Université Abou Bekr Belkaid
Tlemcen Algérie
جامعة أبي بكر بلقايد
تلمسان الجزائر

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITE ABOU BEKR BELKAID TLEMCEN FACULTE DE TECHNOLOGIE
DEPARTEMENT DE GENIE ELECTRIQUE ET ELECTONIQUE

Mémoire

Pour l'obtention du diplôme de

Master en Automatique

Option : *Automatisation et Supervision*

Réalisée par :

❖ ***M^{r} BOUKERMA Ali Seif Eddine et M^{r} ZEHRI Oussama***

Soutenu en octobre 2013 devant le jury :

Mme.M.BENALLEL	*Présidente*
Mr.A.GUEZZEN	*Examinateur*
Mr. A.HASSAM	*Examinateur*
Mme.L.SARI	*Examinatrice*

Encadrée par : Melle L.GHOMRI

Remerciement

Toute histoire a une fin. Celle de cette thèse ne déroge pas à la règle. Il est de coutume dans un mémoire de thèse de débuter par la fin : voici donc venu le temps des remerciements.

Nous tenons tout d'abord à remercier Dieu le tout puissant et miséricordieux, qui nous a donné la force et la patience d'accomplir ce Modeste travail.

En second lieu, nous tenons à remercier notre encadreur Mademoiselle Ghomri Latefa, son précieux conseil et son aide durant toute la période du travail.

Nos vifs remerciements vont également aux membres du jury Madame Benallel Mounira, Madame Sari Lamia, Monsieur Guezzen Amine Hakim, Monsieur Hassam Ahmed pour l'intérêt qu'ils ont porté à notre recherche en acceptant d'examiner notre travail Et de l'enrichir par leurs propositions.

Enfin, nous tenons également à remercier toutes les personnes qui ont participé de près ou de loin à la réalisation de ce travail.

Dédicaces

Toutes les lettres ne sauraient trouver les mots qu'il faut…

Tous les mots ne sauraient exprimer la gratitude, l'amour,

Le respect, la reconnaissance…

Aussi, c'est tout simplement que

Je dédie humblement ce mémoire à :

A celle qui m'a toujours ouvert ses bras et soutenue dans tout ce que j'ai entrepris ; celle qui a su être bonne, gentille et compréhensive avec moi ; celle dont je regrette l'absence à cette étape importante de ma vie ; celle qui me manque terriblement aujourd'hui ma très chère et adorée grand-mère (nana).

A MA TRÈS CHÈRE MÈRE :

Autant de phrases aussi expressives soient-elles ne sauraient montrer le degré d'amour et d'affection que j'éprouve pour toi. Tu m'as comblé avec ta tendresse et affection tout au long de mon parcours. Tu n'as cessé de me soutenir et de m'encourager durant toutes les années de mes études, tu as toujours été présente à mes côtés pour me consoler quand il fallait.

En ce jour mémorable, pour moi ainsi que pour toi, reçoit ce travail en signe de ma vive reconnaissance et mon profond estime. Puisse le tout puissant te donner santé, bonheur et longue vie afin que je puisse te combler à mon tour.

A MON TRÈS CHER PÈRE :

Autant de phrases et d'expressions aussi éloquentes soit-elles ne sauraient exprimer ma gratitude et ma reconnaissance. Tu as su m'inculquer le sens de la responsabilité, de l'optimisme et de la confiance en soi face aux difficultés de la vie. Tes conseils ont toujours guidé mes pas vers la réussite. Ta patience sans fin, ta compréhension et ton encouragement sont pour moi le soutien indispensable que tu as toujours su m'apporter.

Je te dois ce que je suis aujourd'hui et ce que je serai demain et je ferai toujours de mon mieux pour rester ta fierté et ne jamais te décevoir. Que Dieu le tout puissant te préserve, t'accorde santé, bonheur, quiétude de l'esprit et te protège de tout mal.

A MA TRÈS CHERE SŒUR ET SA PETITE FAMILLE

Pour toute la complicité et l'entente qui nous unissent, ce travail est un témoignage de mon attachement et de mon amour.

A MES CHÈRES FRÈRES BRAHIM ET IMAD

Pour toute l'ambiance dont vous m'avez entouré, pour toute la spontanéité et votre élan chaleureux, Je vous dédie ce travail. Puisse Dieu le tout puissant exhausser tous vos vœux.

A MA CHÈRE PETITE SŒUR NESSRINE

Une fille adorable, Une vraie sœur que tout le monde aimerait avoir! , Je te dédie ce travail.

A MA GRANDE FAMILLE :

Je cite en particulier grand-mère, mes tantes, mes oncles ainsi que mes cousins et cousines.

A mes très chers amis Oussama Zehri, Mohamed Henni et Mohamed Belkhamgani

A ma très chère amie D.Sara

A MES CHÈRS AMIS :

Nassredine Mhedji, Mustapha Belhadj, Fouad Abed, Djamel Bendahnoun, Youcef Bendahnoun, Khelladi Sofiane, Hocine Abdellah Benkhatou, Abdou Bouiche, Blell Abdelhak, Miloud Maaziz, Kader Negadi, Zaki Aoued, Adel Belgacem….

A MES CHÈRES AMIES :

T. Fatima, T.Marwa, B.Hassiba, O.Sara, B.Fayrouz, B.Rabab …

A tous ceux dont l'oubli du nom n'est pas celui du cœur.

A tous ceux qui ont contribué de près ou de loin à l'élaboration de ce travail.

BOUKERMA Ali Sief Eddine

Je dédie humblement ce Travail à :

A celle qui s'est toujours dévouée et sacrifiée pour moi ; celle qui m'a aidée du mieux qu'elle pouvait pour réussir ; celle qui m'a accompagnée tout au long de ce parcours périlleux ; celle qui a toujours été là dans mes moments de détresse, ma très chère mère.

A celui qui m'a toujours encouragée et soutenue moralement, mon très cher père.

A celles qui m'ont toujours aidée, écoutée, soutenue et encouragée tout au long de mon parcours ; celles qui ont toujours été présentes pour moi, mes très chère tantes
Zineb et Khaira.

A mon très cher frère qui m'a énormément aidé et à qui je témoigne mon affection et ma profonde reconnaissance.

A mes très chers amis B.Ali Sief Eddine, B.Youcef, K.Sofiane B. Djamel, B.Moustafa, B.Djalal, Z.Omar Youness, A.Foued.M.Hamada, K.Kichem.

A mes très chères amies B.Kassiba, O.Sara, G.Souhila, B, Fayrouz, F.Imen, Farah

ZEKRI Oussama

Table des matières

Introduction Général.....01
1. **Chapitre 1**: Diagnostic des systèmes à évènements discrets 03
1.1. Introduction 03
1.2. Diagnostic des systèmes à évènements discrets..... .03
1.2.1. Introduction aux SED .... .03
1.2.2. Etude de la diagnosticabilité 05
1.2.3. Diagnosticabilité des SED à aspect temporel 05
1.3. Diagnostic des systèmes temporisés 06
1.3.1. Les systèmes temporisés 06
1.3.1.1 Les automates temporisés 06
1.4. Structure et synthèse du diagnostiqueur 08
1.4.1 Synthèse du diagnostiqueur 11
1.4.1.1. Prérequis 11
1.4.2. Architecture de l'algorithme de synthèse du diagnostiqueur 12
1.5 Conditions de diagnosticabilité 18
1.5.1 Diagnosticabilité..... 19
1.6. Conclusion..... 21
2**. Chapitre 2 :** Synthèse des controleurs des SED et SEDT 22
2.1. Synthèse des controleurs des SED 22
2.1.1. Introduction ..... .22
2.1.2. Théorie de la supervision 23
2.1.2.1 Présentation du travail de Ramadge et Woham 23
2.1.2.2 Shéma de la supervision 24
2.1.2.3 Définition d'un superviseur 26
2.1.3. Controlabilité 29
2.1.3.1 Propriétés de la controlabilité 30
2.1.3.2 La condition de la controlabilité 31
2.1.4. Synthèse du controleur 31
2.1.4.1 Algorithme de Kumar 31
2.2. Contrôle des systèmes à évènements discrets temporisés 33
2.2.1. Introduction sur les SEDT 33

2.2.2.Synthèse de controleur des systèmes à évènements discret(SEDT)..... 34
2.2.2.1 Synthèse de controleur en temps discret 35
2.2.2.2 Synthèse de controleur en temps continu 39
2.3. Conclusion 42
3. **Chapitre 3** : Des applications sur les SED
3.1 Introduction.....43
3.2 Présentation du logiciel PHAVer.....43
3.3 Instalation.....45
3.4 Outils annexes.....45
3.5 Commande pour description et simulation 46
3.6 Modélisation et simulation.....46
3.6.1 Application n°1.....46
3.6.2 Application n°2.....52
3.6.3 Application n°3.....55
3.7 Conclusion.....58
Conclusion générale 59
La bibliographie 60

Liste Des Figures

Figure 1.1 chronogramme de l'évolution de l'état d'une machine……..04
Figure 1.2 Exemple d'un système à événement discret et de son générateur…..….07
Figure1.3 Modèle du système de chauffage de liquides vérifiant toutes les hypothèses de modélisation…….....10
Figure 1.4 Exemple du diagnostiqueur d'un système de chauffage de liquides…….11
Figure 1.5 États d'entrée et États d'ombre……...12
Figure 1.6 Principe de construction d'un sommet du diagnostiqueur…….….13
Figure 1.7 Création d'une partition de trois ensembles d'états franchissables…..……15
Figure 1.8 Exemple de construction d'un diagnostiqueur : modèle initial……...16
Figure 1.9 Exemple de construction d'un diagnostiqueur : zones disjointes des états franchissables……..17
Figure1.10 Exemple de construction d'un diagnostiqueur : modèle du diagnostiqueur…...17
Figure 1.11 Schéma d'un circuit électrique simple ……...20
Figure 1.12 Modèles du circuit électrique ……20
Figure 2.1 Schéma de supervision ……23
Figure 2.2 Le concept de commande par supervision……..…….25
Figure 2.3 Système manufacturier…….26
Figure 2.4 Automate à états finis modélisant le système manufacturier……....27
Figure 2.5 Composition synchrone des deux modèles de machines (modèle du procédé)…28
Figure 2.6 Le système manufacturier sous contrainte de stock……28
Figure 2.7 Le modèle automate de la spécification……29
Figure 2.8 Langage suprême contrôlable d'un fonctionnement désiré……..31
Figure 2.9 Modèle automate du système supervisé avec des états interdits ……...32
Figure 2.10- Modèle final de système supervisé sans états interdits ……33
Figure 2.11 Evolution de l'état de la machine ……..34
Figure 2.12 Automate A_{act} ……...36
Figure 2.13. Automate A ……...……....37
Figure 2.14 Automate temporisé……..40
Figure 2.15.- Première type d'incohérence temporelle……41
Figure 2.16 Deuxième type d'incohérence temporelle……41
Figure 3.1 convoyeur ……..……47
Figure 3.2 L'automate à chronomètres pour le système du convoyeur……47
Figure 3.3 Les espaces d'états acceptés pour le convoyeur : graphe x en fonction de t pour les états Marche et Arrêt……..….52
Figure 3.4 L'automate à chronomètres pour le système du Robot…….…..52
Figure 3.5 Schéma d'un convoyeur dans un système de stockage industriel simple……..56
Figure 3.6 Représentation d'Automate A ...……..……..56

Liste des tableaux

Tab. 1.1 – Description des symboles utilisés dans modèle du diagnostiqueur
Tab. 3.1 – Convoyeur – résultat de simulation en PHAVer
Tab. 3.2 – robot résultat de simulation en PHAVer
Tab .3.3 – Convoyeur - résultat de simulation en PHAVer

Introduction Général

L'automatisation des installations industrielles vise à augmenter la productivité des systèmes et à réduire les coûts de la maintenance des équipements de production.

Cependant, la complexité due à l'automatisation des systèmes de production implique des besoins croissants en termes de disponibilité et de performance. Ainsi, il est nécessaire de disposer d'une fonction permettant le diagnostic des défaillances pouvant affecter le fonctionnement du système.

Un module de diagnostic est nécessaire, non seulement pour améliorer les performances et la productivité des systèmes, mais également pour limiter les conséquences des pannes qui peuvent être catastrophiques sur le plan des biens et des vies humaines.

Plusieurs chercheurs ont abordé la thématique de la surveillance industrielle et du diagnostic.

Le principe de diagnostic repose sur la comparaison du comportement prévu par le modèle avec le comportement réellement observé du système. Tout écart entre ces deux comportements sera synonyme de défaillance. L'utilisation d'un modèle du système pour son diagnostic nécessite souvent sa conformité au critère de diagnosticabilité **([31] et [06]).**

Il s'agit de vérifier si chaque défaillance peut être détectée et isolée dans un temps fini après l'occurrence du défaut source de la défaillance. En d'autres termes, ce critère consiste à déterminer si le modèle du système est suffisamment riche en information pour permettre l'identification de tous les défauts affectant son fonctionnement.

De nombreuses approches de diagnostic à base de modèles ont été proposées dans la littérature. On retrouve les approches issues de la communauté des Systèmes à Événements Discrets (SED) **([31]; [06]; [30] et [05]).**

Les systèmes à événements discrets sont des systèmes dynamiques fondamentalement asynchrones pour lesquels l'espace d'états est discret. Leur évolution se fait conformément à l'arrivée des événements caractérisant le changement d'état du système. Au lieu de s'intéresser au déroulement continu des phénomènes, les systèmes à événements discrets ne se soucient que des débuts et des fins de ces phénomènes (les événements discrets) et de leur enchaînement logique, dynamique ou temporel. Leurs domaines principaux d'application incluent, parmi d'autres, la production manufacturière, la robotique, la circulation des véhicules, la logistique, les réseaux de communication et l'informatique.

Il y a maintenant une multitude d'outils permettant l'étude des systèmes à événements discrets, tels que la simulation sur ordinateur, les réseaux de files d'attente, les langages de programmation parallèle / temps réel, les modèles dynamiques algébriques, les chaînes de Markov, les automates et les réseaux de Petri. Nous allons rappeler dans ce mémoire les approches de modélisation et d'analyse des systèmes à événements discrets principalement impliquées dans la recherche sur le diagnostic des SED et la synthèse de contrôleurs, qui est le sujet de ce travail ; c'est-à-dire les automates.

Le mémoire que nous allons présenter ne comprend pas que le diagnostic mais elle aborde aussi le contrôle des systèmes à évènements discrets.

Depuis quelques années, des exigences d'une compétitivité sans cesse croissante ainsi que le progrès technologique ont engendré l'apparition des systèmes de production de plus en plus complexes. Compte tenu des enjeux économiques, on ne peut pas se permettre la synthèse d'une commande par des essais et des corrections successives. Ainsi, là ou dans le

passé le bon sens suffisait, il est devenu crucial de disposer des outils formels et des techniques pour l'analyse et la synthèse de la commande qui permettent de garantir, a priori, que le fonctionnement de système commandé respecte les spécifications imposés par le cahier des charges.

Ce sont Ramadge et Wonham **([10])** qui à l'origine ont introduit la théorie de la synthèse de contrôleurs sur les systèmes à événement discrets. Le système, ainsi que l'ensemble des comportements valides (*Spécification*) sont donnés chacun par un langage. A partir de ces deux langages, le but de la synthèse consiste alors à déterminer un sous ensemble des comportements du système qui appartienne aussi à la spécification. Ce sous ensemble représente les comportements du système subissant l'action du contrôleur et caractérise ce dernier.

Dans la théorie de la supervision, le système complet est généralement divisé en deux parties : le procédé et le superviseur. Le procédé, un SED, est la partie du système qui doit être supervisée. Le comportement du procédé sans le superviseur (boucle ouverte) est souvent jugé non satisfaisant à l'égard de ce qui est souhaité. Il est possible de modifier dans une certaine limite, le langage généré par le système, par l'ajout d'une structure particulière appelée superviseur. Le rôle du superviseur est d'assurer que les aspects non satisfaisants du comportement du procédé ne se produiront pas. Le comportement souhaité est précisé, en imposant que le langage généré par le procédé couplé au superviseur appartienne à un certain langage de spécification. En détectant chacune des évolutions du procédé, donc en connaissant l'état courant du procédé, le superviseur choisit l'action ''corrective'' appropriée. Cette action se traduit par des lois de contrôle transmises au procédé. Ces lois de contrôle contiennent les événements dont l'occurrence est autorisée depuis l'état courant du procédé. Le superviseur intervient sur l'évolution du procédé uniquement en interdisant l'occurrence de certains événements, c'est-à-dire en ne les faisant pas apparaître dans une (ou plusieurs) loi de contrôle. Une autre distinction essentielle entre le procédé et le superviseur est que le procédé est donné, il correspond à un système existant qui ne peut être changé, tandis que le superviseur est un système à construire, donc plus souple.

Notre objectif dans ce travail est d'étudier les SED en vue de leur diagnostic et leur commande. Pour cela il nous a paru nécessaire de commencer le premiers chapitre par une présentation générale des SED Puis nous présentons la notion de diagnosticabilité ainsi que la synthèse du diagnostiqueur.

Le chapitre deux présente dans un premier temps La théorie de la supervision des SED proposée par Ramadge & Wonham est d'abord présentée en détail. Elle est à l'origine de la commande par supervision. La notion de contrôlabilité. Puis la synthèse de contrôleur.

Nous présentons dans le troisième chapitre des applications sur les systèmes à évènements discret, nous avons utilisé le logiciel PHAVER.
C'est un logiciel qui permet le calcul de l'espace atteignable d'un SED. Cette étape est indispensable pour l'analyse de cette classe de systèmes. Et donc elle est indispensable dans les deux cas du diagnostic et de la synthèse de contrôleur.

Chapitre 1

Diagnostic des systèmes à évènements discrets

1. 1 Introduction

Les systèmes automatiques complexes trouvent leurs places aujourd'hui dans de plus en plus d'applications réelles de la vie quotidienne par exemple dans l'industrie automobile et aéronautique entre autres. L'une des exigences pour ces systèmes, est qu'elle soit le plus autonome que possible même dans des cas où de défaillances. Le problème de diagnostic automatique de défaillances dans des systèmes complexes est l'un des domaines de recherche qui ont attiré l'attention aussi bien de la communauté de contrôle de systèmes avec des approches issues de l'automatique que de la communauté de l'automatique des systèmes à événements discrets avec les travaux sur le diagnostic à base de modèles graphiques tels que les automates à états finis et les réseaux de Petri.

Le diagnostic est de nos jours, un élément clef pour accroitre la productivité et la disponibilité des systèmes complexes. La fonction de diagnostic consiste à détecter une défaillance, de localiser son origine et de déterminer ses causes. Son principe général consiste à confronter les données relevées au cours du fonctionnement réel du système avec la connaissance dont on dispose sur son fonctionnement normal et anormal, afin de détecter les incohérences et en déduire les causes de dysfonctionnement.

Notre travail est dans la continuité des travaux précédents. Il vise à associer les travaux de diagnostic à base de modèles temporisés des systèmes à événements discrets avec ceux de la commande.

1.2 Diagnostic des systèmes a événements discrets

1.2.1 Introduction aux SED

Les systèmes à événements discrets (SED) sont des systèmes dynamiques fondamentalement asynchrones pour lesquels l'espace d'états est discret. Leur évolution se fait conformément à l'arrivée des événements caractérisant le changement d'état du système.

Au lieu de s'intéresser au déroulement continu des phénomènes, les modèles SED ne se soucient que des débuts et des fins de ces phénomènes (les événements discrets) et de leur enchaînement logique, dynamique ou temporel.

Formellement, le changement d'état d'un SED peut être décrit par un couple (événement, instant d'occurrence). Un ensemble ordonné de tels couples s'appelle une *séquence* **[01]**. Leurs principaux domaines d'application incluent, parmi d'autres, la production manufacturière, la robotique, la circulation des véhicules, la logistique, les réseaux de communication et l'informatique.

La classe des SED a été largement étudiée dans la littérature **([09]; [06])**. Cet intérêt est justifié par l'existence d'un grand nombre de systèmes réels évoluant d'une manière discrète. Cassendras et Lafortune ont défini un SED dans **([26])** comme suit :

Définition 1.1: Un système à événements discrets est un système dont l'état évolue en fonction de l'occurrence d'événements asynchrones, sur une échelle de temps continue.

Exemple 1.1 : Considérons l'exemple suivant :

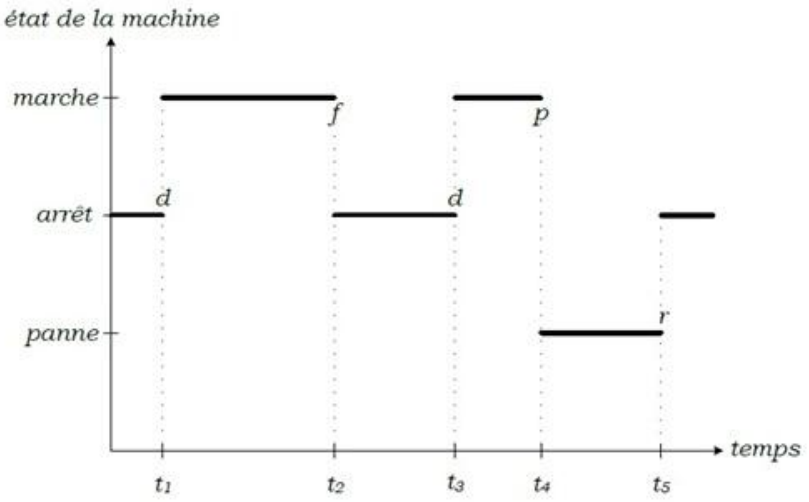


Figure 1.1. Chronogramme de l'évolution de l'état d'une machine

Dans l'état initial la machine est supposée être en arrêt. Au moment t_1 une pièce est détectée à l'entrée, l'événement *d* = début de cycle d'usinage se produit, et la machine évolue dans l'état de marche. Il y a deux évolutions possibles, à partir de cet état. Soit le cycle d'usinage est accompli sans erreur et la machine dépose la pièce usinée à la sortie simultanément à l'occurrence de l'événement *f* = fin de cycle d'usinage, soit la machine tombe en panne, fait signalé par l'occurrence de l'événement *p* = panne. Depuis l'état de panne, la réparation de la machine événement *r* – ramène celle-ci à son état initial. La pièce en cours de traitement est supposée perdue pendant la réparation. Les deux scénarios de fonctionnement décrits précédemment sont présentés dans la figure 1.1 La machine mène à terme le premier cycle d'usinage au moment t_2 (événement *f*), commence un nouveau cycle au moment t_3 (événement *d*), tombe en panne au moment t_4 (événement *p*) et est réparée au moment t_5 (événement *r*). La séquence décrivant ce fonctionnement est comme suit :

$$S = (d, t_1), (f, t_2), (d, t_3), (p, t_4), (r, t_5)$$

1.2.2 Etude de la diagnosticabilité

L'application d'une méthode de diagnostic est soumise à la condition de vérification du critère de diagnosticabilité. Ce critère stipule que le modèle du système dispose de suffisamment d'informations pour effectuer le diagnostic. En d'autres termes, il permet de répondre à l'interrogation suivante : est-il possible de déterminer toutes les occurrences des défauts affectant un système, étant données les informations fournies par son modèle et les observations qu'il génère ?

La définition de la diagnosticabilité pour les SED a été initialement formalisée dans les travaux de Lin et Wonham **([27])**. Plusieurs extensions de cette définition ont été ensuite proposées, dans de différents contextes **([06] et [28] et [29],)**. Les extensions proposées varient en fonction de l'abstraction du modèle utilisé (logique, temporisé, hybride), de la structure de la méthode de diagnostic (centralisée, décentralisée, distribuée), ...

Dans la suite, nous présentons la notion de diagnosticabilité dans le cadre des SED temporisés.

Nous nous intéressons aux travaux les plus importants pour la compréhension de notre travail.

1.2.3 Diagnosticabilité des SED à aspect temporel

Un automate temporisé est dit diagnosticable, s'il est possible de détecter tout défaut non observable, au bout d'un délai u.t. après son occurrence. Il faut souligner que cette définition traite uniquement le problème de détection dans le sens où elle considère l'existence d'un seul type de défauts. Une trace temporisée ! Dite défaillante, si elle contient un défaut et au moins u.t. se sont écoulés après l'occurrence de ce défaut. On définit l'opérateur de projection observable P qui permet de filtrer les événements non observables à partir d'une trace temporisée.

Formellement, la diagnosticabilité est définie comme suit **([30])** :

Un langage temporisé est diagnosticable, si toute paire de traces dans L, la première contient un défaut et la seconde est dépourvue de défauts, les projections observables de ces deux traces doivent être différentes au bout de u.t. de l'occurrence du défaut.

L'intégration du temps dans la définition de diagnosticabilité introduit deux nouvelles notions sur la définition classique de diagnosticabilité :

- dans l'approche classique, l'identification d'un défaut doit se faire au bout d'un délai fini. Ce délai est représenté qualitativement à travers le nombre d'événements qui suivent l'événement du défaut. Par contre, ce délai est défini quantitativement dans l'extension temporisée de cette définition à travers le délai.
- La discrimination des trajectoires de défauts prend en considération le temps d'une Manière explicite.

En effet, la définition classique stipule qu'un langage est diagnosticable s'il est possible de discriminer, d'un point de vue logique, les trajectoires comportant des défauts.

Cela nécessite l'observation d'un événement discriminant, au bout d'un délai fini, permettant de caractériser d'une manière unique une trajectoire défaillante du système.

Dans le contexte temporisé, la discrimination d'une trajectoire défaillante considère en plus, les dates d'occurrence des événements observables.

Ainsi, deux traces peuvent avoir la même projection observable d'un point de vue logique (séquence d'événements) mais pas d'un point de vue temporisé (dates des événements). Ainsi, il est possible de discriminer un comportement défaillant en s'appuyant sur les dates d'occurrence des correspondants événements observables. Dans ce cas, on parle de temps discriminant.

1.3 Diagnostic des systèmes temporisés

1.3.1 LES SYSTEMES TEMPORISE

1.3.1.1 Les Automates Temporisés : présentation et analyse

Définition 1.2: Un automate temporisé est un automate à états finis muni d'un ensemble de variables réelles positives appelées *horloges*. Les horloges sont incrémentées simultanément, avec une dynamique égale à 1 et peuvent être remises à zéro. Les horloges sont des variables fictives dans le sens oïl elles ne sont pas des variables du système, mais elles sont utilisées pour mesurer le temps et définir les contraintes temporelles sur le franchissement des transitions.

Une transition, entre deux sommets d'un automate temporisé, est franchie suite à l'occurrence d'un événement en entrée et la satisfaction d'une contrainte de franchissement, appelée *garde*. Ce franchissement est instantané et peut déterminer la mise à zéro d'un ensemble d'horloges. **[02]**

- *Notations de base* :

Un SEF *G* est défini par le quadruplet $G = (X, \Sigma, \delta, x0)$ tel que :

– X: ensemble d'états

– Σ: ensemble des événements (alphabet)

– δ: ensemble fini de transitions

– $x0$: état initial

L'ensemble d'événements Σ est répartis en deux sous-ensembles :

Σ_o et Σ_{uo} : $\Sigma = \Sigma_o \cup \Sigma_{uo}$.

Σ_o contient les événements observables i.e., les événements dont l'occurrence peut être observée et qui peuvent être les commandes issues du contrôleur ou les valeurs obtenues des capteurs lors de l'exécution du système.

Σ_{uo} contient les événements non observables i.e., les événements dont l'occurrence ne peut pas être observée et qui contiennent les événements de fautes et tout autre événement qui peut causer un changement dans le système mais dont les capteurs ne détectent pas la présence.

L'ensemble des fautes du système est noté Σ_f et représente un sous-ensemble des non observables : $\Sigma_f \subseteq \Sigma_{uo}$.

On partitionne l'ensemble de fautes Σ_f en m ensembles disjoints qui correspondent aux différents types de fautes dont le traitement est assuré par les mêmes actions consécutives.

❖ *Construction du générateur G'*

G' résulte de G en ne décrivant que le comportement du système par rapport aux événements observables.

G' est en général non déterministe et génère le langage P(L) qui résulte de l'application de la fonction de projection P sur toutes les trace de L i.e., G' exprime la partie observable su système modélisé.

Avant de montrer comment construire G' nous commençons par introduire les définitions suivantes :

$X_0 = \{x_0\} \cup \{x \in X$ / il y a une transition étiquetée par un observable et entrante à x$\}$

L (G, x) est l'ensemble des traces de L issues de l'état x de G.

Le générateur G' est défini par : $G' = (X_0, \Sigma_o, \delta_{G'}, x_0)$ où X_0, Σ_o, x_0 ont été déjà définis et où $\delta_{G'}$ est la fonction de transition de G' telle que $\delta_{G'} \subseteq (X_0 \times \Sigma_o \times X_0)$ avec:

$(x, \sigma, x') \in \delta_{G'}$ ssi $(x, s, x') \in \delta$ pour un certain $s \in L_\sigma (G, x)$.

Exemple 1.2 (inspiré de [03])

La figure 1.2 Montre le modèle d'un système à événements discrets G (la partie gauche) ainsi que son générateur G.

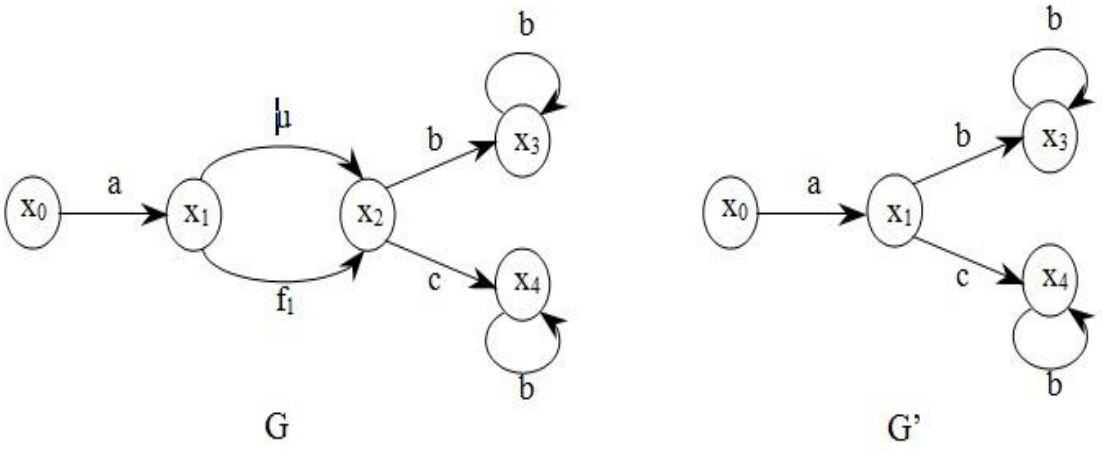


Figure 1.2. Exemple d'un système à événement discret et de son générateur

Le modèle initial G est défini par :

- l'ensemble des états est : X $\{x_0, x_1, x_2, x_3, x_4\}$;
- l'ensemble des événements est : Σ $\{a, b, c, u, f_1\}$;

Les sous-ensembles des événements observables, non observables et de fautes sont respectivement :

$\Sigma_o\{a, b, c,\}$ $\Sigma_{uo}\{u, f\}$ et $\Sigma_f\{f1\}$;

- l'ensemble de transitions est :

δ $\{(x_0, a, x_1), (x_1, f_1, x_2), (x_1, u, x_2), (x_2, b, x_3), (x_2, c, x_4), (x_4, b, x_4), (x_3, b, x_3)\}$;

- l'état initial est : x_0

Le générateur correspondant G' est défini par : $X_0, \Sigma_o, \delta_{G'}, x_0$

- l'ensemble des états est : X_0 $\{x_0, x_1, x_3, x_4\}$;
- l'ensemble des événements est: Σ_o {a, b, c} ;

- l'ensemble de transitions est :

δ_G $\{(x_0, a, x_1), (x_1, b, x_3), (x_1, c, x_4), (x_3, b, x_3), (x_4, b, x_4)\}$;

- l'état initial est : x_0

1.4 Structure et Synthèse du diagnostiqueur :

Nous présentons dans cette partie la structure du diagnostiqueur que nous désirons Construire à partir d'un automate temporisé vérifiant les hypothèses et les spécifications énumérées ci-dessus.

La structure de notre diagnostiqueur est inspirée du modèle du diagnostiqueur proposé dans les travaux de Sampath **[06].**

Notre diagnostiqueur est un automate temporisé déterministe, compilé à partir du modèle du système. Cet automate est exécuté en-ligne avec le système pour permettre l'élaboration du diagnostic. Une évolution d'un sommet à un autre dans cet automate se fait uniquement par le biais de transitions sur des événements observables. Par ailleurs, le diagnostiqueur peut évoluer vers un autre sommet suite à l'écoulement d'un délai seuil d'attente à travers des transitions urgentes **([04]).**

Les sommets du diagnostiqueur correspondent à des macro-états, pouvant contenir plusieurs ensembles d'états du système enrichis par des informations relatives aux défauts. En effet, après l'occurrence de chaque événement observable, le diagnostiqueur évolue vers le prochain sommet qui fournit une estimation de l'état du système, à l'instant du franchissement de cet événement. Par ailleurs, à chacun de ces états est associée une étiquette de diagnostic permettant d'indiquer si un défaut s'est produit dans l'exécution menant vers cet état.

Dans la suite, nous présentons une définition formelle de l'automate temporisé du diagnostiqueur. Nous notons par : Φ = {N, F1, F2, F3…..Fm}un ensemble d'étiquettes, dites étiquettes de diagnostic. Ces étiquettes permettent de déterminer le mode de défauts affectant un ensemble d'états du système. En effet, un ensemble d'états du système est qualifié de Fi-défaillant, s'il est associé à une étiquette Fi. Autrement dit, un défaut de l'ensemble Fi s'est produit avant d'atteindre un état de cet ensemble. De même, un ensemble d'états est qualifié de normal, s'il est associé à l'étiquette N. Dans ce dernier cas, aucun défaut ne s'est produit avant d'atteindre un état de cet ensemble.

Définition 1.3 : Un diagnostiqueur est un automate temporisé déterministe
D = $(Q^d, X, \Sigma o, E^d, q_0^d, Q_f^d, I)$ où :

- $Q^d \subset 2^{QxF}$ x C(X) est un ensemble de sommets du diagnostiqueur. Chaque sommet du diagnostiqueur est défini par un ensemble

$q^d = (\{(q_i; \emptyset i); i \in f1,. . . . ., k\}, z)$, où z désigne une zone d'horloges et $(qi,\emptyset i)$, $i \in \{1, ... ,k\}$ correspond à un ensemble de couples "sommet/étiquette de diagnostic".

- Ed est un ensemble fini de transitions. Une transition entre deux sommets du diagnostiqueur q_d et q'_d est définie par un quintuplet
- (q_d,σ, g, Y,q'_d), où σdésigne unévénement observable ; i.e.,$\sigma \in \Sigma o$, et g est une contrainte de garde de l'ensemble C(X).

- q'_ddésigne le sommet initial du diagnostiqueur. Ce sommet est défini comme suit :q'_d = ({(q_0, *N)*}; z_0), où q0 désigne le sommet initial du modèle du système et z0désigne la zone d'horloge initiale. Cette zone ponctuelle associe la valeur 0 pour toutes les horloges de l'ensemble X ; i.e., $x_1 = x_2 = ... = x_n = 0$.L'étiquette N indique que cet état initial fait partie du comportement normal du système.

- Q'_d désigne l'ensemble des sommets finaux.
- *I* est la fonction qui associe une contrainte d'invariant à chaque sommet.

Remarque1.1 Nous soulignons que, contrairement à la définition originale des automates temporisés introduite dans **[07]**, les conditions des gardes des transitions définies dans l'automate temporisé du diagnostiqueur contiennent des contraintes diagonales de la forme x - - y ∼ c. L'implémentation du diagnostiqueur reste toujours possible en considérant ces contraintes.

Définition 1.4 : Soit $q^d = (\{(q1; \emptyset 1), . . . ,(q_k; \emptyset_k)\}; z)$ un sommet d'un diagnostiqueur.
Nous définissons l'opérateur D*is*, qui renvoie la partie discrète d'un sommet du diagnostiqueur ; i.e. D*is* $(q^d) = \{(q_1; \emptyset_1), ...,(q_k; \emptyset_k)\}$. De même, l'opérateur Z renvoie la zone d'horloges associée à un sommet du diagnostiqueur ; i.e., $Z(q^d)$ = z**.[07]**

Définition 1.5 : Un sommet du diagnostiqueur $q^d = (\{(q_1; \emptyset_1) ….. (q_k; \emptyset_k)\}; z)$ est dit :

1. Fi-certain, si $\emptyset p = Fi, \forall p \in \{1, . . ., k\}$.
2. Fi-incertain, s'il existe $(q ,\emptyset); (\bar{q}, \bar{\emptyset}) \in Dis(q^d)$, où $\emptyset = Fi$ et $\bar{\emptyset} \neq Fi$**.[07]**

Intuitivement, un sommet du diagnostiqueur est Fi-certain lorsque tous les états estimés dans ce sommet sont Fi-défaillants. Lorsque la fonction de diagnostic détecte que le sommet courant du diagnostiqueur est Fi-certain, elle génère une alarme annonçant l'occurrence d'un défaut de l'ensemble Fi. Par contre, si le sommet courant est Fi-incertain, aucune décision ne pourra être prise par cette fonction. En effet, les états estimés comportent à la fois des états Fi-défaillants et d'autres états associés à un autre mode fonctionnement (le mode de fonctionnement normal ou un mode de défaillance $Fj , j \neq i$) ce qui implique une ambiguïté dans la prise de décision**.[07]**

Dans la suite, nous présentons un exemple qui illustre le diagnostiqueur du modèle du système de chauffage de liquides, présenté dans la figure 1.3. Nous détaillons ultérieurement l'algorithme de synthèse de ce diagnostiqueur.

Exemple 1.3 : L'automate temporisé de la figure 1.4 représente le diagnostiqueur du système de chauffage de liquides, obtenu à partir du modèle de la figure 1.3 Chaque sommet de ce diagnostiqueur comporte un ensemble de paires (Sommet, étiquette), associé à une contrainte sur l'horloge x qui définit l'évaluation de cette horloge après le franchissement de la transition d'entrée du sommet. **[02]**

Afin de construire ce diagnostiqueur, nous avons considéré la partition de défauts suivante *:* $\Sigma_f = F_1 \cup F_2$, où F_1 = {*f*uite} correspond à l'occurrence d'une fuite et
F2 = {blocage _V1} correspond au blocage de la vanne ***V1***. Nous allons considérer deux cas d'application de ce diagnostiqueur.
Nous supposons, dans un premier cas, que le système effectue une exécution sur la trace :(fuite, 10) (rempli, 37.4) (évacuer, 60) (vide,13).
En observant la projection observable de cette trace: (rempli, 47.4) (évacuer,60)(vide,13), le diagnostiqueur évolue vers le sommet *({(**rempl**. &**fuite**,F_1)}, x = 0).*
La fonction de décision constate que le sommet est F_1*-certain* et génère, par conséquence, une alarme indiquant l'occurrence d'un défaut de l'ensemble F1.
Nous remarquons pour n'importe quelle suite d'événements observés, le diagnostiqueur va évoluer dans des sommets F1-certain. Ceci est attendu puisque les défauts considérés dans le modèle du système sont permanents.

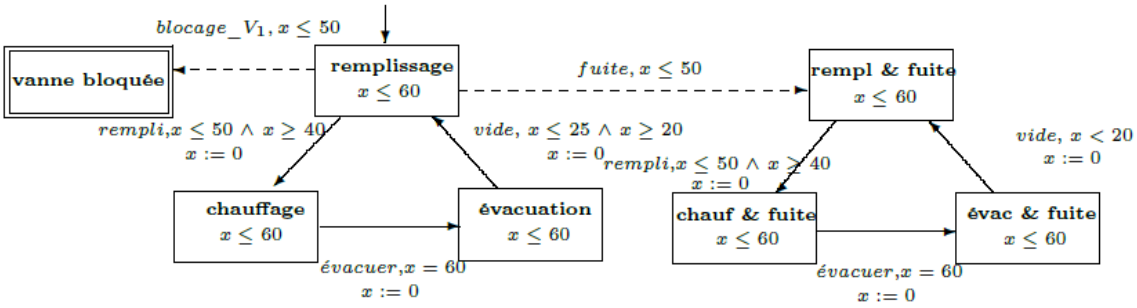


Figure 1.3 *Modèle du système de chauffage de liquides vérifiant toutes les hypothèses de Modélisation*

Nous supposons, dans un deuxième cas, que le système effectue une exécution sur la trace : (*rempli*, 43.4)(*évacuer*,60)(*vide*,23.4)(*blocage*_V1,4).
Le diagnostiqueur accepte la trace (rempli,43.4)(évacuer,60)(vide,23.4), puis il ne reçoit plus d'événements. Lorsque 51 u.t. s'écoulent après l'occurrence de l'événement *vide*, une transition urgente est franchie vers le somme t({(**vanne bloquée**, F2)}; x ≥51) dans le diagnostiqueur.
Ensuite, la fonction de décision constate que le sommet courant est F2-certain puis génère une alarme Indiquant l'occurrence d'un défaut de l'ensemble F2.

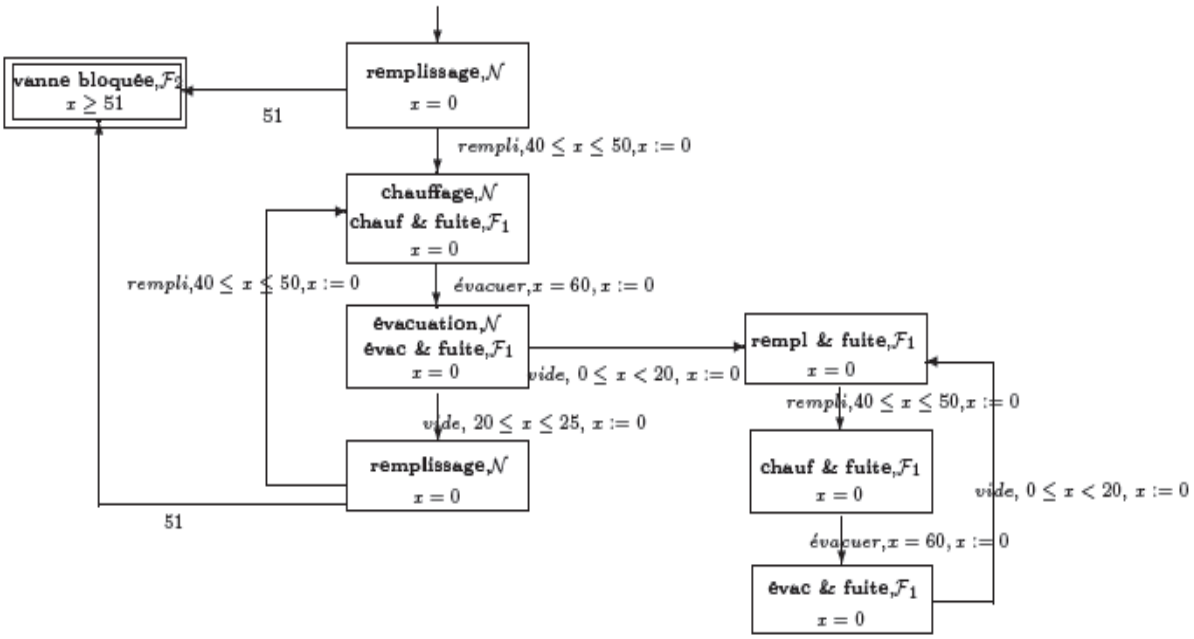


Figure 1.4 *Exemple du diagnostiqueur d'un système de chauffage de liquides*

1.4.1 Synthèse du diagnostiqueur :

Nous présentons dans cette partie notre méthode de synthèse du diagnostiqueur.
Nous commençons par définir un ensemble des fonctions nécessaires pour l'élaboration de cette méthode. Puis, nous exposons l'algorithme de synthèse du diagnostiqueur. **[02]**

1.4.1.1 Prérequis :

Nous définissons l'opérateur de propagation des étiquettes de diagnostic :
$\otimes: \Phi \times \Sigma \rightarrow \Phi$
. Cet opérateur permet de déterminer l'étiquette de diagnostic associée à
Un état donné suite à l'occurrence d'un événement $\sigma \in \Sigma$

$$\emptyset \otimes \sigma \begin{cases} Fi\ , if \emptyset = Fi; \\ Fi, if \emptyset = N and \sigma \in Fi; \\ N, if \emptyset = N and \sigma \notin \Sigma_f \end{cases}$$

Nous pouvons considérer l'extension de cet opérateur pour propager une étiquette sur une séquence d'événements. Dans ce cas, il suffit d'appliquer successivement cet opérateur sur chaque événement de cette séquence, tout en propageant le résultat obtenu à chaque étape.

● Nous définissons la fonction d'accessibilité non-observable ***UR*** qui permet de déterminer l'ensemble d'états accessibles depuis un ensemble d'états de départ,
en franchissant toutes les séquences possibles de transitions sur des événements non observables. Les étiquettes de diagnostic associées aux états de départ sont propagées durant ce calcul d'accessibilité, en utilisant l'opérateur $\otimes$.

$UR : 2^{Q \times C(X) \times \Phi} \longrightarrow 2^{Q \times C(X) \times \Phi}$
$[(q_i, z_i), \emptyset_i] \longrightarrow [(q_j, z_j), \emptyset_j] \mid \exists \mathbf{s} \in (\Sigma_{\mathbf{u0}} \times \mathbb{R}_+)^*, (q_i, z_i) \rightsquigarrow (q_j, z_j)$ et $\emptyset_{j = \Phi j \otimes \| \mathbf{s} \|}$
Dans cette définition de la fonction UR, l'ensemble des états de départ est donné sous la

forme d'un ensemble de triplets ({(q_1; $\emptyset_1$) ….. (q_k; $\emptyset_k$)}; z).

Chaque triplet [(q_i; z_i); $\emptyset$i] représente un état symbolique (q_i; z_i) associé avec une étiquette de diagnostic $\emptyset$i.

1.4.2 Architecture de l'algorithme de synthèse du diagnostiqueur :

L'algorithme de synthèse du diagnostiqueur se base sur deux notions importantes, illustrées dans la figure 1.5, qui sont : les états d'entrée et les états d'ombre **([05]).**

- Les états d'entrée correspondent aux états accessibles suite à l'occurrence d'un événement observable. Chaque sommet du diagnostiqueur est constitué par un ensemble d'états d'entrée. Ces états sont décrits par un ensemble d'états symboliques associés à des étiquettes de diagnostic.

- Les sommets d'ombre correspondent à l'ensemble d'états accessibles depuis un ensemble d'états d'entrée, suite à l'occurrence d'événements non observable et/ou l'écoulement du temps. En effet, ces états sont obtenus en appliquent l'opérateur d'accessibilité non observable UR, sur les états symboliques d'un sommet du diagnostiqueur.

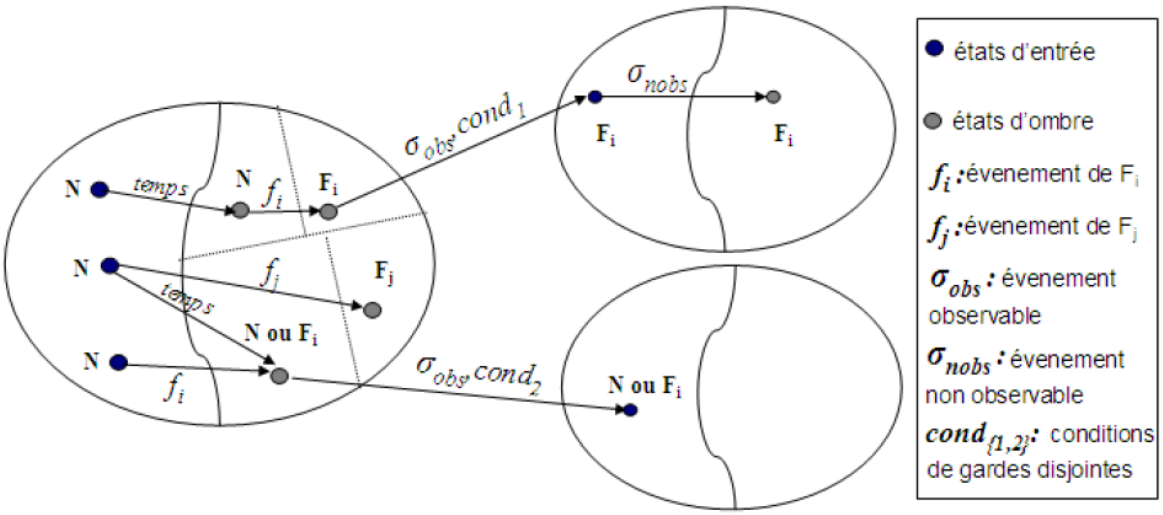


Figure1.5 – *États d'entrée et États d'ombre*

- ***Première étape*** : déterminer l'ensemble des états d'ombre à partir des états d'entrée ;

- ***Deuxième étape***: déterminer, pour chaque événement observable, le sous-ensemble d'états d'ombre permettant de franchir une transition sur $\boldsymbol{\sigma}$. Ce sous ensemble sera appelé *les états franchissables* ;

- ***Troisième étape :*** estimer, pour chaque évaluation d'horloges associée à l'occurrence d'un événement observable, l'ensemble des états du système. Cette étape s'appuie sur une partition de l'ensemble des états franchissables. Deux états franchissables appartenant à deux ensembles différents de cette partition ne doivent pas avoir la

même évaluation d'horloge. En effet, l'évaluation associée à l'occurrence d'un événement observable nous permet de distinguer l'ensemble des états franchissables possibles. Ces derniers doivent appartenir au même ensemble de la partition.

- ***Dernière étape*** : Créer, pour chaque sous-ensemble d'états de la partition des états franchissables, un nouveau sommet du diagnostiqueur. Les états d'entrée constituant ce sommet du diagnostiqueur correspondent aux successeurs discrets des états franchissables considérés. Nous considérons dans la suite un exemple illustrant la construction d'un sommet du diagnostiqueur à partir d'une partie d'un automate temporisé. Cet exemple est décrit dans la figure 1.6

Nous considérons dans la suite un exemple illustrant la construction d'un sommet du diagnostiqueur à partir d'une partie d'un automate temporisé. Cet exemple est décrit dans la figure 1.6

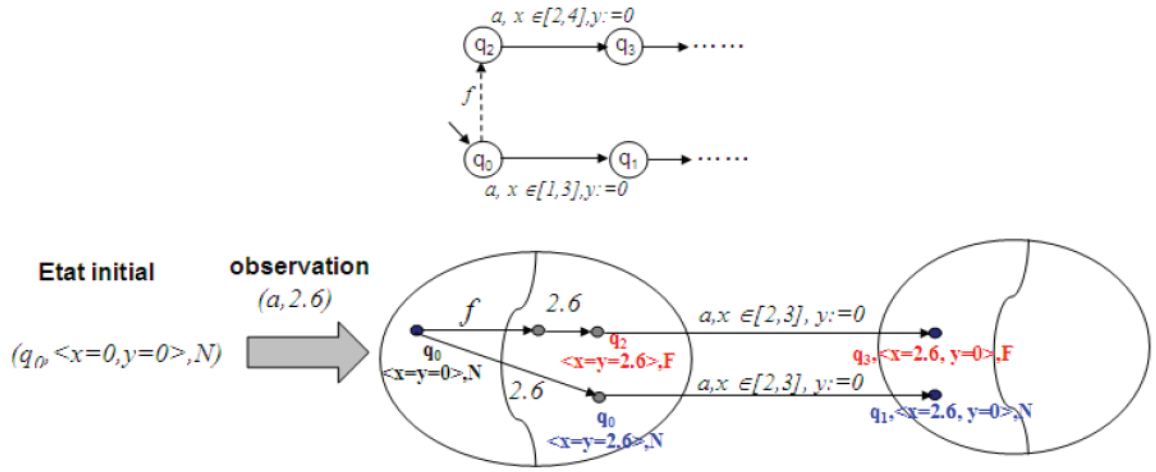


Figure1.6 – *Principe de construction d'un sommet du diagnostiqueur.[0 2]*

Dans cet exemple, l'ensemble des états d'ombre de l'état initial *(q0, x = y = 0, N)*, correspond aux états accessibles suite à l'écoulement du temps et/ou l'occurrence de l'événement de défaut *f*. En considérant K = 5 (borne supérieure des horloges), l'ensemble d'états d'ombre accessibles est égale à

$\{(q0; <x = y \wedge x \leq 5>, N)(q2; <x = y \wedge x \leq 5>, F)\}$.

L'étiquette *F* indique ici l'occurrence du défaut *f*.
L'ensemble des états franchissables sur l'événement a à partir des états d'ombre est égale à

$\{(q0,<x = y \wedge x \in [1,3] >, N)(q2,<x = y \wedge x \leq [2,4]>, F)\}$
Correspondent aux états d'ombres vérifiant les gardes des transitions sur l'événement a

L'étape suivante consiste à créer une partition sur l'ensemble des états franchissables.
Selon la évaluation d'horloges v à l'instant de l'observation de l'événement a, nous pouvons distinguer trois cas possibles

- si v(x) ∈ [1; 2[, l'état estimé du système correspond à {(q0, *v*, *N*)} ;

- si v(x) 2 [2; 3], l'état estimé du système correspond à {(q0, *v*, *N*), {(q2, *v*, *F*)}. Cette situation correspond au cas illustré dans notre exemple. En effet, l'événement a est observé après 2.6 u.t., les états estimés à cet instant sont :
  {(q_0,<x = y = 2.6>, *N*), (q_0,< x = y = 2.6>, *F*)}

- si *v*(x) ∈]3; 4], l'état estimé du système correspond à {(q2, *v*, *F*)}.

A partir de cette partition sur l'ensemble des états franchissables, nous pouvons construire trois sommets successeurs. Dans la figure 6, un seul sommet successeur est illustré. Ce sommet est obtenu en calculant les successeurs discrets de l'ensemble d'états {(q0, <y = x ∈ [2,3], >*N) (*q2, <y = x ≤ [2,3]>, *F*)}
Ainsi, nous obtenons le sommet ({(q_1, *N*),(q3,*F*), <x ∈ [2,3] ∧*y* = 0 > }).

Dans la suite, nous exposons l'ossature de l'algorithme de synthèse du diagnostiqueur d'une manière plus détaillée. Dans ce pseudo-code, nous désignons par q^d le sommet en cours d'élaboration (recherche de ses sommets successeurs) et SOMMETS_A_TRAITER l'ensemble des sommets qui restent à traiter. Cet ensemble est une structure de type file d'attente (FIFO). Par ailleurs, SOMMETS_TRAITÉS désigne l'ensemble des sommets du diagnostiqueur qui ont déjà été traités. Nous rappelons que le sommet initial q_0^d = {{(q_0, *N*)}, z_0} du diagnostiqueur est constitué par le sommet initial q_0, la zone d'horloge z_0 définie par $x_1 = x_2 = \ldots = x_n = 0$ et l'étiquette N indiquant que le système admet un comportement normal à son état initial. L'automate temporisé du diagnostiqueur peut être obtenu en appliquant les étapes du pseudo-code suivant :

1. initialisations :
{
(a) créer le sommet initial q_0^d = {{(q_0, *N*)}, z_0},
(b) ajouter q_0^d à SOMMETS_A_TRAITER
(c) SOMMETS_TRAITÉS ←{ },
}

2. tant que (SOMMETS_A_TRAITER6= fg)
{
(a) retirer un élément q^d de SOMMETS_A_TRAITER et l'ajouter à SOMMETS_TRAITÉS,
(b) déterminer ÉTATS_OMBRE, l'ensemble des états d'ombre issues de q^d.
(c) s'il existe des états finaux parmi les états d'ombre :

i. déterminer l'entier τ qui correspond au délai minimal, après lequel aucun événement observable ne pourra être observé ; i.e., le système sera certainement dans un sommet final :

$\tau = min\{\ d \in \mathbb{N} \mid \forall\ (q, v, \emptyset\ ) \in$ ÉTATS_OMBRE, $v(y) \geq d \Longrightarrow q \in Q_f\ \}$

Où l'horloge *y* mesure le temps écoulé entre deux événements observables successifs (autrement dit, le temps de séjour dans chaque sommet du diagnostiqueur).

ii. grouper ces états finaux dans un sommet final du diagnostiquerq_f^d, puis créer une transition urgente vers ce sommet étiquetée par le délai τ

(d) répéter pour chaque événement observable$\boldsymbol{\sigma}$.

{

i. déterminer à partir des états d'ombre, le sous-ensemble d'états franchissables sur $\boldsymbol{\sigma}$,

ÉTATS_FRANCHISSABLES$^{\sigma}$ = {(q, v, ∅) ∈ ÉTATS_OMBRE tel que il existe une transition $(q, v) \xrightarrow{\sigma} (q', v')$}

ii. créer une partition sur l'ensembleÉTATS_FRANCHISSABLES$^{\sigma}$, telle que : (1) un état appartient à un seul sous-ensemble de la partition et (2) deux états ayant la même évaluation d'horloge doivent appartenir au même sous-ensemble de la partition (voir l'exemple de la partition illustrée dans figure 1.7).

iii. répéter pour chaque zone d'horloges η décrivant les évaluations d'un sous ensemble d'états franchissables de la partition :

- créer un sommet du diagnostiqueur $\tilde{q}^d$, contenant les successeurs discrets Sur$\boldsymbol{\sigma}$du sous-ensemble considéré de la partition d'états franchissables.
- créer une transition de q^d à $\tilde{q}^d$ ayant comme une contrainte de garde le prédicat η.
- ajouter $\tilde{q}^d$ à SOMMETS_A_TRAITER s'il n'a pas été encore visité ; i.e., $\tilde{q}^d$ SOMMETS_TRAITÉS.

}

}

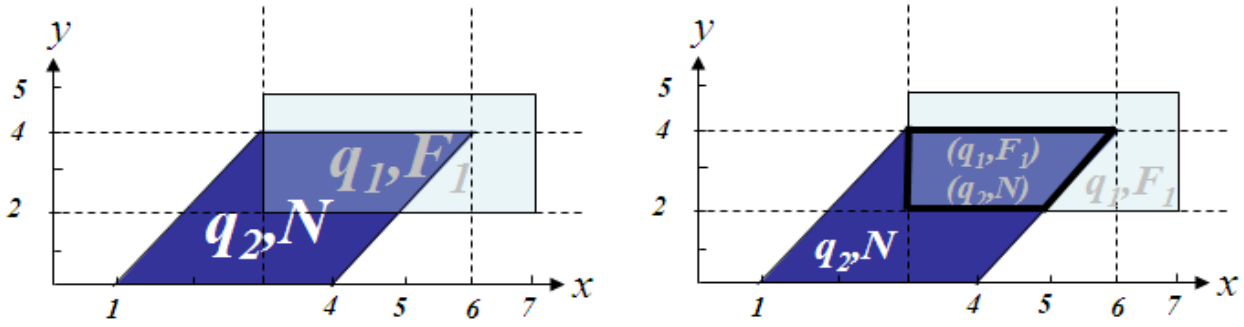


Figure1.7 *Création d'une partition de trois ensembles d'états franchissables.* **[02]**

Afin de mieux illustrer les différentes étapes de ce pseudo-code, nous allons construire dans l'exemple suivant, le diagnostiqueur correspondant à l'automate temporisé de la figure 1.7.

Exemple 1.4 : Dans la figure 1.8, nous présentons une partie d'un automate temporisé dont on désire construire le diagnostiqueur. Il est clair que cette partie vérifie les hypothèses nécessaires pour l'application de l'algorithme de synthèse du diagnostiqueur. Les événements *a* et *b* sont observables tandis que f constitue le seul événement de défaut. Ainsi, nous aurons un seul mode de défauts, *F1* = { *f* }. Par ailleurs, les sommets de ce modèle sont non-finaux, auxquels nous associons la condition d'invariant$<x \leq 20 \ \wedge y \leq 20>$,où 20 est une valeur arbitrairement choisie.

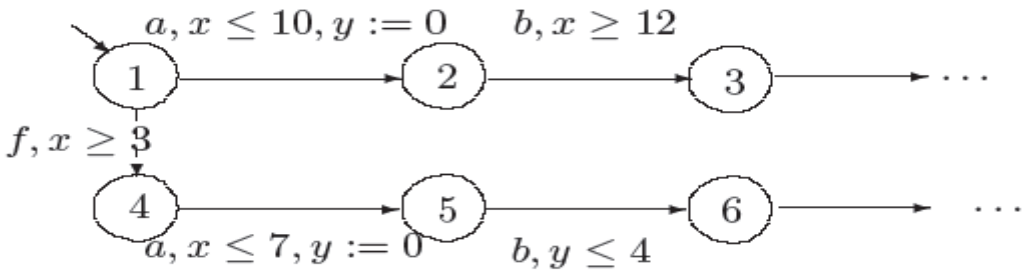


Figure1.8 *Exemple de construction d'un diagnostiqueur : modèle initial*. **[02]**

La construction du diagnostiqueur se déroule comme suit :

- créer du sommet initial ({(1, *N*)},< x = y = 0>),
- à partir du sommet initial, *[(1, <x = y = 0>);N]*, appliquer la fonction d'accessibilité non-observable afin de déterminer les états d'ombre.

obtient ÉTATS_OMBRE=*{[(1,<x = y ∧ x ≤ 20>);N],[(4; <x = y∧3 ≤x ≤ 20>);F1]}*

- calculer les états franchissables pour les transitions sur l'événement *a* :

ÉTATS_FRANCHISSABLES ^{a}= *{[(1, <x = y ∧ x ≤ 10>), N], [(4,< x =y ∧ 3 ≤ x ≤ 7>),F1]}.*

- déterminer les évaluations d'horloges correspondant aux états de ÉTATS_FRANCHISSABLES ^{a} afin d'avoir des zones d'horloges disjointes.

On obtient les états symboliques suivants :

- [(1, <x = y ∧ x ≤ 10 ∧ ¬ (3 ≤ x ≤ 7)>),N] ;
- [(1, <x = y ∧3≤x ≤7>),N] ;
- *[(4; <x = y ∧ 3 ≤ x ≤7>),F1] ;*

- à partir de la partition de l'ensemble d'états franchissables, construire deux nouveaux sommets du diagnostiqueur :
  - q_1^d = *({(2, N) (5, F1)}),<y= 0 ∧ 3 ≤ x ≤7>) ;*
  - q_2^d = *({(2, N), <y= 0 ∧ x ≤ 10 ∧¬ (3 ≤ x ≤ 7)>) ;*

- à partir du sommetq_1^d, appliquer la fonction d'accessibilité non-observable afin de déterminer les états d'ombre. On obtient :

  ÉTATS_OMBRE = {[(2,<3 ≤ x – y ≤ 7 ∧ 3 ≤ x ≤ 20>),N], [5,<3 ≤ x – y ≤ 7 ∧3 ≤x≤20>),F_1]}

- calculer les états franchissables pour les transitions sur l'événement *b* :

ÉTATS_FRANCHISSABLES^{b} *={[(2,<3 ≤ x – y ≤ 7 ∧ 12 ≤ x ≤ 20>), N], [5,<3 ≤ x – y ≤ 7 ∧ x ≥3∧ y ≤ 4 >),F_1]}.(Voir figure 9)* ;

- les zones d'horloges correspondant aux états de ÉTATS_FRANCHISSABLES^{b} sont déjà disjointes : pas besoin d'élaborer une partition sur cet ensemble.
- construire les sommets successeurs deq_1^d, on obtient
  - q_3^d = *({(6, F_1)},<3 ≤ x – y ≤ 7 ∧0 ≤y ≤ 4) ;*
  - q_4^d = *({(3,N)}, <3 ≤ x – y ≤ 7 ∧10 ≤x ≤ 20) ;*

Enfin, nous illustrons le diagnostiqueur résultant de cette construction dans la figure 1.10 La table 1 définie la signification de chaque symbole utilisé dans l'automate temporisé du diagnostiqueur. **[02]**

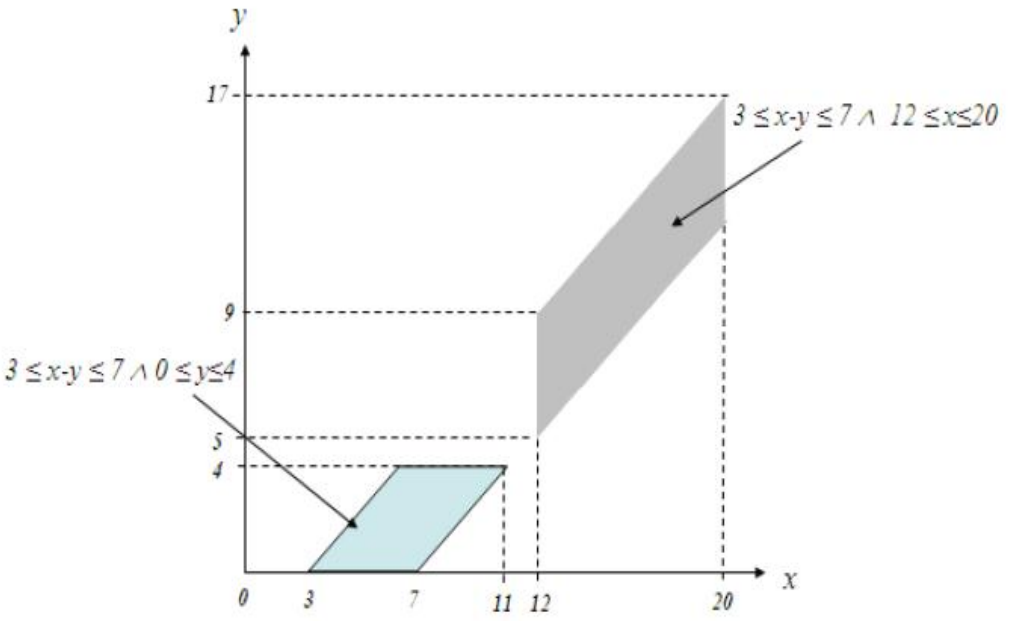


Figure 1.9 *Exemple de construction d'un diagnostiqueur : zones disjointes des états franchissables.*

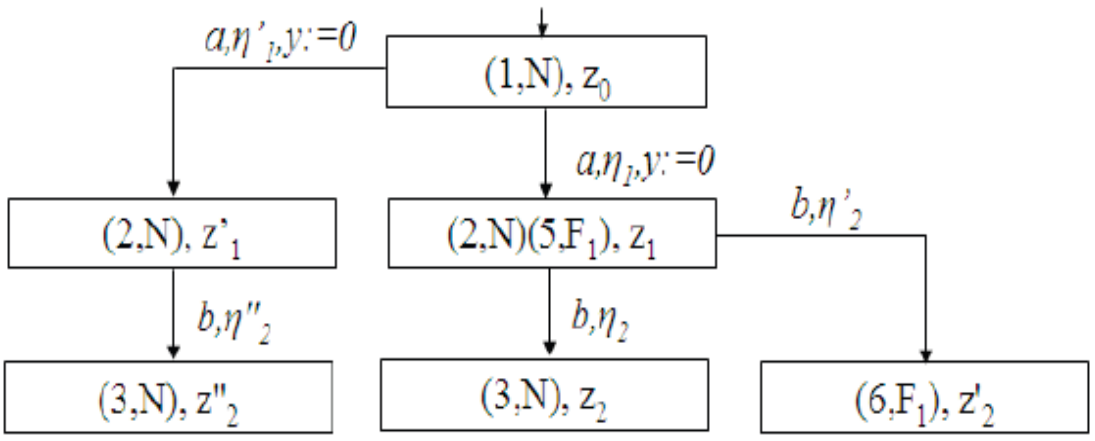


Figure1. 10 *Exemple de construction d'un diagnostiqueur : modèle du diagnostiqueur.*

Symbole	contrainte d'horloges
z_0	$\langle x = y = 0 \rangle$
z_1	$\langle y = 0 \wedge 3 \leq x \leq 7 \rangle$
z'_1	$\langle y = 0 \wedge x \leq 10 \wedge \neg(3 \leq x \leq 7) \rangle$
z_2	$\langle 3 \leq x - y \leq 7 \wedge 12 \leq x \leq 20 \rangle$
z'_2	$\langle 3 \leq x - y \leq 7 \wedge 0 \leq y \leq 4 \rangle$
z''_2	$\langle (0 \leq x - y < 3 \wedge 12 \leq x \leq 20) \vee (7 \leq x - y \leq 10 \wedge 12 \leq x \leq 20) \rangle$
η_1	$\langle x = y \wedge 3 \leq x \leq 7 \rangle$
η'_1	$\langle x = y \wedge x \leq 10 \wedge \neg(3 \leq x \leq 7) \rangle$
η_2	$\langle 3 \leq x - y \leq 7 \wedge 12 \leq x \leq 20 \rangle$
η'_2	$\langle 3 \leq x - y \leq 7 \wedge 0 \leq y \leq 4 \rangle$
η''_2	$\langle (0 \leq x - y < 3 \wedge 12 \leq x \leq 20) \vee (7 \leq x - y \leq 10 \wedge 12 \leq x \leq 20) \rangle$

Tab1.1 *– description des symboles utilisés dans modèle du diagnostiqueur*

1.5 Conditions de diagnosticabilité

Avant d'énoncer les conditions nécessaires et suffisantes pour la diagnosticabilité d'un système à événements discrets G dont le diagnostiqueur est Gd et le générateur est G', donnons quelques définitions utiles :

- Un état q de Q_d est dit ***Fi-certain*** si :∀ (x, l) ∈ q, F_i ∈ l.
- Un état q de Q_d est dit ***Fi-incertain*** si : ∃ (x, l)(y,l')∈ q, F_i ∈l et F_i ∈ l'.
- Un état *q* de Q_d est dit ***ambiguë*** si :∃ (x,l) ∈ q, A∈ l.

Remarque 1.6: Différence entre un état Fi-incertain et un état ambigu.

- Un état *q* est ***Fi-certain*** si toute trace de *q0* à *q* contient obligatoirement la faute *Fi*.

- Un état *q* est ***Fi-incertain*** s'il y a deux traces *s1* et *s2* de *q0* à *q* ayant la même projection observable tel que *s1* contient *Fi* alors que *s2* ne contient pas *Fi* et que dans le système initial *G*, les traces *s1* et *s2* mènent vers deux états différents.

- Un état *q* est ***ambigu*** s'il y a deux traces *s1* et *s2* de *q0* à *q* ayant la même projection observable tel que *s1* contient *Fi* alors que *s2* ne contient pas *Fi* etque dans le système initial *G*, les traces *s1* et *s2* mènent vers un même état.

- Un ensemble d'états x1, …, xn. X forme un cycle dans G si :

$\exists s \in L(G, x1)$ tel-que $s = \sigma 1\ \sigma 2 \ldots\ldots \sigma_n$ et $\Sigma_0 (x_i, \exists_i) = x_{(i+1) \bmod n}$, i= 1,2,...,n.

- Un ensemble d'états *Fi-incertains* $q_1, q_2, \ldots, q_n \in Q_d$ forme un cycle F_i-*indéterminé* si:

a) $q_1, q_2, \ldots, q_n$ forment un cycle dans *Gd* avec :

$\delta_d (q_l, \sigma_l) = q_{(l+1) \bmod n}$ tel-que $\sigma_l \in \Sigma_0$, l=1,2,...,n

b) Pour ce cycle dans Gd, il existe un cycle correspondant dans G' impliquant uniquement des états ayant Fi dans leurs étiquettes dans le cycle de Gd et un cycle similaire dans G' impliquant uniquement des états n'ayant pas Fi dans leurs étiquettes dans le cycle de Gd. Formellement :

$\exists (x_l^k, l_l^k) \in q_1$, l = 1,…m, r = 1,…..m' tel que

1. $F_i \in l_l^k$, $F_i \notin l_l^r$ pour tout l, k, r,
2. Les séquences d'états $\{x_l^k\}$l = 1,…n, k = 1,...m et $\{y_l^r\}$l = 1,…n, r = 1,…m'

   Forment des cycles dans G' (le générateur) avec :

$(x_l^k, \sigma_l, x_{l+1}^k) \in \sigma_{G'}$ $\quad l = 1, \ldots\ldots, n \quad k = 1, \ldots\ldots\ldots, m$

$(x_n^k, \sigma_n, x_l^{k+1}) \in \sigma_{G'}$ $\quad k = 1, \ldots\ldots, m - 1$

$(x_n^m, \sigma_l, x_l^l) \in \sigma_{G'}$

Et

$(y_l^r, \sigma_l, y_{l+1}^r) \in \sigma_{G'}$ $\quad l = 1, \ldots\ldots, n \quad r = 1, \ldots\ldots\ldots, m'$

$(y_n^k, \sigma_l, y_l^{k+1}) \in \sigma_{G'}$ $\quad r = 1, \ldots\ldots, m' - 1$

$(y_n^{m'}, \sigma_l, y_l^l) \in \sigma_{G'}$ **[03]**

1.5.1 Diagnosticabilité

Un langage L sans fautes multiples de même type est diagnosticable si et seulement si son diagnostiqueur Gd satisfait les conditions suivantes :

C1) Il n'y a pas de cycle F_i-indéterminé dans Gd pour tout type d'erreur F_i.

C2) Il n'y a pas d'état q de Q_d qui soit ambigu.

Exemple1.5 *:* Prenons le circuit de la figure 1.11, les états possibles de circuit sont :

(W1, W2) **R1** Marche, **R2** Marche ;

(w1, S2) **R1** Marche, **R2** Court circuitée;

(W1, O2) **R1** Marche, **R2** Ouverte ;

(S1, W2) **R1** Court circuitée, **R2** Marche;

(O1, W2) **R1** Ouverte, **R2** marche.

Ici, nous supposons qu'il existe une seule défaillance dans le circuit pour la simplicité. nous définissons les événements suivants

$\sigma 1: V_{Sortie} = 0$;

$\sigma 2 : V_{sortie} = V_{entrée}$;

$\sigma 3 : I = 0$;

$\sigma 4: I \neq 0$;

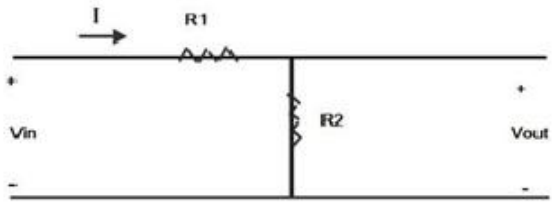


Figure 1.11

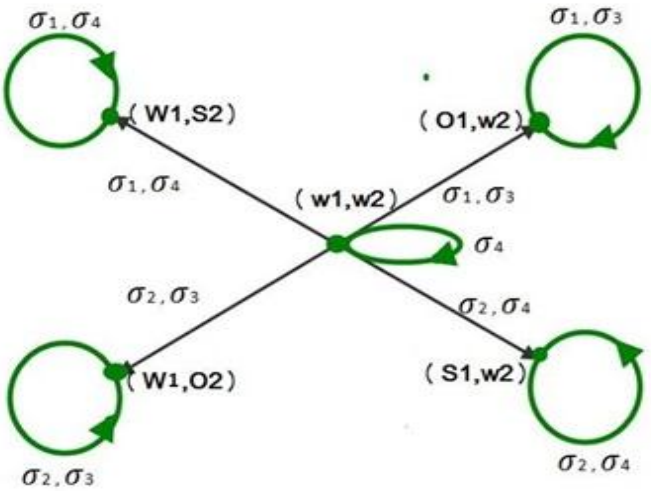


Figure 1.12

En examinant le circuit, nous trouvons :

Que, par exemple $V_{sortie} = 0$ implique que R1 est ouverte ou R2 est court-circuitée tandis que $V_{sortie} = V_{entrée}$ implique que R1 est court-circuitée ou R2 est ouvert.

Par conséquent, les transitions de l'Etat sont données dans la figure 1.12.

La diagnosticabilité de circuit dépend de $\Sigma_0$et T. soit $\Sigma_0=\{ \sigma 1, \sigma 2 \}$ (ce n'est que V_{Sorie} et $V_{entrée}$ qui peuvent être mesurés) et la partition désirée T = {{W1,W2}, {W1,S2}, {W1,O2}, {S1,W2}, {O1,W2}}

1.6 Conclusion

Nous avons présenté au cours de ce chapitre le principe de notre approche de diagnostic des SED à base d'automates temporisés.

Dans un premier volet, nous avons défini Les systèmes à événements discrets.

Ainsi le système à diagnostiquer est modélisé par un automate temporisé décrivant à la fois son comportement normal et ses comportements défaillants.

Ensuite, nous avons présenté, intuitivement puis formellement, notre approche de diagnostic qui se base sur la construction d'un diagnostiqueur. Ce diagnostiqueur est un automate temporisé déterministe qui évolue en fonction des événements observables générés par le système et estime l'état courant du système et les occurrences de défauts.

Un fonction de décision, analyse le sommet courant du diagnostiqueur et annonce l'occurrence d'un défaut du mode Fi lorsque ce sommet est Fi-certain.

Dans un deuxième volet, nous avons étudié la diagnosticabilité du système à diagnostiquer.

Nous avons établi que l'utilisation d'un diagnostiqueur est soumise à la vérification de notion de diagnosticabilité par le modèle du système considéré.

Enfin nous avons présenté une méthode systématique permettant de vérifier la diagnosticabilité d'un modèle.

Cette méthode repose sur la détection dans le modèle du diagnostiqueur des cycles

Fi-indéterminé et des sommets finaux Fi-incertain. Un théorème liant la notion de diagnosticabilité et la vérification de ces conditions a été formalisé.

Chapitre 2

Synthèse de contrôleurs des SED et SEDT

2 Synthèse de contrôleurs des SED

2.1.1 - Introduction

Ce chapitre présente les approches les plus connues de contrôle des SED. Pour commencer, nous clarifions les concepts fondamentaux et le schéma de supervision.

Nous présentons ensuite les bases de la théorie générale de la commande par supervision des SED, développée par Ramadge et Wonham. Dans toute la suite de ce travail nous parlerons de supervision, superviseur et de contrôleur.

Étant donné un système manufacturier, nous nous intéressons maintenant à lui imposer un fonctionnement respectant un cahier des charges désiré.

Pour les systèmes peu complexes, la manière la plus facile de contrôle est la conception directe du modèle supervisé du système. Malheureusement, dès que la complexité du système augmente, cette méthode ne peut pas garantir le respect du cahier des charges.

Pour pallier cet inconvénient, des méthodes formelles permettant la synthèse de contrôleurs pour les systèmes à événements discrets ont été développées. Ces méthodes reposent sur l'utilisation d'outils tels que les automates et les réseaux de Pétri.

Pour plus de détails sur le contrôle des SED on peut se rapporter à **[08].**

Ce sont les travaux de P. J. Ramadge et W. M. Wonham (R&W) qui constituent les bases de la théorie générale de la supervision des SED **([09])**

2.1.2 Théorie de la supervision

La théorie de la supervision des systèmes à événement discrets a été initiée par les travaux de Ramadge et Wonham (R&W) **[0 9] [1 0].**

Dans cette approche des modèles logiques sont utilisés pour décrire le fonctionnement d'un procédé. Un procédé est considéré comme un SED qui évolue spontanément en générant des événements. Le schéma de supervision est

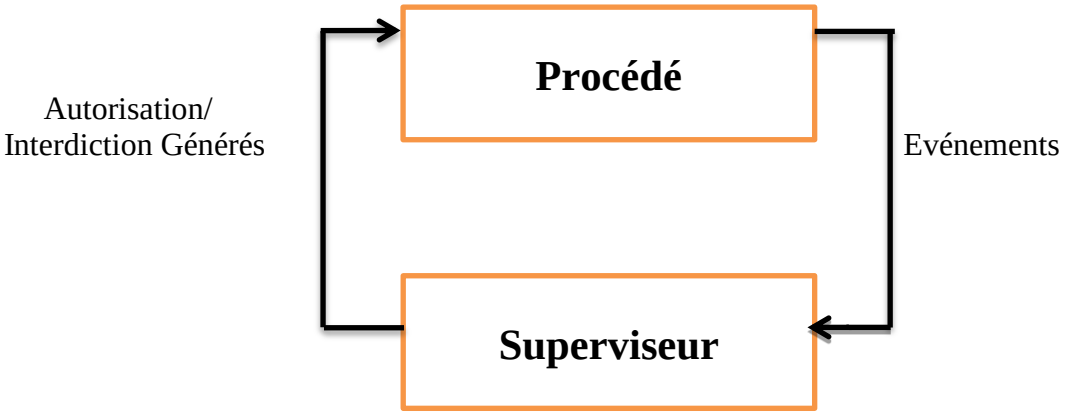


Figure 2.1*- schéma de supervision*

2.1.2.1 Présentation du travail de Ramadge et Woham

L'objectif de l'approche R&W est de synthétiser un contrôleur pour un modèle donné, tel que le fonctionnement du procédé couplé au contrôleur reste toujours compris dans l'ensemble des comportements valides. De plus, ce fonctionnement doit être le plus permissif possible. Le comportement du système non-contrôlé, ainsi que l'ensemble des comportements désirés, sont chacun définis par un langage.

À partir de ces langages, le but de la synthèse est de déterminer le sous-ensemble des comportements du procédé qui appartient aussi à la spécification. Ce sous ensemble caractérise le langage du contrôleur et décrit les comportements du système contrôlé. Il est important, à ce niveau, de mentionner qu'il y a certains événements qui ne peuvent pas être interdits par le superviseur : les événements incontrôlables. L'importance du concept de contrôlabilité pour la théorie de la supervision des SED sera clarifiée dans les sections suivantes.

Un procédé peut être couplé à un ou à plusieurs contrôleurs. Si nous utilisons un contrôleur unique, la commande sera dite centralisée. S'il y a plusieurs contrôleurs couplés au même procédé, la commande sera modulaire.

L'approche de R&W assure l'optimalité du contrôleur pour les systèmes modélisés par des automates à états finis. Cependant, bien que ce formalisme permet la modélisation d'une large classe des SED, il est peu adapté aux systèmes réels à cause de sa grande sensibilité au problème de l'explosion combinatoire du nombre d'états (même dans les petits systèmes). En outre, l'utilisation de cette approche a montré une difficulté dans l'implémentation du système contrôlé. C'est pour ces raisons que de nombreuses tentatives

ont été faites pour adapter la théorie générale de la commande des SED à d'autres formalismes de modélisation **([11], [12], [13], [14], [15] [16] [17]),**

2.1.2.2 Schéma de la supervision

Toutes les informations utiles concernant le procédé physique à contrôler sont intégrées dans son modèle. Le procédé est considéré comme un *générateur d'événements*, c'est-à-dire un SED évoluant de façon spontanée en générant des événements (figure2.2a).

L'approche R&W se place au niveau d'abstraction logique (atemporel). Le procédé est modélisé par un automate $P = (Q, \Sigma, \delta, q_0)$.

Son comportement est alors donné par son langage, L(P). Ce langage contient toutes les séquences d'événements qui peuvent être générées par P. Il est nécessairement préfixe-clos.

Le comportement du procédé en isolation n'est en général pas satisfaisant, dans le sens où il ne respecte pas certaines propriétés désirables pour l'exploitation.

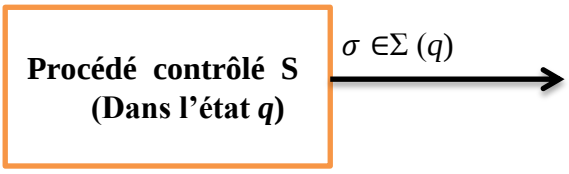
Ces propriétés constituent les *objectifs de contrôle*, ou la *spécification*. Le but du contrôle est de restreindre le comportement du procédé afin d'assurer le respect de ces objectifs. Cette restriction est réalisée par le biais d'un contrôleur, S, qui est lui-même un SED. Dans ce contexte, le contrôleur se révèle comme une fonction $S : L(P) \rightarrow 2^{\Sigma}$ qui, dans chaque état du procédé, envoie à celui-ci la liste d'événements interdits par la spécification —*la politique de contrôle*, Φ (figures 2.1 et 2.2c).

Étant donné un état $q \in Q$ du procédé, on note $\Sigma(q)$ l'ensemble des événements qui peuvent être générés par le procédé depuis cet état, et $\Phi(q)$ la liste des événements interdits par la spécification (*l'action de contrôle*) dans l'état q. Dans chaque état q, le procédé couplé au contrôleur reçoit en entrée la liste des événements interdits $\Phi(q)$ et émet en sortie l'ensemble des événements possibles et autorisés, $\Sigma(q) \setminus \Phi(q)$.

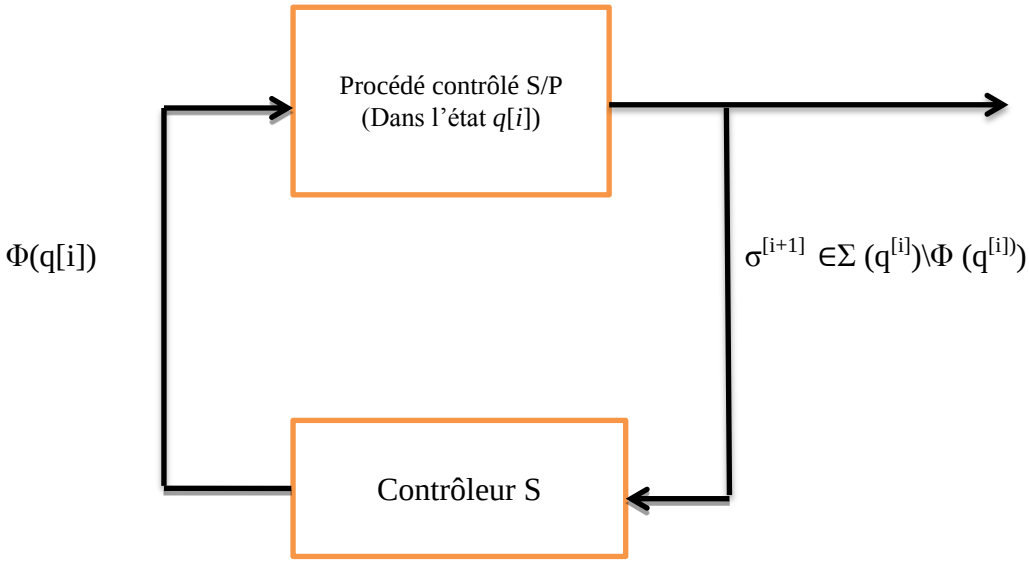
Ce procédé, illustré dans la figure 2.2b, est appelé *procédé contrôlé*. Un événement σest susceptible d'être généré par le procédé contrôlé S/P depuis un état q si, et seulement si, il peut être généré par le procédé en isolation et s'il n'est pas interdit. C'est-à-dire : $\sigma \in \Sigma(q)$ et $\sigma \notin \Phi(q)$. Il va de soit que le procédé contrôlé peut être défini de façon analogue, en spécifiant en entrée la liste des événements autorisés, $\Sigma \setminus \Phi$.

Le fonctionnement du procédé couplé au contrôleur, appelé *fonctionnement en boucle fermée*, est décrit dans la figure 2.2c. Le contrôle ajoute au procédé des contraintes

(a) Procédé en isolation



(b) Procédé supervisé



(c) Schéma de supervision avec indice de temps

Figure 2.2 *– Le concept de commande par supervision*

Supplémentaires, induisant un comportement souhaité pour le système en boucle fermée. En général, il y a plusieurs contrôleurs qui permettent de réaliser un fonctionnement en boucle fermée donné. Fondamentalement, l'observation du procédé par le contrôleur est asynchrone. A chaque instant logique [*i*], le contrôleur construit, en se basant sur l'histoire du système, une liste d'événements interdits, $\Phi\ (q^{[i]})$, et la fournit au procédé.

Après avoir reçu la liste, le procédé contrôlé peut générer à l'instant [*i + 1*] un événement $\sigma^{[i+1]} \in \Sigma\ (q^{[i]}) \backslash \Phi\ (q^{[i]})$.

Si l'occurrence de cet événement conduit le contrôleur vers un nouvel état, la liste des événements interdits est immédiatement actualisée, $\Phi\ (q^{[i+1]}) \neq \Phi\ (q^{[i]})$, et envoyée au procédé, qui réagit en conséquence. La notation S/P sera dorénavant utilisée pour désigner le fonctionnement en boucle fermée du procédé P couplé au contrôleur S.

Remarque 2.1 : Le rôle du contrôleur consiste strictement à restreindre le fonctionnement du procédé.

Pour bien garantir le fait que, après l'occurrence d'un événement quelconque, le contrôleur a le temps nécessaire pour interdire l'occurrence de tout événement indésirable, il est nécessaire de faire l'hypothèse suivante :

Hypothèse 2.1.*Deux événements ne peuvent pas être simultanément générés par le procédé.*

Cette hypothèse est basée sur le fait que les événements ont une durée nulle. Les procédés considérés évoluent à des instants continus du temps, et la probabilité que deux événements soient simultanément générés par le procédé est aussi nulle. **[01]**

Lors de toute discussion sur le contrôle des SED, il est important de réfléchir aussi au problème de l'optimalité. L'interdiction des comportements indésirables (illustrée dans le schéma de supervision) représente un aspect important du contrôle. Néanmoins, il existe un autre aspect, également important : la garantie du fait que le système n'est soumis à aucune restriction inutile. L'optimalité du contrôleur ne peut être assurée que lorsque cette deuxième exigence, *la condition de comportement maximal permissif*, est elle aussi garantie. Nous avons déjà mentionné le fait que l'approche R&W assume l'optimalité du contrôleur pour les modèles automates à états finis.

2.1.1.3Définition d'un superviseur

Un superviseur peut être perçu comme une machine à états dont les sorties sont des listes $\Phi(q^i)$ d'événements interdits.

On peut remarquer qu'entre deux occurrences successives d'événements par exemple, σ^i et σ^{i+1}, la sortie $\Phi(q^i)$ du superviseur reste inchangée. Ainsi, on peut représenter un superviseur par un modèle tel que la sortie ne dépend que de l'état. Un superviseur peut donc être modélisé par une *machine de Moore*.

Définition 2.1 : Le superviseur *S* peut être défini par le 6-uplet $S = (V, \Sigma, \xi, v_0, 2^{\Sigma c}, \theta)$ où *V* est un ensemble fini d'états; Σ est l'alphabet d'entrée; $\xi : V \times \Sigma \rightarrow V$ est la fonction de transition d'états; v_0 est l'état initial; $2^{\Sigma c}$ est l'alphabet de sortie; et $\theta : V \rightarrow 2^{\Sigma c}$ est la fonction d'affectation de sortie.

Le superviseur *S* peut être perçu comme une machine à états déterministe qui évolue conformément à une modification de son entrée (sur l'occurrence d'un événement de Σ généré par le procédé) et qui change d'état selon ξ. Pour chaque état *v*, le superviseur *S* fournit en sortie une liste d'événements interdits $\Phi = \theta(v)$. Rappelons que seuls les événements contrôlables peuvent être interdits par la supervision. Ainsi, chaque sortie de *S* est un élément de $2^{\Sigma c}$, où $2^{\Sigma c}$ est l'ensemble de tous les sous-ensembles de Σ_c. **[1 8]**

Remarque 2.2 : Une machine de Moore est un automate déterministe dans lequel la sortie peut prendre des valeurs dans un ensemble fini de grandeurs discrètes. Par conséquent, la sortie d'une machine de Moore n'est plus nécessairement binaire. Ceci confère à la machine de Moore un pouvoir de description supérieur à celui des modèles accepteurs.

Exemple2.1 : Considérons l'exemple classique d'un système manufacturier composé de deux machines identiques : *M*1 et *M*2, et un stock entre ces deux machines. Conformément à la figure 3.4, les deux machines travaillent de façon indépendante, puisent des pièces brutes en amont et déposent les pièces usinées en aval.

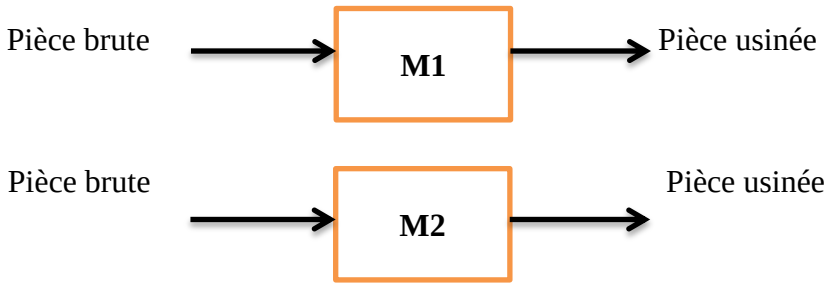


Figure 2.3*- Système manufacturier*

Le procédé est supposé évoluer de façon spontanée en générant des événements

(*générateur d'événements*). Son fonctionnement peut être décrit par un ensemble de séquences d'événements qui constitue un langage formel sur l'alphabet des événements.

Chaque machine de ce procédé peut être modélisée par un automate sans sorties (accepteur). Le graphe de transition d'états de l'automate des machines *M*1 et *M*2 est présenté dans la figure 2.4.

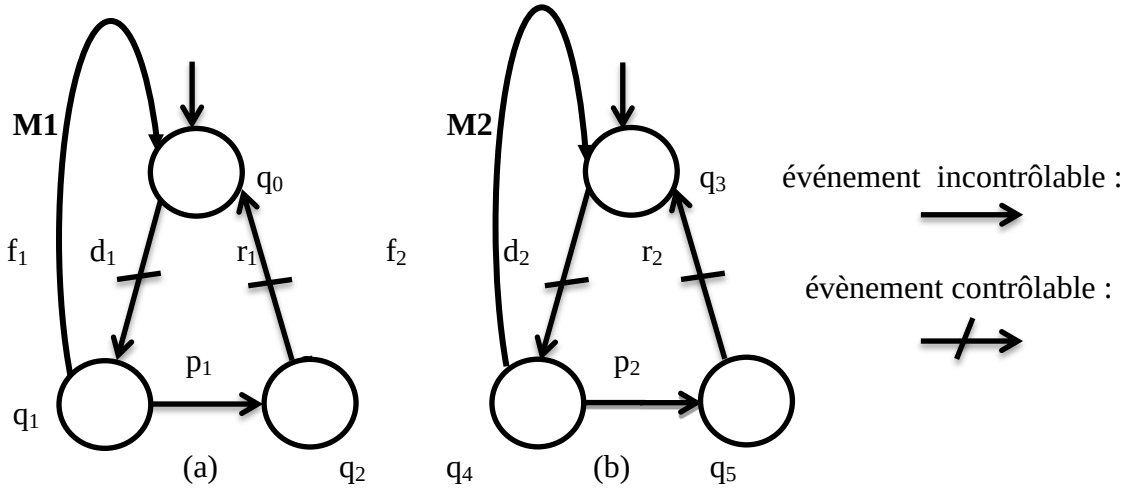


Figure 2.4*- Automate à états finis modélisant le système manufacturier*

Considérons la machine *M*1 (figure 2.4.a). Dans son état initial (état q_0), la machine est à l'arrêt. L'événement d_1 modélise le début du cycle de la machine, l'occurrence de cet événement conduit la machine dans l'état de marche (état q_1).

Dans notre exemple, nous supposons que $d_1$est simultané avec la prise d'une pièce en amont. De même, la fin de cycle (événement f_1) est simultanée avec le dépôt d'une pièce en aval.
Lorsque la machine *M*1 est en marche (état q_1), l'occurrence de l'événement p_1 conduit la machine dans un état de panne (état *q*2). Depuis cet état, la réparation de la machine, ramène la machine dans son état initial. La machine *M*2 possède un fonctionnement similaire à *M*1.

Notons respectivement Σ_1 et Σ_2, les alphabets des machines *M*1 et *M*2. Nous avons :

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

Le fonctionnement du système manufacturier est alors défini sur un alphabet $\Sigma = \Sigma_1 \cup \Sigma_2$.

Nous avons pris la convention de représenter par un arc barré, toute transition associée à un événement contrôlable.

Ainsi, nous supposons que $\Sigma_C = \{d_1, d_2, r_1, r_2\}$ et $\Sigma_U = \{f_1, f_2, p_1, p_2\}$.

Un modèle automate sans sortie de notre système manufacturier peut être obtenu en effectuant la composition synchrone des modèles M1 et M2. Le modèle P défini par

P = M1 $\|_s$ M2est représenté par son graphe de transition d'états dans la figure 2.5

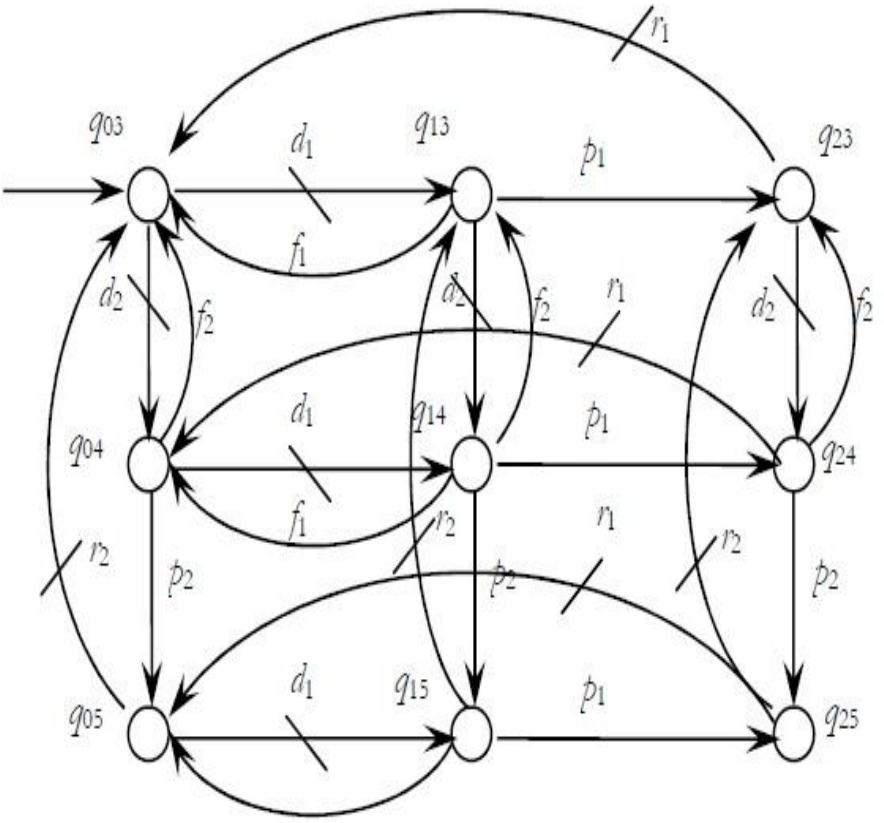


Figure 2.5*- Composition synchrone des deux modèles de machines (modèle du procédé)*

Dans l'automate P, un état est un couple (q_i, q_j), où qi est un état de M1 et q_j est un état de M2. Cet état (q_i, q_j) est noté q_{ij} dans la figure 2.5On considère la spécification de fonctionnement suivante pour notre système manufacturier : le fonctionnement doit respecter la présence d'un stock de capacité limitée à 1, situé entre les 2 machines. Nous supposons donc à présent que les machines travaillent en série. Conformément à la figure 2.6, une pièce doit visiter M1 puis M2. La présence du stock entre M1 etM2 impose que :

(1) la machine M2 ne peut commencer à travailler que si elle peut prendre une pièce dans le stock, c'est-à-dire, si le stock est plein,

(2) la machine M1 ne peut déposer une pièce dans le stock que si celui-ci est vide. Le stock est supposé vide dans son état initial.

Le superviseur S de la figure 2.7 permet de garantir le respect de cette spécification de fonctionnement. Dans cet automate les états v0 et v1 correspondent respectivement aux états: "stock vide" et "stock plein".

Figure 2.6 - Le système manufacturier sous contrainte de stock

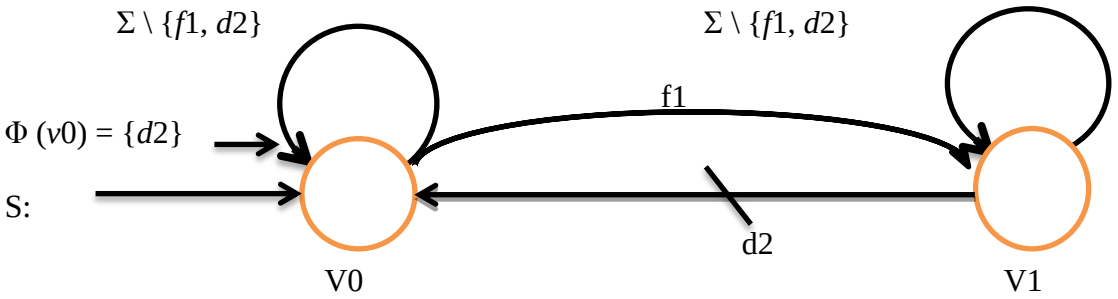


Figure 2.7*- Le modèle automate de la spécification*

Lorsque le stock est vide, l'occurrence de l'événement contrôlable d_2 est interdite (début du cycle de M2).

Sur l'occurrence de l'événement f_1 (fin du cycle de M1 et dépôt d'une pièce dans le stock), l'automate S change d'état et passe dans l'état v1.

Dans cet état, l'occurrence de l'événementf_1 est interdite (fin du cycle de M1).

On appelle *fonctionnement désiré en boucle fermée*, le fonctionnement du procédé couplé à sa spécification.

Conformément à la figure 2.2, un événement peut être généré par le procédé supervisé si, il peut être généré par le procédé P en isolation et s'il est autorisé (non interdit) par le superviseur S. Par extension, une séquence d'événements ω est possible dans le fonctionnementen boucle fermée, si elle est possible dans le procédé en isolation ($\omega \in L(P)$), et si elle est autoriséepar le superviseur ($\omega \in L(S)$). Si on note S/P la machine constituée du procédé P couplé à la spécification S, le langage L(S/P) représente alors le fonctionnement en boucle fermée du système.

Le langage L(S/P) est simplement défini par : $L(S/P) = L(P) \cap L(S)$.

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

P et de S. **[19]**

2.1.3 Contrôlabilité

Considérons maintenant le schéma de contrôle par supervision de la figure 2.2c.

Il y a une question fondamentale qui doit être prise en compte dans ce contexte :

1) Le contrôleur est limité en termes de contrôlabilité des événements possibles dans le procédé, dans ce qui suit, nous allons analyser cette question.

L'existence des événements incontrôlables est une conjoncture usuelle dans la modélisation des systèmes automatisés réels.

Ils peuvent représenter, soit des circonstances fondamentalement inévitables, telles que l'arrivée d'une panne ou la variation de la valeur repérée par un capteur, soit des

actions qui ne peuvent pas être anticipées ou prévues à cause des limitations matérielles, soit des actions qu'on a choisi de désigner comme incontrôlables, telles que les événements ayant une très haute priorité ou le top d'un horloge. Quel que soit le cas de modélisation, le problème du point de vue du contrôle est le même : il faut ajouter des contrôles supplémentaires, car tout événement qui ne peut pas être interdit doit être toujours considéré comme faisant partie du comportement en boucle fermée.

L'incontrôlabilité peut alors très rapidement compliquer la recherche d'un contrôleur garantissant un fonctionnement désiré pour le système supervisé. Si dans le cas des exemples très simples il est, malgré tout, possible de faire appel à l'intuition pour calculer un contrôleur satisfaisant même dans l'éventualité de l'incontrôlabilité, des méthodes formelles sont nécessaires si l'on souhaite résoudre ce problème pour des modèles plus complexes. Pour ces raisons, la contrôlabilité est un concept clef de la théorie R&W, constituant la base même de la synthèse de contrôleur.

Dans ce contexte, on peut définir une partition générale sur l'alphabet du procédés el on la contrôlabilité de ses événements :

Notation2.1: Étant donné l'alphabet Σ d'événements d'un procédé P, nous pouvons écrire: $\Sigma = \Sigma_C \cup \Sigma_U$, où Σ_C *et* Σ_U désignent respectivement les ensembles d'événements contrôlables et incontrôlables sur Σ. Dorénavant, Σ_C sera appelé *l'alphabet des événements contrôlables*, et

Σ_U *l'alphabet des événements incontrôlables*.

La théorie générale de la commande des SED dans l'approche R&W est basée sur les langages formels et les modèles automates à états finis.

La contrôlabilité est une des propriétés de base pour cette approche, ayant des implications sur l'existence du contrôleur ; il faut alors l'étudier en se rapportant à ce cadre.

Nous parlons alors de *la contrôlabilité d'un langage* :

Un langage K *est dit contrôlable par rapport à un langage* L *si :*

$\overline{K}\Sigma_U \cap L \subseteq \overline{K}$, *où* $\overline{K}$ *est la préfixe-clôture de* K

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

2.1.3.1 Propriétés de la contrôlabilité

Étant donnés deux langages L1 et L2, les affirmations suivantes peuvent être énoncés à l'égard de la contrôlabilité des opérations impliquant L1 et L2 :

- Si L1 et L2 sont contrôlables, alors leur union, L∪ = L1 ∪ L2, est aussi contrôlable.
- Si L1 et L2 sont contrôlables, leur intersection, L∩ = L1 ∩ L2, n'est pas nécessairement contrôlable.

- Si L1 et L2 sont contrôlables et si $\overline{L1 \cap L2} = \overline{L1} \cap \overline{L2}$ (i.e. L1 et L2 sont des langages non-conflictuels), alors l'intersection $L\cap = L1 \cap L2$ est aussi contrôlable.
- Si L1 et L2 sont contrôlables et préfixe-clos, alors leur intersection, $L\cap = L1 \cap L2$, est aussi contrôlable et préfixe-close. **[01]**

2.1.3.2 La condition de contrôlabilité

Si on considère K comme le langage de la spécification et L comme le langage du procédé, avec $K \subseteq L$, $K \neq \phi$, on dit que le langage de la spécification est contrôlable par rapport au langage de procédé si :

$\forall$ mot $\omega \in K, \forall \sigma \in \Sigma_U : \omega\sigma \in L \longrightarrow \omega\sigma \in K$. **[01]**

2.1.4 Synthèse du contrôleur

Ramadge et Wonham ont introduit la notion de contrôlabilité pour les SED **([09])** afin de caractériser les langages contrôlés d'un procédé dans le cadre de la théorie générale de la supervision. L'existence d'un contrôleur qui assure un comportement souhaité pour le système en boucle fermée est intimement liée au concept de contrôlabilité des langages, i.e. la possibilité d'imposer que le langage généré par le procédé couplé au contrôleur appartienne à un certain langage de spécification.

Etant donnés un procédé P et une spécification de fonctionnement S_{Spec}, on souhaite synthétiser un superviseur S de façon à ce que le système en boucle fermée S/P, respecte la spécification. Cela signifie qu'il faut déterminer le langage $L(P) \cap L(S_{spec})$.

Ce langage appelé fonctionnement désiré correspond à l'ensemble des séquences qui peuvent être générées par le procédé et qui sont tolérées par la spécification, ce langage est noté L_D. Il n'est pas toujours possible (prise en compte d'événements incontrôlables Σ_U) de restreindre, par la supervision, le fonctionnement d'un procédé à n'importe quel sous-langage de ce fonctionnement. L'existence d'un superviseur S tel que $L(S/P) = L_D$ réside dans le concept de contrôlabilité.

2.1.4.1 Algorithme de Kumar

A partir des modèles automates *P* d'un procédé et S_{spec} d'une spécification de fonctionnement, l'algorithme de Kumar permet de vérifier la contrôlabilité du langage de spécification $L(S_{spec})$. De plus, dans le cas où le langage $L(S_{spec})$ n'est pas contrôlable, cet algorithme permet de synthétiser un modèle automate du langage suprême contrôlable du fonctionnement désiré $supC(L_D)$ (figure 2.8).

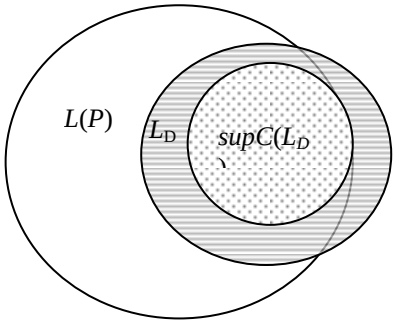


Figure 2.8- Langage suprême contrôlable d'un fonctionnement désiré

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

- ✓ **Pas 1.** On construit le composé synchrone D de P et de S_{spec}, c'est-à-dire, $D = P \| S_{spec}$. Le langage $L(D)$ sera noté L_D.
- ✓ **Pas 2.** On détermine l'ensemble des *états interdits*.
- ✓ **Pas 3.** On détermine l'ensemble des *états faiblement interdits*.
- ✓ **Pas 4.** On supprime de D l'ensemble des états interdits ainsi que l'ensemble des états faiblement interdits (ainsi que les transitions associées à ces états). On supprime de D l'ensemble des états non accessibles, c'est-à-dire, tout état (q, v) tel qu'il n'existe pas de chemin permettant de rejoindre (q, v) depuis l'état initial.

Appliquons cet algorithme sur un exemple inspiré de **[18].** La figure 2.9 donne l'automate final et l'ensemble des états interdits.

Exemple 2.2 : Par application de l'algorithme de Kumar pour notre exemple, nous trouvons les états interdits suivants : {(q_{13},v_1), (q_{14},v_1), (q_{15},v_1) }
Remarque 2.3 : Dans cet exemple il n'y a pas d'états faiblement interdits.

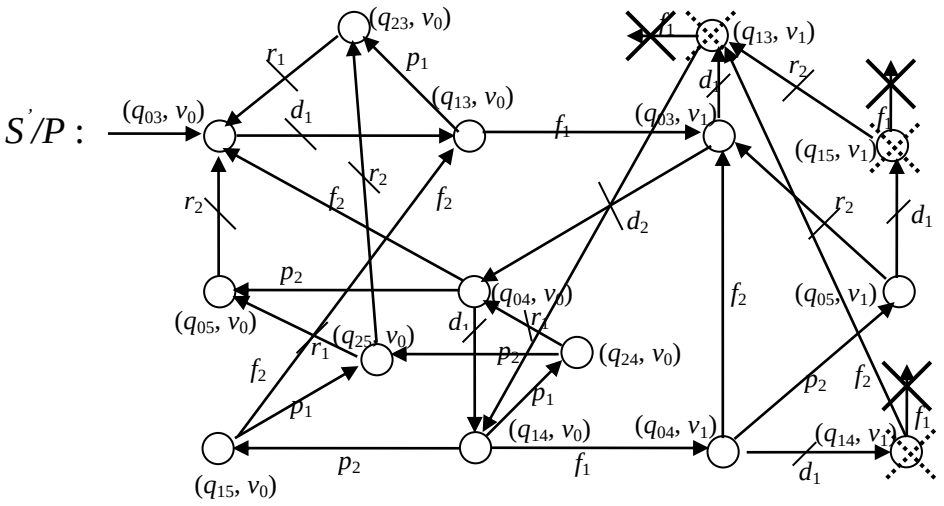


Figure 2.9- *Modèle automate du système supervisé avec des états interdits*

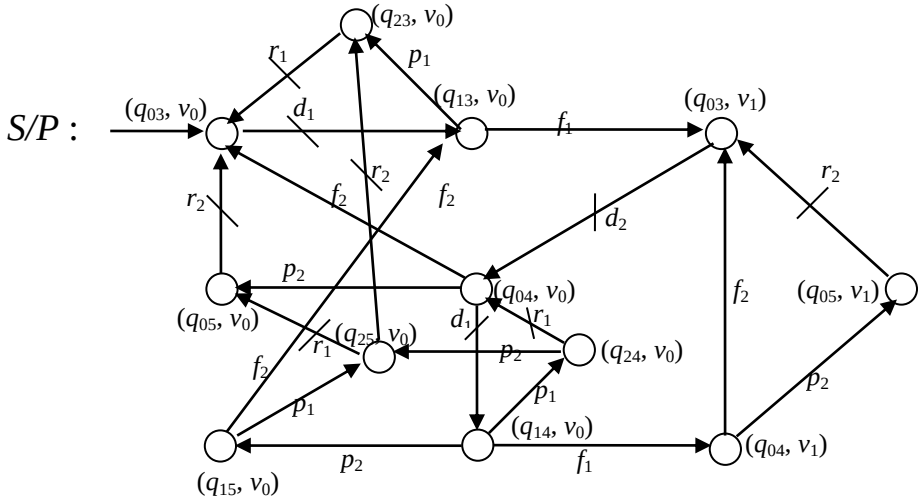


Figure 2.10- *Modèle final de système supervisé sans états interdits*

Si le modèle du procédé comporte n états et le modèle de la spécification comporte m états, alors l'algorithme de Kumar permet de synthétiser un superviseur qui comporte au plus n.m états. Ainsi, la taille du superviseur est souvent bien plus grande que celle du procédé.

Il résulte que dans bien des cas, l'explosion combinatoire due à l'utilisation de modèles automates rend difficile la synthèse de superviseurs. Un grand nombre d'extensions de la théorie de R&W visent à résoudre le problème de la réduction de la taille du superviseur **[2 1].**

2.2 – Contrôle des systèmes à évènements discrets temporisés

2 .2.1 - Introduction sur les SEDT :

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

Considérons, par exemple, une machine qui peut être dans états : *arrêt, marche et panne*. On suppose qu'il peut y avoir l'occurrence de quatre événements : *début traitement, traitement achevé, panne* et *réparation. Ces* événements sont étiquetés par les symboles *a, b, c* et *d* respectivement. Une évolution possible de ce système est présentée dans la figure 2.11.

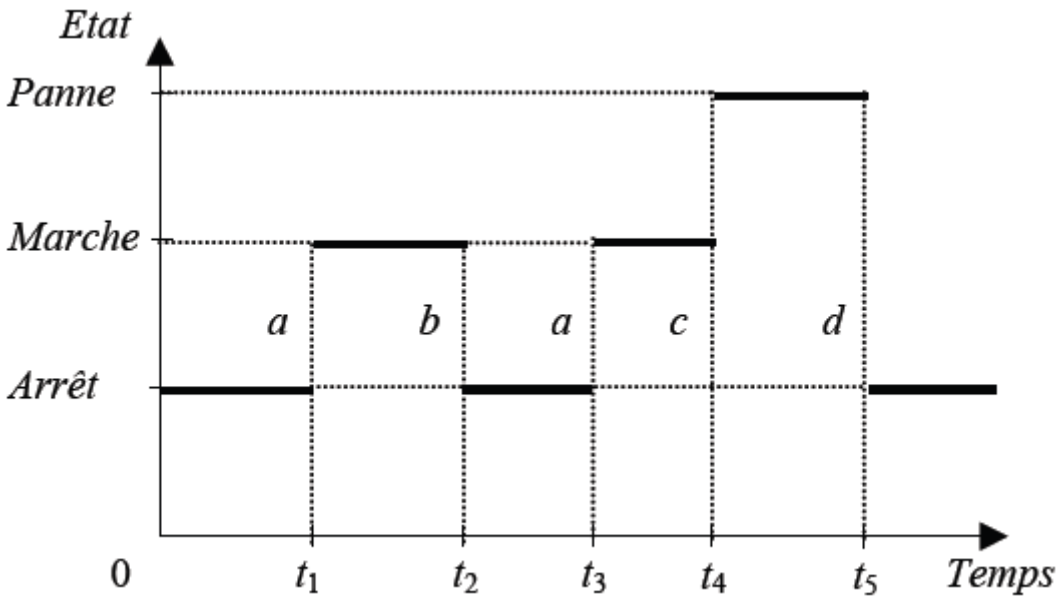


Figure 2.11 *– Evolution de l'état de la machine*

Initialement la machine est en état d'*arrêt.* Suite à l'arrivé de l'événement *a*a l'instant t_1, la machine passe dans l'état *marche*. De façon similaire, le système évolue aux instants t_2, t_3, t_4 *et* $t_5$sur l'occurrence des événements *b, a, c* et *d.*

Les domaines d'application des SED sont nombreux. Différents aspects du comportement d'un système peuvent être considères, selon l'application envisagée. Par conséquent, différents outils pour la modélisation et l'analyse des SED ont été développés. L'évolution d'un SED est caractérisée par l'occurrence des événements. Selon la manière de modéliser l'arrivée des événements, on peut classifier les modèles temporisés des SED en trois catégories : modèles logiques, modèles temporisée nom-stochastiques et modèles temporisés stochastiques.

Dans certaines applications, l'information temporelle est essentielle et doit être prise en compte explicitement par le modèle. Les modèles qui ont cette caractéristique sont appelés *temporisés.* Il s'agit des modèles ou le temps est déterministe ou (réseaux de pétri temporels, automates temporisés) et des modèles ou le temps est aléatoire (chaines de Markov, réseaux des files d'attente).

Les modèles ou le temps est déterministe ont le même principe de fonctionnement que les modèles logique. Cependant, les instants d'occurrence des événements sont explicitement pris en compte. Les SED représentés par ce modèles sont appelés ***systèmes a événements discrets temporisés*** (SEDT).

Les modèles ou le temps est aléatoire sont caractérisés par des distributions de probabilité associées aux instants d'arrivée des événements.

Dans la suite, nous nous intéressent seulement au contrôle des systèmes à événements discrets temporisés. **[23]**

2.2.2- Synthèse de contrôleur des Systèmes a événements discrets (SEDT)

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

2.2.2.1 - Synthèse de contrôleur en temps discret :

Dans **[24]** les auteurs ont proposé une approche pour la synthèse de contrôleur des systèmes à évènements discrets temporisés (SEDT) se basant sur les travaux de Ramadge etWonham, auxquels les contraintes temporelles sur les dates d'occurrence des évènements sont ajoutées. Brandin et Wonham ont eu pour objectif d'élargir la théorie classique de la commande supervisée pour les systèmes à événements discrets afin de l'appliquer à des systèmes temporisés.

La théorie de Ramadge et Wonham repose sur la manipulation des langages. Pour se ramener à cette théorie, le temps est discrétisé et sa prise en compte va se réaliser par la création d'un nouvel événement, appelé *tick*, qui sera présent dans l'écriture du langage de l'automate temporisé. Le modèle temporisé est construit à partir d'un modèle logique, appelé graphe d'activités, représenté sous la forme d'un automate discret. Ce modèle décrit la succession logique des commutations entre les différents états du système. Les transitions du graphe sont étiquetées par des événements discrets. Le graphe d'activités est alors un système à événements discrets, non temporisé, représenté par un automate fini déterministe noté $Aact = (Qact, \Sigma act, \delta act, qact,0)$.

Comme pour les automates à états finis, chaque événement du procédé est instantané et il est exécuté à n'importe quel instant t du temps réel.

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

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

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

A chaque événement $a \in \Sigma_{act}$ on associe un intervalle $[l_a, u_a]$, l_a et $\in N$ et $u_a \in N \cup \{\infty\}$.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 $0 \leq ua < \infty$. L'ensemble des événements prochesest noté Σspe.

- Un événement a est appelé lointain si $ua = \infty$. L'ensemble des événements lointains estnoté Σrem.

Par conséquent, l'ensemble des événements Σact est partitionné en deux sous-ensembles disjoints :

$\Sigma act = \Sigma spe \cup \Sigma rem$

A chaque événement a on associe une temporisation ta, qui mesure le temps écoulé depuis sa dernière validation. Pour modéliser les contraintes temporelles dans le

modèle du comportement d'un SED on utilise un nouvel automate $A = (Q, \Sigma, \delta, q_0)$ dérivé de l'automate A_{act}. Cet automate est défini de la façon suivante :

- Q est l'ensemble des états pour lequel :

$t_a = \begin{cases} [0\ ua] \ si\ a \in \Sigma spe. \\ [0\ la] \ si\ a \in \Sigma rem. \end{cases}$

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

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

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

- q_0 est l'´état initial.

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

Exemple 2.3: Considérons un SED qui a un seul état logique. Supposons que les évènements qui peuvent se produire dans ce système sont *(a, 1, 1) et (b, 2, 3).* Le comportement logique de ce système est modélisé par l'automate A_{act} en figure 2.12. **[2 3]**

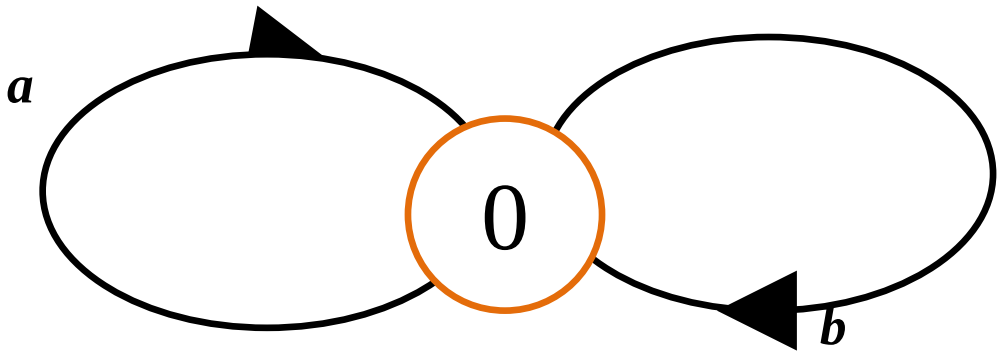


Figure 2.12- *Automate A_{act}*

Le comportement temporel du système est modélisé par l'automate A = (Q, Σ, δ, q_0) où :

- Q = { 0 } × [0,1] × [0,1],
- Σ = {*a*, *b*, *tick*},
- δ = est la fonction de transition,
- q_0 = {0},

Cet automate est représenté dans la figure 2.13. Chacun de ces états est caractérisé par une valeur particulière des temporisations associées aux évènements. Il faut noter que la prise en compte explicite du passage du temps a engendré l'augmentation du nombre d'états. Ainsi, cet automate a 8 états et 11 transitions, contrairement à l'automate modélisant le comportement logique du système qui a seulement un état et deux transitions.

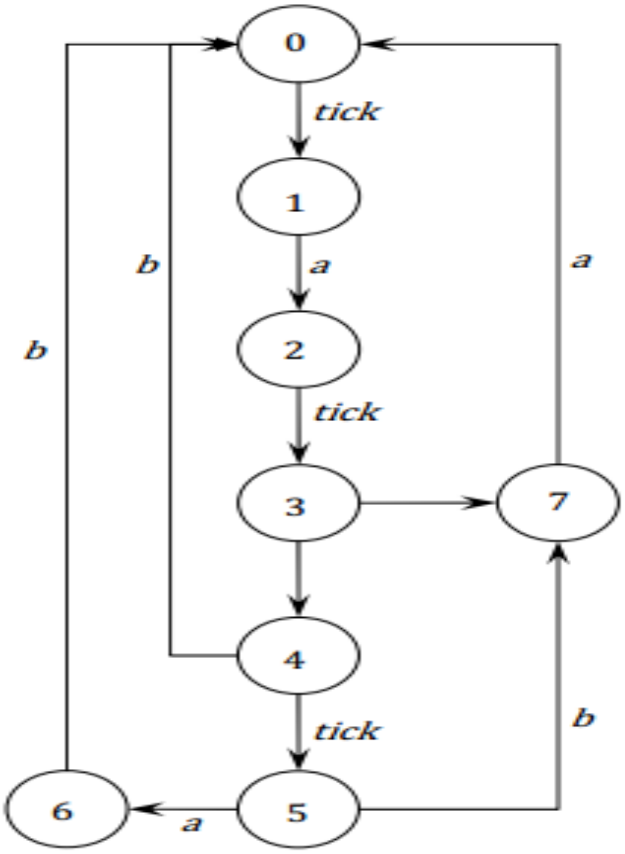


Figure 2.13. *Automate A ;*

De la même manière que pour les SED, certains évènements sont contrôlables tandis que d'autres ne le sont pas. Ainsi, l'ensemble des évènements Σ est partagé en trois sous-ensembles disjoints : $\Sigma = \Sigma_C \cup \Sigma_U \cup \{tick\}$

Où:

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

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

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

$\Sigma u \subseteq \Sigma spe \cup (\Sigma spe - \Sigma c)$

Une autre catégorie d'évènements, essentielle dans la synthèse de contrôleur des SEDT, est représentée par les évènements forçables, un évènement est considéré comme étant forçable
s'il peut se produire spontanément ou être forcé par un système extérieur tout en respectant la contrainte temporelle associée à sa date d'occurrence. Dans l'approche de Brandin et Wonham, un événement forçable peut préempter l'occurrence de l'événement *tick*. Ainsi le superviseur peut forcer l'occurrence d'un événement avant que l'horloge atteigne une certaine valeur. L'ensemble des évènements forçables est noté Σ_{for}. Il n'y a aucune relation entre l'ensemble des évènements forçables et les ensembles des évènements contrôlables et incontrôlables. Un événement contrôlable peut ne pas être forçable dans le sens ou on peut interdire son occurrence, mais on ne peut pas forcer son exécution. De même, certains évènements incontrôlables peuvent être forçables. Intuitivement, le passage du temps ne peut pas être forcé ($tick \notin \Sigma_{for}$), mais il peut être préempté.

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

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

Formellement, un contrôleur est défini par une fonction :

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

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

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

$$S(\omega) \supseteq \begin{cases} \Sigma u \cup \{tick\} si\ S(\omega) \cap \Sigma_{for} = \phi \\ \Sigma u \qquad si \quad S(\omega) \cap \Sigma_{for} \neq \phi \end{cases}$$

De la même manière que dans la théorie de la commande par supervision des SED, les évènements incontrôlables sont toujours autorisés par le superviseur. Par contre, lorsque parmi les évènements éligibles il y a au moins un événement forçable, le contrôleur peut forcer son exécution avant l'occurrence d'un nouveau top d'horloge (*événement tick*). C'est pour cela que dans cette approche, le système de commande est appelé contrôleur.**[23]**

Le comportement en boucle fermée, le concept de contrôlabilité d'un langage, ainsi que le calcul du langage suprême contrôlable sont identiques à ceux de la théorie de R&W. Il est ainsi possible de déterminer le contrôleur maximal permissif.
Même si on est ramené ici à l'approche classique R&W par discrétisation du temps, les nombreuses définitions d'événements la rendent difficile à comprendre. Les notions d'événements proches, lointains, forçables, contrôlables, incontrôlables sont difficiles à classer. Du point de vue de l'automaticien et de manière plus simple, on peut distinguer deux types d'événements comme dans la théorie classique :

1. les événements contrôlables pour lesquels on peut modifier la date d'occurrence dans leur intervalle d'existence (quelque soient ses bornes). Cela peut correspondre par exemple au démarrage d'une machine qui doit avoir lieu dans l'intervalle *[a, b]*. Le contrôle peut alors avoir à réduire cet intervalle pour satisfaire une spécification. Un événement contrôlable peut être forcé.

2. les événements incontrôlables pour lesquels on ne peut rien modifier. Cela peut correspondre par exemple à un intervalle d'occurrence de la fin de fabrication d'une pièce et traduit alors une incertitude sur cette fin de tâche.

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

2.2.2.2- Synthèse de contrôleur en temps continu :

Plusieurs approches de synthèse de la commande ont été proposées pour pallier au problème de l'explosion combinatoire du nombre d'états engendrée par la nature discrète du temps. Ces approches, basées principalement sur l'outil automate temporisé, considère que le temps évolue d'une manière continue. Nous nous intéressent seulement au contrôle des systèmes à événements discrets temporisés. Dans cette section nous présentons une approche de contrôle des SEDT à base d'un automate temporisé qui est en relation directe avec notre travail.

Dans cette approche **[25]** l'auteur propose une méthode pour la synthèse de contrôleur

en s'appuyant sur l'outil automate temporisé.

Un automate temporisé est un automate fini muni d'un ensemble de variables continues par morceaux appelées horloge. Ces variables mesurent l'écoulement du temps. Lorsque le système séjourne dans un sommet, chaque horloge xa une dynamique continue décrite par l'équation $\dot{x} = 1$ chaque sommet on associe un prédicat sur la valeur des horloges vérifie l'invariant associé. Le franchissement d'une transition de l'automate est instantané. A chaque transition on associe une condition de franchissement, appelée *garde,* et une affectation. Une transition peut être franchie depuis un sommet si sa *garde* est vérifiée par la valeur des horloges. Les *gardes* modélisent les contraintes temporelles imposées sur l'évolution du système. L'affectation associée à une transition désigne les horloges mise à zéro par son franchissement. **[23]**

Considérons par exemple, l'automate temporisé dans la figure 2.14. Cet automate a deux horloges $x_1$et x_2, qui sont incrémentées dans les sommets. L'invariant du sommet L_0 est$x_2 \leq 1$. La *garde* associée a la transition de sortie du sommet L_0est$a \wedge x_2=1$. Cette transition peut être franchie depuis L_0 lorsque l'événement *a* arrive et l'horloge x_2 a la valeur 1. L'affectation associée à cette transition est $x_2 = 0$. Ainsi, le franchissement de cette transition entraine la mise à zéro de l'horloge x_2.

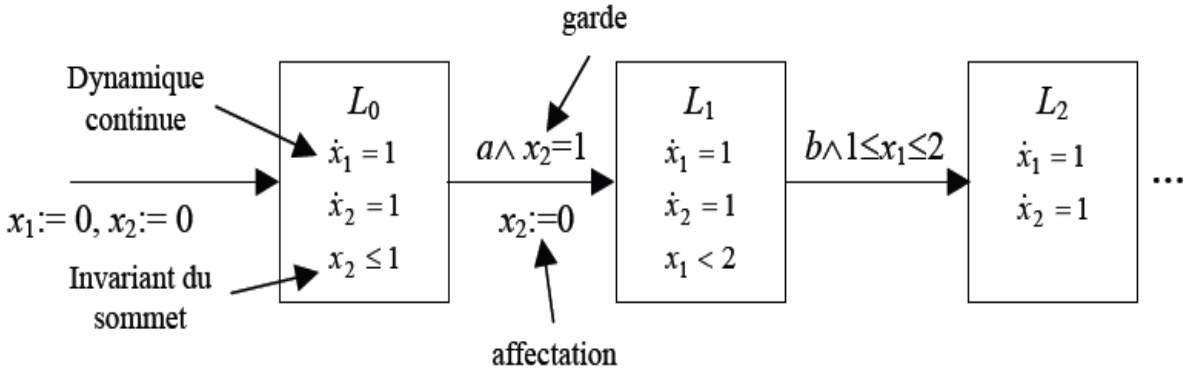


Figure 2.14 *Automate temporisé*

Dans la proche de synthèse de contrôleur par détemporisation, le procédé ainsi que les spécifications sont modélisés par des automates temporisés. La première étape de la synthèse de contrôleur consiste à effectuer la composition synchrone de deux automates. Dans le cas des automates temporisés, pour chaque événement commun il faut prendre en compte les contraintes temporelles sur la date d'occurrence. Ainsi, lors la composition synchrone des automates, la contrainte temporelle associée à un événement commun est la conjonction des contraintes temporelles qu'il doit vérifier dans chacun des automates temporisés. Cette dépendance temporelle mutuelle peut engendrer l'apparition des incohérences temporelles.

Il y a deux catégories d'incohérences temporelles

Les incohérences temporelles de la première catégorie apparaissent lorsqu'on

atteint un sommet avec une valeur des horloges telle qu'aucune garde de ses transitions de sortie ne sera jamais vérifiée. La conséquence de ce type d'incohérence temporelle est le blocage dans un sommet de l'automate.

Exemple 2.4. Considérons la partie d'automate temporisé présentée dans la figure 2.15. Par simplification, nous n'avons pas représenté l'évolution des horloges dans les sommets de l'automate. La valeur de l'horloge $\boldsymbol{x}_1$a l'entrée dabs le sommet L_2 est supérieure à 5, tandis que la garde de l'arc de sortie de la transition de L_2a L_3est $\boldsymbol{x}_1$= 3. Ainsi, cette transition ne pourra jamais être franchie et le système reste bloqué dans le sommet L_2.

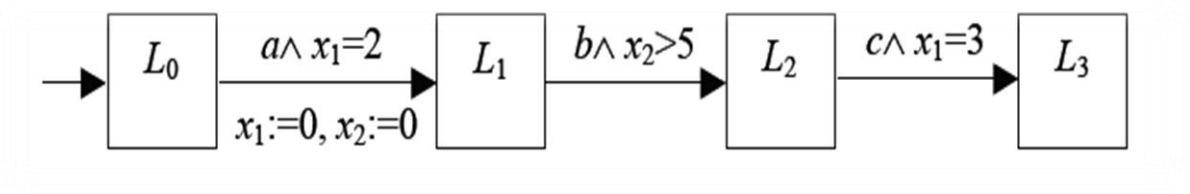


Figure 2.15*.- Première type d'incohérence temporelle*

Les incohérences temporelles de la deuxième catégorie apparaissent lorsque la date de franchissement au plus tôt d'une transition de sortie d'un sommet est plus grande que la date de franchissement au plus tard d'une autre transition de sortie du même sommet. La conséquence de ce type d'incohérence temporelle est le fait qu'une transition de sortie du somme considéré ne sera jamais franchie. Ainsi, elle complique inutilement la représentation de l'automate temporisé.

Exemple 2.5. Considérons la partie d'automate temporisé présentée dans la figure 2.16. La transition de L_1 vers L_3 ne peut jamais être franchie par ce que pendant le séjour du système dans L_1, l'horloge $\boldsymbol{x}_1$ attient toujours la valeur 2 avant que $\boldsymbol{x}_2$ = 6.

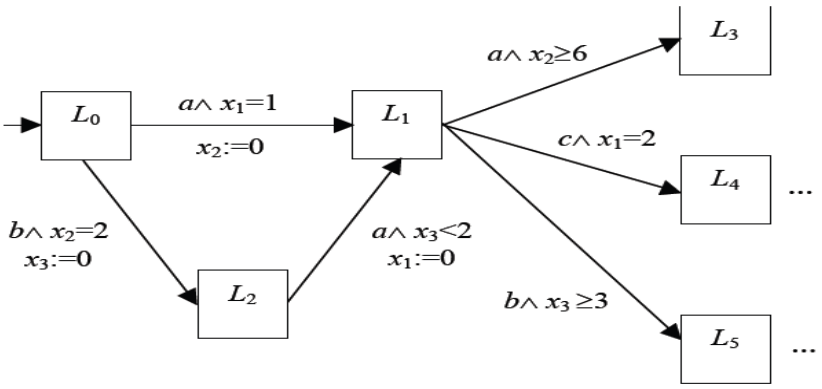


Figure2.16 *Deuxième type d'incohérence temporelle*

La détection des incohérences temporelles est effectuée par une technique de manipulation des intervalles temporels. Cette opération est réalisée à titre préventif avant de commencer la synthèse du contrôleur.

La première étape consiste à détemporisé le modèle. La détemporisation consiste à transformer l'automate temporisé en un automate de régions **[26]**. Dans un automate de régions l'information temporelle est incluse dans les sommets et non dans les transitions. Par conséquent, l'automate de régions peut être considéré comme un automate non temporisé, d'où le lien avec la théorie de base. En suite en applique la théorie de Ramdage et Wonham pour réaliser la synthèse du contrôleur.

2.3 Conclusion

Malgré que La synthèse de la supervision pour les SED soit basée sur la théorie de R & W, il existe encore des difficultés d'implantation que l'on peut regrouper selon les quatre points suivants :

Les événements forcés : Dans la théorie de la supervision, il est supposé que le procédé génère des événements de façon spontanée. Le superviseur peut seulement interdire des événements contrôlables, et ne peut en aucun cas forcer des événements dans le procédé. Ce problème a été résolu par l'interdiction d'une nouvelle catégorie d'événements. Il s'agit des événements forçables. Cette catégorie désigne les événements qui peuvent se produire de façon instantanée ou être forcés par un système extérieur.

L'implantation du superviseur : Le passage de la synthèse à l'implantation n'est pas systématique. En effet, la procédure de synthèse donne un modèle de superviseur qui n'est pas directement utilisable pour faire de la commande.

Les difficultés de modélisation : La synthèse du superviseur requiert la connaissance des modèles automates du procédé et de ses spécifications. Une telle modélisation peut s'avérer parfois difficile et ce même pour des exemples assez simples.

La complexité : Le problème classique de l'explosion combinatoire de l'espace des états du système présente souvent des difficultés d'implantation. **[18]**

Dans la plus part des cas on dispose d'une information supplémentaire sur la date d'occurrence des événements. La prise en compte de cette information a fait l'objet de travaux sur la synthèse Du contrôleur des SEDT, pour élaborer des lois de contrôle moins contraignantes. L'étude de ces approches nous a permis de faire la classification suivante :

1. *Les approches à temps discret* modélisant le passage de temps par l'occurrence d'un événement. Ces approches sont basées sur la théorie de R & W et fournissent des lois de commandes efficaces. Par contre, elles présentent l'inconvénient de fournir des modèles d'analyse de taille importante.
2. *Les approches à temps continu* permettent de contourner le problème de l'exploitation du nombre d'états et de l'approximation engendrée par la discrétisation du temps. Plusieurs approches à temps continu ont été étudiées. Pour rester dans le cadre de notre travail on a choisi L'approche du contrôle par détemporisation du temps. Cette approche utilise l'outil automate temporisé pour la synthèse du contrôleur.

Chapitre 3

Des applications Sur Les SED

3.1 Introduction

Dans ce chapitre nous décrivons le logiciel PHAVer, un outil d'analyse des systèmes à évènements discrets développé par Goran Frehse, du laboratoire verimag de Grenoble, France. Au début nous nous intéressons à l'outil PHAVer. A cause du fait que ce logiciel n'a pas été utilisé en avant dans le laboratoire d'Automatique de Tlemcen, nous présenterons aussi comment l'installer, ses commandes, ses versions et ses outils annexes.

Ensuite on se propose de mettre en pratique cet outil, afin de simuler la surveillance des systèmes.

Les simulations ont été faites pour plusieurs exemples.

3.2 Présentation du logiciel PHAVer

Logiciel PHAVer (Polyhedral Hybrid Automaton Verifier) est un outil de vérification des systèmes.

L'auteur de PHAVer a essayé de le faire autant « user friendly » que possible, en gardant le parser simple La syntaxe a emprunté extensivement depuis les créateur de HyTech [tAH 96] puis leur langage est intuitivement compréhensible. La section suivante décrit la syntaxe de PHAVer, version 0.35 **[31]** en représentant les automates, les localités/états, relations, etc. et aussi quelque description concise des commandes pour faire l'analyse de l'automate.

La description d'un automate peut être faite à l'aide des opérants suivants

1. **identificateurs**

Un *identificateur* est une lettre plus n'importe quelle combination de lettres, chiffres et caractères joindre les identificateurs des localités appartenant aux automates composés. Un nombre peut être donné dans un format virgule mobile, e.g.3.14 ou 6.266e-34 ou comme une fraction, e.g.9/5.

Notez que les nombres sont représentés comme des rationnels exacts et aucune conversion vers le format virgules mobile binaire n'est faite.

2. Constante :

Les constantes sont définies de la manière identificateur :=expression, ou expression est n'importe quelle combinaison d'expressions, identificateurs et nombres avec +,-, /, *, (,).

3. Structure de données

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

- Formules linéaires

Une expression linéaire est spécifiée sur une collection arbitraire de variables, nombres et constantes qui peuvent être combinés en utilisant +,-, /, *, (,), tant qu'elle produit une expression linéaire sur ces variables.

- Ensemble d'états symboliques

Un état symbolique est une combinaison d'un nom d'une location et une formule linéaire, unies par &, e.g. start & x>0&y==0. Un ensemble d'états symboliques d'un automate aut est assigné à une variable dans la forme identificateur=aut.{ensemble d'états symboliques}.

- Relations symboliques

Elles sont obtenues par les algorithmes de relation de simulation.

- automates

Ils vont être traités en ce qui suit.

Pour déclarer un automate il y a plusieurs aspects à considérer. On va les expliquer à travers un exemple :

```
Automaton aut
        Contr_var : var ident, var ident,… ;
        input_var : var ident, var ident,… ;
        parameter : var ident, var ident,… ;
        synclabs : lab_ident, lab_ident,… ;
        loc loc_ident : while invariant wait{derivee};
            when guard synclabel_ident do {trans_real} goto loc_ident;
            when guard synclabel_ident goto loc_ident;
            while …
        initially : initial_states;
end
```

- initialisation

L'automate est déclaré en haut de la description, à l'aide du mot automaton. La dernière ligne de l'automate doit être end, ce qui signifie que ce qui se trouve entre ces deux mots clés constitue un seul automate. Avant la dernière ligne on doit aussi préciser l'état initial de l'automate avec l'instruction initially. L'état initial est caractérisé par le sommet initial et les valeurs initiales des variables.

- Variables

Toutes les variables doivent être précisées justement après la définition de l'automate, en commençant sur la deuxième ligne, à l'aide des mots state_var (les versions du logiciel 0.2.2 et 0.33) ou bien contr_var depuis la version 0.35. il y a aussi l'argument input_var qui désigne les entrées dans le systèmes.

4. Sommets

La déclaration des sommets est effectuée par l'instruction loc. l'invariant et le garde sont des formules linéaires entre les variables d'état et d'entrée et les paramètres. La définition de la dérivée dépend de la dynamique :

- Pour une dynamique linéaire, c'est une formule linéaire convexe entre les variables d'état .e.g ., $0 \leq x' \ \& \ x' < 1$ pour $\dot{x} \in [1,0)$.
- Pour une dynamique affine, c'est une formule linéaire convexe entre les variables et leurs dérivées. .e.g ., $x' == -2 * x$ pour $\dot{x} = -2x$.
-

5. Transition

Une transition est spécifiée par la déclaration when..goto. on doit avoir toujours une étiquette de synchronisation associée à la transition. Une formule linéaire trans_rel spécifie la relation de saut après une déclaration do. Les variables d'état qui ne sont pas changés par la transition doivent être spécifiés explicitement dans la relation de saut, e.g ., $x' == x \ \& y' == y$. La mise à 0 d'une variable sera définie par $z' == 0$.

3.3 Installation

Le logiciel PHAVer est téléchargeable à l'adresse http://www.cs.ru.nl/~goranf/. Il y a deux variantes disponibles parce qu'il peut être utilisé sous deux systèmes d'exploitation : linux et Windows. Pour la variante Windows, on doit installer aussi un émulateur de Linux, qui s'appelle Cygwin. Il est téléchargeable sur internet.

- Remarque : il est nécessaire que les scripts soient dans le même dossier que l'application PHAVer.

3.4 Outils annexes

Pour construire des graphes bidimensionnels sous Linux on peut utiliser la commande graph du progiciel plotutils, disponible à

http://www.gnu.org/software/plotutils/.

Sous Windows/ Cygwin il existe un script Matlab, disponible à http://www.cs.ru.nl/~goranf/.

Pour obtenir la mémoire utilisée pendant le calcul des états atteignables avec PHAVer on utilise mem-time, un logiciel sous Linux, disponibles à

http://www.update.uu.se/~johand/memtime/ .

3.5 Commandes pour description et simulation

Pour les besoins de notre projet, les commandes suivantes sont d'intérêt. Pour plus de détails le guide de l'utilisateur PHAVer est **[32].**

- **&** : les automates sont composés par l'ampersand, e.g. compAut=aut1&aut2.
- **.reachable** : calcule l'ensemble des états qui sont atteignables dans l'automate en partant depuis les états initiaux et en faisant une analyse en avant.
- **.print('fichier', arg)** : écrit une représentation de l'automate dans le fichier 'fichier' ; la variable dénote le format : arg-0(textuellement), 1 (séquence de formules linéaires), 2 (séquence de vertices, utilisable pour construire des graphes).
- **.reverse** : inverse la causalité dans l'automate ; utilisable pour implémenter l'atteignabilité en arrière, en inversant l'automate et puis e faisant de nouveau l'analyse en avant.
- **.initial_states** : remplace les états initiaux de l'automate.
- **.intersection_assign** : construit l'intersection de deux régions.
- **.différence_assign** : fait la différence entre deux régions.

3.6 Modélisation et simulation

Dans ce qui suit nous allons nous concentrer sur la modélisation des certains systèmes de productique et sur le calcul des espaces atteignable exact et acceptable, nécessaire pour la description des automates pour la surveillance. On va discuter la modélisation en automate à chronomètre et les résultats de simulations.

3.6.1 Application N1 : Modèle du convoyeur non-bouclé

Un des équipements les plus utilisés dans un atelier de production est le convoyeur. Par exemple on pourrait demander de transférer une palette sur le convoyeur.
Le convoyeur est destiné à transférer, à vitesse constante, une palette du point A au point B comme dans la figure suivante :

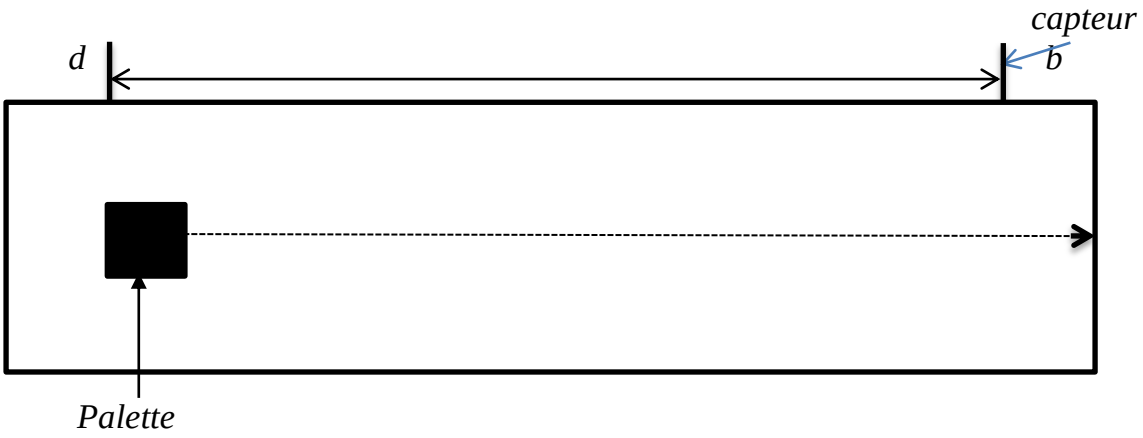


Figure 3.1- convoyeur

Les spécifications imposent que la durée normale de Transfer jusqu'au bout de convoyeur soit dans l'intervalle [10,11] u.t. cette durée peut être dépassée, mais seulement avec un maximum de 1-2 secondes.

On appelle intervalle normale de bon fonctionnement l'intervalle [10,11] u.t. et intervalle de fonctionnement acceptable l'intervalle [10,12] u.t.

Nous avons modélisé le comportement de ce système par l'automate à chronomètre suivant :

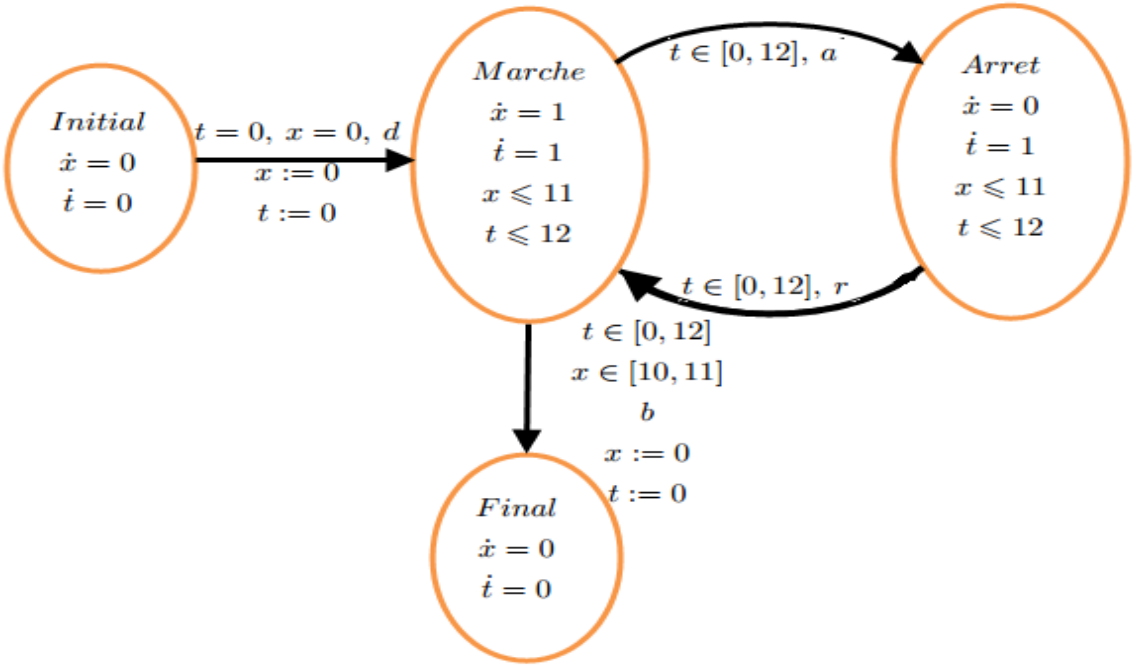


Fig. 3.2- L'automate à chronomètres pour le système du convoyeur

Cet automate à chronomètre est composé d'un ensemble de sommet {*initial, marche, arrêt, final*} reliés par des transitions. Il est équipé de deux horloges : x, qui mesure le temps écoulé pendant le transfert physique de la palette depuis le démarrage de la tâche et t, qui est utilisé pour mesurer le temps global du système, y compris les pauses occasionnées par l'arrêt temporaire de la palette sur le convoyeur.

Le sommet initial de l'automate est ***initial***.il est marqué par une flèche entrante. Les valeurs initiales des horloges sont x=0, t=0. Le sommet initial modélise l'état de disponibilité du convoyeur.

Lorsque le système reçoit l'ordre extérieure d, le convoyeur va démarrer et le système va franchir la transition jusqu'au sommet ***marche.*** *A cause* des contraintes associées à cette transition, x=0 et t=0, le système ne va pas séjourner dans ce sommet, mais en supposant que l'évènement d est arrivé, la transition

Initial ⟶ ***marche*** va être franchie immédiatement, en sortant du sommet initial dès le début de la tâche.

Les affectations **x** :=0 et **t** :=0 associées à cette transition modélisent l'initialisation des deux horloges qui vont être utilisées dans le sommet suivant.

Lorsque le système se trouve dans le sommet de ***marche***, la palette est transférée sur le convoyeur qui se déplace physiquement. Elle va attendre le bout du convoyeur. Par conséquent, les dynamiques des horloges dans le sommet ***marche*** sont 1**.** L'invariant du sommet associé à ***marche*** est $x \leq 11$, $t \leq 12$. Ainsi, le système doit quitter ce sommet au plus tard 11 secondes après y être arrivé, intervalle pendant lequel le convoyeur qui s'est déplacé. En outre, la période de séjour en ce sommet ne doit pas dépasser les 12 unités du temps global prévues dans la spécification.

A cause du fait que les deux horloges marchent en parallèle, le système, le système peut quitter le sommet après que la limite de 11 secondes a été atteinte, en ce cas-ci (le convoyeur ne s'arrête pas) la contraintes $t \leq 12$ étant équivalent à $t \leq 11$.

Depuis l'état de marche il y a deux possibilités pour l'évolution du système. Soit le convoyeur est arrêté pour une période finie de temps, soit la palette arrive au bout du convoyeur. La première variante variante est modélisée par la transition ***marche*** ⟶ ***arrêt***, tandis que la dernière par la transition ***marche*** ⟶ ***final***. Lorsque t est dans l'intervalle [0,12], c'est-à-dire à n'importe qui vérifié le temps global spécifié, et l'évènement d'arrêt *a* arrive, la transition jusqu'au sommet arrêt est franchissable. De la même façon, lorsque t est dans l'intervalle [0,12] et x est entre [10,11], c'est-à-dire le mouvement de la palette s'est effectué pendant l'intervalle normal de fonctionnement, la transition vers le sommet ***final*** est validée. Elle deviendra aussi franchissable si l'évènement b qui marque l'arrivée de la palette au bout du convoyeur apparaît.

Lorsque le système se trouve dans la localité ***arrêt***, la palette ne bouge plus sur le convoyeur, donc la valeur dd l'horloge x reste inchangé, comme à la sortie du sommet ***marche***. On arrête l'écoulement du temps pour cette horloge, ce qui revient

à avoir la dynamique nulle, $\dot{x}$. Les invariants du sommet sont toujours x ≤ 11, t ≤ 12 comme dans la localité ***marche***. Lorsque la contrainte temporelle 0≤ t ≤ 12 est encore satisfaite et l'évènement r de reprise du bon fonctionnement du convoyeur est enregistré, le système va revenir dans l'état ***marche***.

Une fois le système arrivé dans le sommet final, la palette est surement correctement arrivée au bout du convoyeur, dans les délais acceptés. Les deux horloges peuvent être maintenant arrêtées, $\dot{x}$=0, $\dot{t}$=0.

La modélisation qu'on vient d'exposer a été appliqué dans le logiciel PHAVer, voir le code .pha :

```
automaton conv
state_var : t, -- temps ecou
            x; -- temps glob
synclabs:   d, -- demarrage
            b, -- bout du convoyeur
            a, -- arret
            r; -- redemarage

loc  initial:  while true wait {x'==0 & t'==0}
          when x==0 & t==0 sync d do {x'==0 & t'==0} goto marche;

loc  marche: while x<=4 & t<=5 wait {x'==1 & t'==1}
          when t>=0 & t<=12 sync a goto arret;
                        when t>=0 & t<=5 & x>=10 & x<=11 sync b
                           do { x'==0 & t'==0 } goto final;

loc arret:  while x <= 11 & t <= 12 wait {x'==0 & t'==1}
         when t >= 0 & t <= 12 sync r goto marche;

loc final:  while true wait {x' == 0 & t' == 0};
initially initial & x==0 & t==0;

end
echo " Analyse en avant :" ;
echo " " ;
Analyse_avant =conv.reachable;
Analyse_avant.print ;
Analyse_avant.print("c:/resultat_avant.txt", 0);
Reg=conv.{final & x==0 & t==0} ;
Conv.reverse;
Conv.initial_states (reg);
Analyse_arriere=conv.reachable;
Analyse_arriere.print;
Analyse_arriere.print("c:/resultat_arriere.txt", 0);
good=Analyse_avant;
good.intersection_assign(Analyse_arriere) ;
good.print ;
good.print("c:/resultat_intersection.txt", 0);
```

- **Le fichier .pha**

Après avoir décrit l'automate selon le dessin de la figure 3.2, nous allons calculer les trajectoires permises 'espaces atteignable ' qui permettent d'attendre les valeurs acceptées des horloges. Selon ce qu'on a découvert dans la théorie, ceci revient à faire les analyses suivantes :

- En avant : en partant depuis le sommet initial, avec les valeurs des horloges x=0, t=0, on calcule l'espace atteignable avec les commande :

```
Analyse_avant =conv.reachable ;
```

- En arriéré : en partant depuis le sommet final avec les valeurs des horloges x=0, t=0, on calcule l'espace atteignable avec les commandes :

```
Reg=conv.{final & x==0 & t==0} ;
Conv.reverse;
Conv.initial_states (reg);
Analyse_arriere=conv.reachable;
```

- Intersection : entre les analyse en avant et en arrière :

```
good=analyse_avant ;
good.intersection_assign(Analyse_arriere) ;
```

Pour réussir à faire une analyse en arrière, on doit tout d'abord inverser l'automate par la commande **Conv.Reverse;** Toutes **les flèches** des transitions vont leur sens, tandis que les affectations et les grades vont changer leur place respectivement. Toutefois, cette opération n'altère pas l'état initial du système. Dans notre problème on a besoin de partir depuis le sommet ***final***, alors on va modifier cela par la commande convoyeur_initial_states(reg) ;, ou reg est une region déjà définie par nous comme étant le sommet ***final*** et ayant les valeurs des horloges x==0,t==0.

Après ces préparations on peut finalement faire l'analyse en arrière c'est-à-dire

l'analyse en avant de l'automate à chronomètre inversé.

Le résultat final s'obtient en faisant l'intersection des deux analyses avec la commande **analyse_avant.intersection_assign(analyse_arriere);.** On a opté pour sauvegarder le résultat de l'analyse en avant dans une variable nouvelle, good, à cause du fait que la commande **intersection_assign** écrase le premier champ, pour déposer le résultat.

Les résultats de la simulation

La simulation en PHAVer de ce modèle donne les inéquations suivantes :

Sommet	Analyse avant	Analyse arrière	Intersection	Les espaces atteignables
Initial	x= = 0 t = = 0	x= = 0 t = = 0	x= = 0 t = = 0	x == 0 t == 0
Marche	x≤ 11 x ≥ 0 t – x ≥ 0 t ≤ 12	x ≤ 11 t -x ≥ −11 -t + x ≥-2 t≤ 12	x ≤ 11 x ≥ 0 -t + x ≥-2 t– x ≥ 0 t – x ≥ 0 t ≤ 12	-x ≥ -11 x ≥ 0 t - x ≥ 0 -t ≥ -12 | t - x = 0 t ≥ 0 -t ≥ -11
Arrêt	x ≤ 11 x ≥ 0 t – x ≥ 0 t ≤ 12	x ≤ 11 x ≥ −2 -t + x ≥-2 t≤ 12	x ≤ 11 x ≥ 0 -t + x ≥-2 t– x ≥ 0 t ≤ 12	-x ≥ -11 x ≥ 0 t - x ≥ 0 -t ≥ -12
Final	x= = 0 t = = 0	x= = 0 t = = 0	x= = 0 t = = 0	x = = 0 t = = 0

Tab.3.1 – Convoyeur – résultat de simulation en PHAVer

Observons que les résultats sont représentés sous forme de contraintes linéaires. Comme attendu, l'espace accessible depuis l'état ***initial*** de l'automate (analyse en avant) pour les états ***Marche*** et ***Arrêt*** se trouve à la droite de la première bissectrice, restreint par les limites supérieures des intervalles acceptés pour chaque horloge.

On à découvert donc que le temps global mesuré est toujours plus grand ou égal au temps de marche du convoyeur ce qui est tout à fait juste. De plus, les valeurs des deux horloges sont positives, dans le sens de l'écoulement du temps. La nouveauté est la révélation de l'inéquation

– t + x ≥ - 2 dans l'analyse en arrière. Elle ne constitue pas un résultat évident, au contraire, c'est l'inégalité qui donne la limite supérieure de l'espace atteignable (*Figure 3.2*).

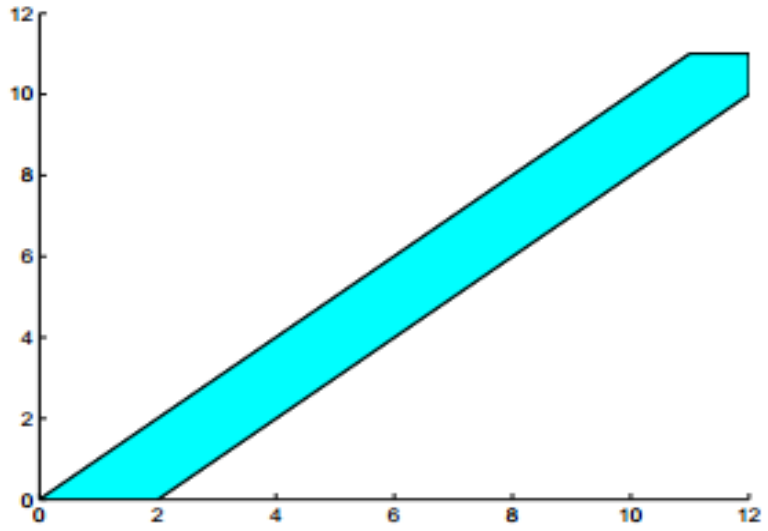


Fig. 3.3 Les espaces d'états acceptés pour le convoyeur : graphe x en fonction de t pour les états Marche et Arrêt

3.6.2 Application N2 : Modèle du Robot

Dans le domaine d'automatisme on a souvent besoin de manipuler des robots, ainsi leur surveillance devient nécessaire. Supposons que la tâche d'un robot est de prendre une palette et de faire usiner pendant un intervalle fini de temps.
Les spécifications imposent que la durée normale de l'exécution de la tâche du robot soit dans l'intervalle [2,3] u.t. cette durée peut être dépassée mais seulement avec un maximum de 1-2 secondes.

- Intervalle normal de bon fonctionnement [2,3] u.t.
- Intervalle de fonctionnement acceptable [2,4] u.t.

Nous avons modélisé le comportement de ce système par l'automate à chronomètre suivant :

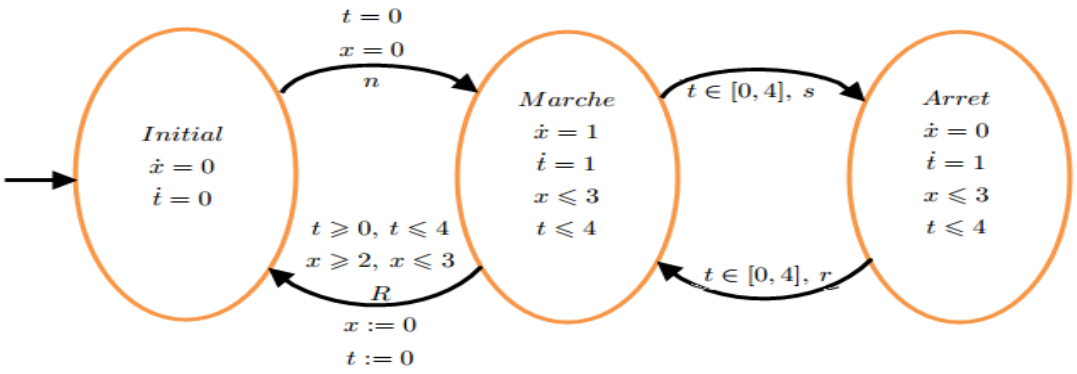


Figure 3.4 - *L'automate à chronomètres pour le système du Robot*

L'automate à chronomètre est composé d'un ensemble de sommet {*initial, marche, arrêt, final*} reliés par des transitions. les deux horloges ressemblent aux ceux du modèle convoyeur : x, qui mesure le temps écoulé pendant l'exécution de la tâche du robot depuis son démarrage et t qui est utilisé pour mesurer le temps global du système, y compris les pauses

occasionnées par l'arrêt temporaire du robot.

Le sommet initial de l'automate est ***initial***.il est marqué par une flèche entrante. Les valeurs initiales des horloges sont x1=0, t1=0. Le sommet initial modélise l'état de disponibilité du robot.

A cause des contraintes associées à la transition ***initial*** ➔ ***marche***, x1=0 et t1=0, le système ne va pas séjourner dans ce sommet, mais lorsque le système reçoit l'évènement n neutre (qui symbolise que la transition va être franchie immédiatement), le robot va commencer tout de suite sa tâche et le système va arriver au sommet ***marche.*** Les affectations x1 :=0 et t1 :=0 associées à cette transition modélisent l'initialisation des deux horloges qui vont être utilisées dans le sommet suivant.

Lorsque le système se trouve dans le sommet de ***marche***, le robot exécute sa tâche par exemple l'usinage d'une pièce. Jusqu'à la fin de la tâche, si le robot est fonctionnel, les dynamiques des horloges dans le sommet ***marche*** sont 1**.** L'invariant du sommet associé à ***marche*** est x1 ≤3, t 1 ≤ 4. Ainsi, le système doit quitter ce sommet au plus tard 3 secondes après y être arrivé, intervalle pendant lequel le convoyeur qui s'est déplacé. En outre, la période de séjour en ce sommet ne doit pas dépasser les 4 unités du temps global prévues dans la spécification.

A cause du fait que les deux horloges marchent en parallèle, le système, le système peut quitter le sommet après que la limite de 6 secondes a été atteinte, en ce cas-ci (le robot ne s'arrête pas) la contrainte t1 ≤ 04 étant équivalent à t 1 ≤ 03

Depuis l'état de marche il y a deux possibilités pour l'évolution du système. Soit le robot immobilisé pour une période finie de temps, soit il a accompli sa tâche. La première variante variante est modélisée par la transition ***marche*** ➔ ***arrêt***, tandis que la dernière par la transition ***marche*** ➔ ***final***. Lorsque t est dans l'intervalle [0,4], c'est-à-dire à n'importe quel moment qui vérifié le temps global spécifié, et l'évènement d'arrêt *s* arrive, la transition jusqu'au sommet arrêt est franchissable. De la même façon, lorsque t1 est dans l'intervalle [0,4] et x1 est entre [2,3], c'est-à-dire le robot a effectué sa tache pendant l'intervalle normal de fonctionnement, la transition vers le sommet ***final*** est validée. Elle deviendra aussi franchissable si l'évènement R (qui marque l'arrivée du robot à la fin de son travail) apparaît.

Lorsque le système se trouve dans la localité ***arrêt***, le robot est bloqué, donc la valeur de l'horloge x1 reste inchangé, comme à la sortie du sommet ***marche***. On arrête l'écoulement du temps pour cette horloge, ce qui revient à avoir la dynamique nulle, $\dot{x} = 0$. Les invariants du sommet sont toujours x1 ≤ 03, t1 ≤ 04 comme dans la localité ***marche***. Lorsque la contrainte temporelle 0≤ t1 ≤ 04 est encore satisfaite et l'évènement r de reprise du bon fonctionnement du robot est enregistré, le système va revenir dans l'état ***marche***.

Une fois le système arrivé de nouveau dans le sommet ***initia***l, le robot est surement correctement fait sa tâche, dans les délais acceptés. Les deux horloges peuvent être maintenant arrêtées, $\dot{x}$=0, $\dot{t}$=0 et un nouveau cycle peut recommencer.

Cette dernière modélisation a été appliqué dans le logiciel PHAVer, voir le code .pha :

```
automaton robot
state_var : x1, -- temps ecoulé pendant le mouvement du robot
            t1; -- temps ecoulé globalement


synclabs: n1, -- Demmarage
          s1, -- arret
           r1, -- redemarrage
           R1; -- fin de tache

loc initial: while true wait {x1'==0 & t1'==0}
      when x1==0 & t1==0 sync n1 goto marche;

loc marche: while x1<=3 & t1<=4 wait {x1'==1 & t1'==1}
      when t1>=0 & t1<=4 sync s1 goto arret;
      when t1>=0 & t1<=4 & x1>=2 & x1<=3 sync R1 do { x1==0 &
t1==0} goto initial;

loc arret: while x1<=3 & t1<=4 wait {x1'==0 & t1'==0}
      when x1>=0 & x1<=4 sync r1 goto marche;



initially initial & x1==0 & t1==0;
end

echo " Analyse en avant-robot1 :" ;
echo " " ;
Analyse_avant =robot.reachable;
Analyse_avant.print ;
Analyse_avant.print("C: /resultat_avant_robot1.txt", 0);
Reg=robot.{final & x1==0 & t1==0} ;
robot.reverse;
robot.initial_states (reg);
Analyse_arriere=robot.reachable;
Analyse_arriere.print;
Analyse_arriere.print("C :/resultat_arriere_robot1.txt", 0);
good=Analyse_avant;
good.intersection_assign(Analyse_arriere) ;
good.print ;
good.print("C:/resultat_Robot1.txt", 0);
```

Remarque 3.1 : la différence **entre** les deux modèles est l'absence de la localité ***Final.*** Selon qu'on a trouvé pour l'exemple précédent, les états initial et final sont équivalent alors on peut supprimer un d'eux. En faisant cela, on assure aussi que le système va cycler.

Les résultats de la simulation

La simulation en PHAVer de ce modèle donne les inéquations dans le tableau 3.2.

Sommet	Analyse avant	Analyse arrière	Intersection	Les espaces atteignables
Initial	x1= = 0 t 1= = 0	x1= = 0 t 1= = 0	x1= = 0 t1 = = 0	x1 = = 0 t1 = = 0
Marche	x ≤ 3 x ≥ 0 t – x ≥ 0 t ≤ 4	x ≤ 3 t -x ≥ −4 -t + x ≥-2 t≤ 4	x ≤ 3 x ≥ 0 -t + x ≥-2 t– x ≥ 0 t ≤ 4	-x ≥ -3 x ≥ 0 t - x ≥ 0 -t ≥ -4 | t - x = 0 t ≥ 0 -t ≥ -4
Arrêt	x ≤ 3 x ≥ 0 t – x ≥ 0 t ≤ 4	x ≤ 3 x ≥ −2 -t + x ≥-2 t≤ 4	x ≤ 3 x ≥ 0 -t + x ≥-2 t– x ≥ 0 t ≤ 4	-x ≥ -3 x ≥ 0 t - x ≥ 0 -t ≥ -4

Tab .3.2. – robot résultat de simulation en PHAVer

Sous Windows/Cygwin, les espaces accessibles en analyse en avant sont obtenues après 4 itérations, en 0.016sec de calcul. Ceux en analyse en arrière nécessitent 3 itérations de calcul en 0.016 secondes. Le résultat final de l'intersection est disponible après un temps global de 0.094sec.

3.6.3 Application N°3 : Modèle d'un convoyeur bouclé :

Comme on remarquait dans la section précédente, le convoyeur peut être modélisé par un autre automate à chronomètre similaire nom bouclé.

Dans ce système (fig.3.3) lorsque l'événement *d* se produise. Le poussoir met une palette sur le convoyer. Cet événement *d* est généré automatiquement dans un intervalle [0,2] *u.t* ou manuellement par l'operateur dans l'intervalle mentionné précédemment. L'hypothèse suivante est considérée : lorsque le contrôleur donne l'ordre *d* le poussoir met instantanément une palette sur le convoyeur. Lorsque la palette atteint le point *B* le capteur produit l'événement *b* dans un intervalle de [3,4[*u.t* cette durée peut être dépassée, mais seulement avec un maximum d'une seule seconde ; l'événement *b* réinitialise les horloges x2 :=0 et y2 :=0.

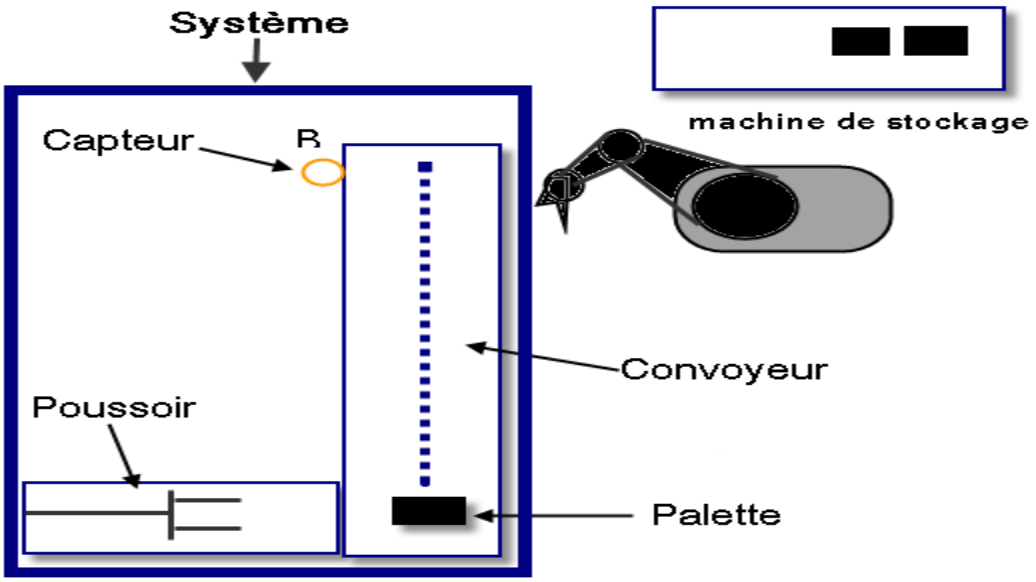


Fig. 3.3 schéma d'un convoyeur dans un système de stockage industriel simple

Dans cet instant la machine terminale d'assemblage commence à transférer la palette ; l'événement *e* est généré automatiquement après le transfer du pellet dans un intervalle de temps indéterminé (Dépendre au transfère de la palette) cet événement réinitialise l'horloge x1 :=0 pour que le système puisse continuer son fonctionnement en boucle continue. Les signaux *s* et *r* représente les sortie d'un capteur logique indiquant la dynamique d'exécution du convoyeur

On appelle intervalle normale de bon fonctionnement l'intervalle [3,4] *u.t.* et intervalle de fonctionnement acceptable l'intervalle [3,5) *u.t.* La modélisation de comportement de ce système est représentée par l'automate à chronomètre (**A**) suivant :

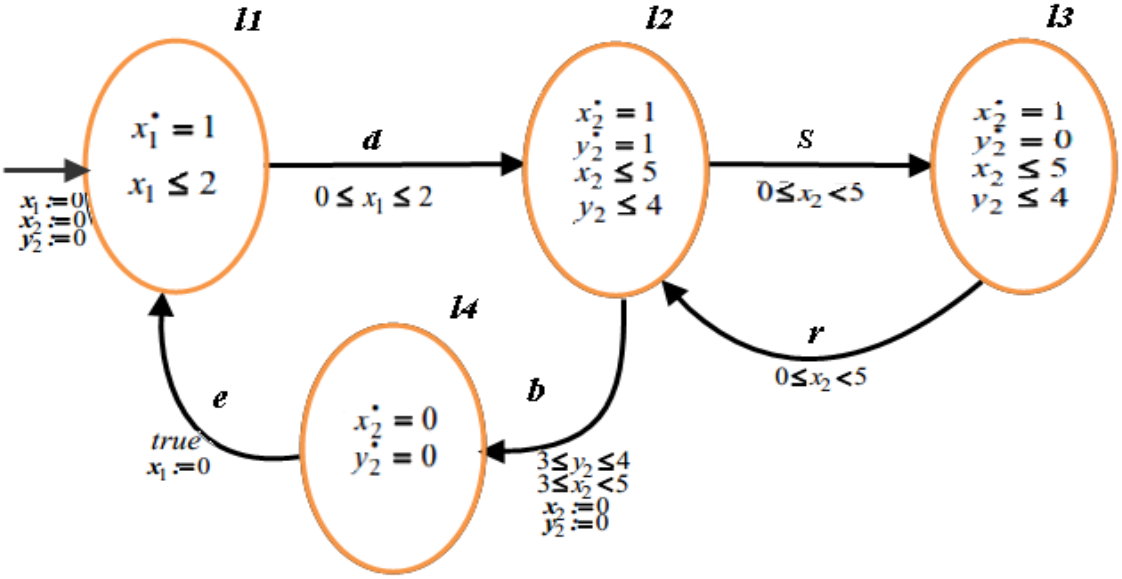


Fig. 3.4 - Représentation d'Automate A.

L'automate A est composé d'un ensemble de quatre sommet {**l1**:*généré par la présence de la palette*, **l2** *: marche*, **l3**: *arrêt*, **l4** *: final*} reliés par des transitions *d, s, b, r, e*. Les trois horloges du modèle convoyeur : x1qui mesure le temps écoulé depuis l'arriver de la pièce sur le convoyeur ; x2 : mesure le temps global du système, y2:mesure le temps écoulé depuis le démarrage du système. Ce modèle nous permet le calcul des espaces atteignables dans chaque sommet de ce système par le logiciel de vérification formelle PHAVer. Cette dernière modélisation a été appliqué dans le logiciel PHAVer, voir le code .pha :

```
automaton conv2
state_var : x1, -- le tempe coulé depuis l'arrivé de la palette
            x2,  -- le temps global du système
             y2; -- le temps écoulé depuis le démarrage de système

synclabs:   d, -- mettre la palette sur le convoyeur
            s, -- dynamique d'execution
            r, -- retour
            e,-- fin dy cycle
            b; -- arriver

loc  l1: while x1<=2 wait {x1'==1}
          when x1>=0 & x1<=2 sync d do {x1'==0 & x2'==x2 &
y2'==y2} goto l2;

loc  l2: while  x2<=5 & y2<=4 wait {x2'==1 & y2'==1}
          when x2>=0 & x2<=5 sync s do {x1'==x1 & x2'==x2 &
y2'==y2} goto l3;
          when x2>=3 & x2<5 & y2>=3 & y2<=4 sync b do {x1'==x1
& x2'==0 & y2'==0} goto l4;

loc  l3: while  x2<=5 & y2<=4 wait {x2'==1 & y2'==0}
          when x2>=0 & x2<=5 sync r do {x1'==x1 & x2'==x2 &
y2'==y2} goto l2;

loc  l4:  while true wait {x2'==0 & y2'==0}
         when true sync e do {x1'==0 & x2'==x2 & y2'==y2} goto l1;
         initially l1 & x2==0 & y2==0 & x1==0;
end
echo " Analyse en avant-conv2 :" ;
echo " " ;
Analyse_avant =conv2.reachable;
Analyse_avant.print ;
Analyse_avant.print("C:\ resultat_avant_conv2.txt", 0);
Reg=conv2.{final & x1==0 & x2==0 & y2==0} ;
conv2.reverse;
conv2.initial_states (reg);
Analyse_arriere=conv2.reachable;
Analyse_arriere.print;
Analyse_arriere.print("C:\ resultat_arriere_conv2.txt", 0);
good=Analyse_avant;
good.intersection_assign(Analyse_arriere) ;
good.print ;
good.print("C:\phavers/resultat_conv2.txt", 0);
```

Les résultats de la simulation

La simulation en PHAVer de ce modèle donne les inéquations suivantes :

Sommet	Analyse avant	Analyse arrière	Les espaces atteignables
L1	$x_2 = y_2 = 0$ $0 \leq x_1 \leq 2$	$x_2 - y_2 < 2 \mid 0 \leq x_1 \leq 2 \mid 0 \leq x_2 \leq 5 \mid 0 \leq y_2 \leq 4$	$x_1 \geq 0$ & $-x_1 \geq -2$ | $y_2 == 0$ & $x_2 == 0$ & $x_1 == 0$
L2, L3	$0 \leq x_2 - y_2 \mid$ $0 \leq x_1 \leq 2 \mid$ $0 \leq y_2 \leq 4 \mid$ $0 \leq x_2 \leq 5 \mid$	$x_2 - y_2 < 2 \mid$ $0 \leq x_2 \leq 5 \mid$ $0 \leq y_2 \leq 4$	$-x_2 \geq -5$ & $-y_2 \geq -4$ & $x_2 \geq 0$ & $-x_2 \geq -5$,
L4	$x_2 = y_2 = 0$ $0 \leq x_1 \leq 2$	$0 \leq x_1 \leq 2 \mid$ $0 \leq y_2 \leq 4 \mid$ $x_2 - y_2 < 2 \mid$ $0 \leq x_2 \leq 5 \mid$	$x_2 = y_2 = 0$ $0 \leq x_1 \leq 2$

Tab .3.3. –Convoyeur - résultat de simulation en PHAVer

Les espaces accessibles en analyse en avant sont obtenues après 3 itérations, en 0.016sec de calcul. Ceux en analyse en arrière nécessitent 3 itérations de calcul en 0.012 secondes. Le résultat final de l'intersection est disponible après un temps global de 0.073sec.

3.7 Conclusion

En calculant les espaces atteignables des automates à chronomètre qui nous concernaient on a constaté que l'espace exacte des trajectoires acceptées est exactement l'intersection des analyses en avant et en arriéré des automates.

Ceci permet d'établir un résultat remarquable : cette solution est capable de détecter un default bien avant qu'il se produise et cela donne le temps nécessaire pour l'algorithme de commande à résoudre le problème.

Phaver est un nouveau logiciel pour l'analyse des systèmes à évènements discrets et aussi les systèmes hybrides.

On l'a utilisé pour calculer les espaces atteignables de nos automates à chronomètre ; des guides d'installation et d'utilisation ont été incluses dans le chapitre ce chapitre

Comme ce logiciel n'avait pas été utilisé jusque maintenant dans le laboratoire d'Automatique de Tlemcen.

Conclusion Général

Notre travail est une investigation dans les domaines du contrôle et du diagnostic des systèmes à événement discret.

Dans un premier temps, nous nous sommes intéressés au diagnostic des systèmes à évènements discrets.

Notre démarche de diagnostic commence par la notion du diagnosticabilité ensuite nous avons présenté le diagnostic des SED temporisé, Le diagnostiqueur est un automate temporisé déterministe qui évolue en fonction des événements observables générés par le système.

Chaque sommet de cet automate permet de retrouver une estimation de l'état courant du système et les occurrences des événements de défauts non observables. Une fonction de décision, analyse le sommet courant du diagnostiqueur et annonce l'occurrence d'un défaut du mode Fi lorsque ce sommet est Fi-certain.

Enfin, nous avons étudié la diagnosticabilité du modèle considéré dans notre démarche. Nous avons établi que l'utilisation d'un diagnostiqueur est soumise à la vérification de la notion de diagnosticabilité du modèle considéré. Une méthode systématique permettant de vérifier la diagnosticabilité du modèle automate temporisé a été élaborée. Cette méthode repose sur la détection des cycles Fi-indéterminé et des sommets finaux Fi-incertain. Un théorème liant la notion de diagnosticabilité et la vérification de ces conditions a été ensuite présenté.

Dans un second temps, nous nous sommes intéressés à la synthèse de contrôleur pour les systèmes à événements discrets et les systèmes à évènements discrets temporisés à partir des automates à état fini et des automates temporisés.

Dans notre travail, nous proposons une méthode systématique et facile de mise en œuvre pour la synthèse du contrôle des systèmes à événements discrets. Nous modélisons les systèmes par des automates. La méthode distincte utilise des conditions pour les transitions contrôlables.

Nous avons alors été confrontés au problème des transitions incontrôlables pour garantir l'optimalité et à la complexité apportée par le nombre des places de contrôle qui peut être très grand. Pour résoudre ces problèmes, nous avons utilisé l'approche principale de Ramadge et Wonham ainsi que le travail de Kumar pour déterminer les états interdits dans le cas général ou il y a des évènements incontrôlables.

Pour mettre en oeuvre ces notions nous avons utilisé le logiciel PHAVer. On tient juste a précisé que ce logiciel est utilisé pour la première fois dans le laboratoire d'Automatique de Tlemcen.

Ce travail constitue un bon complément pour notre formation de Masters en automatique. Ou nous avons principalement et affaire à des systèmes dynamiques continus. Ici nous avons complétés nos connaissances dans les domaines des systèmes dynamiques par cette classe importante des systèmes à événements discrets.

Bibliographie

[01] Andra - Ioana VASILIU, Synthèse de contrôleurs des systèmes à événements discrets basée sur les réseaux de Pétri., [2006].

[02] Haithem DERBEL, Diagnostic à base de modèles des systèmes temporisés et d'une sous-classe de systèmes dynamiques hybrides. (Diagnostic des systèmes temporisés). , [2009].

[03] NOUIOUA F / DAGUE P, Diagnosticabilité des systèmes a événement discret. , [2009].

[04] Barbuti, R et L.tesei. ,«Timed automata with urgent transitions», Acta Inf vol. 40, no 5, p. 317–347, [2004].

[05] Ghazel, M., Surveillance des Systèmes à Evénements Discrets à l'Aide des Réseaux de Petri T-Temporels, thèse de doctorat, l'Ecole Centrale de Lille et l'Université des Sciences et Technologies de Lille., [2005].

[06] Sampath, M., R. Sengupta, S. Lafortune, K. Sinnamohideen et D. Teneketzis., «Diagnosability of discrete event systems», IEEE Transactions on Automatic Control, vol. 40, no 9, p. 1555–1575, [1995].

[07] Alur, R. et D. L. Dill., «A theory of timed automata», Theoretical Computer Science, vol. 126, p183–235 , [1994].

[08] Wonham, W,Supervisory control of discrete event systems, Technical report, University of Toronto, Dept. of Electrical and Computer Engineering, [2011].

[09] Ramadge, P. &Wonham, W. 'Supervisory control of a class of discrete event processes', SIAM J. Control Optim. 25(1), 206–230.,[1987].

[10] Ramadge P. J., Wonham W., "The Control of Discrete Event Systems", Proceedings of the IEEE; *Special issue on Dynamics of Discrete Event Systems*, Vol. 77, No. 1:81-98., [1989].

[11] Uzam, M. &Wonham, W. ,'A hybrid approach to supervisory control of discrete event systems coupling RW supervisors to Petri nets', *Int. J. of Adv. Manuf.Technol.* **28**(7 - 8), 747 – 760., [2006].

[12] Li, Y. &Wonham, W., 'Control of vector discrete-event systems. II. Controller synthesis', IEEE Trans. Autom. Control 39(3), 512–531., [1994].

[13] Kumar, R. & Holloway, L. ,'Supervisory control of deterministic Petri nets with regular specification languages', *IEEE Trans. Autom. Control* **41**(2), 245–249., [1996].

[14] Holloway, L., Guan, X. & Zhang, L., 'A generalization of state avoidance policies for controlled Petri nets', *IEEE Trans. Autom. Control* **41**(6), 804–816. [1996].

[15] Holloway, L. & Krogh, B., 'Synthesis of feedback control logic for a class of controlled Petri nets', *IEEE Trans. Autom. Control* **35**(5), 514–523., [1990].

[16] Dideban, A. & Alla, H., Determination of minimal sets of control places for safePetri nets, *in* 'Proc. IEEE American Control Conference (ACC 2007)', New York(NY), pp. 4975–4980.,[2007].

[17] Dideban, A. & Alla, H., 'Reduction of constraints for controller synthesis basedon safe Petri nets', *Automatica 44*(7), 1697–1706.,[2008].

[18] Abbas DIDEBAN, Synthèse de contrôleurs discrets par simplification de contraintes et de conditions. , [2007].

[19] Latifa GHOMRI, Synthèse de contrôleur de systèmes hybrides à flux continu par réseaux de Pétri hybrides. , [2012].

[20] Kumar, R., Supervisory Synthesis Techniques for Discrete Event Dynamical Systems: Transition Model Based Approach, PhD thesis, Department of Electrical and Computer Engineering, University of Texas at Austin. , [1991].

[21] Dideban A., Alla H., "Determination of Minimal Sets of Control Places fo rSafe Petri Nets", American Control Conference, 11-13 July, New York City, USA.[2007].

[22] Alexandru Tibriu S., "Sur La Syntheses De La Commande Des Systèmes a Evénement Discret Temporisés" Institut National Polytechnique de Gronoble.
France [2001].

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

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

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

[26] Cassandras, C. et S. Lafortune., Introduction to Discrete Event Systems, Kluwer Academic Publisher., [1999].

[27] Lin, F. et W. M. Wonham., «Diagnosability of discrete event systems and its applications», Discrete Event Dynamic Systems, vol. 4, no 2, p. 197–212.; [1994].

[28] Fourlas, K., K. Kyriakopoulos et N. Krikelis., «Diagnosability of hybrid systems», dans Proceedings of MCCA'02, p. 3994–3999., [2002].

[29] Thorsley, D. et D. Teneketzis., «Diagnosability of stochastic discrete-event systems», IEEE Transactions On Automatic Control, vol. 50, no 4, p. 476–492.,[2005].

[30] Tripakis, S., «Fault diagnosis for timed automata», dans Proceedings of the 7th International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems, FTRTFT'02, Springer-Verlag, London, UK, ISBN 3-540-44165-4, p. 205–224., [2002].

[31] Lin, F. et W. M. Wonham., «On observability of discrete-event systems», Information sciences, vol. 44, no 3, p. 173–198 , [1988]

[32] Goran Frehse. Phaver : language overview for PHAVer v 0.35. Laboratoire verimag, Gières, France 0.35 edition, [2006]

[33] Howard Wong-Toi Thomas A. Henzinger, Pei-Psin Ho. A user Guide to HyTech. University of California, Berkeley, USA, October. Describes version 1.04 of Hytech, [1996].

Résumé

Le sujet de mémoire proposé est dans la continuité des travaux précédents. Il vise à développer davantage les travaux de diagnostic à base de modèles temporisés des systèmes à événements discrets. Il s'agit de combiner l'approche de diagnostic à base de modèles temporisés avec la théorie du contrôle supervisé développée par Ramadge et Wonham.

Ce sujet correspond ainsi à une extension temporisée de l'approche de diagnostic et de supervision purement discrète .Il s'agit de proposer une solution intégrée qui permet de coupler le module du diagnostic avec le module du contrôle.

Cette solution repose sur la synthèse d'un superviseur qui permet d'autoriser ou d'inhiber certains événements contrôlables du système afin de déterminer le plus grand sous-langage temporisé diagnosticable. Ainsi, le diagnostiqueur résultant à partir d'une telle approche est construit à partir d'une restriction diagnosticable du modèle du système ce qui permet l'identification de tout défaut au bout d'un délai fini de son occurrence.

MOTS-CLES : Systèmes à événements discret, synthèse de contrôleurs, diagnostic, automates temporisés.

Abstract

This thesis presents an approach to monitoring the controlled discrete events systems.

The study is limited to the interruptible faults: intermittent and permanent. To increase the availability of a system, it is crucial to reduce the unnecessary interruptions.

For that, the acceptable behavior of these systems is introduced. This behavior presents a tolerance to the intermittent faults. An approach of the construction of monitoring system is presented.

Key words: Discrete events systems, Monitoring, control,