier que les 2 processus ne peuvent pas accéder simultanément a la ressource·

Un processus c'est 2 états. Soit il est « en attente », soit il a obtenu la ressource.

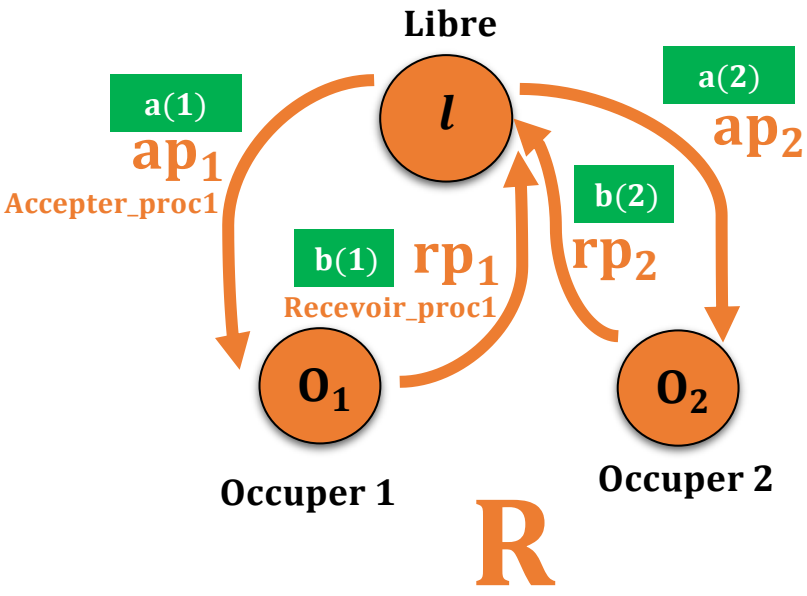
$$P_1 \,||\, R \,||\, P_2$$

Faut préciser comment on va synchroniser et une fois qu'on a fait ça on va construire le graphe.



On a aussi un processus qui représente l'état de la ressource.
Combien d'états il a ? ressource occuper / ressource libre.

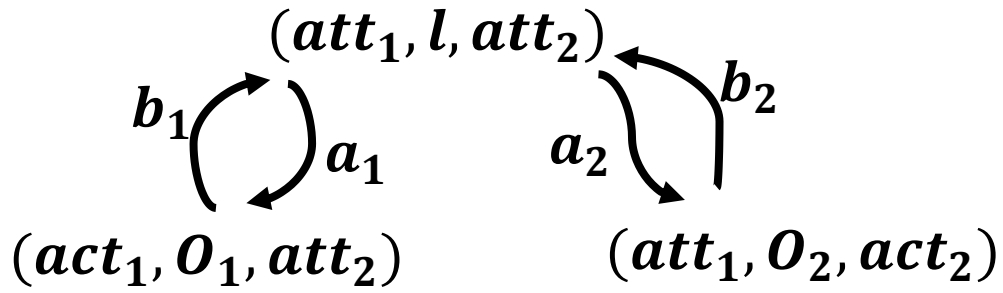
$$P_1 \left[{}^{a_1}/_{dr_1}, {}^{b_1}/_{rr_1}\right] \,||$$
$$R \left[{}^{a_i}/_{ap_i}, {}^{b_i}/_{rp_i}\right] \,||$$
$$P_2 \left[{}^{a_2}/_{dr_2}, {}^{b_2}/_{rr_2}\right]$$



Maintenant, faut développer le graphe.

C'est un système très synchrone, donc pas beaucoup de transitions (plus y'a des trucs asynchrones, plus le graphe va dans tous les sens).

Initialement processus 1 est en attente, la ressource est libre…



Est-ce que on peut dire quelque chose sur « Est-ce que ce système satisfait l'exclusion mutuelle ? » Oui

Soit la ressource est libre, soit elle est occupée par 1, soit occupée par 2.