Etude des systèmes dynamiques hybrides par représentation d'état discrète et automate hybride

Monika Kurovszky

Université Joseph Fourier - GRENOBLE 1

THESE

pour obtenir le grade de

DOCTEUR DE L'UJF

Spécialité : **≪AUTOMATIQUE-PRODUCTIQUE≫**

préparée au Laboratoire d'Automatique de Grenoble

dans le cadre de l'École Doctorale **≪Électronique, Électrotechnique, Automatique, Télécommunication et Signal≫**

présentée et soutenue publiquement

par

Monika KUROVSZKY

le 12 décembre 2002

Titre :

ÉTUDE DES SYSTÈMES DYNAMIQUES HYBRIDES PAR REPRÉSENTATION D'ÉTAT DISCRÈTE ET AUTOMATE HYBRIDE

Directeur de thèse :

M. Hassane ALLA (Laboratoire d'Automatique de Grenoble)

JURY :

Président	M. Alain BARRAUD	Professeur à l'ENSIEG, INP de Grenoble
Rapporteur	M. Jean-Claude HENNET	Directeur de Recherche CNRS, LAAS Toulouse
Rapporteur	M. Jean-Jacques LOISEAU	Directeur de Recherche CNRS, IRCyN-Ecole Centrale de Nantes
Examinateur	M. Yves QUENEC'HDU	Professeur à SUPELEC Rennes
Examinateur	M. Hassane ALLA	Professeur à l'Université Joseph Fourier

à la suite ...

Remerciements

Tout d'abord, je tiens à remercier à mon professeur M. Hassane ALLA, pour son accueil, ses conseils et sa disponibilité pendant tous ces trois ans. Ses connaissances scientifiques et son esprit d'analyse m'ont permis de mener à terme ce travail de recherche.

Je tiens à remercier M. Alain BARRAUD, professeur à l'INPG, qui m'a fait l'honneur de présider le jury réuni pour la soutenance de ma thèse de doctorat.

Je remercie également M. Jean-Claude HENNET, directeur de recherche CNRS à LAAS de Toulouse, et à Jean-Jacques LOISSEAU, directeur de recherche CNRS à IRCCyN - École Centrale de Nantes, qui m'ont fait l'honneur d'avoir accepté la charge d'être rapporteurs sur mon travail.

Je remercie à M. Yves QUENEC'HDU, professeur à SUPELEC de Rennes, qui lui aussi m'a fait l'honneur d'avoir accepté de porter un jugement sur mon travail de recherche et de faire partie du jury de soutenance de ma thèse.

Je tiens exprimer mes plus sincères remerciements et toute ma gratitude à Mme. Simona Iuliana CARAMIHAI, professeur à l'Univeristé "POLITEHNICA" de Bucarest, qui m'a dirigé vers la recherche scientifique.

Je tiens à remercier M.René DAVID pour ses remarques, toujours constructives, lors de mes premières exposés dans l'ancienne équipe SyLODI. Lors de ces échanges, j'ai pu apprécier ses qualités scientifiques et humaines.

Je remercie le directeur M. Luc DUGARD de m'avoir accueilli dans ce laboratoire.

Mes remerciements s'adressent également aux membres des équipes SyLODI et CS^2, pour leurs sympathie, leurs remarques et conseils. Merci Alexia pour les remarques et corrections de la thèse.

Je remercie également à l'ensemble de l'équipe technique et administrative pour leur gentillesse, leur bon humeur et leur efficacité.

Tout particulièrement, je remercie à mes amis Tiberiu SAVA, Rosa ABBOU, Siana Petropol, Cora et Stefan GIURGEA, Ioana FAGARASAN, Daniel MORARU, Jean-Marc BOLLON et Moez YEDDES pour leur soutien moral et les bons moments que nous avons passé ensemble.

Un grand merci à tous ceux qui, par leur soutien sous une forme ou une autre, m'ont aidé dans la réalisation de ce travail.

Une pensée très reconnaissante pour mes parents qui sont à l'origine de ma formation. Merci aussi à toute ma famille pour leur soutien ... loin des yeux mais si proches de cœur.

Merci, enfin, à Catalin pour ... tout.

Table des matières

Table des figures

Introduction

L'automatique traite différemment les problèmes de type continu et ceux de type séquentiel. Chacun de ces domaines a créé un ensemble de théories et de méthodes et développé des solutions performantes pour régler les problèmes homogènes qui se posent, mais sans toujours intégrer les solutions et les apports de l'autre domaine.

En effet, les procédés industriels sont complexes. Pour les piloter d'une manière automatique, on utilise les systèmes de contrôle commande et des automates exécutant des programmes séquentiels couplés à des boucles de régulation, monovariables et parfois multivariables qui peuvent comporter différents modes de fonctionnement. L'évolution dynamique du système de commande est à la fois continue et événementielle. Le procédé peut lui aussi présenter ce double aspect. C'est le cas des productions batch, dans lesquelles la matière est caractérisée par des variables continues et est traitée étape par étape. Pour garantir le bon fonctionnement de l'ensemble d'un système automatisé, il est nécessaire de prendre en compte les aspects continus et événementiels de sa dynamique. On pourra alors concevoir des systèmes de démarrage et d'arrêt plus fiables, étudier de façon plus rigoureuse des régulations continues fonctionnant dans les différents modes de marche et proposer des solutions pour la conception et l'analyse du système de commande.

De manière générale, les systèmes dynamiques faisant intervenir explicitement et simultanément des phénomènes ou des modèles de type dynamique continu et événementiel sont appelés *systèmes dynamiques hybrides* (SDH). Ces systèmes sont classiquement constitués de processus continus interagissant avec (ou supervisés par) des processus discrets.

L'étude des systèmes hybrides a retenu l'attention de la communauté automaticienne, ainsi que celle de la communauté informatique. Les objectifs que l'on peut assigner à l'étude des SDH consistent à apporter une solution en terme de modèle, de méthode, de performance et de qualité globale à des problèmes mal traités par les approches homogènes. Parmi ceux-ci on peut citer les problèmes générés par des variations de structure en fonction de différents modes de marche d'un système, des discontinuités du fonctionnement des machines-outils ou en robotique, la modélisation des phénomènes transitoires rapides par une commutation des modèles, la description des procédés manufacturiers (procédés batch). Pour apporter une solution à ces problèmes, la recherche sur les SDH est concentrée autour de trois axes : la modélisation, l'analyse et la commande.

La *modélisation* cherche à formaliser des modèles précis qui peuvent décrire le comportement riche (et complexe) des SDH. Ce domaine a reçu l'attention des chercheurs et plusieurs formalismes ont été proposés afin d'établir un modèle homogène permettant la conciliation entre les parties discrètes et continues. Dans Chombart [Cho97], les approches de modélisation des systèmes dynamiques hybrides sont classées en trois principales classes : (i) l'approche discrète, consistant à approximer les dynamiques continues de façon à se ramener à un système à événements discrets ; (ii) l'approche continue, consistant à approximer les dynamiques discrètes par des systèmes continus afin d'utiliser la théorie des systèmes continus et (iii) enfin la troisième approche est celle qui considère

à la fois les comportements continus et discrets dans une même structure. L'intérêt de cette dernière approche réside dans le fait qu'elle ne fait aucune abstraction d'information concernant le système à étudier.

L'approche de modélisation à laquelle nous nous sommes intéressés dans le cadre de notre travail fait partie de cette catégorie de modèles et considère le modèle du système hybride comme une extension de l'automate discret en associant une dynamique continue à chaque état discret. La composante continue est décrite par un ensemble d'équations différentielles et la composante discrète par un automate fini. Le modèle résultant de cette approche est connu comme étant le modèle *automate hybride* [Hen96].

Un *automate hybride* opère par une alternance de pas continus, où les variables d'état et le temps évoluent de façon continue, et de pas discrets, où plusieurs transitions discrètes peuvent être franchies. Il s'agit d'une extension de l'automate temporisé où la dynamique continue n'est plus représentée par des horloges (des équations d'état de type $\dot{x} = 1$) mais par des équations différentielles quelconques.

L'*analyse* porte sur le développement des outils de simulation, de validation et de vérification des SDH. Les problèmes rencontrés sont liés à la complexité de cette analyse et à l'interprétation physique de certaines propriétés à analyser telles que la stabilité globale du système à travers ses phases consécutives de fonctionnement.

Un problème central dans la vérification des propriétés des systèmes hybrides modélisés par des automates hybrides est l'analyse d'atteignabilité. Dans ce travail nous proposons une méthode de calcul exacte de l'espace atteint par l'évolution du système hybride étudié, basée sur la représentation en temps discrétisé de la dynamique continue. L'approche consiste à représenter la dynamique continue par un système linéaire discrétisé et la dynamique événementielle par un automate à états finis. L'ensemble donne un automate hybride appelé *à temps-discrets* sur lequel on applique les techniques d'atteignabilité. Ces techniques permettent d'obtenir l'*automate atteignable* qui ne contient que les évolutions possibles du système hybride.

La *commande* concerne la synthèse de contrôleurs discrets ou hybrides (ayant des sorties continues et la capacité de prendre des décisions discrètes) conformément à certains objectifs de performance et de sûreté de fonctionnement du procédé hybride commandé. Certaines approches de la commande des SDH portent sur la formulation et la résolution d'un problème de commande optimale ou des extensions de la théorie de Lyapunov ; d'autres visent à rechercher une stratégie discrète qui permette de restreindre le comportement du SDH pour satisfaire des spécifications imposées.

Notre travail concernant la synthèse de la commande s'inscrit dans cette dernière catégorie. L'objectif est d'apporter une solution originale au problème de la synthèse optimale d'un modèle de commande pour les systèmes dynamiques hybrides. A cet effet, nous proposons une démarche basée sur une extension de la théorie classique de la commande supervisée, développée par Ramadge et Wonham pour les systèmes à événements discrets. L'utilisation du temps discrétisé pour la représentation de la dynamique continue, permet d'abstraire un automate à états finis à partir de l'automate atteignable, modélisant le système hybride. Les techniques de synthèse formelle découlant des travaux de Ramadge et Wonham étendus pour les systèmes temporisés par Brandin et Wonham [BW94], peuvent ainsi être appliquées pour obtenir un superviseur optimal. Le modèle de commande résultant de notre approche est représenté sous la forme d'un automate temporisé.

L'approche que nous proposons pour la synthèse de la commande comporte les étapes suivantes :

- On modélise le comportement d'un système hybride par un automate à états finis

obtenu par le dépliage temporel de la dynamique du système ;
- On applique les techniques formelles de synthèse de superviseur ;
- Afin de réduire la taille du modèle de commande, on réalise le re-pliage temporel du modèle de commande synthétisé. Le modèle résultant est un automate temporisé.

Nous noterons la simplicité du modèle de commande ainsi obtenu. Ce qu'il est intéressant de noter est qu'on obtiens à la fin de la démarche de synthèse un automate temporisé avec une seul horloge, indiquant les dates d'occurrence auxquelles les événements contrôlables doivent être exécutés, alors que le modèle de départ est un automate hybride ayant une dynamique hybride quelconque. On s'affranchit ici de l'aspect hybride du système. On a en quelque sorte ici, une généralisation de la méthode "clock translation". Celle-ci consiste à remplacer une équation différentielle du premier ordre par une horloge moyennant une modification du prédicat associé à la transition discrète. Cette transformation est analytique. Notre démarche s'inspire de celle-ci, mais dans notre cas (système couplé), seule une solution numérique est possible.

Le rapport est organisé en quatre chapitres, comme suit :

Le **premier chapitre** de ce mémoire est constitué d'une étude bibliographique sur les systèmes dynamiques hybrides. L'objectif est de positionner notre travail par rapport aux approches existant dans la littérature.

Le **deuxième chapitre** est dédié à la présentation de l'outil de modélisation et nous introduisons une classe d'application basée sur les systèmes de production. D'abord, nous proposons une structure générique de système hybride qui synthétise ses principales caractéristiques comportementales. Ensuite, le formalisme de modélisation est présenté. Nous verrons comment les modèles des systèmes de production peuvent faire apparaître des comportements intéressants, tels que les commutations autonomes.

Dans le **troisième chapitre**, l'approche d'analyse est présentée. La question à laquelle on veut répondre est : "Comment construire un modèle d'automate qui ne contient que les évolutions possibles du système étudié ?". Pour cela, nous proposons une méthode basée sur l'analyse de franchissabilité des transitions du graphe représentant le modèle automate hybride initial. Le résultat de cette approche est l'*automate atteignable*.

Dans le **quatrième chapitre**, toutes les étapes de la démarche de synthèse d'un modèle de commande pour un système hybride seront présentées. L'objectif de la commande est de faire respecter au procédé hybride les spécifications imposées par le cahier des charges. Dans ce but, nous allons réaliser la synthèse d'un superviseur représentant l'ensemble de toutes les lois de commande qui garantissent que le fonctionnement du système respecte les spécifications. Pour ce faire, nous allons proposer un modèle formel pour l'automate déplié représenté sous la forme d'un automate à états finis qui représente l'évolution du système hybride modélisé par l'automate atteignable. Le modèle ainsi obtenu nous permettra d'utiliser la théorie classique de la commande supervisée de Brandin et Wonham et de garantir l'optimalité des résultats.

Nous achevons ce mémoire par une conclusion sur ce qui a été fait et des perspectives de continuation de ce travail.

Chapitre 1

Étude des systèmes dynamiques hybrides

Dans ce chapitre, une introduction de l'étude des systèmes dynamiques hybrides (SDH) sera faite. Les systèmes dans lesquels les dynamiques discrète et continue interagissent et où leur interaction détermine le comportement qualitatif et quantitatif de ces systèmes, sont appelés systèmes dynamiques hybrides.

Après une brève introduction de la problématique de l'étude des systèmes hybrides, les classes de phénomènes hybrides seront présentés. Parmi les approches de modélisation existantes, les principales méthodes de modélisation axées sur une approche essentiellement continue seront d'abord illustrées. Ensuite, le principe de modélisation événementielle d'un système hybride en vue de la supervision discrète sera discuté. Puis nous détaillons une approche de modélisation dite mixte dans la mesure où elle tient compte des informations continues et discrètes dans la même structure.

Ainsi, ce chapitre présente l'état de l'art qui nous permettra de situer notre travail.

1.1 Problématique

Traditionnellement, l'automatique traite différemment les problèmes de type continu et ceux de type séquentiel. Ainsi, on distingue classiquement deux spécialités : l'automatique des *systèmes continus* et celle des *systèmes à événements discrets*.

Les *systèmes continus* se caractérisent par une évolution continue de leur état dans le temps. Les modèles utilisés sont essentiellement les équations différentielles ou aux différences. Par contre, l'état des *systèmes à événements discrets* prend ses valeurs dans un ensemble dénombrable et évolue d'une manière discontinue en fonction de changements discrets appelés événements. Deux types de modèles sont à distinguer : les modèles non temporisés où seul l'ordre de l'occurrence des événements intervient et les modèles temporisés où la variable temps, de nature discrète ou continue, sert d'horloge sur laquelle l'occurrence des événements est synchronisée. On utilise l'une ou l'autre de ces approches pour l'automatisation des procédés industriels, selon leur nature et celle des produits qu'ils fabriquent. Une synthèse des techniques pour l'étude des systèmes hybrides est présentée dans [***01].

Par ailleurs, dû au développement remarquable de la technologie digitale dans ces dernières décennies, l'utilisation des contrôleurs digitaux a amélioré le degré d'automatisation des systèmes de commande pour les procédés réels. Un exemple typique est le cas des procédés batch rencontrés dans l'industrie chimique, où les ordinateurs sont utilisés pour assurer la supervision des réactions chimiques complexes. La plupart des systèmes réels sont composés par des sous-processus continus qui sont démarrés, reconfigurés et arrêtés par une commande logique discrète, à la base des horloges et des signaux de capteurs installés sur les processus continus. De plus, même les processus continus, à partir d'un certain niveau d'abstraction du modèle, peuvent être vus comme passant d'un état discret vers un autre. Il s'agit, par exemple, de l'embrayage et la collision dans les systèmes mécaniques ou de changement de phase dans les procédés chimiques.

Les systèmes dans lesquels les dynamiques discrète et continue interagissent et où leur interaction détermine le comportement qualitatif et quantitatif de ces systèmes sont appelés *systèmes dynamiques hybrides* (SDH). Ces systèmes peuvent être de natures très diverses. On peut rencontrer des systèmes continus auxquels sont associées des commutations discrètes ou bien des systèmes à événements discrets auxquels sont associées certaines évolutions continues (par exemple, c'est le cas d'un système à événements discrets temporisé).

Les impératifs liés à la maîtrise de la complexité des systèmes automatisés et aux exigences de performances et de sûreté ont conduit à devoir prendre en compte globalement le comportement de systèmes qui comprennent des *phénomènes hybrides*. Puisqu'il n'est pas toujours possible d'opérer un choix satisfaisant entre les approches continue et à événements discrets, cette impossibilité conduit, alors, à des difficultés dans la modélisation, l'analyse de systèmes ainsi que dans la conception et la synthèse de la commande. Le développement d'une théorie dédiée aux systèmes hybrides s'avère donc nécessaire. Pour répondre à ce besoin, dans la littérature, on trouve plusieurs approches théoriques, permettant de modéliser les systèmes hybrides ([ACHH93], [Hen96]), d'analyser leurs évolution ([ACH$^+$95], [Fav99], [ABDM00]) et de synthétiser une commande, telle que l'évolution du système en boucle fermée respecte les spécifications imposées ([AK98], [LTS99], [ABD$^+$00]).

Notre travail concerne l'analyse et la synthèse de la commande pour les systèmes hybrides. Nous proposons d'introduire la problématique de la recherche dans ce domaine en nous appuyant sur un exemple très simple. Ainsi, nous allons illustrer les principales

caractéristiques des systèmes hybrides, qui rendent difficile l'analyse et la synthèse de la commande.

Exemple 1.1. Considérons l'exemple classique d'un thermostat utilisé pour maintenir la température dans une chambre. Le système étudié est composé par un système de chauffage et un capteur de température. Les seuils inférieur et supérieur du thermostat sont fixés à des valeurs θ_m et, respectivement, θ_M, tel que $\theta_m < \theta_M$. Le système de chauffage est en *marche* tant que la température dans la chambre est inférieure au seuil θ_M. Le chauffage est arrêté lorsque le capteur détecte le seuil supérieur θ_M et il reste en *arrêt* jusqu'au moment où la température chute au-dessous du seuil inférieur θ_m.

La température de la chambre et le thermostat peuvent être vus comme un *système dynamique* dont l'*évolution continue* est définie par la variation de la température x dans la chambre et l'*évolution discrète* par le passage de l'état en *marche* du système de chauffage dans l'état d'*arrêt*.

Considérons que l'évolution de la température peut être modélisée par les équations différentielles suivantes :

$$\dot{x} = \begin{cases} f_1(x) = -x + \alpha & \text{si le chauffage est en } \textit{marche} \\ f_2(x) = -x & \text{sinon} \end{cases}$$

où $\alpha \in \mathbb{R}_+$ est une constante réelle positive.

D'une manière graphique, le système considéré peut être représenté par un graphe orienté présenté dans la Figure 1.1. Les sommets du graphe correspondent aux dynamiques continues des états discrets du système. Notamment, la dynamique f_1 est associée au sommet modélisant l'état en *marche* du système de chauffage et f_2 au sommet modélisant l'état d'*arrêt*.

Le passage d'un état vers l'autre est modélisé par des arcs étiquetés.

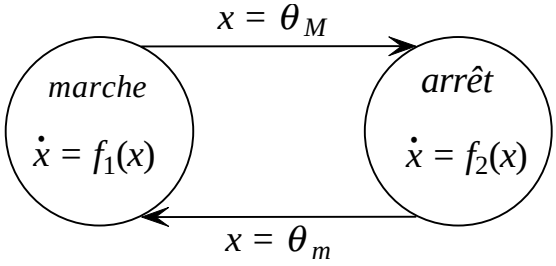


FIG. 1.1 – Modèle du thermostat

Le *problème de l'analyse* consiste à vérifier que la température dans la chambre restera toujours dans l'intervalle désiré, notamment

$$m \leq x \leq M \tag{1.1}$$

Pour cet exemple simple, les solutions analytiques des équations différentielles peuvent être facilement trouvées. Ainsi, pour une valeur initiale de la température $x(0) = \theta_0$, les solutions analytiques trouvées sont : $x(t) = \theta_0 e^{-t} + \alpha(1 - e^{-t})$ pour la dynamique correspondant à l'état de marche du système de chauffage et $x(t) = \theta_0 e^{-t}$ pour l'autre état.

Initialement, supposons que le système est dans l'état en *marche* et la valeur initiale de

la température vérifie la relation $\theta_0 \in [m, M]$. Dans cet état, l'évolution de la température respectera l'expression,

$$x(t) = \theta_0 e^{-t} + \alpha(1 - e^{-t}).$$

L'évolution croissante fait que, au bout de t_1 unités de temps, le seuil θ_M est atteint. Alors, le système de chauffage passera dans l'état d'*arrêt*. Suite au changement d'état du système, la dynamique de la température change et la nouvelle évolution est donnée par

$$x(t) = \theta_0 e^{-(t+t_1)}.$$

Dans cet état, la température aura une évolution décroissante jusqu'au moment où le seuil inférieur θ_m est atteint. À cet instant, le chauffage sera remis en *marche* et le système reviendra dans l'état initial.

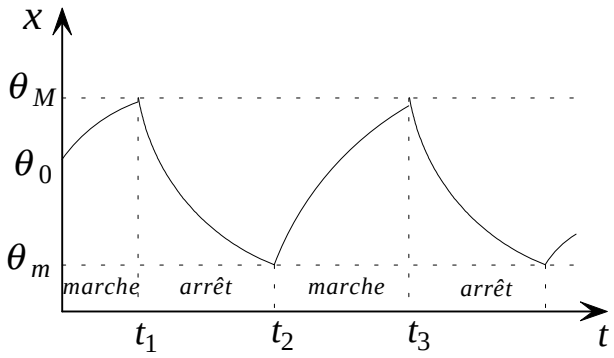


FIG. 1.2 – Trajectoire de la température

La trajectoire de la température est présentée sur la Figure 1.2. Dans cette représentation, nous pouvons remarquer que la température de la chambre restera dans l'intervalle désiré (Relation 1.1) seulement si les seuils θ_m et θ_M vérifient les conditions suivantes :

$$\theta_m \geq m \quad \text{et} \quad \theta_M \leq M \tag{1.2}$$

■

Nous pouvons conclure qu'une résolution analytique pour le problème d'analyse est possible seulement si les solutions des équations différentielles sont connues et si elles ne sont pas trop complexes. Pour les expressions plus complexes, l'utilisation des techniques numériques est indispensable pour simuler l'évolution du système et analyser son comportement.

L'évolution obtenue pour le modèle du thermostat est *déterministe* puisque nous avons considéré des conditions idéales de fonctionnement pour notre système. Notamment, nous avons supposé que le capteur de température détecte d'une manière très précise les seuils de commutations θ_m et θ_M, et dès qu'un de ces deux seuils est détecté, le système change instantanément d'état. Toutes ces conditions sont des conditions de fonctionnement jamais rencontrées dans les systèmes réels, d'une part, puisque la précision des capteurs est limitée, d'autre part, parce que le changement d'état du système ne peut pas se produire instantanément dû aux retards introduits par les composantes du système lors de la transmission d'une information.

L'effet de ces imprécisions est que, en général, les commutations entre les différents états du systèmes ne peuvent pas être garanties à des instants précis. Cela implique une évolution *non-déterministe* du système. Par conséquent, avant de commencer à analyser l'évolution d'un système, nous devons être sûrs que toutes les évolutions du processus

réel sont prises en compte.

Exemple 1.2. Pour pouvoir prendre en compte le non-déterminisme dû à l'imprécision du capteur de température dans l'exemple du thermostat, nous allons modifier les conditions de passage d'un état vers l'autre comme suit : le chauffage est arrêté lorsque la température vérifie la relation $\theta_M - \varepsilon_M \leq x \leq \theta_M + \varepsilon_M$ et il est mis en marche lorsque $\theta_m - \varepsilon_m \leq x \leq \theta_m + \varepsilon_m$, où $\varepsilon_M, \varepsilon_M \in \mathbb{R}^{+1}$ est une constante (Figure 1.3). Les conditions de commutation d'un état vers l'autre du système, exprimées par des intervalles, signifient que le changement d'état peut se faire à n'importe quel instant dès que la température prend des valeurs dans l'intervalle spécifié.

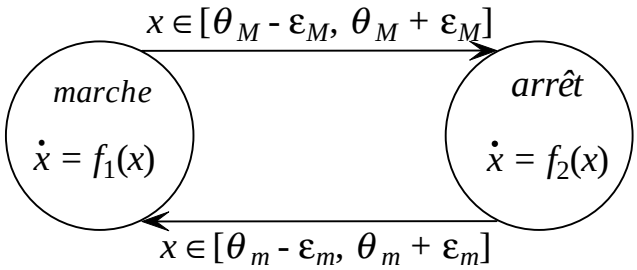


FIG. 1.3 – Modèle non-déterministe du thermostat

Le comportement du système est non-déterministe dans le sens où pour une même condition initiale les trajectoires de la température peuvent être différentes (Figure 1.4).

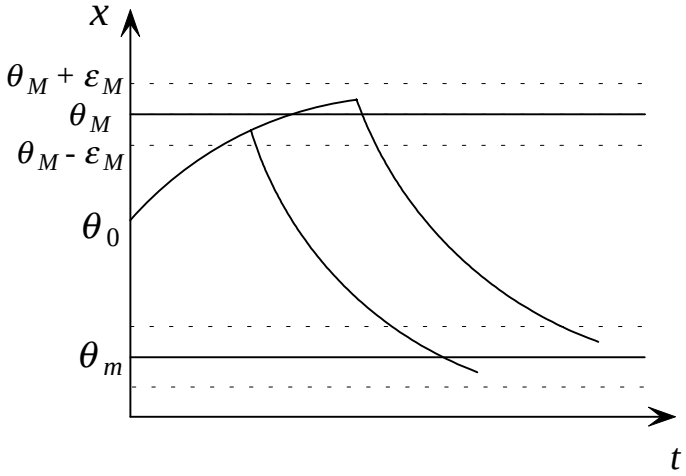


FIG. 1.4 – Deux trajectoires différentes pour la même condition initiale θ_0

■

La plupart des systèmes réels ont une évolution non-déterministe. Parfois, dans les systèmes réels, les conditions initiales ne sont pas bien précisées, dans le sens où la valeur initiale est donnée comme étant incluse dans un intervalle. Par exemple, pour le thermostat la valeur initiale de la température n'est plus définie par une valeur précise $x(0) = \theta_0$ mais par une valeur $x(0)$ appartenant à un intervalle, $x(0) \in [\theta_0 - \varepsilon_0, \theta_0 + \varepsilon_0]$, où $\varepsilon_0 \in \mathbb{R}^+$. Dans ce contexte, la trajectoire unique obtenue (Figure 1.2) sera remplacée

[1]Nous utilisons $\mathbb{R}^+$ pour indiquer l'ensemble des nombres réels positifs.

par un ensemble infini de trajectoires dont chacune correspond à un point appartenant à cet intervalle. Si le non-déterminisme introduit par les composantes physiques du système est aussi pris en compte, alors, même par simulations numériques le problème n'est pas facile à résoudre.

Généralement, l'évolution continue est décrite par une équation différentielle $\dot{x} = f(x, u)$. Un autre type de non-déterminisme peut être introduit par la variation du vecteur d'entrée u. Ainsi, pour une même condition initiale, un ensemble de trajectoires sera généré, chacune de ces trajectoires correspond à une entrée du système. Dans l'exemple du thermostat, des incertitudes de ce genre peuvent être introduites par la variation incontrôlable de la température externe.

Comme nous avons pu le constater, le non-déterminisme rend difficile l'analyse des systèmes 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 de ce vecteur, même en utilisant des techniques de simulation numérique, il est impossible de simuler l'évolution du système pour toutes les valeurs du vecteur d'entrée. Alors, l'approche par simulation s'avère inadaptée lorsque des solutions analytiques simples ne peuvent pas être trouvées. Si pour les systèmes uni-dimensionnels, où l'évolution est déterministe, des méthodes analytiques peuvent être utilisées pour analyser leurs évolutions, dans le cas où les systèmes sont multi-dimensionnels et leur évolution est non-déterministe ces méthodes ne sont plus adaptées. Ainsi, le problème de l'analyse pour les systèmes complexes reste ouvert.

Le problème de *synthèse de la commande* consiste à intervenir dans l'évolution du système au bon moment, pour assurer le respect des spécifications imposées tout au long de son fonctionnement. Pour l'exemple du thermostat, l'*objectif* est de maintenir la température de la chambre dans l'intervalle $[m, M]$. Supposons que nous avons un thermostat dont la structure est fixe, mais dont les seuils de commutation, θ_m et θ_M, peuvent être modifiés tel que l'objectif de la commande puisse être réalisé. Donc, le but est de trouver l'instant de commutation entre les deux états du système tel que la valeur de la température dans la chambre reste toujours entre les limites imposées.

Par l'exemple présenté ainsi que par les remarques faites, pour développer une approche d'analyse et de synthèse des systèmes hybrides nous devons disposer :

- d'outils de modélisation permettant de représenter l'interaction entre les dynamiques continues et discrètes, ainsi que les spécifications relatives au fonctionnement désiré du système ;
- de méthodes d'analyse rigoureuses dans le sens où elles peuvent prendre en compte tous les comportements possibles du système.

Notre approche s'adresse aux systèmes hybrides dont la dynamique continue est linéaire, modélisée par $\dot{x}(t) = f(x, u) = Ax + Bu$, et dont le vecteur d'entrée est constant.

Le travail que nous allons présenter a pour but la synthèse optimale d'un modèle de commande pour les systèmes hybrides, basée sur une extension de la théorie classique de la commande supervisée développée par Ramadge et Wonham pour les systèmes à événements discrets. Par ailleurs, les systèmes hybrides sont souvent des systèmes complexes, ainsi l'analyse et la synthèse automatique de la commande est très souhaitable. Ceci nous a motivé pour proposer une approche algorithmique dans la perspective d'une mise en oeuvre informatique.

1.2 Classes de phénomènes hybrides

La nature *hybride* d'un système peut être inhérente aux phénomènes physiques qui le régissent. Un certain nombre des phénomènes physiques considérés comme hybrides ont été regroupés en quatre catégories principales traduisant leur influence sur les modèles mathématiques utilisés pour décrire les différentes classes des systèmes ([Bra95]).

1.2.1 *Commutation autonome*

Une *commutation autonome* caractérise un phénomène où le champ de vecteur $f(x,t)$ change de façon discontinue lorsque l'état $x(.)$ atteint certains seuils. C'est le cas d'un système à hysteresis.

Le *saut autonome* est un phénomène similaire rencontré dans les systèmes mécaniques, où l'état $x(.)$ effectue un saut lorsqu'il atteint certaines régions de l'espace, c'est-à-dire qu'il passe de façon discontinue de sa valeur courante à une autre. Un exemple de ce phénomène est donné par la collision de 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(a).

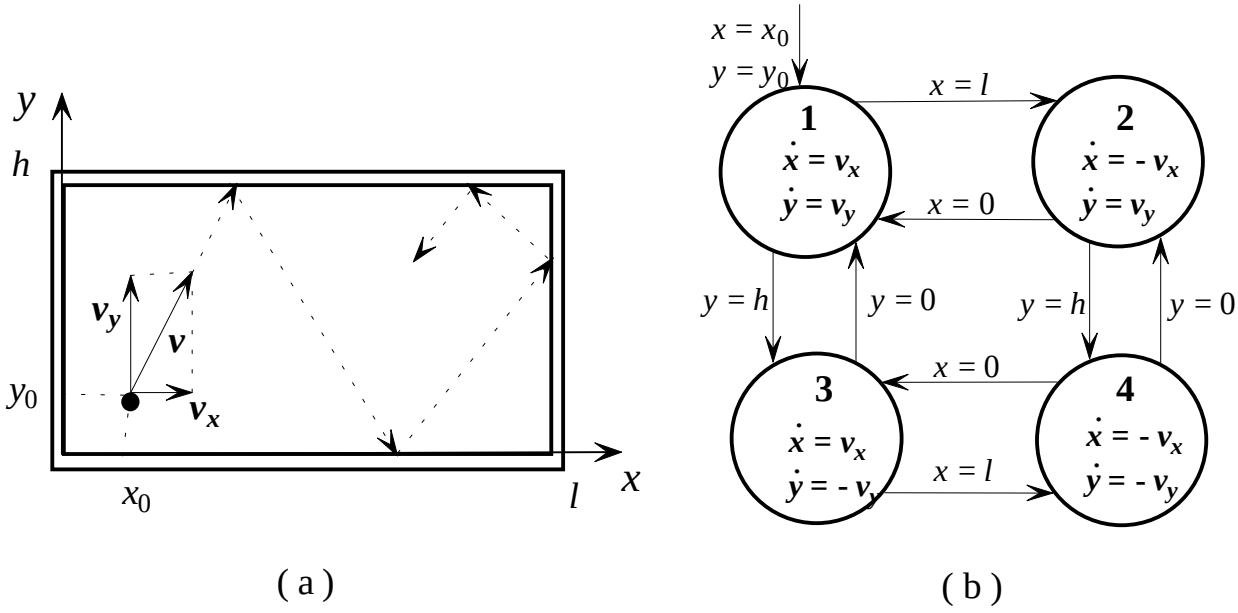


FIG. 1.5 – (a) Trajectoire d'une boule de billard et (b) Automate associé

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. 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 combinaison des signes des composantes de la vitesse donne quatre directions différentes du mouvement de la boule.

L'automate modélisant le mouvement de la boule est représenté dans la Figure 1.5(b). Le vecteur de la vitesse peut avoir quatre états différents $[v_x, v_y]^T$, $[-v_x, v_y]^T$, $[v_x, -v_y]^T$ et $[-v_x, -v_y]^T$. Chaque état de la vitesse caractérise une dynamique des variables x et

y, représentée par un sommet de l'automate. Le passage d'une dynamique à une autre est modélisé par les arcs de l'automate et se produit lorsque la boule atteint un côté de la table, c'est-à-dire, quand x atteint les valeurs 0 ou l et/ou y atteint les valeurs 0 ou h.

1.2.2 *Commutation contrôlée*

Une *commutation contrôlée* traduit une phénomène où le champ de vecteur $f(x,t)$ change de façon discontinue et instantanée en réponse à une entrée de commande.

Le phénomène de commutation contrôlée est illustré à travers l'exemple de l'embrayage mécanique. Il s'agit d'un système composé de deux masses en rotation. Les masses sont couplées par un embrayage mécanique idéal. Chaque masse i, dont l'inertie est J_i, est entraînée par un couple Q_i à une vitesse de rotation ω_i. Quand les masses sont couplées, les valeurs des vitesses de rotation sont identiques. Ces vitesses sont indépendantes quand les masses sont découplées (Figure 1.6).

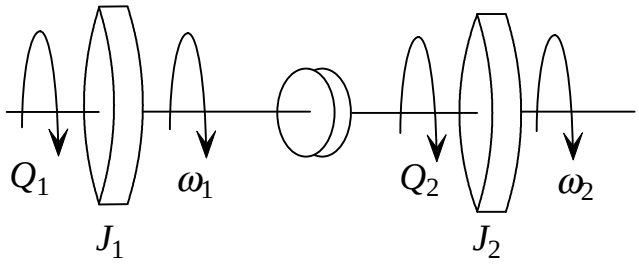


FIG. 1.6 – Système d'embrayage mécanique

Lorsque les axes de rotation sont indépendants le système est *débrayé*. Le modèle mathématique est donné par

$$\begin{pmatrix} J_1 & 0 \\ 0 & J_2 \end{pmatrix} \begin{pmatrix} \dot{\omega}_1 \\ \dot{\omega}_2 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \omega_1 \\ \omega_2 \end{pmatrix} + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} Q_1 \\ Q_2 \end{pmatrix}$$

Quand les axes de rotation sont couplées le système est *embrayé*, les vitesses de rotation sont identiques et le système peut être décrit par l'équation

$$\begin{pmatrix} J_1 & J_2 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \dot{\omega}_1 \\ \dot{\omega}_2 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} \omega_1 \\ \omega_2 \end{pmatrix} + \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} Q_1 \\ Q_2 \end{pmatrix}$$

D'une manière plus générale, le système peut donc être décrit par une équation sous la forme :

$$I_k \dot{x} = A_k x + B_k u$$

où les matrices I_k, A_k et B_k dépendent de l'état k du système. Nous avons donc une équation d'état dans laquelle figure la représentation hybride décrivant la dynamique continue du système dans ses différents états.

1.3 Modélisation des SDH

De façon générale, un système hybride sera modélisé par un ensemble de systèmes à dynamique continue interagissant avec un ou plusieurs systèmes à événements discrets.

Un survol des principales approches proposées dans la littérature, portant sur la modélisation des systèmes hybrides, peut être trouvé dans [AK98]. Le point commun entre ces différentes approches est que l'évolution continue est affectée par les événements discrets et les modèles nécessitent à la fois des variables d'état continues et discrètes.

Dans les travaux de thèse de Chombart [Cho97], les approches de modélisation sont regroupées en trois classes principales :

- L'*approche continue* : il s'agit d'étudier le comportement des modèles continus en présence des discontinuités, et éventuellement, de définir un modèle "étendu".
- L'*approche événementielle* : contrairement à l'approche continue, cette approche consiste dans l'approximation de la dynamique continue de telle manière que le système hybride ne soit représenté que par les événements qui le caractérisent.
- L'*approche mixte* : cette approche repose sur la supposition que le fonctionnement d'un système dynamique hybride est une séquence de deux phases : une transformation continue de l'état continu suivie d'un changement discret instantané. L'approche conduit à rechercher des modifications dans les modèles discrets adaptées à la modélisation des phénomènes hybrides, en particulier en étudiant l'insertion des variables continues.

Dans la suite de cette section nous n'avons retenu, parmi les approches proposées dans la littérature, que les plus intéressantes pour situer notre travail et celles dans lesquelles nous avons trouvé des réponses à l'objectif que nous nous sommes fixé.

1.3.1 *Approche de modélisation continue*

Cette approche consiste à définir une approximation des dynamiques discrètes du système hybride par des équations différentielles (ou aux différences) pour modéliser l'occurrence des événements discrets. L'idée est qu'en utilisant une approche unifiée dans le domaine des systèmes continus, où les théories sont bien établies, les questions de stabilité, de commandabilité et d'observabilité pourront être étudiées selon les théories classiques.

Dans la littérature nous trouvons plusieurs travaux qui traitent la modélisation des systèmes hybrides par une approche continue. Ces travaux peuvent être divisés en deux catégories. Une partie orientée vers la définition d'ensemble de transitions pour pouvoir prendre en compte l'évolution discrète du système hybride. L'autre partie est basée sur l'introduction des variables supplémentaires dans le modèle continu, représenté par des équations différentielles, pour décrire le fonctionnement des systèmes hybrides.

Définition d'un ensemble de transitions

Dans [Wit66], l'auteur considère une classe des systèmes hybrides pour laquelle :

- lors d'une transition discrète, le vecteur d'état $x(.)$ est continu (pas de saut d'état), alors que le champ du vecteur $f(x,t)$ peut subir une discontinuité ;
- une transition a lieu seulement si le vecteur d'état continu satisfait une condition prédéfinie.

Les entrées n'influencent les transitions que par intégration des équations différentielles modélisant les dynamiques continues, mais jamais directement. Cela signifie qu'il n'y a pas de dynamique discrète indépendante.

Le modèle proposé est conçu pour s'appliquer à un certain nombre de systèmes réels comme par exemple les bascules logiques, les relais tout ou rien, ainsi que les ruptures mécaniques. La partie continue du modèle a une forme traditionnelle à laquelle un argu-

ment m prenant en compte l'aspect discret est ajouté : $\dot{x} = f(m, x(t), u(t))$. Ainsi, l'*état* du système hybride est caractérisé par la paire $(m, x) \in M \times E^n$ où E^n est l'espace d'état continu et M est un ensemble fini d'entiers positifs qui indexe les états discrets.

Les *conditions de transition* sont définies par des ensembles. L'état discret change de $m = i$ à $m = j \neq i$ lorsque x atteint un ensemble donné $I_{ij} \subset E^n$ pour chaque paire ordonnée (i, j). L'*ensemble d'arrivée* et *de départ* d'un état discret i sont défini et l'auteur introduit des hypothèses sur ces ensembles pour garantir l'existence et l'unicité de l'évolution dynamique du système. À partir de ce modèle, le problème de la commande optimale d'un système hybride, formulé en terme de minimisation d'un critère choisi est étudié. Il s'agit de trouver les commandes admissibles dans un ensemble de commandes possibles qui minimisent le coût de fonctionnement du système.

D'autres travaux [Tav87] et [BBM94], reprennent la notion d'ensemble de transitions en définissant des frontières correspondant à des valeurs particulières de l'état continu qui entraînent des changements d'état discret non seulement sous forme de commutations de champs de vecteur dans les travaux de Tavernini, mais également des sauts d'état dans les travaux plus récents de Branicky.

Utilisation d'une variable compteur

Dans [Bro93], l'auteur propose des équations différentielles et des phénomènes discrets pour décrire les systèmes dynamiques. Dans [Bro94], une étude reposant sur le principe de modélisation d'un système hybride par une approche continue est présentée. Dans ce travail, l'auteur étudie la modélisation d'une classe de systèmes qui a comme entrées, d'une part, des fonctions continues et, d'autre part, des chaînes de symboles, et dont les sorties se trouvent également sous la forme de chaînes de symboles et de valeurs réelles. L'idée correspond à un échantillonnage en temps régulier.

Afin de bien modéliser et prévoir les aspects significatifs dans le comportement dynamique d'un système hybride, l'auteur considère que le modèle doit :

- donner une description temporelle de l'évolution de l'automate,
- spécifier l'interaction entre les parties discrètes et continues et
- spécifier les équations d'évolution.

Le modèle proposé est donné sous la forme d'équations aux différences :

$$x(k+1) = f(x(k), u(k))$$
$$y(k) = h(x(k))$$

Pour "mixer" les commandes symboliques et continues, une nouvelle variable p, illustrée sur la Figure 1.7, est introduite jouant le rôle de compteur. La variable t_p est utilisée pour dénoter la valeur t quand la variable p atteint une valeur entière.
La commande continue $u(t)$ est exercée à l'instant t et la commande discrète $v\lfloor p \rfloor$ [2] est appliquée lorsque la variable p atteint une valeur entière, c'est-à-dire quand $t = t_p$. Ainsi, le temps t_p peut être interprété comme le temps (valeur de t) au moment de l'occurrence d'un événement discret. Une représentation de la dynamique dans le modèle de Brockett est illustrée dans la Figure 1.8.
Le modèle proposé permet de décrire le fonctionnement de systèmes hybrides régis par des phénomènes de commutations autonomes et de commutations contrôlées, suivant une logique de commande.

[2] Les notations $\lfloor p \rfloor$ et $\lceil p \rceil$ représentent respectivement le plus petit entier supérieur ou égal à p et le plus grand entier inférieur ou égal à p.

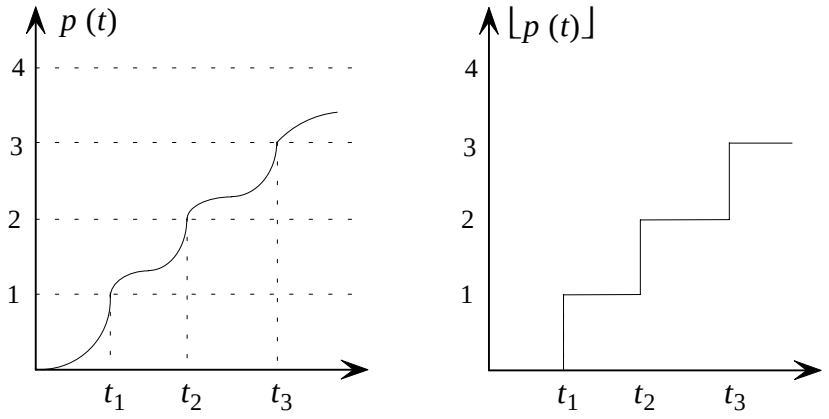


FIG. 1.7 – Fonction compteur p et interpretation de $\lfloor p \rfloor$ et de t_p

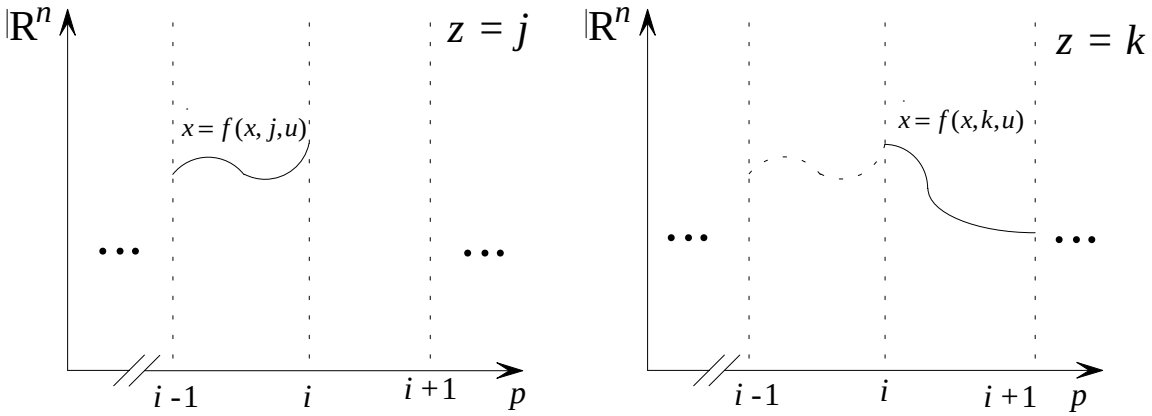


FIG. 1.8 – Evolution dynamique du modèle de Brockett

1.3.2 *Approche de modélisation événementielle*

Avoir une approche purement discrète pour modéliser les systèmes hybrides consiste à supprimer les dynamiques continues ou à faire une approximation de l'évolution continue de façon à ce que le système hybride soit représenté uniquement par les événements qui le caractérisent. La modélisation événementielle d'un SDH permettra ainsi faire appel à la théorie classique de supervision des SED pour la synthèse d'un modèle de commande.

Notions sur la supervision des SED

Un système à événements discrets (SED) est un système dynamique dans lequel l'espace des états est discret. Ses trajectoires d'états sont constantes par morceaux. Un tel système évolue conformément à l'occurrence d'événements physiques à des intervalles de temps généralement irréguliers ou inconnus.

La théorie de la supervision des SED a été initiée par les travaux de Ramadge et Wonham ([RW87], [RW89]). Cette théorie utilise le modèle automate et les langages formels [HU79] pour modéliser le comportement d'un SED ainsi que les spécifications imposées pour son fonctionnement. Un aperçu des concepts de base de la théorie des langages introduite par Ramdge et Wonham pour la commande de SED est présenté dans [LL98].

Le *principe de la supervision* d'un procédé est d'interdire l'occurrence de certains événements dans chacun de ses états et d'autoriser l'occurrence des autres, par l'action

d'un superviseur. L'*objectif* est de construire un superviseur tel que le procédé supervisé évolue avec un maximum de liberté tout en respectant les spécifications imposées pour son fonctionnement. Le schéma général du principe de la supervision est présenté dans la Figure 1.9.

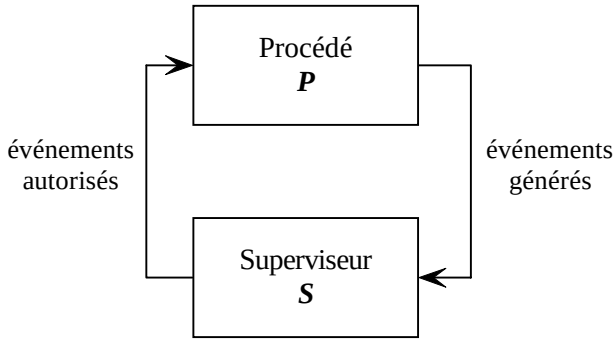


FIG. 1.9 – Schéma du principe général de la supervision

Le superviseur observe l'état du procédé par l'intermédiaire de la séquence d'événements générés par le fonctionnement du procédé. Cette séquence est l'entrée du superviseur. En réponse, il agit sur le comportement du procédé par l'intermédiaire des lois de contrôle qui définissent les événements autorisés depuis l'état courant du procédé.

Généralement, un superviseur ne peut pas agir sur tous les événements qui interviennent dans le fonctionnement d'un procédé. En pratique, certains événements ne peuvent pas être interdits. Ces événements sont appelés *incontrôlables*. Les événements qui peuvent être interdits quel que soit l'état du procédé sont appelés *contrôlables*. Par conséquent, l'ensemble $\sum$ est partagé en deux sous-ensembles disjoints : $\sum = \sum_c \cup \sum_u$. Le sous-ensemble $\sum_c$ mémorise les événements contrôlables tandis que $\sum_u$ dénote le sous-ensemble des événements incontrôlables.

Dans le domaine de l'automatique, les événements contrôlables correspondent en général aux actions appliquées au procédé tandis que les événements incontrôlables sont associés aux sorties (fournies par les capteurs) du procédé.

Une entrée de contrôle pour un procédé P est un sous-ensemble $\gamma \subseteq \sum$ tel que $\sum_u \subseteq \gamma$. Elle représente l'ensemble des événements autorisés par le superviseur depuis un état du procédé. Étant donné que les événements incontrôlables ne peuvent pas être interdits, chaque entrée de contrôle contient tous les événements incontrôlables. L'ensemble des entrées de contrôle est noté avec Γ.
Formellement, un superviseur est défini par la fonction :

$$S : L(P) \rightarrow \Gamma$$

Le fonctionnement du procédé P non supervisé, modélisé par un langage $L(P)$, est appelé fonctionnement en boucle ouverte. Par contre, le fonctionnement du procédé couplé avec son superviseur S, modélisé par le langage $L(S/P)$, est appelé fonctionnement en boucle fermée. Ce fonctionnement doit respecter les spécifications imposées par le cahier des charges.

Définition 1.1. Le langage $L(S/P)$, qui décrit le fonctionnement d'un procédé P supervisé par un superviseur S est défini par :

- $e \in L(S/P)$,
- $wa \in L(S/P) \Leftrightarrow w \in L(S/P),\ a \in S(w)$ et $wa \in L(P)$

■

Un mot wa peut être généré par le procédé supervisé seulement si le mot w a été généré par le procédé supervisé, si l'événement a est autorisé par le superviseur et le mot wa est accepté par le procédé non supervisé. Le mot vide e est compris dans le langage $L(S/P)$. Ainsi, $L(S/P)$ est inclus dans le langage $L(P)$ et il est préfixe-clôt. Par conséquent, un superviseur ne peut que restreindre le comportement d'un procédé.

Nous avons vu que la tâche d'un superviseur est de générer l'entrée de contrôle telle que le procédé évolue avec un maximum de liberté tout en respectant certaines contraintes. Par contre, à cause de l'existence des événements incontrôlables, il n'est pas possible de restreindre le comportement du procédé à n'importe quel sous-ensemble.

On appelle *langage désiré* d'un procédé, l'ensemble de mots générés par le procédé tout en respectant les contraintes imposées par le cahier des charges. Soit K le langage désiré pour un procédé P. L'étude de l'existence d'un superviseur S tel que $L(S/P) = K$ est basée sur la propriété de contrôlabilité du langage K.

Définition 1.2. Un langage K est dit contrôlable par rapport à un langage $L(P)$ si $\overline{K}\sum_u \cap L(P) \subseteq \overline{K}$.

■

Si un langage K est contrôlable par rapport à un langage $L(P)$, alors sa clôture préfixielle, notée $\overline{K}$, est invariante par rapport à l'occurrence des événements incontrôlables.

Si un langage K n'est pas contrôlable par rapport à un langage $L(P)$, alors il n'existe aucun superviseur S tel que $L(S/P) = K$. Dans ce cas, il faut chercher un langage contrôlable et préfixe-clôt inclus dans K. L'objectif étant de construire un superviseur le plus permissif possible, on cherche le plus grand langage préfixe-clôt et contrôlable, inclus en K, noté $SupC(K)$.

Le problème de l'incontrôlabilité d'un langage est illustré dans la figure 1.10.

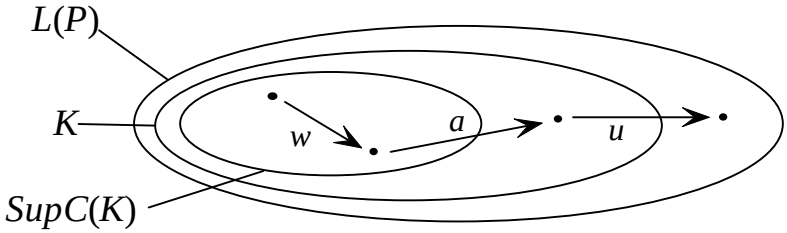


FIG. 1.10 – Incontrôlabilité d'un langage

Soit un mot $wau \in L(P)$, où :
- w est un mot tel que $w \in SupC(K)$,
- a est un événement contrôlable tel que $wa \in K - SupC(K)$,
- u est une séquence d'événements incontrôlables telle que $wau \in L(P) - K$.

Le langage K est incontrôlable parce qu'il est impossible d'interdire l'exécution du mot u à la suite de wa. Par contre, en interdisant l'occurrence de l'événement contrôlable a depuis le mot w, le problème de contrôlabilité ne se pose plus.

Proposition 1.3.1. *Soit un SED P non bloquant de langage $L(P)$ et de langage marqué $L_m(P)$.*

1. *Pour tout langage $K \subseteq L(P)$ il existe un superviseur S tel que $L(S/P) = K$ ssi K est préfixe-clôt et contrôlable par rapport à $L(P)$.*

2. *Pour tout langage $K \subseteq L_m(P)$ il existe un superviseur S tel que $L_m(S/P) = K$ et le système en boucle fermée est non bloquant ssi K est L_m-fermé (c'est à dire $\overline{K} \cap L_m = K$) et contrôlable par rapport à $L(P)$.*

■

Cette proposition est essentielle dans la théorie de la supervision des SED, mais elle n'est pas utilisée pour la construction d'un superviseur. Dans la pratique, la synthèse du superviseur est effectuée à l'aide de l'outil automate. Le comportement du procédé à superviser ainsi que les spécifications imposées pour son fonctionnement sont modélisés par des automates. Le langage désiré du procédé est représenté par l'automate obtenu par la composition synchrone des automates qui modélisent le procédé et les spécifications.

Plusieurs approches ont été proposées pour la supervision des SED. Parmi ces approches, Kumar [Kum91] a proposé un algorithme pour la construction du superviseur le plus permissif basé sur la détermination des états interdits du modèle automate représentant le fonctionnement désiré. Un état est considéré comme interdit s'il ne respecte pas les spécifications. C'est-à-dire, il est atteint depuis un état du procédé par le franchissement d'une transition de sortie sur un événement incontrôlable qui n'est pas autorisé par la spécification. Un état à partir duquel le système peut évoluer vers un état interdit par l'occurrence d'un événement incontrôlable est considéré comme faiblement interdit. Le superviseur est construit en éliminant de l'automate du fonctionnement désiré, tous les états interdits et faiblement interdits. Cet algorithme permet d'obtenir un superviseur le plus permissif.

Dans la suite, nous présentons deux approches de modélisation qui font appel à la théorie de la commande supervisée de Ramadge-Wonham, permettant la synthèse d'un modèle de commande pour un système hybride. La première consiste dans la partition de l'espace d'état continu et l'autre est basée sur les invariants naturels du procédé.

Partition de l'espace d'état continu

Dans [ASL93], les auteurs montrent comment un système de commande hybride peut être représenté par l'interaction de deux systèmes à événements discrets comme l'illustre la Figure 1.11, où l'un des systèmes à événements discrets est le contrôleur et l'autre représente le procédé couplé avec l'interface. L'approche est basée sur le principe de partition de l'espace d'état continu.

La restriction caractéristique de ce type de modèle se situe au niveau de l'interface "contrôleur vers procédé". En effet, les valeurs de la commande continue sont sélectionnées par le contrôleur dans un nombre fini de valeurs constantes. Les transitions entre différentes régions de l'espace d'état permettent de définir l'interface "procédé vers contrôleur".

Dans la classe des systèmes hybrides considérée, les dynamiques continues sont modélisées par un ensemble d'équations différentielles de la forme

$$\begin{aligned} \dot{x}(t) &= f(x(t), u(t)) \\ y(t) &= g(x(t)) \end{aligned}$$

où $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$ et $y \in \mathbb{R}^p$ sont les vecteurs d'état, d'entrée et de sortie du système et $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ et $g : \mathbb{R}^n \to \mathbb{R}^p$ sont deux fonctions continues en x qui modélisent la dynamique du procédé.

Le *contrôleur* est un système à événements discrets modélisé par un automate déterministe qui reçoit, manipule et restitue les événements représentés par les symboles $\tilde{x}[n]$.

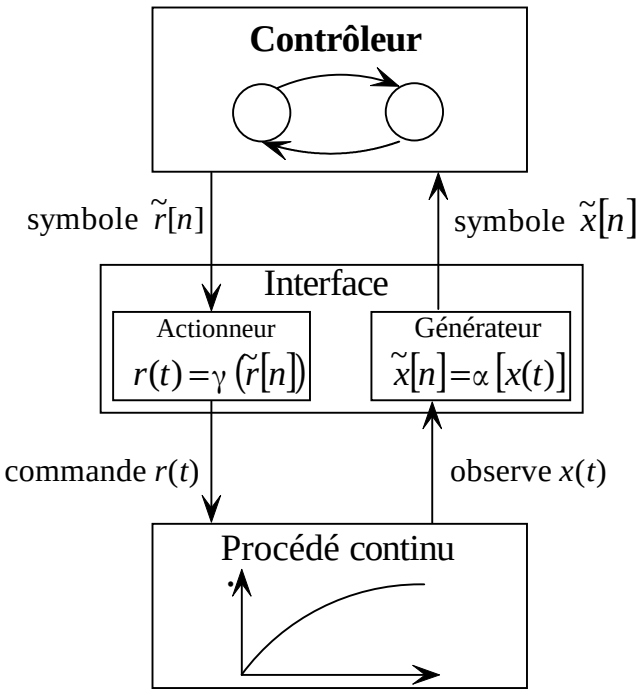


FIG. 1.11 – Modèle d'Antsaklis

Le rôle de l'*interface* est de générer des événements fournissant à la partie discrète une information provenant de l'évolution continue du système, et respectivement, de transmettre les informations provenant du contrôleur au procédé. L'interface est donc constituée de deux sous-systèmes : un *générateur d'événements* qui observe la sortie $x(t)$ du procédé pour générer ensuite les événements $\tilde{x}[n]$ et l'*actionneur* dont le but est de transformer les commandes symboliques $\tilde{r}[n]$ provenant du contrôleur en un signal $r(t)$ constant par morceaux pour l'entrée de commande du procédé.
Le *générateur d'événements* est défini par une application $\alpha : \mathbb{R}^n \times \mathbb{R} \rightarrow \tilde{X}$. La difficulté est de déterminer quand un symbole du procédé doit être généré et quel symbole $\tilde{x}[n]$ doit être transmis au contrôleur. Les auteurs résolvent ce problème introduisant un ensemble de fonctions h_i définies sur l'espace d'état du procédé continu : $\{h_i : \mathbb{R}^n \times \mathbb{R}, i \in I\}$. Un événement se produit lorsque l'état du procédé traverse l'une de ces frontières h_i, c'est-à-dire lorsque la condition suivante est vérifiée :

$$\exists\, i \in I \quad \text{tel que} \quad h_i(x(t)) = 0 \quad \text{et} \quad \tfrac{d}{dt} h_i(x) \neq 0$$

En d'autres termes, l'interface partitionne l'espace d'état continu en régions dépendantes des spécifications du système et délimitées par les frontières h_i.
L'*actionneur* est défini par l'application $\gamma : \tilde{R} \rightarrow \mathbb{R}^m$. L'entrée du procédé est un signal $r(t)$ constant par morceaux qui ne change que lorsqu'il y a un symbole $\tilde{r}[n]$ provenant du contrôleur.
L'approche purement discrète consiste à associer le procédé continu et l'interface. Le système à événements discrets qui représente cette association est appelé *modèle discret du procédé continu*. Modéliser le système de cette façon permet aux auteurs d'appliquer à cette classe de systèmes hybrides les techniques développées pour l'analyse des systèmes à événements discrets. Ainsi, cette approche permet de se rapprocher d'un problème de supervision discrète dont l'analyse peut être réalisée selon la théorie de Ramadge et Wonham [RW89].

À chacune des régions de l'espace d'état continu est associé un état discret $\tilde{p}$. À chaque franchissement de frontière définissant la partition est associé un symbole du procédé. Le comportement de l'automate modélisant le procédé continu est caractérisé par l'ensemble des séquences de symboles qu'il peut générer. Le problème majeur rencontré dans cette méthode est que l'automate obtenu est souvent non déterministe. La partition arbitraire de l'espace d'état peut également engendrer une mauvaise approximation de la dynamique continue. Le générateur doit accomplir deux objectifs : il doit donner au contrôleur suffisamment d'informations pour identifier si l'état courant est ou non dans une région "acceptable" et pour permettre au contrôleur de commander le procédé dans une telle région.

Nerode et Kohn [NK93] présentent une approche par automates, conceptuellement similaire au modèle d'Antsaklis, où le modèle est conçu par l'interaction entre un automate à états fini et un système d'équations différentielles décrivant le procédé continu. L'automate reçoit des informations événementielles sous forme de symboles à intervalles de temps irréguliers, alors que pour la partie continue du modèle une équation différentielle reçoit des entrées en temps réel. Une synchronisation entre les deux parties est envisagée. Le modèle proposé est composé de trois parties : le procédé, l'automate et l'interface.
L'interface se trouve sous forme d'un convertisseur Analogique/Numérique (AN) pour convertir les mesures continues en symboles constituant le langage d'entrée de l'automate, qui génère en sortie des symboles de commande convertis par un convertisseur Numérique/Analogique (NA) en commandes continues par morceaux appliquées en entrée du procédé.

Cette approche, comme la précédente, repose sur le principe de découpage de l'espace d'état continu, en plaçant des capteurs de mesures aux points singuliers du système.

Synthèse d'un contrôleur basé sur la notion d'invariants naturels du procédé

Dans [SAL95] une méthode permettant de résoudre le problème de non déterminisme rencontré dans les cas précédents est donnée. La méthodologie proposée dans cet article présente la construction du contrôleur, l'interface n'étant pas donnée dans ce cas. Cette méthode est basée sur la notion d'*invariants naturels du procédé*.

La particularité d'une telle approche est que chaque objectif de commande est donné par un ensemble de départ et un ensemble cible, chacun d'entre eux étant un sous-ensemble ouvert de l'espace d'état du procédé. Pour atteindre l'objectif, le contrôleur doit être capable de conduire le procédé depuis n'importe quel état de l'ensemble de départ vers un état de l'ensemble cible en utilisant une stratégie de commande possible. Le problème est posé en terme d'atteignabilité.

La solution de ce problème de commande est d'identifier, pour une région cible donnée, les états qui peuvent être conduits dans cette région par application d'un seul symbole de commande. Si la région de départ est contenue dans cet ensemble d'états, l'objectif est atteignable par ce symbole de commande particulier. Sinon, cet ensemble d'états devient le nouvel ensemble cible et le processus est répété. Pour décrire les régions mentionnées, le concept de *flux* est introduit. Le flux d'un procédé est déterminé par la fonction F_k :

$$F_k : X \times \mathbb{R} \rightarrow X \qquad x(t) = F_k(x(0), t)$$

Ce flux représente l'état du procédé au temps t avec un état initial $x(0)$ et une entrée de commande constante $\gamma(\tilde{r}_k)$. Pour une région cible T, ${F_k}^{-}(T)$ est l'ensemble des états initiaux depuis lesquels le procédé peut atteindre T avec une entrée $\gamma(\tilde{r}_k)$.
À partir de ces définitions, les auteurs proposent une procédure de construction du

générateur. La difficulté à identifier l'ensemble du flux qui atteint une région donnée amène les auteurs à ne déterminer qu'un sous-ensemble de $F_k^-(T)$ correspondant à une région B dans laquelle toutes les trajectoires atteignent la région cible T pour un symbole de commande particulier. Le problème revient à déterminer les frontières qui bornent ces ensembles .

Par cette technique de construction du générateur, la conception du contrôleur est anticipée par la détermination de l'interface. Un contrôleur est construit pour une région cible donnée, cela signifie qu'à chaque objectif de commande particulier est associé un automate représentant le contrôleur. Chaque automate est obtenu en associant un état discret du contrôleur à une région déterminée par le découpage de l'espace d'état du procédé présenté précédemment et constitue la première étape de la démarche. La seconde étape consiste à relier entre eux tous les automates, chacun correspondant à un objectif particulier, et cette opération dépend directement de l'ordre dans lequel on désire atteindre chacune des régions cibles.

1.3.3 *Approche de modélisation mixte*

Dans les sections précédentes, des approches d'intégration des aspects hybrides dans des modèles continus ou événementiels ont été présentées. Cependant, dans la structure de ces modèles l'interaction entre la partie continue et événementielle n'est pas représentée explicitement.

L'approche mixte repose sur la supposition que le fonctionnement d'un système hybride est une séquence de deux phases. La première étape correspond à une transformation de l'état continu décrite en terme de temps écoulé durant cette phase. Dans la seconde étape, l'état est soumis à un changement discret instantané. Ainsi, les modèles développés dans le cadre de cette approche reposent sur l'interaction de deux sous-modèles, l'un pour les aspects événementiels, basé sur les automates à états finis, les réseaux de Petri ou des extensions de ces formalismes, et l'autre, formalisé par des équations d'état pour les aspects continus. Chacun des aspects, continu ou événementiel, est ainsi décrit sous une forme classique et l'aspect hybride est clairement pris en compte dans l'interface entre les deux sous-modèles. L'aspect événementiel influe sur le modèle continu en validant certaines des équations continues en fonction de l'état discret actif et l'aspect continu agit sur le modèle événementiel en validant ou en forçant le franchissement de certaines transitions.

Parmi les outils de modélisation résultant de cette approche mixte on retrouve : les *automates hybrides* ([ACHH93], [NOSY93]) représentant le modèle formel fondamental de cette approche ; les *statecharts hybrides* pour apporter des solutions aux problèmes posés par la spécification des modèles, en particulier par l'utilisation de la structuration hiérarchisée [KP92] ; les différents extensions des réseaux de Petri.

Modèle de l'automate hybride

Dans [ACHH93], les auteurs définissent l'automate hybride comme une extension de l'automate discret en associant une évolution continue à chaque état discret. La composante continue est décrite par un ensemble d'équations différentielles et la composante discrète par un automate à états fini. Un automate hybride évolue par une alternance de pas continus, où les variables d'état et le temps évoluent de façon continue, et de pas discrets où plusieurs transitions discrètes et instantanées peuvent être franchies. Il s'agit

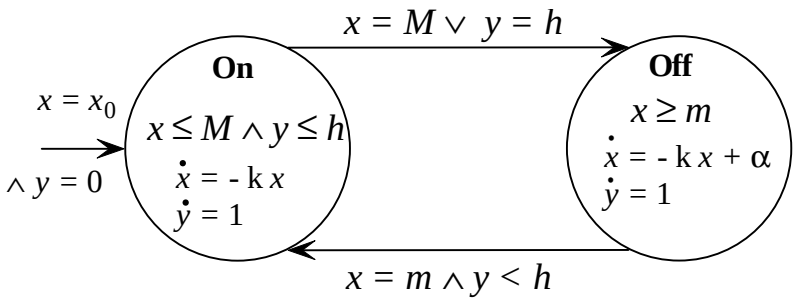


FIG. 1.12 – Automate hybride

d'une extension des automates temporisés [AD94], où la dynamique continue n'est plus représentée par des horloges mais par des équations différentielles quelconques.

D'un point de vue informel et général, un automate hybride apparaît ainsi comme un automate à états fini pilotant un ensemble d'équations différentielles modélisant la dynamique continue du système. Le modèle est composé d'un ensemble fini des variables réelles X et d'un graphe d'événements étiqueté (S, E). L'ensemble S est composé par les sommets du graphe et les éléments de l'ensemble E représentent les transitions discrètes. L'état de l'automate change instantanément lors de l'occurrence d'un événement discret ou par l'écoulement du temps lors de la validation d'une condition logique spécifiée sur la valeur de la variable continue [Hen96].

Considérons l'automate représenté dans la Figure 1.12 modélisant un système hybride. Dans ce modèle, l'évolution continue est représentée par des équations différentielles associées aux sommets du graphe et l'évolution événementielle est modélisée par les arcs étiquetés du graphe.

Les sommets *On* et *Off* représentent les *états discrets* du système où l'évolution continue a lieu. Les prédicats $(x = M \vee y = h)$ et $(x = m \wedge y < h)$ sur les arcs traduisent les *conditions pour l'occurrence d'un événement*. Les prédicats $(x \leq M \wedge y \leq h)$ et $x \geq m$ dans les sommets représentent les *invariants* de l'automate, c'est-à-dire, des conditions imposées aux variables continues du système pour rester dans un état discret (ici les états *On* ou *Off*). L'*état initial* du système est représenté par un arc d'entrée dans le sommet d'origine. L'étiquette de cet arc $x = x_0 \wedge y = 0$ représente la région de l'espace continue à partir de laquelle la dynamique du système hybride démarre.

Ce modèle représente le modèle de base de notre travail de recherche. Une description plus détaillée et formelle de cette approche sera donnée dans le chapitre suivant.

Réseaux de Petri hybrides

Les réseaux de Petri [Pet62] ont été très utilisés comme outils de modélisation, analyse et synthèse pour les systèmes à événements discrets. Dans [BAD91], les auteurs présentent une extension des réseaux de Petri (RdP), les réseaux de Petri hybrides.

Un *RdP hybride* est composé par des places et des transitions continues (C-places et C-transitions) et des places et transitions discrètes (D-places et D-transitions). Le marquage d'une C-place est représenté par un nombre réel, dont l'unité est appelée marque, et le marquage d'une D-place est représenté par un nombre entier dont l'unité est appelée jeton.

Le réseaux de Petri hybride de la Figure 1.13 modélise un système de fabrication qui produit des pièces par lots de 5. À la fin de la production de 2 lots de 5 pièces, un nouveau cycle de production est entamé.

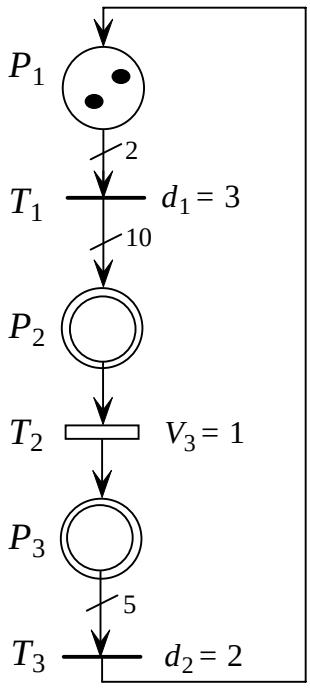


FIG. 1.13 – Modèle de RdP hybride d'un système de fabrication par lots

Le marquage de la place P_1 (D-place) est associé au nombre de lots à l'entrée du système de fabrication. Les durées d_1 et d_2 sont les temps de chargement et déchargement des pièces (D-transitions). La transition T_3 modélise une machine dont la vitesse de production est V_3 (C-transition). Cette machine possède un stock d'entrée et un stock de sortie modélisés par les places P_2 et P_3 (C-places). Le franchissement continu de la transition T_3 correspond à une production continue à la vitesse V_3 quand la place P_2 n'est pas vide. Lorsque P_2 est marquée, le franchissement d'une quantité $V_3\,dt$ de T_3 correspond à retirer $V_3\,dt$ marques à P_2 et à ajouter la même quantité à P_3.

Ce modèle hérite tous les avantages du modèle de réseaux de Petri tel que la représentation du parallélisme, de la synchronization et des conflits (David et Alla, 2001).

D'autres approches autour des RdP ont été proposées. Parmi ces approches, Petterson et Lennartson [PL95] utilisent les "bond graphs" pour la vérification de systèmes représentés par des RdP hybrides. Il s'agit d'étendre la représentation à des objets événementiels tout en gardant les propriétés des graphes. Les "*High-Level Petri Nets*" sont proposés par Giua et Usai [GU96], caractérisés par l'utilisation de jetons colorés dans la partie discrète du réseau.

Pour l'analyse quantitative des réseaux de Petri hybrides périodiques, dans [All98] l'auteur propose un algorithme permettant de construire l'automate hybride décrivant l'évolution d'un réseaux de Petri hybride.

1.4 Modélisation en vue de l'analyse et de la synthèse de la commande

La modélisation, comme nous venons de la présenter, consiste à proposer des modèles précis qui peuvent décrire le comportement riche et complexe des systèmes hybrides. En s'appuyant sur les modèles ainsi obtenus, l'*analyse* porte sur le développement d'outils de

simulation, de validation et de vérification des SDH. Les problèmes rencontrés sont liés à la complexité de cette analyse et l'interprétation de certaines propriétés à analyser.

La *commande* consiste à effectuer la synthèse des contrôleurs discrets ou hybrides (ayant des sorties continues et la capacité de prendre des décisions discrètes) conformément à certains objectifs de performance et de sûreté du procédé hybride commandé. Une partie des approches de la commande des SDH est basée sur le principe de recherche d'une stratégie discrète qui permette de restreindre le comportement du système pour satisfaire certaines spécifications imposées. L'autre partie portent sur la formulation et la résolution d'un problème de commande optimale ou des extensions de la théorie de Lyapunov.

1.4.1 *Analyse des SDH*

L'approche la plus générale pour l'analyse des SDH consiste à simuler leurs modèles. Dans cette optique, plusieurs méthodes et outils de simulation ont été développés. L'intérêt de la simulation est qu'elle s'applique à des nombreux modèles, mais c'est une approche expérimentale qui ne permet pas de considérer toutes les évolutions et ne garantit donc pas l'absence d'erreurs dans le modèle établi. Par conséquent, d'autres approches plus formelles sont développées afin de caractériser l'ensemble des trajectoires d'un système hybride donné. En parallèle, des travaux sur les propriétés structurelles des modèles, telles que la stabilité, sont développés. Un ensemble de travaux s'intéresse à la vérification formelle qui permet de garantir que dans toutes les exécutions possibles du SDH modélisé un certain nombre de propriétés sont satisfaites ou non.

Pour les systèmes dynamiques, on distingue classiquement trois types de propriétés comportementales : la *sûreté* qui exprime les configurations qui ne doivent jamais être réalisées, la *vivacité* qui exprime des évolutions qui doivent être possibles ou inévitables et le *temps de réponse* qui exprime des contraintes sur le temps minimum ou maximum qui sépare certains événements ou caractérise certaines évolutions. Parmi ces trois propriétés qui peuvent être vérifiées, pour les systèmes hybrides les propriétés prépondérantes sont celles de sûreté, exprimées généralement par des sous-domaines de l'espace d'état hybride.

Pour les systèmes hybrides modélisés par des automates hybrides, il existe deux approches pour appliquer les techniques de vérification. Une première approche repose sur le calcul de l'espace atteignable à partir d'un état initial du système. Une seconde approche consiste à construire un modèle événementiel qui est une abstraction du comportement du système hybride et à mener les calculs de vérification sur ce modèle événementiel avec des outils classiques pour ce type de système.

Calcul de l'espace atteignable

Un algorithme de vérification des SDH est basé généralement sur le calcul de l'espace atteignable à partir d'un état initial ou, à l'inverse, le calcul des états à partir desquels il est possible d'atteindre un sous-domaine particulier. En effet, pour prouver que l'état reste toujours dans un domaine particulier il suffit de calculer l'espace atteignable et de vérifier qu'il est bien inclus dans le domaine spécifié. De même, pour vérifier que l'état n'atteint jamais un domaine particulier il suffit de calculer l'espace atteignable à partir de l'état initial et de vérifier que son intersection avec le domaine spécifié est vide ou de calculer l'ensemble des états à partir desquels le domaine interdit est accessible et de vérifier que l'état initial n'en fait pas partie [ACH+95].

La difficulté majeure liée à cette approche se trouve dans le calcul effectif de l'espace atteignable. À l'exception de quelques cas particuliers, il s'agit d'un problème pour lequel

les algorithmes de calcul peuvent ne pas converger. D'un point de vue théorique cette non convergence s'explique par le fait qu'il est prouvé que le calcul de l'espace atteignable par un automate hybride est en général un *problème non-décidable* ([ACH+95], [HKPV95], [PV94]), c'est-à-dire qu'il n'est pas possible de trouver un algorithme qui réalise ce calcul et dont la convergence est assurée en un temps fini.

Les cas particuliers où la convergence de l'algorithme peut être garantie appartiennent principalement à la classe des *automates hybrides linéaires* [ACH+95]. L'intérêt de ces modèles est que toutes les régions de l'espace d'état continu qu'ils définissent (invariants, gardes) sont des régions linéaires convexes et que les images de régions linéaires par leurs fonctions d'évolutions continue et discrète sont également des régions linéaires [HR98].

La frontière entre les classes des systèmes hybrides pour lesquelles le problème d'atteignabilité est décidable et celles pour lesquelles il est non-décidable n'est pas complètement définie. Une conséquence de ces résultats théoriques est que le calcul exact de l'espace atteignable n'est pas toujours possible et dans la pratique on se contentera d'un calcul approché. Dans la littérature il existe plusieurs travaux concernant le calcul de l'espace atteignable par des méthodes approximatives. Même si le calcul approché prend différentes formes, les approximations mises en œuvre visent toutes à surestimer l'espace atteignable. La plupart de ces approches sont basées sur le calcul des polyèdres pour la vérification de certaines classes des automates hybrides.

Dans [HHWT95], l'application HyTech est présentée, en réponse à un besoin de méthodes formelles automatisées pour la vérification des systèmes hybrides pour lesquels l'aspect temporel joue un rôle fondamental dans leurs comportements. S'appuyant sur la bibliothèque de calculs symboliques sur les polyèdres, HyTech peut manipuler des modèles incomplètement spécifiés dans lesquels certaines valeurs sont définies comme des paramètres. Il permet, ainsi, non seulement de vérifier les propriétés, mais aussi d'apporter une assistance à la conception en précisant les conditions qui assurent que la propriété est vérifiée.

Dans [HH95], une méthode de calcul approximatif consistant, dans un premier temps, à remplacer un ensemble des polyèdre par leur enveloppe convexe est présentée. L'enveloppe convexe est définie comme le plus petit polyèdre convexe dans lequel tous les polyèdres du départ sont inclus. La deuxième opération utile est celle d'élargissement, qui consiste à retirer des limites d'un polyèdre. L'opérateur d'élargissement s'avère très puissant pour accélérer la convergence des calculs itératifs (comme le calcul de l'espace atteignable), puisqu'en remplaçant l'élargissement d'une région obtenue à une itération par celle obtenue à l'itération suivante, la convergence vers une surestimation du résultat en un nombre fini d'itérations est garantie. Cependant, les surestimations auxquelles ils conduisent ne permettent pas toujours de conclure sur la validité ou non d'une propriété. Leur manipulation demande un savoir-faire certain de la part de l'utilisateur qui doit souvent mettre en œuvre une stratégie associant les calculs directs et inverses avec les différents opérateurs d'approximation pour mener à bien la vérification.

Dans certains cas, il n'est pas possible de calculer de manière exacte l'espace d'état atteignable car le calcul ne converge pas, alors qu'il est possible d'en donner une expression analytique. Une autre approche présentée par Henzinger et Rusu [HR98], consiste alors à "deviner" une expression caractérisant le domaine atteignable ou le terme de l'itération de son calcul, puis à prouver que cette expression est correcte. Les difficultés liées à une telle preuve d'atteignabilité sont : (1) il faut deviner la région atteignable, ce qui se fait en général à partir des premières itérations du calcul d'atteignabilité, et (2) il faut prouver soit la récurrence, soit l'invariance de la région, ce qui peut en général être automatisé.

Les techniques de calcul ou de preuve présentées ne s'appliquent pas que pour la classe

particulière des automates hybrides présentée.

Pour la vérification des automates hybrides, dont la dynamique continue est plus complexe que celles des automates linéaires, il existe plusieurs approches. Une approche consiste à calculer une abstraction linéaire de l'automate du départ, c'est-à-dire à trouver l'automate linéaire dont le comportement contient celui de l'automate de départ, afin d'obtenir une surestimation de l'espace d'état atteignable [HHWT98], [PV95]. Cette approche repose sur la transformation de l'automate hybride soit en un automate temporisé soit en un automate linéaire dont la dynamique continue est limitée pour chaque variable d'état continue à l'inclusion de sa dérivée dans un intervalle. Ces techniques de transformation sont intéressantes puisqu'elles permettent d'utiliser les résultats et les outils développés dans le cadre des automates linéaires.

Une autre approche consiste à utiliser le calcul sur les intervalles pour obtenir une surestimation de l'espace atteignable directement à partir de l'automate de départ [HHMWT00]. Cette approche est basée sur le calcul numérique et l'idée est en fait que les approximations introduites par le calcul numérique sont plus faibles que celles issues de l'abstraction. La difficulté d'utiliser le calcul numérique pour la vérification est de garantir que le résultat est bien une surestimation de l'espace atteignable afin de pouvoir conclure au respect de la propriété. Cette garantie est apportée par le calcul sur les intervalles ([Moo96],[Rih94]). En effet, les méthodes de calcul manipulent des intervalles et non des nombres réels et fournissent comme résultat non pas une valeur avec des erreurs arrondies mais un intervalle dont on peut garantir qu'il contient la valeur exacte du résultat.

Dans [Dan00], une approche de vérification basée sur l'approximation de la dynamique continue par des polyèdres orthogonaux [BMP99] est présentée. L'auteur présente deux méthodes pour le calcul de l'espace atteignable, une pour les systèmes linéaires et l'autre pour les systèmes avec des dynamiques plus complexes. Une technique de vérification de la propriété de sûreté est aussi présentée pour une classe large des systèmes hybrides. L'algorithme de vérification proposé peut être utilisé pour les systèmes hybrides multi-dimensionnels due à la représentation des régions atteignables par polyèdres.

Abstraction par des systèmes événementiels

Une autre approche de vérification de propriétés de systèmes hybrides consiste à calculer une abstraction purement discrète pour utiliser les méthodes et les outils de vérification des systèmes à événements discrets. L'idée est de construire une abstraction du modèle de départ, c'est-à-dire d'obtenir un automate qui modélise toutes les trajectoires autorisées par le modèle initial, afin de garantir que toute zone de l'espace d'état continu atteignable correspondra à une zone de l'espace discret. Ainsi, toutes les propriétés de sûreté prouvées sur le modèle discret seront garanties sur le modèle hybride de départ.

Le problème pour obtenir un modèle purement discret d'un système hybride réside dans l'abstraction du modèle continu. L'approche la plus directe est présentée dans [PBVV96] et consiste à découper l'espace d'état continu en pavés rectangulaires, associés, chacun, à un état discret. La difficulté principale est alors de trouver un compromis entre la précision et le nombre d'états discrets par exemple en appliquant le concept de grille à des sous-espaces de l'espace d'état continu et en particulier aux surfaces de commutation du système hybride [CK98].

1.4.2 *Synthèse et commande des SDH*

Contrairement aux systèmes continus ou à événements discrets 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 discrète, 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 permette de restreindre la fonction de transition du système pour satisfaire les spécifications ([ACH+95], [AM99]).

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, ce modèle est un modèle de base pour la commande des procédés batch. La notion de contrôlabilité d'un système hybride est définie à partir de l'existence d'une loi de commande qui amène 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].

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.

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

des systèmes type AHS (Automated Highway Systems).

1.5 Conclusions et objectifs de notre travail

Les systèmes dynamiques hybrides sont des systèmes qui combinent une partie discrète et une partie continue. Récemment, ces systèmes ont reçu beaucoup d'attention et plusieurs formalismes ont été proposés afin d'établir un modèle homogène permettant la modélisation de l'interaction entre les parties discrètes et continues.

Dans ce chapitre, nous avons passé en revue les principales approches de modélisation, analyse et commande des systèmes dynamiques hybrides.

Parmi ces approches, l'approche mixte est celle qui considère les comportements continus et événementiels dans une même structure. L'avantage de cette approche est sa généralité, car elle ne fait pas d'hypothèse sur le type de phénomène à modéliser, laissant ainsi toute liberté à l'utilisateur pour construire son modèle. En particulier, elle n'impose pas des contraintes sur le type de systèmes à modéliser et peut être ainsi utilisée pour construire un modèle composé d'une partie correspondant à la partie opérative du système et d'une autre à la commande. Dans notre travail nous nous sommes particulièrement intéressés à l'approche consistant à modéliser le système par un automate hybride [Hen96].

Une fois le modèle du système obtenu, le problème de la validité du modèle se pose. Étant donnée la liberté de description de modèles, la validité du modèle ne peut pas être garantie par sa construction et elle doit donc être vérifiée. Ainsi, on retrouve la problématique de l'analyse.

La vérification des systèmes hybrides repose sur le calcul de l'espace atteignable à partir d'une région initiale pour déterminer si l'intersection de cet espace avec la région à éviter est bien vide. Pour les automates hybrides linéaires, la résolution de ce problème nécessite des outils de calcul basés sur les polyèdres. Mais le calcul exact n'est pas toujours possible et des opérateurs d'approximation doivent être utilisés. Lorsqu'on s'intéresse à des systèmes hybrides dont la dynamique est plus complexe le problème de la vérification devient plus complexe.

Un premier travail a été présenté dans [Dan00]. L'approche est basée sur l'approximation de la dynamique continue par des polyèdres orthogonaux. L'algorithme de vérification proposé peut être utilisé pour les systèmes hybrides multi-dimensionnels du à la représentation des régions atteignables par polyèdres. La méthode présentée fournit comme résultat une surapproximation de la région atteignable par l'évolution du système. L'utilisation du temps continu pour la modélisation de la dynamique continue implique une grande complexité de calcul.

Dans notre travail, nous proposons une méthode exacte de calcul de l'espace atteignable, basée sur la représentation en temps discret de la dynamique continue. L'approche proposée sera présentée dans le troisième chapitre de ce mémoire.

Une autre difficulté est liée à la synthèse de la commande. Des méthodes efficaces permettant de résoudre ce problème ont été développées pour les systèmes continus et les systèmes à événements discrets. Le fait que les systèmes hybrides fassent intervenir deux types de dynamiques, une dynamique continue et une autre discrète, implique une complexité plus importante du problème de synthèse pour cette classe de systèmes.

Le problème général de *synthèse de la commande* peut être formulé comme suit : étant donné un système dynamique hybride, trouver une manière systématique d'intervenir dans son évolution telle que les contraintes imposées sur les variables continues soient toujours

vérifiées.

Dans la littérature, la plupart des approches s'adressent à une classe assez restreinte de systèmes hybrides. Notamment, à des systèmes hybrides dont les fonctions d'évolutions continues sont supposées linéaires découplées. Ainsi, le problème de la synthèse de la commande reste toujours ouvert.

L'*objectif* de notre travail est d'apporter une solution originale au problème de la synthèse optimale d'un modèle de commande pour les systèmes hybrides, basée sur une extension de la théorie classique de la commande supervisée développée par Ramadge et Wonham pour les systèmes à événements discrets. Les évolutions non désirées du système seront modélisées par des états interdits et le but sera de supprimer toutes les trajectoires qui mènent le système vers ces états. Une présentation détaillée de cette approche sera faite dans le dernier chapitre de ce mémoire.

Chapitre 2

Structure générique et classe d'application considérée

Dans ce chapitre nous proposons de faire apparaître une structure générique des systèmes hybrides qui synthétise ses principales caractéristiques comportementales.

L'introduction de ce chapitre est constituée par une discussion sur la structure commune des différentes approches présentées dans le chapitre précédent.

La structure générique d'un systèmes hybride et le formalisme de modélisation, basé sur l'outil automate hybride, seront d'abord présentés. Nous passons ensuite à la description de la classe particulière des systèmes qui nous intéresse, notamment les systèmes de production. Dans ce contexte, la nature hybride de ces systèmes sera discutée en mettant en évidence les aspects continus et discrets qui déterminent leur évolution.

La modélisation par l'outil automate hybride sera illustrée sur un exemple simple, choisi dans le domaine manufacturier, représenté par une ligne de production. Les modèles ainsi obtenus constituent le point de départ pour l'approche d'analyse d'atteignabilité que nous proposerons dans le chapitre suivant.

2.1 Introduction

Dans le chapitre précédent nous avons présenté l'état de l'art sur les différentes approches de modélisation existant dans la littérature, à savoir :

- l'approche événementielle en vue de la supervision discrète est une méthode qui fait appel à la théorie de supervision de Ramadge-Wonham ;
- l'approche continue basée sur la définition d'un ensemble de transitions est une méthode plus générale capable de prendre en compte une grande partie des phénomènes hybrides existant. De ce fait, elle n'est pas forcément la plus adaptée à des systèmes qui présentent un nombre important de commutations discrètes ; et
- l'approche mixte qui combine les parties continues et discrètes dans une même structure est une méthode qui se veut être une méthode orientée vers la simulation et la vérification du fonctionnement du système.

Ainsi, dans le cas de l'*approche événementielle* l'idée repose sur la partition de l'espace d'état continu telle que ses dynamiques soient représentées d'une manière significative (c'est-à-dire, utile pour l'analyse) par un automate à états finis. Le problème particulier de la synthèse du modèle discret d'un procédé continu est de déterminer les hypersurfaces permettant le découpage de l'espace d'état continu qui sont utilisées pour générer les symboles du procédé. Les événements étiquetant les arcs du graphe, seront générés par l'interface, qui relie les parties continues et discrètes du système, à l'instant où les frontières délimitées par les hypersurfaces sont franchies.

L'*approche continue* s'adresse aux systèmes hybrides dont les phénomènes discrets subis par le procédé sont de type commutations autonomes et/ou contrôlées. Le système est supposé être défini par un ensemble de systèmes dynamiques continus indexés dont le passage de l'un à l'autre est géré par une fonction de commutation. Un système ainsi défini peut être illustré par un automate, où chaque sommet est constitué par un système dynamique et chaque arc, étiqueté par une condition de transition appropriée. Une relation de réinitialisation éventuelle de l'état continu, représente la transition entre deux systèmes dynamiques.

Une structure commune de ces approches apparaît clairement tant dans la représentation graphique utilisée que dans la formulation mathématique. Dans la *représentation graphique*, la structure représentative est celle d'un graphe d'état. Par cette structure, les différentes combinaisons possibles entre les entrées du système dynamique hybride considéré, correspondant à des événements spontanés ou à des commandes, sont modélisées. Ce qui différencie toutes ces approches, c'est la manière de déterminer les transitions du graphe d'état. Par conséquent, l'objectif le plus important est d'utiliser au mieux l'information continue disponible pour la combiner à celle événementielle déjà contenue dans cette structure de référence.

Le modèle le plus général, issu de l'*approche mixte*, est le modèle de l'*automate hybride* qui combine les parties continues et discrètes dans une même structure. Les sommets du graphe contiennent les informations sur l'état continu et discret du système et les arcs reliant les sommets sont franchis lorsqu'une condition spécifiée sur les valeurs des variables continues et/ou discrètes est vérifiée. Par conséquent, la structure du modèle est celle d'un graphe de transitions autour duquel il y a un certain nombre d'éléments structurels et topologiques permettant de lier les comportements continus et événementiels. Dans notre travail nous nous intéressons plus particulièrement à la modélisation d'un système hybride en utilisant ce modèle.

Étant donné l'intérêt de notre travail pour la synthèse d'une commande supervisée pour les systèmes hybrides, la description de la classe d'application considérée va toujours

se faire en terme de système à commander, notamment le *procédé*, et ses *spécifications de fonctionnement*.

Dans la suite de ce chapitre, nous proposons de faire apparaître d'abord la structure générique d'un SDH. Les caractéristiques de cette structure justifieront le choix du modèle d'*automate hybride* qui sera introduit d'une manière formelle. Nous passons ensuite à la description de la classe particulière de systèmes qui retient notre attention, notamment les *systèmes de production*. La nature hybride de ces systèmes sera discutée en mettant l'accent sur les aspects discrets et continus qui régissent son fonctionnement. Les modèles mathématiques directement liées à la notion de système et qui sont à la base de la modélisation seront présentés pour quelques situations particulières, fréquemment rencontrées dans ces systèmes. En fin de chapitre, nous illustrons l'approche de modélisation par des automates hybrides sur un exemple choisi dans le domaine manufacturier.

2.2 Une structure générique des SDH

D'une manière générale, en vue de la commande, un système dynamique peut être décrit en terme de système à commander, représentant le *procédé*, et par ses *spécifications de fonctionnement*, décrivant le fonctionnement désiré du système en boucle fermée. Ci-dessous, ces deux composantes seront traitées séparément dans le but de faire apparaître les spécificités d'un système dynamique hybride.

2.2.1 *Le procédé*

Les systèmes dynamiques hybrides sont des systèmes dont le comportement dynamique est défini par l'interaction entre ses dynamiques continues et discrètes. Par conséquent, le procédé lui aussi possédera les deux aspects : l'aspect continu et l'aspect discret.

(a) *Aspects continus*

L'évolution dynamique d'un système est déterminée souvent par les processus physiques qui ont lieu. Ainsi, le modèle mathématique est obtenu à partir des propriétés physiques du système permettant de trouver une description de celui-ci sous la forme de représentation d'état.

Le modèle mathématique général est du type :

$$\dot{x}(t) = f(x(t), u(t), t) \tag{2.1}$$

où $u(t)$ représente le vecteur d'entrées et $x(t)$ est le vecteur des variables d'état. D'autres types de modèles peuvent être obtenus dans des situations particulières. Par exemple, si la modélisation est basée sur des modèles des sous-systèmes du système initial, les équations résultantes sont souvent de type algébro-différentielles.

Le modèle présenté par la relation 2.1 est assez général et l'analyse de ses propriétés doit être réalisée en temps continu. Pourtant, dans les applications de commande des procédés que l'on peut rencontrer dans la nature, des modèles plus simples peuvent être pertinents, tout dépend de la manière dont on génère l'entrée de commande. Par exemple, c'est le cas des systèmes où l'entrée de commande est contrainte d'être linéaire

par morceaux. Ainsi, la dynamique continue sera décrite par un ensemble d'équations différentielles simples (où aux différences) au lieu des représentations, souvent non-linéaires, complexes.

Par exemple, dans le cas des procédés batch il est naturel de spécifier des procédures globales contenant un nombre de phases du procédé. Chacune de ces phases détermine une évolution simple des variables observées. Dans ce contexte, les modèles décrivant le comportement du procédé seront de type :

$$\dot{x}(t) = r_k \qquad t_k \leq t < t_{k+1} \tag{2.2}$$

où r_k donne la variation de la variable x dans une phase du procédé et t_k donne les instants auxquels les commutations entre les phases du système ont lieu.

(b) *Aspects discrets*

D'une manière générale, l'évolution discrète d'un système dynamique hybride est déterminée par l'occurrence des événements d'origine externe ou interne dont la nature peut être contrôlable ou non. L'occurrence des événements implique des changements de la dynamique continue du système.

Dans les systèmes hybrides ces changements peuvent être déterminés soit par une structure particulière du procédé physique, soit par des entrées/sorties discrètes générées par différentes composantes du système.

Changements de dynamique continue dus à la structure du système

Dans le cas des systèmes réels, les variables d'état $x(.)$ (Relation 2.1) peuvent modéliser des variations de volume, de température ou bien de concentration. Le champ de vecteur $f(x(t))$ est, en général, une fonction continue mais parfois il peut présenter des discontinuités. Celles-ci reflètent des changements de la dynamique continue dus aux caractéristiques du procédé physique. Pour illustrer ce phénomène considérons un réservoir représenté par la Figure 2.1.

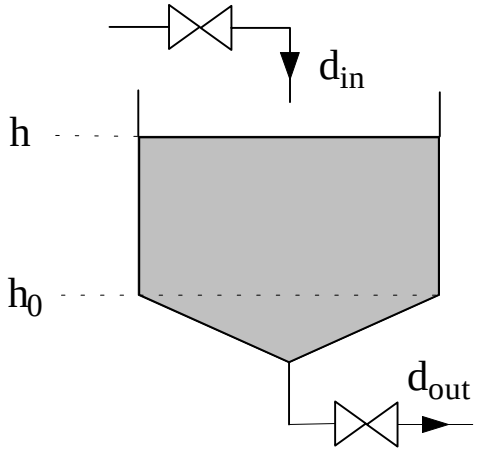


FIG. 2.1 – Exemple d'une structure physique particulière

La forme du réservoir implique un changement de modèle mathématique, qui modélise la variation du niveau de fluide dans le réservoir, lorsque le niveau est au-dessous du seuil

h_0. Ainsi, si le niveau dans le réservoir respecte la relation $h > h_0$, l'équation modélisant la vitesse de variation du niveau dans le réservoir est donnée par

$$S\dot{h}(t) = d_{in} - d_{out}$$

où d_{in} et d_{out} représentent les débits d'entrée et de sortie. Si $h \leq h_0$ alors le niveau dans le réservoir varie en conformité avec la dynamique décrite par

$$S\left(\frac{h(t)}{h_0}\right)^2\dot{h}(t) = d_{in} - d_{out}$$

Dans ce cas, les équations différentielles modélisant la variation de niveau dans le réservoir ne présentent pas de discontinuités, cependant dans la dynamique du système on distingue deux comportements différents. Pour différencier ces deux comportements, la solution repose sur l'utilisation d'une variable discrète associée à chaque dynamique continue.

Dans les procédés réels, cette distinction peut se faire en utilisant un capteur qui détecte le seuil où le changement de modèle intervient. Dans l'exemple considéré, pour distinguer les deux comportements du système, l'utilisation d'un capteur qui détecte le seuil h_0 peut fournir cette information.

Les changements de comportement dont l'origine se trouve dans la structure physique du système correspondent au phénomène de *commutation autonome*. L'utilisation des capteurs, pour modéliser d'une manière explicite les discontinuités introduites par la structure physique du système, implique naturellement l'occurrence d'un événement généré au moment où le changement du comportement continu du système intervient. De tels événements sont des *événements incontrôlables* dans le sens où leur occurrence ne peut pas être empêchée.

Changements de dynamique continue générés par des sorties discrètes

Dans le cas décrit ci-dessus, nous avons présenté une utilisation des capteurs qui permet la modélisation de discontinuités dans l'évolution du système.

Dans la plupart des systèmes réels, les capteurs sont utilisés pour signaler aux opérateurs les dépassements de certains seuils qui peuvent générer un fonctionnement non désiré du système. Ainsi, leur utilisation est étroitement liée à la manière dont les spécifications de fonctionnement du système sont formulées. Les informations qu'ils fournissent sont explicitement prises en compte lors de la synthèse d'un modèle de commande.

La remarque concernant la nature incontrôlable des événements générés par des capteurs en cas de détection d'un seuil, par exemple, reste toujours vraie.

Changements de dynamique continue générés par des entrées discrètes

Les éléments fréquemment utilisés dans les systèmes réels qui peuvent introduire des discontinuités dans leur fonctionnement sont les actionneurs discrets (par exemple, les vannes avec leurs états correspondants : ouvert ou fermé). En général, ce sont des éléments auxquels sont associées des fonctions de contrôle du système (par exemple, la commande manuelle/automatique de fermeture/ouverture de vannes).

Pour pouvoir intégrer des actionneurs discrets dans le modèle global d'un processus, il faut d'abord définir les concepts de modélisation qui leur correspondent. Les événements modélisant un changement d'état du système seront forcément des événements dont la nature est contrôlable pour pouvoir agir sur leur date d'occurrence.

Dans les cas précédents, les événements ont été introduits pour modéliser les changements de la dynamique continue du système. Ici, les événements sont utilisés pour modéliser les états des actionneurs. Il s'agit donc de modéliser les états discrets du système global et de définir les transitions entre ces états. Ceci permettra de construire un modèle global de fonctionnement du système.

2.2.2 *Les spécifications*

En général, les spécifications introduisent des restrictions dans l'évolution du procédé. Ainsi, pour formuler le problème de synthèse de la commande, il est nécessaire qu'après la description du procédé (système à commander), une description de ses spécifications de fonctionnement soient faite.

L'objectif de la synthèse consistera à restreindre l'évolution du procédé telle que le fonctionnement en boucle fermée du procédé couplé avec son système de commande respecte toujours les spécifications imposées.

Dans le cas des systèmes hybrides, les restrictions imposées par des spécifications du système peuvent être décrites en les divisant en deux groupes : les spécifications correspondant à la partie continue et les spécifications correspondant à la partie discrète du système.

(a) *Spécifications continues*

Les spécifications continues correspondent à des restrictions imposées sur les valeurs des variables d'état continues $x(.)$, exprimées par des conditions logiques qui limitent leur évolution à une certaine région de l'espace d'état, appelée *région désirée*. Toute évolution sortant de cette région est non désirable. Si nous utilisons des automates pour modéliser le système étudié, l'évolution non-désirée sera représentée par un sommet appelé *sommet interdit*.

Considérons la partie d'un automate hybride présentée dans la Figure 2.2.

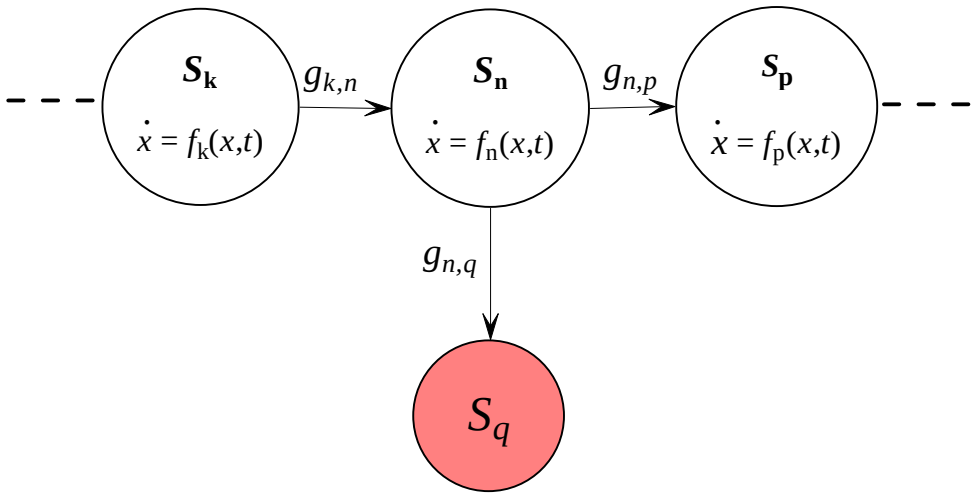


FIG. 2.2 – Partie d'un automate hybride

L'évolution dynamique de l'automate hybride a lieu par une alternance de pas discrets et continus. Ainsi, l'évolution continue a lieu dans les sommets de l'automate tandis que l'évolution discrète est réalisée par le franchissement des transitions (arcs) du graphe. Le

sommet interdit L_q est atteint depuis le sommet L_n par la validation de la transition étiquetée par $g_{n,q}$. Ainsi, la garde de la transition $T_{n,q}$ représentera la *région interdite*.

Définition 2.1. Un *sommet interdit* modélise la situation où les spécification continues du système ne sont plus vérifiées.

■

Exemple 2.1. Pour illustrer la notion de sommet interdit, reprenons l'exemple classique d'un thermostat utilisé pour maintenir la température du procédé dans un intervalle imposé (Exemple 1.1). Le système de chauffage sera arrêté ou redémarré en fonction des informations envoyées par le capteur de température du procédé. L'intervalle de température imposé, représente la spécification continue. Le dépassement des seuils de cet intervalle peut impliquer des défaillance dans le système, provoquées par exemple par le sur-chauffage des différentes composantes. Les états interdits permettent la modélisation d'une telle situation. Ainsi, le modèle automate où les évolutions non désirées sont représentées est illustré dans la Figure 2.3.

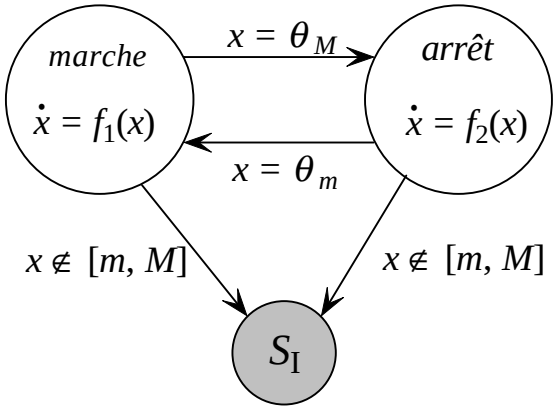


FIG. 2.3 – Modèle du thermostat avec les spécifications de fonctionnement

■

On observe que les contraintes imposées sur la valeur de la température, qui représente ici la variable d'état continue du système, doivent être respectées indépendamment de l'état du système de chauffage (en marche ou non). Ainsi, on peut conclure cette partie par :

Remarque 2.1. Les spécifications imposées aux variables d'état continues sont des contraintes globales du système.

■

Dans notre travail des situations de ce type seront considérées, mais nous pouvons imaginer des systèmes plus généraux auxquels sont associés des spécifications beaucoup plus complexes.

(b) *Spécifications discrètes*

La partie discrète d'un système hybride peut être vue comme une machine à états fini. En général, les spécifications discrètes sont données sous la forme de conditions logiques décrivant l'ordre d'occurrence des événements dans le système pendant son fonctionnement.

L'outil de modélisation permettant de prendre en compte tous les aspects présentés dans une même structure est l'*automate hybride*. Une présentation détaillée de cet outil est faite dans la section suivante de ce chapitre.

2.3 Automate hybride

Les exemples informels présentés dans le chapitre précédent (Exemples 1.1 et 1.2) et les aspects discutés jusqu'ici, nous permettent de constater que pour développer une approche d'analyse et de synthèse de commande pour les systèmes hybrides, il faut disposer d'outils de modélisation permettant de représenter l'interaction entre les dynamiques continues et discrètes, ainsi que les spécifications relatives au fonctionnement désiré du système. De plus, le modèle doit permettre de prendre en compte des comportements non-déterministes. De tels comportements peuvent être dus à des incertitudes dans l'évolution continue et discrète du système lors de l'occurrence des événements incontrôlables ou au manque de précision dans la connaissance du système.

Traditionnellement, les dynamiques continues sont modélisées par des équations différentielles (ou aux différences) et les dynamiques discrètes sont souvent modélisées et analysées par des automates.

L'*automate hybride* combine ces deux modèles et fournit un formalisme possédant les aspects cités ci-dessus. Dans un automate hybride, les dynamiques continues sont modélisées par des équations différentielles associées aux sommets et l'évolution discrète par le franchissement des transitions entre ces sommets.

2.3.1 *Syntaxe*

D'un point de vue informel, un automate hybride apparaît comme un automate à états fini pilotant un ensemble d'équations dynamiques continues. Les équations, modélisant le comportement continu à un instant donné, dépendent de l'état de l'automate mais ce dernier évolue en fonction de la valeur des grandeurs continues. Ainsi, l'automate hybride apparaît comme un système état-transition (Q, E) étendu avec un ensemble X des variables continues (Figure 2.4). L'ensemble fini Q est composé par des sommets représentant l'*état discret* du système où les évolutions continues ont lieu. L'ensemble fini E est composé par des arcs orientés modélisant les transitions discrètes qui relient les sommets du graphe. Tout arc orienté doit avoir un sommet destination.

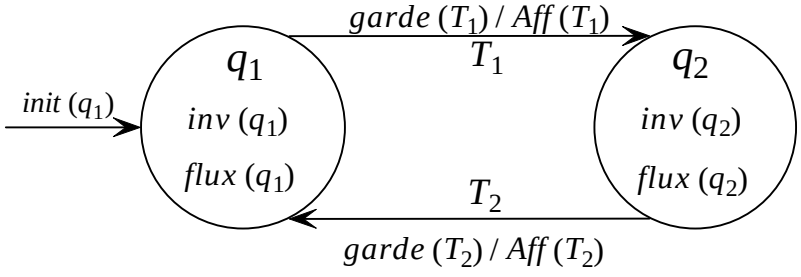


FIG. 2.4 – Automate hybride

Définition 2.2. Un sommet avec un arc d'entrée sans sommet source représente un *sommet initial*. Un sommet avec aucun arc de sortie est appelé *sommet puits*.

■

En reprenant la signification d'un sommet interdit, i.e. le sommet du graphe où l'évolution du système ne respecte plus les spécifications de fonctionnement imposées par le cahier de charges, et en prenant en compte la définition formulée ci-dessus, nous pouvons conclure que,

Remarque 2.2. Un sommet puits de l'automate hybride modélise un sommet interdit du système.

■

L'*état continu* du système est modélisé par un vecteur d'état continu $x(.)$ représentant un point dans un espace $X \subset \mathbb{R}^n$. Dans chaque sommet, la dynamique continue est modélisée par des conditions de flux telle que les équations différentielles (ou aux différences).

A chaque transition on associe un prédicat qui concerne l'état interne du système, appelé *garde*. Ce prédicat détermine les dates possibles pour le franchissement de la transition. Ainsi, une transition de l'automate hybride peut être franchie à l'instant t si et seulement si sa garde est vérifiée par la valeur des variables d'état continues du système à l'instant de temps considéré.

En général, la garde d'une transition est exprimée par une région de l'espace d'état continu, qui peut se ramener à des intervalles. Considérons le modèle partiel d'automate illustré par la Figure 2.5. La garde de la transition T_i est exprimée sous la forme $garde(T_i) = x_i \in [a, b]$. Cette transition devient franchissable à l'instant où $x_i = a$ et doit être franchie au plus tard à l'instant où $x_i = b$. Donc, la transition restera validée tant que $a \leq x_i \leq b$.

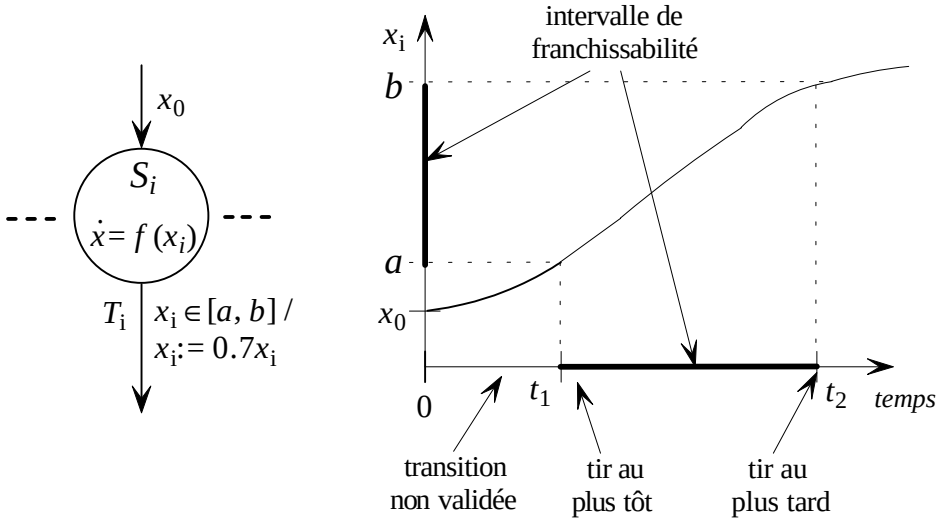


FIG. 2.5 – Validation d'une transition

L'ensemble des variables d'état continues mises à jour lors du franchissement d'une transition est décrit par une *affectation*. Les initialisations spécifiées par l'affectation peuvent correspondre à des fonctions, calculant la nouvelle valeur de l'état à partir de sa valeur avant le franchissement, plus complexes que la remise à zéro des horloges dans les automates temporisés. Ainsi, dans l'exemple illustré dans la figure précédente (Figure 2.5) la variable x_i sera mise à jour lors du franchissement de la transition T_i à 70% de la valeur de x_i avant le franchissement.

Formellement, nous définissons l'automate hybride à partir des définitions trouvées dans [ACHH93], [Hen96] et [HHWT98].

Définition 2.3. Un *automate hybride* d'ordre n est défini par :

$$\mathcal{A} = (Q, X, flux, inv, garde, Aff, init)$$

tel que :

- $Q = \{q_1, q_2, ..., q_m\}$ est un ensemble fini des sommets du graphe représentant les états discrets du système modélisé ;
- $X \subseteq \mathbb{R}^n$ est l'espace d'état continu. L'état continu du système est caractérisé à tout instant par le vecteur $x = [x_1\ x_2\ ...x_n]^T$ dans l'espace Euclidien $\mathbb{R}^n$;
- $flux(q_i)$ est la fonction qui affecte à chaque sommet une représentation pour l'évolution continue. Durant le séjour dans un sommet q_i de l'automate hybride, l'évolution des variables continues est exprimée généralement sous la forme d'une équation d'état $flux(q_i) : \dot{x} = \varphi(q_i)(t, x, u)$, où $x \in X \subset \mathbb{R}^n$, $u \in U \subset \mathbb{R}^p$ et $\varphi : X \times U \rightarrow X$;
- l'invariant $inv(q_i)$ est une fonction qui associe à chaque sommet $q_i \in Q$ une contrainte sur les variables d'états continues $x(.)$. Le système peut séjourner dans un sommet tant que l'invariant du sommet est satisfait ;
- $garde(T_i)$ est une fonction qui associe une condition de franchissement à chaque transition $T_i \in E$. Cette condition est en général une fonction logique entre des prédicats, portant sur les variables $x \in X$ et/ou ses dérivées $\dot{x} \in X$. Une transition $T_i \in E$ ne peut pas être franchie que si la condition $garde(T_i)$ est vrai ;
- l'affectation $Aff(T_i)$ est une fonction qui associe à chaque transition $T_i \in E$ une relation qui permet de mettre à jour la valeur des variables d'état après l'exécution de la transition T_i. Une affectation est donnée sous la forme de prédicats simples de type $x_i := c_i | x_i \in X$ où c_i est une constante réelle dans$X$;
- $init$ est une fonction qui affecte un état initial $x_0 \in X$ au sommet initial $q_{in} \in Q$. La condition initiale $init(q_{in})$ est un prédicat sur X.

■

D'une manière intuitive, les sommets de l'automate hybride permettent de différencier les dynamiques continues du système. L'état d'un système hybride est caractérisé à tout instant par la paire (q_i, x) représentant l'*état global* du système.

Définition 2.4. L'*état global* du système à l'instant t est déterminé par un sommet $q_i \in Q$ et par la valeur du vecteur d'état $x \in \mathbb{R}^n$ à l'instant de temps considéré.

■

L'évolution dynamique d'un automate hybride commence à partir d'une région initiale R_0. Dans chaque sommet, les valeurs des variables changent avec le passage du temps en respectant la fonction de flux continu et l'invariant. Ainsi, l'invariant limite l'espace d'évolution du vecteur d'état dans chaque sommet de l'automate.

Une transition discrète $T_i \in E$, entre les sommets q_i et q_j de l'automate, peut être franchie si la condition $garde(T_i)$ est vraie. L'état résultant du franchissement de la transition est la paire (q_j, x') où x' représente la valeur du vecteur d'état après l'initialisation par la fonction $Aff(T_i)$. L'exécution d'une transition ne prend pas de temps, autrement dit, les transitions sont instantanées. Par conséquent, l'écoulement du temps ne se produit que dans les sommets de l'automate.

Nous pouvons conclure qu'un automate hybride évolue par une alternance de pas continus, où les variables continues et le temps évoluent de façon continue, et de pas discrets, où plusieurs transitions discrètes et instantanées peuvent être franchies.

Exemple 2.2. Reprenant l'exemple du thermostat (Exemple 1.1, pour le cas déterministe et Exemple 1.2 pour le cas non-déterministe) et complétant avec les notions de la définition 2.3, les modèles des automates hybrides correspondants aux deux exemples sont présentés dans la Figure 2.6. Les deux états discrets correspondants aux états en *marche* et en *arrêt*

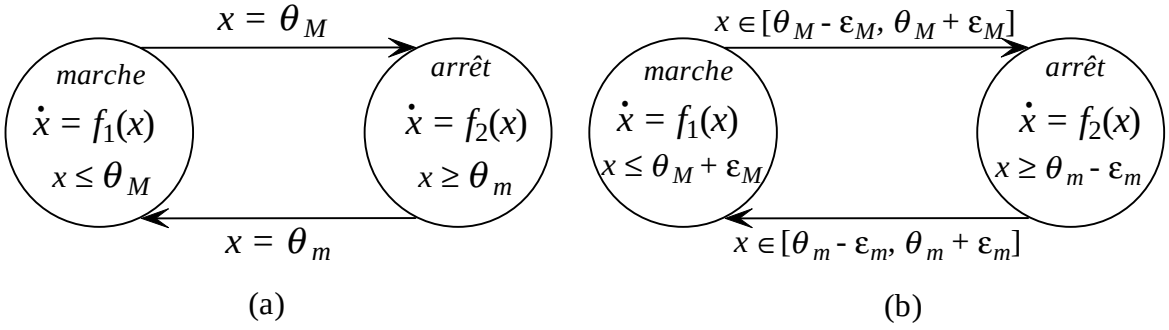


FIG. 2.6 – Automate hybride modélisant le cas déterministe (a) et le cas non-déterministe (b) d'un thermostat

du système sont modélisés par les sommets du graphe. La variable continue x modélise la température et ses dynamiques, associées aux deux états discrets du système, sont données sous la forme d'équations différentielles. Dans ces deux modèles, on remarque l'utilisation de l'invariant pour limiter l'évolution continue du système dans ses états discrets. Ainsi, pour le cas déterministe,l'état en *marche*, est représenté par la condition $x \leq \theta_M$. Les conditions de franchissement, qui déterminent le passage d'un état discret vers l'autre, sont spécifiées par les gardes des transitions. L'utilisation de l'invariant dans les sommets de l'automate limite l'évolution de ses variables continues.

■

Même si l'évolution continue dans chaque état discret est déterministe, l'évolution hybride peut avoir un caractère non-déterministe pour les raisons suivantes :

– Le résultat de l'intersection entre la région atteignable par la dynamique continue, les régions qui définissent l'invariant dans un sommet et les conditions de franchissement associées avec les transitions de sortie du sommet ne se ramènent pas à un seul point de commutation. Ainsi, il y aura des points où le système étant dans un état discret q_i peut soit évoluer vers un autre état q_j, soit rester dans le même sommet. Pour illustrer ce phénomène, considérons toujours l'exemple du thermostat. Supposons que le système commence évoluer à partir de l'état en *marche*. Au bout d'un certain temps, la température du système atteindra le seuil θ_M validant ainsi la transition vers l'état en *arrêt*. Dans le cas déterministe (Figure 2.6(a)), le système peut séjourner dans le sommet correspondant à l'état *marche* tant que la température reste inférieure ou égale à θ_M. Donc, lorsque la valeur de la variable d'état x a atteint la valeur θ_M, la transition vers le sommet correspondant à l'état *arrêt* sera franchie. Par contre, dans le cas non-déterministe (Figure 2.6(b)), l'invariant permet le séjour du système dans ce sommet tant que la condition $x \leq \theta_M + \varepsilon_M$ est vraie. Autrement dit, le système modélisé par la Figure 2.6(b), permettra la commutation vers l'autre état du système à n'importe quel instant tant que la valeur de la température restera dans l'intervalle $x \in [\theta_M - \varepsilon_M, \theta_M + \varepsilon_M]$. Le comportement du système illustré dans la Figure 2.6(a) est déterministe car le franchissement des transitions aura lieu à des instants précis quand la valeur de la température atteindra certains seuils, alors que le comportement du système représenté dans la Figure 2.6(b) est non-déterministe.

Le non-déterminisme provient du fait qu'on ne sait pas a priori l'instant precis de la commutation. Il est ainsi possible de franchir une transition si la valeur du vecteur d'état continu est dans l'intervalle défini par sa garde et il est tel que sa transformation par l'affectation satisfait l'invariant du sommet d'arrivé.
- Si la fonction d'affectation est définie par un intervalle, alors elle peut générer une évolution non-déterministe des variables continues au moment du franchissement d'une transition.
- Dans le modèle de l'automate plusieurs transitions de sortie d'un sommet peuvent être validées en même temps.

2.3.2 *Sémantique*

La sémantique des automates hybrides est définie en considérant qu'à chaque instant, l'état d'un automate hybride est donné par la paire (q, x) correspondant à l'association d'un état discret du système et d'une valeur du vecteur d'état. Nous présentons la sémantique d'un automate hybride $\mathcal{A}$, en terme de tous les comportements qui peuvent être générés par l'évolution du système modélisé. Dans ce sens, la sémantique d'un système continu, définie par un ensemble d'équations différentielles, est donnée par l'ensemble de toutes ses solutions, notamment de toutes ses trajectoires.

Comme nous venons de le présenter, l'évolution de l'automate hybride est réalisé par :
- Une *évolution continue* de ses variables continues par la progression du temps dans l'état discret courant, en respectant les équations de flux continu $flux(q_i)$ associées à cet état.
- Une *évolution discrète* par l'exécution instantanée d'une transition qui change l'état discret et la valeur du vecteur d'état continu.

L'état global du système, défini par la paire (q, x), change par l'intermédiaire des flux continus (x change en fonction de l'équation de flux continu) ou bien en conséquence au franchissement d'une transition, qui implique ainsi le changement de l'état discret du système.

Plus formellement, une trajectoire d'un automate hybride est définie par la séquence :

$$(q_0, \delta_0, x_0) \rightarrow (q_1, \delta_1, x_1) \rightarrow (q_2, \delta_2, x_2) \rightarrow \cdots \rightarrow (q_i, \delta_i, x_i) \rightarrow \cdots$$

où $\delta_i \in \mathbb{R}^+$ correspond à la durée pendant laquelle l'automate reste dans la situation q_i, sa valeur est remise à zéro à chaque commutation, et x_i est la fonction vectorielle qui décrit l'évolution de l'état continu pendant cette durée.

2.4 Classe d'application considérée

2.4.1 *Généralités*

Dans la suite de notre travail nous nous intéressons plus particulièrement à une classe de systèmes hybrides dont la partie continue est représentée par une structure de système continu linéaire évoluant sur un espace d'état délimité par des spécifications de fonctionnement (contraintes sur l'état). Ainsi, l'évolution continue peut être décrite par une équation différentielle linéaire de type $\dot{x} = Ax + Bu$, où $x \in \mathbb{R}^n$ représente le vecteur de variables d'état continues du système étudié.

L'évolution du système est réalisée tant par des changements d'état discret que par des commutations entre les différentes dynamiques continues associées au même état discret, appelés *commutations autonomes*.

La plupart des systèmes réels ont une nature hybride, dans le sens où ils possèdent à la fois des aspects discrets et des aspects continus. Cependant, il est assez difficile de trouver des systèmes réels où la dynamique continue est associée à une dynamique discrète riche. Les systèmes de production font partie de cette catégorie.

2.4.2 *Systèmes de production - Une architecture générique*

Les systèmes de production complexes peuvent être vus comme des systèmes hybrides ayant des dynamiques continues, représentant les flux de pièces dans le système et divers aspects discrets tels que l'état des ressources ou les routages. De plus, les aspects de concurrence, conflit, synchronisation entre les différentes ressources du système sont des situations fréquemment rencontrées dans ce type de systèmes.

Considérons une architecture générique de système de production représentée dans la Figure 2.7. Le système considéré est composé d'un enchaînement des ressources : stocks et machines. Les stocks (S_1, S_2, S_2'...) sont utilisés pour emmagasiner les pièces (matière première ou pièces en cours de traitement) jusqu'au moment où une machine en aval est disponible pour commencer un nouveau traitement.

Un modèle continu peut approximer le flux des pièces. L'idée de la modélisation continue a été étudiée par certains auteurs ([GS80],[DF82]) ; elle fournit une très bonne approximation de la dynamique des pièces dans les stocks intermédiaires d'un système de production ([DA90], [MC91]).

Même si modéliser le nombre de pièces dans un stock par un nombre réel est généralement une approximation acceptable, les états des machines sont forcément discrets (en marche ou non, par exemple). Nous pouvons alors conclure que les systèmes de production contiennent les caractéristiques de base d'un système hybride [Kur02]. C'est ce que nous allons détailler ci-dessous.

2.4.3 *Nature hybride des systèmes de production*

Dans la suite de cette section, la nature hybride des systèmes de production sera discutée en mettant en évidence les aspect continus et discrets qui, par leur interaction, déterminent l'évolution globale du système.

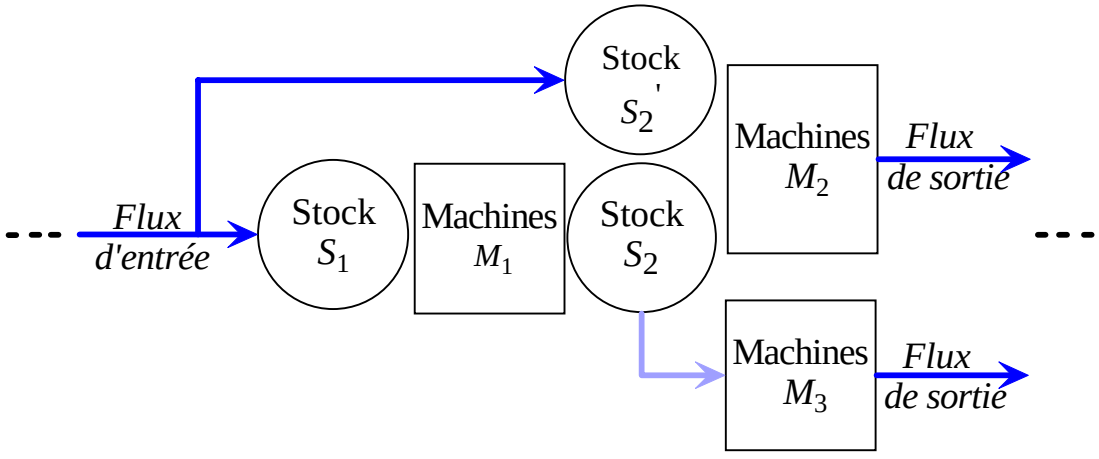


FIG. 2.7 – Architecture générique d'un système de production

(a) *Aspects continus*

Dans la Figure 2.7, le système est constitué d'une succession de stocks (S_1, S_2, S_2'...) et de groupes de machines ($M_1, M_2, M_3, ...$). Chaque groupe de machines est composé d'un nombre de machines identiques. Pour illustrer la modélisation mathématique d'un tel système, nous allons considérer deux situations :

- le cas où le nombre de pièces qui peuvent être prises en charge par un groupe de machines M_i est illimité ; les machines sont des *serveurs infinis*, et
- le cas où le nombre de pièces qui peuvent être prises en charge par le groupe de machines M_i est limité ; les machines sont des *serveurs finis*.

Cas serveurs infinis

D'une manière théorique, dans l'architecture générique d'un système de production (Figure 2.7) les machines qui précédent les stocks peuvent être vues comme des serveurs infinis. C'est-à-dire que le groupe de machines M_i en aval d'un stock S_i pourra prendre en charge toutes les pièces qui sont emmagasinées dans le stock en amont, quel que soient leur nombre. Dans ce contexte, le flux de pièces dans le système peut être approché par un modèle continu décrit par une équation différentielle linéaire

$$\dot{x} = Ax + Bu \tag{2.3}$$

où x est un vecteur modélisant la variation du niveau de pièces dans les stocks et u le débit d'entrée.

La situation où le nombre de serveurs qui compose un groupe de machines est infini est purement théorique, elle n'est jamais rencontrée dans des systèmes réels.

Cas serveurs finis

Dans la plupart des systèmes réels, le nombre de machines qui compose un groupe M_i est fini. Ainsi, chaque groupe de machines de l'architecture générique est constitué par un nombre de machines identiques. Le flux de pièces dans un tel système peut être modélisé par l'équation différentielle

$$\dot{x}(t) = A\,\mathbf{min}(x, k) + B \tag{2.4}$$

où x représente le vecteur d'état dans lequel la composante x_i modélise le niveau de pièces dans le stock S_i et k est un vecteur de constantes dont l'élément k_i exprime le nombre de machines qui compose le groupe M_i. L'utilisation de la fonction **min** exprime le fait que la productivité du système est limitée par le nombre de serveurs qui compose chaque groupe de machines intervenant dans le système. Le modèle obtenu est général et décrit un ensemble de comportements dynamiques du système en fonction de la valeur courante du vecteur d'état. Les différents comportements sont déterminés par le résultat de l'évaluation $\mathbf{min}(x, k)$.

Afin d'illustrer ce propos, considérons la Figure 2.7 et analysons l'influence des groupes de machines M_2 et M_3 sur la variation du niveau des pièces dans le stock S_2. Le modèle mathématique correspondant est :

$$\dot{x}_2(t) = -a_2\mathbf{min}(x'_2, k_2) - a_3\mathbf{min}(x_2 - x'_2, k_3) + a_1\mathbf{min}(x_1, k_1) \tag{2.5}$$

où la variable x'_2 vérifie la condition $0 \leq x'_2 \leq x_2$. A partir de cette expression huit comportements différents peuvent être générés. Citons ci-dessous trois parmi les huit possibles :

- Si $\mathbf{min}(x'_2, k_2) = x'_2$, $\mathbf{min}(x_2 - x'_2, k_3) = x_2 - x'_2$ et $\mathbf{min}(x_1, k_1) = x_1$ la relation 2.5 devient $\dot{x}_2(t) = -a_2 x'_2 - a_3(x_2 - x'_2) + a_1 x_1$;
- Si $\mathbf{min}(x'_2, k_2) = x'_2$, $\mathbf{min}(x_2 - x'_2, k_3) = k_3$ et $\mathbf{min}(x_1, k_1) = k_1$ on obtiens $\dot{x}_2(t) = -a_2 x'_2 - a_3 k_3 + a_1 k_1$;
- Si $\mathbf{min}(x_2", k_2) = k_2$, $\mathbf{min}(x_2 - x_2", k_3) = k_3$ et $\mathbf{min}(x_1, k_1) = k_1$ la dynamique continue est régie par $\dot{x}_2(t) = -a_2 k_2 - a_3 k_3 + a_1 k_1$.

Nous avons illustré l'influence des deux groupes de machines, les groupes M_2 et M_3, sur la variation du niveau dans le stock S_2. Cependant, les vecteurs x et k sont n-dimensionnels ($x, k \in \mathbb{R}^n$, où n représente le nombre de stocks dans le système) et par l'évaluation de la fonction $\mathbf{min}(x, k)$ plusieurs comportements dynamiques pourront être générés. Le nombre maximal de comportements possibles est 2^n ($n \in \mathbb{N}$).

Dans la suite, nous illustrons la généralité des expressions mathématiques proposées au début de cette section, à travers deux exemples qui traitent des situations fréquemment rencontrées dans les systèmes de production. Ces situations correspondent à celles de conflit et/ou synchronisation entre les ressources du système (illustrées aussi dans l'architecture générique présentée dans la Figure 2.7).

Exemple 2.3. *Situation de conflit*

Le comportement d'un système de production, comme nous venons de le présenter, est déterminé par l'évolution de ses variables continues modélisant le niveau de pièces dans les stocks. Lorsque deux ou plusieurs machines ont un stock d'entrée commun, on dit qu'il y a une situation de conflit structurel.

Un exemple de conflit est illustré par la Figure 2.8. Le modèle du système de production choisi pour illustrer la situation de conflit est composé par deux groupes de machines M_1 et M_2, dont le nombre de machines qui le composent est k_1 et k_2. Aux machines qui constituent le groupe M_1 sont associées des taux de production a_1, et à celles qui font partie du groupe M_2 sont associées des taux de production a_2. Les machines ont un stock d'entrée commun S_1 alimenté à un débit constant d.

Un conflit structurel devient effectif lorsque le nombre de pièces présentes à un certain instant dans le stock S_1 ne permet pas d'alimenter avec des pièces brutes les machines M_1 et M_2. Deux cas sont à distinguer :

- le cas de *serveurs infinis* ($k_i \gg x_1$, où $i = 1, 2$) - dans cette situation il y a toujours conflit effectif, et
- le cas de *serveurs finis* (k_1 et k_2 sont des nombres finis) - dans cette situation le

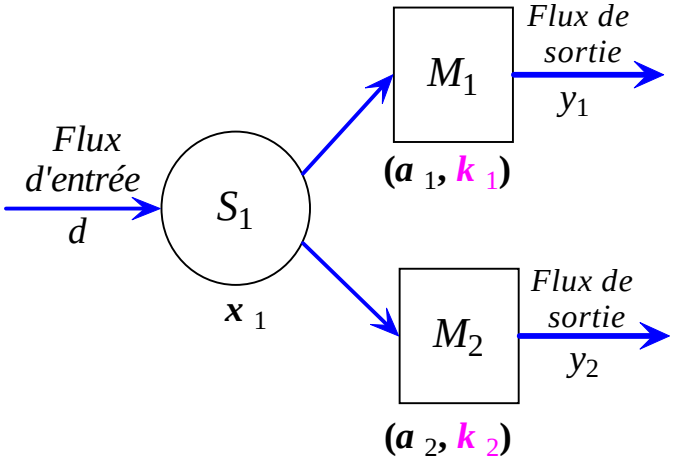


FIG. 2.8 – Conflit structurel dans un système de production

conflit devient effectif lorsque $x_1 < k_1 + k_2$.

Un conflit effectif introduit un non-déterminisme dans l'évolution du système. Dans l'exemple considéré, la variable continue représente le niveau des pièces x_1 dans le stock S_1. L'expression mathématique, modélisant la dynamique de cette variable, est donnée par l'équation différentielle suivante :

$$\dot{x}_1(t) = -a_1 \mathbf{min}(x_1', k_1) - a_2 \mathbf{min}((x_1 - x_1'), k_2) + d \tag{2.6}$$

où la variable x_1' vérifie la relation $0 \leq x_1' \leq x_1$. Cette expression représente le modèle mathématique dans le cas où le nombre de machines qui compose chaque groupe est fini.

Dans le cas d'un système *serveurs infinis*, l'expression mathématique devient :

$$\dot{x}_1(t) = -a_1 x_1' - a_2(x_1 - x_1') + d \tag{2.7}$$

Le non-déterminisme est représenté dans le modèle par l'utilisation de la variable x_1'. Ce non-déterminisme peut être éliminé en choisissant une politique de résolution de conflit appropriée, par priorité ou par partage proportionnel. Un exemple de politique de résolution est d'imposer une priorité de traitement au groupe de machines M_1 et les pièces restant dans le stock, après la saturation de ce groupe de machines, seront traitées par le deuxième groupe de machines M_2.

■

Exemple 2.4. *Synchronisation entre les ressources du système*

La synchronisation représente une situation spécifique dans les systèmes où, par exemple, interviennent des opérations d'assemblage. Considérons un système où la synchronisation entre ses ressources se situe au niveau de la machine M_1, comme l'illustre la Figure 2.9. Cette machine doit effectuer l'assemblage d'une pièce du stock S_1 avec une pièce provenant du stock S_2.

Les variables d'état continues sont les niveaux de pièces dénotées par x_1 et x_2, dans les deux stocks S_1 et S_2. M_1 représente un groupe de k_1 machines identiques dont le taux de production est a_1. Le modèle qui décrit le comportement dynamique du système est donné par l'ensemble des équations différentielles

$$\begin{cases} \dot{x}_1(t) = -a_1 \mathbf{min}(x_1, x_2, k_1) + d_1 \\ \dot{x}_2(t) = -a_1 \mathbf{min}(x_1, x_2, k_1) + d_2 \end{cases} \tag{2.8}$$

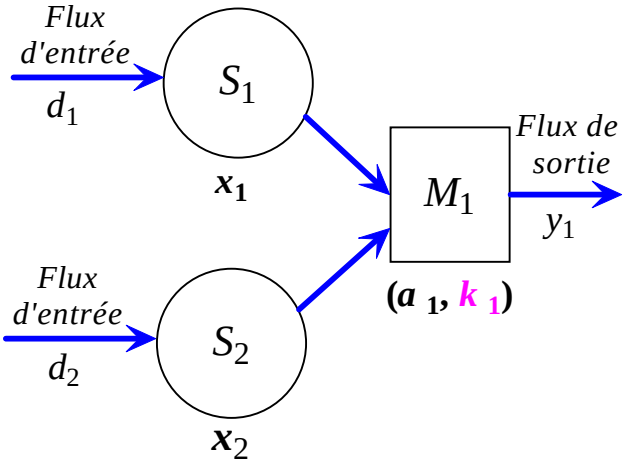


FIG. 2.9 – Synchronisation au niveau de la machine M_1

Cette expression modélise l'évolution du niveau des pièces dans les stocks S_1 et S_2 pour un système serveurs finis. Si le nombre de machines qui compose le groupe M_1 est infini, l'expression mathématique devient

$$\begin{cases} \dot{x}_1(t) = -a_1\mathbf{min}(x_1, x_2) + d_1 \\ \dot{x}_2(t) = -a_1\mathbf{min}(x_1, x_2) + d_2 \end{cases} \tag{2.9}$$

Dans la situation où il y a une synchronisation entre certaines ressources du système nous pouvons définir le débit de sortie, comme suit

$$y_1 = a_1\mathbf{min}(x_1, x_2, k_1) \tag{2.10}$$

■

Nous remarquons que dans un système de production, la variable continue du système est x. Elle modélise le niveau de pièces dans les stocks du système à un instant donné. Alors, il est nécessaire d'établir la propriété suivante :

Propriété 2.4.1. *La variable continue $x(t)$ prendra toujours des valeurs positives, i.e., $\forall\ t,\ x(t)\geq 0$.*

Preuve : Considérons un système de production comme celui illustré sur la Figure 2.10. Ce système est constitué par un succession des stocks et des groupes de machines. Un groupe de machine peut commencer un traitement lorsqu'il y a des pièces dans son (ses) stock(s) d'entrée. La variable continue dans un système de production représente le niveau des pièces dans les stocks, notée $x(t)$ et est modélisée par un vecteur n-dimensionnel ($x(t)\in\mathbb{R}^n$). Le flux d'entrée du système représente le débit des pièces brutes d (pièces par unité de temps) et il est forcément une quantité positive ($d \geq 0$). Le flux de sortie est représenté par le débit des produits fini (pièces par unité de temps) et ne peut être que positif ou zéro.

D'abord, nous décomposons le système en étages, comme illustre la Figure 2.10, tel que la sortie d'un étage constitue l'entrée de l'étage suivant. Ensuite, par une démarche inductive nous montrons que à tout instant pendant le fonctionnement du système le niveau des pièces dans les stocks est positif.

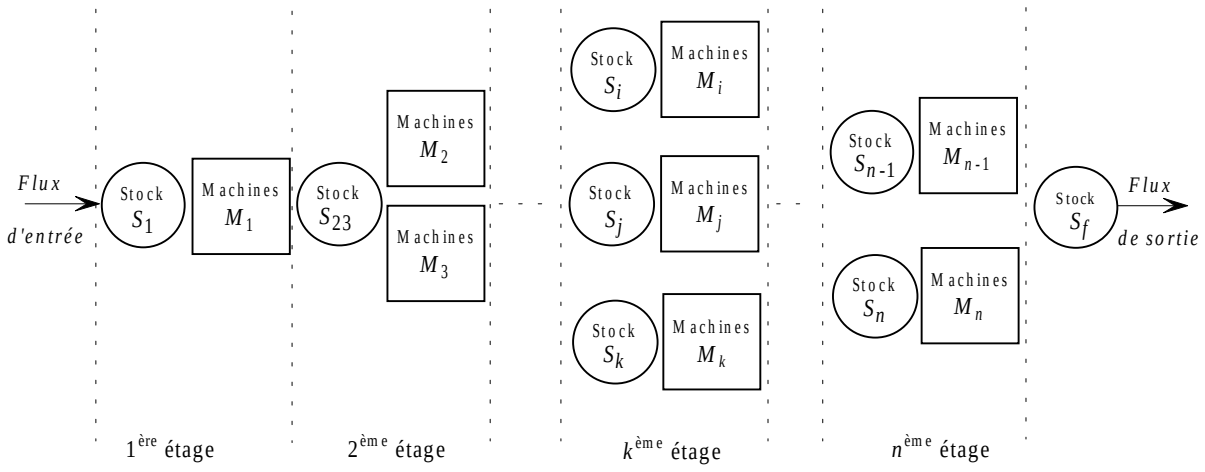


FIG. 2.10 – Architecture d'un système de production

Pour cela considérons un étage du système de production comme illustre la Figure 2.11.

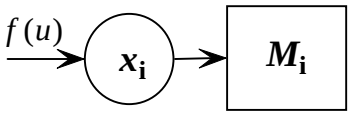


FIG. 2.11 – Un étage du système de production

Le modèle mathématique modélisant la variation du niveau des pièces dans le stock en amont de la machine M_i est :

$$\dot{x}_i = f(u) - a_i \mathbf{min}(x_i, k_i)$$

avec le flux d'entrée $f(u) \geq 0$, $\forall t$ et $x_i(0) \geq 0$. Cela implique que $\dot{x}_i$ est de signe quelconque.

Si $\dot{x}_i > 0$, $x_i(t)$ est également positif $\forall t$.

Supposons $\dot{x}_i < 0$. Alors x_i décroît et il existe un instant t_k tel que $x_i(t_k) = 0$. Cela implique que $\dot{x}_i(t_k) = f(u(t_k)) \geq 0$. D'où $x_i(t) \geq 0$, $\forall t$.

■

Le fait d'avoir des synchronisations ou des conflits ne modifient en rien la démonstration. Il y seulement des termes en plus qu'ils interviennent dans l'expression de la fonction **min**.

En *boucle ouverte*, on commence par le flux d'entrée $f(u) = d > 0$. Cela implique que le flux de sortie est $y_1 > 0$. Le débit d'entrée du deuxième étage est le débit de sortie du premier, i.e. $f(u) = y_1$. Donc, le débit sortie de cet étage sera $y_2 > 0$ et ainsi de suite. En *boucle fermée*, on commencera par un débit d'entrée $f(u) = y_i$ correspondant au débit de sortie de la machine M_i et le système évoluera en respectant le même principe que dans le cas de système boucle ouverte.

Nous avons discuté les aspects continus d'un système de production. Grace aux exemples présentées ci-dessus, nous avons vu que le modèle mathématique représenté par la relation 2.4 constitue un bon modèle pour décrire l'évolution continue d'un système de production. Nous nous intéressons ci-dessous aux aspects discrets des systèmes de production.

(b) *Aspects discrets*

Il s'agit des événements dont l'origine est discrète telle que, l'intervention d'un opérateur pour arrêter ou redémarrer le flux d'entrée, ou bien la panne d'une machine. L'occurrence d'un de ces événements implique le changement d'état discret du système (passage de l'état flux d'entrée *On* vers l'état flux d'entrée *Off*, par exemple).

Une dynamique continue est associée à chaque état discret du système. Alors, un changement d'état implique forcément le changement de la dynamique continue. Les événements qui sont à l'origine d'un changement de ce type peuvent être *contrôlables* ou *non*.

Dans les états discrets du système, l'expression mathématique de la dynamique continue est exprimée en utilisant l'opérateur **min** (Relation 2.4). Alors, par l'écoulement du temps et, implicitement, par l'évolution des variables continues du système, l'utilisation de cet opérateur implique des changements de dynamiques continues en restant dans le même état discret. Ces changements sont dus à des commutations dont la nature est *incontrôlable* (c'est-à-dire, leur occurrence ne peut pas être empêchée). Ils sont appelés

commutations autonomes. Ainsi, on constate que dans les systèmes de production, il est possible d'avoir un grand nombre de commutations autonomes. Ceci rend cette classe d'application intéressante pour notre étude.

La limitation du nombre de machines dans un système de production conduit à l'apparition de commutations autonomes dans le modèle hybride du système. Ces commutations ont une nature incontrôlable et impliquent des changements de dynamique sans changement de l'état discret du système. De telles situations seront illustrées dans la section suivante lors de la modélisation d'une ligne de production.

2.5 Exemple de modélisation

Dans cette section nous présentons un exemple de modélisation en nous basant sur l'outil automate hybride qui nous permettra d'illustrer les notions théoriques présentées dans ce chapitre. L'objet de notre étude est représenté par une ligne de production ouverte.

2.5.1 *Description de l'exemple*

Nous considérons une ligne de production (Figure 2.12) composée par deux groupes de machines serveurs finis M_1 et M_2 et deux stocks S_1 et S_2.

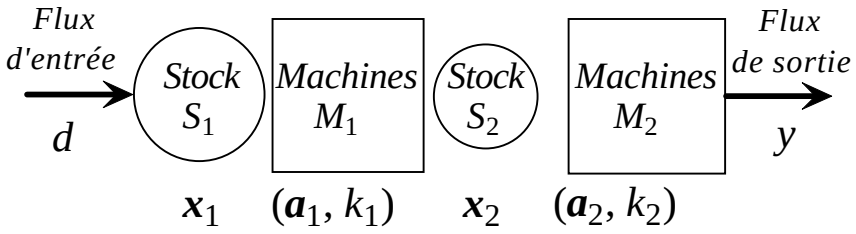


FIG. 2.12 – Ligne de production

Chaque groupe de machines, M_1 et M_2, est considérée comme un serveur fini. Le nombre de serveurs qui compose chaque groupe de machines est donné par les constantes $k_1 = 3$ et $k_2 = 7$ serveurs.

Le premier groupe de machines M_1 est alimenté avec des pièces brutes à partir du stock S_1. Une fois que les pièces brutes sont usinées par les machines qui constituent le groupe M_1, elles sont mises en attente dans le stock intermédiaire S_2 jusqu'au moment où les machines M_2 en aval sont disponibles pour commencer un nouveau traitement. L'entrée du système est représenté par un débit constant d (1 pièce /unité de temps) de matières premières (pièces brutes).

Des taux de production sont associés à chaque machine. Ainsi, aux machines du groupe M_1 est associé le taux de production $a_1 = 0.2$ pièces/unité de temps et aux machines du groupe M_2 est associé le taux de production $a_2 = 0.1$ pièces/unité de temps.

Les spécifications de fonctionnement imposées au système sont :

- *spécifications continues* : dans le stock S_1 le niveau de pièces x_1 doit être maintenu dans l'intervalle [1,4] et dans le stock S_2 le niveau de pièces x_2 doit rester dans l'intervalle [1,8] pendant l'évolution du système.
- *spécifications discrètes* : le flux d'entrée peut être arrêté et nous pouvons décider de l'instant d'arrêt. Par contre, une fois le flux d'entrée arrêté, il redémarre automatiquement après 4 unités de temps.

Dans la suite de cette section, nous nous proposons d'illustrer la modélisation mathématique des flux des pièces dans le système considéré. En utilisant cette représentation, nous construirons le modèle de l'automate hybride qui contient toutes les trajectoires du système.

2.5.2 *Modélisation mathématique*

Pour construire le modèle mathématique il faut étudier le comportement du système dans le cas où le *flux d'entrée* est *démarré* et respectivement dans le cas où *flux d'entrée* est *arrêté*. C'est le seul événement provenant de l'environnement considéré ici.

Flux d'entrée démarré

Les variables continues du système sont les niveaux de pièces contenues dans les stocks S_1 et S_2. Ainsi, l'expression mathématique modélisant la dynamique de ces variables dans la situation considérée est représentée par :

$$\begin{cases} \dot{x}_1(t) = -a_1 \mathbf{min}(x_1, k_1) + d \\ \dot{x}_2(t) = -a_2 \mathbf{min}(x_2, k_2) + a_1 \mathbf{min}(x_1, k_1) \end{cases} \tag{2.11}$$

où d représente le débit d'entrée du système, a_1 et a_2 sont les taux de productions associés aux groupes de machines et k_1 et k_2 sont les nombres de serveurs qui composent les groupes des machines M_1 et M_2.

Le modèle mathématique ainsi obtenu est donné sous la forme générale contenant des discontinuités, similaire à celle décrite par la relation 2.4. Par l'évaluation de la fonction **min**, des dynamiques simples linéaires peuvent être obtenues.

Flux d'entrée arrêté

Le modèle mathématique obtenu dans cette situation sera similaire à celui exprimé par la relation 2.11, sauf que dans ce cas la valeur du débit d'entrée aura la valeur $d = 0$. L'expression mathématique ainsi obtenue est

$$\begin{cases} \dot{x}_1(t) = -a_1 \mathbf{min}(x_1, k_1) \\ \dot{x}_2(t) = -a_2 \mathbf{min}(x_2, k_2) + a_1 \mathbf{min}(x_1, k_1) \end{cases} \tag{2.12}$$

2.5.3 *Modèle de l'automate hybride*

Nous pouvons constater que dans notre système, il y a deux états discrets modélisant les deux situations possibles : *flux d'entrée arrêté* et *flux d'entrée démarré*. L'automate hybride modélisant la ligne de production considérée est illustré dans la Figure 2.13.

Les sommets $q_1(ON)$ et $q_2(OFF)$ correspondent aux deux états discrets du système. Dans le modèle, le sommet initial correspond à l'état du système où le flux d'entrée est démarré ($q_1(ON)$) et la condition initiale est représentée par le vecteur, qui modélise le niveau initial dans les deux stocks, $x_0 = [2 \quad 1]^T$. Ainsi, initialement le stock S_1 contient 2 pièces et S_2 contient 1 pièce. Généralement, le contenu des stocks n'est pas connu avec précision. Donc, la condition initiale n'est pas donnée par un point mais par une région initiale.

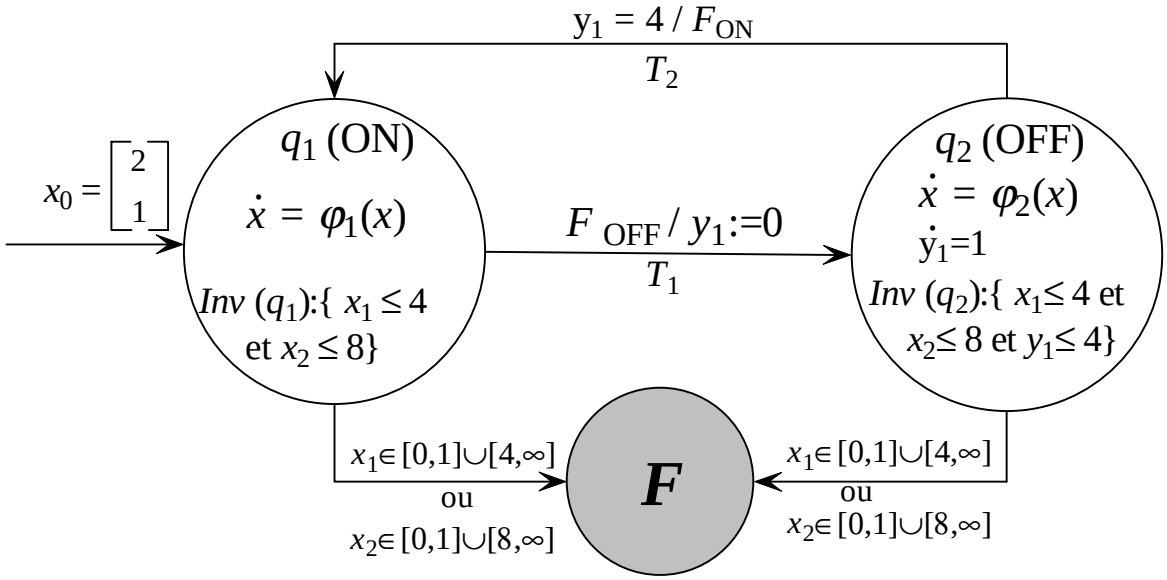


FIG. 2.13 – Automate hybride modélisant la ligne de production

Les dynamiques continues associées aux états discrets sont modélisées par l'équation de flux

$$\dot{x} = \varphi_1(x)$$

associée au sommet q_1, et respectivement

$$\dot{x} = \varphi_2(x) \text{ et } \dot{y}_1 = 1$$

associées au sommet q_2. Ces équations de flux correspondent aux expressions mathématiques trouvées lors de la modélisation mathématique (la relation 2.11 pour le sommet q_1 et la relation 2.12 pour le sommet q_2).

L'invariant limite l'évolution continue du système dans le sommet de l'automate hybride. Ainsi, dans le sommet $q_1(ON)$, l'invariant associé $Inv(q_1)$ est représenté par le prédicat spécifié sur les variables continues

$$Inv(q_1) : (x_1 \leq 4) \wedge (x_2 \leq 8).$$

Dans le sommet $q_2(OFF)$, l'invariant associé $Inv(q_2)$ est représenté par le prédicat spécifié sur les variables continues

$$Inv(q_2) : (x_1 \leq 4) \wedge (x_2 \leq 8) \wedge (y_1 \leq 4).$$

Les spécifications continues du système nous permettent de déduire la garde associée à la transition qui mène vers le sommet interdit du système, modélisé par le sommet F. Ainsi, cette garde est représentée par le prédicat

$$(x_1 \in [0, 1] \cup [4, \infty]) \vee (x_2 \in [0, 1] \cup [8, \infty]).$$

L'évolution vers ce sommet est possible seulement si les variables continues du système ne respectent plus les spécifications continues imposées par le cahier de charges.

Les spécifications discrètes sont modélisées par des événements générés par l'arrêt du flux d'entrée F_{OFF} et par son redémarrage F_{ON}. Le premier événement est contrôlable, l'instant de son occurrence peut être décidé. Par contre, l'autre est incontrôlable car, quelle que soit l'évolution des variables d'état, le flux sera redémarré automatiquement après 4 unités de temps. Pour compter l'écoulement du temps dans le sommet q_2, l'équation modélisant une horloge $\dot{y}_1 = 1$ est ajoutée aux équations modélisant le flux continu. Ainsi, l'arc reliant le sommet q_2 à q_1 est étiqueté par la condition $y_1 = 4$ et lors du franchissement de cette transition l'événement F_{ON} est généré.

Le modèle de l'automate hybride ainsi obtenu contient dans ses sommets des équations

de flux complexes. Pour mettre en évidence les commutations autonomes dans un même état discret du système (soit dans le sommet avec le flux d'entrée démarré - $q_1(ON)$, soit dans le sommet avec le flux d'entrée arrêté - $q_2(OFF)$), il faut expliciter ces dynamiques en fonction du résultat de l'évaluation des opérateurs **min** utilisées.

En effet, dans le cas de notre étude, nous pouvons observer que chaque dynamique complexe génère quatre dynamiques simples. Ainsi, le modèle de l'automate hybride dont les dynamiques associées au sommet sont linéaires sera constitué par 9 sommets reliés par des arcs.

Ce modèle est obtenu par la décomposition structurelle du modèle initiale (Figure 2.13) et il est illustré dans la Figure 2.14.

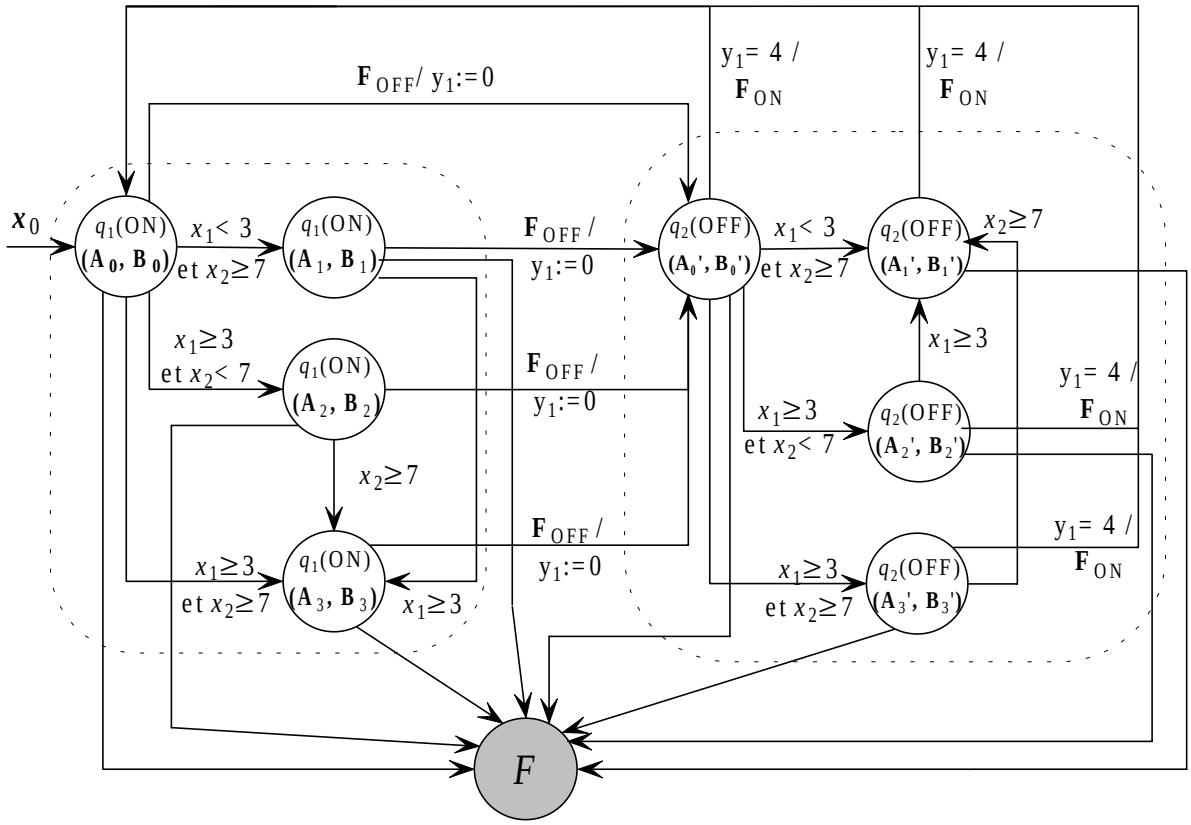


FIG. 2.14 – Automate hybride avec commutations autonomes

Dans ce modèle, il y a quatre sommets correspondant à un même état discret. A chaque sommet est associé une dynamique simple. Le système évolue dans un sommet tant que ses variables continues respectent les conditions logiques associées.

Dans la figure, nous avons indiqué la paire de matrices de la représentation d'état correspondant à chaque situation particulière. Ainsi, par exemple, la paire (A, B) denote le résultat de l'évaluation de la relation 2.4 dans le cas où

$$x_1 < k_1 \text{ et } x_2 < k_2.$$

Les transitions sortant de ce sommet indiquent les autres conditions logiques qui nous permettront d'identifier les autres paires des matrices A_i, B_i,où i=1,2,3.

D'une manière similaire sont obtenues les dynamiques correspondant au cas où le flux d'entrée est arrêté. Les valeurs explicites de ces matrices sont données dans le tableau 2.1.

Etat du flux / Cond. logique	*Démarré*		*Arrêté*	
	Sommet	Dynamique continue	Sommet	Dynamique continue
$(x_1 < k_1)$ et $(x_2 < k_2)$	(A_0, B_0)	$\begin{bmatrix} -a_1 & 0 \\ a_1 & -a_2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}+\begin{bmatrix} d \\ 0 \end{bmatrix}$	(A_0', B_0')	$\begin{bmatrix} -a_1 & 0 \\ a_1 & -a_2 \end{bmatrix}\begin{bmatrix} k_1 \\ x_2 \end{bmatrix}$
$(x_1 < k_1)$ et $(x_2 \geq k_2)$	(A_1, B_1)	$\begin{bmatrix} -a_1 & 0 \\ a_1 & -a_2 \end{bmatrix}\begin{bmatrix} x_1 \\ k_2 \end{bmatrix}+\begin{bmatrix} d \\ 0 \end{bmatrix}$	(A_1', B_1')	$\begin{bmatrix} -a_1 & 0 \\ a_1 & -a_2 \end{bmatrix}\begin{bmatrix} x_1 \\ k_2 \end{bmatrix}$
$(x_1 \geq k_1)$ et $(x_2 < k_2)$	(A_2, B_2)	$\begin{bmatrix} -a_1 & 0 \\ a_1 & -a_2 \end{bmatrix}\begin{bmatrix} k_1 \\ x_2 \end{bmatrix}+\begin{bmatrix} d \\ 0 \end{bmatrix}$	(A_2', B_2')	$\begin{bmatrix} -a_1 & 0 \\ a_1 & -a_2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$
$(x_1 \geq k_1)$ et $(x_2 \geq k_2)$	(A_3, B_3)	$\begin{bmatrix} -a_1 & 0 \\ a_1 & -a_2 \end{bmatrix}\begin{bmatrix} k_1 \\ k_2 \end{bmatrix}+\begin{bmatrix} d \\ 0 \end{bmatrix}$	(A_3', B_3')	$\begin{bmatrix} -a_1 & 0 \\ a_1 & -a_2 \end{bmatrix}\begin{bmatrix} k_1 \\ k_2 \end{bmatrix}$

TAB. 2.1 – Dynamiques correspondant aux sommets de l'automate hybride obtenu par la décomposition structurelle

Toutes les trajectoires qui mènent vers le sommet interdit F du système seront franchissables dès que les variables continues du système ne vérifieront plus les spécifications continues imposées, notamment quand

$$(x_1 \notin [1, 4] \text{ ou } x_2 \notin [1, 8]).$$

Le modèle illustré dans la Figure 2.14 modélise toutes les trajectoires potentielles du système. Lorsqu'une région initiale est choisie, il est alors possible que certaines transitions ne soient jamais franchies. Dans ce cas, un automate plus simple peut être obtenu. C'est l'étape d'analyse qui permettra d'obtenir ce que nous appellerons l'automate atteignable.

2.6 Conclusions

Dans ce chapitre nous avons présenté une structure générique de système hybride décrit en terme de *procédé* et de *spécifications de fonctionnement*. La complexité du système qui découle de cette structure justifie le choix de l'automate hybride comme outil de modélisation en vue de l'analyse et la synthèse de la commande.

L'*automate hybride* considère à la fois les comportements continus et événementiels dans une même structure. Cet outil peut être vu comme une extension de l'automate discret et il constitue le formalisme principalement utilisé pour la vérification de certaines propriétés du système. Cependant, en ce qui concerne la modélisation, on peut mentionner que le développement de modèles à l'aide des automates hybrides n'est pas aisé car la sémantique associée à ce formalisme est plutôt orientée vers l'analyse de comportements.

La classe d'application à laquelle nous nous intéressons plus particulièrement dans notre travail est représentée par les systèmes de production. Nous avons conclu sur la nature hybride de ces systèmes, par la mise en évidence de ses aspects continus et discrets qui régissent leur fonctionnement. Ces systèmes font partie de la catégorie des systèmes hybrides dont la dynamique discrète est très riche.

Par l'approche d'analyse et synthèse d'une commande supervisée, nous proposons

l'utilisation du modèle de l'automate hybride à cause de sa grande puissance d'analyse. Comme nous avons pu le constater, dans la modélisation présentée à la fin de ce chapitre, la taille des modèles obtenus par cet outil augmente considérablement avec la complexité du système étudié. Ainsi, il faut répondre premièrement à la question "Est-ce que toutes les trajectoires modélisées sont effectivement réalisables dans le système ?". Répondre à cette question revient à l'analyse d'atteignabilité des états dans le graphe.

Dans le chapitre suivant, nous introduisons la problématique de l'analyse d'atteignabilité. Nous proposons l'algorithme qui permet la construction du modèle d'automate atteignable qui ne modélise que les évolutions possibles du système pour une condition initiale donnée.

Chapitre 3

Analyse du comportement dynamique des systèmes hybrides

Dans ce chapitre, nous proposons une méthode permettant d'obtenir le modèle d'automate atteignable à partir d'un modèle d'automate hybride obtenu lors de l'étape de modélisation d'un système hybride. L'automate atteignable ne modélise que les évolutions possibles du système à partir d'une condition initiale donnée. L'approche proposée est basée sur la représentation en temps-discret de la dynamique continue.

L'analyse d'atteignabilité constitue un problème central dans la vérification des propriétés des systèmes hybrides par des automates hybrides. L'objectif de l'analyse est le calcul de l'espace d'état atteignable par l'évolution de l'automate hybride. Pour mener ce calcul il faut choisir un formalisme de représentation des régions calculées. Dans notre travail nous avons choisi de représenter ces régions par des polyèdres, car ce sont des objets simples à décrire et à utiliser.

L'analyse du comportement dynamique des systèmes hybrides constitue une étape importante de notre approche. En effet, nous cherchons vérifier si une région de l'espace d'état continu peut être atteinte à partir d'une région initiale donnée. Ceci nous permettra de décider de la franchissabilité des transitions de l'automate hybride initial modélisant le système.

Nous proposons un algorithme permettant la construction de l'automate atteignable à partir d'un modèle initial donné. Reprenant le modèle d'automate hybride modélisant la ligne de production, présenté dans le chapitre précédent, nous illustrons l'automate atteignable résultant par l'utilisation de l'algorithme proposé pour ce système, modélisant les évolutions possibles pour une condition initiale particulière.

3.1 Problématique

Afin de s'assurer de l'adéquation des systèmes par rapport à leurs spécifications, des validations sont nécessaires en vue de détecter les erreurs dans les phases initiales de la conception d'un système de commande. En particulier, la validité du modèle ne peut pas être garantie par sa construction et elle doit être vérifiée. Ainsi, on retrouve la problématique générale de l'analyse.

L'approche la plus générale pour l'analyse d'un système hybride en vue de la validation est la simulation des modèles élaborés. L'intérêt de la simulation est qu'elle s'applique à des nombreux modèles mais il est évident que c'est une approche expérimentale ne permettant pas de considérer toutes les évolutions et ne garantissant donc pas l'absence d'erreurs dans le modèle établi. Par conséquent, afin de pouvoir caractériser l'ensemble de toutes les trajectoires d'un système hybride, d'autres approches plus formelles doivent être utilisées.

Classiquement, on distingue trois propriétés comportementales à vérifier dans les systèmes dynamiques hybrides :

- la *propriété de sûreté* qui exprime des configurations qui ne doivent jamais être réalisées ;
- la *propriété de vivacité* qui exprime des évolutions qui doivent être possibles ou inévitables ; et
- la *propriété de temps de réponse* qui exprime des contraintes sur le temps minimum ou maximum séparant certains événements ou caractérisant certaines évolutions.

Parmi ces trois types de propriétés, dans les systèmes hybrides on s'interesse particulièrement aux propriétés de sûreté. Généralement, une propriété de sûreté s'exprime par un sous-domaine de l'espace d'état continu (par exemple, quand une variable d'état ne doit pas franchir un seuil) ou bien par un automate qui évolue vers un état particulier selon la satisfaction ou non de la propriété.

L'analyse des systèmes hybrides modélisés par des automates hybrides est généralement basée sur le calcul des régions de l'espace d'état atteignables à partir d'une région initiale donnée. D'abord, il faut vérifier si toutes les trajectoires modélisées par l'automate hybride sont effectivement réalisables dans le système. Répondre à cette question revient à mener une *analyse d'atteignabilité* des états dans le graphe.

Le *problème d'atteignabilité* peut être formulé comme suit : étant donné un système hybride et une région initiale, trouver la région de l'espace d'état qui peut être atteinte par l'évolution du système. Si le modèle du système est donné par un automate hybride, alors ce problème peut être réduit à vérifier l'atteignabilité des gardes, représentant les régions de validation associées avec les transitions du modèle. Les résultats de cette analyse seront utilisés pour apporter des modifications dans le modèle d'automate hybride initial. S'il y a des régions qui ne sont jamais atteintes (c'est-à-dire, des transitions jamais franchies), le modèle initial peut être simplifié par la suppression des transitions étiquetées par ces régions. Le résultat de cette procédure de vérification sera concrétisé par l'*automate atteignable*, modélisant les évolutions possibles du système pour une condition initiale donnée.

Ainsi, l'*analyse d'atteignabilité* pose le problème de l'existence d'une trajectoire à partir d'une région initiale R_0 qui peut amener la dynamique du système dans une région finale R_F.

En général, l'exécution d'un automate hybride est représentée par une alternance de *pas continus* et de *pas discrets*. Les *pas continus* sont obtenus en restant dans le même sommet de l'automate hybride et en laissant évoluer les variables selon leur fonctions $flux(q)$. Un *pas discret* est réalisé par le franchissement d'une transition. Par conséquent,

un *successeur continu* est obtenu en restant dans un même sommet et en laissant le temps s'écouler. Un *successeur discret* est obtenu en franchissant une transition de l'automate.

Dans la littérature, deux méthodes d'analyse sont utilisées pour vérifier l'atteignabilité de certaines régions de l'espace : la *méthode d'analyse en avant* et la *méthode d'analyse en arrière* :

- La *méthode d'analyse en avant* consiste à calculer l'ensemble de tous les successeurs d'une région initiale R_0 donnée, d'une manière itérative. La région cible de l'espace d'état (qu'on désire atteindre) est R_D, alors l'atteignabilité de cette région est vérifiée par l'intersection de cette région avec la région calculé R_A. Si $R_A \cap R_D = \emptyset$, alors la région cible R_D n'est pas atteignable à partir de la région initiale R_0. Si le résultat de l'intersection est non vide, alors la région cible R_D est atteignable à partir de R_0.
- La *méthode d'analyse en arrière* représente une méthode basée sur le calcul des prédécesseurs de la région cible R_D. Toute région depuis laquelle on peut atteindre une région donnée est un prédécesseur de cette région. Ainsi, cette méthode consiste à calculer l'ensemble de tous les prédécesseurs de la région R_D. Si la région initiale R_0 est incluse dans la région calculée, alors la région cible R_D est atteignable à partir de R_0.

Dans le cas où le système étudié est représenté par un système hybride, la difficulté majeure liée à ces méthodes se trouve dans le calcul effectif de la région de l'espace atteignable. Notamment, l'utilisation de ces méthodes implique le calcul :

- des successeurs et prédécesseurs d'une région initiale R_0 donnée ;
- de l'union des régions successeurs/prédécesseur pour être capable d'obtenir la région atteinte R_A par l'évolution du système ;
- de l'intersection des régions pour pouvoir décider si la région cible R_D a été atteinte ou non.

Nous pouvons alors conclure que, pour vérifier l'atteignabilité d'une certaine région, il faut d'abord la définir en terme de domaine de l'espace d'état. Ensuite, il faut calculer itérativement l'image directe où inverse d'un domaine par la fonction d'évolution du système et considérer l'intersection entre ces différents domaines. Le calcul de l'image d'un domaine par la fonction d'évolution est donc le point fondamental de l'analyse d'atteignabilité.

Dans les sections suivantes de ce chapitre, les étapes de notre approche d'analyse seront détaillées. D'abord, le cadre théorique de notre travail sera défini. Ensuite, une étude sur la validation d'une transition de l'automate hybride sera réalisée. Nous allons conclure ce chapitre par l'algorithme permettant d'obtenir l'automate atteignable à partir d'un automate hybride initial et pour des conditions initiales données.

3.2 Calcul de l'espace atteignable

Considérons un automate hybride $\mathcal{A} = (Q, X, flux, inv, garde, Aff, init)$ pour lequel on cherche à déterminer les états atteignables à partir d'une région initiale R_0 dans laquelle se trouve la variable continue de l'état. Les exécutions possibles pour l'automate sont représentées par des trajectoires constituées chacune par une succession des transitions continues, pendant lesquelles le système évolue selon la dynamique continue du sommet courant - définie par $flux$ - et de transitions discrètes correspondant aux franchissement des transitions de l'automate.

Par conséquent, le calcul de l'espace atteignable revient au calcul itératif des succes-

seurs et/ou prédécesseurs d'une région initiale.

Dans la suite de cette section nous présentons le cadre formel de notre travail.

3.2.1 *Représentation en temps-discret de la dynamique continue du système*

En général, les équations de $flux$, associées aux sommets de l'automate hybride, sont du type :

$$\dot{x} = \varphi(x) = Ax + Bu \tag{3.1}$$

La solution exacte d'une telle équation est donnée par :

$$\xi_x(t) = e^{At}\, x(0) + \int_0^t e^{A(t-\tau)}\; B\, u(\tau)\, d\tau \tag{3.2}$$

Le calcul analytique de la solution d'une telle équation n'est pas toujours possible. Par conséquent, pour résoudre et étudier l'évolution d'un système de ce type, en général on doit faire appel à des techniques de simulation numérique.

La simulation numérique permet d'approximer la solution d'une équation différentielle à partir d'une condition initiale donnée. La résolution numérique des équations différentielles est un outil très puissant. Cependant, la solution est approximative car l'intégration numérique de l'équation différentielle est seulement à certains instants discret, vue l'utilisation des systèmes de calcul digitaux. Ainsi la solution analytique ξ_x sera approchée par $\hat{\xi}_x$, qui représente la solution obtenue par intégration numérique.

Dans le but de réduire la complexité de calcul, engendrée par l'utilisation des modèles à temps continu, et d'augmenter la précision de ce calcul, nous avons choisi d'utiliser dans notre travail une *représentation en temps-discret* de la dynamique continue [KA01b][KA01c]. Nous verrons dans la suite que ceci comporte une autre avantage. Ainsi, les dynamiques continues seront modélisées par l'équation :

$$x_{k+1} = A_d x_k + B_d u_k \tag{3.3}$$

où : x_k représente le vecteur d'état continu a la $k$ième pas d'itération, la matrice A_d et le vecteur B_d sont obtenus à partir de leurs équivalents continus A et B par la procédure de discrétisation et le vecteur u_k représente l'entrée du système.

La période d'échantillonnage dépend de la nature de la dynamique continue et doit être choisie de manière à vérifier les conditions imposées par la théorème de Shannon. Dans ce contexte, les erreurs introduites par l'échantillonnage peuvent être définies et leur expression dépendra seulement de la méthode de discrétisation utilisée.

L'utilisation d'une telle représentation pour la dynamique continue du système engendre la nécessité d'avoir un outil de modélisation adéquat. Ainsi, nous allons définir l'automate hybride à temps-discret comme suit :

Définition 3.1. Un *automate hybride à temps-discret* est un automate hybride (Def. 2.3) dont les dynamiques associées à chaque sommet sont données par des équations du type :

$$x_{k+1} = A_d x_k + B_d u_k$$

■

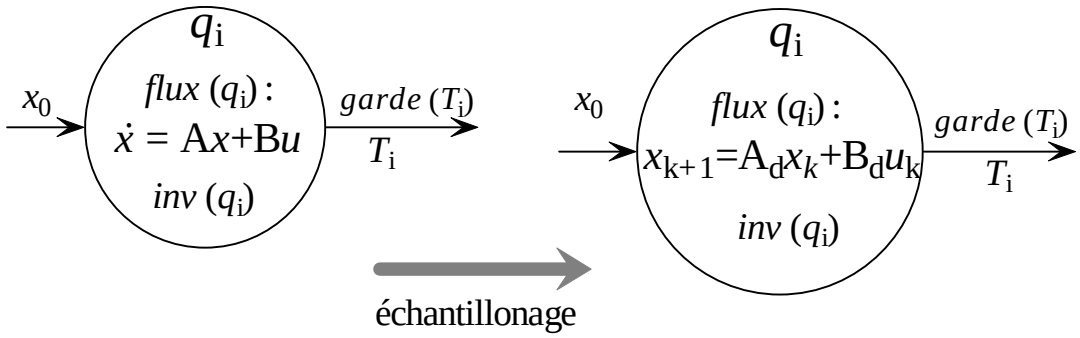


FIG. 3.1 – Passage à l'automate hybride en temps-discret

Par conséquent, la fonction de flux dans un sommet de l'automate hybride en temps continu sera remplacée par une équation équivalente en temps discret (Figure 3.1). La figure illustre le passage d'un sommet de l'automate hybride dans lequel les dynamiques continues sont représentées en temps continu vers un sommet dont les dynamiques sont exprimées en temps discret. Pour obtenir l'automate hybride en temps discret, cette transformation doit être réalisée pour tous les sommets de l'automate hybride initial.

Dans la suite de notre travail, nous considérons que notre système est modélisé par un automate hybride en temps discret.

3.2.2 *Représentation des régions par des polyèdres*

Vu la difficulté de calcul exact de successeurs continus et/ou de la région atteinte par l'évolution du système hybride modélisé par un automate hybride, pour des conditions initiales données, il faut tout d'abord trouver une représentation simple pour les régions de l'espace d'état continu qui interviennent dans ce calcul.

Les *polyèdres* sont des objets faciles à manipuler et représenter dans l'espace $\mathbb{R}^n$. Ainsi, dans notre travail nous avons choisi d'utiliser une telle représentation justifié par la simplicité de la représentation et leur pouvoir de description unique. Ainsi, une caractéristique importante des polyèdres peut être exprimée par la propriété suivante : :

Propriété 3.2.1. *Un polyèdre dans l'espace $\mathbb{R}^n$ peut être défini d'une manière unique par l'ensemble ordonné de ses sommets.*

■

Exemple 3.1. Pour illustrer ce propos, considérons une région comme celle représentée dans la Figure 3.2.

La région est représentée dans $\mathbb{R}^2$ et elle peut être décrite par l'ensemble de ses sommets. Chaque élément de l'ensemble représentera une paire des points de coordonnés. Ainsi, la région considérée peut être complètement caractérisée et uniquement décrite par l'ensemble ordonné : $\{(a_1, e_2), (d_1, f_2), (f_1, f_2), (f_1, d_2), (e_1, a_2), (b_1, b_2), (c_1, c_2)\}$. Ainsi, nous pouvons conclure que cette description nous donne une représentation unique de la région considérée.

■

Par conséquent, toutes les opérations qui devaient s'effectuer sur des ensemble réels sont remplacées par des opérations sur des polyèdres. Ainsi, le successeur continu à l'instant t sera décrit par un polyèdre. De même, la région atteignable par l'évolution du système à partir d'une région initiale donnée, sera obtenue par l'union de ses successeurs

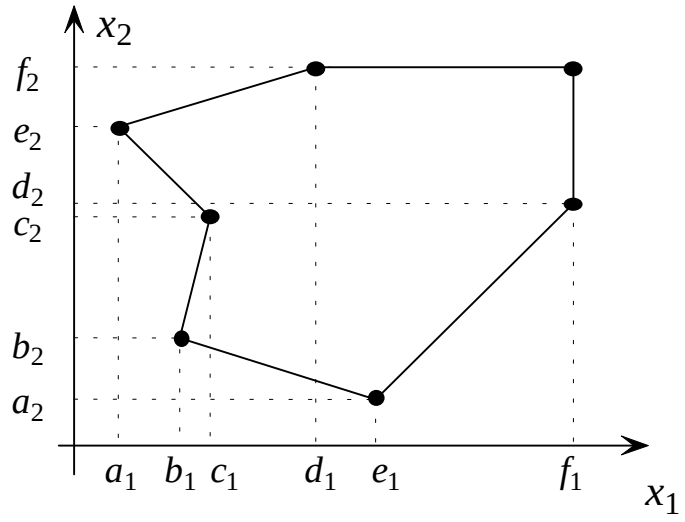


FIG. 3.2 – Représentation des régions par polyèdres en $\mathbb{R}^2$

continus.

Les polyèdres sont des objets qui peuvent être classés en deux ensembles : les polyèdres *convexes* et les polyèdres *non-convexes*. La première classe des polyèdres est la plus simple à utiliser. Cependant, la limitation d'utiliser seulement des régions convexes représente une contrainte assez forte, car pour obtenir la région atteignable par l'évolution du système, nous devons réaliser l'union de tous les successeurs continus. L'approximation de cette union par un polyèdre convexe peut engendrer des solutions peu précises qui ne permettront pas de conclure sur l'atteignabilité d'une certaine région. Ainsi, il va falloir utiliser aussi des polyèdres concaves lorsqu'on veut exprimer d'une manière très précise l'espace d'état atteignable par l'évolution du système.

L'utilisation des polyèdres non-convexes dans un espace $\mathbb{R}^n$, où $n > 2$, implique une complexité croissante au niveau des manipulations numériques lors de l'utilisation des techniques de simulation. Ainsi, dans notre travail, nous avons supposé que les régions initiales et les régions représentant les gardes des transitions de l'automate hybride (appelées *régions cibles*) qu'on désire atteindre, sont exprimées sous la forme de régions convexes. Nous nous proposons de construire les trajectoires exactes du système en nous basant sur le calcul des successeurs continus. Par conséquent, dans les algorithmes que nous allons présenter, nous utiliserons seulement la manipulation des polyèdres convexes. Les techniques de calcul proposées seront détaillées dans la suite de cette section.

3.2.3 *Calcul des successeurs continus*

Dans le calcul des successeurs continus, nous nous sommes basés sur une propriété importante des systèmes linéaires qui est énoncée et prouvée dans la suite de cette section.

Considérons un système continu linéaire $\mathcal{C} = (X, f)$ où $X \subseteq \mathbb{R}^n$. La dynamique continue du $\mathcal{C}$ est donnée par

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{3.4}$$

où $A \in \mathbb{R}^{nxn}$, $B \in \mathbb{R}^{nxm}$ et $u(t) \in \mathbb{R}^m$. Une propriété fondamentale des systèmes linéaires peut être formulée comme suit :

Lemme 3.2.2. *L'image d'une région convexe $R_0 \subseteq X$ par une application linéaire à chaque instant $t \geq 0$ est une région convexe.*

Preuve : Soit ξ_x la trajectoire du système $\mathcal{C}$ à partir d'une condition initiale $x \in X$. La solution de l'équation différentielle 3.4 est donnée par :

$$\xi_x(t) = e^{At}\, x + \int_0^t e^{A(t-\tau)}\; B\, u(\tau)\, d\tau \tag{3.5}$$

Soit $f(t, F)$ la région atteignable à partir d'une région initiale F en t unités de temps. Ainsi, cette région peut être exprimée sous la forme :

$$f(t, F) = e^{At}\, F + \int_0^t e^{A(t-\tau)}\; B\, u(\tau)\, d\tau \tag{3.6}$$

L'exponentielle matricielle e^{At} est un opérateur linéaire. Par conséquent, la convexité sera conservée. Donc, la région $f(t, F)$ est convexe.

■

Dans notre travail nous avons choisi de représenter les conditions initiales et les gardes associées aux transitions par des régions convexes. Ainsi, en nous basant sur la propriété 3.2.1, nous pouvons conclure que le calcul des successeurs continus se résume au calcul des trajectoires générées par chaque sommet de la région initiale. Dans le but de réduire la complexité des calculs, nous avons décidé d'utiliser une représentation en temps-discret de la dynamique continue. Ainsi, en conformité avec la propriété énoncée par le lemme 3.2.2, à chaque pas discret une région convexe sera obtenue.

La Figure 3.3 illustre le principe de calcul des successeurs continus à partir d'une région initiale R_0.

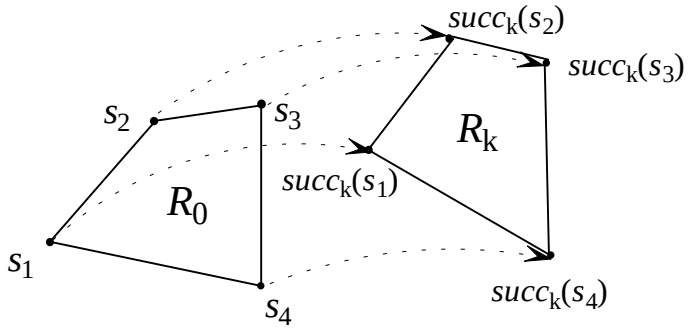


FIG. 3.3 – Calcul des successeurs d'une région initiale convexe R_0

Dans cette figure nous avons noté par R_k la région successeur de la région R_0 atteinte en k pas discrets. Cette région est décrite par ses quatre sommets, notés $succ_k(s_i)$ (pour $i = \overline{1,4}$). Le sommet $succ_k(s_i)$ est obtenu par le calcul de la trajectoire du sommet initial s_i. La trajectoire du système respecte la dynamique décrite par l'équation de flux en temps discret

$$x_{k+1} = A_d x_k + B_d \quad ,$$

en k pas discrets et pour $x_0 = s_i$. De cette manière on obtient les quatre points de coordonnées qui délimitent la région R_k.

Le principe de calcul des successeurs d'une région initiale est synthétisé dans les étapes de l'algorithme suivant. Nous supposons comme étant connus la région initiale R_0 et les matrices A_d et B_d.

Algorithme 3.1. *Calcul des successeurs d'une région initiale*

Pas 1 : Initialiser la région successeur $succ_0$ par la région initiale R_0.

Pas 2 : Initialiser la variable compteur $k := 0$.

Pas 3 : RÉPÉTER

3.1. Pour chaque sommet de la région courante, calculer le sommet successeur atteint par un pas discret $x_{k+1} = A_d x_k + B_d$, où A_d et B_d sont connus. D'une manière symbolique on peut écrire que la région successeur d'une région R_k est obtenue par $R_{k+1} := \varphi(R_k)$, où $\varphi(R_k)$ est la fonction qui permet le calcul des successeurs de tous les sommets de R_k atteints par un pas discret.

3.2. Actualiser le compteur $k := k + 1$.

JUSQU'A $R_{k+1} = R_k$

■

Par l'analyse de cet algorithme, nous pouvons constater que si on veut calculer tous les successeurs d'une région initiale la condition d'arrêt de l'algorithme est, en effet, une condition par laquelle on teste la convergence de la dynamique continue dans un sommet.

D'une manière plus formelle, la convergence de la dynamique continue est testée par le calcul de la distance entre deux régions successeurs continues d'une région initiale donnée. La distance entre deux régions successeurs R_k et R_{k+1} est obtenue en calculant pour chaque sommet de la région successeur R_{k+1} l'écart par rapport aux sommets correspondants de la région successeur R_k. Ainsi, la convergence des régions peut être définie comme suit :

Définition 3.2. Une dynamique continue dans un sommet de l'automate hybride converge au bout de k pas d'itération, si pour une valeur de ε fixée, le maximum de la distance entre la région R_k et les régions successeurs obtenues à partir de cette région vérifient la relation

$$\mathbf{max}(\delta(R_k, R_{j+1})) \leq \varepsilon \tag{3.7}$$

quel que soit $j \in \mathbb{N}$, $j \geq k$.

■

Nous illustrons le calcul de tous les successeur continus dans un sommet d'un automate hybride à temps-discret, par l'exemple suivant.

Exemple 3.2. Considérons le sommet de l'automate hybride représenté dans la Figure 3.4(a). Dans les automates hybrides, l'utilisation des invariants limite l'évolution dynamique du système. Dans l'exemple considéré, l'invariant associé au sommet q_i est représenté par la région $inv(q_i)$: $(x_1 \in [1, 6],\ x_2 \in [1, 10])$. En général, les conditions initiales sont données sous la forme d'intervalles qui délimitent une région initiale de l'espace d'état. Supposons que la dynamique en temps discret associée au sommet q_i est donnée par l'équation :

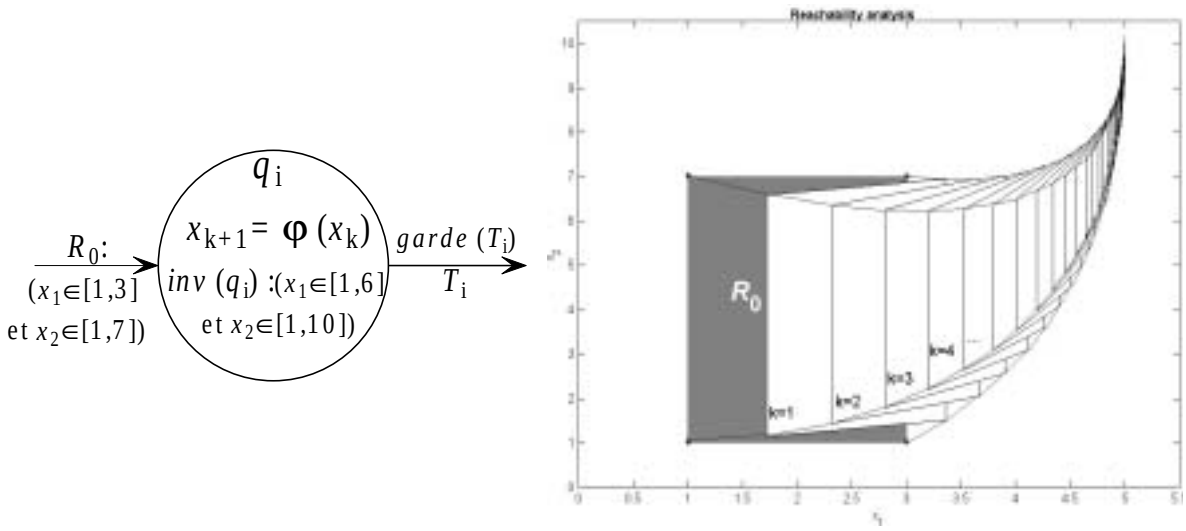
$$x_{k+1} = \varphi(x_k) = A_d x_k + B_d \tag{3.8}$$

où

$$A_d = \begin{pmatrix} 0,81 & 0 \\ 0,17 & 0,9 \end{pmatrix} \quad \text{et} \quad B_d = \begin{pmatrix} 0,9 \\ 0,9 \end{pmatrix}.$$

La région initiale est représentée par R_0 : $(x_1 \in [1, 3],\ x_2 \in [1, 7])$. L'ensemble ordonné de sommets de cette région est $\{(1, 1),\ (3, 1),\ (3, 7),\ (1, 7)\}$.

Les successeurs continus dans le sommet de l'automate considéré, à partir de la région



(a) (b)

FIG. 3.4 – Exemple de calcul des successeurs continus dans un sommet de l'automate hybride

initiale R_0 sont représentés dans la Figure 3.4(b).

Nous pouvons remarquer la convergence de la dynamique continue vers le point de coordonné $(5, 10)$. Dans ce contexte, nous avons la certitude que tous les successeurs continus, atteignable par l'évolution du système en respectant la dynamique associée au sommet de l'automate hybride considéré, ont été calculés. La convergence de la dynamique garantit aussi que l'algorithme de calcul se termine dans un nombre fini d'itérations.

■

Remarque 3.1. Nous verrons dans la suite que dans le cas général on ne calcule pas les successeurs continus d'une région initiale jusqu'à la convergence des dynamiques ou à la valeur $t = \infty$. La condition associée aux transitions (les gardes) de sortie du sommet fera que le franchissement aura lieu dans un temps fini. Ce critère d'arrêt ne joue son rôle que dans le cas des sommets puits.

■

3.3 Franchissabilité des transitions

Dans cette section nous allons nous intéresser plus particulièrement à l'analyse de la franchissabilité d'une transition dans un automate hybride à temps discret.

Comme nous l'avons déjà présenté dans le chapitre 2, l'évolution d'un automate hybride est réalisée par une alternance des pas continus et discrets. L'évolution continue, en elle-même, ne pose pas de problème, toute la difficulté provient de l'interaction événementielle. Ainsi, il faut déduire l'instant de commutation en tenant compte de l'évolution des variables d'état. Il découle une notion importante, celle de la validation d'une transition discrète.

Dans ce travail, nous ne considérons que des automates hybrides à temps discret pour lesquels les régions représentant la région initiale, l'invariant et la région cible sont convexes et les dynamiques sont linéaires.

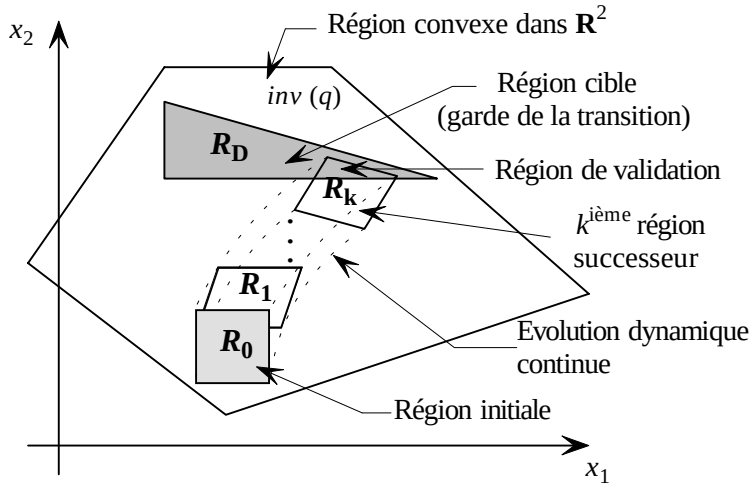


FIG. 3.5 – Représentation de la dynamique dans un sommet de l'automate hybride

Dans le but de déterminer l'instant où une transition devient franchissable, une procédure basée sur le calcul de successeurs continus dans un sommet de l'automate hybride sera utilisée. En effet, il s'agit de vérifier l'atteignabilité de la garde associée à la transition de sortie d'un sommet hybride, appelée aussi *région cible*.

D'une manière intuitive, pour un sommet de l'automate hybride la procédure de vérification d'atteignabilité d'une région cible peut être résumée comme suit :

TANT QUE *l'invariant du sommet est vérifié et aucune région cible n'est atteinte*
CALCULER *le successeur continu de la région courante.*

La vérification de la franchissabilité d'une transition sera alors réalisée par l'intersection entre la région atteinte et la région cible. La Figure 3.5 illustre le principe de l'analyse de la franchissabilité. A l'instant où le résultat de l'intersection de la région successeur courante et la région cible est non vide, on dit que la transition dont la garde est représentée par la région cible, devient validée. L'invariant limite l'évolution du système dans le sommet. Il est utilisé pour arrêter le calcul des successeurs continus.

En général, deux cas peuvent se poser :

- La région cible est atteinte alors que l'invariant est toujours vérifié. La transition est franchissable.
- La région cible n'est pas atteinte alors que l'invariant n'est plus vérifié. La construction est alors arrêtée. Il y a incohérence dont l'origine est souvent dans la modélisation.

Dans la suite de cette section, nous présentons les algorithmes qui nous permettront de décider sur la franchissabilité des transitions discrètes d'un automate hybride. Pour ce faire nous étudions deux situations particulières : 1) le cas d'un sommet possédant une seule transition d'entrée et une transition de sortie, et 2) le cas d'un sommet possédant une transition d'entrée et plusieurs transitions de sortie.

3.3.1 *Cas d'un sommet possédant une transition d'entrée et une transition de sortie*

Pour commencer l'analyse et établir des conditions et propriétés pour la validation d'une transition, considérons la structure partielle de l'automate hybride représentée dans la Figure 3.6. Il s'agit d'un sommet q_j possédant une transition (arc) d'entrée et une seule tran-

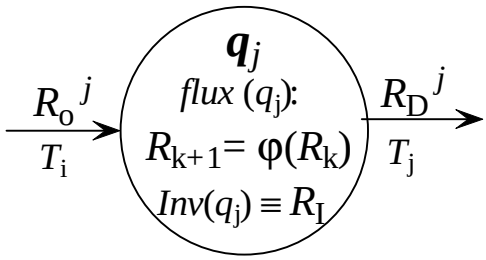


FIG. 3.6 – Sommet avec une transition d'entrée et une transition de sortie

sition de sortie (SISOT - *Single Input Single Output Transition*). L'analyse d'un sommet possédant plusieurs entrées revient au cas d'une entrée car le fonctionnement déterministe de l'automate impose qu'il n'y ait qu'une seule entrée possible au sommet à un instant donné.

Dans le sommet de la Figure 3.6, l'équation de flux continu représente l'évolution dynamique du système tant que l'invariant du sommet est vérifié. La dynamique continue est décrite par l'équation de flux $R_{k+1} = \varphi(R_k)$, où la région R_{k+1} représente le successeur continu de R_k et cette région sera obtenue en suivant la démarche présentée auparavant. L'invariant associé au sommet est représenté par $Inv(q_j) \equiv R_I$. La condition initiale est représentée par la région $R_0{}^j$ et la région cible, dont l'atteignabilité doit être vérifiée, est donnée par $R_D{}^j$ représentant la garde de la transition de sortie T_j.

Définition 3.3. Pour tout automate hybride $\mathcal{A}$, le *temps de séjour* δ_j dans un sommet $q_j \in Q$ est défini par le temps passé dans le sommet depuis l'entrée et avant le franchissement d'une transition de sortie.

■

Propriété 3.3.1. *Pour tout sommet $q_j \in Q$, une transition de sortie T_j est* validée *seulement s'il existe une valeur finie $\delta_j \in \mathbb{R}^+$ pour laquelle la région cible est atteinte.*

■

La valeur du $\delta_j \in \mathbb{R}^+$ est fonction de la condition initiale (i.e., région initiale), de la condition de franchissement de la transition de sortie T_j et de la trajectoire définie par l'équation de flux continu dans le sommet.

Propriété 3.3.2. *Si la transition T_j est validée, alors elle sera franchissable à partir de l'instant $t = \delta_j$, représentant le* temps de séjour minimal *du système dans le sommet.*

■

Remarque 3.2. Si la transition T_j n'est jamais validée et l'invariant est toujours vérifié, l'évolution dynamique restera infiniment régie par l'équation de flux continu dans le sommet q_j (c'est-à-dire, $\delta_j \to \infty$). Dans ce cas, le sommet q_j est appelé *sommet puits.*

■

L'algorithme suivant résume les étapes de l'analyse de franchissabilité pour un sommet de l'automate hybride possédant une transition de sortie.

Algorithme 3.2. *Analyse de franchissabilité d'une transition dans le cas SISOT*

Pas 1 : Initialiser la région successeur courante $succ_0 := R_0^j$.

Pas 2 : Initialiser la région atteinte par l'évolution du système dans le sommet par
$$R_A{}^j := \emptyset$$

Pas 3 : Initialiser la variable compteur $k := 0$.

Pas 4 : Tant que $(R_A{}^j \neq R_{k+1})$ et $(R_A{}^j \subseteq Inv(q_j))$ **calculer**

4.1. La région courante atteinte dans le sommet est $R_A{}^j := R_A{}^j \cup R_k$

4.2. **Si** $R_A{}^j \cap R_D{}^j \neq \emptyset$ **alors**

4.2.1. $R_D{}^j$ est atteignable à partir de $R_0{}^j$ en k pas discrets $\Rightarrow$ la transition T_j est franchissable

4.2.2. Aller au Pas 5.

4.2.3. Arrêter le calcul.

4.3. **Sinon**

4.3.1. La région successeur de la région R_k obtenue par $R_{k+1} := \varphi(R_k)$.

4.3.2. Actualiser le compteur $k := k + 1$.

Pas 5 : Tant que $(R_A{}^j \subseteq R_D{}^j)$ et $(R_A{}^j \subseteq Inv(q_j))$ **calculer**

5.1. La région successeur de la région R_k obtenue par $R_{k+1} := \varphi(R_k)$

5.2. Actualiser le compteur $k := k + 1$

5.3. La région courante atteinte dans le sommet est $R_A{}^j := R_A{}^j \cup R_k$

Pas 6 : Si $(R_A{}^j \cap R_D{}^j = \emptyset)$ **alors**
$R_D{}^j$ n'est pas atteignable à partir de $R_0{}^j \Rightarrow$ la transition T_j n'est pas validée.

■

L'algorithme se termine soit :
– s'il y a une convergence de la dynamique continue - à partir de cet instant on retrouve des successeurs continus identiques ;
– si l'invariant n'est plus vérifié - dans cette situation le système ne peut plus séjourner dans le sommet ;
– si la région cible est atteinte - dans cette situation la transition de sortie est validée et peut être franchie tant que la région atteinte dans le sommet est incluse dans la région cible.

Les successeurs de la région initiale $R_0{}^j$ sont calculés pour chaque pas discret k. La région cible dont on teste l'atteignabilité est $R_D{}^j$ et elle représente la garde associée à la transition de sortie T_j.

L'ensemble des régions atteignables $R_A{}^j$, pendant le séjour du système dans le sommet, est obtenu par l'union des régions successeurs continus atteintes avant la validation de la transition de sortie du sommet q_j. Le test d'atteignabilité est fait par l'intersection entre la région courante atteinte $(R_A{}^j)$ et la région cible $(R_D{}^j)$. L'instant où la transition de sortie devient franchissable correspond au *temps de séjour minimal* du système dans le sommet. À partir de cet instant, les successeurs continus de la région courante sont calculés tant que l'évolution continue vérifie la garde de la transition et l'invariant. Au moment où les régions

atteintes ne vérifient plus l'inclusion dans la région de franchissement ou l'invariant du sommet, on arrête le calcul des successeurs continus et cet instant correspond au *temps de séjour maximal* du système dans le sommet analysé. Les temps de séjour seront exprimés en terme de nombre de pas discrets.

Nous pouvons remarquer que si jamais l'invariant du sommet, représenté par une région R_I n'est plus vérifié, le calcul s'arrête même si la transition de sortie n'est pas validée.

Remarque 3.3. La *région atteinte* lors d'une visite dans un sommet de l'automate représente l'image de l'évolution continue du système dans ce sommet jusqu'au moment où, soit une transition de sortie devient franchissable, soit l'invariant n'est plus vérifié, soit il y a une convergence des dynamiques continues dans le sommet.

■

3.3.2 *Cas d'un sommet possédant une transition d'entrée et plusieurs transitions de sortie*

Considérons la structure partielle représentée dans la Figure 3.7 où le sommet a une transition d'entrée et plusieurs transitions de sortie (SIMOT - *Single Input Multiple Output Transitions*).

Propriété 3.3.3. *Une transition $T_j{}^l$ $(l = \overline{1,n})$ est validée seulement si la garde associée est atteignable par l'évolution continue dans un temps $t = \delta_j{}^l$ fini.*

■

Remarque 3.4. Plusieurs transitions peuvent être validées mais seulement une sera franchie.

■

Dans le cas d'un sommet dont plusieurs transitions sont validées à la fois, le fonctionnement devient non-déterministe. Afin d'assurer le déterminisme de l'évolution du système, il faut donner une politique pour décider, parmi toutes les transitions validées, quelle est la transition qui sera franchie.

Un choix possible est celui de franchir la première transition de sortie validée. Dans cette situation, considérons Δ, l'ensemble de toutes les transitions validées, $R_D{}^l$ la garde

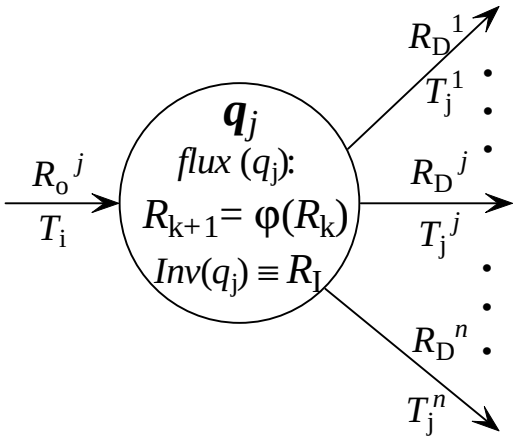


FIG. 3.7 – Sommet avec une transition d'entrée et plusieurs transitions de sortie

associée à la transition T_j^l ($l = \overline{1,n}$) et ${\delta_j}^{min}$ le temps de séjour minimal correspondant à Δ, tel que :

$${\delta_j}^{min} = \mathbf{min}({\delta_j}^l) \tag{3.9}$$

où ${T_j}^l \in \Delta$.

Propriété 3.3.4. *Dans le cas d'une évolution déterministe, la transition T_j^l pour laquelle ${\delta_j}^l = {\delta_j}^{min}$ est franchissable à partir de l'instant ${\delta_j}^{min}$.*

■

Le comportement décrit par cette propriété est naturel, car c'est le premier seuil qui détermine l'exécution d'une transition discrète.

Remarque 3.5. Si aucune transition ${T_j}^l$ ($l = \overline{1,n}$) n'est validée, l'ensemble de transitions validées Δ sera l'ensemble vide et le sommet q_j représente un *sommet puits*.

■

Dans le cas d'un sommet possédant plusieurs transitions de sortie, l'algorithme d'analyse de franchissabilité consiste dans une version itérative de l'Algorithme 3.2. Dans cet algorithme, une condition supplémentaire pour arrêter le calcul des successeurs continus sera utilisée et cette condition dépendra de la manière dont on choisit quelle est la transition qui sera franchie parmi l'ensemble des transitions franchissables.

Nous pouvons observer que du point de vue du fonctionnement de l'automate hybride, les propriétés dynamiques pour le cas SIMOT peuvent être déduites à partir des résultats obtenus dans l'analyse de la situation SISOT, en prenant en compte l'indéterminisme du franchissement entre les différentes transitions de sortie. Le fonctionnement de l'automate impose que pour sortir d'un sommet à un certain instant t seulement une transition pourra être exécutée.

Dans cette section nous avons présenté les techniques permettant d'analyser la franchissabilité d'une transition de sortie d'un sommet hybride. Cependant, pour pouvoir décider de l'atteignabilité des transitions d'un automate hybride nous avons besoin d'une méthode cohérente qui permet de réaliser une analyse globale du modèle. Proposer une telle méthode est l'objectif de la section suivante, en se basant toujours sur les résultats présentés jusqu'ici.

3.4 Automate atteignable

Afin d'achever l'analyse d'atteignabilité pour un système modélisé par un automate hybride, nous allons proposer une technique pour la construction d'un modèle qui ne contient que les trajectoires effectivement réalisables dans le système à partir d'une condition initiale spécifiée. Ce modèle est appelé *automate atteignable* et son construction est basée sur l'analyse de la franchissabilité des transitions de l'automate hybride initiale. La franchissabilité d'une transition revient à l'analyse d'atteignabilité des régions de gardes qui lui sont associées.

L'analyse d'atteignabilité constitue un problème central dans la vérification des propriétés des systèmes hybrides modélisés par des automates hybrides. Celle-ci nous permet de mieux connaître l'évolution du système dans sa globalité. Par conséquent, les informations concernant le comportement du système hybride, obtenues dans cette étape, seront ensuite utilisées lors de la synthèse de la commande. L'objectif est de retenir toutes les

informations concernant l'évolution du système. Dans ce but, l'évolution complète du système dans un sommet de l'automate q_j, lors de chaque visite, sera stockée dans une variable $trace_j$ qui lui est associée.

Dans un sommet de l'automate hybride l'évolution du système est continue. Dans la section précédente (Section 3.3, nous avons établie les conditions d'arrêt pour le calcul de l'espace atteignable dans un sommet de l'automate hybride. Dans un automate hybride, l'évolution du système est hybride (alternance des évolutions continues et discrètes) et peut comporter aussi des évolutions cycliques - démarre dans un état physique et au bout de certain temps revient dans l'état physique initial. Il s'agit donc d'une évolution plus complexe et plus difficile à analyser. Par exemple, reprenons le cas d'un thermostat : on démarre avec l'état *On* du système de chauffage, après un certain temps on arrête le chauffage car les seuil supérieur de l'intervalle de température dans lequel on veut y rester a été atteint. Ainsi, le système passe dans l'état *Off*. Le système reste dans cet état tant que la valeur de la température atteindra le seuil inférieur. A cet instant, on redémarre le chauffage et le système passe dans l'état *On*, correspondant à l'état physique initiale. Par conséquent, un sommet de l'automate modélisant un état physique du système peut être visité plusieurs fois dans un fonctionnement donné. Ces différentes visites dans le même sommet correspondent à des conditions d'entrée qui sont généralement différentes.

Propriété 3.4.1. *La* région atteinte *dans un sommet de l'automate, par l'évolution du système, est obtenue par l'union des régions atteintes à chaque visite dans ce sommet.*

■

Remarque 3.6. Due à l'évolution des variables continues, la condition initiale pour une visite dans un sommet de l'automate est généralement différente de celle obtenue lors d'une visite précédente.

■

L'espace atteignable d'un automate hybride $\mathcal{A}$ est déterminé par l'union des successeurs continus après une exécution de l'automate. Ainsi, le calcul des successeurs, et en conséquence le calcul de l'espace atteignable, s'arrête lorsqu'il y a une convergence des dynamiques. Cependant, le calcul de l'espace atteignable peut ne pas converger. La non-convergence s'explique par le fait que le calcul de l'espace atteignable par un automate hybride est, en général, un problème *non décidable* [HKPV95].

La convergence ne peut pas être vérifiée à priori, car on ne dispose pas d'une propriété garantissant la *stabilité globale* d'un système hybride. Par conséquent, pour arrêter la construction de l'automate atteignable nous avons recours à des technique numériques. Ainsi, nous allons vérifier si l'écart entre les conditions initiales à l'entrée d'un sommet lors de deux visites précédentes (successives ou non) est inférieur à une valeur ε fixée. Cette valeur imposera la précision de la construction du modèle de l'automate atteignable. Si au bout de certain temps, on retrouve à l'entrée d'un sommet une condition initiale approchée à un ε-près, alors nous avons détecté une *boucle* dans l'évolution du système. Ainsi, on décidera d'arrêter la construction de l'automate atteignable et aussi le calcul de l'espace atteignable.

Nous avons développé un algorithme pour la construction de l'automate atteignable à partir d'un automate hybride donné [KA01a]. La présentation de cet algorithme fait l'objet de la section suivante.

3.4.1 *Algorithme de construction de l'automate atteignable*

Nous utilisons les notations suivantes :

- Q représente l'ensemble des sommets de l'automate hybride
$$(Q = \{q_0, q_1, ..., q_m, q_{m+1}, ..., q_r\})\,;$$
- T représente l'ensemble des transitions de l'automate hybride. La transition T_0 est la transition d'entrée dans le sommet initial q_0, étiquetée par la d'une région initiale R_0. Une transition $T_{m,m+1}$ représente l'arc reliant les sommets q_m et q_{m+1} étiqueté par la région cible (garde de la transition) ${R_D}^{m,m+1}$ (Figure 3.8) ;
- $nb_{sommets}$ est le compteur qui mémorise le nombre des sommets ;
- C_m dénote l'ensemble des successeurs continus obtenus par l'évolution du système dans le sommet q_m ;
- $trace_n$ est une pile associée au sommet q_n. Chaque élément de la pile mémorise les caractéristiques d'une visite dans le sommet :
  - l'*indice* pour indiquer le numéro associé à la visite courante ;
  - la *condition initiale* à l'entrée du sommet pour la visite courante ;
  - l'*ensemble des successeurs continus* obtenus durant le séjour du système dans le sommet ;
  - la date à laquelle le système quitte le sommet ;
- E représentant la liste des transitions de l'automate hybride franchies pendant l'évolution du système.

Initialement, tous les ensembles et vecteurs sont vides. L'automate hybride est décrit par deux ensembles : l'ensemble de ses sommets Q et par l'ensemble de ses transitions T.

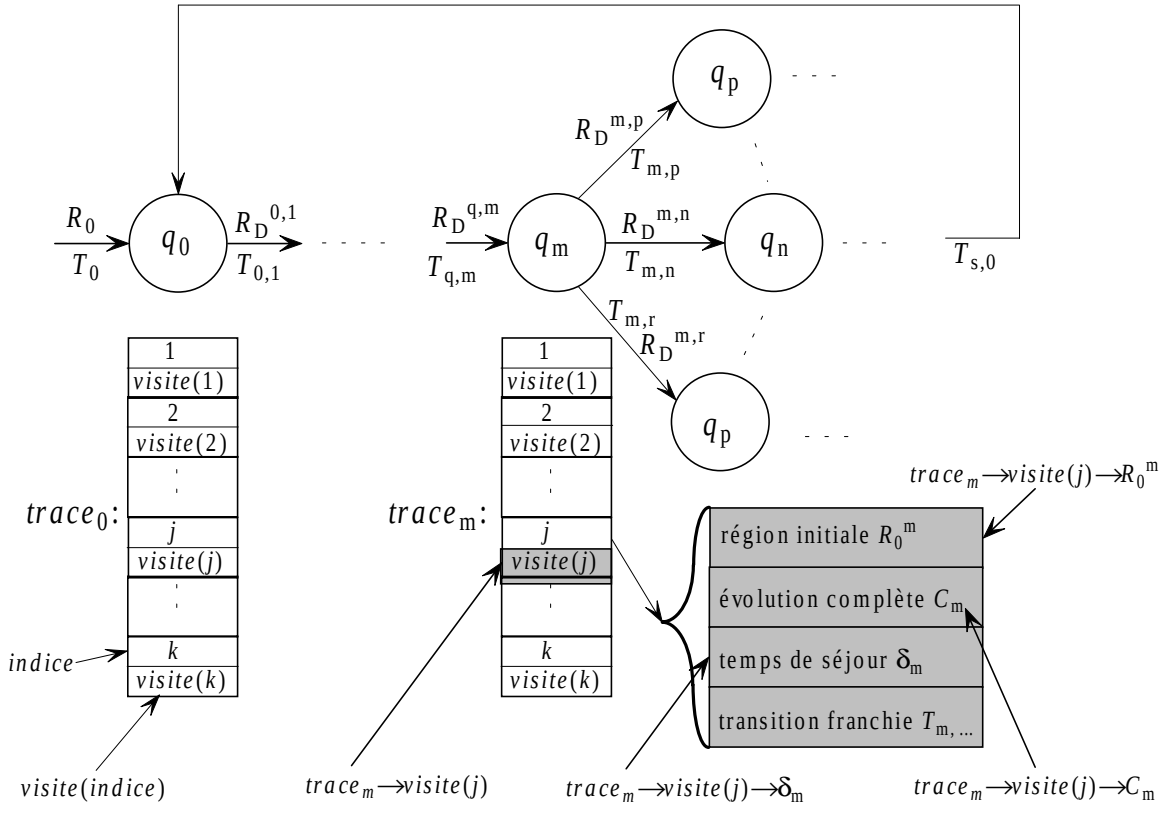


FIG. 3.8 – Présentation des variables

Algorithme 3.3. *Construction de l'automate atteignable*

Pas 1 : *Initialisation*

- Identifier le sommet initial q_0 et créer la pile $trace_0$ associée ;
- Créer la liste des transitions franchies $E = \{\}$;
- Ajouter à la liste E la transition d'entrée T_0 :
$$E := E \cup \{T_0\} ;$$
- Initialiser les variables auxiliaires :
$$compteur := 0 ; \; m := 0 ; \; n := 1 ; \; indice := 1$$

Pas 2 : Pour chaque sommet de l'automate hybride (i.e., $compteur = 0 : nb_{sommets}$) **faire**

2.1. Le sommet courant est q_m. **Si** c'est la première visite ($indice = 1$) **alors** créer le vecteur $trace_m$ associé au sommet courant q_m

2.2. Analyse de franchissabilité des transitions de sortie du sommet courant q_m (*Aller au Pas 3*)

2.3. Franchissement d'une transition de sortie validée (*Aller au Pas 4*)

Pas 3 : *Analyse de franchissabilité* *(Single Input Multiple Output Transitions)*

3.1. *Initialisation*
- La région initiale à l'entrée du sommet courant q_m est R_0 :
$$trace_m \rightarrow visite(indice) \rightarrow R_0^m := R_0 ;$$
- L'ensemble des transitions de sortie du sommet q_m est donnée par l'ensemble $\{T_{m,n} | n \in \mathbb{N}\}$, où n prend ses valeurs entre zéro et le nombre de transitions de sortie du sommet analysé ;
- Initialiser le vecteur qui mémorise l'évolution dynamique complète du système dans le sommet q_m :
$$trace_m \rightarrow visite(indice) \rightarrow C_m := \emptyset ;$$
- Initialiser la variable compteur $k := 0$.

3.2. **Pour** chaque transition de sortie $T_{m,n}$ (pour n de 1 jusqu'à la nombre de transitions de sortie) **faire**

3.2.1. Appliquer les étapes de l'Algorithme 3.2 ;

3.2.2. **Si** la transition $T_{m,n}$ est franchissable **alors**
- Mémoriser l'intervalle de franchissement dans une variable $\delta_m(n)$;
- Mémoriser l'évolution complète du système dans le sommet q_m jusqu'à la validation de la transition $T_{m,n}$ dans une variable $C_m(n) := R_A^m$.

3.3. Mémoriser la liste des transitions franchissables et parmi ces transitions, choisir celle (notée $T_{m,p}$, où $1 \leq p \leq n$) qui correspond à la politique de franchissement imposée et *Aller au Pas 2.3*.

Pas 4 : *Franchissement de la transition de sortie* $T_{m,p}$

4.1. *Actualisation de la pile* $trace_m$ *associée au sommet* q_m
- Mémoriser dans la pile $trace_m$:
  - le numéro de la variable *indice* correspondant à la visite courante dans le sommet q_m :
$$trace_m \rightarrow indice := indice ;$$

– l'ensemble C_m qui mémorise l'évolution du système pendant son séjour dans q_m, lors de la visite courante :
$$trace_m \rightarrow visite(indice) \rightarrow C_m := C_m(p)\,;$$
– le temps de séjour $\delta_m$du système dans le sommet q_m avant le franchissement de la transition $T_{m,p}$
$$trace_m \rightarrow visite(indice) \rightarrow \delta_m := \delta_m(p)\,;$$
– la transition $T_{m,p}$ dans la liste E_{indice} correspondant à la visite courante : $E := E \cup T_{m,p}$

4.2. *Changement de sommet dans l'automate*

4.2.1. Le sommet courant est q_p (sommet destination de la transition $T_{m,p}$;

4.2.2. **SI** q_p est le sommet initial q_0 **ALORS**
- Mettre à zéro la valeur de la variable *compteur* qui compte le nombre de sommets de l'automate : $compteur := 0$;
- **Si** la condition initiale ${R_0}^p$ à l'entrée du sommet q_p est identique ou s'approche à ε-près
$$dist(R_0, trace_p \rightarrow visite(\overline{s) \rightarrow R_0^p}) \leq \varepsilon,$$
pour une région initiale $s = \overline{1,\ indice - 1}$ à l'entrée du sommet q_p obtenue lors d'une visite précédente, **alors** une *boucle* est détectée dans l'évolution du système ; *Aller au Pas 5* ;
- **Sinon** Incrémenter la variable correspondant au nombre de visites $indice := indice + 1$ et réinitialiser la variable $n := 1$; *Aller au Pas 2.*

4.2.3. **Si** q_p est le sommet interdit **alors**
- Une fois que le système a atteint ce sommet, il restera indéfiniment.
- **S**'il y a d'autres transitions de sortie contrôlables à partir du sommet source, **alors** franchir une de ces transitions et *Aller au Pas 4.*

4.2.4. **Si** q_p ne représente pas le sommet initial q_0, ni le sommet interdit F et $compteur < nb_{sommets}$ **alors**
- Incrémenter le nombre de sommets analysés
$$compteur := compteur + 1$$
et réinitialiser la variable $n := 1$;
- La condition initiale à l'entrée du sommet est donnée par la région atteinte dans le sommet q_m avant le franchissement de la transition $T_{m,p}$: ${R_0}^p := {R_A}^m(\delta_m(p))$;
- Actualiser les valeurs des variables : $m := p$;
- Identifier les transitions de sortie du sommet q_m ; *Aller au Pas 2.1.*

Pas 5 : *Automate atteignable*

5.1. Comparer l'ensemble des transitions franchies, stockés dans la liste E lors de chaque visite, avec l'ensemble des transitions de l'automate hybride initial T ;

5.2. **S**'il y a des transitions qui ne sont jamais franchies **alors**
- Actualiser l'automate hybride initial par la suppression de toutes ces transitions et de tous les sommets qui ne sont plus reliés aux autres sommets.

5.3. **FIN.**

■

L'algorithme proposé synthétise les étapes de l'analyse d'atteignabilité pour un automate hybride et fournit comme résultat le modèle d'automate atteignable, c'est à dire

l'automate qui modélise toutes les trajectoires possibles du système à partir des conditions initiales données. Toute l'évolution du système est mémorisée dans l'ensemble des piles *trace*.

Nous illustrons ci-dessus les résultats obtenus par l'utilisation de cet algorithme sur le modèle d'automate hybride de la ligne de production présenté dans le chapitre précédent.

3.4.2 *Exemple d'application*

Afin d'illustrer les résultats obtenus par l'utilisation de cet algorithme, reprenons l'exemple de la ligne de production considéré dans le chapitre 2 (Section 2.5).

Considérons le modèle de l'automate hybride présenté dans la Figure 2.14 et appliquons l'algorithme 3.3. Pour améliorer la clarté de la présentation, nous reprenons l'automate hybride dans la Figure 3.9 avec ses dynamiques représentées dans le Tableau 3.1.

Le modèle comporte neuf sommets, dont un modélise le sommet interdit. Aux sommets où les dynamiques continues ont lieu, sont associées des équations différentielles continues linéaires. Nous illustrons ici que l'utilisation de l'algorithme pour mener l'analyse dans l'état *On* du système lors d'une première visite. Dans l'Annexe B, nous présentons une itération complète de l'algorithme jusqu'au moment où le système revient dans l'état initial avec le flux d'entrée démarré.

Pas 1 : *Initialisation*

• Initialement, le niveau des pièces dans les deux stocks de la ligne de production est $x_1 \in [1, 2]$ pièces dans le stock S_1 et $x_2 \in [1, 7[$ pièce dans le stock S_2. On observe que la condition initiale est exprimée par une région. Ceci nous permettra de faire une analyse puis une synthèse qui seront robustes vis-à-vis des incertitudes du modèle. Le nombre de serveurs qui composent les deux groupes de machines, M_1 et M_2, est $k_1 = 3$ et $k_2 = 7$ machines.

• Le sommet initial est (A_0, B_0). Initialisation des variables *compteur* $:= 0$, *indice* $:= 1$, $m := 0$ et $n := 1$.

• Le sommet courant a 5 transitions de sortie dont les conditions de franchissement sont :

– $(x_1 < 3)$ et $(x_2 \geq 7)$;
– $(x_1 \geq 3)$ et $(x_2 < 7)$;
– $(x_1 \geq 3)$ et $(x_2 \geq 7)$;
– $F_{OFF}/y_1 := 0$ où F_{OFF} est un événement contrôlable dont la date d'occurrence peut être décidée, et
– la transition qui mène vers le sommet interdit dont la garde est $(x_1 \notin [1, 4])$ et $(x_2 \notin [1, 8])$.

• *Aller au Pas 2.*

Analyse d'atteignabilité dans le sommet (A_0, B_0)

Pas 2 : Étant la première visite dans le sommet (A_0, B_0), créer le vecteur trace correspondant au sommet - $trace_{(A_0, B_0)}$ - et commencer l'analyse de franchissabilité des transitions de sortie.

Pas 3 :*Analyse de la franchissabilité des transitions de sortie du sommet* (A_0, B_0)

Le système peut séjourner dans le sommet (A_0, B_0) tant que l'invariant du sommet - représenté par la condition logique $(x_1 \in [1, 4])$ et $(x_2 \in [1, 8])$ - est vérifié.

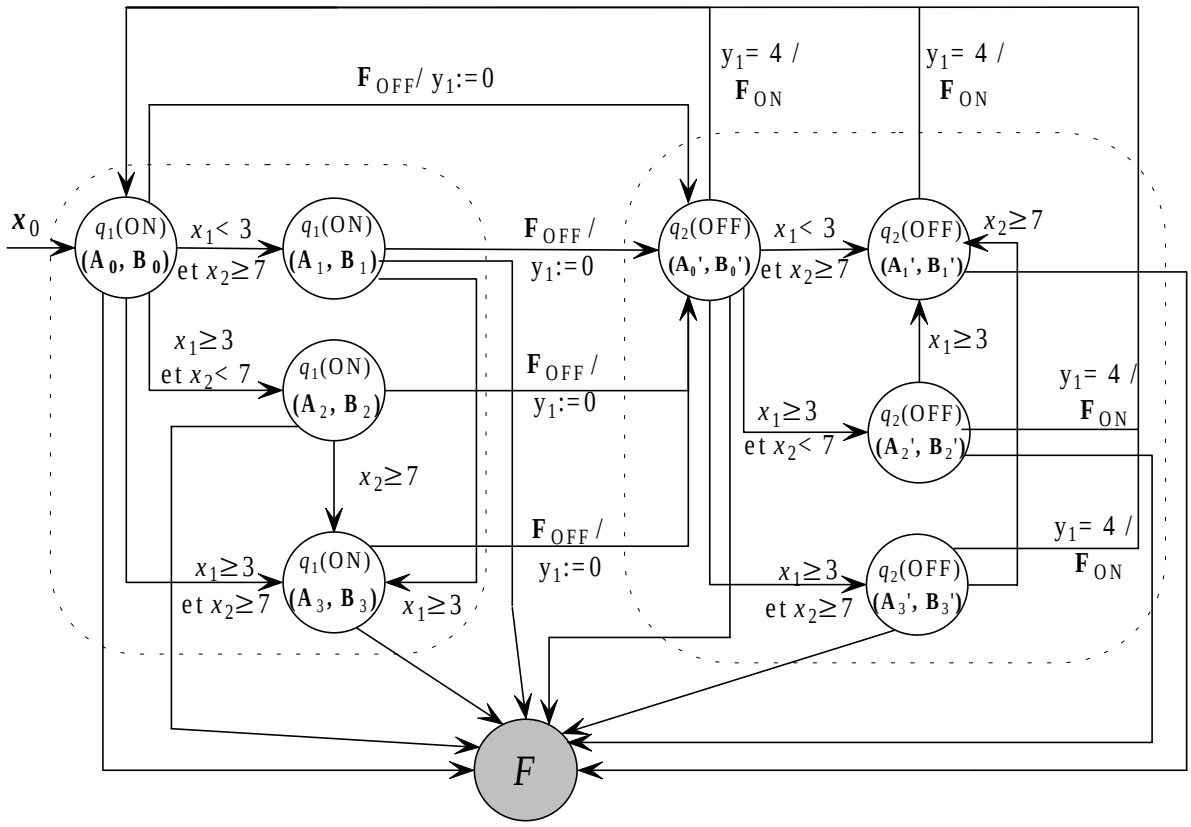


FIG. 3.9 – Automate hybride avec commutations autonomes modélisant la ligne de production

Etat du flux / Cond. logique	*Démarré*		*Arrêté*	
	Sommet	Dynamique continue	Sommet	Dynamique continue
$(x_1 < k_1)$ et $(x_2 < k_2)$	(A_0, B_0)	$\begin{bmatrix} -a_1 & 0 \\ a_1 & -a_2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}+\begin{bmatrix} d \\ 0 \end{bmatrix}$	(A_0', B_0')	$\begin{bmatrix} -a_1 & 0 \\ a_1 & -a_2 \end{bmatrix}\begin{bmatrix} k_1 \\ x_2 \end{bmatrix}$
$(x_1 < k_1)$ et $(x_2 \geq k_2)$	(A_1, B_1)	$\begin{bmatrix} -a_1 & 0 \\ a_1 & -a_2 \end{bmatrix}\begin{bmatrix} x_1 \\ k_2 \end{bmatrix}+\begin{bmatrix} d \\ 0 \end{bmatrix}$	(A_1', B_1')	$\begin{bmatrix} -a_1 & 0 \\ a_1 & -a_2 \end{bmatrix}\begin{bmatrix} x_1 \\ k_2 \end{bmatrix}$
$(x_1 \geq k_1)$ et $(x_2 < k_2)$	(A_2, B_2)	$\begin{bmatrix} -a_1 & 0 \\ a_1 & -a_2 \end{bmatrix}\begin{bmatrix} k_1 \\ x_2 \end{bmatrix}+\begin{bmatrix} d \\ 0 \end{bmatrix}$	(A_2', B_2')	$\begin{bmatrix} -a_1 & 0 \\ a_1 & -a_2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$
$(x_1 \geq k_1)$ et $(x_2 \geq k_2)$	(A_3, B_3)	$\begin{bmatrix} -a_1 & 0 \\ a_1 & -a_2 \end{bmatrix}\begin{bmatrix} k_1 \\ k_2 \end{bmatrix}+\begin{bmatrix} d \\ 0 \end{bmatrix}$	(A_3', B_3')	$\begin{bmatrix} -a_1 & 0 \\ a_1 & -a_2 \end{bmatrix}\begin{bmatrix} k_1 \\ k_2 \end{bmatrix}$

TAB. 3.1 – Dynamiques correspondant aux sommets de l'automate hybride obtenu par la décomposition structurelle

L'analyse de franchissabilité des transitions de sortie nous permet d'observer que seulement 3 transitions parmi les 5 sont franchissables. Ces transitions sont :

– celle qui mène vers le sommet interdit ;
– celle étiquetée par l'événement contrôlable F_{OFF} dont la date d'occurrence n'est pas

fixé (i.e. $k \in [0, \infty]$) ;

– celle étiquetée par la condition logique ($x_1 \geq 3$) et ($x_2 < 7$).

Parmi les trois transitions de sortie deux sont validées : celle étiquetée par F_{OFF} et celle étiquetée par la condition logique ($x_1 \geq 3$) et ($x_2 < 7$). La première est validée dès que le système a atteint le sommet et la deuxième par l'écoulement du temps au bout de 2 pas discrets. L'écoulement du temps ne peut pas être empêchée, donc le franchissement de cette dernière transition ne peut pas être évité. Donc, cette transition sera franchie dès que la condition de garde associée est validée. L'intérêt est d'avoir des résultats de l'analyse pertinents pour la synthèse de la commande, nous allons franchir la transition vers le sommet (A_2, B_2) mais on prendra aussi en compte le non-déterminisme introduit par la présence de la transition contrôlable.

Pas 4 : *Franchissement de la transition* $T_{(A_0, B_0),(A_2, B_2)}$

Actualisation de la pile $trace_{(A_0, B_0)}$ *associée au sommet* (A_0, B_0)

Mémoriser dans la pile $trace_{(A_0, B_0)}$:

– l'indice correspondant à la visite courante : $indice = 1$;

– la condition initiale à l'entrée du sommet (A_0, B_0) est représentée par la région $R_0^{(A_0, B_0)} = [1, 2] \times [1, 7]$;

– l'évolution complète $C_{(A_0, B_0)}(1)$ obtenue par calcul numérique ;

– le temps de séjour dans le sommet (A_0, B_0) avant l'évolution vers (A_2, B_2) est $\delta_{(A_0, B_0)}(1) = 2$ pas discrets ;

– actualiser $E(1) := \{T_0, T_{(A_0, B_0),(A_2, B_2)}\}$ la liste des transitions franchies.

Changement de sommet dans l'automate

- Le sommet courant est (A_2, B_2) ;
- La condition initiale à l'entrée du sommet est la région

$$R_A^{(A_0, B_0)} = \{(2.318, 1.444),\ (2.989, 1.740),\ (2.989, 6.653),\ (2.318, 6.356)\}$$

- Le sommet (A_2, B_2) n'est pas le sommet initial et le $compteur = 1 < 9$, alors on incrémente le nombre des sommets analysés $compteur := compteur + 1 = 2$;
- Il s'agit d'une première visite dans le sommet (A_2, B_2) (car $indice = 1$), donc créer la pile $trace_{(A_2, B_2)}$
- La condition initiale à l'entrée du sommet est $R_0^{(A_2, B_2)} := R_A^{(A_0, B_0)}$;
- Il y a 3 transitions de sortie ; *Aller au Pas 2.1.*

Analyse d'atteignabilité dans le sommet (A_2, B_2)

Pas 2 : Étant la première visite dans le sommet (A_2, B_2), commencer l'analyse de franchissabilité des transitions de sortie.

Pas 3 :*Analyse de la franchissabilité des transitions de sortie du sommet* (A_2, B_2)

Parmi les transitions de sortie 2 sont franchissables pendant le séjour du système dans le sommet. La transition étiquetée par l'événement contrôlable F_{OFF} est validée dès que le système a atteint le sommet (A_2, B_2), tandis que l'autre qui mène vers le sommet interdit est validée au bout de 3 pas discrets.

Pas 4 : *Franchissement de la transition* $T_{(A_2, B_2),F}$

Actualisation de la pile $trace_{(A_2, B_2)}$ *associée au sommet* (A_2, B_2)

Mémoriser dans la pile $trace_{(A_2, B_2)}$:

– l'indice correspondant à la visite courante : $indice = 1$;

– la condition initiale à l'entrée du sommet (A_2, B_2) est représentée par la région

$R_0^{(A_2,\,B_2)}$;

– l'évolution complète $C_{(A_2,\,B_2)}(1)$ obtenue par calcul numérique ;

– le temps de séjour dans le sommet $(A_2,\,B_2)$ avant l'évolution vers F est $\delta_{(A_2,\,B_2)}(1) = 2$ pas discrets ;

– actualiser $E(1) := \{T_0,\, T_{(A_2,\,B_2),F}\}$ la liste des transitions franchies.

Changement de sommet dans l'automate

• Le sommet courant est F ;

• La condition initiale à l'entrée du sommet est la région
$R_A^{(A_2,\,B_2)} = \{(3.602, 2.560),\, (4.052, 2.960),\, (4.052, 6.599),\, (3.602, 6.200)\}$

• Le sommet F est le sommet interdit et une fois atteint, le système y restera indéfiniment ;

• Il y a une transition contrôlable qui est validée avant que $T_{(A_2,\,B_2),F}$ devienne franchissable ; il s'agit de la transition $T_{(A_2,\,B_2),(A_0',\,B_0')}$

Pas 4 : *Franchissement de la transition* $T_{(A_2,\,B_2),(A_0',\,B_0')}$

Actualisation de la pile $trace_{(A_2,\,B_2)}$ *associée au sommet* $(A_2,\, B_2)$

Mémoriser dans la pile $trace_{(A_2,\,B_2)}$:

– l'indice correspondant à la visite courante : $indice = 1$;

– la condition initiale à l'entrée du sommet $(A_2,\, B_2)$ est représentée par la région $R_0^{(A_2,\,B_2)}$;

– l'évolution complète $C_{(A_2,\,B_2)}(1)$ obtenue par calcul numérique ;

– le temps de séjour dans le sommet $(A_2,\, B_2)$ avant l'évolution vers F est $\delta_{(A_2,\,B_2)}(1) \in [0,\, 2]$ pas discrets ;

– actualiser $E(1) := \{T_0,\, T_{(A_2,\,B_2),F},\, T_{(A_2,\,B_2),(A_0',\,B_0')}\}$ la liste des transitions franchies.

Changement de sommet dans l'automate

• Le sommet courant est $(A_0',\, B_0')$;

• Les conditions initiales à l'entrée du sommet dépendent de l'instant de commutation. Ainsi,

1.) si la transition $T_{(A_2,\,B_2),(A_0',\,B_0')}$ est franchie à l'instant 0
$R_A^{(A_2,\,B_2)}(0) = \{(2.318, 1.444),\, (2.989, 1.740),\, (2.989, 6.653),\, (2.318, 6.356)\}$;

2.) si la transition $T_{(A_2,\,B_2),(A_0',\,B_0')}$ est franchie à l'instant 1
$R_A^{(A_2,\,B_2)}(1) = \{(2.804, 1.796),\, (3.353, 2.180),\, (3.353, 6.625),\, (2.804, 6.241)\}$;

3.) si la transition $T_{(A_2,\,B_2),(A_0',\,B_0')}$ est franchie à l'instant 2
$R_A^{(A_0',\,B_0')}(2) = \{(3.202, 2.199),\, (3.652, 2.641),\, (3.652, 6.663),\, (3.202, 6.221)\}$.

• Le sommet $(A_0',\, B_0')$ n'est pas le sommet initial et le $compteur = 2 < 9$, alors on incrémente le nombre des sommets analysés $compteur := compteur + 1 = 3$;

• Il s'agit d'une première visite dans le sommet $(A_0',\, B_0')$ (car $indice = 1$), donc créer la pile $trace_{(A_0',\,B_0')}$

• La condition initiale à l'entrée du sommet est $R_0^{(A_0',\,B_0')} := R_A^{(A_2,\,B_2)(i)}$, où $i = 0,\, 1,\, 2$ est fonction de l'instant de commutation ;

• Il y a 5 transitions de sortie ; *Aller au Pas 2.1.*

■

Pour une précision de calcul fixée à $\varepsilon = 10^{-3}$ et pour des conditions initiales spécifiées, en utilisant l'Algorithme 3.3, par calcul numérique on obtient des informations concernant les transitions qui ne sont jamais franchies. Finalement, pour le sommet $(A_0,\, B_0)$ dont les transitions de sortie sont en nombre de 5, seulement 2 transitions seront franchies :

– la transition dont la garde est représentée par l'événement contrôlable F_{OFF}, toujours franchissable depuis l'instant ou le système va entrer dans le sommet (A_0, B_0),

– la transition dont la garde est représentée par la condition ($x_1 \geq 3$)et($x_2 < 7$) qui devient franchissable par l'écoulement du temps qui engendre également l'évolution des variables d'états continues (dynamique croissante), et

Les transitions jamais franchissables à partir de cet sommet sont la transition qui mène vers le sommet interdit F, car la dynamique correspondante au sommet est croissante. Par conséquent, avant la validation de la condition de garde associée à la transition qui mène vers le sommet interdit l'autre transition étiquetée par la condition logique ($x_1 \geq 3$) et ($x_2 < 7$) est franchissable. Pendant le séjour du système dans le sommet analysé son évolution ne valide jamais les transitions étiquetées par la condition logique ($x_1 < 3$) et ($x_2 \geq 7$), respectivement ($x_1 \geq 3$) et ($x_2 \geq 7$). Ainsi, on détermine l'ensemble des transitions qui ne sont jamais franchies pendant l'évolution du système.

Avec la précision considérée, la convergence de l'algorithme est obtenue au bout de 5 visites dans le sommet initial $q_1(ON)$, où le flux d'entrée est démarré. La condition initiale retrouvée à l'entrée du sommet pour cette visite, correspond à la condition initiale obtenue lors de la 3ème visite. Par conséquent, l'algorithme de construction de l'automate atteignable à partir du modèle de l'automate hybride initiale (Figure 3.9) se termine dans un nombre fini d'itérations. Le modèle de l'automate atteignable résultant pour la ligne de production est illustré dans la Figure 3.10.

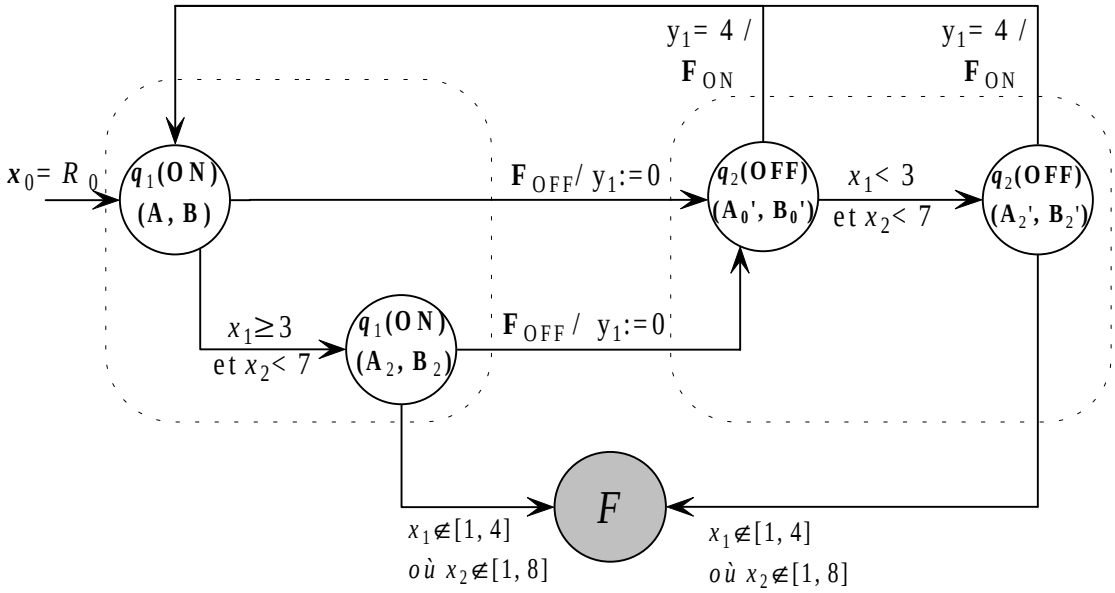


FIG. 3.10 – Automate atteignable pour la ligne de production

Nous pouvons remarquer que dans le système, suite à des simplifications du modèle résultant de l'analyse de franchissabilité des transitions du modèle initial, la dimension du modèle d'automate hybride a été réduite. Il comporte 5 sommets dont un représente le sommet interdit. L'évolution hybride est régie seulement par quatre dynamiques continues dont l'interaction est réalisée par le franchissement d'une transition. Les quatre dynamiques correspondant aux sommets de l'automate atteignable sont données dans le Tableau 3.2.

Nous pouvons, alors, conclure que les résultats d'analyse nous fournissent des informations exhaustives sur le comportement dynamique du système. Ceci nous permet d'ap-

Etat du flux / **Cond. logique**	*Démarré*		*Arrêté*	
	Sommet	Dynamique continue	Sommet	Dynamique continue
$(x_1 < k_1)$ et $(x_2 < k_2)$	(A_0, B_0)	$\begin{bmatrix} -a_1 & 0 \\ a_1 & -a_2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}+\begin{bmatrix} d \\ 0 \end{bmatrix}$	(A_0', B_0')	$\begin{bmatrix} -a_1 & 0 \\ a_1 & -a_2 \end{bmatrix}\begin{bmatrix} k_1 \\ x_2 \end{bmatrix}$
$(x_1 \geq k_1)$ et $(x_2 < k_2)$	(A_2, B_2)	$\begin{bmatrix} -a_1 & 0 \\ a_1 & -a_2 \end{bmatrix}\begin{bmatrix} k_1 \\ x_2 \end{bmatrix}+\begin{bmatrix} d \\ 0 \end{bmatrix}$	(A_2', B_2')	$\begin{bmatrix} -a_1 & 0 \\ a_1 & -a_2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

TAB. 3.2 – Dynamiques correspondantes aux sommets de l'automate hybride atteignable

porter des simplifications dans le modèle initial, donc de réduire la complexité du modèle. Les conditions initiales sont des régions de l'espace d'état continu. Ceci nous permet de construire l'automate atteignable pour toute une classe de conditions initiales ponctuelles, donc confère une certaine robustesse au modèle. Les résultats ainsi obtenus pourront ensuite être exploitées dans l'étape de synthèse de la commande.

3.4.3 *Analyse de l'algorithme*

L'analyse dynamique d'un système hybride modélisé par un automate hybride, dont les dynamiques continues sont exprimées par des équations d'état où la matrice A est quelconque, est un problème difficile et dans plupart des cas, seulement une solution numérique est possible.

L'algorithme proposé permet la construction de l'automate atteignable à partir d'un modèle d'automate hybride initial et dans des conditions initiales données. Cependant, la convergence de l'algorithme ne peut pas être garanti a priori. La non-convergence s'explique par le fait que le calcul de l'espace atteignable par un automate hybride est en général un problème non décidable [HKPV95].

En effet, pendant le fonctionnement du système les sommets de l'automate seront visitées plusieurs fois. Due aux dynamiques continues associées aux variables continues, la valeur de la condition initiale est différente à chaque visite dans le sommet de l'automate. Ne disposant pas d'une propriété qui permette de vérifier la stabilité globale d'un système hybride, la construction de l'automate atteignable peut être tout de même arrêtée de manière numérique. Ainsi, pour arrêter la construction de l'automate atteignable, pour la visite courante il faut retrouver à l'entrée du sommet initial une condition initiale qui va approcher à un ε-près la valeur trouvée lors d'une visite précédente, où ε est une précision qu'on impose. Il est évident que la précision du modèle de l'automate atteignable dépendra directement de la valeur ε choisie.

L'algorithme que nous avons proposé pour la construction de l'automate atteignable, qui ne modélise que les évolutions possibles du système pour une condition initiale donnée, a été développé en concordance avec notre objectif final, qui est la synthèse de la commande par supervision. Par conséquent, le modèle de l'automate atteignable modélise aussi les évolutions non-désirées du système par des sommets interdits. La structure du modèle est minimale dans le sens où dans chaque sommet on prend en compte seulement les transitions de sortie qui peuvent être franchies. Cependant, elle dépend de la condition initiale.

La classe d'application considérée dans notre travail est représenté par des systèmes

pour lesquelles les évolutions dans les sommets de l'automate sont stables. On arrête faire évoluer les stocks des que l'évolution vers les sommets interdits est possible. Ainsi, toutes les variables d'état sont bornées par les spécifications de fonctionnement. Par conséquent, nous n'aurions jamais un système divergent en raison des hypothèses de construction.

3.5 Conclusions

Notre approche de synthèse de la commande est basée sur la modélisation du système à commander par un automate hybride en temps discret. Généralement, l'analyse du comportement dynamique des systèmes hybrides modélisés par des automates hybrides passe par la construction de l'espace atteignable par l'évolution du système. Le résultat de ce calcul est concrétisé par le modèle d'automate hybride atteignable qui ne modélise que les évolutions effectivement réalisables par le système à partir d'une condition initiale donnée.

Ce chapitre nous a permis d'introduire l'algorithme que nous proposons pour construire l'automate atteignable. La structure de ce modèle est minimale, dans le sens où il représente strictement les évolutions possibles depuis chaque sommet. Ce modèle a été développé en concordance avec l'objectif de synthèse d'un superviseur.

L'objectif de notre démarche de synthèse d'une commande supervisée pour les systèmes hybrides est de déterminer les instants de commutation d'un sommet de l'automate hybride vers l'autre tel que les sommet interdits ne soient jamais atteignables. La méthode que nous proposons est basée sur une extension de la théorie de la commande supervisée développée pour les systèmes à événements discrets temporisés par Brandin et Wonham [BW94]. Afin d'appliquer la théorie proposée par Brandin et Wonham, tout d'abord nous cherchons à répondre à la question "Quelles sont toutes les évolutions possibles du système ?". Répondre à cette question revient à abstraire un automate à états finis à partir de l'automate atteignable en utilisant les résultats de l'analyse.

Dans le chapitre suivant, nous introduisons la problématique de la synthèse d'une commande supervisée et nous proposons une approche pour la synthèse d'un superviseur optimal.

Chapitre 4

Synthèse de la commande

Jusqu'à présent, nous avons présenté la démarche à suivre pour obtenir l'automate atteignable à partir d'un automate hybride initial modélisant le système étudié. Un algorithme permettant la construction de ce modèle, qui ne modélise que les évolutions possibles dans le système pour une condition initiale donnée, a été aussi proposé. L'automate atteignable représente le modèle de départ pour la synthèse de la commande.

Ce chapitre est consacré à la présentation de la méthode que nous avons développée permettant la synthèse d'un modèle de commande pour un système dynamique hybride. À partir de l'automate atteignable, l'utilisation du temps discrétisé permet d'obtenir un modèle équivalent, représenté par un automate à états finis, modélisant le système hybride. Les techniques de synthèse d'une commande supervisée proposées par les travaux de Ramadge et Wonham peuvent être appliquées pour obtenir un superviseur optimal. Le modèle de commande résultant de notre approche est représenté par un automate temporisé.

4.1 Problématique

Dans notre travail, le procédé à commander et les spécifications de fonctionnement imposées par le cahier de charges sont modélisés par un automate hybride. Les évolutions non-désirées du système, où les spécifications ne sont plus vérifiées sont modélisées par des sommets interdits. L'analyse d'atteignabilité des sommets du graphe nous a permis de vérifier l'atteignabilité de ces sommets. Par conséquent, si l'automate atteignable résultant de l'analyse contient au moins un sommet interdit, alors la synthèse d'un modèle de commande est nécessaire afin d'empêcher l'évolution du système vers ces sommets.

La *problématique de la synthèse* peut être formulée comme suit : étant donné un système dynamique hybride, trouver une manière systématique d'intervenir dans son évolution telle que les spécifications imposées par le cahier de charges soient toujours vérifiées.

Des méthodes efficaces permettant de résoudre ce problème ont été développées pour les systèmes continus et les systèmes à événements discrets. Le fait que les systèmes hybrides fassent intervenir deux types des dynamiques, une dynamique continue et une autre discrète, implique une complexité plus importante du problème de synthèse.

Dans la littérature, la plupart des approches s'adressent à une classe assez restreintes de systèmes hybrides. Notamment, à des systèmes hybrides dont les fonctions d'évolutions continues sont supposées linéaires découplées.

Dans ce chapitre, nous présentons notre approche de synthèse de la commande pour les systèmes hybrides basée sur une extension de la théorie classique de la commande supervisée développée par Ramadge et Wonham pour les systèmes à événements discrets ([RW87],[RW89]). Un aperçu globale de la démarche de synthèse est présenté dans [Kur02]. La présentation détaillée de cette démarche fait l'objet de ce chapitre.

L'approche de synthèse de la commande se décompose en plusieurs étapes (Figure 4.1).

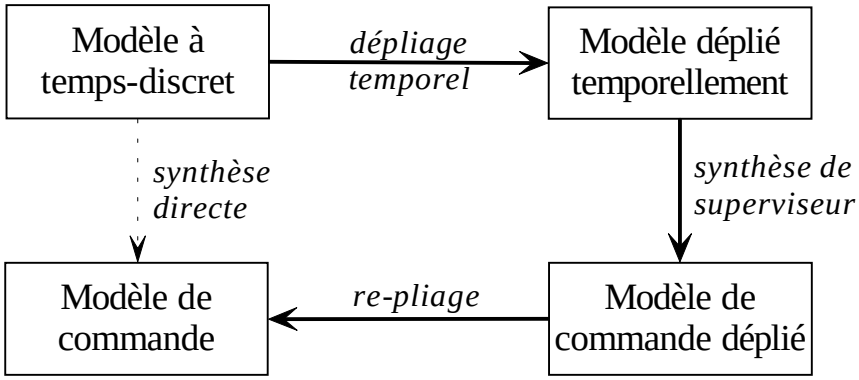


FIG. 4.1 – Étapes de la synthèse d'un superviseur

La démarche de synthèse démarre avec un modèle automate à temps discret représenté par le modèle automate atteignable obtenu lors de l'analyse d'atteignabilité. La construction de ce modèle est détaillée dans le chapitre précédent (Section 3.4). À partir de ce modèle, en se basant sur le cadre théorique développé par Brandin et Wonham pour les systèmes à événements discrets temporisés [BW94], l'idée est de construire un modèle équivalent représenté par un automate à états finis classique. Ce modèle peut être obtenu par le dépliage temporel de la dynamique continue du système, en se basant sur la représentation en temps discret de la dynamique continue du système et en exploitant les

résultats de l'analyse d'atteignabilité. Le modèle résultant est appelé *modèle déplié*. Pour ce modèle, la théorie classique de la commande supervisée devient applicable. S'il existe, le superviseur résultant sera le plus permissif possible. Le modèle de commande obtenu par la technique de synthèse de superviseur est donné sous la forme d'un automate à états finis dont la taille en général est importante en terme de nombre de sommets. Ce modèle est appelé *modèle de commande déplié*. Dans le but d'avoir une représentation plus compacte du modèle de commande, une procédure de re-pliage est proposée. Le modèle de commande ainsi obtenu sera représenté par un automate temporisé.

Dans la suite de ce chapitre, chaque étape de la démarche de synthèse sera détaillée.

4.2 Dépliage temporel

Parmi les approches proposées dans la littérature, nous n'avons retenu que celle qui est en relation directe avec notre recherche et dans laquelle nous avons trouvé des éléments de réponse à l'objectif que nous nous sommes fixés.

Notre objectif est d'élargir la théorie de la commande supervisée développée par Ramadge et Wonham pour l'appliquer à des systèmes hybrides. Dans ce contexte, nous nous sommes particulièrement intéressés à l'approche proposée par Brandin et Wonham [BW94] élaborée pour modéliser les processus temporisés et procéder ensuite à la synthèse de superviseur.

4.2.1 *Approche Brandin-Wonham*

Cette approche pour la commande supervisée de systèmes à événements discrets temporisés (SEDT) se base sur les travaux de Ramadge et Wonham, auxquels les contraintes temporelles sur la dates d'occurrence des événements sont incorporées.

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. Donc, la prise en compte du temps 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 modélise 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 $\mathcal{A}_{act} = (Q_{act}, \sum_{act}, \delta_{act}, q_{act,0}, Q_{act,m})$ avec :

- Q_{act} un ensemble fini d'états logiques ;
- $\sum_{act}$ un ensemble fini d'événements ;
- δ_{act} la fonction de transition ;
- $q_{act,0}$ l'état initial ;
- $Q_{act,m}$ l'ensemble des états marqués.

Chaque événement du procédé est instantané et il est exécuté à n'importe quel instant t du temps réel.

Exemple 4.1. Cet exemple est une représentation d'un automate non temporisé qui permettra d'illustrer progressivement la construction du modèle temporisé. Considérons le graphe d'activités représenté dans la Figure 4.2.

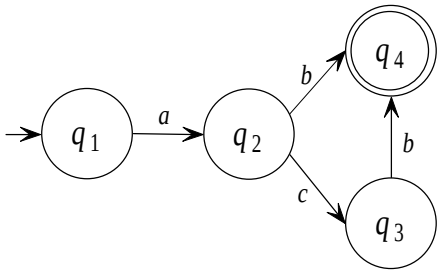


FIG. 4.2 – Modèle logique

L'automate est composé de quatre états, dont l'état initial est q_1 et dont l'état final est q_4. L'évolution de l'automate est réalisée par l'occurrence des l'un de trois événements $\{a, b, c\}$, qui étiquettent les arcs représentant les transitions du modèle.

Chacun des ensembles définis ci-dessus peut être explicité :

- $Q_{act} = \{1, 2, 3, 4\}$;
- $\sum_{act} = \{a, b, c\}$;
- $q_{act,0} = q_1$;
- $Q_{act,m} = \{q_4\}$.

■

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.

À chaque événement $a \in \sum_{act}$ on associe un intervalle $[l_a, u_a]$, $l_a \in \mathbb{N}$ 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 $\sum_{spe}$.
- Un événement a est appelé *lointain* si $u_a = \infty$. L'ensemble des événements lointains est noté $\sum_{rem}$.

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

$$\sum_{act} = \sum_{spe} \cup \sum_{rem}.$$

À 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 $\mathcal{A} = (Q, \sum, \delta, q_0, Q_m)$ dérivé de l'automate $\mathcal{A}_{act}$. Cet automate est défini de la façon suivante :

- Q est l'ensemble des états pour lequel :

$$t_a = \begin{cases} [0,\, u_a] & \text{si} \quad a \in \sum_{spe} \\ [0,\, l_a] & \text{si} \quad a \in \sum_{rem} \end{cases}$$

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

$$q = \{q_{act} | \{t_a | a \in \sum_{act}\}\}$$

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

$$\sum = \sum_{act} \cup \{tick\}$$

- δ est la fonction de transition. Un événement a peut être exécuté depuis un état q, i.e. $\delta(q, a)!$, si $\delta_{act}(q_{act}, a)!$, et la contrainte temporelle associée à l'occurrence de a est vérifiée.
- q_0 est l'état initial
- Q_m est l'ensemble des états marqués.

Un événement a est dit *autorisé* depuis un état q (i.e., $\delta(q_{act}, a)!$) 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 événement 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 4.2. Reprenons l'exemple représenté par le modèle logique (Figure 4.2). Son évolution est réalisée par l'occurrence des trois événements : a, b et c. Si la dimension temporelle est prise en compte, aux événements sont associés des intervalles temporels qui correspondent à la période pendant laquelle leur occurrence a lieu. Supposons que :

- à l'événement a est associé l'intervalle [1, 1],
- à l'événement b est associé l'intervalle [2, ∞), et
- à l'événement c est associé l'intervalle [1, 3].

Le comportement temporel du système est modélisé par un automate $\mathcal{A} = (Q, \sum, \delta, q_0, Q_m)$ dont les différents ensembles sont définis comme suit :

- $Q = \{q_1, q_2, q_3, q_4\} \times t_a \times t_b \times t_c = \{q_1, q_2, q_3, q_4\} \times [0, 1] \times [0, 2] \times [0, 3]$
- $\sum = \{a, b, c, tick\}$
- $q_0 = q_1$
- $Q_m = \{q_4\}$

Le modèle temporisé obtenu à partir du modèle logique par la prise en compte des contraintes temporelles associées à chaque événement, est représenté dans la Figure 4.3. La représentation de l'automate temporisé fait intervenir des transitions associées à l'événement *tick* et multiplie le nombre de transitions associées aux événements b et c. Les intervalles de temps associés à ces événements sont [2, ∞) pour b et [1, 3] pour c, ils décrivent l'incertitude sur ses dates d'occurrence.

Dans le modèle temporisé (Figure 4.3), à partir de l'état $q_{2'}$ les événements b et c sont autorisés mais seulement c est admissible. Donc, dans cet état l'événement b est mis en attente.

■

Par conséquent, seulement un événement admissible peut se produire.

De la même manière que pour les SED, certains événements sont contrôlables tandis que les autres ne le sont pas. Ainsi, l'ensemble des événements $\sum$ est partagé en trois sous-ensembles disjoints :

$$\sum = \sum_c \cup \sum_u \cup \{tick\}$$

où :

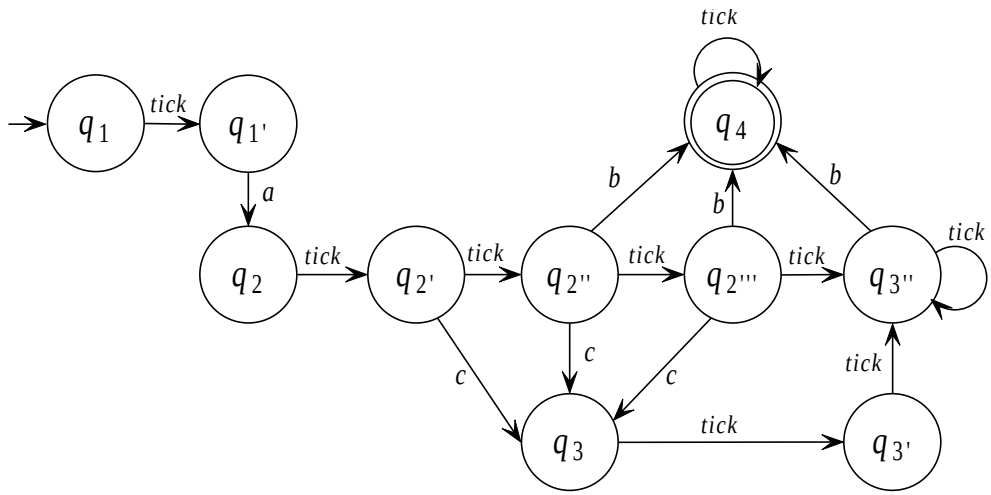


FIG. 4.3 – Modèle temporisé

- $\sum_c$ est l'ensemble des événements contrôlables ;
- $\sum_u$ est l'ensemble des événements incontrôlables.

Un événement contrôlable peut être interdit indéfiniment. Seulement les événements lointains peuvent être contrôlables, $\sum_c \subseteq \sum_{rem}$.

Par opposition, les événements proches ont une date d'occurrence au plus tard, au delà de laquelle ils ne peuvent plus être interdits. Ainsi, les événements proches sont incontrôlables $\sum_{spe} \subseteq \sum_u$. Certains événements lointains peuvent être incontrôlables par leur nature. Ainsi, l'ensemble des événements incontrôlables est défini par :

$$\sum_u = \sum_{spe} \cup (\sum_{rem} - \sum_c)$$

Une autre catégorie d'événements, essentielle dans la commande supervisée des SEDT, est représentée par les *événements forçables*. Un événement est considéré 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 événements forçables est noté $\sum_{for}$. Il n'y a aucune relation entre l'ensemble des événements forçables et les ensembles des événements contrôlables et incontrôlables. Un événement contrôlable peut ne pas être forçable dans le sens où on peut interdire son occurrence, mais on ne peut pas forcer son exécution. De même, certains événements incontrôlables peuvent être considérés forçables. Intuitivement, le passage de temps ne peut pas être forcé, donc $tick \notin \sum_{for}$.

Remarque 4.1. Dans notre travail de recherche, nous proposons une nouvelle classification des événements. Nous considérons qu'un événement peut être contrôlable ou incontrôlable. Un événement est contrôlable si on peut fixer sa date d'occurrence à l'intérieur d'un intervalle donné. Par contre, un événement est considéré incontrôlable si on ne peut pas agir sur sa date d'arrivée. Ainsi, nous considérons que la notion de forçage d'un événement est équivalente à la notion de contrôlabilité. Cette classification sera détaillée dans la section 4.3.

■

Jusqu'ici dans notre travail nous nous sommes intéressés à l'analyse de l'évolution d'un système hybride. Cependant, les résultats de l'analyse, tels qu'ils ont été présentés dans

le chapitre précédent, sont difficilement exploitables pour la synthèse d'une commande supervisée. La difficulté dans l'exploitation des résultats obtenus par l'analyse est générée par le fait que les trajectoires du système ne sont pas explicitement représentées. Par conséquent, nous allons proposer un modèle d'automate dérivé de l'automate temporisé, tel qu'il est construit par Brandin et Wonham, qui simule l'évolution d'un système hybride modélisé par un automate hybride à temps discret, représenté sous la forme d'un automate à états finis. De plus, le concept d'événements forçables sera lui aussi exploité lors de la synthèse du superviseur (Section 4.3).

4.2.2 *Modèle proposé : Automate déplié*

Afin d'appliquer la théorie de la commande supervisée aux systèmes hybrides, l'idée est d'abstraire un automate à états finis à partir de l'automate atteignable, explicitant ainsi toutes les trajectoires possibles du système pour une condition initiale donnée. Un tel modèle peut être obtenu par le dépliage temporel de l'évolution continue discrétisée du système dans chaque sommet de l'automate hybride. La technique est similaire à celle du dépliage temporel dans le cas des systèmes temporisés.

Le modèle déplié est obtenu à partir d'un automate hybride. Donc, on dispose d'un modèle qui prend en compte tant les aspects temporels que les aspects continus et discrets de l'évolution d'un système. Le modèle proposé est similaire à celui proposé par Brandin et Wonham sur de nombreux points. Cependant, dans ce modèle l'évolution continue des variables d'états du système doit être prise en compte.

Nous allons définir formellement le modèle automate déplié à partir des éléments d'un automate hybride à temps-discret (voir Definition 3.1) et des éléments d'un automate temporisé, tel qu'il a été défini par Brandin et Wonham.

Soit un automate hybride à temps-discret $\mathcal{A} = (Q_H, X, flux, inv, garde, Aff, init)$ et un automate temporisé $\mathcal{A}_T = (Q, \sum, \delta, q_0, Q_m)$.

Définition 4.1. Un *automate déplié* $\mathcal{A}_d$ est un automate à états finis temporisé $\mathcal{A}_d = (Q_d, \sum_d, \delta, q_0, Q_m)$ avec :

- Q_d l'ensemble fini des sommets de l'automate. Chaque sommet $q \in Q_d$ mémorise :
  l'état global du système hybride représenté par la paire (q_i, R_A), avec $q_i \in Q_H$ l'état logique du système et $R_A \subseteq X$ représentant la région continue atteinte par l'évolution du système ;
  - la temporisation t_a associée à chaque événement $a \in \sum$, intervenant dans l'évolution du système.
- $\sum_d$ un ensemble fini d'événements. Le passage du temps est modélisé par l'occurrence d'un événement *tick*. Cet événement modélise l'écoulement d'une période d'échantillonnage

$$\sum_d = \sum_{act} \cup \{tick\}$$

- δ une fonction de transition. Un événement a peut être exécuté depuis un sommet q_d si son occurrence est possible à la fois sur plan logique et sur le plan temporel.
- q_0 un état initial
- Q_n un ensemble fini d'états marqués.

■

Le modèle automate déplié ainsi défini représente un automate temporisé étendu avec les aspects de l'évolution continue des variables d'état d'un système hybride. Pour la construction de ce modèle, il faut connaître l'évolution complète du système. Ainsi, les résultats obtenus lors de l'analyse d'atteignabilité seront exploités.

4.2.3 *Construction du modèle déplié*

Dans notre travail, nous avons utilisé la représentation en temps-discret de la dynamique continue du système hybride. Ainsi, dans le modèle déplié l'écoulement du temps va être pris en compte en associant un événement *tick* à chaque pas discret. Donc, l'écoulement d'une période d'échantillonnage se manifestera au niveau du modèle déplié par l'occurrence d'un événement *tick*. Cela ne signifie pas qu'il s'est écoulé une période d'échantillonnage pendant l'occurrence de cet événement, mais cela correspond au passage d'un pas discret à l'autre. Pour prendre en compte les aspects continus de l'évolution des variables d'état, on associe à chaque sommet une région de l'espace d'état continu. Cette région représente la région successeur courante.

Le dépliage temporel peut se faire d'une manière systématique à partir de l'automate atteignable, en utilisant les résultats de l'analyse stockés dans les vecteurs *trace* associés à chaque sommet de l'automate hybride. La détermination de ces vecteurs est présentée dans le chapitre précédent, lors de la construction de l'automate atteignable (Section 3.4). En résumé, dans un vecteur *trace* associé à un sommet de l'automate hybride est stocké l'historique de l'évolution du système pour chaque visite. Ainsi, en utilisant ces informations, la trajectoire complète de l'état continu, durant le fonctionnement du système, peut être explicitement reconstruite.

L'idée de base du dépliage temporel d'un sommet de l'automate hybride repose sur la création d'un nouveau *sommet* dit *discret*, associé à chaque successeur continu obtenu lors d'une visite dans le sommet hybride considéré.

Pour illustrer ce propos, considérons un sommet q_j de l'automate hybride, illustré dans la Figure 4.4.

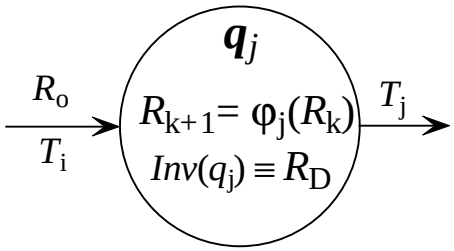


FIG. 4.4 – Sommet de l'automate hybride

Supposons que le calcul de l'espace atteignable dans ce sommet a été déjà réalisé pour une région initiale R_0, représentant la condition initiale à l'entrée du sommet lors de la $k$ième visite. L'évolution dynamique du système pour une telle initialisation est ainsi stockée dans le vecteur *trace*.

D'une manière systématique, le modèle déplié équivalent est obtenu par l'exécution des étapes décrites par l'algorithme suivant :

Algorithme 4.1. *Équivalent déplié d'un sommet de l'automate hybride*

Pas 1 : *Initialisation*
- Identifier le vecteur $trace_j(k)$ associé au sommet q_j pour la visite courante k
- L'ensemble des successeurs continus calculés pendant le séjour du système dans ce sommet est représenté par l'ensemble $C_j(k)$
- La date au plus tard à laquelle le système quitte le sommet (i.e.,l'invariant du sommet n'est plus vérifié) est $\delta_j(k)$
- Initialiser le compteur $c := 0$

Pas 2 : *Dépliage temporel*
TANT QUE la valeur du compteur $c \leq \delta_j(k)$ **FAIRE**

2.1. Le successeur continu courant est représenté par R_c, correspondant au $c$ième élément de l'ensemble $C_j(k)$

2.2. Créer un nouveau sommet discret S_c^j associé à la région R_c

2.3. **SI** $c < \delta_j(k)$ **ALORS**
Créer une transition de sortie étiquetée par l'événement *tick*

2.3. Actualiser la valeur du compteur : $c := c + 1$

■

L'algorithme présenté ci-dessus fait appel à des informations stockées dans le vecteur *trace* associé à la visite courante k dans le sommet q_j. Les transitions de sortie étiquetées par l'événement *tick* sont ajoutées jusqu'au dernier successeur continu. Le dépliage est arrêté lorsque la durée de séjour maximale $\delta_j(k)$ est atteinte.

Dans le chapitre précédent, nous avons vu que la condition d'arrêt pour le calcul des successeurs continus est donnée par l'invariant associé au sommet de l'automate hybride analysé.

Propriété 4.2.1. *La construction de l'équivalent déplié d'un sommet hybride s'arrête lorsque l'invariant du sommet n'est plus vérifié.*

■

L'algorithme précédent illustre les étapes du dépliage temporel de la dynamique continue dans un sommet de l'automate hybride et ne prend pas en compte les aspects engendrés par la validation des transitions de sortie du sommet. Afin d'obtenir l'équivalent déplié d'un automate hybride, il va falloir prendre en compte aussi les aspects discrets, qui déterminent des changements entre les différentes dynamiques continues du système par le franchissement des transitions de l'automate hybride considéré.

Rappelons que les changements de dynamique continue dans un système hybride sont déterminés par :

– l'occurrence des événements discrets, dont l'origine peut être interne ou externe (par exemple, l'intervention d'un opérateur humain ou la panne d'une machine), ou

– la validation d'une condition logique spécifiée sur les valeurs des variables d'état qui évoluent d'une manière continue avec l'écoulement du temps.

Par conséquent, les transitions de sortie d'un sommet hybride sont étiquetées soit par des conditions de franchissement spécifiées sur les variables continues (régions de franchissement), soit par des événements, soit par des prédicats logiques qui combinent les deux.

Dans le vecteur *trace*, l'information concernant les différentes dates de validation des transitions de sortie est aussi présente. Par conséquent, l'algorithme 4.1 doit être complété par des étapes qui permettent d'identifier l'évolution future du système. Trois situations peuvent être rencontrées et ces situations sont énumérées dans la suite :

Situation 1 : *Une transition de sortie est validée et l'invariant du sommet est vérifié* (illustrée par la Figure 4.5(a))

Dans une telle situation, à partir de l'instant auquel la transition est validée et tant qu'elle reste validée, dans le modèle déplié, à chaque nouveau sommet discret créé, il faudra ajouter une transition supplémentaire étiquetée par l'événement généré lors du franchissement de cette transition (Fig. 4.5(b)).

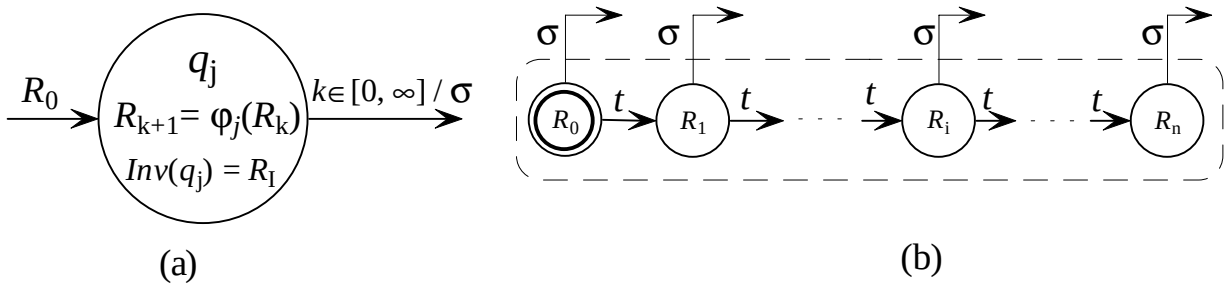


FIG. 4.5 – Illustration de la 1ère situation : (a) sommet de l'automate hybride, (b) modèle déplié équivalent

Le modèle déplié du sommet q_j est composée par $n+1$ sommets, par n transitions étiquetées par l'événement *tick* noté t reliant les sommets et par $n+1$ transitions de sortie associées au chaque sommet modélisant le non-déterminisme sur la date d'occurrence de l'événement σ. Une région successeur continue successeur de la région initiale R_0 est associée à chaque sommet. La région R_n représente le dernier successeur continu qui vérifie l'invariant associé au sommet q_j.

■

Situation 2 : *Une transition de sortie qui mène vers un sommet interdit est validée et l'invariant est toujours vérifié*

Dans cette situation, une transition de sortie étiquetée par un événement *tick* est créée si la transition est validée par l'écoulement du temps, ou bien par l'événement discret incontrôlable qui représente l'étiquette de la transition validée. Le sommet destination de ce nouvel arc restera le sommet interdit.

Reprenons la Figure 4.5(a). Considérons qu'il y a une deuxième transition de sortie du sommet q_j qui mène vers un sommet interdit et que cette transition est étiquetée par une région de franchissement R_F (Figure 4.6(a)), tel que $R_D \cap R_F \neq \phi$. Supposons que le dernier successeur continu avant que la transition vers le sommet interdit soit validée est R_i. Le modèle déplié du sommet q_j obtenu correspond au modèle illustré sur la Figure 4.6(b). Le successeur R_i précède le successeur R_n qui représente le dernier successeur pour lequel l'invariant du sommet q_j est vérifie. Cependant, la construction

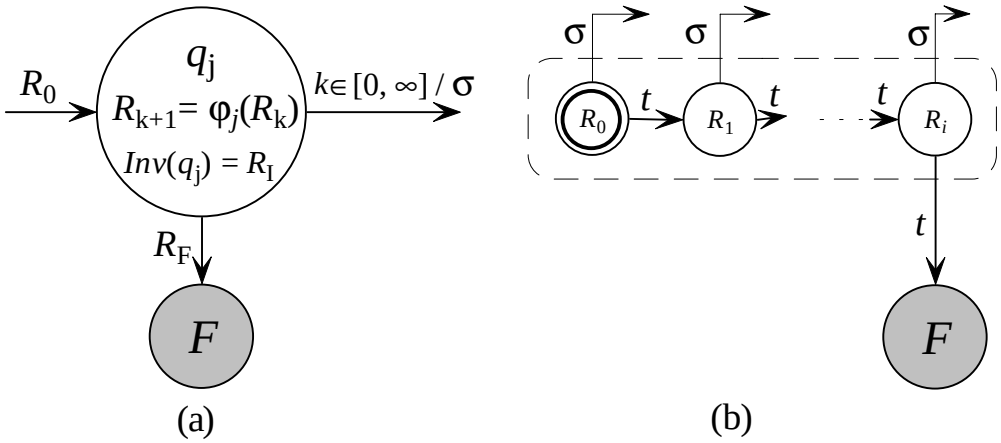


FIG. 4.6 – Illustration de la 2ème situation : (a) partie d'un automate hybride, (b) modèle déplié équivalent

du modèle déplié s'arrête lorsque la transition vers le sommet interdit F a été franchie.

Remarque 4.2. Une exception à la Propriété 4.2.1 est représentée par la Situation 2, où le sommet interdit de l'automate hybride est atteint alors que l'invariant du sommet en cours de dépliage est toujours vérifié. Étant donné que le sommet interdit est en général un sommet puits de l'automate hybride, il n'y a aucun intérêt à continuer le dépliage temporel, car l'évolution du système restera dans le domaine non-désirable. Par conséquent, la construction de l'équivalent déplié va s'arrêter lors de la validation d'une telle transition.

■

La technique de dépliage est illustrée dans l'exemple suivant.

Exemple 4.3. Considérons une partie d'un automate hybride modélisé dans la Figure 4.7(a). Le modèle considéré est composé par deux sommets dont un représente le sommet interdit, noté F et l'autre, noté q_j, représente le sommet hybride dont l'équivalent déplié sera construit. La région initiale à l'entrée de q_j est représentée par une région R_0 : $\{[1, 3]\times[1, 7]\}$. Les deux transitions de sortie sont T_j et T_f. La transition T_f est étiquetée par la condition de franchissement spécifiée sur la valeur de la variable d'état x_1, par $x_1\in[4, 5]$. La transition T_j est étiquetée par une condition spécifiée sur la valeur de k correspondant au pas discret, représenté par la condition $k \in [0, \infty)$. Par le franchissement

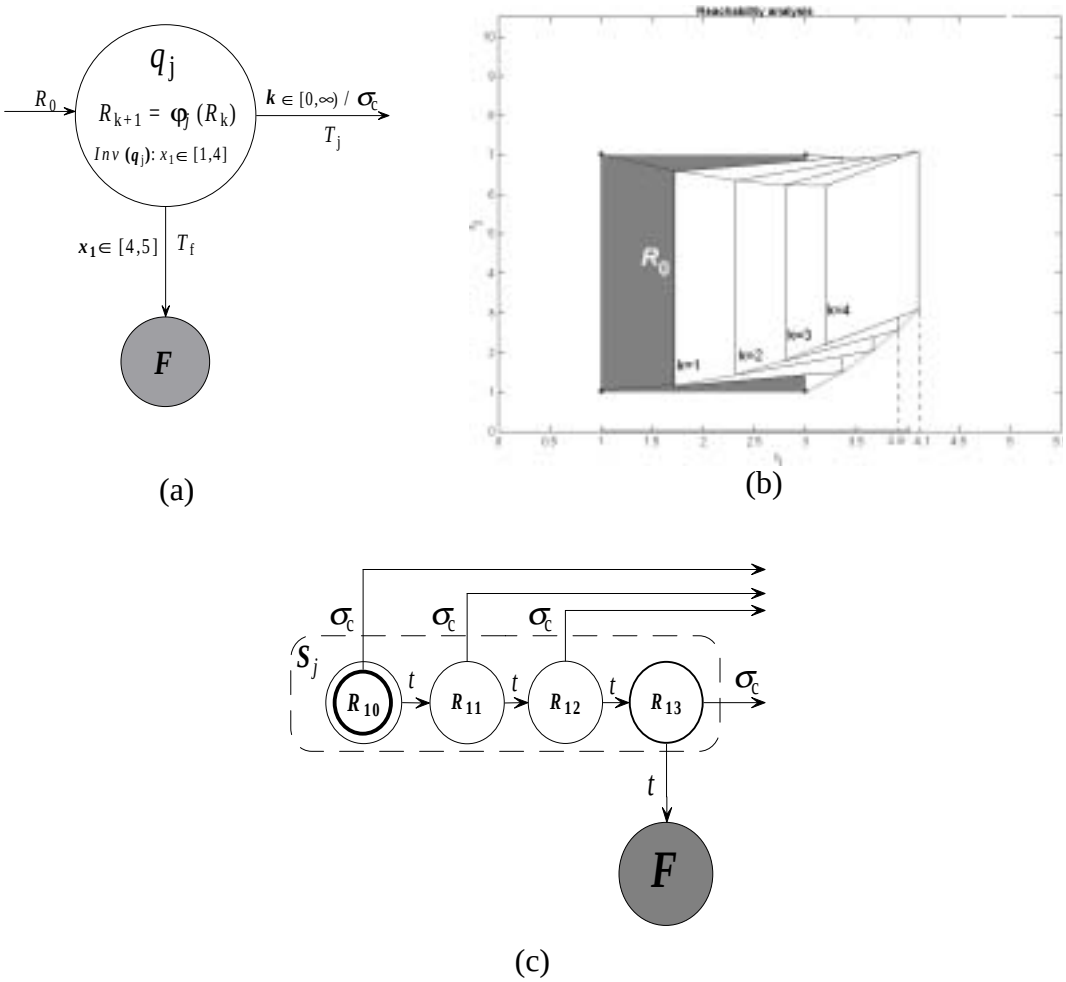


FIG. 4.7 – Illustration du dépliage temporel : (a) sommet de l'automate hybride, (b) successeurs continus et (c) équivalent déplié

de cette transition l'événement σ_c est généré.

La dynamique continue est associée au sommet q_j et les successeurs continus sont calculés en respectant la fonction $R_{k+1} = \varphi_j(R_k)$. Le système peut séjourner dans ce sommet tant que l'invariant $Inv(q_j) : x_1 \in [1, 4]$ est vérifié.

La Figure 4.7(b) illustre les résultats de l'analyse, plus particulièrement les successeurs continus jusqu'au moment où l'évolution vers l'état interdit du système devient possible et l'invariant du sommet n'est plus vérifié. Par l'analyse des résultats, on observe que la transition T_j est toujours validée. L'invariant du sommet est vérifié pour les 3 premiers successeurs calculés. Cependant, le 4-ème successeur vérifie la condition de franchissement de la transition T_f et l'évolution du système vers le sommet interdit devient possible.

L'équivalent déplié obtenu à partir de ces résultats est illustré dans la Figure 4.7(c). Le modèle est obtenu par la création d'un nouveau sommet discret associé à chaque région successeur continu, y compris la région initiale, tant que l'invariant du sommet est vérifié. Ainsi, le premier sommet discret créé, correspond à la valeur $c = 0$ du compteur et il est associé à la région initiale R_{10}. La transition de sortie étiquetée par l'événement t (notation utilisée pour l'événement *tick*) est aussi créée. Tant que la valeur de la variable compteur c vérifie la relation $c < \delta_j$, où $\delta_j = 4$ pour notre cas, on répète la démarche et on va créer les sommets discrets R_{11}, R_{12} et R_{13} avec des transitions correspondantes étiquetée par l'événement t. La transition T_j est validée depuis la valeur $k = 0$, le modèle doit être complété en ajoutant une transition de sortie supplémentaire étiquetée par l'événement σ_c à chaque sommet discret. Ceci permet de prendre en compte l'incertitude liée de la date d'occurrence de cet événement. L'évolution du système vers le sommet interdit est possible lorsque la garde de la transition T_f est validée. Dans notre exemple, la transition est validée, par l'écoulement du temps et l'évolution des variables d'état continues, par le 4-ème successeur continu. Les valeurs de la variable x_1 passe de $x_1(3) \in [2.804753\ 3.902377]$ à $x_1(4) \in [3.202684\ 4.101342]$. C'est à partir de ce moment que la transition T_f est validée. Donc, le sommet destination de la transition de sortie du sommet R_{13}, étiquetée par l'événement t, est le sommet interdit F.

■

Afin d'obtenir l'équivalent déplié d'un automate hybride, la technique de dépliage présentée ci-dessus doit être appliquée à chaque sommet de l'automate hybride lors de chaque visite. Le modèle résultant contiendra toutes les trajectoires possibles dans le système. Celles-ci sont ainsi explicitement représentées. Quelques remarques peuvent être formulées :

Remarque 4.3.

1. Le modèle déplié garde la complexité du modèle automate hybride initial, aucune approximation de son comportement n'est faite. Ce modèle représente l'évolution du système hybride.
2. L'inconvénient de ce modèle est l'explosion combinatoire du nombre de sommets pour des systèmes complexes engendré par l'utilisation du temps échantillonné.
3. L'avantage est qu'on obtient un automate à états finis pour lequel les techniques formelles pour la synthèse de la commande supervisée peuvent être appliquées. Donc, l'optimalité des résultats de synthèse peut être garantie.

■

4.3 Synthèse de superviseur

Le modèle déplié a été élaboré pour modéliser les trajectoires complètes d'un système hybride et pour procéder ensuite à la synthèse de superviseurs. Dans la suite, nous explicitons cette notion de synthèse.

4.3.1 *Concept de supervision*

Soit un procédé dynamique hybride, dont la représentation décrit l'ensemble de tous les comportements possibles et soit un critère (par exemple, les spécifications de fonctionnement) permettant de partitioner l'ensemble des comportements. Le sous-ensemble contenant les comportements qui ne vérifient pas le critère est considéré comme non acceptable. Le procédé peut être vu comme le système à superviser. Le superviseur est un autre système qui interagit avec le procédé. Il observe l'évolution du procédé à superviser et génère des actions qui influencent le comportement du procédé supervisé.

D'une manière générale, le superviseur restreint le comportement du procédé sous supervision afin d'obtenir un comportement qui ne soit jamais inclus dans le sous-ensemble des comportements non acceptables.

Dans l'approche classique de supervision le rôle du superviseur se limite à une autorisation/interdiction de l'occurrence de certains événements. Dans la réalité, un procédé n'a pas toujours une évolution spontanée. Le superviseur est, alors, chargé d'une fonction supplémentaire, il doit forcer l'occurrence de certains événements. Le rôle du superviseur est donc doublé d'un rôle de système de commande. Les entrées du superviseur, représentant également les sorties du procédé, sont des informations issues des capteurs. Les sorties du superviseur, qui sont aussi les entrées du procédé, représentent les commandes envoyées au procédé.

Généralement, les travaux sur la commande par supervision existant dans la littérature classifient les événements qui intervient dans le fonctionnement d'un système en trois catégories : incontrôlables, contrôlables et forçables. Un événement est dit *incontrôlable* si on ne peut pas interdire son occurrence. Par opposition, on dit qu'un événement est *contrôlable* si on peut toujours interdire son occurrence. Un événement est appelé *forçable* s'il peut se produire spontanément ou être forcé par un système extérieur (par exemple, le superviseur). Dans le sens de Brandin et Wonham, un événement forçable peut uniquement "préempter" le tick de l'horloge [BW94]. Autrement dit, le superviseur peut forcer l'événement à se produire avant l'occurrence du tick d'horloge.

Dans notre travail on se limite à seulement deux catégories des événements : contrôlables et incontrôlables.

Dans notre acception, un événement est *contrôlable* si on peut fixer sa date d'occurrence. Par exemple, le démarrage d'une tâche dans un système de production est, en général, un événement contrôlable.

Considérons l'exemple de la ligne de production présenté dans la sous-section 2.5.1. L'action d'arrêter le flux d'entrée du système est un événement contrôlable. L'instant d'occurrence de cet événement peut être fixé à n'importe quel moment dans l'intervalle $k \in [0, \infty)$.

Remarque 4.4. Un événement contrôlable est toujours forçable, i.e. $\sum_{for} \subseteq \sum_{c}$.

■

Pour forcer l'occurrence d'un événement, à un instant donné, la commande supervisée associe à l'événement une date unique d'occurrence qui correspond à l'instant considéré.

Un événement est appelé *incontrôlable* si on ne peut pas agir sur sa date d'exécution. Un exemple d'événement incontrôlable dans un système de production est en général l'accomplissement d'une tâche.

L'objectif de la synthèse de la commande est alors de déterminer les contraintes sur la date d'occurrence des événements contrôlables, telle que l'évolution du système respecte les spécifications de fonctionnement imposées. Dans la suite nous proposons les structures de contrôles qui seront utilisées pour la synthèse de la commande.

4.3.2 *Structures de contrôle*

La méthode de synthèse de la commande, développée dans le travail de recherche présenté dans ce mémoire, est basée sur le modèle automate déplié. La construction de ce modèle a été présentée en détail dans Section 4.2.2.

L'objectif de la commande revient alors, à contrôler toutes les trajectoires du système qui mènent vers les états interdits. Ainsi, la synthèse de la commande sera réalisée par l'identification de certaines structures de contrôle dans le modèle déplié, permettant de contrôler ainsi les évolutions futures du système (Figure 4.8).

Les notations utilisées dans les structures de contrôle sont :

- σ_c pour dénoter les événements contrôlables,
- σ_u pour dénoter les événements incontrôlables, et
- t pour l'événement tick.

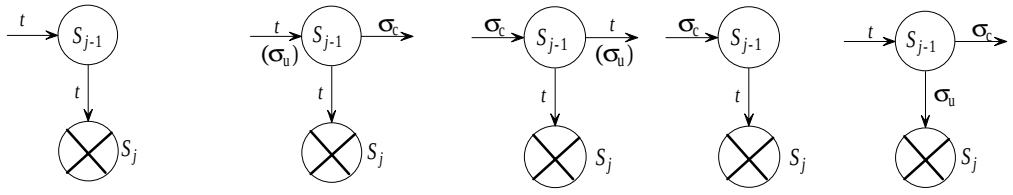
Chaque structure est composée par deux sommets : S_{j-1} et S_j. Le premier sommet est un sommet quelconque du graphe et l'autre modélise le sommet interdit.

Afin d'éviter l'évolution du système vers les sommets interdits du graphe, un superviseur externe peut agir sur l'occurrence des événements contrôlables. Ainsi, un événement contrôlable peut être soit *désactivé*, dans le sens d'interdire indéfiniment son occurrence, soit *activé*, son occurrence est forcée à un certain moment.

Les structures de contrôle présentées dans la Figure 4.8 sont utilisées lors de la synthèse. Elles correspondent aux différentes cas de figures que l'on peut rencontrer. Le principe est de remonter les branches de l'automate déplié à partir du sommet interdit et d'identifier une de ces six structures de contrôle. En fonction de la structure identifiée dans le modèle déplié, une décision sera prise en en ce qui concerne l'occurrence de l'événement contrôlable.

Supposons que la **Structure 1** soit identifiée dans notre modèle (Figure 4.8(a)). Dans cette situation, l'évolution vers le sommet interdit a lieu par l'écoulement du temps (occurrence des événements *tick* d'horloge). Comme l'écoulement du temps ne peut pas être empêché, le sommet S_{j-1} deviendra lui aussi interdit.

La **Structure 2** illustrée dans la Figure 4.8(b), correspond à une structure appelée de *contrôle en aval*. Si cette structure est identifiée, alors nous pouvons constater que



(a) *Structures* 1 (b) *Structure* 2 (c) *Structures* 3 (d) *Structure* 4

FIG. 4.8 – Structures de contrôle

le sommet S_{j-1} est atteint soit par l'occurrence d'un événement incontrôlable σ_u, soit par l'écoulement du temps (l'occurrence du tick d'horloge). À partir du sommet S_{j-1}, le sommet interdit est atteint par l'occurrence de l'événement t. La transition de sortie du sommet S_{j-1} est étiquetée par un événement contrôlable. Par conséquent, l'évolution vers le sommet interdit S_j peut être évitée en forçant l'occurrence de l'événement contrôlable σ_c.

La **Structure 3** illustrée dans la Figure 4.8(c) est une structure de *contrôle en amont*. L'évolution vers le sommet interdit S_j est déterminée par l'occurrence de l'événement contrôlable σ_c suivi de l'occurrence du tick d'horloge. La transition de sortie du S_{j-1} est étiquetée soit par l'événement incontrôlable σ_u soit par t. Ainsi, l'évolution vers le sommet interdit S_j peut être empêchée seulement en interdisant l'occurrence de l'événement σ_c.

L'occurrence des événements discrets étant considérée comme instantanée, dans la **Structure 4** une fois le sommet S_{j-1}, atteint l'évolution du système vers le sommet interdit S_j ne peut plus être évitée en forçant l'occurrence de l'événement contrôlable σ_c avant l'occurrence de l'événement σ_u. Ainsi, le sommet S_{j-1} devient interdit.

Les structures de contrôle proposées revient à considérer l'événement t, modélisant un top d'horloge, comme :

1 - événement contrôlable et préemptible, et

2 - événement incontrôlable

La problématique de d'existence d'un superviseur sera abordée dans la section suivante. En se basant sur la contrôlabilité de l'événement t, dans le sens qu'il y a des événements contrôlables qui peuvent préempter son occurrence, sera utilisée dans la démonstration de l'existence d'un superviseur.

4.3.3 *Existence du superviseur*

Le modèle déplié synthétisé à partir du modèle d'automate atteignable est un modèle temporisé. Dans la suite de la démarche de synthèse, on s'appuie seulement sur les aspects temporels modélisées (date d'occurrence d'événements) par ce modèle.

Dans un premier temps, il est nécessaire de définir la manière dont les transitions du modèle temporisé peuvent être contrôlées par le superviseur. La méthode décrite ci-dessous est celle proposée par Brandin et Wonham [BW94]. La recherche de superviseur s'effectue en respectant deux critères :

- la supervision ne peut que restreindre le comportement d'un système non supervisé (i.e., le langage du procédé sous supervision est inclus dans le langage du procédé non supervisé) ;
- le comportement du procédé sous supervision doit pouvoir être optimisé afin d'obtenir le maximum de permissivité.

Le superviseur peut intervenir sur l'évolution du procédé par l'intermédiaire des événements contrôlables. De la même façon que dans la supervision SED, les événements incontrôlables sont toujours autorisés par le superviseur. Par contre, lorsque parmi les événements éligibles il y a au moins un événement forçable, le superviseur peut forcer son occurrence avant l'occurrence d'un événement *tick*. L'ensemble des événements forçables est dénoté $\sum_{for}$, les éléments de cet ensemble ont la possibilité de préempter l'occurrence de l'événement *tick*. L'évolution du temps n'est pas interrompue puisque l'occurrence d'un événement est instantanée et la notion de préemption traduit celle de précédence et correspond donc uniquement au fait que l'événement forçable se produit avant l'apparition de l'événement *tick*.

Soit P un procédé à superviser. On modélise son comportement temporisé avec un automate $\mathcal{P} = (Q, \sum, \delta, q_0, Q_m)$ qui génère un langage $L(\mathcal{P})$. Considérons un mot $w \in L(\mathcal{P})$. Alors il existe un état $q \in Q$ atteint par l'exécution du mot w depuis l'état initial q_0.

Les possibilités d'évolution du procédé depuis un état q de l'automate sont décrites par l'ensemble d'événements éligibles dans l'état q. A chaque mot $w \in L(\mathcal{P})$ on associe un ensemble d'événements éligibles, $Elig_{\mathcal{P}}(w)$, défini par :

$$Elig_{\mathcal{P}}(w) = \{a \in \sum | wa \in L(\mathcal{P})\}$$

Dans cet ensemble le superviseur peut ponctuellement supprimer autant d'événements contrôlables que cela est nécessaire pour garantir la contrôlabilité.

L'événement *tick* doit faire l'objet d'un traitement particulier. Si, à un instant donné, l'ensemble des événements éligibles contient l'événement *tick* et au moins un événement de $\sum_{for}$, comme illustre la Structure 2 de la Figure 4.8(b), alors le superviseur peut ponctuellement supprimer l'événement *tick* de l'ensemble considéré. Cela signifie que l'occurrence d'un événement est garantie avant l'apparition de *tick*. Le statut de l'événement *tick* est donc ambigu : il peut être considéré comme un événement contrôlable s'il est en concurrence avec au moins un événement forçable et il doit être vu comme un événement incontrôlable dans le cas contraire.

Les événements qui peuvent être ponctuellement supprimées de l'ensemble des événements éligibles appartiennent à l'ensemble $\sum_{hib}$ défini comme suit :

$$\sum_c := \sum_{hib} \cup \{tick\},$$

où $\sum_{hib} \subseteq \sum_{for}$.

Formellement, un superviseur S est défini comme une application

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

telle que pour tout mot $w \in L((P))$:

$$S(w) \cap Elig_{\mathcal{P}}(w) \neq \phi$$

$$S(w) \supseteq \begin{cases} \sum_u \cup \{tick\} & \text{si } S(w) \cap \sum_{for} = \phi \\ \sum_u & \text{si } S(w) \cap \sum_{for} \neq \phi \end{cases}$$

De la même façon que dans la théorie de la supervision de SED, les événements incontrôlables sont toujours autorisés par le superviseur. Par contre, lorsque parmi les événements éligibles il y a au moins un événement forçable, le superviseur peut forcer son exécution avant l'occurrence d'un nouveau top d'horloge (événement *tick*). Dans cette approche, le superviseur joue aussi un rôle d'un système de commande.

Le comportement en boucle fermée et le concept de contrôlabilité d'un langage sont traités de la même façon que dans la théorie de Ramadge et Wonham.

Soit un procédé P de langage $L(\mathcal{P})$ et de langage marqué $L_m(\mathcal{P})$. Le langage du procédé sous supervision $L(S/P)$, qui décrit le fonctionnement d'un procédé P supervisé par un superviseur S, est défini par :

- $e \in L(S/P)$,
- si $w \in L(S/P)$, $a \in S(w)$ et $wa \in L(\mathcal{P})$ alors $wa \in L(S/P)$

Le comportement marquée du procédé sous supervision est déterminé par

$$L_m(S/P) = L(S/P) \cap L_m(P),$$

telle que $0 \subseteq L_m(S/P) \subseteq L(P)$. Cela correspond à une restriction par rapport au langage du procédé en boucle ouverte.

Par définition, un superviseur S est dit *non bloquant* si tous les mots de son langage sont des préfixes des mots marqués :

$$\overline{L_m}(S/P) = L(S/P)$$

Remarque 4.5. Dans notre travail, nous considérons que toutes les évolutions du système sont marquées, donc le langage généré par le procédé est identique au langage marqué. Dans ce contexte le problème de non blocage ne se pose plus.

■

Les spécifications que le procédé doit respecter sont mises sous la forme d'un langage K inclus dans le langage de l'automate modélisant le comportement du procédé en boucle ouverte (i.e., $K \subseteq L(\mathcal{P})$). Ces spécifications permettent de définir après chaque mot $w \in \sum^*$ un ensemble d'événements éligibles

$$Elig_K(w) := \{a \in \sum | wa \in \overline{K}\}$$

La contrôlabilité est une propriété fondamentale dans la supervision. C'est elle qui permet d'assurer que le procédé respectera constamment les spécifications. Elle est définie par rapport à un autre langage. Le langage K est contrôlable par rapport à $L(\mathcal{P})$ si pour tout mot $w \in \overline{K}$

$$Elig_K(w) \supseteq \begin{cases} Elig_{\mathcal{P})}(w) \cap (\sum_u \cup \{tick\}) & \text{si } Elig_K(w) \cap \sum_{for} = \phi \\ Elig_{\mathcal{P}}(w) \cap \sum_u & \text{si } Elig_K(w) \cap \sum_{for} \neq \phi \end{cases}$$

La contrôlabilité de K par rapport à $L(\mathcal{P})$ implique que quel que soit l'état considéré, K admet tous les événements incontrôlables qui sont admis par le procédé. De plus, l'événement *tick* est au moins admis par K lorsque l'ensemble d'événements éligibles par K ne contient pas d'événements forçables.

Si l'objectif de la supervision est de faire respecter au procédé considéré un ensemble de contraintes, il faut s'assurer que toute exécution commencée par le procédé permet d'atteindre un état marqué (un état de l'automate dans notre cas). Donc, chaque mot inclus dans le langage des spécifications K est contenu dans le langage marqué $L_m(\mathcal{P})$: $K = \overline{K} \cap L_m(\mathcal{P})$. Cette propriété est connue sous le nom de L_m-clôture et a été discuté dans le chapitre 1 (Section 1.3.2).

Dans notre cas, $L_m(\mathcal{P}) = L(\mathcal{P})$. Donc il faudra s'assurer que chaque mot inclus dans le langage des spécifications K est contenu dans le langage du procédé $L(\mathcal{P})$: $K = \overline{K} \cap L(\mathcal{P})$.

En se basant sur toutes ces notions, le théorème classique relatif à l'existence d'un superviseur peut être étendu au cas des automates temporisés [BW94].

Théorème 4.3.1. *Soit $K \subseteq L(\mathcal{P})$, avec $K \neq \phi$. Il existe un superviseur S pour le procédé P tel que $L(S/P) = K$ si et seulement si*

- *K est contrôlable par rapport à $L(\mathcal{P})$*
- *et K est $L(\mathcal{P})$-clôt (i.e., $\overline{K} \cap L(\mathcal{P}) = K$).*

Preuve :

"$\Longleftarrow$" Soit $w \in \overline{K}$. Le superviseur $S(w)$ est défini par

$$S(w) := \begin{cases} \sum_u \cup (\sum_c \cap Elig_K(w)) \cup \{tick\} & \text{si } Elig_K(w) \cap \sum_{for} = \phi \\ \sum_u \cup (\sum_c \cap Elig_K(w)) & \text{si } Elig_K(w) \cap \sum_{for} \neq \phi \end{cases}$$

Pour $w \in L(\mathcal{P}) - \overline{K}$ le superviseur est défini comme

$$S(w) := \sum.$$

On observe que $Elig_K(w) \cap \sum_{for} \neq \phi$ implique $S(w) \cap \sum_{for} \neq \phi$, donc S est un superviseur.

D'abord, on montre que $L(S/P) \subseteq \overline{K}$.

Puisque $K \neq \phi$ est contrôlable par rapport à $L(\mathcal{P})$, alors $e \in \overline{K}$.

Supposons que $wa \in L(S/P)$ (i.e., $w \in L(S/P)$ et $a \in S(w) \cap Elig_P(w)$). Si $w \in \overline{K}$, alors

$a \in \sum_u$. En se basant sur la contrôlabilité du K, cela implique que $a \in Elig_P(w) \cap \sum_u$, donc $a \in Elig_K(w)$.
Si on suppose que $a \in \sum_c$, alors

$$a \in \sum\nolimits_c \cap S(w) = \begin{cases} (\sum_c \cap Elig_K(w)) \cup tick & \text{si } Elig_K(w) \cap \sum_{for} = \phi \\ \sum_c \cap Elig_K(w) & \text{si } Elig_K(w) \cap \sum_{for} \neq \phi \end{cases}$$

par définition de $S(w)$. Dans la première situation, par la contrôlabilité de K,

$$tick \cap Elig_P(w) \subseteq Elig_K(w).$$

Dans la deuxième situation, $a \in Elig_K(w)$. Donc, dans les deux cas $a \in Elig_K(w)$, cela implique $wa \in \overline{K}$.

Nous allons démontrer l'inclusion inverse, i.e. $L(S/P) \supseteq \overline{K}$.
Par la définition nous avons $e \in L(S/P)$.
Supposons que $wa \in \overline{K}$. Cela signifie que $a \in Elig_K(w)$, donc $a \in Elig_P(w)$, cela implique $wa \in L(P)$. Supposons que $w \in L(S/P)$ et que $a \in \sum_u$. L'implication directe est que $a \in S(w)$, donc $wa \in L(S/P)$. Si $a \in \sum_c$ et $a \in Elig_K(w)$, alors $a \in S(w)$, par conséquent $wa \in L(S/P)$ et l'inclusion inverse a été démontrée.

Nous allons démontrer que $L(S/P) = K$.
Par la définition nous avons

$$L_m(S/P) = L(S/P) \cap L(\mathcal{P}).$$

Nous avons prouvé que $L(S/P) = \overline{K}$, donc

$$L_m(S/P) = \overline{K} \cap L(\mathcal{P}).$$

Comme K est $L(\mathcal{P})$-clôt, implique $L_m(S/P) = K$. Comme toutes les évolutions de notre système son marquées, cela implique que $L(S/P) = K$.

"$\Longrightarrow$" Soit un superviseur S avec $L(S/P) = K$.

D'abord on montre que K est $L(\mathcal{P})$-clôt.
En se basant sur le fait que toutes les évolutions du système sont marquées, la propriété de non blocage pour le superviseur S est garantie. Alors $L(S/P) = \overline{K}$, donc

$$K = L_m(S/P) = L(S/P) \cap L(\mathcal{P}) = \overline{K} \cap L(\mathcal{P}),$$

i.e., K est $L(\mathcal{P})$-clôt.

Nous allons démontrer que K est contrôlable.
Soit $w \in \overline{K}$. Cela implique $w \in L(S/P)$ et d'ici $Elig_P(w) \neq \phi$.
Puisque S est un superviseur, $S(w) \cap Elig_P(w) \neq \phi$,

$$\{a | wa \in L(S/P)\} \neq \phi,\ L(S/P) = K$$

tel que $Elig_K(w) \neq \phi$.
Soit $a \in Elig_P(w)$. Alors, $wa \in \overline{K}$ ssi $wa \in L(S/P)$,

$$a \in Elig_K(w) \text{ ssi } a \in S(w).$$

Si $a \sum_u$ alors $a \in S(w)$ et en conséquence $a \in Elig_K(w)$. Si $Elig_K(w) \cap \sum_{for} = \phi$ et $a = tick \in Elig_P(w)$ alors $a \in S(w)$ et d'ici $a \in Elig_K(w)$. Dans les deux cas, $a \in Elig_K(w)$.

■

Le concept de superviseur peut être étendu pour prendre en compte la notion de marquage. La structure générale de commande ne change pas, mais pour $K \subseteq L_m(\mathcal{P})$ le superviseur est défini pour le couple (K, P) et il sera tel que $L_m(S/P) = L_m(S/P) \cap K$.

Si le langage K considéré n'est pas contrôlable par rapport à $L(\mathcal{P})$, il est possible de définir l'ensemble de sous-langages de K qui seront contrôlables par rapport à $L(\mathcal{P})$. Cet ensemble est défini par :

$$C(K) = \{K' \subseteq K | K' \text{ est contrôlable par rapport à } L(\mathcal{P})\}$$

Cette famille est close sous l'union des langages et par conséquent, il existe un plus grand langage contrôlable unique noté $supC(K)$. Quel que soit le langage des spécifications donné, il est possible de déterminer un langage contrôlable pour définir le superviseur. Le calcul du langage suprêmal contrôlable est traité de la même façon que dans la théorie de Ramadge et Wonham. Ce langage correspond au superviseur le plus permissif. Pour démontrer ce propos on s'appuie sur la proposition suivante [RW87] :

Proposition 4.3.2. *Soit $K \subseteq \sum^*$ un langage $L_m(\mathcal{P})$-marqué*[1]*, alors $supC(K \cap L_m(\mathcal{P}))$ est $L_m(\mathcal{P})$-clôt.*

■

Le théorème précédent relatif à l'existence du superviseur nécessitait que le langage des spécifications soit contrôlable par rapport au langage du procédé. En utilisant le plus grand langage contrôlable inclus dans K il est possible d'obtenir un nouveau théorème :

Théorème 4.3.3. *Soit $K \subseteq \sum^*$ un langage $L_m(\mathcal{P})$-marqué et soit $K' = supC(K \cap L_m(\mathcal{P}))$. Si $K' \neq \phi$, il existe un superviseur S pour P tel que $L_m(S/P) = K'$.*

Preuve : K est contrôlable et par la Proposition 4.3.2, il est $L_m(\mathcal{P})$-clôt. Le résultat de ce théorème est une conséquence du Théorème 4.3.1.

■

Quelles que soient les spécifications données, il est possible de définir un superviseur qui ne permettra peut être pas au procédé d'adopter exactement le comportement décrit par les spécifications mais ce comportement sera nécessairement inclus dans celui proposé par les spécifications.

Le langage généré par le procédé couplé à son superviseur correspond au langage suprêmal contrôlable. Un superviseur est dit le plus permissif lorsqu'il permet d'obtenir le plus grand langage contrôlable inclus dans les spécifications, si ces dernières représentent exactement le comportement souhaité.

La démonstration des théorèmes précédents décrit la manière d'agir sur l'occurrence des événements intervenant dans l'évolution du procédé afin de synthétiser le superviseur le plus permissif. L'algorithme de *Kumar* implémente la démonstration . Dans [KGM91] les auteurs proposent un algorithme pour la construction du superviseur le plus permissif, lorsque les spécifications sont incluses dans le langage de l'automate du procédé, et que ces spécifications correspondent au langage d'un automate. A partir du langage K modélisant les spécifications, l'algorithme permet de vérifier si le langage K est contrôlable et s'il ne l'est pas, de déterminer le plus grand contrôlable inclus dans K. Le langage ainsi déterminé est celui du superviseur. Cet algorithme est celui qui est le plus fréquemment utilisé.

Pour conclure cette section, nous allons présenter l'algorithme qui synthétise les différentes étapes suivies lors de la synthèse du modèle de commande et nous allons illustrer son utilisation sur un exemple pédagogique.

4.3.4 *Algorithme de synthèse du superviseur*

L'algorithme proposé permet d'éliminer toutes les évolutions non désirées du système, représentées dans l'automate déplié.

Les étapes de la synthèse de superviseur peuvent être systématisées à travers l'algorithme suivant, faisant appel aux structures de contrôle présentées dans la Section 4.3.2 :

[1] Un langage K est dit $L_m(\mathcal{P})$-marqué si tout préfixe du K appartenant à $L(\mathcal{P})$ appartient aussi à K (i.e., $\overline{K} \cap L(\mathcal{P})$)

Algorithme 4.2. *Synthèse de superviseur*

Étant donné le modèle automate déplié

1. Identifier dans ce modèle les sommets interdits
2. *Synthèse effective*

   À partir du sommet interdit courant, **pour** chaque sommet interdit **faire**

   2.1. Remonter les branches dans le modèle déplié jusqu'au moment où une des structure de contrôle est identifiée ou jusqu'à ce que tous les sommets aient été explorés.

   2.2. **Si** la Structure 1 est identifiée **alors**
   Le sommet S_{j-1} devient interdit ;
   Supprimer ses transitions de sortie.

   2.3. **Si** la Structure 2 est identifiée **alors**
   Forcer l'occurrence de l'événement contrôlable σ_c ;
   Supprimer la transition de sortie qui mène au sommet S_j.

   2.4. **Si** la Structure 3 est identifiée **alors**
   Interdire l'occurrence de l'événement contrôlable σ_c ;
   Supprimer la transition de sortie étiquetée par σ_c.

   2.5. **Si** la Structure 4 est identifiée **alors**
   Le sommet S_{j-1} devient interdit ;
   Supprimer ses transitions de sortie.

   2.6. Retourner au Pas 2.1.

3. **Si** tous les sommets ont été explorés **alors**
   Actualiser le modèle automate déplié en supprimant toutes les transitions et sommets isolés du graphe.

■

La synthèse de superviseur est illustrée dans la suite sur un exemple pédagogique.

Exemple 4.4. Considérons l'automate déplié illustré dans la Figure 4.9. Pour éliminer toutes les trajectoires qui mènent vers le sommet interdit, noté F sur la figure, les étapes de l'algorithme précédent seront appliquées.

Le modèle contient plusieurs trajectoires qui mènent vers le sommet interdit. Le modèle choisi contient un sommet interdit, donc la synthèse de superviseur va démarrer à partir de ce sommet. À partir du sommet F on remonte les branches afin d'identifier les structures de contrôle présentées dans la Figure 4.8. Le sommet interdit peut être atteint par trois séquences d'événements à partir du sommet initial R_{10} appartenant au groupe S_1^1.

La première séquence est représentée par la succession $\sigma_c.t$. À partir du sommet interdit F, en remontant la branche correspondant, on observe que la Structure 3 est identifiée. L'évolution vers le sommet interdit peut alors être empêchée en interdisant l'occurrence de l'événement contrôlable σ_c. Donc l'arc étiqueté par cet événement doit être supprimé.

À partir du même sommet initial, la deuxième possibilité d'atteindre le sommet F est obtenue par la séquence d'événements $t.\sigma_c.2t$. En remontant la branche correspondant à partir du sommet interdit, on identifie dans un premier temps la Structure 1. Donc, le sommet R_{21} appartenant au groupe S_2^2 devient interdit et l'arc le reliant au sommet F doit être supprimé. À partir de ce nouveau sommet interdit, en remontant la branche du graphe on identifie la Structure 3. Par conséquent, l'évolution non désirée peut être évitée par l'interdiction de l'occurrence de l'événement contrôlable σ_c. L'arc étiqueté par

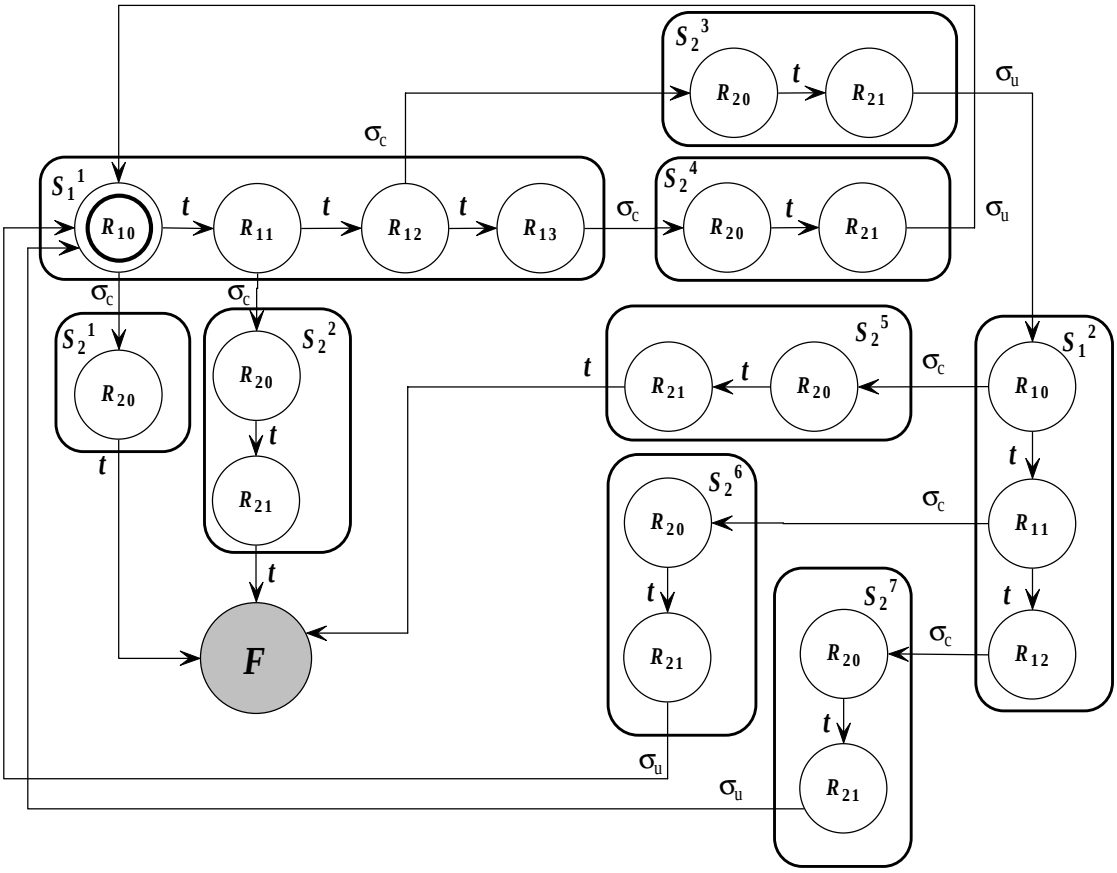


FIG. 4.9 – Automate déplié

cet événement doit être supprimé.

La troisième et dernière trajectoire qui conduit le système vers le sommet interdit F est représentée par la séquence $2t.\sigma_c.t.\sigma_u.\sigma_c.2t$. Afin d'empêcher cette évolution on va procéder comme dans le cas précédent. D'abord la Structure 1 peut être identifiée. Donc, le sommet R_{21} appartenant au groupe S_2^5 devient interdit. L'arc reliant ce sommet avec le sommet interdit doit être supprimé. À partir de ce nouveau sommet interdit, en remontant la branche du graphe la Structure 3 est identifiée. Pour éviter l'évolution du système vers ce sommet, l'occurrence de l'événement contrôlable σ_c doit être interdit. La transition étiquetée par cet événement doit être supprimée.

Pour achever la synthèse du superviseur il faut actualiser le graphe. Donc, on supprime tous les sommets du modèle qui ne sont plus reliés. C'est le cas des sommets : R_{20} appartenant au groupe S_2^1 ; R_{20} et R_{21}, ainsi que l'arc qui les relie, du groupe S_2^2, et R_{20} et R_{21}, ainsi que l'arc qui les relie, du groupe S_2^5. Le modèle ainsi obtenu est illustré dans la Figure 4.10.

■

Nous pouvons remarquer que par les structures de contrôles proposées nous ne faisons qu'appliquer l'approche de Brandin et Wonham pour la synthèse de superviseur. Par conséquent, nous avons éliminé les évolutions non-désirées soit par le forçage de l'occurrence des événements contrôlables, soit par l'interdiction de leur occurrence. Donc, on peut conclure que :

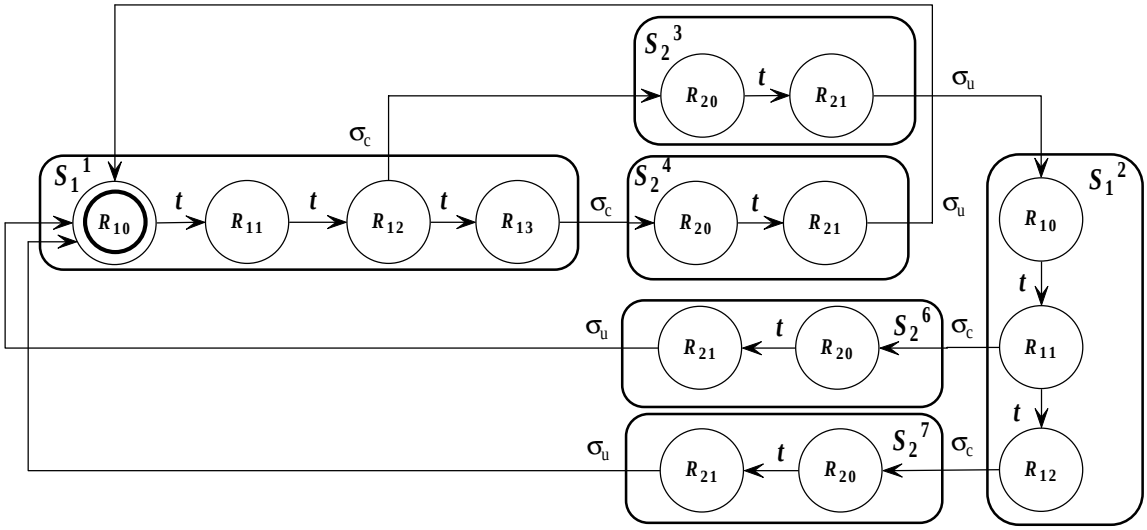


FIG. 4.10 – Résultats de la synthèse

Propriété 4.3.4. *Le superviseur ainsi obtenu est le plus permissif possible. Alors, les résultats de la synthèse sont optimaux.*

■

Remarque 4.6. L'optimalité est due à la représentation en temps-discret de la dynamique continue.

■

Le modèle du superviseur résultant est encore assez complexe en nombre des sommets. Dans la suite nous proposons une procédure qui permet de simplifier ce modèle et de lui donner une représentation plus compacte.

4.4 Modèle de commande

Dans cette section nous allons proposer une démarche permettant de réduire la taille du modèle de commande obtenu lors de la synthèse.

Le dépliage du modèle d'automate hybride a fait apparaître des sommets intermédiaires modélisant des états du système qui ont été atteints par l'écoulement du temps. Parmi tous ces états il y en a plusieurs qui sont inutiles, dans le sens où ils ne contiennent pas d'informations importantes sur l'évolution du système en vue de l'implementation du modèle de commande. L'idée est de regrouper tous ces sommets du modèle déplié. Pour ce faire, nous proposons dans la suite une procédure de re-pliage qui sera appliquée au modèle de commande déplié obtenu lors de la synthèse.

Le re-pliage du modèle va se faire dans deux étapes. D'abord, une procédure de *re-pliage temporel* du modèle déplié sera exécutée. Une fois que le modèle re-plié temporellement a été obtenu, la *fusion* entre les sommets du graphe qui sont similaires sera réalisée. On dira dans la suite ce que l'on entend par sommets similaires.

La procédure de re-pliage temporel est une procédure dual à celle du dépliage temporel. Afin de réaliser le re-pliage temporel il faut identifier les groupes de sommets discrets obtenus lors du dépliage à partir d'un même sommet hybride.

Propriété 4.4.1. *Deux sommets discrets du modèle déplié peuvent être re-pliés si et seulement si ils ont été obtenus par le dépliage temporel du même sommet de l'automate hybride.*

■

Autrement dit, deux sommets discrets de l'automate déplié peuvent être re-pliés si et seulement si les successeurs continus associés à ces sommets ont été générés par la même dynamique continue.

Afin d'illustrer ce propos considérons l'exemple suivant.

Exemple 4.5. Prenons par exemple le groupe de sommets discrets représenté dans la Figure 4.11(a). Supposons que ce groupe ait été obtenu par le dépliage temporel d'un

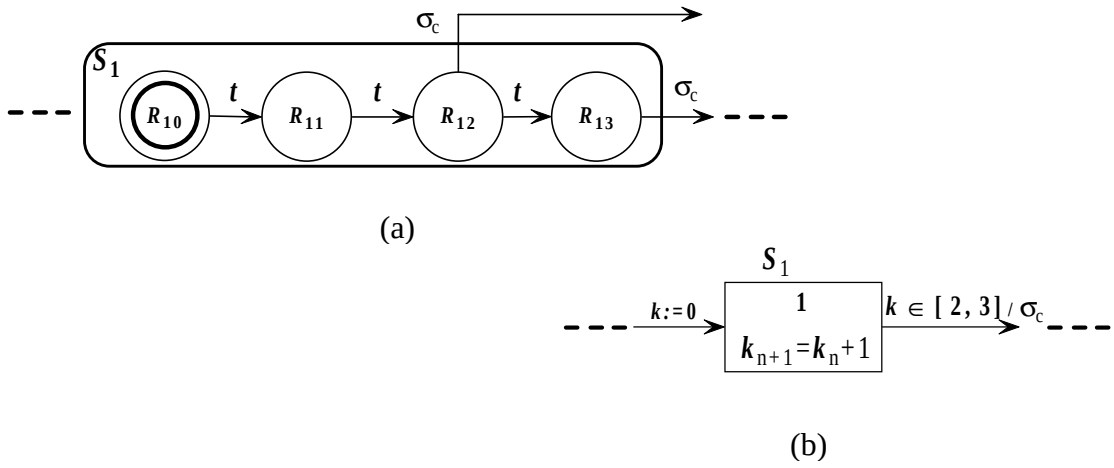


FIG. 4.11 – Re-pliage temporel : (a) groupe des sommets discrets et (b) équivalent re-plié

sommet d'un automate hybride. Donc les successeurs continus ont été générés par une même dynamique.

L'équivalent plié de ce modèle est représenté dans la Figure 4.11(b). On observe la dualité par rapport à la technique de dépliage temporel présentée en détail dans la section 4.2.2. Les quatres sommets discrets sont regroupés dans un seul auquel on associe une horloge qui compte le nombre de périodes d'échantillonnage écoulées. Ce nouveau sommet aura une transition d'entrée qui met à zéro l'horloge et une transition de sortie qui modélise les contraintes temporelles sur l'occurrence de l'événement σ_c.

■

À partir du modèle synthétisé par l'Algorithme 4.2, en procédant ainsi un modèle de commande simplifié en terme de nombre de sommets peut être établi. Le modèle de commande obtenu par le re-pliage temporel du modèle superviseur illustré dans la Figure 4.10, est représenté ci-dessous (Figure 4.12).

Ce modèle a été obtenu par le re-pliage des groupes de sommets discrets du modèle initial, dénotés par S_1^1, S_1^2, S_2^3, S_2^4, S_2^6 et S_2^7. Les contraintes temporelles spécifiées sur la date d'occurrence des événements σ_c et σ_u sont prises en compte par les étiquettes des transitions de sortie des sommets re-pliés. À chaque sommet du modèle re-plié on associe des horloges qui sont remises à zéro lors du franchissent d'une transition. Le modèle ainsi obtenu est déjà moins complexe en terme de nombre des sommets qui le composent.

Dans le but d'obtenir une représentation plus compacte du modèle de commande, la *fusion* entre les sommets du modèle re-plié temporellement peut être réalisée.

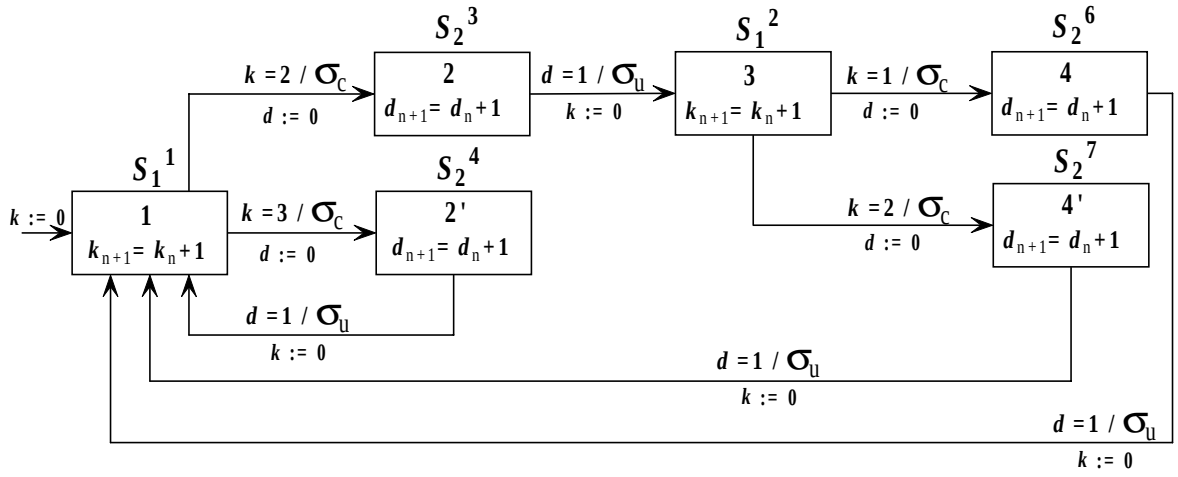


FIG. 4.12 – Modèle de superviseur après le re-pliage temporel

Propriété 4.4.2. *Deux sommets re-pliés peuvent fusionner si et seulement si*
– *leurs transitions d'entrée ont le même sommet source, et*
– *leurs transitions de sortie ont le même sommet destination et elles sont associées à des conditions de gardes identiques.*

■

Pour illustrer la démarche qui permet de réaliser la fusion entre deux sommets du modèle re-plié, reprenons le modèle illustré dans la Figure 4.12. Sur ce modèle on observe que les conditions de gardes associées aux transitions de sortie des sommets S_2^6 et S_2^7 sont identiques. Les transitions d'entrée de ces sommets sont étiquetées par des conditions de gardes différentes, cependant l'événement généré est le même. Par conséquent, il est possible fusionner ces deux sommets sous la contrainte que la transition d'entrée du nouveau sommet sera étiquetée par une condition de garde donnée sous la forme d'un intervalle représentant l'union des conditions de franchissement des transitions fusionnées.

Le modèle de commande final obtenu par l'exécution de la procédure de re-pliage pour le modèle de commande déplié est représenté dans la Figure 4.13.

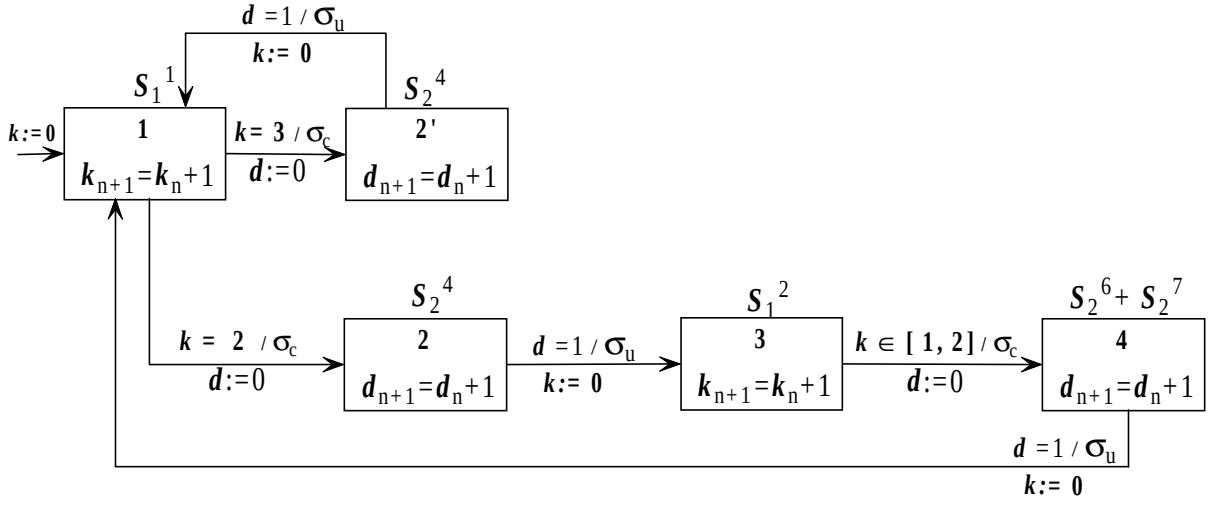


FIG. 4.13 – Modèle de superviseur après le re-pliage temporel

La garde de la transition d'entrée dans le sommet résultant après la fusion, spécifiée sous la forme d'intervalle permet de prendre en compte l'évolution continue du système et l'incertitude associée avec la date d'occurrence des événements.

D'une manière synthétique, la procédure de re-pliage peut être résumée par les étapes de l'algorithme suivant :

Algorithme 4.3. *Procédure de re-pliage*

Étant donné le modèle de commande déplié

1. Replier temporellement tous les groupes de sommets discrets
2. **POUR** chaque dynamique continue
   - 2.1. Identifer tous les sommets repliés temporellement qui correspondent à la même dynamique continue
   - 2.2. Sauvegarder les sommets ainsi obtenus, avec les informations concernant ses transitions d'entrée/sortie dans un vecteur noté S_{RT}
   - 2.3. **SI** pour la dynamique courante $S_{RT} \neq \emptyset$ **alors**
     - Trouver tous les éléments de S_{RT} qui ont la même garde associée avec ses transitions de sortie
     - Sauvegarder les résultats dans un vecteur S_F
     - Fusionner tous les sommets contenus dans le vecteur S_F

■

Dans l'algorithme présenté ci-dessus les vecteurs S_{RT} et S_F sont utilisés pour stocker les informations intermédiaires nécessaires pour l'opération de re-pliage du modèle.

Remarque 4.7. Le modèle de commande déplié et le modèle de commande obtenu par re-pliage sont *bisimilaires*.

■

Le fait que les systèmes soient bisimilaires indique aussi que les deux systèmes sont équivalents. Une preuve de la propriété ci-dessus, reste à développer.

Nous pouvons remarquer la simplicité du modèle du commande obtenu pour un système hybride. Le modèle de commande est donné sous la forme d'un automate temporisé. Il indique les dates d'occurrence auxquelles les événements contrôlables doivent être exécutés. On notera que l'on s'affranchit ici de l'aspect hybride du système.

Les résultats que nous venons de présenter dans cette dernière section, concernant le re-pliage d'un modèle déplié, sont des points de départ et restent à être affinés dans un travail de recherche futur.

4.5 Exemple d'application

Dans cette section nous allons illustrer la démarche de synthèse de la commande à travers un exemple. Nous allons reprendre l'exemple de la ligne de production introduit dans le chapitre 2 (Section 2.5.1).

Rappelons que la ligne de production considérée, (Figure 2.12) est composée par deux groupes de machines serveurs finis M_1 et M_2 et deux stocks S_1 et S_2. Le nombre de serveurs qui compose chaque groupe de machines est limité à la valeur 3 pour le premier groupe et à 7 pour le deuxième. Les spécifications de fonctionnement imposées au système sont :

– *spécifications continues* : Dans le stock S_1 le niveau de pièces x_1 doit être maintenu dans l'intervalle [1,4] et dans le stock S_2 le nombre de pièces x_2 doit rester dans l'intervalle [1,8] pendant l'évolution du système.

– *spécifications discrètes* : Le flux d'entrée peut être arrêté et nous pouvons décider de l'instant d'arrêt. Par contre, une fois le flux d'entrée arrêté, il redémarre automatiquement après 4 unités de temps.

L'objectif de la commande est de décider de l'instant d'arrêt du flux d'entrée du système tel que le niveau des pièces dans les stocks soit maintenu dans les intervalles spécifiés.

La synthèse de la commande démarre à partir du modèle automate atteignable obtenu par l'analyse d'atteignabilité et illustré dans la Figure 3.10 (Section 3.4.2). Pour améliorer la clarté de la présentation nous reprenons ce modèle dans la Figure 4.14.

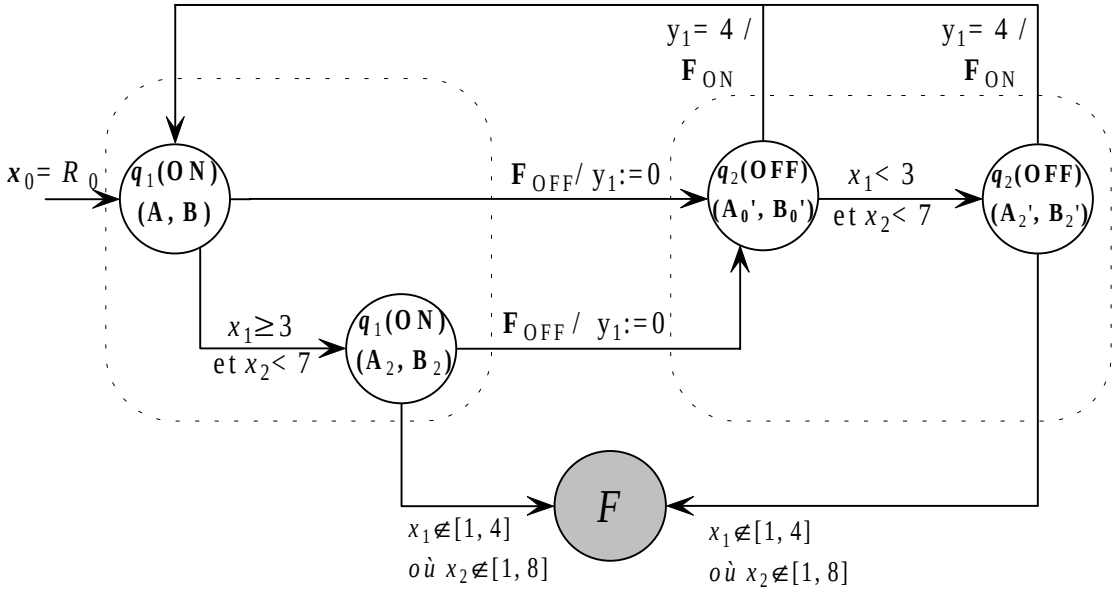


FIG. 4.14 – Automate atteignable pour la ligne de production

L'évolution hybride est régie seulement par quatre dynamiques continues dont l'interaction est réalisée par le franchissement d'une transition. Les quatre dynamiques correspondantes aux sommets de l'automate atteignable sont données dans le Tableau 4.1.

Etat du flux / Cond. logique	*Démarré*		*Arrêté*	
	Sommet	Dynamique continue	Sommet	Dynamique continue
$(x_1 < k_1)$ et $(x_2 < k_2)$	(A_0, B_0)	$\begin{bmatrix} -a_1 & 0 \\ a_1 & -a_2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}+\begin{bmatrix} d \\ 0 \end{bmatrix}$	(A_0', B_0')	$\begin{bmatrix} -a_1 & 0 \\ a_1 & -a_2 \end{bmatrix}\begin{bmatrix} k_1 \\ x_2 \end{bmatrix}$
$(x_1 \geq k_1)$ et $(x_2 < k_2)$	(A_2, B_2)	$\begin{bmatrix} -a_1 & 0 \\ a_1 & -a_2 \end{bmatrix}\begin{bmatrix} k_1 \\ x_2 \end{bmatrix}+\begin{bmatrix} d \\ 0 \end{bmatrix}$	(A_2', B_2')	$\begin{bmatrix} -a_1 & 0 \\ a_1 & -a_2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

TAB. 4.1 – Dynamiques correspondant aux sommets de l'automate hybride atteignable

La première étape de la synthèse de la commande consiste dans la construction de l'équivalent déplié de l'automate atteignable.

4.5.1 *Dépliage temporel de l'automate atteignable*

Afin de réaliser le dépliage de la dynamique continue du système il faut exploiter les résultats de l'analyse d'atteignabilité obtenus par simulation numérique. Pour une première exécution de l'automate, c'est-à-dire une succession d'états *flux d'entrée arrêté* et *flux d'entrée redémarré* le modèle déplié sera construit.

Pour ce faire, le dépliage temporel du groupe $q_1(ON)$ doit d'abord être réalisé. Les régions successeurs obtenues à partir d'une région initiale R_0 : $\{[1, 2] \times [1, 7]\}$ pour la première visite dans ces sommets sont représentées dans le tableau suivant :

Pas discret	**q_1(ON)**		**q_2(OFF)** après 4 pas discrets
	(A_0, B_0)	**(A_2, B_2)**	
$k = 0$	(1, 1), (2 , 1), (2, 7), (1 , 7)		(0.449329, 1.112302), (0.898658, 1.554284), (0.898658, 5.576205), (0.449329, 5.134222)
$k = 1$	(1.725077, 1.167610), (2.543808 , 1.339823), (2.543808 , 6.768848), (1.725077 , 6.596634)		(0.775127, 1.545126), (1.143007, 2.022428), (1.143007, 5.661612), (0.775127, 5.184309)
$k = 2$	(2.318720, 1.444138), (2.989040 , 1.740959), (2.989040 , 6.653343), (2.318720, 6.356522)		(1.041868, 1.992868), (1.343062, 2.488102), (1.343062, 5.780972), (1.041868, 5.285737)
$k = 3$	(2.804753, 1.796583), (3.353565, 2.180597), (3.353565, 6.625506), (2.804753 , 6.241493)		(1.260257, 2.443936), (1.506854, 2.943914), (1.506854, 5.923425), (1.260257, 5.423449)
$k = 4$		(3.202684, 2.199191), (3.652013 , 2.641173), (3.652013 , 6.663093), (3.202684 , 6.221111)	(1.428383, 2.896616), (1.674980, 3.365433), (1.674980, 6.061407), (1.428383, 5.592589)
$k = 5$		(3.602684, 2.560886), (4.052013, 2.960808), (4.052013, 6.599992), (3.602684, 6.200070)	

TAB. 4.2 – Successeurs continus

Les successeurs continus sont représentés sous la forme des polyèdres convexes, décrits dans le tableau par l'ensemble de ses quatre sommets.

Le système évolue en respectant la dynamique associé au sommet (A_0, B_0) pendant les premiers 3 pas discrets. Un changement de dynamique continue a lieu à cet instant car la valeur de la variable d'état x_1 devient supérieure à $k_1 = 3$. À partir de cet instant l'évolution du niveau des pièces dans les stock respectera la deuxième dynamique décrite par la paire (A_2, B_2).

Au 5-ème pas discret on observe que l'invariant du sommet hybride représenté par la condition logique $Inv(q_{ON})$: $\{x_1 \leq 4 \,\text{et}\, x_2 \leq 8\}$ est violé, alors le calcul des successeurs continus est arrêté. À chaque successeur continu calculé on associe un sommet discret, et les transitions reliant ces sommets sont étiquetées par l'événement t. Le sommet correspondant au dernier successeur, obtenu pour le 5-ème pas discret représentera le sommet interdit. À chaque sommet discret, une transition supplémentaire est étiquetée par l'événement contrôlable F_{OFF}.

La condition initiale à l'entrée du groupe des sommets hybrides $q_2(OFF)$, change en

fonction de l'instant auquel le flux d'entrée est arrêté. Dans la deuxième colonne du Tableau 4.2 sont indiquées les régions successeurs obtenues après un délai de 4 unités de temps depuis l'arrêt du flux d'entrée. Ainsi, la première région correspond au successeur obtenu si le flux d'entrée est arrêté pour la valeur de $k = 0$, la deuxième pour la valeur de $k = 1$, etc. On observe que si la décision d'arrêter le flux d'entrée correspond aux valeurs de $k = 0$ ou $k = 1$, l'évolution du système est conduite vers le sommet interdit.

Le modèle déplié correspondant à une succession d'états *flux d'entrée arrêté* et *flux d'entrée redémarré* est illustré dans la Figure 4.15.

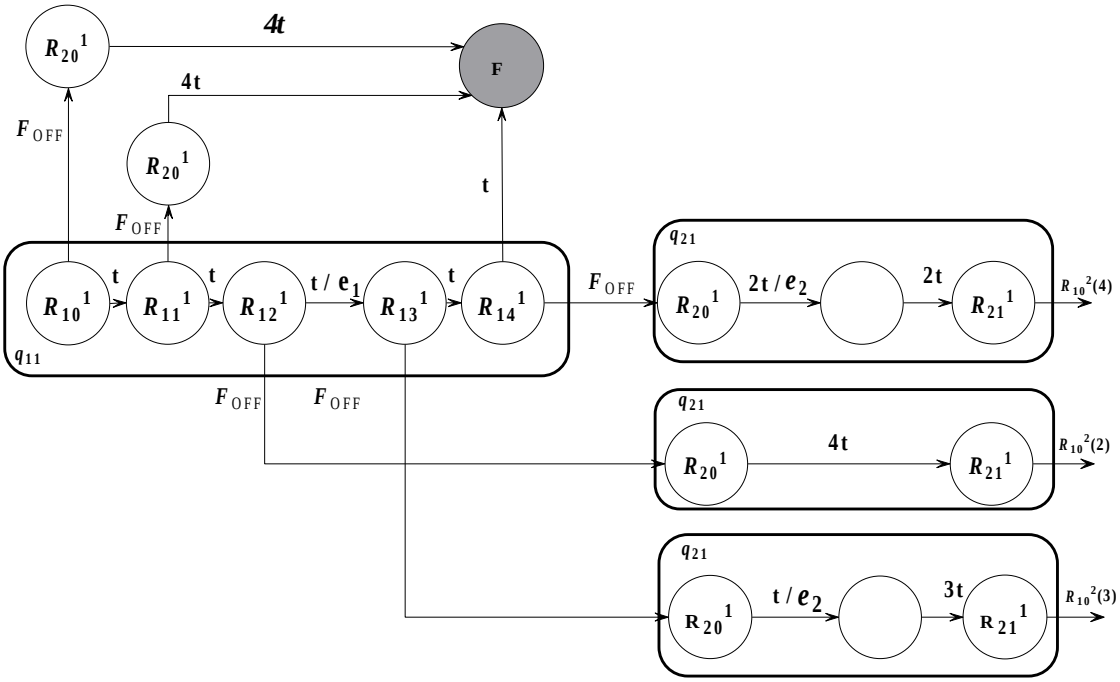


FIG. 4.15 – Dépliage partiel de l'automate atteignable

Dans ce modèle les commutations autonomes générées par la validation des conditions logiques spécifiées sur la valeur des variables d'état et le nombre de serveurs qui composent le groupe des machines sont aussi illustrées par les événements e_1 et e_2. Pour alléger les notations, nous notons par l'événement e_1 la validation de la condition $x_1 \geq 3$ et $x_2 < 7$ et par e_2 la validation de la condition $x_1 < 3$ et $x_2 < 7$.

Le modèle déplié complet de l'automate atteignable, correspondant aux résultats de l'analyse ne sera pas représenté vu sa complexité en terme de nombre de sommets qui le compose.

4.5.2 *Synthèse de superviseur*

La technique de synthèse du superviseur sera illustrée sur le modèle déplié partiel obtenu pour la ligne de production (Figure 4.15).

Le sommet interdit est modélisé sur le modèle déplié par le sommet F. Ce sommet peut être atteint par trois séquence d'événements :

- la séquence $F_{OFF}.4t$,
- la séquence $t.F_{OFF}.4t$,

– la séquence $5t$.

Les deux premières séquences contiennent chacune l'événement contrôlable F_{OFF}. Ainsi, l'évolution vers l'état interdit dans ces deux situations peut être empêchée en interdisant l'occurrence de cet événement.

Dans le cas de la dernière séquence, l'évolution vers l'état interdit peut être empêchée seulement par le forçage de l'occurrence de l'événement contrôlable F_{OFF} avant l'occurrence du dernier tick d'horloge (l'événement t). Ainsi, l'évolution du système vers le sommet interdit ne sera plus possible et après 4 tick d'horloge, l'événement F_{OFF} se produira.

Le modèle partiel obtenu par la synthèse de superviseur est représenté dans la Figure 4.16.

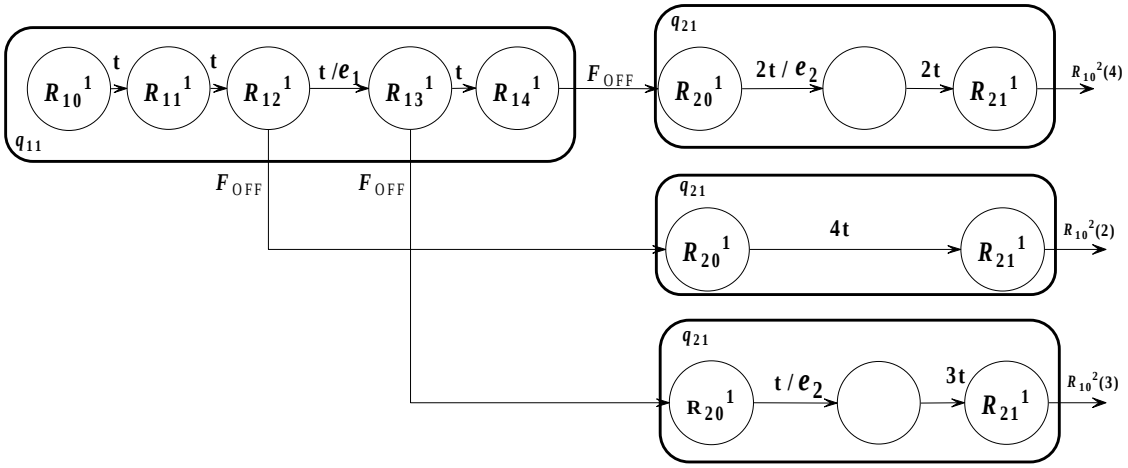


FIG. 4.16 – Modèle partiel du superviseur

Nous pouvons constater sur ce modèle partiel que l'occurrence de l'événement contrôlable est limitée par la synthèse à l'intervalle de temps [2,4]. On retrouve ici l'idée de base de n'importe quelle technique de synthèse qui est de restreindre l'évolution initiale du système afin d'éviter les évolutions non désirées du système.

4.5.3 *Procédure de re-pliage*

Afin d'illustrer le re-pliage d'un modèle déplié, considérons le modèle partiel du superviseur obtenu pour la ligne de production (Figure 4.16).

Dans ce modèle on identifie 4 groupes de sommets discrets. Un groupe est obtenu en évoluant par la dynamique continue associée à l'état du système dans lequel le flux d'entrée est démarré, noté q_{11}, et les trois autres générés par la dynamique associée à l'état du système où le flux d'entrée est arrêté, notés toutes les trois par q_{21}.

Par le re-pliage temporel du modèle, on peut regrouper les sommets appartenant au groupe q_{11} dans un seul sommet, noté ON. On peut procéder de la même façon avec les autres trois groupes. Les trois transitions de sortie du nouveau sommet ON sont des transitions d'entrée pour les sommets obtenus par le re-pliage des autres trois groupes. Ainsi, le modèle re-plié temporellement, correspondant au modèle partiel du superviseur est représenté dans la Figure 4.17(a).

Les conditions de gardes associées aux transitions d'entrée correspondent aux instants où la décision d'arrêter le flux est prise et par l'événement contrôlable F_{OFF}, généré lors du franchissement de la transition. L'horloge associée aux sommets compte l'écoulement

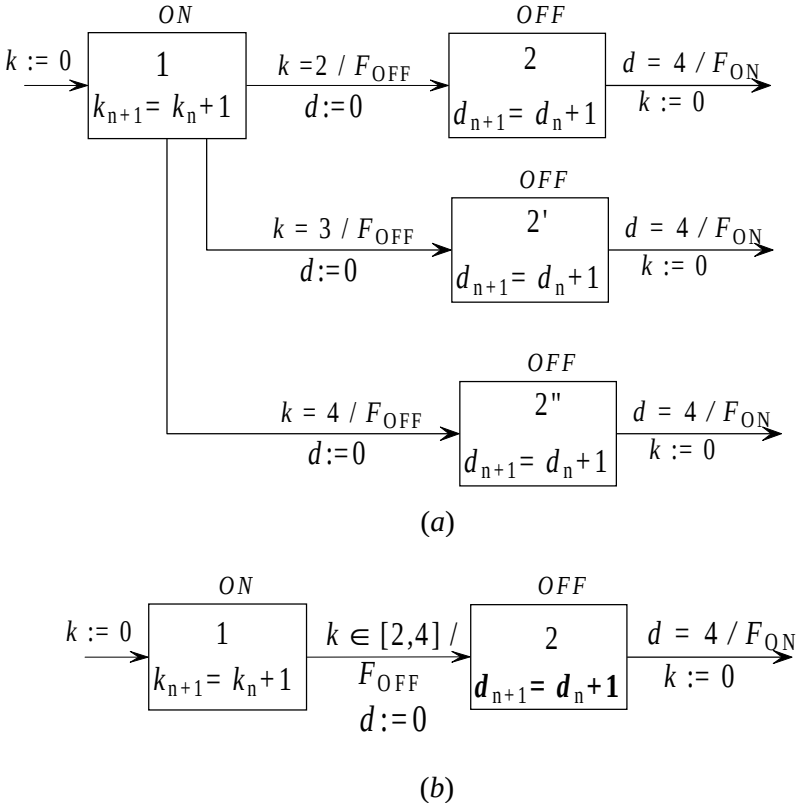


FIG. 4.17 – (a) Résultat du re-pliage temporel et (b) Modèle partiel de commande

du temps pendant le séjour du système dans le sommet. Les transitions de sortie du sommet OFF seront toutes étiquetées par le délai avant le redémarrage du flux d'entré.

Dans cette figure on peut observer que les trois sommets appelés OFF (2, 2' et 2") ont les mêmes conditions de gardes associées aux transitions de sortie. L'événement étiquetant la transition d'entrée dans ces sommets est aussi identique. Donc, les trois sommets appelés OFF peuvent être fusionnés et le modèle partiel de commande est donné sous la forme d'un automate temporisé représenté dans la Figure 4.17(b).

Le modèle de commande complet est obtenu par synthèse numérique et il est illustré dans la Figure 4.18.

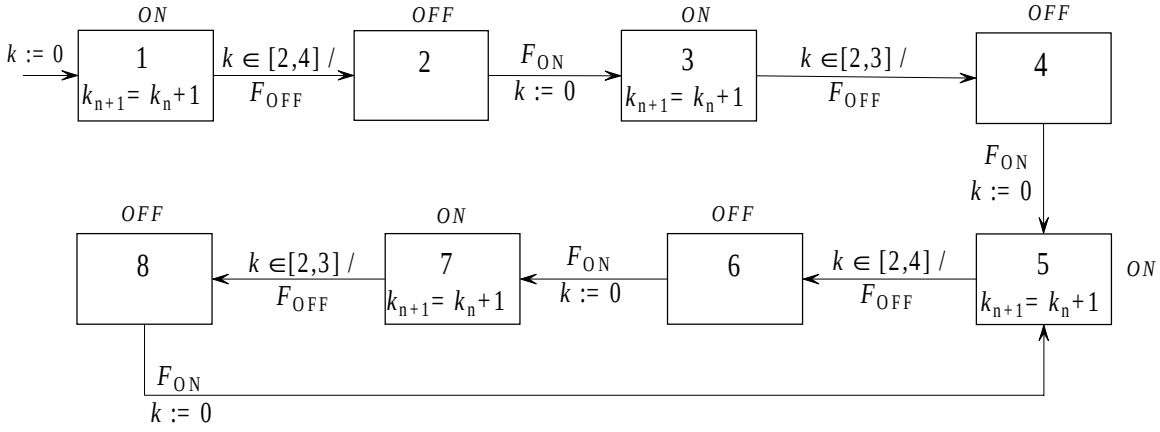


FIG. 4.18 – Modèle complet de commande pour la ligne de production

Dans ce modèle, sont indiquées les dates d'occurrence auxquelles l'événement contrôlable F_{OFF} doit être exécuté. Le superviseur ainsi obtenu est optimal.

4.6 Conclusions

Ce chapitre nous a permis de présenter la démarche que nous proposons pour la synthèse de la commande pour un système dynamique hybride. Notre approche est basée sur la représentation en temps discret de la dynamique continue et se propose d'exploiter les résultats de l'analyse d'atteignabilité.

D'abord, nous avons proposé un modèle représenté par un automate à états finis abstrait à partir de l'automate atteignable. Ce modèle a été construit par dépliage temporel de l'évolution continue du système. Le modèle résultant permet l'utilisation des techniques de synthèse formelles basées sur les travaux de Brandin et Wonham dans le domaine de la commande par supervision des systèmes à événements discrets.

Une fois le modèle déplié construit, nous avons illustré la technique de la synthèse du superviseur. L'idée est d'éliminer toutes les trajectoires du graphe qui conduisent vers le sommet interdit en agissant sur la date d'occurrence des événements contrôlables. Des structures de contrôle ont été proposées et un algorithme systématisant les étapes de la synthèse a été présenté.

Afin de réduire la complexité du modèle de commande en nombre de sommets, une procédure de re-pliage est proposée. Cette procédure permet obtenir les instants d'occurrence des événements contrôlables sous la forme d'une représentation plus compacte. Le résultat du re-pliage est le modèle de commande exprimé sous la forme d'un automate temporisé.

L'inconvénient de la démarche proposée réside dans l'explosion combinatoire du nombre de sommets lors du dépliage temporel, à cause de l'utilisation du temps-discret pour la représentation de la dynamique continue. Cependant, cet inconvénient doit être relativisé car le modèle de commande est très simple. L'automate temporisé obtenu caractérise toutes les lois de commande qui garantissent que le système hybride commandé respecte les spécifications imposées. De plus, le grand avantage est que le résultat de la synthèse est optimal.

Conclusions et perspectives

L'objectif du travail présenté dans ce mémoire a été de proposer une méthodologie permettant la synthèse d'un modèle de commande optimale pour un système dynamique hybride.

Pour atteindre cet objectif, nous avons orienté notre travail vers l'analyse et la commande des systèmes hybrides en combinant la puissance d'analyse de l'automate hybride avec les avantages d'une représentation en temps discret de la dynamique continue.

D'abord, nous avons proposé une approche permettant d'analyser le comportement dynamique d'un système hybride modélisé par un automate hybride à temps discret. Cette approche nous a permis de vérifier la validité du modèle automate hybride modélisant le système étudié, ainsi que d'apporter des simplifications dans ce modèle en supprimant toutes les évolutions qui ne sont jamais réalisables pour un ensemble de conditions données. Le résultat de cette démarche est l'automate atteignable qui contient toutes les trajectoires possibles du système hybride.

Une première contribution du travail présenté est le développement d'un algorithme pour construire d'une manière systématique l'automate atteignable qui constitue le modèle de départ pour la synthèse de la commande. Les comportements non désirées du système hybride sont modélisés par des sommets interdits de l'automate atteignable.

Une fois le modèle automate atteignable obtenu, nous avons abordé la deuxième partie importante de notre travail de recherche - la synthèse de la commande. L'utilisation du temps discrétisé permet d'obtenir un modèle automate à états finis sur lequel les techniques de synthèse formelle classiques peuvent être appliquées. Ainsi, l'approche de synthèse est composée de trois étapes :

- Dans un premier temps, nous modélisons le système étudié par un modèle équivalent représenté sous la forme d'un automate à états fini temporisé, appelé automate déplié. Pour cela nous nous sommes basées sur le cadre théorique développé par Brandin et Wonham pour les systèmes à événements discrets temporisés.

  Ce modèle est obtenu à partir du modèle automate atteignable par le dépliage temporel de la dynamique continue. Un algorithme permettant la construction systématique de ce modèle est également proposé.

  Le modèle ainsi obtenu représente l'évolution du système hybride et il est approprié pour l'utilisation des techniques classiques de synthèse du superviseur.

- D'une manière générale, la problématique de synthèse de la commande est de trouver une manière systématique d'intervenir dans l'évolution du système telle que les spécifications imposées par le cahier de charges soient toujours vérifiées. Le superviseur restreint le comportement du procédé sous supervision afin d'obtenir un comportement qui ne soit jamais inclus dans l'ensemble des comportements non désirés.

A partir du comportement du procédé modélisé par un automate à états finis, cette représentation permet d'avoir une description finie des trajectoires du système. Afin d'éviter l'évolution vers les sommets interdits de ce modèle, le superviseur peut intervenir dans l'évolution du système par l'intermédiaire des événements contrôlables, les événements incontrôlables étant toujours autorisés. L'objectif de la commande consiste alors à contrôler toutes les trajectoires qui mènent vers les sommets interdits en agissant sur la date d'occurrence des événements contrôlables.

Afin de contrôler les évolutions futures du système, nous avons proposé un ensemble des structures de contrôle. Chaque structure de commande modélise une configuration possible qui peut être rencontrée dans le modèle déplié et fournit une décision permettant d'empêcher une évolution non désirée du système. La technique de synthèse est basée sur l'analyse des trajectoires vers le sommet interdit. A partir de ce sommet, on remonte les branches en cherchant à en identifier une parmi les structures de contrôle proposées. Un algorithme qui synthétise toutes les étapes de la synthèse a été aussi proposé.

Notre démarche permet de déterminer toutes les trajectoires admissibles pour le fonctionnement du système en respectant les spécifications imposées.

Les conditions d'existence du superviseur ont été formulées en se basant sur le cadre théorique existant pour les systèmes temporisés. Étant donné que la synthèse de superviseur est basée sur la théorie classique de la supervision, l'optimalité du résultat est ainsi garantie. Donc, le superviseur obtenu est le plus permissif possible. Une preuve formelle de toutes ces concept a été également fournie.

- Le dépliage temporel en vu de la synthèse de superviseur a fait apparaître un ensemble de sommets intermédiaires, augmentant la taille du modèle de commande résultant. Parmi ces sommets nous avons vu qu'il y en a plusieurs qui sont inutiles, dans le sens où ils ne contiennent pas d'informations pertinentes pour la commande. Afin de réduire la complexité du modèle de commande, nous avons proposé une démarche de re-pliage du modèle en supprimant tous ces états inutiles.

  Nous noterons plus particulièrement la simplicité du modèle de commande obtenu après re-pliage. Il est représenté sous la forme d'un automate temporisé avec une seul horloge, alors qu'au départ il s'agissait d'un système hybride avec des dynamiques quelconques.

  Le modèle obtenu est un modèle "nécessaire et suffisant" qui répond aux spécifications, où seule l'information pertinente est conservée. C'est en ce sens que c'est une généralisation de la méthode "clock translation".

Par ce travail de recherche nous avons essayé de répondre aux problèmes d'analyse et synthèse de la commande pour un système hybride dans le cas général. Les résultats obtenus peuvent s'appliquer aussi bien au pilotage des systèmes de production, qui ont constitué l'objet de notre étude de cas, qu'au contrôle des flux dans les procédés batch.

Les perspectives que nous envisageons pour notre travail de recherche sont de plusieurs ordre. Tout d'abord, une perspective immédiate pour la continuation du travail de recherche présenté dans ce mémoire est d'affiner les résultats présentées dans la dernière section, concernant la technique de re-pliage proposée et de développer un cadre formel concernant ce problème. Il est également nécessaire de valider les algorithmes proposés pour d'autres classes d'exemples d'application. Ensuite, étant donné le volume de calcul engendré par les différents algorithmes proposés, ainsi que la complexité des modèles obtenus, il nous est paru nécessaire d'implémenter les algorithmes proposés. Les résultats

présentés dans ce travail ont été obtenus par calcul numérique étape par étape. Ceci constitue une des perspectives les plus importantes afin de pouvoir valider la démarche sur des autres exemples d'application. Enfin, il est important d'étudier la complexité des algorithmes proposés.

Bibliographie

[***01] ******. *Systèmes Dynamiques Hybrides, sous la direction de Janan Zaytoon.* Hermes Science, 2001.

[ABD+00] E. Asarin, O. Bournez, T. Dang, O. Maler, and A. Pnueli. Effective synthesis of switching controllers for linear systems. *In Proceedings of IEEE*, 2000.

[ABDM00] E. Asarin, O. Bournez, T. Dang, and O. Maler. Approximate reachability analysis of piecewise-linear dynamical systems. *In Hybrid Systems : Computation and Control, LNCS*, 1790 :20–31, 2000.

[ACH+95] R. Alur, C. Courcoubetis, N. Halbawacs, T.A. Henzinger, X. Nicolin P.H. Ho, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138 :3–34, 1995.

[ACHH93] R. Alur, C. Courcoubetis, T.A. Henzinger, and P.H. Ho. Hybrid automata : an algorithmic approach to the specification and verification of hybrid systems. *In Hybrid Systems, LNCS*, 736 :209–229, 1993.

[AD94] R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126 :183–235, 1994.

[AK98] P. Antsaklis and X. Koutsoukos. On hybrid control of complex systems : a survey. *Symposium ADPM'98, Reims, France*, 1998.

[All98] M. Allam. *Sur l'analyse quantitative des RdP hybrides : une approche basée sur les automates hybrides.* PhD thesis, Laboratoire d'Automatique de Grenoble - Institut National Polytechnique de Grenoble, 1998.

[AM99] E. Asarin and O. Maler. As soon is possible : time optimal control of timed automata. *In Hybrid Systems : Computation and Control HSCC'99, LNCS*, 1569 :19–30, 1999.

[ASL93] P. Antsaklis, J. Stiver, and M. Lemmon. Hybrid system modeling and autonomous control systems. *Lecture Notes in Computer Science*, 736 :366–392, 1993.

[BAD91] J. Le Bail, H. Alla, and R. David. Hybrid petri nets. *In Proceedings of the European Control Conference, Grenoble - France*, pages 1472–1477, 1991.

[BBM94] M.S. Branicky, V.S. Borkar, and S.K. Mitter. A unified framework for hybrid control. *In Proceedings of the 33rd Conference on Decision and Control*, pages 4228–4234, 1994.

[BM95] M.S. Branicky and S.K. Mitter. Algorithms for optimal hybrid control. *In Proceedings of the 34rd Conference on Decision and Control*, pages 2661–2666, 1995.

[BMP99] O. Bournez, O. Maler, and A. Pnueli. Orthogonal polyhedra : Representation and computation. *In Hybrid Systems : Computation and Control, LNCS*, 1569 :46–60, 1999.

[Bra95] M.S. Branicky. *Studies in hybrid systems : Modeling, Analysis and Control*. PhD thesis, Deptartement of Electrical Engineering and Computer Science-MIT, Cambridge, 1995.

[Bra98] M.S. Branicky. Multiple lyapunov functions and other analysis tools for switched and hybrid systems. *In IEEE Transaction on Automatic Control, Special Issue on Hybrid Systems*, 43 :475–482, 1998.

[Bro93] R. Brockett. Hybrid models for motion control systems. *Essays in Control : Perspectives in the Theory and its Applications*, pages 29–53, 1993.

[Bro94] R. Brockett. Language driven hybrid systems. *In Proceedings of the 33rd Conference on Decision and Control*, pages 4210–4214, 1994.

[BW94] B. Brandin and W.M. Wonham. Supervisory control of timed discrete event systems. *IEEE Transactions on Automatic Control*, 39(2) :329–341, 1994.

[Cho97] A. Chombart. *Commande Supervisée de Systèmes Hybrides*. PhD thesis, Laboratoire d'Automatique de Grenoble-Institut National Polytechnique de Grenoble, 1997.

[CK98] A. Chutinan and B. Krogh. Computing polyhedral approximations to flow pipes dor dynamic systems. *In Proceedings of 37th IEEE Conference on Decision and Control*, 1998.

[DA90] R. David and H. Alla. Autonomous and timed continuous petri nets. *11th International Conference on Application and Theory of Petri Nets, Paris, France*, pages 367–386, 1990.

[Dan00] T. Dang. *Vérification et synthèse des systèmes hybrides*. PhD thesis, Verimag-Institut National Polytechnique de Grenoble, 2000.

[DF82] D. Dubois and J.P. Forestier. Productivité et en-cours moyens d'un ensemble de deux machines séparées par un stock. *RAIRO Automatique*, 16(2) :105–132, 1982.

[Fav99] A. Favela. *Modélisation et analyse du comportement dynamiques des systèmes hybrides : une approche basée sur le modèle de l'automate hybride*. PhD thesis, Laboratoire d'Automatique de Grenoble-Institut National Polytechnique de Grenoble, 1999.

[GS80] S.B. Gershwin and I.C. Schick. Continous model of an unreliable two-stage material flow system with finite interstage buffer. *Technical Report - MIT LIDS-R-1032*, 1980.

[GU96] A. Giua and E. Usai. High-level petri nets : A definition. *In Proceedings of the 35th Conference on Decision and Control, Kobe, Japon*, 1996.

[Hen96] T.A. Henzinger. The theory of hybrid automata. *Proceedings of the 11th Annual Symposium on Logic in Computer Science, LNCS*, pages 278–292, 1996.

[HH95] T.A. Henzinger and P.H. Ho. A note on abstract interpretation strategies for hybrid automata. *In Hybrid Systems II, LNCS*, 999 :252–263, 1995.

[HHMWT00] T.A. Henzinger, B. Horowitz, R. Majumdar, and H. Wong-Toi. Beyond hytech : hybrid system analysis using interval numerical methods. *In Proceedings of 3rd Int. Workshop on Hybrid Systems : Computation and Control HSCC'00, LNCS(1790)*, 1790 :130–144, 2000.

[HHWT95] T.A. Henzinger, P.H. Ho, and H. Wong-Toi. A user guide to hytech. *In Tools and Algorithms for the Construction and Analysis of Systems, LNCS(1019)*, pages 41–71, 1995.

[HHWT98] T.A. Henzinger, P.H. Ho, and H. Wong-Toi. Algorithmic analysis of non-linear hybrid systems. *In IEEE Transactions on Automatic Control*, 43(4) :540–554, 1998.

[HKPV95] T.A. Henzinger, P. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata ? the algorithmic analysis of hybrid systems. *Proceeding of the 27th Annual Symposium on Theory of Computing*, pages 373–382, 1995.

[HR98] T.A. Henzinger and V. Rusu. Reachability verification for hybrid automata. *In Proceedings of 1st International Workshop on Hybrid Systems : Computation and Control, LNCS*, 1386 :190–204, 1998.

[HU79] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Langages and Computation.* Addison-Wesley, 1979.

[KA01a] M. Kurovszky and H. Alla. Analysis and control of hybrid systems using a discrete-event and discrete-time approach. *Note Interne AP01-266, Laboratoire d'Automatique de Grenoble - France*, 2001.

[KA01b] M. Kurovszky and H. Alla. Analysis and control of manufacturing systems : a hybrid approach. *IFAC/IFOR/IFIP Symposium on Large Scale Systems, Bucharest- Romania*, 2001.

[KA01c] M. Kurovszky and H. Alla. An approach for the analysis and control of hybrid systems. *IEEE International Symposium On Intelligent Control 2001, Mexico City - Mexic*, 2001.

[KGM91] R. Kumar, V. Garg, and S.I. Marcus. On contrôlability and normality of discrete event dynamical systems. *System Control Letters*, 17 :157–168, 1991.

[KP92] Y. Kesten and A. Pnueli. Timed and hybrid statecharts and their textual representation. *Formal techniques in real-time and fault-tolerant systems, LNCS*, 571 :591–620, 1992.

[Kum91] R. Kumar. *Supervisory Synthesis Techniques for Discrete Event Dynamical Systems.* PhD thesis, University of Texas AT Austin, 1991.

[Kur02] M. Kurovszky. Analyse et commande des systèmes hybrides : une approche à temps discrets. *Conférence Internationale Francophone d'Automatique - CIFA'02, Nantes - France*, 2002.

[LGS96] J. Lygeros, D.N. Godbole, and S. Sastry. Multiagent hybrid system design using game theory and optimal control. *In Proceedings of the 35th IEEE Conference on Decision and Control, Kobe, Japon*, pages 1190–1195, 1996.

[LGS98] J. Lygeros, D.N. Godbole, and S. Sastry. Verified hybrid controllers for automated vehicles. *In IEEE Transactions on Automatic Control, Special Issue on Hybrid Systems*, 43 :522–539, 1998.

[LL98] L. Libeaut and J.J. Loiseau. Commande de sed - théorie des langages commandables. *26ème Ecole de Printemps d'Informatique théorique "Algèbre Max-plus et applications en informatique et automatique, Noirmoutier - France*, 1998.

[LTS99] J. Lygeros, C. Tomlin, and S. Sastry. Controllers for reachability specifications for hybrid systems. *Automatica*, 35(3), 1999.

[MC91] A. Mandelbaum and H. Chen. Discrete flow networks : Bottleneck analysis and fluid approximations. *Math. Operations Research*, 16 :408–446, 1991.

[Moo96] R.E. Moore. *Interval Analysis*. Prentice-Hall, 1996.

[NK93] A. Nerode and W. Kohn. Models for hybrid systems : Automata, topologies, controllability, observability. *In Hybrid Systems, LNCS*, 736 :317–356, 1993.

[NOSY93] X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. An approch to the description and analysis of hybrid systems. *Lecture Notes in Computer Science*, 736, 1993.

[PBVV96] A. Puri, V. Borkar, V., and P. Varaiya. ϵ-approximation of differential inclusions. *In Proceedings of Hybrid Systems III Workshop : Verification and Control, LNCS*, 1066 :362–376, 1996.

[Pet62] C. Petri. *Kommunication mit automaten*. PhD thesis, Univeristy of Bonn, Germany, 1962.

[PL95] S. Pettersson and B. Lennartson. Hybrid modeling focused on hybrid petri nets. *In Proceedings of the 2nd European Workshop on Real-Time and Hybrid Systems, Grenoble, France*, pages 303–309, 1995.

[PV94] A. Puri and P. Varaiya. Decidability of hybrid systems with rectangular differential inclusions. *In D.L.Dill editor, CAV'94, Computer-aided Verification, LNCS*, 818 :85–104, 1994.

[PV95] A. Puri and P. Varaiya. Verification of hybrid systems using abstractions. *In Hybride Systems II, LNCS*, 999 :359–369, 1995.

[Rih94] R. Rihm. Interval methods for initial value problems in ode. *Topics in validated computations, North-Holland*, 1994.

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

[RW89] J.G. Ramadge and W.M. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1) :81–97, 1989.

[SAL95] J.A. Stiver, P.J. Antsaklis, and M.D Lemmon. Interface and controller design for hybrid control systems. *Rapport Technique, ISIS Group, University of Notre Dame*, 1995.

[SAL96a] J.A. Stiver, P.J. Antsaklis, and M.D Lemmon. An invariant based approach to the design of hybrid control systems. *IFAC 13th Triennial World Congress, San Francisco Ca.*, J :467–472, 1996.

[SAL96b] J.A. Stiver, P.J. Antsaklis, and M.D Lemmon. A logical des approch to the design of hybrid control systems. *Mathl. Comput., Modelling*, 23(11/12) :55–76, 1996.

[Tav87] L. Tavernini. Differential automata and their discrete simulators. *Nonlinear Analysis, Theory, Methods and Applications*, 11(6) :665–683, 1987.

[TE98] M. Tittus and B. Edgardt. Control design for integrator hybrid systems. *IEEE Transactions on automatic Control, Special Issue on Hybrid Systems*, 43 :491–500, 1998.

[TPS98] C. Tomlin, G.J. Pappas, and S. Sastry. Conflict resolution for air traffic management : A study in multi-agent hybrid systems. *IEEE Transactions on Automatic Control, Special Issue on Hybrid Systems*, 43 :509–521, 1998.

[Wit66] H.S. Witsenhausen. A class of hybrid-state continuous-time dynamic systems. *IEEE Transactions in Automatic Control*, 11(2), 1966.

Annexe A

Généralisation de la méthode "Clock Translation"

Dans cette annexe, nous allons présenter la méthode *clock translation* proposée par Henzinger, Ho et Wong-Toi [HHWT98] ainsi que la généralisation que nous la proposons pour cette méthode.

La méthode *clock translation* consiste dans la construction d'un modèle bisimilaire d'un automate hybride en remplaçant les variables non linéaires de l'automate par des horloges. La méthode est applicable pour les automates hybrides continu-linéaires invariants dans le temps découplé H_{CLID} [Fav99].

Étant donné un automate H_{CLID}, pour obtenir un automate temporisé on remplace chaque variable no linéaire x par une horloge t_x si les conditions suivantes sont vérifiées :

- On n'a que des prédicats simples [1] dans les conditions initiales et de franchissement des transitions.
- La variable x a une courbe de solution unique depuis n'importe quel point initial (découplage continu).
- La variable x a des courbes de solutions rigoureusement monotones pour déterminer, avec la valeur de t_x la valeur de vérité des prédicats tel que $x > c$ pour quel que soit $c \in \mathbb{R}$.
- A un certain instant, la valeur initiale dans la courbe des solution de x et le temps écoulé à partir de cet état initial doivent être connus.

Ainsi, la construction de l'automate bisimilaire se fait en deux pas :

1 - Chaque sommet de l'automate est subdivisé en k sommets tel qu'on obtient un sommet pour chaque valeur initiale c_k de x (chaque entrée du sommet). Ensuite, on ajoute une horloge t_x dans chaque nouveau sommet et on fait une remise à zéro de t_x à chaque transition discrète.

2 - La condition initiale, les conditions de franchissement des transitions et les invariants qui sont fonction de la variable x, sont remplacées par des relations en fonction de la variable t_x. L'horloge t_x mesure le temps de séjour dans chaque sommet de l'automate. Ainsi, les conditions de franchissement des transitions et les invariants de l'automate temporisé sont déterminées à partir du calcul des temps de séjour dans chaque sommet de l'automate hybride.

[1] Un prédicat est dit simple lorsqu'il existe une relation $x \sim c$, pour $x \in X$ tel que $\sim = \{<, \leq, =, \geq, >\}$ et $c \in \mathbb{R}$.

Nous illustrons l'utilisation de la méthode sur l'exemple suivant :

Exemple A.1. Considérons le modèle automate hybride de la Figure A.1, qui décrit le fonctionnement d'un thermostat. On veut obtenir un automate temporisé, bisimilaire linéaire de l'automate hybride par la méthode "clock translation".

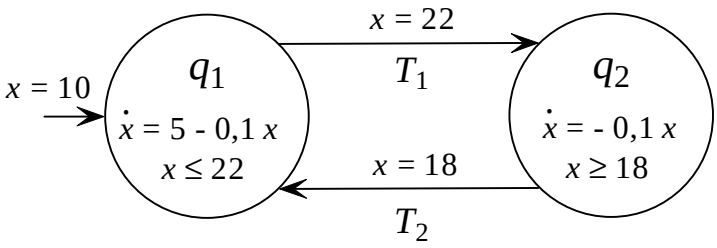


FIG. A.1 – Automate hybride modélisant un thermostat

Par la méthode "clock translation" on a :

Pas 1 :

Le sommet q_1 a deux entrées, l'entrée liée à la condition initiale $x = 10$ et l'entrée liée à la valeur initiale $x = 18$ (c_1 et c_2 respectivement). Le sommet q_2 a une seule entrée liée à la valeur initiale $x = 22$. Comme illustre la Figure A.2, on construit un sommet pour chaque entrée du sommet q_1, on ajoute une horloge t_x dans chaque nouveau sommet et on fait une remise à zéro de l'horloge t_x à l'instant quand les commutations discrètes ont lieu.

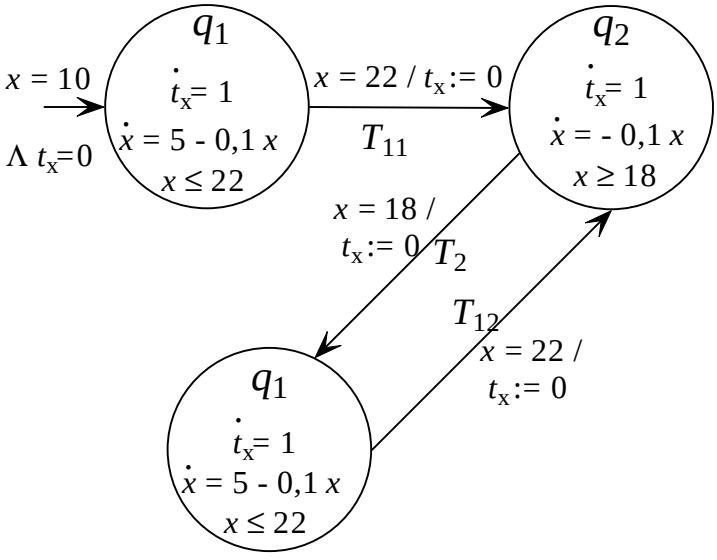


FIG. A.2 – Pas 1 pour la construction de l'automate bisimilaire

Pas 2 :

Les invariants ainsi que les conditions de franchissement des transitions sont calculés en fonction des temps de séjour dans les sommets. Les conditions initiale et d'entrée dans les sommets sont nulles. Ainsi on obtient l'automate temporisé équivalent illustré par la Figure A.3.

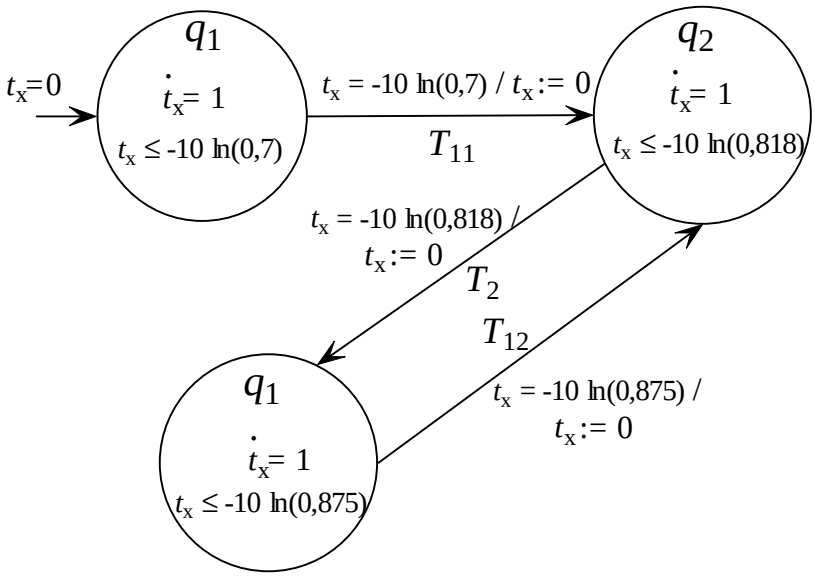


FIG. A.3 – Pas 1 pour la construction de l'automate bisimilaire

Il s'agit d'une méthode efficace mais la multitude des hypothèses considérées (comme par exemple, la solvabilité des variables) limite considérablement les champs d'application de cette méthode. Ainsi, la méthode "clock translation" est applicable seulement aux automates H_{CLID}.

La méthode présentée ci-dessus est une méthode analytique. Par notre démarche nous proposons en quelque sorte une généralisation de la méthode "clock translation". Dans notre travail, les dynamique continues du système sont couplées et seule une solution numérique est possible.

La Figure A.4 illustre la manière dont nous obtenons un modèle temporisé équivalent d'un automate hybride avec des dynamiques continues quelconques.

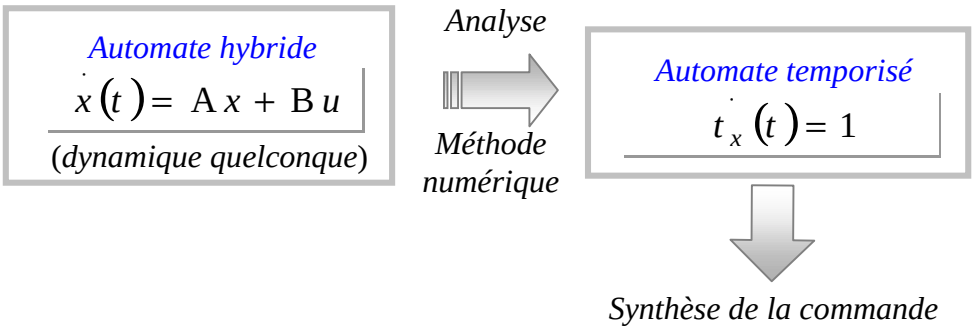


FIG. A.4 – Généralisation de la méthode "Clock Translation"

Au départ, nous avons le modèle automate hybride du système étudié. Pour ce modèle, on applique l'Algorithme 3.3 permettant la construction de l'automate atteignable, modélisant seulement les évolutions possibles pour une condition initiale spécifiée. Par cet algorithme, lors de l'exécution des différentes étapes qui le composent, nous allons mémoriser dans la pile *trace* associées au sommets de l'automate l'évolution complète du système dans chaque état avant la franchissement d'une transition de sortie ainsi que les temps de séjour du système dans chaque sommet lors de chaque visite. Par conséquent, une fois l'automate atteignable obtenu, nous pouvons également exprimer les conditions

de franchissements temporelles associées aux transitions du modèle par des conditions temporelles. Ce modèle est approprié pour lui appliquer la démarche de synthèse qu'on propose dans le Chapitre 4.

Remarque A.1. Pour les systèmes hybrides modélisés par des automates hybrides avec des dynamiques continues couplées, le modèle temporisé équivalent contient une seule horloge, tant que pour les automates hybrides avec des dynamiques continues découplées nous aurions autant d'horloges que des dynamiques découplées.

■

Dans l'annexe suivante, nous allons illustrer ce propos pour l'exemple de la ligne de production. Ainsi, l'automate atteignable sera exprimé sous la forme d'un modèle temporisé.

Annexe B

Exemple de construction de l'automate atteignable

Dans cette annexe nous illustrons une itération complète de l'Algorithme 3.3 (une succession d'états physiques flux d'entrée démarré/ flux d'entrée arrêté) permettant de construire de l'automate atteignable pour l'exemple de la ligne de production. Reprenons l'exemple de la ligne de production introduit dans le chapitre 2 (Section 2.5).

Considérons le modèle de l'automate hybride présenté dans la Figure 3.9 avec les dynamiques correspondantes aux sommets représentées dans le tableau 3.1. Pour plus de clarté, nous reprenons ci-dessous ce modèle avec les dynamiques correspondantes (Figure B.1 et Tableau B.2).

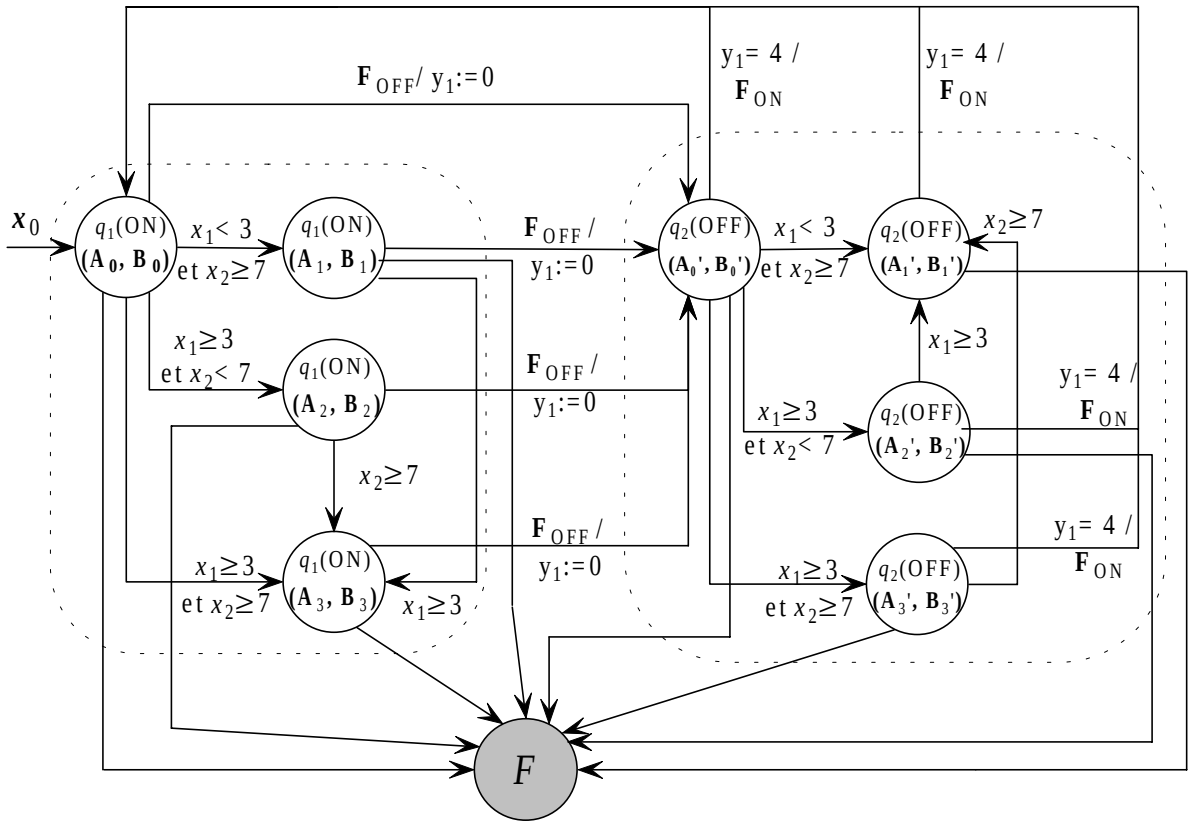


FIG. B.1 – Automate hybride avec commutations autonomes modélisant la ligne de production

Etat du flux / **Cond. logique**	*Démarré*		*Arrêté*	
	Sommet	Dynamique continue	Sommet	Dynamique continue
$(x_1 < k_1)$ et $(x_2 < k_2)$	(A_0, B_0)	$\begin{bmatrix} -a_1 & 0 \\ a_1 & -a_2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}+\begin{bmatrix} d \\ 0 \end{bmatrix}$	(A_0', B_0')	$\begin{bmatrix} -a_1 & 0 \\ a_1 & -a_2 \end{bmatrix}\begin{bmatrix} k_1 \\ x_2 \end{bmatrix}$
$(x_1 < k_1)$ et $(x_2 \geq k_2)$	(A_1, B_1)	$\begin{bmatrix} -a_1 & 0 \\ a_1 & -a_2 \end{bmatrix}\begin{bmatrix} x_1 \\ k_2 \end{bmatrix}+\begin{bmatrix} d \\ 0 \end{bmatrix}$	(A_1', B_1')	$\begin{bmatrix} -a_1 & 0 \\ a_1 & -a_2 \end{bmatrix}\begin{bmatrix} x_1 \\ k_2 \end{bmatrix}$
$(x_1 \geq k_1)$ et $(x_2 < k_2)$	(A_2, B_2)	$\begin{bmatrix} -a_1 & 0 \\ a_1 & -a_2 \end{bmatrix}\begin{bmatrix} k_1 \\ x_2 \end{bmatrix}+\begin{bmatrix} d \\ 0 \end{bmatrix}$	(A_2', B_2')	$\begin{bmatrix} -a_1 & 0 \\ a_1 & -a_2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$
$(x_1 \geq k_1)$ et $(x_2 \geq k_2)$	(A_3, B_3)	$\begin{bmatrix} -a_1 & 0 \\ a_1 & -a_2 \end{bmatrix}\begin{bmatrix} k_1 \\ k_2 \end{bmatrix}+\begin{bmatrix} d \\ 0 \end{bmatrix}$	(A_3', B_3')	$\begin{bmatrix} -a_1 & 0 \\ a_1 & -a_2 \end{bmatrix}\begin{bmatrix} k_1 \\ k_2 \end{bmatrix}$

FIG. B.2 – Dynamiques correspondant aux sommets de l'automate hybride obtenu par la décomposition structurelle

Initialement le flux d'entrée du système est démarré. Dans la suite nous allons appliquer les étapes de l'algorithme 3.3.

Pas 1 : *Initialisation*

- Initialement, le niveau des pièces dans les deux stocks de la ligne de production est $x_1 \in [1, 2]$ pièces dans le stock S_1 et $x_2 \in [1, 7]$ pièce dans le stock S_2. On observe que la condition initiale est exprimée par une région. Ceci nous permettra de faire une analyse puis une synthèse qui seront robustes vis-à-vis des incertitudes du modèle. Le nombre de serveurs qui composent les deux groupes de machines, M_1 et M_2, est $k_1 = 3$ et $k_2 = 7$ machines.
- Le sommet initial est (A_0, B_0). Initialisation des variables *compteur* := 0, *indice* := 1, $m := 0$ et $n := 1$.
- Le sommet courant a 5 transitions de sortie dont les conditions de franchissement sont :
  - $(x_1 < 3)$ et $(x_2 \geq 7)$;
  - $(x_1 \geq 3)$ et $(x_2 < 7)$;
  - $(x_1 \geq 3)$ et $(x_2 \geq 7)$;
  - $F_{OFF}/y_1 := 0$ où F_{OFF} est un événement contrôlable dont la date d'occurrence peut être décidée, et
  - la transition qui mène vers le sommet interdit dont la garde est $(x_1 \notin [1, 4])$ et $(x_2 \notin [1, 8])$.
- *Aller au Pas 2.*

Analyse d'atteignabilité dans le sommet (A_0, B_0)

Pas 2 : Étant la première visite dans le sommet (A_0, B_0), créer le vecteur trace correspondant au sommet - $trace_{(A_0, B_0)}$ - et commencer l'analyse de franchissabilité des transitions de sortie.

Pas 3 :*Analyse de la franchissabilité des transitions de sortie du sommet* (A_0, B_0)

Le système peut séjourner dans le sommet (A_0, B_0) tant que l'invariant du sommet -

représenté par la condition logique ($x_1 \in [1, 4]$) et ($x_2 \in [1, 8]$) - est vérifié.

L'analyse de franchissabilité des transitions de sortie nous permet d'observer que seulement 3 transitions parmi les 5 sont franchissables. Ces transitions sont :

- celle qui mène vers le sommet interdit ;
- celle étiquetée par l'événement contrôlable F_{OFF} ;
- celle étiquetée par la condition logique ($x_1 \geq 3$) et ($x_2 < 7$).

Parmi les trois transitions de sortie deux sont validées : celle étiquetée par F_{OFF} et celle étiquetée par la condition logique ($x_1 \geq 3$) et ($x_2 < 7$). La première est validée dès que le système a atteint le sommet et la deuxième au bout de 2 pas discrets. L'intérêt est d'avoir des résultats de l'analyse pertinents pour la synthèse de la commande, donc nous allons franchir la transition vers le sommet (A_2, B_2) en prenant aussi en compte le non-déterminisme introduit par la présence de la transition contrôlable.

Pas 4 : *Franchissement de la transition* $T_{(A_0, B_0),(A_2, B_2)}$

Actualisation de la pile $trace_{(A_0, B_0)}$ *associée au sommet* (A_0, B_0)

Mémoriser dans la pile $trace_{(A_0, B_0)}$:

- l'indice correspondant à la visite courante : $indice = 1$;
- la condition initiale à l'entrée du sommet (A_0, B_0) est représentée par la région $R_0^{(A_0, B_0)} = [1, 2] \times [1, 7]$;
- l'évolution complète $C_{(A_0, B_0)}(1)$ obtenue par calcul numérique ;
- le temps de séjour dans le sommet (A_0, B_0) avant l'évolution vers (A_2, B_2) est $\delta_{(A_0, B_0)}(1) = 2$ pas discrets ;
- actualiser $E(1) := \{T_0, T_{(A_0, B_0),(A_2, B_2)}\}$ la liste des transitions franchies.

Changement de sommet dans l'automate

- Le sommet courant est (A_2, B_2) ;
- La condition initiale à l'entrée du sommet est la région
  $R_A^{(A_0, B_0)} = \{(2.318, 1.444), (2.989, 1.740), (2.989, 6.653), (2.318, 6.356)\}$
- Le sommet (A_2, B_2) n'est pas le sommet initial et le $compteur = 1 < 9$, alors on incrémente le nombre des sommets analysés $compteur := compteur + 1 = 2$;
- Il s'agit d'une première visite dans le sommet (A_2, B_2) (car $indice = 1$), donc créer la pile $trace_{(A_2, B_2)}$
- La condition initiale à l'entrée du sommet est $R_0^{(A_2, B_2)} := R_A^{(A_0, B_0)}$;
- Il y a 3 transitions de sortie ; *Aller au Pas 2.1.*

Analyse d'atteignabilité dans le sommet (A_2, B_2)

Pas 2 : Étant la première visite dans le sommet (A_2, B_2), commencer l'analyse de franchissabilité des transitions de sortie.

Pas 3 :*Analyse de la franchissabilité des transitions de sortie du sommet* (A_2, B_2)

Parmi les transitions de sortie 2 sont franchissables pendant le séjour du système dans le sommet. La transition étiquetée par l'événement contrôlable F_{OFF} est validée dès que le système a atteint le sommet (A_2, B_2), tandis que l'autre qui mène vers le sommet interdit est validée au bout de 3 pas discrets.

Pas 4 : *Franchissement de la transition* $T_{(A_2, B_2),F}$

Actualisation de la pile $trace_{(A_2, B_2)}$ *associée au sommet* (A_2, B_2)

Mémoriser dans la pile $trace_{(A_2, B_2)}$:

- l'indice correspondant à la visite courante : $indice = 1$;
- la condition initiale à l'entrée du sommet (A_2, B_2) est représentée par la région

$R_0^{(A_2,\,B_2)}$;
– l'évolution complète $C_{(A_2,\,B_2)}(1)$ obtenue par calcul numérique ;
– le temps de séjour dans le sommet $(A_2,\,B_2)$ avant l'évolution vers F est $\delta_{(A_2,\,B_2)}(1) = 2$ pas discrets ;
– actualiser $E(1) := \{T_0,\, T_{(A_2,\,B_2),F}\}$ la liste des transitions franchies.

Changement de sommet dans l'automate

- Le sommet courant est F ;
- La condition initiale à l'entrée du sommet est la région
  $R_A^{(A_2,\,B_2)} = \{(3.602, 2.560),\, (4.052, 2.960),\, (4.052, 6.599),\, (3.602, 6.200)\}$
- Le sommet F est le sommet interdit et une fois atteint, le système y restera indéfiniment ;
- Il y a une transition contrôlable qui est validée avant que $T_{(A_2,\,B_2),F}$ devienne franchissable ; il s'agit de la transition $T_{(A_2,\,B_2),(A_0',\,B_0')}$

Pas 4 : *Franchissement de la transition* $T_{(A_2,\,B_2),(A_0',\,B_0')}$

Actualisation de la pile $trace_{(A_2,\,B_2)}$ *associée au sommet* $(A_2,\, B_2)$

Mémoriser dans la pile $trace_{(A_2,\,B_2)}$:
– l'indice correspondant à la visite courante : $indice = 1$;
– la condition initiale à l'entrée du sommet $(A_2,\, B_2)$ est représentée par la région $R_0^{(A_2,\,B_2)}$;
– l'évolution complète $C_{(A_2,\,B_2)}(1)$ obtenue par calcul numérique ;
– le temps de séjour dans le sommet $(A_2,\, B_2)$ avant l'évolution vers F est $\delta_{(A_2,\,B_2)}(1) \in [0,\, 2]$ pas discrets ;
– actualiser $E(1) := \{T_0,\, T_{(A_2,\,B_2),F},\, T_{(A_2,\,B_2),(A_0',\,B_0')}\}$ la liste des transitions franchies.

Changement de sommet dans l'automate

- Le sommet courant est $(A_0',\, B_0')$;
- Les conditions initiales à l'entrée du sommet dépendent de l'instant de commutation. Ainsi,
  1.) si la transition $T_{(A_2,\,B_2),(A_0',\,B_0')}$ est franchie à l'instant 0
  $R_A^{(A_2,\,B_2)}(0) = \{(2.318, 1.444),\, (2.989, 1.740),\, (2.989, 6.653),\, (2.318, 6.356)\}$;
  2.) si la transition $T_{(A_2,\,B_2),(A_0',\,B_0')}$ est franchie à l'instant 1
  $R_A^{(A_2,\,B_2)}(1) = \{(2.804, 1.796),\, (3.353, 2.180),\, (3.353, 6.625),\, (2.804, 6.241)\}$;
  3.) si la transition $T_{(A_2,\,B_2),(A_0',\,B_0')}$ est franchie à l'instant 2
  $R_A^{(A_0',\,B_0')}(2) = \{(3.202, 2.199),\, (3.652, 2.641),\, (3.652, 6.663),\, (3.202, 6.221)\}$.
- Le sommet $(A_0',\, B_0')$ n'est pas le sommet initial et le $compteur = 2 < 9$, alors on incrémente le nombre des sommets analysés $compteur := compteur + 1 = 3$;
- Il s'agit d'une première visite dans le sommet $(A_0',\, B_0')$ (car $indice = 1$), donc créer la pile $trace_{(A_0',\,B_0')}$
- La condition initiale à l'entrée du sommet est $R_0^{(A_0',\,B_0')} := R_A^{(A_2,\,B_2)(i)}$, où $i = 0,\, 1,\, 2$ est fonction de l'instant de commutation ;
- Il y a 5 transitions de sortie ; *Aller au Pas 2.1.*

Analyse d'atteignabilité dans le sommet $(A_0',\, B_0')$

Pas 2 : Étant la première visite dans le sommet $(A_0',\, B_0')$, commencer l'analyse de franchissabilité des transitions de sortie.

Pas 3 :*Analyse de la franchissabilité des transitions de sortie du sommet* $(A_0',\, B_0')$

Dans ce sommet le système peut séjourner $y_1 = 4$ unités de temps. En fonction de

l'instant de commutation il y a plusieurs possibilités :

1. Si le changement de sommet (franchissement de la transition $T_{(A_2,\,B_2),(A'_0,\,B'_0)}$) a lieu à l'instant 0, seulement la transition étiquetée par l'événement incontrôlable $y_1 = 4/F_{ON}$ est franchissable.
2. Si le changement de sommet (franchissement de la transition $T_{(A_2,\,B_2),(A'_0,\,B'_0)}$) a lieu à l'instant 1 ou 2, alors 2 transitions de sortie sont franchissables. La transition étiquetée par l'événement incontrôlable $y_1 = 4/F_{ON}$ et la transition étiquetée par la condition logique ($x_1 < 3$ et $x_2 < 7$).

Pas 4 : *Franchissement de la transition*

4.1. Commutation à l'instant 0 **de** $(A_2,\, B_2)$: Franchissement de la transition $T_{(A'_0,\,B'_0),(A_0,\,B_0)}$

Actualisation de la pile $trace_{(A'_0,\,B'_0)}$ *associée au sommet* $(A'_0,\, B'_0)$

Mémoriser dans la pile $trace_{(A'_0,\,B'_0)}$:

– l'indice correspondant à la visite courante : $indice = 1$;
– la condition initiale à l'entrée du sommet $(A'_0,\, B'_0)$ est représentée par la région $R_0^{(A'_0,\,B'_0)}$;
– l'évolution complète $C_{(A'_0,\,B'_0)}(1)$ obtenue par calcul numérique ;
– le temps de séjour dans le sommet $(A'_0,\, B'_0)$ avant l'évolution vers $(A_0,\, B_0)$ est $\delta_{(A'_0,\,B'_0)}(1) = 4$ pas discrets ;
– actualiser $E(1) := \{T_0,\, T_{(A_2,\,B_2),F},\, T_{(A_2,\,B_2),(A'_0,\,B'_0)},\, T_{(A'_0,\,B'_0),(A_0,\,B_0)}\}$ la liste des transitions franchies.

Changement de sommet dans l'automate

- Le sommet courant est $(A_0,\, B_0)$;
- La condition initiale à l'entrée du sommet est
  $R_A^{(A'_0,\,B'_0)} = \{(1.041, 1.992),\, (1.343, 2.488),\, (1.343, 5.780),\, (1.041, 5.285)\}$;
- Le sommet $(A_0,\, B_0)$ est le sommet initial. Alors, il faut remettre à zéro le *compteur* $:= 0$ et $indice := indice + 1 = 2$;
- Il s'agit d'une deuxième visite dans le sommet $(A_0,\, B_0)$ (car $indice = 2$). Donc, on compare la condition initiale à l'entrée de ce sommet mémorisé pour la visite précédente, mémorisée en $trace_{(A_0,\,B_0)}$ avec la condition initiale lors de cette visite $trace_{(A'_0,\,B'_0)}$.
- La condition initiale à l'entrée du sommet pour la visite courante est $R_0^{(A_0,\,B_0)} := R_A^{(A'_0,\,B'_0)}$. On la compare avec la région initiale de la visite précédente
  $$R_0^{(A'_0,\,B'_0)} = \{(1, 1),\, (2, 1),\, (2, 7),\, (1, 7)\}.$$
  La précision considérée est $\varepsilon = 10^{-3}$. En calculant l'écart on observe qu'on est loin de la précision désirée, donc il n'y a pas convergence.
- Il y a 5 transitions de sortie ; *Aller au Pas 2.1* pour continuer l'analyse.

4.2. Commutation à l'instant 1 **ou** 2 **:** Franchissement de la transition $T_{(A'_0,\,B'_0),(A'_2,\,B'_2)}$

Actualisation de la pile $trace_{(A'_0,\,B'_0)}$ *associée au sommet* $(A'_0,\, B'_0)$

Mémoriser dans la pile $trace_{(A'_0,\,B'_0)}$:

– l'indice correspondant à la visite courante : $indice = 1$;
– les conditions initiales à l'entrée du sommet $(A'_0,\, B'_0)$, sont représentées par les régions
  $R_A^{(A_2,\,B_2)}(1) = \{(2.804, 1.796),\, (3.353, 2.180),\, (3.353, 6.625),\, (2.804, 6.241)\}$;
  $R_A^{(A'_0,\,B'_0)}(2) = \{(3.202, 2.199),\, (3.652, 2.641),\, (3.652013, 6.663),\, (3.202, 6.221)\}$.
– l'évolution complète $C_{(A'_0,\,B'_0)}(1)$ obtenue par calcul numérique pour les deux conditions initiales ;
– le temps de séjour dans le sommet $(A'_0,\, B'_0)$ avant l'évolution vers F est

$\delta_{(A'_0,\,B'_0)}(1) = 0$ pour le cas où la transition $T_{(A_2,\,B_2),(A'_0,\,B'_0)}$ est franchie à l'instant 1 et $\delta_{(A'_0,\,B'_0)}(1) = 1$ pour le cas où la transition $T_{(A_2,\,B_2),(A'_0,\,B'_0)}$ est franchie à l'instant 2 pas discrets ;

– actualiser $E(1) := \{T_0,\, T_{(A_2,\,B_2),F},\, T_{(A_2,\,B_2),(A'_0,\,B'_0)},\, T_{(A'_0,\,B'_0),(A_0,\,B_0)},\, T_{(A'_0,\,B'_0),(A'_2,\,B'_2)}\}$ la liste des transitions franchies.

Changement de sommet dans l'automate

- Le sommet courant est $(A'_2,\, B'_2)$;
- Le sommet $(A'_2,\, B'_2)$ n'est pas le sommet initial et le *compteur* $= 3 < 9$, alors on incrémente le nombre des sommets analysés *compteur* := *compteur* $+ 1 = 4$;
- Il s'agit d'une première visite dans le sommet $(A'_2,\, B'_2)$ (car *indice* $= 1$), donc créer la pile $trace_{(A'_2,\,B'_2)}$
- Les conditions initiales à l'entrée du sommet dépendent de l'instant de commutation de $(A_2,\, B_2)$ vers $(A'_0,\, B'_0)$. Ainsi, $R_0^{(A'_2,\,B'_2)}(i) := R_A^{(A'_0,\,B'_0)}(i)$, où $i = 1,\, 2$;
- Il y a 3 transitions de sortie ; *Aller au Pas 2.1.*

Analyse d'atteignabilité dans le sommet $(A'_2,\, B'_2)$

Pas 2 : Étant la première visite dans le sommet $(A'_2,\, B'_2)$, commencer l'analyse de franchissabilité des transitions de sortie.

Pas 3 :*Analyse de la franchissabilité des transitions de sortie du sommet* $(A'_2,\, B'_2)$

Dans ce sommet le système peut séjourner $y_1 = 4$ unités de temps. Parmi les 3 transitions de sortie seulement celle étiquetée par la condition $y_1 = 4/F_{ON}$ peut être franchie.

Pas 4 : *Franchissement de la transition*

Franchissement de la transition $T_{(A'_2,\,B'_2),(A_0,\,B_0)}$

Actualisation de la pile $trace_{(A'_2,\,B'_2)}$ *associée au sommet* $(A'_2,\, B'_2)$

Mémoriser dans la pile $trace_{(A'_2,\,B'_2)}$:

– l'indice correspondant à la visite courante : *indice* $= 1$;

– il y a deux conditions initiales à l'entrée du sommet $(A'_2,\, B'_2)$; elles sont représentées par la région $R_0^{(A'_2,\,B'_2)}(i)$, où $i = 1,\, 2$;

– l'évolution complète $C_{(A'_2,\,B'_2)}(1)$ obtenue par calcul numérique ;

– le temps de séjour dans le sommet $(A'_2,\, B'_2)$ avant l'évolution vers $(A_0,\, B_0)$ est $\delta_{(A'_0,\,B'_0)}(1) = 4$, respectivement $\delta_{(A'_0,\,B'_0)}(1) = 3$ pas discrets en fonction de la région initiale à l'entrée ;

– actualiser $E(1) := \{T_0,\, T_{(A_2,\,B_2),F},\, T_{(A_2,\,B_2),(A'_0,\,B'_0)},\, T_{(A'_0,\,B'_0),(A_0,\,B_0)},\, T_{(A'_0,\,B'_0),(A'_2,\,B'_2)},\, T_{(A'_2,\,B'_2),(A_0,\,B_0)}\}$ la liste des transitions franchies.

Changement de sommet dans l'automate

- Le sommet courant est $(A_0,\, B_0)$;
- La condition initiale à l'entrée du sommet est soit $R_A^{(A'_0,\,B'_0)}(1) = \{(1.260, 2.443),\, (1.506, 2.943),\, (1.506, 5.923),\, (1.260, 5.423)\}$, soit $R_A^{(A'_0,\,B'_0)}(2) = \{(1.428, 2.896),\, (1.674, 3.365),\, (1.674, 6.061),\, (1.428, 5.592)\}$;
- Le sommet $(A_0,\, B_0)$ est le sommet initial. Alors, il faut remettre à zéro le *compteur* := 0 et *indice* := *indice* $+ 1 = 2$;
- Il s'agit d'une deuxième visite dans le sommet $(A_0,\, B_0)$ (car *indice* $= 2$). Donc, on compare la condition initiale à l'entrée de ce sommet mémorisé pour la visite précédente, mémorisée en $trace_{(A_0,\,B_0)}$ avec la condition initiale lors de cette visite $trace_{(A'_0,\,B'_0)}$.

- La condition initiale à l'entrée du sommet pour la visite courante est $R_0^{(A_0,B_0)} := R_A^{(A'_0,B'_0)}$. On la compare avec la région initiale de la visite précédente $R_0^{(A'_0,B'_0)} = \{(1,1),(2,1),(2,7),(1,7)\}$. La précision est $\varepsilon = 10^{-3}$. En calculant l'écart des régions, on observe il n'y a pas convergence.
- Il y a 5 transitions de sortie ; *Aller au Pas 2.1* pour continuer l'analyse.

Pour une précision $\varepsilon = 10^{-3}$, la convergence de l'algorithme est obtenu au bout de 5 itérations. Le modèle automate atteignable est illustré dans la Figure B.3

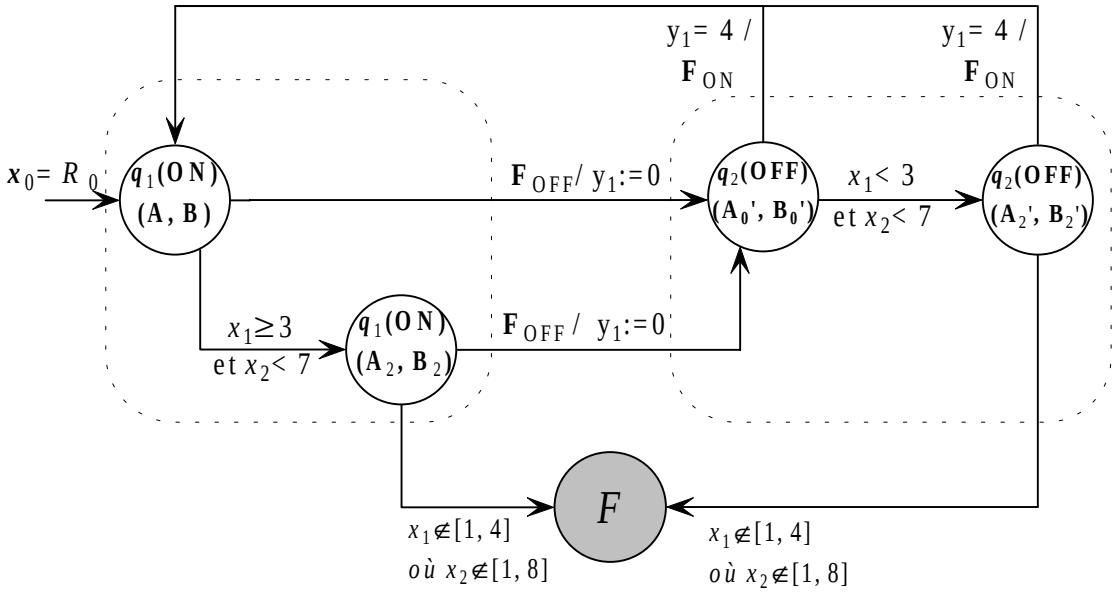


FIG. B.3 – Automate atteignable pour la ligne de production

D'une manière synthétique, les résultats complets de l'analyse d'atteignabilité peuvent être exprimés par un modèle temporisé comme illustre la Figure B.4.

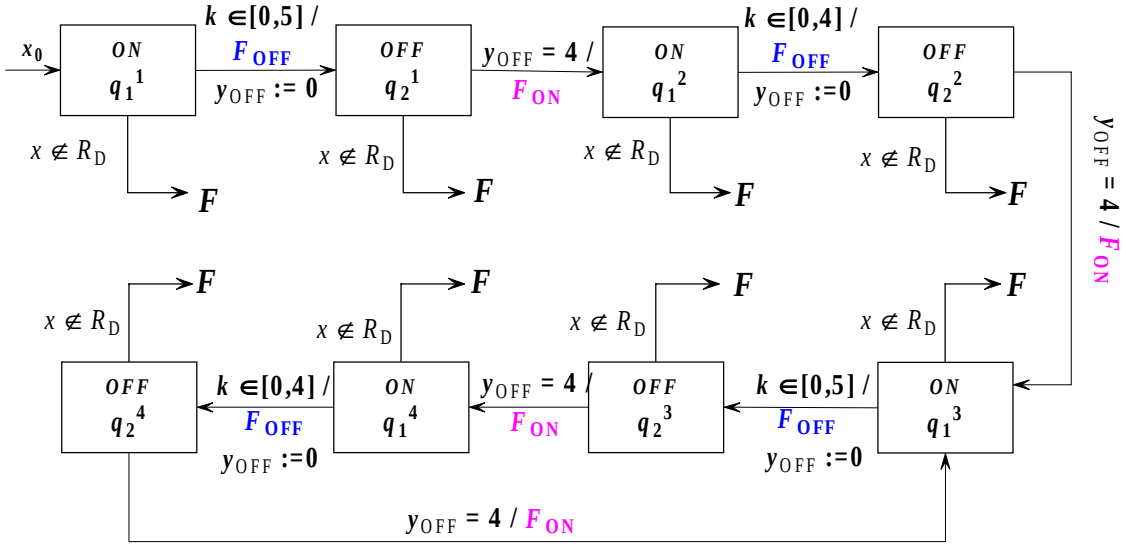


FIG. B.4 – Résultats synthétiques de l'analyse d'atteignabilité

Dans ce modèle, nous avons une représentation plus précise de l'évolution du système par la représentation explicite des temps de séjour du système dans chaque sommet de

l'automate.

Le modèle déplié temporellement est construit à partir du modèle automate atteignable, en exploitant les résultats de l'analyse représentés synthétiquement dans la figure précédente et en utilisant les informations stockées dans les vecteurs *trace* associés au sommets de l'automate hybride. Étant donné la complexité du modèle déplié en terme de nombre de sommets, il ne sera pas présenté ici.

***Résumé* :** Le travail présenté dans ce mémoire propose une méthodologie de synthèse de la commande pour des systèmes hybrides, qui permet de calculer l'ensemble de toutes les lois de commande telles que le fonctionnement du système respecte les spécifications imposées par le cahier des charges.

Notre approche consiste à représenter la dynamique continue par un système linéaire discrétisé et la dynamique événementielle par un automate à états finis. L'ensemble donne un automate hybride sur lequel les techniques d'analyse d'atteignabilité sont appliquées. Ces techniques permettent d'obtenir l'*automate atteignable*, qui ne contient que les trajectoires possibles du système pour une condition initiale donnée. En quelque sorte, nous avons ici une généralisation de la méthode *clock translation*.

L'utilisation du *temps discrétisé* permet d'obtenir un automate à états finis modélisant le système hybride. Ce modèle est obtenu par le *dépliage temporel* de la dynamique continue du système dans chaque sommet de l'automate hybride. La technique est similaire avec celle proposée par Brandin et Wonham pour les systèmes temporisés. Par ce modèle les trajectoires du système hybride seront explicitement représentées.

L'approche de synthèse de la commande présentée dans ce mémoire est basée sur une *extension* de la *théorie* classique *de la commande supervisée*. Le modèle de commande synthétisé est représenté par un automate temporisé. Celui-ci indique les dates d'occurrence auxquelles les événements contrôlables intervenant dans le fonctionnement du système doivent être exécutés. On notera que l'on s'affranchit ici de l'aspect hybride du système. Les résultats de la synthèse sont optimaux.

Les résultats de recherche de ce travail peuvent s'appliquer aussi bien au pilotage des systèmes de production qu'au contrôle des flux dans un procédé batch.

***Mots clés* :** Automate hybride**,** Analyse, Synthèse, Commande supervisée, Systèmes de production.

Study of Dynamical Hybrid Systems using discrete-time representation and hybrid automaton

Abstract: This thesis presents a control synthesis approach for hybrid systems. This approach proposes a methodology for computing the set of all control laws, which guarantees that the considered system respects the imposed specifications.

In our approach, the *discrete-time representation* is used to model the continuous dynamics of the system and finite state machine models the discrete evolutions. These two models combined give the hybrid automaton on which the reachability techniques are applied. These techniques let us to obtain the *reachable automaton*, which contains only the possible trajectories of the system for a given initialization. We have here a generalization of the *clock translation* method.

The use of the discrete-time representation for the continuous dynamics let to obtain a finite state machine, modeling the hybrid system. This model is obtained by the *timed unfolding* of the continuous dynamics in each location of the hybrid automaton. The unfolding procedure is similar with the technique developed by Brandin and Wonham for the timed discrete event systems. The unfolded model represents explicitly the complete evolution of the hybrid system.

The proposed *control synthesis* approach is based on the *extension* of the *supervisory control theory* developed for the discrete event systems. The obtained *control model* is given as a timed automaton. This model state the dates at which the controllable events of the system must occur in order to assure the respect of the specifications. We note that, at the control level, the hybrid aspects of the system evolution disappear. The synthesis results are optimal.

The results presented in this thesis can be applied for the control of manufacturing systems but also for the flow control in the batch processes.

***Keywords*:** Hybrid automaton**,** Analysis, Control, Supervised control, Manufacturing systems.