LORIC un simulateur réseau de Petri ecrit en Maclisp

V.I. Leopoulos

INRIA
CENTRE DE ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél (3) 954 9020

Rapports de Recherche

N° 371

LORIC
UN SIMULATEUR RÉSEAU DE PETRI ÉCRIT EN MACLISP

Vrassidas-Ioannis LEOPOULOS

Mars 1985

LORIC

Un simulateur Réseaux de Petri écrit en Maclisp

Vrassidas-Ioannis LEOPOULOS

PROJET SAGEP

Resumé

Cette communication est consacrée à la présentation du logiciel de simulation LORIC de R.d.P. élaboré à l'INRIA et écrit en langage Maclisp.

Abstract

This paper is devoted to the presentation of the simulation software LORIC concerning the Petri Nets and elaborated at INRIA. It is written in Maclisp language.

INTRODUCTION

Cette publication présente la spécification du noyau d'un simulateur de R.d.P. utilisant le langage Maclisp.

Les R.d.P. sont utilisés comme un outil de modélisation, d'analyse et de simulation des systèmes informationnels concurrents et plus récemment des systèmes automatisés de production discontinue. Notre simulateur est orienté vers ces derniers systèmes : on s'intéresse à l'aide à la conception, à la gestion et à la simulation des ateliers flexibles.

La communication est divisée en trois parties.

Dans la première partie,on rappelle les différentes classes de R.d.P. : les R.d.P. autonomes (ordinaires, à arcs inhibiteurs, généralisés, à priorités) [MOA 1,2], non autonomes (synchronisés, temporisés synchronisés, interprétés) [MOA 1,2] et colorés [JEN 3].

Dans la deuxième partie ,on donne les caractéristiques principales du langage Maclisp.

Dans la dernière partie,on décrit l'algorithme de simulation de fonctionnement d'un R.d.P. utilisé pour le développement du logiciel et la manière dont le réseau et le marquage initial,ainsi que les conditions vérifiées au début de la simulation du fonctionnement,sont introduits.

I. LES RESEAUX DE PETRI

I.1.RESEAUX DE PETRI AUTONOMES

I.1.1. LA STRUCTURE D'UN RESEAU DE PETRI ORDINAIRE

La structure d'un réseau de Petri est un quadruplet $R = (P,T,a,b)$ où :

(1) P est un ensemble de places.

(2) T est un ensemble de transitions.

(3) $a \subseteq P \times T$ est une relation d'incidence avant.

(4) $b \subseteq T \times P$ est une relation d'incidence arrière.

Le graphe d'un R.d.P. est la représentation d'une structure de R.d.P. comme un graphe biparti orienté. Les noeuds du graphe sont les cercles qui représentent les places et les rectangles qui représentent les transitions.

Les places et les transitions sont connectées par des arcs directs. Il y a un arc de la place p_r à la transition t_i si et seulement si $(p_r, t_i) \in a$. De même, il y a un arc de la transition t_j à la place p_s si et seulement si $(t_j, p_s) \in b$.

On note encore :

(1) $t \in T$, $^{\cdot}t = \{p \in P \mid (p,t) \in a\}$
$^{\cdot}t$ est l'ensemble des places d'entrée de t.

(2) $t \in T$, $t^{\cdot} = \{p \in P \mid (t,p) \in b\}$
$t^{\cdot}$ est l'ensemble des places de sortie de t.

(3) $p \in P$, $^{\cdot}p = \{t \in T \mid (t,p) \in b\}$
$^{\cdot}p$ est l'ensemble des transitions d'entrée de p.

(4) $p \in P$, $p^{\cdot} = \{t \in T \mid (p,t) \in a\}$
$p^{\cdot}$ est l'ensemble des transitions de sortie de p.

Marquages d'un R.d.P.

Un marquage M d'un RdP R= (P,T,a,b) est une application de P dans $\mathbb{N}$, $\mathbb{N}$ étant l'ensemble des entiers naturels.

Le marquage M peut être défini aussi bien par un n-vecteur :

$$M = \begin{bmatrix} m_1 \\ m_2 \\ \vdots \\ m_n \end{bmatrix} \quad \text{et } m_i = M(p_i)$$

où $n = |P|$ et chaque $m_i \in \mathbb{N}$.

Au niveau du graphe, le marquage correspond à une distribution dans les places d'objets appelés marques. Une marque est représentée par un point ; chaque place p_i aura donc m_i marques.

L'évolution du marquage dans un RdP est contrôlée par le nombre et la distribution des marques.

Le marquage d'un RdP évolue par la mise à feu des transitions.

Pour que la mise à feu d'une transition soit possible il faut que la transition soit validée.

Une transition t de R est dite validée par le marquage M si et seulement si $\forall p \in {}^{\cdot}t$ on a $M(p) \geq 1$, c'est-à-dire si et seulement si toute place d'entrée de t contient au moins une marque.

Si $\mathcal{M}$ est l'ensemble des marquages de R et t une transition de R, la mise à feu d'une transition t est definie comme étant l'opération qui à un marquage $M_i \in \mathcal{M}$ fait correspondre le marquage M_j tel que :

$$\forall p \in P,\ M_j(p) = \begin{cases} M_i(p)-1 & \text{si } p \in {}^{\cdot}t - t^{\cdot} \\ M_i(p)+1 & \text{si } p \in t^{\cdot} - {}^{\cdot}t \\ M_i(p) & \text{sinon.} \end{cases}$$

La mise à feu d'une transition est définie pour les marquages M qui valident t.

I.1.2. RESEAUX DE PETRI GENERALISES (RdPG)

Un RdPG est un doublet $R_G = (R,W)$ où :

et
- R est un RdP (P,T,a,b)
- W est une application de l'ensemble U= a∪b des arcs orientés de R dans $\mathbb{N}-\{0\}$.

Sur le graphe RdPG,on représente cette application soit par l'association à chaque arc $u \in U$ de la valeur W(u) qu'on appelle poids de l'arc, soit par des arcs multiples.

Evolution du marquage.

Une transition t est validée par un marquage M si et seulement si chacune de ses places d'entrée contient un nombre de marques au moins égal au poids de l'arc qui la connecte à t. L'opération mise à feu de t est definie pour M si et seulement si M valide t. L'exécution consiste alors à enlever de chaque place d'entrée p_r de la transition un nombre de marques égal à $W((p_r,t))$ et à déposer dans chaque place de sortie p_s un nombre de marques égal à $W((t,p_s))$.

I.1.3. RESEAUX DE PETRI A PRIORITES

Un RdPP est un doublet $R_p = (R,O)$ tel que :

et $\begin{cases} R \text{ est un RdP } (P,T,a,b) \\ O \text{ est une relation d'ordre partiel sur } T. \end{cases}$

Si $(t,t') \in O$ et $t \neq t'$, on dit que t est plus prioritaire que t'.

Opération sur les marquages.

Une transition t est validée par un marquage M si et seulement si chacune de ses places d'entrée contient au moins une marque. L'opération mise à feu de t est définie pour M si et seulement si M valide t et ne valide aucune transition plus prioritaire. Son éxécution s'effectue comme pour le RdP ordinaire.

I.2. RESEAUX DE PETRI NON AUTONOMES

I.2.1. RESEAUX DE PETRI SYNCHRONISES (RdPS)

Un RdPS, noté R_S est defini par :

(1) un RdP $R = (P,T,a,b)$
(2) un ensemble E d'événements externes contenant un élément e appelé "événement toujours présent".
(3) une application μ de T dans E.

On représente un RdPS par le graphe du RdP associé en portant sur chaque transition t_j le renseignement $< \mu(t_j) >$ précisant l'événement associé à t_j.

Opération sur les marquages.

Soit $X \subseteq E$ un sous-ensemble d'événements.

Une transition t de R_S est dite réceptive à X pour un marquage M si et seulement si t est validée par M et $\mu(t) \in X$.

Soit $T_{X,M} = \{t_1, t_2, \ldots, t_r\}$ l'ensemble des transitions réceptives à X pour M.

On appelle :

'séquence de simulation complète par rapport à X pour le marquage M' toute séquence de simulation σ_c à partir de M qui vérifie les propriétés suivantes :

(a) les transitions de σ_c sont exclusivement des transitions de $T_{X,M}$.

(b) toute transition de $T_{X,M}$ n'apparaît qu'une fois au plus dans σ_c.

(c) toute séquence σ'_c obtenue en permutant les transitions de σ_c est aussi une séquence de simulation à partir de M.

(d) σ_c est maximale , c'est-à-dire qu'il n'existe pas d'autre séquence plus longue qui contienne toutes les transitions de σ_c et qui vérifie les propriétés a,b et c.

On appelle :

' tir sur occurence de X appliqué à M' :

l'exécution à partir de M d'une séquence de simulation complète σ_c.

On note :

$$M \xrightarrow{X/\sigma_c} M'$$

Le marquage M' est dit <u>marquage atteint</u> par suite du tir sur occurence de X appliqué à M selon σ_c.

On appelle :

'tir itéré sur occurence de X appliqué à un marquage stable M' :

une séquence de tirs composée d'un tir sur occurence de X appliqué à M, suivi d'une itération de tirs sur occurence de {e} jusqu'à atteindre un marquage stable M'.

I.2.2. RESEAUX DE PETRI TEMPORISES SYNCHRONISES

On définit un environnement pour un RdPTS par un ensemble d'éléments externes E contenant l'événement toujours présent e et tel que, à tout événement $x \in E$, $x \neq e$, est associée une fonction croissante τ_x de $\mathbb{N}-\{0\} \longrightarrow \mathbb{R}^+-\{0\}$, $\tau_x(i)$ étant l'instant d'occurence de la i-ème occurence de x à partir de l'instant initial "0".

L'ensemble des événements survenant à l'instant τ est égal à :

$$\{x \in E-e \mid \exists i \in \mathbb{N}-\{0\} \text{ et } \tau_x(i) = \tau\} \cup \{e\}$$

Un RdPTS est défini par :

et
- un RdPs $R_S = (R, E, \mu)$
- un ensemble $\nu = \{\nu_p,\ p \in P\}$ d'applications croissantes de $\mathbb{R}^+ \longrightarrow \mathbb{R}^+$,
- où P est l'ensemble des places de R.

On représente un RdPTS par le RdPS associé en spécifiant à côté de chaque place p l'application ν_p.

Opérations sur les marquages.

Dans un RdPTS une marque peut être soit disponible, soit indisponible, selon qu'elle est restée dans la place correspondante le temps nécessaire ou non. Alors à chaque instant τ le marquage M est la somme de deux marquages M_d (disponible) et M_i (indisponible). Une transition est validée par M si et seulement si elle est validée par M_d.

On dira que $\sigma = \{t_1, t_2, \ldots, t_n\}$ est une séquence de simulation instantanée à partir de M à l'instant τ si on peut mettre à feu les transitions $t_1, \ldots, t_n$ à partir de M à l'instant τ.

Les opérations de tir sur occurence d'événements à un instant τ et de tirs itérés sur occurence d'événements à l'instant τ s'effectuent de la même manière que dans un RdPS en utilisant pour chaque tir une séquence de si-

mulation complète instantanée à l'instant τ.

On peut définir un RdP Temporisé (RdPT) comme étant un RdPTS tel que l'ensemble E se réduit à l'évenement toujours présent e.

I.2.3. RESEAUX DE PETRI INTERPRETES

On définis un RdPI par :

(1) un sous-système opératif (V,OP,C) tel que :
- $V = \{v_1, v_2, \ldots, v_m\}$ est un ensemble fini de variables prenant leurs valeurs respectivement dans les domaines $D_1, D_2, \ldots, D_m$.
- $OP = \{OP_1, OP_2, \ldots, OP_n\}$ est un ensemble fini d'opérateurs définis comme des applications internes de $D_1 x D_2 x \ldots x D_m$.
- $C = \{C_1, C_2, \ldots, C_r\}$ est un ensemble de conditions (prédicats) sur les variables de V.

(2) un RdPTS (R,E,μ,v).

(3) une application φ : P $\longrightarrow$ OP, P étant l'ensemble des places de R.

(4) une application ψ : T $\longrightarrow$ C, T étant l'ensemble des transitions de R.

Opérations sur les marquages.

Les opérations sont celles définies pour les RdPTS avec deux particularités :

(a) une transition n'est considérée comme étant réceptive à un ensemble d'événements X à l'insant r que si :
- a1) t est validée par le marquage du RdPTS à cet instant.
- a2) le prédicat $\psi(t)$ qui lui est associé est vérifié.
- a3) $\mu(t) \in X$.

(b) chaque fois qu'une marque dans une place p passe de l'état indisponible à l'état disponible, on effectue la transformation définie par l'opérateur $\psi(p)$.

I.3. RESEAUX DE PETRI COLORES

Pour les RdP colorés, à chaque marque on associe une couleur qui indique son identité. De plus un ensemble de couleurs est associé à chaque place et chaque transition. Chaque transition peut être mise à feu par rapport à chaque couleur qui lui est associée.

Soit A un ensemble non vide et $\mathbb{D} = \mathbb{N}$ ou $\mathbb{Z}$.

On note : $[A \to \mathbb{D}]_f$ l'ensemble des fonctions $g \in [A \to \mathbb{D}]$ tel que :

$\{a \in A \mid g(a) \neq 0\}$ soit fini.

Si A est fini on a $[A \longrightarrow \mathbb{D}]_f = [A \longrightarrow \mathbb{D}]$.

Un réseau de Petri coloré est un doublet RdPC= (R,C).

R est un RdP (P,T,a,b),

où a et b sont des relations d'incidence avant et arrière :

et C la fonction "couleur" définie de PUT dans des ensembles non vides.

$$\left.\begin{matrix} a(p,t) \\ b(t,p) \end{matrix}\right\} \in [C(t) \longrightarrow [C(p) \to \mathbb{Z}]_f] \quad \forall (p,t) \in P \times T.$$

Marquage d'un RdPC.

Un marquage d'un RdPC est une application :

$$m(p) \in [C(p) \longrightarrow \mathbb{N}]_f \quad \forall p \in P.$$

Si p est une place et t une transition alors C(p) et C(t) sont appelés couleurs.

II. LE LANGAGE MACLISP

Maclisp est un dialecte de Lisp développé dans le projet MAC par le laboratoire d'intelligence artificielle du M.I.T.

Comme son nom (List Processing) l'indique, LISP est un langage construit de manière à faciliter la manipulation des listes. Grâce à une série de fonctions prévues à cet effet, on peut facilement construire une liste à partir d'éléments, ajouter des éléments au fur et à mesure des besoins, vérifier l'appartenance d'un élément à la liste, remplacer un élément par un autre, ou changer l'ordre des éléments dans la liste.

Les éléments d'une liste peuvent être soit des nombres, des combinaisons de nombres et de lettres,ou des combinaisons de lettres, soit des listes plus petites dont la concatenation donne la liste finale.

Notre réseau est donc représenté par une liste dont les éléments sont des listes, chacune représentant une transition. Les éléments de cette liste contiennent toutes les informations nécessaires à la détermination d'une transition.

Le marquage est présenté sous la forme d'une liste particulière, appelée "liste d'association" qui se schématise ainsi :

((indicateur 1 . valeur 1)(indicateur 2 . valeur 2)...)

où (indicateur 1) peut être le nom de la place et (indicateur 2) le nombre de jetons d'une certaine couleur contenus dans la place.

Il faut encore souligner que, en LISP, on n'est pas obligé de déclarer la taille de la liste qu'on va utiliser. On peut alors introduire n'importe quel nombre de modules (dans les limites de la capacité de traitement de notre ordinateur) contenant n'importe quel nombre de transitions qui ont n'importe quel nombre de places d'entrée et de sortie, et qui peuvent être mises à feu par rapport à n'importe quel nombre de couleurs (on parlera aussi de manières.

Si la forme de listes est très pratique pour la représentation intérieure d'un réseau de Petri et de l'évolution du marquage, elle est difficilement lisible par l'utilisateur non averti.

III. PRESENTATION DU LOGICIEL

III.1. STRUCTURE DU LOGICIEL

Le logiciel que nous proposerons permet de constituer un réseau de Petri à partir de modules créés antérieurement, stockés sur disque et repérés par un mot clef affecté au moment de la génération du module.

Chaque module est un ensemble de transitions.

Chaque transition est renseignée comme indiqué ci-dessous :

$$t, L_1, F_1, L_2, F_2, T_2, H, C, E$$

où :

t est le nom de la transition.

L_1 est la liste des places d'entrée de la transition.

F_1 est la liste des relations d'incidence avant.

L_2 est la liste des places de sortie de la transition.

F_2 est la liste des relations d'incidence arrière.

T_2 est la liste des temps de séjour des jetons dans les places de sortie.

H est la liste des places d'où sont issus des arcs inhibiteurs qui aboutissent à la transition.

C est la liste des conditions nécessaires pour que la transition soit sensibilisée.

E est la liste des événements auxquels la transition est réceptive.

Chaque transition peut être mise à feu de plusieurs manières, décrite dans F_2.

Ces manières représentent une réalité : si, par exemple, on veut modéliser une machine capable d'effectuer plusieurs opérations différentes, alors chaque manière de mettre une transition à feu correspond à une opération.

A chaque manière de mettre une transition à feu correspond, évidemment, une manière de la valider, décrite dans F_1.

Dans notre exemple chaque opération différente que peut effectuer la machine nécessite éventuellement des outils et des matières premières spécifiques à cette opération.

Une condition nécessaire pour qu'une transition soit validée d'une certaine manière est la présence d'un nombre minimal de jetons disponibles dans chacune des places d'entrée.

On peut utiliser des jetons de couleurs différentes et, dans ce cas, la validation d'une transition peut exiger la présence simultanée dans chaque place d'entrée d'un nombre minimal de jetons disponibles de chaque couleur.

Considérons par exemple une machine qui assemble 2 pièces de type A et une pièce de type B pour donner un produit C.

La transition qui représente la fin de l'assemblage est validée par la présence de deux jetons de couleur A et d'un jeton de couleur B dans l'unité place d'entrée. La mise à feu provoque la disparition des jetons de la place d'entrée et l'apparition d'un jeton de couleur C dans la place de sortie.

Par conséquent, la liste F_1 prend la forme suivante :

(Manière A de valider la transition
(pour la 1re place d'entrée
(couleur Ac_1^1, nombre de jetons nécessaires An_1^1)
(couleur Ac_2^1, nombre de jetons nécessaires An_2^1)
⋮
(couleur $Ac_{K_1}^1$, nombre de jetons nécessaires $An_{K_1}^1$))
(pour la 2me place d'entrée
(couleur Ac_1^2, nombre de jetons nécessaires An_1^2)
⋮
(couleur $Ac_{K_2}^2$, nombre de jetons nécessaires $An_{K_2}^2$))
⋮
(pour la N_A^{me} place d'entrée
(couleur $Ac_1^{N_A}$, nombre de jetons nécessaires $An_1^{N_A}$)
⋮
(couleur $Ac_{K_{N_A}}^{N_A}$, nombre de jetons nécessaires $An_{K_{N_A}}^{N_A}$))
⋮

⋮

(Manière X de valider la transition

(pour la 1re place d'entrée

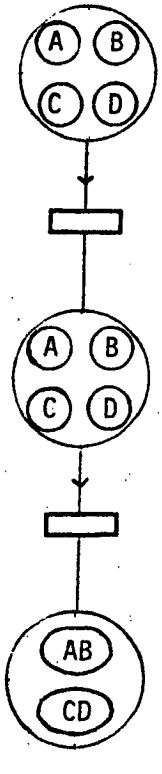
(couleur Xc_1^1, nombre de jetons nécessaires Xn_1^1)

⋮

(couleur $Xc_{K_{N_X}}^{N_X}$, nombre de jetons nécessaires $Xn_{K_{N_X}}^{N_X}$)))

La liste F_2 est du même type. A chaque manière de valider la transition correspond une manière de la mettre à feu. Pour chaque place de sortie de la transition, on connait l'ensemble des couleurs et le nombre de jetons de chaque couleur à déposer dans cette place lorsque la transition est mise à feu.

Représentons par exemple une machine capable d'effectuer simultanément deux opérations.



P_1 : stockage amont

t_1 : entrée dans la machine

P_2 : assemblage

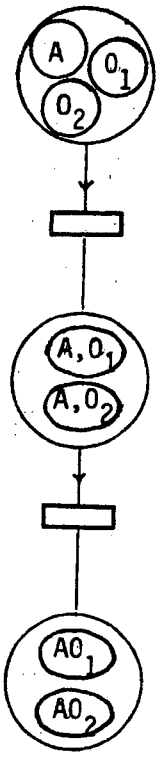
t_2 : sortie machine

P_3 : stockage aval

Si, dans le lieu de stockage amont, se trouvent une pièce A et une pièce B, on déclenche l'entrée dans la machine, l'assemblage et le stockage aval. Il apparaît un produit AB. De même, si les pièces C et D apparaissent dans le stockage amont, elles se sont assemblées et la pièce CD apparaît dans le stockage aval.

Il y a donc deux manières de valider et de mettre à feu les transitions, et ces deux manières peuvent se réaliser simultanément.

L'ordre dans lequel les différentes manières de valider une transition se présentent dans F_1 est important : il détermine l'ordre des mises à feu et entraine des résultats différents en cas de conflit.



P_1 : disponibilité de la matière première A et des outils O_1 et O_2

t_1 : début de fabrication

P_2 : fabrication

t_2 : fin de fabrication

P_3 : stockage du produit (AO_1) ou du produit (AO_2)

Si un seul jeton de couleur A se trouve dans P_1, les outils O_1 et O_2 étant tous deux disponibles, alors un seul des deux produits AO_1 ou AO_2 pourra être obtenu, suivant l'ordre de la mise à feu, déterminé par l'ordre des

manières dans F_1.

Nous avons signalé que notre logiciel permet le stockage de modules, puis leur utilisation.

Supposons qu'un R.d.P. soit en mémoire. Il est possible d'appeler un module pour le compléter.

Le module s'appelle par le mot-clé qui le caractérise. On indique également les places communes entre le module et le réseau. La liaison se fait automatiquement et le système veille à ce que les noms des places et des transitions ne figurent pas en double.

Il est donc possible de construire une bibliothèque de modules, ce qui permet de construire rapidement un réseau.

On peut, en conséquence, remplacer un module par un autre afin de détailler une partie du système étudié.

Considérons par exemple un atelier formé d'un convoyeur et de différentes machines. Les produits passent sur un nombre éventuellement variable de machines et toujours dans le même ordre. Ils sont transportés par le convoyeur. La figure 1 schématise cette situation :

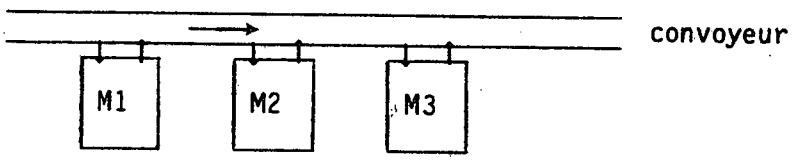


Figure 1

Cet atelier peut être représenté par le réseau donné par la figure 2 :

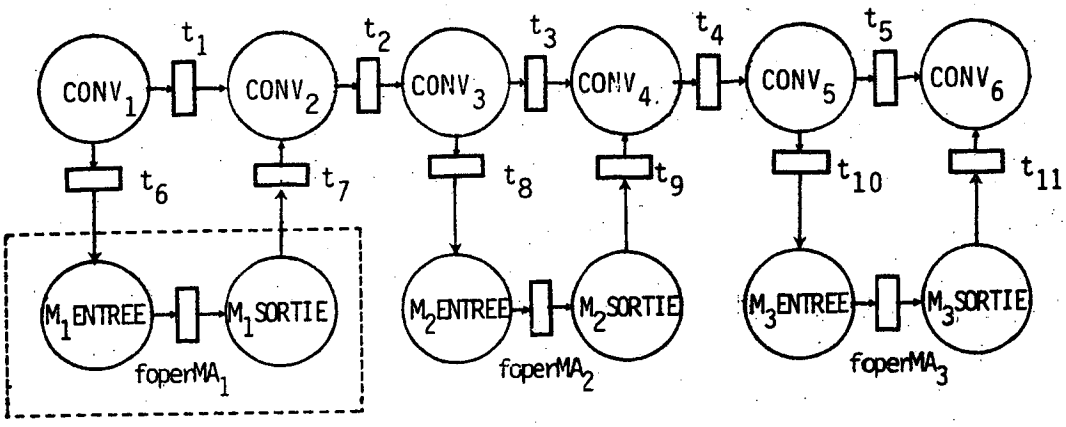


Figure 2

Un produit qui entre dans le système se dirige vers la première machine ou continue sur le convoyeur vers les machines suivantes. Un jeton dans une place M_iENTREE représente un produit qui subit une ou plusieurs opérations dans la machine, un jeton dans une place M_iSORTIE représente un produit en cours de recharge sur le convoyeur.

On peut maintenant remplacer la partie (M_iENTREE, foperMA_i, M_iSORTIE) du réseau par un module plus détaillé qui modélise séparément chaque opération que le produit subit sur une machine.

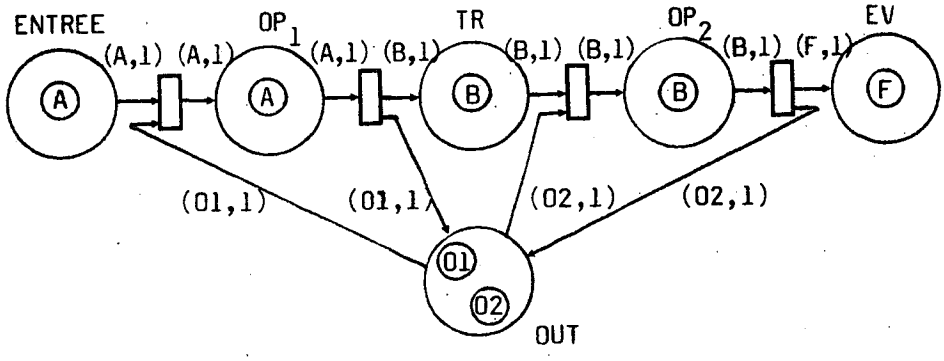


Figure 3

Par exemple, le produit entre dans la machine et est amené devant le premier outil (jeton de couleur A dans la place ENTREE, jeton de couleur O1 dans la place OUT). Ensuite le produit subit la première opération (jeton de couleur A dans la place OP_1) et se transforme en produit intermédiaire qui va être transporté devant le deuxième outil tandis que l'outil n° 1 redevient disponible (jeton de couleur B dans la place TR, jeton de couleur O1 dans la place OUT). Dès que l'outil n° 2 devient disponible (jeton de couleur O2 dans la place OUT) le produit intermédiaire subit la deuxième opération (jeton de couleur B dans la place OP_2) pour se transformer en produit final tandis que l'outil n° 2 redevient disponible (jeton de couleur F dans la place EV, jeton de couleur O2 dans la place OUT). Le produit final va enfin sur le convoyeur pour sortir du système.

Ce module peut être introduit dans le schéma de la figure 2 pour détailler le fonctionnement du système.

La liaison s'effectue en remplaçant les places M_iENTREE et M_iSORTIE du module 1 par les places ENTREE et EV du module 2.

III.2. LE MARQUAGE

Le marquage se présente comme une liste qui contient pour chaque couleur de jeton les places marquées, le nombre de jetons existants et les instants où les jetons deviennent disponibles.

Cette liste a la forme suivante :

Couleur 1 (place 1,1, nombre total de jetons présents $n1,1$)

dont : $(n1,1_{t_1^1}$, disponibles à l'instant $t_1^1)$

$(n1,2_{t_2^1}$, " " $t_2^1)$

:

:

$(n1,1_{t_n^1}$, " " $t_n^1)$

(place 1,2, nombre de jetons présents n1,2)

dont : ($n1,2_{t_1^2}$, disponibles à l'instant t_1^2)

($n1,2_{t_2^2}$, disponibles à l'instant t_2^2)

⋮

($n1,2_{t_n^2}$, " " t_n^2)

⋮

(place 1,K, nombre total de jetons présents n1,K)

⋮

⋮

Couleur N (place N,1, nombre total de jetons présents nN,1)

⋮

(place N,J, nombre total de jetons présents nN,J)

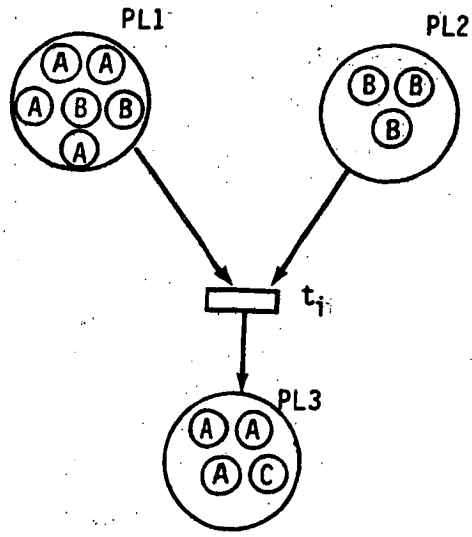
dont : ⋮

($nN,J_{t_1^J}$, disponibles à l'instant t_1^J)

Bien entendu $t_1^1 < t_2^1 < t_3^1 < \ldots < t_n^1$

et $n1,1_{t_1} + n1,1_{t_2} + \ldots + n1,1_{t_n} = n1,1$

Considérons par exemple le réseau suivant :



Le marquage est présenté par la liste suivante :

{Couleur A (PL1,4)(2,t_1)(2,t_2)
(PL3,3)(1,t_3)(1,t_4)(1,t_5)
Couleur B (PL1,2)(2,t_6)
(PL2,3)(1,t_7)(2,t_8)
Couleur C (PL3,1)(1,t_9)}

III.3. L'ALGORITHME

Si les événements extérieurs peuvent arriver à n'importe quel instant non prévu, on exécute une itération à chaque unité de temps. Bien entendu, la simulation est longue dans ce cas. Si, par contre, les instants où les événements extérieurs interviennent sont connus, il est possible de faire une itération chaque fois qu'un événement se produit.

On examine alors l'événement présent et les transitions qui sont ré-

ceptives à cet événement, ainsi que les transitions auquelles l'événement "toujours présent" est attribué. Si il n'y a pas d'événement, on examine seulement ces dernières.

Pour les transitions réceptives à l'événement ou à l'événement "toujours présent", on vérifie si les coditions nécessaires sont satisfaites.

Si c'est le cas, on vérifie si les transitions sont sensibilisées par rapport à certaines manières en considérant le nombre de jetons de chaque couleur qui sont disponibles à cet instant.

On examine ensuite la liste des conflits structurels et le nombre total de jetons disponibles par couleur pour vérifier si la mise à feu d'une transition empêche complètement la mise à feu d'une autre transition. Dans ce cas, le système donne la main à l'utilisateur.

Si il n'y a pas de conflit on procède aux opérations sur le marquage en enlevant les jetons nécessaires des places d'entrée et en ajoutant les jetons nécessaires aux places de sortie. Bien entendu, on procède à la mise à feu d'une transition par rapport au plus grand nombre de manières possibles,prises dans l'ordre donné.

IV. CONCLUSIONS - PERSPECTIVES

Comme nous l'avons déjà souligné, notre simulateur vise à la fois à simuler le fonctionnement d'un réseau de Petri dans le cas le plus général (RdP généralisés temporisés-synchronisés) et permet une communication aisée avec l'environnement extérieur.

La coloration de jetons augmente la puissance de modélisation tandis que la possibilité d'introduire le réseau à partir de modules permet la modélisation à différents niveaux de détails.

On peut envisager maintenant l'édition de statistiques sur l'évolution du marquage ainsi que l'amélioration de la présentation des resultats.

On doit cependant souligner que Maclisp est particulièrement bien adapté à l'écriture d'un logiciel R.d.P.

rdpl.lisp 01/16/85 1258.1 hfh Wed

```
(defun rdpl (L M P) (prog (L1 L2 l1 l2 l3 l4 a b c l5 l10 l11 ts m ev te lt ta
                           l12 l13 l14 l15 l16 m2 m3 m4 M1 m6 m7 ti L3 in t1
                           L4 w1 tap l21)

;
; L:LA LISTE QUI CONTIENT LE RESEAU
; M:LE MARQUAGE INITIAL
; P:LA LISTE DES CONDITIONS VERIFIEES ET DE L' EVENEMENT PRESENT
;
; PASSER A LA BASE 10
;
(setq base 12)
;
(setq hes (openo "hes.lisp"))
; IMPRESSION DU MARQUAGE INITIAL
;
(princ 'MARQUAGE-INITIAL=)(terpri)
(print 'MARQUAGE-INITIAL= hes.lisp)
(princ M)(terpri)
(print M hes.lisp)(terpri)
;
; SAUVEGARDE DE LA LISTE DU RESAU
;
(setq L3 L)
(setq ti '0 try '0)
(setq ve '(c1 c2 c3 c4 c5 c6 c7 c8 c9 c10 c11 c12 c13 c14 c15 c16 c17
           c18 c19 c20 c21 c22 c23 c24 c25 c26 c27 c28 c29 c30 )  vr vc)
;
; CONFIGURATION DU RESEAU PAR PLACES
;
tag3
(cond ((null L)  (go tag4)))
(setq l1 (cadar L))
(setq l5 (caddar L))
tag54
(cond ((null l5)(go tag2)))
(setq l26 (cadr l5) l27 l1)
tag1
(cond ((null l1)(setq l1 l27 l5 (cddr l5)) (go tag54)))
```

```
  12:LISTE DU RESAU PAR PLACES
;
(setq 12 (append 12 (ncons (append
 (ncons (car 11))(ncons (caar L))
                                (ncons (car 15)) (ncons(car 126))))))
(setq 11 (cdr 11))
(setq 126 (cdr 126))
(go tag1)
tag2
(setq L (cdr L))
(go tag3)
tag4
;
;CONSTRUCTION DE LA LISTE DES CONFLITS STRUCTURELLES
;
tag7
(setq 13 12)
(cond ((null 12) (go tag8)))
(setq a (caar 13))
(setq b (cadar 13))
(setq c (caddar 13))
(setq d (car (last (car 13))) d1 d)
tag5
(setq lo (cadr 13))
(cond ((null 13) (go tag6)))
(cond ((eq a (car lo))
                    (cond ((eq b (cadr lo))(go tag55))
                          (t (go tag56)))))
(go tag55)
tag56
(setq mol (member (car d)(cadddr lo)))
(cond ((null d)(cond((null wl)(go tag55))
                    (t (go tag57))))
      ((not mol)(setq d (cddr d))(go tag56)))
(setq wl (append wl (ncons (car d))(ncons(plus(cadr d)(cadr mol)))))
(setq d (cddr d))(go tag56)
tag57
;
; 14:LISTE DES CONFLITS STRUCTURELS
;
(setq 14 (append 14 (ncons (append
  (ncons a)(ncons b)(ncons c)(ncons  (cadr lo))
                    (ncons (caddr lo))(ncons (car ve))(ncons wl))))
      ve (cdr ve))
tag55
(setq 13 (cdr 13) wl nil d d1)
(go tag5)
tag6
(setq 12 (cdr 12))
(go tag7)
tag8
```

```
(princ 'LISTE-DES-CONFLITS-STRUCTURELES=)(terpri)(princ 14)(terpri)
(print 'LISTE-DES-CONFLITS-STRUCTURELES= hes.lisp)(terpri)
(print 14 hes.lisp)(terpri)
(setq 140 14)
tag31
(cond ((eq ti 29.)(return t)))
(cond ((greaterp ti '0)(setq L L3 ts nil ev nil M1 nil t1 nil 14 140 121 nil)))
(setq L1 L3 L2 L3 L4 L3 ve vr)
;
; RECHERCHE DE L' EVENEMENT PRESENTE
;
tag10
(cond ((null L2)(go tag18)))
(setq ev (car (last(car L2))))
(cond ((eq (car P) ev)(go tag9)))
(setq L2 (cdr L2))
(go tag10)
;
; RECHERCHE DE L' EXISTENCE DES ARCS INHIBITEURS
;
tag9
;
; in:MOT CLE QUI SIGNIFIE L' EXISTENCE DES ARCS INHIBITEURS
;
(setq inh (cadr (member 'in (car L2))))
tag91
(cond ((null inh)(go tag92))
      ((assoc (car inh) M)(go tag27)))
(setq inh (cdr inh))
(go tag91)
;
; RECHERCHE DES CONDITIONS ASSOCIEES AUX TRANSITIONS
;
tag92
;
; te:LISTE DES CONDITIONS ASSOCIEES A LA TRANSITION EXAMINEE
;
(setq te (cadr(cddar L2)))
tag14
(cond ((null te)(go tag13))
      ((member (car te)(cdr P))(setq te (cdr te))(go tag14)))
      (setq L2 (cdr L2))(go tag10)
(setq te (cdr te))
(go tag14)
tag13
;
```

```
, RECHERCHE DES TRANSITIONS SENSIBILISEES
;
(cond ((null L2)(setq L2 L4 L1 L4))
      (t (go tag45)))
tag46
(cond ((null L2)(cond ((null ts)(go tag47))
                      (t (go tag18))))
      ((member (caar L2) t1)(go tag45)))
tag48
(setq L2 (cdr L2))(go tag46)
tag45
;
; 110:LISTE DES PLACES D' ENTREE POUR LA TRANSITION EXAMINEE
; 111:LISTE DES MANIERES (COULEURS) PSR RAPPORT AUXQUELLES LA TRANSITION
;     PEUT ETRE SENSIBILISEE
;
(setq 110 (cadar L2) 122 110)
(setq 111 (caddar L2) 121 nil)
tag44
(cond ((null 111)
               (cond ((null t1)(go tag43))
               (t (go tag48))))
      ((member (car 111) t1)(setq 111 (cddr 111))(go tag44)))
tag12
(cond ((null 110)(setq ts (append ts (ncons (caar L2)) (ncons(car 111))))
                (cond ((null t1)
                                  (setq L2 (cdr L2))(go tag10))
                      (t (go tag48)))))
;
, m0:LES PLACES D' ENTREE DE LA TRANSITION EXISTENT ELLES DANS LE MARQUAGE ?
;
(setq m0 (assoc (car 110) M))
(cond (m0 (go tag11))
      ((null t1)(go tag43)))
(go tag48)
tag43
(setq L2 (cdr L2))
(go tag10)
tag27
(setq 110 122 111 (cddr 111))
(cond ((null 111)(cond ((null t1)
                                  (go tag43))
                       (t (go tag48)))))
tag11
(cond ((null 121)(setq 121 (cadr 111)))
      (t (setq 121 (cdr 121))))
(setq 118 (car 121))
tag34
(cond ((null 118)(setq 110 (cdr 110) )(go tag12)))
(cond ((member (car 118) M)(go tag33)))
(go tag27)
tag33
;
```

```
; LA COULEUR EXISTE ELLE DANS LE MARQUAGE
;
(setq ml (member (car 118) M))
(setq m (assoc (car 110) ml))
(cond ((not m)(go tag27)))
(setq ml0 ml)
tag42
(cond ((null ml0)(go tag27))
      ((equal m (cadr ml0))(go tag41))
      ((atom (cadr ml0))(go tag27)))
(setq ml0 (cdr ml0))
(go tag42)
tag41
;
; Y A T IL DES JETONS DISPONIBLES A CE MOMENT LA POUR LA PLACE ET LA
; COULEUR ?
;
(setq m6 (cdr(member m ml)))
(cond ((signp g (difference (cdar m6) ti))(go tag27)))
(setq m7 (car m6))
tag28
(setq m6 (cdr m6))
(cond ((null m6)(go tag29)))
(cond((signp n (caar m6))
      (cond ((signp g (difference (cdar m6) ti))(go tag29))
            (t (rplaca m7 (plus (car m7)(caar m6)))
               (rplacd m7 (cdar m6))
  (rplaca  m6 '(0 . 0))
               (go tag28)))))
tag29
;
; LES JETONS DISPONIBLES SONT ILS SUFFISANTS
;
(cond ((signp l (difference (car m7) (cadr 118)))(go tag27)))
      (setq 118 (cddr 118))
(go tag34)
tag18
(cond ((null ts)
                (cond ((null t1) (go tag25))
                      (t (go tag47)))))
;
; RECHERCHE DES CONFLITS REELS
;
tag60
(cond ((null 14) (go tag19)))
(setq lt (car 14) vq ts)
tag15
;
```

```
; SI PLUSIEURS TRANSITIONS PARTAGENT LA MEME PLACE D' ENTREE ET SONT
; SENSIBILISEES PEUT ON LES METTRE A FEU TOUTES ?
;
(cond ((null ts)(setq ts vq)
                (cond ((< (length ta) 4.)(go tag17))
                      (t (setq tap (caddr(cdddr lt)))(go tag16))))
   ((member (car ts) lt)(cond  ((member (cadr ts) lt)
                (setq ta (append(ncons(car ts))(ncons(cadr ts)) ta)) ))))
tag58
(setq ts (cddr ts))
(go tag15)
tag16
(setq vo (cadr(member tap lt)))
tag61
(cond ((null vo)(go tag17)))
(setq m13 (member (car vo) M) v1 (assoc (car lt) m13))
(setq v2 (caadr (member v1 M)))
;
; SI CONFLIT IL Y A PRINT CONFLIT ET RETURN NIL
;
(cond ((> v2 (cadr vo))(go tag66)))
(princ 'conflit:)(princ ta)(terpri)
(print 'conflit: hes.lisp)(print ta hes.lisp)(terpri)
(princ 'quelle-transition-voulez-vous-eliminer-?)(terpri)
(print 'quelle-transition-voulez-vous-eliminer-? hes.lisp)(terpri)
(setq tre (read t))
(print tre hes.lisp)
(rplaca (member tre ts) nil)(setq  ts (delete nil ts))
(princ 'par-rapport-a-la-quelle-maniere ?)(terpri)
(print 'par-rapport-a-la-quelle-maniere ? hes.lisp)(terpri)
(setq tri (read t))
(print tri hes.lisp)
(rplaca (member tri ts) nil)(setq ts (delete nil ts))
(go tag17)
tag66
(setq vo (cddr vo))(go tag61)
tag17
(setq l4 (cdr l4) ta nil )
(go tag60)
;
```

```
; MISE EN FEU DES TRANSITIONS SENSIBILISEES
; ON ENLEVE LES JETONS DES PLACES EN AMONT
;
tag19
(cond ((null L1) (go tag21)))
(setq 112 (car L1))
(setq m11  (member (car 112) ts))
(cond ((member (car 112) ts) (setq 113 (cadr 112) 116 (caddr 112)))
      (t (go tag52)))
(setq 121 (cadr (member(cadr m11) 116)))
tag20
(cond ((null 113)(go tag52)))
(setq 119 (car 121))
tag35
(cond ((null 119)(go tag36)))
(setq m8 (member (car 119) M))
(setq m2 (assoc (car 113) m8))
(rplacd (assoc (car 113) m8) (difference (cdr m2) (cadr 119)))
(rplaca (cadr(member m2 m8)) (difference (caadr (member m2 m8)) (cadr 119)))
(setq 119 (cddr 119))
(go tag35)
tag36
(setq 113 (cdr 113))
(setq 121 (cdr 121))
(go tag20)
tag52
(setq L1 (cdr L1))(go tag19)
tag21
;
; t1:LA LISTE DES TRANSITIONS SENSIBILISEES ET DES MANIERES (COULEURS) PAR
;    RAPPORT AUXQUELLES ELLES SONT SENSIBILISEES
;
(setq t1 (append t1 ts) ts nil)
(princ 'TRANSITIONS-SENSIBILISEES=)(princ t1)(terpri)
(print 'TRANSITIONS-SENSIBILISEES= hes.lisp)(print t1 hes.lisp)(terpri)
(go tag13)
tag47
;
; ON AJOUTE DE JETONS AUX PLACES AVAL
;
(setq L1 L4)
tag51
(cond ((null L1)(go tag255)))
(setq 112 (car L1))
(setq 114 (cadr (cdddr 112)) 123 114)
(setq 115 (caddr(cdddr 112)))
(setq 117 (car (cddddr(cdddr 112))) 124 117)
tag49
;
```

```
; LA TRANSITION APPARTIENNE A LA LISTE DES TRANSITIONS SENSIBILISEES ?
; 125 : LA MANIERE (COULEUR) A METTRE A FEU LA TRANSITION
;
(setq 122 (car 115))
(cond ((null 115)(go tag50))
      ((member 122 t1)(setq 125 (cadr 115))(go tag22)))
(setq 115 (cddr 115))(go tag49)
tag22
;
; 120 : (COULEUR,POIDS RESPECTIF)
;
(cond ((null 114) (go tag23)))
(setq 120 (car 125))
tag37
(cond ((null 120)(go tag38)))
;
;(JETONS A AJOUTER TEMPS DE DISPONIBILITE)
;
(setq m5 (cons (cadr 120)(plus ti (car 117))))
;
;LA COULEUR EXISTE DEJA DANS LE MARQUAGE ?
;
(setq m9 (member (car 120) M) m19 m9)
(cond ((not m9)(go tag39)))
;
;LA PLACE EST DEJA MARQUEE POUR CETTE COULEUR ?
;
(setq m3 (assoc (car 114) m9))
tag72
(cond ((not m3)(go tag70))
      ((atom (cadr m19))(go tag70))
      ((equal (car m3) (caadr m19))(go tag71)))
(setq m19 (cdr m19))(go tag72)
tag71
;
; ON AJOUTE LES JETONS DANS LE TROIS CAS
;
tag73
(setq m19 (cdr m19))
 (cond ((null m19)(setq M (append M (ncons m5)))(go tag24))
       ((signp ge (caadr m19))(go tag73)))
 (rplacd (assoc (car 114) m9) (plus (cdr m3)(cadr 120)))
         (rplacd m19 (append (ncons m5) (cdr m19)))
```

```
 (go tag24)
tag70
(rplacd m9
 (append
 (ncons(cons (car 114)(cadr 120)))(ncons m5) (cdr m9)))
(go tag24)
tag39
(setq M (append M (ncons(car 120))(ncons(cons (car 114)(cadr 120)))(ncons m5)))
tag24
(setq 120 (cddr 120))
(go tag37)
tag38
(setq 114 (cdr 114))
(setq 125 (cdr 125))
(setq 117 (cdr 117))
(go tag22)
tag23
(setq 114 123 117 124 115 (cddr 115))(go tag49)
tag50
(setq L1 (cdr L1))
(go tag51)
tag255
tag25
;
; ON ENLEVE DU MARQUAGE LES PLACES VIDES ET LES COULEURS NON UTILISEES
;
(cond ((null M) (go tag30)))
(setq m4 (car M) m40 (cadr M))
(cond ((atom m4)(cond ((atom m40)(go tag26))
                      (t  (go tag40))))
      ((signp ge (car m4)) (go tag32))
      ((zerop (cdr m4)) (go tag26)))
tag32
(cond ((signp le (car m4)) (go tag26)))
tag40
(setq M1 (append M1 (ncons m4)))
tag26
(setq M (cdr M))
(go tag25)
;
```

```
;IMPRESSION DU MARQUAGE PAR COULEUR
;
tag30
(setq M M1)
tag62
(cond ((null M1)(go tag63))
      ((atom (car M1))(cond ((atom (cadr M1))(go tag64))
                            (t (terpri)))))
 (princ (car M1))
 (print (car M1) hes.lisp)
tag64
(setq M1 (cdr M1))(go tag62)
tag63
(terpri)
(setq ti (plus 1 ti))
(princ "UNITES-DE-TEMPS-PASSEES=)(princ ti)(terpri)
(print "UNITES-DE-TEMPS-PASSEES= hes.lisp)(princ ti hes.lisp)
(print "---------------------------------------------------------------- hes.lisp)
(princ (runtime))(terpri)
(princ (difference (runtime) try))(terpri)
(setq try (runtime))
(go tag31)))
(setq li (openi "li.lisp"))
(setq ro (openi "ro.lisp"))
(setq pev (openi "pev.lisp"))
(princ  (rdpl (read li)(read ro)(read pev)))


r 13:05 1.182 26 level 2
```

ANNEXE II

UN EXEMPLE

L'exemple qui suit montre les évolutions possibles des états d'une module station de travail d'un atelier flexible de mastiquage de carrosseries et se trouve dans l'article : "APPLICATION DES RDP A L'ETUDE DE L'AUTOMATISATION D'UN ATELIER FLEXIBLE DE MASTIQUAGE DE CARROSSERIES" (B.CAVANNA, F.DOLLE, M.MOALLA).

Ici, il nous sert comme exemple pour voir de quelle manière on introduit le réseau et on suit l'évolution du marquage,

Les places représentent les états suivants :

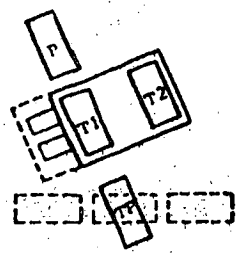
P : TABLE PIVOT POSITION PIVOT
H : TABLE PIVOT POSITION HORIZONTALE
TPnc : TABLE PIVOT NOUVELLE CARROSSERIE
TPl : TABLE PIVOT LIBRE
T2p : TABLE TAMPON CHARGEMENT OCCUPEE
T2l : TABLE TAMPON CHARGEMENT LIBRE
G : TAMPON A GAUCHE
D : TAMPON A DROITE
Pl : TABLE DE TRAVAIL LIBRE
Pt : CARROSSERIE EN COURS DE MASTIQUAGE
Pp : TABLE DE TRAVAIL OCCUPEE PAR CARROSSERIE MASTIQUEE
Tll : TABLE TAMPON DECHARGEMENT LIBRE
Tlp : TABLE TAMPON DECHARGEMENT OCCUPEE
TPcm : TABLE PIVOT CARROSSERIE MASTIQUEE A EVACUER
CP : CARROSSERIES PRETES

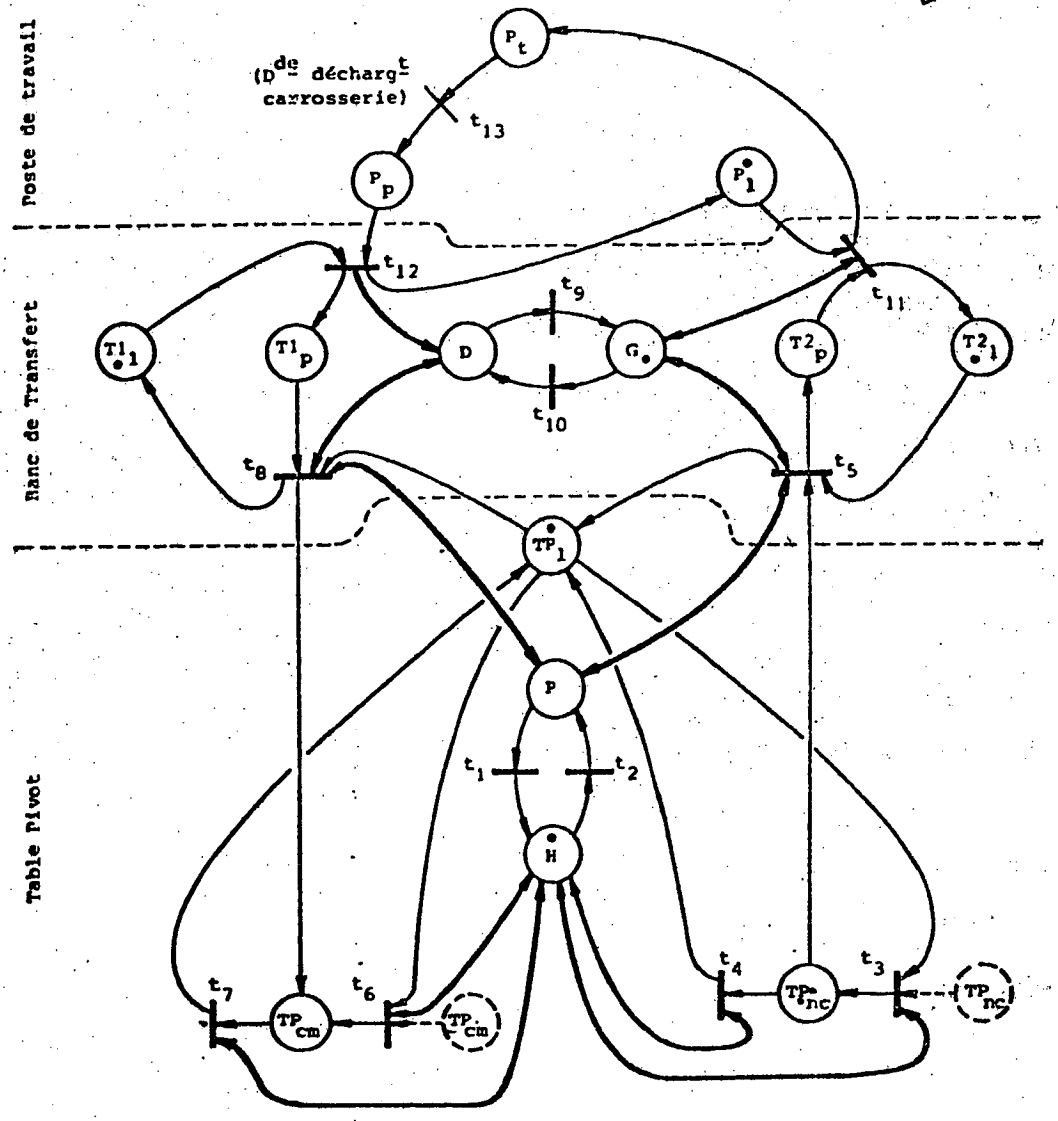
Dans notre exemple les places sont temporisées.

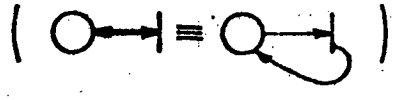
Les jetons doivent rester 1 unité de temps à chaque place, sauf la place CARROSSERIE EN COURS DE MASTIQUAGE où les jetons doivent rester 5 unités de temps.

Les jetons ont tous la même couleur INC.

Une condition toujours vraie CO et l'événement toujours présent EV sont associés aux transitions.


P
T2
T1
T'


Poste de travail
Banc de Transfert
Table Pivot
(Dde décharge carrosserie)
Pt
t13
Pp
P1
t12
t11
t9
t10
T11
T1p
D
G
T2p
T21
t8
t5
TP1
P
t1
t2
H
t7
t6
t4
t3
TPcm
TPcm
TPnc
TPnc



RESULTATS DE LA SIMULATION

```
((t1 (TABLE-PIVOT-POSITION-PIVOT) (MA1 ((INC 1.))) (CO)
     (TABLE-PIVOT-POSITION-HORIZONTAL) (MA1 ((INC 1.))) (1.) nil EV)
 (t2 (TABLE-PIVOT-POSITION-HORIZONTAL) (MA2 ((INC 1.))) (CO)
     (TABLE-PIVOT-POSITION-PIVOT) (MA2 ((INC 1.))) (1.) nil EV)
 (t3 (TPnc TABLE-PIVOT-LIBRE TABLE-PIVOT-POSITION-HORIZONTAL)
                                        (MA3 ((INC 1.) (INC 1.) (INC 1.))) (CO)
     (TABLE-PIVOT-NOUVELLE-CARROSSERIE TABLE-PIVOT-POSITION-HORIZONTAL)
                                          (MA3 ((INC 1.) (INC 1.)))(1. 1.) EV)
 (t4 (TABLE-PIVOT-NOUVELLE-CARROSSERIE TABLE-PIVOT-POSITION-HORIZONTAL)
                                            (MA4 ((INC 1.) (INC 1.))) (CO)
     (TABLE-PIVOT-LIBRE TABLE-PIVOT-POSITION-HORIZONTAL)
                                          (MA4 ((INC 1.) (INC 1.))) (1. 1.) EV)
 (t5 (TABLE-PIVOT-NOUVELLE-CARROSSERIE TABLE-PIVOT-POSITION-PIVOT
                      TABLE-TAMPON-CHARGEMENT-LIBRE TAMPON-A-GAUCHE)
                                (MA5 ((INC 1.) (INC 1.) (INC 1.) (INC 1.))) (CO)
     (TABLE-PIVOT-LIBRE TAMPON-A-GAUCHE TABLE-TAMPON-CHARGEMENT-OCCUPEE
      TABLE-PIVOT-POSITION-PIVOT)
                    (MA5 ((INC 1.) (INC 1.) (INC 1.)(INC 1.))) (1. 1. 1. 1.) EV)
 (t6 (TPcm TABLE-PIVOT-POSITION-HORIZONTAL TABLE-PIVOT-LIBRE)
                                        (MA6 ((INC 1.) (INC 1.) (INC 1.))) (CO)
     (TABLE-PIVOT-POSITION-HORIZONTAL TABLE-PIVOT-CARROSSERIE-MASTIQUEE)
                                          (MA6 ((INC 1.) (INC 1.))) (1. 1.) EV)
 (t7 (TABLE-PIVOT-CARROSSERIE-MASTIQUEE TABLE-PIVOT-POSITION-HORIZONTAL)
                                            (MA7 ((INC 1.) (INC 1.))) (CO)
     (TABLE-PIVOT-POSITION-HORIZONTAL TABLE-PIVOT-LIBRE CARROSSERIES-PRETES)
                                (MA7 ((INC 1.) (INC 1.)(INC 1.))) (1. 1. 1.) EV)
 (t8 (TABLE-TAMPON-DECHARGEMENT-OCCUPEE TAMPON-A-DROITE TABLE-PIVOT-LIBRE
                                          TABLE-PIVOT-POSITION-PIVOT)
                              (MA8 ((INC 1.) (INC 1.) (INC 1.) (INC 1.))) (CO)
     (TABLE-TAMPON-DECHARGEMENT-LIBRE TABLE-PIVOT-CARROSSERIE-MASTIQUEE
                        TAMPON-A-DROITE TABLE-PIVOT-POSITION-PIVOT)
              (MA8 ((INC 1.) (INC 1.) (INC 1.) (INC 1.))) (1. 1. 1. 1.) EV)
 (t9 (TAMPON-A-DROITE) (MA9 ((INC 1.))) (CO)
      (TAMPON-A-GAUCHE) (MA9 ((INC 1.))) (1.) EV)
 (t10 (TAMPON-A-GAUCHE)
     (MA10 ((INC 1.)))(CO) (TAMPON-A-DROITE) (MA10 ((INC 1.))) (1.) EV)
 (t11 (TABLE-TAMPON-CHARGEMENT-OCCUPEE TAMPON-A-GAUCHE TABLE-P-LIBRE)
                                        (MA11 ((INC 1.) (INC 1.) (INC 1.))) (CO)
```

```
(CARROSSERIE-EN-COURS-DE-MASTIQUAGE TAMPON-A-GAUCHE
                                                    TABLE-TAMPON-CHARGEMENT-LIBRE)
                                (MA11 ((INC 1.) (INC 1.) (INC 1.))) (5. 1. 1.) EV)
(t12 (TABLE-P-OCCUPEE-PAR-CARROSSERIE-MASTIQUEE
  TABLE-TAMPON-DECHARGEMENT-LIBRE TAMPON-A-DROITE)
                                          (MA12 ((INC 1.) (INC 1.) (INC 1.))) (CO)
(TABLE-TAMPON-DECHARGEMENT-OCCUPEE TAMPON-A-DROITE TABLE-P-LIBRE)
                               (MA12 ((INC 1.) (INC 1.) (INC 1.)))(1. 1. 1.) EV)
(t13 (CARROSSERIE-EN-COURS-DE-MASTIQUAGE) (MA13 ((INC 1.))) (CO)
     (TABLE-P-OCCUPEE-PAR-CARROSSERIE-MASTIQUEE)
                                (MA13 ((INC 1.)))(1.) EV))
```

```
r 13:07 0.304 30 level 2
```

The only difference between a rut and a grave is their dimensions.

```
Leopoulos.Theosys logged out 01/16/85  1308.6 hfh Wed
CPU usage 6 sec, memory usage 4.5 units, cost $7.78.
hangup
```

```
MARQUAGE-INITIAL=
(INC (TABLE-PIVOT-LIBRE . 1.) (1. . 0.) (TPnc . 2.) (1. . 0.) (1. . 3.) (TABLE-
\cPIVOT-POSITION-HORIZONTAL . 1.)
(1. . 0.) (TABLE-P-LIBRE . 1.) (1. . 0.) (TABLE-TAMPON-CHARGEMENT-LIBRE . 1.) (
\c1. . 0.) (TAMPON-A-GAUCHE . 1.)
(1. . 0.) (TABLE-TAMPON-DECHARGEMENT-LIBRE . 1.) (1. . 0.))
LISTE-DES-CONFLITS-STRUCTURELES=
((TABLE-PIVOT-POSITION-PIVOT t1 MA1 t5 MA5 c1 (INC 2.)) (TABLE-PIVOT-POSITION-P
\cIVOT t1 MA1 t8 MA8 c2 (INC 2.))
(TABLE-PIVOT-POSITION-HORIZONTAL t2 MA2 t3 MA3 c3 (INC 2.)) (TABLE-PIVOT-POSITI
\cON-HORIZONTAL t2 MA2 t4 MA4 c4
(INC 2.)) (TABLE-PIVOT-POSITION-HORIZONTAL t2 MA2 t6 MA6 c5 (INC 2.)) (TABLE-PI
\cVOT-POSITION-HORIZONTAL t2 MA2
t7 MA7 c6 (INC 2.)) (TABLE-PIVOT-LIBRE t3 MA3 t6 MA6 c7 (INC 2.)) (TABLE-PIVOT-
\cLIBRE t3 MA3 t8 MA8 c8 (INC 2.)
) (TABLE-PIVOT-POSITION-HORIZONTAL t3 MA3 t4 MA4 c9 (INC 2.)) (TABLE-PIVOT-POSI
\cTION-HORIZONTAL t3 MA3 t6 MA6
c10 (INC 2.)) (TABLE-PIVOT-POSITION-HORIZONTAL t3 MA3 t7 MA7 c11 (INC 2.)) (TAB
\cLE-PIVOT-NOUVELLE-CARROSSERIE
t4 MA4 t5 MA5 c12 (INC 2.)) (TABLE-PIVOT-POSITION-HORIZONTAL t4 MA4 t6 MA6 c13
\c(INC 2.)) (
TABLE-PIVOT-POSITION-HORIZONTAL t4 MA4 t7 MA7 c14 (INC 2.)) (TABLE-PIVOT-POSITI
\cON-PIVOT t5 MA5 t8 MA8 c15 (INC
2.)) (TAMPON-A-GAUCHE t5 MA5 t10 MA10 c16 (INC 2.)) (TAMPON-A-GAUCHE t5 MA5 t11
\c MA11 c17 (INC 2.)) (
TABLE-PIVOT-POSITION-HORIZONTAL t6 MA6 t7 MA7 c18 (INC 2.)) (TABLE-PIVOT-LIBRE
\ct6 MA6 t8 MA8 c19 (INC 2.)) (
TAMPON-A-DROITE t8 MA8 t9 MA9 c20 (INC 2.)) (TAMPON-A-DROITE t8 MA8 t12 MA12 c2
\c1 (INC 2.)) (TAMPON-A-DROITE t9
MA9 t12 MA12 c22 (INC 2.)) (TAMPON-A-GAUCHE t10 MA10 t11 MA11 c23 (INC 2.)))
conflit:
(t3 MA3 t2 MA2)
quelle-transition-voulez-vous-eliminer-?
t2
par-rapport-a-la-quelle-maniere-?
MA2
TRANSITIONS-SENSIBILISEES=
(t3 MA3 t10 MA10)
INC
(TAMPON-A-DROITE . 1.)
(1. . 1.)
(TABLE-PIVOT-NOUVELLE-CARROSSERIE . 1.)
(1. . 1.)
(TPnc . 1.)
(1. . 3.)
(TABLE-PIVOT-POSITION-HORIZONTAL . 1.)
(1. . 1.)
(TABLE-P-LIBRE . 1.)
(1. . 0.)
(TABLE-TAMPON-CHARGEMENT-LIBRE . 1.)
(1. . 0.)
(TABLE-TAMPON-DECHARGEMENT-LIBRE . 1.)
(1. . 0.)
UNITES-DE-TEMPS-PASSEES= 1.
```

```
conflit:
(t4 MA4 t2 MA2)
quelle-transition-voulez-vous-eliminer-?
t4
par-rapport-a-la-quelle-maniere-?
MA4
TRANSITIONS-SENSIBILISEES=
(t2 MA2 t9 MA9)
INC
(TAMPON-A-GAUCHE . 1.)
(1. . 2.)
(TABLE-PIVOT-POSITION-PIVOT . 1.)
(1. . 2.)
(TABLE-PIVOT-NOUVELLE-CARROSSERIE . 1.)
(1. . 1.)
(TPnc . 1.)
(1. . 3.)
(TABLE-P-LIBRE . 1.)
(1. . 0.)
(TABLE-TAMPON-CHARGEMENT-LIBRE . 1.)
(1. . 0.)
(TABLE-TAMPON-DECHARGEMENT-LIBRE . 1.)
(1. . 0.)
UNITES-DE-TEMPS-PASSEES= 2.
-----------------------------------------------------------
conflit:
(t5 MA5 t1 MA1)
quelle-transition-voulez-vous-eliminer-?
t1
par-rapport-a-la-quelle-maniere-?
MA1
conflit:
(t10 MA10 t5 MA5)
quelle-transition-voulez-vous-eliminer-?
t10
par-rapport-a-la-quelle-maniere-?
MA10
TRANSITIONS-SENSIBILISEES=
(t5 MA5)
INC
(TABLE-TAMPON-CHARGEMENT-OCCUPEE . 1.)
(1. . 3.)
(TABLE-PIVOT-LIBRE . 1.)
(1. . 3.)
(TAMPON-A-GAUCHE . 1.)
(1. . 3.)
(TABLE-PIVOT-POSITION-PIVOT . 1.)
(1. . 3.)
(TPnc . 1.)
(1. . 3.)
(TABLE-P-LIBRE . 1.)
(1. . 0.)
(TABLE-TAMPON-DECHARGEMENT-LIBRE . 1.)
(1. . 0.)
UNITES-DE-TEMPS-PASSEES= 3.
-----------------------------------------------------------
```

```
conflit:
(t11 MA11 t10 MA10)
quelle-transition-voulez-vous-eliminer-?
t10
par-rapport-a-la-quelle-maniere-?
MA10
TRANSITIONS-SENSIBILISEES=
(t1 MA1 t11 MA11)
INC
(TABLE-TAMPON-CHARGEMENT-LIBRE . 1.)
(1. . 4.)
(CARROSSERIE-EN-COURS-DE-MASTIQUAGE . 1.)
(1. . 8.)
(TABLE-PIVOT-POSITION-HORIZONTAL . 1.)
(1. . 4.)
(TABLE-PIVOT-LIBRE . 1.)
(1. . 3.)
(TAMPON-A-GAUCHE . 1.)
(1. . 4.)
(TPnc . 1.)
(1. . 3.)
(TABLE-TAMPON-DECHARGEMENT-LIBRE . 1.)
(1. . 0.)
UNITES-DE-TEMPS-PASSEES= 4.
```

```
conflit:
(t3 MA3 t2 MA2)
quelle-transition-voulez-vous-eliminer-?
t2
par-rapport-a-la-quelle-maniere-?
MA2
TRANSITIONS-SENSIBILISEES=
(t3 MA3 t10 MA10)
INC
(TAMPON-A-DROITE . 1.)
(1. . 5.)
(TABLE-PIVOT-NOUVELLE-CARROSSERIE . 1.)
(1. . 5.)
(TABLE-TAMPON-CHARGEMENT-LIBRE . 1.)
(1. . 4.)
(CARROSSERIE-EN-COURS-DE-MASTIQUAGE . 1.)
(1. . 8.)
(TABLE-PIVOT-POSITION-HORIZONTAL . 1.)
(1. . 5.)
(TABLE-TAMPON-DECHARGEMENT-LIBRE . 1.)
(1. . 0.)
UNITES-DE-TEMPS-PASSEES= 5.
```

```
conflit:
(t4 MA4 t2 MA2)
quelle-transition-voulez-vous-eliminer-?
t4
par-rapport-a-la-quelle-maniere-?
MA4
TRANSITIONS-SENSIBILISEES=
(t2 MA2 t9 MA9)
INC
(TAMPON-A-GAUCHE . 1.)
(1. . 6.)
(TABLE-PIVOT-POSITION-PIVOT . 1.)
(1. . 6.)
(TABLE-PIVOT-NOUVELLE-CARROSSERIE . 1.)
(1. . 5.)
(TABLE-TAMPON-CHARGEMENT-LIBRE . 1.)
(1. . 4.)
(CARROSSERIE-EN-COURS-DE-MASTIQUAGE . 1.)
(1. . 8.)
(TABLE-TAMPON-DECHARGEMENT-LIBRE . 1.)
(1. . 0.)
UNITES-DE-TEMPS-PASSEES= 6.
```

```
conflit:
(t5 MA5 t1 MA1)
quelle-transition-voulez-vous-eliminer-?
t1
par-rapport-a-la-quelle-maniere-?
MA1
conflit:
(t10 MA10 t5 MA5)
quelle-transition-voulez-vous-eliminer-?
t10
par-rapport-a-la-quelle-maniere-?
MA10
TRANSITIONS-SENSIBILISEES=
(t5 MA5)
INC
(TABLE-TAMPON-CHARGEMENT-OCCUPEE . 1.)
(1. . 7.)
(TABLE-PIVOT-LIBRE . 1.)
(1. . 7.)
(TAMPON-A-GAUCHE . 1.)
(1. . 7.)
(TABLE-PIVOT-POSITION-PIVOT . 1.)
(1. . 7.)
(CARROSSERIE-EN-COURS-DE-MASTIQUAGE . 1.)
(1. . 8.)
(TABLE-TAMPON-DECHARGEMENT-LIBRE . 1.)
(1. . 0.)
UNITES-DE-TEMPS-PASSEES= 7.
```

```
TRANSITIONS-SENSIBILISEES=
(t1 MA1 t10 MA10)
INC
(TAMPON-A-DROITE . 1.)
(1. . 8.)
(TABLE-PIVOT-POSITION-HORIZONTAL . 1.)
(1. . 8.)
(TABLE-TAMPON-CHARGEMENT-OCCUPEE . 1.)
(1. . 7.)
(TABLE-PIVOT-LIBRE . 1.)
(1. . 7.)
(CARROSSERIE-EN-COURS-DE-MASTIQUAGE . 1.)
(1. . 8.)
(TABLE-TAMPON-DECHARGEMENT-LIBRE . 1.)
(1. . 0.)
UNITES-DE-TEMPS-PASSEES= 8.
```

```
TRANSITIONS-SENSIBILISEES=
(t2 MA2 t9 MA9 t13 MA13)
INC
(TABLE-P-OCCUPEE-PAR-CARROSSERIE-MASTIQUEE . 1.)
(1. . 9.)
(TAMPON-A-GAUCHE . 1.)
(1. . 9.)
(TABLE-PIVOT-POSITION-PIVOT . 1.)
(1. . 9.)
(TABLE-TAMPON-CHARGEMENT-OCCUPEE . 1.)
(1. . 7.)
(TABLE-PIVOT-LIBRE . 1.)
(1. . 7.)
(TABLE-TAMPON-DECHARGEMENT-LIBRE . 1.)
(1. . 0.)
UNITES-DE-TEMPS-PASSEES= 9.
```

```
TRANSITIONS-SENSIBILISEES=
(t1 MA1 t10 MA10)
INC
(TAMPON-A-DROITE . 1.)
(1. . 10.)
(TABLE-PIVOT-POSITION-HORIZONTAL . 1.)
(1. . 10.)
(TABLE-P-OCCUPEE-PAR-CARROSSERIE-MASTIQUEE . 1.)
(1. . 9.)
(TABLE-TAMPON-CHARGEMENT-OCCUPEE . 1.)
(1. . 7.)
(TABLE-PIVOT-LIBRE . 1.)
(1. . 7.)
(TABLE-TAMPON-DECHARGEMENT-LIBRE . 1.)
(1. . 0.)
UNITES-DE-TEMPS-PASSEES= 10.
```

```
conflit:
(t12 MA12 t9 MA9)
quelle-transition-voulez-vous-eliminer-?
t9
par-rapport-a-la-quelle-maniere-?
MA9
TRANSITIONS-SENSIBILISEES=
(t2 MA2 t12 MA12)
INC
(TABLE-P-LIBRE . 1.)
(1. . 11.)
(TABLE-TAMPON-DECHARGEMENT-OCCUPEE . 1.)
(1. . 11.)
(TABLE-PIVOT-POSITION-PIVOT . 1.)
(1. . 11.)
(TAMPON-A-DROITE . 1.)
(1. . 11.)
(TABLE-TAMPON-CHARGEMENT-OCCUPEE . 1.)
(1. . 7.)
(TABLE-PIVOT-LIBRE . 1.)
(1. . 7.)
UNITES-DE-TEMPS-PASSEES= 11.
```

```
conflit:
(t8 MA8 t1 MA1)
quelle-transition-voulez-vous-eliminer-?
t1
par-rapport-a-la-quelle-maniere-?
MA1
conflit:
(t9 MA9 t8 MA8)
quelle-transition-voulez-vous-eliminer-?
t9
par-rapport-a-la-quelle-maniere-?
MA9
TRANSITIONS-SENSIBILISEES=
(t8 MA8)
INC
(TABLE-PIVOT-CARROSSERIE-MASTIQUEE . 1.)
(1. . 12.)
(TABLE-TAMPON-DECHARGEMENT-LIBRE . 1.)
(1. . 12.)
(TABLE-P-LIBRE . 1.)
(1. . 11.)
(TABLE-PIVOT-POSITION-PIVOT . 1.)
(1. . 12.)
(TAMPON-A-DROITE . 1.)
(1. . 12.)
(TABLE-TAMPON-CHARGEMENT-OCCUPEE . 1.)
(1. . 7.)
UNITES-DE-TEMPS-PASSEES= 12.
```

```
TRANSITIONS-SENSIBILISEES=
(t1 MA1 t9 MA9)
INC
(TAMPON-A-GAUCHE . 1.)
(1. . 13.)
(TABLE-PIVOT-POSITION-HORIZONTAL . 1.)
(1. . 13.)
(TABLE-PIVOT-CARROSSERIE-MASTIQUEE . 1.)
(1. . 12.)
(TABLE-TAMPON-DECHARGEMENT-LIBRE . 1.)
(1. . 12.)
(TABLE-P-LIBRE . 1.)
(1. . 11.)
(TABLE-TAMPON-CHARGEMENT-OCCUPEE . 1.)
(1. . 7.)
UNITES-DE-TEMPS-PASSEES= 13.
```

```
conflit:
(t7 MA7 t2 MA2)
quelle-transition-voulez-vous-eliminer-?
t2
par-rapport-a-la-quelle-maniere-?
MA2
conflit:
(t11 MA11 t10 MA10)
quelle-transition-voulez-vous-eliminer-?
t10
par-rapport-a-la-quelle-maniere-?
MA10
TRANSITIONS-SENSIBILISEES=
(t7 MA7 t11 MA11)
INC
(TABLE-TAMPON-CHARGEMENT-LIBRE . 1.)
(1. . 14.)
(CARROSSERIE-EN-COURS-DE-MASTIQUAGE . 1.)
(1. . 10.)
(CARROSSERIES-PRETES . 1.)
(1. . 14.)
(TABLE-PIVOT-LIBRE . 1.)
(1. . 14.)
(TAMPON-A-GAUCHE . 1.)
(1. . 14.)
(TABLE-PIVOT-POSITION-HORIZONTAL . 1.)
(1. . 14.)
(TABLE-TAMPON-DECHARGEMENT-LIBRE . 1.)
(1. . 12.)
UNITES-DE-TEMPS-PASSEES= 14.
```

```
TRANSITIONS-SENSIBILISEES=
(t2 MA2 t10 MA10)
INC
(TAMPON-A-DROITE . 1.)
(1. . 15.)
(TABLE-PIVOT-POSITION-PIVOT . 1.)
(1. . 15.)
(TABLE-TAMPON-CHARGEMENT-LIBRE . 1.)
(1. . 14.)
(CARROSSERIE-EN-COURS-DE-MASTIQUAGE . 1.)
(1. . 18.)
(CARROSSERIES-PRETES . 1.)
(1. . 14.)
(TABLE-PIVOT-LIBRE . 1.)
(1. . 14.)
(TABLE-TAMPON-DECHARGEMENT-LIBRE . 1.)
(1. . 12.)
UNITES-DE-TEMPS-PASSEES= 15.
```

```
TRANSITIONS-SENSIBILISEES=
(t1 MA1 t9 MA9)
INC
(TAMPON-A-GAUCHE . 1.)
(1. . 16.)
(TABLE-PIVOT-POSITION-HORIZONTAL . 1.)
(1. . 16.)
(TABLE-TAMPON-CHARGEMENT-LIBRE . 1.)
(1. . 14.)
(CARROSSERIE-EN-COURS-DE-MASTIQUAGE . 1.)
(1. . 18.)
(CARROSSERIES-PRETES . 1.)
(1. . 14.)
(TABLE-PIVOT-LIBRE . 1.)
(1. . 14.)
(TABLE-TAMPON-DECHARGEMENT-LIBRE . 1.)
(1. . 12.)
UNITES-DE-TEMPS-PASSEES= 16.
```

```
TRANSITIONS-SENSIBILISEES=
(t2 MA2 t10 MA10)
INC
(TAMPON-A-DROITE . 1.)
(1. . 17.)
(TABLE-PIVOT-POSITION-PIVOT . 1.)
(1. . 17.)
(TABLE-TAMPON-CHARGEMENT-LIBRE . 1.)
(1. . 14.)
(CARROSSERIE-EN-COURS-DE-MASTIQUAGE . 1.)
(1. . 18.)
(CARROSSERIES-PRETES . 1.)
(1. . 14.)
(TABLE-PIVOT-LIBRE . 1.)
(1. . 14.)
(TABLE-TAMPON-DECHARGEMENT-LIBRE . 1.)
(1. . 12.)
UNITES-DE-TEMPS-PASSEES= 17.
```

```
TRANSITIONS-SENSIBILISEES=
(t1 MA1 t9 MA9)
INC
(TAMPON-A-GAUCHE . 1.)
(1. . 18.)
(TABLE-PIVOT-POSITION-HORIZONTAL . 1.)
(1. . 18.)
(TABLE-TAMPON-CHARGEMENT-LIBRE . 1.)
(1. . 14.)
(CARROSSERIE-EN-COURS-DE-MASTIQUAGE . 1.)
(1. . 18.)
(CARROSSERIES-PRETES . 1.)
(1. . 14.)
(TABLE-PIVOT-LIBRE . 1.)
(1. . 14.)
(TABLE-TAMPON-DECHARGEMENT-LIBRE . 1.)
(1. . 12.)
UNITES-DE-TEMPS-PASSEES= 18.
```

```
TRANSITIONS-SENSIBILISEES=
(t2 MA2 t10 MA10 t13 MA13)
INC
(TABLE-P-OCCUPEE-PAR-CARROSSERIE-MASTIQUEE . 1.)
(1. . 19.)
(TAMPON-A-DROITE . 1.)
(1. . 19.)
(TABLE-PIVOT-POSITION-PIVOT . 1.)
(1. . 19.)
(TABLE-TAMPON-CHARGEMENT-LIBRE . 1.)
(1. . 14.)
(CARROSSERIES-PRETES . 1.)
(1. . 14.)
(TABLE-PIVOT-LIBRE . 1.)
(1. . 14.)
(TABLE-TAMPON-DECHARGEMENT-LIBRE . 1.)
(1. . 12.)
UNITES-DE-TEMPS-PASSEES= 19.
```

```
conflit:
(t12 MA12 t9 MA9)
quelle-transition-voulez-vous-eliminer-?
t9
par-rapport-a-la-quelle-maniere-?
MA9
TRANSITIONS-SENSIBILISEES=
(t1 MA1 t12 MA12)
INC
(TABLE-P-LIBRE . 1.)
(1. . 20.)
(TABLE-TAMPON-DECHARGEMENT-OCCUPEE . 1.)
(1. . 20.)
(TABLE-PIVOT-POSITION-HORIZONTAL . 1.)
(1. . 20.)
(TAMPON-A-DROITE . 1.)
(1. . 20.)
(TABLE-TAMPON-CHARGEMENT-LIBRE . 1.)
(1. . 14.)
(CARROSSERIES-PRETES . 1.)
(1. . 14.)
(TABLE-PIVOT-LIBRE . 1.)
(1. . 14.)
UNITES-DE-TEMPS-PASSEES= 20.
```

```
TRANSITIONS-SENSIBILISEES=
(t2 MA2 t9 MA9)
INC
(TAMPON-A-GAUCHE . 1.)
(1. . 21.)
(TABLE-PIVOT-POSITION-PIVOT . 1.)
(1. . 21.)
(TABLE-P-LIBRE . 1.)
(1. . 20.)
(TABLE-TAMPON-DECHARGEMENT-OCCUPEE . 1.)
(1. . 20.)
(TABLE-TAMPON-CHARGEMENT-LIBRE . 1.)
(1. . 14.)
(CARROSSERIES-PRETES . 1.)
(1. . 14.)
(TABLE-PIVOT-LIBRE . 1.)
(1. . 14.)
UNITES-DE-TEMPS-PASSEES= 21.
```

```
TRANSITIONS-SENSIBILISEES=
(t1 MA1 t10 MA10)
INC
(TAMPON-A-DROITE . 1.)
(1. . 22.)
(TABLE-PIVOT-POSITION-HORIZONTAL . 1.)
(1. . 22.)
(TABLE-P-LIBRE . 1.)
(1. . 20.)
(TABLE-TAMPON-DECHARGEMENT-OCCUPEE . 1.)
(1. . 20.)
(TABLE-TAMPON-CHARGEMENT-LIBRE . 1.)
(1. . 14.)
(CARROSSERIES-PRETES . 1.)
(1. . 14.)
(TABLE-PIVOT-LIBRE . 1.)
(1. . 14.)
UNITES-DE-TEMPS-PASSEES= 22.
-----------------------------
```

BIBLIOGRAPHIE

[1] M.MOALLA ,
"Description des systèmes décrits par des réseaux de Petri", communication présentée au 5ème Séminaire Tuniso-Français d'Informatique : l'Informatique dans la gestion et le traitement des données, 1982, Tunis.

[2] J.SIFAKIS - M.MOALLA,
"Analyse des systèmes décrits par des réseaux de Petri", communication présentée au 5ème Séminaire Tuniso-Français d'Informatique : l'Informatique dans la gestion et le traitement des données, 1982, Tunis.

[3] K.JENSEN,
"Coloured Petri nets and the invariant method", Computer Science Department, Aarhus University, 1980.

[4] G.W.BRAMS,
"Réseaux de Petri. Théorie et pratique", Masson, 1983.

[5] J.PETERSON,
"Petri net theory and the modeling of systems", Prentice-Hall, 1981.

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique