

Travaux Dirigés n°1

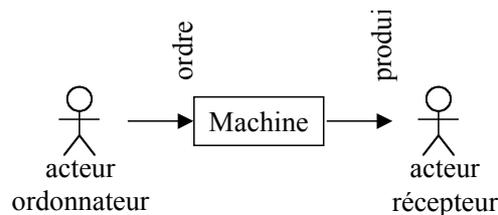
Ingénierie des protocoles - Réseaux de Petri

Correction

Question 1. Modélisation d'un atelier de fabrication

Question 1.1. Modélisation d'une machine de fabrication simple

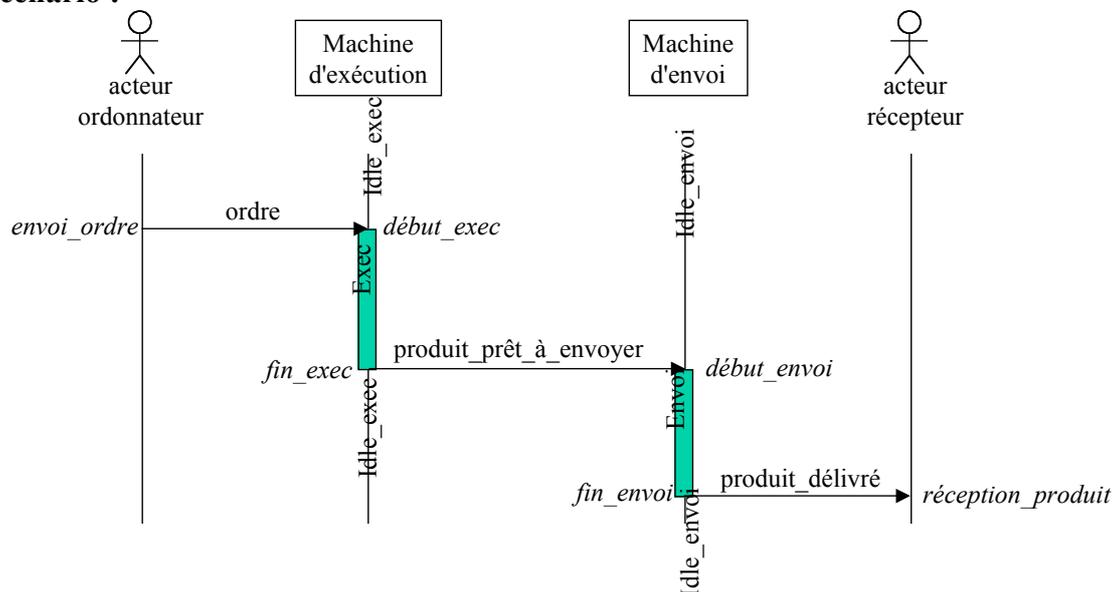
Considérons une première analyse du système selon une approche à la UML. Le diagramme des cas d'utilisation ci-dessous donne l'ensemble des acteurs externes du système ainsi que les interactions.



Toutefois, cette figure n'aide pas à préciser le mode des interactions entre les deux acteurs et la machine : est-ce une communication synchrone, du type rendez-vous, ou asynchrone ? Pour répondre à cette question, il est nécessaire d'imaginer plusieurs scénarios.

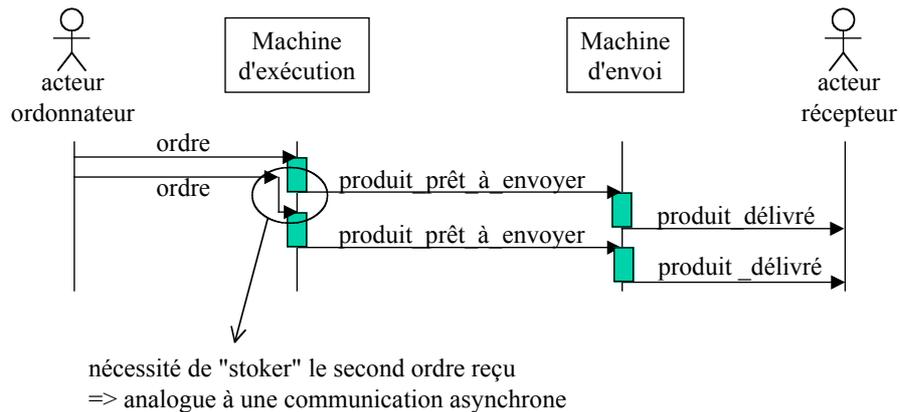
La machine est décomposée, selon l'énoncé, en deux machines organisées en pipe-line : une machine d'exécution et une machine d'envoi. On peut alors décrire l'ensemble des scénarios par des diagrammes de séquences. Trois scénarios sont envisageables :

1^{er} scénario :



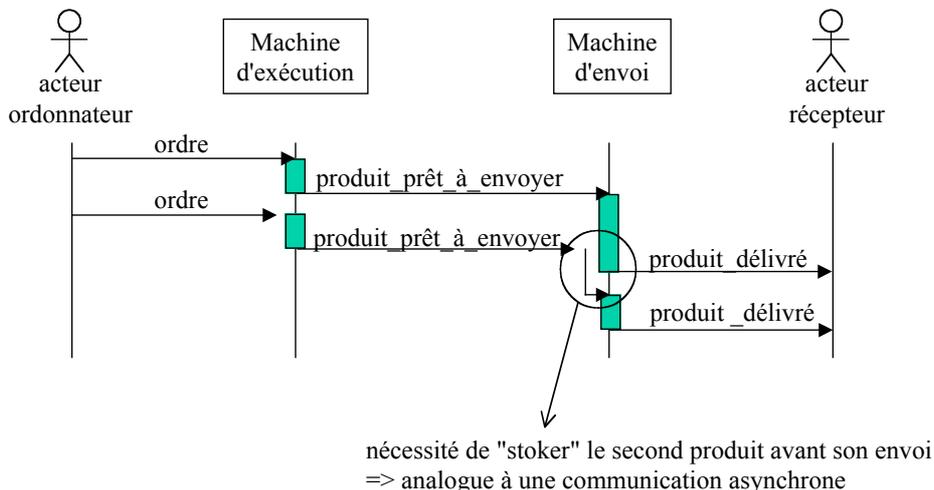
Ce premier scénario montre les états et les événements présents dans le système. Les états Idle_exec, Exec, Idle_envoi et Envoi deviendront des places, tandis que les événements envoi_ordre, début_exec, fin_exec... réception_produit deviendront des transitions. Toutefois, ce premier scénario n'indique toujours pas le besoin en synchronisation entre les différentes parties du système. Ce besoin apparaît avec les deux scénarios suivants.

2^{ème} scénario : arrivée d'un ordre alors que la machine d'exécution travaille



Ce second scénario montre que la communication entre l'acteur ordonnateur et la machine d'exécution nécessite une place pour stocker les ordres (qui seront donc modélisés comme des jetons) en attendant leur traitement par la machine.

3^{ème} scénario : sortie d'un produit à envoyer alors que la machine d'envoi travaille

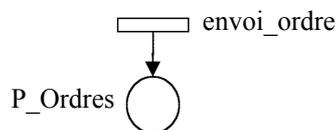


Ce troisième scénario montre que la communication entre la machine d'exécution et la machine d'envoi nécessite une place pour stocker les produits à envoyer (qui seront donc modélisés comme des jetons) en attendant leur traitement par la machine d'envoi.

De même on fera l'hypothèse que la communication entre la machine d'envoi et l'acteur récepteur est asynchrone. Les produits délivrés seront stockés dans une place avant réception effective (événement réception_produit) par l'acteur récepteur.

Modélisation de l'acteur « ordonnateur » :

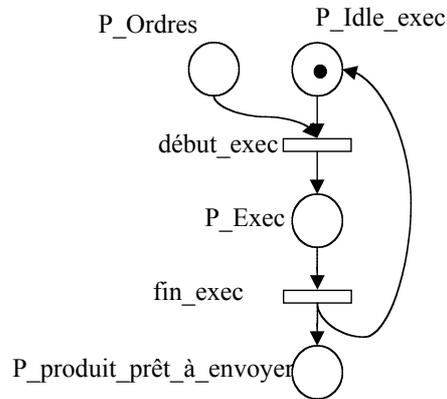
L'acteur « ordonnateur » se contente, d'après ces trois scénarios, d'envoyer un ordre indépendamment de l'état des machines. Cet ordre est stocké en attendant son traitement par la machine. Ce comportement peut être modélisé par le RdP suivant :



La place P_Ordres contiendra autant de jetons que d'ordres émis et en attente de traitement. Cette place assurera la communication de type asynchrone entre l'acteur ordonnateur et la machine d'exécution identifiée par le deuxième scénario. L'envoi d'un ordre est modélisé par le tir de la transition « envoi_ordre », transition qui n'a aucune précondition, et qui peut donc être tirée à tout moment.

Modélisation de la machine d'exécution :

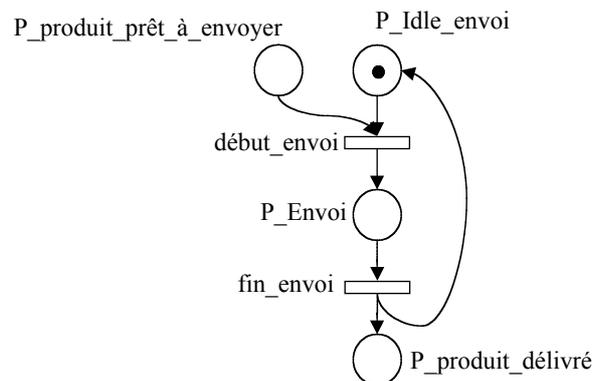
Un modèle simple de cette machine est donné par le RdP suivant :



Cette machine, selon ce modèle, a deux états : idle et en exécution. Ces deux états correspondent aux deux places P_Idle_exec et P_Exec. On retrouve les deux états identifiés sur le digramme de séquence du premier scénario. La fin d'une exécution provoque la production d'un produit à envoyer, modélisé par un jeton dans la place « P_produit_prêt_à_envoyer ». Cette dernière place assurera la communication de type asynchrone entre les machines d'exécution et d'envoi identifiée par le troisième scénario.

Modélisation de la machine d'envoi :

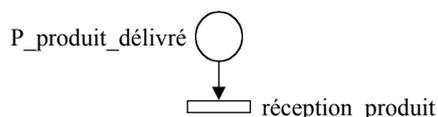
La machine d'envoi est similaire à la machine d'exécution :



Cette machine a deux états : idle et en envoi. Elle passe d'idle à envoi sur réception d'un produit à envoyer. Enfin, la fin de l'envoi provoque la production d'un « produit_délivré » modélisé par la production d'un jeton dans la place P_produit_délivré. Cette dernière place assurera la communication de type asynchrone entre la machine d'envoi et l'acteur récepteur.

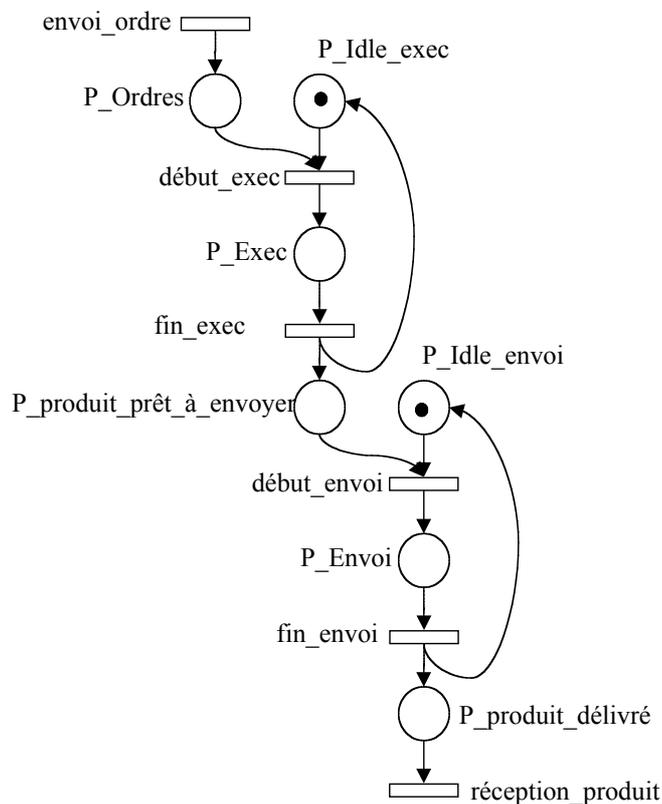
Modélisation de l'acteur « récepteur » :

L'acteur « récepteur » se contente, d'après ces trois scénarios, d'attendre et consommer les « produits » les uns après les autres dès que ceux-ci sont délivrés. Ce comportement peut être modélisé par le RdP suivant :



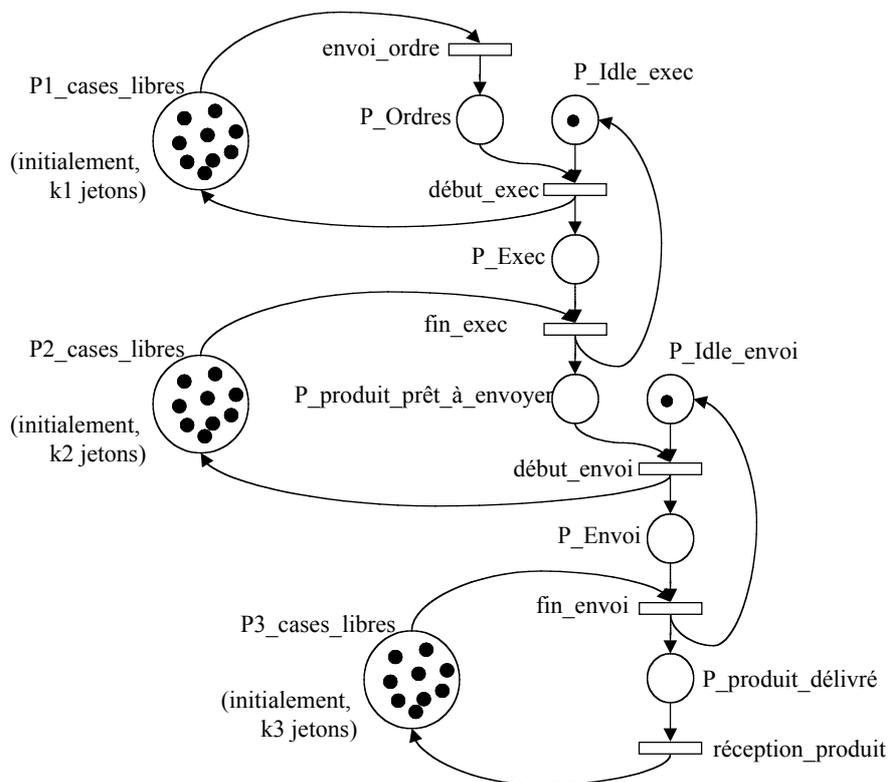
Modélisation du système global :

Les communications entre les différents acteurs et machines étant asynchrones, le modèle global est construit en fusionnant les places de même nom. Ce qui donne le modèle RdP suivant :



Il apparaît clairement que la place « P_ordres » n'est pas bornée. De même, les places « $P_produit_prêt_à_envoyer$ » et « $P_produit_délivré$ » ne sont pas bornées (il est facile de construire un marquage $M1$ tel que $M1(P_Idle_exec)=1$, $M1(P_Idle_envoi)=1$, $M1(P_produit_prêt_à_envoyer)=1$ et $M1(P_produit_délivré)=1$ et $M1(p)=0$ pour toutes les autres places p . Ce marquage est strictement supérieur au marquage initial, en particulier pour les places « $P_produit_prêt_à_envoyer$ » et « $P_produit_délivré$ », ce qui montre que ces places ne sont pas bornées.

Ce caractère non borné traduit une incorrection dans la modélisation. Pour corriger ce problème, il est nécessaire d'introduire une capacité maximale pour ces places, capacités notées respectivement k_1 , k_2 et k_3 , et introduire un mécanisme interdisant la production d'un ordre, d'un produit prêt à être envoyé, et d'un produit délivré, lorsque ces capacités sont respectivement atteintes. Un tel mécanisme peut être modélisé par l'ajout de places supplémentaires caractérisant les « cases » libres pouvant stocker des ordres, des produits prêts à être envoyés, et des produits délivrés. Le modèle global est donné par le RdP suivant :



Selon ce nouveau modèle, l'acteur ordonnateur ne peut envoyer un ordre que s'il y a encore au moins un jeton dans la place P1_cases-libres, c'est-à-dire s'il y a encore un espace pour stocker cet ordre. De même pour la production d'un produit prêt à être envoyé et d'un produit délivré.

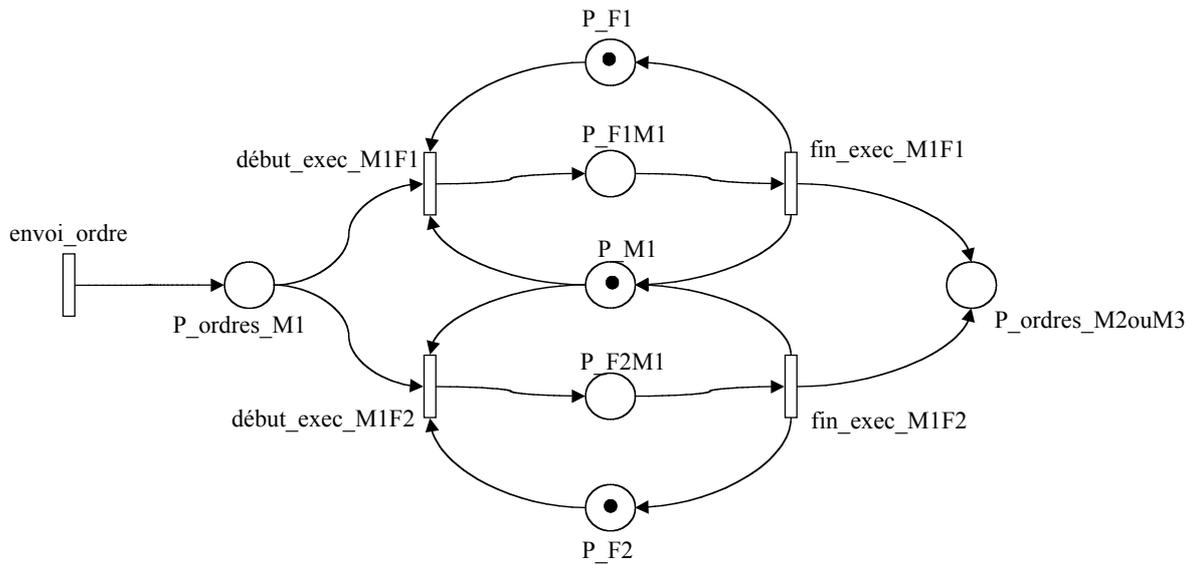
Il est facile de voir que les couples de places (« P1_cases_libres », « P_ordres »), (« P2_cases_libres », « P_produit_prêt_à_envoyer ») et (« P3_cases_libres », « P_produit_délivré ») forme chacun des composants conservatives positives. Le nombre de jetons contenus dans ces places est donc borné, et cette borne est donnée par le marquage initial :

- le nombre maximum de jetons dans P_ordres est k1
- le nombre maximum de jetons dans P_produit_prêt_à_envoyer est k2
- le nombre maximum de jetons dans P_produit_délivré est k3

Question 1.2. Modélisation d'un atelier de fabrication

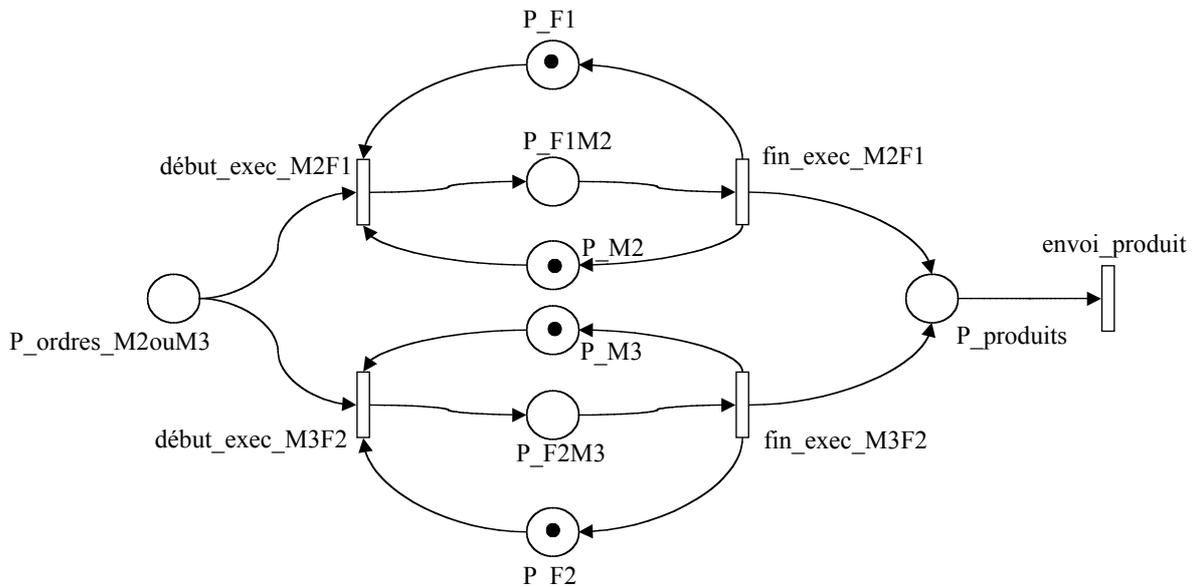
On suppose que les machines sont maintenant atomiques. Nous ne modéliserons donc plus à ce niveau le fonctionnement interne de chaque machine comme dans la question précédente, mais nous nous concentrerons en revanche sur le fonctionnement global de l'atelier.

La première étape de l'atelier est composée de la machine M1 avec les deux opérateurs F1 et F2. A ce niveau, M1, F1 et F2 peuvent être modélisés comme des sémaphores, c'est-à-dire des ressources libres ou utilisées. Cette étape peut être modélisée par le RdP suivant :



L’envoi d’un ordre provoque la production d’un jeton dans la place P_ordres_M1 (notons que là encore nous avons fait le choix d’une communication asynchrone entre l’atelier et l’acteur ordonnateur externe). Ensuite, si la machine M1 est libre (un jeton dans P_M1) et si l’opérateur F1 (resp. F2) est libre (un jeton dans P_F1 (resp. dans P_F2)), alors la transition $début_exec_M1F1$ est franchissable, ce qui conduit à occuper M1 et F1 (resp. F2) (on retire le jeton dans P_M1 et P_F1 (resp. P_F2)), et produire un jeton dans P_M1F1 (resp. P_M1F2). En fin d’exécution, on rend à nouveau disponible la machine et l’opérateur en remettant un jeton dans les places respectives, puis on produit un ordre vers la seconde étape de l’atelier (un jeton dans P_ordres_M2ouM3).

La seconde étape est à peu près similaire à la seconde, à ceci près qu’elle met en jeu deux machines M2 (opérateur F1) et M3 (opérateur F3). Là encore F1, F2, M2 et M3 sont vus comme des sémaphores :

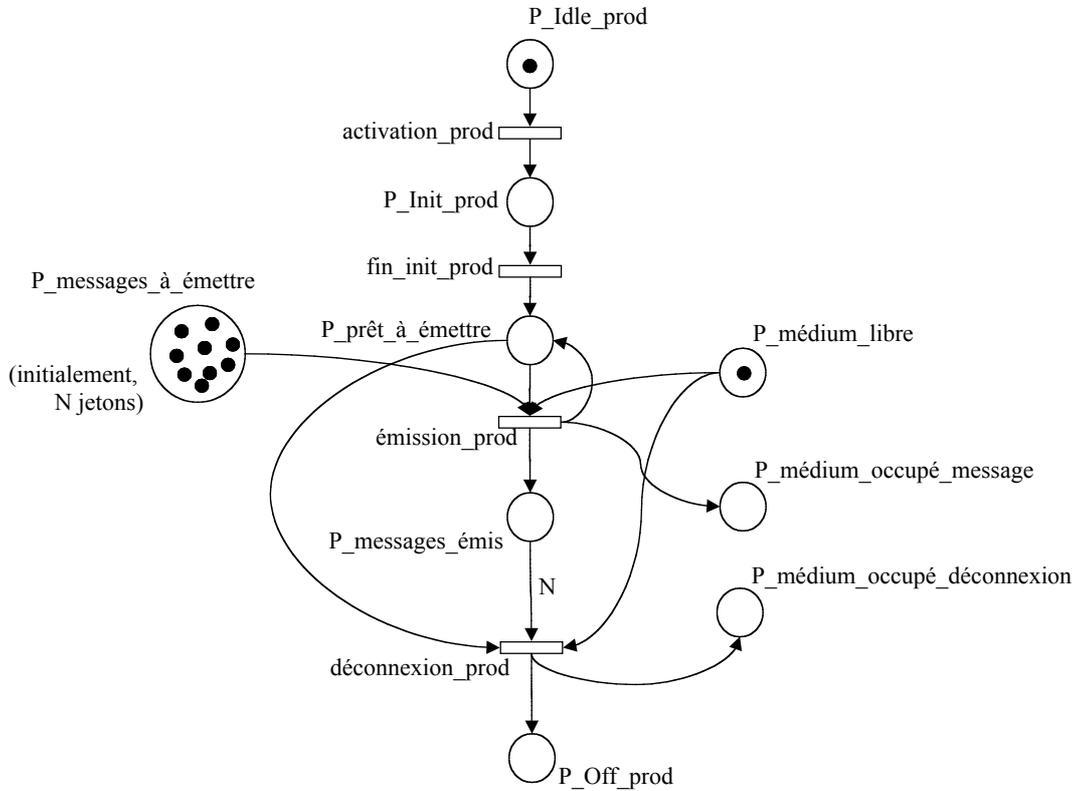


Il est clair que les places « P_ordres_M1 », « P_ordres_M2ouM3 » et « $P_produits$ » ne sont pas bornées. Par corriger ce problème on applique la même méthode que dans la question précédente.

Question 2. Protocole Producteur - Consommateur

Question 2.1. Médium = buffer à une case

Modèle du producteur :



Modèle du consommateur :

