Introduction à la modélisation et à la simulation de systèmes à évènements discrets par réseaux de Petri

Culture Sciences de l'Ingénieur

Yan MONIER – avec la relecture de Jean-Jacques Lesage

Édité le 01/07/2021

école normale supérieure paris—saclay

Cette ressource est issue du travail personnel de Yan Monier, étudiant en master 2 Industrie du Futur et Système Intelligent, cursus Conception des Systèmes Critiques au département Génie Mécanique de l'ENS Paris Saclay. Jean-Jacques Lesage est enseignant et chercheur au LURPA (Laboratoire Universitaire de Recherche en Production Automatisé), son travail porte majoritairement sur la modélisation et l'identification de systèmes à évènements discrets, sous forme d'automates ou de réseaux de Petri.

1 – Introduction sur les systèmes et l'automatisation

1.1 – La modélisation système dans le monde de l'industrie

L'un des problèmes majeurs de l'industrie a toujours été : « comment produire plus à moindre coût ». Avec l'arrivée de l'automatisation, les industriels ont pu réduire les coûts de production et produire des biens de plus en plus sophistiqués. L'automatisation a par ailleurs permis de réduire la pénibilité du travail, et nous sommes tous confrontés à elle de manière directe ou indirecte (téléphone portable, ordinateur, etc.). Cependant, toujours dans une idée de diminuer les coûts tout en améliorant la performance, l'ensemble des problèmes liés à l'utilisation de systèmes automatisés n'a pas encore été résolu de manière optimale. La recherche dans le domaine de la modélisation des systèmes a pour but de répondre à ce besoin.

1.2 – Exemple de problème : le comportement non souhaité

Lors de la conception de systèmes automatisés, on voudrait pouvoir vérifier que lorsqu'ils seront mis en marche, ils feront bien ce, et seulement ce, pourquoi ils ont été conçus. La modélisation système peut être un outil utilisé dans le but de vérifier qu'un système est bien conçu, qu'il ne va pas agir d'une manière imprévue et non souhaitée par son concepteur. L'un des comportements non voulus le plus connu est le « deadlock », c'est le fait pour un système de se mettre dans une situation où il ne peut plus évoluer, il reste donc bloqué.

Prenons par exemple le cas du blocage d'un bras robotisé. On présente *figure 1* un système permettant de cuire des pots en terre cuite. Ce système est composé de deux convoyeurs (convoyeurs 1 et 2), d'un bras robot et d'un four. Le bras robot peut prendre un pot non cuit du convoyeur 1 et le mettre dans le four. Le robot peut aussi prendre un pot cuit du four et le déposer sur le convoyeur 2. On a pour contrainte que le four ne peut contenir que deux pots et que le robot ne peut transporter qu'un pot à la fois. Un des problèmes que l'on peut être amené à rencontrer avec ce genre de système est que le robot commence l'action de prendre un pot non cuit sur le convoyeur 1 pour le mettre dans le four, alors qu'il y avait déjà deux pièces dans le four. Dans ce cas, on se retrouve dans une situation de blocage où les actions programmées possibles par le système ne permettent pas de débloquer le système. (Il n'est généralement pas prévu que le robot puisse lâcher une pièce non cuite autre part que dans le four).

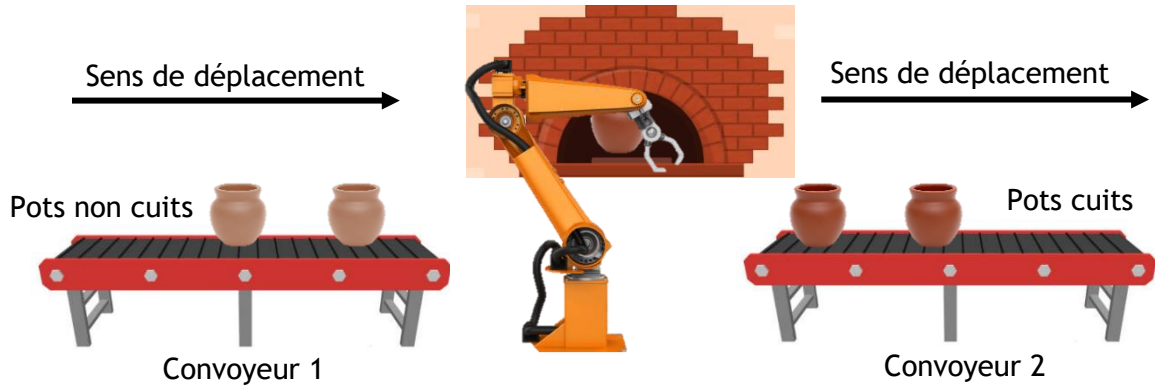


Figure 1 : Système de chauffe de pots en terre cuite

1.3 – Autre exemple de problème : la rapidité de maintenance

Les systèmes manufacturiers sont des systèmes complexes dont le vieillissement conduit souvent à des pannes. Le problème de ce genre d'incident est qu'il peut avoir de lourdes conséquences pour les industriels. Par exemple dans le cas des chaînes de production, si une machine tombe en panne sur la ligne de production, cela peut générer un blocage et stopper la production jusqu'à ce que la machine en question soit réparée. De plus si la panne n'est pas détectée, la chaine de production va continuer à produire des pièces qui sont potentiellement mauvaises et donc inutilisables par la suite (exemple : si la machine qui pose les écrans lors de la fabrication des téléphones portables ne fonctionne plus et que le problème n'est pas détecté, on risque de se retrouver avec des téléphones terminés, mais sans écran). La recherche dans le domaine des systèmes à évènements discrets permet de proposer des solutions dans la détection de telles pannes et d'identifier la source de la panne. De cette manière, une équipe de maintenance peut rapidement intervenir pour remettre au plus tôt la chaîne de production en marche. C'est ce qui est appelé en anglais : « fault detection and isolation » (détection et isolation de fautes).

2 – Les systèmes à évènements discrets

2.1 – Définitions

D'après la définition Larousse, un **système** est un « ensemble d'éléments considérés dans leur relation à l'intérieur d'un tout fonctionnant de manière unitaire ». De manière plus pratique, un système est un ensemble d'éléments interagissant entre eux selon certains principes ou règles dans le but de réaliser une tache [1]. Dans le cadre des systèmes à évènements discrets (SED), un système est déterminé par :

- Sa **frontière d'isolement**, c'est-à dire la limite entre ce qui fait partie du système et ce qui ne fait pas partie du système (l'environnement extérieur du système qui va déterminer les entrées et les sorties du SED.) ;
- Les **interactions** entre le système et son environnement ;
- Ses **fonctions**, c'est-à-dire l'ensemble des comportements des entités faisant partie du système, leur organisation et leurs interactions.

Un système à évènements discrets, par opposition à un système continu, est un système qui satisfait les deux propriétés suivantes :

- Son **espace d'état est discret** (le nombre d'état du système est dénombrable) ;
- Le passage d'un état à un autre est déclenché par un **évènement**. (C'est-à-dire que le passage du système d'un état à un autre se fait par saut et non pas de manière continue).

2.2 - Exemples

Une lampe torche comme présentée *figure 2* n'a que deux états : allumée ou éteinte. La lampe passe de l'état éteinte à allumée s'il y a eu l'évènement « allumer », et passe de l'état allumée à éteinte lorsqu'il y a eu l'évènement « éteindre ». Dans le cas de la *figure 3* est présenté un convoyeur sur lequel à tout moment, un objet peut être déplacé entre 2 positions : 0m et 2m, ceci est un système continu.

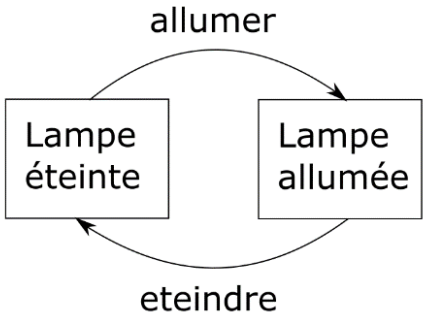


Figure 2 : Systèmes à évènement Discret : la lampe

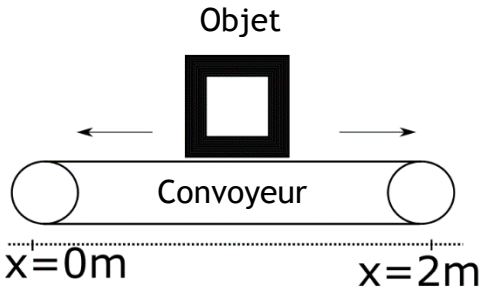


Figure 3 : Système continu : l'objet sur convoyeur

Il est cependant important de noter qu'un système ayant des comportements physiques continus peut être assimilé à un système à évènements discrets lorsque l'aspect continu du système n'est pas utile. Par exemple, dans le cas d'un convoyeur qui transporte des objets d'un point A à un point B, seules les informations indiquant qu'un objet est à l'entrée du convoyeur ou qu'un objet est à la sortie du convoyeur sont nécessaires. Si un objet entre sur le convoyeur, le convoyeur est mis en route, lorsqu'un objet sort du convoyeur, celui-ci est arrêté. Cela rentre bien dans la catégorie des systèmes à évènements discrets.

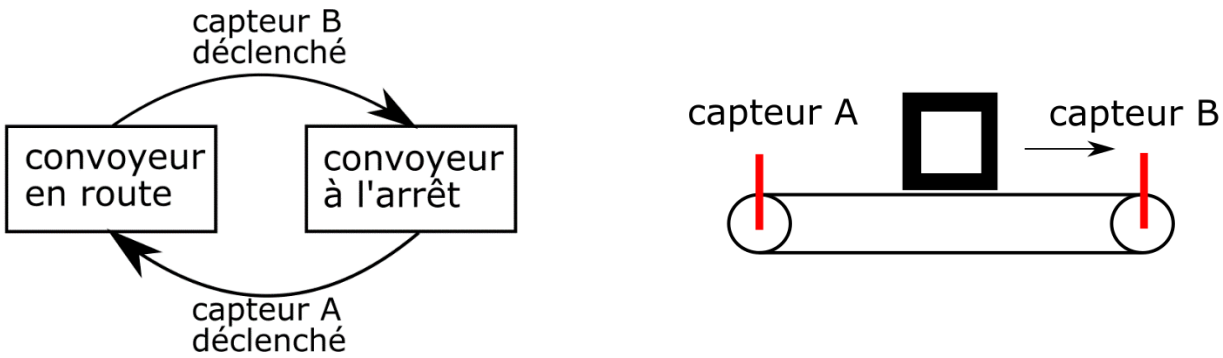


Figure 4 : Système à évènements discrets : un objet transporté par un convoyeur d'un point A à un point B

Il existe plusieurs formalismes permettant de modéliser les systèmes à évènements discrets : les principaux sont les automates à états et les réseaux de Petri. Dans cette ressource, nous ne présenterons que les réseaux de Petri tels qu'ils ont été originellement conçus.

3 – Les Réseaux de Petri

3.1 - Principe des réseaux de Petri

Un réseau de Petri est un formalisme permettant de modéliser des systèmes à évènements discrets sous la forme d'une représentation graphique du système relativement facile à comprendre pour

un humain. Il a l'avantage de pouvoir représenter des systèmes à évènements discrets complexes sous un format plus compact que les automates à états finis.

Dans le principe, un réseau de Petri peut être considéré comme un graphe orienté bipartite contenant des **places** (des cercle vides), des **transitions** (des rectangles noirs) et des **arcs** (des flèches reliant les transitions et les places entre elles). On peut considérer les places comme des récipients dans lesquelles on peut placer des jetons (pastille noire à l'intérieur des places). Les places peuvent représenter une partie du système, un lieu dans le système, un compteur, ou autre. L'ensemble des jetons dans les places représente le marquage du réseau de Petri et un marquage est associé à un état du système.

Par exemple, dans le cas de la lampe, le réseau de Petri qui décrit son comportement est le suivant :

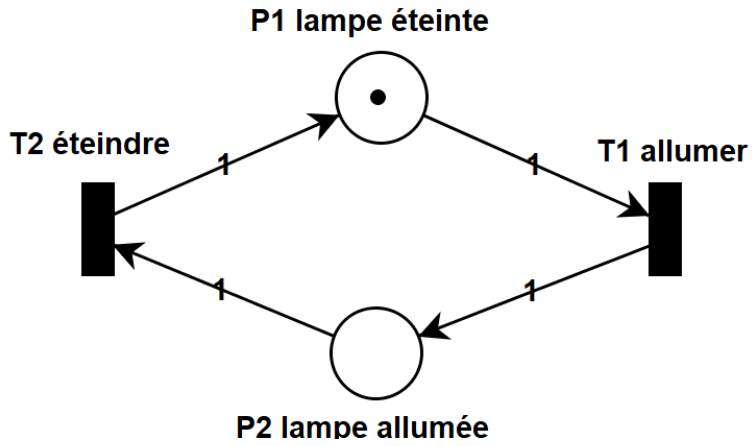


Figure 5 : Réseau de Petri d'une lampe torche

Le jeton représente ici dans quel état est la lampe, s'il est dans la place P1 la lampe est éteinte et s'il est dans la place P2, la lampe est allumée. Ici le marquage initial du système est défini par un jeton présent dans la place P1 et aucun dans la place P2. Cela correspond à l'état du système où la lampe est éteinte.

3.2 – Définition d'un réseau de Petri [3][4]

De manière plus formelle, un réseau de Petri est défini comme un 5-uplets (c'est-à-dire un ensemble de 5 éléments) : $\{P, T, A, W, M_0\}$

Les éléments de ce 5-uplets sont définis comme suit :

- P est l'ensemble des **places** ;
- T est l'ensemble des **transitions** ;
- A est l'ensemble des **arcs** allant d'une place à une transition, ou d'une transition à une place. Il est important de noter qu'un arc ne peut pas relier deux transitions ou deux places entres elles ;
- W est appelée la fonction poids (aussi appelée matrice d'incidence dans la représentation matricielle des réseaux de Petri), elle associe à chaque arcs un nombre entier. Ce nombre sera utilisé par la suite pour définir la façon dont les jetons peuvent passer d'une place à l'autre ;
- M_0 est appelé marquage initial. A un instant donné, le marquage du réseau de Petri est défini par le nombre de jetons présents dans chacune des places du réseau de Petri. Le

marquage initial est donc le nombre de jetons initialement présent dans chaque place, il représente l'état initial du système.

3.3 – Loi d'évolution d'un réseau de Petri

Les jetons à l'intérieur du réseau de Petri peuvent évoluer de place en place lorsqu'une transition est dite **franchie**. Lorsqu'une transition est franchie, un nombre spécifique de jetons des places en entrée de la transition est consommé (retiré de la place) et des jetons sont ajoutés dans les places en sortie de la transition. Le nombre de jetons ajoutés ou supprimés dépend du poids W associés aux arcs liant les places et transitions. De plus, pour qu'une transition soit franchie, elle doit être franchissable. Une transition est franchissable s'il y a suffisamment de jetons pouvant être consommés dans les places en entrée de la transition.

Exemples :

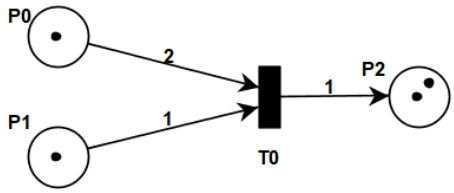


Figure 6 : Exemple 1, transition non franchissable

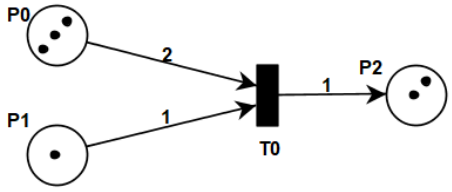


Figure 7 : Exemple 1, transition franchissable

Sur la figure 6, la transition T0 n'est pas franchissable car il n'y a pas assez de jeton dans la place P0. En effet, l'arc qui va de la place P0 à la transition T0 est de poids w=2, il faut donc au moins deux jetons dans la place P0 pour que la transition T0 soit franchissable. On peut quand même noter que la condition concernant la place P1 est respectée, l'arc sortant est de poids w=1 et il y a bien un jeton dans la place. Cependant pour que la transition soit franchissable, il faut que les conditions de toutes les places en entrée de T0 (P0 et P1) soient satisfaites. Ainsi, dans le cas de la *figure 7*, il y a assez de jetons dans la place P0 et dans la place P1, la transition T0 est donc franchissable. Lorsqu'elle est franchie, on obtient le marquage *figure 8*.

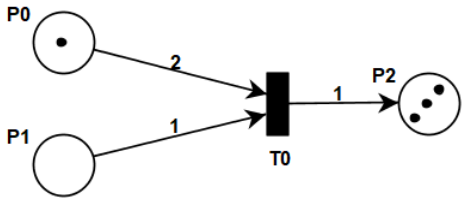


Figure 8 : Exemple 1, transition franchie à partir du marquage figure 7

La transition T0 a été franchie, comme défini par les poids sur les arcs correspondants :

- 2 jetons sont supprimés de la place P0 ;
- 1 jeton est supprimé de la place P1 ;
- 1 jeton est ajouté à la place P2.

4 – Tutoriel modélisation et simulation de réseau de Petri avec PIPE2, application au système de cuisson de pot *figure 1*

4.1 – Installation de PIPE2

PIPE 2 est un logiciel libre de droit disponible à tous. Pour installer PIPE 2 [5], veuillez le télécharger directement sur le site des développeurs : Platform Independent Petri net Editor 2 (sourceforge.net) ou directement sur leur lien Forge : Platform Independent Petri Net Editor download | SourceForge.net.

Vérifiez que vous avez bien java d'installé sur votre ordinateur, le logiciel PIPE2 a besoin de java pour pouvoir fonctionner

Apres avoir télécharger le logiciel, vous obtiendrez un fichier compressé, décompressez-le quelque part sur votre PC. Ouvrez le dossier compressé et lancez le fichier .bat : launch.bat en double cliquant dessus. Une fenêtre de terminale devrait s'ouvrir puis le logiciel devrait se lancer.

4.2 - Première approche de PIPE2

Après avoir lancé PIPE2, vous devriez obtenir cette fenêtre :

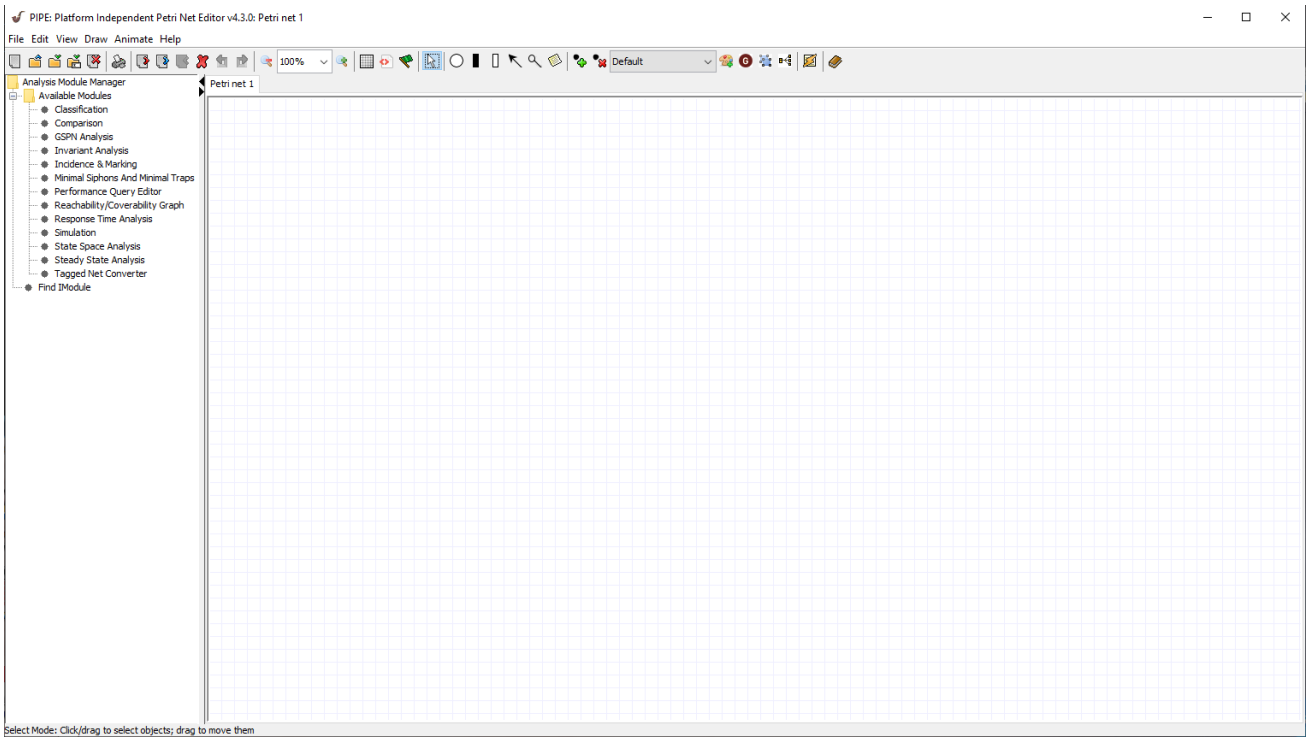
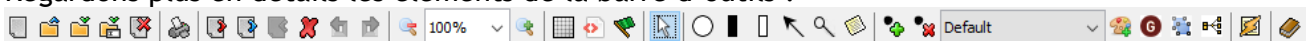


Figure 9 : Ecran d'accueil PIPE2

Regardons plus en détails les éléments de la barre d'outils :



Dans cette barre d'outils, les éléments importants sont :

- Outils création de place : en cliquant sur la grille avec cet outil, on crée une place ;
- Outils création de transition : en cliquant sur la grille avec cet outil, on crée une transition ;
- Outils création d'arc : en cliquant sur une transition puis sur une place (ou sur une place puis une transition) avec cet outil on crée un arc de la transition vers la place (de la place vers la transition) ;
- Outils ajout de jeton : en cliquant sur une place avec cet outil, on ajoute un jeton dans cette place ;
- Outils suppression de jetons : en cliquant sur une place contenant des jetons avec cet outils, on supprime un jeton de cette place ;
- Outils sélection : permet de déplacer une transition ou une place sur la grille. En sélectionnant un objet (transition, place, arc) et en appuyant sur le bouton Suppr du clavier, on supprime l'objet. En double cliquant sur un objet (ou clique droit), on peut accéder à ses propriétés (nom des places ou transitions par exemple) ;
- Outils simulation : permet de lancer la simulation du réseau de Petri.

4.3 - Première Modélisation du système de cuisson de pot (modélisation simplifiée)

Les différentes étapes d'opérations dans le système sont :

1. Un pot non cuit arrive sur le convoyeur 1
2. Le robot prend un pot non cuit convoyeur 1
3. Le robot met un pot non cuit dans le four
4. Le pot cuit
5. Le robot prend un pot cuit dans le four
6. Le robot pose un pot cuit sur le convoyeur 2
7. Un pot cuit sort du convoyeur 2

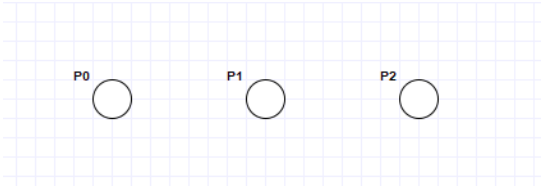
Les contraintes imposées par le système sont :

- Ct1 : Le four ne peut contenir que 2 pots en même temps
- Ct2 : Le robot ne peut transporter qu'un pot en même temps

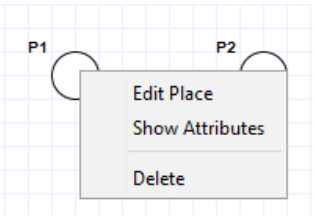
Pour le moment, commençons à modéliser le système sans prendre en compte les contraintes Ct1 et Ct2.

La première étape est de modéliser les différentes opérations du système, où l'on va considérer qu'un jeton représente un pot et que les places représentent des lieux où peuvent se situer les pots. Les transitions représentent les évènements faisant passer un pot d'un lieu à un autre.

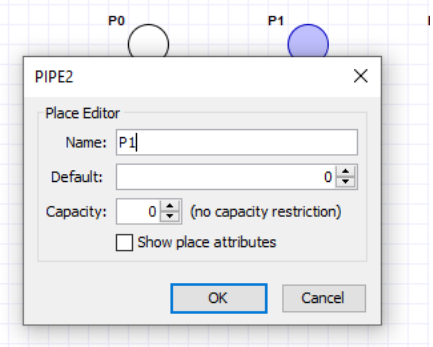
On s'intéresse tout d'abords aux opérations 1,2 et 3. Les lieux dans lesquels les pots (non cuits) peuvent se situer sont : sur le convoyeur 1, embarqué sur le robot entre le convoyeur 1 et le four et enfin, dans le four. On crée donc ces trois places dans le logiciel à l'aide de l'outils création de place.



En faisant une clique droite sur une place, on obtient cette fenêtre :

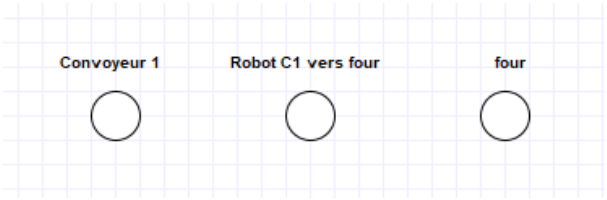


On appuie ensuite sur le bouton Edit Place, et on obtient cette fenêtre :

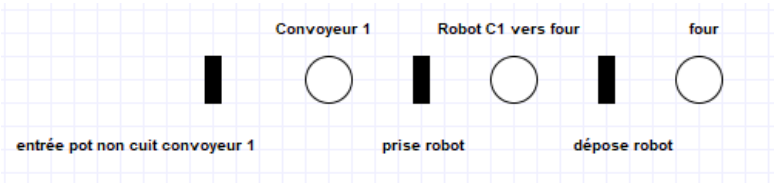


Dans la zone de texte « Name » on peut changer le nom de la Place.

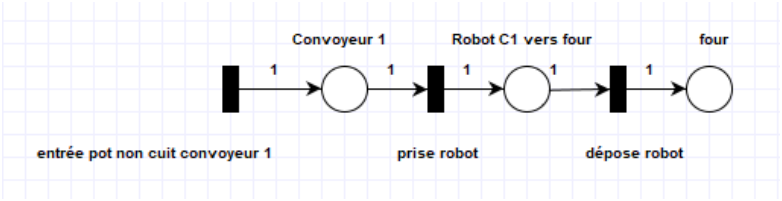
On renomme ainsi nos trois places par les lieux correspondants :



A l'aide de l'outils transition, on place ensuite des transitions et on les renomme avec l'évènement correspondant :

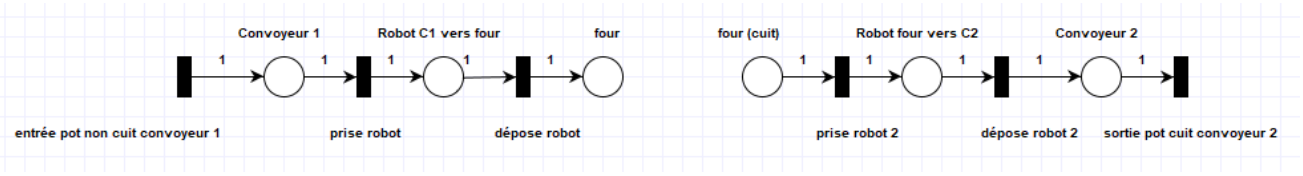


On relie ensuite les transitions et les places dans l'ordre de déplacement des pots non cuits :

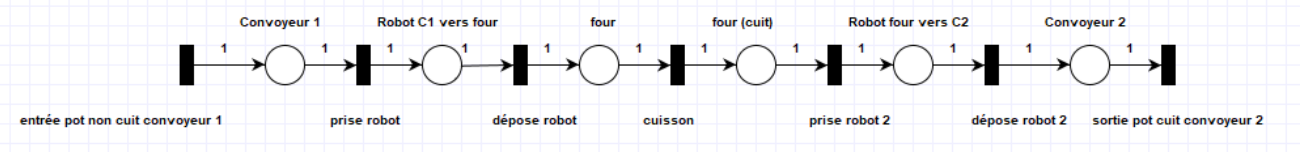


On va maintenant s'occuper de la partie avec les pots cuits (opérations 5, 6 et 7). On applique la même méthode qu'avec les pots non cuits : les lieux dans lesquels les pots (cuits) peuvent se trouver sont : dans le four, embarqué sur le robot pour le transfert du four vers le convoyeur 2 et sur le convoyeur 2. Les événements sont : chargement du robot pour passer de four à robot (prise robot 2), déchargement pour passer du robot au convoyeur 2 (dépose robot 2) et sortie d'un pot cuit du convoyeur 2 (sortie pot cuit convoyeur 2) pour sortir du convoyeur 2.

On obtient le résultat suivant :



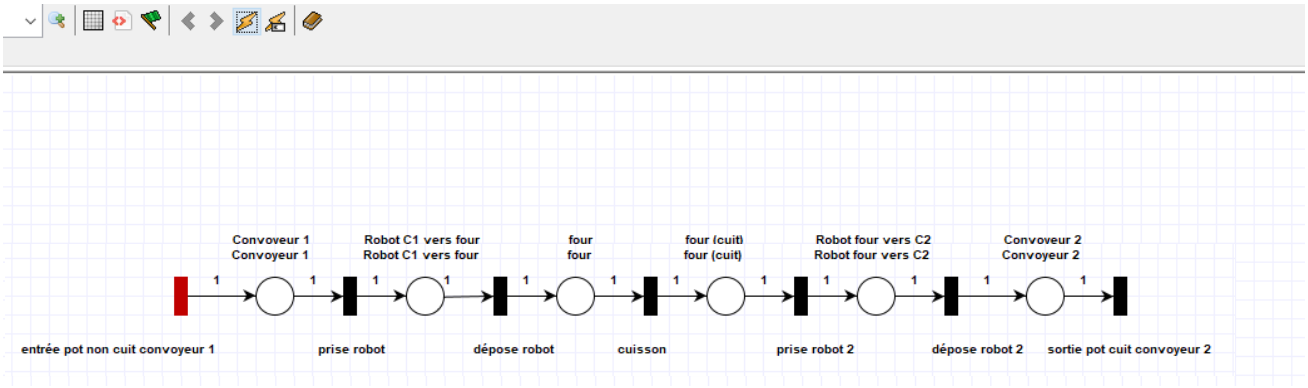
On relie ensuite les 2 parties par une transition représentant le fait qu'un pot non cuit va cuire (opération 4) pour devenir un pot cuit, et on obtient notre première version du réseau de Petri de notre système :



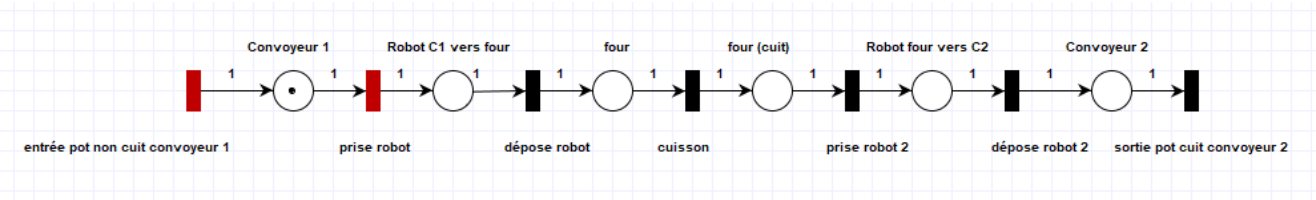
4.4 - Première simulation du modèle.

Nous allons maintenant procéder à la première simulation de ce modèle. Pour cela, cliquez sur l'outils simulation : 

Vous devriez alors vous retrouver dans cette situation :



Les transitions en rouge sont les transitions franchissables. Vous pouvez cliquer sur une transition franchissable pour la franchir. Si on clique sur la transition « entrée pot non cuit convoyeur 1 », on obtient le résultat suivant :



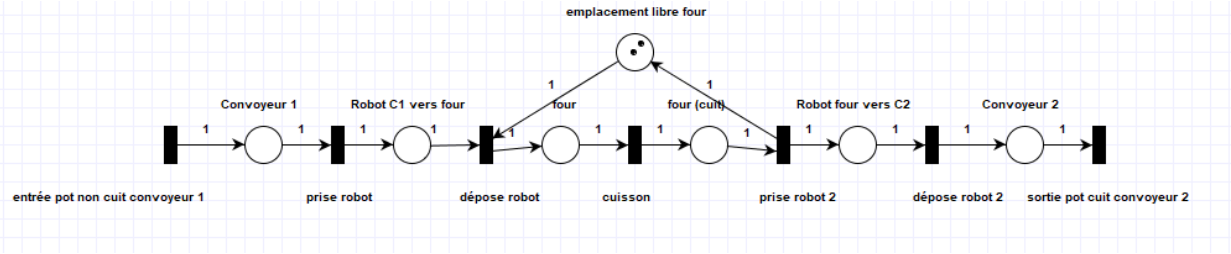
Un jeton a été rajouté dans la place Convoyeur 1. Cela symbolise qu'il y a maintenant un pot non cuit sur le convoyeur 1. Une nouvelle transition « prise robot » est donc maintenant franchissable : puisqu'il y a un pot non cuit sur le convoyeur 1, le robot peut maintenant prendre le pot non cuit. Vous pouvez si vous le souhaitez continuer à franchir des transitions franchissables et observer l'évolution du réseau de Petri. Pour quitter la simulation, il suffit de réappuyer sur l'outils simulation : 

On a ici une première modélisation et simulation de la chaine d'opération que peut faire notre système. Cependant, nous n'avons pas encore pris en compte les contraintes de notre système (Ct1 et Ct2). Si vous le voulez, vous pouvez franchir des transitions de sorte à obtenir plus de deux jetons dans les places four et four (cuit), ce qui représenterait le fait qu'il y ait plus de deux pots dans le four, alors que cela n'est pas possible dans les caractéristiques de notre système. De plus vous pouvez charger le robot plusieurs fois sans le décharger, ce qui signifierait que le robot peut porter plusieurs pots à la fois, or ceci n'est pas censé être faisable par notre système.

4.5 - Modélisation réaliste du système de cuisson de pot

La contrainte la plus facile à modéliser est la contrainte du nombre de pots présents dans le four (Ct1).

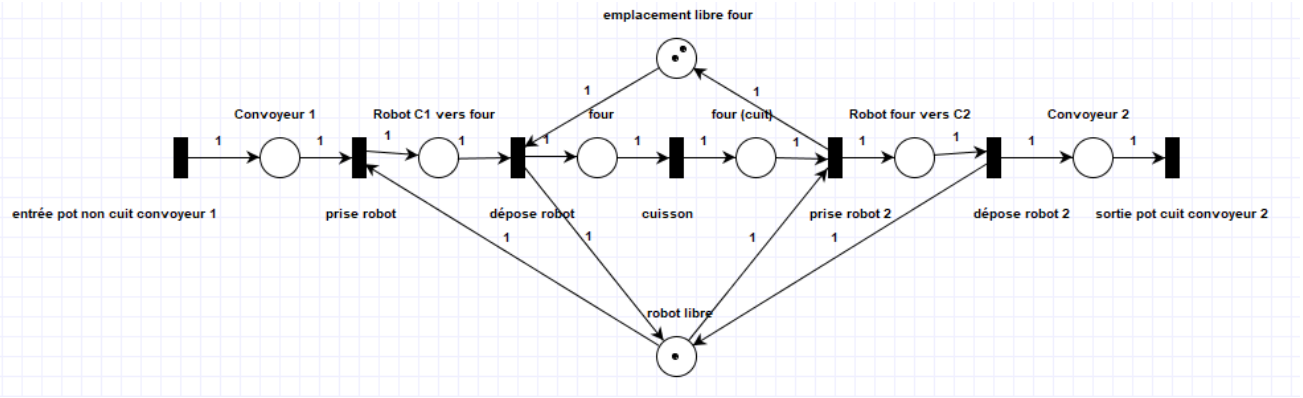
Pour cela, nous allons créer une place représentant le nombre d'emplacements libres dans le four. C'est-à-dire que s'il y a deux jetons dans cette place, cela signifiera qu'il y a deux emplacements libres dans notre four (dans ce cas, le four est vide). Ainsi, dès qu'un pot entre dans le four, on enlève un jeton à cette place, et dès qu'un pot quitte le four, on rajoute un jeton à cette place. On obtient le résultat suivant :



Il y a dans la nouvelle place « emplacement libre four » initialement deux jetons car initialement, le four est vide et peut accueillir deux pots. La place est ensuite reliée à « dépose robot », symbolisant le fait que lorsqu'un pot est ajouté dans le four, il prend un emplacement vide, il supprime donc un jeton de la place « emplacement libre four ». S'il n'y a plus de place dans le four (la place « emplacement libre four » ne contient plus de jeton), le robot ne pourra plus mettre de pièces dans le four puisque la transition « dépose robot » ne sera plus franchissable (il manquera des jetons dans la place « emplacement libre four »). La place « emplacement libre four » est ensuite reliée à la transition « prise robot 2 » parce qu'à chaque fois que le robot enlève un pot du four, cela rajoute un emplacement libre dans le four. On obtient alors bien un système où le four ne peut pas contenir plus de deux pots.

On va ensuite répéter ce même processus pour limiter le nombre de pot que le robot peut porter. Si vous le voulez, vous pouvez essayer de le faire vous-même avant de regarder la solution.

Pour éviter que le robot puisse porter plus d'un pot à la fois, on va, de la même manière que pour le four, créer une place « robot libre » et la relier aux transitions de chargement (prise robot) et de déchargement (dépose robot) du robot. On obtient le résultat suivant :



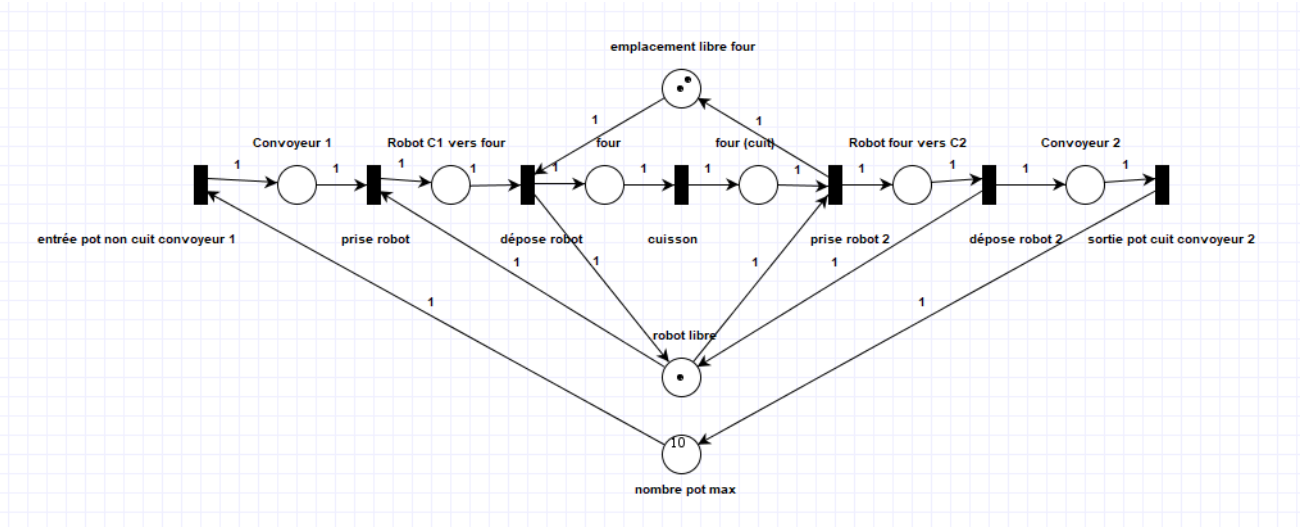
Ainsi à chaque fois que le robot prend un pot, le jeton de la place « robot libre » est supprimé et dès que le robot lâche un pot, un jeton est remis dans cette place.

4.6 - Application d'outils de vérification du modèle

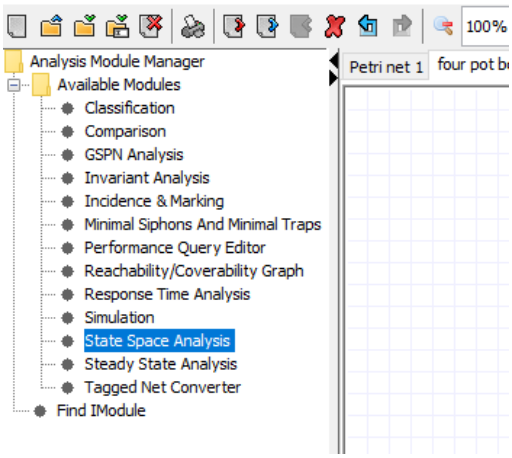
Afin de tester notre modèle, nous allons considérer une hypothèse réaliste qui consiste à supposer qu'il ne peut pas y avoir un nombre infini de pots simultanément dans notre système. On choisira arbitrairement un nombre maximal de dix pots. Le réseau de Petri est un type de modèle pour lequel a été développé un grand nombre de méthode d'analyses. Nous allons ici appliquer l'une d'entre elle, la recherche de « deadlock ».

Nous allons vérifier qu'il n'y pas de situation de blocage comme énoncé dans la partie 1.2. (Si on ne fait pas l'hypothèse qu'il y a un nombre fini de pots dans le système, on ne pourra jamais observer de cas de blocage où le système ne peux plus évoluer puisque l'on pourra toujours faire rentrer un pot non cuit sur le convoyeur 1).

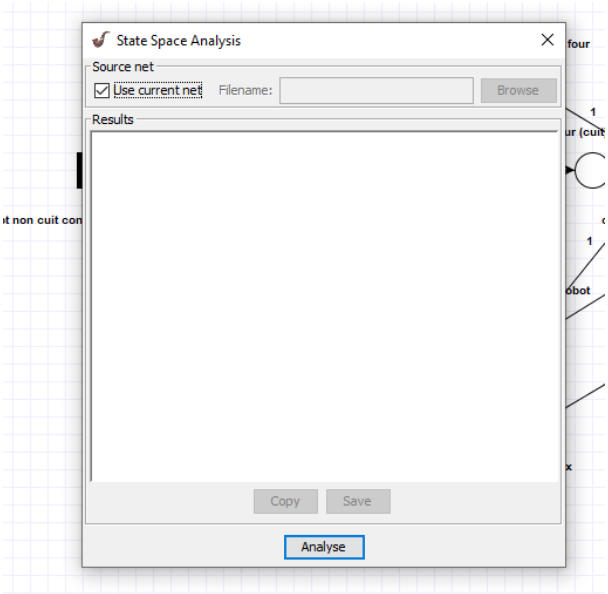
Le modèle retenu est donc le suivant :



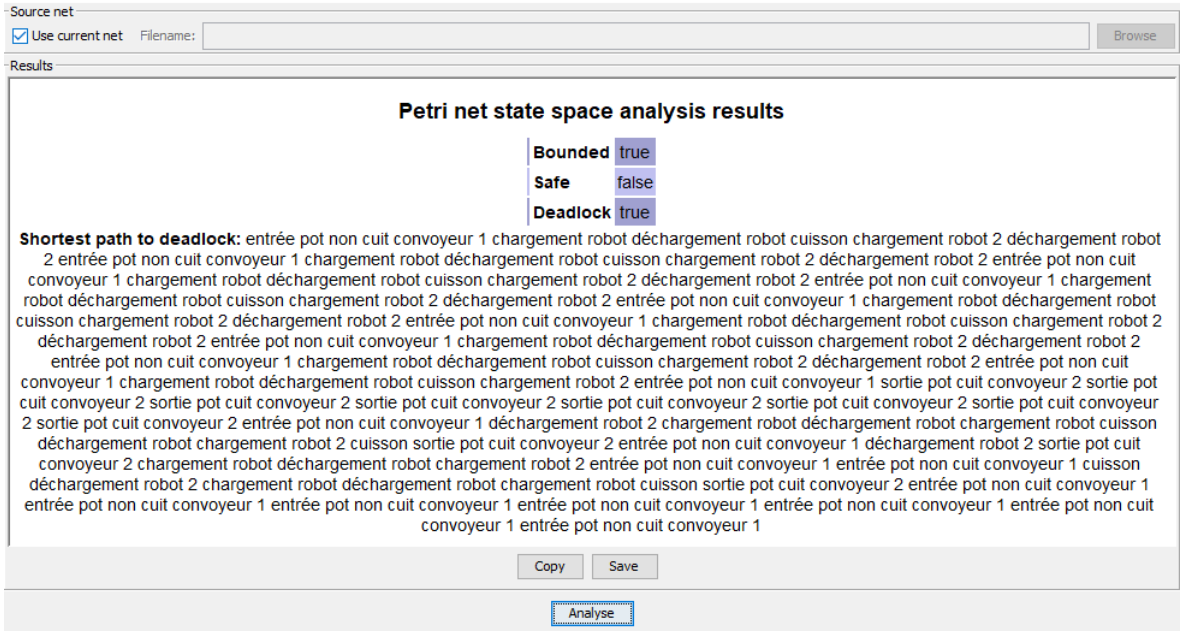
Nous allons ensuite utiliser l'outils « State Space Analysis » pour vérifier qu'il n'y a pas de deadlock (blocage) dans notre système. Pour ce faire on va double cliquer sur cet outil :



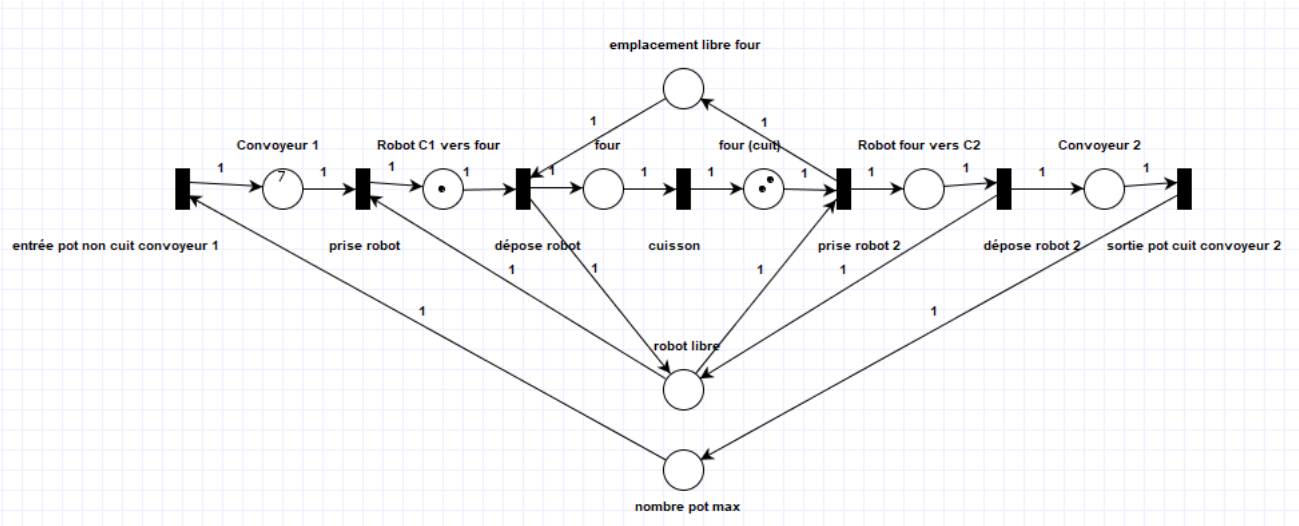
La fenêtre suivante va apparaitre :



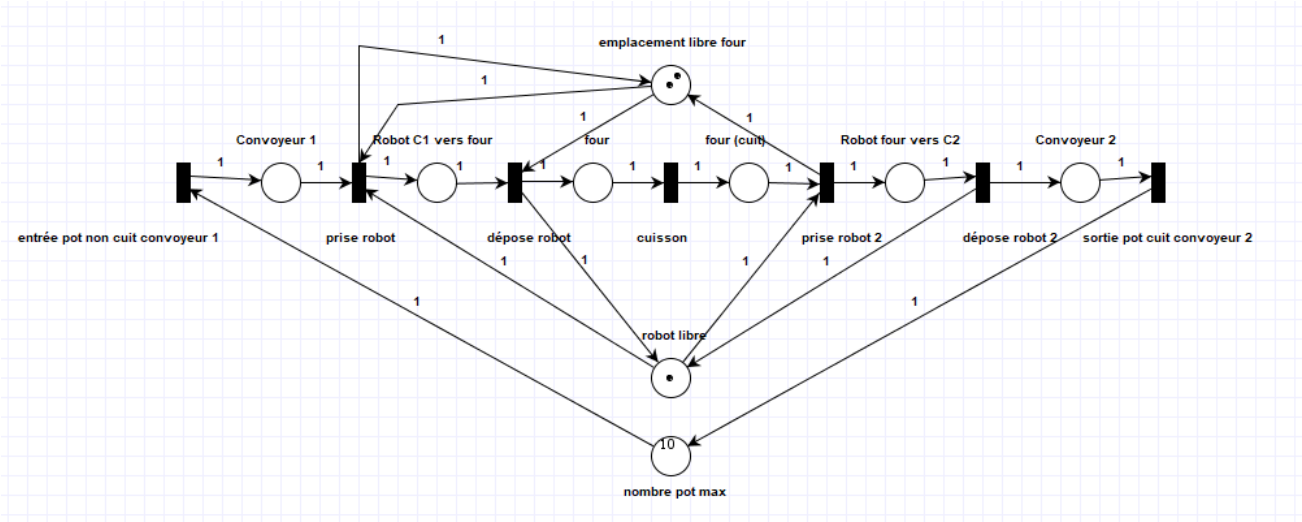
Cliquez ensuite sur « Analyse » et vous obtiendrez le résultat suivant :



On peut voir que la case « deadlock » est suivie de la mention « true », cela signifie qu'il y a un deadlock que l'on peut atteindre si l'on opère la séquence de transition décrite juste après : « shortest path to deadlock ». Cela correspond au cas où le robot s'est chargé d'un pot non cuit alors que le four est plein, le robot ne peut donc pas poser le pot dans le four, ni vider le four :

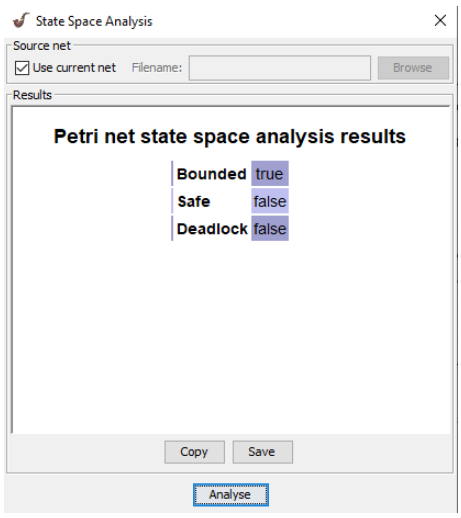


A ce stade, le concepteur du système observe qu'il y a un problème qu'il n'avait pas prévu, il va pouvoir ensuite modifier son modèle pour le corriger. Pour qu'il n'y ait pas de deadlock dans notre modèle, il faut qu'on rajoute une contrainte qui est que le robot ne peut pas se charger d'un pot non cuit si le four est plein. Pour cela, il suffit de rajouter deux arcs entre la transition « prise robot » et « emplacement libre four » :



De cette manière, la transition « prise robot » ne fait pas varier le nombre de jeton dans la place « emplacement libre four » (il en supprime un et en ajoute un), cependant, la transition « prise robot » n'est pas franchissable s'il n'y a pas d'emplacement libre dans le four.

En réappliquant l'analyse de deadlock précédente avec ce nouveau modèle, on obtient ce résultat :



Le système ne possède maintenant plus de deadlock, il n'y a pas de situation imprévue où il risque de se retrouver bloqué.

5 – Conclusion

Ceci conclu cette introduction à la modélisation de systèmes à évènement discret par réseau de Petri. Vous pouvez maintenant vous amuser si vous le souhaitez à modéliser de petits systèmes tels qu'une machine à café, un distributeur de billet etc.

Si vous voulez en apprendre plus sur les réseaux de Petri, découvrir d'autres utilisations pour l'industrie et plus particulièrement le formalisme mathématique matriciel qui est sous-jacent, je vous conseille d'aller lire cet article de R. Zurawski et MengChu Zhou [7]. Un site est aussi dédié aux réseaux de Petri et fourni des informations récentes sur l'avancée de la recherche dans ce domaine [2].

Si vous êtes plus intéressés par le fonctionnement de Pipe 2 et la théorie mathématique implémentée dans ce logiciel, vous pouvez aller lire l'article de Nadeem Akharware MEng. MIEE [6]

Références :

[1]: C. G. Cassandras et S. Lafortune, *Introduction to discrete event systems*, 2. ed. New York, NY: Springer, 2008

[2]: Petri Nets World : Petri Nets World: Online Services for the International Petri Nets Community (uni-hamburg.de)

[3]: T. Murata, « Petri Nets: Properties, Analysis and Applications », *PROCEEDINGS OF THE IEEE*, vol. 77, nº 4, p. 40, 1989.

[4]: Petri net - Wikipedia

[5]: Pipe2 : Platform Independent Petri net Editor 2 (sourceforge.net)

[6]: N. A. Me. Miee, « PIPE2: Platform Independent Petri Net Editor »

[7]: R. Zurawski et MengChu Zhou, « Petri nets and industrial applications: A tutorial », *IEEE Trans. Ind. Electron.*, vol. 41, nº 6, p. 567-583, déc. 1994, doi: 10.1109/41.334574.