BULLETIN D'INFORMATIQUE APPROFONDIE ET APPLICATIONS

COMPUTATION - INFORMATION

N° 83 – JUIN 2009

DIRECTEUR

Jean - Michel Knippel

FONDATEUR

Edmond Bianco

SERVEUR DE PUBLICATION

Christian Blanvillain

SECRETARIAT

Kalassoumi Adjilani

Université de Provence
Equipe Hermès. Case 33
3 place Victor Hugo
F - 13331 Marseille Cedex 3
Téléphone: (0)4 91 10 62 30
Télécopie : (0)4 91 50 91 10

DEPOSITAIRE

Université de Provence
Bibliothèque Universitaire
1 place Victor Hugo
F - 13331 Marseille Cedex 3
Téléphone : (0)4 91 10 85 29
Télécopie : (0)4 91 95 75 57

IMPRIMEUR

Université de Provence
Service Reprographie
3 place Victor Hugo
F - 13331 Marseille Cedex 3
Téléphone : (0)4 91 10 60 48

1 **EDITORIAL**
Il faut revoir le temps de travail des enseignants !
Le décret de 1950 est vraiment trop vieux !

par Laurent Tarillon

3 **Introduction aux Réseaux de Petri**

par Robert Valette

23 **Exemple du photographe - Pas 1**
Le réseau de Petri vu comme un ensemble d'automates communicants

par Robert Valette

37 **VOUZZAVEDIBISAR**
Petri Nets 2009. Call for Papers and Announcement

par Jean - Baptiste Voron

http://www.univ-provence.fr/biaa

Publication trimestrielle, gratuite, de l'Université de Provence

Aix – Marseille Université

Impression : juillet 2009

ISSN 0291 - 5413

EDITORIAL

Il faut revoir le temps de travail des enseignants ! Le décret de 1950 est vraiment trop vieux !

Laurent Tarillon

Actuellement, le temps de travail d'un enseignant de collège ou de lycée est de 18 heures par semaine. C'est, pour les professeurs certifiés, le seul élément fixe et clair relatif au temps de travail qui leur est demandé. Il a été fixé par une décret datant de 1950. Rendez-vous compte ! Ce qui détermine ce qu'un enseignant doit à la nation date de 56 ans. Il est vraiment temps de se pencher sur un problème aussi lourd de conséquences.

18 heures par semaine ! Quel salarié ne voudrait pas travailler aussi peu pour d'aussi bons salaires ?

Comment le législateur a-t-il pu créer en 1950 un statut aussi avantageux ? Il faut peut-être y regarder de plus près. En fait, ce temps a été conçu en prévoyant qu'un enseignant travaille 1,5 heure chez lui pour une heure devant élève afin de préparer ses cours, évaluer les élèves et actualiser ses connaissances dans sa discipline. Cela fait 18 fois 2,5 heures (1 devant les élèves et 1,5 à la maison), soit 45 heures hebdomadaires. En effet, le temps de travail légal de l'époque, s'il était légalement de 40 heures par semaine, était en réalité de environ 42 h par semaine, sur 50 semaines.

Mais que s'est-il passé depuis pour les enseignants ? Rien ! Alors que pour les autres salariés il y a eu la troisième semaine de congés payés en 1956, puis quatre en 1969. Les 40 heures réelles ont été atteintes au début des années 70 (elle était un droit depuis 1936). Mais ce n'est pas fini, il y a eu les 39 heures et la cinquième semaine en 1982, puis les 35 heures en 2000. En somme le temps de travail hebdomadaire pour les salariés a baissé de 25 %. Mais les enseignants doivent toujours le même service.

C'est au moins un enseignant qui écrit cela vous dites vous, en lecteur éclairé ! Certes je l'avoue, je fais partie de ces privilégiés. Car, comment peut-on parler de temps de travail sans parler des vacances ? Eh bien justement, le législateur a tout prévu et cela de deux façons. D'abord 45 heures dues quand les autres devaient 42, ça c'est pour les petites vacances (Toussaint, Noël...). Car les vacances c'est pour que les élèves se reposent. Donc, notre temps de travail était annualisé. Mais, et les deux mois d'été alors ? Là, c'est un tout petit peu plus compliqué. Certains enseignants ne le savent même pas, d'ailleurs. Cela se situe au niveau de la grille des salaires. Notre grille a

été, elle aussi, fixée en 1950 au même niveau que les autres cadres de la fonction publique recrutés avec un concours au niveau bac + 3. Mais à cette grille, il nous a été retiré deux mois de salaire, puis le résultat a été divisé par 12. (Par exemple si un inspecteur des impôts est payé 2000 € par mois il recevra 24 000 € par an, mais pour la même qualification, un enseignant recevra aussi 2000 € par mois mais sur 10 mois, soit 20000 € par an. Cette somme est ensuite divisée par 12 et donne 1667 € par mois.) Et oui chers lecteurs les enseignants ne sont pas payés pendant les grandes vacances.

Oui bon d'accord, peut-être que nous ne sommes pas si privilégiés que cela concernant le temps de travail. Mais côté salaires, quand même, nous ne sommes pas à plaindre ! Eh bien soit, comparons. Nous sommes nettement en dessous de la moyenne des cadres du privé comme du public (qu'on nous prouve le contraire). Mais l'exemple le plus frappant pour moi de la dégradation de la valeur que la nation accorde à ceux qui éduquent ses enfants est le suivant. Le salaire de départ d'un enseignant en 1970 était 2 fois supérieur au SMIC. Aujourd'hui, il n'est plus que 1,2 fois plus élevé. Autrement dit, si comme le PS l'a écrit dans son projet le SMIC augmentera de 25 % au cours des cinq ans à venir (et l'UMP l'a augmenté au même rythme annuel dès cette année), un enseignant débutant gagnera moins que le SMIC. Faudra-t-il en arriver là, pour que la société se rende compte de la dégradation de notre situation ?

Alors oui le décret de 1950 est vieux ! Il est vraiment temps de le toiletter comme le disent nos gouvernants ! Mais dans quel sens ? Est-ce en faisant en sorte que nous devions travailler plus pour gagner autant comme veut le faire Monsieur de Robien ?

A suivre…

Laurent Tarillon
Enseignant de sciences économiques et sociales à Grenoble.

Tarillon.laurent@free.fr

Introduction aux Réseaux de Pétri

Robert Valette

LAAS-CNRS Toulouse,

version d'avril 2007

Pré-requis : Ce texte suppose que l'on a entendu parler des systèmes linéaires et des automates finis. Son but est d'expliciter la genèse des Réseaux de Petri à partir des automates et donc d'expliciter également certaines des similitudes et des différences existant entre ces deux techniques de modélisation.

Table des matières

1 Systèmes à événements discrets **5**

1.1 Les types de variables 5

1.2 Types de systèmes 5

1.3 Représentation de la dynamique 7

2 Rappel sur les automates finis **7**

2.1 Définition des automates finis 7

2.2 Exemple 8

2.3 Réceptivité 8

3 Limite des automates à états finis **9**

4 Communications entre automates **11**

4.1 Types de communications 11

4.2 Automates avec communications synchrones 12

4.3 Automates avec communications asynchrones 12

5 Des automates aux réseaux de Pétri **13**

5.1 Principe 13

5.2 Comparaison vis-à-vis de la réceptivité 13
5.3 Représentation d'automates indépendants 14
5.4 Représentation des communications 16
5.4.1 Communication synchrone : 16
5.4.2 Communication asynchrone : 17
6 Définition informelle des réseaux de Petri **18**
6.1 Définition informelle des concepts 18
6.2 Exemple 19

1 Systèmes à événements discrets

Les réseaux de Petri sont un outil permettant la modélisation et l'analyse de systèmes à événements discrets. Avant de présenter les réseaux de Petri, précisons ce que sont ces systèmes à événements discrets (en Anglais *"Discrete Event Dynamic Systems"* ou DEDS).

1.1 Les types de variables

Lorsque l'on construit un modèle mathématique d'un système, on a l'habitude de classer les variables en deux catégories

- Les *variables continues* sont des variables qui prennent leurs valeurs sur le domaine des réels $\mathcal{R}$.
- Les *variables discrètes* sont des variables qui prennent leurs valeurs sur un domaine dénombrable comme l'ensemble des entiers naturels $\mathcal{N}$, ou bien sur des ensembles dont le nombre d'éléments est fini.

A partir de ce classement, les systèmes dynamiques peuvent être rangés dans diverses catégories selon les modèles mathématiques qui sont utilisés pour les représenter. Nous allons rapidement passer en revue ces diverses catégories.

1.2 Types de systèmes

Les systèmes continus : Le modèle d'un système continu ne comprend que des variables continues. De plus, le temps est également une variable continue. Quand le temps est représenté par une variable continue, on dit qu'il s'agit d'un *temps dense*. C'est le grand domaine des systèmes représentés par des équations algébro-différentielles. Un exemple de système continu avec une seule variable d'état est représenté figure 1.

Les systèmes échantillonnés : Ce sont ceux pour lesquels on a choisi d'avoir une représentation discrète du temps associée à des variables d'état continues. Le temps est en effet représenté comme une suite infinie d'instants repérés par des entiers naturels
$\theta_0, \theta_1, \ldots, \theta_{n-1}, \theta_n, \theta_{n+1}, \ldots$
De plus, on suppose en général que ces instants de temps sont également répartis. La valeur d'une variable d'état à un instant donné ($x(\theta)$, par exemple) est un nombre réel. Un exemple de système échantillonné avec une seule variable d'état est représenté figure 2, le pointillé représentant un modèle continu sous-jacent.

Cette technique de modélisation peut être utilisée dans deux cas :

- le système modélisé (le modèle continu sous-jacent) comporte effectivement un échantillonneur que le modèle mathématique représente fidèlement,
- le système représenté ne comporte pas d'échantillonneur, il s'agit d'un système continu, mais pour éviter une représention fine par un système algébro-différentiel complexe, on choisit une représention plus grossière en ne s'intéressant aux valeurs des variables qu'à certains instants prédéterminés.

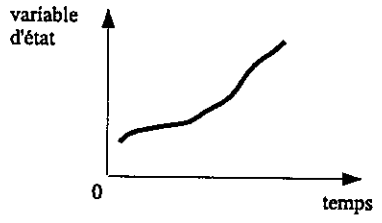

FIG. 1 – *Système continu*

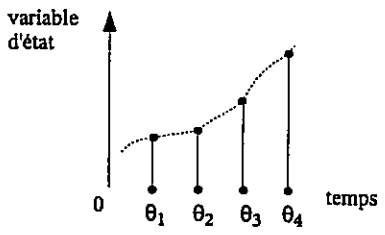

FIG. 2 – *Sytème échantillonné*

Le deuxième cas correspond à une vision *qualitative* par discrétisation du temps.

Les systèmes discrets : Ce sont ceux pour lesquels, on a choisi une représention continue du temps (temps dense) associée à une représention discrète de l'état. Cela peut être le cas lorsque l'état est naturellement discret. Par exemple une lampe est allumée ou éteinte, un équipement est dans un état de bon fonctionnement ou bien il est défaillant *etc.* Cela peut également être le cas lorsque l'on souhaite n'avoir qu'une vue qualitative d'un système. Par exemple, pour un ascenseur, on ne souhaite pas entrer dans le détail des phases d'accélération et de ralentissement et en conséquence l'état de la cabine sera codé par un ensemble fini de valeurs : {*"arrêté au rez-de chaussée"*, *"en mouvement entre le rez-de chaussée et le premier étage"*, *"arrêté au premier étage"*, *"en mouvement entre le premier et le second étage"*, ..., *"arrêté au dernier étage"*}. Un exemple de système discret avec une seule variable d'état est représenté figure 3.

Si l'on a une vision *qualitative*, la grande différence par rapport au cas précédent est que l'on discrétise les variables d'état, mais par contre le temps reste dense. On s'intéresse à la position de l'ascenseur à tous les instants du temps dense, même si la position est approximative (entre deux étages). Dans le cas des systèmes échantillonnés, on s'intéressera par exemple à la position exacte de l'ascenseur toutes les secondes.

Les systèmes à événements discrets : Ce sont ceux pour lesquels on ne conserve de la représentation du système qu'une suite *d'événements* discrets. Le temps n'est codé que par la suite des événements. Il est donc discrétisé. Les événements sont discrets car ils correspondent à des changements d'états entre des états discrets du système. Temps et états sont donc discrétisés. Les événements ne peuvent pas être répartis également dans le temps comme dans le cas des systèmes échantillonnés. En effet, ils correspondent à des passages d'un état qualitatif du système à un autre. La durée, vue d'un point de vue continu, s'écoulant entre deux événements successifs, dépend de la dynamique du système. Si nous revenons à l'exemple de l'ascenseur, les événements peuvent être, par exemple, l'arrivée de l'ascenseur à chaque étage. Suivant les appels et les arrêts, ces événements ne seront pas du tout répartis également dans le temps. Un exemple de système à événements discrets avec une seule variable d'état est représenté figure 4.

En prenant toujours le cas de l'ascenseur, On sait que partant du rez-de-chaussée, il n'atteindra le deuxième étage qu'après être passé par le premier étage. Mais, le temps mis pour atteindre cet étage n'est pas spécifié.

Les systèmes hybrides : Ce sont ceux qui comprennent à la fois des variables d'état continues et des variables d'état discrètes. On peut aussi à la fois manipuler du temps dense, codé par des réels et du temps discret sous la forme d'un ensemble d'événements. C'est donc une classe de systèmes qui comprend les quatre précédentes.

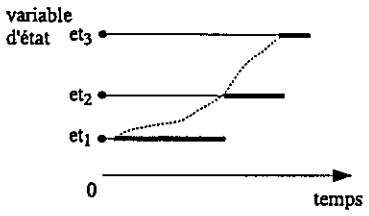

FIG. 3 – *Système discret*

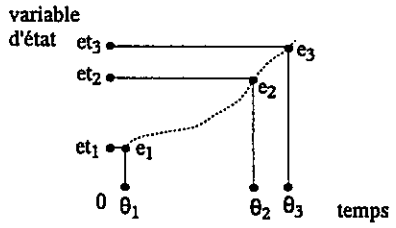

FIG. 4 – *Sytème à événements discrets*

1.3 Représentation de la dynamique

Le comportement dynamique d'un système continu est, de façon classique, représenté par un système d'équations différentielles. Dans le cas le plus simple (systèmes linéaires), il est possible de calculer de façon analytique l'évolution des variables d'état au cours du temps, à partir d'un état initial. Il est donc possible d'avoir des caractérisations fortes des domaines de variation des variables d'état de tels systèmes.

Dans le cas des systèmes à événements discrets, chaque changement d'état est un événement. On ne dispose pas de la notion de dérivée et l'ensemble des états doit être défini par énumération. Soit il s'agit d'une énumération explicite comme dans le cas des automates finis, soit il s'agit d'une énumération implicite lorsque l'on met plusieurs automates finis en parallèle (l'ensemble des états potentiels est le produit cartésien des ensembles des états de chaque automate) ou que l'on utilise des réseaux de Petri, comme nous allons le voir par la suite.

Ce qu'il faut bien garder à l'esprit c'est que la différence fondamentale entre les systèmes continus et les systèmes à événements discrets résulte de l'absence de la notion de dérivée dans le second cas. Le grand problème des systèmes à événements discrets est donc celui de l'explosion combinatoire du nombre des états puisque ces états doivent en général être énumérés, sous une forme ou sous une autre, pour étudier les propriétés de ces systèmes.

2 Rappel sur les automates finis

2.1 Définition des automates finis

La technique la plus ancienne et la plus classique pour représenter le comportement dynamique d'un système à événements discrets est d'utiliser un automate fini (c'est-à-dire un automate ayant un nombre fini d'états). Un automate fini (on dit aussi machine à états finis), en anglais *"finite state automaton"* ou *"finite state machine"* est formée d'un ensemble *fini* d'états et d'une fonction définissant les transitions entre états (ce sont les événements).

La définition formelle d'un automate fini est la suivante :

Définition 1 *Un automate fini est un quadruplet* $\mathcal{A} = (S, E, T, s_0)$ *avec :*

- S *est un ensemble* fini *d'états,*
- E *est un alphabet,*

Définition de l'automate :

- $S = \{s_0, s_1, s_2\}$
- $E = \{e_1, e_2, e_3, e_4\}$
- s_0 est l'état initial

(pour T voir figure)

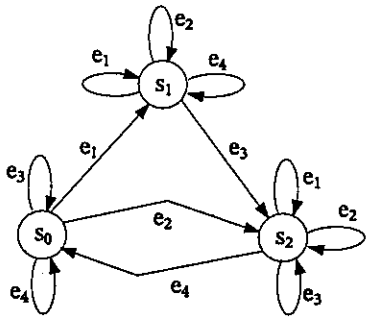

FIG. 5 – *Exemples d'automate fini*

- *T est la fonction de transition $T : S \times E \rightarrow S$*
- *$s_0 \in S$ est l'état initial.*

Nous retrouvons bien les deux notions fondamentales des systèmes à événements discrets : les états (les éléments de S) et les événements (les éléments de E). La fonction de transition exprime que lorsque le système est dans un état donné, les événements sont susceptibles de le faire changer d'état. Elle décrit donc la dynamique du système à événements discrets.

2.2 Exemple

Considérons l'automate fini de la figure 5. Il possède 3 états, 4 événements et 12 arcs.

Quand il est dans l'état s_0, l'occurrence de l'événement e_1 le fait passer dans l'état s_1. Nous avons donc :

$$T(s_0, e_1) = s_1 \tag{1}$$

Si l'événement e_1 survient quand le système est dans l'état s_1 le système reste dans cet état :

$$T(s_1, e_1) = s_1 \tag{2}$$

Nous voyons qu'un état peut avoir deux états suivants. Par exemple, s_0 a comme états suivants s_1 et s_2. Ces états suivants sont différenciés par les événements [1].

2.3 Réceptivité

La réceptivité consiste à exprimer le fait qu'un automate dans un état donné peut très bien n'être *réceptif* qu'à un sous ensemble des événements de E. Cela veut dire que dans cet état il ne peut reconnaître qu'un sous ensemble de E. Le domaine de définition de T n'est donc plus le produit cartésien ($S \times E$) des ensembles S et E mais un sous ensemble de cet ensemble.

1. Un automate tel que défini par la définition 1 est donc déterministe. Des automates non déterministes ont également été définis dans la littérature.

Définition de l'automate :

- $S = \{s_0, s_1, s_2\}$
- $E = \{e_1, e_2, e_3, e_4\}$
- s_0 est l'état initial

(pour T voir figure)

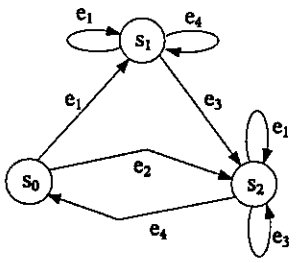

FIG. 6 – *Exemples d'automate fini avec la notion de réceptivité*

Il est très différent de dire que l'occurrence d'un événement e_1 ne change pas un état s_1 ou bien de dire que cet état n'est pas réceptif à cet événement [GI 73]. Dans le second cas, si e_1 survient, on se trouve dans une situation mathématiquement non définie qui, *dans la pratique* sera en général interprétée comme une *erreur*. Dans le premier cas, aucune erreur ne peut bien évidemment être détectée puisque la transition est définie.

Si nous revenons à l'exemple de l'ascenseur, cela n'apporte pas une information particulièrement significative de spécifier (cas de l'automate classique) que lorsque l'ascenseur est arrêté au premier étage, l'événement *"arrivée au deuxième étage"* ne le fait pas changer d'état. Il est par contre beaucoup plus utile de préciser qu'un tel événement est anormal et correspond probablement à une défaillance (par exemple d'un capteur). Par contre, dans l'état *"arrêté au premier étage"*, l'ascenseur peut être réceptif au bouton d'appel du premier étage. Cet événement ne provoque pas de changement d'état de l'automate représentant le système, mais on peut souhaiter le noter car si l'on augmente le niveau de détail du modèle cet événement deviendra nécessaire. Par exemple, le bouton d'appel peut relancer la temporisation[2] associée à l'ouverture de la porte.

Dans l'exemple de la figure 6, l'état s_0 n'est réceptif qu'aux événements e_1 et e_2, l'état s_1 aux événements e_1, e_3 et e_4 et l'état s_2 aux mêmes événements e_1, e_3 et e_4. Si lorsque le système est dans l'état s_0 l'un des événements e_3 ou e_4 a lieu, une incohérence sera détectée.

Un état peut toujours avoir plusieurs états suivants en préservant le déterminisme, et un événement peut apparaître plusieurs fois dans le graphe. Par exemple les états s_0, s_1 et s_2 sont tous les trois réceptifs à l'événement e_1.

3 Limite des automates à états finis

Un système complexe est toujours décrit par morceaux, de façon modulaire. Si on a un système à événements discrets complexe, on va donc être amené à le décrire à l'aide de plusieurs automates.

Considérons par exemple un système formé de deux sous-systèmes, chaque sous-système étant représenté par un automates possédant deux états et deux événements comme représenté figure 7. Si nous les considérons ensemble comme un seul système à événements discrets, il faut

2. A ce niveau plus détaillé la spécification de temporisations indique qu'un temps dense est introduit. Le modèle détaillé est donc en fait un modèle hybride et non plus un système à événements discrets. Dans ce modèle hybride l'événement provoque un changement d'état.

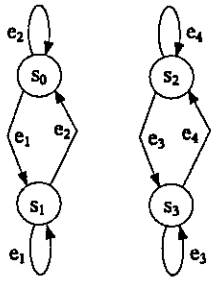

FIG. 7 – *Automates considérés séparément*

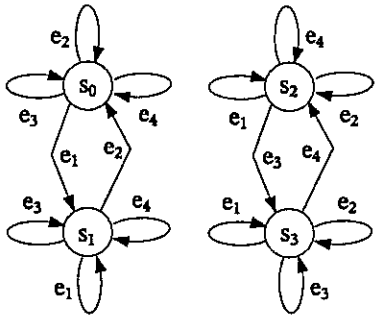

FIG. 8 – *Automates considérés ensemble*

d'abord que l'on plonge chacun des automates représentant un sous-système dans l'ensemble des événements du système global[3]. Si E est cet ensemble, nous avons

$$\begin{aligned} E &= E_1 \cup E_2 \\ &= \{e_1, e_2\} \cup \{e_3, e_4\} \\ &= \{e_1, e_2, e_3, e_4\} \end{aligned} \tag{3}$$

Les ambiguïtés provenant de l'absence d'explicitation des réceptivités vont se renforcer. Une boucle élémentaire sur un état peut alors signifier :

- l'événement est possible dans cet état mais ne le change pas à ce niveau de détail (il y aura des changements d'états avec des modèles plus détaillés),
- l'événement n'est pas possible (fonctionnement anormal et détection éventuelle de défaillance),
- l'événement concerne une autre partie du système.

Ensuite il faut engendrer l'ensemble des états possibles en faisant le produit cartésien des ensembles d'états des deux sous-systèmes

$$\begin{aligned} S &= S_1 \times S_2 \\ &= \{s_0, s_1\} \times \{s_2, s_3\} \\ &= \{s_{02}, s_{12}, s_{03}, s_{13}\} \end{aligned} \tag{4}$$

avec $s_{ij} = s_i \times s_j$. On obtient finalement l'automate fini de la figure 9

Nous rencontrons alors un second problème, celui de l'explosion combinatoire du nombre des états. En effet, si l'on considère deux sous systèmes ayant chacun 20 états, le système résultant aura $20 \times 20 = 400$ états. L'énumération explicite des états est donc un processus lourd et lorsque

3. Pour simplifier nous considérons ici le cas où il n'y a pas d'événements en commun entre les deux sous-systèmes.

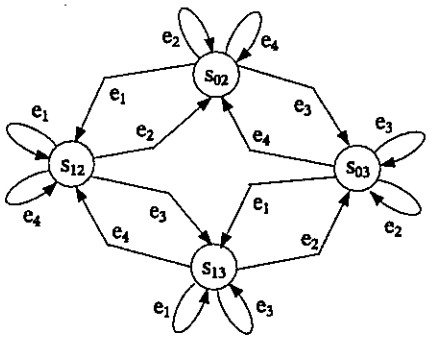

FIG. 9 – *Automate produit représentant les deux sous-systèmes*

l'on étudie des trajectoires d'un tel système cela implique de raisonner dans un espace d'état de grande taille que les ingénieurs auront du mal à conceptualiser.

En résumé nous voyons les deux limites des automates finis pour la représentation de systèmes à événements discrets complexes (c'est-à-dire composé d'un ensemble de sous systèmes) : la non explicitation de la notion de réceptivité et l'explicitation de de tous les états lorsque l'on construit l'automate fini global. Il est donc important de travailler le plus possible au niveau de la représentation décomposée en sous-systèmes (celle de la figure 7). Il est d'autant plus important de conserver cette vision décomposée que l'automate global (celui de la figure 9) fait disparaître complètement la structure du système. Par exemple, si l'on observe la séquence d'événements $(e_1 ; e_3)$[4], rien ne permet de remarquer sur la figure 9 qu'il s'agit en fait de la composition de 2 événements e_1 et e_3 qui sont indépendants car ils appartiennent à deux sous-systèmes différents. L'événement e_3 n'est pas la conséquence de l'événement e_1, même si dans ce scénario il apparaît temporellement après lui.

Dans l'exemple simple que nous venons de voir, les deux sous-systèmes sont complètement indépendants et la construction de l'automate produit est en fait inutile. Dans le cas général des communications ont lieu entre les sous-systèmes et ces communications sont essentielles pour comprendre le fonctionnement du système global. De plus, elles peuvent induire des dysfonctionnements graves comme nous le verrons plus en détail lors de l'analyse et la vérification des réseaux de Petri. Comment peuvent-elles être représentées?

4 Communications entre automates

4.1 Types de communications

Il y a deux grands types de communications : les communications *"synchrones"* et les communications *"asynchrones"*. Le mot *"synchrone"* implique l'existence d'un référentiel temporel (une horloge) commun. Le mot *"asynchrone"* implique son absence. Dans le premier cas, l'échange d'information se fait par des actions (émission et réception) qui ont lieu de façon exactement simultanée dans les deux sous-systèmes en train de communiquer. Dans le second cas émission et réception se font indépendamment, la seule contrainte étant que l'émission précède

4. Le connecteur ";" signifie ici *"en séquence avec"*

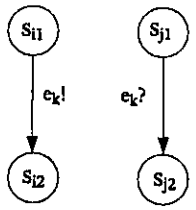

FIG. 10 – *Communication synchrone*

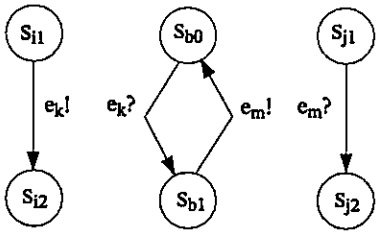

FIG. 11 – *Communication asynchrone*

la réception.

Un exemple de communication synchrone est la communication téléphonique entre deux personnes. Pour que la communication ait lieu il faut que l'appelé décroche et quand l'une des deux personnes parle (émission) l'autre doit écouter (réception). Un exemple de communication asynchrone est l'envoi d'un courrier (ou un SMS). Quand une personne compose et émet le message le destinataire n'a rien à faire. Une fois le message émis, le destinataire peut le lire quand il le souhaite. Au moment de la lecture, la personne ayant émis le message n'a rien à faire. La mise en œuvre d'une communication asynchrone nécessite l'existence d'une *boîte aux lettres*, c'est-à-dire d'un *tampon* (en anglais *"buffer"*) permettant d'accumuler les messages émis et non encore reçus.

4.2 Automates avec communications synchrones

Comment ces deux types de communication peuvent-ils être représentés lorsqu'ils ont lieu entre deux sous-systèmes représentés par des automates finis? Soient deux automates finis i et j avec une communication synchrone entre eux. Cette communication est vue comme en événement e_k commun aux deux automates (voir la figure 10). L'événement e_k fait passer l'automate i de l'état s_{i1} à l'état s_{i2} et *simultanément*, il fait passer l'automate j de l'état s_{j1} à l'état s_{j2}.

Dans le cas où c'est nécessaire, on peut préciser la partie émission de l'événement e_k en la notant $e_k!$ et la partie réception de ce même événement e_k en la notant $e_k?$. Mais il est toujours supposé que $e_k!$ et $e_k?$ sont parfaitement synchrones c'est-à-dire qu'ils ont lieu exactement au même moment.

4.3 Automates avec communications asynchrones

Soient deux automates finis i et j avec une communication asynchrone. Cette communication implique deux événements différents e_k et e_m et un tampon de mémorisation b (voir la figure 11), jouant le rôle de boîte aux lettres. Le tampon (*"buffer"*) peut être vu comme un troisième automate à deux état: s_{b0} quand il est vide et s_{b1} quand il contient un message émis mais non encore reçu.

La communication asynchrone peut alors être vue comme une séquence de deux communications synchrones[5] : la première prend place entre les automates i et b (l'émission), la seconde entre b et j (la réception).

5. Paradoxalement, on peut également voir une communication synchrone comme une séquence de communica-

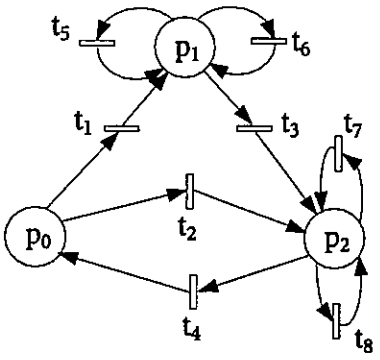

FIG. 12 – *Réseau de Petri associé à l'automate de la figure 6*

5 Des automates aux réseaux de Petri

5.1 Principe

Ce que l'on appelle maintenant les réseaux de Petri est né de la thèse de Carl Adam Petri [PE 62] intitulée *Communications entre automates*. À partir de systèmes décrits sous la forme d'un ensemble de sous-systèmes communicants, l'idée était de décrire de façon *unifiée* les changements d'états internes des automates associés à des sous-systèmes et les communications entre ces automates.

Pour cela C.A. Petri a proposé d'utiliser un graphe avec deux types de nœuds : les **places** et les **transitions**. Les places, représentées graphiquement par des *cercles* sont associées à des états (cette notion sera raffinée par la suite) et les transitions, représentées par des *rectangles*[6] décrivent les changements d'états.

Dans un premier temps, cela consiste donc simplement à ajouter un rectangle au milieu de chaque arc décrivant la transition d'un état vers un autre. Ainsi, l'automate de la figure 6 prend la forme de la figure 12.

Le changement est bien plus important qu'il n'apparaît au premier abord. En effet, au lieu de privilégier les états (seuls nœuds des automates), les transitions entre états deviennent également des nœuds. Dans cette technique de représentation des systèmes à événements discrets, les événements vont être des éléments de base au même titre que les états.

Voyons tout d'abord les conséquences de cette technique de représentation vis-à-vis de la notion de réceptivité.

5.2 Comparaison vis-à-vis de la réceptivité

Quand on utilise un automate pour représenter un système à événements discrets, de façon classique, les états sont des entités internes au modèle, et les événements sont associés à une vision

tions asynchrones. Par exemple au téléphone, *A* compose le numéro, *B* entend la sonnerie, *B* décroche et parle, *A* écoute *etc.*

6. Les rectangles pouvant être très aplatis, les transitions sont parfois représentées par des barres. Parfois, à l'opposé, elles sont représentées par des carrés.

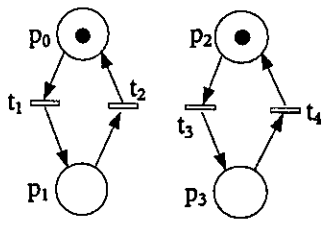

FIG. 13 – *Réseau de Petri associé aux automates de la figure 7*

externe. La fonction de transition explicite comment l'occurrence d'un événement modifie l'état interne de l'automate. Un événement donné peut être la cause de plusieurs types de changements d'état. Par exemple, dans le cas de l'automate de la figure 6, l'événement e_1 peut faire passer de l'état s_0 à l'état s_1, ou bien maintenir le système dans l'état s_1. Il n'y a pas identité entre l'ensemble des événements et l'ensemble des transitions entre états, et la relation n'est simple que si l'on ne prend pas en compte la notion de réceptivité.

Dans un réseau de Petri l'ensemble des transitions entre états est explicité (comme l'est l'ensemble des états). Si l'on désire retrouver une vision extérieure d'événements pouvant provoquer divers changements d'états, il faut le faire explicitement en étiquetant (ou interprétant) les transitions du réseau de Petri.

Prenons l'exemple du réseau de Petri de la figure 12. Il y a 8 transitions (de t_1 à t_8) alors que l'alphabet de l'automate de la figure 6 ne comportait que 4 événements. Pour retrouver la même information, il faut *étiqueter* les transitions par la fonction d'interprétation I suivante :

t	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
$I(t)$	e_1	e_2	e_3	e_4	e_1	e_4	e_1	e_3

Cela veut dire que, par exemple, les transitions d'état t_1, t_5 et t_7 sont vues de la même façon par l'extérieur du système.

De façon totalement symétrique, on peut étiqueter les places d'un réseau de Petri, et supposer que les états associés aux places p_1 et p_2 sont vues de la même façon de l'extérieur du système. Si nous revenons à notre exemple d'ascenseur, nous pouvons parfaitement avoir plusieurs états internes correspondant à la vision de la cabine arrêtée au premier étage.

La réceptivité est donc naturellement prise en compte dans un réseau de Petri. Si on met une boucle élémentaire, c'est que l'on désire expliciter un événement pouvant se produire dans un état donné sans changer cet état au niveau d'abstraction considéré. C'est au contraire le fait que plusieurs transitions d'état sont vues comme un seul événement qui nécessite une information supplémentaire : la *fonction d'interprétation* ou d'étiquetage I.

5.3 Représentation d'automates indépendants

Considérons l'ensemble de deux automates des figures 7 et 8 en supposant que tous les événements provoquent des changements d'états (les boucles élémentaires sont donc des représentations inutiles, sans sens physique). La représentation sous la forme d'un réseau de Petri est celle de la figure 13[7].

7. On peut considérer que cette figure ne représente qu'*un seul* réseau de Petri car le graphe associé à un réseau de Petri n'est pas nécessairement connexe

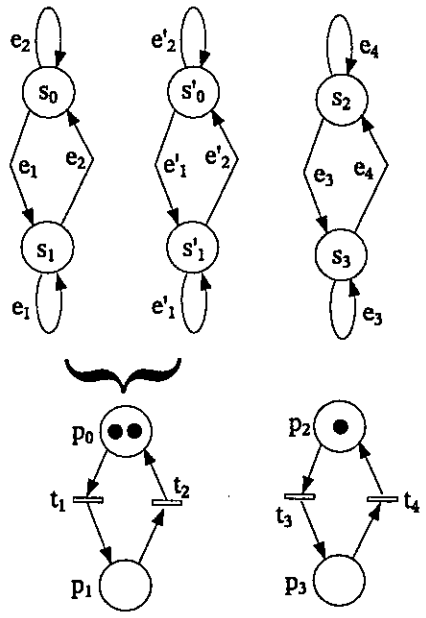

FIG. 14 – *Représentation de deux instances du même automate*

On peut alors faire deux remarques :

1. Il est clair que les transitions entre états t_3 et t_4 n'ont aucune influence sur le sous système décrit par les places p_0 et p_1 car aucun arc ne les relie à ces places.
2. On a bien conservé la structure du système formée de deux entités indépendantes

L'état initial du système est spécifié en mettant un **jeton** dans chacune des places p_0 et p_2. À partir de cet état le système ne peut évoluer que de deux façons différentes, soit par la transition t_1, soit par la transition t_3. En effet, le nœud t_1 (respectivement t_3) possède une seule place en entrée, la place p_0 (respectivement p_2) et cette place contient un jeton. Les autres transitions ont leur place d'entrée vide. On dit que la transition t_1 (respectivement t_3) est **sensibilisée**. Cela veut dire que le système, dans l'état initial, n'est *réceptif* qu'aux événements associés aux transitions t_1 et t_3.

Si l'événement associé à t_1 se produit, alors on peut procéder au **franchissement** de cette transition (on dit également **tir** d'une transition). Le franchissement de t_1 consiste à enlever le jeton de p_0 et à mettre un jeton dans p_1.

Dans ce nouvel état, seules les transitions t_2 et t_3 sont sensibilisées. L'état du système est codé par la présence des deux jetons, l'un dans la place p_1 et l'autre dans la place p_2. Une distribution de jetons dans les places s'apelle un **marquage**. Un marquage définit l'état du système.

Si le système avait été composé de deux automates du type de celui comprenant les états s_0 et s_1 et par un automate du type de celui comprenant les états s_2 et s_3, le réseau de Petri le représentant serait toujours celui de la figure 13 du point de vue de la structure. Simplement son marquage initial serait formé de **deux** jetons dans la place p_0 et d'un jeton dans la place p_2. Une place peut donc *contenir plusieurs jetons* (voir la figure 14).

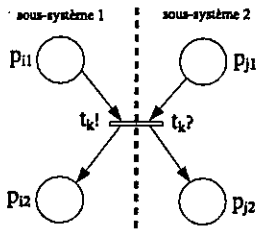

FIG. 15 – *Communication synchrone*

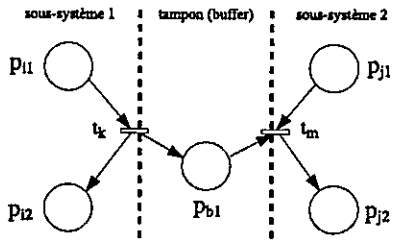

FIG. 16 – *Communication asynchrone*

Notons le fait que dans la représentation sous la forme du réseau de la figure 14, (réseau de Petri ordinaire) les événements e_1 et e'_1 (respectivement e_2 et e'_2) ne peuvent pas être différenciés car ils sont tous les deux associés aux franchissements de la transition t_1 (respectivement t_2). Les réseaux de Petri colorés qui seront introduits dans le chapitre intitulé *"Les données et le temps"* permettent de différencier les divers jetons situés dans une même place. Ils permettent de retrouver cette information.

5.4 Représentation des communications

La représentation de deux automates indépendants ne présente qu'un intérêt limité. Nous avons vu qu'il y avait deux types de communication et que les réseaux de Petri avaient également deux types de nœuds. Nous avons déjà mentionné le fait que le but de C.A. Petri était de pouvoir utiliser les mêmes éléments de base (les mêmes primitives) pour décrire les automates associés aux sous-systèmes et pour décrire les communications entre ces automates. Nous allons maintenant voir qu'effectivement les transitions permettent de décrire les communications synchrones et les places les communications asynchrones.

5.4.1 Communication synchrone :

Comme nous l'avons vu précédemment (voir les sous-sections 4.1 et 4.2), une communication synchrone est caractérisée par l'occurrence simultanée de la variante émission et de la variante réception du même événement. Cela implique une transition d'état simultanée dans chacun des deux sous-systèmes communicants. Si le système est représenté par un réseau de Petri, une communication synchrone est alors simplement représentée par *la fusion de deux transitions*, l'une associée à l'émission et l'autre à la réception (voir figure 15).

Cela n'est possible que parce les *transitions sont représentées par des nœuds* dans un réseau de Petri et non par des arcs comme dans les automates.

L'occurrence d'une communication synchrone entre deux sous-systèmes est donc ici[8] associée au franchissement d'une transition ayant deux places en entrée et deux places en sortie (figure 15). Cette transition est **sensibilisée** si chacune de ces deux places contient *au moins un jeton*[9]. Cela veut dire qu'il y a *au moins* une instance d'un sous-système prête à émettre et *au*

8. Nous verrons dès la sous-section suivante que ce n'est qu'un cas particulier.
9. Il y aura plus d'un jeton s'il y a plus d'un sous-système ayant le même comportement.

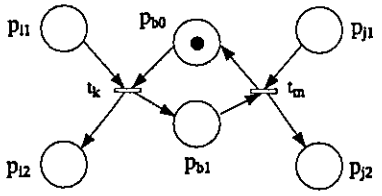

FIG. 17 – *Un seul message stocké*

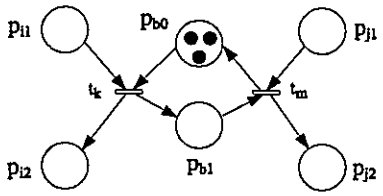

FIG. 18 – *Possibilité de recevoir 3 messages*

moins une instance prête à recevoir. Le **franchissement** de cette transition se fait en **enlevant un jeton** à chacune des places d'entrée et en **ajoutant un jeton** à chacune des places de sortie. Cela veut dire qu'une instance de sous-système a effectué une émission et a changé d'état en conséquence, et qu'une instance de sous-système a effectué une réception et a également changé d'état en conséquence. Ces deux changements d'état sont *synchrones*.

5.4.2 Communication asynchrone :

Comme nous l'avons vu précédemment (voir les sous-sections 4.1 et 4.3), pour un tel type de communication, l'émission et la réception ne se font pas au même moment. Par contre ce type de communication nécessite un tampon (*"buffer"*) capable de conserver le message émis jusqu'à ce qu'il soit reçu. Ce tampon est commun entre les deux entités et il correspond à un état (message émis et non encore reçu) et non à un événement.

La représentation par réseau de Petri est donc fondée sur *une place partagée* entre les deux sous-systèmes (voir figure 16, la place partagée est la place p_{b1}).

On retrouve le fait qu'une communication asynchrone peut être vue comme la séquence de deux communications synchrones : l'émission (ici la transitition t_k) et la réception (ici la transition t_m).

En comparant les figures 11 et 16 on remarque que dans le réseau de Petri on ne retrouve pas l'équivalent de l'état s_{b0}. Cela vient du fait que si l'on choisit de représenter le système par un ensemble d'automates finis, il faut que le tampon soit également un automate fini et il faut énumérer ses états. Les réseaux de Petri sont en fait *plus généraux que les automates finis* et on peut ne spécifier que l'existence de la partie du tampon contenant le ou *les* messages émis et non reçus. La différence de représentation correspond à une différence de comportement. Il est possible d'avoir plusieurs messages en attente avec la représentation donnée par le réseau de Petri.

Dans le cas de la figure 11, une fois le message émis, il faut attendre qu'il soit consommé pour pouvoir en émettre un autre car il faut que l'automate fini représentant le tampon soit dans l'état s_{b0} pour qu'une émission soit possible. Il y a donc au plus un message non lu en attente. Dans la solution décrite par le réseau de Petri, rien n'empêche d'émettre plusieurs fois avant de recevoir. Les messages vont s'accumuler dans le tampon, et cela sera représenté par *plusieurs jetons* dans la place p_{b1}. Comme ce nombre de jetons n'est pas *à priori* limité, le système aura un nombre d'états accessibles infini *et ne pourra donc pas être représenté par un automate fini*[10].

10. Il faudrait ici un automate à pile.

Il est tout à fait possible d'avoir un réseau de Petri ayant le comportement de l'ensemble des automates de la figure 11. Un tel réseau est représenté figure 17. Si l'on veut limiter le nombre de messages que l'on peut stocker en attente dans le tampon à 3, il suffit de mettre initialement 3 jetons dans la place p_{b0} (voir la figure18). Le nombre de jetons mis initialement dans la place p_{b0} définit la capacité du tampon (la nombre maximal de messages que peut contenir la boîte aux lettres).

6 Définition informelle des réseaux de Petri

6.1 Définition informelle des concepts

Récapitulons ce que nous avons vu concernant les réseau de Petri.

Tout d'abord un réseau de Petri peut être vu comme étant un *ensemble d'automates finis communicants.* Cette vision doit être tempérée par deux remarques. La première remarque est que les réseaux de Petri sont plus généraux que les automates finis puisque, comme nous l'avons vu, des piles impliquant un nombre infini d'états peuvent être représentées. La deuxième remarque c'est que, par contre, les communications prises en compte sont *statiques*, c'est-à-dire qu'elles ne peuvent pas dépendre de données internes aux processus représentés par les automates [11].

Un réseau de Petri est d'abord un *graphe orienté* ayant *deux types de nœuds* : des *places* représentées par des *cercles* et des *transitions* représentées par des *rectangles.* Les arcs du graphe ne peuvent relier que des places vers des transitions, ou des transitions vers des places (par d'arcs entre places ni entre transitions).

Un réseau de Petri décrit un *système dynamique à événements discrets.* Les places permettent la description des états (qui sont donc discrets) et les transitions permettent la description des événements (changements d'état).

Un réseau de Petri est un graphe muni d'une *sémantique opérationnelle*, c'est-à-dire qu'un comportement est associé au graphe, ce qui permet de décrire la dynamique du système représenté. Pour cela un troisième élément est ajouté aux places et aux transitions : *les jetons.*

Une répartition des jetons dans les places à un instant donné est appelée le *marquage* du réseau de Petri. Un marquage donne *l'état* du système. Le nombre de jetons contenus dans une place est un *entier positif ou nul.* Il ne peut pas être négatif.

Pour un marquage donné, une transition peut être *sensibilisée* ou non. Une transition est sensibilisée si chacune de ses places d'entrée contient au moins un jeton [12]. L'ensemble des transitions sensibilisées pour un marquage donné défini l'ensemble des changements d'états possibles du système depuis l'état correspondant à ce marquage. C'est un moyen de définir l'ensemble des événements auxquels ce système est *réceptif* dans cet état.

Quand une transition est sensibilisée, elle peut être franchie. Le *franchissement* d'une transition se fait en enlevant un jeton à chacune des places d'entrée de la transition et en ajoutant un jeton à chacune des places de sortie de cette même transition [13]. Le franchissement d'une

11. Cette remarque deviendra plus claire après l'étude du chapitre concernant les données et le temps.

12. Nous verrons lors de la définition formelle que des poids peuvent être associés aux arcs ce qui impose alors la présence de plusieurs jetons dans une place donnée.

13. Les poids associés aux arcs permettent d'enlever et d'ajouter plusieurs jetons à une place donnée.

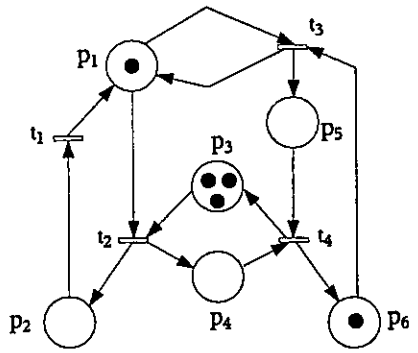

FIG. 19 – *Exemple de Réseau de Petri*

transition décrit le changement d'état du système à événements discrets lors de l'occurrence de *l'événement* associé à la transition.

6.2 Exemple

Considérons le réseau de Petri de la figure 19 à titre d'exemple. Il est formé des 6 places $\{ p_1, p_2, p_3, p_4, p_5, p_6 \}$ et des 4 transitions $\{ t_1, t_2, t_3, t_4 \}$. On remarque qu'il est tout à fait possible d'avoir entre une place et une transition un arc dans chaque sens. C'est le cas entre la place p_1 et la transition t_3. Cela forme ce que l'on appelle, dans le cas des réseaux de Petri, une *boucle élémentaire*[14]. Par contre il n'y a pas d'arc entre deux places ni entre deux transitions.

Le marquage initial est défini par la répartition suivante des jetons :

- un jeton dans la place p_1,
- trois jetons dans la place p_3,
- un jeton dans la place p_6,
- aucun jeton dans les autres places.

Ce marquage définit l'état initial du système.

Pour ce marquage, seules les transitions t_2 et t_3 sont sensibilisées. En effet, pour t_2, les places p_1 et p_3 contiennent au moins un jeton et pour t_3, c'est le cas des places p_1 et p_6. Les deux autres transitions ont leurs places d'entrée vides.

Choisissons de franchir la transition t_2. Le tir de cette transition consiste à diminuer d'une unité le contenu en jetons des places p_1 et p_3 et à incrémenter le contenu des places p_2 et p_4. Le marquage (le nouvel état) atteint est représenté figure20.

Pour ce marquage, seule la transition t_1 est franchissable. En effet, si la transition t_3 a encore une place d'entrée contenant au moins un jeton (la place p_6), il n'en est plus de même pour

14. En théorie des graphes une boucle élémentaire est un arc orienté dont l'origine et la destination sont confondues, l'arc reboucle sur le même nœud. Dans un réseau de Petri c'est impossible car un arc relie une place à une transition ou une transition à une place. C'est pourquoi la notion de boucle élémentaire est différente

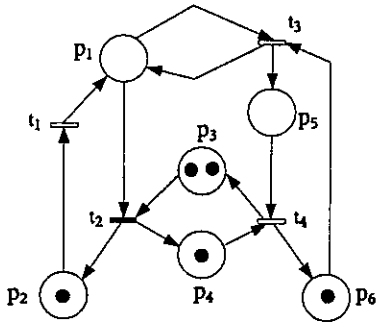

FIG. 20 – *Après le tir de t_2*

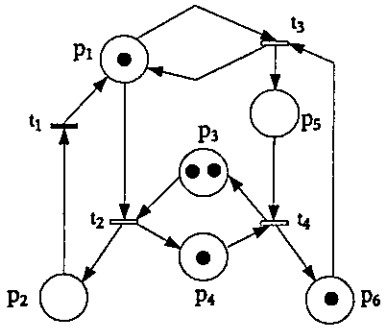

FIG. 21 – *Après le tir de t_2 puis t_1*

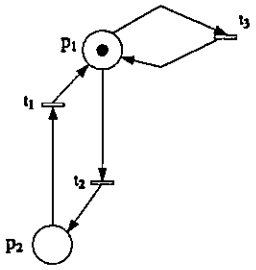

FIG. 22 – *Sous-système 1*

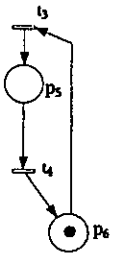

FIG. 23 – *Sous-système 2*

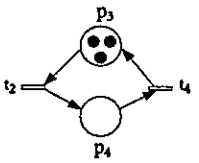

FIG. 24 – *Tampon*

l'autre place (p_1) qui est maintenant vide. Une place d'entrée vide suffit à rendre non sensibilisée une transition. Franchir une transition non sensibilisée entraînerait un *nombre négatif* de jetons dans la (ou les) places d'entrée vides, ce qui est interdit. Comme p_1 est vide, la transition t_2 est également non sensibilisée.

Franchissons alors la transition t_1. Le marquage atteint est alors celui de la figure21. Si l'on recommence le franchissement de la *séquence* (t_2 ; t_1), on aura alors 2 jetons dans la place p_4 et un seul jeton dans la place p_3. Si l'on recommence encore une fois, il y aura 3 jetons dans p_4 et la place p_3 sera vide. Alors la seule transition sensibilisée sera t_3 (t_2 ne le sera plus) et la séquence (t_3 ; t_4) permettra de décrémenter le contenu en jeton de p_4 et d'incrémenter celui de p_3.

Cet exemple de réseau de Petri peut être vu comme la composition de l'automate fini (représenté sous la forme d'un réseau de Petri) de la figure 22 avec celui de la figure 23. Ces deux automates ont une communication synchrone (la transition t_3) et une communication asynchrone via le tampon (*"buffer"*) de la figure 24. Ce tampon permet l'accumulation d'au plus trois messages.

Références

[GI 73] P. Girard, P. Naslin : Construction de machines séquentielles Editions Dunod 1973

[PE 62] C.A. Petri : Communication with automata, Supplement 1 to technical report RADC-TR-65-377, Vol 1, New-York 1966, Translated from "Kommunikation mit Automaten" PhD Bonn 1962

Exemple du photographe – Pas 1
Le réseau de Petri vu comme un ensemble d'automates communicants

Robert Valette

LAAS-CNRS Toulouse,

version d'avril 2007

Pré-requis : Ce texte illustre le chapitre 1 *Introduction aux réseaux de Petri*. Il suppose une connaissance au moins superficielle de UML. Il aboutit à la construction d'un réseau de Petri qui sera repris par la suite pour illustrer d'autres chapitres.

Table des matières

1 Système à représenter **25**

1.1 Description générale 25

1.2 Les processus associés à la station graphique 26

1.2.1 Processus associé au cas d'utilisation « Cas 1 » : 26

1.2.2 Processus associé au cas d'utilisation « Cas 2 » : 26

1.3 Les processus associés à la station d'impression 26

1.3.1 Processus associé au cas d'utilisation « Cas 1 » : 26

1.3.2 Processus associé au cas d'utilisation « Cas 2 » : 27

2 Description des interactions **27**

2.1 Diagrammes de séquence 29

2.2 Diagrammes « *états-transitions* » 29

2.3 Discussion 29

3 Modélisation par un réseau de Petri **30**

3.1 Réseau de Pétri décrivant la station graphique 30

3.2 Réseau de Pétri décrivant la station d'impression 30

3.3 Réseau de Pétri global 32

4 Discussion **36**

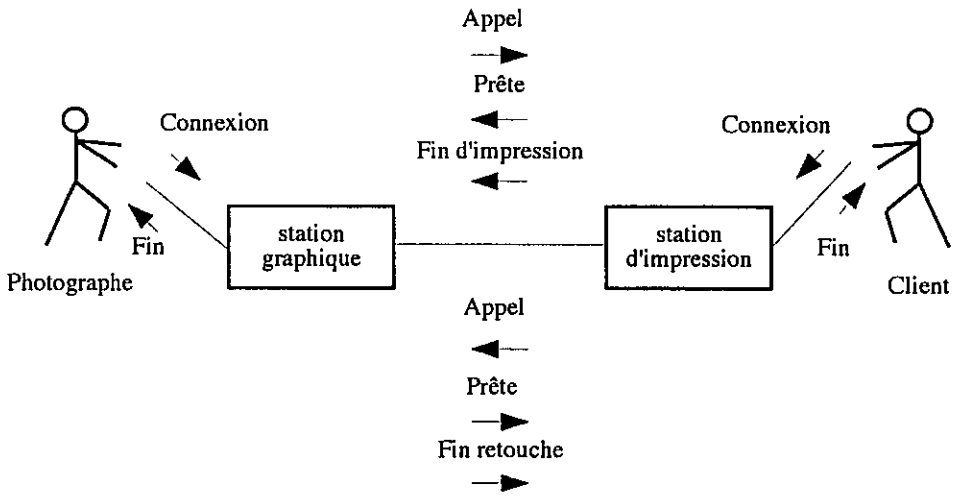

FIG. 1 – Diagramme de collaboration

1 Système à représenter

1.1 Description générale

On cherche à représenter et analyser le fonctionnement et les interactions entre les divers équipements d'un photographe. Ce photographe dispose d'une *station d'impression* qui peut fonctionner en mode automatique pour les clients. Ceux-ci chargent eux mêmes leurs photos et les font imprimer.

Le photographe dispose également d'une *station graphique* qui lui permet la retouche manuelle des photos numériques. Il souhaite *l'interconnecter* avec la station d'impression pour d'une part pouvoir imprimer immédiatement les photos retouchées manuellement, et d'autre part permettre à un client d'exécuter une fonction de retouche automatique juste avant l'impression.

Il s'agit d'un exemple simple (*"toy example"* en Anglais) pour illustrer le fait qu'un réseau de Petri permet de décrire un ensemble d'automates communicants. Il s'agit donc d'une description très simplifiée au réalisme limité. De plus le système est volontairement erroné afin d'illustrer les problèmes pouvant apparaître dans les systèmes à événements discrets. Cet exemple sera ainsi à nouveau utilisé pour illustrer le chapitre d'analyse des propriétés d'un réseau de Petri.

En s'inspirant d'UML, la vision globale du système est donnée par le diagramme de collaboration de la figure 1. Il y a deux acteurs extérieurs, le *photographe* et le *client*, et deux objets, la *station graphique* et la *station d'impression*. L'acteur *photographe* peut se *connecter* sur la *station graphique* et l'acteur *client* peut se *connecter* sur la *station graphique*. Chacun des acteurs est à l'origine d'un cas d'utilisation :

- *Cas 1 :* une retouche manuelle puis une impression de la part du photographe,
- *Cas 2 :* un chargement de la photo sur la station d'impression, une retouche automatique puis une impression de la part du client.

La *station graphique* peut envoyer trois types de messages à la *station d'impression* :

- un *appel* pour obtenir un service d'impression,
- un signalement qu'elle est *prête* à faire une retouche automatique,

– un signalement de *fin de retouche.*

La *station d'impression* peut envoyer trois types de messages à la *station graphique* :

– un *appel* pour obtenir un service de retouche automatique,
– un signalement qu'elle est *prête* à faire une impression,
– un signalement de *fin d'impression.*

Décrivons maintenant un peu plus en détail les processus que devront exécuter chacun des objets.

1.2 Les processus associés à la station graphique

1.2.1 Processus associé au cas d'utilisation "Cas 1" :

Ce processus correspond aux activités de la station graphique quand le photographe retouche et imprime une photo.

Événement : Le photographe se *connecte* sur la station graphique de retouche.

Activité : Le photographe retouche manuellement la photo.

Événement : *Appel* à la station d'impression.

Activité : Attente.

Événement : La station d'impression est *prête* et disponible.

Activité : Transfert des données vers la station d'impression et attente.

Événement : *Fin d'impression "photographe"* (sur demande de la station graphique)

1.2.2 Processus associé au cas d'utilisation "Cas 2" :

Ce processus correspond aux activités de la station graphique quand un client imprime une photo.

Événement : Réception d'un *appel depuis la station d'impression* (demande de retouche automatique).

Activité : Retouche automatique de l'image.

Événement : *Fin de la retouche* automatique.

1.3 Les processus associés à la station d'impression

1.3.1 Processus associé au cas d'utilisation "Cas 1" :

Ce processus correspond aux activités de la station d'impression quand un client imprime une photo.

Événement : Un client se *connecte* sur la station d'impression.

Activité : Initialisation de la station d'impression et chargement de la photo à partir de la carte mémoire du client.

Événement : *Appel* à la station graphique pour une retouche automatique.

Activité : Attente.

Événement : La station graphique est *prête* pour la retouche automatique.

Activité : Transfert de données pour la retouche, attente et retour des données

Événement : *Fin de retouche* automatique.

Activité : Impression de la photo du client.

Événement : Fin de l'impression du client.

1.3.2 Processus associé au cas d'utilisation "Cas 2" :

Ce processus correspond aux activités de la station d'impression quand le photographe imprime une photo après retouche manuelle.

Événement : Réception d'un appel d'impression depuis la station graphique.

Activité : Réception des données et impression.

Événement : Fin d'impression sur demande de la station graphique

2 Description des interactions

2.1 Diagrammes de séquence

Nous allons maintenant préciser les interactions en donnant un diagramme de séquence pour chaque cas d'utilisation.

On peut commencer par construire les diagrammes de séquence pour chaque cas d'utilisation. Un tel diagramme décrit l'enchaînement des événements et des activités pour chaque objet pour un cas d'utilisation donné. Nous avons ici deux cas d'utilisation, nous obtenons les diagrammes des figures 2 et 3.

Chacun comprend deux lignes de temps verticales, une pour la station d'impression et une pour la station graphique. Les rectangles gris décrivent les activités et les flèches sont associées aux communication entre les objets et entre les objets et les acteurs extérieurs (ici le photographe et le client).

On remarque que les deux communications associées aux appels sont des communications asynchrones puisque l'on ne se préoccupe pas de l'état du destinataire quand la communication est effectuée, et que l'on prévoit une attente pour la réponse (objet prêt). Par contre les 4 autres sont plutôt des communication synchrones car elles correspondent à des phases d'interaction fortes (échanges de données).

Comme tout *semble* normal, nous pouvons énumérer les états des objets sous la forme d'automates finis en les annotant pour marquer les communications.

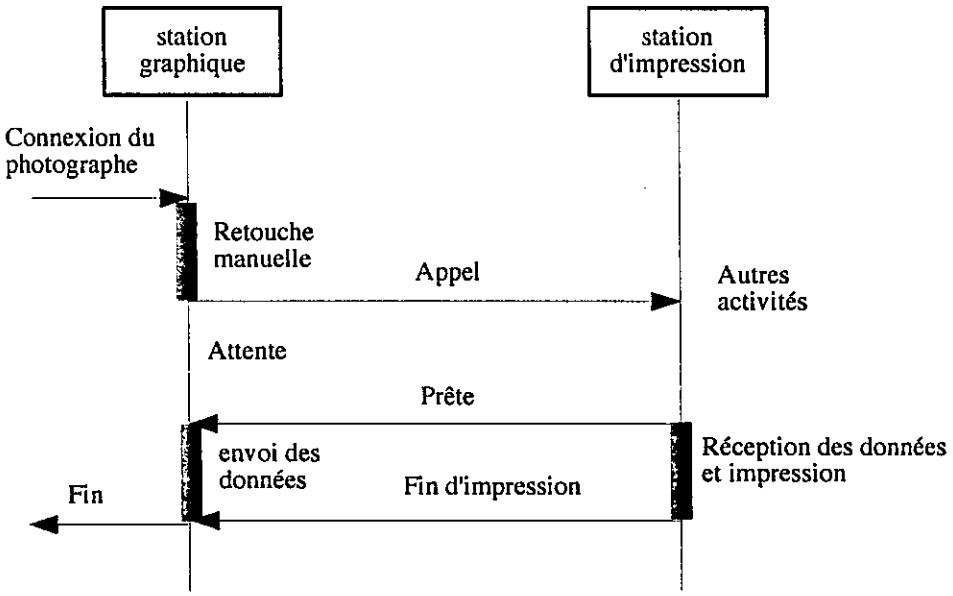

FIG. 2 – Diagramme de séquence pour le "cas 1"

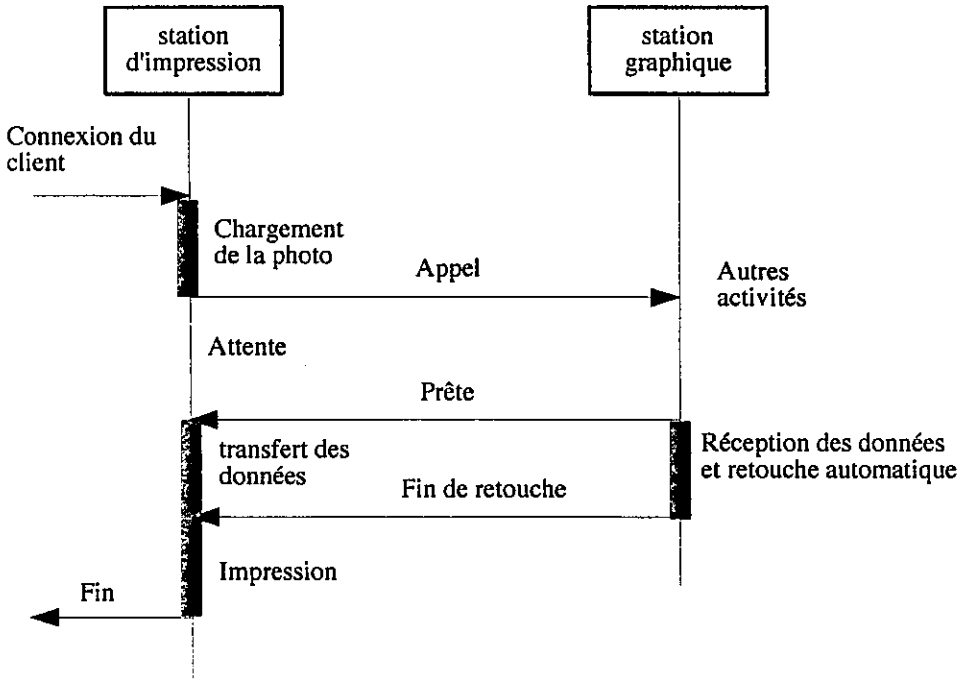

FIG. 3 – Diagramme de séquence pour le "cas 2"

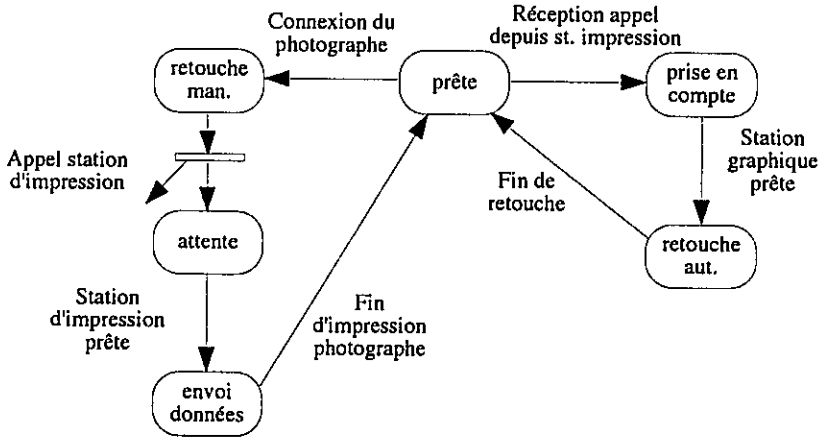

FIG. 4 – Diagramme d'états pour la station graphique

2.2 Diagrammes *"états-transitions"*

Commençons par la station graphique. Nous obtenons le diagramme des états te des transitions de la figure 4. Il s'agit d'un automate fini ayant 6 états. Pour que la figure reste claire, on ne fait figurer que les événements provoquant un changement d'état, c'est-à-dire auxquels l'automate est *réceptif*. Les états sont représentés par des ovales. Le rectangle représente une communication asynchrone correspondant à un appel vers l'autre objet du système : la station d'impression.

L'automate fini de la station d'impression est représenté figure 5. Il comprend 7 états et une communication asynchrone correspondant à un appel vers la station graphique. La réception de la communication asynchrone correspondant à la demande de service de la station graphique est prise en compte sous la forme du changement d'état associé. Le tampon (ou boîte-aux-lettres, ou *"buffer"* associé à cet échange n'est pas explicité.

Les communications synchrones ne sont spécifiées que par le fait que le même événement apparaît dans les deux automates finis, celui de la station graphique et celui de la station d'impression.

2.3 Discussion

La technique de description du système que nous venons d'utiliser est fortement inspirée du langage UML. Les diverses vues du systèmes obtenues donnent l'impression que le système fonctionne correctement. Mais l'interaction entre les deux objets n'apparaît pas clairement et est représentée à l'aide d'annotations associées aux automates finis que sont en fait les diagrammes d'états quand on supprime toutes les annotations. Nous allons maintenant décrire le même système à l'aide d'un réseau de Petri pour bien mettre en évidence le fait qu'il s'agit bien d'un ensemble d'automates communicants et que les communications sont représentées par les mêmes primitives graphiques que les changements d'états internes.

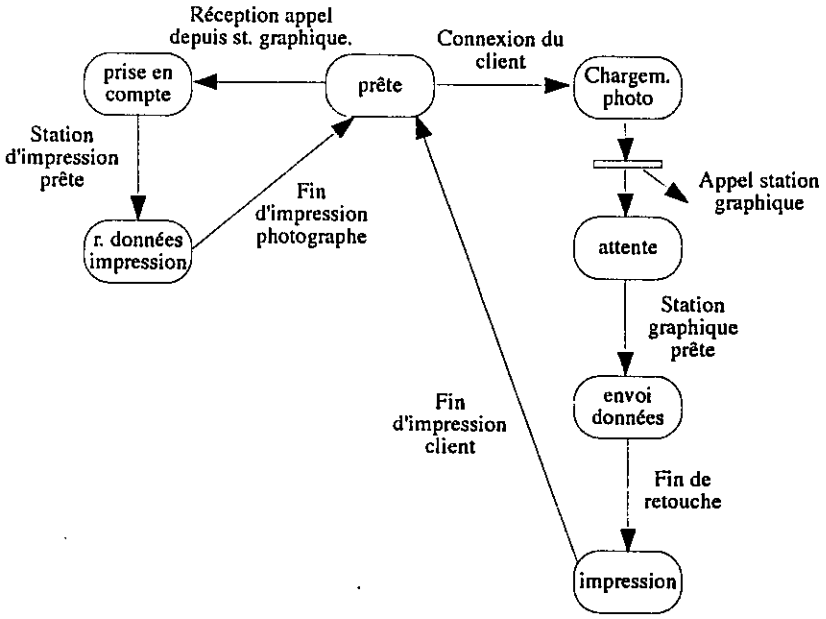

FIG. 5 – Diagramme d'états pour la station d'impression

3 Modélisation par un réseau de Petri

3.1 Réseau de Petri décrivant la station graphique

Commençons par la construction du réseau de Petri décrivant le comportement de la station graphique. Il suffit de partir de l'automate fini de la figure 4 (le diagramme "états-transitions"). Les états sont représentés par des places (des cercles). L'état initial, c'est-à-dire l'état marqué *"prête"*, comprend initialement un jeton. Les changements d'états sont représentés par des transitions (les rectangles). Pour expliciter les deux communications asynchrones et préparer la construction du réseau de Petri global, nous avons représentés les deux tampons jouant le rôle de boîtes-aux-lettres.

Nous obtenons le réseau de Petri de la figure 6. La signification des places et des transitions (leur *interprétation*, ou bien encore leur *sémantique*) est donnée par la table 3.1.

3.2 Réseau de Petri décrivant la station d'impression

On procède de la même façon pour la station d'impression à partir de l'automate fini exprimé sous la forme du diagramme d'états et de transitions de la figure 5. Les états sont transformés en places et les changements d'états en transitions.

Toujours comme précédemment, les tampons associés aux deux communications asynchrones sont représentés par des places. Nous obtenons le réseau de Petri de la figure 7. La signification des places et des transitions est donnée par la table 3.2.

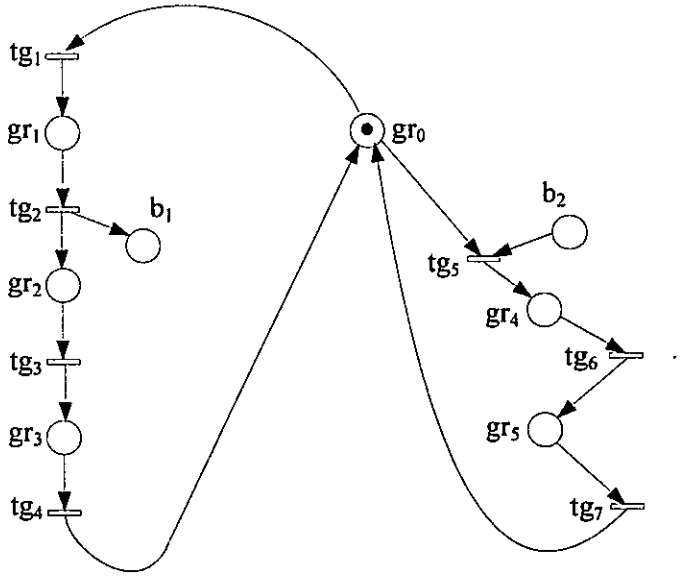

FIG. 6 – Réseau de Petri pour la station graphique

	Etat (marquage) initial
gr_0	*prête*
	Cas d'utilisation "Cas 1"
tg_1	*Connexion du photographe*
gr_1	*retouche manuelle*
tg_2	*Appel station d'impression*
gr_2	*attente*
tg_3	*La station d'impression est prête*
gr_3	*envoi des données*
tg_4	*Fin d'impression "photographe"*
	Cas d'utilisation "Cas 2"
tg_5	*Réception de l'appel depuis la station d'impression*
gr_4	*prise en compte de cet appel*
tg_6	*La station graphique est prête*
gr_5	*retouche automatique*
tg_7	*Fin de la retouche automatique*
	Tampons pour les communications asynchrones
b_1	*appel de la station d'impression*
b_2	*appel par la station d'impression*

TAB. 1 – Interprétation du réseau de Petri de la figure 6

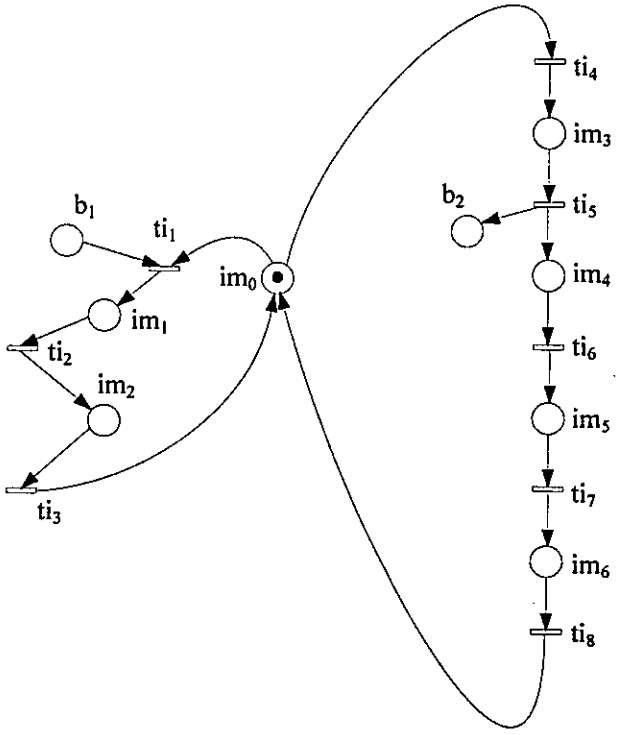

FIG. 7 – Réseau de Petri pour la station d'impression

3.3 Réseau de Petri global

Le réseau de Petri global décrivant à la fois le comportement interne des deux sous-systèmes (la station graphique et la station d'impression) et leur interaction est obtenu en fusionnant les réseaux de Petri des figures 6 et 7.

Les communications asynchrones sont prise en compte en confondant simplement les places correspondant aux tampons, c'est-à-dire les places b_1 et b_2.

En ce qui concerne les communications synchrones, il faut examiner les tableaux 3.1 et 3.2 pour trouver les paires de transitions qui correspondent en fait à un unique événement, vu à la fois du côté de la station graphique et du côté de la station d'impression. Par exemple les transitions tg_3 et ti_2 correspondent toutes les deux à l'occurrence de l'événement : *La station d'impression est prête*. Ces paires de transitions qui doivent fusionner sont données par le tableau 3.3.

En juxtaposant simplement les deux réseaux de Petri des figures 6 et 7, on obtient le graphe de la figure 8. Les deux communications asynchrones sont bien visibles car elles sont représentées par des places en gras (les tampons). Les quatre communications synchrones se remarquent par le fait qu'elles correspondent à deux transitions côte à côte.

En rendant les fusions effectives, nous obtenons alors le réseau de Petri global du système de la figure 9.

	Etat (marquage) initial
im_0	*prête*
	Cas d'utilisation "Cas 1"
ti_1	*Réception de l'appel depuis la station graphique*
im_1	*prise en compte de cet appel*
ti_2	*La station d'impression est prête*
im_2	*réception des données et impression*
ti_3	*Fin d'impression "photographe"*
	Cas d'utilisation "Cas 2"
ti_4	*Connexion du client*
im_3	*chargement de la photo*
ti_5	*Appel de la station graphique*
im_4	*attente*
ti_6	*La station graphique est prête*
im_5	*envoi et réception des données*
ti_7	*Fin de la retouche automatique*
im_6	*impression*
ti_8	*Fin d'impression "client"*
	Tampons pour les communications asynchrones
b_1	*appel par la station graphique*
b_2	*appel de la station graphique*

TAB. 2 – Interprétation du réseau de Petri de la figure 7

transition sta. graph.	transition sta. imp.	Événement associé	transition globale
tg_3	ti_2	*La station d'impression est prête*	t_{32}
tg_4	ti_3	*Fin d'impression "photographe"*	t_{43}
tg_6	ti_6	*La station graphique est prête*	t_{66}
tg_7	ti_7	*Fin de la retouche automatique*	t_{77}

TAB. 3 – Tableau des communications synchrones

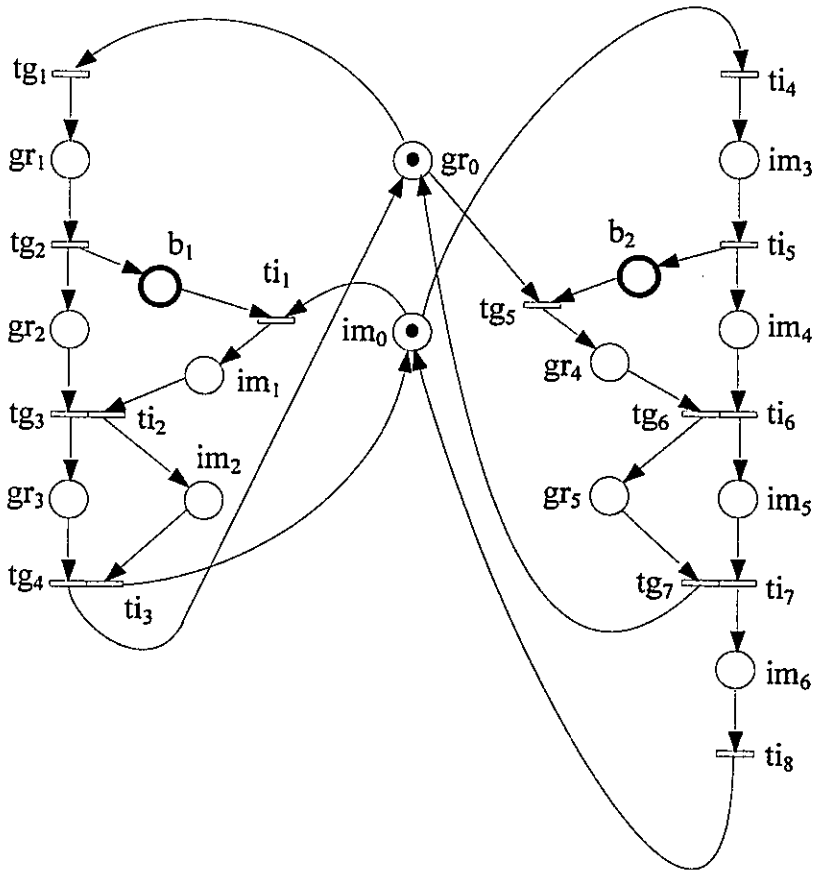

FIG. 8 – Les deux réseaux de Petri connectés

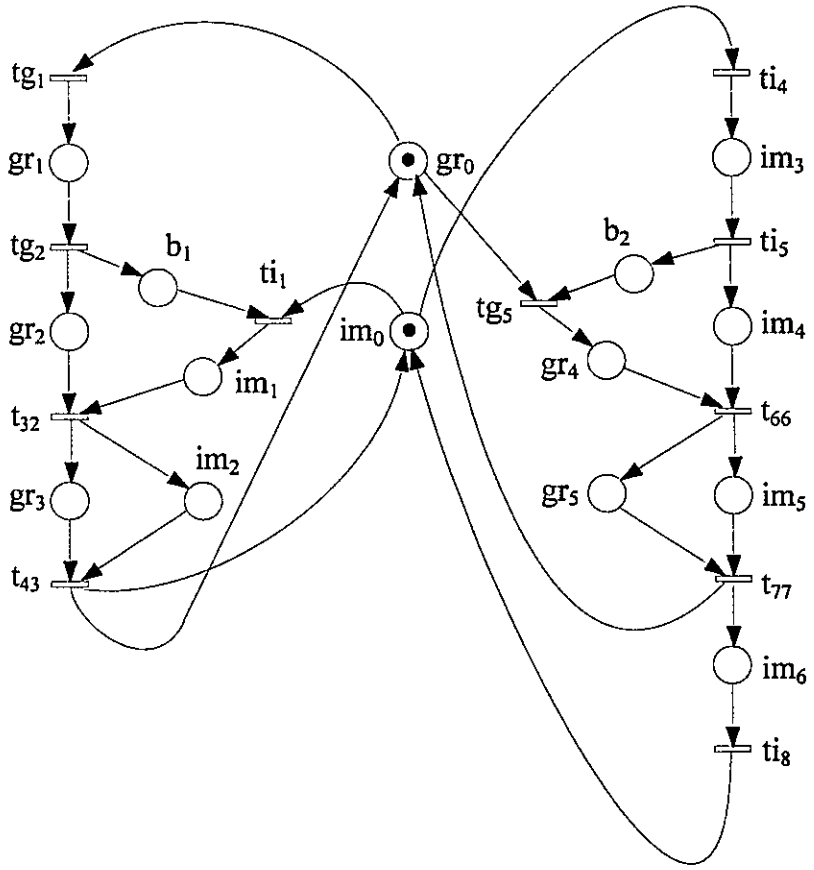

FIG. 9 – Le réseau de Petri global

4 Discussion

La représentation de la figure 8 illustre le fait qu'un réseau de Petri peut être vu comme un ensemble d'automates interconnectés par des communications asynchrones (via des tampons) et des communications synchrones.

On peut remarquer que le réseau de Petri du système (figure 9) est identique et donc que le comportement de chacun des éléments du sous-système peut être obtenu en effaçant simplement les éléments graphiques qui ne le concernent pas.

Ainsi, si on efface toutes les places im_i (station d'impression) et b_i (tampons de communication asynchone) avec les arcs entrants et sortants associés, ainsi que toutes les transitions ti_i, on obtient un réseau de Petri dont la structure est identique à l'automate fini décrivant le comportement de la station graphique.

Inversement, si on efface toutes les places gr_i (station graphique) et b_i avec les arcs entrants et sortants associés, ainsi que toutes les transitions tg_i, on obtient un réseau de Petri dont la structure est identique à l'automate fini décrivant le comportement de la station d'impression.

Le réseau de Petri conserve donc bien directement la représentation des comportements des sous-systèmes composant le système. Si on construit l'automate résultant du produit de deux automates, on ne conserve absolument pas les graphes des automates de départ. Cela vient du fait que dans un automate, l'état est décrit par *un seul nœud*, donc les sous-systèmes sont intimement mélangés. Dans un réseau de Petri, un état est un marquage et un marquage implique en général plusieurs places, et donc *plusieurs nœuds* du graphe. On peut donc conserver la structure des représentations des sous-systèmes. Un réseau de Petri permet de décrire le *comportement global* d'un système tout en *préservant la représentation de son architecture en sous-systèmes.*

Les communications asynchrones sont représentées par des places et les communications synchrones par des transitions. Les communications entre les sous-systèmes sont donc bien représentées à l'aide des mêmes primitives que les états et les changements d'états à l'intérieur des sous-systèmes.

VOUZZAVEDIBISAR

Petri Nets 2009. Call for Papers and Announcement

Jean - Baptiste Voron

30th INTERNATIONAL CONFERENCE ON APPLICATION AND THEORY OF PETRI NETS AND OTHER MODELS OF CONCURRENCY

Paris, France, June 22-26, 2009

Additional information about the conference will be published via
http://PetriNet2009.lip6.fr

Contact e-mail: PetriNet2009@lip6.fr

In 2009 the Petri Nets conference is co-located with the RSP'09.

Important Dates:

Submission of Papers:	January 5, 2009
Notification:	March 1, 2009
Final Version Due:	April 1, 2009
Workshops & Tutorials:	June 22-23, 2009
Conference:	June 24-26, 2009

The deadline for submission of papers is STRICT.
However, if you submit the title page by Jan. 5
it is sufficient to submit the full paper by Jan. 10.

Some of the best papers accepted for the conference will be invited as submission to a special issue of the *Fundamenta Informaticae* journal.

The 30th annual international Petri Net conference and tutorials will be organised by the Université Pierre & Marie Curie together with CNAM, Université Evry-Val-d'Essone, Université Paris-Dauphine, Université Paris 12, Université Paris 13 and Telecom ParisTech (members of the MeFoSyLoMa group). Papers presenting original contributions in any area of application and theory of Petri nets are sought. The language of the conference is English.