

Chapitre II : la synchronisation de processus

processus → pinger (thread)
↓ local (processus)

//isme → vrai //isme : multiprocesseur

faux //isme : multiprogram
temps partagé

Ressources → partageable [sans risque de pb]
→ non partageable → ressources propres au processus
- → ressources utilisées en commun

Dans un processus la ressource qui est utilisée alors que les règles de Bernstein ne sont pas vérifiées est appelée critique.

Et la partie du code qui la manipule est appelé section critique.

Qu'est ce qu'on doit faire à une section critique?

On doit abandonner le //isme => revenir au séquentiel dans la section critique
solution : on doit rajouter un protocole d'entrée et de sortie

protocole entrée
SC
protocole sortie

2) les conditions à respecter pour trouver une solution au pb de sélectio critique

3 4 conditions

1/ EN : exclusion mutuelle dans 1 SC. On ne doit pas avoir plus d'1 processus \Rightarrow 1 ou 0 (EN doit être assurée)

Dwg : si EN n'est pas assurée ça ne sert à rien de vérifier le reste

2/ BN : le blocage mutuel doit être évité

si deux processus veulent entrer dans leur SC, la SC ne doit rester libre alors que les processus attendent définitivement
BN doit être evitée

3/ progression : 1 processus en dehors de sa SC ne doit pas empêcher l'autre d'y entrer dans sa SC
 \Rightarrow progression doit être assurée

4/ AB : attente Bannee : tout processus qui désire rentrer doit le faire au bout d'un temps

3) les solutions proposées basées sur les variables d'état (on suppose 2 processus P_i, P_j)

1er solution : Elle utilise 2 variables $D_i = \text{faux}$
 $D_j = \text{faux}$

P_i
désire

$D_i = \text{vrai};$
b) ($\wedge P_j = \text{vrai}$) faire rien;

$SC_i;$
 $D_i = \text{faux};$

P_j

\equiv

$D_j = \text{vrai};$
 $\wedge (D_i = \text{vrai}) \text{ faire rien};$

$SC_j;$

$D_j = \text{vrai};$

1) condition d'entrée de P_i : $D_j = \text{faux}$

2) Invariant

P_i en SC $\Rightarrow D_i = \text{vrai}$

3) Analyse de la solution

a) EN : P_j en SC et P_i désire rentrer risque t-il de le joindre ?

P_j en SC \Rightarrow (invariant) $\Rightarrow D_j = \text{vrai}$

a) P_i désire rentrer $D_i = \text{vrai}$ et test ($D_j = \text{vrai}$)

P_i reste en attente $\wedge P_j$ en SC

$\Rightarrow P_i$ ne peut rejoindre P_j en SC

b) SC libres

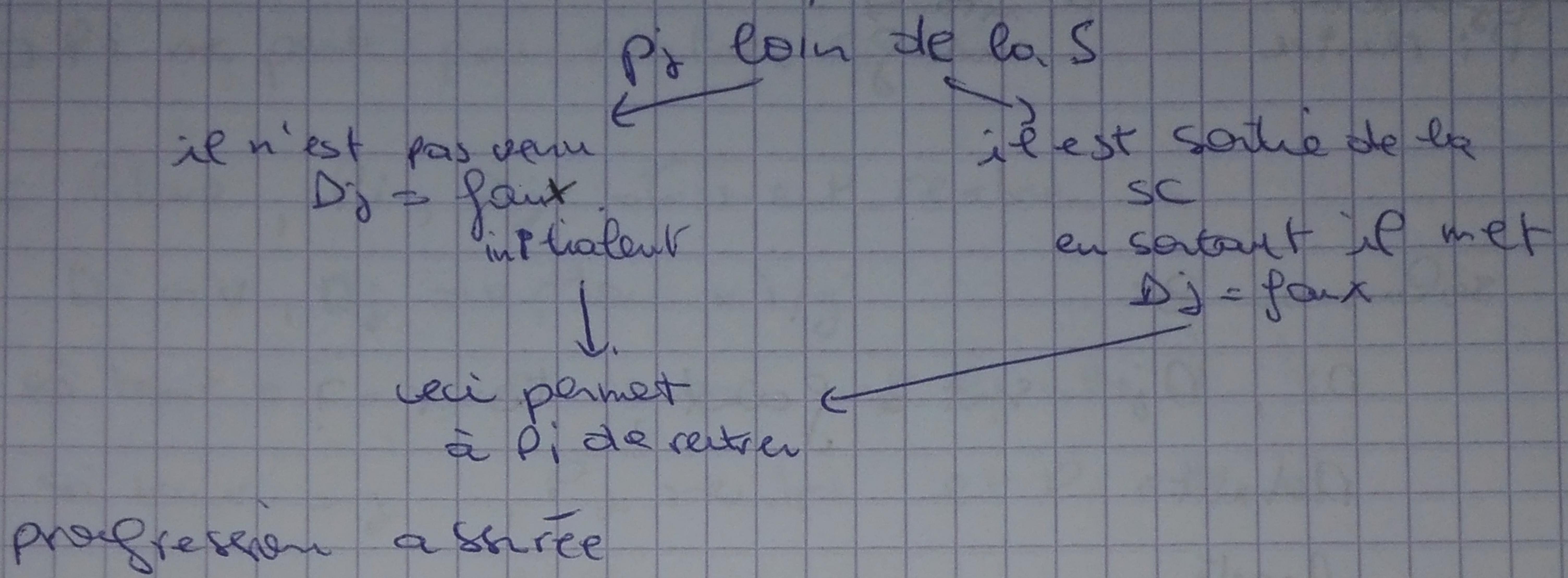
P_i, P_j désirent rentrer, risquent-ils de rentrer en même temps $D_i = \text{vrai}, D_j = \text{vrai}$

P_i, P_j vont attendre sous la loc B \Rightarrow

pouvoir réentrer \Rightarrow en SC (a) + (b) \Rightarrow EN assurée

ils déclenchent immédiatement
d'après 1.b, le BN n'est pas élué

3) progression: P_j loin de la SC . P_i désire la SC
- lorsque t'il de boucler infiniment?



Exo2 série 2

tour initialisé aléatoirement

Début

répéter

$\text{tg tour} = j$ fourie rien; fait

$\langle \text{SC} \rangle$

 tour = j

 jusqu'à faux

fin

condition d'entrée pour P_i : tour = i

Invariant: tour = i

a) EN

 alors P_j en SC \Rightarrow tour = $j \Rightarrow P_i$ ne peut rentrer

b) SC vide: P_j, P_i désirent rentrer \Rightarrow

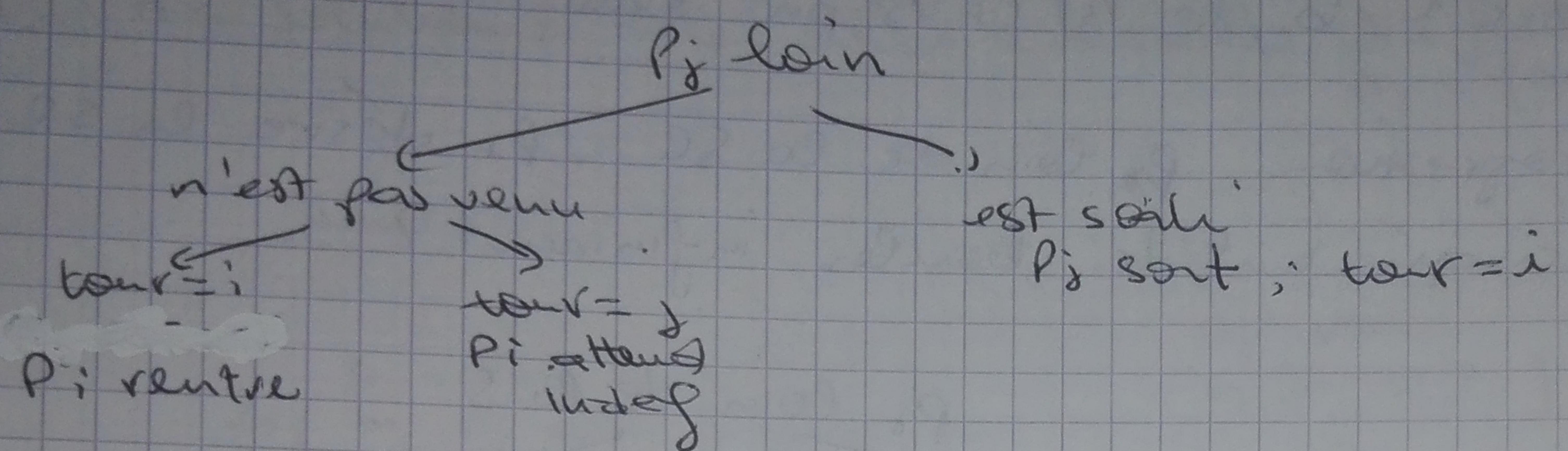
 si tour = i P_i rentre et P_j attend

 si tour = j P_j rentre et P_i attend

\Rightarrow EN assurée

2/ BN entre d'après (1c)

3/ progression



Solution 3:

D_i, D_j sont à faux; tour = ?

Début

répéter

D_i = vrai;

-tour = j;

tq D_j = vrai et tour = j faire rien

SC;

D_i = faux;

jusqu'à faux

condition d'entrée pour P_i: D_j = faux ou tour = i

Invariant, P_i en SC : D_i = vrai

D_j = faux, tour = i

D_i = vrai \wedge P_i en SC
P_i en SC \Leftrightarrow D_i = vrai
D_i = vrai \wedge P_i en SC

P_i: D_i = vrai, tour = j, test(D_j) = \top rentrer

D_j = V, tour = i attendre

1/ EN

a/ P_i est en SC, P_i désire rentrer. Risque il de le faire

P_j en SC $\Rightarrow D_j = \text{vrai}$

P_i désire rentrer $\Rightarrow D_i = \text{vrai}$, $\text{tour} = j$

$D_i = \text{vrai}$, $D_j = \text{vrai}$, $\text{tour} = j$

(La condition d'entrée n'est pas vérifiée pour P_i)

$\Rightarrow P_i$ ne peut pas rentrer en SC

b/ SC libre, P_i , P_j désirent rentrer \Rightarrow

$D_i = v$, $D_j = v$, $\text{tour} = i/j$

si $\text{tour} = i$, P_i rentre et P_j attend

si $\text{tour} = j$, P_j rentre et P_i attend

(a) + (b) exclusion \cap ressource

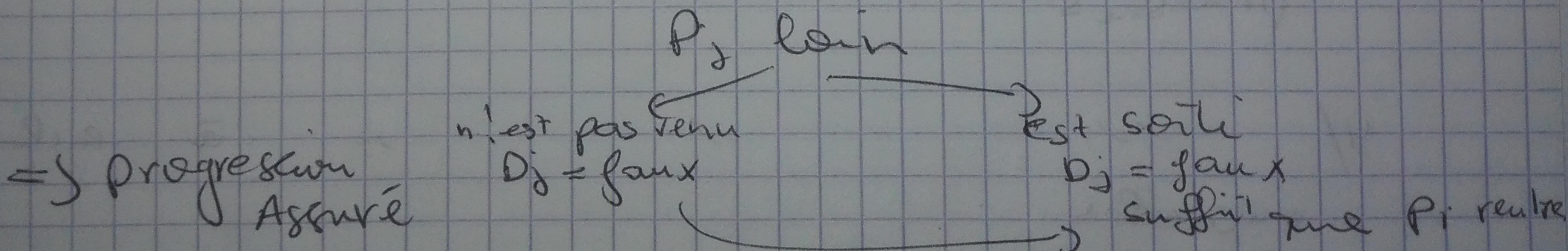
2/ BN évite

P_i , P_j désirent rentrer risquent d'attendre indef
d'après a b le BN est évité

\Rightarrow A d'entrée eux rentre

3/ progression

P_j loin de la SC, P_i désire rentrer risque t'il
d'attendre indef



4. A.B

P_j en SC

P_i en attente, risque-t-il d'attendre indefinitely si P_j continue à demander SC

P_j en SC $\Rightarrow D_j = \text{vrai}$

P_j en attente $\Rightarrow D_j = \text{vrai}$ et tour = j

$\Rightarrow P_j$ sort de la SC $\Rightarrow D_j = \text{faux}$ et P_j redemande la SC

$D_j = \text{vrai}$ et tour = i

$\Rightarrow D_i = \text{vrai}$, $D_j = \text{vrai}$, tour = i

$\Rightarrow P_i$ rentre et P_j attende $\Rightarrow A.B$ assurée.

Analyse

les solutions logicielles présentent les inconvénients suivants:

1/ l'attente est activée

2/ ils n'assurent pas toute les conditions

3/ ils sont pas généralisable facilement à N processus

4/ -- -- -- à exclusion mutuelles

ex: K imprimante

5/ ils sont valables pour des problèmes de compétition & non valables pour des problèmes de synchronisation