Université ABOU-BEKR BELKAÏD – Tlemcen
Faculté de génie électrique et électronique

Thèse

Pour obtenir le grade de docteur

Spécialité : Productique

Préparé au laboratoire de productique de Tlemcen

Par

Latéfa GHOMRI

Le 08 Mars 2012

Synthèse de contrôleur de systèmes hybrides à flux continu par réseaux de Petri hybrides

Devant le jury composé de :

Président
GHOUALI Noureddine
Professeur à l'université de Tlemcen

Examinateurs
GHOUL HADIBY Rachida
Professeur à l'université d'Oran

HENNET Jean-Claude
Directeur de recherche au CNRS

DEMONGODIN Isabelle
Professeur à l'université Saint Jérôme

Directeurs de thèse
ALLA Hassane
Professeur à l'université Grenoble 1

SARI Zaki
Professeur à l'université de Tlemcen

Sommaire

Introduction générale

La modélisation des systèmes dynamiques est depuis longtemps étudiée pour l'analyse et la commande. Ainsi il est possible d'étudier le comportement dynamique d'une variété considérable de procédés et systèmes physiques pour observer leur comportement puis le modifier par une commande. Chronologiquement, les systèmes continus ont été les premiers à être étudiés, leur modélisation s'effectue généralement au moyen d'équations différentielles. Ces systèmes traitent des grandeurs continues, comme la température, la pression, le flux, etc.

Le progrès de l'informatique et l'utilisation des ordinateurs pour la commande de systèmes ont créé des situations dans lesquelles la connaissance exacte des variations des grandeurs continues n'est pas intéressante et seules certaines valeurs seuils de ces variables continues présentent un intérêt. Ces systèmes sont les systèmes à événements discrets.

Pendant longtemps l'automatique a traité séparément les systèmes continus et les systèmes à événement discrets, pour chacune de ces deux classes de systèmes existe une théorie, des méthodes et des outils pour résoudre les problèmes qui se posent à elle. Cependant, la séparation entre le monde des systèmes continus et celui des systèmes à événements discrets, n'est pas aussi nette et la plupart des systèmes réels présentent à la fois des aspects continus et d'autres discrets. La prise de conscience de ce fait rapproche de plus en plus les deux communautés, ce qui a permis de favoriser les échanges entre les différents spécialistes, qu'ils soient issus du monde de l'automatique continue ou de celui de l'automatique événementielle. Les résultats de travaux dans ce domaine portent essentiellement sur les problèmes de modélisation, d'analyse et de commande des systèmes dynamiques hybrides. Au monde des automaticiens s'est associée la communauté informatique qui, en s'intéressant au problème de la vérification s'est approchée de la modélisation des systèmes à événements discrets puis à celle des systèmes hybrides.

Dans ce travail nous nous sommes intéressés à une classe particulière des systèmes dynamiques hybrides, à savoir la classe des systèmes à flux continus. Cette classe a pour particularité de comporter une dynamique continue linéaire et positive commandée par une dynamique discrète. Un système hybride est dit positif si ses variables d'état prennent des valeurs positives dans le temps. Et il est dit linéaire par morceaux si les lois décrivant son évolution continue sont formulées au moyen d'équations différentielles linéaires au sens des automates

hybrides, les variables continues ont une évolution affine en fonction du temps. Un effort particulier a été apporté à l'étude de cette classe pour deux raisons principales. D'abord, elle est suffisamment riche pour permettre une modélisation réaliste de nombreux problèmes. Ensuite, sa simplicité relative permet une conception facile d'outils et de modèles pour sa description et son analyse. Cette classe englobe plusieurs problèmes réalistes, comme les procédés batch, les systèmes manufacturiers traitant une quantité importante de produits, les systèmes de transport, de communication et autres

Pour intégrer les aspects continu et événementiel au sein d'un même formalisme, trois approches sont possibles. Cela dépend du modèle dominant, à partir duquel s'effectue l'extension. Ces approches sont l'approche continue, l'approche discrète et l'approche mixte.

La première approche consiste à intégrer l'aspect discret dans un formalisme continu, et ceci par l'introduction de variables booléenne ou entière dans un système d'équations.

L'approche discrète a pour principe d'intégrer l'aspect continu dans un formalisme pour les systèmes à événements discrets comme les réseaux de Petri ou les automates à états finis. Le champ d'application de ces derniers a été étendu pour prendre en compte les systèmes dynamiques hybrides. Pour les réseaux de Petri, le premier pas dans cette direction a été pris en 1987 par David et Alla [DA87], en introduisant les réseaux de Petri continus, puis les réseaux de Petri hybrides. Depuis, plusieurs extensions des réseaux de Petri ont été présentées pour permettre la modélisation de l'aspect hybride.

Le modèle réseaux de Petri hybride présente l'avantage d'être un modèle puissant pour la description des systèmes hybrides, dans le sens où le modèle est conçu de manière intuitive. De plus, ce modèle hérite de tous les avantages des réseaux de Petri. Des exemples de ces avantages sont : ce modèle n'exige pas une énumération exhaustive de l'espace d'état et peut représenter d'une manière finie un système dont l'espace d'états atteignable est infini. Il permet une modélisation modulaire, où la structure de chaque module est conservée dans le modèle composé. Et enfin, les réseaux de Petri permettent une représentation explicite du parallélisme, de la synchronisation et des conflits.

La troisième approche pour l'intégration des aspects continu et discret dans un même modèle est dite approche mixte. Cette approche repose sur l'utilisation d'un modèle à événements discrets comme moniteur de système d'équations. A un instant de temps donné, seul un des modèles est actif, soit il y a modification de la structure des équations formant le modèle continu et le temps n'évolue pas, soit le temps évolue mais la structure des équations ne change pas. Cette caractéristique se trouve dans le modèle automates hybrides, qui constitue le modèle le plus général pour la modélisation des systèmes dynamiques hybrides puisqu'ils peuvent modéliser la plus grande variété de dynamiques continues. Ce modèle permet une analyse formelle des systèmes dynamiques hybrides.

Dans cette thèse nous nous sommes intéressés à la synthèse de contrôleur des systèmes à flux continu. Un système à flux continu peut représenter soit un flux de matière soit un flot important de produits, dans ce dernier cas il constitue une approximation. Ce flux peut être

interrompu, ralenti ou accéléré par des événements donnant ainsi un modèle hybride. Notre objectif est de contrôler ces flux en leur imposant des contraintes. La synthèse de contrôleur pour les systèmes dynamiques (autonomes, temporisé ou hybride) passe en général par les trois étapes suivantes :

1. La représentation du comportement du système par un modèle ;
2. La définition des spécifications exigées sur le comportement de ce système ;
3. La synthèse du contrôleur qui restreint le comportement du système au comportement exigé, tout en lui assurant le comportement le plus libre possible.

Pour la première étape qui est une étape de modélisation, nous avons utilisé une variété des réseaux de Petri hybrides, dite réseaux de Petri hybrides D-élémentaires. Ce modèle est bien adapté pour représenter les influences réciproques des parties discrètes et continues. Il combine un réseau de Petri T-temporel et un réseau de Petri continu. Les intervalles de temps associés aux transitions du RdP T-temporel introduisent un indéterminisme quand au comportement du modèle hybride. Cet indéterminisme permet d'utiliser les transitions discrètes qui seront contrôlables, comme point de commande pour le contrôle des systèmes à flux continu.

Les algorithmes de synthèse de contrôleur utilisent traditionnellement les automates à cause de leur facilité de manipulation formelle. Cependant ce formalisme n'est pas le plus adapté pour la description du système. C'est pour cette raison que nous avons penser à combiner les avantages des réseaux de Petri hybrides D-élémentaires à ceux des automates hybrides en utilisant un algorithme permettant la traduction des réseaux de Petri hybrides D-élémentaire en automates hybrides. Cela permet d'associer la puissance de modélisation des réseaux de Petri à la puissance d'analyse des automates. C'est le modèle automate hybride qui sera utilisé pour la détermination formelle de contrôleur.

Dans un système dynamique hybride, les spécifications peuvent porter sur la partie discrète ou sur la partie continue. Nous ne considérons ici que les spécifications qui portent sur la partie continue, elle correspond à la dynamique à contrôler. Ces spécifications reviennent à limiter l'espace atteignable par les variables d'état continues à un espace d'état désiré. La synthèse de contrôleur consiste à déterminer les contrôles à appliquer pour restreindre ces espaces. Ceci est réalisé en modifiant les gardes des transitions contrôlables. Les gardes sont obtenues à partir des formules algébriques, permettant le calcul des dates limites de séjour dans les sommets et assurant le respect des spécifications.

Cette thèse est composée de quatre chapitres :

Dans le premier chapitre, nous introduisons les notions fondamentales relatives aux systèmes dynamiques hybrides. Leur définition ainsi que les principaux phénomènes hybrides, sont présentés. Nous passons en revue les principaux formalismes de modélisation des systèmes hybrides dont nous avons besoin, à savoir : Les réseaux de Petri hybrides D-élémentaires et les automates hybrides.

Dans le deuxième chapitre, nous introduisons les réseaux de Petri hybrides D-élémentaires. Notre principale motivation pour utiliser les RdP pour modéliser les systèmes dynamiques hybrides est de conserver tous les avantages des RdP. Un algorithme permettant la traduction des réseaux de Petri hybrides D-élémentaires en automates hybrides est présenté. Cet algorithme permet de combiner les avantages des deux modèles.

Dans le troisième chapitre nous présentons les principales contributions autour de la synthèse de contrôleur. Elle s'adresse aux domaines des systèmes à évènements discrets, des systèmes temporisés ou des systèmes dynamiques hybrides. Nous portons une attention particulière aux travaux de Ramadge et Wonham qui constituent les travaux fondateurs de la synthèse de contrôleur.

Dans le quatrième chapitre nous présentons notre approche de synthèse de contrôleur développée pour les systèmes à flux continus. Nous commençons par décrire cette approche au niveau d'un sommet. Puis nous présentons la manière dont est déterminé le contrôleur global

Chapitre 1

Systèmes Dynamiques Hybrides : Modélisation et Analyse

Dans ce premier chapitre nous présentons la classe de système que nous considérons dans ce travail. Il s'agit d'une sous classe de systèmes dynamiques hybrides que nous appellerons les systèmes a flux continu. Ainsi nous commençons par présenter de manière générale le comportement hybride des systèmes dynamiques, les spécificités des systèmes à flux continu et les outils que nous utiliserons pour la modélisation de ces systèmes. Nous illustrons tout cela par des exemples.

I. Comportement hybride des systèmes dynamiques

Les systèmes dynamiques peuvent être classifiés suivant la nature des variables d'état. Ainsi, les premiers systèmes dynamiques étudiés sont les systèmes continus où les variables d'état évoluent de manière continue dans le temps, telles que la position, la vitesse, la température, la pression, etc. (figure 1.1). L'outil classique de modélisation de la dynamique des systèmes continus correspond aux équations différentielles.

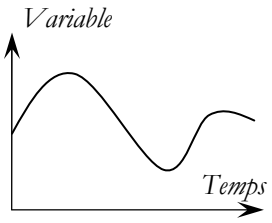


Figure 1.1. Comportement d'une variable d'état dans un système continu

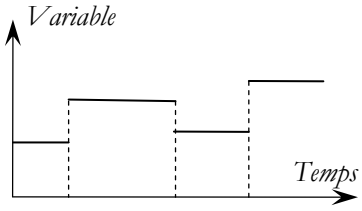


Figure 1.2. Comportement d'une variable dans un SED

Les systèmes à événements discrets (SED) sont des systèmes où l'état change de valeurs de manière discrète dans le temps (nombre de pièces dans un stock, état d'une machine, …). Le changement d'état est causé par l'occurrence d'un événement instantané, par exemple l'ouverture d'une vanne, l'occurrence d'une panne, ou la frappe d'une touche de clavier (figure 1.2). L'outil de modélisation de base des SED sont les automates à état finis.

La fusion de ces deux classes de systèmes a donné lieu à une nouvelle classe dite systèmes dynamiques hybrides (SDH). Les SDH sont donc des systèmes comportant des dynamiques continues et des dynamiques événementielles en interaction. Ils sont représentés par des variables d'état continues, qui prennent leurs valeurs dans un ensemble continu (ℝ) et d'autres discrètes, prenant leurs valeurs dans un ensemble d'événements discret (ℕ). Ces variables continues et discrètes dépendent de la variable indépendante *temps*, qui peut être à son tour considérée comme continue ou discrète.

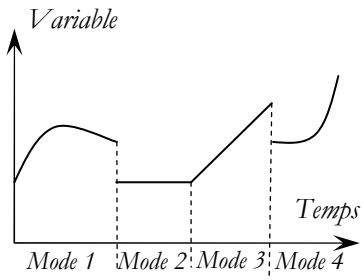


Figure 1.3. Comportement d'une variable dans un SDH

La théorie des SDH existant de nos jours est le fruit de recherche dans plusieurs disciplines ; elle a évolué comme la fusion de deux points de vue complémentaires. D'un côté, les systèmes hybrides sont considérés comme une extension des systèmes continus classiques, avec parfois des événements discrets provoquant un changement du comportement dynamique. D'un autre point de vue, les SDH sont considérés comme une extension des systèmes purement évènementiels, où chaque état du système est caractérisé par un comportement continu.

Ainsi le comportement d'un SDH est un ensemble de modes, chaque mode est caractérisé par la dynamique des variables d'état continues. Des événements instantanés provoquent le changement de modes. Ce changement peut préserver la continuité des variables d'état, où il peut causer des discontinuités. Les dates d'occurrence de ces événements peuvent considérablement influer sur le comportement du système. On rencontre dans de nombreux domaines des exemples de systèmes abstraits par des SDH, dans l'électronique, la robotique, la production manufacturière, le transport, la communication ou la biologie.

I.1. Classes de phénomènes hybrides

Dans un SDH, le système change de mode (ou de phase) suite à l'occurrence instantané d'un événement discret. Cette occurrence entérine une discontinuité dans l'état continu. Branicky, Borkar et Mitters [BBM94] ont identifié les principaux phénomènes physiques pouvant causer des discontinuités dans les SDH. Les auteurs distinguent deux types de discontinuités : des changements instantanés dans l'état, dits sauts, et des changements instantanés dans la dynamique (le champ de vecteur), dits commutations. Ces discontinuités peuvent être autonomes ou commandées.

- ***Les commutations autonomes***

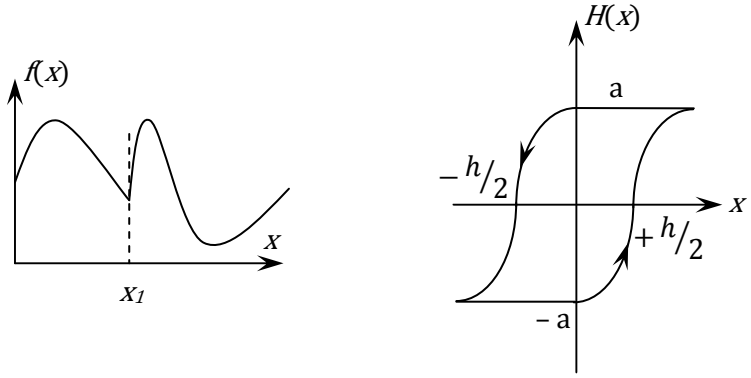
Ce phénomène est observé quand la fonction d'évolution change d'une façon discontinue lorsque l'une des variables d'état continu atteint certains seuils, comme schématisé sur la figure 1.4.a pour une seule variable d'état. C'est le cas des systèmes à hystérésis (figure 1.4.b).

- ***Les commutations contrôlées***

Dans ce cas, le champ de vecteur commute en réponse à une loi de commande. Un exemple d'un tel phénomène est donné par Zaytoon et Quenec'Hdu [ZQ2001] pour le modèle simplifié d'une transmission manuelle.

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = \dfrac{\left(-a\left(\dfrac{x_2}{v}\right) + u\right)}{1+v} \end{cases}$$

La variable x_1 représente la vitesse relative par rapport à un point fixe, x_2 la vitesse de rotation de l'engin, $u \in \{0, 1\}$ la position d'accélérateur, a un paramètre du système et $v \in \{1, 2, 3, 4\}$ la position du levier de vitesse.



-a- -b-

Figure 1.4. a. Commutation à la valeur seuil x_1. b. La fonction d'hystérésis.

- ***Les sauts autonomes***

Lorsque l'état atteint certaines zones prédéfinies de l'espace d'état, il effectue un saut (discontinuité ou impulsion) de sa valeur courante à une autre. Un exemple très connu de ce phénomène est le choc entre deux corps où la vitesse change brutalement et subit un saut. Considérons, par exemple, une table de billard de longueur *l* et de largeur *h*, avec une boule, comme l'illustre la Figure 1.5.b. La position initiale de la boule est (x_0, y_0) et après avoir été frappée elle commence à se déplacer avec une vitesse *V (Vx,Vy)*. Quand la boule arrive à un côté de la table parallèle à l'axe *y*, elle rebondit et le signe de la composante de la vitesse v_x change. De même, le signe de la composante de la vitesse v_y change lorsque la boule arrive à un côté parallèle à l'axe *x*. La figure 1.5.c représente la composante verticale *Vy* de la vitesse *v en* fonction de *y*.

- ***Les sauts commandés***

L'état ou une partie change instantanément sous l'influence d'une action extérieure. Considérons par exemple un modèle simple d'un stock où on dépose les quantités $\alpha_1, \alpha_2, \cdots$, de matière aux instants $t_1 < t_2 < \cdots$. L'évolution de la quantité de matière *y* est régie par l'équation suivante [ZQ2001] :

$$\dot{y} = -Q(t) + \sum_i \delta(t - t_i)\alpha_i$$

Avec Q la fonction d'utilisation de la matière et δ l'impulsion de Dirac.

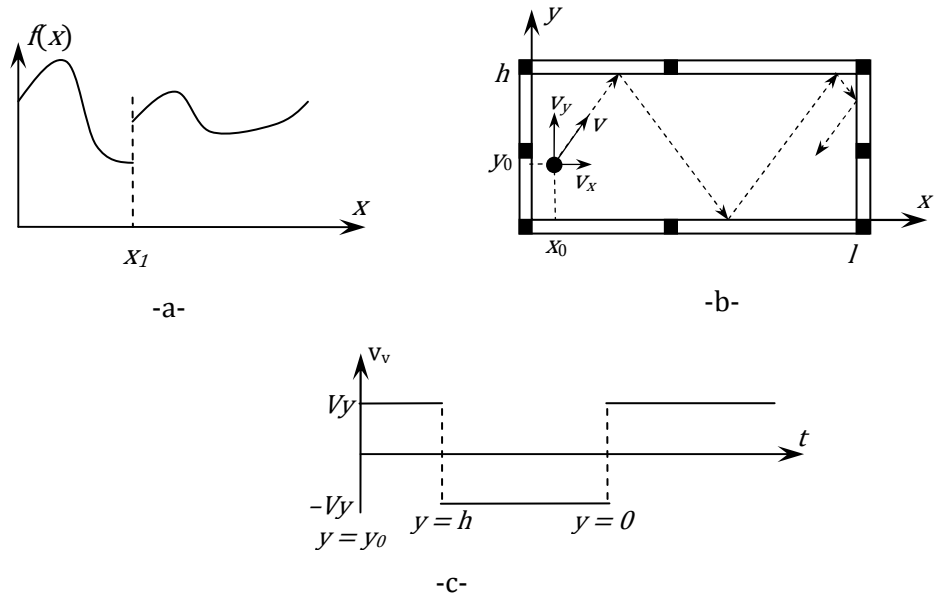


Figure 1.5 a. Saut à la valeur seuil x_1. b. La trajectoire d'une balle de billard. c. Evolution de vitesse v_Y de la balle de billard

I.2. La classe de systèmes considérée (*les systèmes à flux continu*)

Dans ce travail nous nous intéressons à une sous classe de SDH, que nous appelons, *systèmes à flux continu* (SàFC). Ces systèmes ont les trois caractéristiques suivantes :

1. Positifs, Un système dynamique est dit positif si toutes ses variables d'état prennent toujours des valeurs positives dans le temps, la positivité est souvent une conséquence de la nature du système.
2. Linéaire, Nous considérons des systèmes dont les variables continues sont linéaires au sens de l'automate hybride qui sera définie plus tard, *i.e.* elles évoluent suivant des équations différentielles du type $\dot{x} = k$, où k est une constante rationnelle.
3. La partie discrète contrôle la partie continue, dans le sens ou la configuration de la partie discrète est complètement autonome et influe sur celui de la partie continue.

Exemples illustratifs

Les exemples des SàFC sont nombreux, nous pouvons citer ici quelques exemples : une machine qui contrôle un flux de production, un feu tricolore qui contrôle un flux de véhicules, une décision de routage qui contrôle le flux d'écoulement de messages, …etc. Nous présenterons dans la suite deux exemples de SàFC, le premier est un système de bacs ou l'état (discret) de la vanne contrôle le flux (continu) du liquide et le second est un système manufacturier traitant un flux (continu) de produits contrôlé par l'état de la machine.

Exemple 1.1. Soit le système de bacs schématisé en figure 1.6 ci contre. Ce système comporte deux bacs qui sont vidés en permanence (sauf dans le cas où les bacs sont vides) à un débit de 5 et 7 litres/seconde respectivement. Les bacs sont aussi alimentés à tour de rôle, avec une vanne dont le débit est 12 litres/seconde. Cette dernière à deux positions, quand elle est en position A, elle alimente le bac1 et elle alimente le bac2 en position B. Dans ce système les variables d'état

continues positives sont h_1 et h_2 les volumes du liquide dans les bacs. Le comportement des variables h_1 et h_2 est contrôlé par la variable discrète, position de la vanne.

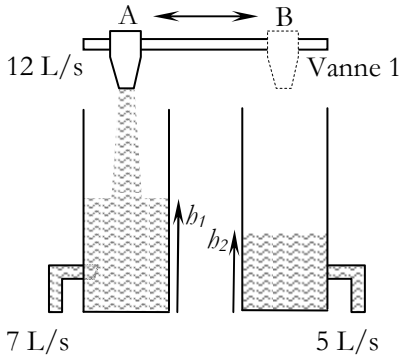


Figure 1.6. Système de bacs

❑

Exemple 1.2 : La figure 1.7 représente un système manufacturier comportant 3 machines et 2 stocks tampons. Ce système est utilisé pour satisfaire une demande périodique comme indiqué sur la figure 1.7 ci-après. Les machines 1 et 2 restent opérationnelles en permanence, tandis que la machine 3 peut être arrêtée. Les machines ont des taux de fabrication de 10, 7, et 22 pièces/min respectivement. Dans ce système les contenus des stocks sont des variables d'état continues, positives et linéaires. L'état continu du système dépend des variables discrètes : 1) l'état de la machine 3, qui peut prendre deux valeurs : arrêt et marche ; et 2) l'état de la demande qui prend deux valeurs 15 pièces/min et 0 pièces/min.

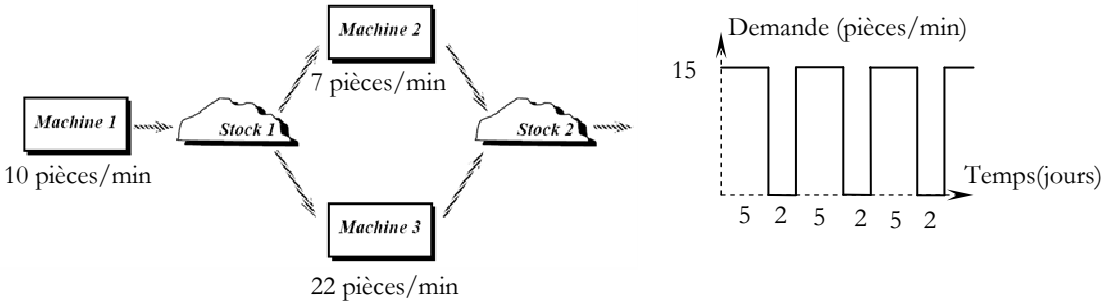


Figure 1.7. Système manufacturier

❑

De nombreux formalismes ont été développés pour la description du comportement hybride des systèmes dynamiques. L'outil de base est l'automate hybride (AH). Il regroupe dans un même formalisme les équations différentielles qui sont l'outil de base pour la description des systèmes continus ; et les automates a états finis qui sont l'outil de base de description des SED. Nous introduirons dans la suite ce formalisme.

II. Les automates hybrides

Un automate hybride est un automate à états finis qui a été étendu par des variables continues. Dans chaque sommet discret, la dynamique des variables continues est définie par des équations différentielles. Et les transitions entre les états discrets dépendent des valeurs des variables continues. Nous commençons dans la suite par définir les automates à états finis. C'est à partir de ce formalisme que les automates hybrides ont été définis.

II .1 Les automates à états finis

Définition 1.1 (automate à états finis) : un automate à états finis est un quadruplé $AEF = (Q, \Sigma, T, q_0)$ où :

- Q = est un ensemble fini de sommets ;
- Σ est un ensemble fini d'évènements ;
- δ est un ensemble de transitions entre sommets. Une transition est un triplet $t_i = (q, \sigma, q')$, ou q est un sommet source, σ est un événement de Σ et q' est un sommet but.
- q_0 est l'état initial de l'automate ;

❑

Dans un automate à états finis chaque sommet représente un état du SED. La transition $t_i = (q, \sigma, q')$ traduit le passage du système du sommet q vers le sommet q' suite à l'occurrence de l'événement σ.

Exemple 1.3 : Considérons la vanne 1 dans le système de bac (exemple 1.1). Cette vanne peut avoir deux états (position A ou position B). Notons par α le passage de la position A vers la position B et par β le passage de la position B vers la position A. Initialement la vanne est supposée en position A. La figure 1.8 ci-contre représente l'automate décrivant ce cette dynamique événementielle.

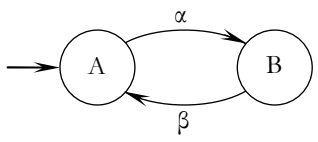


Figure 1.8. Automate à états finis modèle d'une vanne

❑

Le comportement global d'un SED est décrit par l'ensemble des séquences d'événements qui peuvent être exécutées en parcourant l'automate à partir de son état initial. Cet ensemble correspond à un langage issu de l'alphabet Σ, exprimé par une expression régulière permettant d'agréger les séquences d'événements répétitives. Ainsi l'expression régulière (αβ)*(ε + α) correspond au langage décrit par l'automate de la figure 1.8. Le symbole ε correspond à la séquence de longueur nulle (événement vide) et (αβ)* indique que la séquence αβ peut être exécutée autant de fois que l'on veut.

Les automates hybrides ont été définis par Allur *et al.,* [ACH+95] comme un automate à états finis pilotant un ensemble d'équations différentielles continues. A chaque sommet q d'un automate hybride on associe une fonction d'évolution F_q, sous la forme $\dot{x} = F_q(x)$ et un prédicat

sur la valeur des variables appelé invariant du sommet. Le système peut séjourner dans le sommet *q* tant que les valeurs des variables continues vérifient l'invariant associé. Le franchissement d'une transition de l'automate hybride est instantané, à chaque transition T on associe une condition de franchissement, appelée garde, et une affectation qui réinitialise les valeurs des variables continues.

Une transition ne peut être franchie que si sa garde est vérifiée par les valeurs des variables (figure 1.9). L'affectation associée à une transition désigne les variables qui effectuent un saut suite au franchissement de la transition. Un automate hybride est formellement défini comme suit :

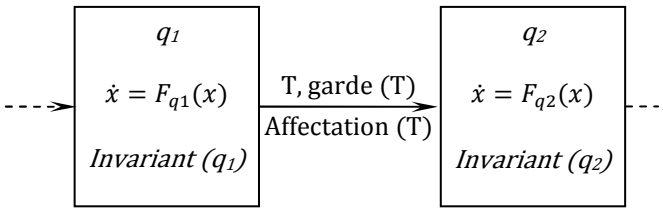


Figure 1.9. Automate hybride

Définition 1.2 (Automate hybride) : un automate hybride est un sextuple A = (*Q*, *X*, *E'*, *δ*, *F*, *Inv*) tel que :

- $Q = \{q_1, q_2, \ldots\}$ est un ensemble fini de sommets ;
- $X \in \Re^n$ est un vecteur d'état comportant n variable réelles ;
- Σ est un ensemble fini d'évènements ;
- δ est un ensemble fini de transitions, chaque transition est un quintuple $t_i = (q, \sigma, g, \gamma, q')$ tel que :
  i. $q \in Q$ est le sommet source ;
  ii. $\sigma \in \Sigma$ est un évènement associé à la transition t_i ;
  iii. g est la garde de la transition t_i, c'est un prédicat sur X ; la transition t_i ne peut être franchie que si sa garde g est vérifiée ;
  iv. γ est la fonction de réinitialisation qui affecte une expression aux variables continues quand la transition t_i est franchie ;
  v. $q' \in Q$ est le sommet but ;
- F est une fonction qui associe à chaque sommet q une fonction continue f_q qui représente l'évolution dynamique du vecteur d'état dans le sommet ;
- *Inv* est une fonction qui associe à chaque sommet q , un prédicat $Inv(q)$, qui doit être vérifié par les valeurs des variables continues lors du séjour de l'automate dans le sommet q.

❑

Exemple 1.4 : Considérons une balle de masse *m* soumise à l'action de gravité. On la laisse tomber d'une hauteur x_0 avec une vitesse initiale nulle. Les variables continus *x* et *v* modélisent respectivement la hauteur de la balle par rapport au sol et sa vitesse. On peut modéliser ce système par l'automate hybride de la figure 1.10. Cet automate a un seul sommet et une seule

transition. Les équations différentielles suivantes décrivent la fonction d'évolution dans le sommet.

$$F(q) = \begin{cases} \dot{x} = v \\ \dot{v} = -g \end{cases}$$

Où g est la constante gravitationnelle. L'invariant $x \geq 0$ représente le fait que la balle ne pénètre pas le sol qui a une hauteur nulle. Quand la balle touche le sol ($x=0$) elle rebondit, ceci implique une commutation autonome dans sa vitesse et un saut autonome dans sa vitesse. Cette dernière change de direction et est réduite par un facteur c (perte d'énergie). Ce changement instantané d'état est modélisé par la transition T, dont la garde est :

$$g(T) = (x = 0 \ \wedge v < 0)$$

Et l'affectation est :

$$\gamma(T) = (v' := -cv)$$

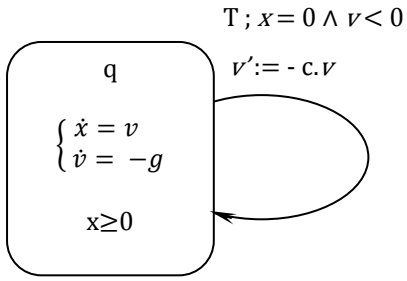


Figure 1.10. Automate hybride modèle de la balle rebondissant

Dans le système de balle rebondissante, la variable x (hauteur de la balle) présente des commutations continues, et la variable v (vitesse de la balle) présente des sauts continus.

❑

L'état d'un automate hybride à un instant *t* est déterminé par le couple (*q*, *v*) tel que $q \in Q$ est un sommet discret et $v \in \Re^n$ est la valeur du vecteur d'état à l'instant considéré. A partir d'un état, le système peut évoluer, soit en franchissant une transition discrète qui change le sommet actif et réinitialise certaines variables, soit par la progression du temps dans le sommet courant, ce qui entraîne un changement permanent de l'état continu *v* conformément à la fonction d'évolution de ce sommet.

Le comportement d'un automate hybride est défini par ses exécutions possibles qui sont des séquences de délais et de transitions discrètes. Formellement une exécution d'un automate hybride est définie comme suit :

Définition 1.3 (Exécution d'un automate hybride) : Une exécution φ est une séquence finie ou infinie de la forme :

$$\varphi = (q_0, v_0) \xrightarrow{t_0, f_0} (q_1, v_1) \xrightarrow{t_1, f_1} (q_2, v_2) \xrightarrow{t_2, f_2} \dots (q_i, v_i) \xrightarrow{t_i, f_i} \dots$$

Avec (q_i, v_i) est un état de l'automate hybride ; la fonction f_i est la fonction d'évolution dans le sommet q_i, et t_i est le temps de séjour dans le sommet q_i. Les conditions suivantes doivent être respectées :

- $f_i(0) = v_i$
- f_i(t) vérifie $Inv(q_i)$ $\forall\ t \in [0, t_i]$
- $\forall\ i, \exists\ T = (q_i, \sigma, g, \gamma, q_{i+1}) \in \delta$ *tel que :*
  - $f_i(t_i)$ vérifie g ;
  - $f_{i+1}(0) = \gamma(f_i(t_i))$;

❑

Un automate hybride est dit *déterministe* si, à partir d'un état initial donné, au plus une seule exécution (trajectoire) est possible. Il est dit non déterministe dans le cas contraire, *i.e.,* à partir d'un état initial plusieurs exécutions sont possibles La plupart des systèmes réels sont non déterministes. Le non déterminisme est une imprécision :

i. Dans les conditions initiales ;
ii. Dans les gardes des transitions ;
iii. Dans les fonctions d'évolution des sommets ;

Pour ces trois paramètres les valeurs peuvent ne pas être précisément connues lors de la modélisation, dans ce cas les valeurs sont données sous la forme d'un intervalle comportant toutes les valeurs possibles. Le non déterminisme rend difficile l'analyse des SDH car, pour caractériser toutes les évolutions possibles, l'ensemble des trajectoires générées par le système doit être pris en compte. Si, pour un vecteur d'entrée constant le problème n'est pas facile à résoudre, dans le cas où il y a également une variation dans ce vecteur, même en utilisant des techniques de simulation numérique, il est difficile de simuler l'évolution du système pour toutes les valeurs du vecteur d'entrée. Les exemples 1.5 et 1.6 ci-après illustrent les notions de déterminisme et non déterminisme dans les SDH.

Exemple 1.5. Considérons l'exemple classique d'un thermostat [Lyg04] utilisé pour maintenir la température d'une chambre dans l'intervalle $[\theta_{min}, \theta_{max}]$. Ce procédé est composé d'un système de chauffage et d'un capteur de température. Le système de chauffage est en mode *ON* jusqu'à la détection du seuil supérieur θ_{max}, et il reste en mode *OFF* jusqu'au moment où la température descend en dessous d'un seuil inférieur θ_{min}. Ce système peut être abstrait par un SDH dont l'évolution continue est définie par la variation de la température x et l'évolution discrète par le passage du système de chauffage entre les états *ON* et *OFF*. L'automate hybride en figure 1.11.a décrit le fonctionnement global de ce système. Dans l'état *OFF* l'évolution de la température

correspond à un système du premier ordre $\dot{x} = -ax$ avec a > 0, et dans l'état ON elle est décrite par $\dot{x} = -ax + b$, avec b > θ_{max}.

❑

L'analyse d'un SDH revient à déterminer son espace d'état atteignable qui est déterminé par l'ensemble des exécutions possibles du système. Pour cet exemple simple, les solutions analytiques des équations différentielles peuvent être facilement trouvées, car ces équations différentielles sont simples à résoudre et la valeur de la température *x* à l'entrée de chaque sommet est déterministe et peut être calculée. L'unique trajectoire de la température est présentée sur la Figure 1.11.c.

❑

Exemple 1.6. Considérons à nouveau le système du thermostat (exemple 1.5), et supposons que le capteur ne soit pas précis et qu'il comporte un écart de mesure de valeur maximale ε. Cela implique une modification des gardes de transitions entre les états ON et Off de l'automate hybride (figure 1.11.b). Les conditions de commutation d'un état vers l'autre de l'automate hybride, exprimées par les intervalles, signifient que le changement d'état peut se faire à n'importe quel instant dès que la température prend une valeur dans l'intervalle spécifié. Le comportement du système est non déterministe dans le sens où pour une même condition initiale les trajectoires de la température peuvent être multiples (Figure 1.11.d).

❑

Une classe restreinte des SDH est la classe des automates hybrides linéaires (figure 1.12). Un automate hybride est dit linéaire si :

- La fonction d'évolution dans tous les sommet est une fonction linéaire de la forme : $\dot{X} = k$, avec k un vecteur constant.
- Les invariants des sommets ainsi que les gardes des transitions sont des prédicats linéaires de la forme : $AX \prec b$; avec A un vecteur rationnel de dimension *n* et *b* une constante réelle. $\prec$ est une relation d'ordre $\prec \in \{=, <, >, \leq, \geq\}$.

Il s'agit ici d'une linéarité par rapport au temps, *i.e.* pour chaque variable, $x_i = k_i.t + x_0$. Les fonctions réinitialisation associées aux transitions sont des fonctions affines. *i.e.,* elles sont de la forme : *X'*:= *AX,* avec A une matrice carrée d'ordre *n.*

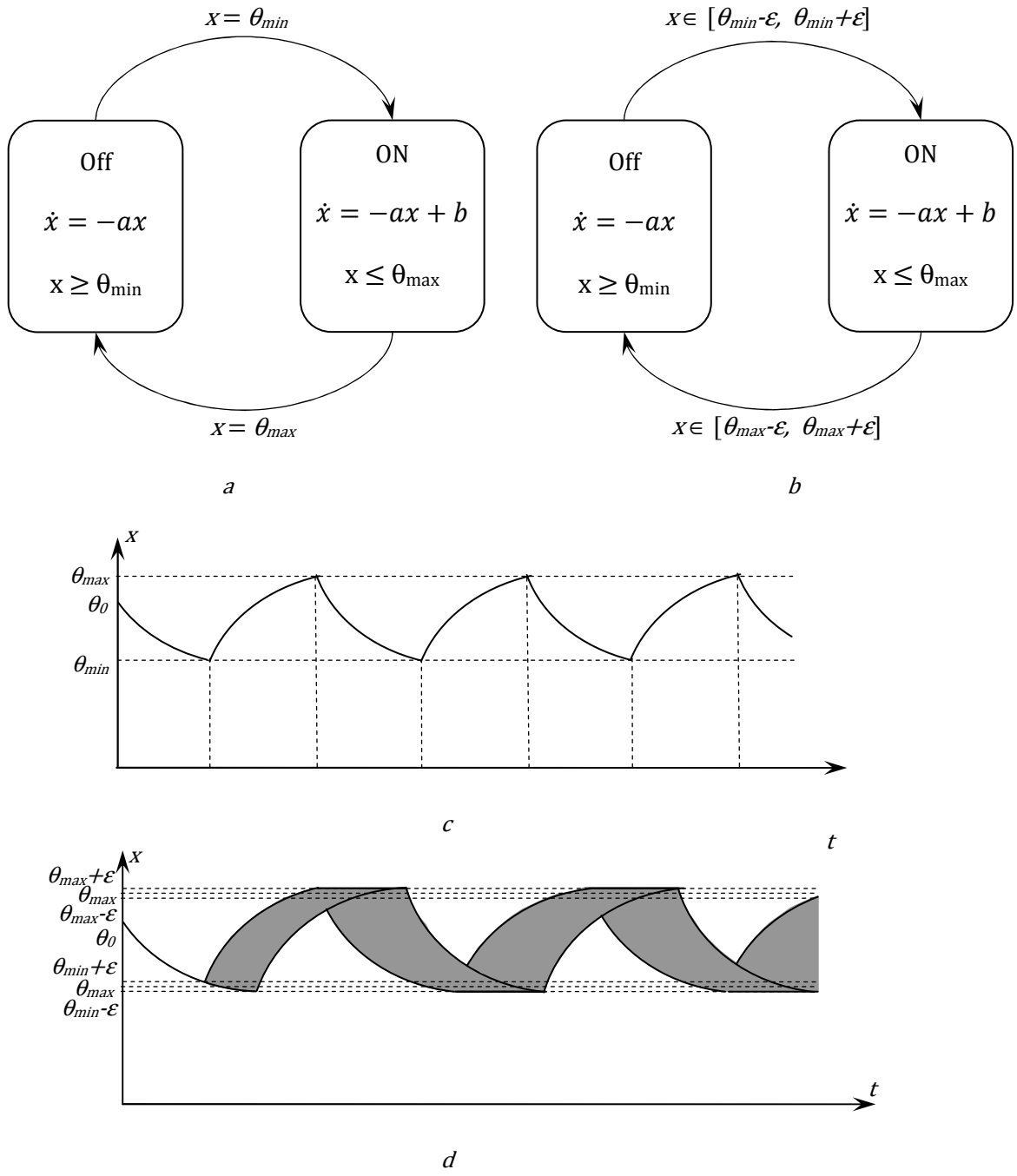


Figure 1.11. a. automate hybride déterministe. b. automate hybride non déterministe. c. exécution de t'automate hybride de la figure 1.11.a. d. exécution de t'automate hybride de la figure 1.11.b.

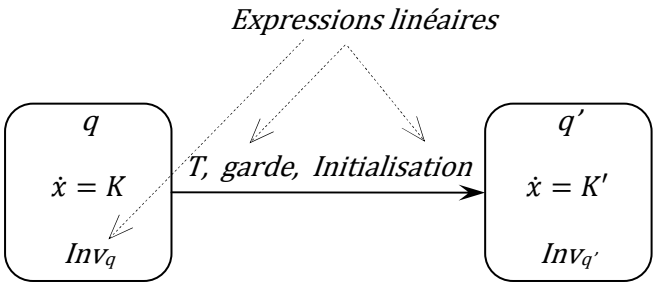


Figure 1.12. Automate hybride linéaire

L'intérêt des automates hybrides linéaires est le fait que tous les paramètres qui les définissent (invariants, gardes, conditions initiales, …) sont linéaires, et que si les régions qui défissent les espaces d'état atteignables définis ci-dessous sont convexes, alors leurs images par les fonctions linéaires d'évolutions continues et discrètes sont également des régions convexes. Une classe restreinte des automates hybrides linéaires est la classe des automates temporisés [AD94]. Dans un automate temporisé les variables d'état continues évoluent uniformément avec le passage du temps ($\dot{x} = 1$), ce sont des horloges. De plus les affectations associées aux transitions peuvent soit remettre à zéro une horloge, soit lui laisser sa valeur avant le franchissement. Le calcul de l'espace des états atteignables est prouvé être décidable pour la classe des automates temporisés, *i.e.* l'algorithme de calcul de l'espace atteignable termine en un temps fini.

II .2 Analyse d'atteignabilité des automates hybrides

L'espace d'états atteignables par un automate hybride, à partir d'un espace d'état initial (q, $\mathcal{X}$), où q est un sommet et $\mathcal{X}$ un espace d'état, est défini comme étant l'ensemble des états visités par toutes les exécutions commençant à partir de (q, $\mathcal{X}$). Il existe deux types d'évolutions à partir d'un état initial, à savoir : une évolution continue, en restant dans le même sommet et en laissant le vecteur d'état évoluer suivant la fonction d'évolution du sommet, ou de manière discrète en franchissant une transition discrète.

Définition 1.4 (successeur continu) : Soit un ensemble d'état (q, $\mathcal{X}$) où $q \in Q$ et $\mathcal{X}$ un espace d'état continu. On définit l'ensemble des successeurs continus de (q, $\mathcal{X}$), qu'on note $R_{Cont}(q, \mathcal{X})$, comme suit :

$$R_{cont}(q, \mathcal{X}) = \{(q, x') / \exists x \in \mathcal{X}, \exists t > 0, (q, x) \xrightarrow{t \ ,f_q} (q, x')\}$$

❑

La figure 1.13.a ci-après représente le successeur continu de l'ensemble d'état (q, $\mathcal{X}$). Le comportement d'un automate hybride dans un sommet discret q est contraint par l'invariant de ce sommet délimité par un trait pointillé sur la figure. Dans cette figure l'état (q, z) ne fait pas partie des successeurs continus de l'ensemble (q, $\mathcal{X}$) tandis que l'état (q, y) en fait partie.

Définition 1.5 (successeur discret) : Soit une transition T, dont la garde est g et la fonction de réinitialisation γ, reliant q à q' et $\mathcal{X}$ un espace d'état continu, où q, $q' \in Q$. On définit l'ensemble des successeurs discret de (q, $\mathcal{X}$), par rapport à la transition T, qu'on note $R_{dis}(q, \mathcal{X})$, comme suit :

$$R_{dis}(q, \mathcal{X}) = \{(q', x') / \exists x \in \mathcal{X} \cap g \wedge x' \in \gamma \cap inv(q')\}$$

❑

Considérons la figure 1.13.b. Le successeur discret de l'ensemble d'état (q, $\mathcal{X}$) est représenté par l'espace d'état gris. Cet espace est l'intersection des la garde g, représentée par un

rectangle et de l'invariant du sommet but *q'*. Il est supposé ici que le vecteur d'état n'est pas réinitialisé.

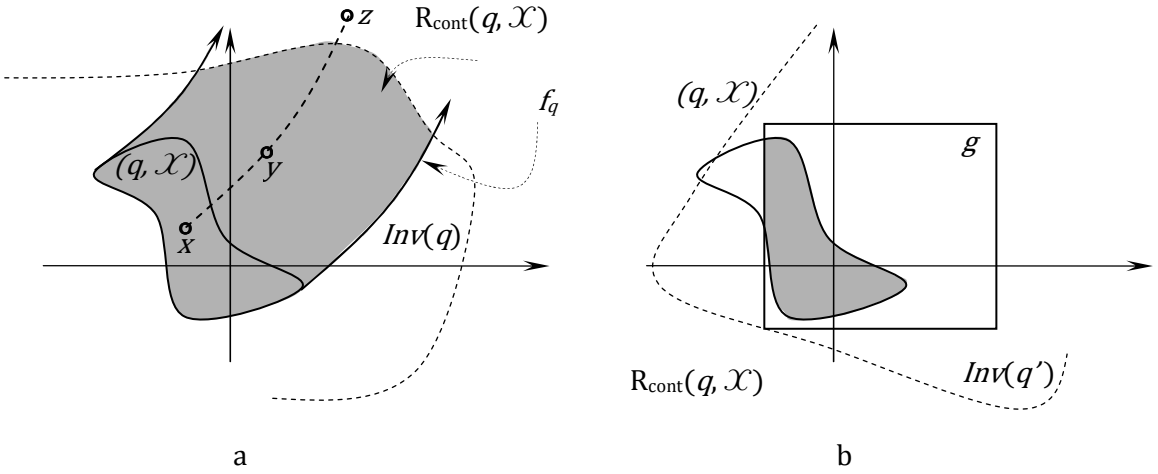


Figure 1.13.a. Successeur continu, b. successeur discret

A l'exception de quelques classes très simples de SDH, le calcul de l'espace d'états atteignables est généralement complexe. Ce problème est prouvé être indécidable [ACH 95, HKPV95]. Cela oblige souvent à calculer une sur-approximation de cet espace. L'algorithme de base pour le calcul de l'espace des états accessibles est un calcul de point fixe. A partir de l'espace des états initiaux, on ajoute l'espace des successeurs continus et l'espace des successeurs discrets jusqu'à ce que cet espace se stabilise, et donc un point fixe est atteint. Le logiciel PHAver a été développé par la communauté informatique, il permet de déterminer l'espace des états accessibles. Lorsque l'algorithme de calcul converge, il donne cet espace pour chaque sommet sous la forme d'inégalités entre les différentes variables continues. Cette formalisation analytique sera utilisée pour le calcul du contrôleur dans le chapitre 3. Les détails du logiciel PHAver sont donnés en Annexe.

III. Les réseaux discrets

Plusieurs modèles construits à partir du formalisme des réseaux de Petri (RdP) ont été définis pour étudier des dynamiques complexes. Le premier pas pour introduire une dynamique continue dans les RdP a été pris par David et Alla en définissant les réseaux de Petri continus [DA87]. Depuis plusieurs travaux ont été consacrés à étendre les réseaux de Petri pour pendre en charge les systèmes hybrides. Une liste de ces travaux peut être trouvée sur la page Web [1].

Nous utilisons ici le modèle RdP hybride D-élémentaire, qui est une combinaison d'un RdP T-temporel et d'un RdP continu à vitesse constante. Pour présenter ces modèles, nous commençons par définir les RdP T-temporel et les RdP à vitesse constante (RdPCC) avant de définir le modèle hybride et une de ses restrictions que sont les RdP hybrides D-élémentaires.

III.1 Notion de base des réseaux de Petri

Si les automates à états finis sont le modèle le plus classique de modélisation des systèmes à événements discrets, les réseaux de Petri ont connu depuis leur invention en 1962 par Carl Adams Petri [Pet62] un réel succès en raison de leur simplicité mathématique, des avantages de leur représentation graphique et de leur compacité. C'est l'un des outils les plus populaires pour la modélisation de systèmes à événements discrets et ses domaines d'applications sont très vastes. Une littérature très abondante existe sur les réseaux de Petri, leurs fondements théoriques et leurs applications pratiques. Nous citons cette liste non exhaustive des articles et ouvrages les plus importants traitant les réseaux de Petri et leurs propriétés ; G.W. Brams [Bra82], [Bra82], T. Murata [Mur89], R. David et H. Alla [DA92], J.L. Peterson [Pet81].

D'une manière informelle, un RdP est un graphe biparti, c'est-à-dire avec deux types de nœuds, les places (représentées par des cercles) et les transitions (représentées par des barres), des arcs permettent de relier une place à une transition ou une transition à une place. Un poids (nombre entier strictement positif) est affecté à chaque arc, ce poids vaut 1 quand ce n'est pas précisé. Le RdP est dit ordinaire si les valeurs de tous ses poids valent 1, et il est dit généralisé dans le cas contraire. L'ensemble des places ainsi que l'ensemble des transitions sont finis et non vides. Chaque place contient un nombre entier (qui peut être nul) de jetons ou marques, c'est le mouvement de ces jetons entre les places qui décrit la dynamique du système. Le marquage d'un RdP est un vecteur dont la dimension est égale au nombre de places et dont les composantes sont des entiers positifs ou nuls. La $i^{ième}$ composante indique le nombre de jetons dans la $i^{ième}$ place.

D'une manière plus formelle les RdP sont définis comme suit :

Définition 1.6 (Réseau de Petri) : Un RdP marqué est un quadruplet $PN = (P, T, Pré, Post, M_0)$ tel que :

- P est un ensemble fini et non vide de places ;
- T est un ensemble fini et non vide de transitions. Les ensembles P et T sont disjoints $P \cap T = \emptyset$;
- *Pré* est l'application d'incidence avant, telle que :

  Pré : $(P \times T) \rightarrow \mathbb{N}$

  $(P_i, T_j) \rightarrow$ Pré (P_i, T_j) = Poids de l'arc reliant la place P_i à la transition T_j ;
- *Post* est l'application d'incidence arrière, telle que :

  Post : $(P \times T) \rightarrow \mathbb{N}$

  $(P_i, T_j) \rightarrow$ Post (P_i, T_j) = Poids de l'arc reliant la transition T_j à la place P_i ;
- M_0 : $P \rightarrow \mathbb{N}$

  $P_i \rightarrow M_0(P_i)$ est le marquage initial de la place P_i.

❑

Dans la suite, les notations suivantes seront utilisées :

i. ${}^{o}T_j$ (resp. ${}^{o}P_i$) représente l'ensemble des Places (*resp.* transitions) d'entrée de la transition T_j (*resp.* place P_i).
ii. T_j^{o} (resp. P_i^{o}) représente l'ensemble des Places (*resp.* transitions) de sortie de la Transition T_j (*resp.* Place P_i).

- ${}^{o}T_j = \{P_i \in P / \text{Pré}(P_i, T_j) > 0\}$;
- $T_j^{o} = \{P_i \in P / \text{Post}(P_i, T_j) > 0\}$;
- ${}^{o}P_i = \{T_j \in T / \text{Post}(P_i, T_j) > 0\}$;
- $P_i^{o} = \{T_j \in T / \text{Pré}(P_i, T_j) > 0\}$;

Les transitions dans un RdP modélisent les événements dont l'occurrence change l'état du système, et chaque état est modélisé par un marquage particulier du RdP. Ce marquage change chaque fois qu'une transition est franchie. Le franchissement d'une transition est conditionné par sa validation. Les définitions de la validation, du franchissement d'une transition et l'ensemble de marquages accessibles sont formalisées ci-dessous :

Définition 1.7 (Validation et q-validation d'une transition) : Une transition T_j est dite validée par le marquage M, si le marquage M satisfait :

$$\forall P_i \in {}^{o}T_j, \frac{m_i}{Pré(P_i,T_i)} > 0$$

T_j est dite q-validée par le marquage M, ce qui signifie qu'elle a la possibilité d'être franchie β fois d'un seul coup, ($\beta \leq q$) , si :

$$\forall P_i \in {}^{o}T_j, \min\left(\frac{m_i}{Pré(P_i,T_i)}\right) = q$$

q est un entier positif, il est appelé *degré de validation* de la transition T_j.

❑

Définition 1.8 (Franchissement d'une transition) : Soit M un marquage d'un RdP et T_j une transition validée par le marquage M. Franchir la transition T_j consiste à :

- Retirer *Pré* (P_i, T_j) jetons de toute place $P_i \in {}^{o}T_j$.
- Ajouter *Post* (P_i, T_j) jetons à toute place $P_i \in T_j^{o}$.

Le franchissement de la transition T_j mène le RdP du marquage M au marquage M', ce qui est noté : M [Tj ⟩ M'. Le marquage M' est donné par :

$$\forall P_i \in P : M'(P_i) = M(P_i) - Pré(P_i, T_j) + Post(P_i, T_j)$$

❑

Définition 1.9 (Séquence de franchissement) : Soit M_I un marquage d'un RdP et S = $T_I T_{II} T_{III} \dots T_N$ une séquence finie de N transitions. S est dite franchissable s'il existe une suite de marquage M_{II} M_{III} $M_{IV} \dots M_{N+1}$.telle que :

$$\forall i \in \{I, II, III, \dots, N\} : M_i [T_i \rangle M_{i+1}$$

Ce qui est noté :

$$M_I [S \rangle M_{N+1}$$

❑

Définition 1.10 (Ensemble des marquages accessibles) : Soit un RdP marqué *PN* et M_0 son marquage initial. L'ensemble des marquages accessibles par *PN* à partir de M_0 est l'ensemble des marquages tel qu'il existe une séquence de franchissement y menant depuis M_0.

❑

Cet ensemble est généralement représenté par un graphe, dit graphe des marquages accessibles du RdP dont les nœuds sont les marquages et les arcs sont étiquetés par les transitions faisant passer d'un marquage à un autre. L'une des méthodes les plus importantes pour l'étude et l'analyse des propriétés des RdP, est l'utilisation du graphe des marquages accessibles. Cet outil permet de générer la totalité des états atteignables à partir de l'état initial, ainsi que les séquences de franchissement nécessaires pour atteindre chaque état. Ce graphe peut très rapidement exploser en fonction du nombre de jetons mis en jeu, ce qui réduit beaucoup le nombre de propriétés pouvant être pratiquement étudiée, et cela constitue la limitation majeure dans l'utilisation des RdP.

Le franchissement d'une transition entraîne une modification du marquage. Cette modification est exprimée par la matrice d'incidence du RdP qui est définie comme suit :

Définition 1.11 (Matrice d'incidence d'un RdP) : La matrice d'incidence W d'un RdP est une application de P x T dans $\mathbb{Z}$ définie par :

$$\forall P_i \in P, \forall T_j \in T, W(P_i, T_j) = Post(P_i, T_j) - Pré(P_i, T_j).$$

❑

Une séquence de franchissement S est caractérisée par un vecteur noté $\bar{S}$. C'est un vecteur de dimension m (nombre de transitions du RdP) dont la $I^{ième}$ composante représente le nombre d'occurrences de la $i^{ième}$ transition dans la séquence S.

Le marquage M obtenu après le franchissement d'une séquence S est donné par l'*équation d'état* suivante :

$$M = M_0 + W\bar{S}$$

Où : M_0 est le marquage initial ;

W est la matrice d'incidence du RdP ;

$\bar{S}$ est le vecteur caractéristique de la séquence S ;

M est le marquage atteint en franchissant S à partir de M_0 ;

III.2 Les réseaux de Petri T-temporels

La prise en compte explicite du temps est un des sujets les plus importants qu'a traité l'informatique des trois dernières décennies. Plusieurs extensions temporisées des RdP ont été proposées. En effet le temps peut être associé à chacun des éléments d'un RdP : les places [Sif79], [CR83], [Kha97], les transitions [Mer74], [Ram74], [Sta78], [RH80], [HV87], les arcs [Wal83], ou sur plusieurs de ces éléments en même temps, [Cer99]. Dans [Boy01] on trouve des présentations complètes de différentes extensions temporisées des RdP ainsi que des comparaisons entre ces modèles. La totalité de ces modèles peuvent être scindés en deux familles, les RdP temporisés d'un coté, et qui trouvent leur origine dans le travail de Ramchandani [Ram74], et les RdP temporels de l'autre et dont l'origine est la thèse de Merlin [Mer74]. De façon informelle, les RdP temporisés utilisent la notion de *durée* fixe à opposer à la notion de *délai* de franchissement pour les RdP temporels.

Les RdP T-temporels (Time Petri nets) ont été introduits par Merlin dans sa thèse [Mer74]. L'idée fondatrice des RdP T-temporels est d'associer un intervalle de temps $[\alpha_j, \beta_j]$ à chaque transition T_j. Si cette dernière est validée de façon continue pendant au moins α_j unités de temps, elle peut être franchie. De plus si elle est validée pendant β_j-α_j unités de temps de manière continue, elle *doit être* franchie. On trouve ici les notions de délais minimum et maximum dans un état au lieu de durée d'un état dans les modèles temporisés. Intuitivement, les RdP temporels généralisent les RdP temporisés et autonomes. C'est cet outil que nous avons retenu dans notre travail.

Définition 1.12 (Les réseaux de Petri T-temporels) : Un RdP T-temporel PN_T est un couple $PN_T = (PN, SIM)$ tel que :

- PN est un RdP autonome marqué ;
- SIM : $T \rightarrow \mathbb{Q}+ \times (\mathbb{Q}+ \cup \{\infty\})$

$$T_j \rightarrow [\alpha_{js}, \beta_{js}]$$

$[\alpha_{js}, \beta_{js}]$ est l'intervalle statique de franchissement de la transition T_j.

❑

SIM, la fonction d'intervalle statique (Static Interval Mapping), associe à chaque transition un intervalle statique de franchissement. Cet intervalle est dit *statique* car pendant l'évolution d'un

tel RdP, des occurrences d'intervalles de franchissement *dynamiques* (qui évoluent avec le temps) apparaissent, ces derniers sont notés [αj, βj].

Comme présenté auparavant, une transition T_j d'intervalle temporel $[\alpha_j, \beta_j]$ peut être franchie à un instant t si et seulement si $t \in [\theta+\alpha_j, \theta+\beta_j]$, θ représente l'instant ou le marquage a validé la transition T_j. C'est donc l'instant du dernier franchissement d'une transition. Un des problèmes posés par l'analyse des RdP T-temporel est la gestion de la multi-validation (Définition 1.7)

Berthomieu et Diaz souligne dans [BD91] la difficulté d'analyse des RdP T-temporels intégrant la notion de la multi-validation. Dans [Boy01], l'auteur a présenté les difficultés liées à la gestion de la multi-validation. Généralement, pour analyser un RdP T-temporel, on suppose que le degré de validation maximal d'une transition est de 1.

L'état d'un RdP T-temporel a été défini de deux manières. Dans la définition originelle, présentée dans [BD91] on considère que l'état d'un RdP T-temporel est constant entre deux franchissements successifs de transitions, seules les transitions discrètes entres états sont considérées. Par contre, d'autres auteurs considèrent deux types de transitions d'état [Lim04], à savoir les transitions discrètes, causées par le franchissement d'une transition, et les transitions continues causées par l'écoulement du temps. La définition originelle est présentée ci-après.

Définition 1.13 (État d'un RdP T-temporel) : Soit le RdP T-temporel PN_T, un état *E* de PN_T est un couple (M, I) avec :

- M : $P \rightarrow \mathbb{N}$ est un marquage du PN_T.
- I : $T \rightarrow (\mathbb{Q}^+ x (\mathbb{Q}^+ \cup \{\infty\})) \cup \{\emptyset\}$

$$T_j \rightarrow I(T_j) = I(T_j) = \begin{cases} [\alpha_j \beta_j] \; si \; T_j \in T^M \\ \varnothing \quad si \; T_j \notin T^M \end{cases}$$

I est le vecteur des intervalles de franchissement possibles pour les transitions validées, avec T^M l'ensemble des transitions validées par le marquage M.

❑

L'état initial d'un RdP T-temporel $S_0 = (M_0, I_0)$ est constitué de son marquage initial M_0 et de vecteur I_0 qui comporte les intervalles statiques des transitions validées par M_0. Il est clair que le nombre d'états atteignables par un RdP T-temporel est infini. Construire un graphe des marquages accessibles est en général impossible, en effet les transitions peuvent être franchies à tout instant dans leurs intervalles de franchissement. Cela entraine que chaque état admet une infinité de successeurs, d'où l'idée de regrouper certains états dans une *classe d'états*, cette idée à pour but de permettre une représentation finie de ce graphe.

Exemple 1.7 : La figure 1.14 représente une station de travail comportant une machine, précédée d'un stock de capacité 3 dans lequel les pièces arrivent une à une et attendent la disponibilité de la

machine si nécessaire. Considérons les contraintes temporelles suivantes sur le fonctionnement du système.

i. Le temps opératoire de la machine prend ses valeurs dans l'intervalle [2, 3]
ii. La période entre deux arrivées successives de deux pièces varie entre 2 et 5 unités de temps.

❑

Dans cet exemple un indéterminisme est associé aux dates d'occurrence des événements. Le comportement de ce système est modélisé par le RdP T-temporel de la figure 1.14.b.

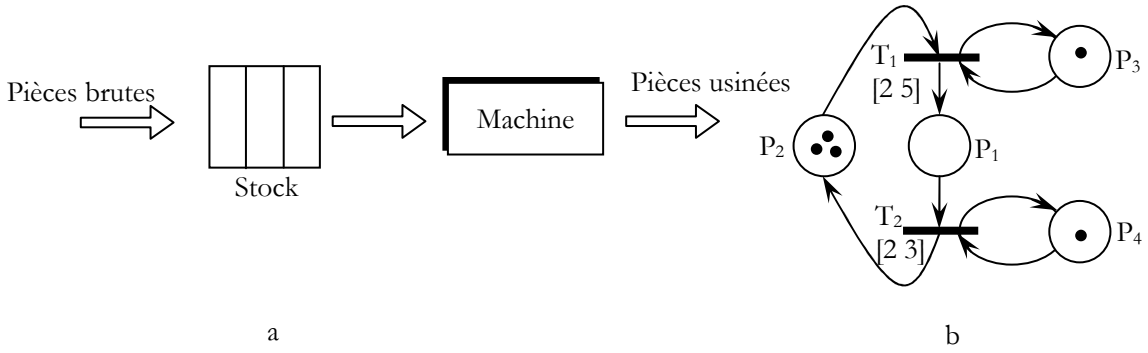


Figure 1.14. a. un système manufacturier simple. b. RdP T- temporel modélisant le système manufacturier

Dans un RdP T-temporel, une transition T_j est franchissable à l'instant t ($t \in R^+$) depuis l'état E = (M, I), ce qui est noté $E[T_j\rangle_t$, si et seulement si les trois conditions suivantes sont satisfaites :

i. T_j est validée par le marquage M.
ii. t appartient à l'intervalle de franchissement de T_j, $t \in I(T_j) = [\alpha_j, \beta_j]$.
iii. Aucune autre transition ne doit être franchie avant t.

$$\forall T_k \in T^M, E[T_k\rangle, T_k \neq T_j, t \leq \beta_k.$$

On note par $E[T_j\rangle_t E'$, le fait que la transition T_j est franchissable depuis l'état E à l'instant t, et son franchissement donne lieu à l'état E'. L'état E' est tel que :

- M' est calculé comme pour un RdP autonome.

$$\forall P_i \in P, M'(P_i) = M(P_i) - Pré(T_j, P_i) + Post(T_j, P_i).$$

- I' est calculé en trois étapes :
  i. Remplacer par ∅ tous les intervalles associés à des transitions validées par M, mais non validée par M – Pré (T_j). Cet ensemble noté D(E, T_j) comporte les transitions dé-validées par le franchissement de T_j depuis E.

$$D(E, T_j) = \{T_k \text{ tel que } E[T_k\rangle, \exists P_i \in P, M(P_i) - Pré(T_j, P_i) < Pré(T_k, P_i)\}$$

Cette première étape permet de définir un vecteur I_1' comme suit :

$$I'_1(T_k) = \begin{cases} \varnothing & si\ T_k \in D(E, T_j) \\ I(T_k) & sinon \end{cases}$$

ii. On décale dans le temps tous les intervalles de I'_1 pour obtenir I'_2.

$$I'_2(T_k) = \begin{cases} [Max(0, \alpha_k - t), \beta_k] & si\ I'_1(T_k) = [\alpha_k, \beta_k] \\ \varnothing & sinon \end{cases}$$

iii. On initialise les transitions nouvellement validées par le nouveau marquage M'. Ainsi l'intervalle I' résultat est obtenu par :

iv.

$$I'(T_k) = \begin{cases} [\alpha_k, \beta_k] & si\ M'[T_k\rangle \text{ et } I'_1(T_k) = \varnothing \\ I'_1(T_k) & sinon \end{cases}$$

La règle de franchissement que nous venons de décrire, définit une relation d'accessibilité sur l'ensemble des états d'un RdP T-temporel. Les *séquences de franchissements* sont définies de la même manière que pour les RdP classiques. Un *échéancier de franchissement* associe une séquence de franchissement S à une séquence de dates ω. C'est une séquence de couples :

$$(T_I, t_I) \rightarrow (T_{II}, t_{II}) \rightarrow \ldots \rightarrow (T_N, t_N)$$

Cet échéancier (S, ω) est dit *réalisable* depuis un état E si les transitions de la séquence S sont successivement franchissables depuis l'état E, aux dates relatives de franchissement qui leur correspondent dans la séquence ω.

Le fonctionnement d'un RdP T-temporel peut être caractérisé par l'ensemble des états accessibles depuis son état initial ou, de façon duale, par l'ensemble des échéanciers réalisables depuis son état initial.

III.3 Analyse d'accessibilité : La méthode des classes d'états

Comme précédemment mentionné, l'ensemble des états atteignables par un RdP T-temporel est en général infini. La méthode des classes d'états permet de grouper un certain nombre d'état en une classe d'état. Cette méthode est une méthode d'analyse par énumération pour les réseaux T-temporels introduit dans [Ber91] et améliorée dans [Lil99]. Elle permet, pour une large classe de réseaux temporels, une analyse d'accessibilité semblable à la méthode du graphe des marquages utilisée pour l'analyse d'accessibilité des RdP autonome.

Une classe d'états regroupe tous les états obtenus depuis l'état initial en franchissant une même séquence S. Tous ces états ont le même marquage, et leurs intervalles de franchissement ne diffèrent que par un décalage de certaines composantes.

Définition 1.14 (Classe d'état d'un RdP T-temporel) : Soit le RdP T-temporel PN_T, une classe d'état du PN_T est un couple C = (M, D) telle que :

- M est un marquage du RdP ;
- D est un polyèdre appelé domaine de franchissement ; Les inéquations définissant D sont de deux types [BD91]

$$\begin{cases} \propto_i \leq t_i \leq \beta_i, \forall i \; tel \; que \; T_i \; est \; validée \\ \gamma_{kj} \leq t_j - t_k \leq \gamma_{jk}, \forall \, j, k \; tel \; que \; j \neq \; k \; et \; T_t, T_k \; sont \; validées \end{cases}$$

t_i représente l'instant de franchissement de la transition validée T_i relativement à l'instant où l'on est entré dans la classe.

❑

La classe initiale d'un RdP T-temporel est donnée par $C_0 = (M_0, D_0)$, où M_0 est le marquage initial et D_0 est le domaine donné par les intervalles de franchissement statiques des transitions validées par M_0. La transition T_j est franchissable depuis la classe C = (M, D) si :

i. T_j est validée par M.
ii. Le domaine D n'est pas vide, autrement dit le système d'inéquations définissant D admet des solutions.

Soit une classe C= (M, D) et une transition T_j franchissable depuis C ; la classe C' = (M', D') successeur de C par franchissement de T_j est donné par :

- Le nouveau marquage M' est calculé de façon classique par :

$$\forall P_i \in P, M'(P_i) = M(P_i) - Pré\,(T_j, P_i) + Post\,(T_j, P_i) ;$$

- Le nouveau domaine de franchissement *D'* est calculé selon les étapes suivantes :
  i. Changement des variables, $\forall i, t_i = t_j + t_i'$;
  ii. $\forall T_i \neq T_j$, Ajout des contraintes $t_i' \geq 0$;
  iii. Elimination des variables correspondant à des transitions dé-validées par le franchissement de T_j (ce qui inclut T_j) ;
  iv. Ajout des inéquations relatives aux transitions nouvellement validées par le franchissement de T_j ;

Le changement de variables modélise l'écoulement du temps pour les transitions validées par un changement d'origine du temps. La nouvelle origine devient l'instant de franchir T_j. Ceci exprime que T_j est franchie en premier et que toutes les autres transitions franchissables seront franchies après. Dans [BER 91] les auteurs ont démontré que le nombre de classes atteints par un RdP T-temporel est fini si :

- Le RdP autonome sous-jacent est borné ;
- Les bornes des intervalles temporelles associés aux transitions prennent leurs valeurs dans $\mathbb{Q}$ (ensemble des rationnels), dans la définition originelle donnée par Merlin [Mer74] [MF76] ces bornes temporelles sont des réels positifs.

Le graphe des classes d'états accessibles (GCEA) par un RdP T-temporel est un graphe dont chaque nœud représente une classe d'états, et les arcs sont étiquetés par les noms de transitions. Ce graphe possède la propriété suivante : Il existe une séquence S menant de la classe $C_0 = (M_0, D_0)$ à la classe C = (M, D) dans le GCEA si et seulement si il existe un échéancier (S, ω) menant de M_0 vers M. En conséquence le GCEA ne peut être utilisé que pour analyser des propriétés d'accessibilité de marquages, qui sont des propriétés non temporelles, ce qui est insuffisant pour l'analyse des propriétés *quantitatives*. D'où l'idée de traduire les RdP T-temporels en automate temporisé, et l'utilisation de ces derniers pour une analyse quantitative (temporelle). Malheureusement, il n'y a pas d'extension de ces techniques pour les RdP hybrides.

IV. Conclusion

Dans ce premier chapitre de la thèse nous avons présenté la classe de systèmes dynamiques hybrides qui nous intéresse, à savoir la classe des systèmes à flux continu. Ses principales caractéristiques sont la linéarité et la positivité. Nous avons aussi présenté le formalisme de base pour la modélisation et l'analyse des systèmes dynamiques hybrides. Les automates hybrides, qui constituent le modèle le plus général et le plus utilisé en analyse, sont présentés. Ce modèle nécessite une énumération exhaustive de l'espace d'état, ce qui permet son analyse formelle. Le chapitre suivant sera consacré à la présentation d'un deuxième modèle qui est le RdPH D-élémentaire. Notre principale motivation pour utiliser les RdP pour modéliser les systèmes dynamiques hybrides est de conserver tous les avantages des RdP.

Chapitre 2

Systèmes Dynamiques Hybrides : Modélisation et Analyse par réseaux de Petri

Dans ce deuxième chapitre nous présentons le modèle réseau de Petri hybride. Nous montrons que c'est un outil élégant de modélisation, cependant pour faire son analyse on est amené à passer au modèle automates hybride. Ce passage de fait par une traduction systématique et permet ainsi une analyse formelle du modèle.

I. Les réseaux de Petri continus

Les RdP continus ont été définis par David et Alla [DA87] comme une limite des RdP discrets, obtenus à partir de ces derniers en fluidifiant les marques. Une place d'un RdP continu est dite place continue ou C-place. Son marquage est un nombre réel positif ou nul. De même, une transition dans un RdP continu est dite transition continue, ou C-transition, elle est validée si toutes ses places d'entrée sont marquées. A l'inverse des RdP discrets où le franchissement d'une transition est un événement instantané, une C-transition est franchie continument dans le temps. Les RdP continus sont un outil de modélisation des systèmes dynamiques où toutes les variables d'état sont continues et positives.

Exemple 2.1 : La figure 2.1.a schématise une station de travail comportant une machine M de capacité unitaire, La machine puise les pièces dans un stock S_1 de capacité C_1 et alimente un deuxième stock S_2 de capacité C_2. Ce système possède $2(1 + C_1)(1 + C_2)$ états, ce nombre peut très rapidement exploser en fonction des valeurs numériques de C_1 et C_2.

Le RdP de la figure 2.1.b, qui modélise le système pour $C_1 = C_2 = 6$, possède 98 marquages accessibles. Il peut atteindre 242 marquages pour $C_1 = C_2 = 10$. Comme mentionné précédemment, il est pratiquement difficile d'analyser un RdP dont le nombre d'états atteignables

est important. La modélisation de ce système par un RdP continu peut pallier à ce problème. Le RdP continu en figure 2.1.c peut modéliser un tel système.

❑

Définition 2.1 (Réseau de Petri continu autonome marqué) : Un RdP continu autonome et marqué est un quintuplé $PN_C = (P, T, Pré, Post, M_0)$ ou on retrouve les même composants d'un RdP discret avec les différences suivantes :

i. Les applications d'incidences avant et arrière *Pré* et *Post* prennent leurs valeurs dans $\mathbb{Q}^+$ (ensemble des rationnels positifs) au lieu de $\mathbb{N}$.

ii. De même les composants du vecteur M_0 prennent leurs valeurs dans $\mathbb{R}^+$ (ensemble des réels) et non dans $\mathbb{N}$.

❑

Définition 2.2 (Validation et q-validation d'une C-transition) : Comme pour un RdP discret, La C-trasition T_j est dite validée par le marquage M si ce dernier satisfait :

$$\forall P_i \in {}^{o}T_j , \frac{m_i}{Pre(P_i,T_j)} > 0$$

T_j est dite q-validée (avec q un réel positif) si et seulement si :

$$\forall P_i \in {}^{o}T_j , \min_i \left(\frac{m_i}{Pre(P_i,T_j)}\right) = q$$

Une C-transition T_j q-validée peut être franchie β fois simultanément, avec β un réel inférieur ou égal à q. β est dite quantité de franchissement de la C-transition T_j. La notation $[T_J]^{\beta}$ dénote le franchissement simultané de la transition T_j β fois.

❑

Le RdP continu en figure 2.1.c modélise le système décrit en exemple 2.1. Pour différencier un RdP continu d'un RdP discret, un double cercle est utilisé pour représenter une C-place et une barre vide pour représenter une C-transition. L'état d'une C-transition est booléen, elle est soit en franchissement soit non. Cela nous permet de remplacer l'ensemble $\{T_2, T_3, P_3, P_4\}$ du RdP discret (figure 2.1.b), qui possède au fait deux états (Soit P_3 marquée, soit P_4 marquée) par la C-transition T_2 dans le RdP continu (figure 2.1.c).

Le marquage atteignable d'un RdP continu est non dénombrable, d'où l'impossibilité de construction d'un graphe des marquages accessibles. Ce dernier est remplacé par un graphe d'atteignabilité, ou chaque nœud modélise un ou plusieurs marquages. La construction de ce graphe est basée sur la notion du macro-marquage [DA05].

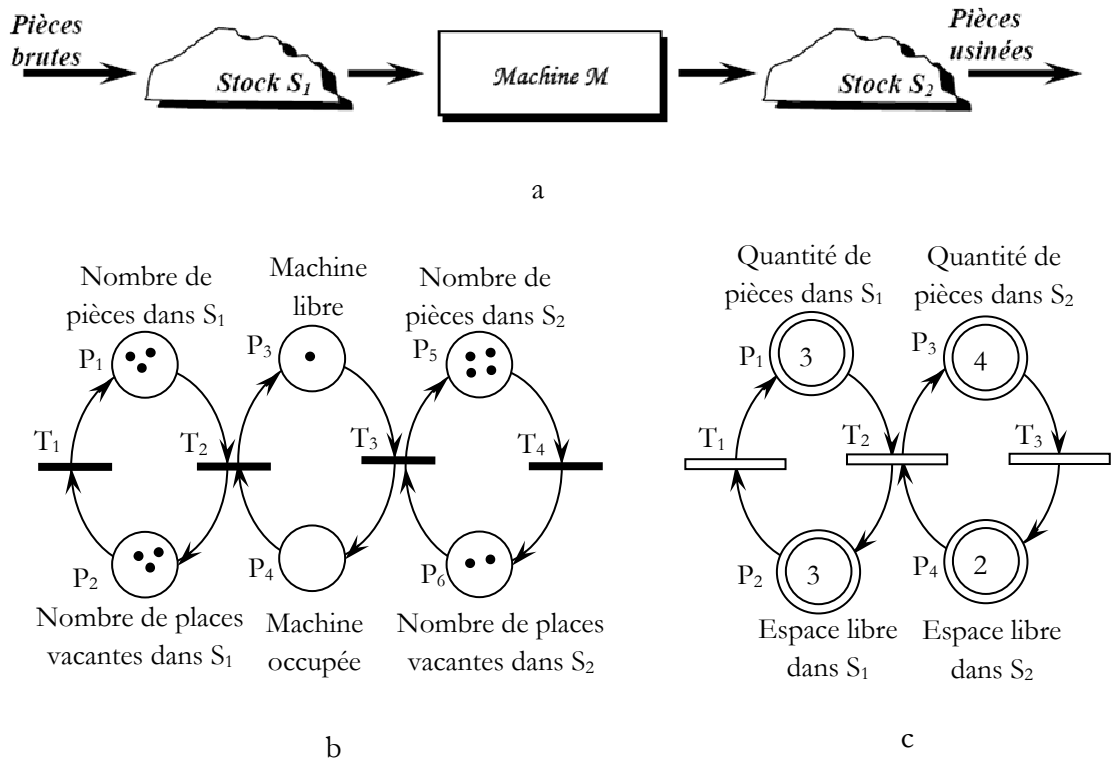


Figure 2.1. a. Station de travail. b. RdP discret modélisant la station de travail. c. RdP continu modélisant la station de travail

Définition 2.3 (macro-marquage) : Soit M_k un marquage d'un RdP continu, l'ensemble des places P peut être divisé en deux sous-ensembles :

i. $P^+(M_k)$ l'ensemble des places P_i tel que $m_i > 0$
ii. $P^0(M_k)$ l'ensemble des places P_i tel que $m_i = 0$

Un macro-marquage, noté M* est l'union de tous les marquages M_k ayant le même ensemble $P^+(M_k)$ de places marquées. Un macro-marquage est caractérisé par son ensemble de places marquées $P^+(M_k)$.

❑

Propriété 2.1 : Un RdP continu ayant n places peut atteindre au maximum 2^n macro-marquages.

❑

Le RdP continu en figure 2.1.c peut au maximum atteindre $2^4 = 16$ états, en réalité il n'a que 7 états atteignables (figure 2.2), puisque les places P_1 et P_2 sont complémentaires, m1 + m2 = 6, quelque soit le marque atteint. De même P_3 et P_4 sont complémentaires, $m_3 + m_4 = 6$. On ne pourra donc jamais atteindre un macro-marquage où m_1 et m_2, ou m_3 et m_4 sont nuls en même temps. Par exemple le macro-marquage $[0\ 0\ 0\ 0]^T$ n'est pas atteignable.

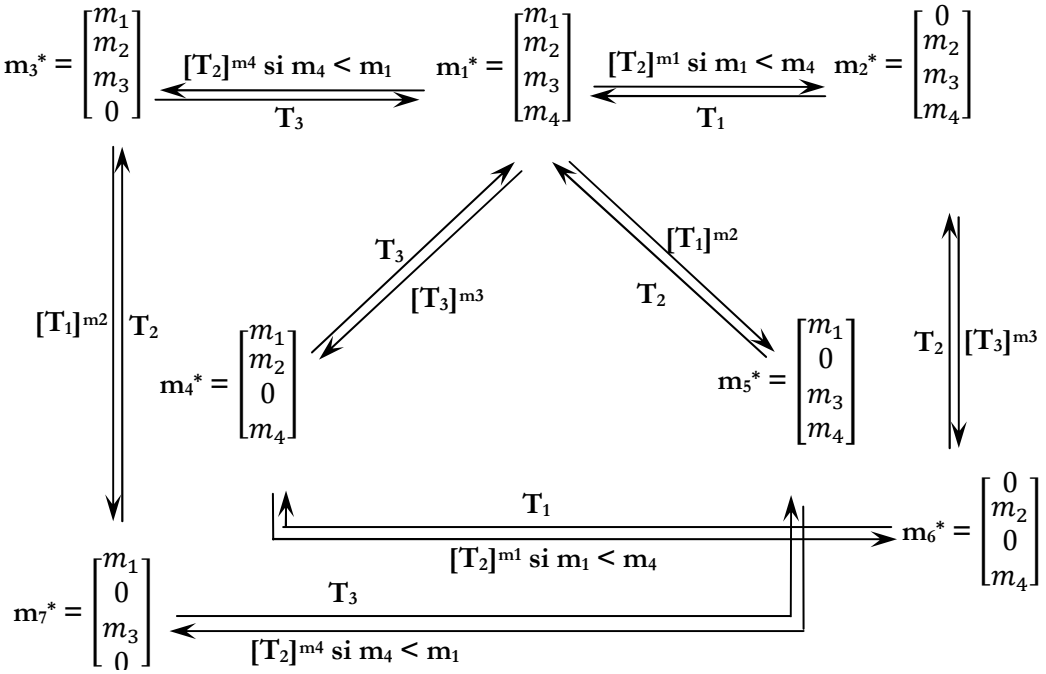


Figure 2.2. Graphe d'atteignabilité du RdP continu de la figure 2.2.c

L'intégration explicite du temps dans les RdP continus a été inspirée des RdP discrets T-temporisés. Dans un RdP continu temporisé, on associe à chaque transition T_j une vitesse maximale de franchissement V_j. Les RdP continus sont utilisés pour la modélisation des systèmes à flux continus, sous ce nom nous regroupons les systèmes continus positifs et les systèmes à événements discrets dont le flux de marques est important. Quatre types de RdP continu ont été définis, la différence entre ces modèles réside dans la manière de définir les vitesses de franchissement des transitions. Dans ce qui suit les RdP continus à vitesse constante seront présentés en détail car ce sont ceux-là qui nous intéressent dans notre recherche, les autres modèles seront brièvement présentés ensuite.

I.1 Le RdP continu à vitesse constante

Des quatre modèles RdP continu, celui à vitesse constante (RdPCC) a été le premier à être défini. C'est aussi le plus simple et le plus aisé à comprendre sur le plan intuitif. Une vitesse de franchissement maximale est associée à chaque transition T_j. Tant que la place d'entrée de la transition n'est pas vide, la transition est franchie selon sa vitesse maximale. Si la place d'entrée est vide mais qu'elle est alimentée en amont par une autre transition T_k, alors la vitesse de franchissement est le minimum des vitesses maximales des deux transitions T_j et T_k.

Définition 2.4 (RdP continu à vitesse constante) : Un RdPCC est un couple $PN_{CC} = (PN_C, V)$ tel que :

- PN_C est RdP continu autonome ;
- $V : T \rightarrow \mathbb{R}^+$

$T_j \rightarrow V_j$, la vitesse de franchissement maximale de la transition T_j ;

❑

Le fonctionnement de ce modèle est clairement exprimé à travers l'équation fondamentale d'un RdP. L'équation fondamentale du RdP discret qui est

$$M' = M + W \,.\, \bar{S}$$

devient pour un RdPCC

$$\dot{M} = W.\bar{S}$$

Où W est la matrice d'incidence du RdP continu et *v(t)* est le vecteur des vitesses de franchissements instantanées des transitions. A titre d'illustration considérons les deux exemples suivants :

Exemple 2.2 ***:*** Considérons un système constitué de deux réservoirs et de trois vannes, comme illustré dans la figure 2.3.a. Les vannes sont caractérisées par leur débit V_1, V_2 et V_3 (litres/seconde), et sont supposées être ouvertes en permanence.

❑

Exemple 2.3 : Un système manufacturier comporte trois machines M_1, M_2 et M_3, dont les vitesses de traitement des pièces sont respectivement V_1, V_2 et V_3 (pièces/seconde). Le système comporte aussi deux stocks tampons S_1 et S_2 pour l'attente des disponibilités des machines (figure 2.3.b).

❑

Les deux systèmes décrits dans les exemples 2.2 et 2.3 sont modélisés par le RdPCC de la figure 2.3.c. pour les valeurs numériques suivantes : $V_1 = 3$, $V_2 = 4$ et $V_3 = 6$. Pour l'exemple 2.2, le modèle est exact alors que pour l'exemple 2.3 il s'agit d'une approximation.

Les marquages des places P_1 et P_2 représentent, les quantités de liquide dans les réservoirs (exemple 2.2) ou le nombre des pièces dans les stocks (exemple 2.3). Les vitesses associées aux transitions du RdP continu modélisent les débits des vannes ou les vitesses des machines.

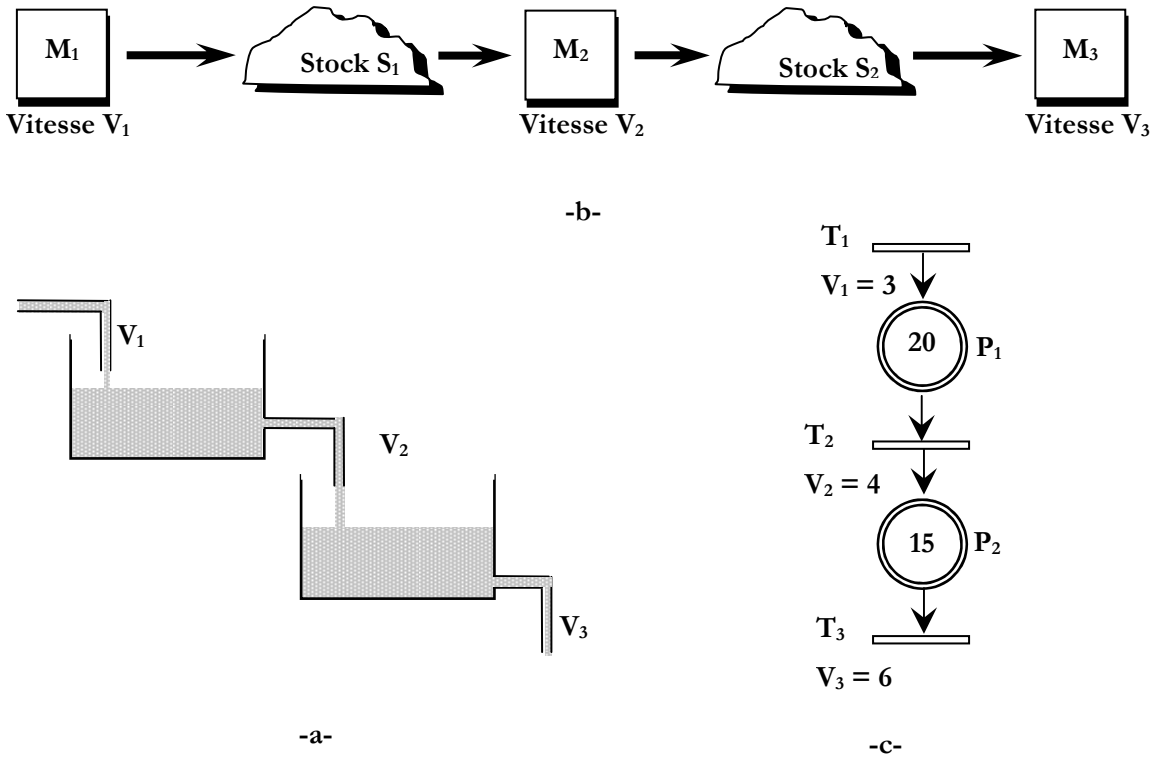


Figure 2.3.-a-Système hydraulique. –b- Système manufacturier.
-c- RdPCC modèle des systèmes en figures a et b

A l'instant initial, les trois transitions sont fortement validées, une transition est dite fortement validée si toutes ses places d'entrée sont marquées, et elles sont franchies à leurs vitesses maximales. Le marquage des places P_1 et P_2 (Figure 1.2.14) évolue suivant les équations suivantes :

$$m_1 (t + dt) = m_1 (t) + (V_1 – V_2).dt$$

$$m_2 (t + dt) = m_2 (t) + (V_2 – V_3).dt$$

$$\text{Puisque } M_0 = [20\ 15]^T$$

$$m_1 (t) = 20 - t$$

$$m_2 (t) = 15 - 2.t$$

Ces équations restent vraies aussi longtemps que $m_1 > 0$ et $m_2 > 0$. A t = 7.5s m_2 s'annule, ce qui empêche T_3 d'être franchie à sa vitesse maximale. Mais puisque la place P_2 est toujours alimentée à une vitesse de 4 litres/s par la transition T_2, T_3 est aussi franchie avec cette même vitesse, qui n'est plus sa vitesse maximale, T_3 est dite faiblement validée (Sa seule place d'entrée n'est pas marquée mais elle est alimentée). A t = 20s m_1 s'annule, ce qui empêche T_2 d'être franchie à sa vitesse maximale, elle sera franchie à la vitesse de T_1.

Par convention les vitesses maximales de franchissement sont notées par des majuscules alors que les vitesses instantanées de franchissement par des minuscules. Le vecteur des vitesses instantanés du RdPCC en figure 2.3.c est donné par :

$$\begin{pmatrix} v_1(t) \\ v_2(t) \\ v_3(t) \end{pmatrix} = \begin{cases} [V_1\ V_2\ V_3]^T\ Pour\ 0 \le t \le 7.5 \\ [V_1\ V_2\ V_2]^T\ Pour\ 7.5 \le t \le 20 \\ [V_1\ V_1\ V_1]^T\ Pour\ t \ge 20 \end{cases}$$

L'évolution du marquage en fonction du temps est illustré en figure 2.4.

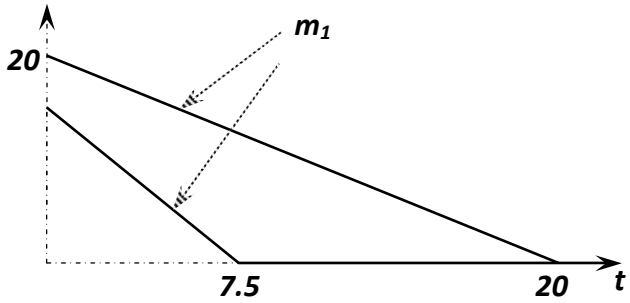


Figure 2.4 Illustration du comportement du RdPCC en figure 2.3.c.

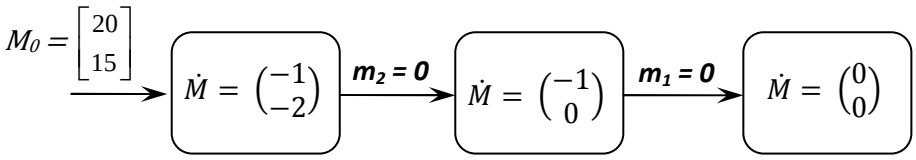


Figure 2.5 Graphe d'évolution du RdPCC en figure 2.3.c

L'état d'un RdPCC est caractérisé soit par le vecteur des franchissements instantanés ou, par dualité, par les dérivés des marquages. Le seul événement susceptible de changer l'état d'un RdPCC est que le marquage d'une place devient nul. Le comportement de ce modèle est généralement représenté par un graphe d'évolution dont les nœuds correspondent aux vecteurs des dérivées des marquages et les arcs sont étiquetés par les places dont l'annulation de marquage a causé le changement d'état. Il est évident que chaque état correspond à un macro-marquage. Une des propriétés les plus importantes du modèle RdPCC est que son graphe d'évolution a toujours un nombre fini de nœuds même si le RdPCC est non borné. À titre d'illustration le graphe d'évolution du RdPCC de la figure 2.3.c est représenté dans la figure 2.5. Le graphe d'évolution d'un RdPCC est facilement assimilable à un automate hybride particulier, où les variables d'état sont données par les marquages des C-places et les gardes des transitions sont de la forme $m_i = 0$.

I.2 Le RdP continu à Vitesses Variables [DA92]

Dans ce modèle la vitesse maximale de franchissement d'une transition dépend du marquage des places en amont de la transition et d'une valeur constante associée à la transition. Les marquages et les vitesses sont donc des fonctions continues du temps, alors que dans le RdP continu à vitesse constante cette propriété est vraie pour les marquages mais pas pour les vitesses qui sont simplement constantes par

morceaux. Ce modèle fournit une meilleure approximation du RdP discret, en particulier lorsque le nombre de marques est petit, mais les simulations sont plus longues. Ici, le comportement événementiel est perdu, il faut discrétiser le temps pour faire une simulation.

I.3 Le RdP continu Asymptotique [Bail et al. 93]

Il est possible de distinguer plusieurs phases d'évolution du marquage. Chaque phase est caractérisée par une période durant laquelle le vecteur des vitesses de franchissement est constant. La vitesse de franchissement d'une transition est alors constante par morceaux, et l'évolution du marquage est donc composée de plusieurs phases de fonctionnement. Il s'agit de l'approximation du RdP Continu à Vitesses Variables, mais la simulation est facilitée.

I.4 Le RdP continu à Vitesses Fonction du Temps [Dubois et al. 94]

Cette extension des RdP continus temporisés permet de prendre en compte l'environnement du système modélisé comme par exemple la fluctuation de l'approvisionnement d'un atelier ou l'influence d'une commande extérieure. La vitesse maximale de franchissement d'une transition est soit une fonction continue du temps, soit une fonction constante par morceaux. La vitesse de franchissement à un instant donné est déterminée par une relation identique à celle du RdP continu temporisé, avec comme seule différence une vitesse maximale fonction du temps. Dans le cas de vitesses constantes par paliers, le RdP obtenu peut être vu comme une suite de RdP continus à vitesses constantes.

Le modèle RdP continu est donc utilisé soit pour modéliser un système continu, soit pour constituer une approximation d'un système à événements discret, plus ou moins finement selon le modèle utilisé. Dans ce dernier cas, il permet de réduire le nombre d'événements qui devient important lorsqu'un modèle RdP discret contient un grand nombre de jetons. Ces modèles ne traitent pas la dynamique du système événement par événement mais suivent les évolutions moyennes.

II. Les réseaux de Petri hybrides

Les réseaux de Petri continus sont adaptés pour la modélisation d'un système à flux continu mais ne permettent pas de représenter l'influence de conditions logiques sur ce flux. Considérons le cas de l'exemple 2.2, le système décrit dans cet exemple comporte des vannes, le transfert de matière à travers ces vannes a deux états, si la vanne est ouverte, le débit est traduit par une vitesse de franchissement d'une transition $v(t)$, et si la vanne est fermée, la vitesse correspondante devient alors nulle. Ce comportement est équivalent à avoir brusquement un autre RdP continu, ou bien à faire varier la vitesse de franchissement d'une transition dans un réseau de Petri continu à vitesse fonction du temps. Les RdP hybrides [DA01] contenant des nœuds discrets (des D-places et des D-transitions) et des nœuds continus (des C-places et des C-transitions) permettent de représenter au sein d'un même formalisme ce couplage entre un fonctionnement continu et un fonctionnement logique (représentation graphique unifiée des variables continues et des variables discrètes).

Un RdP hybride (RdPH) est un modèle combinant un RdP discret et un autre continu, aucune condition n'est imposée sur la nature des RdP discret et continu, en effet ces derniers

peuvent être de n'importe quelle nature. Toutefois le modèle RdPH de base combine un RdPCC et un RdP discret T-temporisé, et c'est le modèle défini comme suit.

Définition 2.5 (RdP hybride) : Un RdP hybride est une structure $PN_H = (P, T, h, E, \Sigma, Pré, Post, Tempo, V, M_0)$, tel que :

- $P = \{P_1, P_2, \ldots, P_n\}$ est un ensemble de n places, $P = P^C \cup P^D$ avec :

  $P^C = \{P_1, P_2, \ldots, P_{nC}\}$ est l'ensemble fini de places continues (ou C-places) ;

  $P^D = \{P_{nC+1}, \ldots, P_n\}$ est l'ensemble fini de places discrètes (ou D-places) ;

- $T = \{T_1, T_2, \ldots, T_m\}$ est un ensemble de m transitions, $T = T^C \cup T^D$ avec :

  $T^C = \{T_1, T_2, \ldots, T_{mC}\}$ est l'ensemble fini de transitions continues (ou C-transitions) ;

  $T^D = \{T_{mC+1}, \ldots, T_m\}$ est l'ensemble fini de transitions continues (ou C-transitions) ;

- $h : P \cup T \rightarrow \{D, C\}$ est une application qui désigne les nœuds discrets, $h(x) = D$, et les nœuds continus, $h(x) = C$;
- E est un ensemble fini d'événements ;
- $\Sigma : T^D \rightarrow E$ est une fonction qui associe à chaque transition discrète un événement de E;
- *Pré* et *Post* désignent respectivement les applications d'incidence avant et arrière ; ces applications doivent satisfaire la condition suivante :

$$\forall (P_i, T_j) \in P^D \times T^C : Pré(P_i, T_j) = Post(P_i, T_j) ;$$

- $Tempo : T^D \rightarrow \mathbb{Q}^+$ est une application qui associe à chaque D-transition la durée de sa temporisation.
- $V : T^C \rightarrow \mathbb{R}^+$ est une application qui associe à chaque C-transition sa vitesse maximale de franchissement :
- M_0 est le marquage initial, les D-places contiennent un marquage entier positif et les C-places contiennent un marquage réel positif ;

❑

La condition sur les applications d'incidence avant et arrière est repérée sur le RdP par des boucles reliant les D-Places aux C-transitions, elle signifie qu'une marque discrète ne peut pas être fluidifiée par une transition continue. Le modèle RdP hybride ainsi défini permet donc la modélisation des conditions logiques influant sur le comportement du système, mais il permet aussi la modélisation de transformation de marques continues en marques discrètes et vice-versa (formation et éclatement de lots).

Nous numérotons les nœuds du RdP de telle sorte que les nœuds continus aient les indices les plus petits, comme présenté en définition 2.5. Cela fait que la matrice d'incidence a la forme suivante :

$$W = \begin{pmatrix} W_C & W_{cD} \\ 0 & W_D \end{pmatrix}$$

Les RdP élémentaires constituent une classe particulière de RdP hybrides où il n'y a pas de transformation de marquage, du discret vers le continu ou du continu vers le discret. Dans ce modèle le RdP T-temporisé contrôle le comportement du RdP continu C via des boucles connectant certaines D-places à certaines C-transitions, ce qui signifie que ces dernières ne sont validées et par conséquent ne peuvent être franchies que si les D-places sont marquées. Le RdP continu C à son tour peut influencer le comportement du RdP T-temporisé, une D-transition T_j peut avoir comme condition de franchissement le marquage d'une C-place P_i qui atteint un seuil *S*. Graphiquement, ceci est représenté de deux manières soit par une boucle (un arc de P_i vers T_j et un arc de T_j vers P_i) dont le poids est *S* si ce seuil est un seuil supérieur, c'est-à-dire si le marquage de P_i ne peut être supérieur à *S* (figure 2.6.a). Dans le cas contraire, si le marquage de P_i ne doit pas être inférieur à *S*, un arc inhibiteur est utilisé pour relier T_j à P_i (Figure 2.6.b). Et dans les deux cas le franchissement de T_j ne modifie pas le marquage de P_i.

Définition 2.6 (RdP hybrides élémentaire) : Un RdPH élémentaire est un couple (PN_H, I) tel que :

- PN_H est un RdP dont les applications Pré et Post satisfont la condition suivante :

$$\forall (P_i, T_j) \in (P^D \times T^C) \cup (P^D \times T^C), \text{ alors Pré } (P_i, T_j) = \text{Post } (P_i, T_j)$$

- I : $(P_i, T_j) \rightarrow \mathbb{R}$, est une application d'inhibition, si un arc inhibiteur de poids S relie la place P_i à la transition T_j, le franchissement de T_j n'est possible que si le marquage de P_i est inférieur à S.

❑

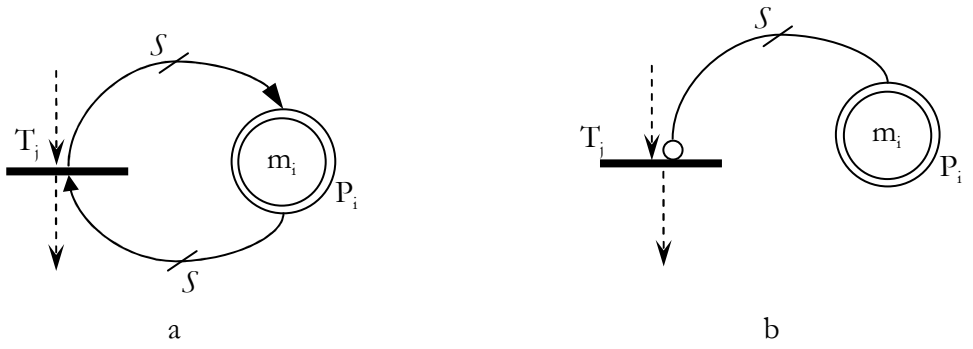


Figure 2.6.a. validation de T_j si $m_i \geq S$; b. validation de T_j si $m_i < S$

Exemple 2.4 : Reprenons l'exemple 2.2, et supposons que l'"on désire limiter le niveau de liquide dans le bac 2 entre S_{min} et S_{max}. En fermant et ouvrant la vanne 2 au bon moment (Figure 2.7.a).

Ce système est modélisé par le RdPH élémentaire de la figure 2.7.b. Dans ce modèle le franchissement de la C-transition T_3 n'est possible que si la D-place P_4 est marquée. De même franchir la D-transition T_4 (T_5) n'est possible que si le marquage de la C-place P_2 est supérieur à S_{mas} (inférieur à S_{min}).

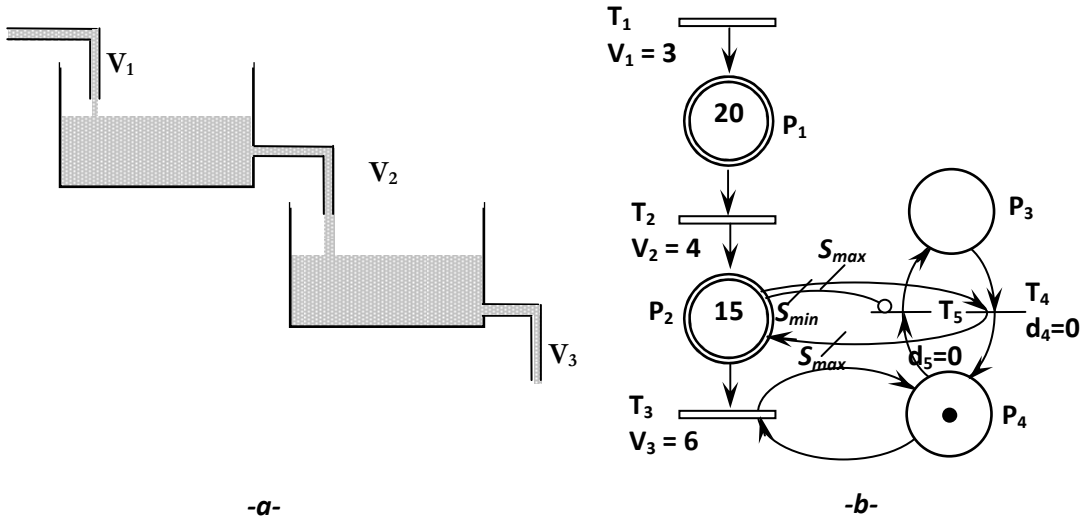


Figure 2.7..a. Système de réservoirs, b. RdPH élémentaire le modélisant

❑

Le modèle RdPH présenté en définition 2.5 est un modèle déterministe dans le sens où, connaissant son état à l'instant initial, on peut connaître exactement son état à n'importe quel instant t et cet état est unique (à condition de choisir une politique de résolution des conflits). L'état de ce modèle est donné par l'état des parties discrète et continue. Les événements qui peuvent changer l'état d'un RdPH sont de deux types, ce sont les événements qui peuvent changer l'état d'un RdP T-temporisé et ceux qui change l'état d'un RdP continu C, à savoir :

i. Une D-transition est franchie ;
ii. Le marquage d'une C-place s'annule ;

Entre deux événements successifs l'état du modèle, donné par les dérivées des marquages des C-places et les marquages des D-places, est constant. Cet état est appelé IB-état (Invariant Behavior state) ou état comportemental invariant.

Définition 2.7 (IB-état d'un RdPH) : Un IB-état d'un RdPH est un intervalle de temps pendant lequel :

i. Le marquage des D-places est constant ;
ii. Les dérivés des marquages des C-places sont constants ;

❑

Le comportement d'un RdPH est généralement modélisé par un graphe d'évolution dont les nœuds représentent les IB-états et les arcs sont étiquetés par les événements (dont

l'occurrence fait changer d'IB-état) avec leurs dates d'occurrence. Le graphe d'évolution du RdPH élémentaire en figure 2.7.b possède 7 IB-états comme présenté en figure 2.8.

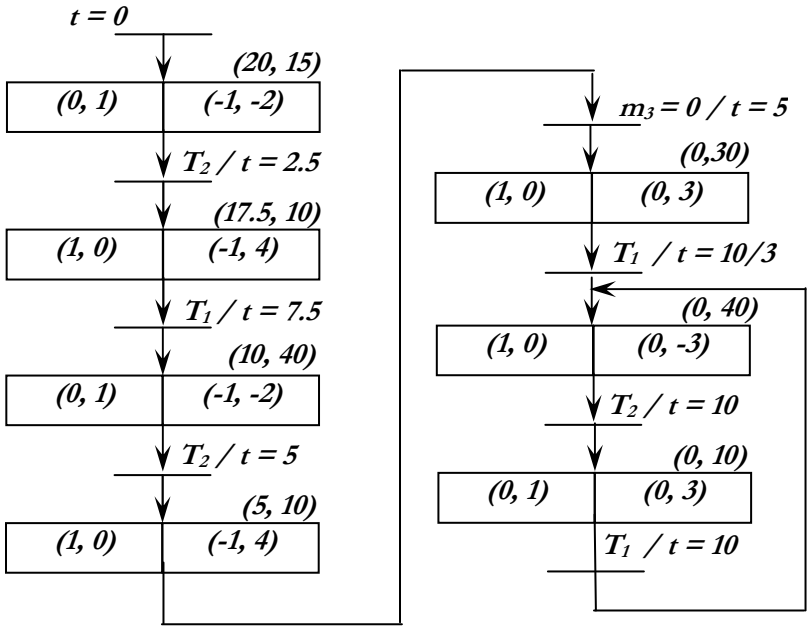


Figure 2.8 Graphe d'évolution du RdPH élémentaire de la figure 2.7.b

III. Les réseaux de Petri hybrides D-élémentaires

Les RdPH D-élémentaires ont été introduits dans [Gho05][GA07][GA08]. Ce formalisme est la combinaison d'un RdP continu C et d'un RdP T-temporel. Le fait que la partie discrète est représentée par un RdP T-temporel confère au modèle hybride un comportement non-déterministe. Rappelons que, pour un RdP T-temporel, les dates de franchissement des transitions ne sont pas déterministes mais prises dans des intervalles de franchissement. De plus dans un RdPH D-élémentaire la partie discrète évolue de manière indépendante par rapport à la partie continue qu'elle contrôle.

Définition 2.8 (RdP hybride D-élémentaire) : Un RdP hybride D-élémentaire est une structure $PN_{HD} = (P, T, h, E, \Sigma, Pre, Post, U, V, M_0)$. Les définitions des paramètres *P*, *T*, *h*, *E*, Σ, *V*, M_0, sont identiques à celles du RdP hybride et :

- *Pre* et *Post* sont les applications d'incidence avant et arrière. Ces applications sont telles que :

  $\forall (P_i, T_j) \in P^C \times T^D$, $Pre(P_i, T_j) = Post(P_i, T_j) = 0$; (1)

  And: $\forall (P_i, T_j) \in P^D \times T^C$, $Pre(P_i, T_j) = Post(P_i, T_j)$; (2)
- U: $T^D \rightarrow \mathbb{R}^+ \times (\mathbb{R}^+ \cup \{\infty\})$ associe à chaque D–transition T_j son intervalle de franchissement $[\alpha_j, \beta_j]$.

❑

Les conditions (1) and (2) sur les applications d'incidence avant et arrière signifient qu'aucun arc ne relie une place continue à une transition discrète. et si un arc relie une place

discrète P_i a une transition continue T_j, l'arc connectant T_j à P_i doit exister. Ceci apparait graphiquement comme des boucles connectant des places discrètes à des transitions continues. Physiquement ceci signifie que le RdP discret évolue indépendamment du RdP continu, mais contrôle l'évolution de ce dernier.

Exemple 2.5 : Reprenons le système de bacs décrit dans l'exemple du chapitre 1 et repris dans la figure 2.9.a. Ce système est modélisé par le RdPH D-élémentaire en figure 2.9.b, avec les hypothèses nouvelles suivantes : A partir de l'instant où on décide de changer la position de la vanne, cette opération prend 3 unités de temps ; les débits des Vanne 1, Vanne 2 et Vanne 3 sont respectivement de 8, 7 et 5 litres/secondes. La partie discrète du RdPH D-élémentaire modélise l'état de la vanne 1. Les D-transitions T_5 et T_6 modélisent le changement de position de la vanne, et les D-places P_3 et P_4 commandent le franchissement des C-transitions T_1 et T_3 respectivement.

Comme un RdPH D-élémentaire combine un RdP discret et un RdP continu, son état à l'instant t est donné par l'état des deux sous-modèles. Le couplage fort entre des RdP discret et continu rend difficile l'analyse du modèle hybride. Sa traduction en automate hybride permet de combiner la puissance de modélisation des RdP discrets à la puissance d'analyse des automates hybrides.

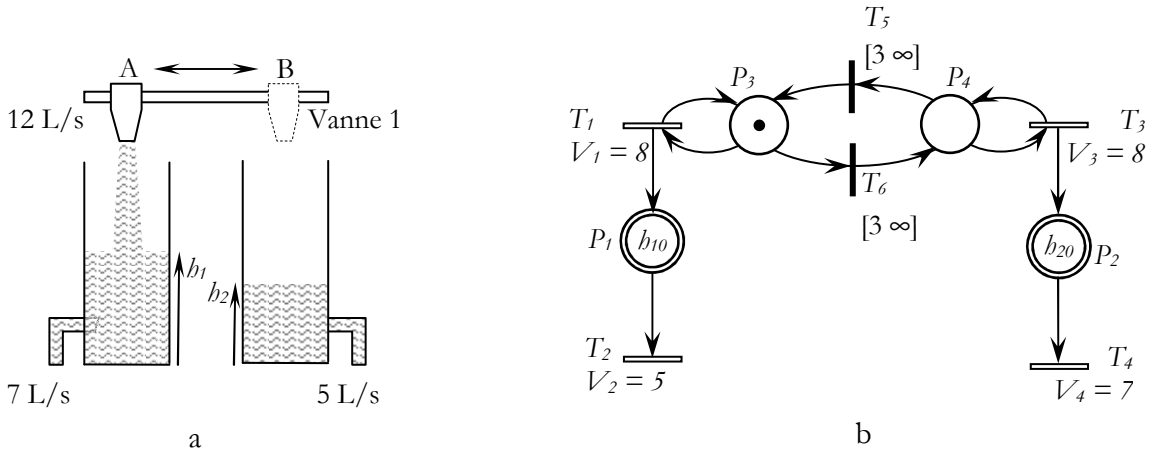


Figure 2.9. a. Système de bacs ; b. RdPH D-élémentaire modélisant le système de bacs

❑

IV. Traduction des RdPH D-élémentaires en automates hybrides

IV.1 La relation entre les automates et les réseaux de Petri

La relation entre les automates et les RdP est ancienne, le principal outil d'analyse des RdP est le graphe des marquages accessibles qui peut être vu comme un automate à états fini étiqueté par les noms des transitions. Ceci est le cas pour plusieurs autres extensions et abréviations des RdP, leurs analyse se fait après leurs traduction en automate.

D'une manière générale, les automates (discrets temporisés ou hybrides) sont des modèles qui permettent une manipulation formelle facile mais sont difficilement utilisables pour la modélisation. Les RdP, au contraire, présentent l'avantage d'avoir une modélisation intuitivement claire et ne nécessitent pas une énumération exhaustive de l'espace d'état. Cependant, ceci rend difficile la tâche d'analyse. D'où l'idée de coupler la puissance d'analyse des automates à la puissance de modélisation des RdP. Cette idée a été utilisée principalement pour les extensions des réseaux de Petri temporels et hybrides.

Dans le domaine des RdP temporels, le premier travail consacré à la traduction des RdP en automates a été proposé par Sifakis et Yovine [SY96] qui étudient une sous-classe des réseaux de Petri temporels à flux, dont le réseau sous-jacent est sauf (STPN). Dans ce modèle, étant donnée une transition T_j et une place amont P_i de T_j, un intervalle de temps [α, β] est associé à l'arc (P_i T_j). Un jeton arrivant dans la place P_i doit attendre une durée comprise entre α et β avant d'être disponible pour la transition T_j. Les auteurs proposent une traduction de ces RdP en automates temporisés ; une horloge est associée à chaque place du RdP. Elle est remise à zéro à chaque fois que la place reçoit un jeton. Les sommets de l'automate correspondent au graphe des marquages du RdP. Les gardes et les invariants sont dérivés des intervalles associés aux arcs.

Dans [BST98], Bornot, Sifakis et Tripakis s'intéressent aux RdP avec échéances (PND) qui sont des RdP saufs, étendus avec des horloges. Un PND peut être défini comme un automate temporisé à échéances (TAD) dont la structure discrète est le graphe des marquages du RdP. Les transitions de ce TAD sont soumises aux mêmes contraintes temporelles que les transitions du PND. Le PND et le TAD ont le même nombre d'horloges. Les auteurs proposent une traduction des RdP temporels en TAD avec une horloge par arc entrant du RdP temporel initial. Les automates temporisés à échéances pouvant être considérés comme des automates temporisés standard, la méthode fournit donc, par transitivité, une traduction des RdP temporels dont le RdP sous-jacent est sauf vers les automates temporisés. Le résultat possède un nombre d'horloges supérieur ou égal au nombre de transitions du RdP initial.

Sava et Alla [SA01] considèrent les RdP T-temporels bornés dont le RdP sous-jacent n'est pas nécessairement sauf et proposent un algorithme calculant le graphe des marquages du RdP T-temporel sous la forme d'un automate temporisé. L'automate résultat possède une horloge pour chaque transition du RdP T-temporel. Ce travail nécessite le calcul de l'espace d'état du RdP T-temporel. L'automate résultat est ensuite utilisé pour la synthèse de contrôleur.

Lime et Roux [LR03] proposent une extension du graphe des classes d'état qui permet de construire le graphe des classes d'états comme un automate temporisé. Les auteurs prouvent la minimalité relative de nombre d'horloges de l'automate résultat.

Cassez et Roux ont développé une méthode de traduction structurelle d'un RdP T-temporel en un automate temporisé [CR04]. Chaque transition du RdP est encodée dans un automate temporisé avec variables. Tous les automates ainsi obtenus sont combinés avec un superviseur en un produit synchronisé d'automates temporisés. Le calcul est très rapide et applicable à tous les RdP T-temporels même non bornés car il est structurel. La méthode n'est cependant pas bien adaptée à la vérification car le produit d'automates possède une horloge par transition du RdP T-temporel et ce nombre ne peut être réduit par des algorithmes à cause de la structure du résultat.

La plupart des extensions des RdP vers l'hybride ont été traduites en automates hybrides en vue de leur analyse. C'est le cas des RdPH présentés à la section. Allam et Alla ont présenté, dans [AA98], un algorithme qui permet la construction de l'automate hybride équivalent au RdPH, l'automate résultat à autant de sommets que d'IB-états du RdPH, les variables d'état étant les marquages des places continues et les horloges qui mesurent le temps pour les transitions validées. L'automate résultat est linéaire et déterministe. Pour assurer la convergence de cet algorithme de passage, le RdPH doit être borné et les temporisations associées aux transitions doivent être des nombres rationnels positifs.

En s'inspirant du travail de Allam et Alla [AA98], Demongodin et Rouibia [DR03] ont présenté une procédure qui permet de systématiser le passage des RdP lots en automates hybrides. L'automate résultat a un sommet pour chaque IB-état du RdP lots. Si ce dernier vérifie les conditions de bornitude et de rationalité des temporisations associées aux transitions discrètes, les auteurs montrent la convergence de l'algorithme. La méthode d'analyse en avant est ensuite utilisée pour calculer la région atteignable, à partir de la région initiale définie par le marquage initial. Le calcul de la région atteignable, permet de décrire le fonctionnement périodique du RdP lots.

IV.2 Principe de l'algorithme de traduction des RdPH D-élémentaires en automates hybrides

Comme mentionné ci-dessus, dans un RdPH D-élémentaire, la partie discrète a un comportement indépendant de la partie continue, son évolution ne dépend que de son marquage initial et de la variable continue indépendante qui est le temps. Elle peut donc être étudiée seule. Pour chaque marquage accessible par le RdP T-temporel, correspondra une configuration du RdPCC. Le modèle RdPH D-élémentaire peut donc être étudié d'une manière hiérarchique, d'abord la partie discrète est considérée, ensuite, pour chaque marquage accessible de cette dernière, la configuration continue est étudiée. L'algorithme de traduction des RdPH D-élémentaires en automate hybride que nous proposons dans ce mémoire comporte les trois étapes suivantes.

1. Isoler la partie RdP T-temporel du modèle hybride et construire son automate temporisé équivalent

Dans le modèle hybride, les parties discrète et continue sont reliées via des boucles (deux arcs de même poids dans les deux sens) connectant certaines D-places à certaines C-transition. Ces boucles constituent les limites entre le RdP T-temporel et le RdP continu C. En cassant ce lien, on aura les deux parties discrètes et continue séparées.

Cette première étape de l'algorithme de traduction consiste à traduire le RdP T-temporel en automate temporisé. Comme précédemment précisé plusieurs travaux ont été consacrés à la traduction des RdP T-temporels en automate temporisés. La plupart d'entre eux ne sont applicables qu'aux RdP T-temporels bornés [SA01] et [LR03]. Puisque, ce n'est que pour cette classe des RdP T-temporel que l'algorithme de traduction converge, l'automate résultat a un nombre fini de sommets.

Cette étape ayant déjà été résolue, nous réutilisons les résultats existants. Considérons à titre d'exemple le RdPH D-élémentaire de la figure 2.10. La limite entre le RdP T-temporel et le RdPCC est schématisée sur la figure 2.10 par la ligne en pointillés. La figure 2.11 représente la partie discrète du RdPH D-élémentaire de la figure 2.10 (figure 2.11.a), ainsi que son automate temporisé équivalent (figure 2.11.b).

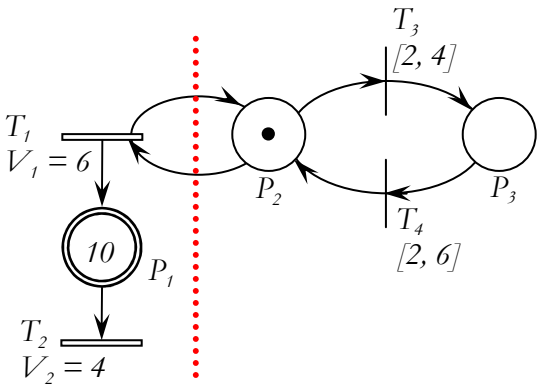


Figure 2.10. Limites entre les parties discrètes et continue dans un RdPH D-élémentaire

2. Construire l'automate hybride correspondant à chaque sommet de l'automate temporisé résultat de l'étape précédente

Nous désignons par macro-sommets, les sommets de l'automate temporisé qui résulte de l'étape précédente, puisque chacun d'eux va correspondre à plusieurs sommets dans l'automate hybride final. Chaque macro-sommet correspond à un marquage du RdP T-temporel, et donc à une configuration du RdPCC, puisque dépendant du marquage des D-places. On peut imaginer la construction de l'évolution du RdPCC indépendamment de celle du modèle discret. Ensuite, on ajoute l'influence ce de dernier. Celle-ci se traduira par le fait que certaines C-transitions peuvent être franchissables ou infranchissables selon l'état de la partie discrète. Une transition infranchissable sera éliminée de l'évolution du RdPCC.

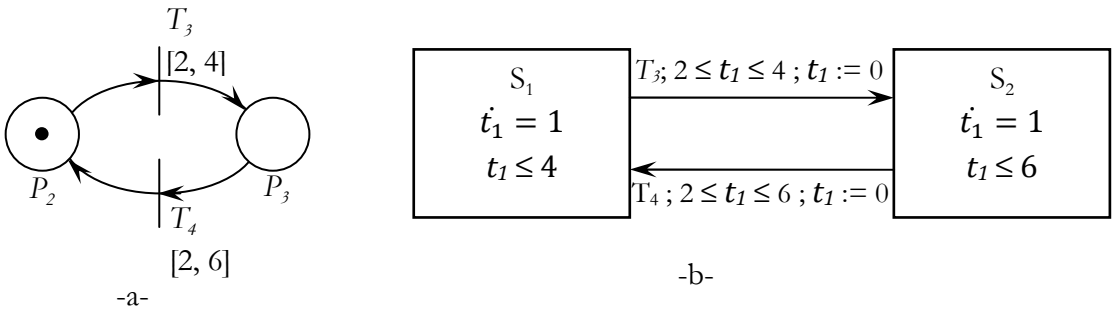


Figure 2.11 a. RdP T-temporel correspondant au modèle hybride de la figure 2.
b. Automate temporisé correspondant

La figure 2.12 représente la configuration du RdPCC pour les deux marquages accessibles par le RdP T-temporel. Pour $[m_3\ m_4]^T = [1\ 0]^T$, Le modèle hybride se réduit au RdP continu C de la figure 2.12.a, et il se réduit au RdP continu C de la figure 2.12.b pour $[m_3\ m_4]^T = [0\ 1]^T$.

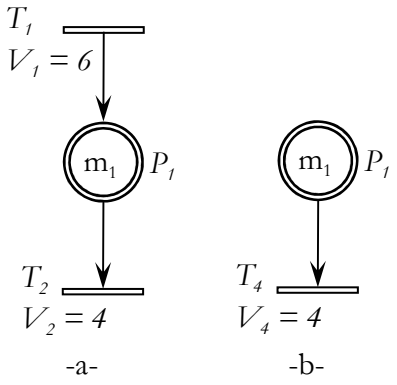


Figure 2.12. -a- RdP continu C correspondant au marquage discret $[1\ 0]^T$ *-b- RdP continu C correspondant au marquage discret* $[0\ 1]^T$.

Le but de cette étape est de construire les automates hybrides modélisant les comportements des RdPCC correspondant à chaque macro-sommet. L'évolution d'un RdPCC est généralement modélisée par son graphe d'évolution, le graphe d'évolution d'un RdPCC est assimilable à un automate hybride linéaire, dans lequel les variables d'état correspondent au marquage des C-places et les transitions modélisent l'événement : marquage d'une C-place s'annule. La difficulté dans la construction des automates hybrides correspondant au RdPCC, dans notre cas, réside dans le fait que le marquage initial ne correspond pas à une valeur déterministe mais à un ensemble souvent infini de valeurs, à cause de l'incertitude dans les dates de commutation entre deux macro-sommets. Pour cette raison, nous allons procéder à une traduction structurelle des RdPCC en automates hybrides, et ceci en supposant que le marquage initial correspond a un macro-marquage dont le support est tout P^C. La notion de macro-marquage, présentée auparavant, à été introduite par David et Alla [DA05] dans le but de représenter d'une manière finie le nombre de marquages infini accessibles par un RdP continu autonome. Pour un RdP continu, autonome ou non, à n places, le nombre maximum de macro-marquages accessibles est 2^n, ceci est dû à la nature booléenne de l'état d'une place dans un macro-marquage.

Propriété 2.2 : Pour un RdPCC, le nombre de macro-marquages atteignables est maximal pour le macro-marquage initial qui a pour support P^C (l'ensemble de toutes les C-places).

❑

Sachant que le seul événement pouvant changer l'état d'un RdPCC est que le marquage d'une C-place s'annule, l'occurrence de cet événement n'est possible que pour les C-place marquées et dont le marquage est décroissant. Un macro-marquage m^* dans lequel la C-place P_i n'est pas marquée, est atteignable depuis le macro-marquage m_0^*, dans lequel P_i est marquée. Le contraire n'est pas possible.

Pour cette raison et pour prendre en compte tous les marquages accessibles par les RdPCC, nous allons considérer que le macro-marquage initial a pour support P^C et construire

l'automate hybride modélisant le comportement de chaque RdPCC sans considération pour son marquage initial. Pour ce faire, nous procédons comme suit :

i. Initialement $m_0^* = P^C$, ce qui signifie que toutes les places sont considérées initialement marquées et par conséquent toutes les transitions sont franchies à leurs vitesses maximales.

Créer le sommet initial de l'automate hybride et lui associer l'activité

$$\dot{M} = W.V$$

Avec W la matrice d'incidence du RdP continu C et V le vecteur des vitesses de franchissement maximales.

ii. Créer depuis le sommet initial une transition pour chaque place P_i dont le marquage est décroissant, et lui affecter la garde $m_i = 0$. Après cette étape on aura une transition pour chaque place dont le marquage est décroissant.

iii. Pour chaque transition T_j construite dans l'étape ii. créer un sommet destination et lui affecter l'activité :

$$\dot{M} = W.v(t)$$

Avec $v(t)$ le vecteur des vitesses de franchissement instantanées [DA92]. Fusionner les sommets ayant la même activité.

iv. Vérifier si pour chaque place P_i dont le marquage est initialement décroissant, le marquage a atteint un bilan nul ($\dot{m}_i = 0$). Si oui arrêter l'algorithme et sinon refaire l'étape ii. pour chaque sommet créé en étape iii.

Si on considère à nouveau le RdPH D-élémentaire de la figure 2.10, après cette deuxième étape, on aura une forme hiérarchique de l'automate hybride, comme schématisée sur la figure 2.13. La figure 2.12.a représente la configuration de la partie continue correspondante au marquage $[1\ 0]^T$, Ce RdPCC a une seule C-place dont le bilan de marquage est croissant, elle est donc modélisée par un automate hybride possédant un seul sommet. La deuxième configuration de la partie continue, est donnée par la figure 2.12.b, et correspond au marquage discret $[0\ 1]^T$. Pour ce RdPCC le marquage de la place P_1 est décroissant et peut donc s'annuler. Ce RdPCC est équivalent à un automate hybride avec deux sommets (Figure 2.13).

3. Remplacer les transitions entre les macro- sommets par des transitions entre les sommets internes

Après l'application de la deuxième étape de l'algorithme de traduction des RdPH D-élémentaires en automate hybrides, on obtient une forme hiérarchique d'un automate hybride comportant des macro-sommets et chacun de ces derniers comporte un automate hybride qui décrit la dynamique continue correspondant au macro-sommet.

Les deux événements susceptibles de changer l'état d'un RdPH D-élémentaire et qui sont : le franchissement d'une D-transition ou le marquage d'une C-place qui s'annule, correspondent respectivement, dans la forme hiérarchique de l'automate hybride que nous avons

obtenu, aux transitions entre les macro-sommets et aux transitions entre les sommets internes aux macro-sommets. Dans une dernière étape, nous remplaçons les transitions entre les macro-sommets par des transitions entre leurs sommets internes.

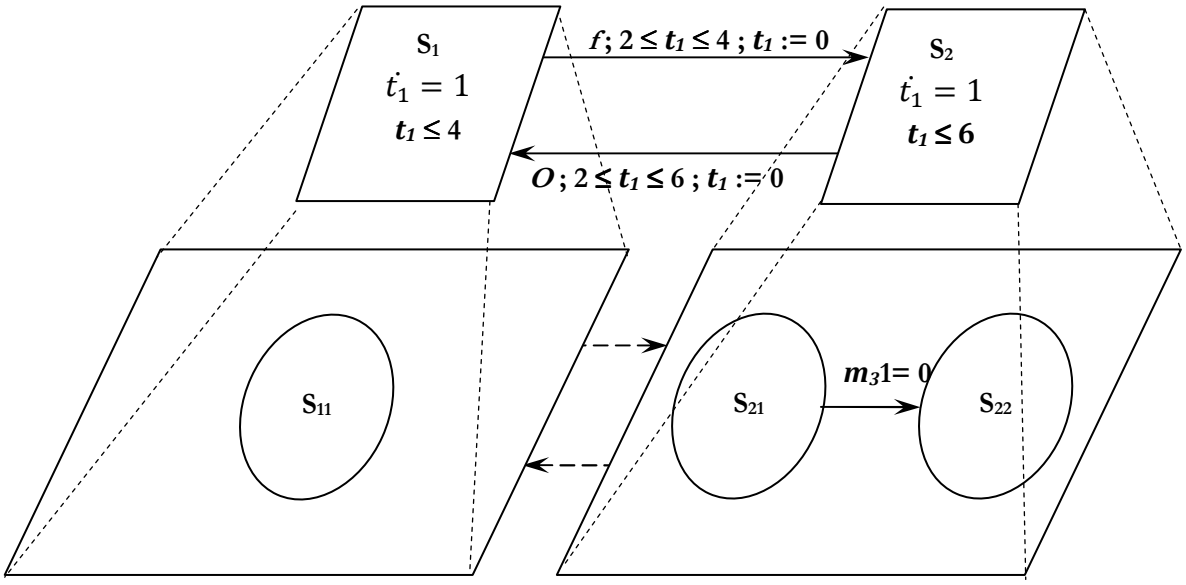


Figure 2.13. Forme hiérarchique de l'automate hybride après la deuxième étape de l'algorithme

Pour cela, considérons la notion de macro-marquages, un macro-marquage d'un RdPCC de n places, peut être caractérisé par un vecteur booléen $\overline{M}$ de dimension n, avec :

$$\overline{M_i} = \begin{cases} 1 \; si \; P_i \; est \; marqué \\ 0 \; sinon \end{cases}$$

Cette étape consiste à remplacer chaque transition entre deux macro-sommets par des transitions entre tous les sommets internes, à condition que le vecteur caractéristique du sommet source soit inférieur ou égal (composante à composante) au vecteur caractéristique du sommet but. Ceci signifie que franchir une transition discrète dans un RdPH D-élémentaire, peut marquer une place vide, mais ne peut pas annuler le marquage d'une place marquée.

La figure 2.14 modélise l'automate hybride résultat de l'application de l'algorithme de traduction au RdPH D-élémentaire de la figure 2.10. Dans la forme hiérarchique de l'automate hybride, résultat de l'étape 2 de l'algorithme, nous avons deux transitions entre les macro-sommets étiquetées respectivement *f* et *O*. La transition f est remplacée dans l'automate hybride final par une seule transition reliant les sommets S_{11} et S_{21}, tandis que la transition *O*, est remplacée par deux transition O_1 et O_2.

Dans la seconde étape de l'algorithme, nous avons construit un automate hybride équivalent au comportement du RdPCC correspondant à chaque macro-sommet, et ceci sans considération pour le marquage initial du RdPCC à l'entrée du macro-sommet. Ceci peut engendrer des sommets dans l'automate final qui ne seront jamais visités. Ce modèle constitue une sur-approximation du modèle exact dans la mesure où il contient toutes les trajectoires réelles ainsi que d'autres qui ne sont pas réalisables. Pour éliminer les sommets non atteignables ainsi que

les transitions associées, nous allons analyser l'automate résultat par le logiciel PHAver que nous présenterons en détail dans le chapitre suivant. Cet algorithme a le grand avantage de proposer une traduction structurelle ?

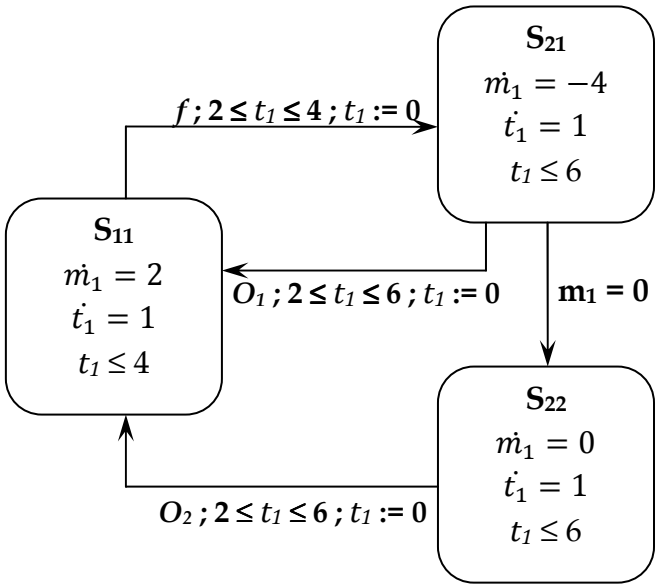


Figure 2.14. Automate hybride résultat de l'application de l'algorithme de traduction au RdPH D-élémentaire de la figure 2.10

IV.3 Particularités de l'automate hybride résultant de la traduction

L'algorithme de traduction des RdP D-élémentaires en automates hybrides donne lieu à un automate hybride linéaire avec un certain nombre de spécificités.

i. Les variables d'état continues sont de deux types :
   - les horloges t_i : elles correspondent aux durées de validation des D-transitions dans le modèle RdPH D-élémentaire ;
   - Les variables m_i : elles correspondent aux marquages des C-places dans le modèle RdPH D-élémentaire ;

   Ainsi X le vecteur d'état est de la forme $X = \begin{pmatrix} M_C \\ - \\ t_D \end{pmatrix}$, tel que : $M_C = (m_1, m_2, ..., m_{nC})^T$ est un vecteur de n_C variables réelles et $t_D = \{t_1, t_2, \ldots, t_k\}$ est le vecteur d'horloges.

ii. Les horloges peuvent être réinitialisées lors de franchissement de transitions, tandis les variables réelles mi ne le sont jamais, elles n'effectuent donc jamais de sauts

iii. Une garde de chaque transition ne dépend que d'une seule variable réelle. Elle peut être sous une des deux formes suivantes :
   - $m_i = 0$, si la transition modélise l'annulation du marquage d'une C-place dans le RdPH D-élémentaire ;
   - $\alpha_j \leq t_j \leq \beta_j$, si la transition modélise une D-transition dans le RdPH D-élémentaire ;

Ces remarques sont importantes, elles constituent les éléments de départ pour la synthèse du contrôleur.

V. Conclusion

Dans ce deuxième chapitre de la thèse nous avons présenté les modèle teéseau de Petri hybride et le RdPH D-élémentaire. La principale motivation pour utiliser les RdP pour modéliser les systèmes dynamiques hybrides, est que tous les avantages qui font des RdP un modèle puissant pour les SED, restent vrais pour les RdPH, à savoir que le RdP n'exige pas une énumération exhaustive de l'espace d'état et peut donc représenter d'une manière finie et concise des systèmes dont l'espace d'état est infini ou très important. De plus leur utilisation pour la modélisation est intuitivement claire. Un algorithme de traduction structurelle d'un RdPH D-élémentaire un automate hybride a été présenté. Sa structure favorable sera exploitée pour la synthèse de contrôleur.

Le chapitre suivant sera consacré à la synthèse du contrôleur pour les systèmes dynamiques hybrides.

Chapitre 3

Systèmes Dynamiques Hybrides : Synthèse de contrôleur

Dans le chapitre précédent, nous avons présenté les différents outils de modélisation des systèmes à événements discrets et hybrides. Nous allons voir dans celui-ci-ci comment les exploiter pour synthétiser des systèmes de commande. La notion de synthèse de contrôleur que nous considérons ici a pour origine la théorie de Ramadge et Wonham. Cette théorie a été étendue dans plusieurs directions, en particulier pour la commande des systèmes temporisés et hybrides. Ce troisième chapitre de la thèse est consacré à cette théorie. Nous commençons par présenter la notion de la synthèse de contrôleur pour les systèmes à évènements discrets où seul l'aspect logique est pris en compte, ensuite nous verrons les systèmes temporisés puis les systèmes dynamiques hybrides.

I. Synthèse de contrôleur pour les systèmes à évènements discrets

La théorie de la commande par supervision des systèmes à événement discrets a été initiée par les travaux de Ramadge et Wonham (R&W) [RW87]. Ces derniers considèrent un système à évènements discrets modélisé à un niveau d'abstraction logique (non temporisé), et dont le comportement doit être modifié par une commande à retour d'état, dans le but de satisfaire certaines spécifications. Le procédé est considéré comme un SED qui évolue spontanément en générant des événements. Le schéma de commande par supervision est présenté dans la figure 3.1. Dans tout ce chapitre, nous parlerons de superviseur ainsi que cela est défini dans la théorie originelle. Par la suite pour notre travail nous parlerons plutôt de contrôleur pour rester proche d'une terminologie familière aux automaticiens.

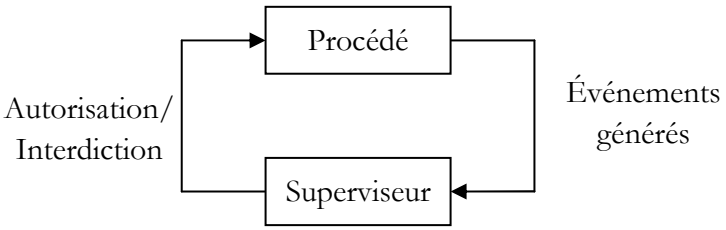


Figure 3.1. Schéma de supervision

Dans ce schéma, un procédé est couplé à un superviseur. Les entrées du superviseur sont les sorties du procédé. Le rôle de superviseur est *d'autoriser ou d'interdire* l'occurrence d'événements dans le procédé. Par ce rôle, le superviseur peut changer le fonctionnement du procédé. Etant donnés un procédé et un ensemble de spécifications logiques de fonctionnement, l'objectif de la théorie de R&W est de synthétiser un superviseur tel que le fonctionnement du procédé couplé au superviseur respecte les spécifications. De plus, on souhaite que le fonctionnement ainsi obtenu soit le plus *permissif* possible. La synthèse d'un tel superviseur est basée sur le concept de contrôlabilité.

Dans la figure 3.1, un superviseur unique est couplé au procédé. La supervision sera qualifiée de *centralisée*. En revanche, lorsque plusieurs superviseurs sont couplés à un même procédé, on parlera alors de supervision modulaire. Il existe d'autres architectures de commande telle que la commande hiérarchisée ou les commandes avec des systèmes partiellement contrôlable ou observable.

I.1 Concept de supervision

Supposons que le système à contrôler est modélisé par un automate P. Son comportement est donc donné par le langage $L(P)$. Le comportement de P peut s'avérer ne pas être entièrement satisfaisant dans le sens où il ne respecte pas certaines propriétés appelées *objectifs de contrôle* ou *spécifications*. Il est donc nécessaire de réduire ce comportement dans le but d'assurer ces spécifications. Cette restriction est réalisée par le biais d'un superviseur qui peut être vu comme une fonction qui, à partir de l'histoire du système va envoyer à celui-ci l'ensemble des événements qui doivent être interdits pour toujours respecter les spécifications. Contrôler un système consiste donc à lui ajouter des contraintes supplémentaires, induisant une réduction de son comportement à un comportement souhaité.

Un procédé couplé à un superviseur peut être perçu comme un système qui reçoit en entrée une liste d'événement interdits et génère en sortie l'ensemble des événements autorisés (qui est l'ensemble des évènements possible moins l'ensemble des évènements interdits). Dans la figure 3.2, le procédé P reçoit en entrée la liste d'événements interdits $\boldsymbol{\Phi}$ et en sortie donne l'ensemble des événements autorisés dans chaque état. L'ensemble des événements possibles dans l'état q de l'automate correspondant est présenté par $\sum(q)$. Ce procédé sera appelé *procédé supervisé* (S/P).

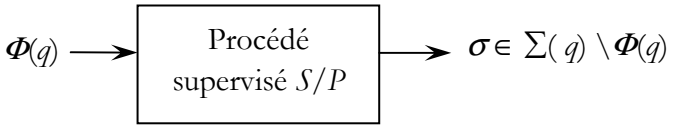


Figure 3.2. Procédé supervisé

Depuis un état q, le procédé supervisé S/P évolue de façon spontanée en produisant un événement $\sigma \in \Sigma(q) \setminus \Phi$, où $\Sigma(q) \setminus \Phi$ dénote l'ensemble des événements qui appartiennent à l'ensemble $\Sigma(q)$ et qui n'appartiennent pas à l'ensemble Φ. Cela signifie que le procédé supervisé peut générer un événement σ, si σ peut être généré par le procédé en isolation et si σ n'est pas interdit.

Nous pouvons remarquer qu'un procédé supervisé peut être défini de façon équivalente en spécifiant en entrée la liste des événements autorisés. Si Σ est l'alphabet des événements du procédé alors, la liste des événements autorisés correspond à l'ensemble $\Sigma \setminus \Phi$. La liste des événements interdits ou autorisés est fournie par le superviseur en fonction de l'ensemble des événements de sortie du système. Cette idée est présentée dans la figure 3.3 où l'indice i représente le temps logique. Dans cette figure, le procédé supervisé S/P est supposé être dans un état q. Depuis cet état, S/P peut générer à l'instant logique i+1, l'événement σ^{i+1}. Cet événement est un élément de l'ensemble de $\Sigma(q^i) \setminus \Phi^i$. Fondamentalement, l'observation du procédé par le superviseur est asynchrone. L'occurrence de σ^{i+1} conduit le superviseur dans un nouvel état. Immédiatement, la liste d'événements interdits Φ^{i+1} est alors fournie au procédé et ainsi de suite. On appellera *fonctionnement en boucle fermée*, le fonctionnement du procédé couplé à son superviseur.

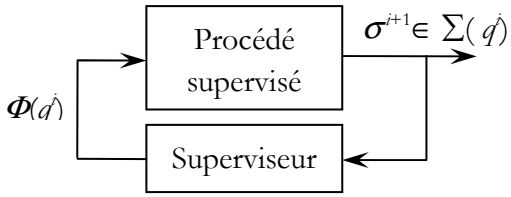


Figure 3.3. Schéma de supervision avec indice de temps

Remarque 3.1 : Le rôle du superviseur se cantonne à interdire l'occurrence d'événements dans le procédé. Il ne peut en aucun cas forcer des événements à se produire. Il s'ensuit donc que le superviseur ne peut que restreindre le fonctionnement du procédé.

❑

En pratique, certains événements générés par un procédé ne peuvent pas être interdits. Prenons l'exemple de la panne d'une machine, il paraît naturel que cet évènement ne puisse pas être interdit. Un tel événement sera appelé *événement incontrôlable*. Au contraire, on appellera *événement contrôlable* tout événement qui peut être interdit à n'importe quel moment.

De manière générale, si Σ est l'alphabet des événements d'un procédé, on peut définir la partition suivante : $\Sigma = \Sigma_C \cup \Sigma_U$, où Σ_C et Σ_U dénotent respectivement les ensembles d'événements contrôlables et incontrôlables sur Σ. Comme les événements incontrôlables ne

peuvent pas être interdits par la supervision, il est nécessaire d'exiger qu'une liste Φ d'événements interdits ne contienne aucun événement incontrôlable. Ainsi, dans la figure 3.3 nous aurons pour tout i : $\Phi(\dot{q}) \cap \Sigma_U = \varnothing$.

I.2 Définition d'un superviseur

Conformément à la figure 3.3, un superviseur peut être perçu comme une machine à états dont les sorties sont des listes $\Phi(\dot{q})$ d'événements interdits. On peut remarquer qu'entre deux occurrences successives d'événements par exemple, σ^i et σ^{i+1}, la sortie $\Phi(\dot{q})$ du superviseur reste inchangée. Ainsi, on peut représenter un superviseur par un modèle tel que la sortie ne dépende que de l'état. Un superviseur peut donc être modélisé par une machine de Moore.

Définition 3.1 (Superviseur) : Le superviseur S peut être défini par le 6-uplet $S = (V, \Sigma, \xi, v_0, 2^{\Sigma c}, \theta)$ où :

- V est un ensemble fini d'états ;
- Σ est l'alphabet d'entrée ;
- $\xi : V \times \Sigma \rightarrow V$ est la fonction de transition d'états ;
- v_0 est l'état initial ;
- $2^{\Sigma c}$ est l'alphabet de sortie ;
- $\theta : V \rightarrow 2^{\Sigma c}$ est la fonction d'affectation de sortie.

❑

Le superviseur S est donc une machine à états déterministe qui évolue conformément à une modification de son entrée (sur l'occurrence d'un événement de Σ généré par le procédé) et qui change d'état selon ξ. Pour chaque état v, le superviseur S fournit en sortie une liste d'événements interdits $\Phi = \theta(v)$. Rappelons que seuls les événements contrôlables peuvent être interdits par la supervision. Ainsi, chaque sortie de S est un élément de $2^{\Sigma c}$, où $2^{\Sigma c}$ est l'ensemble de tous les sous-ensembles de Σ_c.

Nous présentons ci-dessous à travers un exemple la démarche de la synthèse de contrôleurs qui comporte les étapes suivantes : 1) modélisation du procédé physique, 2) modélisation des spécifications, et 3) calcul du fonctionnement désiré en boucle fermée.

Exemple 3.1 : Considérons l'exemple classique d'un système manufacturier composé de deux machines identiques : M_1 et M_2, et un stock entre ces deux machines. Conformément à la figure 3.4, les deux machines travaillent de façon indépendante, puisent des pièces brutes en amont et déposent les pièces usinées en aval.

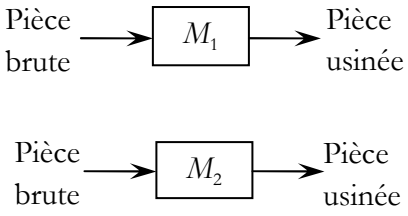


Figure 3.4 Système manufacturier ;

Le procédé est supposé évoluer de façon spontanée en générant des événements (*générateur d'événements*). Son fonctionnement peut être décrit par un ensemble de séquences d'événements qui constitue un langage formel sur l'alphabet des événements. Chaque machine de ce procédé peut être modélisée par un automate sans sorties (accepteur). Le graphe de transition d'états de l'automate des machines M_1 et M_2 est le même que celui de l'exemple qui est présenté au début du chapitre (figure 3.5).

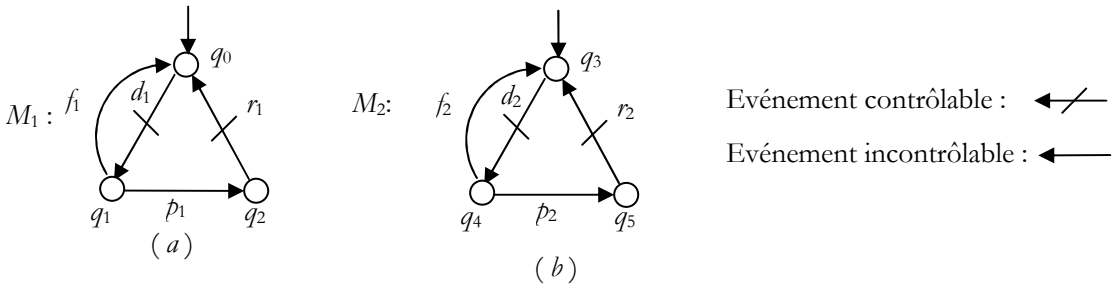


Figure 3.5. Automate à états finis modélisant le système manufacturier

Considérons la machine M_1 (figure 3.5.a). Dans son état initial (état q_0), la machine est à l'arrêt. L'événement d_1 modélise le début du cycle de la machine, l'occurrence de cet événement conduit la machine dans l'état de marche (état q_1). Dans notre exemple, nous supposons que d_1 est simultané avec la prise d'une pièce en amont. De même, la fin de cycle (événement f_1) est simultanée avec le dépôt d'une pièce en aval. Lorsque la machine M_1 est en marche (état q_1), l'occurrence de l'événement p_1 conduit la machine dans un état de panne (état q_2). Depuis cet état, la réparation de la machine, ramène la machine dans son état initial. La machine M_2 possède un fonctionnement similaire à M_1.

Notons respectivement Σ_1 et Σ_2, les alphabets des machines M_1 et M_2. Nous avons :

$$\Sigma_1 = \{d_1, f_1, p_1, r_1\} \quad \text{et} \quad \Sigma_2 = \{d_2, f_2, p_2, r_2\}$$

Le fonctionnement du système manufacturier est alors défini sur un alphabet $\Sigma = \Sigma_1 \cup \Sigma_2$. Nous avons pris la convention de représenter par un arc barré, toute transition associée à un événement contrôlable. Ainsi, nous supposons que $\Sigma_C = \{d_1, d_2, r_1, r_2\}$ et $\Sigma_U = \{f_1, f_2, p_1, p_2\}$.

Un modèle automate sans sortie de notre système manufacturier peut être obtenu en effectuant la composition synchrone des modèles M_1 et M_2. Le modèle P défini par $P = M_1 \|_s M_2$ est représenté par son graphe de transition d'états dans la figure 3.6.

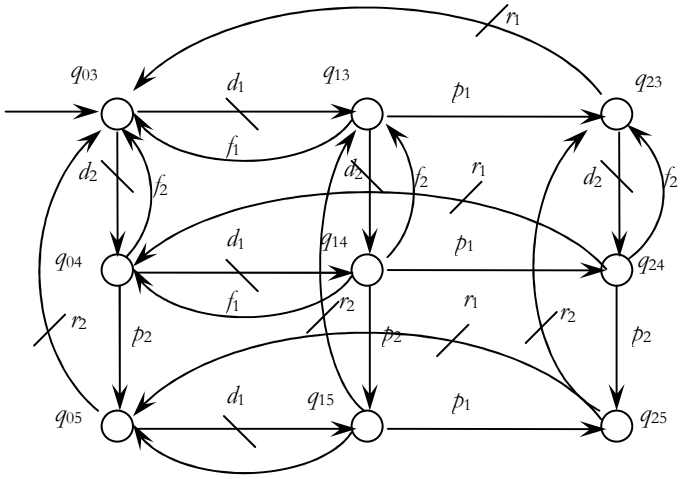


Figure 3.6. Composition synchrone des deux modèles de machines (modèle du procédé)

Dans l'automate P, un état est un couple (q_i, q_j), où q_i est un état de M_1 et q_j est un état de M_2. Cet état (q_i, q_j) est noté q_{ij} dans la figure 3.6

On considère la spécification de fonctionnement suivante pour notre système manufacturier : le fonctionnement doit respecter la présence d'un stock de capacité limitée à 1, situé entre les 2 machines. Nous supposons donc à présent que les machines travaillent en série. Conformément à la figure 3.7, une pièce doit visiter M_1 puis M_2. La présence du stock entre M_1 et M_2 impose que : (1) la machine M_2 ne peut commencer à travailler que si elle peut prendre une pièce dans le stock, c'est-à-dire, si le stock est plein, et (2) la machine M_1 ne peut déposer une pièce dans le stock que si celui-ci est vide. Le stock est supposé vide dans son état initial.

Le superviseur S de la figure 3.8 permet de garantir le respect de cette spécification de fonctionnement. Dans cet automate les états v_0 et v_1 correspondent respectivement aux états : "stock vide" et "stock plein".

Figure 3.7. Le système manufacturier sous contrainte de stock

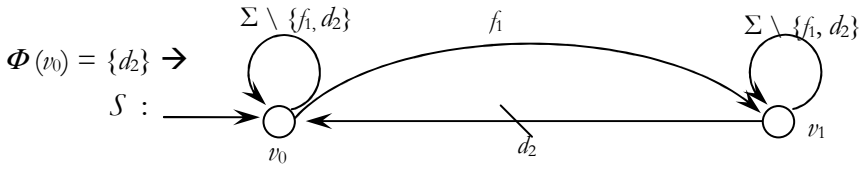


Figure 3.8. Le modèle automate de la spécification

Lorsque le stock est vide, l'occurrence de l'événement contrôlable d_2 est interdite (début du cycle de M_2). Sur l'occurrence de l'événement f_1 (fin du cycle de M_1 et dépôt d'une pièce dans le stock), l'automate S change d'état et passe dans l'état v_1. Dans cet état, l'occurrence de l'événement f_1 est interdite (fin du cycle de M_1).

On appelle *fonctionnement désiré en boucle fermée*, le fonctionnement du procédé couplé à sa spécification. Conformément à la figure 3.3, un événement peut être généré par le procédé supervisé si, il peut être généré par le procédé P en isolation et s'il est autorisé (non interdit) par le superviseur S. Par extension, une séquence d'événements ω est possible dans le fonctionnement en boucle fermée, si elle est possible dans le procédé en isolation ($\omega \in L(P)$), et si elle est autorisée par le superviseur ($\omega \in L(S)$). Si on note S/P la machine constituée du procédé P couplé à la spécification S, le langage $L(S/P)$ représente alors le fonctionnement en boucle fermée du système. Le langage $L(S/P)$ est simplement défini par :

$$L(S/P) = L(P) \cap L(S).$$

Le modèle automate reconnaissant $L(S/P)$ est obtenu en effectuant le composé synchrone de P et de S.

Définition 3.2 (langage du procédé en boucle fermée) : Le langage $L(S/P)$ généré par le procédé supervisé en boucle fermée est défini de manière récursive par :

- $\varepsilon \in L(S/P)$
- $[\omega\sigma \in L(S/P)] \Leftrightarrow [(\omega \in L(S/P)) \wedge (\sigma \in S(\omega)) \wedge (\omega\sigma \in L(P))]$

Où $\sigma \in S(\omega)$ signifie que l'occurrence d'événement σ après le mot ω ne doit pas être interdit par le superviseur.

❑

Un mot $\omega\sigma$ peut être généré par le procédé supervisé si le mot ω a été généré par le procédé supervisé et si l'événement σ est autorisé par le superviseur et le mot $\omega\sigma$ est accepté par le procédé non supervisé. Le mot vide ε est compris dans le langage $L(S/P)$.

Le modèle de fonctionnement désiré en boucle fermé du système supervisé est donné dans la figure 3.9.

I.3 Contrôlabilité

Ramadge et Wonham ont introduit la notion de contrôlabilité pour les SED afin de caractériser les langages supervisés d'un procédé. Etant donnés un procédé P et une spécification de fonctionnement S_{Spec}, on souhaite synthétiser un superviseur S de façon à ce que le système en boucle fermée S/P, respecte la spécification. Cela signifie qu'il faut déterminer le langage $L(P) \cap L(S_{spec})$. Ce langage appelé *fonctionnement désiré* correspond à l'ensemble des séquences qui peuvent être générées par le procédé et qui sont tolérées par la spécification, ce langage est noté L_D. Il n'est pas toujours possible (prise en compte d'événements incontrôlables Σ_U) de restreindre, par la supervision, le fonctionnement d'un procédé à n'importe quel sous-langage de ce

fonctionnement. L'existence d'un superviseur *S* tel que $L(S/P) = L_D$ réside dans le concept de contrôlabilité.

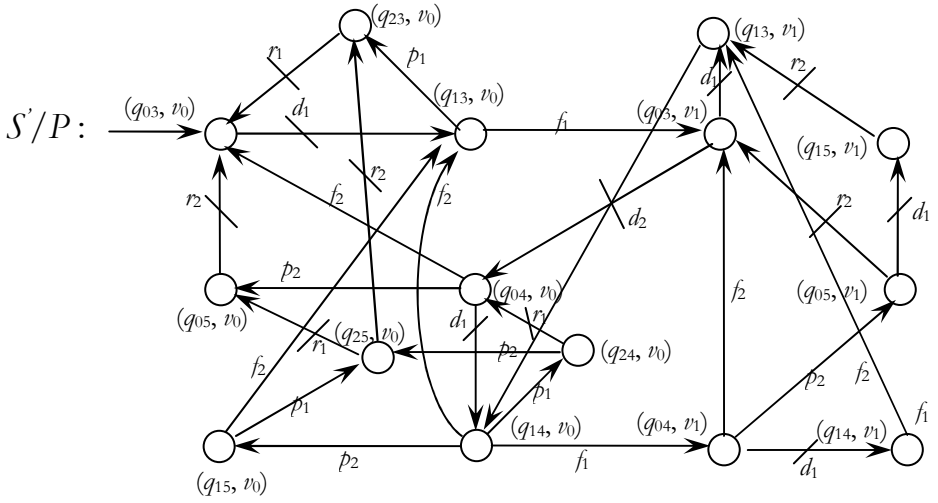


Figure 3.9. Modèle du fonctionnement désiré en boucle fermée

Définition 3.3 (contrôlabilité) : Un langage *K* est dit contrôlable par rapport à un langage *L(P)* si :

$$\overline{K\Sigma_U} \cap L(P) \subseteq \overline{K}$$

Où $\overline{K}$ représente le préfixe-clôture du langage de spécification et *L*(*P*) le langage du procédé.

❑

On peut dire que le langage de spécification *K* est contrôlable par rapport à un langage *L*(*P*) si pour toute chaîne ω de *K* et pour tout événement incontrôlable τ de Σ_U , la chaîne ωτ appartient à *L*(*P*), implique qu'elle appartient aussi à *K*.

La théorie de R&W permet de résoudre ce problème. Nous allons présenter directement l'algorithme de Kumar [Kum91] qui permet de déterminer le langage contrôlable maximal permissif. Pour présenter cet algorithme, il est d'abord nécessaire de définir quelques notions importantes.

Définition 3.4 (ensemble des états interdits) : Soit $P = (Q, \Sigma, \delta, q_0)$ et $S_{spec} = (V, \Sigma, \xi, v_0)$ les modèles automates du procédé et de la spécification. Par composition synchrone des deux modèles, L'ensemble des *états interdits* Q_I sera défini comme ci-dessous :

$$M_I = \{(q, v) / \ \exists \sigma \in \Sigma_U \text{ avec } \delta(q, \sigma) \text{ défini, et } \xi(v, \sigma) \text{ non défini}\}$$

❑

Il y a un autre type d'état interdit qu'il faut ajouter à l'ensemble défini ci-dessus : ce sont les états *faiblement interdits* :

Définition 3.5 : Soit M_{PS} l'ensemble des états possibles et autorisés par la spécification et M_I l'ensemble des états interdits. L'ensemble des *états faiblement interdits* M_F sera défini comme ci-dessous :

$$M_F = \{q_i \mid q_i \in M_{PS}, q_j \in M_I \text{ ou } q_j \in M_F \text{ et } \sigma \in \Sigma_U, q_i \xrightarrow{\sigma} q_j$$

❑

1.4 Algorithme de Kumar

A partir des modèles automates P d'un procédé et S_{spec} d'une spécification de fonctionnement, l'algorithme de Kumar permet de vérifier la contrôlabilité du langage de spécification $L(S_{spec})$. De plus, dans le cas où le langage $L(S_{spec})$ n'est pas contrôlable, cet algorithme permet de synthétiser un modèle automate du langage suprême contrôlable du fonctionnement désiré $supC(L_D)$ (figure 3.10).

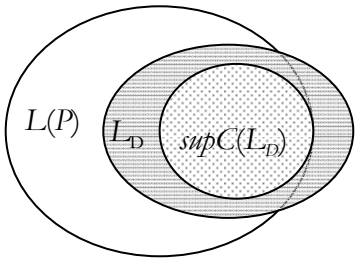


Figure. 3.10. Langage suprême contrôlable d'un fonctionnement désiré

Algorithme de Kumar : Soit $P = (Q, \Sigma, \delta, q_0)$ et $S_{spec} = (V, \Sigma, \xi, v_0)$ les modèles automates du procédé et de la spécification. L'algorithme est basé sur les 4 pas suivants :

Pas 1. On construit le composé synchrone D de P et de S_{spec}, c'est-à-dire, $D = P \| s\ S_{spec}$. Le langage $L(D)$ sera noté L_D.

Pas 2. On détermine l'ensemble des *états interdits*.

Pas 3. On détermine l'ensemble des *états faiblement interdits*.

Pas 4. On supprime de D l'ensemble des états interdits ainsi que l'ensemble des états faiblement interdits (ainsi que les transitions associées à ces états). On supprime de D l'ensemble des états non accessibles, c'est-à-dire, tout état (q, v) tel qu'il n'existe pas de chemin permettant de rejoindre (q, v) depuis l'état initial.

❑

La figure 3.11 donne l'automate final et l'ensemble des états interdits du système manufacturier. Par application de l'algorithme de Kumar sur cet exemple, nous trouvons les états interdits suivants : {(q_{13},v_1), (q_{14},v_1), (q_{15},v_1) }. Dans cet exemple il n'y a pas d'états faiblement interdits.

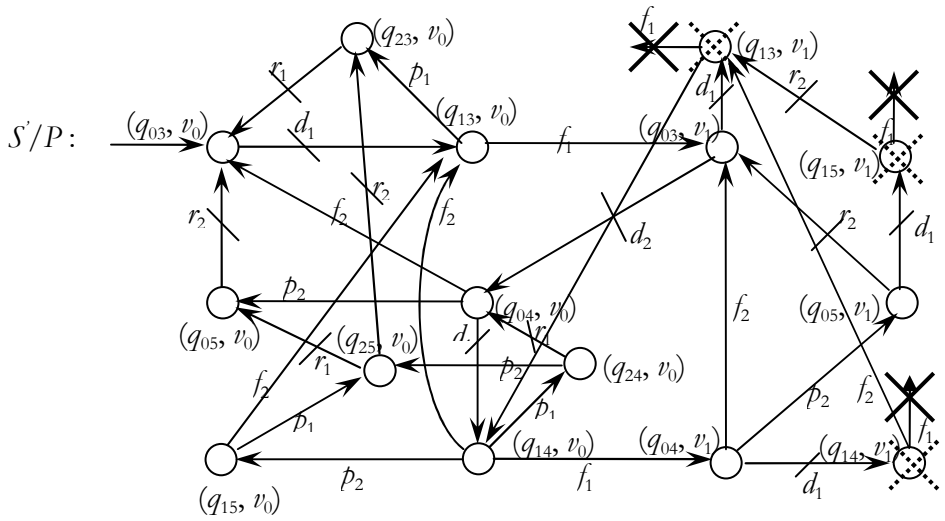


Figure. 3.11. Modèle automate du système supervisé avec des états interdits

Le modèle final de cet automate est présenté dans la figure 3.12.

Si le modèle du procédé comporte *n* états et le modèle de la spécification comporte *m* états, alors l'algorithme de Kumar permet de synthétiser un superviseur qui comporte au plus *n.m* états. Ainsi, la taille du superviseur est souvent bien plus grande que celle du procédé. Il résulte que dans bien des cas, l'explosion combinatoire due à l'utilisation de modèles automates rend difficile la synthèse de superviseurs. Un grand nombre d'extensions de la théorie de R&W visent à résoudre le problème de la réduction de la taille du superviseur [Did07].

La théorie de R&W constitue actuellement une activité importante au sein de plusieurs groupes de recherche sur le plan international. Cette théorie possède diverses extensions. Les travaux basés sur cette théorie sont très nombreux. L'approche classique utilise des automates à états finis et un point de vue centralisé. Un outil informatique "logiciel TCT" à été développé dans l'équipe du Professeur Wonham *«System Control Group»* à l'université de Toronto. En exploitant les concepts de la théorie R&W. Cet outil permet la synthèse de la commande par supervision des SED. Une fois la commande obtenue, une implantation est nécessaire. Là aussi, différents travaux proposent des interprétations sous forme de schémas *Ladder Diagram* ou de *Grafcet*.

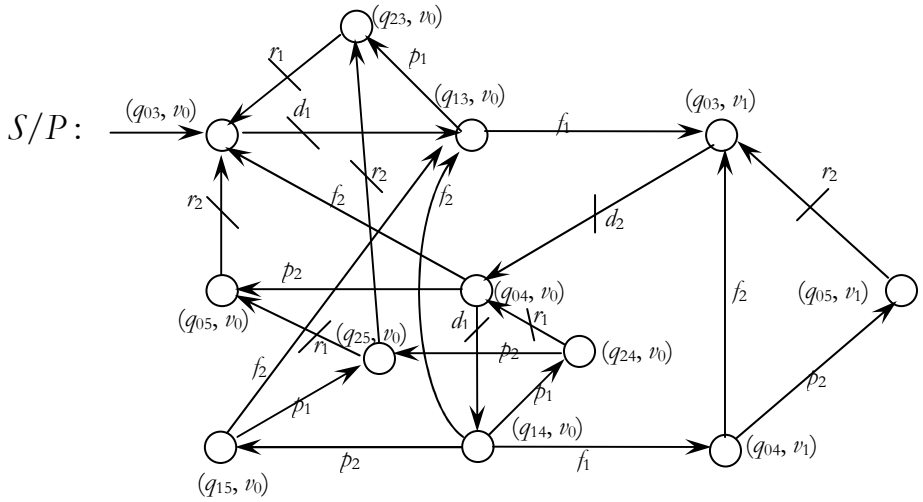


Figure. 3.12. Modèle final de système supervisé sans états interdits

Pour pallier aux problèmes d'explosion combinatoire du nombre d'états dans la modélisation par automates, de nombreux travaux utilisent les réseaux de Petri pour avoir des modèles concis. Ces approches sont en plein développement, elles ne sont pas développées dans cette thèse car elles ne sont pas concernées par notre travail.

La prise en compte du temps nous concerne directement. Nous considérons à nouveau la théorie RW par son extension pour introduire des contraintes temporelles [BW94][Sav02]. C'est l'objet de la prochaine section. Dorénavant, nous ne parlerons plus que de synthèse de contrôleurs.

II. Synthèse de contrôleur pour les systèmes à évènements discrets temporisés

La synthèse de contrôleur pour les systèmes à évènements discrets ne prend en considération que l'ordre d'occurrence des évènements, et agit sur le procédé en autorisant ou interdisant des évènements. La prise en charge explicite du temps, permet de synthétiser des lois de commande moins contraignante dans le sens où on peut restreindre les contraintes temporelles sur l'occurrence des évènements sans l'interdire totalement. Il est évident cependant, que la prise en compte du temps rend le modèle plus complexe et par conséquent la synthèse du contrôleur plus difficile.

Plusieurs travaux ont été élaborés et des approches basées sur différents outils de modélisation ont été proposées pour l'analyse et la synthèse de contrôleur des systèmes à évènements discrets temporisés. Ces travaux peuvent être classifiés en deux catégories, à savoir : la synthèse de contrôleur en temps discret et en temps continu.

II.1 Synthèse de contrôleur en temps discret

Dans [BW94] les auteurs ont proposé une approche pour la synthèse de contrôleur des systèmes à évènements discrets temporisés (SEDT) se basant sur les travaux de Ramadge et

Wonham, auxquels les contraintes temporelles sur les dates d'occurrence des évènements sont ajoutées. Brandin et Wonham ont eu pour objectif d'élargir la théorie classique de la commande supervisée pour les systèmes à événements discrets afin de l'appliquer à des systèmes temporisés. La théorie de Ramadge et Wonham repose sur la manipulation des langages. Pour se ramener à cette théorie, le temps est discrétisé et sa prise en compte va se réaliser par la création d'un nouvel événement, appelé *tick*, qui sera présent dans l'écriture du langage de l'automate temporisé. Le modèle temporisé est construit à partir d'un modèle logique, appelé *graphe d'activités*, représenté sous la forme d'un automate discret. Ce modèle décrit la succession logique des commutations entre les différents états du système. Les transitions du graphe sont étiquetées par des événements discrets. Le *graphe d'activités* est alors un système à événements discrets, non temporisé, représenté par un automate fini déterministe noté $A_{act} = (Q_{act}, \Sigma_{act}, \delta_{act}, q_{act,0})$. Comme pour les automates à états finis, chaque événement du procédé est instantané et il est exécuté à n'importe quel instant *t* du temps réel.

L'ajout du temps sur cette structure va permettre de définir un modèle d'automate temporisé. On suppose que le temps est mesuré à l'aide d'une horloge digitale qui incrémente un compteur de top d'horloge défini par :

$$tickcount : \mathbb{R}^+ \rightarrow \mathbb{N}, \text{ tel que } tickcount(t) = n \text{ lorsque } n \leq t < n+1$$

Lorsque un événement arrive à l'instant t, avec $n \leq t < n+1$, on considère dans le modèle qu'il est arrivé à l'instant $t = n$. Par conséquent, l'espace du temps est discret et la résolution temporelle pour la modélisation est d'un top d'horloge. Les contraintes temporelles sont spécifiées toujours en termes de top d'horloges.

A chaque événement $a \in \Sigma_{act}$ on associe un intervalle $[l_a, u_a]$, l_a *et* $\in \mathbb{N}$ et $u_a \in \mathbb{N} \cup \{\infty\}$. Un triplet (a, l_a, u_a) dénote un événement temporel. Les événements sont classés en deux catégories selon la valeur de la borne supérieure de l'intervalle associé.

- Un événement a est appelé *proche* si $0 \leq u_a < \infty$. L'ensemble des événements proches est noté Σ_{spe}.
- Un événement a est appelé *lointain* si $u_a = \infty$. L'ensemble des événements lointains est noté Σ_{rem}.

Par conséquent, l'ensemble des événements Σ_{act} est partitionné en deux sous-ensembles disjoints :

$$\Sigma_{act} = \Sigma_{spe} \cup \Sigma_{rem}$$

A chaque événement a on associe une temporisation t_a, qui mesure le temps écoulé depuis sa dernière validation. Pour modéliser les contraintes temporelles dans le modèle du comportement d'un SED on utilise un nouvel automate $A = (Q, \Sigma, \delta, q_0)$ dérivé de l'automate A_{act}. Cet automate est défini de la façon suivante :

- Q est l'ensemble des états pour lequel :

$$t_a = \begin{cases} [0\ u_a]\ si\ a \in \Sigma_{spe} \\ [0\ l_a]\ si\ a \in \Sigma_{rem} \end{cases}$$

Chaque état $q \in Q$ mémorise un état logique $q_{act} \in Q_{act}$ ainsi que la valeur de la temporisation t_a associée à chaque événement $a \in \Sigma_{act}$.

- Σ est l'ensemble des événements. Le passage du temps est modélisé par l'occurrence d'un évènement particulier. Cet événement, noté *tick*, modélise l'occurrence d'un top d'horloge.

$$\Sigma = \Sigma_{act} \cup \{tick\}$$

- δ est la fonction de transition. Un évènement a peut être exécuté depuis un état q dans A, si a peut être exécuté depuis un état q_{act} dans Q_{act} et la contrainte temporelle associée `à l'occurrence de a est vérifiée.
- q_0 est l'´etat initial

Un évènement a est dit *autorisé* depuis un état q s'il peut se produire dans l'automate non-temporisé (dans le modèle logique). Il devient *admissible* ou *éligible*, si son occurrence est possible à la fois sur un plan logique et sur le plan temporel. Un évènement autorisé mais non admissible est *en attente*. La prise en compte du temps enrichit le modèle étudié, mais en contre-partie, augmente sa complexité.

Exemple 3.2 : Considérons un SED qui a un seul état logique. Supposons que les évènements qui peuvent se produire dans ce système sont (a, 1, 1) et (b, 2, 3). Le comportement logique de ce système est modélisé par l'automate A_{act} en figure 3.13.

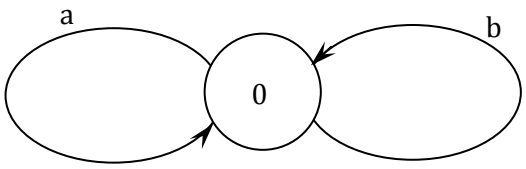


Figure 3.13. Automate A_{act}

Le comportement temporel du système est modélisé par l'automate $A = (Q, \Sigma, \delta, q_0)$ où :

- $Q = \{0] \times [0, 1] \times [0, 3]$;
- $\Sigma = \{a, b, tick]$;
- δ est la fonction de transition ;
- $q_0 = \{0\}$

Cet automate est représenté dans la figure 3.14. Chacun de ces états est caractérisé par une valeur particulière des temporisations associées aux évènements. Il faut noter que la prise en compte explicite du passage du temps a engendré l'augmentation du nombre d'états. Ainsi, cet

automate a 8 états et 11 transitions, contrairement à l'automate modélisant le comportement logique du système qui a seulement un état et deux transitions.

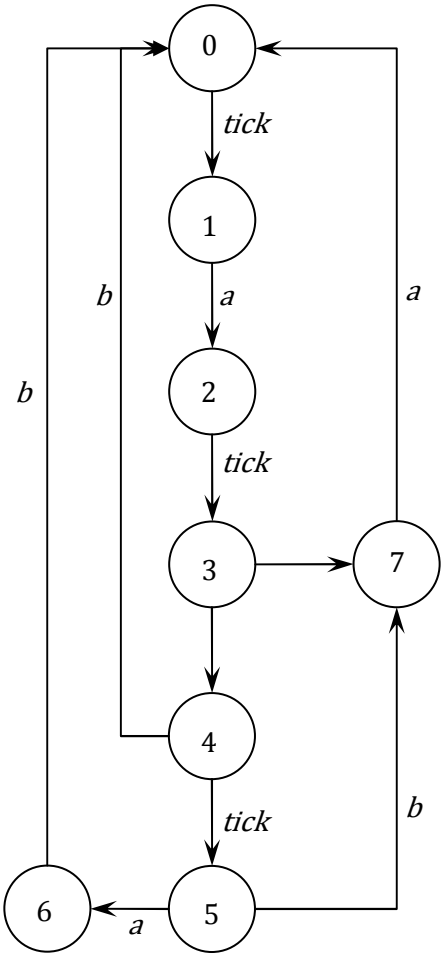


Figure 3.14. Automate A ;

❑

De la même manière que pour les SED, certains évènements sont contrôlables tandis que d'autres ne le sont pas. Ainsi, l'ensemble des évènements Σ est partagé en trois sous ensembles disjoints :

$$\Sigma = \Sigma_C \cup \Sigma_U \cup \{tick\}$$

où :

- Σ_C est l'ensemble des évènements contrôlables ;
- Σ_u est l'ensemble des évènements incontrôlables ;

Dans cette approche, un événement contrôlable peut être interdit indéfiniment, seuls les évènements lointains peuvent être contrôlables $\Sigma_C \subseteq \Sigma_{rem}$;

Par opposition, les évènements prévus ont des dates d'occurrence au plus tard, au delà de laquelle ils ne peuvent plus être interdits. Ainsi, les évènements prévus sont incontrôlables $\Sigma_{spe} \subseteq \Sigma_u$. Certains évènements lointains peuvent être incontrôlables par leur nature. Ainsi, l'ensemble des évènements incontrôlables est défini par :

$$\Sigma_u \subseteq \Sigma_{spe} \cup (\Sigma_{spe} - \Sigma_c)$$

Une autre catégorie d'évènements, essentielle dans la synthèse de contrôleur des SEDT, est représentée par les évènements forçables, un évènement est considéré comme étant forçable s'il peut se produire spontanément ou être forcé par un système extérieur tout en respectant la contrainte temporelle associée à sa date d'occurrence. Dans l'approche de Brandin et Wonham, un événement forçable peut préempter l'occurrence de l'événement *tick*. Ainsi le superviseur peut forcer l'occurrence d'un événement avant que l'horloge atteigne une certaine valeur.

L'ensemble des évènements forçables est noté Σ_{for}. Il n'y a aucune relation entre l'ensemble des évènements forçables et les ensembles des évènements contrôlables et incontrôlables. Un événement contrôlable peut ne pas être forçable dans le sens ou on peut interdire son occurrence, mais on ne peut pas forcer son exécution. De même, certains évènements incontrôlables peuvent être forçables. Intuitivement, le passage du temps ne peut pas être forcé ($tick \notin \Sigma_{for}$), mais il peut être préempté.

L'approche globale de la synthèse de contrôleur en temps discret consiste à modéliser le comportement temporisé d'un procédé avec un automate $P = (Q, \Sigma, \delta, q_0)$ qui génère un langage L(P). A tout mot $\omega \in$ L(P), il existe un état $q \in Q$ atteint par l'exécution du mot ω depuis l'état initial q_0. Les possibilités d'évolution du procédé depuis cet état sont décrites par l'ensemble des évènements éligibles dans l'état q.

A chaque mot $\omega \in L(P)$ on associe un ensemble d'évènements éligibles : $Elig_P(\omega)$, défini par :

$$Elig_P(\omega) = \{a \in \Sigma \mid a\,\omega \in L(P)\}$$

Formellement, un contrôleur est défini par une fonction

$S(\omega) : L(P) \rightarrow 2^{\Sigma}$

Tel que $\forall \omega \in$ L(P) :

$$S(\omega) \cap Elig_P(\omega) \neq \varnothing$$

$$S(\omega) \supseteq \begin{cases} \Sigma_u \cup \{tick\} si\ S(\omega) \cap \Sigma_{for} = \varnothing \\ \Sigma_u \qquad\qquad si\ S(\omega) \cap \Sigma_{for} \neq \varnothing \end{cases}$$

De la même manière que dans la théorie de la commande par supervision des SED, les évènements incontrôlables sont toujours autorisés par le superviseur. Par contre, lorsque parmi

les évènements éligibles il y a au moins un événement forçable, le contrôleur peut forcer son exécution avant l'occurrence d'un nouveau top d'horloge (événement tick). C'est pour cela que dans cette approche, le système de commande est appelé contrôleur.

Le comportement en boucle fermée, le concept de contrôlabilité d'un langage, ainsi que le calcul du langage suprême contrôlable sont identiques à ceux de la théorie de R&W. Il est ainsi possible de déterminer le contrôleur maximal permissif.

Même si on est ramené ici à l'approche classique RW par discrétisation du temps, les nombreuses définitions d'événements la rendent difficile à comprendre. Les notions d'événements proches, lointains, forçables, contrôlables, incontrôlables sont difficiles à classer. Du point de vue de l'automaticien et de manière plus simple, on peut distinguer deux types d'événements comme dans la théorie classique :

1) les événements contrôlables pour lesquels on peut modifier la date d'occurrence dans leur intervalle d'existence (quelque soient ses bornes). Cela peut correspondre par exemple au démarrage d'une machine qui doit avoir lieu dans l'intervalle [a, b]. Le contrôle peut alors avoir à réduire cet intervalle pour satisfaire une spécification. Un événement contrôlable peut être forcé.
2) les événements incontrôlables pour lesquels on ne peut rien modifier. Cela peut correspondre par exemple à un intervalle d'occurrence de la fin de fabrication d'une pièce et traduit alors une incertitude sur cette fin de tâche.

C'est cette notion d'événement que nous retiendrons dans notre travail.

Quoique la solution donnée par cette approche au problème de synthèse de contrôleur soit intéressante, néanmoins, elle a un inconvénient majeur au niveau de la modélisation. En effet la nature discrète du temps engendre l'explosion combinatoire du nombre d'états du modèle. De plus, cet aspect discret du temps est une approximation dans la modélisation du système. La solution en temps continu s'impose alors d'elle-même. Nous allons la développer ci-dessous et nous verrons que cela entraîne une complexité certaine de la méthode de synthèse.

II.2 Synthèse de contrôleur en temps continu

Plusieurs approches de synthèse de la commande ont été proposées pour pallier au problème de l'explosion combinatoire du nombre d'états engendrée par la nature discrète du temps. Ces approches, basées principalement sur l'outil automate temporisé, considère que le temps évolue d'une manière continue. Nous présentons dans la suite quelques une de ces approches.

1. Dans [Gou99] l'auteur propose une approche pour la synthèse de contrôleur en s'appuyant sur l'outil automate temporisé. La notion clé dans cette approche est la suppression du temps. En effet, son principe de base est de supprimer le temps dans l'automate temporisé modélisant le procédé en boucle fermée (composition synchrone de l'automate temporisé modélisant le procédé et de l'automate temporisé modélisant la spécification). Avant d'appliquer la théorie de R&W classique (pour les systèmes à

événements discrets logiques), la suppression consiste à transformer l'automate temporisé en un automate de régions [Yov98]. Dans un automate de régions l'information temporelle est incluse dans les sommets et non dans les transitions. Par conséquent, l'automate de régions peut être considéré comme un automate non temporisé, d'où le lien avec la théorie de base.

2. Dans [AGP+99] les auteurs proposent l'utilisation des réseaux de Petri à retard pour la modélisation du processus à commander, La synthèse du contrôleur est basée sur l'outil automate temporisé à retard. Les RdP à retard ont la structure discrète des RdP de base, tandis que l'information temporelle est modélisée de la même manière que dans les automates temporisés. Un RdP à retard est un RdP muni d'horloges et des contraintes temporelles sur le franchissement de transitions. Chaque transition modélise l'occurrence d'un événement. Ainsi, les transitions sont classifiées en contrôlables ou incontrôlables selon la nature de l'événement modélisé. Comme pour les automates temporisés, à chaque transition d'un RdP à retard est associée une condition de franchissement et une affectation. Les RdP à retard évoluent de la même façon qu'un automate temporisé, sauf qu'ils héritent de la clarté de modélisation des RdP classiques. Ainsi ils peuvent explicitement représenter le parallélisme, la synchronisation, le partage de ressources, …etc.
   Après avoir modélisé le procédé en boucle ouverte par un RdP à retard, les auteurs procèdent en une traduction de ce dernier en un automate temporisé à retard. Un automate temporisé à retard est dérivé du modèle automate temporisé en remplaçant les invariants des sommets par des conditions de franchissement au plus tard, associées aux transitions.
   La spécification imposée au fonctionnement du procédé est exprimée par une propriété qui doit être satisfaite par les états de l'automate temporisé à retard correspondant au procédé. Soit un automate temporisé à retard A et une propriété S, l'objectif de la synthèse d'un superviseur est de construire un automate à retard A_S tel que tous ses états vérifient la propriété S.
   Soit Q le sous-ensemble des états de l'automate A qui satisfont la propriété S, la procédure de synthèse de contrôleur consiste à déterminer l'ensemble $\pi(Q)$ des états à partir desquels on peut atteindre n'importe quel état $q \in Q$ par franchissement d'une transition contrôlable. On enlève de l'ensemble $\pi(Q)$ les états à partir desquels on peut évoluer par franchissement d'une transition incontrôlable vers un état qui n'appartient pas au sous-ensemble Q.
   La procédure de calcul est itérative. Elle est initialisée avec $Q(0) = Q$. A chaque itération *i* on calcule l'ensemble $\pi(q(i))$ et on réactualise $Q(i+1) = Q(i) \cap \pi(q(i))$. L'algorithme s'arrête lorsque $Q(i+1) = Q(i)$. L'ensemble obtenu est noté Q*. L'automate A_S a la même structure discrète que l'automate A sauf pour les gardes des arcs contrôlables qui sont modifiées de telle façon que seuls les états $q \in Q^*$ sont atteignables.
   Cette approche paraît séduisante, elle est cependant difficile à mettre en application dans des cas concrets que l'on peut rencontrer en automatique. La plupart des exemples présentés sont de taille réduite et faciles à résoudre à la main.

3. Dans [Kou99] les auteurs proposent l'utilisation d'un modèle RdP à arcs temporels pour la modélisation du procédé et des spécifications. La synthèse de contrôleur est faite sur l'outil automate à temps continu.
   L'outil RdP à arcs temporels est dérivé du modèle RdP autonome en lui associant des intervalles temporels aux arcs de sortie des places. Ces intervalles modélisent les contraintes temporelles qui interviennent dans le fonctionnement du système. L'occurrence des évènements dans le procédé est modélisée par le franchissement des transitions. Les contraintes temporelles sur la date d'occurrence d'un événement sont représentées par les intervalles temporels associés aux arcs d'entrées dans la transition correspondante. Une transition dans un RdP à arcs temporels peut être contrôlable ou incontrôlable selon la nature de l'événement modélisé.
   Le modèle du comportement désiré du procédé est obtenu en effectuant le produit synchrone du RdP à arcs temporels qui modélise le procédé en boucle ouverte et celui modélisant la spécification. L'arc temporel n'apporte ici qu'une modélisation plus concise par rapport aux modèles RdP prenant en compte le temps. La synthèse du contrôleur se fait toujours en passant par l'automate, ainsi le modèle global est ensuite traduit en automate à temps continu.
   La synthèse du contrôleur consiste à analyser un par un tout les sommets interdits et à restreindre les gardes des transitions contrôlables telles que ces sommets ne soient plus atteignables. Cette méthode est basée sur une technique de calcul du temps minimal et maximal de séjour du système dans un sommet de l'automate.

   Considérons la partie d'automate à temps continu présentée dans la figure 3.15

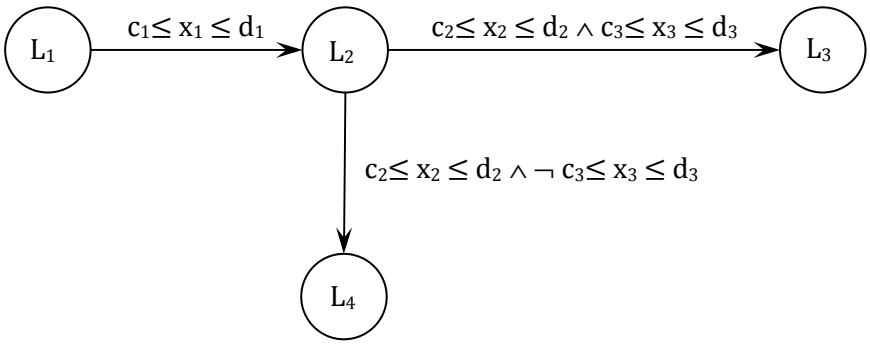


Figure 3.15. Partie d'automate à temps continu ;

Le sommet interdit L_4 est atteint à partir du sommet L_2. L'objectif de la synthèse du contrôleur est de garantir qu'à partir du sommet L_2, la transition vers L_3 est toujours franchie avant la transition vers le sommet interdit L_4. Par conséquent il faut déterminer les nouvelles valeurs pour les gardes des transitions contrôlables d'entrée et de sortie de L_2 qui garantissent que le temps maximal de séjour dans L_2 avant d'évoluer vers L_3 soit plus petit que le temps minimal de séjours dans L_2 avant d'évoluer vers L_4.

La technique de synthèse de contrôleur proposé par cette approche est basé sur deux procédures : La première consiste à modifier la garde de la transition de sortie de L_2 vers L_3 telle que cette transition soit franchie toujours avant la transition vers le sommet interdit L_4. Cette procédure appelée contrôle aval peut être appliquée seulement si la transition de L_2 vers L_3 est contrôlable. La deuxième procédure consiste à modifier la garde de la transition d'entrée de L_2 telle que la valeur des horloges dans ce sommet ne permettent pas de franchir la transition vers L_4. Cette procédure appelée contrôle amont peut être appliquée seulement pour les transitions d'entrée qui sont contrôlables. Lorsqu'une de ces procédures ne permet pas de trouver une solution, le sommet L_2 devient un sommet interdit. Par conséquent, cette approche cherche une solution locale pour éviter l'évolution du système vers un sommet interdit.

4. Dans [SA01] les auteurs se sont intéressés au problème de synthèse de contrôleur pour les SED temporisés. Leur approche est plus générale que celle proposée dans [Kou99], dans le sens ou elle n'est pas locale à un sommet. Le procédé en boucle ouverte ainsi que la spécification sont modélisés par un RdP T-temporel. Le fonctionnement désiré en boucle fermée est modélisé par la composition synchrone des modèles du procédé et de la spécification. Les auteurs ont également proposé un algorithme de traduction des RdP T-temporel en automate temporisé, afin d'exploiter ce dernier dans la phase de synthèse de contrôleur.
   Pour mieux expliquer l'approche de synthèse de contrôleur proposé dans [SA01], considérons la figure 3.16 ci-après.
   La transition $T_{n,q}$ qui mène vers le sommet interdit L_q est incontrôlable, ainsi il n'est pas possible d'agir sur sa date de franchissement. Ainsi reste deux possibilités pour forcer le franchissement de $T_{n,p}$ avant celui de $T_{n,q}$: soit déterminer une nouvelle garde pour la transition $T_{n,p}$, ce qui est possible seulement si cette transition est contrôlable. Soit déterminer les gardes des transitions contrôlables en amont du sommet L_n, telle que toutes les valeurs des horloges dans ce sommet valident $T_{n,p}$ avant $T_{n,q}$.

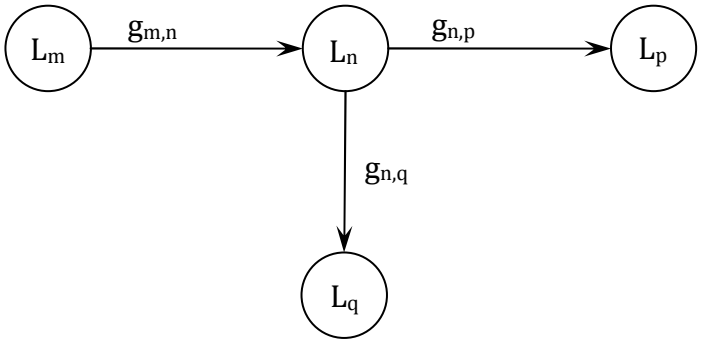


Figure 3.16. Partie d'automate temporisé ;

Le calcul des gardes des transitions contrôlables telles que la transition $T_{n,p}$ est toujours franchie avant $T_{n,q}$ est effectué comme suit :

Si $T_{n,p}$ est contrôlable, on modifie la garde de cette transition telle qu'elle soit toujours franchie avant $T_{n,q}$. Ensuite on calcule D_n le nouvel espace des horloges dans le sommet L_n, cette étape est dite traitement aval. On vérifie s'il est possible d'atteindre le sommet L_n avec les valeurs des horloges qui n'appartiennent pas à l'espace D_n. si tel est le cas, une seconde étape est entamée. Cette étape est dite traitement amont, consiste à remonter les branches de l'automate et à calculer de nouvelles gardes pour les transitions contrôlables. Il faut garantir que toutes les valeurs des horloges dans le sommet L_n appartiennent à l'espace des horloges désiré.

La modification des gardes de certaines transitions peut rendre certains sommets de l'automate temporisé non atteignables. Ainsi après avoir calculé les nouvelles gardes pour centaines transition contrôlables, une actualisation de l'automate est nécessaire afin de prendre en compte ces modifications, c'est la dernière étape de l'approche. Cette approche complète n'a pas été prouvée comme étant optimale, elle souffre en particulier de la perte de convexité des espaces atteignables.

III. Synthèse de contrôleur pour les systèmes hybrides

Contrairement aux systèmes à événements discrets avec ou sans la prise en compte du temps et pour lesquels la notion de la commande et les problèmes de synthèse associés sont bien identifiés et clairement définis, la commande des systèmes hybrides est une notion beaucoup plus large. En effet, le fait que ces systèmes fassent intervenir deux types de dynamiques, une dynamique continue et une autre événementielle, favorise la diversité des formulations du problème de la commande hybride rencontrées dans la littérature.

Parmi les méthodes qui traitent explicitement de la commande des systèmes hybrides, certaines accordent une importance plus grande à la partie discrète qu'à la partie continue. Les systèmes hybrides concernés sont généralement modélisés sous la forme d'automates hybrides. Le problème de commande correspondant est formulé sous la forme de recherche d'une stratégie discrète qui permet de restreindre la fonction de transition du système pour satisfaire les spécifications [ACH+95], [AM99].

1. Tittus et Egardt [TE98], étudient la synthèse de contrôleurs pour une classe de systèmes hybrides où la dynamique continue est décrite par des intégrateurs en utilisant le modèle de l'automate hybride linéaire. Bien que le modèle hybride soit très limité, il constitue un modèle de base pour la commande des procédés batch. La notion de contrôlabilité pour cette classe de système hybride est définie à partir de l'existence d'une loi de commande qui conduit le système vers des sous-ensembles prédéfinis dans l'espace d'état hybride. Une méthodologie pour analyser la contrôlabilité et synthétiser un contrôleur hybride pour un procédé est présentée.
   En se basant sur le modèle d'Antsaklis, dans [SAL96a] les auteurs modélisent le procédé et l'interface par une automate fini et ils l'utilisent pour l'analyse du système hybride. La notion de contrôlabilité est utilisée pour obtenir une méthode de synthèse de contrôleurs. En utilisant les invariants du procédé, ils proposent une méthode pour la conception du contrôleur [SAL96b].

2. D'autres approches s'intéressent davantage à la dynamique continue d'un système hybride. La classe des systèmes hybrides envisagée est celle des systèmes continus avec commutation de modèle. La commande recherchée est hybride (commande continue et contrôle discret) et les spécifications portent essentiellement sur la partie continue (régularisation autour de l'origine, optimisation d'un critère portant sur les variables continues, etc.). Parmi ces approches on trouve ainsi, des approches fondées sur la formulation et la résolution d'un problème de commande optimale, où le critère à minimiser porte uniquement sur les variables continues, et des approches fondées sur la théorie de Lyapunov.
   Une première approche générale de la commande optimale des SDH est proposée par Branicky dans [BM95]. Sur la base de son modèle unifié, dans [Bra98], Branicky présente des outils d'analyse appliqués aux systèmes à commutations et hybrides. Particulièrement, l'auteur introduit "les fonctions multiples de Lyapunov" comme un outil d'analyse pour la stabilité des systèmes à commutation. L'idée est que même si à chaque système individuel est associée une fonction Lyapunov, il faut imposer des restrictions sur la commutation pour garantir la stabilité.

3. D'autres approches s'intéressent davantage à la dynamique continue d'un système Dans [LGS96], les auteurs présentent une méthode pour la conception des contrôleurs de systèmes multi-agents basée sur la commande optimale et la théorie du jeu. Le système hybride est vu comme un jeu entre deux joueurs, le contrôleur et la perturbation. Les deux joueurs concourent sur des fonctions de coût, lesquelles ont un rapport avec les propriétés que le système en boucle fermée doit satisfaire. Le contrôleur gagne le jeu s'il peut tenir le système dans un fonctionnement "sûr" en présence de perturbations. Dans [TPS98], cette approche est utilisée pour la gestion du trafic aérien, et dans [LGS98] pour la commande.

IV. Conclusion

Dans ce troisième chapitre du mémoire, nous avons présenté les différentes approches pour concevoir un contrôleur. Nous avons commencé par la synthèse de contrôleur pour les SED sans prise en compte du temps qui correspond aux travaux fondateurs du Professeur Wonham. Ceux-ci reposent sur une modélisation par automate et par l'analyse du langage reconnu. Des spécifications sont ajoutées pour contraindre le procédé physique à vérifier certaines propriétés. Le concept clé de la contrôlabilité permet de déterminer un contrôleur maximal permissif. Une première extension de cet outil a consisté à prendre en compte le temps, il est évident que dans ce cas on améliore la conception d'un contrôleur en ajoutant l'intervalle d'existence des événements. Mais en même temps, les procédures de synthèse sont plus complexes et font actuellement l'objet d'une recherche active pour proposer des solutions implémentables. Cela n'a pas empêché les chercheurs d'aller au-delà de la prise en compte du temps en considérant des dynamiques de plus en plus complexes. Cela va des SED où l'on ajoute des dynamiques du type $\dot{x} = k$, aux systèmes continus auxquels on ajoute des commutations. Pour notre part, on s'intéresse à la sous-classe des systèmes hybrides positifs décrits par RdP hybrides D-élémentaires. Cette sous-classe est suffisamment générale pour décrire des applications les plus diverses comme les systèmes de

production, de transport ou de communication. Grâce à la modélisation par RdP, les modèles obtenus restent autant que possible proche du système physique ; et le passage aux automates hybrides permettra de faire une synthèse formelle du contrôleur. Ce travail constitue l'objet du quatrième chapitre.

Chapitre 4

Synthèse de contrôleur pour les systèmes à flux continu

Ce dernier chapitre de la thèse est consacré à l'approche de synthèse de contrôleur proposée pour les systèmes à flux continus, c'est une contribution importante de notre travail. Nous pouvons résumer cette approche comme suit : d'abord le procédé en boucle ouverte est modélisé par un RdP D-élémentaire que nous traduisons en automate hybride. Ce dernier à une structure particulière, en particulier les gardes des transitions sont fonction d'une seule variable, ceci facilitera la synthèse de contrôleur. Nous commençons par présenter l'approche formelle de synthèse de contrôleur au niveau d'un sommet. L'approche globale qui prend en considération tout l'automate hybride sera seulement esquissée.

I. Présentation intuitive de l'approche de la synthèse de contrôleur

La synthèse de contrôleur des systèmes dynamiques hybrides est un problème crucial qui est d'actualité. Bien que plusieurs contributions aient été apportées dans ce domaine, il en est encore à ses débuts et beaucoup reste à faire. Dans ce travail nous nous intéressons au problème de la synthèse de contrôleur des systèmes à flux continu. Ce sont des systèmes dynamiques hybrides, où les variables d'états continues sont positives et linéaires aux sens des automates hybrides, c'est.-à-dire qu'elles évoluent suivant des équations différentielles de la forme $\dot{x} = k$, où k est une constante rationnelle.

Dans l'approche de synthèse de contrôleur que nous proposons dans ce travail, nous commençons par considérer le procédé vu comme un système hybride. Nous considérons son comportement dynamique en boucle ouverte que nous voulons contraindre à respecter certaines spécifications. Nous ne considérerons dans ce travail que les spécifications agissant sur la dynamique continue du système, une perspective de ce travail consisterait à restreindre la dynamique événementielle. Le fonctionnement contraint portera sur les variables d'état continues et sera appelé fonctionnement désiré en boucle fermée. Pour restreindre l'espace d'état

atteignable, nous avons besoin de variables de commande qui seront nos points de contrôle, de telle sorte que le changement de valeur de ces variables va interdire l'espace d'état continu d'atteindre toute valeur indésirable.

Dans notre cas ces variables de contrôle serons les variables discrètes, plus précisément les dates d'occurrence des évènements contrôlables associés aux transitions discrètes. C'est une manière naturelle de contrôle d'un système hybride où c'est la partie discrète qui agit sur la partie continue. Il y a un couplage entre les dynamiques discrète et continue, ce couplage est formellement décrit dans le modèle RdP hybride D-élémentaire par la sémantique d'évolution de ce modèle. Il s'agit alors d'agir sur les éléments qui apparaissent dans la structure pour en limiter la dynamique et garantir le respect de spécifications, et cela de la manière la plus permissive.

Nous allons dans un premier temps présenter notre approche de manière intuitive à travers un exemple simple de système de production.

Exemple 4.1 :

Considérons un système producteur consommateur, composé d'une machine qui alimente un stock avec un taux de production de 20 pièces par minutes (figure 4.1.a). Le stock est utilisé pour satisfaire une demande de 13 pièces /minutes. L'arrêt et la marche de la machine sont effectifs après un délai de 2 minutes. Initialement, on suppose que le stock contient 50 pièces et la machine est en marche. La figure 4.1.b, ci-après montre le RdP hybride D-élémentaire modélisant le système du producteur consommateur. La présence ou l'absence de la marque dans la place P_2 autorise ou interdit l'alimentation du stock.

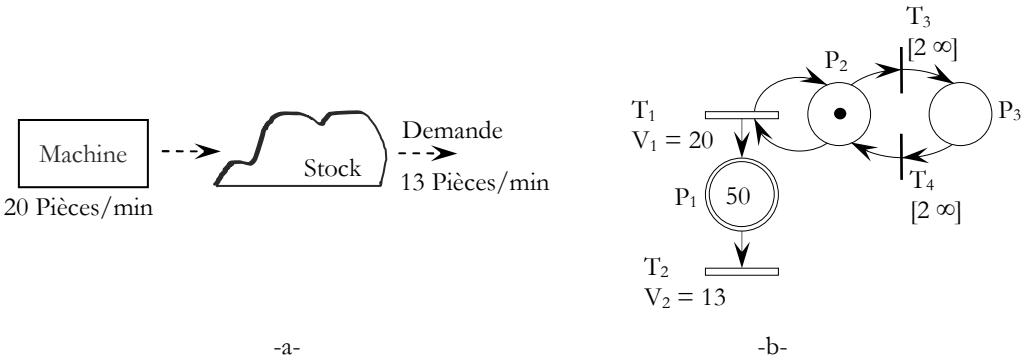


Figure 4.1.a Un système producteur consommateur ;
b) RdP D-élémentaire modélisant le système producteur consommateur

Dans cet exemple simple, il est évident que le nombre de pièces dans le stock peut être infini, *i.e.* le franchissement de la transition T_3 peut être infiniment retardé. Supposons que nous voulons que le niveau de stock ne dépasse jamais 100 pièces, ceci constitue la spécification. Dans le but de contrôler le niveau du stock pour satisfaire la spécification, nous devons agir sur les instants d'arrêt et de marche de la machine, *i.e.* calculer les intervalles de franchissement $[\alpha_3\ \beta_3]$ et $[\alpha_4\ \beta_4]$ des transitions T_3 et T_4 de telle sorte que le marquage de la C-place P_1 ne dépasse jamais 100. Même dans cet exemple simple il est difficile de calculer ces intervalles temporels

Il est aisé de constater que la spécification vient limiter l'espace d'état continu atteignable, représenté ici par m_1 le marquage de la C-place P_1. Il est donc nécessaire de pouvoir déterminer cet espace d'état pour calculer le contrôle qui permettra de le restreindre. Comme nous l'avons vu dans les chapitres précédents, le RdP hybride fournit un modèle élégant aisément compréhensible par l'utilisateur et concis dans son graphisme. Cependant, il n'est pas possible de caractériser de manière formelle son comportement dynamique. C'est pour cela que nous effectuons la traduction du RdP D-élémentaire en un automate hybride linéaire (figure 4.2.a). Il est ainsi possible de calculer l'espace d'état dynamique atteignable en utilisant des outils spécialement développés à cette fin. Dans ce travail, nous utilisons l'outil PHAVer (The Polyhedral Hybrid Automaton Verifier) [2]. C'est un outil de vérification des propriétés de sureté pour les systèmes dynamiques hybrides développé par la communauté informatique. Une annexe lui est consacrée à la fin de ce mémoire. Le principe du calcul de l'espace atteignable repose sur la détermination des successeurs continus et discrets partant d'une région initiale. Cet espace est donné sous la forme d'inégalités entre les différentes variables d'état continues. La convergence de l'espace et donc la décidabilité de l'algorithme est prouvée pour les automates temporisés, mais dans le cas des automates hybrides linéaires, la terminaison n'est pas garantie. Cependant en Automatique, les algorithmes de calcul terminent le plus souvent car les modèles possèdent souvent les propriétés suffisantes de convergence.

Nous ne présentons ici que l'utilisation de PHAVer pour calculer l'espace d'état atteignable de l'automate hybride modélisant le procédé en boucle ouverte. Dans le cas de l'automate hybride de la figure 4.2.a, l'utilisation de PHAVer pour le calcul de l'espace atteignable donne l'espace d'état suivant :

q_1 & (t_1 >= 0 & m_1 - 7*t_1 >= 0 | m_1 - 7*t_1 == 50 & t_1 >= 0 | t_1 == 0 & m_1 == 50),

q_2 & (t_1 >= 0 & m_1 + 13*t_1 >= 14 & m_1 >= 0 | t_1 >= 0 & m_1 + 13*t_1 >= 64 & m_1 >= 0),

q_3 & (m_1 == 0 & 13*t_1 >= 14 | m_1 == 0 & 13*t_1 >= 64)};

On retrouve bien des inégalités entre les deux variables continues m_1 et t_1. L'espace d'état atteignable est représenté sous la forme d'un automate déplié en figure 4.2.b. Chacun des sommets q_1, q_2 et q_3 est visité deux fois. Une visite de sommet est caractérisée par un espace d'état donné à l'entrée du sommet. Dans chaque sommet de l'automate nous avons indiqué l'espace atteignable qui est représenté ici par une portion du plan dans ce cas simple où il n'y a que deux variables continues.

Nous allons utiliser l'automate déplié dans notre stratégie de contrôle. Ainsi, notre but maintenant est d'introduire formellement la spécification, ce qui permettra de calculer les nouvelles gardes des transitions, tout en maintenant le contrôle maximal permissif.

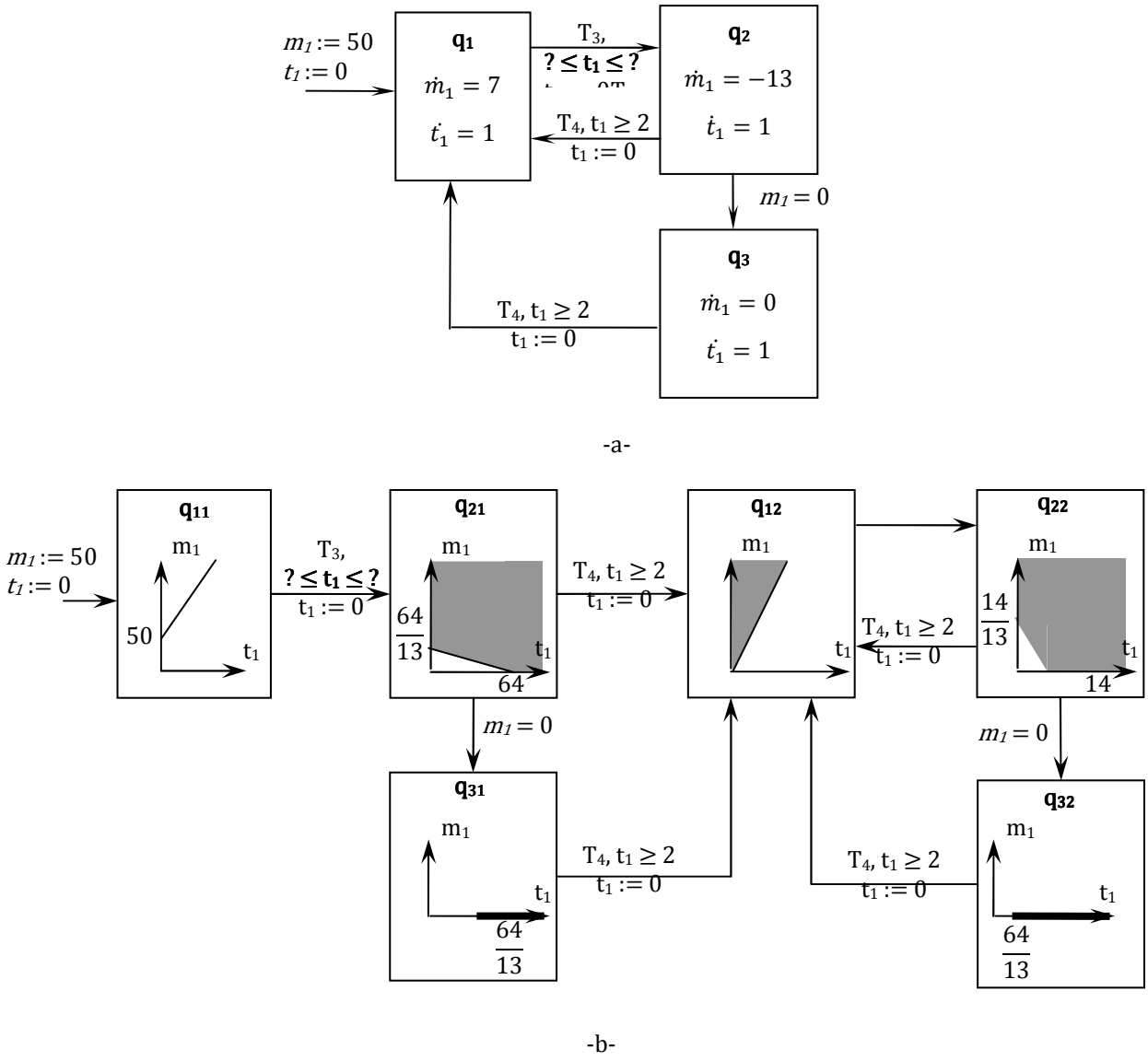


Figure 4.2.a. Automate hybride modélisant le système producteur consommateur
b. espace d'état atteignable par l'automate hybride

Considérons la spécification $m_1 \leq 100$, nous souhaitons garantir que tout espace atteignable dans un sommet vérifie cette contrainte. Il est évident que si la dynamique de m_1 dans un sommet est négative ou constante, et à condition que l'espace d'entrée dans ce sommet vérifie la spécification, alors l'espace d'état atteignable dans ce sommet ne comporte aucun état violant la spécification. Ceci est le cas des sommets q_2 et q_3 dans l'automate hybride de la figure 4.2.a. Par contre si la dynamique de m_1 est positive dans un sommet, comme c'est le cas du sommet q_1, alors les gardes des transitions de sortie des sommets q_{11} et q_{12} doivent être correctement modifiées pour pouvoir quitter le sommet en question avant d'attendre un état interdit.

Notre approche de synthèse de contrôleur est basée sur les trois étapes suivantes :

1. Modéliser le système en boucle ouverte (sans spécifications) par un RdP D-élémentaire [GA08] ;
2. Traduire le RdP D-élémentaires en automate hybride [GA08] ;

3. Modéliser les spécifications et calculer les nouvelles gardes des transitions qui assure le respect des spécifications [GA11] [GA'11].

Ces trois étapes sont résumées dans la figure 4.3 ci-après. Chaque bloc dans cette figure correspond à une étape. Nous allons expliquer chacune des trois étapes tout au long de ce chapitre. Il est évident que l'étape novatrice et la plus délicate est celle du pas 3. Le contrôleur final sera un automate temporisé qui restreindra la dynamique continue du système.

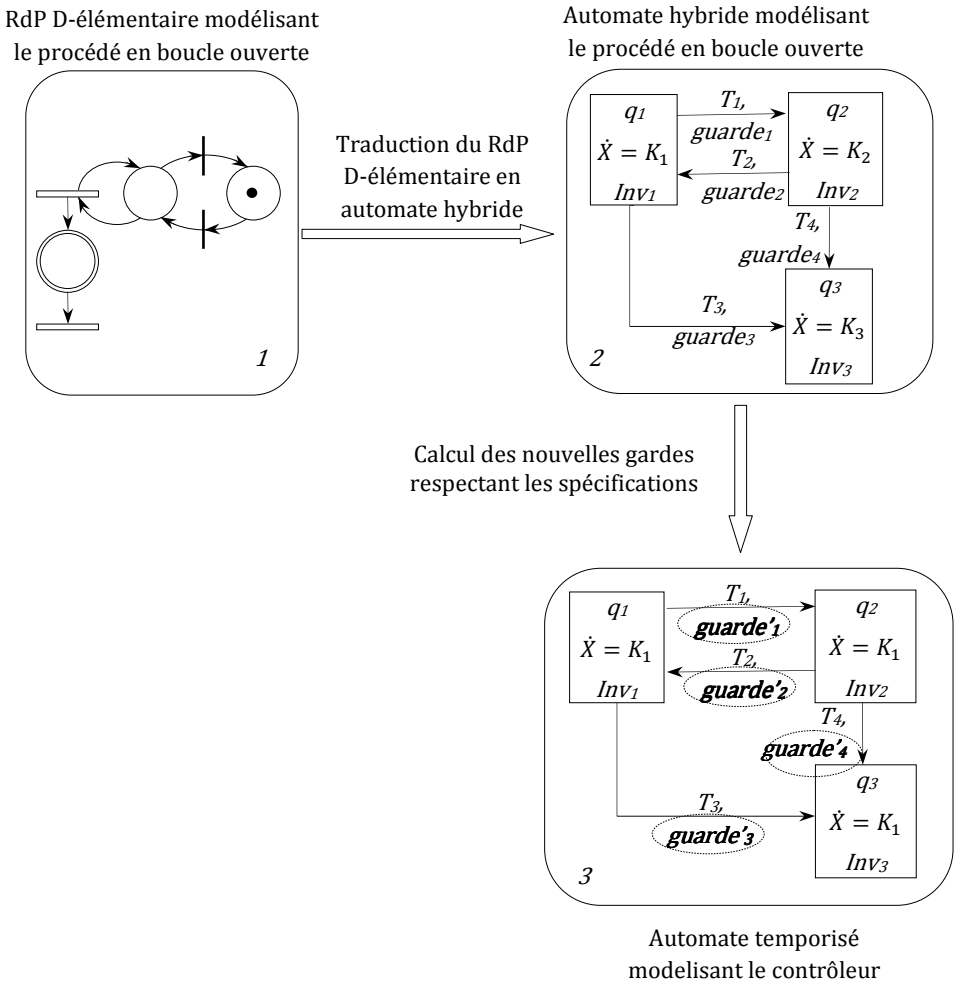


Figure 4.3. Principe de l'approche de synthèse de contrôleur

II. Modélisation du procédé en boucle ouverte

Les procédés que nous considérons ici font partie de la classe des systèmes à flux continu, qui sont des systèmes dynamiques hybrides dont les variables d'état sont linéaires et positives. La partie discrète a un comportement dynamique autonome, c'est-à-dire qu'elle évolue comme si elle était seule, mais elle commande la partie continue. Nous considérons ici que tous les événements de la partie discrète sont contrôlables. C'est une hypothèse tout à fait justifiée dans la mesure où nous considérons ici un système hybride piloté par la partie discrète. Imaginer une cascade d'événements incontrôlables est possible mais relève davantage d'une démarche intellectuelle que

de la prise en compte d'une réalité. Néanmoins cette extension est tout à fait possible et peut constituer une perspective à ce travail.

Hypothèse 4.1 : Les évènements de la partie discrète sont tous contrôlables.

❑

Le RdP hybride D-élémentaire est un modèle bien adapté à la modélisation des systèmes à flux continu. Les parties discrète et continue ainsi que l'interaction entre elles sont distinctement représentées. Comme décrit dans deuxième chapitre de ce mémoire, un RdP hybride D-élémentaire est la combinaison d'un RdP T-temporel modélisant la partie discrète du procédé et d'un RdP continu à vitesse constante modélisant la dynamique continue. Les transitions du RdP T-temporel sont toutes associées à des évènements contrôlables comme nous venons de le poser. C'est l'étape 1 de notre approche.

Le RdP hybride D-élémentaire est ensuite traduit en un automate hybride linéaire. Cette traduction décrite en détail dans la section IV de la première partie permet de combiner la puissance de modélisation des RdP D-élémentaires à la puissance d'analyse des automates hybrides. C'est l'étape 2 de notre approche.

Nous notons dans la suite par ***A*** l'automate résultant de la traduction d'un RdP D-élémentaire. Cet automate a certaines particularités. Il est défini comme suit :

Définition 4.1 (Automate *A*) : L'automate ***A*** résultant de la traduction d'un RdP hybride D-élémentaire est une structure $\boldsymbol{A} = (Q, X, \Sigma', \delta, F, Inv)$ tel que :

- $Q = \{q_1, q_2, \ldots\}$ est un ensemble fini de sommets ;
- $X = \begin{pmatrix} M_C \\ - \\ t_D \end{pmatrix}$ est le vecteur d'etat continu. X est la concaténation de deux vecteurs : $M_C = (m_1, m_2, \ldots, m_{nC})^T$ est le vecteur de n_C variables réelles qui correspondent au marquage des C-places ; et $t_D = \{t_1, t_2, \ldots, t_k\}$ est un vecteur de k horloges qui correspondent aux transitions validées.
- $\Sigma' = \Sigma \cup \Sigma_U$ est l'ensemble des évènements, où Σ est l'ensemble des évènements contrôlables, et Σ_U est l'ensemble des évènements incontrôlables ;
- δ est un ensemble fini de transitions, chaque transition est un quintuple $T = (q, a, g, \gamma, q')$ tel que :
  - $q \in Q$ est le sommet source ;
  - $a \in \Sigma$ est l'événement associé à la transition T ;
  - g est la garde de la transition T. Les gardes portent sur une seule variable continue. Les gardes des transitions contrôlables sont de la forme $(\alpha_i \leq t_i \leq \beta_i)$, les gardes des transitions incontrôlables sont de la forme $(m_i = 0)$.
  - γ est la fonction de réinitialisation qui affecte une expression linéaire aux horloges du vecteur t_D lors de franchissement de la transition T. La fonction γ ne change jamais les valeurs des variables réelles m_i du vecteur M_C.

- $q' \in Q$ est le sommet but ;

- F est une fonction qui affecte à chaque sommet une fonction d'évolution continue et linéaire. Quand l'automate séjourne dans le sommet q, les variables m_i de M_C évoluent suivant des équations différentielles de la forme $\dot{m}_i = B_i$, où B_i est le bilan dynamique de la C-place p_i. et les horloges $t_j \in T_D$ évoluent suivant des équations différentielles de la forme $\dot{t}_j = 1$.
- *Inv* est une fonction qui affecte à chaque sommet q, un prédicat linéaire $Inv(q)$, qui doit être vérifié par les variables continues pour pouvoir séjourner dans le sommet q.

❑

Les évènements issus de la partie discrète sont tous contrôlables, ils sont dans l'ensemble Σ dont la taille est égale au nombre de transitions discrètes. Ceux issues de la partie continue sont tous incontrôlables, ils sont dans l'ensemble Σ_U dont la taille est égale au nombre de places continues, ils n'apparaissent pas de manière explicite dans le modèle RdP hybride D-élémentaire car ils correspondent au marquage d'une C-place qui passe à 0. Dans le modèle automate hybride, ces évènements incontrôlables apparaissent de manière explicite.

Remarque 4.1:

Dans l'automate hybride linéaire $\boldsymbol{A}$, les gardes des transitions sont fonction d'une seule variable. Les transitions contrôlables correspondent au franchissement des transitions discrètes temporisées dans le RdP D-élémentaire. Elles ont des gardes de la forme : $\alpha_i \leq t_i \leq \beta_i$, où t_i est une horloge et a_i, β_i sont des constantes réelles. Les transitions incontrôlables correspondent au marquage des places continues qui s'annulent. Elles ont des gardes de la forme $m_i = 0$.

❑

Soit q un sommet de l'automate $\boldsymbol{A}$, Nous notons par $R_0(\boldsymbol{A}, q)$ l'espace d'état à l'entrée du sommet q de l'automate $\boldsymbol{A}$. C'est l'ensemble de tous les états à l'entrée de q. Et nous notons par $R(\boldsymbol{A}, q)$ l'espace d'état atteignable dans q de $\boldsymbol{A}$. C'est l'ensemble des états atteignables depuis $R_0(\boldsymbol{A}, q)$ par la fonction d'évolution continue $F(q)$ et qui satisfont $Inv(q)$. La figure 4.4 ci-après représente un exemple des espaces d'état $R_0(\boldsymbol{A}, q)$ et $R(\boldsymbol{A}, q)$ dans $\mathbb{R}^2$.

La démarche générale de la synthèse de contrôleur consiste, dans un premier temps à résoudre le problème de restriction de l'espace d'état, d'abord pour un seul sommet puis à itérer sur tous les états de l'automate $\boldsymbol{A}$. Nous allons résoudre formellement ce problème pour un sommet, puis nous donnerons une idée de l'algorithme général qui constitue une perspective immédiate de ce travail.

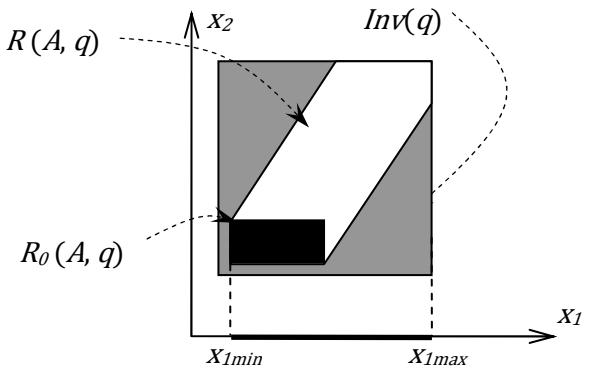


Figure 4.4. Espace d'état atteignable dans un sommet q

La figure 4.5 donne une représentation structurelle d'un sommet *q*. Nous supposons dans un premier temps que le sommet *q* a une seule transition T_j de sortie et que cette transition est contrôlable (Hypothèse 4.1). Comme précisé en remarque 4.1, la garde de la transition T_j est de la forme $\alpha_i \leq t_i \leq \beta_i$. La figure 4.5.b schématise R(***A***,*q*), l'espace d'état atteignable dans *q*,. Cet espace dépend: 1) de l'espace d'état à l'entrée de q $R_0(A, q)$, 2) de la fonction d'évolution dans le sommet *q* F(*q*), et 3) de l'invariant du sommet *Inv*(*q*). L'outil PHAVer nous permet de calculer R(*q*, ***A***), par un calcul d'analyse avant.

Ainsi arrivé à cette étape, nous disposons pour tous les sommets de l'ensemble des comportements dynamiques atteignables. Ces derniers sont exprimés sous la forme d'inégalités sur les variables continues, l'aspect discret est traduit par la commutation entre les sommets. Dans la section suivante, nous allons introduire la spécification en contraignant cet ensemble d'inégalités et nous allons étudier les conséquences de cette contrainte sur le comportement dynamique.

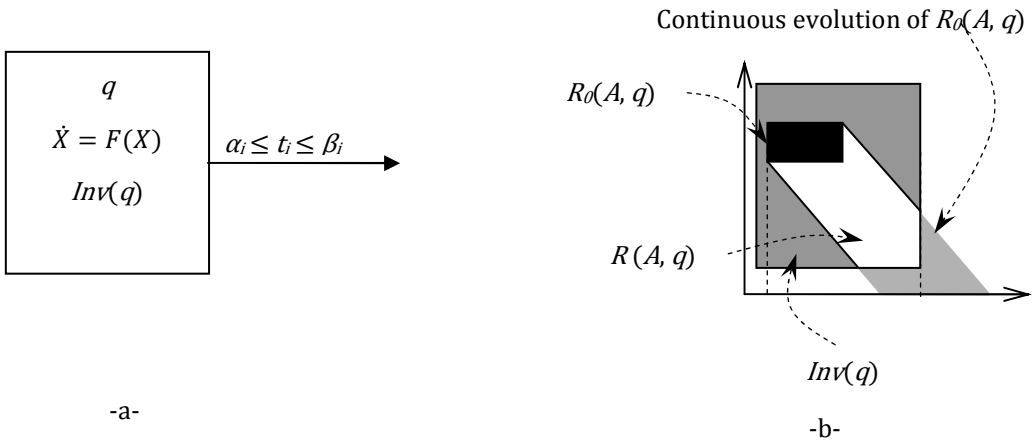


*Figure 4.5.a. un sommet q de l'automate **A**, b) L'espace d'état atteignable dans le sommet q*

III. Synthèse de contrôleur

La synthèse de contrôleur consiste à réaliser l'étape 3 de notre approche, décrite par le bloc 3 de la figure 4.3. Nous développons dans cette section la méthode du calcul des nouvelles gardes qui garantissent que l'espace d'état dans le sommet vérifie les spécifications. Ce travail sera présenté en détail au niveau d'un sommet.

III.1 Modélisation des spécifications

Dans un système dynamique hybride, les spécifications peuvent porter soit sur la partie discrète soit sur la partie continue. Nous ne considérons ici que les spécifications qui portent sur le comportement continu du système hybride. Cela signifie que les spécifications sont uniquement relatives au vecteur continu M_C, autrement dit, relatives aux variables d'état continues m_i et non pas aux horloges. Une spécification est de la forme d'une inégalité linéaire. C'est l'équation d'un hyperplan affine qui vient partager l'espace atteignable en deux régions.

Définition 4.2 (Spécification) : Soit $S^T = (s_1, s_2, \ldots, s_{nC})^T$ un vecteur réel constant de dimensions n_C (nous appelons dans la suite le vecteur S, vecteur des facteurs de spécifications) et b une constante réelle. Une spécification *Spec* sur le comportement continu de l'automate ***A***, est un prédicat linéaire de la forme.

$$S^T.M_C \leq b$$

Rappelons que M_C est le vecteur réel de dimension n_C et dont les composants m_i représentent les marquages des C-places. Nous notons par *Spec*(*q*) la forme générale d'une spécification. C'est la conjonction de toutes les spécifications imposées sur l'espace d'état du sommet *q*.

$$Spec(q) = Spec_1 \wedge Spec_2 \wedge \ldots \wedge Spec_L$$

❑

Une spécification continue Spec(q) est un ensemble de contraintes sur l'espace d'état continu atteignable par l'automate hybride dans le sommet *q*. L'automate hybride peut séjourner dans le sommet *q* uniquement si *Spec*(*q*) est satisfaite, et doit quitter *q* en franchissant une transition avant la violation de *Spec*(*q*).

Exemple 4.2 :

Considérons le sommet q_1 décrit dans la figure 4.6.a. Nous avons pris le cas de 2 variables continues et d'une horloge, d'un espace d'entrée non déterministe et d'une spécification sur les variables continues.

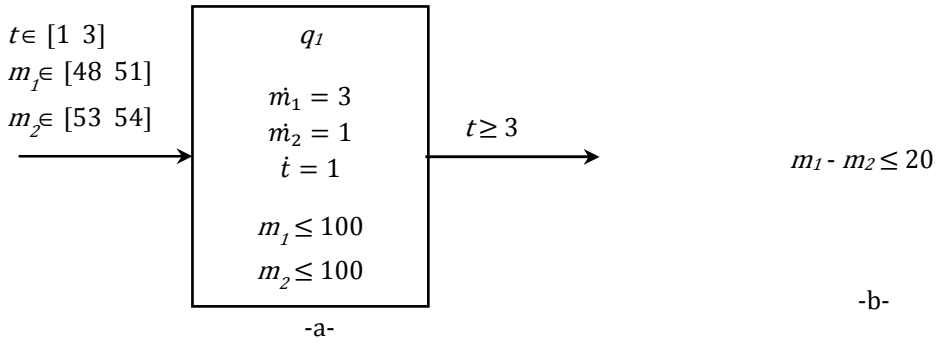


Figure 4.6.a. Comportement non contraint, b. Spécification

Le calcul de l'espace d'état atteignable dans le sommet q_1 est donné par l'ensemble des inégalités suivant :

$$R(A, q_1) = \begin{cases} m_1 - 3t \geq 39 \\ m_2 - t \geq 50 \\ 3m_2 - m_1 \geq 108 \\ m_2 \geq 53 \\ m_1 - 3m_2 \geq -114 \\ m_1 \geq 48 \\ m_2 - t \leq 53 \\ t \geq 1 \\ m_1 - 3t \leq 48 \\ m_1 \leq 100 \end{cases}$$

Nous disposons ainsi d'une caractérisation analytique de l'espace atteignable dans le sommet q_1. Nous souhaitons que cet espace ne viole jamais la spécification $m_1 - m_2 \leq 20$, elle doit être vérifiée à tout instant. On peut aisément vérifier que l'espace atteignable dans q_1, décrit par l'ensemble des inégalités ci-dessus, contient un sous-ensemble de valeurs qui violent cette spécification. Par exemple pour $t = 20$, $m_1 = 99$, $m_2 = 70$, on $m_1 - m_2 = 29$.

Le contrôle consiste à modifier la garde « $t \geq 3$ » de telle manière à ce que $m_1 - m_2$ soit toujours inférieur à 20. La méthode de calcul des nouvelles fait l'objet de la prochaine section.

III. 2. Calcul du contrôleur

Dans l'approche générale présentée dans la figure 4.3, nous sommes au niveau du block 3 (Etape 3). Le calcul du contrôleur va consister à ajouter à l'espace d'état atteignable de l'automate ***A*** la spécification et ainsi le modifier. Tous les intervalles d'existence des variables vont être réduits, c'est le cas en particulier de celui de l'horloge. C'est celui-là qui nous intéresse et qui correspond au contrôle temporisé de la transition de sortie du sommet.

Considérons un sommet q de l'automate ***A*** (figure 4.7.a), et supposons que ce sommet ne possède qu'une seule transition de sortie T_j dont la garde est fonction de l'horloge t_i : $a_i \leq t_i \leq \beta_i$. Le cas de plusieurs transitions de sortie sera aisément traité plus tard.

Le problème du contrôle maximal permissif consiste à calculer l'intervalle le plus grand $[\alpha'_i\ \beta'_i] \subseteq [\alpha_i\ \beta_i]$ tel que la garde $\alpha' \leq t_i \leq \beta'_i$ respecte les spécifications.

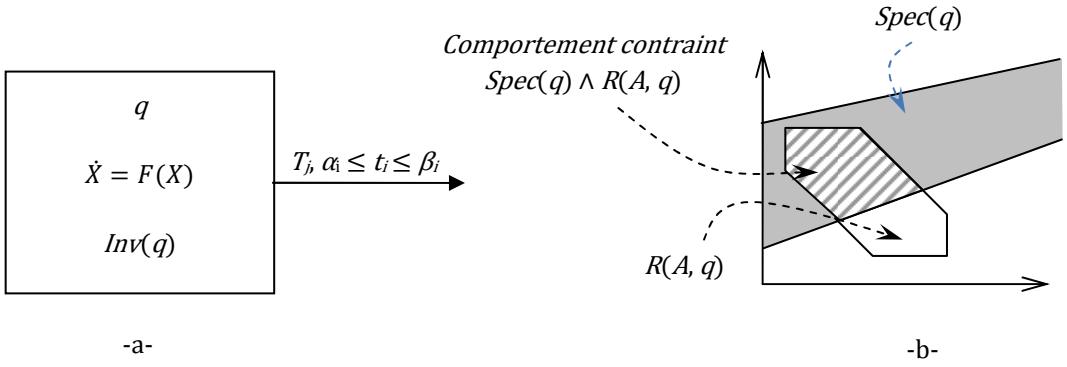


*Figure 4.7.a. Sommet q de **A**, b) Comportement contraint*

Pour faire ce calcul, nous allons nous aider des deux caractéristiques suivantes : 1) chaque variable m_i évolue suivant une fonction linéaire pendant le séjour de l'automate dans un sommet donné (la dérivée des variables d'état est constante) ; et 2) l'espace d'état à l'entrée dans le sommet est défini par une région convexe.

Ceci peut être formalisé comme suit :

Propriété 4.1 : Dans un sommet q, chaque variable continue m_i, peut être exprimée sous la forme

$$m_i = c_i(t - t_0) + d_{i0}$$

Où $t_0 \in [t_{0min}\ t_{0max}]$ et $d_{i0} \in [d_{0imin}\ d_{0imax}]$ sont respectivement les valeurs de la variable m_i et du temps à l'entrée du sommet.

$[t_{0min}\ t_{0max}]$ et $[d_{0imin}\ d_{0imax}]$ sont des intervalles convexes données par la projection orthogonale de $R_0(A, q)$ sur les axes de m_i et t.

avec $t_{0min}, c_i, d_{0imin} \in \mathbb{R}^+$ and $t_{0max}, d_{0imax} \in \mathbb{R}^+ \cup \{\infty\}$

❑

La convexité des intervalles construits ci-dessus peut être aisément démontrée par le fait que la région initiale de l'automate est convexe, et chaque transformation effectuée dans un sommet est linéaire. On sait que la propriété de convexité est conservée par application linéaire.

Pour faciliter l'énoncé du théorème et les démonstrations ci-dessous, nous proposons les notations suivantes pour tout sommet q de A :

- $C = (c_1\ c_2\ \dots\ c_{nC})^T$ est le vecteur des pentes des m_i variables, (dynamique des marquages) ;
- $D_{0min} = (d_{01min}\ d_{02min}\ \dots\ d_{0nCmin})^T$ est le vecteur des valeurs d'entrée minimales de $R_0(\mathbf{A}, q)$;

- $D_{0max} = (d_{01max}\ d_{02max}\ \dots\ d_{0nCmax})^T$ est le vecteur des valeurs d'entrée maximales de $R_0(\boldsymbol{A}, q)$
- d_{max} est le temps de séjour maximal dans le sommet q.

❑

Nous décomposons le vecteur S (vecteur des facteurs de spécification) en deux vecteurs S^- et S^+ tel que : $S = S^- + S^+$ pour séparer les composantes positives des composantes négatives. Cette décomposition nous sera utile pour calculer les intervalles optimaux, et ainsi garantir le contrôle maximal permissif.

$$S^- = \begin{pmatrix} \min(s_1, 0) \\ \min(s_2, 0) \\ \vdots \\ \min(s_{nC}, 0) \end{pmatrix} \quad \text{et} \quad S^+ = \begin{pmatrix} \max(s_1, 0) \\ \max(s_2, 0) \\ \vdots \\ \max(s_{nC}, 0) \end{pmatrix}$$

Théorème 4.1 : Le contrôle maximal permissif permettant de garantir le respect de la spécification $S^T.M_C \leq b$, dans un sommet q avec pour seule transition de sortie T_j, de garde : $a_i \leq t_i \leq \beta_i$ est obtenu avec la nouvelle garde de T_j : $a'_i \leq t_i \leq \beta'_i$

Tel que : $a'_i = \max(a_i, t_{imin})$ et $\beta'_i = \min(\beta_i, t_{imax})$

Où : $t_{imin} = t_{0min}$ et $t_{i\,max}$ sont calculés comme suit:

- Si $S^T.C > 0$ $\quad t_u = \frac{b + S^T C.t_{0\,min} - S^{+T} D_{0max} - S^{-T} D_{0min}}{S^T C}$

  i. Si $t_u < a'_i$ Le sommet q est interdit ;
  ii. Si $t_u \geq a'_i$ $\quad t_{i\,max} = t_u$

- Si $S^T.C < 0$ $\quad t_u = \frac{b + S^T C.t_{0\,min} - S^{+T} D_{0min} - S^{-T} D_{0max}}{S^T C}$

  i. Si $t_u > t_{0min}$ Le sommet q est interdit ;
  ii. Si $t_u \leq t_{0min}$ $\quad t_{imax} = d_{max}$

- Si $S^T.C = 0$,

  i. Si $S^T D_0 \leq b$, La spécification est toujours vérifiée ;
  ii. Si $S^T D_0 > b$, Le sommet q est interdit ;

❑

Preuve :

La valeur de t_{imin} est évidente ; elle correspond à la durée minimale de séjour dans le sommet q de $\mathcal{A}$. Elle est obtenue à partir de l'espace d'état atteignable dans q ;

La valeur de t_{imax} est obtenue à partir de la spécification ;

$$S^T.M_C \leq b$$

D'après la Propriété 1 : $M_C = C(t - t_0) + D_0$

Nous pouvons écrire : $S^T.(C(t - t_0)+D_0) \leq b$

Le signe du scalaire $S^T.C$ est très important lors du calcul de la valeur maximale de la borne supérieure du temps. Ce scalaire combine les poids des variables d'état dans la spécification et les pentes de ces variables. Dépendant du signe de $S^T.C$, trois cas sont distingués :

1er cas :

$S^T.C > 0$ $\Rightarrow$ la condition sur le temps vérifiant la spécification est :

$$t \leq \frac{b+S^T.C.t_0-S^T.D_0}{S^T.C}$$

et la borne la plus contraignante sur t et donnant le comportement maximal permissif est celle qui correspond au à la plus petite valeur de $\frac{b+S^T.C.t_0-S^T.D_0}{S^T.C}$, soit :

$$t_u = \frac{b+S^T C.t_{0min} - S^{+T} D_{0max} - S^{-T} D_{0min}}{S^T.C}$$

- Si $t_u < a'_i$, donc le nouvelle garde est vide et le sommet q est interdit ;
- Si $t_u \geq a'_i$, donc $t_{imax} = t_u$ et $\beta'_i = \min(\beta_i, t_{imax})$;

2ième cas :

$S^T.C < 0$ $\Rightarrow$ la condition sur le temps vérifiant la spécification est :

$$t \geq \frac{b+S^T.C.t_0-S^T.D_0}{S^T.C}$$

Et la borne la plus contraignante de t et donnant le comportement maximal permissif est celle qui correspond à la plus grande valeur de $\frac{b+S^T.C.t_0-S^T.D_0}{S^T.C}$, soit :

$$t_u = \frac{b+S^T.C.t_{0min} - S^{+T}.D_{0min} - S^{-T} D_{0max}}{S^T.C}$$

- Si $t_u > t_{0\ min}$ donc la spécification n'est pas vérifiée par l'espace d'entrée du sommet *q*, et celui-ci est interdit ;
- Si $t_u \leq t_{0\ min}$, le temps de séjour n'est plus donné que par le système, la spécification n'a aucun effet contraignant, donc :

$t_{i\ max} = d_{max}$ (temps maximum du séjour dans le sommet *q* de *A*)

et $\beta'_i = \min(\beta_i, t_{imax})$

3ième cas :

$S^T.C = 0$

- Si $S^T.D_0 \leq b$, ceci signifie que la spécification est toujours vérifiée ;
- Si $S^T.D_0 > b$, Ceci signifie que la spécification n'est pas vérifiée pour quelque valeurs des variables continue à l'entrée du sommet *q*, Ce dernier est donc interdit.

❑

Considérons la figure 4.8 ci-dessous, nous allons illustrer les deux premiers cas du théorème 4.1. Dans la figure 4.8.a, la spécification $m_1 - m_2 \leq 20$ donne :

$$S^T = (1\ -1), \text{ et } S^T.C = (1\ -1).\binom{3}{1} = 2$$

Donc : $$t_{i\ max} = \frac{20+2-51+53}{2} = 12$$

et $\beta'_i = \min(12, \infty) = 12$ et la nouvelle garde est [3, 12].

Dans la Figure 4.8.b, la spécification $m_2 - m_1 \leq 20$ donne :

$$S^T = (-1\ 1), \text{ et } S^T.C = (-1\ 1).\binom{3}{1} = -2$$

Donc : $$t_{i\ max} = \frac{20-2-53+51}{-2} = -8$$

Comme $t_{i\ max} < 1$ (borne inférieure de t_0), dans ce cas, il est nécessaire de calculer le temps maximum de séjour dans le sommet q_1 imposé par la dynamique et l'invariant :

$$d_{max} = \text{Min}\left(\frac{100-48}{3}, 100-53\right) = 17.33$$

La nouvelle garde est donc [3, 17.33].

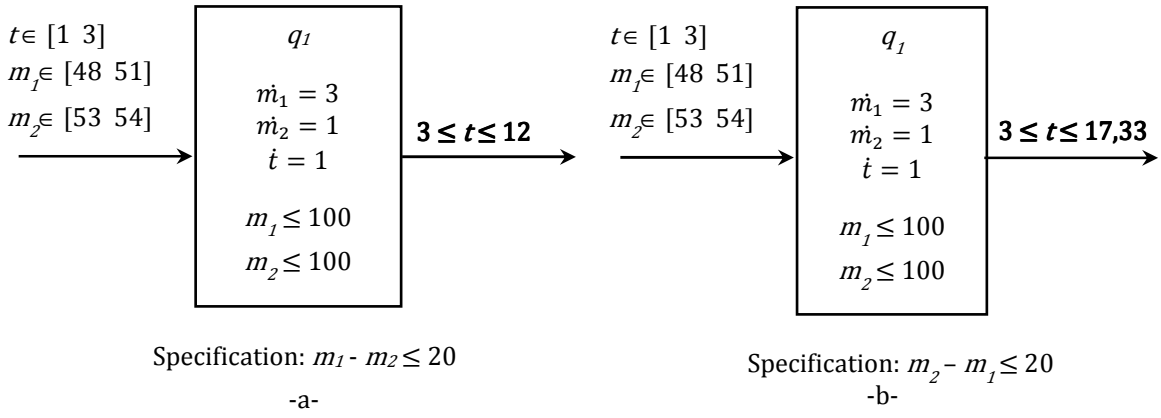


Figure 4.8 Calcul du contrôle : a) $S^T.C > 0$, b) $S^T.C < 0$

Remarque 4.2 :

Dans le cas général, la spécification est de la forme :

$$Spec(q_i) = Spec_{i1} \wedge Spec_{i2} \wedge \ \ldots \wedge Spec_{ik}$$

Pour déterminer la nouvelle garde de sortie du sommet, chaque $Spec_{ik}$ est étudiée séparément suivant le théorème 4.1. Le résultat final consiste à considérer la conjonction de toutes les gardes $[a'_{ik}\ \beta'_{ik}]$ pour obtenir la garde finale. Notons que la détermination de l'espace d'état atteignable dans le sommet q en question est effectuée une seule fois.

❑

Remarque 4.3 :

Quand un sommet a plusieurs transitions de sortie, la garde de chaque transition est calculée séparément. Nous aurons alors plusieurs transitions en concurrence dans la dynamique de sortie.

❑

Remarque 4.4 :

L'approche que nous proposons ici est puissante puisqu'elle donne une formulation algébrique au calcul du contrôleur. Ceci permet de prendre en considération n'importe quel type de spécification qui peut être différente d'un sommet à un autre. Cependant nous avons besoin de connaitre l'espace d'état à l'entrée des sommets et dans certain cas le temps de séjour maximum dans le sommet. Ces informations peuvent être obtenues en utilisant la programmation linéaire sur l'espace d'état atteignable. Le critère d'optimisation sera simplement la minimisation d'une variable donnée sur l'espace atteignable correspondant, par exemple t_0, qui donnera $t_{0\,min}$, ou t_{0miax}.

❑

III.3. Contrôleur global

Nous venons de résoudre le problème du contrôle pour un sommet en déterminant la garde associée à la transition contrôlable de sortie. Il reste maintenant à déterminer le contrôleur global. L'idée générale consiste à itérer le calcul donné par le théorème 4.1 et de descendre les branches de l'automate jusqu'à obtenir la convergence des gardes calculées. C'est un problème de calcul de point fixe qui n'est pas aisé à résoudre car se pose très vite le problème de terminaison de l'algorithme. On sait que pour les automates hybrides linéaires la construction de l'espace atteignable n'est en général pas décidable. Mais du point de vue de l'automaticien, il est possible de mettre en place des hypothèses d'initialisation des variables tout à fait réalistes qui font que l'algorithme termine. On pourra ainsi proposer un algorithme général de synthèse d'un contrôleur maximal permissif. C'est un automate temporisé ainsi que c'est défini depuis le départ puisque c'est l'action sur les horloges des transitions temporisées qui forcent les commutations, de telle façon à ce que les spécifications soient vérifiées en tout point de l'espace d'état.

L'exemple présenté dans la figure 4.1 dont l'automate hybride correspondant a été donné dans la figure 4.2 est repris ci-dessous. Le contrôleur optimal pour ce système a été calculé jusqu'à la convergence.

Exemple 4.3 : Considérons à nouveau le système de producteur consommateur, ainsi que la spécification qui impose à la taille du stock de ne jamais excéder 100 pièces. Le contrôleur garantissant le respect de la spécification est schématisé en figure 4.9. Ce contrôleur est construit sommet par sommet en appliquant le théorème 4.1 sur l'automate déplié (figure 4.2) par une technique descendante jusqu'à la convergence. Le modèle du contrôleur est un automate temporisé, puisque les points de contrôle sont des transitions discrète temporisées. La dynamique continue n'apparait pas dans le modèle du contrôleur, elle est prise en compte dans les valeurs des nouvelles gardes.

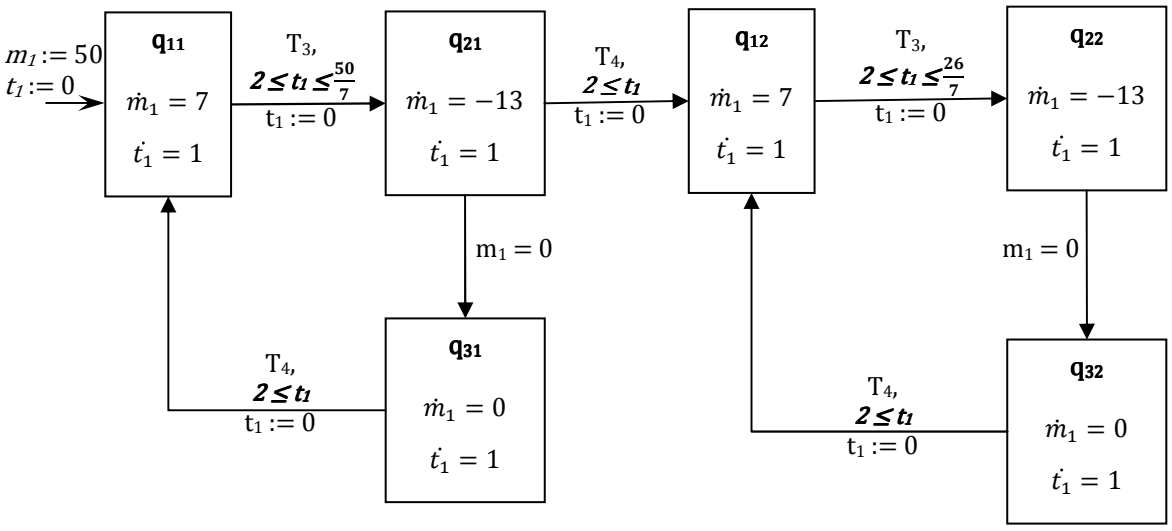


Figure 4.9. Contrôleur du système producteur consommateur

IV. Conclusion

Nous avons présenté dans cette partie une approche de la synthèse de contrôleur pour les systèmes à flux continu modélisé par des RdP D-élémentaires. Ce modèle est traduit en automate hybride linéaire, afin d'utiliser ce dernier pour la synthèse de contrôleur. La synthèse de contrôleur a pour but de limiter l'espace d'état atteignable à un espace d'état désiré. Ceci est réalisé en modifiant les gardes des transitions contrôlables. Nous avons déterminé des formules algébriques, permettant le calcul des dates limites de séjour dans les sommets permettant le respect des spécifications. Pour ce calcul nous avons besoin des valeurs limites des variables dans l'espace d'état à l'entrée des sommets, qui sont obtenue par programmation linéaire. Le contrôleur final est un automate temporisé. Il est optimal dans le sens où il donne l'espace d'état maximal garantissant les spécifications.

Ce calcul a été présenté et prouvé pour un sommet. La solution générale a été esquissée en mettant en avant les difficultés et des pistes ont été données pour garantir la terminaison de l'algorithme de construction. Ceci constitue une perspective importante pour notre recherche future.

Conclusion générale et perspectives

Ce travail s'articule autour de la synthèse de contrôleur des systèmes dynamique hybrides. Une classe particulière des systèmes hybrides a été considérée, à savoir la classe des systèmes hybrides où les variables continues sont positives et linéaires par morceaux. Cette classe englobe plusieurs problèmes réalistes, comme les procédés batch, les systèmes manufacturiers traitant une quantité importante de produits, les systèmes de transport, de communication et autres.

Pour cela nous avons adopté la démarche suivante :

- Tout d'abord nous avons utilisé le modèle réseaux de Petri hybrides D-élémentaires pour la modélisation. Cet outil a une grande capacité de modélisation et il est bien adapté pour la représentation de la classe des systèmes à flux continu. Il se distingue du RdPH classique dans les deux points suivants :

    - dans un RdPH D-élémentaire, la partie discrète est représentée par un RdP T-temporel et non par un RdP T-temporisé comme dans le cas des RdPH de base. Les RdP T-temporels introduisent un indéterminisme quant aux dates d'occurrences des évènements discrets, c'est ce degré de liberté qui nous a permis de calculer une commande.
    - dans un RdPH D-élémentaire la partie événementielle commande le comportement de la partie continue, tandis que cette dernière n'a aucune influence sur la partie discrète. Cela correspond, le plus souvent à un comportement réel.

- Un algorithme pour la traduction des réseaux de Petri hybrides D-élémentaires en automates hybrides à été utilisé pour pouvoir combiner la capacité de modélisation des automates hybrides à la capacité d'analyse des automates hybrides.

- L'automate hybride obtenu par la traduction représente le comportement non contraint du procédé, et donc souvent son espace d'état atteignable comporte une partie indésirable. Le comportement désirable est représenté par celui du procédé auquel sont ajoutées des contraintes encore dites spécifications. Les spécifications dans notre cas sont des inégalités linéaires sur les variables d'état continues.

- Pour pouvoir respecter les spécifications et donc restreindre l'espace d'état continu à un espace désirable, nous agissons sur les variables discrètes, à savoir les dates d'occurrence des événements discrets. Cela correspond dans le modèle automate hybride aux gardes des transitions.

L'approche présentée dans cette thèse a permis de déterminer des formules algébriques, permettant le calcul des dates limites de séjour dans les sommets en garantissant le respect des spécifications. Le contrôleur final est un automate temporisé. Il est optimal dans le sens où il donne l'espace d'état maximal garantissant les spécifications.

Cependant l'optimalité a été prouvée pour un sommet. La solution générale a été esquissée en mettant en avant les difficultés. C'est un problème de calcul de point fixe qui n'est pas aisé à résoudre car se pose très vite le problème de terminaison de l'algorithme. On sait que pour les automates hybrides linéaires la construction de l'espace atteignable n'est en général pas décidable. Mais du point de vue de l'automaticien, il est possible de mettre en place des hypothèses d'initialisation des variables tout à fait réalistes qui font que l'algorithme termine. On pourra ainsi proposer un algorithme général de synthèse d'un contrôleur maximal permissif. Ceci constitue une perspective importante pour notre recherche future.

Une perspective à court terme consiste à prendre en compte les événements incontrôlables dans la partie discrète. L'idée qui peut être exploitée consiste à vérifier qu'il ya une solution si l'intervalle associé à la transition discrète incontrôlable n'est pas réduit lors de la synthèse.

Enfin une perspective à plus long terme est la mise en œuvre globale de l'approche : modélisation par Réseaux de Petri hybride, traduction an automates hybrides, synthèse du contrôleur. Elle permettra la validation de l'approche sur de études de cas réels justifiant ainsi tout l'intérêt de notre travail.

Annexe

L'outil PHAVer

Cette annexe est consacrée à logiciel PHAVer (Polyhedral Hybrid Automaton Verifier). PHAVer est un outil de vérification des systèmes dynamiques hybride. Il est développé par Goran Frehse, du laboratoire Verimag de Grenoble – France. Il présente beaucoup de similitudes avec l'outil HyTech [3] développé à l'université Berkeley aux Etats-Unis. Nous allons par la suite présenter La syntaxe de PHAVer et ses points de similitudes avec HyTech.

I. Présentation de PHAVer

PHAVer est un outil pour la vérification de propriétés de sureté pour les systèmes dynamiques hybrides linéaires par morceaux. PHAVer utilise une arithmétique exacte dont la robustesse est garantie par l'utilisation de la bibliothèque *Parma Polyhedral Library* [BRZ+02]. La vérification des propriétés de sureté pour les systèmes dynamiques hybrides est ramenée à un problème de calcul d'atteignabilité, qui n'est décidable que pour une sous-classe des automates hybrides dite automates hybrides rectangulaires initialisés. PHAVer utilise un algorithme à la volée qui donne une sur-approximation des dynamiques affines par des automates hybrides linéaires. Un ensemble d'algorithmes a été développée pour réduire le nombre de bits et le nombre de contraintes qui sont nécessaires pour représenter les régions polyédrique et améliorer l'efficacité globale de l'algorithme de vérification. PHAVer a aussi la capacité de calculer les relations de simulation et de décider de l'équivalence et de raffinement entre automates hybrides.

II. Syntaxe de PHAVer

PHAVer utilise les automates hybrides à entrées/sorties [FRE05] (hybrid Input/Output automata). Dans un automate hybride à entrées/sorties l'ensemble des variables continues est scindé en trois sous-ensembles. $X = X_I \cup X_O \cup X_L$, tel que X_I est l'ensemble des variables d'entrées, X_O est l'ensemble des variables de sortie et X_L est l'ensemble des variables locales. Ces sous-ensembles sont deux à deux disjoints. $X_I \cap X_O = X_I \cap X_L = X_O \cap X_L = \varnothing$. Les variables

appartenant à X_L où à X_O sont les variables contrôlées par l'automate, tandis que les variables appartenant à X_I sont des variables non-contrôlées.

La caractéristique principale de PHAVer est qu'il différencie entre les variables d'entrée (**input_var**) et les variables de contrôle (**contr_var**). Ce qui est très important pour la vérification d'équivalence et de relation de simulation.

La syntaxe qu'utilise PHAVer pour la description textuelle des automates hybrides est similaire à celle de HyTech. La structure générale de la description d'un automate hybride est comme suit :

```
automaton automate
    contr_var: var 1, var 2,... ;
    input_var: var 3, var 4,... ;
    parameter: var 5, var ident6,... ;
    synclabs: lab_ident1, lab_ident2,... ;
    loc Sommet_1: while invariant wait { dynamique };
        when guard_1 sync label do { initialisation } goto Sommet_2;
        when ...
    loc Sommet_3: while ...
end
```

Structure de données

Il y a quatre types de structures de données qui peuvent être affectées aux identificateurs : formules linéaires, ensembles d'états symboliques, relations symboliques et automates.

- *Formules linéaires :* Elles sont spécifiées sur une collection de variables, nombre et constantes qui peuvent être combinés en utilisant +, -, /, *, (et). tant que la combinaison est linéaire sur les variables.
- *Ensemble d'états symboliques :* Un état symbolique est une combinaison d'un sommet et d'une formule linéaire, unie par &. *e.g.* **S1 & m1+m2 == 20 & t1 >0**. Un ensemble d'états symboliques d'un automate **aut** est affceté à une variable par la formule **identificateur = aut.{ensemble d'état symboliques}**.
- *Relation symbolique :* Elles sont obtenues par les algorithmes de simulation.
- *Automates :* L'affectation d'un automate à l'identificateur **aut** se fait par description de tous ces paramètres qui sont :

Variables : Toutes les variables doivent être déclarées à l'aide des instructions **state_var** (pour les variables contrôlées) et **Input_var** (pour les variables d'entrées). Notons que l'instruction **contr_var** n'est utilisée que depuis la version 0.35 et que le mot **state_var** était utilisé dans les versions antérieures.

Sommets : La déclaration des sommets est effectuée par l'instruction **loc**. L'invariant est une expression linéaire qui combine les variables d'entrées, les variables de sortie et les constantes. La définition des pentes de variables dépend des dynamiques :

i. Pour une dynamique linéaire, c'est une formule linéaire entre les variables d'état. *e.g.*, **0 ≤ m1' & m1' ≤ 10** pour $\dot{m1} \in [0\ 10]$.
ii. Pour les dynamiques affines, c'est une formule linéaire de la forme entre les variables d'état et leurs dérivés. *e.g.,* m1' == 2*m1 pour $\dot{m1} = 3m1$,

Transitions : Une transition est spécifiée par les instructions **when** …**goto**. On doit avoir toujours une étiquette de synchronisation associée à la transition. Une formule linéaire **Initialisation** spécifiées relation de saut après le mot **do**. Les variables d'état qui ne changent pas de valeur lors du franchissement de la transition doivent être spécifiées explicitement par la relation de type **x' == x**, et les variables changeant de valeur pas la relation **x' == x0**, avec x0 constant.

PHAVer dispose d'un ensemble de commandes pour l'analyse des automates hybrides.

III. Commande de PHAVer

Les principales commandes de PHAVer sont présentées ci-après :

- **& :** L'ampersand est utilisé pour la composition d'automates. *e.g.,* **Aut** = **Aut1** & **Aut2**
- **.reachable :** calcule l'ensemble des états atteignable par l'automate depuis ses états initiaux, et en faisant une analyse avant. Ainsi l'instruction **Atteignable** = **Aut.reachable** affecte l'espace d'état atteignable par l'automate **Aut** à la variable **Atteignable**.
- **.print('fichier', arg) :** écrit une description de l'automate dans le fichier **'fichier'**. le mot clé **arg** dénote le format du fichier. Elle peut prendre les valeurs 0, 1 et 2.
- **.reverse :** Inverse la causalité d'un automate. Cette commande peut être utilisée pour l'analyse en arrière d'un automate, en l'inversant puis en effectuant son analyse avant.
- **.intial_states :** remplace les états initiaux de l'automate.
- **.difference_assign :** fait la différence entre deux espaces d'états.

PHAVer donne la possibilité de représenter l'espace d'état atteignable sur des graphes bidimensionnels. Sous Linux on utilise la commande **graph** du logiciel **plotutils**, disponible sur http://www.gun.org/software/plotutils/. Sous windows/Cygwin il existe un script matlab, disponible à http://www.es.ru.nl/~goranf/.

Références bibliographiques

[1] Bibliography on Hybrid Petri Nets. http://bode.diee.unica.it/~hpn/

[2] http://www-verimag.imag.fr/~frehse/phaver_web/

[AA98] M. Allam et H. Alla, "From hybrid Petri nets to hybrid automata", Journal Européen des Systèmes Automatisés (APII-JESA), 32(9-10) : pp. 1165-185, 1998.

[ACH+95] R. Alur, C. Courkoubetis, N. Halbwachs, T.A. Henzinger, P.H. Ho, A. Olivero, J. Sifakis et S. Yovine, "*The algorithmic Analysis of Hybrid Systems*" Theoretical computer science, (138): 3-34, 1995.

[AD94] R. Alur et D. Dill, "The theory of timed automata", Theoretical Computer Science, 126: pp. 183-235, 1994.

[BBM98] Branicky. M., V. Borkar, and S. Mitter. A unified framework for hybrid control : Model and optimal control theory. IEEE Transactions on Automatic Control, 43(1), 1998.

[BD91] Bernard Berthomieu et Michel Diaz, "Modeling and verification of time dependant systems using Time Petri Nets", IEEE Transactions on Software Engineering, 17(3):259_273, Mars, 1991.

[Boy01] M. Boyer, "Contribution à la modélisation des systèmes à temps contraint et application au multimédia. Thèse de doctorat, Université Toulouse 3, 2001.

[Boy01] M. Boyer, "Contribution à la modélisation des systèmes à temps contraint et application au multimédia. Thèse de doctorat, Université Toulouse 3, 2001.

[Bra82] G.W. Brams, "Réseaux de Petri : Théorie et Pratique", Tome 1, Théorie et analyse, Masson, 1982.

[Bra83] G.W. Brams, "Réseaux de Petri : Théorie et Pratique", Tome 2, modélisation et applications, Masson, 1983.

[BRZ+02] Bagnara. R., E. Ricci, E. Zaffanella, et P. M. Hill, “Possibly not closed convex polyhedra and the Parma Polyhedra Library,” 2002.

[BST98] S. Bornot, J. Sifakis, et S. Tripakis, "Modeling urgency in timed systems". Lecture Notes in Computer Science, 1536 : 103-129, 1998.

[BW94] Brandin B. A., W. M. Wonham, 1994, “Supervisory control of Timed Discrete Event Systems”. IEEE Transactions on Automatic Control,Vol. 39 (2) : 329-342.

[Cer99] A. Cerone et A. Maggiolo-Schettini, Time-based expressivity of time Petri nets for system specification". Theoretical Computer Science, 216. Elsevier, 1999.

[CR04] F. Cassez et O.H. Roux. "Structural translation from time Petri nets to timed automata". In The 4th International Workshop on Automated Verification of Critical Systems (AVoCS 2004), London, United Kingdom, Septembre 2004.

[CR83] J.E. Coholahan et N. Roussopoulos, "Timed requirements for timed driven systems using augmented Petri nets", IEEE Transactions in Software Engineering, 9. IEEE Computer Society, 1983.

[DA05] R. David et H. Alla, " Discrete, Continuous, and Hybrid Petri Nets", Springer, 2005.

[DA07] Dideban A., Alla H., 2007, “Determination of Minimal Sets of Control Places for Safe Petri Nets”, American Control Conference, 11-13 July, New York City, USA.

[DA87] R. David et H. Alla, “Continuous Petri Nets” Dans les proceedings of the eight European workshop on application and theory of Petri nets, Pages 275-294, Zaragoza (Espagne), Juin 1987.

[DA92] R. David et H. Alla, "Du Grafcet aux Réseaux de Petri", Hermès, 1992.

[DR03] I. Demongodin et S. Rouibia, "Modélisation par Réseaux de Petri Lots et Analyse de L'état Stable Par Automates Hybrides", 4e conférence francophone de modélisation et simulation MOSIM’03, (Toulouse) France, 2003.

[FRE05] Goran Frehse. Compositional Verification of Hybrid Systems using Simulation Relations. Thèse de doctorat, Radboud Universiteit Nijmegen, Octobre 2005.

[GA07]

[GA07] L. GHOMRI, H. ALLA, "Modeling and analysis using hybrid Petri nets", Non Linear Analysis ; Hybrid Systems 1, Elsevier 2007, 141-153.

[GA08] L. GHOMRI, H. ALLA "Modelling and analysis of hybrid dynamic systems using hybrid Petri nets", in Petri Net, Theory and Applications, Edited by: Vedran Kordic , Publisher: InTech, ISBN 978-3-902613-12-7, February 2008, 113-130.

[GA11] L. GHOMRI, H. ALLA, "Contrôle temporisé optimal des systèmes dynamiques Hybrides utilisant les réseaux de Petri hybrides », ROADEF 2011, 12e congrès annuel de la Société française de Recherche Opérationnelle et d'Aide à la Décision, Saint Etienne, 2-4 mars 2011.

[GA11'] L. GHOMRI, H. ALLA, "Continuous flow Systems and Control Methodology Using Hybrid Petri nets», 2011 IEEE Conference on Automation Science and Engineering, Trieste , Italy, 24-27 août 2011.

[Gho05] GHOMRI Latéfa, Modélisation structurelle utilisant les automates hybrides et les réseaux de Petri hybrides en vue de la synthèse de contrôleur des systèmes dynamiques hybrides

[Gou99] A. Gouin, Contribution à la commande supervisée des systèmes à évènements discrets temporisés : synthèse de superviseur dans le cadre du modèle automates temporisés. PhD thesis. LISA – Université d'Angers, 1999.

[HKP+95] T.A. Henzinger, P.W. Kopke, A.Puri et P. Varaiya, "What's decidable about hybrid automata? The algorithmic analysis of hybrid systems", Proceedings of 27th annual ACM Symposium on theory of computing, pp. 373-382, 1995.

[HV87] M.A. Holliday et M.K. Vernon, "A generalized timed Petri net model for performance analysis". IEEE Transactions in Software Engineering, 13. IEEE Computer Society, 1987.

[Kha97] W. Khansa, "Réseaux de Petri p-temporels : contribution à l'étude des systèmes à événements discrets". Thèse de doctorat, Université de Savoie, 1997.

[Kou99] N. El Kouhen, Commande supervisée des systèmes à évènements discrets temporisés, PhD thesis. Laboratoire d'Automatique et d'Automatique Industrielle – Ecole Mohammadia d'ingenieur, Rabat, 1999.

[Kum91] Kumar R., "Supervisory Synthesis Techniques for Discrete Event Dynamical Systems", Thesis for the degree of Doctor of Philosophy, Université du Texas, Austin, 1991

[Lil99] J. Lilius, "Eficient state space search for time Petri nets", In MFCS Workshop on Concurrency '98, volume 18 of ENTCS. Elsevier, 1999.

[Lim04] Didier Lime, "Vérification d'applications temps réel à l'aide de réseaux de Petri temporels étendus", Thèse de PhD, Université de Nante, 2004.

[LR03] Didier Lime et Olivier H. Roux, "State class timed automaton of a time Petri net", In 10th International Workshop on Petri Nets and Performance Models, (PNPM'03). IEEE Computer Society, September 2003.

[Lyg04] Lygeros, J., editor (2004). Lecture Notes on Hybrid Systems, volume 2034. Department of Electrical and Computer Engineering University of Patras.

[Mer74] P.M. Merlin, "A study of recoverability of communication protocols". Ph.D. Thesis, Department of Computer Science, University of California, 1974.

[Mer74] P.M. Merlin, "A study of recoverability of communication protocols". Ph.D. Thesis, Department of Computer Science, University of California, 1974.

[MF76] Philip M. Merlin and David J. Farber. Recoverability of communication rotocols implications of a theoretical study. IEEE Transactions on Communications, COM-24(9):1036, 1043, 1976.

[Mur89] T. Murata, "Petri-nets: Properties, Analysis and Applications", Dans les proceedings de IEEE, 77(4) : 541-580, 1989.

[Pet62] C.A. Petri, "Kommunication mit Automaten", Thèse de PhD, Université de Bonn, Allemagne, 1962.

[Pet81] J.L. Peterson, "Petri nets theory and the modeling of systems", Prentice-Hall, 1981.

[Ram74] C. Ramchandani, "Analysis of asynchronous concurrent systems using Petri nets", PhD Thesis, Project MAC, MAC-TR 120, MIT, 1974.

[Ram74] C. Ramchandani, "Analysis of asynchronous concurrent systems using Petri nets", PhD Thesis, Project MAC, MAC-TR 120, MIT, 1974.

[RH80] C.V. Ramamoorhy et G.S. Ho, "Performance evaluation of asynchronous concurrent systems using Petri nets". IEEE Transactions in Software Engineering, 6. IEEE Computer Society, 1980.

[RW87] J.G. Ramadge et W.M. Wonham, "Supervisory control of a class of discrete event processes", SIAM J., Control and Optimisation, 25 :206-230, 1987.

[SA01] Alexandru Tiberiu Sava, "Sur la synthèse de la commande des systèmes à événements discrets temporisés", PhD thesis, Institut National polytechnique de Grenoble, Grenoble, France, novembre 2001.

[Sav02] Sava A. T., 2002, "Sur la synthèse de la commande des systèmes à événements discrets temporisés", thèse de doctorat, INPG, France.

[Sif79] J. Sifakis. "Performance evaluation of systems using nets. Net theory and applications", Advanced course on general net theory of processes and systems, LNCS 84. Springer, 1979.

[Sta78] P. H. Starke. Free Petri nets languages. Mathematical Foundations of Computer Science, LNCS 64. Springer, 1978.

[SY96] I. Sifakis et S. Yovine "Compositional specification of timed systems (extended abstract)". In 13th Syposium on Theoretical Aspects of Computer Science, pages 347-359, Grenoble, France, february 1996. Springer-Verlag.

[Wal83] B. Walter, "Timed Petri-nets for modelling and analyzing protocols with real-time characteristics". Third IFIP workshop on protocols specification, testing and verification. North-Holland, 1983.

[Yov98] S. Yovine, Model checking timed automata, embdded systems, G. Rosenberg and F. Vaandrager eds., LNCS, 1494, 1998

[ZQ01] Zaytoon, J. and Quenec'Hdu, Y. (2001). Systèmes dynamiques hybrides, chapter Sur la modélisation des systèmes hybrides, pages 87–91. Hermès Science publications, Paris.