Université d'Evry Val d'Essonne, Master Informatique et Systèmes

Génie Logiciel 2
TD 1 : Conception de systèmes

Tarek Melliti & Pascal Poizat

2011-2012

Exercice - 1 *Produit de LTSs*

Considérez les deux LTSs S_1 et S_2 représentés dans la figure 1.

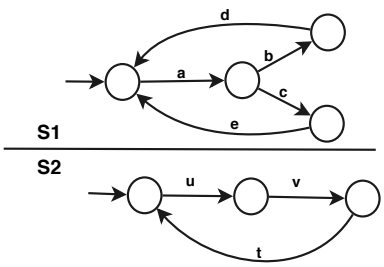

FIGURE 1 – LTSs de deux systèmes

1- Écrivez leurs définitions en termes de quintuplets vus en cours.

On veut construire un système S composé de S_1 et S_2 en parallèle.

2- Calculez le produit cartésien.
3- Calculez le produit cartésien avec ϵ.
4- Calculez le produit synchronisé en considérant le vecteur de synchronisation suivant :

S	S_1	S_2
act1	a	u
act2	b	v
act3	c	v
act4	d	t
act5	e	ϵ

5- Que constatez vous ? Modifiez le vecteur de synchronisation pour éviter ce cas.

Exercice - 2 *Correction : Produit de LTSs*

1- Voici la représentation des deux systèmes sous forme de n-uplet
$S1 = \langle \{a, b, c, d, e\}, \{s_0, s_1, s_2, s_3\}, \{(s_0, a, s_1), (s_1, b, s_2), (s_1, c, s_3), (s_2, d, s_0), (s_3, e, s_0)\}, s_0, \emptyset \rangle$
$S2 = \langle \{u, v, t\}, \{s_0, s_1, s_2\}, \{(s_0, u, s_1), (s_1, v, s_2), (s_2, t, s_0)\}, s_0, \emptyset \rangle$

2- La figure 2

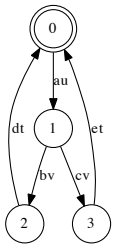

FIGURE 2 – $S_1 \times S_2$

3- La figure 3

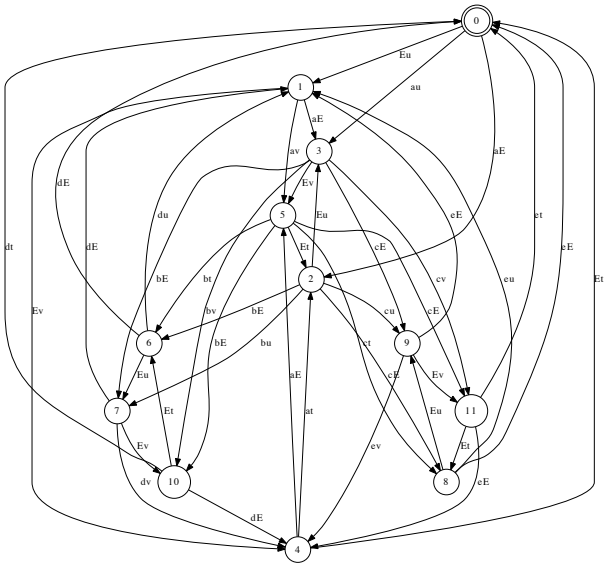

FIGURE 3 – $S_1 \times^{\varepsilon} S_2$

4- La figure 4

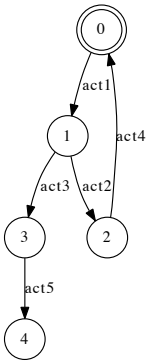

FIGURE 4 – $S_1|V|S_2$

5- On remarque que le système bloque (état 4). Pour corriger, il faut remplacer le vecteur de l'action $act5 = (e, \varepsilon)$ par le couple $act5 = (e, t)$. Cela donné le système de la figure 5.

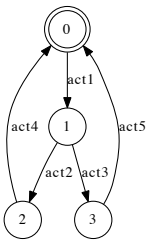

FIGURE 5 – $S_1|V|S_2$ non bloquant après modification de V

Attention on pourrait penser que si on fait avancer le système $S2$ par t en ajoutant $act6 = (\varepsilon, t)$ cela réglerait le problème puisque çela ramène le système $S2$ vers son état initial et donc tout le système vers un état où $act1$ peut se faire. En faisant cela, un non déterminisme est introduit quand à la synchronisation de l'action $act4$ et alors le système peut décider de faire t seulement ($act6$) au lieu de synchroniser avec d ($act4$). Le système résultat est donné dans la figure 6, où l'on constate la persistence (mais le déplacement) du blocage.

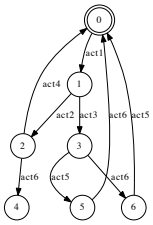

FIGURE 6 – $S_1|V|S_2$ bloquant après une mauvaise modification de V

Par contre si en plus de l'action $act6$ on rajoute $act7 = (d, \varepsilon)$ alors le système fonctionnera sans blocage.

Exercice - 3 *Batteries (Synchronisation et contrôlleurs)*

Soit une machine alimentée par trois batteries. Entre chaque batterie et la machine se trouve un interrupteur. Le but du système est de jouer sur ces trois interrupteurs à intervalles réguliers pour commuter ou non les batteries et éviter qu'une même batterie ne débite trop longtemps mais tout en évitant les surcharges (ce qui se produit si plusieurs batteries débitent en même temps).

1- Faites un dessin du système.

Chaque interrupteur peut se retrouver dans deux états différents : fermé (le courant passe) ou ouvert (le courant ne passe pas). Ces deux états peuvent être représentés par une proposition (au sens logique propositionnelle) F_i et sa négation $\neg F_i$ avec i étant l'identité de l'interrupteur (ici i=1..3).

2- Exprimez les propriétés suivantes : (P_1) pas de surcharge, (P_2) système éteint et (P_3) système allumé.

On considère le système global comme la composition parallèle (n horloges / n processeurs) des trois interrupteurs.

3- Construisez son LTS.
4- L'absence de surcharge est-elle vérifiée ?
5- En supposant que l'état initial de l'interrupteur 1 est fermé et les deux autre ouverts, donnez une matrice de synchronisation permettant d'avoir un système sans surcharge (donnez la matrice puis le LTS résultat).
6- Que constatez vous une fois le système éteint ?

Le constat de la question précédente montre qu'il est parfois impossible d'obtenir un comportement voulu pour un système uniquement en synchronisant les actions de ses composantes. Dans un tel cas, une solution consiste à ajouter un sous-système "contrôleur" additionnel.

7- Donnez le LTS d'un contrôleur pour le système ainsi que la nouvelle matrice de synchronisation.

Exercice - 4 *Correction : Synchronisation*

1- Fait en cours

2-

- pas de surcharge : $\neg(F_1 \wedge F_2) \wedge \neg(F_1 \wedge F_3) \wedge \neg(F_2 \wedge F_3)$
- allumé : $F_1 \vee F_2 \vee F_3$
- éteint : $\neg F_1 \wedge \neg F_2 \wedge \neg F_3$

3- Le comportement de chaque batterie est simple, il comporte deux états et deux transitions ; un état éteint et un état allumé. De l'état éteint on peut faire l'action on_i pour l'allumer et de l'état allumé ont peut faire off_i pour l'éteindre. Les trois systèmes sont présentés dans la figure 7.

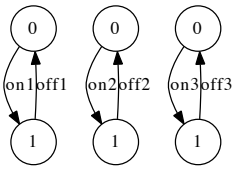

FIGURE 7 – Les trois batteries respectivement B_1, B_2, B_3

La figure 8 représente le produit libre des trois sous-systèmes

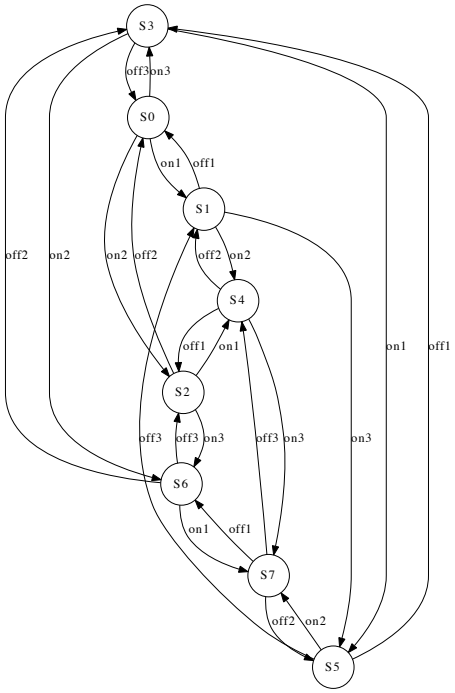

FIGURE 8 – $B_1|||B_2|||B_3$

4- Pour s'assurer qu'une seule batterie est utilisée il suffit de mettre en place une matrice de synchronisation qui empêche que deux batteries soient allumées en même temps, voir le tableau 1. La figure 9 représente le produit des trois batteries synchronisées sur cette matrice.

S	B_1	B_2	B_3
ϵ	ϵ_1	ϵ_2	ϵ_3
$switch_{1-2}$	off_1	on_2	ϵ_3
$switch_{1-3}$	off_1	ϵ_2	on_3
$switch_{2-1}$	on_1	off_2	ϵ_3
$switch_{2-3}$	ϵ_1	off_2	on_3
$switch_{3-1}$	on_1	ϵ_2	off_3
$switch_{3-2}$	ϵ_1	on_2	off_3
off_1	off_1	ϵ_2	ϵ_3
off_2	ϵ_1	off_2	ϵ_3
off_3	ϵ_1	ϵ_2	off_3

TABLE 1 – La matrice (V) de synchronisation qui empêche deux batteries d'être allumées à la fois

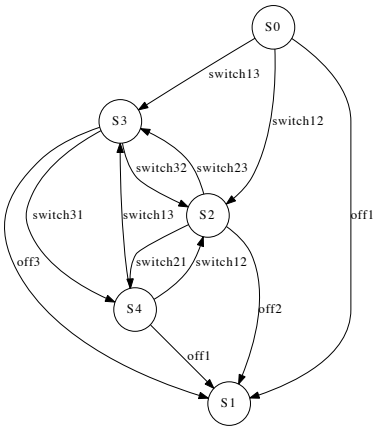

FIGURE 9 – Le produit synchrone des trois batteries sur M : $((B_1|V|B_2)|V|B_3)$

Il faut noter que cette matrice ne fonctionne que si un des sous-systèmes est à l'état allumé au départ (dans le cas de figure 9 on a choisi pour B_1 d'être allumé). Cette contrainte est due tout simplement au fait qu'on est incapable de mettre en place une matrice pour les trois systèmes (tels qu'ils sont définis dans la question précédente figure 7) au même état de départ. Pourquoi ? Si on suppose qu'au départ les trois batteries sont dans leurs états éteint respectifs, alors pour allumer tout le système tout en respectant la contrainte d'une seule batterie allumée à la fois, l'action on_i pour un $i = 1...3$ doit être synchronisée avec les actions $\varepsilon_{j \neq i}$ ce qui donne que à chaque fois que le sous-système est capable d'allumer le système B_i il peut le faire indépendamment des états des autres. Il est trivial de voir que ceci peut mener à un système qui viole la contrainte (plus d'une batterie allumée à la fois).

Il faut également remarquer (cela a un rapport direct avec l'explication plus haut) que le système résultat une fois éteint (les trois dans l'état éteint) on ne peut plus le rallumer exactement pour la même raison que l'explication plus haut.

5- Il existe deux solutions à notre problème :

La première consiste à modifier nos sous-systèmes pour dire que à chaque fois qu'une batterie est éteinte alors on peut l'éteindre et de même pour l'état allumé. Dans ce cas là les actions $suitch_{ij}$ désigneront en même temps les actions d'allumage de la batterie j après un état éteint du système global et un vrai basculement de i vers j.

La deuxième consiste à lever l'ambiguïté de l'action on_i utilisée pour allumer le système en allumant la batterie i et également pour le switch d'une batterie j vers i en rajoutant un autre sous-système (qu'on appelle un **contrôleur**). Le comportement de ce contrôleur (noté **C**) est simple : il contient deux états, éteint et allumé pour tout le système ainsi deux actions on et off. Les actions $switch_{ij}$ ne sont pas considérées (voir figure 10). Le tableau 2 représente la nouvelle matrice qui inclue le contrôleur C.

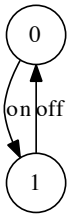

FIGURE 10 – le comportement du contrôleur

S	C	B_1	B_2	B_3
ϵ	ϵ	ϵ	ϵ	ϵ
$switch_{1-2}$	ϵ	off_1	on_2	ϵ
$switch_{1-3}$	ϵ	off_1	ϵ	on_3
$switch_{2-1}$	ϵ	on_1	off_2	ϵ
$switch_{2-3}$	ϵ	ϵ	off_2	on_3
$switch_{3-1}$	ϵ	on_1	ϵ	off_3
$switch_{3-2}$	ϵ	ϵ	on_2	off_3
off_1	off	off_1	ϵ	ϵ
off_2	off	ϵ	off_2	ϵ
off_3	off	ϵ	ϵ	off_3
on_1	on	on_1	ϵ	ϵ
on_2	on	ϵ	on_2	ϵ
on_3	on	ϵ	ϵ	on_3

TABLE 2 – La nouvelle matrice qui inclue le contrôleur

La figure 11 représente le comportement du système corrigé avec le contrôleur et la matrice ci-dessus

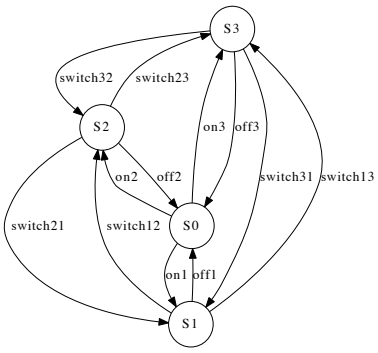

FIGURE 11 – Le comportement du système avec l'ajout d'un contrôleur et avec la matrice de synchronisation du tableau 1

Exercice - 5 *Parking (Sémaphores)*

On considère un parking qui dispose de deux entrées (Est et Ouest) et qui contient une seule place. Le système est composé de trois sous-systèmes : la porte Est, la porte Ouest et le compteur de places de parking occupées que l'on considère comme une variable à valeur dans [0..1] (on peut consulter sa valeur et la modifier en lui affectant 0 ou 1). Le comportement d'une porte est le suivant : consulter la disponibilité de la place, si libre alors laisser entrer la voiture (un observateur sera utilisé) puis incrémenter le nombre de places occupées. On ne s'occupe pas des sorties du parking.

1- Modélisez les deux portes ainsi que le compteur [0..1].
2- Que constatez vous ? Expliquez l'origine du problème et proposez une modification de vos modèles pour régler le problème.

Exercice - 6 *correction : semaphore*

1- La figure 12 représente le comportement des trois sous-systèmes, respectivement, la porte Est (PE), la variable (X) et la porte Ouest (PO). Notez bien qu'ici on ne s'intéresse qu'à une partie de ses systèmes ; dans la variable on a considéré que les actions de test à zéro et l'affectation de 1 et pour les portes seulement l'entrée et mais pas la sortie. Cette simplification à pour objectif de rendre les LTS lisibles et garde le problème visé par cet exercice intact.

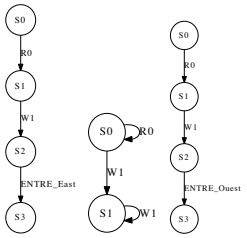

FIGURE 12 – Comportement des trois sous systèmes PE, X, PO.

Dans le système global on a les deux portes qui sont des systèmes indépendants et les deux partagent des actions avec la variable. Pour écrire l'équation du système global $PARKING$ on a deux possibilités : ou bien définir une matrice de synchronisation de tout le système (cas 1) ou procéder de manière incrémentale (cas 2) :

cas 1 La matrice de synchronization se trouve dans le tableau 3. On constate que les portes se synchronisent avec la variable mais pas entre elles. L'équation du système $PARKING = ((PO|V1|X)|V1|PE)$

cas 2 On constate que les deux portes sont indépendantes c.a.d elles ne synchronisent sur aucune de leurs actions ce qui nous permet de construire un système intermédiaire appelé $PORTES$ et qui obtenu par un produit libre des deux portes, $PORTES = PE|||PO$. Ensuite on compose le système PORTES avec la variable sur la matrice du tableau 4, $PARKING = PORTES|V2|X$.

$PARKING$	PE	X	PO
ϵ	ϵ_1	ϵ	ϵ
$R0$	$R0$	$R0$	ϵ
$W1$	$W1$	$W1$	ϵ
$ENTREE\_East$	$ENTREE\_East$	ϵ	ϵ
$R0$	ϵ	$R0$	$R0$
$W1$	ϵ	$W1$	$W1$
$ENTREE\_Ouest$	ϵ	ϵ	$ENTREE\_East$

TABLE 3 – La matrice (V1) de synchronisation du cas 1

$PARKING$	$PORTES$	X
ϵ	ϵ_1	ϵ
$R0$	$R0$	$R0$
$W1$	$W1$	$W1$
$ENTREE\_East$	$ENTREE\_East$	ϵ
$ENTREE\_Ouest$	$ENTREE\_East$	ϵ

TABLE 4 – La matrice (V2) de synchronisation du cas 1

Ces deux approches donneront le même système représenté dans la figure 13

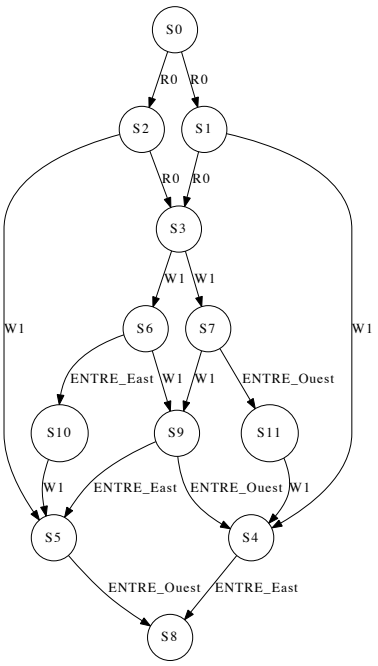

FIGURE 13 – Comportement du Système parking

On constate dans le comportement du système parking qu'il existe la possibilité que deux voitures entrent dans le parking alors qu'il y a une seule place. Cette situation est due au parallélisme ; chaque porte teste la valeur à zéro de la variable avant de laisser entrer une voiture mais entre temps (avant de mettre sa valeur à 1) l'autre porte peut faire la même chose ce qui fait que les deux tests passent et les deux portes laisseront entrer deux voitures. Ce cas est connu sous le problème de lecture sale d'une variable partagée. Le deuxième test qui passe est un test qui est fait sur variable dont la valeur est en cours de changement (vers la valeur 1) donc ce n'est pas sa vrai valeur. Pour résoudre le problème il faut rendre les deux actions (lire 0 et écrire 1) atomique en d'autres termes rendre l'accès à la variable comme une section critique. Pour résoudre le problème on peut rajouter un moniteur qui règle l'accès à cette ressource. Ce moniteur fonctionne comme un Jeton (noté ici J) ; celui qui veut utiliser la variable doit prendre le jeton avant et une fois fini il doit le remettre. Tout processus voulant utiliser la variable doit entourer les tronçons de son comportement où il utilise la variable pour la changer par les actions de prise de jeton (noté P) et sa libération (noté V). La figure 14 représente le comportement du système Jeton alors que les systèmes PEJ et POJ représentent le changement du comportement des deux portes pour accéder à la ressource variable.

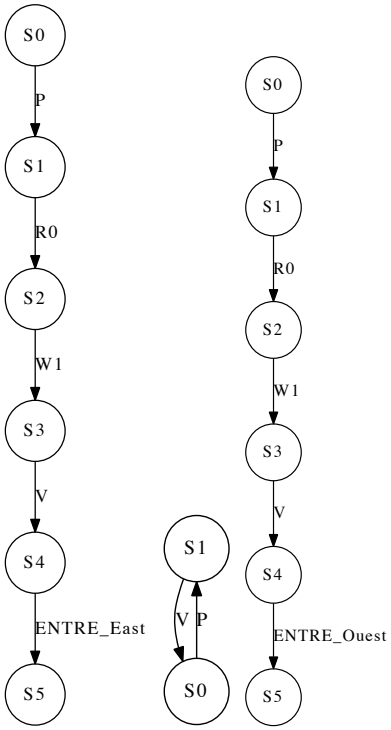

FIGURE 14 – Comportement des trois sous système PEJ, J, POJ.

Pour obtenir le système global il faut inclure le système Jeton dans la matrice et synchroniser les actions P et V qui apparaissent dans les nouvelles portes avec le système Jeton. Voici l'extension de la matrice de la question 1 (cas 1).

La figure 15 représente le système PARKING obtenu par l'ajout du jeton. Il faut noter que le système ne permet qu'une seule entrée, ce qui est souhaité.

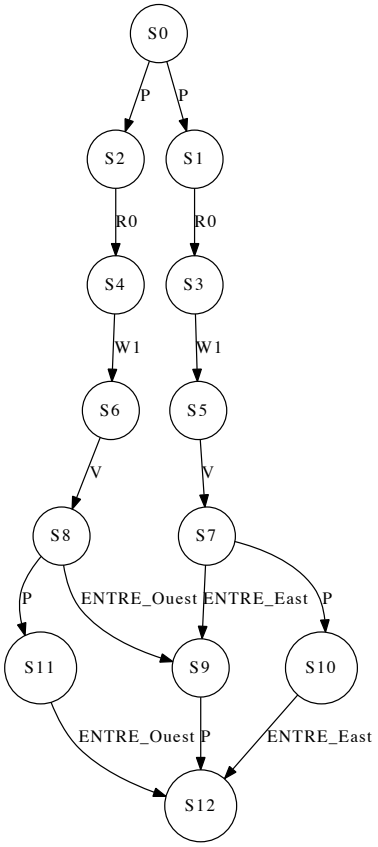

FIGURE 15 – Comportement du $PARKING$ corrigé avec le jeton.

Exercice - 7 *Dîner des Philosophes*

On souhaite modéliser le problème du dîner des philosophes pour $N = 2$. Le système est donc composé de quatre sous-systèmes, $Philo1$, $Philo2$, $Four1$ et $Four2$, qui représentent respectivement le philosophe 1, le philosophe 2, la fourchette 1 et la fourchette 2.

1- Ecrivez le comportement de chaque sous-système, sachant qu'un philosophe répète le processus suivant : prendre la fourchette à sa droite, puis celle à sa gauche, manger, poser la fourchette à sa droite, puis celle à sa gauche, penser.
2- Donnez la matrice de synchronisation de telle façon que la fourchette 1 ($Four1$) soit à droite du philosophe 1 ($Philo1$) et la fourchette 2 à sa gauche (et *vice versa* pour le philosophe 2). Construisez le système global.
3- Caractérisez sur le modèle du système global la situation de blocage vue en cours.
4- Proposez une modification de la matrice de synchronisation de la question 2 permettant d'éviter le blocage.
5- Ecrivez cette matrice pour le cas général (N quelconque).

Exercice - 8 *Correction : Dîner des Philosophes*

1- Le système est composé de deux types de systèmes, deux exemplaires de chaque type sont requis pour réaliser le diner des philosophes, le tableau 5 représente la configuration du dîner.

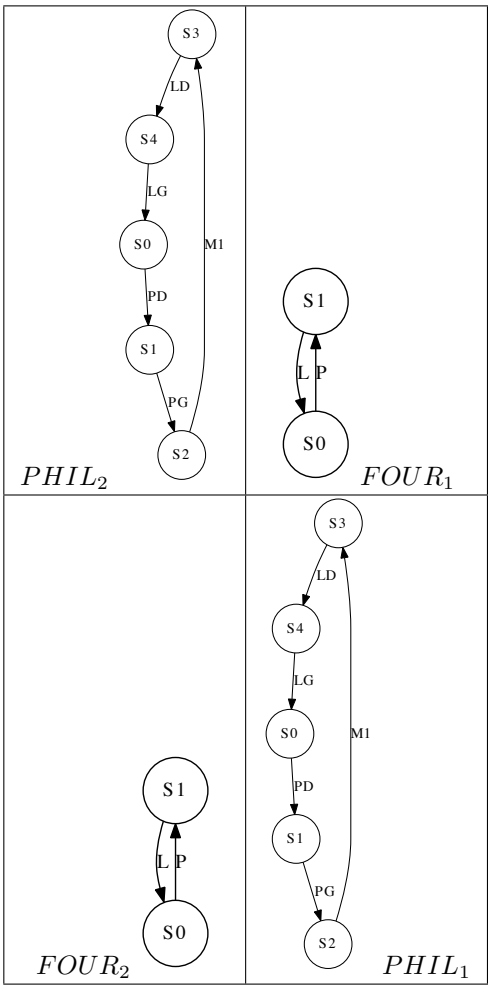

TABLE 5 – Les quatre sous-systèmes : $PHIL_1$, $PHIL_2$, $FOUR_1$, $FOUR_2$

2- La disposition des systèmes dans le tableau 5 correspond exactement à l'architecture décrite pour cette question. Pour obtenir une telle architecture il faut bien décrire les vecteurs de synchronization. Le tableau 6 représente la synchronisation qui permet d'obtenir cette architecture.

$DINER$	$PHIL_1$	$PHIL_2$	$FOUR_1$	$FOUR_2$
ϵ	ϵ	ϵ_1	ϵ	ϵ
$PFD1$	PD	ϵ	ϵ	P
$PFG1$	PG	ϵ	P	ϵ
$M1$	M	ϵ	ϵ	ϵ
$LFD1$	LD	ϵ	ϵ	L
$LFG1$	LG	ϵ	L	ϵ
$PFD2$	ϵ	PD	P	ϵ
$PFG2$	ϵ	PG	ϵ	P
$M2$	ϵ	M	ϵ	ϵ
$LFD2$	ϵ	LD	L	ϵ
$LFG2$	ϵ	LG	ϵ	L

TABLE 6 – La matrice (V) de synchronisation du diner des philosophe

3- La figure 17 représente le système $DINER = PHIL_1|V|PHIL_2|V|FOUR_1|V|Four_2$

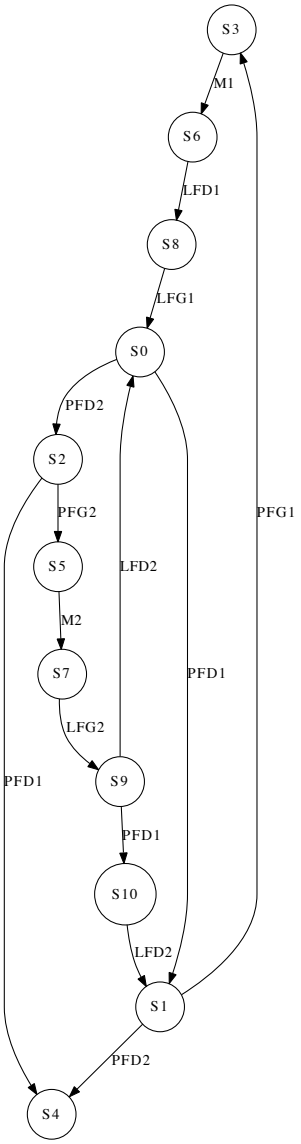

FIGURE 16 – Le dîner des philosophes

4- La situation de blocage se matérialise dans l'état $S4$. Deux traces peuvent mener au blocage ($PFD1, PFD2$ et $PFD2, PFD1$). Cette situation correspond à l'état où chaque philosophe détient dans sa main une fourchette et attend la deuxième sauf que chacun détient la deuxième de l'autre.
5- Pour régler le problème, on peut jouer sur la position des philosophes et faire en sorte que deux philosophes assis l'un à coté de l'autre se disputent la même fourchette avant de demander la deuxième. Pour cela (dans le cas de N=2) on va modifier le vecteur de synchronization de telle façon que l'un des deux philosophes commence par prendre celle qui est à sa gauche avant celle qui est à sa droite. Sans changer le comportement du philosophe ou créer un nouveau type on va seulement modifier la matrice de synchronisation comme suit (la différence avec le tableau précédent est soulignée) :

$DINER$	$PHIL_1$	$PHIL_2$	$FOUR_1$	$FOUR_2$
ϵ	ϵ	ϵ_1	ϵ	ϵ
$PFD1$	PD	ϵ	ϵ	P
$PFG1$	PG	ϵ	P	ϵ
$M1$	M	ϵ	ϵ	ϵ
$LFD1$	LD	ϵ	ϵ	L
$LFG1$	LG	ϵ	L	ϵ
$PFG2$	ϵ	PD	ϵ	P
$PFD2$	ϵ	PG	P	ϵ
$M2$	ϵ	M	ϵ	ϵ
$LFG2$	ϵ	LD	ϵ	
$LFD2$	ϵ	LG	L	ϵ

TABLE 7 – La matrice (V') qui permet d'éviter le blocage

La figure **??** représente le système dîner corrigé, $DINER = PHIL_1|V'|PHIL_2|V'|FOUR_1|V'|Four_2$

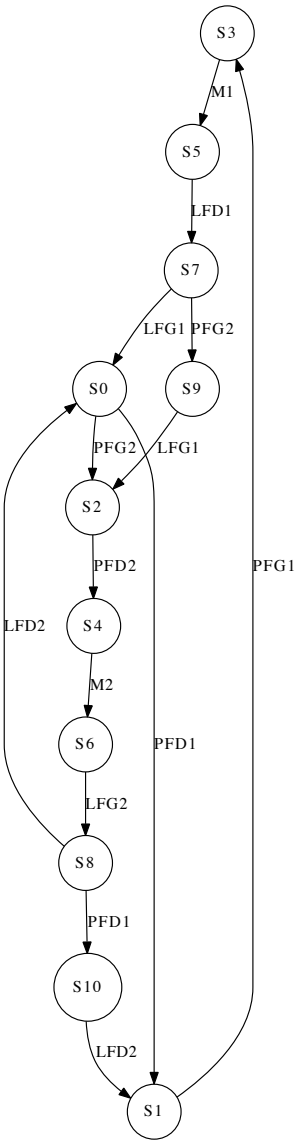

FIGURE 17 – Le dîner des philosophes sans blocage

6- Le cas général de la question précédente serait de faire en sorte qu'il y ait deux sortes de philosophes un premier type qui commence par la droite et le deuxième genre commence par la gauche ensuite d'alterner

sur la table les types de philosophe. La matrice de synchronization dans le cas général est défini comme suit :

$\forall i \leq N$ telle que $imod2 = 1$

$DINER$	...	$PHIL_i$	....	$FOUR_i$	...	$FOUR_{i+1modeN}$	...
$PFDi$	... ϵ ...	PD	... ϵ ...	ϵ	... ϵ ...	P	... ϵ ...
$PFGi$	... ϵ ...	PG	... ϵ...	P	... ϵ...	ϵ	...ϵ ...
Mi	... ϵ ...	Mi	... ϵ ...	ϵ	... ϵ ...	ϵ	... ϵ ...
$LFDi$	... ϵ ...	LD	... ϵ ...	ϵ	... ϵ ...	L	... ϵ ...
$LFGi$	... ϵ ...	LG	... ϵ...	L	... ϵ...	ϵ	...ϵ ...

$\forall j \leq N$ tel que $jmod2 = 0$

$DINER$	...	$PHIL_j$	....	$FOUR_j$	...	$FOUR_{j+1modeN}$	...
$PFGj$	... ϵ ...	PD	... ϵ ...	P	... ϵ ...	ϵ	... ϵ ...
$PFDj$	... ϵ ...	PG	... ϵ...	ϵ	... ϵ...	P	...ϵ ...
Mj	... ϵ ...	Mj	... ϵ ...	ϵ	... ϵ ...	ϵ	... ϵ ...
$LFGj$	... ϵ ...	LD	... ϵ ...	L	... ϵ ...	ϵ	... ϵ ...
$LFDj$	... ϵ ...	LG	... ϵ...	ϵ	... ϵ...	L	...ϵ ...

TABLE 8 – La matrice générale qui permet d'éviter le blocage dans le cas d'un problème de taille N