Automates et Vérification Formelle

Séance pratique

Ciprian Teodorov et Luka Le Roux

Prélude

Le résultat du produit synchrone avec bégaiement entre deux automates A et B de la Figure 1 est présenté dans la Figure 2.

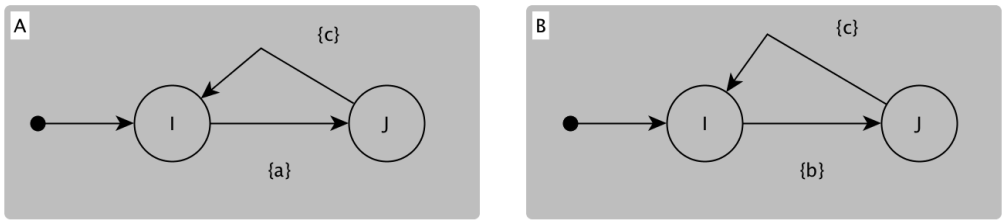

Figure 1 Deux automates indépendants.

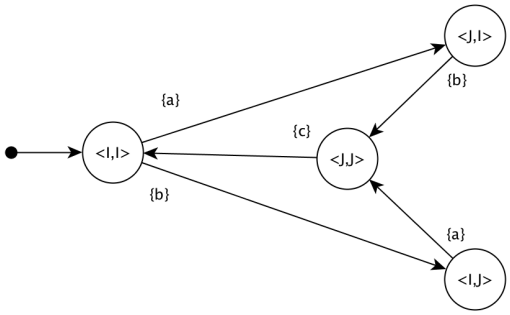

Figure 2 Le produit synchrone avec bégaiement des automates de la Figure 1

Exercice 1 : Complétez les automates Figure 3 pour que leur produit synchrone ait la même topologie que celle Figure 4.

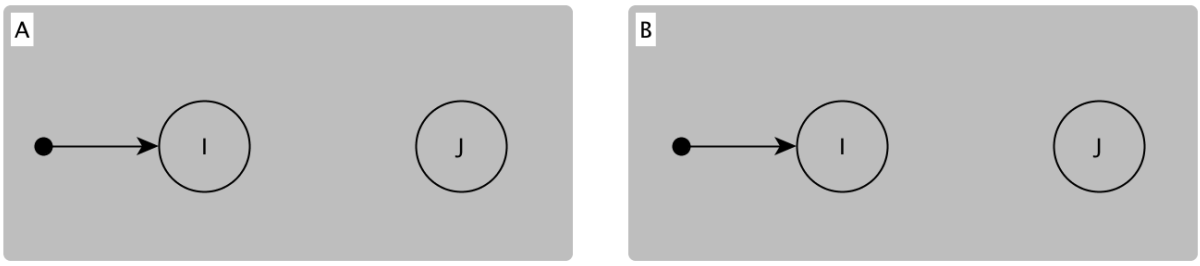

Figure 3 Les deux automates à compléter

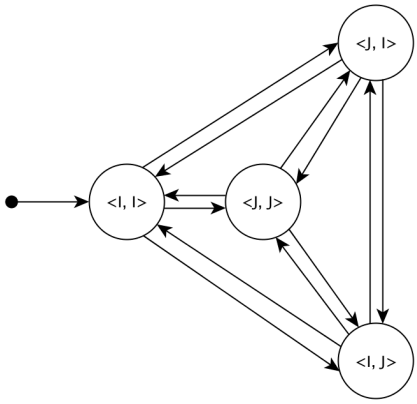

Figure 4 Le squelette du produit demandé

Le problème de Alice et Bob

Les exercices suivants sont basés sur l'histoire de deux personnages, Alice et Bob [1][1].
Alice et Bob sont voisins et partagent leur jardin, une configuration similaire à celle de la Figure 5. Chacun d'eux a un animal de compagnie, Alice un chien et Bob un chat. Les deux souhaitent laisser leur animal sortir prendre l'air dans le jardin. Malheureusement le chien d'Alice ne supporte pas le chat de Bob. De plus – comme ils habitent en Finistère – à cause de la pluie ils ne veulent pas sortir dehors avec leur animal. Du coup ils sont obligés de trouver une stratégie leur permettant de sortir leurs animaux sans risquer une rencontre même s'ils ne sont pas joignables par téléphone.

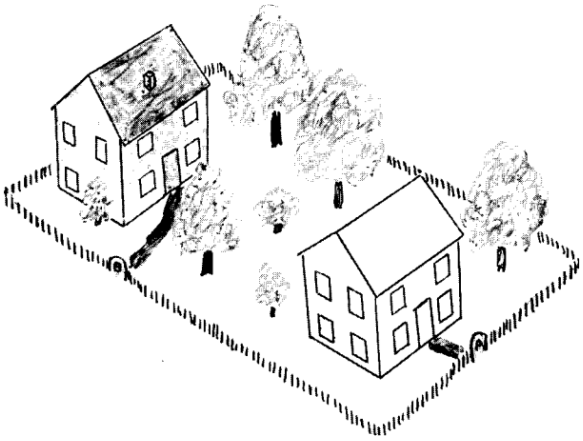

Figure 5 Le jardin de Alice et Bob

Comme Alice et Bob sont très occupés, ils n'ont pas le temps pour créer une telle stratégie par eux même et ils nous ont demandé de les aider.

Modélisation du problème et vérification de propriétés de sûreté

Conceptuellement, on pourrait modéliser le comportement souhaité par nos deux personnages avec des automates, comme illustré dans la Figure 6. Tous simplement, l'état **I** représente le cas où le chien d'Alice (ou le chat de Bob) est à l'intérieur. L'état **CS** représente le cas où les animaux sont dehors. En effectuant l'action {a1} Alice peut faire sortir son chien, réciproquement, Bob peut effectuer l'action {b1} pour faire sortir le chat. Quand ils veulent faire rentrer leur animaux Alice effectue l'action {a2} et Bob l'action {b2}. Pour simplifier le modèle on considère qu'au début les deux animaux sont à l'intérieur (l'état initial de l'automate Alice est l'état I, pareil pour Bob).

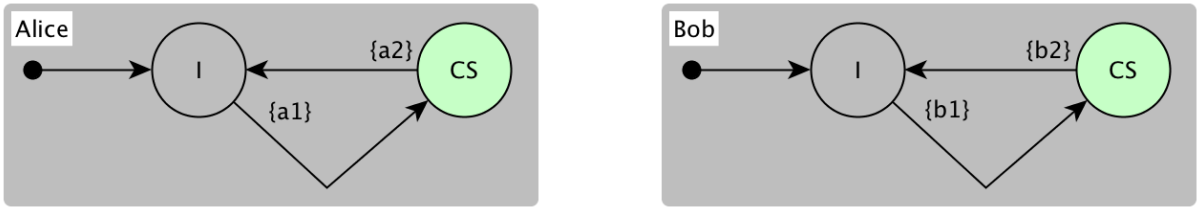

Figure 6 Deux automates modélisant le comportement de Alice et Bob.

Exercice 2 : Quel est le produit des automates Alice et Bob de la Figure 6 ? Que pourriez-vous dire du résultat ?

[1] Ces deux personnages et les exercices sont inspirés de l'article de Leslie Lamport [1].

Automates à variables partagées

Pour nous faciliter la tâche de modélisation, dans la suite on va utiliser une extension des automates : les automates à variables partagées. Ceux-ci peuvent être formalisé comme suit :

$\mathcal{A}$=(Q, ∑, δ, q0, F, V, P, E) ou

- Q – un ensemble fini d'états
- ∑ – un alphabet
- $q_0 \in Q$ – un états initial
- V – un ensemble non vide de variables partagées
- P – un ensemble de préconditions sur la valuation des variables
- E – un ensemble d'actions sur les variables
- $R \subseteq Q \times \mathcal{P}(P) \times \mathcal{P}(E) \times Q$ est une relation de transition conditionnelle entre 2 états avec une action

---

Les relations conditionnelles seront représentées comme suit :

$$S_i \xrightarrow{\{symbole\}[condition]/action} S_j$$

Ou :

- *symbole* – un élément de ∑
- *condition* – est une expression booléenne
- *action* – est une ou plusieurs affectations (« *v = exp* ») indiquant le changement de valeur de la variable *v* à la valeur indiquée par l'expression *exp*

Pour éviter la situation précédente, ou les deux animaux pouvaient se retrouver ensemble dans le jardin, on va considérer la mise en place d'un système de signalisation.
On donne un drapeau à chacun de nos personnages et on leur demande de lever le drapeau pour indiquer l'intention de laisser sortir leur animal. Ensuite, si le drapeau du voisin n'est pas levé, alors ils le laisseront sortir dans le jardin. Cette stratégie évitera les accidents à condition que les deux jouent le jeux en levant et remontant le drapeau systématiquement.

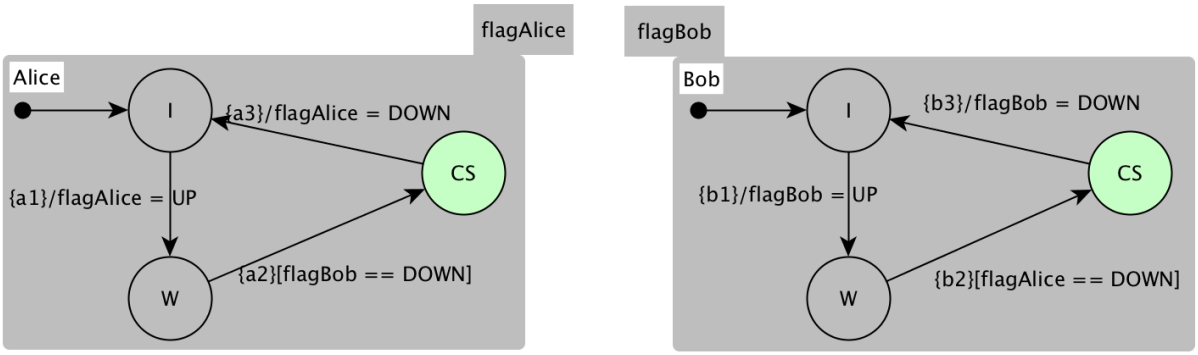

Figure 7 Les automates représentant la stratégie basée sur les drapeaux.

Les automates à variables partagées, illustrés dans la Figure 7, encodent cette stratégie.

Exercice 3 : Quel est le produit des automates Alice et Bob de la Figure 7 ? Que pourriez-vous dire du résultat ?
Indice : Chaque état du produit des automates à variables partagées contient non seulement les états de chaque automate mais aussi les valeurs des variables dans ces états.
Par exemple le quadruplet <I, false, I, false> est l'état initial du produit Alice × Bob.

Gêné par la situation précédente, Bob très gentil, propose une issue : « si jamais cette situation arrive, il descendra son drapeau et réessaiera plus tard ». Cet amendement fait changer les automates associés à nos personnages comme illustré dans la Figure 8.

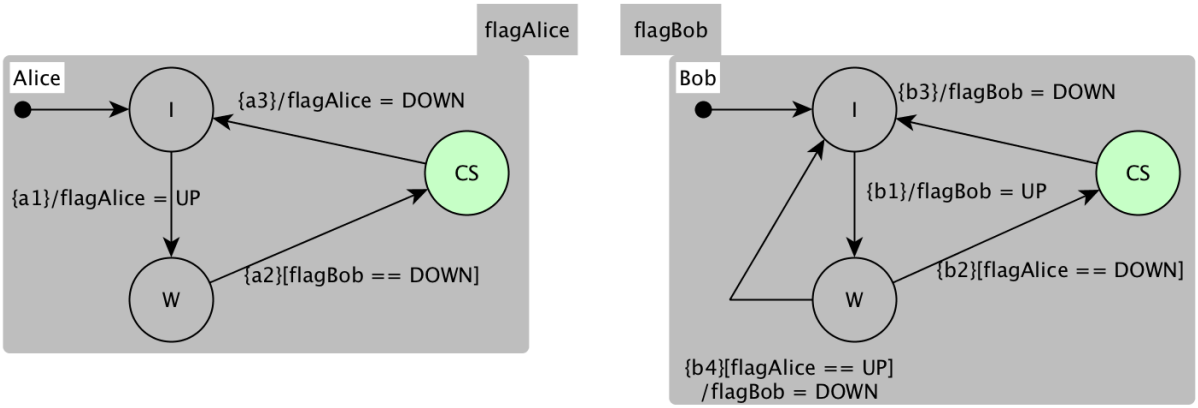

Figure 8 Bob propose de baiser son drapeau, s'il voit celui d'Alice hissé.

Exercice 4 : Modifiez le produit Alice × Bob, obtenu lors de l'exercice précédent, en considérant ce changement.

Toutes les propriétés vues jusqu'à maintenant étaient des propriétés dites « de sûreté ». Pour leur vérification on n'a besoin que de regarder si tous les états du produit respectent la propriété.
Par exemple :

- *(P1)* Tout état doit vérifier « not (Alice@CS and Bob@CS)) » – pour garantir l'exclusion mutuelle
- *(P2)* Tout état doit avoir une transition sortante – pour garantir l'absence d'un deadlock.

Exercice 5 : Assurez-vous que dans le produit obtenu pour l'exercice précédent les propriétés *(P1)* et *(P2)* sont respectées.

Ces deux propriétés sont bien respectées par notre modèle, néanmoins on peut observer un « livelock » sur le produit.

Vérification de propriétés de vivacité

Dans la suite on considèrera le cas des propriétés dites « de vivacité ». Pour la vérification de ces propriétés on utilisera le schéma suivant (Figure 9) :

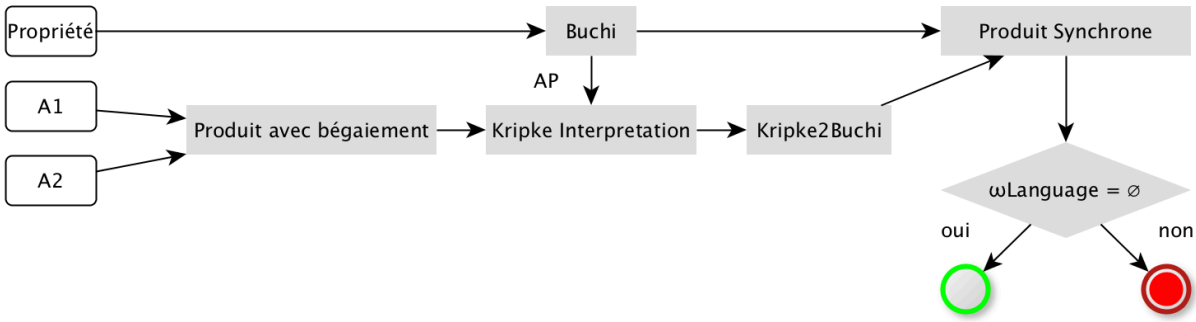

Figure 9 Schéma de vérification pour les propriétés de vivacité

Pour la vivacité, la propriété sera encodée avec un automate de Buchi B_P. La vérification de ce type de propriété nécessite un ensemble de traitements supplémentaires sur le produit des automates. Notamment, les propositions atomiques (AP) présentes dans l'automate B_p doivent être évaluées, pour obtenir une interprétation du produit sous forme de structure de Kripke. Ensuite, cette structure de Kripke peut-être transformée dans un automate de Buchi représentant le système B_s. Finalement, la vérification de la propriété sera faite en vérifiant que le langage accepté par le produit synchrone de B_p et B_s est le langage vide (aucun ω-mot n'est pas accepté).

Rappel : Un automate de Buchi accepte un mot s'il y a un chemin infini qui contient un état d'acceptation. La Figure 10 illustre un tel automate qui accepte l'ω-mot $aa(baa)^{\omega}$.

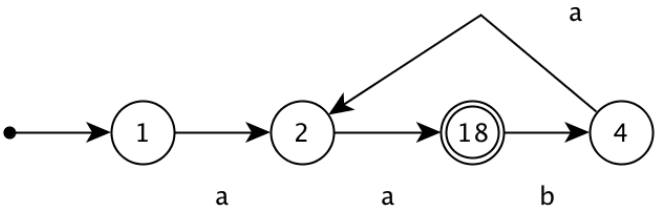

Figure 10 Automate de Buchi acceptant le mot infini $aa(baa)^{\omega}$

Exercice 6 : A partir du résultat de l'exercice 4, créez l'automate de Kripke pour la vérification de la propriété exprimée avec l'automate de Buchi illustré dans la Figure 11.

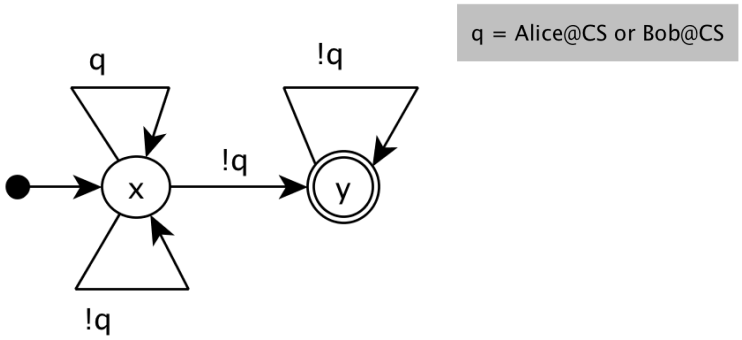

Figure 11 Automate de Buchi, représentant une propriété de vivacité.

Exercice 7 : Transformez l'automate de Kripke de l'exercice 6 dans un automate de Buchi.

Exercice 8 : Effectuez le produit synchrone entre l'automate de Buchi de la Figure 11 et l'automate de Buchi obtenu lors de l'exercice 7. Vous pouvez vous arrêter dès que vous avez trouvé que le langage accepté par le produit n'est pas vide (un ω-mot est accepté par le produit).

Exercice 9 : Représentez la trace de ce mot dans le résultat du produit Alice × Bob. Quel sens donnez-vous à l'ω-mot que vous avez trouvé lors de l'exercice 8 ? Qu'elle propriété encode-t-il l'automate de Buchi de la Figure 11 ?

Retournons à Alice et Bob. On n'a toujours pas une solution à leur donner, mais maintenant on sait que la propriété encodée par l'automate de Buchi de la Figure 11 doit être vérifiée. Une solution pourrait être de mémoriser l'intention de Bob de faire sortir son chat, et dans ce cas sans annoncer son intention il pourrait directement le faire sortir dès que le drapeau d'Alice n'est plus levé. Les automates de la Figure 12 illustre cette approche.

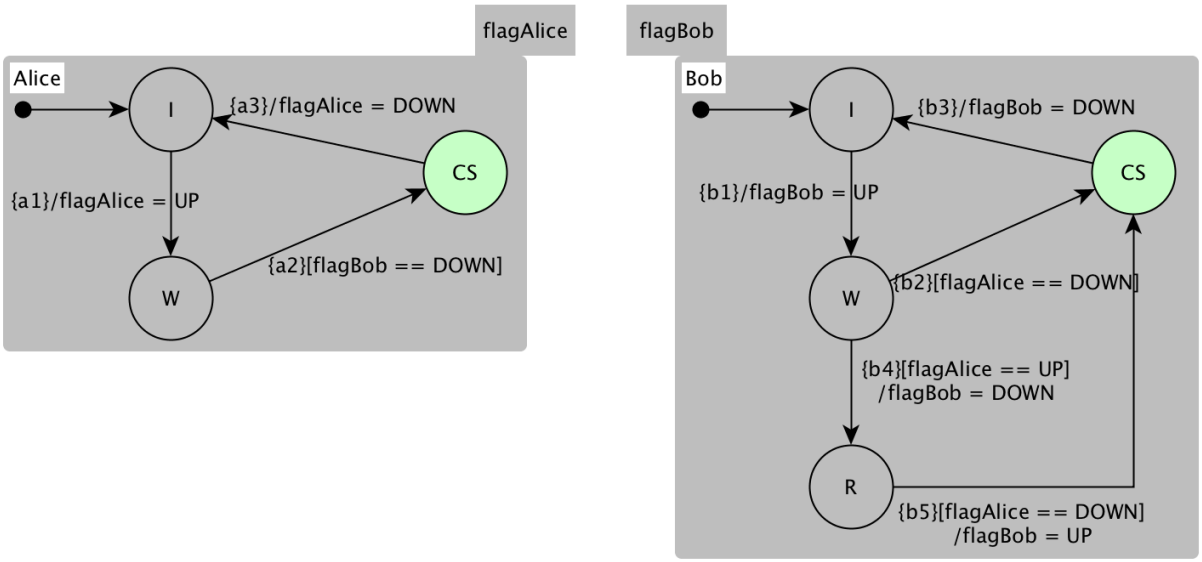

Figure 12 Bob mémorise son intention de faire sortir son chat et agit dès qu'Alice baisse son drapeau

Le produit des automates Alice et Bob de la Figure 12, est donné dans la Figure 13.

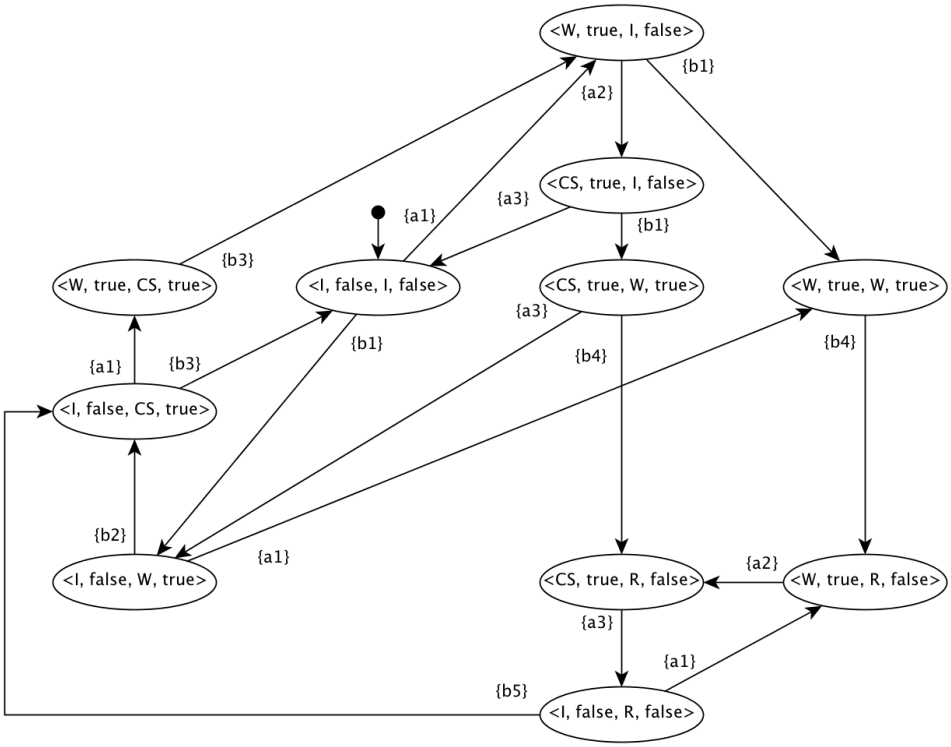

Figure 13 Le produit de Alice et Bob de la Figure 12.

Exercice 10 : Est-ce que cette solution respecte l'ensemble de propriétés que l'on a considéré jusqu'ici ? Analysez la Figure 13 afin d'identifier des cas qui pourront poser des problèmes à nouveau.

Exercice 11 : Vérifiez si le produit de la Figure 13 vérifie la propriété d'équité exprimé par l'automate Buchi donné dans la Figure 14.

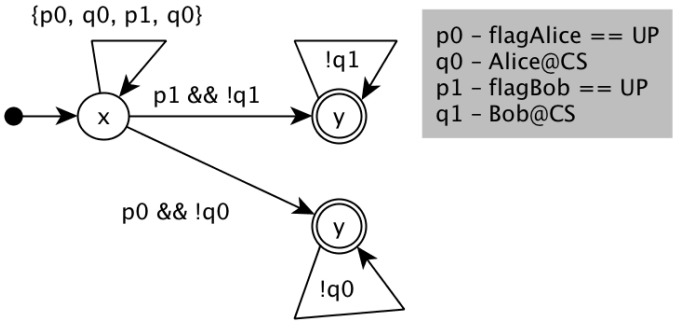

Figure 14 Automate de Buchi correspondant à une propriété d'équité.

L'algorithme d'exclusion mutuelle de Petterson

La solution à notre problème consiste à faire alterner les sorties des animaux. Cette solution communément appelé l'algorithme d'exclusion mutuelle de Petterson a été proposée par Petterson dans [2]. La Figure 15 illustre cette stratégie.

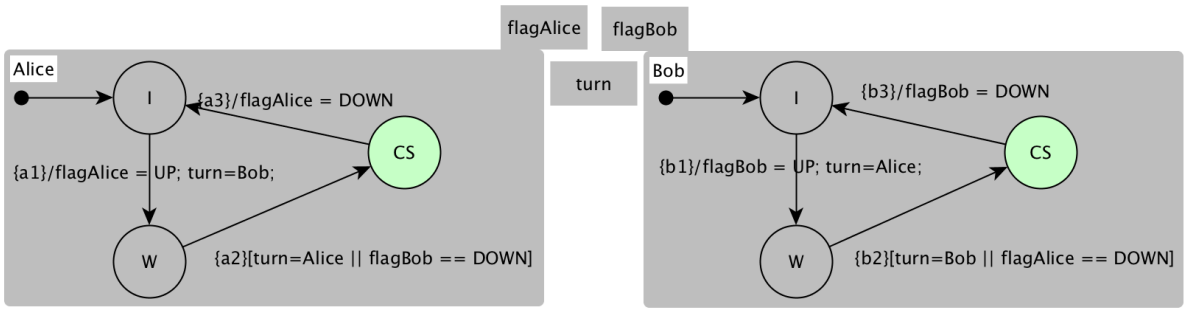

Figure 15 L'algorithme d'exclusion mutuelle de Petterson pour Alice et Bob

La Figure 16 donne le résultat du produit des automates de la Figure 15.

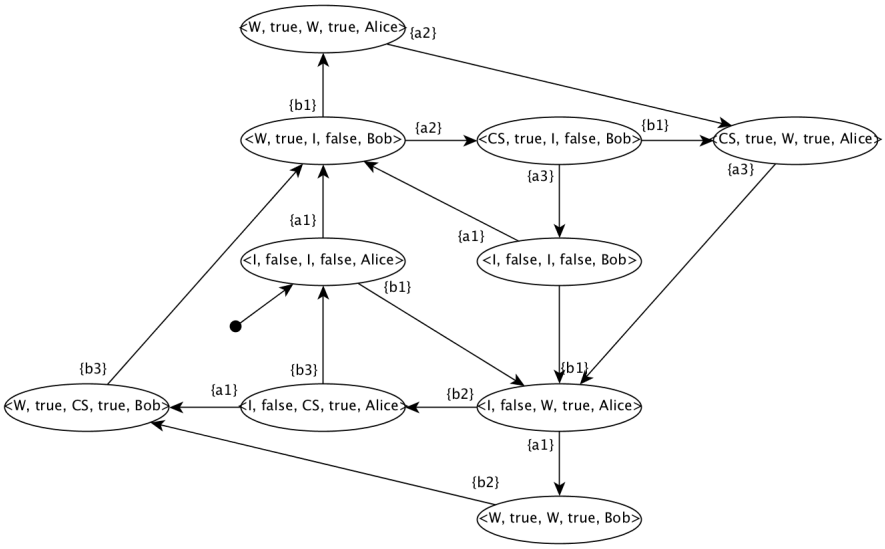

Figure 16 Produit de Alice et Bob suivant l'algorithme de Petterson.

Exercice 12 : Vérifiez que la solution respecte l'ensemble des propriétés considérées jusqu'ici. A savoir :

- *(P1)* Tout état doit vérifier « not (Alice@CS and Bob@CS)) » – pour garantir l'exclusion mutuelle
- *(P2)* Tout état doit avoir une transition sortante – pour garantir l'absence d'un deadlock.
- *(P3)* Un des deux (Alice@CS ou Bob@CS) arrivera finalement dans la section critique.
- *(P4)* Si un des deux hisse son drapeau il arrivera finalement dans la section critique.

Les propriétés de vivacité nous permettent de vérifier des cas plus subtils et notamment d'assurer que lors de l'exécution de notre système tout comportement voulu à une chance d'arriver.

Références bibliographiques :

[1] Lamport, Leslie. "Solved problems, unsolved problems and non-problems in concurrency." *ACM SIGOPS Operating Systems Review* 19.4 (1985): 34-44.
[2] Peterson, Gary L. "Myths about the mutual exclusion problem." *Information Processing Letters*12.3 (1981): 115-116.