Université de Nantes Ecole Centrale de Nantes

Diplôme d'Etudes Approfondies

Formation doctorale Automatique et Informatique Appliquée.

Modèles algébriques de Systèmes à Evénements Discrets

Présenté par Benoît Marx,
à l'IRCCyN, le 21 septembre 2000.

Remerciements.

Je tiens, par ces quelques lignes, à adresser mes plus vifs remerciements à Jean-Jacques Loiseau pour son aimable disponibilité et ses conseils. Je voudrais aussi exprimer ma sincère reconnaissance aux professeurs du DEA AIA pour nous avoir fait découvrir de nouvelles notions.

Table des matières.

INTRODUCTION 7

CHAPITRE 0 9

PRÉSENTATION DE L'EXEMPLE. 9

CHAPITRE 1 12

ALGÈBRE DES DIOÏDES. 12
1 LES DIOÏDES. 12
1.1 Quelques définitions concernant les dioïdes. 12
1.2 Exemples. 13
1.3 Extension aux dioïdes matriciels. 13
2 LES TREILLIS. 13
2.1 Quelques définitions concernant les treillis. 13
2.2 Exemples. 14
2.3 Le théorème du point fixe de Knaster-Tarski. 14
3 APPLICATION 1 : L'ÉQUATION $X = AX \oplus B$. 15
3.1 L'équation $x = ax \oplus b$ dans les dioïdes complets. 15
3.2 Equations implicites matricielles : $x = Ax \oplus b$ et notions d'analyse spectrale 16
3.2.1 Résolution de l'équation implicite matricielle $x = Ax \oplus b$. 16
3.2.2 Notions d'analyse spectrale. 17
4 APPLICATION 2 : L'ÉQUATION $Y = A.X$ 22
4.1 Résolution de l'équation $y = A.x$ 22
4.2 Application : atteignabilité et commandabilité. 24
CONCLUSION. 25

CHAPITRE 2 26

LES GRAPHES D'EVÉNEMENTS TEMPORISÉS (GET). 26
1 INTRODUCTION AUX RÉSEAUX DE PETRI. 26
1.1 Définition des RdP. 26
1.2 Evolution des RdP. 27
2 LES GET. 27
2.1 Définition et propriétés des graphes d'événements temporisés. 27
2.2 Mise en équation des GET. 28
3 REPRÉSENTATION DANS $M_{in}^{ax}[[\gamma,\delta]]$. 29
3.1 Transformées en γ et en δ. 29
3.2 Le Dioïde $M_{in}^{ax}[[\gamma,\delta]]$. 30

3.2.1 Les dioïdes $\overline{\mathbf{Z}}_{\min}[[\delta]]$ et $\overline{\mathbf{Z}}_{\max}[[\gamma]]$. 30
2.3.2 Définition de $M_{in}^{ax}[[\gamma,\delta]]$. 31
2.3.3 Réalisabilité, rationalité et périodicité. 33
CONCLUSION 33

CHAPITRE 3 36

LES RÉSEAUX DE PETRI TEMPORISÉS (RDPT). 36
1 DÉFINITION ET RÈGLES D'ÉVOLUTION. 36
2 MISE EN ÉQUATION. 37
3 FORME D'ÉTAT DES RDPT. 38
4 FONCTIONNEMENT « DÈS QUE POSSIBLE ». 39
5 ILLUSTRATION. 40
6 CARACTÉRISATION DES TRAJECTOIRES. 41
7 EXEMPLE DU RÉSEAU FERROVIAIRE. 42
CONCLUSION. 46

CHAPITRE 4 48

AUTOMATES ET LANGAGES. 48
1 DÉFINITIONS ET OPÉRATIONS SUR LES LANGAGES. 48
1.1 Définitions. 48
1.2 Opérations sur les langages. 49
2 LES AUTOMATES. 49
2.1 Définitions et représentation. 49
2.2 Déterminisation d'un automate. 51
3 LE THÉORÈME DE KLEENE. 52
4 PRODUITS DE LANGAGES ET D'AUTOMATES. 53
4.1 Produits de langages. 53
4.2 Produits d'automates. 54
5 SUPERVISION D'AUTOMATE. 55
5.1 Notations et définitions. 55
5.1.1 Partition commandable/non commandable. 55
5.1.2 Loi de commande, superviseur, système en boucle fermée. 56
5.2 Résolution des problèmes de commande. 57
5.3 Commande optimale. 57
5.4 Calcul des éléments suprémaux. 59
CONCLUSION. 61

CHAPITRE 5 63

AUTOMATES TEMPORISÉS : LE MODÈLE DE BRANDIN ET WONHAM. 63
1 DÉFINITION D'UN SEDT. 63
2 FONCTIONNEMENT DES SEDT. 65
3 PROGRAMME DE TEMPORISATION DE SEDT, SOUS MATLAB. 66
4 COMPOSITION DE SEDT. 66
5 COMMANDE DES SEDT. 67
5.1 Partition de Σ_{act}. 67

5.2 Superviseurs. 67
5.3 Résolution du problème de commande. 68
6 EXEMPLE DU RÉSEAU FERROVIAIRE. 69
CONCLUSION 72

CHAPITRE 6 74

AUTOMATES TEMPORISÉS : LE MODÈLE D'ALUR ET DILL. 74
1 ω-AUTOMATES. 74
2 AUTOMATES TEMPORISÉS. 75
2.1 Langages temporisés. 76
2.2 Tables de transitions temporisées. 76
3 AUTOMATES RÉGIONS. 78
3.1 Restriction de l'étude au cas des contraintes entières. 78
3.2 Régions temporelles et automates région. 78
4 COMPOSITION D'AUTOMATES TEMPORISÉS. 80
5 COMMANDE D'AUTOMATES TEMPORISÉS. 80
6 EXEMPLE DU RÉSEAU FERROVIAIRE. 82
CONCLUSION. 84

CONCLUSION 86

BIBLIOGRAPHIE 88

ANNEXE 90

PROGRAMME DE TEMPORISATION DE SED. 92
1 PROGRAMMES. 92
2 EXEMPLES. 100

Introduction

On présentera ici quelques techniques de modélisation ou d'analyse des systèmes à événements discrets (SED). Les SED sont utilisés pour la modélisation de systèmes pour lesquels les techniques continues ne sont pas adaptées.

En effet la théorie classique des systèmes continus s'attache à la description des équations différentielles ou aux dérivées partielles, comme $x' = f(x,u)$, où x représente l'état, u une commande et où f est un opérateur causal suffisamment régulier pour que x soit définie presque partout. Ce type de modélisation n'est pas facilement utilisable pour prendre en compte les phénomènes de synchronisations (par exemple synchronisation d'arrivées de pièces pour un assemblage ou synchronisation de feux dans un réseau de transport), de transitions d'états (système informatique), de saturations (utilisation de stocks de dimension finie), gestion des conflits (utilisation d'une ressource par plusieurs utilisateurs) et de façon plus générale tous les phénomènes de nature discrète.

Les systèmes à événements discrets sont précisément des systèmes où se produisent des changements brusques d'état. Ces changements d'état interviennent à l'occurrence ou constituent, des événements discrets. Contrairement au cas des systèmes linéaires, il n'existe pas pour les SED de modèle naturel. De nombreuses théories complémentaires se sont développées et permettent chacune de traiter des phénomènes spécifiques : équations logiques, chaînes de Markov, réseaux de files d'attente, graphes d'événements, systèmes min-plus (ou max-plus) linéaires, systèmes de transitions, automates et langages…

Mon objectif est de commencer à étudier une classe assez générale de SED, pour laquelle la théorie n'est pas très développée. Cette classe correspond aux systèmes modélisables par des réseaux de Petri temporisés. Ce rapport, de nature essentiellement bibliographique, a pour objet de présenter quelques directions et cas particuliers pour lesquels des théories sont connues. Ce travail a pour prétexte une étude de cas : un réseau ferroviaire. En utilisant les différentes approches des SED on tentera dans un premier temps de modéliser le réseau, puis dans un second de le commander afin d'en sécuriser le fonctionnement.

Les modèles de base utilisés pour l'étude des SED sont essentiellement graphiques : Grafcet, automates, réseaux de Petri. Des modèles algébriques ont été développés, et utilisés dans une optique de commande, en particulier dans deux cas : les automates et les graphes d'événements. Il s'agit respectivement de la théorie des langages et de la supervision de Ramadge et Wonham, et de la théorie des systèmes max-plus linéaires. Ces théories utilisent les mêmes outils mathématiques : monoïdes, dioïdes, treillis, structures ordonnées. Les problèmes

de commande se ramènent à des problèmes de points fixes, de résiduation d'opérateurs dans des algèbres tropicales[1].

Dans le chapitre introductif, on présentera l'exemple qui sera régulièrement illustré au cours de ce rapport, à savoir un réseau ferroviaire constitué de deux trains dont on désire éviter la collision. Dans un premier chapitre on donnera les bases de la théorie algébrique des dioïdes, en se limitant aux résultats intéressants pour l'étude des systèmes à événements discrets. On donnera aussi deux des applications les plus utiles de cette théorie, la résolution des équations y = Ax et x = ax + b. Ensuite, le présent rapport est construit autour de deux axes principaux de recherche. Les réseaux de Petri constituent le premier, et les automates le second. Ainsi dans le deuxième chapitre on présentera les réseaux de Petri en vue d'en étudier un cas particulier : les Graphes d'Evénements Temporisés (GET). On verra leur analyse dans l'algèbre (Max,+) et leur représentation dans un dioïde particulier. Le troisième chapitre aura pour objet une approche plus générique que les GET : les Réseaux de Petri Temporisés, et les possibilités de représentation qu'ils offrent. On entame la seconde partie avec le quatrième chapitre dans lequel on exposera des notions de base sur les langages commandables et les automates, ainsi que leur principale application : la supervision par automate. Les deux chapitres suivants ont pour objet l'introduction du temps dans les modèles d'automates. On exposera les deux conceptions les plus connues. D'abord le modèle établi en discrétisant le temps, proposé par Brandin et Wonham, puis celui d'Alur et Dill, qui utilise une représentation dense du temps.

[1] On peut consulter à ce propos le site http://amadeus.inria.fr/tropical

Chapitre 0

Présentation de l'exemple.

Dans ce chapitre on présente l'exemple qu'on retrouvera souvent comme illustration des théories présentées. Il s'agit d'un réseau ferroviaire constitué de deux trains empruntant chacun un circuit formé de deux tronçons. Les deux réseaux ont un tronçon commun, sur lequel les deux trains circulent en sens inverse, il s'agit donc d'éviter les collisions ainsi que les situations de blocage, les deux trains se trouvant « nez à nez ».

On utilisera les notations suivantes. Le train n°1, parcourt le plus grand circuit, composé du tronçon A et du tronçon commun C, tandis que le train n°2 roule sur les tronçons B et C. Les tronçons A, B et C sont parcourus par les trains en, respectivement, 2 unités de temps (u.t.), 1 u.t. et 1 u.t. (les deux trains ont la même vitesse sur ce tronçon). On trouvera ci-contre le schéma du réseau. De plus on suppose que le train 1 (resp. 2) se trouve initialement sur le tronçon A (resp. B).

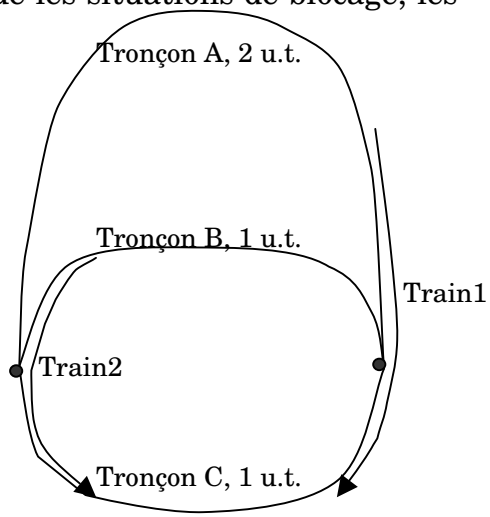


Fig. *schéma du réseau ferroviaire.*

Avant d'étudier les outils de modélisation des SED, on peut dégager certaines caractéristiques typiques des SED sur cet exemple. Tout d'abord il convient de noter que ce système n'a rien d'intrinsèquement discret, on pourrait très bien considérer les abscisses curvilignes le long des trajectoires de chaque train, les vitesses ou accélérations… En revanche on adopte un point de vue discret en choisissant de tracer le chronogramme du réseau en codant l'état du système par un couple $(x_1,x_2) \in \{A,B,C\}^2$. Avec comme état initial (A,B), on a le chronogramme suivant.

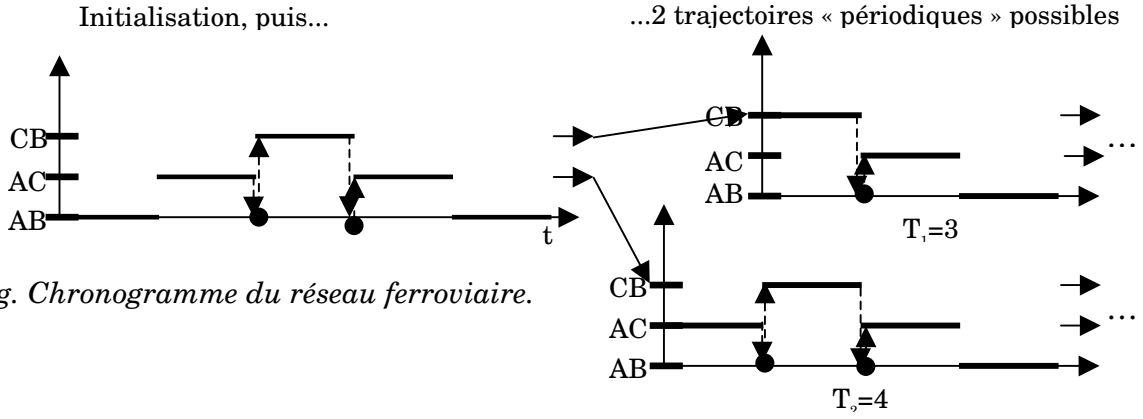


Fig. Chronogramme du réseau ferroviaire.

On observe qu'après un fonctionnement d'initialisation de 5 u.t., le système peut prendre deux trajectoires différentes qui bouclent toutes deux sur ce point de divergence. On a des comportements typiques des SED :

- des trajectoires présentant des transitions d'état ou sauts ;
- des synchronisations : pour qu'un train emprunte un tronçon il faut que le train se présente en entrée de ce tronçon d'une part et que le tronçon se libère d'autre part ;
- des choix à effectuer, à $t = 5$, on peut faire entrer un train ou l'autre sur le tronçon C.

On peut être naturellement tenté de modéliser ce système graphiquement par un automate à états finis du type :

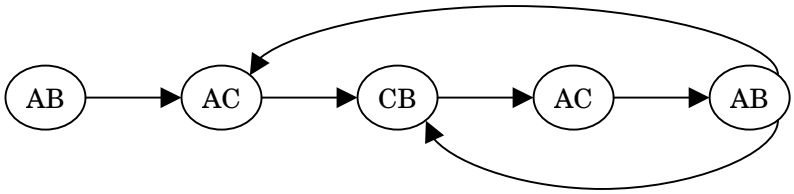


Fig. Modélisation du réseau ferroviaire.

Cependant on pourrait aussi faire apparaître les passages « fugaces » du système en (A,B) entre (A,C) et(C,B), on pourrait tenter d'affecter des durées aux différents états…
C'est dans ce but que des outils ont été mis au point, on présentera dans le présent rapport les réseaux de Petri, et les automates (non temporisés et temporisés).

Chapitre 1

Algèbre des dioïdes.

On présente dans ce chapitre les résultats élémentaires de la théorie des dioïdes, appliquée à l'étude des systèmes à événements discrets, ainsi que les possibilités de résolution des équations du type y = Ax et x = ax +b. La plupart des résultats et démonstrations de ce chapitre proviennent de *[9]*

1 Les dioïdes.

1.1 Quelques définitions concernant les dioïdes.

Définition 1.1 (Dioïde) Un ensemble D muni de deux opérations internes, l'une notée additivement $\oplus$, l'autre notée multiplicativement $\otimes$, est un dioïde si les axiomes suivants sont vérifiés:

Axiome 1.1 Associativité de $\oplus$ et $\otimes$

$$\forall\, \alpha,\beta,\gamma \in D,\ (\alpha \oplus \beta) \oplus \gamma = \alpha \oplus (\beta \oplus \gamma) \text{ et } (\alpha \otimes \beta) \otimes \gamma = \alpha \otimes (\beta \otimes \gamma) \ .$$

Axiome 1.2 Commutativité de $\oplus$

$$\forall\, \alpha,\beta \in D,\ \alpha \oplus \beta = \beta \oplus \alpha \ .$$

Axiome 1.3 Distributivité de $\otimes$ sur $\oplus$

$$\forall\, \alpha,\beta,\gamma\, D,\ (\alpha \oplus \beta) \otimes \gamma = (\alpha \otimes \beta) \oplus (\alpha \otimes \gamma) \ .$$

Axiome 1.4 Eléments neutres de $\oplus$ ("zéro") et de $\otimes$ ("identité")

$$\exists \varepsilon \in D : \forall \alpha \in D,\ \alpha \oplus \varepsilon = \alpha \ ,$$

$$\exists e \in D : \forall \alpha \in D,\ \alpha \otimes e = e \otimes \alpha = \alpha \ .$$

Axiome 1.5 Propriété d'absorption du zéro pour $\otimes$

$$\forall a \in D,\ \alpha \otimes \varepsilon = \varepsilon \otimes \alpha = \varepsilon \ .$$

Axiome 1.6 Idempotence de l'addition

$$\forall \alpha \in D,\ \alpha \oplus \alpha = \alpha \ .$$

Définition 1.2 (Dioïde commutatif) Un dioïde est dit commutatif si et seulement si la multiplication, notée $\otimes$, est commutative.

Définition 1.3 (Dioïde complet) Un dioïde est complet si et seulement si il est fermé pour les sommes infinies et si l'axiome 1.3 s'étend aux sommes infinies.

Théorème 1.1 Dans un dioïde, la relation suivante, notée $\geq$, est une relation d'ordre partiel.

$$\alpha \geq \beta \Leftrightarrow \alpha = \alpha \oplus \beta \ (\Leftrightarrow \exists \gamma : \alpha = \beta \oplus \gamma) \ .$$

L'élément minimal de D est ε. Deux éléments α et β de D ont toujours une borne supérieure, égale à $\alpha \oplus \beta$. De plus l'addition et la multiplication sont isotones pour cet ordre, autrement dit:

$$\forall\, \alpha,\beta,\gamma \in D,\ \alpha \geq \beta \Leftrightarrow \alpha \oplus \gamma \geq \beta \oplus \gamma\ ,$$
$$\forall\, \alpha,\beta,\gamma \in D,\ \alpha \geq \beta \Leftrightarrow \alpha \otimes \gamma \geq \beta \otimes \gamma\ .$$

1.2 Exemples.

Exemple 1.1 L'ensemble $3 \cup \{ -\infty \}$ muni de l'opération max, notée ⊕, et de l'addition usuelle, notée ⊗, que nous noterons 3_{max} est un dioïde (on a alors $\varepsilon=-\infty$) et est communément appelé "algèbre (max, +)".

Exemple 1.2 L'ensemble $3 \cup \{ +\infty \}$ muni de l'opération min, notée ⊕, et de l'addition usuelle, notée ⊗, que nous noterons 3_{min} est un dioïde (on a alors $\varepsilon=+\infty$) et est communément appelé "algèbre (min, +)".

1.3 Extension aux dioïdes matriciels.

Le concept de dioïde peut être étendu aux matrices carrées. A partir d'un dioïde "scalaire", considérons les matrices carrées n×n à valeurs dans D. Les sommes et les produits de matrices sont définis à partir des sommes et produits usuels de D. L'ensemble des matrices n×n muni des opérations ⊕ et ⊗ définies par :

$$A = (A_{ij}),\ B = (B_{ij}),\ (A \oplus B)_{ij} = A_{ij} \oplus B_{ij},\ (A \otimes B)_{ij} = \bigoplus_{i=1}^{n} (A_{ij} \otimes B_{ij}).$$

possède aussi la structure de dioïde et est noté $D^{n\times n}$. L'élément neutre de l'addition est la matrice dont les coefficients sont tous égaux à ε, sauf les termes diagonaux égaux à ***e*.**

2 Les treillis.

2.1 Quelques définitions concernant les treillis.

Définition 1.4 (demi-treillis) (T,+) est un demi-treillis si la loi de composition vérifie l'associativité, l'idempotence et la commutativité.

Lemme 1.1 Soit (T,⊕) un demi-treillis, la relation : $\alpha \leq \beta \Leftrightarrow \alpha \oplus \beta = \beta$ définit une relation d'ordre sur T.

Définition 1.5 ((sup) et (inf) demi-treillis)) Un ensemble ordonné (E, ≤) pour lequel deux éléments quelconques ont toujours une plus petite borne supérieure (appelée sup) est un (sup) demi-treillis. On peut définir de manière analogue un (inf) demi-treillis en vérifiant l'existence d'une plus grande borne inférieure (appelée inf).

Autrement dit, dans un (sup) demi-treillis on a toujours :

$$\forall \alpha, \beta, \exists \gamma \text{ tel que : } \alpha \leq \gamma, \beta \leq \gamma ,$$
$$\forall\ \delta, \alpha \leq \delta, \beta \leq \delta \Rightarrow \gamma \leq \delta .$$

Définition 1.6 (treillis) Un ensemble (T , ≤) est un treillis si toute partie finie de T possède un sup et un inf dans T.
Le sup de deux éléments α et β est noté $\alpha \vee \beta$ et l'inf est noté $\alpha \wedge \beta$.

Propriétés 1.1 Soit T un treillis, pour tous α, β, γ ∈ T, on a:
1. $\alpha \vee \alpha = \alpha$ (idempotence),
2. $\alpha \vee \beta = \beta \vee \alpha$ (commutativité),
3. $\alpha \vee (\beta \vee \gamma) = (\alpha \vee \beta) \vee \gamma$ (associativité),
4. $\alpha \vee (\beta \wedge \gamma) = \alpha$ (absorption),
5. $\alpha \vee (\beta \wedge \gamma) \leq (\alpha \vee \beta) \wedge (\alpha \vee \gamma)$,
6. $\alpha \wedge (\beta \vee \gamma) \geq (\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$.

On peut vérifier que, réciproquement, un ensemble T muni de deux opérations ∨ et ∧ vérifiant les propriétés 1.1 est un treillis

Définition 1.7 (Top) Dans un treillis le plus grand élément est la somme de tous les éléments du treillis. Il est noté T (Top).

2.2 Exemples.

Exemple 1.3 3 ordonné usuellement est un treillis, on a alors $\alpha \vee \beta = max(\alpha, \beta)$ et $\alpha \wedge \beta = min(\alpha, \beta)$.

Exemple 1.4 Etant donné un ensemble F, l'ensemble des parties de F ordonnées par l'inclusion est un treillis, où les bornes sup et inf sont données respectivement par:

$$A \wedge B = A \cap B,\ A \vee B = A \cup B .$$

2.3 Le théorème du point fixe de Knaster-Tarski.

Un des principaux résultats concernant les treillis est le théorème du point fixe, en effet un certain nombre de preuves d'existences d'éléments suprémaux ou de solutions d'équations reposent sur ce théorème.

Théorème 1.2 Une application croissante d'un treillis complet dans lui même admet un point fixe.

Démonstration. On veut démontrer que $x_{inf} = inf\{ t \in T \mid t \geq f(t)\}$ est un point fixe de f (i.e. $f(x_{inf}) = x_{inf}$), pour cela on montre que l'ensemble $A = \{t \in T \mid t \geq f(t)\}$, est non vide car étant complet il admet un plus grand élément T satisfaisant $T \geq f(T)$. De plus par définition de x_{inf} et par monotonie f on a :

$$\forall\ t \in A,\ t \geq f(t) \Rightarrow t \geq x_{inf} \Rightarrow f(t) \geq f(x_{inf}) ,$$

donc $x_{inf} = inf(A) \geq f(x_{inf})$ et par monotonie de f on a : $x_{inf} \geq f(x_{inf}) \Rightarrow f(x_{inf}) \geq f(f(x_{inf}))$ ce qui montre que f(x_{inf})∈ A, donc $f(x_{inf}) \geq inf(A) = x_{inf}$, finalement il vient $f(x_{inf}) = x_{inf}$.

3 Application 1 : l'équation $x = ax \oplus b$.

3.1 L'équation $x = ax \oplus b$ dans les dioïdes complets.

On peut d'abord vérifier que T est toujours solution de $x = ax \oplus b$ (pour a≠ε, sinon b est l'unique solution). C'est le plus grand point fixe de $f(x) = ax \oplus b$. On peut aussi chercher son plus petit point fixe. On pourrait considérer la fonction $f(x) = ax \oplus b$, et appliquer le théorème du point fixe en remarquant que la fonction f est croissante (on rappelle que ⊕ désigne le max donc $f(x) = max(a+x,b)$). Mais il est plus intéressant de chercher la plus petite solution, pour cela on introduit un nouvel opérateur l'étoile de Kleene.

Définition 1.8 (L'étoile de Kleene) On définit l'étoile de Kleene par **:**

$$A^* = \oplus_{n \in \angle} a^n.$$

Notation 1.1 (Opérateur « plus ») L'opérateur « plus », qui dérive de l'étoile est défini par :

$$a^+ = a^* a = \oplus_{n \in \angle^*} a^n.$$

On a de plus la relation :

$$e \oplus a^+ = a^*.$$

Proposition 1.1 Soit D un dioïde complet, α, β ∈ D, et c un entier strictement positif. On a :

1.$(\alpha^*)^* = \alpha^*$,
2. $(\alpha^+)^* = a^*$,
3. $\alpha^* = (\alpha^0 \oplus \ldots \oplus \alpha^{c-1})(\alpha^c)^*$,
4. $(\alpha \oplus \beta)^* = (\alpha^*\beta)^*\alpha^*$,
5. $(\alpha \oplus \beta)^* = \beta^*(\alpha\beta^*)^*$,
6. $\alpha^* = \alpha^*\alpha^*$,
7. $(\alpha\beta^*)^+ = \alpha(\alpha \oplus \beta)^*$,
8. $(\alpha\beta^*)^* = e \oplus \alpha(\alpha \oplus \beta)$.

De plus si α et β commutent, on a :

9. $(\alpha \oplus \beta)^* = \alpha^*\beta^*$.

On peut trouver les preuves de ces propositions à la page 30 de [9].

Théorème 1.3 L'inéquation $x \geq ax \oplus b$, dans un dioïde complet admet une plus petite solution, égale à a^*b. De plus $x_0 = a^*b$ réalise l'égalité :

$$x_0 = ax_0 + b \ .$$

3.2 Equations implicites matricielles : $x = Ax \oplus b$ et notions d'analyse spectrale .

Dans ce paragraphe on considère toujours l'équation implicite $x = Ax \oplus b$, mais A est ici une matrice carrée, x et b sont des vecteurs, tous à coefficients dans un dioïde complet D. En se plaçant dans le dioïde des matrices carrées $D^{n\times n}$, et en complétant b et x en des matrices carrées par des colonnes de ε, on se ramène à appliquer le théorème 1.3, donc la plus petite solution de l'équation matricielle est donnée par $x = A^*b$. En outre, on vérifie l'égalité. Cependant l'expression $x = A^*b = \oplus_n A^n b$ n'est pas directement utilisable, on cherche donc un algorithme permettant de calculer les étoiles de matrices quand les étoiles de coefficients sont connues

3.2.1 Résolution de l'équation implicite matricielle $x = Ax \oplus b$.

Lemme 1.2 Pour la matrice suivante, partitionnée en quatre blocs, on a :

$$A^* = \begin{bmatrix} a & b \\ c & d \end{bmatrix}^* = \begin{bmatrix} a^* \oplus a^* b(ca^* b \oplus d)^* ca^* & a^* b(ca^* b \oplus d)^* \\ (ca^* b \oplus d)^* ca^* & (ca^* b \oplus d)^* \end{bmatrix}.$$

Démonstration. (on reprend ici la démonstration surtout à titre d'exemple de résolution d'équations implicites, plus que pour son intérêt théorique)
Soit X définie par :

$$X = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{bmatrix}.$$

A* est la plus petite solution de $X \geq AX \oplus Id$, autrement dit, on a le système suivant :

$$\begin{gathered} x_{11} \geq ax_{11} \oplus bx_{21} \oplus e, \\ x_{12} \geq ax_{12} \oplus bx_{22,} \\ x_{21} \geq cx_{11} \oplus dx_{21,} \\ x_{22} \geq c\, x_{12} \oplus d\, x_{22} \oplus e. \end{gathered}$$

Les deux premières équations permettent d'éliminer respectivement x_{11} et x_{12} puis de reporter les expressions obtenues dans les deux suivantes, on a alors les inégalités suivantes:

$$\begin{gathered} x_{11} \geq a^*(bx_{21} \oplus e) \Rightarrow x_{21} \geq (d \oplus ca^* b)x_{21} \oplus ca^*, \\ x_{12} \geq a^* bx_{22} \Rightarrow x_{22} \geq (d \oplus ca^* b)x_{22} \oplus e. \end{gathered}$$

On a les solutions pour x_{21} et x_{22}, à partir desquelles on obtient celles de x_{11} et x_{12} pour obtenir le système suivant :

$$\begin{gathered} x_{11} \geq a^* \oplus a^* b(ca^* b \oplus d)^* ca^*, \\ x_{12} \geq a^* b(d \oplus ca^* b)^*, \\ x_{21} \geq (ca^* b \ \ d)^* ca^*, \\ x_{22} \geq (ca^* b \oplus d)^*. \end{gathered}$$

On a montré l'inégalité, pour prouver l'égalité il faut reporter les expressions trouvées dans $AX \oplus Id$ et de vérifier l'égalité X = AX ⊕ Id, on a alors le résultat souhaité.

3.2.2 Notions d'analyse spectrale.

3.2.2.1Graphe et matrice associée.

On peut associer une matrice à un graphe orienté. Un certain nombre de propriétés de la matrice (par exemple les valeurs ou vecteurs propres) sont alors associées à des propriétés du graphe. On donne par la suite quelques notions d'analyse spectrale.

Définition 1.9 (Matrice associée à un graphe orienté) On rappelle qu'un graphe orienté est un couple de deux ensembles (S,A), avec $A \subset S^2$, S est l'ensemble des sommets, A l'ensemble des arêtes. On pourra repérer chaque arête (i,j) par une lettre d'un alphabet Σ. A un graphe à n sommets, on associe la matrice $M \in P(A^*)^{n\times n}$ telle que :

$$M_{ij} = \begin{cases} lettre\ valuant\ l'arc\ j \to i, s'il\ existe \\ \varepsilon \quad autrement \end{cases}.$$

Exemple 1.5 On a ci-dessous un exemple de graphe et de matrice associée :

$$M = \begin{bmatrix} \varepsilon & d & \varepsilon \\ f & \varepsilon & c \\ b & \varepsilon & a \end{bmatrix}$$

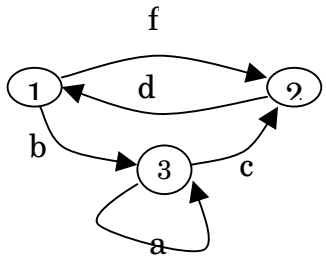


Fig. Graphe et matrice associée.

On appelle chemin de longueur k de j à i une suite $p = (i_k, i_{k-1}, \ldots, i_1)$, avec $i_1=j$ et $i_k= i$ telle que pour tout $l = 1, \ldots, k-1$, $i_1 \to i_{i+1}$ soit un arc. Si $i = j$, on parle de circuit. Le poids d'un chemin p est défini par :

$$w(p) = M_{i^k i^{k-1}} \otimes \ldots \otimes M_{i^2 i^1}.$$

Quitte à identifier le mot w(p) et le singleton {w(p)}, on pourra voir M comme une matrice à coefficients dans P(A*). Le théorème suivant établit l'équivalence entre la multiplication de matrices et la concaténation de chemins.

Théorème 1.4 M^k_{ij} est égal à la somme des poids des chemins de longueur k allant de j à i. M^*_{ij} est égale à la somme des poids des chemins du graphe de j à i (de longueur arbitraire).

Démonstration. La propriété pour les M^k entraîne la propriété pour M*. On se limitera à M^2, le cas général est analogue. Pour M^2 on a :

$$M^2_{ij} = \oplus_k M_{ik} M_{kj}.$$

$M_{ik} M_{kj}$ est non nul ssi il y a un arc de *j* à *k* et de *k* à *i*, c'est à dire ssi *(i, k, j)* est un chemin de longueur 2 de *j* à *i*, auquel cas $w(i, j, k) = M_{ik} M_{kj}$ est le poids de ce chemin.

Une des conséquences du théorème précédent est qu'on obtient les poids extrémaux en considérant uniquement les coefficients de M*. En effet prendre la somme c'est prendre le minimum (ou le maximum si on se place dans ($3 \cup \{-\infty, +\infty\}$, max, +) et non dans ($3 \cup \{-\infty, +\infty\}$, min, +)).

Théorème 1.5 M^*_{ij} est égal au poids minimum (3_{min})d'un chemin de j à i.

Si on se place dans 3 et non dans $3 \cup \{-\infty, +\infty\}$, on peut se demander quand M* converge. M* converge à la condition qu'il n'y ait pas de chemin de poids négatif (on peut retrouver la preuve à la page 37 de [9]).

3.2.2.2 Matrices irréductibles.

Il est intéressant d'étudier les valeurs propres et vecteurs propres d'une matrice, pour cela définissons préalablement le concept de matrice irréductible.
Soit (S, ⊕, ⊗) un semianneau positif, c'est à dire vérifiant les deux propriétés suivantes :

$$\alpha \oplus \beta = \varepsilon \Rightarrow \alpha = \varepsilon \text{ et } \beta = \varepsilon ,$$
$$\alpha \otimes \beta = \varepsilon \Rightarrow \alpha = \varepsilon \text{ ou } \beta = \varepsilon .$$

Dans un graphe orienté, on définit la relation de forte connexité : i$\Re$j s'il existe un chemin de i à j et aussi un chemin de j à i. Par convention, il existe toujours un chemin (de longueur nulle) de i à i. Les classes d'équivalence sont appelées les composantes fortement connexes. Un graphe est dit fortement connexe s'il y a une seule classe d'équivalence pour la relation $\Re$.

Proposition 1.2 Soit S un semianneau positif et $M \in (S^+)^{n\times n}$. les propositions suivantes sont équivalentes :

1. le graphe associé à la matrice M est fortement connexe;
2. on ne peut pas partitionner la matrice M sous la forme triangulaire par blocs :

$$A = P\begin{bmatrix} B & C \\ \varepsilon & D \end{bmatrix}P^{-1}$$, où P est une matrice de permutation;

3.$(I \oplus A)^{n-1}$ a tous ses coefficients strictement positifs.

Définition 1.10 (Matrice irréductible) La matrice M est dite irréductible lorsqu'elle vérifie les conditions de la proposition 1.2.

L'intérêt de l'irréductibilité tient, entre autre, à ce qu'un vecteur propre d'une matrice irréductible a toutes ses composantes non nulles.

Lemme 1.3 (Solidarité des matrices irréductibles) Soit S un semianneau positif, $M \in S^{n\times n}$ irréductible et $u \in (3^+)^n \setminus \{\varepsilon\}$ tel que $Au = ru$. Alors $u_i \neq \varepsilon$ pour tout $i \in \{1,\ldots,n\}$.

Démonstration. Soit j un indice tel que $u_j \neq \varepsilon$. De $ru_i = (Au)_i = \oplus_k A_{ik}u_k$, on déduit que tout i accessible en un coup depuis j (i.e. tel que $A_{ij} \neq \varepsilon$) vérifie $u_i \neq \varepsilon$. En raisonnant de proche en proche, tout i étant accessible en un nombre fini de coups depuis j, on a que $u_i \neq \varepsilon$.

3.2.2.3 Valeurs propres, vecteurs propres et cyclicité.

Théorème 1.6 Une matrice $M \in (3_{max})^{n\times n}$ irréductible admet une unique valeur propre, notée ρ(M), donnée par :

$$\rho(M) = \bigoplus_{i=1}^{n} (tr\, M^i)^{1/i}.$$

En écrivant la trace comme la somme des poids des circuit de A^i et en remarquant que le poids d'un circuit composé de plusieurs circuits élémentaires est dominé par le plus grand poids moyen de ses circuits élémentaires, le rayon spectral apparaît comme la moyenne géométrique maximale (dans l'algèbre(max,+), ce qui correspond à la moyenne arithmétique dans l'algèbre usuelle) des poids des circuits élémentaires du graphe.

$$\rho(M) = \oplus_c\, w(c)^{1/l(c)}$$

Exemple 1.6 Graphe critique d'une matrice.

$$M = \begin{bmatrix} 1 & \varepsilon & 2 \\ e & \varepsilon & \varepsilon \\ \varepsilon & 2 & 3 \end{bmatrix}$$

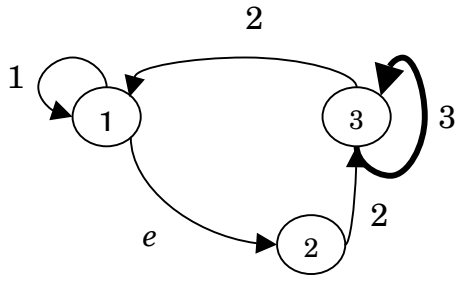


Fig. Graphe critique et matrice.

Pour la matrice M ci-dessus, le graphe associé n'admet que trois circuits élémentaires (1,1), (3,3) et (1, 2, 3, 1), on a donc :

$$\rho(M) = max(A_{11}, A_{33}, (A_{32} + A_{21} + A_{13})/3) = 3.$$

Définition 1.11 (Graphe critique) Les circuits qui réalisent le maximum dans la formule du rayon spectral sont qualifiés de critiques. On appelle graphe critique, noté GC(M), le sous-graphe de G(M) formé des sommets et arêtes appartenant à un circuit critique. On notera $C^c = (C^c_1,\ldots, C^c_q)$, l'ensemble des q composantes fortement connexes du graphe critique .

Notation 1.2 les composantes du graphe critique sont représentées en gras sur le graphe (circuit (3, 3) sur l'exemple).

Définition 1.12 (Cyclicité dans le cas (max, +)) On définit comme suit la cyclicité d'une matrice $M \in (3_{max})^{n\times n}$:

1. La cyclicité d'une composante connexe du graphe critique est égale au pgcd des longueurs des circuits de cette composante;
2. La cyclicité de la matrice M est égale au ppcm des cyclicités des composantes connexes du graphe critique. On la notera c(M).

Théorème 1.7 (cyclicité) Soit $M \in (3_{max})^{n\times n}$ irréductible. Il existe un entier N tel que :

$$k \geq N \Rightarrow M^{k+c(M)} = (\rho(M))^{c(M)}\, M^k.$$

3.2.2.4 Interprétation : régime périodique et taux de production.

Soit un système de production en régime autonome vérifiant l'équation aux dateurs :

$$x(n+1) = Mx(n)\ .$$

On s'intéresse aux valeurs propres et aux vecteurs propres de la matrice M, i.e. les vecteurs $u \in (3_{max})^n \setminus \{\varepsilon\}$ et scalaires $\lambda \in 3_{max}$ tels que $Mu = \lambda u$. Si u est un vecteur propre de M, et si on prend la condition initiale $x_0=u$, on obtient $x(n+1) = \lambda \otimes x(n) = \lambda + x(n)$. Autrement dit une pièce est produite toutes les λ unités de temps : λ est donc l'inverse du taux de production. En outre, le vecteur propre u donne une condition initiale autorisant un fonctionnement périodique du système.

Si M est irréductible, on peut relaxer la condition initiale, en effet le théorème de cyclicité entraîne que le comportement est périodique i.e. :

$$x(k + c) = \rho(M)^c\, x(k)\ ,$$

pour k assez grand. En revenant à l'algèbre usuelle cela s'écrit :

$$x(k + c) = c * \rho(M) + x(k)\ .$$

Autrement dit, c événements se produisent tous les $c\rho(M)$ unités de temps, et $\rho(M)$ est le temps moyen de production d'une pièce (l'inverse du taux de production). On peut donc conclure par le résultat suivant :

un graphe d'événements temporisé fortement connexe fonctionnant en régime autonome atteint, après un régime transitoire fini, un régime périodique. Le taux de production est l'inverse de la valeur propre de la matrice de la dynamique (max, +) du système.

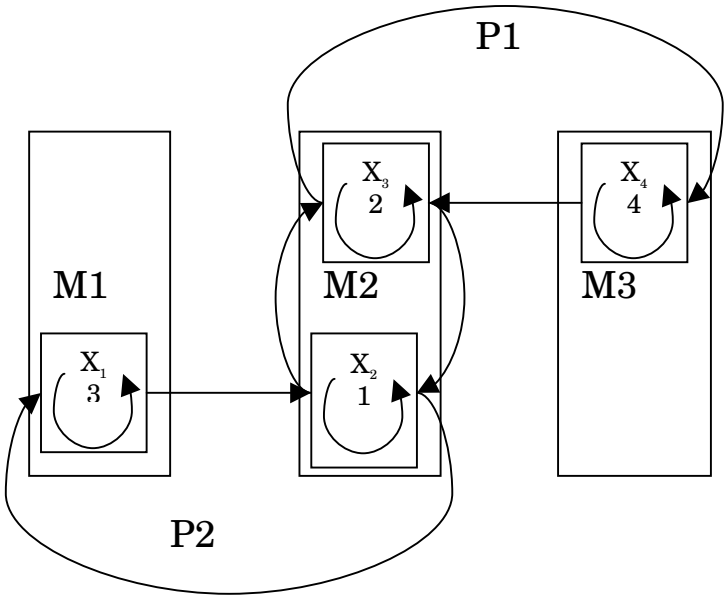
3.2.2.5 Exemple.

On considère un atelier flexible composé de trois machines M1, M2 et M3, qui uϕsinent deux types de pièces P1 et P2. les temps d'usinage en fonction de la pièce et de la machine sont donnés dans le tableau suivant :

	P1	P2
M1	/	3
M2	2	1
M3	4	/

Le fonctionnement de l'atelier respecte les cycles suivants :

- M2 traite P1 puis P2 ;
- P1 est usiné par M3 puis M2 ;
- P2 est usiné par M1 puis M2.



Mise en équation sous forme d'un système max-linéaire :

$$\begin{cases} x_1(k+1) = x_1(k).3 \oplus x_2(k).1 \\ x_2(k+1) = x_1(k+1).3 \oplus x_3(k+1).2 \\ x_3(k+1) = x_2(k).1 \oplus x_4(k+1).4 \\ x_4(k+1) = x_3(k).2 \oplus x_4(k).4 \end{cases},$$

on met ce système sous forme matricielle $x(k+1)=A.x(k+1) \oplus Bx(k)$:

$$x(k+1) = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ 3 & \varepsilon & 2 & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & 4 \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \end{bmatrix} x(k+1) \oplus \begin{bmatrix} 3 & 1 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & 1 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 2 & 4 \end{bmatrix} x(k).$$

En appliquant la résolution des équations implicites on a une solution du type $x = Ax+B$, on a une solution $x(k+1) = A^*B.x(k)$. Reste à calculer A^*, pour cela on remarque que A^* est solution de $x = A.x \oplus I_d$, on peut donc calculer toutes les composantes de la matrice en explicitant le système :

$$\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{14} \\ x_{21} & x_{22} & & \\ \vdots & & \ddots & \vdots \\ x_{41} & \cdots & & x_{44} \end{bmatrix} = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ 3 & \varepsilon & 2 & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & 4 \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \end{bmatrix} . \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{14} \\ x_{21} & x_{22} & & \\ \vdots & & \ddots & \vdots \\ x_{41} & \cdots & & x_{44} \end{bmatrix} \oplus \begin{bmatrix} e & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & e & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & e & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & e \end{bmatrix}.$$

En respectant les règles de calcul de l'algèbre (max,+) et en resolvant variable par variable on obtient les résultats suivants :

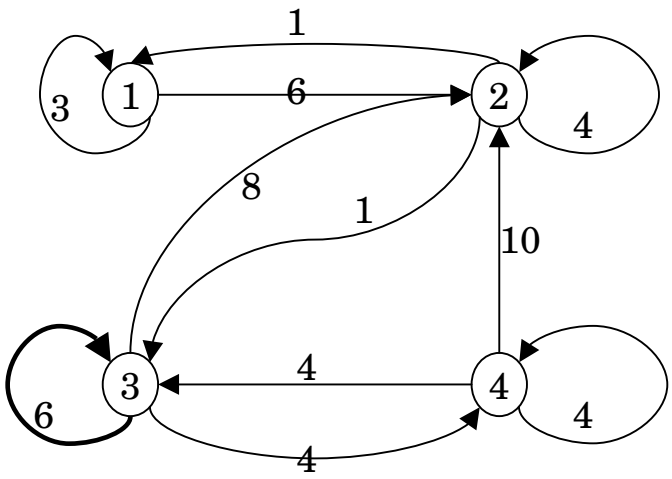
$$\begin{cases} x_{11} = e, \\ x_{12} = x_{13} = x_{14} = \varepsilon \end{cases} et \begin{cases} x_{44} = e \\ x_{41} = x_{42} = x_{43} = \varepsilon \end{cases}$$

$$\begin{cases} x_{31} = 4.x_{41} = \varepsilon \\ x_{32} = 4.x_{42} = \varepsilon \\ x_{33} = e \oplus 4.x_{43} = e \\ x_{34} = 4.x_{44} = 4. \end{cases} et \begin{cases} x_{21} = 3.x_{11} \oplus 2.x_{31} = 3.x_{11,} = 3 \\ x_{22} = e \oplus 3.x_{12} \oplus 2.x_{32} = e, \\ x_{23} = 3.x_{13} \oplus 2.x_{33} = 2.x_{33} = 2, \\ x_{24} = 3.x_{14} \oplus 2.x_{34} = 2.x_{34} = 6. \end{cases}$$

On peut alors écrire A^* et calculer A^*B :

$$A^* = \begin{bmatrix} e & \varepsilon & \varepsilon & \varepsilon \\ 3 & e & \varepsilon & 6 \\ \varepsilon & \varepsilon & e & 4 \\ \varepsilon & \varepsilon & \varepsilon & e \end{bmatrix}, A^*B = \begin{bmatrix} e & \varepsilon & \varepsilon & \varepsilon \\ 3 & e & \varepsilon & 6 \\ \varepsilon & \varepsilon & e & 4 \\ \varepsilon & \varepsilon & \varepsilon & e \end{bmatrix} \begin{bmatrix} 3 & 1 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & 1 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 2 & 4 \end{bmatrix} = \begin{bmatrix} 3 & 1 & \varepsilon & \varepsilon \\ 6 & 4 & 8 & 10 \\ \varepsilon & 1 & 6 & 8 \\ \varepsilon & \varepsilon & 2 & 4 \end{bmatrix}.$$

Ce qui s'interprète par le graphe fortement connexe suivant :



La valeur propre du système est donnée par :

$$\rho(A^*B) = \bigoplus_{i=1}^{n} (\mathrm{tr}\,(A^*B)^{\,i})^{1/i}.$$

Dans ce cas on a le max pour le circuit (3,3),de longueur 1, autrement dit : ρ(A*B)=w(3,3)=6. Ce résultat s'interprète comme un **temps de cycle du système égal à 6 u.t.**

4 Application 2 : l'équation *y = A.x* .

4.1 Résolution de l'équation *y = A.x* .

Dans ce chapitre on étudie la résolution, dans l'algèbre (max, +), de l'équation $y = A.x$, où x et y $\in 3^n$ et A $\in 3^{n\times n}$. Les résultats exposés ici ont été établis par J.M. Prou et E. Wagneur dans 16].

On définit les multiplications max et min, respectivement notée « . » (parfois omis) et « $.^{/}$ » par :

$$(A \,.\, x)_i = \bigvee_{j=1}^{n} a_{ij}\, x_j,$$

$$(A \,.^{/}\, x)_i = \bigwedge_{j=1}^{n} a_{ij}\, x_j \,.$$

On définit aussi les notations suivantes, A^{-} est appelée la résiduée de A:

$$A^{-} = (a_{ij}^{-1}),$$
$$A^{-T} = (A^{-})^T = (a_{ji}^{-1})^T,$$
$$A^{\clubsuit} = (A \,.\, A^{-T}) \,.$$

Théorème 1.8 (Résolution de l'équation y = A.x) Un vecteur $x \in 3^n$ est solution de l'équation $y = A.x$ si et seulement si il vérifie les deux propriétés suivantes :

$$x = A^{-T} .^{/} y \,,$$
$$A.(A^{-T} .^{/} y) = y \,.$$

La dernière propriété peut aussi s'écrire : $A^{\clubsuit} .^{/} y = y$, autrement dit y est un point fixe de $A^{\clubsuit}$.

Théorème 1.9 La matrice $A^{\clubsuit}$ vérifie les deux relations suivantes:

$$A^{\clubsuit} .^{/} A = A \,,$$
$$(A^{\clubsuit})^{\wedge *} = A^{\clubsuit} \,.$$

Le terme $(A^{\clubsuit})^{\wedge *}$ représente l'étoile obtenue en prenant les puissances calculées par min-multiplication.

Démonstration. La première relation se démontre par le calcul du premier terme, en décomposant les différents opérateurs min et max et en utilisant la propriété d'absorption.
Pour montrer la deuxième relation, il faut remarquer que $A^{\clubsuit} = (A^{\clubsuit})^{\wedge *}$ est équivalent à $\forall k \;\; (A^{\clubsuit})^k \leq A^{\clubsuit}$, en particulier pour k = 0 on a $A^{\clubsuit} \geq I$ donc par monotonie il vient $(A^{\clubsuit})^2 \geq A^{\clubsuit}$, or pour k = 2, on a $(A^{\clubsuit})^2 \leq A^{\clubsuit}$, on donc l'égalité : $(A^{\clubsuit})^2 = A^{\clubsuit}$. La réciproque est évidente par récurrence ($(A^{\clubsuit})^2 = A^{\clubsuit} \Rightarrow (A^{\clubsuit})^k = A^{\clubsuit}$ pour tout entier k). On a établi que la proposition est équivalente à $A^{\clubsuit 2} = A^{\clubsuit}$, or pour obtenir cette égalité il suffit de max-multiplier à droite les deux membres de la première relation par A^{-T}.

La première propriété montre que les colonnes de A sont des combinaisons min-linéaires des colonnes de $A^{\clubsuit}$, plus précisément A est un point fixe de $A^{\clubsuit}$.

Théorème 1.10 Si on note $\mathrm{Im}^{\wedge} A^{\clubsuit}$ (resp. $\mathrm{Im}^{\vee} A^{\clubsuit}$) l'espace engendré par les combinaisons min-linéaires (resp. max-linéaires) des colonnes de $A^{\clubsuit}$, on a :

$$Im^{\vee} A^{\clubsuit} \subset Im^{\wedge} A^{\clubsuit}.$$

Démonstration. Il suffit de prouver $\mathrm{Im}^{\vee} A^{\clubsuit} \subset \mathrm{Im}^{\vee} A \subset \mathrm{Im}^{\wedge} A^{\clubsuit}$. La première inclusion vient de la définition de $A^{\clubsuit}$ et de (AB =C $\Rightarrow$ Im C $\in$ Im A). On a la deuxième car $\forall x \in \mathrm{Im}^{\vee}A \Rightarrow \exists y$ tel que $A.y = x$, or d'après le théorème 1.8 ce n'est possible que si x est un point fixe de $A^{\clubsuit}$ donc x $\in$ Im $A^{\clubsuit}$.

Théorème 1.11 $\mathrm{Im}^{\vee} A^{\clubsuit} = \mathrm{Im}^{\wedge} A^{\clubsuit}$ si et seulement si $\mathrm{Im}^{\vee} A^{\clubsuit}$ est un treillis.

On peut de même établir les équations duales en considérant l'équation $x=A.^{/}y$ et en définissant $A^{\blacklozenge} = A.^{/}A^{-T}$, il suffit alors de reprendre les résultats de ce chapitres en remplaçant $A^{\clubsuit}$ par $A^{\blacklozenge}$ et en permutant les opérateurs $\vee$ et $\wedge$.

4.2 Application : atteignabilité et commandabilité.

Une application intéressante de la résolution des équations $y = Ax$ est l'étude de l'atteignabilité et de la commandabilité des systèmes décrits par les équations d'état classiques, où x est l'état du système, u la commande et y la sortie:

$$x(k+1) = Ax(k) + Bu(k+1) ,$$
$$y(k+1) = Cx(k+1).$$

En remplaçant successivement $x(i)$ par son expression dans celle de $x(k+1)$ on obtient:

$$x(k+1) = A^{k+1}x(0) + A^{k}Bu(1) + A^{k-1}Bu(2) + \ldots + Bu(k+1),$$

en écrivant cette relation sous forme matricielle on obtient:

$$x(k+1) = \begin{bmatrix}\Gamma_k & A^{k+1}\end{bmatrix}\begin{bmatrix} u_k \\ x(0)\end{bmatrix}, \text{ où } \Gamma_k = \begin{bmatrix}B & AB & A^2B & \ldots & A^kB\, A^{k+1}\end{bmatrix} \text{ et } u_k = \begin{bmatrix} u(k+1) \\ u(k) \\ \ldots \\ u(1)\end{bmatrix}.$$

On est donc ramené au problème précédent car étudier l'atteignabilité ou la commandabilité revient à chercher des solutions en $[u_k\ x(0)]^T$, autrement dit à chercher un état initial, et une commande permettant d'atteindre l'état $x(k+1)$ en k+1 étapes. Si le problème a une solution on dit que x(k+1) est atteignable, si de plus on peut prendre $x(0)$ = [0] on dit l'état commandable, dans ce cas le système devient : $x(k+1) = [\Gamma_k][u_k]$. D'après le théorème 1.8 il suffit de vérifier que $x(k+1)$ est une combinaison max-linéaire des colonnes de $[\Gamma_k\ A^{k+1}]^{\clubsuit}$.

Exemple.

L'état $X = \begin{bmatrix}11 & 14\end{bmatrix}^T$ est-il atteignable, commandable par le système défini par $A = \begin{bmatrix}1 & \varepsilon \\ 3 & 2\end{bmatrix}, B = \begin{bmatrix}1 \\ 3\end{bmatrix}$?

On recherche dans un premier temps si X est atteignable en une étape, on a $[\Gamma_0\ A] = [B\ A]$ et $[\Gamma_0\ A]^{\clubsuit} = \begin{bmatrix} e & 2^{-1} \\ \varepsilon & e\end{bmatrix}$, et $\lambda\begin{bmatrix} e \\ \varepsilon\end{bmatrix} \wedge \mu\begin{bmatrix} 2^{-1} \\ e\end{bmatrix} = \begin{bmatrix}11 \\ 14\end{bmatrix}$, pour $\lambda = 11$ et $\mu = 14$, donc X est atteignable et $\begin{bmatrix} u(2) \\ x(1)\end{bmatrix} = \begin{bmatrix}\Gamma_0 & A\end{bmatrix}^{-T}.^{/}\begin{bmatrix}11 \\ 14\end{bmatrix} = \begin{bmatrix}10 & 10 & 12\end{bmatrix}^T$, autrement dit, X est

accessible depuis $X(1) = [10\ 12]^T$ avec la commande $u(1) = 10$. On peut se demander si X est commandable en deux étapes on prend alors $x(0) = 0$: $\begin{bmatrix}11\\14\end{bmatrix} = [\Gamma_1]\begin{bmatrix}u(2)\\u(1)\end{bmatrix} = \begin{bmatrix}1 & 2\\3 & 5\end{bmatrix}\begin{bmatrix}u(2)\\u(1)\end{bmatrix}$, or $[\Gamma_1]^* = \begin{bmatrix}e & 2^{-1}\\3 & e\end{bmatrix}$ et on vérifie facilement que $X \in Im\ \Gamma_1^*$ on a donc $\begin{bmatrix}u(2)\\u(1)\end{bmatrix} = [\Gamma]^{-T} \cdot \begin{bmatrix}11\\14\end{bmatrix} = \begin{bmatrix}1^{-1} & 3^{-1}\\2^{-1} & 5^{-1}\end{bmatrix}\begin{bmatrix}11\\14\end{bmatrix} = \begin{bmatrix}10\\9\end{bmatrix}$, X est donc commandable avec $u(1) = 9$ et $u(2) = 10$.

Conclusion.

Outre l'intérêt théorique des structures algébriques de dioïdes et de treillis, le but de ce chapitre est de présenter le cadre mathématique utile aux différentes approches d'étude des SED. Les algèbres (min,+) et (max,+) sont surtout dédiées aux RdP temporisés et plus particulièrement aux Graphes d'Evènements Temporisés (qui font l'objet du chapitre suivant) mais on retrouve par exemple l'opérateur étoile dans la théorie des automates et des langages, de même que la structure de treillis permet d'assurer l'existence d'éléments suprémaux pour le calcul d'automates superviseurs.

Chapitre 2

Les Graphes d'Evénements Temporisés (GET).

On présente dans ce chapitre les réseaux de Petri, langage graphique permettant de décrire les systèmes à événements discrets . Le lecteur peut compléter la présentation en se reportant au livre de R. David et H. Alla [7]. On s'attardera ici sur une sous-classe de réseaux de Petri : les graphes d'événements temporisé, c'est à dire où le temps est intégré dans la structure du réseau. Cette sous-classe présente l'intérêt de pouvoir être décrite par des équations récurrentes (max,+) linéaires sur les dateurs, ou (min,+) linéaires sur les compteurs. On présentera aussi les transformées en γ et δ qui permettent d'exploiter plus commodément les équations aux compteurs et dateurs, en particulier au moyen de la représentation graphique dans le plan (γ, δ). On pourra se reporter pour plus de précisions à la thèse de B. Cottenceau [6].

1 Introduction aux réseaux de Petri.

Les Réseaux de Petri (RdP) sont des graphes orientés, i.e. sont définis par un ensemble de sommets S et un ensemble A d'éléments, arcs ou arêtes, de S^2. Les RdP sont des graphes avec deux types de sommets, les places (représentées par des cercles) et les transitions (représentées par des rectangles). S se partitionne selon : $S = T \cup P$ où P représente l'ensemble des places et T celui des transitions. Les arcs orientés relient les transitions aux places et les places aux transitions, mais deux sommets d'un même type ne peuvent être directement reliés. Etant donné un sommet s on note $\Gamma(s)$, l'ensemble des successeurs de s, et $\Gamma^{-1}(s)$, l'ensemble de ses prédécesseurs. A chaque arc a est associé un poids n_a. à chaque place p_i est associé un marquage $m_i \in \angle$ qui représente le nombre de jetons présents dans la place, on définit ainsi $M \in \angle^n$, le vecteur de marquage dont chaque composante i donne le marquage de p_i.

1.1 Définition des RdP.

Plus rigoureusement un RdP est défini par un quadruplet $<P,T,Pre,Post>$, où Pre est une application $P \times T \to \angle$ dite d'incidence (avant) et Post une application $P \times T \to \angle$ dite d'incidence arrière. On a la représentation graphique en opérant comme suit : si $Pre(P, T) > 0$, on trace un arc de P à T, et $n_{(P,T)}=Pre(P,T)$, si $Post(P, T) > 0$, on trace un arc de T à P, et $n_{(T,P)}=Post(T,P)$.

1.2 Evolution des RdP.

Une transition est tirable (ou franchissable) si $\forall P \in \Gamma^{-1}(T)$, $M(P) \geq n_{(P,T)}$. Si la transition est franchie, le marquage M devient M', avec les règles suivantes :

si $P \in \Gamma^{-1}(T)$, $P \notin \Gamma(T)$, $M'(P) = M(P) - n_{(P,T)}$,
si $P \notin \Gamma^{-1}(T)$, $P \in \Gamma(T)$, $M'(P) = M(P) + n_{(T,P)}$,
si $P \in \Gamma^{-1}(T) \cap \Gamma(T)$, $M'(P) = M(P) + n_{(T,P)} - n_{(P,T)}$,
Sinon, $M'(P) = M(P)$.

Autrement dit le franchissement donne le nouveau marquage M' :

$$M'(P) = M(P) - Pre(P,T) + Post(P,T).$$

(la présentation est très brève, pour avoir des informations complémentaires le lecteur intéressé pourra se reporter à *[7]*)

2 Les GET.

2.1 Définition et propriétés des graphes d'événements temporisés.

Un graphe d'événements est un RdP tel qu'en chaque place il n'entre et ne sort qu'un seul arc, de plus tous les arcs ont un poids unitaire. Cette configuration permet de modéliser les phénomènes de synchronisation (plusieurs places aboutissant sur une même transition), mais en revanche n'est pas utilisable pour l'interprétation de concurrence ou de conflit, par exemple quand un jeton venant d'une transition peut servir à plusieurs places.

Proposition 2.1 Pour un graphe d'événements, la somme des marquages des places d'un circuit est constante.

Définition 2.1 (Vivacité) On dit qu'un graphe est vivant si et seulement si pour tout marquage accessible M et toute transition T, il existe une séquence de franchissements à partir de M contenant T.

Proposition 2.2 Un graphe d'événements est vivant si et seulement il ne contient pas de circuit sans jeton.

On introduit le caractère temporel en associant une durée aux places, elle représente le temps de séjour minimal et obligatoire de chaque jeton (on peut de manière équivalente associer le temps aux transitions).

Exemple 2.1

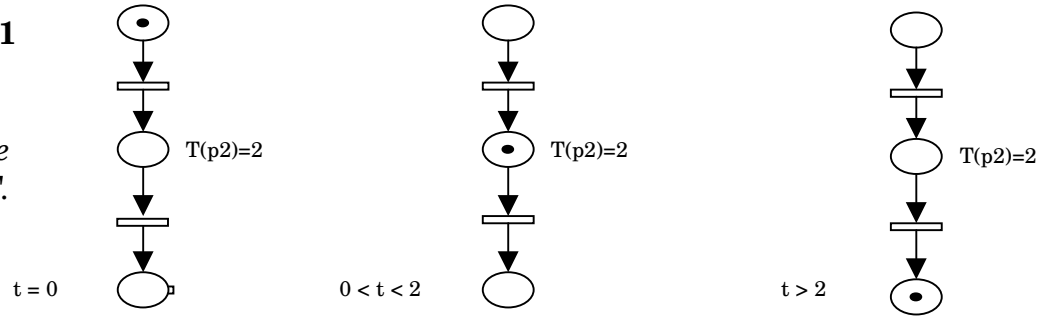


Fig. Exemple de GET.

2.2 Mise en équation des GET.

On peut considérer le GET de deux points de vue :

- événementiel, on étudie $x_i(n)$, fonction non décroissante, qui représente la date du n^{ième} franchissement de la transition i, pour cette approche on se place dans 9_{max} .
- temporel, on étudie $x_i(t)$, fonction non décroissante, qui représente le nombre de franchissements de la transition i à l'instant t, pour cette approche on se place dans 9_{min}.

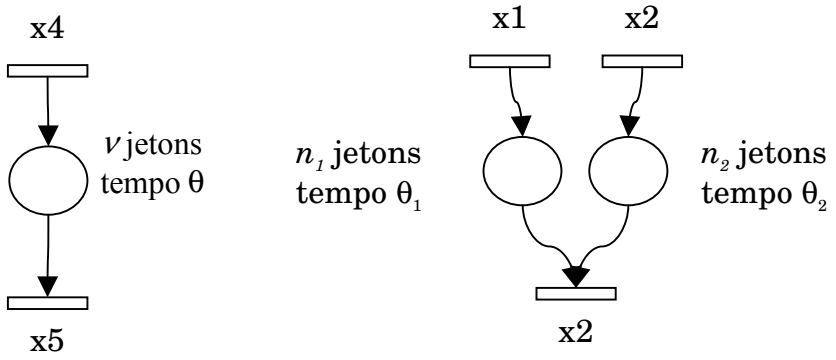


Fig. GET élémentaires

Equations aux compteurs :
$x_5(t) \leq x_4(t - \theta) + \nu,$
$x_3(t) \leq min(x_1(t-\theta_1)+n_1,\ x_2(t-\theta_2)+n_2).$

Equations aux dateurs :
$x_5(k) \geq x_4(k-\nu) + \theta,$
$x_3(k) \geq max(x_1(k-n_1)+ \theta_1,\ x_2(k-n_2)+ \theta_2).$

Dans les algèbre (min,+) et (max,+) on a les expressions suivantes:

Equations aux compteurs :
$x_3(t) \leq n_1 \otimes x_1(t-\theta_1) \oplus n_2 \otimes x_2(t-\theta_2).$

Equations aux dateurs :
$x_3(k) \geq \theta_1 \otimes x_1(k-n_1) \oplus \theta_2 \otimes x_2(k-n_2).$

Les fonctions dateurs au plus tôt du GET sont donc solutions d'un système implicite dans le dioïde 3_{max} de la forme :

$$x_n \geq \bigoplus_{i=0}^{n} A_i\, x(n-i) \oplus \bigoplus_{j=0}^{b} B_j\, u(n-j),$$

$$y_n \geq \bigoplus_{i=0}^{n} C_i\, x(n-i).$$

où les $u(n)$ sont les vecteurs d'entrée du système et $x(n)$ les vecteurs d'état. En utilisant les résultats sur la résolution d'équations implicites on a:

$$x_n = \bigoplus_{i=0}^{n} \underline{A}_i\, x(n-i) \oplus \bigoplus_{j=0}^{b} \underline{B}_j\, u(n-j),$$

$$y_n = \bigoplus_{i=0}^{n} C_i\, x(n-i).$$

où $\underline{A_i} = A_n^* A_i$ et $\underline{B_i} = B_n^* B_i$, de plus la solution minimale de $x \geq ax \oplus b$ vérifie l'égalité, donc le fonctionnement "au plus tôt" réalise l'égalité.
En augmentant l'état on peut mettre cette équation sous la forme d'état:

$$x(t) = Ax(t-1) \oplus Bu(t), y(t) = Cx(t).$$

Exemple 2.2
Considérons le GET suivant:

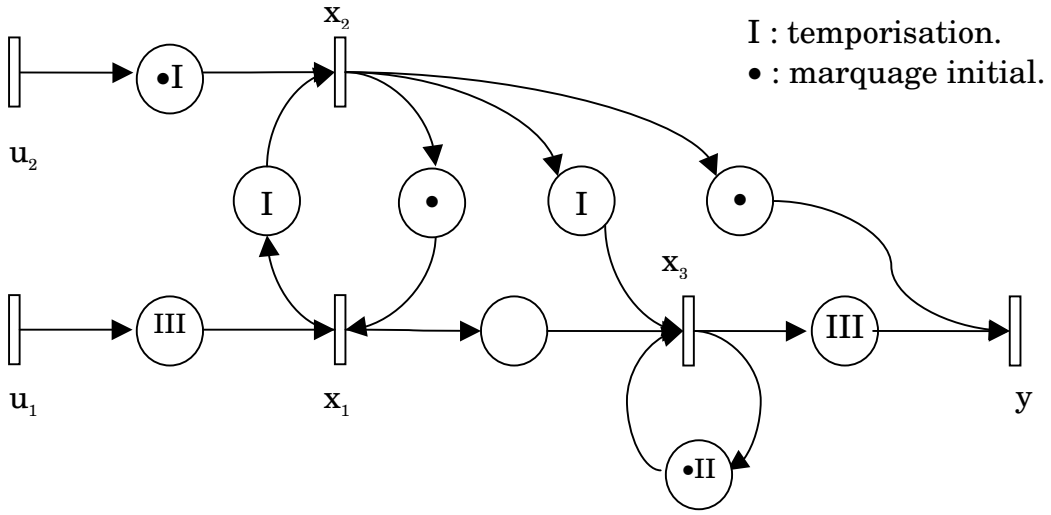


Fig. Exemple de GET.

Mise en équation:

$$x_1(n) \geq 3u_1(n) \oplus x_2(n-1),$$
$$x_2(n) \geq 1u_2(n-1) \oplus 1x_1(n),$$
$$x_3(n) \geq x_1(n) \oplus 1x_2(n) \oplus 2x_3(n-1),$$
$$y(n) \geq 3x_3(n) \oplus x_2(n-1).$$

On a sous forme matricielle :

$$X(n) \geq \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon \\ 1 & \varepsilon & \varepsilon \\ e & 1 & \varepsilon \end{bmatrix} X(n) \oplus \begin{bmatrix} \varepsilon & e & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 2 \end{bmatrix} X(n-1) \oplus \begin{bmatrix} 3 & \varepsilon \\ \varepsilon & 1 \\ \varepsilon & \varepsilon \end{bmatrix} \begin{bmatrix} u(n) \\ u(n-1) \end{bmatrix},$$

$$Y(n) \geq \begin{bmatrix} \varepsilon & \varepsilon & 3 \end{bmatrix} X(n) \oplus \begin{bmatrix} \varepsilon & e & \varepsilon \end{bmatrix} X(n-1).$$

C'est à dire un système de la forme :

$$x(n) \geq A_0 x(n) \oplus A_1 x(n-1) \oplus B_0 u(n) \oplus B_1 u(n-1),\ y(n) \geq C_0 x(n) \oplus C_1 x(n-1).$$

L'étude du comportement au plus tôt permet d'établir les égalités.

3 Représentation dans $M_{in}^{ax}[[\gamma, \delta]]$.

3.1 Transformées en γ et en δ.

On introduit ici les transformées en γ et en δ qui sont des opérateurs permettant un décalage respectivement événementiel et temporel, à l'aide desquels on exprime une fonction sous la forme d'une série entière.

Transformée en γ : l'opérateur γ effectue un décalage en numérotation, on a : $\gamma d(k) = d(k-1)$, pour toute fonction dateur $d(k)_{k\in 9}$ on peut écrire :

$$D(\gamma)=\oplus_{k\in 9} d(k)\gamma^k.$$

Définition 2.2 (Stationnarité) Un système est dit stationnaire s'il commute avec γ, i.e. $S\gamma = \gamma S$ (où S est un système (max,+) linéaire).

Par exemple, avec cette notation les équations d'état deviennent :

$$x = A\gamma x \oplus Bu,$$
$$y = Cx.$$

Transformée en δ : l'opérateur δ effectue un décalage temporel, autrement dit : $\delta c(t) = c(t-1)$, pour toute fonction compteur $c(k)_{k\in 9}$ on peut écrire :

$$C(\delta)=\oplus_{t\in 9} c(t)\delta^t.$$

En utilisant ces deux transformées on peut écrire la forme d'état de l'exemple précédent :

$$X=\begin{bmatrix} \varepsilon & \gamma & \varepsilon \\ \delta & \varepsilon & \varepsilon \\ e & \delta & \gamma\delta^2 \end{bmatrix} X \oplus \begin{bmatrix} \delta^3 & \varepsilon \\ \varepsilon & \gamma\delta \\ \varepsilon & \varepsilon \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix},\ Y=\begin{bmatrix} \varepsilon & \gamma & \delta^3 \end{bmatrix} X.$$

3.2 Le Dioïde $M_{in}^{ax}[[\gamma,\delta]]$.

On cherche à coder les trajectoires de GET en intégrant les équations aux compteurs et celles aux dateurs, pour cela on définit des séries formelles des deux variables commutatives γ et δ, à exposant dans 9 et à coefficients booléens, on note ce dioïde $M_{in}^{ax}[[\gamma,\delta]]$, avec l'élément neutre $e = \gamma^0\delta^0$, l'élément absorbant $\varepsilon = \gamma^{+\infty}\delta^{\infty}$.

3.2.1 Les dioïdes $\overline{\mathbf{Z}}_{\min}[[\delta]]$ et $\overline{\mathbf{Z}}_{\max}[[\gamma]]$.

Un ensemble de séries formelles à coefficients sur un dioïde complet peut également être muni d'une structure de dioïde complet dont la loi ⊕ est la somme de séries formelles et la loi ⊗ est le produit de séries formelles. Les transformées en γ (resp. δ) des dateurs (resp. compteurs) peuvent donc être considérées comme appartenant à des dioïdes de séries formelles en γ (resp. δ).
Nous noterons $\overline{\mathbf{Z}}_{\max}[[\gamma]]$ le dioïde complet des séries formelles en γ à coefficients dans $\overline{\mathbf{Z}}_{\max}$ et exposants dans 9, et $\overline{\mathbf{Z}}_{\min}[[\delta]]$ le dioïde des séries formelles en δ à coefficients dans $\overline{\mathbf{Z}}_{\min}$ et exposants dans 9. On rappelle que l'élément neutre de l'addition dans $\overline{\mathbf{Z}}_{\max}[[\gamma]]$ est la série $\varepsilon(\gamma) = \oplus_{k\in 9}\varepsilon\gamma^k$ (où ε=-∞ est l'élément neutre de

l'addition dans $\overline{\mathbf{Z}}_{\max}$) et l'élément neutre de la multiplication est la série $e(\gamma) = e\gamma^{\theta}$ (où $e = 0$ est l'élément neutre de la multiplication de $\overline{\mathbf{Z}}_{\max}$).

Ces dioïdes permettent de coder tout type de trajectoires de GET, or compteurs comme dateurs sont des fonctions monotones (le k+1$^{\text{ième}}$ tir d'une transition est toujours postérieur au k$^{\text{ième}}$). Or sur $\overline{\mathbf{Z}}_{\min}[[\delta]]$ l'ordre est inverse de l'ordre usuel, on écrit donc la monotonie des variables par:

$$\forall k,\ t \in \mathbb{9},\ d(k) \geq d(k-1) \quad \text{et} \quad c(t+1) \geq c(t).$$

Les variables compteurs sont donc monotones non décroissantes sur $\overline{\mathbf{Z}}_{\max}$ et les variables dateurs sont monotones non croissantes sur $\overline{\mathbf{Z}}_{\min}$, et vérifient respectivement:

$$d(k) = d(k-1) \oplus d(k) \text{ sur } \overline{\mathbf{Z}}_{\max},$$
$$c(t) = c(t+1) \oplus c(t) \text{ sur } \overline{\mathbf{Z}}_{\min}.$$

2.3.2 Définition de $M_{in}^{ax}[[\gamma,\delta]]$.

On note $\mathbb{B}[[\gamma,\delta]]$ le dioïde des séries formelles des variables commutatives γ et δ à exposants dans $\mathbb{9}$ et à coefficients dans le dioïde de Boole $\mathbb{B}$. D'après les deux égalités précédentes on a que tout élément de $\mathbb{B}[[\gamma,\delta]]$ vérifie :

$$X(\gamma,\delta) \geq (\gamma \oplus \delta^{-1})X(\gamma,\delta)$$

On définit donc $M_{in}^{ax}[[\gamma,\delta]]$ par $(\gamma \oplus \delta^{-1})\mathbb{B}[[\gamma,\delta]]$, où un élément $X(\gamma,\delta)$ ne définit plus un point mais un cône Sud-Est de sommet ce point.

Règles de calcul et de simplification sur $M_{in}^{ax}[[\gamma,\delta]]$.

- $\gamma^n\delta^t \oplus \gamma^{n'}\delta^t = \gamma^{\min(n,\,n')}\delta^t$;
- $\gamma^n\delta^t \oplus \gamma^n\delta^{t'} = \gamma^n\delta^{\max(t,\,t')}$;
- $\gamma^n\delta^t \wedge \gamma^{n'}\delta^{t'} = \gamma^{\max(n,\,n')}\delta^{\min(t,\,t')}$.

On peut traduire cela graphiquement par:

- la somme $\oplus$, (ou le sup) de deux monômes est représentée par l'union des cônes correspondant à chaque monôme;
- le produit $\otimes$ de deux monômes $\gamma^n\delta^t$ et $\gamma^{n'}\delta^{t'}$ est $\gamma^{n+n'}\delta^{t+t'}$, et est représenté par le cône de sommet $(n+n',\ t+t')$, soit la somme vectorielle des sommets correspondant aux deux monômes);
- l'inf de deux monômes est représenté par l'intersection des cônes correspondant à chaque monôme.

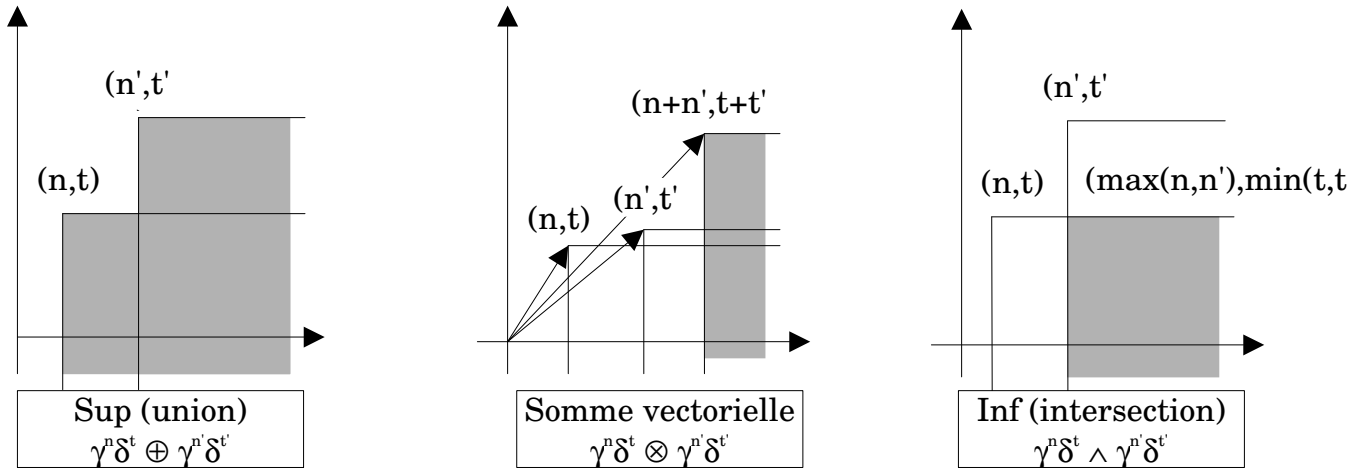


Fig. Interprétation graphique des opérations dans $M_{in}^{ax}[[\gamma,\delta]]$.

Exemple 3.1

On reprend le graphe de la figure et on obtient les équation suivantes:

$$X_1 = \gamma X_2 \oplus \delta^3 U_1,$$
$$X_2 = \delta X_1 \oplus \gamma\delta U_2,$$
$$X_3 = X_1 \oplus \delta X_2 \oplus \gamma\delta^3 X_3,$$
$$Y = \gamma X_2 \oplus \delta^3 X_3.$$

En remplaçant X_1 par son expression dans celle de X_2, on obtient:
$X_2 = \gamma\delta X_2 \oplus \delta^4 U_1 \oplus \gamma\delta U_2$, soit en résolvant en X2 : $X2 = (\gamma\delta)^* (\delta^4 U_1 \oplus \gamma\delta U_2)$.

De manière analogue on résout l'équation implicite en X_3 avant d'injecter les expressions de X_1, X_2 .

$$X_3 = (\gamma\delta^3)^*(X_1 \oplus \delta X_2),$$
$$X_3 = (\gamma\delta^3)^*(\delta(\gamma\delta)^* (\delta^4 U_1 \oplus \gamma\delta U_2) \oplus \gamma(\gamma\delta)^* (\delta^4 U_1 \oplus \gamma\delta U_2) \oplus \delta^3 U_1),$$
$$X_3 = (\gamma\delta^3)^*(\delta^5 U_1 \oplus \gamma\delta^3 \oplus \gamma\delta^4 U_1 \oplus \gamma^2\delta U_2 \oplus \delta^3 U_1),$$
$$X_3 = (\gamma\delta^3)^*(\delta^5 U_1 \oplus \gamma\delta^3).$$

En reportant X_2 et X_3 dans Y, il vient :

$$Y = (\gamma\delta^3)^*(\delta^3 U_1 \oplus \gamma\delta^5 U_2) \oplus (\gamma\delta)^* (\gamma\delta^4 U_1 \oplus \gamma^2\delta U_2),$$
$$Y = (\gamma\delta^3)^*(\delta^3 U_1 \oplus \gamma\delta^5 U_2).$$

On a alors une expression simplifiée de Y en fonction des deux entrées U_1 et U_2 qui peut se traduire par le GET ci-dessous, bien plus simple que celui de la figure initiale.

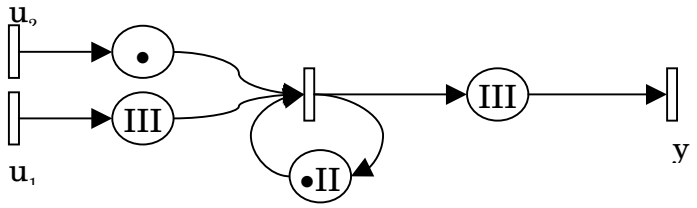


Fig. GET simplifié.

Il apparaît alors un comportement périodique (identique vis à vis des deux entrées) contenu dans le terme $(\gamma\delta^2)^*$, qui se traduit par un circuit élémentaire, on caractérisera par la suite la périodicité.

2.3.3 Réalisabilité, rationalité et périodicité.

Définition 2.3 (Causalité) Une série $s \in M_{in}^{ax}[[\gamma,\delta]]$ est dite causale si $s = \varepsilon$ (la série est nulle) ou si son représentant minimal est à exposants dans $\angle$. Une matrice est dite causale si toutes ses composantes sont causales.

Ainsi tout élément de $M_{in}^{ax}[[\gamma,\delta]]$ est causal si les sommet de son représentant graphique appartiennent au cadran Nord-Est du plan 9^2. De plus l'ensemble des séries causales de $M_{in}^{ax}[[\gamma,\delta]]$est stable par la somme et le produit de $M_{in}^{ax}[[\gamma,\delta]]$.

Définition 2.4 (**Rationalité**) Un élément $s \in M_{in}^{ax}[[\gamma,\delta]]$ est dit rationnel si son représentant minimal appartient à la clôture rationnelle de l'ensemble {γ,δ,e,ε}, c'est à dire peut s'écrire avec un nombre fini de sommes, produits et étoiles d'éléments de l'ensemble {γ,δ,e,ε}. Une matrice est dite rationnelle si toutes ses composantes sont rationnelles.

Définition 2.5 (**Réalisabilité**) Une matrice $H \in M_{in}^{ax}\left[[\gamma,\delta]\right]^{p\times m}$ est dite réalisable si il existe quatre matrices A_1, A_2, A_3 et A_4 de taille respective $n\times n$, $n\times n$, $n\times m$ et $p\times n$ à coefficients dans l'ensemble {ε,e} telles que $H = C(\gamma A_1 \oplus \delta A_2)^* B$.

Définition 2.6 (Périodicité) Une série s est dite périodique s'il existe deux polynômes p et q de $M_{in}^{ax}[[\gamma,\delta]]$où $p=\bigoplus_{i=0}^{\lambda}\gamma^{n_i}\delta^{t_i}$ et $q=\bigoplus_{j=0}^{\mu}\gamma^{N_j}\delta^{T_j}$ et un monôme causal $r = \gamma^{\gamma}\delta^{\tau}$ tels que :

$$s = p \oplus qr^*.$$

Dans cette expression le polynôme p représente le transitoire et q le motif du régime périodique.

Théorème 2.1 Soit $H \in M_{in}^{ax}\left[[\gamma,\delta]\right]^{p\times m}$. Les propositions sont équivalentes :

- H est réalisable;
- H est rationnelle;
- H est périodique et causale.

Conclusion

La richesse des résultats de l'étude des GET a motivé une abondante littérature sur l'analyse et la commande des GET. C'est sans doute parmi les recherches sur les SED la théorie la plus développée, avec celle de Ramadge et Wonham concernant les automates. La principale limite de cet outil est d'ordre structurel. En effet, on ne s'intéresse qu'aux RdP tels que chaque place n'est reliée, au plus, à une transition en amont et en aval. Cette contrainte interdit la structure de

choix ou de conflit, par exemple les structures typiques de partages de ressource(s) ou d'exclusions mutuelles. Ainsi notre exemple du réseau ferroviaire n'est pas facilement modélisable par un GET car le tronçon commun est une ressource pour laquelle une exclusion mutuelle est requise (le but étant d'interdire la présence simultanée des deux trains sur le même tronçon). La possibilité pour le tronçon C d'être parcouru par l'un ou l'autre des trains se traduit par un jeton qui peut prendre deux trajectoires différentes. Autrement dit une place est suivie de plusieurs transitions, on n'est donc plus dans le cas d'un GET, mais dans celui, plus général d'un RdP temporisé.
En somme, la richesse des résultats concernant les GET se paye par une puissance de modélisation moindre. C'est ce qui a motivé l'étude des RdP Temporisés, qui font l'objet du chapitre suivant, pour lesquels la condition structurelle est relaxée.

Chapitre 3

Les Réseaux de Petri Temporisés (RdPT).

On présente dans ce chapitre les Réseaux de Petri Temporisés (RdPT), cette interprétation des RdP peut être vue comme une généralisation des GET, en effet tout en gardant le caractère temporel on relaxe la condition structurelle, i. e. les nombres de transitions amont et aval de chaque place ne sont pas limités à l'unité. Après avoir présenté et défini le modèle des RdPT, le but de notre étude est de caractériser les fonctions qui peuvent coder le fonctionnement des RdPT, d'en donner une représentation d'état et enfin de s'intéresser à un fonctionnement particulier : le fonctionnement "dès que possible". Ce chapitre reprend entre autre les résultats établis par L. Libeau et J.J. Loiseau dans [7].

1 Définition et règles d'évolution.

On définit comme suit les matrices W^+ et W^-, les matrices d'incidence avant et arrière:

$$W_{ij}^{+} = \begin{cases} w(\tau_j, p_i)\ si\ (\tau_j, p_i) \in F \\ 0\ autrement \end{cases},$$

$$W_{ij}^{-} = \begin{cases} w(p_i, \tau_j)\ si\ (p_i, \tau_j) \in E \\ 0\ autrement \end{cases},$$

où E et F sont les ensembles d'arcs reliant respectivement les places aux transitions et les transitions aux places. Le couple (W^+, W^-) détermine complètement un RdP dans la mesure où un nouveau marquage $m \in \angle^n$ peut être donné en fonction d'un précédent si on connaît celui-ci ainsi que les tirs de transitions effectués. Si $\pi_0 \in \angle^n$ est le marquage initial et si la composante x_i d'un vecteur x donne le nombre de tirs de la transition i, on a la relation suivante:

$$m = \pi_0 + (W^+ - W^-)x\ .$$

On sait que le marquage ne peut être strictement négatif, on a donc l'inégalité : $m = \pi_0 + (W^+ - W^-)x \geq 0$, néanmoins cette condition n'implique pas que x soit un vecteur de tirs. Dans le cas des RdP acycliques (un RdP est dit acyclique s'il n'existe pas de chemin d'une transition à elle même) on a le résultat suivant:

lemme 3.1 soit G un RdP acyclique, $x \in \angle^q$ est un vecteur de tirs si et seulement si $\pi_0 + (W^+ - W^-)x \geq 0$.

On introduit le temps en attribuant un délai à chaque place (on peut de manière équivalente l'attribuer aux transitions), cette durée correspond au temps de séjour minimal d'un jeton dans la place.

Définition 3.1 (Reseau de Petri Temporisé) Un RdPT est défini par un quadruplet $<G, \pi_0, \nu, T>$ où :

- $<G, \pi_0>$ est un réseau de Petri,
- ν est un vecteur non négatif de dimension n où chaque composante ν_i est égale au délai attribué à la place p_i (si les délais sont constants le RdPT est dit déterministe, si les ν_i sont des variables aléatoires il est dit stochastique),
- T est un ensemble totalement ordonné par la relation ≤ ; les éléments de T, appelés instants sont les dates de tir d'une ou plusieurs transitions (T peut être par exemple 3^+ ou $\angle^+$).

Règles d'évolution d'un RdPT.
Après son arrivée dans une place p_i, un jeton reste indisponible durant une durée égale à ν_i, ensuite il devient disponible. Seuls les jetons disponibles sont pris en compte pour la validation des transitions aval. Le franchissement des transitions obéit aux règles usuelles des RdP, à ceci près que seuls les jetons disponibles sont déplacés. Le tir d'une transition est considéré comme instantané, autrement dit s'il est initialisé à $t \in$ T, il se terminera à t.

On voit que les tirs de transitions ne peuvent survenir qu'à certains instants $t_k \in T$, $k \in \angle$, qui forment un ensemble discret $S = \{t_0, t_1, \ldots, t_k, \ldots\}$. On suppose que S est, soit fini, soit infini et $sup(S) = +\infty$. De plus une séquence de t_k est non décroissante (autrement dit on n'a pas de repliement du temps, on n'a donc pas un nombre infini de tirs durant un intervalle fini de temps).

2 Mise en équation.

On considère ici un RdPT où $\nu \in (3^+)^n$ est un vecteur constant et $T = 3^+$. Pour décrire le fonctionnement du RdPT on définit deux variables temporelles. Le vecteur de tir, θ(t), est une fonction de 3 dans $\angle^q$, telle que sa $j^{ième}$ composante soit le nombre de tirs de la transition τ_j à l'instant t. Le vecteur de marquage, $m(t) \in \angle^n$ est égal au marquage à l'instant t. Par convention, les jetons commencent à être déplacés à $t = 0$, autrement dit on a pour tout $t<0$, $\theta(t) = 0$ et $m(t) = \pi_0$. De plus on considère que θ est constant par morceaux et que les points de discontinuité sont les t_k, autrement dit on a $\forall t \in [t_k\ t_{k+1}[$, $\theta(t) = \theta(t_k)$.

Sans tenir compte du caractère temporisé du RdP, c'est à dire sans tenir compte des délais associés aux places, on peut écrire que le marquage est instantanément déterminé par la séquence de tirs, soit:

$$m(t) = \pi_0 + (W^+ - W)\theta(t) \geq 0,$$

mais cette inégalité n'assure pas que θ(t) soit un vecteur de tir, même dans le cas d'un RdP acyclique, car les délais ne sont pas pris en compte. Pour pallier cet inconvénient on définit $\tilde{m}$ fonction de 3 dans $\angle^n$, telle que chaque composante $\tilde{m}_i(t)$ corresponde au nombre de jetons disponibles à l'instant t dans la place p_i. De

plus on pose : $\tilde{m}(t)=\pi_0$ pour tout $t < 0$, autrement dit tous les jetons sont disponibles à l'instant initial. On peut alors déduire une relation vérifiée par $\tilde{m}$ traduisant que le marquage est déduit de $\pi_{0,}$ en prenant en compte les jetons disponibles dans chaque place amont p_i à $t - \nu_i$. Le terme $W^+\theta(t)$ devient alors $W^+\theta(t-\nu_i)$ (on note que pour $\nu_i = 0$, on retrouve le cas des RdP non temporisés), et on a pour chaque place :

$$\tilde{m}_i(t)=\pi_0 + \sum_{\tau_k\in\Gamma^{-1}(p_i)} w(\tau_k,p_i)\theta_k(t-\nu_i) - \sum_{\tau_l\in\Gamma(p_i)} w(p_l,\tau_k)\theta_l(t),$$

qui devient en utilisant la notation matricielle:

$$\tilde{m}_i(t) = \pi_{0i} + [W^+\theta(t-\nu_i)-W^-\theta(t)]_i.$$

En écrivant la positivité du marquage disponible on a le théorème suivant, où S est le sous-RdP déduit de G en supprimant tous les arcs menant à une place p_i telle que $\nu_i > 0$ (ce sous-RdP non temporisé donne le comportement instantané du RdPT, et son caractère acyclique garantit de ne pas avoir de boucle infiniment parcourue en un intervalle de temps fini).

Théorème 3.1 Soit (G, π_0, ν, 3^+) un RdPT tel que S soit acyclique. Une fonction non-décroissante θ de 3 dans $\angle^q$ correspond à une évolution du réseau si et seulement si, pour tout $t \in 3$, et pour toute place p_i on a :

$$\pi_{0i} + [W^+\theta(t - \nu_i) - W^-\theta(t)] > 0.$$

3 Forme d'état des RdPT.

On peut établir une mise en équation sous forme d'état, pour cela on suppose que les délais ν_i sont constants, non négatifs et finis, de plus le sous-réseau S tel que défini plus avant est toujours acyclique. On définit ici les matrices $W^+(k)$ par :

$$(W^+(k))_{ij} = \begin{cases} W^+_{ij} \; si \; \nu_i = k, \\ 0 \; autrement \end{cases}.$$

Le nombre de places étant fini on a une borne supérieure des délais, notée K, on a alors $W^+ = \sum_{i=0}^{K} W^+(i)$, on peut alors écrire le théorème 3.1 de la manière suivante: $\pi_0 + \sum_{i=0}^{K} W^+(i)\theta(t-i) - W^-\theta(t) \geq 0$. On peut ne faire apparaître qu'un seul terme dans la somme en définissant les deux vecteurs $\bar{\theta}(t) = [\theta(t) \quad \theta(t-1) \quad \ldots \quad \theta(t-K+1)]^T \in \angle^{qK}$ et $\overline{\pi_0} = [\pi_0 \quad 0 \quad \ldots \quad 0]^T \in \angle^{n+q(K-1)}$, ainsi que les deux matrices E et $F \in 9^{(n+q(K-1))\times(qK)}$ (I_q est la matrice identité q×q):

$$E = \begin{bmatrix} W^+(0)-W^- & 0 & \cdots & 0 \\ 0 & I_q & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & I_q \end{bmatrix}, et\, F = \begin{bmatrix} W^+(1) & W^+(2) & \cdots & W^+(K) \\ -I_q & 0 & \ldots & 0 \\ & \ddots & \ddots & \vdots \\ 0 & & -I_q & 0 \end{bmatrix}.$$

Théorème 3.2 Avec les notations ci-dessus une fonction non-décroissante $\bar{\theta}(t)$ correspond à une évolution du RdPT si et seulement si pour tout $t \in 3^+$:

$$E\bar{\theta}(t) + F\bar{\theta}(t-1) + \overline{\pi_0} \geq 0$$

Les matrices E et F permettent d'augmenter l'état et de décomposer les places affectées de délais non unitaires en ν_i places de délais unitaires.

On peut encore se rapprocher de la forme d'état usuelle en partitionnant l'ensemble des transitions, T, en deux ensembles X et U, où U est l'ensemble des transitions sources, c'est à dire la commande du système, et X le complémentaire de U par rapport à T. Les compteurs associés à X forment alors l'état, les transitions de U et leurs compteurs sont respectivement notées u_i et $u_i(t)$, on suppose de plus que les entrées ne peuvent être tirées à $t=0$ (autrement dit $u(0)=0$). On introduit aussi un vecteur de sortie $y(t)$ de dimension p (cet ensemble de transitions ne correspond pas forcément aux transitions puits).
En décomposant E et F du corollaire 3.1sur X et U on déduit $E \in \angle^{n\times|X|}$, $F \in \angle^{n\times|X|}$ et $G \in \angle^{n\times|U|}$, de plus on prend $H \in 3^{p\times|X|}$.

Corollaire 3.1 Soit(G, π_0, ν, $\angle$) un RdPT, tel que le sous-RdP S soit acyclique, avec les notations et hypothèses ci-dessus, une fonction non-décroissante x de $\angle$ dans $\angle^{|X|}$ correspond à une évolution du RdPT si et seulement si , pour tout $t \in \angle$, on a :

$$\begin{cases} Ex(t+1) + Fx(t) + \pi_0 + Gu(t+1) \geq 0, \\ x(0) = x_0, \\ y(t) = Hx(t), \end{cases}$$

où u est une fonction non-décroissante de $\angle$ ans $\angle^{|U|}$, arbitrairement choisie et x_0 est une condition initiale vérifiant $Ex_0 + \pi_0 \geq 0$.

4 Fonctionnement « dès que possible ».

Jusqu'ici on a défini les délais comme des durées minimales de séjour dans les places, ce qui laisse supposer qu'un mécanisme non modélisé retient les jetons une fois les transitions aval validées. Mais on peut fixer la règle d'évolution suivante : dès qu'une transition est validée elle est franchie, dans ce cas on dit que le RdPT fonctionne en mode « dès que possible ».

Si tous les poids des arcs sont égaux à 1 (on dit dans ce cas que le RdP est ordinaire) dès que toutes les places amont d'une transition ont au moins un jeton disponible la transition est franchie. Ce qui se traduit plus formellement par le théorème suivant :

Théorème 3.2 Soit <G, π_0, ν, 3^+> un RdPT ordinaire, tel que le sous-réseau S soit acyclique et $\nu \in (3^+)^n$. Une fonction non-décroissante θ de 3 dans $\angle^q$ correpond à une évolution du RdPT fonctionnant dès que possible si et seulement si pour tout $t \in 3^+$, et pour toute transition τ_j, on a :

$$\min_{p_i \in \Gamma^{-1}(\tau_j)} \left\{ \pi_0 + \left[W^+ \theta(t - \nu_i) - W^- \theta(t) \right]_i \right\} = 0 .$$

Etant donné l'introduction de l'opérateur *min* dans les équations il est intéressant de se servir des possibilités qu'offre l'algèbre (min,+) pour reformuler les équations d'état. On définit pour cela la matrice $N \in \angle_{\min}^{|X|\times n}$ par :

$$N = \begin{cases} 0 si\ p_i \in \Gamma^{-1}(\tau_j) \\ +\infty autrement \end{cases}.$$

Corollaire 3.2 Soit (G, π_0, ν, 3^+) un RdPT ordinaire, tel que le sous-réseau S soit acyclique et que $\forall\ i\ \nu_i \in \{0,1\}$. Une fonction non-décroissante x, de $\angle$ dans $\angle^{|X|}$ correspond à une évolution du RdPT fonctionnant dès que possible si et seulement si, pour tout $t \in \angle$, on a :

$$\begin{cases} \tilde{m}(t+1) = Ex(t+1) + Fx(t) + \pi_0 + Gu(t+1), \\ y(t) = Hx(t), \\ N \otimes \tilde{m}(t) = 0, \\ x(0) = x_0, \end{cases}$$

où ⊗ est le produit matriciel de l'algèbre (min,+), u une fonction non-décroissante de $\angle$ dans $\angle^{|U|}$, choisie arbitrairement et x_0, une condition initiale vérifiant $Ex_0 + \pi_0 \geq 0$.

5 Illustration.

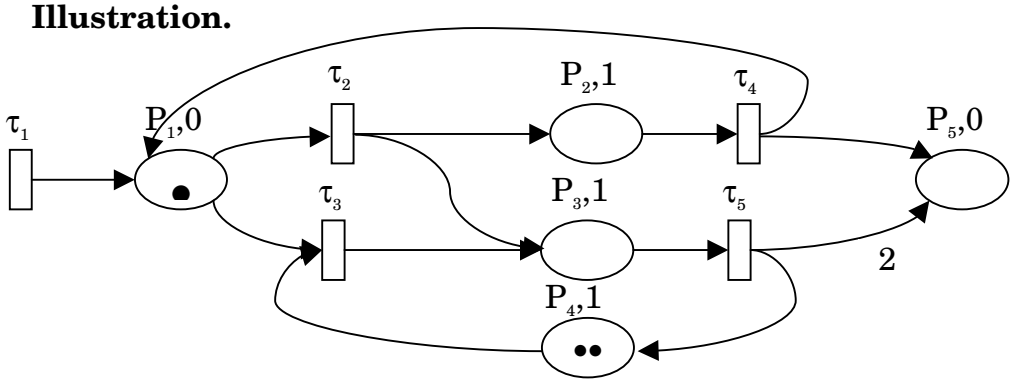


Fig. Un réseau de Peti temporisé.

On illustre par un exemple les différents résultats établis au cours du chapitre, pour ce faire on considère le réseau de Petri temporisé ci-dessus.
Sur le graphe, les délais sont indiqués par les chiffres suivant les places. On voit bien que le sous-réseau de Petri obtenu en supprimant tous les arcs menant à une place de délai non nul est acyclique car toute la partie centrale étant supprimée, on ne peut plus faire de circuit fermé. On est donc bien dans le domaine d'application des différents théorèmes de ce chapitre. On a le marquage initial $\pi_0 = [1\ 0\ 0\ 2\ 0]^{-T}$.

On peut chercher, dans un premier temps, à appliquer le théorème 3.2, c'est à dire à mettre sous la forme d'état : $E\overline{\theta}(t) + F\overline{\theta}(t-1) + \overline{\pi_0} \geq 0$. On sait que toute fonction θ vérifiant la forme d'état correspond à une évolution possible du RdPT.

Aucun délai n'étant supérieur à 1, on a $\overline{\theta} = [\theta(t) \quad \theta(t-1)]$ ainsi que $\overline{\pi}_0 = \pi_0$ et il suffit d'écrire les matrices W^-, *W(0)* et *W(1)* pour déterminer les matrices *E* et *F*:

$$W^- = \begin{bmatrix} 0 & -1 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, W(0) = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 \end{bmatrix} et\, W(1) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

D'après les définitions de *E* et *F* et les dimensions des matrices W^-, *W(0)* et *W(1)*, on a $E=W(0)+W^-$ et $F=W(1)$, soit :

$$E = \begin{bmatrix} 1 & -1 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 \end{bmatrix}, F = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

On peut, à présent, décomposer l'état en une entrée *u*, ici la transition source τ_1, et *x* l'état à proprement parler, à savoir toutes les autres transitions. On définit aussi une sortie, pour bien illustrer le fait que la sortie *y* ne coïncide pas forcément avec les transitions puits on s'intéressera au compteur de τ_2 et à la différence des compteurs de τ_5 et de τ_3. On obtient *E* et *F* en supprimant la première colonne des matrices déterminées plus haut, *G* traduit le fait que la commande du système se fait par la transition τ_1, enfin *H* donne la sortie en fonction des transition s de *x*.

$$\begin{cases} Ex(t+1) + Fx(t) + \pi_0 + Gu(t1) \geq 0, \\ y(t+1) = Hx(t+1), \end{cases}$$

avec :

$$E = \begin{bmatrix} -1 & -1 & 1 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 2 \end{bmatrix}, F = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, G = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} et\, H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix}.$$

6 Caractérisation des trajectoires.

La forme d'état permet d'écrire des inégalités et des égalités en les *q* variables x_i, venant du fonctionnement en mode dès que possible. Ces relations, si on y ajoute les conditions initiales, permettent de déterminer un ensemble de trajectoires élémentaires admissibles par le système. Si on appelle *K* le plus grand délai

apparaissant dans les relations en x_i on peut construire des trajectoires élémentaires de $K+1$ points dans $\angle^q$. Il faut que les délais soient commensurables. Chaque point donne l'état du système, décrit par les variables x_i, à un instant t. Le temps est discrétisé et deux points consécutifs donnent l'état du système a deux instant séparés d'une unité de temps. On peut ensuite établir une relation de succession entre les différentes trajectoires élémentaires, une trajectoire v_2 succédant à une trajectoire v_1 si et seulement si les K premiers points de v_2 correspondent aux K derniers points de v_1.
Cette relation permet de construire un automate (la terminologie des automates utilisée ici est exposée dans le chapitre suivant) dont les états sont des trajectoires élémentaires. On peut déterminer, d'après les conditions initiales, quelles trajectoires élémentaires peuvent constituer un état initial. Les transitions de l'automate sont les tirs de transitions du RdPT auxquels il faut ajouter un événement « *top* » symbolisant l'écoulement d'une unité de temps sans tir de transition (analogue du *tick* des automates temporisé de Brandin et Wonham, exposés au chapitre 5).

Une trajectoire est admissible si elle est admise par l'automate ainsi établi.

L'automate permet de mettre en évidence certaines propriétés du système, comme des cycles de fonctionnement, d'éventuels blocages, on peut aussi calculer un taux de production à partir de l'évolution des compteurs des transitions sur un cycle élémentaire et de la longueur du cycle. En utilisant les outils d'analyse spectrale déjà présentés, il est possible de déterminer le taux de production maximal pour chaque variable en pondérant les arcs par l'évolution du compteur considéré.

7 Exemple du réseau ferroviaire.

Comme on l'a évoqué en fin du chapitre précédent on peut modéliser le réseau en utilisant la structure de partage exclusif de ressource, la ressource commune étant alors le tronçon C. On obtient alors le RdPT de la figure ci-après.
Le RdP est déterminé par sa matrice d'incidence W et le marquage initial M_0, on a (avec P_1=A, P_2=C_1, P_3=C_2, P_4=B, P_5=ressource):

$$W = \begin{pmatrix} -1 & 1 & 0 & 0 \\ 1 & -+1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & -1 & 1 \\ -1 & 1 & -1 & 1 \end{pmatrix} \text{ et } M_0 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}.$$

Le graphe de couverture est le suivant, on peut vérifier que la spécification est respectée, puisque C_1 et C_2 n'ont jamais un marquage non nul simultanément.

$$\begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \begin{matrix} \leftarrow X_2 \\ - X_1 \rightarrow \end{matrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} \begin{matrix} - X_3 \rightarrow \\ \leftarrow X_4 - \end{matrix} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}.$$

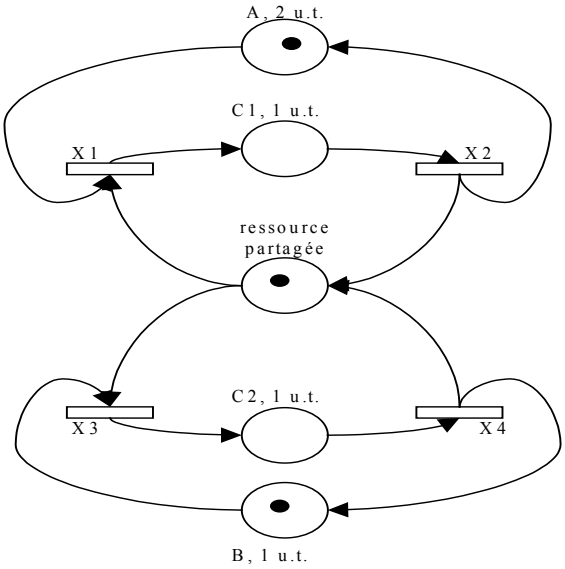


Fig. RdPT du réseau ferroviaire.

On peut déterminer les invariants de places (combinaisons linéaires des composantes du marquage, constantes dans le temps) en résolvant $W^tY = [0]$, ainsi que les invariants de transitions (séquences de tirs telles qu'on ne modifie pas le marquage) en résolvant $WX=[0]$, on obtient:

$$Y=\begin{pmatrix}1&0&0\\1&0&1\\0&1&1\\0&1&0\\0&0&1\end{pmatrix}, autrement\ dit : \begin{cases} m_1(t)+m_2(t)=1,\forall t,\\ m_3(t)+m_4(t)=1,\forall t,\\ m_2(t)+m_3(t)+m_5(t)=1,\forall t,\end{cases} \quad et\ X=\begin{pmatrix}1&0\\1&0\\0&1\\0&1\end{pmatrix}$$

.

La solution en X met juste en évidence que tirer X_1 et X_2 ou X_3 et X_4 fait revenir à la situation initiale, ce qui est assez trivial. Pour plus complêtement le système il faut tenir compte du temps. Pour cela on met en équation le système en utilisant les variables $X_i(t)$ qui sont les compteurs de tirs de la transition i à l'instant t. On traduit à présent que le marquage disponible de chaque place P_i est positif ou nul, autrement dit :

$$M_{0i} + [W^+\theta(t - v_i) - W^-\theta(t)] \geq 0,$$

ce qui en adoptant la forme d'état s'écrit :

$$\begin{pmatrix}-1&0&0&0&&&&\\0&-1&0&0&&&&\\0&0&0&-1&&0&&\\0&0&-1&0&&&&\\-1&1&-1&1&&&&\\&&&&1&0&0&0\\&&0&&0&1&0&0\\&&&&0&0&1&0\\&&&&0&0&0&1\end{pmatrix}\begin{pmatrix}X(t)\\X(t-1)\end{pmatrix}+\begin{pmatrix}0&0&0&0&0&1&0&0\\1&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&1&0&0&0&0\\0&0&0&0&0&0&0&0\\-1&0&\cdots&0&&&&\\0&\ddots&\ddots&\vdots&&0&&\\\vdots&\ddots&\ddots&0&&&&\\0&\cdots&0&-1&&&&\end{pmatrix}\begin{pmatrix}X(t-1)\\X(t-2)\end{pmatrix}+\begin{pmatrix}1\\0\\0\\1\\1\\0\\\vdots\\\vdots\\0\end{pmatrix}\geq\begin{pmatrix}0\\\vdots\\0\end{pmatrix}$$

Autrement dit on a le système d'inéquations suivant :

$$\begin{cases} X_2(t-2)-X_1(t)+1\geq 0, & (1)\\ X_1(t-1)-X_2(t)\geq 0, & (2)\\ X_3(t-1)-X_4(t)\geq 0, & (3)\\ X_4(t-1)-X_3(t)+1\geq 0, & (4)\\ X_2(t)+X_4(t)-X_1(t)-X_3(t)+1\geq 0, & (5) \end{cases} \text{ , avec: } \begin{cases} X_2(t)=0,\ 0\leq t\leq 1,\\ X_4(t)=0,\ 0\leq t\leq 1,\\ 0\leq X_1(t)\leq 1,\ 0\leq t\leq 2,\\ X_3(t)\leq 1, 0\leq t\leq 1,\\ X_i=0, t=0, \forall i\in\{1,\ldots,4\} \end{cases}$$

On suppose que RdPT fonctionne en mode dès que possible. On peut remarquer que lorsqu'une place P_i est la seule place amont de toutes ses transitions aval, on a une égalité et non une inégalité, i.e. si :

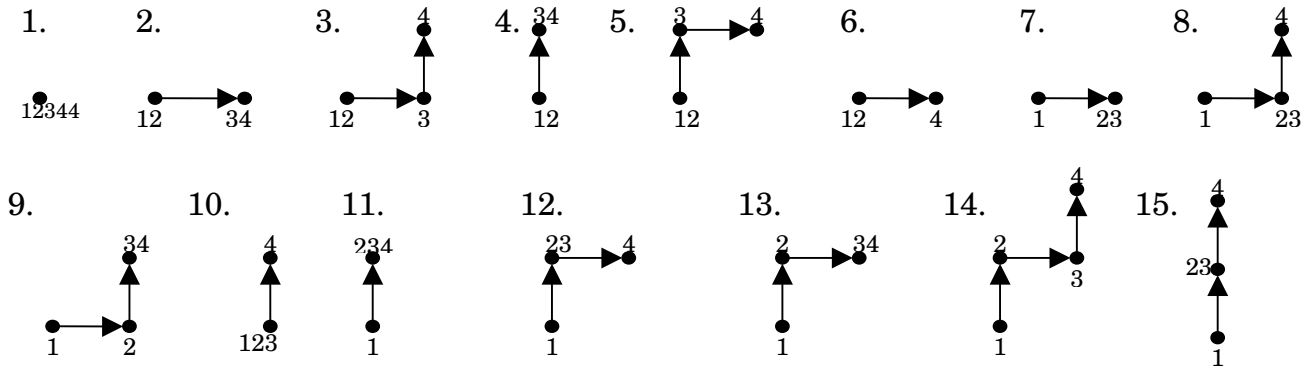
$$|\Gamma^{-1}(T)|=1,\ \forall\ T\in\Gamma(P_i)\Rightarrow \text{égalité en } (i)$$

Ainsi les relations (2) et (3) deviennent : $X_2(t)=X_1(t-1) et\ X_4(t)=X_3(t-1)$. On peut alors décrire le système avec les seules variables X_1 et X_3. En injectant les égalités (2) et (3) dans (1) (4) et (5) on a finalement le système suivant :

$$\begin{cases} X_1(t)\leq 1+X_1(t-3), \forall t\geq 3,\\ X_3(t)\leq 1+X_3(t-2), \forall t\geq 2, \\ X_1(t)+X_3(t)\leq 1+X_1(t-1)+X_3(t-1), \forall t\geq 1. \end{cases} \quad (7)$$

Les deux premières équations montrent que le taux de production est borné par $[1/3\ \ 1/2]^T$, et qu'un régime « de type » périodique doit s'établir sur une « période » de 6 unités de temps. Il faut souligner que le temps t est une variable continue. Si on échantillonne le système avec un pas égal à 1, on obtient une version discrète du même système d'inéquations, où $t\in\angle$

Dans ces équations on a $K=3$, on va donc déterminer toutes les trajectoires de 4 points de $\angle^2$. Etant donné que les variables utiles sont au nombre de deux on représentera graphiquement les trajectoires élémentaires dans le plan (X_1, X_3). Les conditions initiales imposent $X_i=0$ pour $t=0$. Autrement dit, on peut déterminer des trajectoires élémentaires admissibles comme trajectoires initiales. A $t=0$, aucune transition n'est tirée donc les deux premiers points sont confondus. On trouve 15 trajectoires élémentaires, dont six acceptables comme initiales (1…6,10).



On rassemble dans un tableau la relation de succession.

état	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
successeur	1, 6, 10	7, 8	9	11, 12, 15	13, 14	2,3	1,6, 10	4,5	11, 12, 15	4,5	1,6, 10	2,3	7,8	9	4,5

On peut représenter par un automate les transitions possibles d'un état à l'autre, les états initiaux sont en gras.

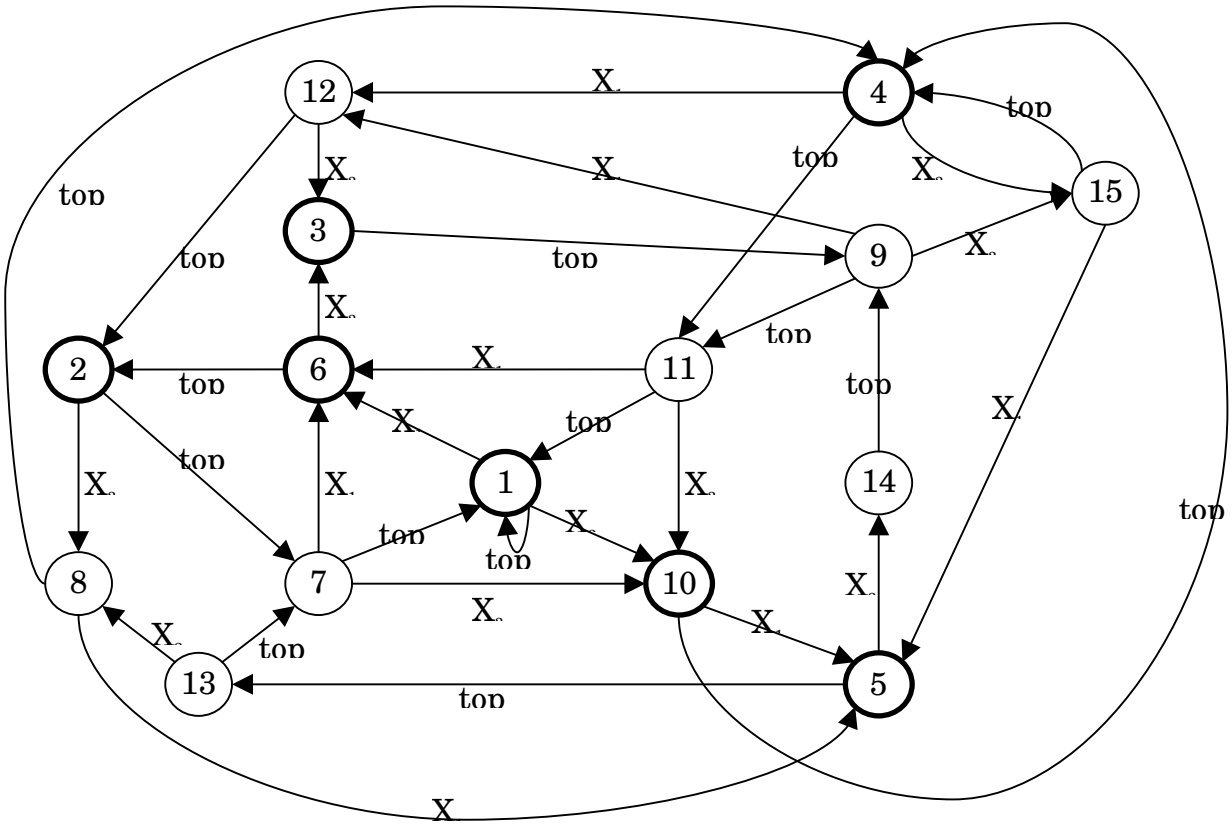
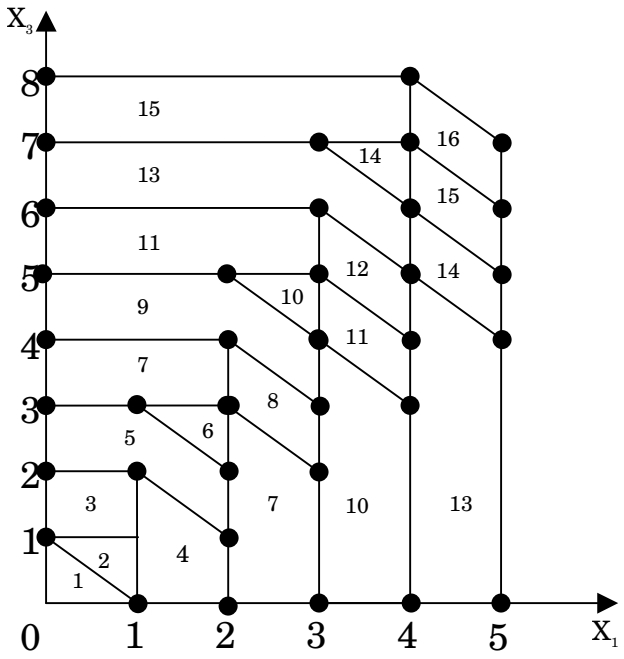


Fig. Automate des trajectoires admissibles.



Cet automate reconnaît toutes les trajectoires admissibles satisfaisant le système *(7)*. Un chemin dans cet automate correspond à une trajectoire $(X_1(t),X_3(t))$ qui est définie de façon unique dès lors qu'on suppose que $X_1(0)=X_3(0)=0$. L'état initial de ce chemin correspond à la condition initiale $X(t),\ 0 \le t \le 3$, dans le système *(7)*. On peut d'abord utiliser cette représentation pour établir l'état accessible en un temps donné.

Le caractère périodique apparaît par le fait qu'on peut déduire la zone atteignable en *n+6* unités de temps en connaissant celle atteignable en *n* unités de temps et en appliquant un algorithme relativement simple.

Algorithme :

- translater de $[2,2]^T$ le sommet de coordonnée X_1 maximum ;
- tracer une ligne brisée en faisant alterner les segments de direction $[-1,0]^T$ et $[-1,-1]^T$ (en commençant par le second type) ;
- s'arrêter à l'intersection avec la ligne horizontale déduite de la limite précédente translatée verticalement de trois unités.

On vérifie bien qu'en 6 u.t. on n'atteint jamais la production $[3\ 2]^T$.

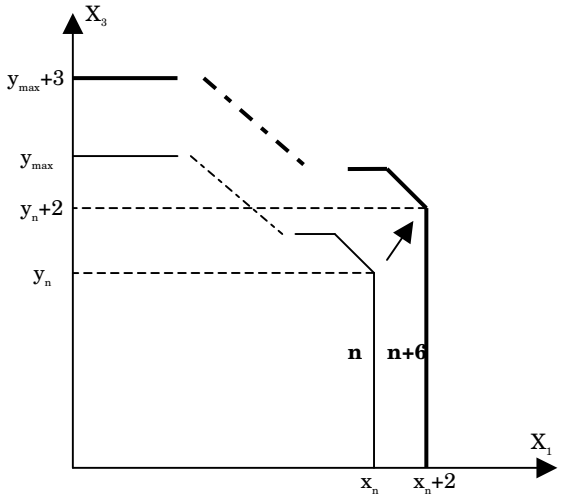


Fig. Fonctionnement périodique.

On a donc réussi à caractériser les différentes trajectoires possibles du système et on peut observer la zone atteignable en une durée donnée, mais cela se fait au prix d'une étude un peu fastidieuse. En effet il faut tester toutes les possibilités de trajectoires avant de pouvoir exhiber le caractère périodique. Cette propriété est prouvable, ici, en recensant tous les cas de figure, mais n'a pas valeur générale. De plus cette méthode n'est utilisable que pour un système ne comportant qu'un faible nombre de variables et pour des systèmes ne faisant pas intervenir de trop grands délais. En effet il faut considérer au plus $(K+1)^{n+1}$ trajectoires élémentaires avec *n* le nombre de variables utiles pour décrire le système.

Conclusion.

La structure de RdPT est très générale, car ne répondant à aucune condition restrictive structurelle, cela en fait un outil de modélisation très puissant. Néanmoins, les résultats sont moins forts que pour les GET, en effet on ne peut plus inverser un système et donner une solution, mais uniquement savoir si une fonction donnée appartient à la classe des fonctions admissibles par le système. De plus la caractérisation des trajectoires, dans le cas de délais commensurables, et l'analyse du système se font au prix de lourds calculs, comme nous l'avons illustré sur l'exemple du réseau ferroviaire. La méthode que nous avons utilisée est originale et serait susceptible d'être généralisée.

Chapitre 4

Automates et langages.

Dans les chapitres précédents on s'est attaché à l'étude de systèmes présentant des phénomènes de synchronisations. Sauf dans le cas des RdPT généraux pour lesquels les résultats sont moins complets (on sait si une fonction est solution, mais on ne donne pas une solution unique), la structure de choix n'est pas possible. Pour ce type particulier de systèmes s'est développée la théorie des automates, construite sur les bases théoriques des langages. Cette approche permet de définir un superviseur qui contrôle le système. On commencera par définir les outils concernant les langages, puis d'autres pour l'étude des automates avant d'aborder les problèmes de supervision, tels que les ont introduits Wonham et Ramadge. Les notions introduites ici, sont exposées plus en détails dans J.L. Ferrier [8] et L. Libeaut et J.J. Loiseau[15].

1 Définitions et opérations sur les langages.

1.1 Définitions.

Un alphabet Σ, est un ensemble (fini ou infini) de lettres. Un mot est une liste finie de lettre de Σ. On note Σ* l'ensemble de tous les mots formés sur Σ. La longueur du mot u donne le nombre de lettres de ce mot, et est notée $|u|$. Le mot vide noté ε (parfois 1) est de longueur nulle.

Exemple
Soit Σ = {a,b,c}, $u = abacb$, $u(1) = a$, $u(2) = b$,…, $|u| = 5$.

Définition 4.1 (Concaténation) On définit la concaténation de deux mots u et v par le mot de longueur $|u| + |v|$, noté $w = u.v$, où :

$$w(i) = \begin{cases} u(i) \; si \; 1 \le 1 \le |u|, \\ v(i - |u|) \; si \; |u| + 1 \le i \le |u| + |v| \; . \end{cases}$$

De plus, le mot vide est l'élément neutre de la concaténation.

Définitions 4.2 (Ordre préfixiel, préfixe suffixe, fermeture, langage fermé, langage fermé par un langage) Soient A,B ∈ Σ deux alphabets, A* muni du produit de concaténation a une structure de monoïde. L'ordre induit par ce monoïde (i.e. un ensemble muni d'une loi associative et d'un élément neutre) est appelé l'ordre préfixiel :

$$\forall u, v \in \Sigma^* \; u \le v \Leftrightarrow \exists w \in \Sigma^*, uw = v.$$

On dit alors que u est un préfixe de v et que w est un suffixe de v. L'ensemble des préfixes des mots de A est appelé la fermeture de A (ou fermeture préfixielle de A) et est noté pref(A) ou $\overline{A}$. On dit que A est fermé si $A = \overline{A}$. On dit que A est B-fermé, ou fermé par B si $A = \overline{A} \cap B$.

Propriétés 4.1 $\forall$ A, B $\subset \Sigma^*$, $\overline{A+B} = \overline{A} + \overline{B}$ et $\overline{A \cap B} \subset \overline{A} \cap \overline{B}$

1.2 Opérations sur les langages.

Un langage est une partie de $L \subset \Sigma^*$. L'ensemble $\Pi(\Sigma^*)$ des parties de Σ^* est muni des opérations d'union $\cup$ aussi notée +, et d'intersection $\cap$. L'union est associative, commutative, idempotente, d'élément neutre $\emptyset$. L'intersection est associative, commutative, idempotente et d'élément neutre Σ^*. Le produit de concaténation est associatif, non commutatif, admet $\{\varepsilon\}$ pour élément neutre et $\emptyset$ comme élément absorbant. Le produit de concaténation est distributif par rapport à l'union, mais pour ce qui est de l'intersection on doit se contenter d'une inclusion :

$$\forall A, B, C \subset \Sigma^*, A.(B + C) = A.B + A.C \text{ et } (B + C).A = B.A + C.A ,$$
$$\forall A, B, C \subset \Sigma^*, A.(B \cap C) \subset A.B \cap A.C .$$

Proposition 4.1 L'ensemble $(\Pi(\Sigma^*), +, .)$ est un dioïde

On retrouve des résultats déjà établis au premier chapitre concernant la résolution de l'équation implicite $X = A.X + B$.

Définition 4.3 (Etoile de Kleene) L'étoile de Kleene d'un langage A est définie par $A^* = \cup_{n \geq 0} A^n$.

Proposition 4.2 Pour tout langage A et B $\subset \Sigma^*$ les propositions suivantes sont vérifiées :

1. $A^* = \varepsilon + A.A^* = \varepsilon + A^*.A$,
2. $(A^* + B^*)^* = (A^*.B^*)^* = (A + B)^* = (A^*.B)^*.A^*$,
3. $(B.A)^*.B = B(A.B)^*$.

Théorème 4.1 Pour tout langage A et B $\subset \Sigma^*$, A^*B est la plus petite solution de l'équation $X = AX+B$, de plus si $\varepsilon \notin A$, A^*B est l'unique solution.

2 Les automates.

2.1 Définitions et représentation.

Définition 4.4 (Automate) Un automate est défini par le quintuplet $A=(Q, \Sigma, \delta, Q_0, Q_m)$, où Q est un ensemble d'états, Σ un ensemble de symboles, δ une

fonction de transition $Q \times \Sigma \times Q$, $Q_0 \subset Q$ un sous-ensemble d'états initiaux et $Q_m \subset Q$ un sous-ensemble d'états finaux ou marqués.
Si Q est fini, l'automate est dit fini.

La manière la plus simple de spécifier un automate est de le représenter par un graphe. Les états réalisables sont alors les sommets du graphe orienté. Les états initiaux sont indiqués par une flèche. Les états finaux sont indiqués par une croix. La flèche $q \to q'$ est valuée par $a \in \Sigma$ si et seulement si $\delta(q,a)$ est défini.

	1	2
1		a
2	b	

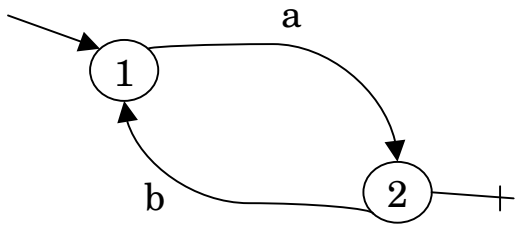


Fig. Représentation matricielle et graphique d'un automate.

Une séquence telle que : $q_0(q_0,\sigma_1,q_1)q_1(q_1,\sigma_2,q_2)q_2 \ldots q_{n-1}(q_{n-1},\sigma_n,q_n)q_n$ est appelée chemin et le mot $q_0q_1q_2 \ldots q_n$ est appelé la trace de ce chemin. L'ensemble des traces des chemins de A est noté $L(A)$. L'ensemble des traces des chemins qui permettent d'arriver à un état marqué est noté $L_m(A)$. $L(A)$ est le langage reconnu par l'automate, $L_m(A)$ est le langage marqué de l'automate, tous deux sont des éléments de Σ^*. Réciproquement tout langage peut être interprété comme l'ensemble des traces d'un automate. On peut de plus remarquer que $L(A)$ est fermé. Plus rigoureusement on écrit:

$$L(A) = \{ w \in \Sigma^* \mid \delta(q_0,w) \in Q \},$$
$$L_m(A) = \{w \in \Sigma^* \mid \delta(q_0, w)! \text{ et } \delta(q_0, w) \in Q_m\}.$$

Proposition 4.3 Soit $A \in \Sigma^*$, alors il existe un automate $A = (Q,\Sigma,\delta,Q_0,Q_m)$ tel que $L_m(A) = A$. Si de plus A est fermé, on peut choisir A pour que $L(A) = A$.

Une des informations principales concernant un automate est de savoir quels sont les états auxquels on peut parvenir depuis un état initial, ainsi que de savoir quels sont les états qui permettent de rejoindre un état final.

Définitions 4.4 (Atteignabilité, coatteignabilité, automate émondé) Un état $q \in Q$ est atteignable s'il est l'extrémité d'un chemin de l'automate, soit :

$$\exists w \in \Sigma^* \mid \delta(q_0,w)=q.$$

Un état q est dit coatteignable si on peut atteindre un état marqué depuis q, soit :

$$\exists w \in \Sigma^* \mid \delta(q,w) \in Q.$$

On note respectivement Q_a et Q_c l'ensemble des états atteignables et coatteignables. Un automate A est atteignable si tous ses états le sont. A est coatteignable si tous ses états le sont. L'automate A est émondé s'il est à la fois atteignable et coatteignable.

Corollaire 4.1 Un automate est coatteignable si et seulement si tous ses états appartiennent à un chemin menant à un état marqué, autrement dit ssi :

$$\overline{L_m(A)} = L(A).$$

Définition 4.5 (Automate restreint à Q') Soit un automate A = (Q,Σ,δ,q_0,Q_m), on définit l'automate restreint à $Q' \subset Q$ par : $A_{/Q'}$ = (Q',Σ,δ',q_0,$Q_m \cap Q'$) où $\delta = \delta \cap (Q' \times \Sigma \times Q')$ est notée $\delta_{/Q'}$.

Proposition 4.4 Avec les notations définies précédemment on a :

$$L(A) = L(A_{/Q_a}),$$
$$L_m(A) = L(A_{/Q_a \cap Q_m}) = L_m(A_{/Q_a}).$$

Ce qui signifie qu'on ne modifie pas le langage $L(A)$ en supprimant les états non atteignables. De la même manière on ne modifie pas le langage reconnu par l'automate $L_m(A)$ en l'émondant.

2.2 Déterminisation d'un automate.

On a dit en introduction que les automates permettent de modéliser les situations de choix de trajectoires, mais il faut distinguer les cas où une même trace permet plusieurs chemins et celui où chaque chemin correspond à une trace et chaque trace à un chemin. Dans le second cas on parle d'automate déterministe.

Définition 4.6 (Automate déterministe) Un automate A = (Q,Σ,δ,Q_0,Q_m) est déterministe s'il satisfait les deux conditions suivantes:

1. $Q_0 = \{q_0\}$ est réduit à un seul état initial;
2. $\forall q', q'', q'' \in Q$, $\forall \sigma \in \Sigma$, si $(q,Q,q') \in \delta$ et si $(q,Q,q'') \in \delta$, alors $q' = q''$.

Dans ce cas, si $\delta(q,\sigma)$ est définie, elle l'est de manière unique, on écrit $\delta(q,\sigma)!$, on a : $\delta(q,\varepsilon) = q$ et $\delta(q,\omega\sigma) = \delta(\delta(q,\omega),\sigma)$.

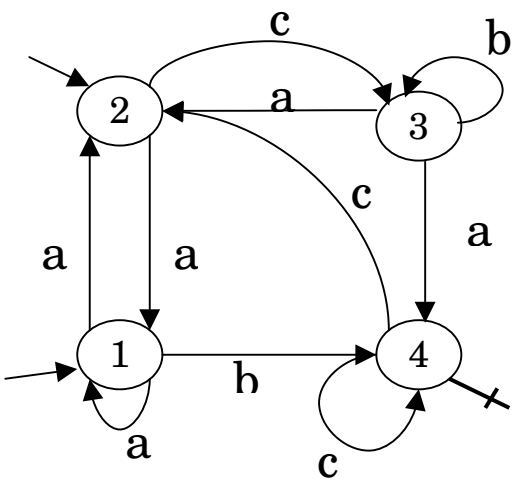
Il est bien sûr plus pratique de travailler avec des automates déterministes car leur comportement à l'occurrence d'un événement donné est connu, d'où l'intérêt de la proposition suivante:

Proposition 4.5 Soit $A = (Q,\Sigma,\delta,Q_0,Q_m)$ un automate. Alors il existe un automate déterministe B, tel que $L(B) = L(A)$. Il existe également un automate, X, déterministe et émondé tel que $L_m(X) = L_m(A)$.

Pour déterminiser l'automate $A = (Q,\Sigma,\delta,Q_0,Q_m)$ on ne considère plus des états mais des ensembles d'états. On construit $B = (Q',\Sigma,\psi,q'_0,Q'_m)$ de la manière suivante. L'état initial de B est l'ensemble des états initiaux de A, $q'_0 = Q_0$. Les états finaux sont les ensembles dans lesquels apparaissent un ou plusieurs états finaux de l'automate à déterminiser. Chaque état de B est construit itérativement par : $\psi(q',\sigma) = \bigcup_{q_i \in q'} \delta(q_i,\sigma)$.

Exemple de déterminisation d'automate.

On veut déterminiser l'automate de la figure ci-dessous, $A = (Q,\Sigma,\delta,Q_0,Q_m)$ avec Σ={a,b,c}, Q_0={1,2}, Q_m={4} donné par le graphe ci-dessous. L'état initial de l'automate déterministe sera {1,2}, ensuite on calcule la fonction de transition, on obtient alors le tableau ci-dessous, qui permet de tracer le graphe déterminisé ci-après.



	A	b	C
P_0={1,2}	P_0	P_1={4}	P_2={3}
P_1={4}	/	/	P_3={2,4}
P_2={3}	P_3={2,4}	P_2={3}	/
P_3={2,4}	P_4={1}	/	P_5={2,3,4}
P_4={1}	P_0={1,2}	P_1={4}	/
P_5={2,3,4}	P_6={1,2,4}	P_2={3}	P_5={2,3,4}
P_6={1,2,4}	P_0={1,2}	P_1={4}	P_5={2,3,4}

Fig. Automate non déterministe.

Les états finaux de l'automate déterministe sont P_1, P_3, P_5, P_6. On obtient le graphe suivant :

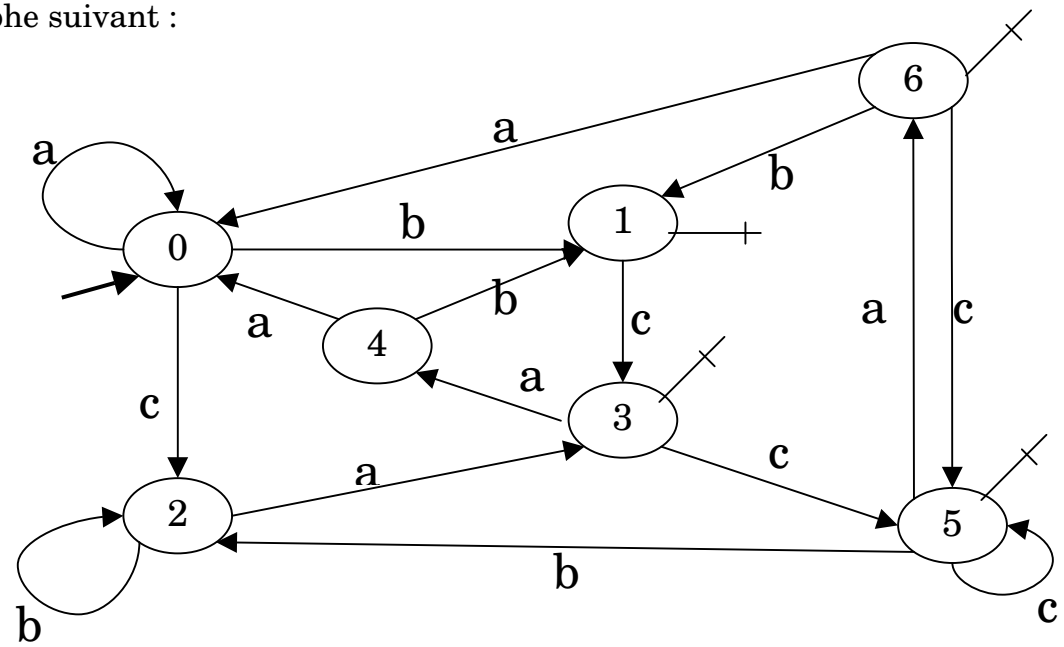


Fig. Automate déterminisé.

3 Le théorème de Kleene.

Dans les deux paragraphes précédents on a défini sommairement les notions de langage et d'automate, on a aussi établi un lien les unissant en remarquant que les traces des chemins d'un automate forment un langage et que réciproquement

tout langage peut être interprété comme l'ensemble des traces des chemins d'un automate. En fait l'équivalence langage/automate est établie par un théorème fondamental : le théorème de Kleene.
Avant d'énoncer le théorème de Kleene, il nous faut définir ce qu'est un langage reconnaissable.
Définition 4.7 (Langage reconnaissable) Un langage $A \in \Sigma^*$ est dit reconnaissable si il existe un automate fini A tel que : $A = L(A)$.

Proposition 4.6 Tout langage fini est reconnaissable.

Proposition 4.7 Si $A \in \Sigma^*$ est reconnaissable, alors $\overline{A}$ et A^* le sont aussi.

Proposition 4.8 Si $A, B \in \Sigma^*$ sont reconnaissables, alors $A+B$, $A.B$, et $A \cap B$ sont reconnaissables.

Proposition 4.9 Si $A \in \Sigma^*$ est reconnaissable, alors A s'exprime sous la forme d'une expression algébrique où interviennent des éléments de Σ et les symboles +, ., *,(,) en nombre fini. Un tel langage est dit rationnel.

Autrement dit on peut maintenant écrire le théorème de Kleene:
Théorème 4.2 (Théorème de Kleene) Un langage $A \in \Sigma^*$ est reconnaissable si et seulement si il est rationnel.

4 Produits de langages et d'automates.

Un procédé réel est souvent composé de plusieurs procédés élémentaires, en interaction plus ou moins forte. De plus, à partir d'un certain niveau de complexité il devient difficile de construire un automate "en un bloc". Dans ce cas il peut être plus commode de construire le système à partir de sous-systèmes élémentaires. On parle alors de produit ou de composition d'automates.

4.1 Produits de langages.

On définit d'abord les opérateurs de projection et de projection inverse.
Définitions 4.8 (Projection et projection inverse) Soient Σ_1 et Σ_2, deux alphabets (non nécessairement disjoints). On définit la projection naturelle $P_i: (\Sigma_1 \cup \Sigma_2)^* \to \Sigma_i^*$, $i = 1,2$ et l'application inverse $P_i^{-1} : \Sigma_i^* \to (\Sigma_1 \cup \Sigma_2)^*$, $i = 1,2$, par:

$$P_i(\varepsilon) := \varepsilon,$$

$$P_i(\sigma) := \begin{cases} \sigma \; si \; \sigma \in \Sigma_i, \\ \varepsilon \; si \; \sigma \notin \Sigma_i, \end{cases}$$

$$P_i(w\sigma) := P_i(w)P_i(\sigma) \; pour \; w \in (\Sigma_1 \cup \Sigma_2)^*, \sigma \in \Sigma_1 \cup \Sigma_2,$$

$$P_i^{-1}(v) := \{w \in (\Sigma_1 \cup \Sigma_2)^* \,|\, P_i(w) = v\}.$$

On peut étendre ces notions aux langages, pour $L \subset (\Sigma_1 \cup \Sigma_2)^*$ et $L_i \subset \Sigma_i$, avec :

$$P_i(L) := \{v \in \Sigma_i^* \mid \exists w \in L,\ P_i(w) := v\},$$
$$P_i^{-1}(L) := \{w \in (\Sigma_1 \cup \Sigma_2)^* \mid \exists v \in L_i,\ P_i(w) = v\}$$

Si L_1 et L_2 sont deux langages on définit le produit de ces deux langages $L_1 \times L_2$ par : $L_1 \times L_2 = P_1^{-1}(L_1) \cap P_1^{-1}(L_1)$.

4.2 Produits d'automates.

Soient deux automates $A_1 = (Q_1, \Sigma_1, \delta_1, q_{01}, Q_{m1})$ et $A_2 = (Q_2, \Sigma_2, \delta_2, q_{02}, Q_{m2})$ supposés déterministes. On définit leur produit $A_1 \times A_2$ par :

$$A_1 \times A_2 = (Q_1 \times Q_2,\ \Sigma_1 \cup \Sigma_2,\ \psi,\ (q_{01}, q_{02}),\ Q_{m1} \times Q_{m2}).$$

où:

$$\psi((o_1, o_1), w) = \begin{cases} (\delta_1(q_1,\sigma), \delta_2(q_2,\sigma)) & si\ \sigma \in \Sigma_1 \cup \Sigma_2,\ \delta_1(q_1,\sigma)!\ et\ \delta_2(q_2,\sigma)!, \\ (\delta_1(q_1,\sigma), q_2) & si\ \sigma \in \Sigma_1, \delta_1(q_1,\sigma)!\ et\ \sigma \notin \Sigma_2\ ou\ (\sigma \in \Sigma_2\ et\ \delta_2(q_2,\sigma)\ non\ défini), \\ (q_1, \delta_2(q_2,\sigma)) & si\ \sigma \in \Sigma_2, \delta_2(q_2,\sigma)!\ et\ \sigma \notin \Sigma_1\ ou\ (\sigma \in \Sigma_1\ et\ \delta_1(q_1,\sigma)\ non\ défini), \\ non\ défini & autrement. \end{cases}$$

On vérifie que le produit est commutatif et associatif, de plus on a :

$$L(A_1 \times A_2) = P_1^{-1}[L(A_1)] \cap P_2^{-1}[L(A_2)],$$
$$L_m(A_1 \times A_2) = P_1^{-1}[L_m(A_1)] \cap P_2^{-1}[L_m(A_2)].$$

Deux cas particuliers sont intéressants. Quand les deux langages Σ_1 et Σ_2 sont distincts, le produit est dit asynchrone, alors si A_1 et A_2 ont respectivement n_1 et n_2 états, $A_1 \times A_2$ a exactement $n_1 \times n_2$ états. Quand Σ_1 et Σ_2 sont égaux, on parle de produit synchrone, une transition $\psi(q_1,q_2,\sigma)$ n'existe que si $\delta_1(q_1,\sigma)$ et $\delta_2(q_2,\sigma)$ sont définis simultanément. Dans ce cas les langages $L(A_1 \times A_2)$ et $L_m(A_1 \times A_2)$ sont donnés par :

$$L(A_1 \times A_2) = L(A_1) \cap L(A_2),$$
$$L_m(A_1 \times A_2) = [L_m(A_1)] \cap L_m(A_2).$$

Exemple de produit synchrone d'automate.

On se contentera d'un exemple de produit synchrone, en effet on peut considérer le produit asynchrone comme le cas particulier du produit synchrone lorsqu'aucun événement (i.e. élément du langage) n'est commun aux deux automates (de plus l'exemple de supervision en fin de chapitre donne un exemple de produit asynchrone).

Soient Σ_1={a, b} et Σ_2={b, c}, donc $\Sigma_1 \cap \Sigma_2$={b}, les deux automates A_1 et A_2 sont représentés ci-dessous :

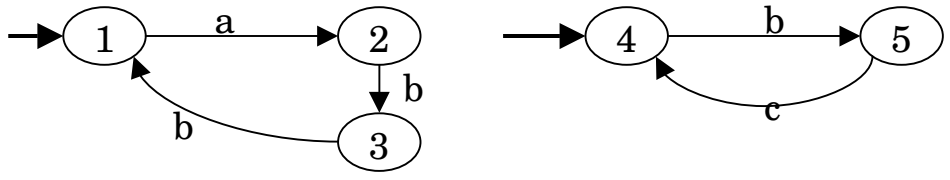


Fig. Automates A_1 et A_2.

L'état initial est l'état (1,4), $\psi((1,4),b)$ n'est pas défini car $\delta_1(1,b)$ n'existe pas, donc seul l'événement "a" peut se produire, on va alors en (2,4). En revanche $\psi((2,4),b)!$ et $\psi((2,4),b)=(3,5)$. On continue ainsi la construction de $A_1 \times A_2$ et on obtient finalement :

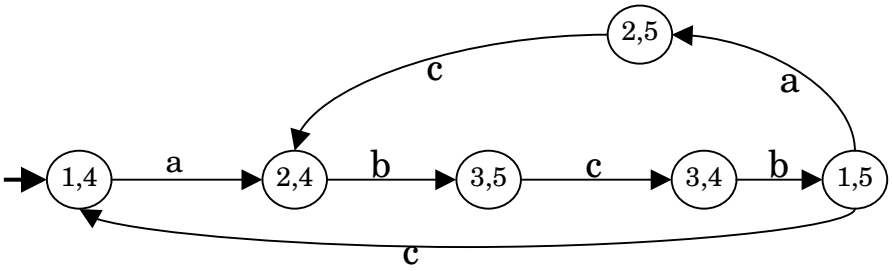


Fig. Produit synchrone des automates A_1 et A_2.

5 Supervision d'automate.

On présente dans ce paragraphe la notion de supervision d'un automate c'est à dire qu'on cherche à le commander en respectant une spécification. Pour cela nous allons dans un premier temps définir les notations de bases concernant les systèmes commandés, avant de s'attacher à la formulation et la résolution des problèmes de commande.

5.1 Notations et définitions.

Par la suite nous considérerons l'automate $M=(Q,\Sigma,\delta,q_0,Q_m)$, qui reconnaît le langage $M=L(M)=\{w \in \Sigma^* \mid \delta(q_0,w)!\}$. M représente l'ensemble des séquences de changements d'états possibles. Le langage marqué, M_m, de l'automate représente les séquences qui conduisent à la réalisation d'un certain objectif Il est défini par : $M_m=L_m(M)=\{w \in \Sigma^* \mid \delta(q_0,w)! \text{ et } \delta(q_0,w) \in Q_m\}$. Les états non coatteignables sont des états à partir desquels on ne peut plus achever l'objectif. on dit que M est non bloquant s'il est coatteignable i.e. $\overline{M_m} = M$.

5.1.1 Partition commandable/non commandable.

Tous les événements liés à un processus ne dépendent pas de la volonté de l'opérateur, typiquement si on s'intéresse au fonctionnement d'une machine, la survenue d'une panne "p", est par essence non commandable, tandis que la réparation "r", l'arrêt a, ou la mise en marche "m" sont commandables.

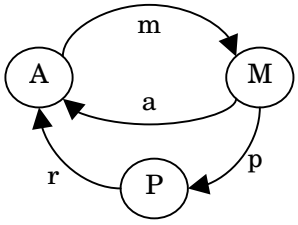


Fig. Fonctionnement d'une machine.

Il est intéressant de partitionner Σ en deux sous-ensembles distincts : Σ_c et Σ_n constitués respectivement des événements commandables et des événements non commandables. Un événement est dit commandable quand on peut empêcher sa survenue. On a donc : $\Sigma=\Sigma_c\cup\Sigma_c$, $\Sigma_c\cap\Sigma_c=\emptyset$. Dans l'exemple précédent on a : Σ={a,m,p,r}, Σ_c={a,m,r} et Σ_n={p}.

5.1.2 Loi de commande, superviseur, système en boucle fermée.

Une loi de commande est l'application qui, à chaque état autorise certaines évolutions. Elle est définie par :

$$f:\Sigma^* \to \Pi(\Sigma),$$
$$w \mapsto f(w), \text{ où } \Sigma_n \subset f(w) \subset \Sigma.$$

Ce qui signifie que *f(w),* c'est à dire l'ensemble des événements de Σ autorisés après *w,* contient les Σ_n, autrement dit *f* n'a pas le pouvoir d'inhiber les événements incontrôlables, *f* est aussi appelée superviseur.

A partir de cette fonction on peut définir le système en boucle fermée de manière récursive par :

$$\varepsilon \in L_f \text{ et } L_f=\{w\in\Sigma^* \mid w=w'\sigma,\ w\in M,\ w'\in L_f, \sigma\in f(w')\}.$$

Autrement dit le système peut toujours rester en l'état et sont autorisés les états résultant d'un déplacement autorisé à partir d'un état autorisé.

Proposition 4.10 Les trois assertions suivantes sont vérifiées :

(*i*) $\overline{L_f} = L_f \subset M$, ($L_f$ *est fermé*),

(*ii*) $\overline{L_f \cap M_m} \subset L_f$,

(*iii*) $L_f\Sigma_n \cap M \subset L_f$.

De façon équivalente, on peut définir un superviseur comme un langage *S* tel que :

1. ε∈*S*,
2. *si* $w\in S$ *et* $\sigma\in f(w)$ *alors* $w\sigma\in S$.

Dans ce cas tout superviseur *f* tel que défini précédemment correspond à un unique langage défini comme ci-dessus. De plus le langage *S* est fermé, $S = \overline{S}$, et il satisfait la condition $S\Sigma_n \subset S$, autrement dit tout mot de *S* suivi d'un événement non commandable est encore dans *S*, ce qui correspond à la propriété $\Sigma_n \subset f(w)$. Un tel *S* est dit admissible.

Réciproquement, si un langage *S* est fermé et vérifie $S\Sigma_n \subset S$ alors il correspond à un superviseur *f* défini par :

$$f(w) = \begin{cases} \{\sigma\in\Sigma \mid w\sigma\in S\}, si\ w\in S, \\ indifférent\ autrement. \end{cases}$$

Avec cette approche le système en boucle fermée est reconnu par le produit synchrone des automates qui reconnaissent M et S, et $L_f=S\cap M$. le superviseur est non bloquant si $S \cap M = \overline{S \cap M_m}$.

5.2 Résolution des problèmes de commande.

Ce qu'on appelle problème de commande est de savoir si, étant donné un automate reconnu par S et une spécification donné $L\subset\Sigma^*$, il existe un superviseur reconnu par S (autrement dit S est fermé et vérifie $S\Sigma_n \subset S$), satisfaisant certaines conditions. On distingue deux classes de problèmes suivant que tous les événements sont contrôlables ou non. On a regroupé les résultats dans le tableau ci-dessous.

<table>
<tr><td colspan="2">Tous les événements sont contrôlables, i.e. $\Sigma_n=\emptyset$</td><td colspan="2">Cas général, $\Sigma_n\neq\emptyset$</td></tr>
<tr><td colspan="2">S superviseur $\Leftrightarrow \overline{S} = S$.</td><td colspan="2">S superviseur $\Leftrightarrow \begin{cases} \overline{S} = S \\ S\Sigma_n \subset S \end{cases}$</td></tr>
<tr><td>Cas n°1</td><td>Cas n°2</td><td>Cas n°5</td><td>Cas n°6</td></tr>
<tr><td>Pour L donné, il existe un S tel que $S\cap M=L$ ssi $\begin{cases} L = \overline{L}, \\ L \subset M, \end{cases}$ (soit $\overline{L} = L \cap M$).</td><td>Pour L donné, il existe un S tel que $S\cap M_m=L$ ssi $L = \overline{L} \cap M_m$.</td><td>Pour L donné, il existe un S tel que $S\cap M=L$ ssi $\begin{cases} L = \overline{L}, \\ L \subset M, \\ \overline{L}\Sigma_n \cap M \subset \overline{L}. \end{cases}$</td><td>Pour L donné, il existe un S tel que $S\cap M= \overline{L}$ ssi $\begin{cases} \overline{L} \subset M, \\ \overline{L}\Sigma_n \cap M \subset \overline{L}. \end{cases}$</td></tr>
<tr><td>Cas n°3</td><td>Cas n°4</td><td>Cas n°7</td><td>Cas n°8</td></tr>
<tr><td>Pour L donné, il existe un S non bloquant, tel que $S\cap M=L$ ssi $L = \overline{L \cap M_m}$.</td><td>Pour L donné, il existe un S non bloquant, tel que $S\cap M_m=L$ ssi $L = \overline{L} \cap M_m$.</td><td>Pour L donné, il existe un S, non bloquant, tel que $S\cap M=L$ ssi $\begin{cases} \overline{L \cap M_m} = L, \\ \overline{L}\Sigma_n \cap M \subset \overline{L}. \end{cases}$</td><td>Pour L donné, il existe un S, non bloquant, tel que $S\cap M_m=L$ ssi $\begin{cases} L = \overline{L} \cap M_m, \\ \overline{L}\Sigma_n \cap M \subset \overline{L}. \end{cases}$</td></tr>
</table>

On trouvera les preuves de toutes ces propositions dans *L. Libeaut et J.J. Loiseau [15].*

5.3 Commande optimale.

On cherche le superviseur maximalement permissif, c'est à dire qui autorise le maximum d'états, tout en respectant la spécification L. Pour cela on définit différentes familles, qui correspondent aux différents cas envisagés dans le tableau. Les éléments suprémaux de ces familles, quand ils existent, sont les solutions des problèmes de commande optimale.

Pour les quatre premiers cas on définit respectivement $\Phi_1=\{K \subset L | K \cap M = \overline{K}\}$. $\Phi_2=\{K \subset L | \overline{K} \cap M_m = K\}$, $\Phi_3=\{K \subset L | \overline{K \cap M_m} = K\}$ et $\Phi_4=\{K \subset L | \overline{K} \cap M_m = K\}$.

Lemme 4.1 Les langages sup(Φ_1) et sup(Φ_2) sont définis. De plus ils sont rationnels et effectivement calculables dès que L, M, et M_m sont rationnels.

Démonstration L'existence des éléments suprémaux repose sur le fait que les deux familles sont fermées pour l'union des langages et que chacune d'entre elle contient au moins ∅.

Lemme 4.2 Les langages sup(Φ_3) et sup(Φ_4), sont bien définis. De plus ils sont rationnels et effectivement calculables si L et M_m sont rationnels. Si on définit Φ_f comme le plus grand langage fermé de L (soit $F_f=sup\{X | X \subset L, X = \overline{X}\}$) alors on a : sup($\Phi_3$)= $\overline{F_f \cap M_m}$.

Démonstration On remarque que $\Phi_2=\Phi_4$ donc une partie du résultat est déjà établie par le lemme 4.1. Pour Φ_3 la méthode est la même,Φ3 est non vide car contient au moins ∅ et le *sup* existe car elle est fermée pour l'union.

Pour les quatre derniers cas on définit les familles de langages de L respectivement commandables, fermés, fermés non bloquants et M_m-fermés par :X(L)=$\{K \subset L | \overline{K}\Sigma_n \cap M \subset \overline{K}\}$, Φ(L)=$\{K \subset L | K = \overline{K}\}$, Φ_{nb}(L)=$\{K \subset L | K = \overline{K \cap M_m}\}$, Φ_m(L)=$\{K \subset L | K = \overline{K} \cap M_m\}$.

Lemme 4.3 Les familles $X(L)$, $\Phi(L)$, $\Phi_{nb}(L)$ et $\Phi_m(L)$, ainsi que leurs intersections, en particulier $X(L)\cap\Phi(L)$, $X(L)\cap\Phi_{nb}(L)$ et $X(L)\cap\Phi_m(L)$, sont fermées pour l'union. Chacune de ces familles admet donc un élément maximal, noté sup(.), qui est l'union de tous ses éléments.

Corollaire 4.2 Il existe un superviseur S respectant la spécification L, $S\cap M\subset L$, et maximalement permissif.S est tel que $S\cap M=sup(X(L)\cap\Phi_{nb}(L))$.

Corollaire 4.3 Il existe un superviseur S respectant la spécification L, $S\cap M_m\subset L$ et maximalement permissif. S est tel que $S\cap M=sup(X(L)\cap\Phi_m(L))$.

Le lemme 4.3 assure la justification des cas n°5 et 6, et les corollaires 4.2 et 4.3 se rapportent respectivement aux cas 7 et 8.

Dans ce paragraphe on a prouvé l'existence des différents éléments suprémaux, mais on n'a pas vu comment les calculer, c'est l'objet du chapitre qui suit.

5.4 Calcul des éléments suprémaux.

On peut remarquer que la commandabilité du superviseur (c'est à dire $S.\Sigma_n \subset S$) n'a d'intérêt que sur *M*, en effet on ne s'occupe que des évolutions possibles du système supervisé. On peut donc relaxer la condition d'admissibilité (commandabilité et fermeture) pour ne retenir que la suivante

$$(S \cap M)\Sigma_n \subset S.$$

On cherche à simplifier le calcul des éléments suprémaux pour cela on a les propositions 4.11 et 4.12 dont on peut trouver les preuves dans *L. Libeaut et J.J. Loiseau [15]*

Proposition 4.11 $sup(X(L) \cap \Phi_{nb}(L)) = \overline{\sup(C(L \cap M_m))}$.

Proposition 4.12 $sup(X(L) \cap \Phi_m(L)) = sup\ X(sup\ \Phi_m(L))$.

Ces propositions reposent sur l'équivalence des assertions suivantes:

1. si $X \subset Y$, alors _sup(X(X))⊂sup(X(Y));
2. si *X* est fermé, alors *sup(X(X))* aussi;
3. si *X* est commandable, alors *X* = *sup(X(X))*;
4. si *X* est commandable, alors $\overline{X}$ est commandable;
5. si $\overline{X}$ est commandable, alors *X* est commandable;
6. $\sup(C(X)) = X \cap \overline{\sup(C(X))}$.

Les différents calculs se ramènent tous au calcul du plus grand langage commandable inclus dans un langage. On note L↑ le plus grand langage commandable inclus dans *L*, autrement dit L↑= sup(X(L)). L↑ apparaît être le point fixe de l'opérateur Ω, défini comme suit :

$$\Omega(K) = \sup\left\{J \subset L \middle| \overline{J}\Sigma_n \cap M \subset M\right\}.$$

Ω(K) est bien défini car l'inclusion dans *L* et la commandabilité sont stables par l'union.

On a les proposition suivantes :

Proposition 4.13 Ω(K) est *L*-fermé, soit $\Omega(K) = \overline{\Omega(K)} \cap L$.

Proposition 4.14 L↑=Ω(L↑).

Démonstration L↑ est inclus dans *L* et est commandable donc L↑⊂Ω(L↑), de plus Ω(L↑)⊂L et $\overline{\Omega(L\uparrow)}\Sigma_n \cap M \subset \overline{L\uparrow} \subset \Omega(L\uparrow)$, donc Ω(L↑) est commandable et inclus dans *L*, d'où le résultat.

Proposition 4.15 Définissons la suite K_i par : $K_0=L$ et $K_i = \Omega(K_{i-1})$ *pour* $i\geq 1$. La suite est non croissante (car $K_i \subset K_{i-1}$, *pour* $i\geq 1$) et converge vers une limite notée K_∞. De plus $L\uparrow \subset K_\infty$.

Ainsi pour déterminer l'automate maximalement permissif en regard d'une spécification, on effectue les opérations suivantes :

1. on supprime tous les états interdits par la spécification;
2. on déclare interdits tous les états bloquants;
3. on déclare interdits tous les états qui ne peuvent être atteints que par l'intermédiaire d'un état interdit;
4. on déclare interdits tous les états qui mènent à un état interdit à l'occurrence d'un événement incontrôlable;
5. on supprime les états interdits;
6. si l'opération 5 a changé l'automate on reprend à l'opération 2, sinon l'automate est maximalement permissif et respecte la spécification.

Exemple.
Deux trains circulent sur un réseau constitué de quatre sections.

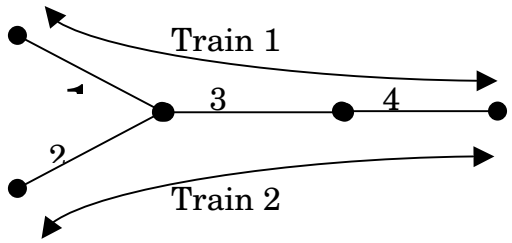


Fig. Réseau ferroviaire.

Chaque train parcourt un circuit, et on souhaite éviter les accidents, autrement dit éviter que deux trains soient simultanément sur une voie. Pour ce faire on installe un système de feux, permettant d'interdire l'accès d'une section à un train. On suppose que tous les changements d'états sont commandables, autrement dit, les feux sont bien vus et respectés par les conducteurs. En faisant le produit asynchrone des automates correspondant à chaque train (le train 1 a un parcours du type : 1343134…, le train 2 a un parcours du type : 2343234…)on a l'automate représenté ci-dessous.

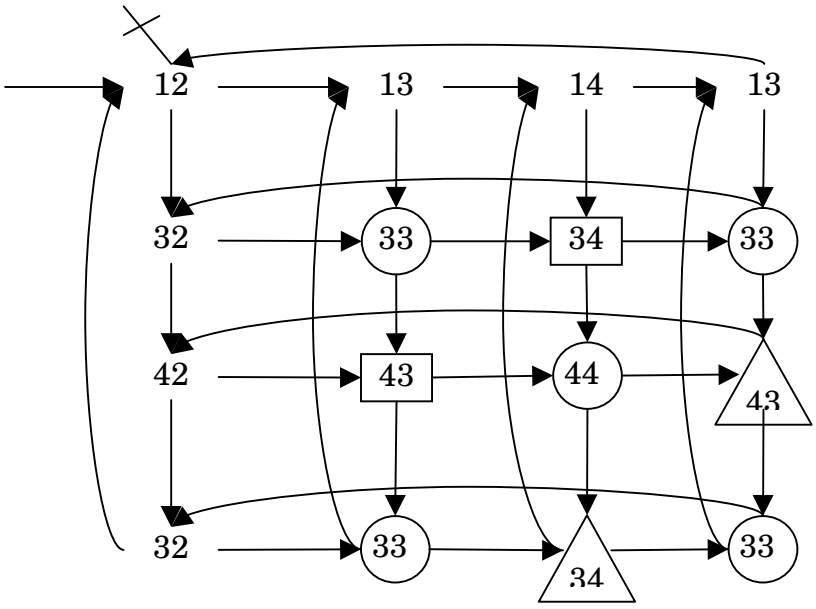


Fig. Réseau ferroviaire.

Les états entourés sont les états interdits. Les états encadrés sont ceux qui deviennent bloquants. Les états dans un triangles deviennent non atteignables. Au final on obtient l'automate suivant:

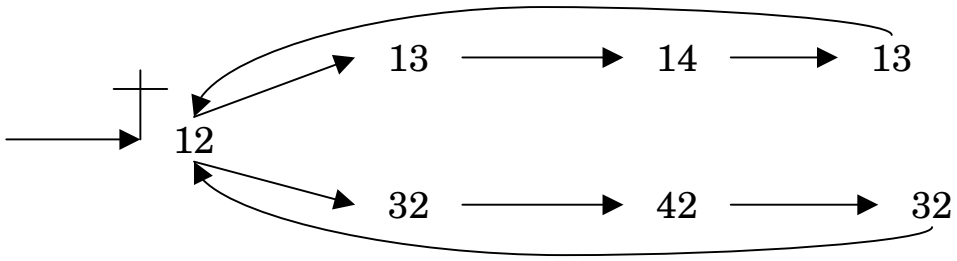


Fig. Réseau supervisé.

Conclusion.

En termes de puissance de modélisation, l'approche de Ramadge et Wonham a l'avantage de permettre la structure de choix au cours d'une trajectoire. Cette approche est particulièrement adaptée à la commande car les concepts propres à la théorie de la commande (notions d'évènements contrôlables ou incontrôlables, évènements forçables, langages commandables, …) y sont intégrés dès les bases. Une abondante littérature en témoigne (la référence [17] en est une synthèse par les initiateurs de cette théorie).
La principale carence de cette approche est de ne pas inclure le temps. C'est cette remarque qui a conduit au développement des automates temporisés, en fait il serait plus juste d'écrire « aux développements », en effet plusieurs modèles sont apparus, entre autres le modèle de Brandin et Wonham qui fait l'objet du

chapitre suivant et celui d'Alur et Dill qui sera présenté dans un chapitre ultérieur.

Chapitre 5

Automates temporisés : le modèle de Brandin et Wonham.

La théorie des automates et des langages commandables proposée par Ramadge et Wonham, si elle permet de modéliser de nombreux systèmes, présente un inconvénient majeur : le temps n'est pas pris en compte. C'est pourquoi d'autres travaux sont venus compléter cette théorie pour y inclure le caractère temporel. Ce chapitre est consacré au modèle de Systèmes à Evénements Discrets Temporisés (SEDT) développé par Brandin et Wonham dans [2], en continuation de [17]. Dans cette approche le temps est discrétisé et il apparaît sous la forme d'un événement appelé *tick*.

1 Définition d'un SEDT.

Définition 5.1 (générateur actif) Le SEDT est construit autour d'un automate classique, a priori indépendant du temps. C'est le générateur actif, noté G_{act} et défini par :

$$G_{act} = <\Sigma_{act}, A, \delta_{act}, a_0, A_m>,$$

où Σ_{act} est un alphabet fini, A l'ensemble des états du système, δ_{act} la fonction de transition du système : $\Sigma_{act} \times A \rightarrow A$ (δ_{act} est une fonction partielle), a_0 l'état initial (on prendra des automates déterministes), A_m l'ensemble des états marqués.

Le temps est introduit en ajoutant une horloge temps réel au générateur actif. Elle génère régulièrement l'événement *tick*, dont deux occurrences successives sont exactement séparées d'une unité de temps. Le temps ainsi interprété est discrétisé, on utilise un compteur entier des occurrences de *tick* ($\forall\, t \in 3^+$, $\forall\, t \in [n, n+1[$ avec $n \in \angle$, on a compt(t) = n). Les événements sont supposés instantanés (si on veut vraiment rendre compte de la durée d'une action il suffit de définir deux événements, un premier pour le début de l'action et un deuxième pour sa fin). Quand un événement survient dans l'intervalle temporel $[n, n+1[$, on considère qu'il apparaît à n unités de temps. Si différents événements surviennent entre deux *tick*, on les classe par ordre chronologique sans autre information temporelle.

Définition 5.2 (événement temporisé) Les événements sont des événements temporisés, en effet pour chacun d'eux on introduit une fenêtre temporelle à l'intérieur de laquelle sa survenue est possible. A chaque événement σ de Σ_{act} on associe un intervalle défini par une borne inférieure $i_\sigma \in \angle$ et une borne supérieure $s_\sigma \in \angle$. L'événement doit se produire entre ces deux limites, autrement

dit au moins i_σ *tick* et au plus s_σ *tick* doivent s'écouler entre deux occurrences de σ (si celui ci reste autorisé durant cette période). On définit ainsi l'ensemble des événement temporisés :

$$\Sigma_{temp}=\{(\sigma, i_\sigma, s_\sigma) \mid \sigma\in\Sigma_{act}\}.$$

De plus, on partitionne Σ_{act} en distinguant deux types d'événements : les événements lointains appartenant à Σ_{loin} et les événements futurs appartenant à Σ_{fut}. Un événement futur doit impérativement se produire entre deux limites temporelles bornées, autrement dit pour $\sigma \in \Sigma_{fut}$, on a $0\leq i_\sigma\leq s_\sigma<\infty$. Tandis qu'un événement lointain doit survenir après une certaine durée, mais sans limite supérieure, autrement dit pour $\sigma \in \Sigma_{loin}$, on a $0\leq i_\sigma<\infty$.

$$\Sigma_{act}=\Sigma_{loin}\cup\Sigma_{fut}.$$

Définition 5.3 (intervalle compteur) Pour chaque événement $\sigma\in\Sigma_{act}$ on associe un intervalle de ∠, noté T_σ, appelé intervalle compteur, avec :

$$\begin{cases} T_\sigma = [0, s_\sigma], si\ \sigma \in \Sigma_{fut}, \\ T_\sigma = [0, i_\sigma], si\ \sigma \in \Sigma_{loin}. \end{cases}$$

On peut introduire à présent la notion d'état étendu.

Définitions 5.4 (état étendu, fonction compteur) Un état étendu est un élément de la forme $q=\{a,\{t_\sigma \mid \sigma\in\Sigma_{act}\}\}$, où $a\in A$ et $t_\sigma\in T_\sigma$, a est appelé la partie active de q. La fonction t_σ est un compteur de σ qui associe à l'événement σ une valeur entière de l'intervalle compteur. Les compteurs évoluent selon la fonction de transition δ et permettent de savoir dans quel horizon temporel σ doit ou peut survenir. L'état initial q_0 est donné par $q_0=\{a_0,\{t_{\sigma 0} \mid \sigma\in\Sigma\}\}$, où $t_{\sigma 0}$ est la valeur par défaut du compteur de σ, définie par :

$$\begin{cases} t_{\sigma 0} = s_\sigma, si\ \sigma \in \Sigma_{fut}, \\ t_{\sigma 0} = i_\sigma, si\ \sigma \in \Sigma_{loin}. \end{cases}$$

Définition 5.5 (SEDT). Le SEDT associé à G_{act}, noté G, est défini par :

$$G=<Q,\Sigma,\delta,q_o,Q_m>,$$

où Q est l'ensemble des états, $\Sigma=\Sigma_{act}\cup\{\text{tick}\}$ est l'alphabet des événements, δ la fonction de transition $(\Sigma\times Q\rightarrow Q)$, q_0 est l'état initial et Q_m l'ensemble des états marqués. En fait on peut écrire : $G=G_{act}\cup\{tick\}$.

Définitions 5.6 (événements admissibles, inadmissibles, permis, interdits). Soient q un état de G, et $\sigma\in\Sigma$, on définit les événements admissibles, inadmissibles, permis et interdits par :

$$\begin{cases} si\ \delta(\sigma,q)!,\ \sigma\ est\ \text{admissible}\ par\ le\ SEDT, \\ \quad si\ \overline{\delta(\sigma,q)!},\ \sigma\ est\ \text{inadmissible}, \end{cases}$$

$$\begin{cases} si\ \delta_{act}(\sigma,q)!,\ \sigma\ est\ \text{permis}\ par\ le\ SED, \\ \quad si\ \overline{\delta_{act}(\sigma,q)!},\ \sigma\ est\ \text{interdit}. \end{cases}$$

Pour que le système temporisé admette un événement il faut que le système non temporisé l'autorise, i.e. un événement ne peut être admissible que s'il est permis.

Un événement futur permis est admissible si son compteur est égal à une valeur entière de l'intervalle $[s_\sigma\text{-}i_\sigma, 0]$, si le compteur est nul on dit que l'événement est *imminent*, c'est à dire qu'il doit obligatoirement se produire avant le prochain *tick*. Un événement permis et inadmissible est dit *en attente*.
Un événement lointain permis est admissible si son compteur est nul (il n'est, par définition jamais imminent).

2 Fonctionnement des SEDT.

Si le système G est en l'état $q=\{a,\{t_{\sigma_1},\ldots,t_{\sigma_2}\}\}$ et que $\sigma\in\Sigma$ survient, la transition est franchie si et seulement si une des trois conditions est vérifiée :
1. $\sigma=tick$ et $\forall\tau\in\Sigma_{fut}$, $t_\tau>0$,
2. $\sigma\in\Sigma_{fut}$ et $\delta_{act}(\sigma,a)$! et $0\le t_\sigma\le s_\sigma\text{-}i_\sigma$,
3. $\sigma\in\Sigma_{loin}$ et $\delta_{act}(\sigma,a)$! et $t_\sigma=0$,

Autrement dit si 1.) $\sigma=tick$ et qu'aucun événement n'est imminent ou, 2.) si un événement futur σ est permis et que i_σ unités de temps se sont écoulées depuis sa dernière occurrence ou, 3.) idem pour un événement lointain.

De plus si la transition est franchie on atteint l'état $q'=\{a',\{t'_{\sigma_1},\ldots,t'_{\sigma_2}\}\}$ déduit de q et de la nature de σ : si $\sigma=tick$, la partie active n'est pas modifiée et on décrémente les compteurs des événements permis (sauf ceux qui sont déjà à zéro) et $\sigma\neq tick$, la partie active de q' est donnée par $a'=\delta_{act}(\sigma,a)$ et les compteurs des événements interdits en a', ainsi que celui de σ sont remis à leur valeur par défaut, les autres ne sont pas modifiés.
On rassemble les règles d 'évolution dans l'algorithme suivant :

- Si $\sigma=tick$
  - $a'=a$
  - si $\tau\in\Sigma_{fut}$, $t'_\tau=\begin{cases} t_\tau\text{-}1 \text{ si } \delta_{act}(\tau,a) \text{ !} \\ S_\tau \text{ sinon} \end{cases}$
  - si $\tau\in\Sigma_{loin}$, $t'_\tau=\begin{cases} max(0,t_\tau\text{-}1) \text{ si } \delta_{act}(\tau,a) \text{ !} \\ i_\tau \text{ sinon} \end{cases}$

- Si $\sigma\in\Sigma_{act}$
  - $a'=\delta_{act}(\sigma,a)$
  - si $\sigma\in\Sigma_{fut}$, $t'_\sigma=s_\sigma$
  - si $\sigma\in\Sigma_{loin}$, $t'_\sigma=i_\sigma$
  - si $\tau\neq\sigma\in\Sigma_{fut}$, $t'_\tau=\begin{cases} t_\tau \text{ si } \delta_{act}(\tau,a') \text{ !} \\ s_\tau \text{ sinon} \end{cases}$
  - si $\tau\neq\sigma\in\Sigma_{loin}$, $t'_\tau=\begin{cases} t_\tau \text{ si } \delta_{act}(\tau,a') \text{ !} \\ i_\tau \text{ sinon} \end{cases}$

On impose que le SEDT ne doit pas comporter de transition bouclant sur un même état étendu, à l'occurrence d'un mot de Σ_{act}^{+} (c'est à dire tout événement autre que *tick* ou l'absence d'événement). Dans le cas contraire rien n'empêche de boucler indéfiniment en préemptant le *tick*, autrement dit cela reviendrait à arrêter le temps ! On impose donc :

$$\forall q \in Q,\ \forall \sigma \in \Sigma_{act}^{+},\ \delta(\sigma,q) \neq q.$$

3 Programme de temporisation de SEDT, sous Matlab.

On trouvera en annexe un programme Matlab, que j'ai fait, qui permet de temporiser un automate. Il suffit de donner l'automate sous forme matricielle et de donner une matrice renseignant sur la nature des événements (lointain ou futur) et les limites temporelles. Le programme renvoie alors deux matrices, l'une est l'automate temporisé et l'autre détaille les états étendus en indiquant l'état correspondant du graphe non temporisé et les valeurs des compteurs. La forme matricielle, et non graphique des résultats ainsi que la simple numérotation des événements n'est pas très conviviale mais ça évite d'alourdir la programmation.

4 Composition de SEDT.

Comme dans le cas des automates non temporisés il est utile de définir un produit de composition qui permet de construire des automates complexes à partir de plusieurs de taille plus raisonnable.

Définition 5.7 (produit de composition de SEDT) Soient G_1 et G_2 deux SEDT, avec $G_i=G_{i,act} \cup \{tick\},\ i \in \{1,2\}$, la fonction de transition δ de $G=comp(G_1, G_2)$ est donnée par :

- si $\sigma \in (\Sigma_{1,act}-\Sigma_{2,act}) \cap (\Sigma_{2,act}-\Sigma_{1,act})$, $\delta_{act}((a_1,a_2),\sigma)$! ssi $\delta_{1,act}(a_1,\sigma)$! et $\delta_{2,act}(a_2,\sigma)$!
  et $\delta_{act}((a_1,a_2),\sigma) = ((\delta_{1,act}(a_1,\sigma),\delta_{2,act}(a_2,\sigma))$,
- si $\sigma \in \Sigma_{1,act}$ et $\sigma \notin \Sigma_{2,act}$ (ou que $\delta_{2,act}$ n'est pas définie en (a_2,σ)),
  $\delta_{act}((a_1,a_2),\sigma) = ((\delta_{1,act}(a_1,\sigma),a_2)$
- si $\sigma \in \Sigma_{2,act}$ et $\sigma \notin \Sigma_{1,act}$ (ou que $\delta_{1,act}$ n'est pas définie en (a_1,σ)),
  $\delta_{act}((a_1,a_2),\sigma) = (a_1,(\delta_{1,act}(a_1,\sigma))$.

Les bornes temporelles de σ restent les mêmes si $\sigma \in (\Sigma_{1,act}-\Sigma_{2,act}) \cup (\Sigma_{2,act}-\Sigma_{1,act})$, en revanche pour les événements $\sigma \in \Sigma_{1,act} \cap \Sigma_{2,act}$ on les redéfinit par :

$$(i_\sigma, s_\sigma) = (\max(i_{\sigma,1}, i_{\sigma,2}), \min(s_{\sigma,1}, s_{\sigma,2})).$$

Dans le cas où $i_\sigma > s_\sigma$, la composition est impossible.

Remarque : en général on n'a pas le même résultat si on temporise la composition de deux automates et si on compose les deux automates après les

avoir temporisés. On obtient le même résultat si tous les événements en commun aux deux automates ont les mêmes bornes temporelles.

5 Commande des SEDT.

5.1 Partition de Σ_{act}.

Le modèle est construit sur les bases de celui de Ramadge et Wonham (R&W) exposé dans [17], on peut donc reprendre les notions liées à la commandabilité des SED. On partitionne Σ_{act} par $\Sigma_{act}=\Sigma_c \cup \Sigma_n$, où Σ_c et Σ_n sont les sous ensembles d'événements respectivement contrôlables et incontrôlables. On rappelle qu'un événement est dit contrôlable quand on peut différer sa survenue et incontrôlable quand on ne peut l'empêcher. Le *tick* est défini contrôlable car il est préempté par les événements imminents car même s'il ne peut être repoussé dans le temps il l'est dans la succession chronologique (cela est possible car les événements sont supposés instantanés).

5.2 Superviseurs.

La supervision est toujours conçue comme un outil qui permet de limiter les trajectoires du système afin de ne pas le laisser sortir d'une spécification. Le contrôle est donc une réduction du langage reconnu par l'automate.
La commande du système se fait par l'interdiction et le forçage de certains événements. Un événement, σ, est forçable quand on peut imposer sa survenue, cette propriété, notée $\sigma \in \Sigma_{for}$, est totalement distincte de celle de commandabilité. En revanche l'interdiction suppose que l'événement soit contrôlable. On a donc $\Sigma_c = \Sigma_{int} \cup \{tick\}$.
Par construction, un événement futur est susceptible de devenir imminent, c'est à dire obligé de survenir avant le *tick* suivant. Pour éviter d'avoir à interdire un événement imminent (ce qui reviendrait à arrêter le temps ou ne plus respecter l'algorithme), on supposera que seuls les événements lointains peuvent être interdits :

$$\Sigma_{int} \subset \Sigma_{loin.}$$

Définition 5.8 ($Adm_G(s)$) On définit l'ensemble, noté $Adm_G(s)$, des événements admissibles par G en $s \in L(G)$ par :

$$Adm_G(s)=\{\sigma \in \Sigma \mid s\sigma \in L(G)\}.$$

Un superviseur ne doit pas mener à une situation de blocage et ne peut pas empêcher la survenue d'événements incontrôlables, en d'autres termes doit permettre l'occurrence des éléments de Σ_n (auquel il convient d'ajouter *tick* si aucun $\sigma \in \Sigma_{for}$ n'est admissible, car alors rien ne peut préempter *tick*). Plus rigoureusement on donne la définition suivante d'un superviseur.

Définition 5.9 (superviseur) Un superviseur V sur un générateur G, d'alphabet Σ est une fonction $V : L(G) \to P(\Sigma)$, $s \mapsto V(s)$, telle que :

$$\forall s \in L(G) \begin{cases} V(s) \cap Adm_G(s) \neq \emptyset, \\ V(s) \supset \begin{cases} \Sigma_n \cup \{tick\}, si V(s) \cap \Sigma_{for} = \emptyset, \\ \Sigma_n \quad , si V(s) \cap \Sigma_{for} \neq \emptyset. \end{cases} \end{cases}$$

V(s) représente l'ensemble des événements autorisés après une trajectoire *s*, pour qu'un événement autorisé survienne il faut aussi qu'il soit admissible, i.e. $\sigma \in V(s) \cap Adm_G(s)$.

Propriété 5.1 (fermeture de la famille des superviseurs) Soit S_G la famille des superviseurs de *G,* S_G est fermée par l'union (autrement dit si *V* et *V'* $\in S_G$ alors $V \cup V' \in S_G$)

On note G/V le système *G* bouclé par le superviseur *V*. les résultats établis pour les SED par R&W restent vrais pour les SEDT. Ainsi on peut considérer les automates sous l'angle de la théorie des langages, le système supervisé devient $L(G/V)$. $L(G/V) \subset L(G)$ est construit itérativement de la manière suivante :

$$\begin{cases} \varepsilon \in L(G/V), \\ s\sigma \in L(G/V) \text{ ssi } s \in L(G/V), \sigma \in V(s), \ s\sigma \in L(G). \end{cases}$$

Le langage marqué du système supervisé est défini par $Lm(G/V)=L(G/V) \cap L_m(G)$, et G/V est non bloquant ssi : $L(G/V) = \overline{L_m(G/V)}$.

5.3 Résolution du problème de commande.

Le problème de commande consiste à savoir si pour un générateur *G* et une spécification (donnée sous forme d'un langage ou d'un automate), il existe un superviseur *V* tel que, G/V respecte la spécification. Pour cela il est utile de définir la notion de commandabilité d'un langage.

Définition 5.10 (commandabilité d'un langage) On dit qu'un langage $K \subset L(G)$ est commandable si $\forall s \in Pref(K)$ (où $Pref(K)$ est la fermeture préfixielle de *K*) :

$$Adm_K(s) \supset \begin{cases} Adm_G(s) \cap (\Sigma_n \cup \{tick\}), & si \quad Adm_K(s) \cap \Sigma_{for} = \emptyset, \\ Adm_G(s) \cap \Sigma_n, & si \quad Adm_K(s) \cap \Sigma_{for} \neq \emptyset, \end{cases}$$

autrement dit tout événement incontrôlable autorisé par *G* après une trajectoire *s*, doit être autorisé par le superviseur.

On donne maintenant les deux principaux résultats énoncés par Brandin et Wonham concernant la commande des SEDT, ils sont analogues à ceux présentés dans le chapitre précédent. Le premier est appelé le Problème de Supervision Non-bloquante (PSN).

Théorème 5.1 (résolution du PSN) Soit $K \subset L_m(G)$ non vide. Il existe un superviseur *V,* non bloquant sur le générateur *G* tel que $L_m(G/V)=K$ ssi

$$\begin{cases} K \text{ est commandable par rapport à } G, \\ K \text{ est } L_m(G) \text{ fermé}. \end{cases}$$

Rappel : K est $L_m(G)$ fermé signifie $K = \overline{K} \cap L_m(G)$.

On peut également chercher un superviseur qui mène le système en des états marqués particuliers. On parle alors de Problème de Supervision Non-bloquante Marquée et on a le résultat suivant :

Théorème 5.2 (résolution du PSNM) Soit $M \subset L_m(G)$ et $K \subset L_m(G)$ non vides. Il existe un superviseur V non-bloquant tel que $L_m(G/V) \subset L(G/V) \cap M$ et $L_m(G/V)=K$ ssi :

K est commandable par rapport à G.

Généralement, on donne la spécification sous forme d'automate. pour appliquer le théorème 5.1, il faut composer les automates du système et de la spécification, cette composition est l'équivalent de l'intersection des langages. On vérifie que l'automate obtenu n'est pas bloquant. Enfin on applique l'algorithme de Kumar, en supprimant les états défendus et faiblement défendus. On rappelle qu'un état q est défendu si la survenue d'un événement incontrôlable lorsque le système est en q mène le système hors la spécification, de même un état q est faiblement défendu si une séquence d'événements incontrôlables survenant en q mène le système en un état défendu. Pour les automates temporisés il convient de considérer *tick* comme un événement incontrôlable lors de la détermination des états défendus et faiblement défendus, en effet sinon il faudrait arrêter le temps pour superviser le système.

6 Exemple du réseau ferroviaire.

On applique la méthode de temporisation aux automates modélisant le fonctionnement de chaque train. Les automates non temporisés sont les suivants.

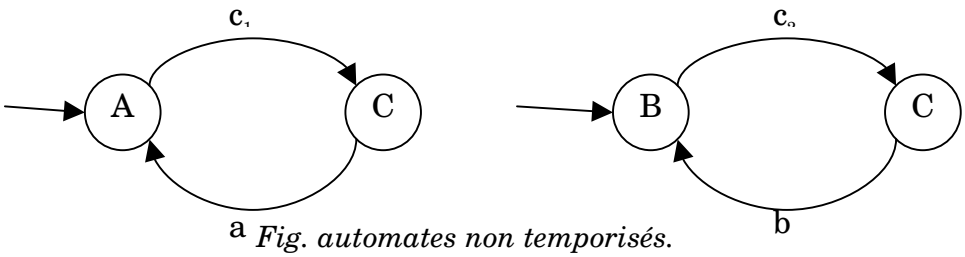


Fig. automates non temporisés.

Rien ne forçant un train à pénétrer sur un tronçon avant une date donnée, on décide que les événements sont lointains, la borne inférieure temporelle est donnée par le temps de parcours des tronçons, on a donc :

$\Sigma_{temp}=\{(a,1,\infty),(b,1,\infty),(c_1,2,\infty),(c_2,1,\infty)\}$.

On a donc les automates temporisés suivants.

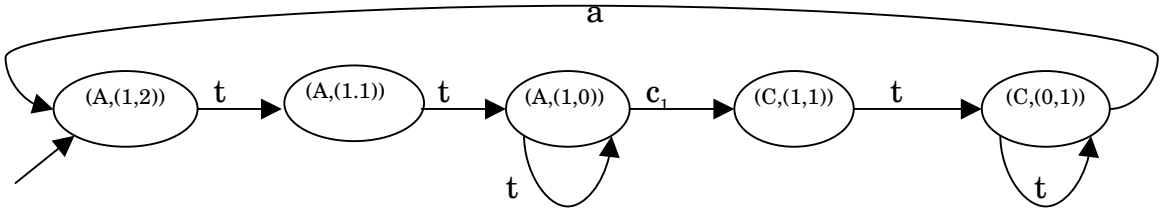


Fig. Automate temporisé, train1.

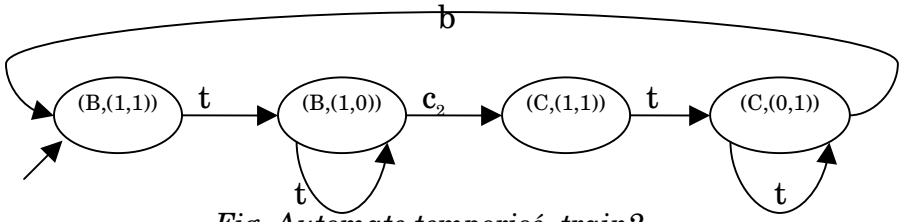


Fig. Automate temporisé, train2.

On compose les deux automates obtenus, dans ce cas étant donné qu'aucun événement n'est commun au deux automates on peut commencer par composer avant de temporiser, néanmoins il est plus aisé de temporiser des petits automates avant de composer car le nombre de compteurs est plus faible. Le produit de composition donne l'automate suivant, dans un soucis de lisibilité on ne fait pas apparaître les compteurs. les indices des noms de place indiquant le temps écoulé depuis l'entrée sur le tronçon.

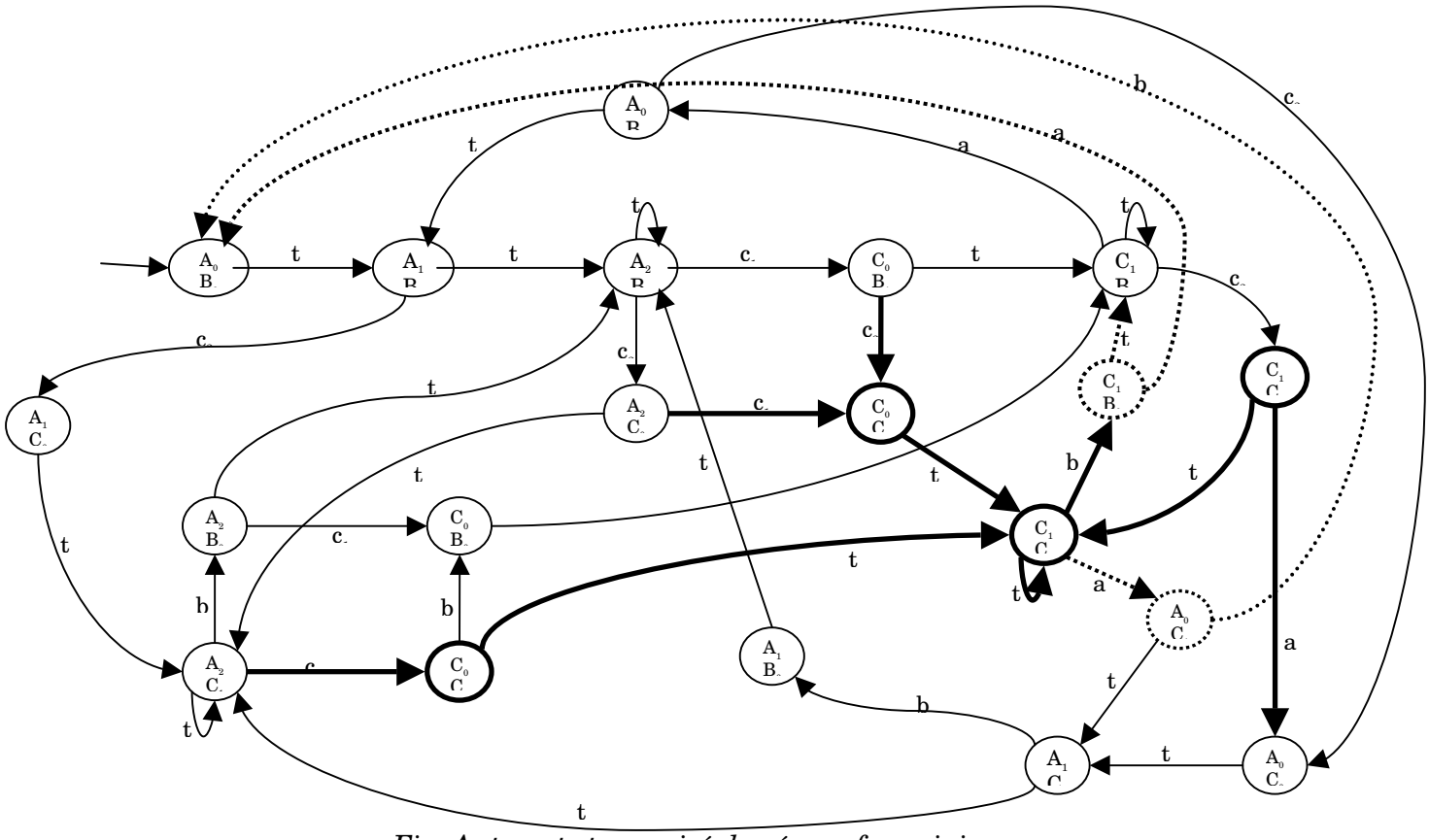
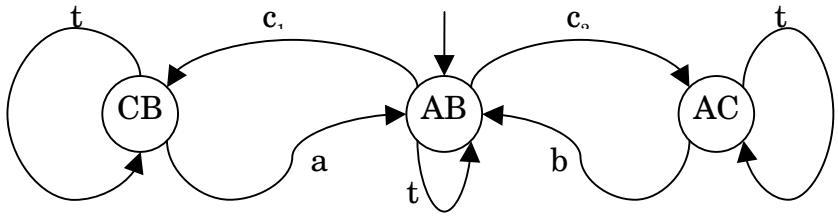


Fig. Automate temporisé du réseau ferroviaire.

On trouvera en annexe une vérification de l'automate temporisé, non supervisé, a titre d'exemple du programme de temporisation de SED.

La spécification est de ne pas se trouver dans la situation où les deux trains occupent le tronçon C, ce qui se traduit par l'automate suivant.

Fig. Automate de la spécification.



On peut supposer que les conducteurs doivent rester maître de leur véhicule on considère donc que les événements c_1 et c_2 sont contrôlables. On peut vérifier la contrôlabilité du langage K. La première condition est vérifée car $\Sigma_n=\varnothing$. La seconde également en effet on a :

$$K = (c_1 a + c_2 b)^*, donc\ \overline{K} = (c_1 a + c_2 b)^* + c_1 + c_2 + \varepsilon,$$
$$\overline{K} \cap L_m = (c_1 a + c_2 b)^* = K.$$

Les états en gras sont les états interdits, c'est à dire hors spécification et qui disparaissent lors de la composition avec l'automate de la spécification. On a représenté en pointillés les états qui ne sont accessibles que par des états interdits. Il n'y a pas d'état défendu ou faiblement défendu, on peut alors décrire le système supervisé par l'automate suivant.

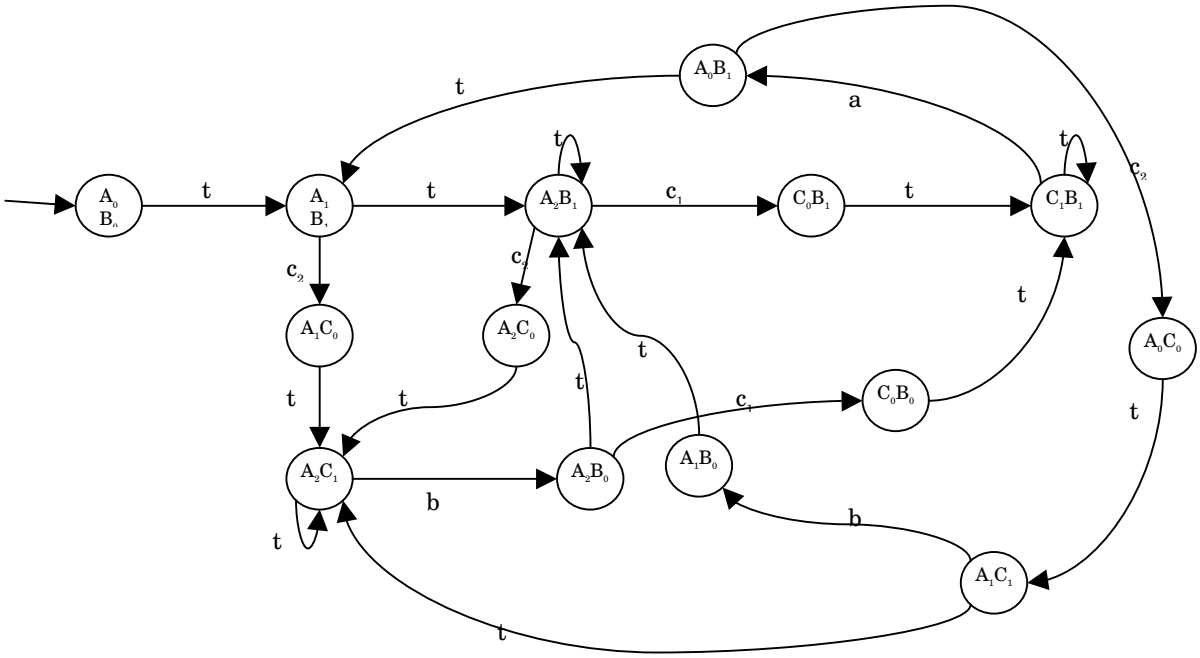


Fig. Automate temporisé supervisé.

Dans le cas où on considère que les événements c_1 et c_2 son incontrôlables les états C_0B_1, C_1B_1, A_2C_0, C_0B_1 et A_2C_1 deviennent défendus et donc A_1C_0 et A_1C_0 et

A_0B_0 deviennent faiblement défendus. Comme A_0B_1 n'est plus accessible A_0C_0, A_1B_0 et A_1C_1 ne le sont plus également. Au final l'automate est donc réduit à néant, autrement dit la supervision n'est guère possible.

On peut vérifier qu'il est possible d'atteindre des régimes périodiques qui autorisent les passages alternés des trains 1 et 2 sur le tronçon C, en particulier les deux régimes mis en évidence dans le chapitre 0. En effet après le transitoire décrit par (*tick tick* c_1 *tick)*, on peut avoir deux régimes périodiques différents. Celui correspondant à *(*c_1 *tick a* c_2 *tick b tick)* a une période de 3 unités de temps et permet d'alterner exactement les passages des deux trains. Alors que le régime décrit par (c_2*tick b* c_1 *tick a* c_2 *tick b tick*) a une période plus grande d'une unité de temps et autorise deux passages du train 2 pour un seul du train 1, le temps moyen de parcours du circuit 1 en est donc diminué. On peut aussi remarquer un autre régime périodique alternant les passages des deux trains en 3 unités de temps. Il est atteint après le transitoire (*tick tick* c_1*tick)* et le motif périodique est (a *tick* c_2 *tick b* c_1 *tick*).

Conclusion

La représentation par automate temporisé permet de modéliser la plupart des processus. Elle garde la valeur générique des automates en y ajoutant la prise en compte du caractère temporel. De plus la construction du modèle permet d'appliquer les résultats établis dans le cadre des automates non temporisés :commande supervisée, calculs d'éléments suprémaux... L'algorithme d'évolution des SEDT est relativement simple à mettre en application, y compris dans le cas de systèmes de grandes dimensions, quitte à le programmer. En revanche ce modèle impose de discrétiser le temps, c'est à dire qu'on perd une partie de l'information. On pourrait subdiviser le temps mais cela conduit vite à des automates trop lourds. De plus cette modélisation demande de connaître une grande quantité d'information sur la survenue des événements. On peut essayer de s'en accommoder en prenant $i_\sigma=0$, $s_\sigma=\infty$ pour tous les événements pour lesquels on ne dispose pas d'information précise, mais là encore l'automate risque de devenir trop complexe. Enfin il est difficile de modéliser les phénomènes de synchronisation entre plusieurs événements (alors que les GET, ou les RdPT le permettent aisément). En effet les compteurs de chaque événement $\sigma \in \Sigma$ évoluent par rapport à la dernière occurrence de σ mais ne tiennent pas compte des autres événements.

Il existe d'autres modélisations par automate temporisé, souvent issues des travaux sur les systèmes temps réel. Le temps y est généralement dense et chaque événement a une horloge ou un compteur qui peut être comparé aux autres compteurs. On dépasse donc deux des principales limitations du modèle de Brandin et Wonham. Le modèle le plus utilisée est celui d'Alur et Dill [1], il est présenté dans le chapitre suivant.

Chapitre 6

Automates temporisés : le modèle d'Alur et Dill.

L'approche utilisée par R. Alur et D.L. Dill s'inspire des systèmes temps réel. Le temps est donc dense au lieu d'être discrétisé. Les événements ne sont plus affectés de compteurs à valeurs dans $\angle$, mais d'horloges temps réel, qui sont accessibles et modifiables en tout point de l'automate. Cette nouvelle approche permet donc de prendre en compte les phénomènes de synchronisation, il suffit pour cela de faire des tests simultanés sur plusieurs horloges. Du point de vue théorique, on manipule des langages infinis, notés ω-langages. Du fait de l'introduction d'un temps dense, les automates acquièrent une double nature : à la fois continus et discrets. En effet on peut construire un automate, appelé automate région, dont chaque état correspond à un intervalle de $3^{|X|}$ (|X| est le nombre d'horloges du système) à l'intérieur duquel le système évolue continûment. En revanche, le franchissement d'une transition traduit une évolution discrète du système (saut d'état).

1 ω-automates.

Dans ce paragraphe on définit les notions utiles à la compréhension du modèle d'Alur et Dill, en particulier les concepts de langages infinis.

Définition 6.1 (ω-langage et ω-automate) Un ω-langage est composé de mots infinis, il peut être fait sur un alphabet fini auquel cas le ω-langage est un élément de Σ^{ω} (ensemble de tous les mots de longueur infinie sur Σ). Un ω-automate donne une représentation finie de certains ω-langages.

On peut citer les automates de Büchi et de Muller qui seront définis plus loin.

Définition 6.2 (table de transitions) Une table de transitions est un quadruplet $A=<\Sigma,S,S_0,E>$, où Σ est un alphabet, S un ensemble fini d'états de l'automate, S_0 l'ensemble des états initiaux et $E \subset S \times S \times \Sigma$ l'ensemble des arcs : $<s,s',a> \in E$ signifie qu'un arc d'étiquette *a* va de *s* en *s'*.

Définition 6.3 (automate déterministe) Un automate est dit déterministe ssi il n'a qu'un état initial et que pour chaque état *s* et chaque événement *a* il existe au plus un arc d'étiquette *a* sortant de *s*.

Définition 6.4 (chemin) Soit un mot $\sigma=\sigma_1\sigma_2\ldots\in\Sigma^*$. $r = s_0 \xrightarrow{\sigma_1} s_1 \xrightarrow{\sigma 2} s_2 \xrightarrow{\sigma_1} \ldots.$ est un chemin de l'automate sur le mot σ si $s_0\in S_0$ et si $\forall i>1$ $<s_i,s_{i+1},a_{i+1}>\in E$. On note *inf(r)* l'ensemble des états qui sont répétés un nombre infiniment grand de fois le long de *r*, *inf(r)={s | s=s_i, pour un nombre infini de i≥0}*.

On définit ici deux classes d'ω-automates particulièrement utilisées, les automates de Büchi et de Muller.

Définition 6.5 (automate de Büchi, langage ω-régulier) Un automate de Büchi est défini par un automate $<\Sigma,S,S_0,E>$ et un ensemble $F\subseteq S$. Un chemin *r* est admissible par un automate de Büchi ssi $inf(r)\cap F\neq\emptyset$. un langage est ω-régulier ssi il est reconnu par un automate de Büchi.

Définition 6.6 (automate de Muller) Un automate de Muller est défini par un automate $<\Sigma,S,S_0,E>$ et une famille $F\subseteq 2^S$ (ensemble des sous-ensemble de S). un chemin *r* est admissible par un automate de Muller ssi $inf(r)\subset F$.

Notation. La répétition infinie d'un événement est notée ω, par exemple le langage reconnu par l'automate ci dessous, qui admet les chemins comportant une suite finie de *a* ou de *b* suivie d'une répétition infinie de *a,* est noté : $L=(a+b)^*a^\omega$. Pour $F=\{a\}$, c'est un automate de Büchi ou de Muller. On note qu'il n'est pas déterministe, cependant on peut le déterminiser en appliquant la méthode donnée dans le chapitre 4.

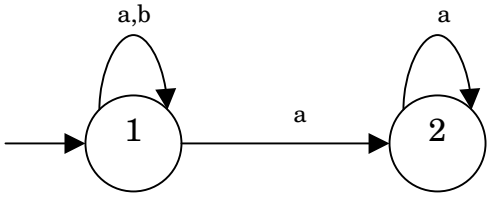


Fig. ω-automate non déterministe.

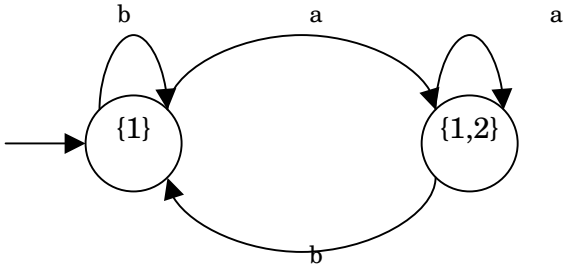


Fig. Automate déterministe.

Propriété 6.1 la classe des ω-langages est fermée pour les opérations booléennes.

Propriété 6.2 Les classes de langages acceptés par les automates de Büchi, par les automates de Muller ou par les automates déterministes de Muller sont égales

2 Automates temporisés.

Dans ce paragraphe après avoir imposé la croissance du temps on ajoute le caractère temporel aux automates, en définissant les langages temporisés, les tables de transitions temporisées,…

2.1 Langages temporisés.

Définition 6.7 (séquence temporelle) Une séquence infinie $\tau = \tau_1\tau_2\ldots$avec $\tau_i \in \Im$ pour tout i est une séquence temporelle ssi elle est strictement croissante $\tau_{1+1} > \tau_i$ $\forall i > 0$ et progressive i.e. $\forall t \in \Im, \exists i \; tq \; \tau_i = t$.

Définition 6.8 (mot temporisé) un mot temporisé est une paire (σ,τ), où $\sigma \in \Sigma$ et τ est une séquence temporelle, τ_i est la date de l'occurrence de σ_i (parfois on limite la condition de monotonie de (τ_i) à une croissance simple, ce qui autorise plusieurs événements à survenir en même temps, en accord avec leur instantanéité). Un langage temporisé est un ensemble de mots temporisés.

Notation. Soit L un langage temporisé, on note *Untime(L)* le ω-langage formé des $\sigma \in \Sigma$ tels que $\exists \tau$ *tel que* $(\sigma,\tau) \in L$.

2.2 Tables de transitions temporisées.

Le modèle inclut une horloge temps réel générale à partir de laquelle d'autres horloges (à valeurs réelles positives) sont définies. Ces autres horloges sont supposées connues et modifiables en tout point de l'automate. On appelle X l'ensemble de ces horloges. Outre les étiquettes usuelles, on peut associer différents éléments aux arcs reliant les états. Ils sont de deux types :

- des conditions sur les horloges, ou *contraintes temporelles* (une horloge doit être égale, inférieure ou supérieure à un nombre,…), repérées par le symbole « ? » ;
- la remise à zéro d'une ou plusieurs horloges, notée « $x := 0$ ».

Plus rigoureusement on définit les contraintes d'horloge ou temporelles et la table de transition temporisée comme suit :

Définition 6.9 (contrainte d'horloge) L'ensemble $\Phi(x)$ des contraintes d'horloges δ est défini par des combinaisons de : $\{x \geq c\}$ et $\{x \leq c\}$, puis par combinaisons et négations de ces éléments, x est une horloge et c est un rationnel.

Définition 6.10 (table de transitions temporisées) Une table de transitions temporisées (TTT) est un quintuplet $<\Sigma, S, S_0, C, E>$, où Σ est un alphabet fini, S un ensemble d'états, S_0 un ensemble d'états initiaux, C un ensemble fini d'horloges et $E \subseteq S \times S \times \Sigma \times 2^C \times \Phi(C)$ donne l'ensemble des transitions de l'automate, un arc $<s,s',a,\lambda,\delta>$ représente le passage de s à s' à l'occurrence de a si δ est vérifiée, λ est le sous-ensemble d'horloges remises à zéro.

Définition 6.11 (trajectoire) Une trajectoire r notée $(\bar{s}, \bar{n})$ d'une TTT $<\Sigma, S, S_0, C, E>$ sur un mot temporisé (σ,τ) est une séquence infinie de la forme :

$$r : < s_0, \nu_0 > \xrightarrow[\tau_1]{\sigma_1} < s_1, \nu_1 > \xrightarrow[\tau_2]{\sigma_2} < s_2, \nu_2 > \ldots$$

avec $s_i \in S$, $\nu_i \in [C \to \mathbb{R}]$ $\forall i$ (ν_i est une interprétation c'est à dire une fonction qui attribue une valeur à chaque horloge $x \in X$) et satisfaisant les deux conditions suivantes :

- $s_0 \in S_0$, $\nu_0(x) = 0$, $\forall x \in C$.
- $\forall i \geq 1$, $\exists < s_{i-1}, s_i, \sigma_i, \lambda_i, \delta_i > \in E$, *tel que* δ_i *soit vérifié et les horloges de* λ_i *remises à zéros*

Définitions 6.12 (automates de Büchi, de Muller temporisés, langage régulier teporisé) On peut définir une version temporisée des automates de Büchi (resp. de Muller), comme étant les automates temporisés augmentés d'une famille d'états F et admettant les trajectoires temporisées r telles que $inf(r) \cap F \neq \emptyset$ (resp. $inf(r) \subset F$). De même, on définit un langage régulier temporisé comme étant tel qu'il existe un automate de Büchi temporisé qui le reconnaisse.

Propriété 6.3 La classe des langages temporisés réguliers est fermée pour l'union et l'intersection (finie).

Propriété 6.4 La classe des langages reconnus par un automate de Büchi temporisé et celle des langages reconnus par un automate de Muller temporisé sont égales.

exemple.
On illustre ici les différentes notions définies plus haut.

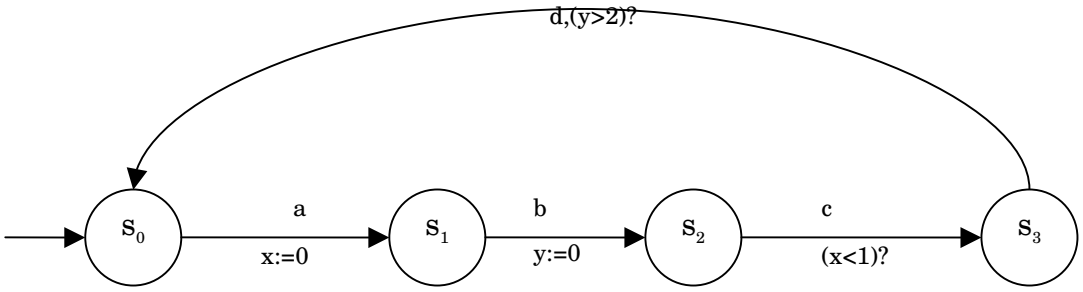


Fig. exemple d'automate temporisé.

Soit l'automate ci-dessus, d'état initial et final s_0, l'ensemble des horloges est $C=\{x,y\}$. La transition entre s_0 et s_1 met x à zéro, tandis que y est remise à zéro lors du franchissement de la transition entre s_1 et s_2. La transition entre s_2 et s_3 ne peut être franchie à l'occurrence de c qu'à la condition que x soit inférieure à 1, de même la dernière transition n'est franchie que si y est supérieure à 1. Le langage est composé des mots formés d'une répétition infinie de $(abcd)$, autrement dit $L=(abcd)^{\omega}$. En tenant compte des contraintes temporelles des transitions, on peut préciser le caractère temporel de L :

$$L=\{((abcd)^{\omega}, \tau) \mid \forall j, ((\tau_{4j+3} < \tau_{4j+1}+1) \cap (\tau_{4j+4} > \tau_{4j+2}+2))\},$$

autrement dit, le $4n+3^{ème}$ événement (un passage de s_2 à s_3) se produit moins d'une unité de temps après le $4n+3^{ème}$ événement (un passage de s_0 à s_2) et de même tous les occurrences de d doivent survenir plus de 2 unités de temps après celles de b.

3 Automates régions.

Dans ce paragraphe on construit un autre automate à partir de la table de transitions temporisées d'un automate temporisé, c'est l'automate région. Chaque état de ce nouvel automate correspond à un ensemble de points de fonctionnement de l'automate que l'on peut parcourir continûment.

3.1 Restriction de l'étude au cas des contraintes entières.

On ne modifie pas le comportement d'un automate temporisé si on multiplie toutes les contraintes temporelles par un même rationnel t, il suffit d'opérer une homotétie de rapport t aux horloges. Ainsi en prenant t égal au p.p.c.m. de toutes les contraintes apparaissant dans les transitions on se ramène au cas d'un automate qui a toutes ses contraintes entières.

Désormais, on se placera toujours dans ce cas de figure.

3.2 Régions temporelles et automates région.

Etant donné qu'en tout point de fonctionnement, le futur de l'automate est déterminé par l'état dans lequel il se trouve et les valeurs des différentes horloges, il est pertinent de regrouper toutes ces informations dans une même variable, appelée *état étendu*.

Définition 6.13 (état étendu) Pour une TTT $<\Sigma,S,S_0,C,E>$, un état étendu est une paire $<s, \nu>$, où $s \in \Sigma$ et ν est une interprétation des horloges de C.

Dans le cas de contraintes entières, seules les parties entières des horloges, noté $\lfloor \sim \rfloor$, sont utiles à la vérification des contraintes. Les parties fractionnelles, notées *fract*, sont utiles pour connaître l'ordre des changements de partie entières des différentes horloges. De ce fait un automate va avoir des trajectoires proches à partir de deux états étendu de même état et dont les horloges ont la même partie entière. On définit donc une classe d'équivalence sur l'ensemble des interprétations.

Définition 6.14 (région temporisée) Soit une TTT $<\Sigma,S,S_0,C,E>$, pour tout $x \in C$ soit c_x le plus grand entier tel que ($c_x \leq x$) ou($c_x \geq x$) soit une contrainte temporelle. La relation d'équivalence ~ est définie sur l'ensemble des interprétations des horloges de C par $\nu \sim \nu'$, ssi :

- $\forall x \in C, \lfloor \nu(x) \rfloor = \lfloor \nu'(x) \rfloor$ *où* $\nu(x) > c_x$ *et* $\nu'(x) > c_x$,
- $\forall (x,y) \in C^2$ *tq* $\nu(x) \leq c_x$ *et* $\nu(y) \leq c_y$, $fract(\nu(x)) \leq fract(\nu(y))$ *ssi* $fract(\nu'(x)) \leq fract(\nu'(y))$,
- $\forall x \in C$ *tq* $\nu(x) \leq c_x$, $fract(\nu(x))=0$ *ssi* $fract(\nu'(x))=0$.

Notation. On note $[\nu]$ la région temporelle contenant ν.

Définition 6.14 (automate région) Un automate région est défini par des états $\langle s, \alpha \rangle$, où $s \in S$ et α est une région temporelle, on a un arc d'étiquette a entre $\langle s, \alpha \rangle$ et $\langle s', \alpha' \rangle$ ssi il existe un arc d'étiquette a entre $\langle s, \nu \rangle$ et $\langle s', \nu' \rangle$, avec $\nu \in \alpha$ et $\nu' \in \alpha'$.
Pour faciliter la construction de cet automate il est utile de définir une relation de succession temporelle définie par :
Définition 6.15 (successeur temporel) Une région temporelle α' est un successeur d'une autre région temporelle α ssi pour tout ν de α il existe $t \in \mathbb{R}^+$ tel que $\nu + t \in \alpha'$.

Alur et Dill donnent au paragraphe 4.3 de [1] la manière de déterminer les successeurs temporels d'une région temporelle, ou d'un ensemble de régions temporelles proches. Plus simplement, est candidat à la succession toute région pouvant être atteinte par une translation le long de la droite $x_1 = x_2 = \ldots$

Exemple.

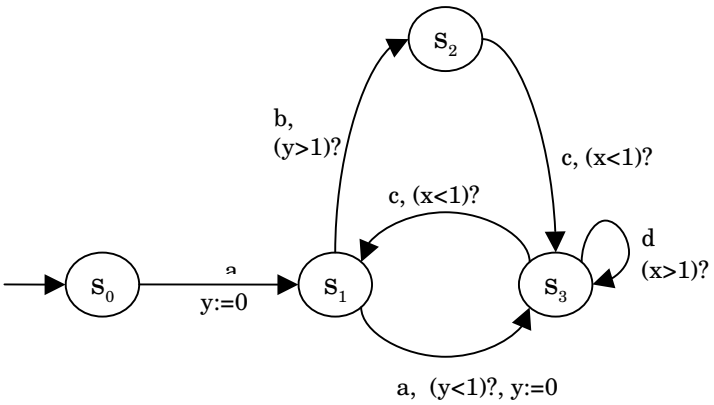


Fig. Exemple d'automate.

Soit l'automate ci-contre. Les contraintes maximales sur x et y sont *(x<1)* et *(y<1)*, on a donc les régions temporelles données par le graphe ci dessous.

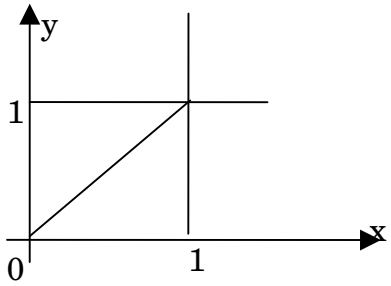


Fig. régions temporelles.

Les régions temporelles sont au nombre de 14, il y a 5 segments (par exemple x=0 et 0<y<1), 4 demi droites (par exemple x<1 et y=1), 2 polygones (par exemple 0<x<y<1) et trois portions de plan (par exemple x>1 et y>1).
On a alors l'automate région.

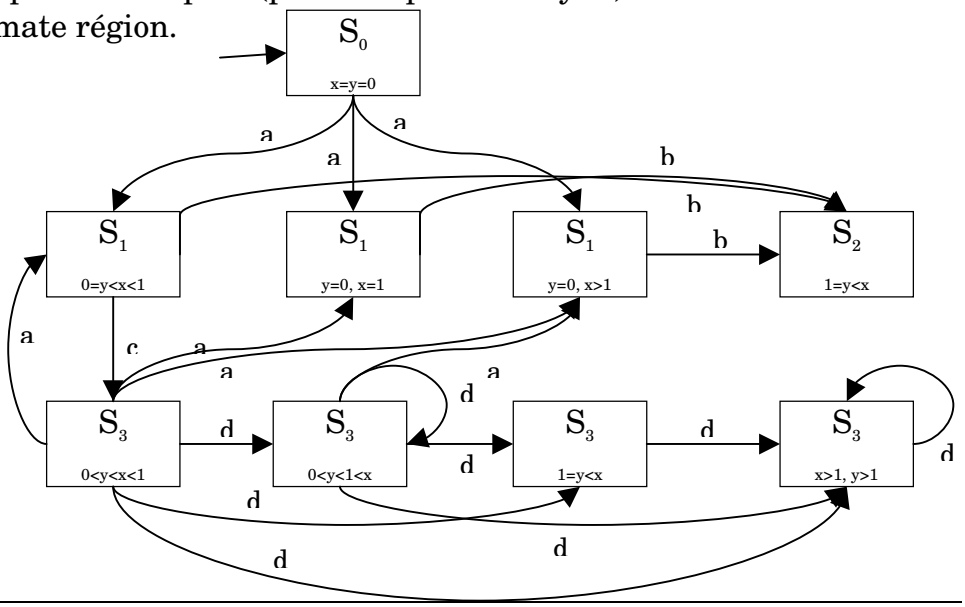


Fig. automate Région.

Pour qu'un chemin soit admissible il faut qu'il corresponde à un mot infini, or si on veut respecter la croissance du temps et des horloges, il faut que les horloges soient régulièrement remises à zéro ou vérifie une condition du type($x>c_x$). ce qui se traduit plus rigoureusement par la définition suivante :

Définition 6.15 (chemin progressif) Un chemin est progressif ssi pour toutes les horloges $x \in X$ il y a une infinité de $i \geq 0$ tels que α_i satisfait [(x :=0)$\cup$($x>c_x$)].

Sur l'exemple seul la boucle en (s_3, x>1,y>1) vérifie cette propriété.

4 Composition d'automates temporisés.

On peut composer des automates temporisés si leur ensembles d'horloges sont disjoints. On utilise un produit synchrone usuel, les transitions sont définies en prenant les intersections des contraintes temporelles et les unions des ensembles d'horloges à remettre à zéro, plus rigoureusement on a :

$$\begin{cases} < s_1, t_1, a, \lambda_1, \delta_1 > \\ < s_2, t_2, a, \lambda_2, \delta_2 > \end{cases} \Rightarrow < (s_1, s_2), (t_1, t_2), a, \lambda_1 \cup \lambda_2, \delta_1 \cap \delta_1 > .$$

5 Commande d'automates temporisés.

Le but premier de ce modèle est de faire de la vérification de système temps réel. C'est à dire de vérifier si le système défini par un automate temporisé et donc un langage respecte une contrainte, généralement d'ordre temporel. Cependant des méthodes de commandes existent en utilisant ce modèle.

Une première méthode consiste à utiliser la méthode de Brandin et Wonham. Pour cela on partitionne l'ensemble des événements en des sous-ensemble contrôlable et non contrôlable, on introduit le concept d'événements interdits, d'événements forçables. On modélise le temps par deux événements *tack* et *tock* suivant que le temps puisse être préempté par un événement forçable ou non. Enfin on synthétise un superviseur par intersection avec une spécification. Cette méthode, qu'on ne détaillera pas ici car très proche de celle exposée dans le chapitre précédent est présentée A. Gouin, L. Libeaut et J.L. Ferrier dans [12].
Cette approche permet d'allier la complexité du modèle d'Alur et Dill et l'approche classique de commandabilité des SED.

On peut aussi appliquer une version modifiée de l'algorithme de Kumar. La démarche est la même que pour la supervision des automates de Brandin et Wonham.

- On compose les automates du système et de la spécification ce qui permet d'obtenir un automate excluant tous les états hors spécification.
- Puis on définit les familles X_f et X_{sf} qui sont respectivement l'ensemble des états défendus et celui des états faiblement défendus.
- On efface de l'automate tous les états appartenant à X_f ou X_{sf}, ainsi que tous les arcs y menant et en sortant et les états qui deviennent inaccessibles.

Pour éviter d'être trop restrictif, A. Gouin et J.L. Ferrier proposent dans [11] une modification de l'algorithme de Kumar, qui permet de réduire les familles X_f et X_{sf}.
Cette méthode est motivée par le fait que certains états sont défendus ou faiblement défendus car ils font sortir le système de la spécification, à l'occurrence d'une séquence d'événements incontrôlables, mais qu'ils permettent parfois d'autres trajectoires qui respectent la spécification. Il faut donc tenter d'obliger le système supervisé à emprunter ces trajectoires.

On note σ l'événement qui, survenant en q, fait sortir de la spécification, et σ' l'événement qui permet d'y rester.

Pour pouvoir supprimer $q \in X_f$ de X_f, il faut réunir deux conditions :
- L'automate supervisé doit autoriser, en q, une sortie qui ne fait pas sortir de la spécification.

- Il faut que l'événement qui fait sortir de la spécification soit associé à une contrainte de l'un des types suivants :
    - Si σ est associé à une contrainte de la forme (c_i>a) ou(c_i≥a) ou (c_i=a) où c_i est une horloge, et σ' est forçable. Dans ce cas on ajoute la contrainte (c_i<a) à σ' et on le force avant que σ ne puisse survenir. Sinon on peut ajouter une contrainte en amont de q pour que la condition rajoutée soit compatible avec celle existant déjà.

    - Si σ est associé à une contrainte de la forme (c_i<a), ou (c_i=a) ou (c_i≥a) on peut associer une contrainte de la forme (c_i>a) à la transition qui permet d'atteindre q, néanmoins il faut que cette nouvelle contrainte soit compatible avec σ'.

Sinon on peut appliquer cette méthode aux transitions amont mais cela augmente souvent le nombre de transitions à modifier et il faut veiller à ce que les contraintes rajoutées soient compatibles avec un plus grand nombre de contraintes déjà existantes

Attention…
Si q a plusieurs transitions en entrée, il faut toutes les traiter successivement.

6 Exemple du réseau ferroviaire.

On modélise le réseau ferroviaire en introduisant quatre horloges x, y, u et v. L'horloge x donne la durée écoulée depuis la dernière entrée du train 1 en A, y le temps écoulé depuis sa dernière entrée en C, u indique le temps écoulé depuis la dernière entrée du train 2 en C, enfin v donne le temps écoulé depuis sa dernière entrée en C. Le modèle impose que les trains restent au moins 1 (ou 2) unité de temps sur les tronçons. Cela se traduit par une contrainte temporelle de la forme (c_i>1). On a donc :

$$X = \{x,y,u,v\},\ c_x=2,\ c_y=1,\ c_u=1,\ c_v=1.$$

On a donc les automates suivant pour modéliser le réseau et la spécification.

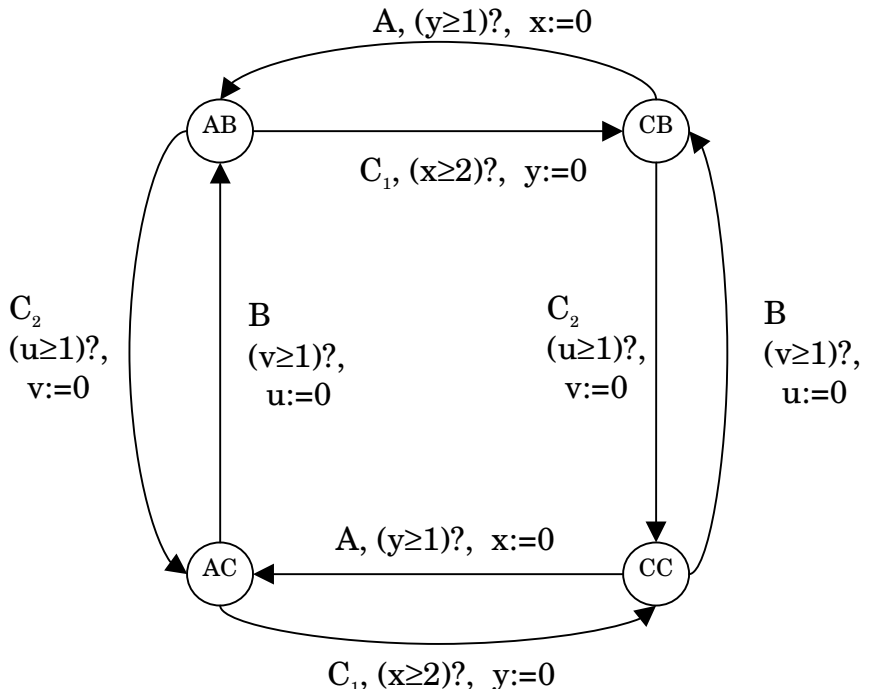


Fig. Automate du réseau.

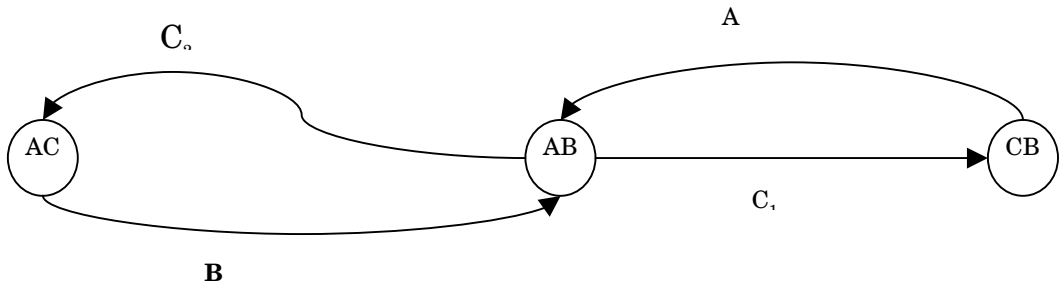


Fig. Automate de la spécification.

Pour avoir le système supervisé on fait le produit des deux automates, on obtient alors le même automate que celui du réseau auquel on enlève la localité (CC).

Si on se place dans le cas $\Sigma_n=\{c_1,c_2\}$, selon l'algorithme de Kumar les états (AC) et (CB) deviennent défendus et (AB) devient faiblement défendu, on essaie donc de réduire les ensembles X_f et X_{sf} . Pour cela on applique l'algorithme de Kumar modifié.

Aucune contrainte n'est de la forme (c_i<n) on ne peut donc pas se placer dans le deuxième cas. On essaie donc d'utiliser la première méthode. Pour cela il faut : $\Sigma_{for}=\{a,b\}$. Etudions les différents cas possibles. On suppose que la spécification est respectée, autrement dit le système n'atteint jamais l'état (CC).
On cherche dans un premier temps à sécuriser le passage en (CB), autrement dit on veut que a se produise avant c_2. Quand on atteint l'état (CB) cela signifie que la séquence d'événements qui vient de se produire est soit (a,c_1) soit (b,c_1). Dans le premier cas, l'automate à est resté au minimum une unité de temps en (CB) puis au minimum deux unités de temps en (AB), en tenant compte des contraintes tempporelles, cela implique :

$$x>2,\ u>3,\ y=0 \text{ en (CB)}$$

autrement dit il faut au moins attendre une unité de temps avant que a soit permis et pendant cette unité de temps c_2 peut se produire. Dans le deuxième cas, le système est resté au moins une unité de temps en (AC) puis en (AB) donc on a :

$$x>2,\ u>1,\ y=0 \text{ en (CB),}$$

ici encore c_2 peut se produire pendant une unité de temps et faire sortir le système de la spécification. De plus on ne peut pas imposer de contrainte sur u. En effet dans le premier cas l'automate doit séjourner une unité de temps en (AB) donc à moins de remettre u à zéro en même temps que y, on a $u>y$, et si on annule les deux horloges en même temps cela ne change rien au problème. Dans le second cas u est au mieux égal à y, mais jamais inférieur, il n'y a donc pas non plus de solution.
En définitive le système n'est pas supervisable si c_2 est incontrôlable.

On cherche maintenant à sécuriser le passage en (AC). On suppose maintenant l'automate en (AC). De manière équivalente au cas précédent on sait que la séquence qui vient de se produire est soit (b,c_2) soit (a,c_2). Dans le premier cas, quand le système parvient à l'état (AC) on a :

$$v=0,\ u>1,\ x>2$$

pendant une unité de temps le système peut diverger. Dans le second cas x a été remis zéro lors du franchissement de la transition qui mène en (AB), on peut donc ajouter la contrainte (x<1) à la transition menant de (AB) en (AC) auquel cas on a :

$$v=0,\ x<1,\ u>1 \text{ en (AC)}$$

autrement dit si b est forçable on dispose d'une unité de temps pour forcer b tandis que c_1 n'est pas permis. De plus cette contrainte interdit la répétition de la séquence (c_2,b) si une séquence (c_1,a) ne vient pas s'intercaler. Le fonctionnement du système est réduit mais le passage en (AC) est sécurisé.

On a alors l'automate suivant, reconnaissant le langage L donné par :

$$L=(c_1a+\varepsilon)(c_2bc_1a)^{\omega}(c_2b+\varepsilon)$$

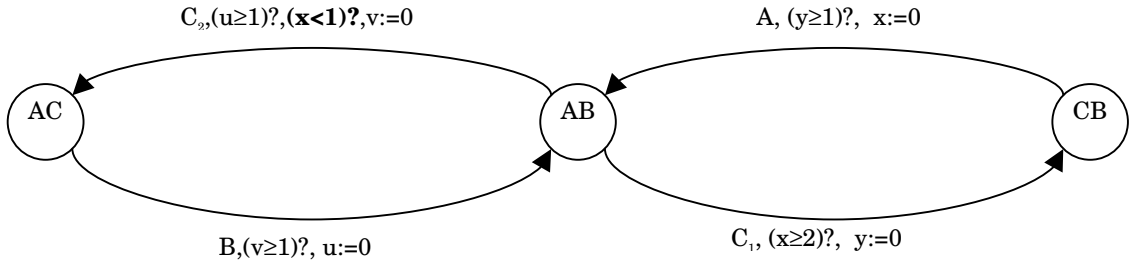


Fig. Automate supervisé.

Ainsi la supervision du réseau ferroviaire nécessite au moins que l'on dispose des moyens d'action suivants :

c_2 contrôlable,
b forçable.

Cette méthode de réduction des états interdits n'est pas applicable au modèle de Brandin et Wonham car on ne fait évoluer les compteurs des événements que s'ils sont permis, on n'a donc pas assez d'information pour pouvoir prévoir l'évolution du système et être plus permissif.

Conclusion.

On a présenté ici l'automate temporisé à temps dense le plus connu mais il en existe évidemment d'autres. On peut en particulier citer le modèle utilisé par V. Rusu dans [18], pour faire de la vérification temporelle. L'automate comporte aussi différentes horloges en relation avec une horloge générale mais en plus elle peuvent évoluer à des vitesses différentes. De plus les variables qui apparaissent dans les transitions continuent d'évoluer à l'intérieur des états. Dans cette catégorie la plupart sont issus des recherches en temps réel et n'ont pas pour objectif premier la commande mais la vérification (on cherche à vérifier une inclusion de langage qui montre que le système vérifie bien une contrainte, donnée par un langage). Du fait de leur taille et de leur complexité, l'outil langage est préféré à l'automate, plus visuel mais moins aisé à manipuler informatiquement.

Conclusion

Ce rapport a pour objet une présentation rapide des différentes techniques d'étude des SED, mais ne prétend pas à l'exhaustivité. On a pu voir que des approches très différentes sont possibles. Chaque approche a été développée dans un but précis : commande de système, évaluation de performances, vérification (temporelle)… cependant, deux axes principaux se dégagent, le premier est la modélisation par réseaux de Pétri et le deuxième utilise les automates.

Les automates sont de par leur conception plus adaptés au problème de commande. Différents algorithmes de supervision existent. De plus l'utilisation d'horloges de type temps réel, permet de prendre en compte les phénomènes de synchronisation d'actions qui ne sont pas facilement mis en œuvre en utilisant les automates de Ramadge et Wonham ou de Brandin et Wonham (B&W). Ces horloges permettent une connaissance plus complète de l'état de l'automate. Cette connaissance autorise une supervision moins restrictive.
On peut remarquer que sur l'exemple du réseau ferroviaire on peut aussi forcer b en gardant c_1 incontrôlable. Néanmoins cette modification si elle est possible ici, n'a pas valeur générale. En effet avec l'automate de B&W on ne peut pas toujours imposer de conditions sur les transitions amont, car si un événement n'est plus permis son compteur est remis à sa valeur par défaut. De plus l'automate de B&W ne permet pas les combinaisons de conditions sur les transitions.

Dans le cas des Réseaux de Petri la structure du modèle inclut les spécifications de fonctionnement. De ce fait l'évaluation de performances paraît être une approche plus naturelle. On a vu par exemple que dans le cas des Réseaux de Petri temporisés on peut caractériser les trajectoires d'un système et borner son taux de production. Cependant, s'il est plus ardu, le contrôle des SED sous forme Réseaux de Petri est une des directions actuelles de recherche, on peut en particulier citer la thèse de L. Libeaut [13] et les travaux effectués au LISA[2], regroupé sous le thème : « Contribution à la commande des Systèmes à Evénements Discrets », en particulier la thèse de B. Cottenceau [6]. Le cas particulier des graphes d'événements temporisés fait aussi l'objet de nombreux travaux très aboutis, il faut cependant noter que sa structure lui interdit la représentation des systèmes permettant des divergences en « ou ».
Il serait sans doute intéressant d'étudier plus en détail une éventuelle relation entre l'automate des trajectoires du RdPT et les automates de B&W (a priori l'échantillonnage fait qu'on se rapproche plus de B&W que de d'A&D). Il semblerait apparaître une relation de type projective de l'automate de B&W vers celui des trajectoires du réseau de Petri…

[2] LISA : Laboratoire d'Ingénierie des Systèmes Automatisés, d'Angers

Bibliographie

[1] R. Alur et D.L. Dill, A Theory of Timed Automata, Theoretical Computer Science, Vol 126, pp. 183-235, 1994.

[2] B.A. Brandin et W.M. Wonham, Supervisory Control of Timed Discrete Event Systems, IEEE, Vol 39, n° 2, pp. 329-341, Feb. 94.

[3] B.A. Brandin et W.M. Wonham, Modular Supervisory of Timed Discrete Event Systems, IEEE, Proceeding of the 32nd Conference on Decision and Control, San Antonio, Texas, Dec. 1993

[4] C.G. Cassandras, Discrete Event Systems – Modeling and Performance Analysis, Aksen Associates Incorporated Publishers, pp. 141-165, 1993.

[5] G. Cohen, P. Moller, J.P. Quadrat et M. Viot, Algebraic Tools for the Performance Evaluation of Discrete Event Systems, IEEE Proceedings, Vol. 77, pp. 39-58, 1989.

[6] B. Cottenceau, Contribution à la commande de systèmes à événements discrets, synthèse de correcteurs pour les Graphes d'Evénements Temporisés dans les dioïdes, Thèse de doctorat, LISA, université d'Angers, 1999.

[7] R. David et H. Alla, Du Grafcet aux Réseaux de Petri, Hermès, France, 1989.

[8] J.L. Ferrier, Systèmes Dynamiques à Evénements Dicrets : du modèle à la commande, notes de cours DEA AIA,1999.

[9] S. Gaubert, 1999, Introduction aux Systèmes Dynamiques à Evénements Discrets, Notes de cours, DEA ATS Orsay.

[10] A. Gouin et J.L. Ferrier, La prise en compte du temps dans les modèles automates : différentes approches, , ATG 1997(Angers, France, 9-11 Déc.1997).

[11] A. Gouin et J.L. Ferrier, Modeling and Supervisory Control of Timed Automata, JESA 99 (numéro spécial du JESA consacré aux systèmes réactifs : vol. 33, n°8-9, nov.1999)

[12] A. Gouin, L. Libeaut et J.L. Ferrier, Supervisory Control of Timed Automata, ECC 99 (Karlsruhe, Allemagne, 31 Août-3 Sept. 1999).

[13] L. Libeaut, Sur l'utilisation des dioïdes pour la commande des systèmes à événements discrets, Thèse de doctorat, Ecole centrale de Nantes, 1996.

[14] L. Libeaut et J.J. Loiseau, An Algebraic Model for Timed Petri Nets, 1999.

[15] L. Libeaut et J.J. Loiseau, Commande des SED-Théorie des langages commandables, 26ème Ecole de printemps d'informatique théorique Algèbres Max-Plus et applications en informatique et automatique, 1998.

[16] J.M. Prou et E. Wagneur, Controllability in the Max-Algebra, Kybernetika, Vol. 35, numéro 1, pp . 13-24, 1999.

[17] P.J.G. Ramadge et W.M. Wonham, The control of Discrete Event Systems, IEEE, Vol 77, n° 1, pp. 91-98, Jan. 1989.

[18] V. Rusu, Vérification Temporelle de programmes ELECTRE, Thèse de doctorat, Ecole Centrale de Nantes, 1996.

ANNEXE

Annexe

Programme de temporisation de SED.

Pour vérifier certains automates temporisé j'ai programmé sous Matlab l'algorithme d'évolution des SEDT de Brandin et Wonham. L'utilisateur doit entrer deux matrices. La première *G* représente le système sous forme matricielle, i.e. à un arc faisant passer le système de l'état *j* à l'état *i* à l'occurrence de l'événement *a* correspond le terme $G_{i,j}$ égal au numéro de *a*. En effet les événements sont repérés par des numéros. La deuxième matrice, Q, renseigne sur la nature et les bornes temporelles des événements. $Qi,_1$ est nul (resp. égal à 1) si l'événement *i* est lointain (resp. futur). $Q_{i,2}$ et $Q_{i,3}$ donnent respectivement les bornes inférieures et supérieures d'occurrence de l'événement *i*. Enfin l'utilisateur doit préciser quel est l'état initial, le programme ne temporisant que des automates déterministes. Le programme donne alors l'automate temporisé G_res et une matrice qui donne en ligne *i* les renseignement suivants : [état de G_res, état de G correspondant, [valeurs des compteurs]].

Syntaxe :

```
[G_res,E] = SEDT(G,Q,ini)
```

On trouvera d'abord les programmes, puis des exemples correspondants au modèle du réseau ferroviaire.

1 Programmes.

Programme principal : SEDT.m.

```
function [G_res,E] = SEDT(G,Q,ini)

% [G_res,E] = SEDT(G,Q,ini)
%
% SEDT donne le graphe temporisé d'un système sous forme
matricielle, ainsi que le détail des états
% temporisés du graphe temporisé
%
% entrées : - G : fonction de transition sous forme
matricielle, matrice n*n
%                   G(i,j) donne le numéro de l'événement qui
correspond à l'arc j->i
%           - Q : Q(i,1)=  0 si sigma(i) est lointain,
%                          1 si sigma(i) est futur.
%                    Q(i,2) est la borne inf d'occurence de
sigma(i).
```

```
%                     Q(i,3) est la borne sup d'occurence de
sigma(i), sans importance si lointain.
%           - ini : numero de l'état initial du système;
%
% sorties : - G_res : donne le graphe temporisé sous forme
matricielle
%           - E : E(i,1) numéro de l'état temporisé;
%                 E(i,2) numéro de l'état non temporisé
correspondant;
%               E(i,j+2) valeur du compteur de sigma(j).

s_G = size(G);
s_Q = size(Q);
n = s_G(1);
m = s_Q(1);
E = [];
G_res = [0];
fait = [];
nouveau = 1;

%on vérifie que le système est déterministe
[deter] = analyse(G);
if deter == 0
   return
end

% initialisation
s_G_res = size(G_res);
[e_init,erreur] = init(G,Q,ini);
E = e_init;
e_cour = e_init;
% arret si erreur détectée
if erreur ~= 0
   return
end



% boucle tant que nbligne(G) < nbcol_G
while (s_G_res(1) > s_G_res(2)) | (s_G_res(1) == 0) | (nouveau
== 1)

   %1. on détermine les evts permis et admissibles en e_cour
   [fait,permis,adm, topadm] = adm_permis(fait,G,e_cour,Q);

   %2. pour top+chaque admissible
   %2.1. après un top
   if topadm == 1
      %2.1.1. calcul de l'état temporisé
      [e_nouv] = posttop(e_cour,E,permis,Q);
      evt = inf;
```

```
      %2.1.2. modification de G_res et E
      [E,G_res] = modifres(E,G_res,e_nouv,e_cour,evt);
   end
   nbadm = length(adm);
   %2.2. après un evt
   for i = 1:nbadm
      evt = adm(i);
      %2.2.1. calcul de l'état temporisé
      [e_nouv] = posteve(e_cour,Q,G,evt,E);
      %2.2.2. modification de G_res et E
      [E,G_res] = modifres(E,G_res,e_nouv,e_cour,evt);
   end

   %test fin boucle
   s_E = size(E);
   s_G_res = size(G_res);
   nbligne = 0;
   nouveau = 0;
   while (nbligne < s_E(1)) & (nouveau == 0)
      nbligne = nbligne + 1;
      if (ismember(E(nbligne,1),fait) == 0)
         nouveau = 1;
      end
   end
   if nouveau == 1
      e_cour = E(nbligne,:);
   end
end
```

Sous-programmes :

- `function [deter] = analyse(G)`

```
% [deter] = analyse(G)
%
% permet de savoir si l'automate est déterministe
%
% entrée : - G : graphe du système sous forme matricielle;
%
% sortie : - deter : booléen égal à 1 si l'automate est
déterministe

s_G = size(G);
nbcol = 0;
bloq = [];
deter = 1;

while (deter == 1)& (nbcol < s_G(2))
   nbcol = nbcol + 1;
   a = G(:,nbcol);
```

```
   a = a';
   abis = [];
   for i = 1:length(a)
      if a(i) ~= 0
         abis = [abis, a(i)];
      end
   end
   if length(abis) == 0
      bloq = [bloq,nbcol];
   end
   ater = unique(abis);
   if length(ater)~= length(abis)
      message = 'le système est non détrerministe.'
      deter = 0;
   end
end
```

- `function [e_init,erreur] = init(G,Q,ini)`

```
%
%[e_init,erreur] = init(G,Q,ini)
%
%init réalise l'initialisation en calculant le premier état de
l'automate temporisé
%
%  entrées  :  -  G  :  fonction  de  transition  sous  forme
matricielle;
%           - Q : Q(i,1) = 0 si sigma(i) est lointain,
%                          1 si sigma(i) est futur.
%                     Q(i,2) est la borne inf d'occurence de
sigma(i).
%                     Q(i,3) est la borne sup d'occurence de
sigma(i), sans importance si lointain.
%           - ini : numero de l'état initial du système;
%
% sorties : - e_init : état initial du graphe temporisé
%                     e_init(1) donne l'état du système
%                      e_init(i+1) donne la valeur du compteur
de sigma(i)
%               - erreur : différent de 0 si une erreur est
détectée
%


s_G = size(G);
s_Q = size(Q);
n = s_G(1);
m = s_Q(1);
erreur = 0;
```

```
if (ini < 1) | (ini > s_G(1))
   erreur = 1;
   'L''état initial spécifié n''existe pas.'
end

e_init(1) = 1;
e_init(2) = ini;
for i = 1:m
   if Q(i,1) == 0
      e_init(i+2) = Q(i,2);
   elseif Q(i,1) == 1
      e_init(i+2) = Q(i,3);
   else
      erreur = 1;
      'La matrice Q n''est pas valide.'
   end
end
```

- `function [fait,permis,adm, topadm] = adm_permis(fait,G,e_cour,Q)`

```
% [fait,permis,adm, topadm] = adm_permis(fait,G,e_cour,Q)
%
% adm_permis détermine les événements admissibles et ceux
permis pour un état donné
%
% entrées : - fait : vecteur daant les composantes donnent les
états pris en compte;
%                 - G : fonction de transition sous forme
matricielle
%           - e_cour : état du système temporisé.
%           - Q : Q(i,1)= 0 si sigma(i) est lointain,
%                         1 si sigma(i) est futur.
%                     Q(i,2) est la borne inf d'occurence de
sigma(i).
%                     Q(i,3) est la borne sup d'occurence de
sigma(i), sans importance si lointain.
%
% sorties : - fait : vecteur daant les composantes donnent les
états pris en compte;
%               - permis : vecteur comportant les événements
permis
%           - adm : vecteur des événements admissibles
%           - topadm : booléen à 1 si top est admis


fait = [fait, e_cour(1)];

% e_cour -> permis
q = e_cour(2);
```

```
permis = setdiff(G(:,q),[0]);
nbpermis = length(permis);

% permis -> admissibles
adm = [];
topadm = 1;

for i = 1:nbpermis
   evt = permis(i);
   % si evt futur
   if Q(evt,1)== 1
      if    (e_cour(evt+2)    <=    (Q(evt,3)-Q(evt,2)))    &
(e_cour(evt+2) > 0)
         adm = [adm, evt];
      elseif e_cour(evt+2) == 0
         topadm = 0;
         adm = [adm, evt];
      end
      % si evt lointain
   elseif Q(evt,1) == 0
      if (e_cour(evt+2) == 0)
         adm = [adm, evt];
      end
   end
end
```

- `function [e_nouv] = posttop(e_cour,E,permis,Q)`

```
% [e_nouv] = posttop(e_cour,E,permis,Q)
%
% posttop calcule le nouvel état temporisé après la survenue
d'un top
%
% entrées : - e_cour : état temporisé précédent;
%             - E : matrice dont chaque ligne correspond à un
état du graphe temporisé,
%               - permis : vecteur comportant les événements
permis;
%           - Q : Q(i,1)= 0 si sigma(i) est lointain,
%                         1 si sigma(i) est futur.
%                     Q(i,2) est la borne inf d'occurence de
sigma(i).
%                     Q(i,3) est la borne sup d'occurence de
sigma(i), sans importance si lointain.
%
% sortie : - e_nouv : nouvel état temporisé.
%

e_nouv = [];
```

```
s_E = size(E);
nb_permis = length(permis);
m = length(e_cour)-2;

e_nouv(1) = s_E(1)+1;
e_nouv(2) = e_cour(2);
for i = 1:m
   if ismember(i,permis) == 1
      e_nouv(i+2) = max(e_cour(i+2)-1,0);
   else
      if Q(i,1) == 1
         e_nouv(i+2) = Q(i,3);
      elseif Q(i,1) == 0
         e_nouv(i+2) = Q(i,2);
      end
   end
end
```

- `function [e_nouv] = posteve(e_cour,Q,G,evt,E)`

```
% [e_nouv] = posteve(e_cour,Q,G,evt,E)
%
% posteve calcule l'état temporisé qui suit la survenue de
sigma en l'état e_cour
%
% entrées : - e_cour : état temporisé courant;
%                  - G : automate non temporisé sous forme
matricielle;
%               - Q : matrice des événements et de leur délai
associé;
%           - evt : événement survenu;
%           - E : matrice des états temporisés
%
% sortie : - e_nouv : nouvel état temporisé.

q = e_cour(2);
s_E = size(E);
s_G = size(G);
s_Q = size(Q);
trouve = 0;
nbligne = 0;

while (trouve == 0) & (nbligne < s_G(1))
   nbligne = nbligne + 1;
   if G(nbligne,q) == evt
      trouve = 1;
   end
end
```

```
e_nouv(1) = s_E(1)+1;
e_nouv(2) = nbligne;
for i = 1:s_Q(1)
   if i == evt
      if Q(i,1) == 0
         e_nouv(i+2) = Q(i,2);
      elseif Q(i,1) == 1
         e_nouv(i+2) = Q(i,3);
      end
   else
      if ismember(i,G(:,e_nouv(2))') == 1
         e_nouv(i+2) = e_cour(i+2);
      else
         if Q(i,1) == 0
            e_nouv(i+2) = Q(i,2);
         elseif Q(i,1) == 1
            e_nouv(i+2) = Q(i,3);
         end
      end
   end
end
```

- `function [E,G_res] = modifres(E,G_res,e_nouv,e_cour,evt)`

```
% [E,G_res] = modifres(E,G_res,e_nouv,e_cour,evt)
%
% modifres modifie E et G_res en fonction de e_nouv.
%
% entrées : - E : matrice dont chaque ligne correspond à un
état du graphe temporisé,
%              - G_res : donne le graphe temporisé sous forme
matricielle;
%           - e_nouv : nouvel état temporisé;
%           - e_cour : ancien état temporisé;
%           - evt : événement survenu;
%
% sorties : - G_res : donne le graphe temporisé sous forme
matricielle;
%             - E : matrice dont chaque ligne correspond à un
état du graphe temporisé,
%              soit l'état du système et les valeurs des
compteurs associés aux événements.
%

s_E = size(E);
nbligne = 0;
trouve = 0;

while (nbligne < s_E(1)) & (trouve == 0)
   nbligne = nbligne + 1;
```

```
   if E(nbligne,2:end) == e_nouv(2:end)
      trouve = 1;
   end
end

% si nouveau
if trouve == 0
   E = [E;e_nouv];
   G_res(e_nouv(1),e_cour(1)) = evt;
% si existe déjà
else
   G_res(nbligne,e_cour(1)) = evt;
end
```

2 Exemples.

On modélise les fonctionnements séparés des trains 1 et 2 par les automates non temporisés présentés au paragraphe 6 du chapitre 5

Le fonctionnement du train1.
L'événement 1 correspond au passage du train de C à A, l'événement 2 au passage de A en C, l'état 1 (reps. 2) signifie que le train est en A (resp. C), on exécute alors :
» G=[0 1;2 0];Q=[0 1 Inf; 0 2 Inf];ini=1;
» [G_res,E] = SEDT(G,Q,ini)

G_res =

0	0	0	0	1
Inf	0	0	0	0
0	Inf	Inf	0	0
0	0	2	0	0
0	0	0	Inf	Inf

E =

1	1	1	2
2	1	1	1
3	1	1	0
4	2	1	2
5	2	0	2

Le fonctionnement du train 2.
L'événement 1 correspond au passage du train de C à B, l'événement 2 au passage de B en C, l'état 1 (reps. 2) signifie que le train est en B (resp. C), on exécute alors :

```
»G=[0 1;2 0];Q=[0 1 Inf; 0 1 Inf];ini=1;
»  [G_res,E] = SEDT(G,Q,ini)

G_res =

   0    0    0    1
 Inf  Inf    0    0
   0    2    0    0
   0    0  Inf  Inf


E =

   1    1    1    1
   2    1    1    0
   3    2    1    1
   4    2    0    1
```

Le fonctionnement du réseau.
Si on veut temporiser le réseau, on a vu qu'il était possible de temporiser après la composition car aucun événement n'est commun aux deux trains, on décrit le réseau par :
Etat 1 : le train 1 est en A, le train 2 est en B,
Etat 2 : le train 1 est en C, le train 2 est en B,
Etat 3 : le train 1 est en A, le train 2 est en C,
Etat 4 : le train 1 est en C, le train 2 est en C,
événement1 : le train1 passe de C à A,
événement2 : le train2 passe de C à B,
événement3 : le train1 passe de A à C,
événement4 : le train2 passe de B à C.

On exécute donc :

```
»[G_res,E] = SEDT([0 1 2 0;3 0 0 2;4 0 0 1; 0 4 3 0],[0 1 Inf;0 2 Inf;0 1 Inf;0 1 Inf],1)

»G_res =
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   2   1   0
 Inf Inf   0   0   0   0   0 Inf   0   0   0   0   0   0   0 Inf   0   0   0   0
   0   3   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   4   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0 Inf   0 Inf   0   0   0   0   0   0   0   0   0   0   0   0   0 Inf Inf
   0   0   4   3   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0 Inf   0   0   0   0   0   0   0   0 Inf   0   0   0   0   0   0   0
   0   0   0   0   1   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

```
    0     0     0     0     4     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0   Inf     0     0   Inf     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0   Inf     0     0     0   Inf     0     0     0   Inf     0     0   Inf     0     0
    0     0     0     0     0     0     3     0     0     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     4     1     0     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0   Inf     0   Inf     0   Inf     0     0   Inf     0     0     0
    0     0     0     0     0     0     0     0     0     1     0     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     2     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     3     0     0     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0     0     1     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0     0     2     0     0     0     0     0     0
    0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     3     2     0     0     0
```

»E =

1	1	1	2	1	1	(A_0B_0)
2	1	1	2	0	0	(A_1B_1)
3	2	1	2	1	0	(A_2B_1)
4	3	1	2	0	1	(A_1C_0)
5	2	0	2	1	0	(C_0B_1)
6	4	1	2	1	1	(A_2C_0)
7	3	1	1	0	1	(A_2C_1)
8	1	1	2	1	0	(B_1B_1)
9	4	0	2	1	1	(C_0C_0)
10	4	0	1	1	1	(A_2B_0)
11	3	1	0	0	1	(C_0C_1)
12	4	1	1	1	1	(A_0B_1)
13	3	1	2	1	1	(C_1C_0)
14	4	0	0	1	1	(C_1C_1)
15	3	1	1	1	1	(C_0B_0)
16	1	1	2	0	1	(A_0C_0)
17	4	1	0	1	1	(A_0C_1)
18	3	1	0	1	1	(C_1B_0)
19	2	0	2	1	1	(A_1C_1)
20	2	1	2	1	1	(A_1B_0)

Le lecteur vertueux pourra vérifier que ces résultats correspondent à l'automate temporisé du réseau ferroviaire donné en fin de chapitre 5.