Surveillance des systèmes à événements discrets commandés: Conception et implémentation en utilisant l'automate programmable industriel

Adib Allahham

Université de Joseph Fourier - Grenoble I

No. attribué par la bibliothèque

THESE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ JOSEPH FOURIER

Spécialité : **AUTOMATIQUE-PRODUCTIQUE**

préparée au Département Automatique de GIPSA-lab Grenoble

dans le cadre de l'École Doctorale :
Électronique, Électrotechnique, Automatique, Traitement du Signal

présentée et soutenue publiquement

par

Adib ALLAHHAM

le 22 Octobre 2008

Titre :

Surveillance des systèmes à événements discrets commandés : Conception et implémentation en utilisant l'automate programmable industriel

Directeur de thése:

M. Hassane ALLA(Professeur à l'Université de Grenoble I)

JURY :

M. Jean-Jacques LESAGE	Rapporteur	Professeur à l'École Normale Supérieure de Cachan
M. Olivier H. ROUX	Rapporteur	Maître de Conférences à l'Université de Nantes
M. Jean Claude HENNET	Examinateur	Directeur de Recherche CNRS
M. Christian COMMAULT	Examinateur	Professeur à l'Institut Polytechnique de Grenoble
M. Hassane ALLA	Examinateur	Professeur à l'Université de Grenoble I

Table des figures

Table des matières

REMERCIEMENTS

Je tiens à remercier les personnes qui ont partagé avec moi ma vie, les bonheurs et aussi mes expériences de la recherche pendant ces années. J'essaie d'écrire ce que je pense et ce que je ressens, et cette partie n'est pas la plus facile de ma thèse.

Ce mémoire est, pour moi, l'occasion de remercier la personne qui m'a toujours épaulé et qui m'a permis d'arriver jusqu'ici. Je tiens à exprimer ma reconnaissance profonde à Monsieur le Professeur Hassane ALLA qui a dirigé mes travaux, m'a accordé sa confiance et m'a soutenue sur de nombreux plans durant cette thèse. Je le remercie grandement, que ce soit pour ces qualités scientifiques ou humaines, faisant toujours preuve de réflexion, de patience et de diplomatie.

Je remercie vivement les rapporteurs de ce mémoire de thèse, Monsieur le professeur Jean-Jacques LESAGE de l'École Normale Supérieure de Cachan, directeur du Laboratoire Universitaire de Recherche en Production Automatisée (LURPA), et monsieur Olivier H. ROUX, maître de conférences à l'Université de Nantes. Je leur suis très reconnaissant pour avoir bien voulu consacrer de leur temps à l'étude de mes travaux. Merci pour leurs remarques et suggestions.
Un grand merci à Monsieur Christian COMMAULT de m'avoir fait l'honneur d'examiner mes travaux et d'avoir accepté de présider ce jury.
J'ai été très heureux de compter parmi mon jury Monsieur Jean-Claude HENNET, directeur de cherche (CNRS). Je le remercie de son regard extérieur, pertinent et critique sur mon travail.

Mes remerciements s'adressent également aux membres de l'équipe SED dont j'ai fait partie durant cette période. Je remercie Mme. Maria Di MASCOLO, Mme. Zinbe SIMEU ABAZI, M. Pierre LADDET, Mme. Alexia GOUIN, M. Bernard DESCOTES-GENON et M. Stephane MOCANU pour leur sympathie, leurs remarques et conseils. Bien évidemment, je tiens également à remercier les doctorants de l'équipe SED (Abbas, Khalid, Haithem, Yamen, Goumana, Imene, Andra, Latifa et Melha) et tous ceux que j'ai oublié.

Je n'oublierai pas l'ensemble de membres de l'équipe administrative et technique du Laboratoire GIPSA-lab et en particulier Daniel, Marie-Thérèse, Virginie et Patricia. Je les remercie pour leur gentillesse, leur bonne humeur et leur efficacité.

Je tiens à remercier ceux qui m'ont accompagnés tous les jours depuis le début. Je pense évidemment à mes amis Ghaith, Fisal, Hayan, Aiman, Sameh, Mouaiad, Cédric,

Hala, Van, Alexandre, Maher, Ahmed, Amine, Li, Nabil, Zeashan et Saleh pour leur soutien moral et les bons moments que nous avons passés ensemble.

La distance de ma famille a toujours été une des difficultés que j'ai eu à surmonter durant mon séjour en France. Je tiens à exprimer tous mes remerciements envers mon père, ma mère et ma sœur qui m'ont toujours soutenu tout au long de mon parcours depuis le premier jour. Pour mon père et ma mère, je n'ai pas de mot spécial, je vous dédie ces travaux.

Un grand merci à tous ce qui, de près ou de loin, ont contribué à la réalisation de ce travail.

Je termine ces remerciements en citant un proverbe Tibétain que mon ami Do-Hieu Trinh m'a toujours répété :

"***Si un problème a une solution, alors it est inutile de s'isnquiéter, s'il n'en a pas, s'inquiéter n'y changera rien***".

Chapitre 1

INTRODUCTION GÉNÉRALE

La surveillance est à l'origine de nombreux travaux depuis les dernières décennies Elle se définit comme l'opération permettant de détecter un défaut, de localiser son origine et de déterminer ses causes. Nos travaux s'articuleront autour d'une définition de la surveillance comportant les deux activités : l'acquisition des données et la détection des défauts. 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 ou anormal. L'intérêt pour la surveillance et la détection des défauts s'explique par la complexité croissante des systèmes réels qui sont de plus en plus exigeants en termes de contraintes de sécurité, de fiabilité, de disponibilité et de performance. En effet, la possibilité qu'un système tombe en panne croît malgré les précautions de manipulations et l'expérience des opérateurs humains. Signaler le plus tôt possible à l'opérateur les écarts détectés par rapport au comportement nominal prévu est fondamental pour la mise en œuvre des actions préventives et correctives sur le système, pour empêcher la propagation de pannes et pour limiter leurs conséquences. Ceci a pour effet aussi de lancer, le plus tôt possible, des phases de diagnostic et de décision. Pour cela, la détection des défauts au plus tôt représente une exigence de plus en plus demandée dans le milieu industriel.

Il existe un grand nombre de méthodes de surveillance [Chow and Willsky, 1984], [Frank, 1990], [Srinivasan and Jafari, 1993], [Zwingelstein, 1995], [Sampath et al., 1995], [Sampath et al., 1996], [Frank, 1996], [Chen and Patton, 1999], [Pandalai and Holloway, 2000], [Lunze, 2000], [Tripakis, 2002], [Boufaied, 2003], [Jiang et al., 2003], [Zad et al., 2003], [Cocquempot et al., 2003], [Cocquempot et al., 2004], [Zad et al., 2005], [Ghazel et al., 2005]... Elles se basent sur un modèle du comportement normal et/ou défaillant du système. Elles se distinguent par les différents critères qu'elles considèrent : la dynamique du système à surveiller (discret, continu ou hybride), l'implémentation du système de surveillance, la nature de l'information disponible (qualitative et/ou quantitative) et sa distribution (centralisée ou distribuée).
Dans le cadre des systèmes à événements discrets, nous trouvons que les méthodes de surveillance sont basées soit sur un modèle complet (normal et défaillant) ou sur un modèle de comportement normal. Les méthodes basées sur le premier type des modèles ont pour objectif de détecter et de diagnostiquer les défauts. Tandis que celles basées sur un modèle de comportement normal sont conçues pour détecter seulement les défauts. Il est important de noter que le modèle complet nécessite une connaissance dite "profonde" du système. Ceci exige de connaître tous les défauts possibles du système surveillé et le comportement du système suite à chaque défaut, tandis que la modélisation de comportement normal demande seulement une connaissance des plages de fonctionnement normal du système. Ces plages de fonctionnement sont en général bien connues. Certains travaux ont introduit le comportement dégradé. Le concepteur prévoit un état intermédiaire entre l'état de fonctionnement normal et l'état de défaut, en prévoyant un

intervalle dit "de tolérance" situé chronologiquement immédiatement après l'intervalle de bon fonctionnement. Les travaux portant sur la surveillance à base d'un modèle de comportement normal ou dégradé, surveillent l'ordre d'occurrence des événements et leurs dates d'occurrence. Ces travaux détectent le retard ou l'absence d'un événement à l'instant de l'expiration de la date au plus tard prévue d'apparition de cet événement. Pour les raisons mentionnées ci-dessus, la détection de ce défaut avant cet instant est un des objectifs de notre méthode de surveillance.
Nous avons restreint le domaine d'étude aux systèmes à événements discrets commandés susceptible aux défauts interruptibles : intermittents et permanents. Cependant, pour réaliser une approche de surveillance de ce type de systèmes, il faut lever un certain nombre de difficultés rencontrées au niveau de la modélisation du système, les informations retenues dans les modèles, la structure de la prise de décision et l'approche d'implémentation.

1.1 Contribution de la thèse

La première contribution de ce mémoire concerne la surveillance des systèmes à événements discrets commandés ayant un caractère temporel.
L'exploration des données temporelles relatives aux événements qui peuvent avoir lieu dans un système à événements discrets s'est avérée très fructueuse dans un processus de surveillance. C'est à partir de ce fait que nous élaborons une approche de surveillance qui surveillera l'ordre d'occurrence des événements et leurs dates d'occurrence dans un comportement appelé "acceptable". Ce comportement peut être adopté dans un système pouvant être sujet aux défauts intermittents, afin d'augmenter sa disponibilité. L'étude se restreint à un type des défauts, appelés les défauts interruptibles qui sont très répandus. Les ressources du système susceptible à ces défauts deviennent indisponibles de manière permanente ou intermittente et interrompent l'exécution des tâches (ou activités). Le comportement de système sujet aux défauts interruptible sera présenté sur un modèle de l'automate à chronomètres. Il est important de noter que les défauts ne sont pas représentés dans le modèle du système. Nous ne considérons que les dynamiques d'exécution des services ou des tâches du système qui sont observables.
Le mécanisme développé dans notre approche a pour principe d'une part, d'exploiter les contraintes temporelles dont on peut disposer sur le comportement acceptable du système. Et d'autre part, d'utiliser les techniques d'analyse d'atteignabilité de l'automate à chronomètres. Nous modélisons, dans un premier temps, le comportement du système sujet aux défauts interruptibles en utilisant l'automate à chronomètres. Nous donnons les propriétés caractérisant les exécutions acceptables des tâches du système. Ensuite, une procédure de synthèse est appliquée à cet automate afin de délimiter seulement les trajectoires vérifiant les propriétés requises. Dans l'automate résultant, les sommets représentent les situations atteignables du système, les équations

différentielles dans chaque sommet reflètent les états des tâches de système dans cette situation. Le franchissement d'une transition est conditionnée seulement par l'occurrence de l'événement associé à cette transition. L'espace temporel synthétisé dans chaque sommet délimite le comportement acceptable dans la situation correspondante du système. La violation de l'espace temporel dans un sommet, nous permet de détecter les défauts interruptibles au plus tôt. Nous entendons par le terme "*détection le plus tôt*" une détection de l'absence ou du retard d'un événement sans attendre l'expiration de la date au plus tard prévue de son apparition. Ceci représente, comme nous avons dit, une exigence de plus en plus demandée dans le milieu industriel.

Dans la deuxième contribution dans ce mémoire il s'agit de proposer une extension de Réseaux de Petri temporels. Ce nouveau modèle est capable de modéliser l'interruption et la reprise d'une horloge.
En effet, modéliser le comportement du système susceptible aux défauts interruptibles par des automates à chronomètres n'est pas toujours simple. De plus, certains cas des systèmes à surveiller ne sont pas exprimables avec les automates, comme le cas des procédés à instance-multiple. Ceci rejoint la notion de multi-sensibilisation dans le modèle réseau de Petri. Cependant, les réseaux de Petri temporels [Merlin, 1974], [Berthomieu and Diaz, 1991] ne sont en général pas suffisants pour modéliser les applications en présence du comportement interruptible. Ils doivent pour cela être étendus afin de pouvoir modéliser une tâche interrompue et reprise au même endroit un peu plus tard. Ceci nous a conduit à proposer une extension de réseau de Petri temporels, appelée "***Réseaux de Petri à chronomètres Post- et Pré-initialisés***".
Ce modèle garde les caractéristiques principales des Réseaux de Petri. Il a la capacité à représenter d'une manière intuitive et naturelle les principaux mécanismes des systèmes à événements discrets : parallélisme, synchronisation et partage des ressources. C'est un modèle de spécification formelle qui allient les avantages d'une description graphique puissante et d'une sémantique formelle.
Une contribution importante dans notre modèle repose sur les concepts de *Pré-* et *Post-initialisation* des horloges. Dans le modèle Réseau de Petri temporels, seul le concept de *Pré-initialisation* est utilisé. Les horloges sont initialisées lorsque les transitions correspondantes à ces horloges sont nouvellement sensibilisées. Dans le modèle réseau de Petri à chronomètres, nous introduisons le concept de *Post-initialisation* où les horloges sont initialisées lorsque les transitions correspondantes à ces horloges sont franchies.
Nous considérons également le problème de calcul de l'espace d'états pour le nouveau modèle réseau de Petri à chronomètres, celui-ci est indécidable comme toutes les extension des réseaux de Petri temporels modélisant l'interruption ou la préemption. Dans ce but, nous proposons une méthode basée sur la traduction de réseau de Petri à chronomètres en automate à chronomètres. Ensuite, une analyse en avant sera appliquée sur l'automate à chronomètres ainsi obtenu en utilisant le model-checker PHAVer.

Dans notre méthode de surveillance, nous voulons combiner la puissance de modélisation de réseau de Petri à chronomètres avec le pouvoir d'analyse de l'automate à chronomètres. Dans ce but, nous modélisons le comportement des systèmes sujets aux défauts interruptibles par le réseau de Petri à chronomètres. L'objectif est d'exploiter les avantages que la modélisation par le réseau de Petri apporte. Ensuite, ce modèle sera traduit en automate à chronomètres afin d'appliquer la procédure de synthèse permettant de calculer les trajectoires représentant l'exécution acceptable du système. Dans le modèle Réseau de Petri à chronomètres d'une tâche sujette aux défauts interruptibles, les chronomètres associés aux transitions ne sont pas indépendants. Cela permet de réduire le nombre de chronomètres dans l'automate à chronomètres. Seuls les chronomètres indépendants subsistent dans l'automate résultant.

La troisième contribution concerne l'implémentation de notre méthode de surveillance. Nous allons voir comment obtenir un programme SFC permettant d'implémenter un système de surveillance par l'automate programmable industriel (API). Une procédure de traduction structurelle de l'automate délimitant le comportement acceptable de système en programme SFC est présentée. Dans le programme SFC, une étape correspond à un sommet de l'automate. Une transition dans l'automate est représentée par une transition dans le programme SFC. La question principale à laquelle nous répondrons concerne la procédure de traduction des informations temporelles ; comment peuvent-elles être représentées dans un programme SFC ? On verra que la réponse à cette question n'est pas triviale et qu'elle nécessite d'associer actions et variables pour résoudre ce problème.
Une validation expérimentale est appliquée sur un système de surveillance d'un procédé industriel. Les validations expérimentales de cette application sont faites à l'automate de surveillance et à son programme de surveillance. Des séquences des signaux binaires représentant les événements du procédé sont générées. Ces séquences sont utilisées à la fois pendant la validation de l'automate de surveillance et du programme de surveillance. La comparaison entre les réactions de l'automate et du programme SFC sur les scénarios de fonctionnement, nous permet de constater la correction de la procédure de traduction proposée.

1.2 Organisation du mémoire de thèse

Ce mémoire est organisé en cinq chapitres.
Le premier chapitre présente le cadre général de notre étude. Il présente le vocabulaire et la terminologie employés dans ce mémoire de façon à bien positionner le cadre de notre travail. Il réunit des définitions discutées au sein de différentes communautés scientifiques. Ensuite, nous présentons une classification générale des méthodes de

surveillance. Nous présentons aussi le type de système considéré : les systèmes à événements discrets commandés ayant un caractère temporel. Enfin, l'accent est mis sur les outils utilisés pour modéliser les systèmes à événements discrets temporels et surtout l'automate à chronomètres et ses techniques d'analyse. Cet outil sera utilisé tout au long de notre travail.
Le deuxième chapitre est consacré à une étude bibliographique assez large sur la problématique de surveillance et de diagnostic des systèmes à événements discrets. Nous analysons les principales approches de diagnostic et de surveillance liées aux systèmes à événements discrets et de modélisation que nous avons organisées selon la classification suivante : les méthodes à base de modèle complet (normal et fautif) et les méthodes à base de modèle de comportement normal. Nous présentons les critères sur lesquelles la conception des méthodes de surveillance des systèmes à événements discrets repose. Ces critères sont le type de modèle, la structure de la prise de décision et l'architecture du système. Donc, nous mettons l'accent sur les principales approches de construction d'un système de surveillance, sur les approches d'implémentation et sur la structure de prise la décision. Enfin, une discussion termine ce chapitre afin de positionner l'approche que nous proposerons dans le chapitre 4 pour la surveillance des systèmes à événements discrets commandés.
Nous avons dédié les trois chapitres restant à présenter les travaux développés.
Le troisième chapitre est consacré à la présentation de notre approche de surveillance. Nous allons présenter, au cours de ce chapitre, la méthode de conception du système de surveillance basé sur l'automate à chronomètres. D'abord, nous commençons par préciser la notion de comportement acceptable, le choix de l'outil de modélisation et le principe de l'approche proposée. Ensuite, nous illustrons intuitivement, par un exemple, le principe de construction du système de surveillance. A travers cette présentation intuitive, nous donnons les motivations conduisant à développer une telle approche. Puis, nous présentons la démarche formelle de la construction du système de surveillance. Elle consiste à construire les modèles des différentes tâches constituant le système en utilisant les automates à chronomètres. Par conséquent, le modèle global du système sera aussi sous la forme d'automate à chronomètres (résultant de la composition synchrone des modèles des différentes tâches). Une procédure de synthèse sera appliquée à cet automate afin de délimiter seulement les trajectoires vérifiant les propriétés qui représentent l'exécution acceptable de toutes les tâches du système. Enfin, nous concluons en appliquant la méthode proposée à un système manufacturier.

Le quatrième chapitre est consacré à notre apport dans le domaine de la modélisation, en utilisant le réseau de Petri. Nous allons présenter au cours de ce chapitre, le ***réseau de Petri à chronomètres Post- et Pré-initialisés***, sa syntaxe et sa sémantique. Ce modèle sera indiqué au cours de ce mémoire par réseau de Petri à chronomètres. Une comparaison entre le réseau de Petri à chronomètres et d'autres

modèles réseaux de Petri sera présentée. Ensuite, l'algorithme de traduction d'un réseau de Petri à chronomètres en un automate à chronomètres et le calcul de l'espace des états du réseau de Petri à chronomètres seront exposés. Une preuve de bisimulation est présentée afin de montrer la correction de la traduction. La modélisation par un réseau de Petri à chronomètres du système de surveillance sera aussi décrite en detail. Puis, deux exemples seront considérés. Le premier exemple est donné afin d'illustrer la modélisation du système de surveillance en utilisant un réseau de Petri à chronomètres. Le deuxième exemple montre d'autres possibilités de modélisation offertes par le modèle réseau de Petri à chronomètres.

Le cinquième chapitre présente l'implémentation et la validation expérimentale de notre méthode de surveillance. Nous présentons d'abord un rappel du SFC et sa syntaxe. Ensuite, le passage d'un automate de surveillance à un programme SFC est décrit. Puis, un procédé industriel sera considéré. Nous construisons le système de surveillance de ce procédé. Ensuite, la validation expérimentale par simulation de ce système de surveillance sera présentée. La simulation se fait pour l'automate de surveillance et le programme SFC de surveillance.

Le dernier chapitre conclut notre travail et propose un certain nombre de perspectives.

Ce travail de thèse a donné lieu aux publications suivantes :

- *Détection des défauts dans les systèmes à événements discrets en utilisant le Grafcet et l'automate temporisé*, Adib Allahham, présenté dans le cadre des Journées Doctorales et Nationales du GDR MACS, Lyon, France, 5-7 Septembre 2005.

- *Monitoring of timed discrete events systems : Application to manufacturing systems*, Adib Allahham et Hassane Alla, In The 32nd Annual conference of IEEE Industrial Electronics Society, pages 3609-3614, Paris, November, 2006.

- *Monitoring of a class of timed discrete events systems*, Adib Allahham et Hassane Alla, IEEE International Conference on Robotics and Automation, pages 1003 - 1008, Roma, Italy, 10-14 April 2007. **Une version étendue est soumise à IEEE Transaction on Automation Science and Engineering**. Elle est en cours de revision.

- *Post and Pre-initialized Stopwatch Petri Nets*, Adib Allahham et Hassane Alla, IFAC Workshop on Dependable Control of Discrete Systems, pages 69 - 74, Cachan, France, 13-15 Juin 2007.

– *Design and implementation of a monitoring system using Grafcet*, Adib Allahham et Hassane Alla, In The 4th International Conference on Informatics in Control, Automation and Robotics, pages 220 - 225, Angers, France, 9-12 May 2007. **Une version étendue est soumise à IEEE Transaction on Control Systems Technology.**

– *Réseaux de Petri à chronomètres Post et Pré-initialisés*, Adib Allahham et Hassane Alla, colloque français MSR'07 2007, pages 263 - 279, Lyon, France, 17 - 19 Octobre, 2007. **Une version étendue a été acceptée dans Nonlinear Analysis hybrid systems.**

Chapitre 2

CADRE DE L'ÉTUDE

2.1 Introduction

Ce chapitre est un chapitre introductif visant à rappeler, dans un premier temps, la terminologie utilisée pour la surveillance, rencontrée dans la littérature et retenue dans ce mémoire. Il met en place les concepts de surveillance, de diagnostic et de supervision. Tant que nous nous intéressons à la détection des défauts, nous situerons la fonction détection par rapport aux modules de surveillance et de diagnostic. Ensuite, une classification des défauts auxquels les systèmes commandés sont sujet, est présentée. Dans la deuxième partie de ce chapitre, nous présentons le type des systèmes vers lesquels nous souhaitons orienter nos travaux. Ce sont les systèmes à événements discrets commandés ayant un caractère temporel. Nous présentons aussi quelques outils de modélisation des systèmes à événements discrets, ainsi que leurs méthodes d'analyse. Nous jugeons que la présentation de ces outils est nécessaires pour la compréhension des travaux présentés dans l'ensemble du mémoire de thèse.

2.2 Surveillance des systèmes à événements discrets commandés

Nous présentons dans la suite quelques définitions inspirées des travaux de [Combacau et al., 2000b], [Combacau et al., 2000a] afin d'être d'accord sur la fonction du système de surveillance. Les définitions considérées sont issues d'un travail de réflexion mené dans le cadre du Groupement de Recherche en Productique.

2.2.1 Surveillance, diagnostic et supervision

Chaque système commandé doit être muni d'un système de surveillance afin de pouvoir alerter l'opérateur lors de l'occurrence de défaillances. L'information du diagnostic établi, sera ensuite remontée afin que le superviseur puisse prendre une décision. Nous voyons ainsi que différents concepts apparaisssent : surveillance, diagnostic, supervision. Nous les définissons ci-dessous.

La surveillance des procédés industriels consiste à générer des alarmes à partir des informations délivrées par des capteurs. Elle recueille les signaux en provenance du procédé et de la commande et reconstitue l'état réel du système commandé. La surveillance est limitée aux fonctions qui collectent des informations, font des inférences, sans agir réellement, ni sur le procédé, ni sur le système de commande. La surveillance a donc un rôle passif vis-à-vis du système de commande et du procédé.

La détection consiste à comparer le comportement courant au comportement de référence et ensuite à prendre une décision en résultat de la comparaison. Elle détecte

tout écart du comportement normal du système et alerte les opérateurs humains de supervision de la présence d'un défaut. Cette fonction permet donc de caractériser le fonctionnement du système de normal ou d'anormal.
Le diagnostic établit un lien de cause à effet entre un symptôme observé et le défaut qui est survenue, ses causes et ses conséquences. Cette fonction est généralement déclinée en trois sous-fonctions : la localisation détermine quel est le sous-système responsable du défaut, l'identification caractérise la cause du défaut et l'explication élabore les conclusions.
La **supervision** recouvre les aspects fonctionnement normal et fonctionnement anormal d'un système :

- en fonctionnement normal, son rôle est de surveiller et de contrôler l'exécution d'une opération et le fonctionnement d'une installation.
- en présence d'un défaut, la supervision doit prendre toutes les décisions correctives nécessaires pour assurer le retour vers un fonctionnement normal, en ayant la connaissance des causes ou des organes ayant subis un défaut.

La supervision a donc un rôle décisionnel et opérationnel en vue de la reprise de la commande. Au cours de cette thèse, nous nous intéressons à la détection des défauts, donc une présentation de la fonction détection sera présentée.
La fonction détection détermine la normalité ou l'anormalité du système en fonctionnement. Cette fonction représente très souvent un sujet de débat concernant sa place. De nombreuses approches considèrent la détection comme un élément à part du diagnostic et le voient plutôt comme une entité de la surveillance [Combacau et al., 2000b] et [Boufaied, 2003]. D'autres travaux la considèrent comme une information primordiale et indissociable du diagnostic et définissent le diagnostic comme la détection, la localisation et l'identification de défauts. Ce sont les méthodes à base de modèles appelées FDI (Fault Detection and Isolation) [Chen and Patton, 1999], [Frank, 1996]. Dans ce cadre, la détection permet de détecter tout écart du comportement normal du système et alerte les opérateurs humains de supervision de la présence d'un défaut. La localisation permet de remonter à l'origine de l'anomalie et de localiser le ou les composants défectueux. Enfin, l'identification détermine l'instant d'apparition de la panne, sa durée et son importance.
Par la suite, nos travaux s'articuleront autour d'une définition de la surveillance comportant les deux activités : l'acquisition des données et la détection des défauts. La fonction d'acquisition des données est chargée de recueillir les signaux en provenance des capteurs de procédé et de la commande.

2.2.2 Terminologie de la surveillance

Avant d'aller plus loin dans ce mémoire, il est nécessaire de bien présenter quelques terminologies nécessaires dans le domaine de la surveillance.
Un défaut (Fault) est considéré comme un écart du comportement normal. Il s'exprime

par une déviation d'une propriété ou d'un paramètre caractéristique du procédé. Un défaut est donc une anomalie de comportement qui précède une possible défaillance. Il peut trouver son origine dans les classes suivantes : défaut d'un composant (actionneur ou capteur,..,etc), défaut de la commande ou un défaut dû à une faute d'un opérateur humain de supervision
Une défaillance (Failure) est une anomalie fonctionnelle qui empêche partiellement ou totalement l'aptitude d'un procédé à remplir sa fonction.
Une panne (Break-down) représente les conséquences d'une défaillance dans la réalisation du fonctionnement nominal du procédé. Elle provoque un arrêt complet du procédé dans un état où le système devient incapable d'assurer le service spécifié.
Une défaillance conduit à un défaut puisqu'il existe un écart entre la caractéristique constatée et la caractéristique spécifiée. Inversement, un défaut n'induit pas nécessairement une défaillance. L'objectif de la surveillance et du diagnostic consiste à détecter de façon précoce un défaut avant qu'il ne conduise à un état de défaillance donc de panne.

Nous utilisons le terme défaut dans ce mémoire de thèse puisqu'il inclut la défaillance et la panne.

2.2.3 Classification des méthodes de surveillance

Les méthodes de surveillance se distinguent selon différents critères : la dynamique du procédé (discret, continu ou hybride), l'implémentation du système de surveillance en ligne et/ou hors ligne, la nature de l'information (qualitative et/ou quantitative) et sa distribution (centralisée, décentralisée ou distribuée). En général, ces méthodes sont divisées en deux catégories [Zwingelstein, 1995] :

- Les méthodes sans modèle (model-free methods) qui sont des méthodes soit à base de connaissance, soit des méthodes de traitement du signal.
- Les méthodes à base d'un modèle (model-based methods) qui représentent des méthodes à base de modèles quantitatifs et/ou qualitatifs.

Les méthodes sans modèle considèrent le système comme une "*boîte noire*" et elles ne nécessitent pas la conception d'un modèle représentant le fonctionnement du système. Elles utilisent uniquement un ensemble de mesures et/ou de connaissance heuristique sur le système. Ces méthodes comprennent les méthodes à base de systèmes experts [Combacau, 2001], de réseaux de neurones [Zwingelstein, 1995] ou à base de reconnaissance des formes comme le cas d'utiliser un capteur spécialisé. Celle-ci, la dernière est du type reconnaissance de la signature d'une défaillance ou d'un défaut.
Parmi les méthodes à base de modèles, on peut distinguer les méthodes basées sur des modèles quantitatifs, les méthodes basées sur des modèles qualitatifs et les méthodes combinant ces deux types de modèles.
Les modèles quantitatifs sont utilisés pour l'estimation des paramètres [Isermann,

1984], [Frank, 1996], d'état [Frank, 1990] ou d'espace de parité [Chow and Willsky, 1984] à travers des modèles mathématiques et/ou structurels pour représenter l'information disponible du fonctionnement d'un procédé. Un défaut provoque alors des changements dans certains paramètres physiques du procédé. Les modèles mathématiques comparent les différentes valeurs des variables avec des seuils de détection afin de générer un résidu qui sera fourni au diagnostic. Les avantages de ces méthodes sont tout d'abord la capacité à détecter les variations abruptes et progressives de pannes à travers une analyse de tendance des signaux. De plus, ces méthodes possèdent la capacité de donner une localisation précise du défaut. Par contre, elles nécessitent une information dite "profonde" sur le comportement du système et de ses pannes, rendent les calculs complexes pour le diagnostic en ligne. Elles sont également très sensibles aux erreurs de modélisation.
Les méthodes à base de modèles qualitatifs permettent de représenter le comportement du procédé avec un certain degré d'abstraction à travers des modèles non plus mathématiques mais des modèles de type symbolique. Les modèles qualitatifs doivent représenter de manière qualitative des systèmes continus [Lunze, 2000], discrets et hybrides [David and Alla, 2005], [Koutsoukos et al., 1998] pour que le système de surveillance soit capable de détecter les déviations du fonctionnement normal, diagnostiquer et localiser les défauts. Dans le cadre des systèmes hybrides, nous citons les travaux suivants : [Cocquempot et al., 2003], [Cocquempot et al., 2004] et [Lunze, 2000]. Ces travaux conçoivent un système de surveillance pour les systèmes où les défaillances peuvent affecter soit le comportement continu au sein d'un mode discret, soit la séquence d'états discrets. Pour les systèmes à événements discrets, de nombreuses approches de surveillance ont été proposée. Une présentation des plus importantes approches sera effectuée dans le chapitre 3. En effet, le modèle auquel le système de surveillance est basé, peut être construit en deux approches. La première consiste à construire un modèle à partir de l'observation du comportement du système réel [Klein et al., 2005]. La seconde approche considère un modèle de connaissance comme nous allons le voir dans le chapitre suivant.

2.2.4 Caractéristiques des méthodes de surveillance à base de modèle

Il existe plusieurs méthodes de surveillance à base de modèle dans la littérature. Ces méthodes se basent sur un modèle du comportement normal et/ou défaillant du système. L'observation réelle de l'état courant du système, sujet de surveillance, est comparée avec l'état estimé par le modèle afin de détecter un défaut. Chacune des méthodes de surveillance doit garantir les caractéristiques suivantes :

- Le système de surveillance doit être facile à implémenter,
- Le système de surveillance doit détecter les défauts au plus tôt possible,
- Le système de surveillance doit être réalisable en temps réel,
- Le système de surveillance doit être concevable de manière algorithmique,

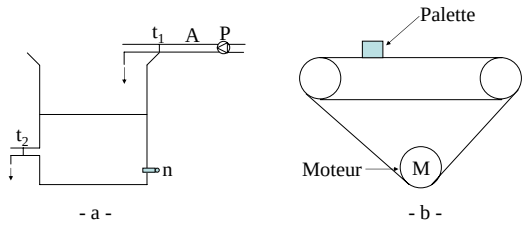

FIG. 2.1 – Exemples des systèmes industriels

– Le nombre de capteurs nécessaire pour la surveillance doit être minimal.

A propos le dernier point, il est à noter que l'information limitée disponible via les capteurs ne permet pas toujours de détecter une situation anormale. Rajouter des capteurs dédiés à la surveillance du procédé est une solution attrayante. Toutefois, ceci ne fait que repousser la limite d'observabilité de l'état réel du procédé.

2.2.5 Classification de défauts

Lorsqu'une application réelle est considérée, rien ne garantit que le fonctionnement du procédé suit l'ensemble des spécifications. Plusieurs raisons peuvent être à l'origine de ce dysfonctionnement : le mauvais fonctionnement suite à l'occurrence de défauts, une mauvaise décision du système de commande ou une instrumentation erronée des capteurs. Ce paragraphe a pour objectif la présentation des types de défauts possibles dans le cadre des systèmes à événements discrets afin d'avoir une idée claire sur le comportement du système après l'occurrence d'un défaut.

Les recherches bibliographiques nous ont conduits à distinguer deux types de défauts [Huang et al., 1996] :

– Les défauts des capteurs discrets et des actionneurs.
– Les défauts des composants constituant la structure principale du système (Figure 2.2.a).

A Défauts des capteurs et des actionneurs

Généralement, les actionneurs et les capteurs ont un mécanisme mécanique, électrique ou électromagnétique. Suite à un défaut dans ces mécanismes, les capteurs et les actionneurs peuvent se bloquer dans une position particulière. Donc, ces défauts se caractérisent par un blocage de la sortie d'un capteur au niveau 1 ou 0 ou par un actionneur qui reste bloqué, inactif ou actif, malgré les ordres envoyés par le système de commande. Dans le système de remplissage présenté dans la figure 2.1, la vanne t_1 peut être sujette au défaut de blocage au niveau 0 correspondant à l'état fermé de t_1. Lorsque ce défaut se produit, le remplissage ne sera pas exécuté bien que le système de commande ait donné

l'ordre de début du remplissage. Le remplissage ne sera exécuté à nouveau que lorsque la vanne est remplacée.
Il est à noter qu'il y a d'autre types de défauts que peut subir un actionneur. Par exemple dans le système de transfert (Figure 2.1.b), l'actionneur (moteur) M peut être accéléré ou ralenti. Ce défaut peut être détecté par les aspects temporels du système de transfert.

B Défauts des composants constituant la structure principale du système

Le système et ses composants peuvent être aussi sujets aux défauts. Quelques défauts comme la défaillance d'un équipement ou la perturbation de l'alimentation électrique, influent sur le fonctionnement de tout le système. Dans le système de remplissage présenté dans la figure 2.1.a, la fuite du bac est un exemple de ce type de défaut. Ce défaut cause une diminution dans le niveau de liquide et affecte la performance du système de bac. Le blocage de la canalisation au point A, représente un autre exemple de ce type de défaut.

C Défauts intermittents et permanents

Une autre classification de défauts peut provenir du recouvrement de défaut.
- **Les défauts permanents** sont définis comme un mauvais fonctionnement d'un composant qui doit être changé ou réparé. Donc, on dit qu'un défaut est permanent si le recouvrement se produit seulement suite à la réparation ou le remplacement du composant fautif.
- **Les défauts intermittents** peuvent, quant à eux, permettre un retour du procédé, du composant ou d'actionneur dans sa dynamique de fonctionnement. Donc, on dit qu'un défaut est intermittent si le recouvrement se produit spontanément. Par exemple, dans la figure 2.1.a, une canalisation bouchée peut être débouchée par une pression interne.

Il est important de distinguer ces deux types de défauts. Le défaut intermittent conduit le système à osciller entre deux états : fautif et normal (non fautif). Le défaut permanent est toujours associé à des événements de recouvrement et le système ne peut pas basculer spontanément de l'état fautif à l'état normal.

Dans notre travail, nous ferons l'hypothèse que le système de commande est exempt de défauts, donc tout origine de défaut du système provient des composants du procédé, des capteurs ou des actionneurs. Nous appelons le défaut qui empêche un composant ou un actionneur de remplir sa fonction par "*défaut interruptible*". Dans les systèmes présentés dans la figure 2.1, nous donnons quelques exemples des défauts interruptibles : le blocage d'une vanne à l'état fermé, le blocage de la canalisation et le blocage de la palette au convoyeur. Dans le dernier cas, le blocage peut être dû à un défaut dans

l'actionneur M ou dans le convoyeur lui-même. Ces défauts interruptibles peuvent être permanents et intermittents.

2.3 Modélisation des systèmes à événements discrets

La classe des systèmes à événements discrets (SED) a été largement étudiée dans la littérature. Son intérêt est justifié par l'existence d'un grand nombre de systèmes réels évoluant à l'occurrence d'événements. Nous nous intéressons à une classe de SED dite "Systèmes à événements discrets commandés". Plusieurs outils de modélisation des SED ont été proposés permettant l'étude et l'analyse des SEDs dans divers contextes. Dans cette section, nous présentons d'abord les SED commandé d'une manière informelle. Ensuite, nous présentons les outils de modélisation des SED nécessaires pour le reste de notre travail. Nous faisons d'abord un rappel sur le modèle automate temporisé. Puis, nous présentons l'automate à chronomètres. Nous passons en revue au cours de cette partie les principales propriétés relatives au modèle automate à chronomètres. Ce modèle permet de capturer quantitativement l'évolution temporelle d'un type de systèmes temps réel. Enfin, nous présentons le modèle des RdP temporels avec un rappel de ses principales propriétés

2.3.1 Présentation des systèmes à événements discrets commandés

Nous nous intéressons par les système à événements discrets commandés. Dans ce paragraphe, nous montrons les éléments qui interviennent dans les SED commandés d'une manière informelle.
Un système à événements discrets commandé est un système constitué d'une association entre un procédé (composants physiques, actionneurs et capteurs) et une unité de commande. La figure 2.2.a illustre l'architecture générale d'un SED commandé. L'unité de commande gère le fonctionnement du procédé selon un ensemble de spécifications établies par l'utilisateur. Elle agit sur les actionneurs du procédé par l'envoi d'une séquence de commandes. En retour, elle reçoit à partir des capteurs un ensemble de signaux permettant de mesurer l'évolution effective du procédé et de déterminer la prochaine séquence de commandes à générer. En effet, nous nous intéressons plus particulièrement aux SED commandés ayant un aspect temporel où le temps apparaît comme paramètre de fonctionnement. L'aspect temporel d'un système est pris en compte à travers la spécification de contraintes temporelles entre les différents événements du procédé (ordres et signaux de capteurs). Ainsi, les spécifications temporelles constituent une partie importantes dans

le bon fonctionnement du système commandé. La description de fonctionnement de ce système peut être exprimé par l'interaction entre les ordres de commandes, les comptes rendus des capteurs et les durées temporelles entre ces événements.
Nous présentons dans la suite un exemple de SED commandé ayant un caractère temporel.

• ***Exemple :*** Considérons un système automatisé de transport de matériaux illustré par la figure 2.2.b. Ce système est composé d'un chariot effectuant une navette entre deux positions. Étant à sa position de départ, le chariot reçoit à partir du système de commande l'ordre de se déplacer vers la position d'arrivée. En effet, le chariot se déplace en avant pendant 5 unités de temps ($u.t$). Lorsque le capteur C_2 indique la présence du chariot à la position finale, la commande lance le chargement du chariot à travers l'ouverture d'une vanne d'alimentation pendant une durée finie de temps puis elle ordonne le retour du chariot à sa position initiale. Cependant, le chariot doit rester 6 $u.t$ dans la position d'arrivée pour permettre son chargement. Lorsque le capteur C_1 détecte le retour du chariot à sa position de départ, le système de commande actionne une pompe pour l'évacuation des matériaux transportés.
Nous remarquons, d'après cet exemple, que le système de commande gère tout le processus de fonctionnement du système, en s'appuyant sur les comptes rendus fournis par les deux capteurs C_1 et C_2. Par ailleurs, les spécifications temporelles constituent une partie importantes dans le bon fonctionnement du système.

2.3.2 L'automate temporisé

Avant de présenter l'automate temporisé, nous introduisons brièvement l'automate à états finis. Cet automate est un graphe étiqueté, dont les nœuds représentent des états et les arcs représentent des transitions entre ces états. Une transition est déclenchée suite à l'occurrence d'un événement.
Le modèle *automate temporisé* est un automate à états finis munis d'un ensemble fini d'horloges [Alur and Dill, 1994], [Bengtsson and Yi, 2004]. Ces horloges sont incrémentés simultanément, avec la même vitesse et peuvent être remises à zéro. Les horloges suivent dans leur évolution une horloge dite *universelle* qui est incrémentée sans être mise à zéro. La dynamique d'une horloge x dans un sommet de l'automate est décrite par l'équation

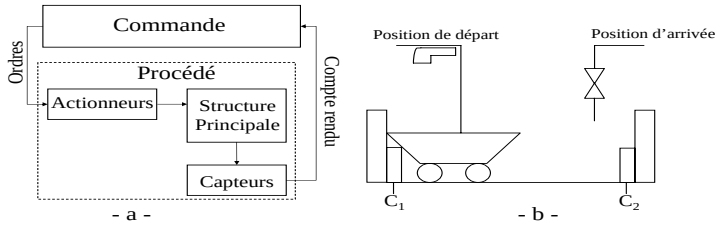

FIG. 2.2 – a- Architecture d'un système commandé b- Un exemple de système commandé

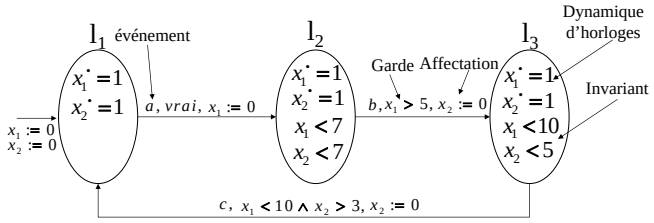

FIG. 2.3 – Exemple d'un automate temporisé

$\dot{x} = 1$. Le franchissement d'une transition de l'automate est effectué suite à l'occurrence d'un événement en entrée et la satisfaction d'une contrainte sur l'ensemble des horloges appelée *garde*. À chaque sommet d'un automate, on peut associer une contrainte sur l'ensemble des horloges appelée *invariant du sommet*. L'automate peut séjourner dans le sommet tant que l'invariant correspondant est vérifié par la valeur des horloges. Le franchissement d'une transition est instantané. Il peut déterminer la mise à zéro de certaines horloges. Ce changement discret de la valeur d'une horloge x_i est modélisé par la relation $x := 0$. Nous illustrons les concepts de base et la notation graphique du modèle automate temporisé à travers l'exemple suivant.

• ***Exemple :*** L'automate de la figure 2.3 comporte 3 sommets l_1, l_2 et l_3. Le franchissement de la transition entre les sommets l_1 et l_2 met à zéro l'horloge x_1. L'invariant du sommet l_2 "$x_1 < 7$" indique que le système peut séjourner dans le sommet l_2 tant que la valeur de l'horloge x_1 est inférieure à 7. La transition entre les sommets l_2 et l_3 ne peut être franchie que si la valeur de l'horloge x_1 satisfait la garde "$x_1 > 5$". Il faut noter que le rôle de l'invariant est d'assurer la progression du système dans le temps et de ne pas permettre par conséquence, le blocage du système dans un sommet. En effet, lorsque la contrainte de l'invariant d'un sommet n'est plus satisfaite par las valeurs des horloges, nous avons fait le choix d'avoir toujours une transition de sortie de ce sommet qui sera validée. D'autres règles de fonctionnement existent [Altisen, 2001].

2.3.3 L'automate à chronomètres

Les automates à chronomètres (stopwatch automata ou SWA) peuvent être vus comme une extension des automates temporisés à l'aide de chronomètres, c'est-à-dire d'horloges pour lesquelles l'écoulement du temps peut être suspendu puis repris plus tard [Henzinger, 1996], [Cassez and Larsen, 2000]. Par exemple, considérons l'automate à chronomètres de la figure 2.4. Dans le sommet l_2 le chronomètre x_3 est actif ; tandis qu'il est inactif dans le sommet l_3.

Nous allons dans la suite définir formellement la syntaxe et la sémantique du modèle automate à chronomètres.

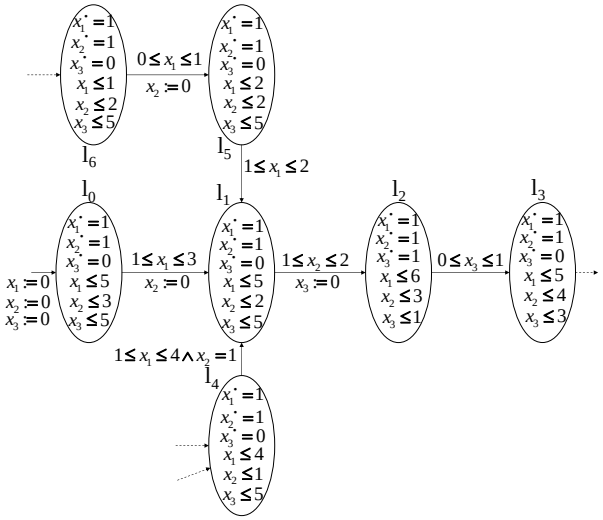

FIG. 2.4 – Exemple d'un automate à chronomètres

A Syntaxe et Sémantique de SWA

Avant d'introduire la syntaxe de l'automate à chronomètres, nous définissons la contrainte atomique. Une *contrainte atomique* sur une variable x est une formule de la forme $x \bowtie c$ pour $c \in \mathbb{Q}_{\geq 0}$ et $\bowtie \in \{<, \leq, \geq, >\}$. Nous notons $C(X)$ l'ensemble des contraintes obtenues par conjonction de contraintes atomiques sur l'ensemble des variables X.

Définition 2.3.1. *(Syntaxe de l'automate à chronomètres) Un automate à chronomètres est un 7-uplet $(L, l_0, X, \Sigma, A, I, Dif)$ où :*

- *L est un ensemble fini de sommets ;*
- *l_0 est le sommet initial ;*
- *X est un ensemble fini de chronomètres à valeurs réelles positives ;*
- *Σ est un ensemble fini d'événements (ou étiquettes) ;*
- *$A \subset L \times C(X) \times \Sigma \times 2^X \times L$ est un ensemble fini de transitions. Soit $a = (l, \delta, \sigma, R, l') \in A$. a est la transition reliant le sommet l au sommet l', avec la garde δ, l'événement σ et l'ensemble de chronomètres à remettre à zéro R.*
- *$I \in C(X)^L$ associe un invariant à chaque sommet ;*
- *$Dif \in (\{0,1\}^X)^L$ associe à chaque sommet l'activité des chronomètres dans ce sommet. $\dot{X}$ désigne l'ensemble des dérivées par rapport au temps des variables de X : $\dot{X} = (Dif(l)(x))_{x \in X}$.*

□

Dans un souci de simplicité, étant donnés un sommet l, un chronomètre x et $b \in \{0, 1\}$, nous noterons $Dif(l)(x) = b$ par $\dot{x} = b$. Soit v une valuation, R un sous-ensemble de X. Nous noterons par $v' = v[R \leftarrow 0]$ la valuation qui associe 0 pour chaque chronomètre $x_i \in R$ et $v' = v(x_i)$ si $x_i \notin R$.

Définition 2.3.2. *(Sémantique d'un SWA) La sémantique d'un automate à chronomètres $\mathbb{A}$ est définie sous la forme d'un système de transitions temporisé $S_A = (Q_A, q_{a_0}, \rightarrow)$ où :*
- $Q_A = L \times (\mathbb{R}^+)^X$;
- $q_{a_0} = (l_0, \overline{0})$ *est l'état initial ;*
- $\rightarrow \in Q_A \times (\Sigma \cup \mathbb{R}) \times Q_A$ *est la relation définie pour $\sigma \in \Sigma$ and $d \in \mathbb{R}^+$, par :*
- *la relation de transition continue :*

$$(l,\upsilon_A) \xrightarrow{d} (l, \upsilon'_A) \text{ ssi}$$
$$\begin{cases} \upsilon'_A = \upsilon_A + \dot{X} * d, \\ \forall t \in [0,d], I(l)(\upsilon_A + \dot{X} * t) = vrai. \end{cases}$$

- *la relation de transition discrète :*

$$(l,\upsilon_A) \xrightarrow{\sigma,\delta,R} (l', \upsilon'_A) \text{ ssi } \exists a = (l, \delta, \sigma, R, l') \in A \text{ tel que :}$$
$$\begin{cases} \delta(\upsilon_A) = vrai, \\ \upsilon'_A = \upsilon_A[R \leftarrow 0] \\ I(l')(\upsilon'_A) = vrai. \end{cases}$$

□

B Composition Synchrone

Nous allons définir dans la suite de cette section une opération fondamentale dans le cadre de la modélisation des SED qui est *la composition synchrone*. Cette opération s'avère nécessaire et même incontournable lorsqu'il s'agit de modéliser des systèmes complexes (i.e. des systèmes ayant un très grand nombre d'états). Nous présentons dans ce qui suit, une définition formelle de la composition synchrone de deux automates à chronomètres. Il est à noter que l'automate résultant de la composition est aussi un automate à chronomètres.

Définition 2.3.3. *Soit deux automates à chronomètres $\mathbb{A}_1$ et $\mathbb{A}_2$ tel que $\mathbb{A}_1 = (L_1, l_{01}, X_1, \Sigma_1, A_1, I_1, Div_1)$ et $\mathbb{A}_2 = (L_2, l_{02}, X_2, \Sigma_2, A_2, I_2, Div_2)$. La composition synchrone de ces deux automates à chronomètres notée $\mathbb{A} = \mathbb{A}_1 \parallel \mathbb{A}_2$ avec $\mathbb{A} = (L, l_0, X, \Sigma, A, I, Div)$ tel que :*
- $L = L_1 \times L_2$.
- $\Sigma = \Sigma_1 \cup \Sigma_2$.
- $X = X_1 \cup X_2$.
- $l_0 = (l_{01}, l_{02})$.

On note $\Sigma' = \Sigma_1 \cap \Sigma_2$. *L'ensemble des transitions* A *est défini par :*

$$Si \begin{cases} \sigma \in \Sigma' \\ l_i \xrightarrow{\sigma,\delta_1,R_1} l_j \in A_1 \\ l'_i \xrightarrow{\sigma,\delta_2,R_2} l'_j \in A_2 \end{cases} \Rightarrow \begin{cases} (l_i, l'_i) \xrightarrow{\sigma,\delta_1\wedge\delta_2,R_1\cup R_2} (l_j, l'_j) \in A \\ I((l_i, l'_i)) = I_1(l_i) \wedge I_2(l'_i) \\ I((l_j, l'_j)) = I_1(l_j) \wedge I_2(l'_j) \\ Div((l_i, l'_i)) = Div_1(l_i) \wedge Div_2(l'_i) \\ Div((l_j, l'_j)) = Div_1(l_j) \wedge Div_2(l'_j) \end{cases}$$

$$Si \begin{cases} \sigma \in \Sigma_1 - \Sigma' \\ l_i \xrightarrow{\sigma,\delta_1,R_1} l_j \in A_1 \\ \forall l_2 \in L_2 \end{cases} \Rightarrow \begin{cases} (l_i, l_2) \xrightarrow{\sigma,\delta_1,R_1} (l_j, l_2) \in A \\ I((l_i, l_2)) = I_1(l_i) \wedge I_2(l_2) \\ I((l_j, l_2)) = I_1(l_j) \wedge I_2(l_2) \\ Div((l_i, l_2)) = Div_1(l_i) \wedge Div_2(l_2) \\ Div((l_j, l_2)) = Div_1(l_j) \wedge Div_2(l_2) \end{cases}$$

$$Si \begin{cases} \sigma \in \Sigma_2 - \Sigma' \\ \forall l_1 \in L_1 \\ l'_i \xrightarrow{\sigma,\delta_2,R_2} l'_j \in A_2 \end{cases} \Rightarrow \begin{cases} (l_1, l'_i) \xrightarrow{\sigma,\delta_2,R_2} (l_1, l'_j) \in A \\ I((l_1, l'_i)) = I_1(l_1) \wedge I_2(l'_i) \\ I((l_1, l'_j)) = I_1(l_1) \wedge I_2(l'_j) \\ Div((l_1, l'_i)) = Div_1(l_1) \wedge Div_2(l'_i) \\ Div((l_1, l'_j)) = Div_1(l_1) \wedge Div_2(l'_j) \end{cases}$$

□

D'après cette définition, les deux automates à chronomètres $\mathbb{A}_1$ et $\mathbb{A}_2$ évoluent indépendamment pour un ensemble d'action $(\Sigma - \Sigma')$ et évoluent d'une manière synchrone (les deux ensemble) pour un ensemble commun d'actions Σ'. Cette définition de la composition synchrone peur être généralisée dans le cas de n automates à chronomètres ($n \geq 2$) en effectuant des compositions successives des couples d'automates. Cette opération nous sera utile dans le cadre de modélisation des SED commandés. En effet, nous allons voir qu'une des techniques de modélisation proposées (dans notre méthode de surveillance) se base essentiellement sur la composition synchrone entre plusieurs modèles élémentaires du système.

C Analyse d'atteignabilité dans l'automate à chronomètres

Un automate à chronomètres admet deux types d'évolution possibles à partir d'un sommet :

- Soit il séjourne dans ce même sommet en laissant le temps s'écouler. Dans ce cas, il faut que les valeurs des chronomètres vérifient la contrainte de l'invariant de ce sommet. Les chronomètres actifs de l'automate avancent d'une valeur égale à la durée du séjour dans le sommet.

– Soit une transition de sortie est franchie. Dans ce cas, il faut vérifier au préalable que les valeurs des chronomètres vérifient la garde de la transition. Puis, après que l'événement est réalisée, il faut remettre à zéro les chronomètres spécifiées sur la transition.

En effet, un sommet d'un automate à chronomètres ne désigne pas un état de l'automate. Un état est défini par un sommet et une valuation des chronomètres, noté par un couple (l, υ) où l désigne un sommet de l'automate et υ désigne une valuation de chronomètres vérifiant l'invariant de ce sommet ($\upsilon \models I(l)$).

L'automate peut séjourner dans ce sommet tant que son invariant est satisfait par la valuation des chronomètres. Ce qui fait qu'un sommet ne correspond pas à un seul état mais à un espace d'états. Les valeurs que les chronomètres peuvent prendre pendant le séjour dans un sommet décrivent un espace de chronomètres. Nous désignons par une région dans un sommet l_n et nous notons (l_n, E_m), un espace de chronomètres E_m dans un sommet l_n. Nous notons par E_0 l'espace de chronomètres correspondant au sommet initial l_0 de l'automate.

Généralement, l'étude d'un système modélisé par un automate à chronomètres est basée sur l'analyse de l'atteignabilité des états de l'automate. Pour savoir si une région E est atteignable depuis une région E_0, deux méthodes peuvent être utilisées. La première méthode est la méthode *d'analyse en avant*. Cette méthode est basée sur le calcul de l'espace de tous les états qui peuvent être atteints depuis les états appartenant à la région E_0. L'ensemble de ces états est appelé régions successeurs de la région E_0. Si l'espace calculé contient des états qui appartiennent également à la région E, alors on peut dire que cette région est atteignable depuis la région E_0.

La deuxième méthode est basée sur le calcul de l'ensemble de tous les états depuis lesquels on peut atteindre des états de la région E. L'ensemble de ces états est appelé prédécesseurs de la région E. Si l'espace calculé contient des états qui appartiennent également à la région E_0, alors on peut conclure que la région E est atteignable depuis E_0. Cette méthode, duale à la méthode d'analyse en avant, est appelée méthode *d'analyse en arrière*. Les méthodes sont détaillées dans [Alur et al., 1995] et [SAVA, 2001]. Ces procédures d'analyse d'atteignabilité ont été implémentées dans les logiciels dédiés à la vérification des systèmes temporisés et hybrides. Parmi ces logiciels nous citons Hytech [Henzinger et al., 1997] et PHAVer [Frehse, 2005], [Asarin et al., 2006]. Dans notre travail, nous nous intéressons seulement aux procédures de calcul des successeurs et prédécesseurs d'une région, que nous présentons par la suite.

a. Principe du calcul des successeurs d'une région

Nous allons présenter au cours de ce paragraphe le principe de calcul des successeurs d'une région de chronomètres de l'automate. Un état d'un automate à chronomètres peut avoir deux types de successeurs : continus et discrets.

Un *successeur continu* d'un état est obtenu en restant dans le même sommet et en laissant

le temps s'écouler.

Définition 2.3.4. *L'état $(l, \upsilon + t)$ est un successeur continu de l'état (l, υ) si :*

$$\exists t \in \mathbb{R}^+ \mid \forall t' \leq t, \upsilon + t' \models I(l).$$

□

Un *successeur discret* d'un état est obtenu par franchissement d'une transition à partir du sommet relatif à cet état.

Définition 2.3.5. *L'état (l', υ') est le successeur discret de l'état (l, υ) par le franchissement de la transition $a = l \xrightarrow{\sigma,\delta,R} l'$ si :*

$$(\upsilon \models \delta) \wedge ((\upsilon' = \upsilon[R \leftarrow 0]) \wedge \upsilon' \models I(l'))$$

□

En effet, la transition a ne peut être franchie que si sa garde est satisfaite par la valeur des chronomètres ($\upsilon \models \delta$). Lors du franchissement de la transition, la valeur des chronomètres est modifiée par la mise à zéro des chronomètres de l'ensemble R. La valeur des horloges doit satisfaire l'invariant du sommet destination l'.
Nous allons maintenant étendre la définition des successeurs d'un état aux successeurs d'un ensemble d'états ou d'une région.

Définition 2.3.6. *L'ensemble des états atteignables à partir de tout état $(l, \upsilon) \in (l, E_m)$ en laissant le temps s'écouler tout en restant dans le même sommet est appelé successeur continu de la région (l, E_m). Cet ensemble, noté $Succ_t(l, E_m)$ est défini par l'expression suivante :*

$$\upsilon' \models Succ_t(l, E_m) \text{ ssi}$$
$$\exists t \in \mathbb{R}^+, \upsilon' - t \models E_m \wedge \forall t' \in \mathbb{R}^+, t' \leq t \Rightarrow \upsilon' - t' \models I(l)$$

□

Exemple : Considérons l'automate à chronomètres de la figure 2.4. Soit (l_1, E_1) une région dans le sommet l_1. Cette région, représentée par la figure 2.5.b, est décrite par l'espace d'horloges suivant : $E_1 = 1 \leq x_1 \leq 2 \wedge 0 \leq x_2 \leq 1$. Le successeur continu de la région (l_1, E_1) est :

$$Succ_t(l_1, E_1) = \begin{cases} \exists t \in \mathbb{R}^+.E_1[x_1 - t, x_2 - t].I(l_0) \\ \exists t \in \mathbb{R}^+.E_3[1 \leq x_1 - t \leq 2 \wedge 0 \leq x_2 - t \leq 1] \wedge x_1 \leq 5 \wedge x_2 \leq 2 \\ 0 \leq x_1 - x_2 \leq 2 \wedge 1 \leq x_1 \wedge 0 \leq x_2 \leq 2 \end{cases}$$

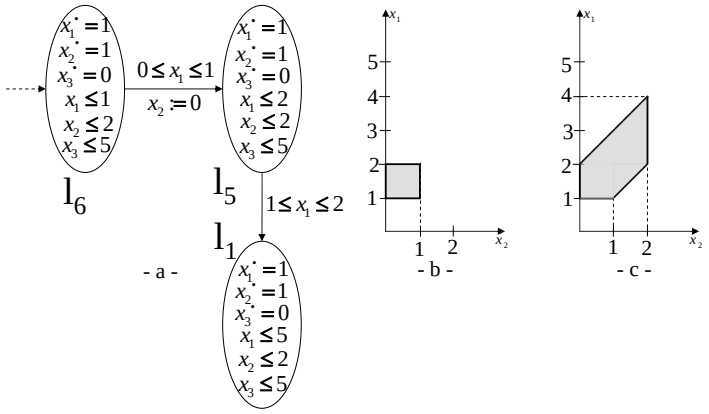

FIG. 2.5 – Successeur continu d'une région : a- La partie considérée de l'automate présenté dans la figure 2.4 b- l'espace E_1 c- $Succ_t(l_1, E_1)$.

Définition 2.3.7. *Soit (l_n, E_m) une région et $a = l_n \xrightarrow{\sigma,\delta,R} l_p$ une transition. L'ensemble des états atteignables depuis tout état $(l_n, \upsilon) \in (l_n, E_m)$ en franchissant la transition a est appelé successeur discret de la région (l_n, E_m). Cet ensemble, noté $Succ_d(l_n, E_m)$ est défini par l'expression :*

$$\upsilon' \models Succ_d(l_n, E_m) \text{ ssi } \exists \upsilon \text{ tel que } \upsilon \models (E_m \wedge \delta) \wedge (\upsilon' = \upsilon[R \leftarrow 0] \wedge \upsilon' \models I(l_p))$$

□

Exemple : Nous allons calculer le successeur discret de la région (l_1, E_2), représenté par la figure 2.6.b, correspondant au franchissement de la transition $a = l_1 \xrightarrow{1 \le x_2 \le 2, x_3 := 0} l_2$.

$$Succ_d(l_1, E_2) = \begin{cases} \exists x'_1, x'_2, x'_3 . E_2[x'_1, x'_2] \wedge x'_2 = x_2 \wedge x'_1 = x_1 \wedge x_3 = 0 \\ \exists x'_1, x'_2, x'_3 . [0 \le x'_1 - x'_2 \le 3 \wedge x'_1 \ge 1 \wedge 1 \le x'_2 \le 2] \\ \wedge x'_2 = x_2 \wedge x'_1 = x_1 \wedge x_3 = 0 \\ \exists x'_1 . [0 \le x_1 - x_2 \le 3 \wedge x_1 \ge 1 \wedge 1 \le x_2 \le 2] \wedge x_1 = x'_1 \wedge x_3 = 0 \\ 0 \le x_1 - x_2 \le 3 \wedge x_1 \ge 1 \wedge 1 \le x_2 \le 2 \wedge x_3 = 0 \end{cases}$$

Dans le calcul précédent, x'_1 et x'_2 désignent les anciennes valeurs des chronomètres avant le franchissement de la transition entre l_1 et l_2. x_1 et x_2 désignent les nouveaux valeurs des chronomètres après le franchissement de cette transition.

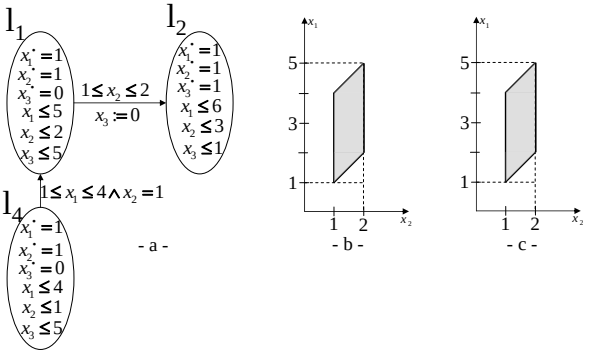

FIG. 2.6 – Successeur discret d'une région : a- La partie considérée de l'automate présenté dans la figure 2.4 b- l'espace E_2 c- Projection de $Succ_d(l_1, E_2)$ sur x_1, x_2

b. Principe du calcul des prédécesseurs d'une région

Tout état depuis lequel on peut atteindre un état donné est un prédécesseur de cet état. Il y a deux types de prédécesseurs : continus et discrets.
Un état depuis lequel on peut atteindre un état donné tout en laissant le temps s'écouler en restant dans le même sommet est un prédécesseur continu de cet état.

Définition 2.3.8. *L'état* (l_m, υ) *est un prédécesseur continu de l'état* $(l_m, \upsilon + t)$ *si :*

$$\exists t \in \mathbb{R}^+ \text{ tant que } \forall t' \leq t, \upsilon + t' \models I(l_m)$$

□

La notion de prédécesseur continu est duale à celle de successeur continu.
Tout état depuis lequel on peut atteindre un état donné par le franchissement d'une transition est un prédécesseur discret de cet état.

Définition 2.3.9. *L'état* (l_m, υ) *est un prédécesseur discret de l'état* (L_{m+1}, υ') *par le franchissement de la transition* $a_{m,m+1} = (l_m, \sigma, \delta_{m,m+1}, R_{m,m+1}, l_{m+1})$ *si :*

$$(l_m, \sigma, \delta_{m,m+1}, R_{m,m+1}, l_{m+1}) \in A \wedge (\upsilon \models \delta_{m,m+1}) \wedge (\upsilon' = \upsilon[R_{m,m+1} \leftarrow 0]) \wedge (\upsilon[R_{m,m+1} \leftarrow 0] \models I(l_{m+1}))$$
$$(l_m, \upsilon) \rightarrow (L_{m+1}, \upsilon')$$

□

La notion de prédécesseur discret est duale à celle de successeur discret. De la même manière que pour les états on peut définir les prédécesseur continus d'une région

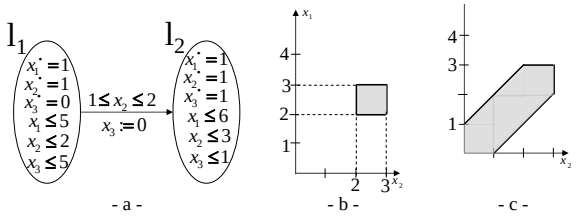

FIG. 2.7 – Prédécesseur continu d'une région : a- La partie considérée de l'automate présenté dans la figure 2.4 b- l'espace E_3 c- $Pre_t(l_2, E_3)$.

Définition 2.3.10. *L'ensemble des états à partir desquels on peut atteindre n'importe quel état* $(l_m, \upsilon) \in (l_m, E_n)$ *en laisaant le temps s'écouler, tout en restant dans le même sommet, est appelé prédécesseur continu de la région* (l_m, E_n)*. Cet ensemble, noté* $Pre_t(l_m, E_n)$*, est défini par l'expression :*

$$\upsilon' \in Pre_t(l_m, E_n) \text{ ssi } \exists t \in \mathbb{R}^+ . \upsilon' + t \in E_n \wedge \forall t' \in \mathbb{R}^+ . t' \leq t \Rightarrow \upsilon' + t' \models I(l_m)$$

□

Exemple : Considérons maintenant la région (l_2, E_3) dans le sommet l_2 de l'automate à chronomètres illustré dans la figure 2.4. L'espace E_3 représenté par la figure 2.7.b, est défini par les contraintes suivantes sur les valeurs des chronomètres : $E_3 = (2 \leq x_1 \leq 3 \wedge 2 \leq x_2 \leq 3 \wedge 0 \leq x_3 \leq 1)$. Le prédécesseur de cette région est calculé de la manière suivante :

$$Pre_t(l_2, E_3) = \begin{cases} \exists t \in \mathbb{R}^+ . E_3[x_1 + t, x_2 + t, x_3 + t] . I(l_2) \\ \exists t \in \mathbb{R}^+ . [t \geq 0 \wedge 2 \leq x_1 + t \leq 3 \wedge 2 \leq x_2 + t \leq 3 \wedge 0 \leq x_3 + t \leq 1] \\ \wedge x_1 \leq 6 \wedge x_2 \leq 3 \wedge x_3 \leq 1 \\ 0 \leq x_1 \leq 3 \wedge 0 \leq x_2 \leq 3 \wedge 0 \leq x_3 \leq 1 \wedge x_2 - x_1 \leq 1 \wedge x_1 - x_2 \leq 1 \\ \wedge 1 \leq x_2 - x_3 \leq 3 \end{cases}$$

L'espace des chronomètres $Pre_t(l_2, E_3)$ est présenté dans la figure 2.7.c.

Définition 2.3.11. *Soit* (l_{m+1}, E_n) *une région et* $a_{m,m+1} = l_m \xrightarrow{\sigma, \delta_{m,m+1}, R_{m,m+1}} l_{m+1}$ *une transition. L'ensemble des états atteignables à partir desquels on peut atteindre n'importe quel état* $(l_{m+1}, \upsilon) \in (l_{m+1}, E_n)$ *en franchissant la transition* $a_{m,m+1}$ *est appelé prédécesseur discret de la région* (l_{m+1}, E_n)*. Cet ensemble, noté* $Pre_d(l_{m+1}, E_n)$ *est défini par l'expression :*

$$\upsilon' \models Pre_d(l_{m+1}, E_n) \text{ ssi } (\upsilon' \models (\delta_{m,m+1} \wedge I(L_m))) \wedge (\upsilon = \upsilon'[R_{m,m+1} \leftarrow 0] \models E_n)$$

□

Remarque 2.3.1. Nous pouvons définir une trajectoire dans un automate à chronomètres par la suivante.

Définition 2.3.12. *Une trajectoire dans un automate à chronomètres SWA est une séquence finie ou infinie des états* $q_0 \longrightarrow_{t_0} q_1 \longrightarrow_{t_1} .....q_i \longrightarrow_{t_i} .....$ *où* $q_i = (l_i, \upsilon_i)$ *et pour* $i \geq 0$ *:*

1. *pour* $t_{i-1} \leq t \leq t_i$ υ_i *vérifie* $I(l_i)$ *(I est l'invariant de* l_i*).*
2. *l'état* q_{i+1} *est le successeur (continu ou discret) de l'état* q_i*.*

□

D Automate inversé

Il est possible de définir pour un automate à chronomètres $\mathbb{A}$ son automate inversé $\mathbb{A}^{-1}$. Généralement, la construction de l'automate inversé $\mathbb{A}^{-1}$ consiste à inverser la causalité de l'automate et les dynamiques des chronomètres (la dynamique 1 est remplacée par la valeur -1). L'inversion permute la garde et la remise à zéro d'une transition. Les autres composants restent inchangés.

Soit Q l'ensemble des états de l'automate $\mathbb{A}$. L'automate inversé $\mathbb{A}^{-1}$ de l'automate $\mathbb{A}$ a la propriété suivante [Henzinger et al., 1998] : $\forall$ q_1, $q_2 \in Q$ tel que $q_1 \xrightarrow{\pi}_{\mathbb{A}} q_2$ où $\pi \in \Sigma \cup \mathbb{R}_{\geq 0}$, nous avons :

$$q_1 \xrightarrow{\pi}_{\mathbb{A}} q_2 \text{ ssi } q_2 \xrightarrow{\pi}_{\mathbb{A}^{-1}} q_1$$

Il s'ensuit que pour chaque région E de l'automate $\mathbb{A}$, pour $\pi \in \Sigma \cup \mathbb{R}_{\geq 0}$, nous avons :

$$Succ^{\pi}_{\mathbb{A}}(E) = Pre^{\pi}_{\mathbb{A}^{-1}}(E) \text{ et } Pre^{\pi}_{\mathbb{A}}(E) = Succ^{\pi}_{\mathbb{A}^{-1}}(E)$$

Donc, l'analyse en arrière d'une région dans un automate, pour $\pi \in \Sigma \cup \mathbb{R}_{\geq 0}$ est équivalente à l'analyse en avant de cette région dans l'automate inversé.

2.3.4 Les réseaux de Petri temporels

Parmi les techniques proposées pour modéliser et analyser des systèmes dans lesquels le temps apparaît comme paramètre, une est largement utilisée : les réseaux de Petri temporels [Merlin, 1974]. Sa puissance de modélisation résulte de sa capacité à représenter d'une manière intuitive et naturelle les principaux mécanismes des systèmes à événements discrets : parallélisme, synchronisation et partage des resources. Par la suite, nous faisons une brève présentation du modèle RdP temporels.

Les réseaux de Petri temporels sont obtenus depuis les RdP en associant deux dates *min* et *max* à chaque transition. Supposons que la transition t soit devenue sensibilisée pour la dernière fois à la date θ, alors t ne peut pas être franchie avant la date $\theta + min$ et doit l'être au plus tard à la date $\theta + max$, sauf si le franchissement d'une autre transition a désensibilisé t avant que celle-ci ne soit franchie. Les réseaux temporels expriment des spécifications "en délais". En explicitant débuts et fins d'actions, il peuvent aussi exprimer des spécifications "en durées". Leur domaine d'application est donc large.

La définition et quelques concepts de bases des réseaux de Petri temporels sont rappelés dans la suite. Ceci sera utilisé dans la suite dans le mémoire.

Définition 2.3.13. *Un RdP temporel est un 6-uplet* $N = \langle P, T, {}^\bullet(.), (.)^\bullet, M_0, I_s \rangle$ *tel que :*
- P *est un ensemble fini et non vide de places ;*
- T *est un ensemble fini et non vide de transitions.*
- ${}^\bullet(.)$, $(.)^\bullet$ *sont respectivement les fonctions d'incidence amont et aval ;*
- $M_0 \in \mathbb{N}^{card|P|}$ *est le marquage initial du réseau ;*
- I_s *est une fonction associant à chaque transition un intervalle donnant son instant de tir au plus tôt* $EFT(t_i) \in \mathbb{Q}^+$ *et au plus tard* $LFT(t_i) \in \mathbb{Q}^+ \cup \{\infty\}$.

□

On parle ici d'intervalle de franchissement statique, car en étudiant la dynamique du RdP, ces intervalles évoluent dans le temps, et on parle dans ce cas d'intervalle dynamique de franchissement.

Nous allons souligner dans la suite deux points : le premier concerne la notion de nouvelle sensibilisation d'une transition et, le second est celle d'évolution temporelle d'une horloge x associée à une transition t. Ces deux concepts permettent de montrer les différences essentielles apportées par l'extension de RdP temporel proposée au chapitre 5.
Une transition t est dite nouvellement sensibilisée par le franchissement de la transition t' à partir du marquage M, si t est sensibilisée par le nouveau marquage $M - {}^\bullet(t') + (t')^\bullet$ mais ne l'était pas par le marquage $M - {}^\bullet(t')$. Étant donné le RdP temporel présenté dans la figure 2.8.a. Supposons que t_1 est franchissable. Les transitions t_1 et t_2 sont sensibilisées par le marquage M et par le marquage $M - {}^\bullet(t_1) + (t_1)^\bullet$ mais pas par $M - {}^\bullet(t_1)$. Les transitions t_1 et t_2 sont donc nouvellement sensibilisées par le franchissement de t_1. Sur la figure 2.8.b, t_1 et t_2 sont sensibilisées par le marquage M et par le marquage $M - {}^\bullet(t_1) + (t_1)^\bullet$ mais aussi par $M - {}^\bullet(t_1)$. Dans ce cas, t_1 est nouvellement sensibilisée par le franchissement de t_1 (car elle est la transition franchissable) mais pas t_2 : t_2 reste sensibilisée [Roux, 2005]. Nous pouvons remarquer dans la figure 2.8 qu'une stratégie *monoserveur* a été adoptée. Celle-ci est appliquée explicitement par les places en boucle sur la transition. Cette stratégie sera considérée tout au long du mémoire. Pour simplifier

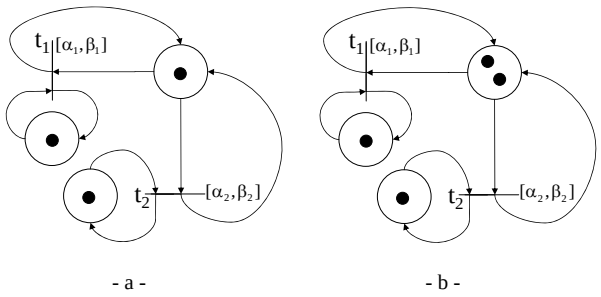

FIG. 2.8 – Example de transitions nouvellement sensibilisées

le graphisme, les places qui sont en boucle sur la transition ne seront pas représentées.
À chaque transition d'un RdP temporel est associée une horloge. L'horloge x_i associée à la transition t_i dans un RdP temporel est active lors que la transition t_i est validée (le nombre de jetons, de chaque place en amont de t_i est plus grand ou égal à la valuation de l'arc entre cette place et la transition). Elle reste active tant que t_i est validée. L'horloge x_i est initialisée lorsque la transition t_i est nouvellement sensibilisée. La désensibilisation de t_i fige l'horloge x_i.

Plusieurs méthodes ont été développées dans la littérature afin d'analyser d'atteignabilité dans le RdP temporel. Parmi les travaux élaborés dans ce cadre, nous citons l'approche appelée *la méthode énumérative* [Berthomieu and Diaz, 1991]. Dans cette méthode, l'état d'un RdP temporel peut être présenté sous la forme d'un couple $c = (M, D)$, où M est le marquage du réseau de D est le domaine de franchissement. c est appelé *une classe d'état*. La construction des classes d'états est basée sur un temps relatif. Autrement dit : dans le domaine de franchissement d'une classe d'états, on contraint les instants de franchissement des transitions sensibilisées exprimés par rapport à la date d'entrée dans la classe. Les valeurs des horloges des états précédent sont oubliées. En effet, il est possible de reformuler l'algorithme calculant cet espace afin de produire des systèmes en "dates absolues" de franchissement, plustôt relative. Dans ce cas, on introduira une variable supplémentaire, *start*, qui représente la date d'initialisation. L'outil *Tina* [Berthomieu et al., 2004] propose toutes les constructions de classes discutées au dessus.
Les auteurs dans [Gardey et al., 2006] ont proposé une approche du calcul de l'espace d'états d'un RdP temporels. La méthode est basée sur les zones et sur le calcul en avant de l'espace d'états de l'automate temporisé correspondant au modèl RdP temporels. Les auteurs ont d'abord considéré la sous-classe des RdP temporels pour laquelle les intervalles de franchissement des transitions sont bornés. Ceci est afin d'adapter l'algorithme utilisé pour l'analyse en avant des automates temporisés. Ils ont ensuite considéré la classe générale des RdP temporels (autorisant l'infini en borne supérieure des intervalles de franchissement) pour laquelle un opérateur de surapproximation des zones peut être

nécessaire pour assurer la terminaison des algorithmes. Ils ont prouvé qu'avec cette sur-approximation l'algorithme d'analyse termine et est exact vis-à-vis de l'accessibilité des marquages. Les travaux présentés dans [SAVA, 2001] considèrent des RdPs bornés dont le réseau sous-jacent n'est pas nécessairement sauf et proposent un algorithme calculant le graphe des marquages d'un RdP sous la forme d'un automate temporisé (avec une horloge par transition du RdP temporels). Ensuite, les techniques d'analyse d'atteignabilté de l'automate temporisé, ont été utilisés afin d'explorer les états accessibles du modèle RdP temporels.
Pour les deux dernières méthodes que nous venons de décrire, les traductions proposées sont limitées à des réseaux bornés.

Remarque 2.3.2. Une stratégie de *multiserveur* peut être considérée. Une transition est multi-sensibilisée par un marquage M s'il existe un entier $k > 1$ tel que $M \geq k.^{\bullet}(t)$. Dans [Berthomieu, 2001], plusieurs interprétations opératoires de la multi-sensibilisation sont distinguée. À la transition t sensibilisée par un marquage $M \geq k.^{\bullet}(t)$ seront associées k horloges. Afin de compléter l'interprétation, il est nécessaire de considérer les instances de sensibilisation. Plusieurs choix peuvent être envisagés : les instances de sensibilisation peuvent être considérées comme indépendantes, ou encore peuvent être ordonnées selon leurs dates de création (stratégie première - sensibilisée première - franchie). Ordonner les instances de franchissement selon leur âge semble raisonnable lorsque ces instances représentent des occurrences d'événements. Le lecteur est renvoyé à [Berthomieu, 2001] pour plus de précisions.

2.4 Conclusion

L'évolution des systèmes réels et surtout les systèmes commandés qui nous intéressent, rend le système de surveillance indispensable pour assurer une rentabilité maximale de ces systèmes. Ce chapitre a présenté, dans un premier temps, notre vision de la surveillance par rapport à la supervision et au diagnostic. En effet, nous considérons la surveillance comme une composition de deux modules : l'acquisition des données et la détection des défauts. Dès lors, une terminologie appropriée à notre vision de la surveillance a été présentée afin de définir chaque terme utilisé dans ce mémoire.
Ensuite, une première étude de classification des différentes méthodes de surveillance a été présentée dans ce chapitre. Deux grandes catégories de méthodes de surveillance ont été dégagées : méthodes sans modèles et méthodes à base de modèles. Les méthodes sans modèles sont essentiellement basées sur les connaissances de l'expert. Les méthodes à bases de modèles sont, quant à elles, représentées par des modèles quantitatifs et/ou qualitatifs. Les méthodes à base de modèles qualitatifs permettent une modélisation du procédé et de la commande. Ils sont par conséquent représentatifs des Systèmes à Événements Discrets (SED) autour desquels ce mémoire de thèse est centré.

Dans notre étude, nous nous intéressons aux SED commandés pour lesquels les spécifications temporelles constituent une partie importantes dans le bon fonctionnement de système. Nous avons donc présenté quelques techniques permettant de représenter des système dans lesquels le temps apparaît comme paramètre. Ce sont les automates temporisés, les automates à chronomètres et les Réseaux de Petri temporels. L'automate à chronomètres sera utilisé pour modéliser le comportement de système à surveiller. Le choix de cet outil sera justifié dans le chapitre 4. La construction du système de surveillance applique les techniques d'analyse d'atteignabilité de l'automate à chronomètres. Ces techniques sont présentées aussi dans ce chapitre. Dans le cadre de la thèse, nous proposons une extension de réseaux de Petri temporels, pour modéliser le comportement considéré de système à surveiller. Pour cela, un rappel rapid des RdPs temporels est présenté.

Nous allons dresser dans le chapitre 3 un état de l'art des travaux menés à partir d'un modèle de système à événements discrets.

Chapitre 3

SUR LA SURVEILLANCE DES SYSTÈMES À ÉVÉNEMENTS DISCRETS : UN ÉTAT DE L'ART

3.1 Introduction et problématique de la surveillance des SED

Pour assurer une rentabilité maximale des systèmes temps-réel, il convient d'assurer un système de surveillance en ligne afin de détecter les défauts. Ceci nous permet de déterminer les interventions préventive et corrective nécessaires. Comme nous l'avons écrit dans le chapitre 2, les approches à base de modèle ont été largement étudiées dans la littérature. Dans ce chapitre, nous présentons un état de l'art des travaux menés à partir d'un modèle de système à événements discrets. Au cours de cette étude, la classification des méthodes de surveillance est faite à base des informations contenues dans le modèle. Nous avons alors les méthodes à base de modèles de comportement dit "complet" et celles à base de modèles de comportement normal. Le modèle complet d'un système décrit son comportement normal et fautif.
En effet, les méthodes basées sur un modèle complet, ont pour objectif de détecter et diagnostiquer les défauts apparaissant dans le système. La vérification que le modèle est capable de diagnostiquer chaque défaut appartenant à l'ensemble des défauts prédéfinis, nécessite d'introduire la notion de diagnosticabilité. Par ailleurs, les méthodes basées sur un modèle du comportement normal, ont pour objectif seulement de détecter les défauts. Chaque violation du comportement normal représente un défaut.
La conception des méthodes de surveillance des SED repose sur un certain nombre de critères touchant au type de modèle, à la structure de la prise de décision et l'architecture du système. Donc, nous mettons l'accent d'abord, sur les principaux travaux existant dans la littérature. Ensuite, nous présentons les approche de construction d'un système de surveillance, les approches d'implémentation et la structure de prise la décision. Enfin, une discussion termine ce chapitre afin de positionner l'approche que nous proposerons dans le chapitre 4 pour la surveillance des systèmes à événements discrets commandés.

3.2 Méthodes de surveillance des SED

Nous allons présenter au cours de cette section, les méthodes de surveillance des systèmes à événements discrets. Ceci nous conduit à détailler le modèle de base du système de surveillance. Nous allons discuter pour chaque méthode présentée ici, le modèle retenu et la méthode de détection ou de diagnostic des défauts. La classification de méthodes de surveillance se basent sur les informations présentées dans le modèle de base. Par conséquent, nous avons deux types des modèles : ***le modèle de comportement normal et défaillant dit "complet" et celui de comportement normal.***

La modélisation du comportement complet d'un système doit considérer les deux

points suivants : les défauts à diagnostiquer (permanents ou intermittents) et la dynamique du système : 1) dynamique logique, on ne s'intéresse qu'à l'ordre d'apparition des événements et, 2) prise en compte explicite du temps. Ceci nous conduit à distinguer les cas suivants : 1) construction du diagnostiqueur à partir d'un modèle logique, la construction tenant en compte l'occurrence des défauts intermittents et, 2) construction du diagnostiqueur à partir d'un modèle temporisé.
La classification des méthodes de surveillance basées sur un modèle du comportement normal est différente. Un modèle du comportement normal d'un système représente les relations entre les événements dans le fonctionnement normal. Ces relations sont l'ordre d'occurrence des événements et leurs dates d'occurrence. Dans cette méthode de surveillance, on considérera directement le temps. Généralement, une contrainte temporelle peut être associée à une activité dans le système. Une activité peut être à instance unique ou à instance multiple. La modélisation et la surveillance du système à travers ces activités (tâches) sera présentée. Le cas dans lequel les contraintes temporelles sont associées aux tâches à instance multiple sera aussi présenté.

3.2.1 Surveillance à base de modèle de comportement complet

Les travaux présentés dans ce cadre reposent sur la conception d'un modèle graphique représentant le système à surveiller. Ce modèle décrit à la fois le comportement nominal ou normal et le comportement fautif du système. Dans ce modèle, un défaut est vu comme l'occurrence d'un événement non observable. Ensuite, on construit un autre modèle déterministe à partir du modèle du système. Ce nouveau modèle permet d'estimer l'état courant du système et d'identifier les défauts produits à partir de l'observation des événements. Il est appelé *diagnostiqueur*. Dans ce dernier, un défaut ne peut pas être détecté qu'après un nouvel événement observable du procédé ou au-delà d'une durée.

La modélisation des défauts suppose une connaissance a priori des défauts que l'on souhaite détecter et diagnostiquer. Ces défauts sont répartis dans plusieurs partitions, $\Sigma_\Pi = \{\Pi_{F_1}, \Pi_{F_2}, \Pi_{F_3}, ..., \Pi_{F_r}\}$. Chaque partition regroupe tous les défauts qui ont, soit le même effet sur le procédé, soit la procédure de reprise à faire, après l'occurrence d'un de ces défauts, est la même. Chaque partition de défauts Π_{F_i} correspond une étiquette F_i appartenant à l'ensemble des étiquettes de comportement défaillant $\Lambda_F = \{F_1, F_2, ..., F_r\}$. L'ajout de l'étiquette N à Λ_F, indiquant un fonctionnement normal du procédé, conduit à l'obtention de l'ensemble des étiquettes Λ de tout comportement possible du procédé. Le comportement défaillant peut s'exprimer soit par l'apparition d'un événement de défaut simple et retourne une étiquette F_i, soit par l'apparition d'un défaut multiple correspondant à la propagation de plusieurs défauts et retournant plusieurs étiquettes différentes Λ_F.

Au cours de la présentation du diagnostic à base d'un modèle complet, nous allons considérer les cas suivants : tout d'abord la construction du diagnostiqueur à partir d'un modèle logique (ou non temporisé), ensuite la construction à partir d'un modèle logique qui considère l'occurrence des défauts intermittents et enfin la construction à partir d'un modèle temporisé.

A Diagnostic à base de modèles logiques

Le diagnostic à base de modèle logique consiste à construire un diagnostiqueur à partir d'un modèle du système où seule la précédence des événements est prise en compte. Ainsi, le temps sera considéré uniquement d'une manière qualitative à travers l'ordre d'occurrence des événements. Le diagnostiqueur résultant considérera le temps, de même, d'un point de vue qualitatif.

A.1 Construction du diagnostiqueur Dans la présentation de la méthode de diagnostic à base de modèles logiques, nous présentons les travaux les plus référencés, développés par [Sampath et al., 1995], [Sampath et al., 1996]. Les auteurs proposent dans ces travaux de construire un diagnostiqueur, sous la forme d'un automate à états finis (non temporisé), permettant d'estimer l'état du système, de détecter et d'identifier les défauts produits. Cet automate observe les événements générés par le système commandé puis déclenche les transitions associées à ces événements. Chaque sommet de cet automate comporte une estimation de l'état actuel du système ainsi que de l'ensemble de défauts produits. Le diagnostiqueur a alors une connaissance de l'accessibilité des états. Chaque état x du diagnostiqueur a une fonction de décision $l(x) \in \Delta$, où Δ est l'ensemble de tous les sous-ensembles des étiquettes $\Delta = \{\{N\}, \{F_1\}, \{F_2\}, ..., \{F_r\}, \{N, F_1\}, \{N, F_2\}, ..., \{N, F_r\}, \{N, F_1, F_2\}, ..., \{N, F_1, F_2, ..., F_r\}, \{F_1, F_2\}, \{F_1, F_2, F_3\}, \{F_1, F_2, ..., F_r\}\}$. Tout état x ayant $l(x)$ qui possède seulement des étiquettes de défaut est un état certain de défaut. Un état de diagnostiqueur est qualifié de normal si les étiquettes appartenant à cet état ne comporte que l'étiquette N. Un état du diagnostiqueur est qualifié de F_i-incertain s'il ne coincide à aucun des deux cas précédents. Par conséquent, un diagnostiqueur est un automate réduit aux événements observables avec mémoire des événements de défauts représentés par des étiquettes indiquant le fonctionnement normal ou défaillant.

Afin de mieux expliquer le fonctionnement du diagnostiqueur, nous considérons l'exemple présenté dans la figure 3.1.a. Cet exemple représente l'automate d'un procédé où l'événement f_1 représente un défaut appartenant à la partition de défaut Λ_{F_1}. Dès lors, ce modèle dispose seulement d'informations provenant des événements. En effet, à partir de l'état initial, il est impossible de savoir si le système se trouve dans l'état

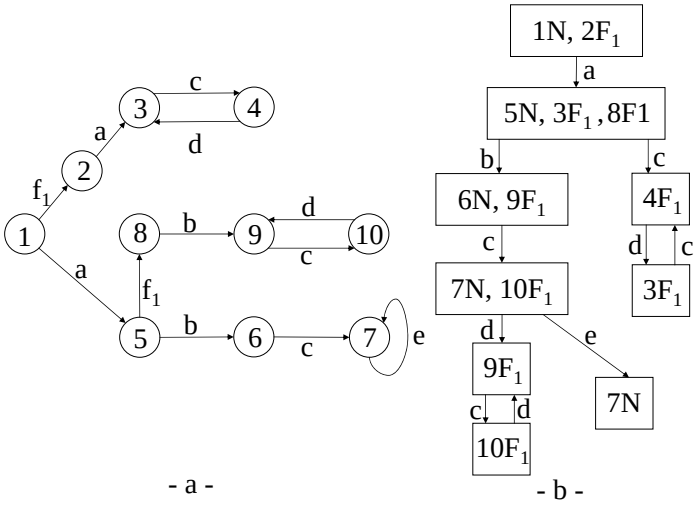

FIG. 3.1 – Construction du diagnostiqueur à base de modèle logique : a- Modèle d'un procédé b- Son diagnostiqueur.

1 en fonctionnement normal ou dans l'état 2 indiquant l'occurrence du défaut f_1. De même, à partir de l'occurrence de l'événement a, il est difficile de savoir si le procédé se trouve dans l'état 5 indiquant un fonctionnement normal du procédé ou dans les états 3 ou 8 indiquant un défaut de type F_1. Le diagnostiqueur de ce modèle est présenté sur la figure 3.1.b. Ainsi, à partir de l'état 1 du modèle du procédé, l'état $1N$ du diagnostiqueur est construit. Suite à l'occurrence de l'événement a, le procédé arrive soit à l'état 3, soit à l'état 5 ou l'état 8. Le diagnostiqueur indique alors que cet événement permet d'accéder soit à l'état 3 ou l'état 8 en détectant un événement de défaut f_1 par le label, ou l'étiquette F_1, soit l'état 5 en fonctionnement normal par l'étiquette N. Cet état du diagnostiqueur ne permet pas d'isoler le défaut et d'assurer du bon fonctionnement du procédé. C'est l'occurrence de l'événement c qui permet d'isoler et de détecter le défaut f_1. Cependant, si l'événement b arrive de l'état incertain $\{3F_1 8F_1 5N\}$ du diagnostiqueur, alors on constate à nouveau un état incertain $\{9F_1 6N\}$. Dès lors, il faut attendre l'occurrence des événements c puis d pour isoler et détecter le défaut dans un état certain 9, du diagnostiqueur pour lequel $l = \{F_1\}$.

A.2 Modélisation du comportement complet du système Nous avons construit le diagnostiqueur dans l'exemple présenté ci-dessus, en supposant disposer du modèle complet du système. Dans la suite nous allons détailler les informations du modèle nécessaires pour la construction du diagnostiqueur.
Le diagnostic des SED nécessite la description complète du comportement du système tant du point de vue de la partie commande que de la partie opérative. Le modèle décrit les évolutions à travers des séquences d'événements qui tiennent compte de l'état du système. Selon le degré de description, le modèle fournit le maximum d'informations

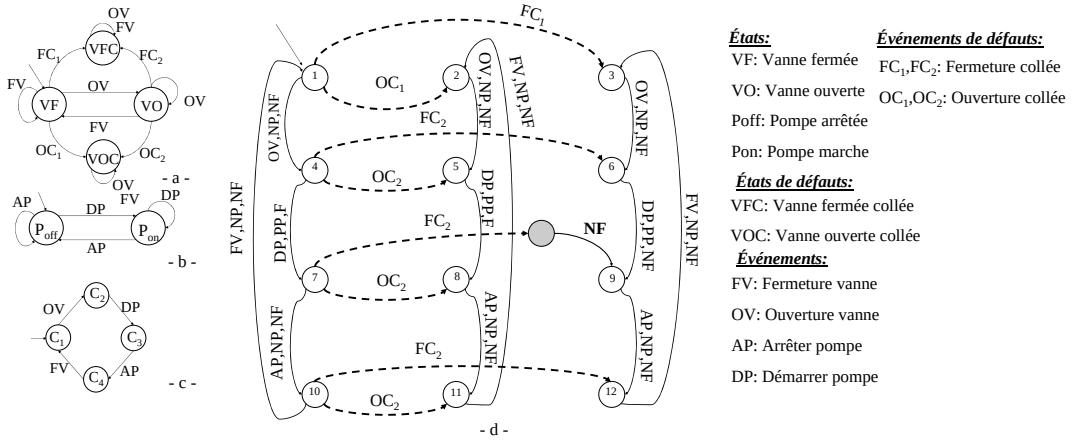

FIG. 3.2 – Modélisation du procédé de distribution d'eau : a- Modèle de la vanne b- Modèle de la pompe c- Modèle de la commande d- Modèle complet du procédé

au diagnostiqueur qui prendra la décision sur la présence d'un défaut et sur son type. Nous allons illustrer maintenant autour d'un exemple la modélisation du comportement complet d'un système.

• ***Spécification de l'exemple :*** L'exemple est constitué d'un procédé de distribution d'eau composé d'une vanne et d'une pompe. Le système a deux capteurs : un capteur de flux et un capteur de pression. Le capteur de flux indique la présence d'un flux dans la canalisation par une mesure F et une non-présence de ce flux par une mesure NF. Le capteur de pression équipé sur la pompe indique s'il y a une pression positive PP dans la pompe ou l'absence de la pression NP.

• ***Modélisation du procédé commandé :*** La modélisation du procédé comporte à la fois une description de la partie opérative de la vanne et de la pompe (les figures 3.2.a et 3.2.b), et un modèle de la commande (Figure 3.2.c). Le modèle de vanne intègre le comportement normal et anormal. Les états VF et VO représentent respectivement la vanne en position fermée et ouverte avec les événements d'ouverture OV et de fermeture de vanne FV. Ce modèle exprime également les événements de défauts par les événements non observables FC_1, FC_2, OC_1 et OC_2 représentant un défaut sur la vanne en fermeture, VFC, et en ouverture, VOC, lorsqu'elle est collée. La commande décrit le comportement désiré à partir de l'état initial C_1 (Figure 3.2.c). A partir de cet état, il est possible d'ouvrir la vanne en activant l'ordre OV (état C_2), ensuite la pompe doit démarrer par l'ordre DP (état C_3). Dans cet état l'événement ou l'ordre AP arrête la pompe et emmène le procédé dans l'état C_4. Il faut alors fermer la vanne en envoyant l'ordre FV (état C_1).

Comme nous l'avons dit, le système a deux capteurs : de flux et de pression. L'ensemble des indications des ces capteurs pour ce système est donné dans le tableau 3.1. Les capteurs de flux et pression fournissent des mesures en fonction de l'état de la vanne et de la pompe par une fonction H(Pompe,Vanne,•) où le • précise que les capteurs sont

$H(P_{off}, VF, \bullet)$ = NP,NF	
$H(P_{off}, VO, \bullet)$ = NP,NF	$H(P_{ON}, VO, \bullet)$ = PP,F
$H(P_{off}, VOC, \bullet)$ = NP,NF	$H(P_{ON}, VOC, \bullet)$ = PP,F
$H(P_{off}, VFC, \bullet)$ = NP,NF	$H(P_{ON}, VFC, \bullet)$ = PP,F

TAB. 3.1 – Liste des mesures des capteurs pour le système considéré.

totalement indépendant de l'état de la commande. Dès lors, on voit que lorsque la vanne est fermée et que la pompe est arrêtée, les captures retournent la mesure de non flux NF et de non pression NP. A partir des modèles de la partie opérative, de la commande et de la liste des mesures des capteurs, il est possible d'obtenir un modèle global de ce procédé par la composition synchrone de tous les modèles (Figure 3.2.d). Dans cette figure, les flèches pointillés représentent les événements non-observables. Dès lors, l'automate résultant décrit, à partir des états initiaux de chaque modèle, le comportement global du système. Ainsi, l'état 1 correspond à la composition de l'état C_1 de la commande, de l'état VF de la vanne et de l'état P_{off} de la pompe. A partir de cet état, il est possible d'évoluer vers l'état 4 et ceci suite à l'événement OV où les capteurs précisent la non présence de flux dans la canalisation et la non présence de pression dans la pompe. Dans l'état 1, si un défaut sur la vanne collée est survenu par l'événement FC_1 (ou par l'événement OC_1) avant l'ordre OV, le système évolue vers l'état 3 (ou vers l'état 2). A partir des états 4, 5 et 6, il faut attendre le démarrage de la pompe par l'ordre DP, amenant aux états 7, 8 et 9. Si le procédé est en fonctionnement normal alors les capteurs indiquent F et PP. En cas de présence de défaut VFC, les mesurent indiquent NF et PP. Ainsi, il est possible de différencier l'ensemble des états normaux des états anormaux dus à l'occurrence d'un défaut FC.

• ***Diagnostiqueur du système considéré :*** La figure 3.3 montre le diagnostiqueur du procédé considéré, présenté dans la figure 3.2.d. Dans cette figure, le défaut F_1 correspond au cas où la vanne est collée en fermeture et F_2 correspond au cas où la vanne est collée en ouverture. Ce diagnostiqueur peut isoler et détecter seulement le défaut F_1. Nous pouvons remarquer que le défaut de type F_1 peut être détecté et isolé à partir des états $(4N, 5F_2, 6F_1)$ et $(7N, 8F_2)$ suite au changement de l'indication de capteur de flux : $F \rightarrow NF$. Le défaut de type F_2 ne peut pas être détecté. Ceci est dû au fait que les états du diagnostiqueur contenant l'étiquette F_2 sont incertains. Ils contiennent à la fois les étiquettes N et F_2. Nous pouvons remarquer aussi que la détection de défaut F_1 (vanne fermée collée) ne peut être faite qu'après l'occurrence de l'événement OV (ouverture vanne).

Remarque 3.2.1. Le modèle du système présenté ci-dessus permet de décrire les évolutions à travers des séquences des événements qui tiennent en compte l'état du système. Cette méthode de modélisation est dite "à base d'événement". Les auteurs dans [Zad et al., 2003], [Zad et al., 2005] proposent d'autres méthodes de modélisation, dite "à base

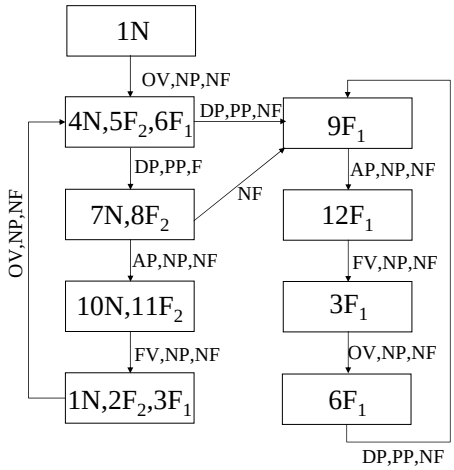

FIG. 3.3 – Diagnostiqueur du procédé présenté dans la figure 3.2.d

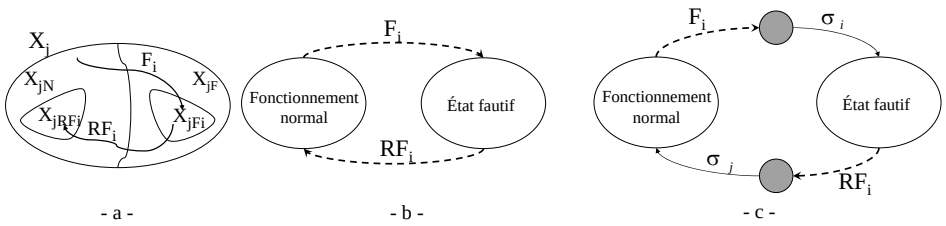

FIG. 3.4 – a,b- Évolution du système c- la solution retenue pour surveiller la dynamique du système.

d'états". Le modèle à base d'états décrit le système par les sorties des capteurs et les commandes ainsi que le chemin qui conduit à ces états. Un défaut est caractérisé par l'atteignabilité d'un état défaillant. Chacun de ces deux modèles conduit à l'obtention d'un diagnostiqueur à base d'événements et à base d'états. Une comparaison rapide entre les deux méthodes de modélisation peut être synthétisée par les points suivants :

• Le diagnostiqueur à base d'événements doit être initialisé en même temps que le procédé afin qu'il puisse suivre l'évolution du procédé. La modélisation à base d'états a l'avantage de ne pas avoir besoin d'initialiser le diagnostiqueur en même temps que le procédé.

• La modélisation à base d'états suppose avoir des capteurs robustes qui ne sont pas considérés comme susceptibles d'être défaillants. Ceci n'est pas le cas de modélisation à base d'événements.

B Diagnostic des défauts intermittents

Plusieurs travaux récents se sont intéressés à étendre l'approche de [Sampath et al., 1995] et [Sampath et al., 1996] présentée ci-dessus dans le cadre des systèmes sujets aux défauts intermittents. Parmi les travaux élaborés dans ce cadre, nous citons [Jiang et al., 2003] et [Correcher et al., 2003]. Le travail présenté dans [Jiang et al., 2003] propose de construire un diagnostiqueur sous la forme d'un automate à états finis. Ce diagnostiqueur est capable de détecter le nombre de fois qu'un défaut répétitif ou intermittent s'est produit, pendant une exécution du système. Les notions de k-diagnosticabilité et [1,k]-diagnosticabilité ont été introduits afin de déterminer pendant une durée bornée, si un défaut s'est produit respectivement k ou j fois où $1 \leq j \leq k$.

Un défaut intermittent peut avoir lieu plusieurs fois pendant le fonctionnement du système. Le nombre de fois qu'un défaut s'est produit et la durée de ce défaut intermittent peuvent amener à transformer un défaut intermittent en un défaut permanent. Dans cet objectif, les travaux présentés dans [Correcher et al., 2003] proposent de construire un diagnostiqueur et une base de données qui sera utilisée pour mémoriser le nombre d'apparition des défauts intermittents, leur durée et leur fréquence d'apparition. Toutes ces informations seront déterminées lorsque le diagnostiqueur détecte un défaut. Nous nous intéressons ici seulement au diagnostiqueur capable de détecter les défauts intermittents d'un système. En effet, un défaut intermittent fait basculer le système vers un état fautif. Lorsque son événement de recouvrement se produit (chaque défaut intermittent a un seul événement de recouvrement), le système revient au fonctionnement normal. Ce comportement de système est représenté dans la figure 3.4.a. Dans cette figure, X_j correspond à l'ensemble des états du système, $X_{jF} \subset X_j$ correspond à l'ensemble des états fautifs et $X_{jN} \subset X_j$ est l'ensemble des états de fonctionnement normal. Les ensembles $X_{jF_i} \subseteq X_{jF}$ et $X_{jRF_i} \subseteq X_{jN}$ correspondent respectivement aux états atteignables suite à l'occurrence d'un défaut de type F_i et aux états atteignables d'un état $x \in X_{jF_i}$ suite à l'occurrence d'un événement de recouvrement. Soit RF_i un événement de recouvrement. Comme le montre la figure 3.4.a, le système revient au fonctionnement normal après la disparition du défaut F_i et ceci est suite à l'occurrence d'un événement de recouvrement RF_i.
Il est à noter que les événements représentant un défaut et son recouvrement sont non-observables. Par conséquent, le modèle peut contenir des cycles (ou contient) des événements non-observable (Fig.3.4.b). La méthode de base développée dans [Sampath et al., 1995] et [Sampath et al., 1996], suppose que le modèle ne contient pas des cycles d'événements non-observables. Pour que le diagnostiqueur soit capable de détecter et diagnostiquer un défaut intermittent, la solution présentée dans la figure 3.4.c est retenue. Dans le modèle du système, le cycle d'événements non-observables constitué d'un défaut intermittent et de son recouvrement, est modifié de telle manière qu'il

contient les événements observables σ_i et σ_j. L'événements σ_i caractérise l'atteignabilité à l'état fautif et l'événement σ_j caractérise le recouvrement de ce défaut et le retour du système au fonctionnement normal. Si les capteurs de système ne permettent pas de vérifier cette solution, des capteurs supplémentaire peuvent être ajoutés. Autrement dit : l'observabilité du système peut être poussée afin qu'un défaut intermittent et son recouvrement soient diagnosticables. Par conséquent, les données caractérisant un défaut intermittent peuvent être déterminées.

• ***Exemple :*** Afin d'expliquer cette modélisation, prenons l'exemple du procédé de distribution d'eau composé d'une vanne et d'une pompe présenté dans le paragraphe A.2. Les modèles de la pompe et de la commande restent inchangés (les figures 3.2.b,c). Dans la figure 3.5.a, nous présentons le modèle de la vanne étant sujette aux défauts intermittents. Nous trouvons que la vanne subit des défauts $F_1 = \{FC_1, FC_2\}$ et $F_2 = \{OC_1, OC_2\}$. Les défauts F_1 et F_2 représentent respectivement le collage de la vanne en fermeture et en ouverture. Les événements de recouvrement sont respectivement RFC et ROC. Les états VFC_1, VFC_2, VOC_1 et VOC_2 représentant la vanne collée en fermeture et en ouvertures, sont les états fautifs. Afin d'expliquer le modèle de vanne, supposons que la vanne est fermée (VF) et elle subit un défaut FC_1 (vanne collée en fermeture), alors la vanne bascule vers l'état VFC_1. Dans cet état, si l'événement de recouvrement RFC_1 se produit la vanne revient à son état normal VF, mais si la vanne reçoit l'ordre d'ouverture OV alors il y a commutation vers l'état fautif VFC_2 et ainsi de suite.
La spécification du procédé mentionnée ci-dessus, indique que le système a deux capteurs : de flux, de pression. Les mesures des capteurs de flux et de pression sont respectivement $\{F, NF\}$ et $\{NP, PP\}$. Dans l'objectif de diagnostiquer les défauts intermittents de la vanne F_1 et F_2, un capteur supplémentaire est équipé sur la vanne. Ce capteur indique que la vanne est ouverte O_1 ou fermée C_1. Il fournit des mesures en fonction de l'état de la vanne. Quand la vanne est fermée et que la pompe est arrêtée, les capteurs retournent NF, NP et C_1. L'ensemble des indications de ces capteurs pour ce système est donné dans le tableau 3.2.
A partir des modèles de la partie opérative, de la commande et des mesures des

$H(P_{off}, VF, \bullet)$ = NP,NF,C_1	
$H(P_{off}, VO, \bullet)$ = NP,NF,O_1	$H(P_{ON}, VO, \bullet)$ = PP,F,O_1
$H(P_{off}, VOC, \bullet)$ = NP,NF,O_1	$H(P_{ON}, VOC, \bullet)$ = PP,F,O_1
$H(P_{off}, VFC, \bullet)$ = NP,NF,C_1	$H(P_{ON}, VFC, \bullet)$ = PP,F,C_1

TAB. 3.2 – Liste des mesures des capteurs pour le système considéré.

capteurs, il est possible d'obtenir un modèle global de ce procédé par composition synchrone de tous les modèles (Fig. 3.5.b). Dès lors, l'automate résultant décrit, à

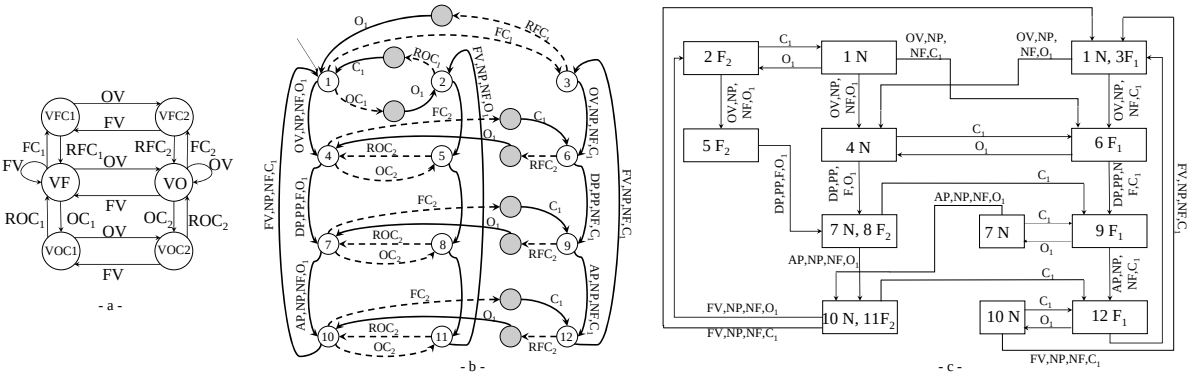

FIG. 3.5 – a- Modèle da la vanne b- Modèle du procédé c- diagnostiqueur

partir des états initiaux de chaque modèle, le comportement global du système. Dans cette figure, les flèches pointillés représentent les événements non-observables. On remarquera que les mesures de capteur ajoutées à la vanne interviennent dans les cycles représentant le défaut intermittent et son recouvrement. Le diagnostiqueur est présenté dans la figure 3.5.c. Il est construit en suivant la méthode de la construction du diagnostiqueur, illustrée dans le paragraphe précédent. Ce diagnostiqueur peut détecter et isoler les défauts de type F_1 et F_2. Nous pouvons remarquer que la détection de défaut F_1 (vanne collée en fermeture) ne peut être faite qu'après l'occurrence de l'événement OV (ouverture vanne). De même, le défaut de type F_2 (vanne collée en ouverture) ne peut être détecté qu'après l'occurrence de l'événement FV (fermeture vanne).

Remarque 3.2.2. La considération des défauts intermittents dans le modèle du procédé, aboutit à avoir des cycles d'événements non-observables. Ces cycles constituent des événements représentant le défaut et son recouvrement. La méthode de diagnostic de base [Sampath et al., 1995], [Sampath et al., 1996], suppose que le modèle du procédé ne contient aucun cycle d'événements non-observables. Ceci a conduit à utiliser des capteurs pour qu'on puisse suivre les dynamiques des composants subissant des défauts intermittents. Autrement dit : les franchissements entre des états normaux et fautifs deviennent observables. Les événements correspondant aux sorties de ces capteurs, interviennent dans les cycles des événements non-observables. Par conséquent, le diagnostiqueur construit à partir de ce modèle est capable de diagnostiquer les défauts intermittents du système. Si les capteurs de système ne permettent pas de synthétiser un diagnostiqueur, des capteurs supplémentaires doivent être ajoutés. Ces capteurs observent les dynamiques des composants sujets aux défauts intermittents.

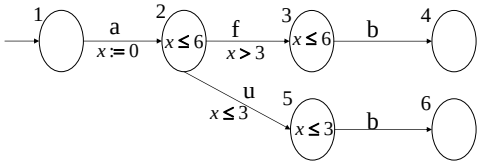

FIG. 3.6 – Une partie de modèle temporel d'un procédé

C Diagnostic des systèmes par les aspects temporels

Afin d'améliorer l'efficacité des approches de [Sampath et al., 1995] et [Zad et al., 2003], plusieurs travaux se sont intéressés à étendre ces approches dans le cadre des modèles temporisés, tel que les automates temporisés, les réseaux de Petri temporels et les automates finis à temps discrétisé [Brandin and Wonham, 1994]. Ces travaux sont motivés par l'apport que peut apporter une exploitation quantitative du temps dans l'identifcation et la détection des défauts surtout dans les systèmes à caractère temporisé (où le système peut avoir plusieurs cycles de fonctionnement comportant les mêmes événements mais qui se déclenchent à des dates différentes). Parmi ces travaux, nous citons les approches à base de modèles automates temporisés, développées dans les travaux [Derbel et al., 2006], [Tripakis, 2002], les approches utilisant les RdP temporels [Ghazel et al., 2005] et les approches utilisant les temps discrets [Zad et al., 2005], [Chen and Provan, 1997]. Nous présentons dans la suite un exemple pour montrer l'apport de l'information temporelle. Celle-ci est exploitée dans les méthodes mentionnées ci-dessus pour le diagnostic des défauts.

Soit le modèle d'un procédé représenté par la figure 3.6. Les événements u,f sont non-observables tandis que les événements a,b sont observables. L'événement f représente un défaut. Les deux séquence $a.f.b$ et $a.u.b$ ont la même projection sur l'ensemble des événements observables a,b. Par conséquent, nous ne pouvons pas distinguer entre les comportement fautif et normal à partir de la séquence d'événements décrivant le comportement observable. Nous allons voir qu'on peut distinguer un défaut f d'un événements non observable u par l'occurrence de b et par le temps de réponse de cet événement. Ainsi, à partir de l'état 2, l'événement b doit conduire dans l'état 4 ou 6 après une certaine durée de temps. S'il survient après plus de 3 unite de temps, $x > 3$, il est alors possible de conclure que c'est le défaut f qui est survenu. Pour cet exemple, si l'information temporelle est absente, le procédé n'est pas diagnosticable.

Nous présentons dans la suite une synthèse sur les principaux travaux développés dans ce cadre. Nous divisons ces travaux dans deux catégories : Approches basées sur un algorithme d'estimation des états accessibles et approches basées sur un observateur sous la forme d'un automate temporisé.

La première approche a été proposée dans [Tripakis, 2002] et permet uniquement la détection des défauts. Elle consiste à concevoir un observateur sous la forme d'un algorithme d'estimation d'état permettant la détection des défauts non observables à

partir de l'observation d'une séquence d'événements générés par le système. Il s'agit d'abord de modéliser le système sous la forme d'un automate temporisé. Ensuite, un test de diagnosticabilité est appliqué sur le modèle. Si le modèle est diagnosticable, on exécute en ligne l'algorithme de l'observateur. L'exécution de cet algorithme détecte les défauts dans le système. Il calcule à la volée l'ensemble des états possibles du système. En partant d'un sous-ensemble R des sommets, un opérateur $Post_{(\delta,a)}(R)$ défini dan l'algorithme, permet de calculer l'ensemble des états possibles du système après l'écoulement de δ unité de temps et l'observation d'un événement a, sachant que le système se trouve à un sommet de l'ensemble R. Pour chaque événement observé, on applique cet opérateur puis on vérifie si tous les états de cet ensemble sont fautifs. Dans ce cas, l'algorithme annonce la détection d'un défaut dans le système.
La deuxième approche [Zad et al., 2005], [Derbel et al., 2006], consiste à construire un observateur sous la forme d'un automate temporisé déterministe. Dans cette approche, l'automate temporisé permet la détection des défauts produits à partir de l'observation d'une séquence temporisée d'événements générés par le système. Suite à l'observation d'un événement ou l'écoulement d'une durée de temps, l'automate de l'observateur évolue à partir du sommet courant, de manière déterministe, vers un autre sommet. Chaque sommet estime les états possibles du système et indique l'existence ou non d'un défaut. Dans [Ghazel et al., 2005], les auteurs ont proposé de construire un observateur d'états à partir d'un modèle RdP temporel du système. L'observateur est en fait un graphe d'états similaire au graphe des classes d'états, où les transitions entre les nœuds se font uniquement par des transitions qui correspondent à des événements observables.

Nous pouvons remarquer, d'après ce paragraphe, l'apport de l'exploitation des propriétés temporelles d'un système temporisé, pour détecter et diagnostiquer les défauts. Nous venons de voir qu'il y a deux principales approches. La première approche est basée sur la construction d'un observateur sous la forme d'un algorithme. Ce dernier nécessite beaucoup de calcul en ligne ce qui le rend inadéquat aux systèmes temps réels. La deuxième approche est basée sur la conception hors ligne d'un observateur sous la forme d'un automate temporisé. Cette approche est plus appropriée pour les systèmes temps réels.

3.2.2 Surveillance à base de modèle de comportement normal

Les travaux portant sur la surveillance à base d'un modèle de comportement normal, s'appuient sur une hypothèse simple et très efficace : "tout ce qui n'est pas normal est forcément anormal". La modélisation du fonctionnement normal d'un système commandé peut sembler *a priori* séduisante car les plages de fonctionnement d'un système sont en général bien connues. Ce modèle met en évidence les différentes relations entre les événements générés par le procédé et la commande (signaux de capteurs et les ordres).

Ces relations sont l'ordre d'occurrence des événements et leurs dates d'occurrence. En effet, l'ordre d'occurrence d'événements peut être modélisé par les automates à états finis et les RdPs. L'exploitation de ce modèle logique ne permet pas de détecter tout les défauts comme par exemple l'absence d'un compte rendu d'un capteur suite à l'émission d'un ordre de commande. Afin de prendre en compte ce problème, il est souvent nécessaire de connaître les relations temporelles parmi les différents événements. Ces relations définissent un ensemble de contraintes entre les dates d'occurrence d'événements. La violation de ces contraintes temporelles caractérise un défaut. Pour modéliser ces relations temporelles entre les événements, ll faut faire appel à des modèles temporels. Parmi les outils utilisés pour modéliser le comportement normal du système, deux sont largement utilisés : les automates temporisés et les RdP temporels.
Dans les méthodes existantes dans la littérature, nous trouvons qu'une place dans un RdP temporel ou un sommet dans un automate temporisé peuvent avoir plusieurs significations. Ils peuvent représenter soit une activité (ou tâche) dans le système, ou un ressource du système. À partir de ce fait, nous présentons, dans la suite de ce paragraphe, la modélisation de comportement normal et le système de surveillance basé sur le modèle résultant. Ensuite, nous présentons le cas dans lequel les activités du système sont à instance-multiple. Nous présentons aussi le système de surveillance de ce cas en utilisant les séquences temporisées (templates). Enfin, nous présentons la notion de comportement dégradé.

A Modélisation de comportement normal du système

Nous considérons le cas où les contraintes temporelles sont associée aux tâches du système. La figure 3.7.a montre un exemple de modélisation d'une activité en utilisant un RdP temporel. L'activité est délimitée par deux événements représentés par deux transitions t_{deb} et t_{fin} du réseau de Petri. Chacune de ces transitions conditionne la validité de l'événement par des tests correspondant à des conditions de franchissement. Ces conditions concernent les places d'entrées (disponibilité de ressources par exemple) de t_{deb}. Lorsque l'activité est terminée, la transition t_{fin} est franchie. Son franchissement entraîne le marquage des places P_m et P_n en général appelées post-conditions. Un fonctionnement anormal est détecté si la durée entre les deux événements de l'activité associés aux transitions t_{deb} et t_{fin}, est violée. Cette durée est définie par deux dates : la date de fin au plus tôt (α) et la date de fin au plus tard (β). Cette durée est mesurée par l'horloge associée à la transition t_{fin}. Par conséquent, deux symptômes peuvent être distingués compte tenu de cette fenêtre temporelle entre ces deux dates :

- occurrence de l'événement associé à la transition t_{fin} avant la date de fin au plus tôt (α). Dans ce cas, la fonction de détection caractérisera un symptôme de défaut.
- absence de réception de l'événement associé à la transition t_{fin}. Ceci est systématiquement détecté par ***le chien de garde*** basé sur la date de fin au plus tard

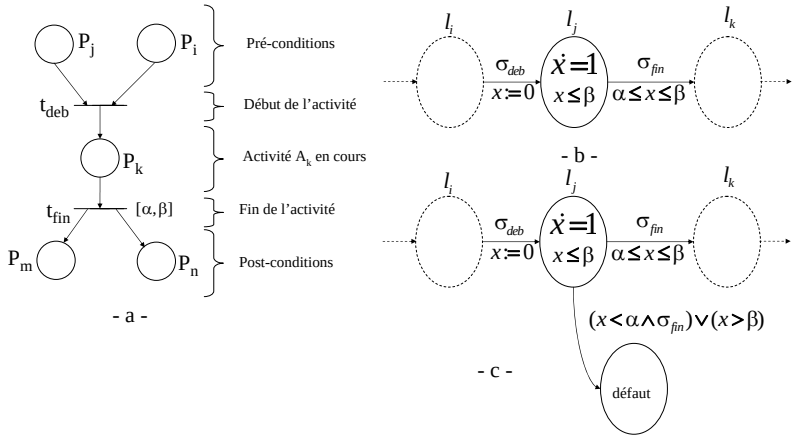

FIG. 3.7 – Le comportement normal d'une activité du procédé modélisée par : a- Un RdP temporel b- Un automate temporisé c- Système de surveillance de l'activité.

(β) de l'opération. Dans ce cas, la fonction détection caractérise un symptôme de défaut.

Cette activité peut être modélisée aussi par un automate temporisé tel que présenté dans la figure 3.7.b. Dans cette figure, les transitions σ_{deb} et σ_{fin} représentent le début et la fin de la tâche. L'horloge x mesure la durée entre l'occurrence de ces deux événements. La valeur de x dans le fonctionnement normal appartient à l'intervalle $\alpha \leq x \leq \beta$.
Les modèles présentés ci-dessus représentent les spécifications de fonctionnement normal. Parmi les travaux nombreux existant dans la littérature, qui utilisent ce type de modèle, nous citons les travaux suivants : [Sahraoui et al., 1987], [Combacau, 1991], [R.Vallett, 1995], [Rayhane, 2004]. La surveillance de ces spécification permet de détecter les défauts par le mécanisme de détection présenté ci-dessus. La figure 3.7.c présente le système de surveillance de la tâche en se basant sur le modèle présenté dans la figure 3.7.b.

Les travaux basés sur ces modèles supposent disposer d'un ensemble de contraintes temporelles. Ces dernières peuvent être mesurées ou données par l'ordonnanceur (cas de modélisation des tâches du système) ou bien par l'apprentissage. Les auteurs dans [Sujit and Holloway, 2000] présentent une méthode d'apprentissage de relations temporelles inter-événements utilisant des observations d'un système manufacturier opérant correctement.

B Surveillance de système par un estimateur d'états

L'idée de l'estimateur est de vérifier si l'état observé est normal et si les événements observés se produisent correctement. Un défaut est détecté si le comportement observé ne correspond pas à celui prévu par l'estimateur. Ce dernier est composé des états discrets possibles correspondant aux situations atteignables du système. Il contient aussi les intervalles temporels délimitant les durées minimales et maximales de séjour dans

chaque situation atteignable du système. Ces conditions seront ensuite surveillées et un défaut sera détecté si elles sont violées.

Le travail présenté dans [Srinivasan and Jafari, 1993], consiste à concevoir un estimateur d'états correspondant au fonctionnement normal. Il s'agit d'abord de modéliser le comportement normal du système commandé sous la forme d'un RdP temporel. L'estimateur est ensuite construit à partir de ce modèle. L'estimateur est composé des marquages possibles de modèle RdP et les séquences d'événements conduisant à chaque marquage. Il contient aussi les intervalles temporels délimitant les durées minimales et maximales de séjour des jetons dans chaque place d'un marquage atteignable. Ces conditions seront ensuite surveillées et un défaut sera détecté si elles sont violées.
Dans le modèle RdP temporel d'un système, les événements sont représentés par des transitions et les différents états des ressources sont représentés par des places. Par exemple, si l'on considère le réseau de Petri de la figure 3.8, ce réseau représente un système composé de deux machines A et B, et d'une ressource partagée R. Les places A_1, A_2 et A_3 représentent les différents états de la machine A. La date prévue d'occurrence d'un événement associé à une transition t_i est représentée par l'intervalle $[\tau_{i_{min}}, \tau_{i_{max}}]$ associé à cette transition, où cette durée représentée par l'intervalle est relative à l'instant de la validation de t_i.
L'approche considère les systèmes ayant des ressources partagées (c'est-à-dire : des ressources en conflit). Dans ce cas, la valeur de séjour d'un jeton dans une place dépend de la séquence de franchissement des transitions. Afin d'illustrer ce point, considérons l'exemple de la figure 3.8. Si l'on veut calculer la durée de séjour d'un jeton dans la place B_1, on trouve que cette durée dépend de la séquence de transitions franchissables après le franchissement de t_{a_1} et avant que la place R soit marquée à nouveau. Il est à noter que la durée maximale de séjour dans une place pour une séquence de franchissement des transitions, n'est pas toujours la somme des bornes supérieures des intervalles associés aux transitions. Considérons la séquence de franchissement t_{a_3}, t_{a_5}. La valeur maximale de séjour du jeton dans la place B_1 pour cette séquence de franchissement est $6 + 5 = 11$ et non pas $8 + 5 = 13$. Par conséquent, l'estimateur d'états doit tenir compte de la séquence de franchissement (séquence d'événements) conduisant à un marquage. [Srinivasan and Jafari, 1993] a calculé la durée de séjour d'un jeton dans une place pour un marquage en utilisant la technique du franchissement en arrière des transitions du RdP [Murata, 1989]. En effet, cette technique de franchissement permet de trouver toutes les séquences possibles conduisant à un marquage du modèle et ceci est pour n'importe quel marquage dans le système. Ensuite, les valeurs maximale et minimale des durées de séjour des jetons dans les places pour un marquage sont calculées. Le calcul est fait pour chaque séquence de franchissement conduisant à ce marquage.

En résumé, l'approche de détection des défauts que nous avons présentée dans ce

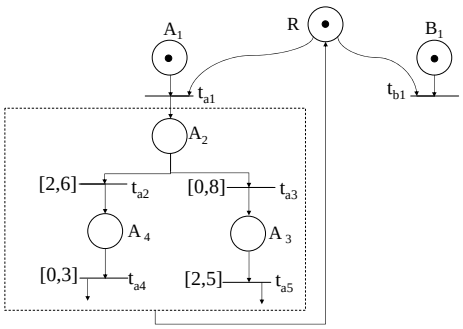

FIG. 3.8 – Un modèle réseau de Petri représentant un système ayant des ressources partagées

paragraphe est basée sur la construction de l'estimateur d'états d'un système dont le comportement normal est connu. L'estimateur contient les situations atteignables des ressources du système, les séquences possibles d'événements conduisant à chaque situation et les intervalles temporels délimitant les durées de séjour des ressources dans chaque situation. Ceci est fait pour chaque séquence possible de franchissement conduisant à cette situation.

Nous présentons maintenant le système de surveillance en utilisant les séquences temporisés (templates) [Pandalai and Holloway, 2000], [Pandalai and Holloway, 1996]. Ces dernières sont aussi construites à partir du modèle sous la forme d'automate temporisé. Cet automate représente le comportement normal du système à surveiller. La séquence temporisé considère le cas où les tâches dans le système sont à instance-multiple.

C Surveillance d'un système par les séquences temporisés

Dans ce paragraphe, nous allons considérer les systèmes dit "*à instance-unique*" et "*à instance-multiple*" [Pandalai and Holloway, 1996] et [Pandalai and Holloway, 2000]. Les systèmes à *instance-unique* possèdent une séquence d'événements se répétant toujours de la même façon avec certaines restrictions temporelles impliquant un délai entre l'occurrence de deux événements. Les systèmes à *instance-multiple* sont des systèmes devant réaliser plusieurs activités (tâches) générant des séquences d'événements fixes pour chaque activités modulables sur l'ensemble des observations.

• ***Exemple :*** Afin de mieux expliquer les notions des systèmes "*à instance-unique*" et "*à instance-multiple*", considérons l'exemple présenté dans la figure 3.9.a. Il représente un système manufacturier, composé d'un vérin devant transférer des palettes d'un convoyeur A vers un convoyeur B. Considérons le vérin pour illustrer l'aspect *instance-unique* du système. Le vérin est commandé en sortie par l'activation d'un événement S qui engendre l'activation de son capteur de fin de course de sortie par l'événement b. La rentrée du

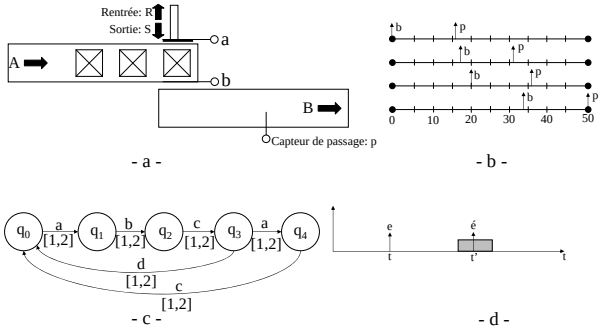

FIG. 3.9 – a- Exemple de système manufacturier b- Illustration d'un système *à instance-multiple* c- Exemple de l'automate temporisé d- La satisfaction de la séquence temporisée.

vérin est, quant à elle, obtenue par l'activation de l'événement R qui entraîne l'activation de son capteur de fin de course de rentrée par l'événement a. Le fonctionnement normal du système implique une séquence d'événements $SbRa$ répétée sur le vérin. L'observation d'une séquence d'événement autre que celle attendue, représente l'apparition d'un défaut. Cette séquence d'événements doit être satisfaite dans un temps donné par l'ajout de contraintes temporelles sur les événements. Par exemple, il existe un délai qui doit être respecté entre l'activation de l'ordre de sortie S et l'activation de son capteur de fin de course de sortie b.

Pour illustrer maintenant l'aspect *instance-multiple* d'un système, considérons la partie du convoyeur B de l'exemple de la figure de 3.9.a. On supposera que lorsqu'une palette sur le convoyeur A est poussée sur le convoyeur B par le vérin (sortie de tige détectée par b), elle met entre 15 et 17 secondes pour arriver jusqu'au capteur de passage p. Chaque palette va alors générer une séquence d'événements b p de 15 à 17 secondes entre les événements b et p. Cependant s'il y a au même moment quatre palettes sur le convoyeur, il est donc possible d'avoir la séquence suivante d'événements observées *bpbbpbpp*. Les délais entre les événements sont respectés comme montrer la figure 3.9.b. Ce type de systèmes est appelé à *instance-multiple*.

Nous pouvons remarquer que les contraintes de temps doivent être appliquées sur les séquences d'événements locales d'une palette et non sur l'observation des séquences globales du système.

Le comportement d'un système à *instance-multiple* ne peut donc pas être modélisé comme un système à *instance-unique*. L'intégration du temps dans un procédé à *instance-unique* est réalisable par des modèles temporels. Il peut être similaire à la modélisation d'une activité présentée dans le paragraphe précédent. Par contre, l'application de contraintes temporelles sur un système à *instance-multiple* est très délicate puisqu'il faut réaliser non seulement un modèle de la partie opérative ou de ses tâches, mais également l'enterlance des tâches.

C.1 Modèle pour la surveillance de système *à instance-unique* Le modèle proposé dans [Pandalai and Holloway, 1996] et [Pandalai and Holloway, 2000] est basé sur des séquences temporisés (appelées "templates"). Ce modèle permet de surveiller les deux types de système. Les modèles de surveillance des systèmes à *instance-unique* et à *instance-multiple* sont générés à partir d'un modèle automate temporisé (Fig. 3.9.c). Cet automate est utilisé afin de représenter les relations temporelles entre les événements et le séquencement entre ces événements pour les deux cas (à *instance-unique* et à *instance-multiple*). Des séquences temporisés, basées sur ces automates, sont construites afin de définir pour chaque événement observé les conséquences normales futures et les conséquences anormales correspondant à un défaut. Une séquence temporisé est de la forme (t, e, C, w) où t est l'instant d'occurrence de l'événement $e \in \Sigma$, C est l'ensemble des conséquences, et w est une étiquette. Une conséquence est une paire (e', τ) où $e' \in \Sigma$ et τ est l'intervalle de délai à borne positive ou négative. Par exemple, la figure 3.9.d montre une séquence temporisée avec une conséquence unique et un intervalle de temps dans lequel l'événement conséquence attendu. La partie grisée indique la période τ durant laquelle l'événement e attend l'occurrence de e'.
Considérons la séquence temporisée $(5, e_1, \{(e_2, [1, 2]), (e_3, [2, 4])\}, w)$. La séquence temporisée est satisfaite pour occurrence de e_2 à la date t, $6 \leq t \leq 7$, ou par l'occurrence de e_3 à la date $7 \leq t \leq 9$. Pour n'importe quelle date $t_f \leq 9$, si la séquence n'est pas satisfaite sous $\rho(]0, t_f])$, alors elle est ouverte. Pour n'importe quelle date $t_f > 9$, si la séquence n'est pas satisfaite sous $\rho(]0, t_f])$, alors elle violée.
Dès lors, chaque événement va être associé à des *séquences temporisées* précisant l'état d'où il vient, les états dans lesquels il peut se rendre et les contraintes de temps à sa transition de sortie. Un système va comporter une série de *séquences temporisées* positives à satisfaire et une autre série de *séquences temporisées* négatives à ne pas satisfaire. Les *séquences temporisées* positives sont les conséquences postérieures à l'état. L'ensemble des conséquences postérieures $C_p(q)$ est l'ensemble des événements de sortie d'un état q et des durées pendant lesquelles ils peuvent être générés par le procédé. Les *séquence temporisées* négatives sont les conséquences interdites. Elles permettent de détecter si un événement a eu lieu plus tôt qu'il ne fallait. L'ensemble des conséquences interdites est l'ensembles des événements d'entrée de l'état q associé aux durées séparant ces événements aux événements de sortie et pendant lesquelles ils ne doivent pas être générés.
Afin d'expliquer ces templates, prenons l'automate présenté dans la figure 3.9.c. Deux des séquences temporisés pouvant être générées de cet automate, sont les suivantes. La *séquence temporisé* $(a, [\{q_0\}, \{b, [1, 2]\}, q_1], [\{q_3\}, \{c, [1, 2]\}, q_4])$ exprime les conséquences postérieures où l'événement a peut provenir soit de l'état q_0 et arrivant à l'état q_1 dans lequel l'événement b doit se produire dans un intervalle de temps $[1, 2]$, soit de l'état q_3 conduisant à l'état q_4 dans lequel l'événement c doit arriver dans l'intervalle $[1, 2]$. La séquence négative $(b, [\{q_1\}, \{a,]-1, 0[\}, pb])$ manifeste le fait que l'événement b ne doit pas survenir si l'événement a apparaît dans un retard minimal compris dans $]-1, 0[$ pour arriver à l'état q_1. Si la séquence temporisé est satisfaite, alors un défaut est détecté.

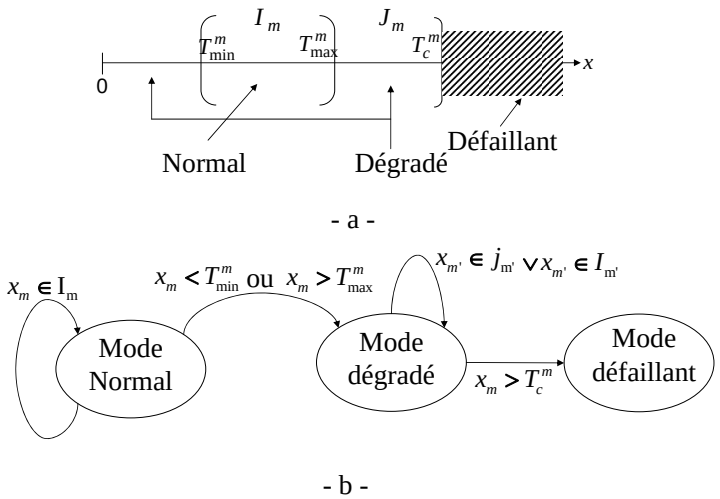

FIG. 3.10 – a- La durée d'exécution d'une tâche b- les modes de fonctionnement

C.2 Modèle pour la surveillance de système *à instance-multiple* L'automate temporisé seul ne peut pas gérer les cas des systèmes à *instance-multiple* puisqu'il est impossible d'être dans plusieurs états dans un même automate [Pandalai and Holloway, 2000]. En effet, il est possible d'avoir plusieurs instances générant le même événement, et il est peut-être impossible de déterminer à quelle instance l'événement appartient. Par conséquent, le modèle de surveillance doit considérer toutes les instances pouvant générer l'événement considéré. En plus des séquences postérieures $C_p(q)$ définies dans le paragraphe précédent, le modèle de surveillance d'un procédé à instance-multiple inclut les conséquences inverses $C_i(q)$. Ces dernières sont l'ensemble des événements d'entrée à l'état q et des durées leur correspondantes.
Les modèles de surveillance de système *à instance unique* et *à instance multiple* peuvent être fusionnés permettant ainsi la création d'un modèle global de surveillance d'instances uniques et multiples s'exécutant en parallèle.

L'avantage des *séquences temporisées* réside dans le fait qu'elles peuvent être associées à une modélisation du système à instance-multiple. Par contre, disposer d'un modèle hors-ligne composé de *séquences temporisées* peut être difficile dans le cas où il y a un grand nombre de conséquences relatives à l'occurrence des événements. Une solution proposée dans [Boufaied, 2003], consiste à utiliser des modèles de réseaux de Petri qui permettent d'exprimer de telles conséquences et de vérifier le bon fonctionnement du système quand il est à *instance-unique ou multiple*. La proposition consiste à utiliser la notion de multi-sensibilisation d'un RdP temporel (Remarque 2.3.2).

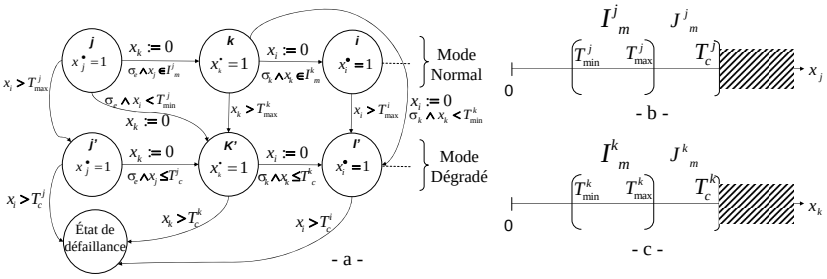

FIG. 3.11 – a- Un exemple d'un modèle de surveillance b,c- les domaines temporels des tâches surveillées dans la figure a

3.2.3 Notion de fonctionnement dégradé d'un procédé

Généralement, les évolutions d'un système dans le temps peuvent être regroupées dans deux modes de fonctionnement : Normal et anormal. Certains travaux ont introduit des nouveaux comportements : le comportement acceptable [Valette et al., 1989] et le fonctionnement dégradé [Rayhane, 2004].
Dans les travaux présentés dans [Rayhane, 2004], l'auteur a distingué trois modes de fonctionnements d'un système de production : *normal, dégradé et défaillant*. Chaque système est composé de plusieurs tâches (activités) qui interagissent. Chaque tâche m correspond à l'exécution d'une opération bien définie. Lorsque la tâche est exécutée normalement, la durée d'exécution peut varier dans un intervalle compris entre T^m_{min} et T^m_{max} (fig. 3.10.a). Un intervalle de tolérance de la tâche m est celui qui correspond au fonctionnement en mode dégradé, noté : $J_m =]T^m_{max}, T^m_c]$. Cet intervalle tient compte des retard qui peuvent être dus, soit au vieillissement des machines par exemple, soit aux surcharges, soit aux problèmes mécaniques, etc.
Le système peut être soit en *fonctionnement normal* si les durées d'exécution des activités (tâches) sont respectées (à l'intérieur d'un intervalle de bon fonctionnement noté I_m dans la figure 3.10.a), soit en *fonctionnement dégradé* si le retard d'arrivée d'événements fin de tâches est compris dans un intervalle de tolérance (noté J_m), soit en *état de défaillance* si ces conditions ne sont pas respectées comme le montre la figure 3.10.a. Le système peut basculer du mode de fonctionnement normal vers le mode dégradé 3.10.b. Dans cette figure, la tâche m bascule vers le mode de fonctionnement dégradé lorsque sa durée mesurée par x_m appartient à l'intervalle J_m. Si la tâche suivante m' a été exécutée tel que la durée de tâche m' appartient soit à l'intervalle $J_{m'}$ ou à l'intervalle $I_{m'}$, le mode de fonctionnement reste dégradé. Du point de vu de modélisation, ces deux modes de fonctionnement ne se différencient que par la tolérance associée à la durée des tâches que l'on accorde. L'état de défaillance quant à lui est un état qui nécessite l'intervention de l'équipe de maintenance.

Pour construire le modèle de surveillance, on procède de la manière suivante : tout d'abord, à partir du grafcet de commande, on construit le graphe des situations qui

représente toutes les évolutions possibles du système. Chaque état de ce graphe est associé à une ou plusieurs horloges ; chacune surveille le temps d'exécution d'une activité. Une horloge est initialisée au début de l'activité qui la surveille. Le modèle obtenu est un automate temporisé. Dans cet automate, les gardes des transitions expriment l'évolution normale du système. Autrement dit : les délais qui contraignent les gardes sont les durées d'exécution normal des tâches. Ensuite et à partir de ce modèle, on tient compte du fonctionnement dégradé. Le modèle du mode dégradé est similaire à celui du mode normal (même ensemble états-arcs), seuls les délais qui contraignent les gardes sont prolongés jusqu'à la borne supérieure de l'intervalle J_m. Le mode de défaillance est considéré comme un état global de défaillance. L'interaction parmi les modes est considérée de la manière suivante. A partir de chaque état dans le mode de fonctionnement normal (par exemple l'état j dans la figure 3.11.a), le système peut évoluer vers l'état suivant en mode normal (état k) ou vers l'état analogue en mode dégradé (état j'). A partir de chaque état j' en mode dégradé, le système peut basculer, soit vers l'état suivant en mode dégradé (état k') ou passer en mode défaillant lorsque le temps dépasse la valeur critique T_c^i (fig.3.11.b).

Remarque 3.2.3. D'après l'étude bibliographique sur la surveillance à base d'un modèle de comportement normal, qui a été faite et présentée ci-dessus, nous pouvons faire les remarques suivantes.
Soit un modèle temporel (un automate temporisé ou un RdP temporel ou des séquences temporisées,..) qui définissent les contraintes temporelles entre les dates d'occurrence d'un ensemble d'événements générés par un procédé. Soit $X = \{x_i, x_j, x_k, ..\}$ l'ensemble des horloges utilisées par le modèle afin de surveiller les contraintes temporelles représentées dans le modèle. Les bornes supérieurs des contraintes surveillées par x_i, x_j et x_k sont respectivement les suivantes : β_i, β_j et β_k. En supposant que ces horloges sont initialisées en même temps, l'espace temporel atteignable des horloges global au niveau de tout le modèle et non pas dans une situation particulière (sommet ou un marquage du RdP) est un hypercube. Les bornes de cet hypercube sont les bornes supérieurs des contraintes surveillées (Fig. 3.12). Un défaut de retard ou d'absence d'un événement est détecté lorsque la valeur d'une horloge dépasse les bornes de cet hypercube.
La méthode présentée dans [Rayhane, 2004] adaptant la notion de fonctionnement dégradé pousse les bornes de l'hypercube de la valeur β_i à la valeur β_i' où $\beta_i < \beta_i'$. Un défaut de retard ou d'absence d'un événement est aussi détecté par la violation de l'espace temporel délimité par les nouvelles bornes tolérées.

3.3 Approche d'implémentation de la surveillance

Dans ce paragraphe, nous présentons les approches de construction d'un module de surveillance, les approches d'implémentation et les différentes structures de la prise de décision.

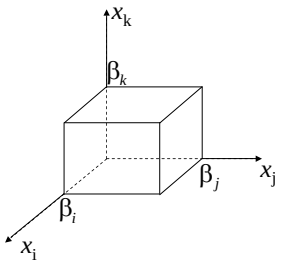

FIG. 3.12 – L'espace temporel atteignable délimitant le fonctionnement normal d'un système caractérisé par un ensemble des contraintes temporelles.

3.3.1 Approche de construction

Il existe trois approches de construction d'un système de surveillance. Elles diffèrent par la place qu'elles donnent au module de surveillance par rapport au système de commande : une surveillance intégrée à la commande, une surveillance séparée de la commande et une approche mixte, c'est-à-dire : une combinaison de deux approches précédentes [Combacau, 2001].

Dans la première approche, le système de surveillance est intégré à la commande. Elle considère que les fonctionnements anormaux doivent être connus à l'avance et introduits dans le système de commande. Cela suppose une connaissance absolue de toutes les évolutions possibles du système. Le modèle de commande contient toute la connaissance sur le fonctionnement normal ou non du système. De plus pour le diagnostic, le système doit être capable d'associer à n'importe quel défaut les causes probables. Dans la seconde approche, les systèmes de commande et de surveillance sont séparés. Toutes les fonctions de surveillance seront séparées de la commande. Cette structure a l'avantage de soulager la commande. Mais elle présente un inconvénient de générer des conflits entre la surveillance et la commande. Ces conflits proviennent de la séparation entre les situations normales et anormales. En effet, ce qui est normal pour la surveillance ne l'est peut-être pas pour la commande. L'approche mixte est un compromis entre les deux précédentes, les fonctions de diagnostic et de décision sont séparées alors que les fonctions de détections et de reprise sont intégrées à la commande. Dans ce cas, le système de commande définit le comportement dit normal. L'avantage de cette approche réside dans le fait que la limite entre normal et anormal est établie dès que l'on spécifie le modèle de commande.

Comme nous l'avons dit dans le chapitre 2, la fonction du système de surveillance est

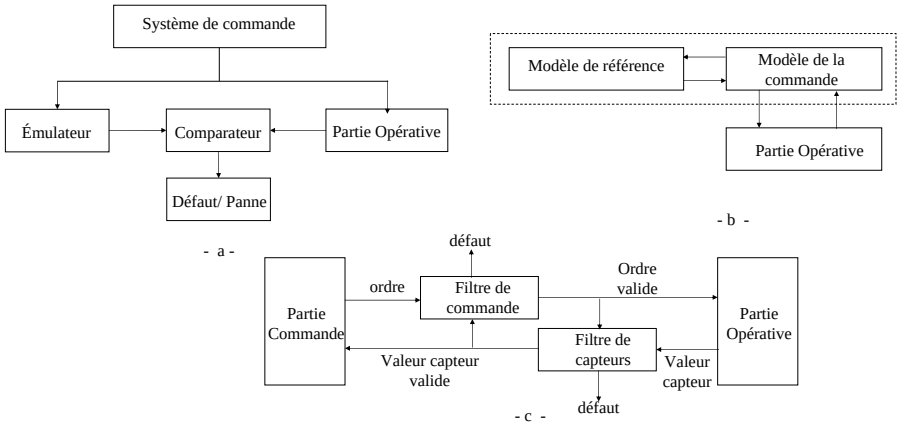

FIG. 3.13 – Approche de surveillance a- Approche comparateur b- Approche par modèle de référence c- Approche Filtre.

limitée à détecter les défauts. Ce système de surveillance qui sera développé au chapitre 4, est séparé de la commande. Cette dernière est construite de manière indépendante du raisonnement de l'approche de surveillance tout en intervenant à travers ses séquences (ordres) pour faire évoluer le processus de surveillance.

3.3.2 Approche d'implémentation

Nous présentons, dans ce qui suit, les différentes approches employées dans la littérature pour l'implémentation d'un système de surveillance. Les approches les plus utilisées sont : l'approche comparateur, l'approche référence et l'approche filtre.

• ***L'approche comparateur*** : Pour ce qui concerne l'approche comparateur 3.13.a, il s'agit de comparer en permanence l'état réel du système, donné par l'ensemble des capteurs, avec celui donné par le modèle de comportement du système. La détection d'un défaut a lieu chaque fois que l'état réel du système s'écarte par rapport à celui donné par le modèle. Les travaux de [Holloway and Krogh, 1991] et [Toguyeni, 1992] placent le modèle du procédé en tant qu'émulateur des évolutions normales de la partie opérative. Son rôle est de calculer les fenêtres temporelles d'occurrence des comptes rendus émis par le procédé quand celui-ci reçoit une commande particulière. Pour une consigne donnée, le comparateur permet de détecter l'apparition non attendu d'un compte-rendu ou lorsqu'il y a non apparition d'un compte rendu pendant la fenêtre temporelle correspondante. Dans cette approche, l'ordre est envoyé en même temps à la partie opérative et à l'émulateur. Donc, un ordre erroné ou qui ne correspond pas à l'état actuel de la partie opérative ne sera pas validé avant son application sur la partie opérative.

• ***L'approche de modèle de référence*** :Dans [Combacau, 1991], un modèle de

référence contenant le modèle de comportements normaux du procédé est présenté. Le principe est de consulter le modèle de référence avant l'envoie de requête dans une fenêtre temporelle (Fig. 3.13.b). S'il y a incohérence entre la requête envoyée et l'état du modèle de référence, alors une erreur de la partie commande est détectée. Les deux modèles doivent évoluer simultanément, sinon seule la référence évolue et un défaut de la partie opérative est détecté. L'avantage de cette approche est qu'un ordre ne sera pas envoyé à la partie opérative avant de le valider. Cependant, les comptes rendus émis par la partie opérative ne sont pas validés avant de les envoyer vers la partie commande.

• ***L'approche filtre*** : Cette approche consiste à insérer un ou plusieurs filtres entre la commande et le procédé, comme le montre la figure 3.13.c. Le principe revient à n'exécuter de commande que lorsque l'état réel du système est cohérent avec cette dernière [Nourelfath, 1997]. L'état réel du système est obtenu à partir de l'ensemble des capteurs à chaque instant. Le filtre de commande teste la cohérence de l'ordre par rapport à la situation de la partie opérative avant de le valider. Le filtre de valeurs capteurs reçoit la valeur des observations avant de les valider et avant de les envoyer à la partie commande. Cette approche demande une connaissance profonde de la partie opérative mais aussi de la partie commande autour de modèles riches.

C'est l'approche de comparateur que nous adapterons pour l'implémentation du module de surveillance (détection de défauts) proposé au chapitre 4. Ce choix est fait car nous supposons que le système de commande est exempt de défauts.

3.3.3 Structure pour la prise de décision

Une structure pour la prise de décision de surveillance doit être définie. Le choix d'une structure dépend de la distribution de l'information disponible : centralisée ou distribuée, et de la taille du procédé : simple ou complexe. Il existe donc trois grandes structures de prise de décision : centralisée, décentralisée et distribuée. Nous donnons dans ce paragraphe une brève description de ces différentes architectures :

• ***La structure centralisée*** : Elle consiste à associer un modèle global du procédé avec un seul module de surveillance (Fig 3.14.a). Ce dernier collecte les différentes informations du procédé avant de prendre sa décision finale sur l'état de fonctionnement du procédé [Sampath et al., 1994]. Bien que performante, la structure centralisée exige la constitution d'un modèle global de procédé qui engendre très souvent des problèmes d'explosion combinatoire.

• ***La structure décentralisée*** : Elle se base sur un modèle global du procédé à qui sont associés plusieurs diagnostiqueurs locaux indépendants [Philippot, 2006] (Figure 3.14.b). Chaque diagnostiqueur reçoit les observations qui lui sont spécifiques et prend une décision locale en se basant sur ses observations locales. Cependant, cette structure implique des problèmes d'indécisions. En effet, certaines spécifications globales ne

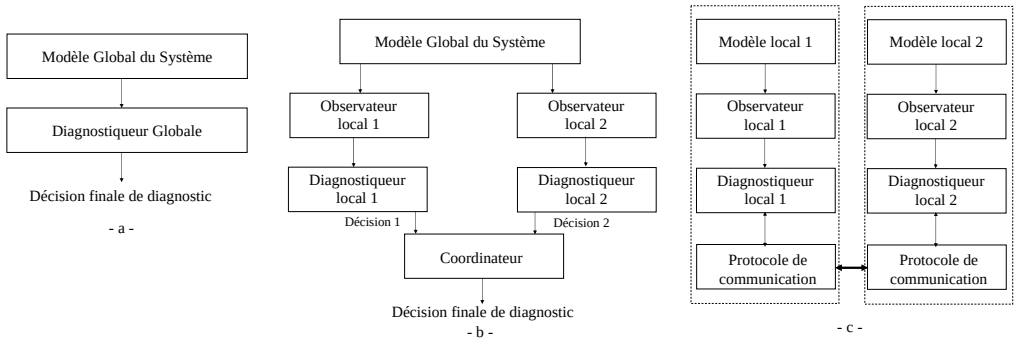

FIG. 3.14 – a- Structure Centralisée b- Structure Décentralisée c- Structure Distribuée, cas de 2 diagnostiqueur locaux

peuvent pas être représentées par un diagnostiqueur local. Afin de résoudre le cas d'indécision, chaque diagnostiqueur envoie sa décision locale à un coordinateur (ou superviseur) qui va gérer les différents problèmes d'ambiguïté entre les diagnostiqueurs et va prendre la décision finale.

• ***La structure distribuée*** : Dans cette structure , le procédé est modélisé à travers ses composants par plusieurs modèles locaux [Boufaied, 2003]. Chacun étant associé à un diagnostiqueur local responsable de son composant (Figure 3.14.c). Chaque diagnostiqueur prend sa décision en se basant sur sa propre observation locale et celle communiquée par les autres diagnostiqueurs locaux. Cette structure permet donc de s'affranchir de la construction d'un coordinateur mais implique une définition d'un protocole de communication entre les diagonstiqueurs parfois difficilement réalisable et engendrant des délais de communications importantes.

En ce qui concerne la structure de la prise de décision, la structure adaptée dans notre système de surveillance est la structure centralisée. Cette structure permet d'éviter l'utilisation d'un coordinateur ou la gestion des conflits décisionnels parmi les différents diagnostiqueurs comme dans le cas des structures décentralisées. Ainsi, il n'a y aura pas besoin d'un protocole de communication complexe entre les diagnostiqueurs locaux, comme dans le cas des structures distribuées. L'inconvénient principal de la structure centralisée est le fait que le modèle du système est global. Cela implique un risque d'explosion combinatoire.

3.4 Conclusion

Ce chapitre a présenté un état de l'art des méthodes de détection et diagnostic des défauts pour les systèmes à événements discrets. Ces méthodes ont été classifiées selon plusieurs critères : le comportement modélisé du système, la structure de prise de décision (centralisée, décentralisée et distribuée) et l'approche de l'implémentation (l'approche de comparateur, modèle de référence et l'approche de filtre).
Comme nous avons pu le noter au cours ce chapitre, il y a deux catégories de modèles : le modèle représentant le comportement complet d'un système et le modèle de comportement normal. Nous résumons ci-dessous les remarques concernant ces modèles.

1. Les objectifs des méthodes basées sur ces modèles sont différents. Les méthodes basées sur un modèle complet a pour objectif de détecter et de diagnostiquer les défauts, tandis que celles basées sur un modèle de comportement normal sont conçues pour détecter seulement les défauts.
2. Le modèle complet nécessite une connaissance dite "profonde" du système. Ceci exige de connaître tous les défauts possibles du système surveillé et le comportement du système suite à chaque défaut, tandis que la modélisation du comportement normal demande seulement une connaissance des plages de fonctionnement normal du système. Ces plages de fonctionnement sont en général bien connues.
3. Un diagnostiqueur est construit à partir d'un modèle représentant le comportement complet du système. Ce modèle contient des événements non-observables. Le diagnostiqueur est un automate à états finis, réduit aux événements observables avec mémoire des événements de défauts représentés par des étiquettes. Ces dernières indiquent le fonctionnement normal ou défaillant. Un état de diagnostiqueur est qualifié de normal si son étiquette ne comporte que l'étiquette correspondante au fonctionnement normal. Un défaut est détecté si l'état de diagnostiqueur ne possède que des étiquettes de défauts.
4. La considération des défauts intermittents engendre des cycles des événements non observables dans le modèle de son comportement. Le diagnostiqueur construit à partir de ce modèle ne permet pas de détecter les défauts intermittents. Un capteur a été utilisé afin d'observer la dynamique de composant sensible aux défauts intermittents. Les mesures (sorties) de ce capteur interviennent dans le cycle d'événements non-observables représentant le défaut et son recouvrement. Par conséquent, le diagnostiqueur conçu à partir du modèle résultant devient capable de détecter les défauts intermittents.
5. L'exploitation des propriétés temporelles d'un système à aspect temporisé, augmente l'efficacité du diagnostiqueur logique. L'étude montre que la conception hors ligne d'un observateur sous la forme d'un automate temporisé est plus approprié aux systèmes temps réels que celui calculé en ligne (observateur sous la forme d'un algorithme d'estimation d'état).

6. Les travaux portant sur la surveillance à base d'un modèle de comportement normal, surveillent l'ordre d'occurrence des événements et leurs dates d'occurrence.

7. Un système peut être soit *à instance unique* ou *à instance multiple*. Les auteurs des [Pandalai and Holloway, 2000], [Pandalai and Holloway, 1996] ont proposé les séquences temporisées pour modéliser le comportement normal de ces systèmes. Dans les travaux présentés dans [Boufaied, 2003], l'auteur propose d'utiliser des modèles de réseaux de Petri qui permettent d'exprimer de telles conséquences et de vérifier le bon fonctionnement du système quand il est à *instance-unique ou multiple*. La proposition consiste à utiliser la notion de multi-sensibilisation d'un RdP temporel pour modéliser le procédé *instance-multiple*.

8. L'estimateur d'états correspondant au fonctionnement normal, détermine si les événements observés se produisent correctement et si l'état observé est normal. Cet estimateur est composé des situations atteignables du système, l'ensemble des séquences d'événements possibles conduisant à chaque situation et les intervalles temporels délimitant les durées de séjour des resources dans chaque situation.

9. Le comportement dégradé a été introduit dans les travaux présentés par [Rayhane, 2004]. L'objectif a été de ne pas pénaliser le système à chaque fois que la situation ne nécessite pas l'arrêt. Dans ce travail, le concepteur a prévu un état intermédiaire entre l'état de fonctionnement normal et l'état de défaut, en prévoyant un intervalle dit "de tolérance" situé chronologiquement immédiatement après l'intervalle de bon fonctionnement. Le système de surveillance détecte un défaut à l'instant d'expiration de cet intervalle toléré.

Cet état de l'art permet de situer l'approche que nous allons proposer dans le chapitre 4, se situant parmi ces méthodes proposées dans la littérature. Il permet aussi de justifier nos différents choix.
Notre contribution est une approche de surveillance, destinée à détecter les défauts interruptibles intermittents et permanents (mentionnés dans le chapitre 2). Elle adapte la notion d'existence d'un état intermédiaire entre l'état de fonctionnement normal et fautif. Notre système de surveillance accepte donc le fait que le système oscille entre le fonctionnement normal et un état intermédiaire. Nous appelons ce fonctionnement, le ***fonctionnement ou comportement acceptable*** du système. Par ailleurs, le modèle dans notre méthode de surveillance ne modélise pas les états fautifs du système.
Comme les méthodes considérant les défauts intermittents, notre approche utilise aussi un capteur afin d'observer la dynamique de l'élément susceptible aux défauts intermittents. Il est également nécessaire de modéliser le système surveillé. Pour cela, nous avons choisi une modélisation par l'automate à chronomètres. Pendant le suivi de la dynamique d'un élément susceptible à un défaut interruptible, la dynamique est soit 1 ou 0.
Le système de surveillance sera alors conçu sous la forme d'un automate à chronomètres. Dans cet automate, les sommets représentent les différentes situations atteignables dans

le système et l'espace temporel dans chaque sommet est calculé de telle manière que les défauts interruptibles seront détectés au plus tôt. Il ne sera en général pas nécessaire d'attendre l'expiration de certaines bornes, comme les méthodes exposées dans ce chapitre (Remarque 3.2.3).

Dans le cadre de notre travail, nous avons proposé une extension de réseaux de Petri temporels, appelée "***Réseaux de Petri à chronomètres Post- et Pré-initialisés***". Ce modèle sera exposé dans chapitre 5, il permet de modéliser en général les systèmes interruptibles. Nous pouvons utiliser ce modèle avec les sémantiques de multi-sensibilisation afin de prendre en compte le cas de système *à instance-multiple*.

Dans le cadre de nos travaux, nous avons souhaité réaliser la construction d'une approche de surveillance (détection de défauts) mais également son implémentation. Le système de surveillance reçoit des entrées du système de commande et de la partie opérative. Une description détaillée de l'approche d'implémentation est fournie dans le chapitre 4.

L'approche de surveillance des SED que nous proposons au chapitre 4 est donc une approche de détection des défauts centralisée, à base d'automate à chronomètres. Cette approche utilise les possibilités offertes dans l'automate afin de calculer l'espace temporel représentant le fonctionnement acceptable dans chaque situation du système. Cet espace sera capable de détecter les défauts interruptibles au plus tôt.

Chapitre 4

SURVEILLANCE DES DÉFAUTS INTERRUPTIBLES DANS LES SYSTÈMES À ÉVÉNEMENTS DISCRETS COMMANDÉS

4.1 Introduction

Pour répondre à la problématique présentée aux chapitres 2 et 3, nous avons développé une méthode de surveillance pour les systèmes à événements discrets commandés. Comme on a pu remarquer dans le chapitre 3, les travaux de recherche réalisés dans le domaine de la surveillance des systèmes à événements discrets ont mis en évidence l'importance du facteur temps dans ce domaine. Ils ont montré que ce facteur est souvent un porteur d'informations dans le cadre de la surveillance. L'exploration des données temporelles relatives aux événements qui peuvent avoir lieu dans un SED s'est avérée très fructueuse dans un processus de surveillance. C'est à partir de ces faits que nous élaborons, dans ce chapitre, une approche de détection pour les systèmes à événements discrets qui ont un comportement appelé "acceptable". Ce comportement peut être adopté dans un système pouvant être sujet aux défauts intermittents, afin d'augmenter sa disponibilité. Ce comportement sera présenté sur un modèle de l'automate à chronomètres.
Le mécanisme développé dans notre approche a pour principe d'une part, d'exploiter les contraintes temporelles dont on peut disposer sur le comportement acceptable du système ; et d'autre part, d'utiliser les techniques d'analyse d'atteignabilité de l'automate à chronomètres. Nous considérons que les dynamiques d'exécution des services ou des tâches du système, sont observables. Les démarches de surveillance que nous proposons se basent tout d'abord sur les événements observables afin d'estimer l'état dans lequel le système peut se trouver. Nous exploitons par la suite les contraintes temporelles calculées dans chaque situation du système, sur les événements dans le but de détecter les défauts. Avant de présenter notre approche de surveillance, il faut fixer au préalable quelques hypothèses qui sont :

- La surveillance se base seulement au niveau de la tâche (service rendu par une ou plusieurs ressources), et non au niveau du matériel (moteur, vanne, pompe, capteur ..). Par conséquent, la notion de la tâche apparaît clairement dans le modèle de surveillance.
- La vitesse d'exécution d'une tâche est constante.
- Un état intermédiaire a été prévu entre l'état de fonctionnement normal d'une tâche et l'état de défaut. Ceci permettra de prendre en compte un intervalle d'exécution dit "acceptable". Par conséquent, *le comportement acceptable* d'une tâche ou d'un système, est introduit.
- Le modèle représente le comportement acceptable et non pas le comportement complet (avec les défauts).
- L'approche surveillera l'ordre d'occurrence des événements et leurs dates d'occurrence dans le comportement acceptable.
- L'étude se restreint à un type des défauts, appelés *les défauts interruptibles* qui sont très répandus dans les SED. L'objectif est de détecter ces défauts au plus tôt. Malgré la restriction que nous avons faite, nous allons voir que le système de surveillance peut prendre en compte d'autres types des défauts.

Nous allons présenter la démarche de surveillance que nous avons développée [Allahham and Alla, 2006], [Allahham and Alla, 2007b], [Allahham and Alla, 2007a], et dont les différentes étapes sont présentées par la suite. Nous modélisons, dans un premier temps, le comportement du système sujet aux défauts interruptibles en utilisant l'automate à chronomètres. Nous donnons les propriétés caractérisant les exécutions acceptables des tâches du système. Ensuite, une procédure de synthèse est appliquée à cet automate afin de délimiter seulement les trajectoires vérifiant les propriétés requises. Dans l'automate résultant, les sommets représentent les situations atteignables du système, les équations différentielles dans chaque sommet reflètent les états des tâches de système dans cette situation et le franchissement d'une transition est conditionnée seulement par l'occurrence de l'événement associé à cette transition. L'espace temporel synthétisé dans chaque sommet délimite le comportement acceptable dans la situation correspondante du système.
L'originalité de notre méthode de surveillance réside dans la combinaison entre le concept des contraintes temporelles d'une tâche et le pouvoir d'analyse de l'automate à chronomètres (analyse en avant et en arrière). Ceci nous permet de calculer un ensemble d'inégalités algébriques (l'espace temporel) dans chaque sommet, où sa violation nous permet de détecter les défauts interruptibles au plus tôt. L'illustration du terme "*détection au plus tôt*" exige de se référer aux méthodes existant dans la littérature, présentées au chapitre 3. Ces méthodes détectent un défaut soit lorsque un événement a eu lieu plus tôt qu'il ne fallait ou lorsque cet événement ne s'est pas produit malgré l'expiration de la date au plus tard prévue de son apparition (voir le remarque 3.2.3). Donc, nous voulons par le terme "*détection au plus tôt*" une détection de l'absence ou du retard d'un événement sans attendre l'expiration de la date au plus tard prévue de son apparition. Ceci représente une exigence de plus en plus demandée dans le milieu industriel.
La démarche de construction du système de surveillance s'effectue d'une part, hors ligne à travers la construction du système de surveillance et d'autre part, en ligne pour son implémentation.

Nous allons présenter dans la suite de ce chapitre notre approche de surveillance. D'abord, nous commençons par préciser la notion de comportement acceptable, notre choix de l'outil de modélisation et le principe de l'approche proposée. Ensuite, nous illustrons intuitivement, par un exemple, le principe de construction du système de surveillance. A travers cette présentation intuitive, nous donnons les motivations conduisant à développer une telle approche. Puis, nous présentons la démarche formelle de la construction de notre système de surveillance. Enfin, nous concluons par appliquer la méthode proposée à un système manufacturier.

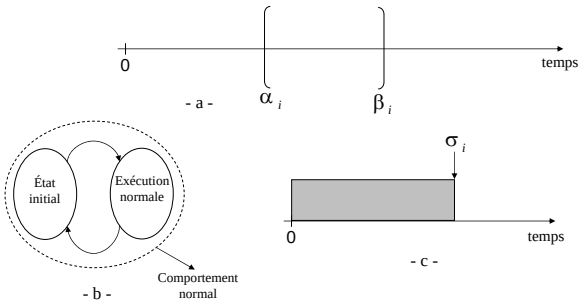

FIG. 4.1 – a- La durée d'exécution normale de $Tâche_i$. b- Le comportement normal c- L'exécution de $Tâche_i$ dans le comportement normal.

4.2 Présentation de l'approche de surveillance proposée

Nous présentons au cours de cette section le principe de notre approche de surveillance des SED. Au cours de cette présentation, nous illustrons la notion de comportement acceptable d'un système, ainsi que la justification du choix de l'outil de modélisation.

4.2.1 Notion de comportement acceptable

Généralement, un système est composé de plusieurs ressources et exécute certaines tâches. Une tâche peut être exécutée par une ou plusieurs ressources. Une des grandeurs pouvant donner l'information sur l'existence d'une déviation par rapport au fonctionnement normal d'une tâche est le temps d'exécution de la tâche. Donc, chaque tâche dans le système (soit $Tâche_i$) est associée à une durée d'exécution $[\alpha_i, \beta_i]$ où α_i et β_i sont respectivement les temps minimal et maximal nécessaires pour l'exécution correcte de la tâche (Fig.4.1.a). Dans le comportement normal du système (Fig.4.1.b), on trouve que $Tâche_i$ bascule de l'état initial à l'état de l'exécution normale. Pendant ce comportement, $Tâche_i$ sera exécutée sans aucune interruption à cause des défauts interruptibles (Fig.4.1.c). L'événement σ_i représentant la fin de $Tâche_i$, se produit dans l'intervalle $[\alpha_i, \beta_i]$. Lorsque $Tâche_i$ se termine, elle revient à l'état initial. Normalement, la durée de séjour $Tâche_i$ dans l'état d'exécution appartient à l'intervalle $[\alpha_i, \beta_i]$. Cet intervalle peut être mesuré ou donné dans les caractéristiques des ressources exécutant $Tâche_i$.

De manière générale, un système composé de ressources exécutant des tâches perd certaines de fonctionnalités de manière provisoire lorsqu'une ou plusieurs ressources sont sujettes aux défauts interruptibles intermittents. Dans la mesure de rentabilité et pour ne pas pénaliser le système, la tolérance à ces défauts est indispensable. Dans cet objectif, le concepteur des systèmes industriels adoptent une tolérance aux durées d'exécution de ses tâches [Rayhane, 2004], [Valette et al., 1989]. Ceci représente un moyen pratique afin

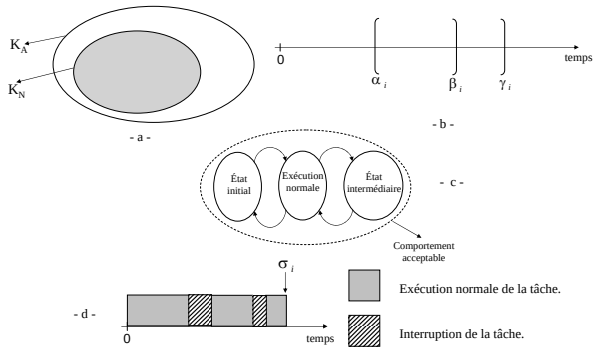

FIG. 4.2 – a- Le comportement acceptable d'un système b- Les durées d'exécution normale et acceptable de $Tâche_i$. c- Le comportement acceptable de $Tâche_i$. d- L'exécution acceptable de $Tâche_i$.

d'augmenter la disponibilité de ces systèmes. La considération de ce point de vue nous conduit à introduire le comportement acceptable du système. Ce comportement exprime le fait que le système exécute ses tâches, même en présence de défauts intermittents. Par conséquent, l'ensemble des états atteignables dans le fonctionnement acceptable est plus grand que l'ensemble correspondant à ceux atteignables en fonctionnement normal. Dans la figure 4.2.a, les états indiqués par K_N et K_A représentent respectivement les états atteignables en comportement normal et acceptable.
Si nous considérons la contrainte temporelle comme nous l'avons précédemment présentée, un autre intervalle sera associé à une tâche, $[\alpha_i, \gamma_i)$ où $\gamma_i > \beta_i$ (Fig. 4.2.b). Cet intervalle est dit "la durée acceptable d'exécution" et représente la durée d'exécution de $Tâche_i$ dans le comportement acceptable. Dans ce comportement de $Tâche_i$ présenté dans la figure 4.2.c, $Tâche_i$ peut être interrompue suite à un défaut intermittent. Par conséquent, l'état de $Tâche_i$ bascule vers un état intermédiaire. Dans cet état, le défaut peut disparaître et l'état de $Tâche_i$ bascule à nouveau vers l'état d'exécution normale. La tâche reprend au même endroit où le défaut est apparu. L'exécution de $Tâche_i$ dans le comportement acceptable est présentée dans la figure 4.2.d. Lorsque $Tâche_i$ se termine, elle revient à l'état initial. L'exécution de $Tâche_i$ est acceptable si la durée d'exécution appartient à l'intervalle $[\alpha_i, \gamma_i)$ lorsque l'événement σ_i apparaît. Cette durée peut être donnée par le concepteur ou par l'ordonnanceur des tâches du système. En effet, nous élaborons une approche de surveillance pour les systèmes dont on connaît a priori les durées exactes et tolérées de ses tâches.

4.2.2 Choix de l'outil de modélisation du comportement acceptable

Plusieurs outils existent dans la littérature permettant la modélisation des SED. Parmi les outils les plus utilisés, nous citons les automates et les réseaux de Petri.

Toutefois, le choix de l'outil de modélisation dépend de la nature du système ainsi que de l'application considérée. Par exemple, quand il s'agit de modéliser un SED pour une application nécessitant une connaissance quantitative de son comportement temporel tel que la synthèse de la commande [SAVA, 2001] ou le diagnostic ou la surveillance, l'information temporelle doit être considérée dans le modèle. Dans notre cas, il faut également prévoir un outil de modélisation permettant d'une part la prise en compte du temps et d'autre part la description comportementale du système, suite aux apparitions et disparations d'un défaut. Tant que nous nous intéressons aux défauts interruptibles, la tâche sujette à un défaut sera suspendue. Elle reprend au même endroit lorsque le défaut disparaît. Pour cela, nous avons choisi d'utiliser l'outil automate à chronomètres pour modéliser le comportement acceptable de ces systèmes. Ce choix est justifié également par la puissance d'analyse de ce modèle. En effet de nombreuses techniques et outils d'analyse d'atteignabilité sont disponibles

En effet, les outils utilisés dans les méthodes existant dans la littérature sont les RdP temporels ou les automates temporisés. Ces outils ne peuvent pas représenter le comportement acceptable du système. Ceci est dû à la modélisation de ce comportement qui a besoin de la notion du chronomètre (l'horloge pour laquelle l'écoulement du temps peut être suspendu puis repris plus tard). Le RdP temporel et l'automates temporisé se basent sur la notion d'horloge classique [Bengtsson and Yi, 2004].

Comme nous l'avons remarqué dans le chapitre 3, un des modèles possibles dans le cas du procédé à *instance-multiple* est le RdP temporel avec la sémantique de multi-sensibilisation [Boufaied, 2003]. Prenons l'exemple présenté dans la figure 3.9.a. Supposons qu'on introduit la notion de comportement acceptable sur la partie composée du convoyeur B. Supposons aussi qu'on veuille surveiller la tâche de transfert des palettes sur ce convoyeur. C'est à partir de ce besoin que nous proposons une extension du réseau de Petri temporel basée sur la notion de chronomètre. Ce modèle appelée "*Réseaux de Petri à chronomètres Post et Pré-initialisés*", sera détaillé au chapitre 5. En plus de l'intrêt précédent de ce modèle, il nous permet, en général, de représenter le comportement acceptable d'un système de manière lisible, concise et simple. Ceci n'est pas le cas de la modélisation directe par les automates à chronomètres (composition synchrone).

Nous allons présenter, au cours de ce chapitre, la méthode de conception de notre système de surveillance basé sur l'automate à chronomètres. Cela consistera à construire les modèles des différentes tâches constituant le système en utilisant les automates à chronomètres. Par conséquent, le modèle global du système sera aussi sous la forme d'automate à chronomètres (résultant de la composition synchrone des modèles des différentes tâches). La conception à partir d'un modèle *réseau de Petri à chronomètres Post et Pré-initialisés* sera détaillé dans le chapitre 5.

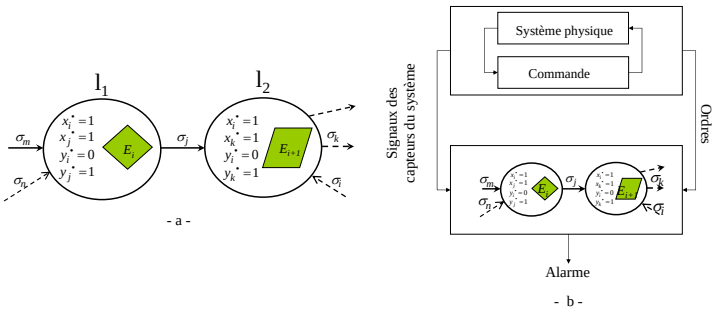

FIG. 4.3 – a- Le modèle du système commandé basé sur l'automate à chronomètres. b- La structure proposée afin d'implémenter notre approche de surveillance.

4.2.3 Principe de l'approche proposée

La construction de système de surveillance représente un travail à réaliser d'abord de manière hors ligne. Le modèle utilisé comme référence dans notre méthode de surveillance est le modèle de comportement acceptable du système. Ce modèle étant un automate à chronomètres (Fig. 4.3.a), les sommets représentent les différentes situations atteignables du système et les équations différentielles dans un sommet reflètent les dynamiques des tâches dans ce sommet : actives ou interrompues à cause des défauts interruptibles. L'espace temporel dans un sommet caractérise les évolutions temporelles possibles du système dans la situation correspondante du système. Il est calculé de telle manière qu'il délimite seulement l'exécution acceptable des tâches du système. Un événement de l'automate correspond à un événement dans le système à surveiller. Il peut être soit un signal du capteur ou un ordre provenant du système de commande. Le franchissement d'une transition peut initialiser certains chronomètres. L'automate est synchronisé sur l'événement provenant du procédé commandé. Il est également temporisé sur les horloges internes. Cela signifie que l'événement associé à une transition peut se produire tant que les valeurs des chronomètres vérifient l'espace temporel associé au sommet source. L'automate peut rester dans un sommet tant que les valeurs des chronomètres vérifient l'espace temporel de ce sommet. Ainsi, cet automate permettra de détecter un défaut suite à la violation de l'espace temporel associé à un sommet. Ceci peut être illustré par l'automate présenté dans la figure 4.3.a. Lorsque l'événement σ_j se produit alors que les valeurs des chronomètres x_i, x_j, y_i et y_j appartiennent à l'espace E_i, la commutation $l_1 \rightarrow l_2$ aura lieu. Une alarme sera déclenchée lorsque la relation suivante est vérifiée : $x_i, x_j, y_i, y_j \not\models E_i$.
La figure 4.3.b montre la structure de notre approche de surveillance. Notre système de surveillance représenté par l'automate à chronomètres, reçoit les signaux provenant du système à surveiller. L'évolution discrete de cet automate se produit suite à l'observation d'un événement généré par le système à surveiller. En cas de violation de l'espace temporel associé à un sommet, le système de surveillance donne une alarme.

4.3 Construction du système de surveillance

Après la discussion présentée dans la section précédente, nous constatons que la construction du système de surveillance doit considérer les deux problèmes suivants. Le premier problème consiste à prendre en compte dans la structure du système de surveillance le changement de la dynamique suite à un défaut. Le système bascule, suite à un défaut intermittent, à un état intermédiaire. Lorsque ce défaut disparaît, le système revient à l'exécution normale. Les entrées du système de surveillance sont des signaux provenant des capteurs et des ordres ; ils doivent donc permettre de suivre les dynamiques d'exécution des tâches du système. Le deuxième problème concerne la détermination de l'espace temporel, dans chaque sommet de l'automate, qui caractérise le comportement acceptable.

Nous détaillons dans la suite une démarche de construction du système de surveillance. Nous construisons d'abord un modèle du système sous la forme d'automate à chronomètres. Ce dernier reflète le changement de la dynamique de système suite à un défaut interruptible. Nous considérons dans la construction du modèle les hypothèses mentionnées dans l'introduction de ce chapitre. Nous noterons dans la suite par $\mathbb{A}$, l'automate obtenu par la composition synchrone des modèles des différentes tâches du système représentées par $\mathbb{A}_1$, $\mathbb{A}_2$, $\mathbb{A}_3$,...., $\mathbb{A}_n$ où n le nombre des tâches du système. Une procédure de synthèse sera appliquée à cet automate afin de déterminer l'espace temporel mentionné ci-dessus. Nous indiquerons par $\mathbb{A}^*$ l'automate après la synthèse.
Afin de bien illustrer la démarche de construction du système de surveillance, nous commençons d'abord par une présentation intuitive, ensuite nous présenterons la démarche formelle permettant cette construction.

4.3.1 Présentation intuitive de l'approche proposée de surveillance

La présentation intuitive de notre méthode de surveillance est faite en considérant un exemple simple de SED commandé.

A Spécification de l'exemple

Considérons un système simple d'un convoyeur (Fig. 4.4.a). Ce système est composé d'un convoyeur, de deux capteurs, d'un moteur M et d'un contrôleur (Fig. 4.4.b). Ce convoyeur est destiné à transférer une palette du point A au point B. Le convoyeur commence sa tâche lorsque le contrôleur donne l'order d. La fin de cette tâche est détectée par le capteur au point B qui produit l'événement b. Le deuxième capteur logique produit

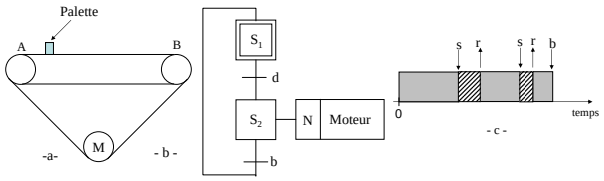

FIG. 4.4 – a- Le système du convoyeur considéré. b- Le système de commande du convoyeur. c- Une exécution acceptable de la tâche du convoyeur.

les événements s et r. Ces événements indiquent l'interruption et la reprise du mouvement de la palette. Autrement dit : ce capteur permet de suivre la dynamique d'exécution de la tâche du convoyeur (Fig. 4.4.c). La palette se déplace sur le convoyeur à une vitesse constante. La durée nécessaire pour exécuter la tâche du transfert est de 10 $u.t$ (Nous avons pris $\alpha = \beta = 10$). Le fait que la tâche du système est exécutée pendant sa durée et sans occurrence des défauts intermittents, représente *le comportement normal* de la tâche de convoyeur. Rappelons qu'un défaut intermittent arrête l'exécution de la tâche du convoyeur pendant une durée indéterminée Alors, un état intermédiaire apparaît dans lequel le convoyeur peut revenir à l'exécution normale de sa tâche. Lorsque ce défaut disparaît, la tâche reprend au même endroit. Ce comportement représente *le comportement acceptable* de la tâche de convoyeur. Comme spécification, nous supposons que la durée acceptable pour exécuter la tâche de transfert est de $[10, 12)$ $u.t$. Comme nous avons dit, cette tolérance a pour but d'augmenter la disponibilité du système. Un défaut intermittent peut être le résultat des perturbations externes. Donc, si la perturbation s'arrête, le défaut disparaît. Ce défaut peut arriver dans le cas du blocage de la palette sur le convoyeur. Il peut se produire à cause d'un défaut interne du moteur M ou de son alimentation. Le défaut peut être dans la partie mécanique du convoyeur lui-même.

Remarque 1. • Comme on a pu la remarquer dans le chapitre 3, le travail de [Correcher et al., 2003] a pu montrer l'importance d'observer le dynamique de l'actionneur subissant des défauts intermittents par des capteurs. Ces capteurs rendent le système diagnostiquable ou détectable. Ainsi, il est possible d'extraire des informations supplémentaires concernant la fréquence et l'instant d'apparition d'un défaut intermittent.
• Le capteur observant l'interuption et la reprise du convoyeur, n'est pas un capteur de défaut. Ceci est dû à la tâche du convoyeur qui peut être exécutée même si elle est interrompue à cause des défauts intermittents. Dans la suite de notre travail, les dynamiques d'exécutions des tâches sont observables par des capteurs logiques. Ces capteurs représentent une source supplémentaire d'informations.

B Conception du système de surveillance

Nous allons construire progressivement le modèle de surveillance du convoyeur présenté dans la figure 4.5. Dès que le contrôleur donne l'ordre d, deux chronomètres x et

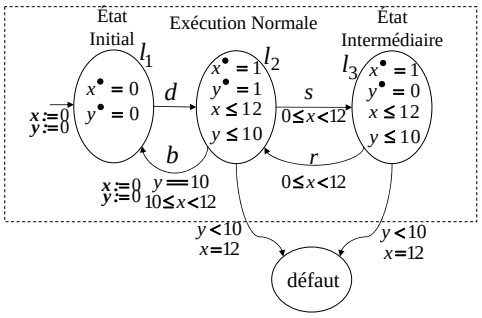

FIG. 4.5 – L'automate à chronomètres représentant le comportement du convoyeur

y sont initialisés (c'est-à-dire : $x = y = 0$). La dynamique de y reflète l'état de la tâche, autrement dit : $\dot{y} = 1$ quand le convoyeur transfère la palette. Tandis que $\dot{y} = 0$ lorsque la palette est arrêtée sur le convoyeur. Le chronomètre x mesure le temps écoulé depuis l'instant du début de la tâche jusqu'à sa fin. C'est-à-dire : la dynamique de x est $\dot{x} = 1$ depuis l'instant de début de la tâche jusqu'à celui représentant sa fin et correspondant à l'occurrence de l'événement b. Lorsque cet événement b se produit, la valeur de x indique si la tâche a été exécutée pendant la durée acceptable $[10, 12)$ $u.t$. Normalement, la palette se déplace sur le convoyeur et arrive au point B (ou $y = 10$) à l'instant $x = 10$. Au cours de l'exécution de la tâche du convoyeur, la palette peut être bloquée temporairement ou de manière permanente à cause d'un défaut inetrruptible. Le blocage de la palette (l'occurrence de l'événement s) implique un changement de l'état du système $l_2 \rightarrow l_3$. Dans ce nouvel état, le chronomètre y s'arrête puisque la palette n'avance pas sur le convoyeur et l'horloge x reste active. Lorsque le défaut disapparaît, la tâche redevient active (l'occurrence de l'événement r), le système revient à son état précédent $l_3 \rightarrow l_2$ et le chronomètre y reprend. Les valeurs $y = 10$ et $10 \le x < 12$ expriment le fait que la palette est arrivé au point B pendant la durée acceptable. Ces valeurs définissent un espace qui sera appelé dans la suite par *l'espace désiré* D. Il est possible que la tâche de convoyeur atteigne l'état fautif. Dans ce cas, les valeurs des chronomètres vérifient la garde : $(x = 12 \wedge y < 10)$. Notre but est de déterminer toutes les trajectoires possibles permettant d'atteindre l'espace D. Autrement dit, on a besoin de construire l'espace temporel contenant les trajectoires qui atteignent seulement l'espace D.
La figure 4.6.a présente l'espace temporel aux sommets l_2 et l_3. Cet espace contient toutes les trajectoires possibles, y compris celles atteignant l'état fautif. La figure 4.6.b présente l'espace temporel (le parallélogramme) après l'application de la procédure de synthèse schématisée dans la figure 4.6.c. Cet espace contient seulement les trajectoires représentant l'exécution acceptable de la tâche. La solution formelle est basée sur l'utilisation de l'analyse en arrière (ou l'opérateur $Pre(D)$) de l'espace D. L'opérateur $Pre(D)$ calcule tous les états qui mènent à D. Le calcul de cet opérateur avec d'autres opérations théoriques constituent les principales étapes de la procedure de synthèse (Fig 4.6.c). Dans cette figure, $\mathbb{A}$ et $\mathbb{A}^*$ indiquent respectivement les automates à chronomètres représentant

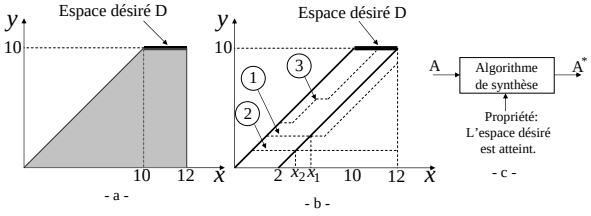

FIG. 4.6 – a- L'espace temporel contenant toutes les trajectoires possibles b- L'espace synthétisé c- Présentation schématique de la procédure de synthèse proposée.

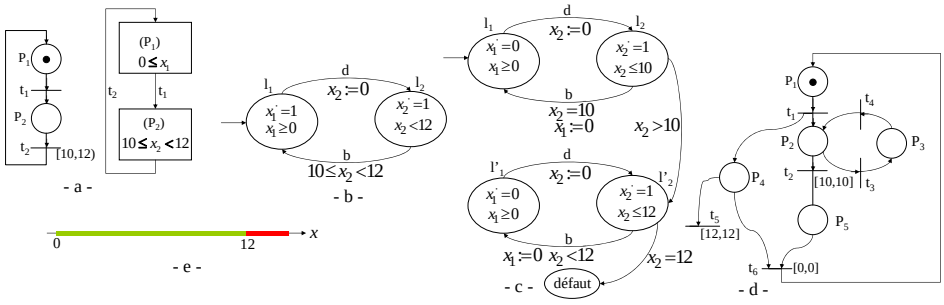

FIG. 4.7 – Surveillance du système de convoyeur par les méthodes existantes dans la littérature :a- par un RdP temporel. b,c- par un automate temporisé. d- représentation de l'interruption et de la reprise par un réseau de Petri temporel. e- l'espace temporel atteignable.

le modèle du système (ici le modèle du convoyeur) avant et après l'application de procédure de synthèse. Une autre entrée de la procédure de synthèse représente la propriété à vérifier : l'espace désiré est atteint.
Afin de montrer l'utilisation de cet espace (le parallélogramme), considérons les cas suivants présentés dans la figure 4.6.b :

• Cas 1 : Le système de surveillance déclenche une alarme à l'instant x_1 même si la palette n'a pas été bloquée définitivement, parce qu'on est sûr quoi qu'il puisse arriver la durée de transfert sera égale ou supérieure à 12 $u.t$.

• Cas 2 : La palette a été bloquée définitivement, notre système de surveillance déclenche donc une alarme à l'instant x_2. Les méthodes existantes détectent ce défaut, comme nous allons le voir dans le paragraphe suivant, à l'instant 12 $u.t$. Ici ce défaut est détecté au plus tôt sans attendre l'instant 12 $u.t$.

• Cas 3 : Le système de surveillance ne détecte pas de défaut même si certaines interruptions ont lieu, car la tâche a été exécutée pendant sa durée acceptable. Ceci renforce le fait que le capteur produisant les événements s et r, n'est pas un capteur de défauts.

C Surveillance du système par les méthodes existantes

Supposons qu'on veuille surveiller le système de convoyeur présenté ci-dessus, en utilisant les méthodes présentées dans le chapitre 3. Les figures 4.7.a,b,c montrent les modèles de surveillance utilisés dans ces méthodes. La figure 4.7.a représente un RdP temporel modélisant le comportement acceptable de l'exemple et son espace temporel atteignable. Les événements d et b sont associés respectivement aux transitions t_1 et t_2. La surveillance de l'espace temporel associé au marquage P_2 permet de détecter un défaut pour les scénarios présentés ci-dessus (cas 1 et 2) à l'instant 12 $u.t$. Cet instant représente la borne supérieure associée à la transition t_2. Autrement dit : le défaut sera détecté à l'instant d'expiration de la date au plus tard prévue d'apparition de l'événement b représentant la fin de tâche de convoyeur. On notera que la surveillance de l'espace temporel dans le sommet l_2 de la figure 4.7.b, permet également de détecter un défaut à l'instant 12 $u.t$ pour les scénarios présentés ci-dessus (cas 1 et 2).
La figure 4.7.c présente le modèle du convoyeur établi en utilisant la méthode détaillée dans [Rayhane, 2004]. Dans ce modèle, nous remarquons que si l'événement b ne se produit pas au delà la durée $x_2 = 10$ $u.t$ le modèle bascule du sommet l_2 au sommet l'_2. Ce sommet appartient au mode de fonctionnement dégradé. Dans ce sommet, si l'événement b arrive pour une valeur de $x_2 < 12$, l'automate bascule vers le sommet l'_1 (contenu dans le mode de fonctionnement dégradé), sinon il bascule vers le sommet représentant l'occurrence d'un défaut.
La question qu'on peut relever est la suivante : si la dynamique d'exécution de la tâche du convoyeur est observable par les événements s et r, est ce-que ces modèles sont capables de profiter de ces événements ? Supposons qu'on modifie le modèle présenté dans la figure 4.7.a par celui présenté dans la figure 4.7.d. Dans ce nouveau modèle, la transition t_1 représente le début de la tâche, et la place P_2 représente l'exécution de la tâche. Les transitions t_3 et t_4 représentent respectivement les événements s et r. Les franchissements de ces transitions t_3 et t_4 représentent respectivement l'occurrence de l'interruption et de la reprise de la tâche de transfert modélisée par la place P_2. On veut mesurer la durée d'exécution effective de la tâche de convoyeur par la transition t_2. La durée totale est mesurée par la transition t_5. Lorsque la valeur de l'horloge x_2 (mesurant la durée d'exécution effective) égale à 10 et la place p_4 est marquée (Autrement dit : la valeur d'horloge associée à t_5 est $x_5 < 12$), les transitions t_2 et t_6 sont franchies immédiatement. Le franchissement de t_6 signifie que la tâche est exécutée pendant sa durée acceptable. Lorsqu'une interruption se produit par le franchissement de t_3, l'horloge x_2 devient inactive. Lorsque la reprise se produit par le franchissement de t_4, l'horloge x_2 devient active à nouveau et t_2 est nouvellement sensibilisée. Par conséquent, x_2 sera initialisée et la durée d'exécution de la tâche avant l'interruption est perdue. Nous pouvons constater que l'horloge ne peut pas représenter le comportement acceptable de la tâche. Ceci montre le besoin de la notion du chronomètre (l'horloge pour laquelle l'écoulement du temps peut être suspendu puis repris plus tard).

Nous pouvons remarquer que, dans tous les modèles présentés, l'espace temporel à surveiller est le segment de temps entre 0 et 12 présenté dans la figure 4.7.e. La surveillance de cet espace détecte un défaut interruptible de la tâche du convoyeur (tel que présenté dans les cas 1 et 2 ci-dessus) à l'instant 12. Cet instant représente la date au plus tard de la fin de la tâche.

4.3.2 Démarche formelle de la construction de système de surveillance

Nous nous intéressons à la surveillance des systèmes à événements discrets commandés présentés dans le chapitre 2. L'aspect temporel d'un système à surveiller est pris en compte à travers les spécifications temporelles représentant les durées d'exécution des différentes tâches du système. Donc, un système sera décomposé en plusieurs tâches.
Généralement, nous considérons le cas dans lequel une tâche peut subir des défauts interruptibles. Par conséquent, nous distinguons dans la suite entre une tâche interruptible et une tâche non-interruptible. Une tâche est interruptible si l'on peut surveiller sa dynamique suite à l'occurrence d'un défaut interruptible, c'est-à-dire : on peut observer son interruption et sa reprise suite à l'apparition et disparation d'un défaut intermittent. Sinon, on considère que la tâche est non-interruptible.

A Modélisation du comportement des SED commandés subissant aux défauts interruptibles

Soit $Tâche_{int}$ l'ensemble des tâches interruptibles dans un système S. Soit $Tâche_i \in Tâche_{int}$ une tâche interruptible dans S. $Tâche_i$ a une durée d'exécution normale connue $I_{nor} = [\alpha_i, \beta_i]$. Cette durée est donnée dans les caractéristiques techniques des ressources qui exécutent la tâche ou elle peut être mesurée directement. Elle représente la durée d'exécution normale présentée dans le paragraphe précédent où α_i correspond au temps nécessaire pour la réalisation de $Tâche_i$ à une vitesse constante. La durée β_i tient compte des retards qui peuvent être dûs aux conditions d'exploitation des ressources exécutant $Tâche_i$ (par exemple, la surcharge ou le vieillissement de ces ressources). Autrement dit : I_{nor} considère le ralentissement de la vitesse étant supposée constante. Remarquons que nous pouvons considérer $I_{nor} = [\alpha_i, \alpha_i]$ où $\alpha_i = \beta_i$. La tâche $Tâche_i$ a aussi une durée d'exécution acceptable, donnée par l'intervalle $I_{acc} = [\alpha_i, \gamma_i)$ où $\beta_i < \gamma_i$. Cette durée considère le retard d'exécution résultant d'occurrence des défauts interruptibles intermittents. Nous appelons, respectivement I_{nor} et I_{acc}, les durées d'exécution normale et acceptable de la tâche $Tâche_i$. Les effets de l'apparition et la disparition d'un défaut sur l'exécution d'une tâche seront indiqués respectivement par l'interruption et la reprise de la tâche.

Afin de surveiller la tâche $T\hat{a}che_i$, nous utilisons comme nous l'avons vu dans le paragraphe précédent, deux chronomètres x_i et y_i. Le chronomètre x_i mesure la durée totale d'exécution depuis l'instant du début de la tâche, tandis que y_i accumule les durées d'exécutions partielles. Autrement dit : il mesure la durée d'exécution effective. Les deux chronomètres ont une valeur nulle au début de la tâche. x_i est utilisé afin de vérifier que la tâche est exécutée avant l'expiration de la date tolérée. y_i mesure la durée effective d'exécution. La tâche $T\hat{a}che_i$ est exécutée correctement si les valeurs $y_i \in [\alpha_i, \beta_i]$ et $x_i \in [\alpha_i, \gamma_i)$ lorsque l'événement fin de tâche se produit.

A.1 Modélisation d'une tâche interruptible Nous allons présenter, par la suite, le modèle *SWA* représentant le comportement d'une tâche interruptible $T\hat{a}che_i$. Cet automate (Fig. 4.8.a) contient trois sommets l_1, l_2 et l_3 qui représentent respectivement les états : *initial, execution normale et interruption*. Les chronomètres x_i et y_i sont initialisés à l'entrée de l_1. Dans ce sommet, les deux chronomètres sont inactifs, autrement dit : les dynamiques des chronomètres x_i et y_i sont 0. Lorsque l'événement de début de tâche (soit l'événement d_i) arrive, l'état de la tâche bascule vers le sommet l_2 représentant l'exécution normale. Dans ce sommet, les deux chronomètres sont actifs, c'est-à-dire : les dynamiques de deux chronomètres sont 1. L'invariant de ce sommet est $I(l_2) = (x_i \leq \gamma_i \wedge y_i \leq \beta_i)$ où γ_i représente la durée maximale totale acceptable pour exécuter $T\hat{a}che_i$ et β_i représente la durée maximale nécessaire pour exécuter $T\hat{a}che_i$. Dans le sommet l_3, la dynamique de y_i devient 0 tandis que la dynamique de x_i reste 1. Ceci est pour exprimer le fait que le temps s'écoule mais la tâche est arrêtée.
La transition $l_2 \xrightarrow{g_1} l_3$ exprime l'interruption de $T\hat{a}che_i$ suite à un défaut interruptible. L'événement de cette transition est e_{s_i} représentant l'événement indiquant l'interruption de $T\hat{a}che_i$ et provenant du système physique. La garde de cette transition étant $g_1 = 0 \leq x_i < \gamma_i$, exprime le fait que l'interruption peut arriver à importe quel instant pendant l'exécution.
La transition $l_3 \xrightarrow{g_2} l_2$ exprime la reprise de $T\hat{a}che_i$ suite à la disparition du défaut. L'événement de cette transition est e_{r_i} représentant l'événement indiquant la reprise de $T\hat{a}che_i$ et provenant du système physique. La garde de cette transition étant $g_2 = 0 \leq x_i < \gamma_i$, exprime le fait que $T\hat{a}che_i$ reprend avant de dépasser la durée acceptable d'exécution la tâche.
Le fait que la tâche $T\hat{a}che_i$ est exécutée correctement, est modélisé par la transition de $l_2 \xrightarrow{g_3} l_1$. L'événement de cette transition est σ_i représentant l'événement indiquant la fin de $T\hat{a}che_i$ et provenant du système physique. La garde de cette transition étant $g_3 = (\alpha_i \leq x_i < \gamma_i \wedge \alpha_i \leq y_i \leq \beta_i)$. Cette garde exprime le fait que $T\hat{a}che_i$ est exécutée $\alpha_i \leq y_i \leq \beta_i$ et pendant la durée acceptable $\alpha_i \leq x_i < \gamma_i$.
Dans la figure 4.8.a, on trouve que $T\hat{a}che_i$ quitte le comportement acceptable vers l'état fautif soit de l_2 ou l_3. Ces faits sont représentés par les transitions : $l_2 \xrightarrow{g_4} l_4$ et $l_3 \xrightarrow{g_4} l_4$.

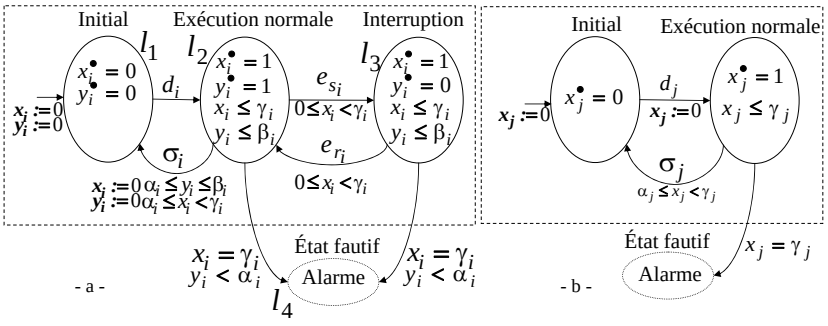

FIG. 4.8 – Le modèle d'une tâche : a- interruptible b- non-interruptible.

Les gardes de ces transitions sont identiques : $g_4 = (x_i = \gamma_i \wedge \alpha_i > y_i)$. Cete garde exprime le fait que la durée d'exécution acceptable de $Tâche_i$ est dépassée (c'est-à-dire : $x_i = \gamma_i$), mais la tâche n'est pas exécutée (c'est-à-dire : $\alpha_i > y_i$). On peut remarquer que $g_3 \times g_4 = 0$. Ceci signifie que l'état de $Tâche_i$ ne peut pas être dans l'état fautif l_4 et les valeurs des chronomètres vérifient g_3.

On peut maintenant définir l'automate à chronomètres représentant le comportement d'une tâche interruptible.

Définition 4.3.1. *L'automate à chronomètres représentant le comportement d'une tâche interruptible* $Tâche_i$ *est* $\mathbb{A}_i = (L, l_1, X, \Sigma_i, A, I, Dif)$ *où :*
- $L = \{l_1, l_2, l_3\}$, *et* l_1 *est le sommet initial,*
- $X_i = \{x_i, y_i\}$, $\Sigma_i = \{\sigma_i, e_{s_i}, e_{r_i}, d_i\}$,
- $I(l_3) = I(l_2) = \{0 \le x_i \le \gamma_i, 0 \le y_i \le \beta_i\}$,
- $Dif(l_2)(x_i) = Dif(l_3)(x_i) = 1$, $Dif(l_1)(x_i) = 0$
$Dif(l_2)(y_i) = 1, Dif(l_3)(y_i) = Dif(l_1)(y_i) = 0$.

□

Pour une tâche interruptible $Tâche_i$ ayant une durée d'exécution normale $I_{nor} = [\alpha_i, \beta_i]$, une durée d'exécution totale $I_{acc} = [\alpha_i, \gamma_i)$ et un modèle comme celui présenté dans la définition 4.3.1, on définit l'espace de commutation sans défauts :

Définition 4.3.2. *L'espace temporel de commutation sans défauts de* $Tâche_i$*, noté* D_i *et appelé l'espace désiré de* $Tâche_i$*, est :*

$$\forall Tâche_i \in Tâche_{int}, \forall x_i, y_i \in X_i : x_i \in I_{acc} \wedge y_i \in I_{nor}$$

□

En effet, l'espace temporel D_i de $Tâche_i$ est la garde de la transition $l_2 \rightarrow l_1$ de l'automate $\mathbb{A}_i$ présenté dans la figure 4.8.a. Cette transition représente l'exécution acceptable de $Tâche_i$.

Remarque 2. Une des trajectoires possibles $L \times X$ dans l'automate $\mathbb{A}_i$ peut être définie comme :
• Initialement : elle commence de l'état $(l_1 \times (x_i = 0,\, y_i = 0))$,
• Des évolutions continues et discrètes de la forme : : $(l_1 \times (0,0))$,...., $(l_2 \times (\alpha_i,\, \alpha_i))$, $(l_1 \times (0,0))$ et $(l_1 \times (0,0))$,...., $(l_4 \times (\varepsilon, \gamma_i))$ où ε est une valeur tel que $\varepsilon < \alpha_i$.

□

Remarquez qu'il y a des trajectoires possibles permettant d'atteindre l'état fautif (la garde $y_i < \alpha_i \wedge x_i = \gamma_i$ sera vérifiée). Le fait que $Tâche_i$ est exécutée pendant sa durée acceptable est présenté par la propriété suivante.

Propriété 1. Les trajectoires $L \times X$ qui caractérisent l'exécution acceptable de $Tâche_i$ vérifient la relation :

$$(L \times X) \cap D_i \neq \phi$$

□

Remarque 3. Soit $Tâche_j$ une tâche non-interruptible ayant une durée d'exécution acceptable $[\alpha_j, \gamma_j)$. Nous modélisons cette tâche par un automate comme celui présenté dans la figure 4.8.b. Dans cet automate, σ_j et d_j sont respectivement les événements représentant la fin et le début de $Tâche_j$.
□

A.2 L'automate à chronomètres du système Comme nous venons de l'expliquer, les modèles des tâches individuelles du système sont d'abord construits. Le modèle du système complet est ensuite obtenu en composant les modèles construits selon la définition 2.3.3. Soit $\mathbb{A}$ l'automate à chronomètres obtenu par la composition de $\mathbb{A}_i$ et $\mathbb{A}_j$, correspondant aux tâches $Tâche_i$ et $Tâche_j$. Nous trouvons dans l'automate $\mathbb{A} = \mathbb{A}_i \parallel \mathbb{A}_j$ que les deux automates à chronomètres $\mathbb{A}_i$ et $\mathbb{A}_j$ évoluent indépendamment pour un ensemble d'actions (Σ-Σ') et évoluent d'une manière synchrone (les deux ensembles) pour un ensemble commun d'actions Σ'.
Dans l'automate $\mathbb{A}$ résultant de la composition, on définit l'ensemble : $\mathbb{D} = \{D_1, D_2, ...\}$. Cet ensemble contient les espaces temporels désirés de toutes les tâches interruptibles dans le système S représenté par l'automate $\mathbb{A}$. Autrement dit, $\forall\ Tâche_i \in Tâche_{int}$: son espace désiré $D_i \in \mathbb{D}$. L'exécution acceptable de toutes les tâches dans S peut être représentée par une propriété. Cette dernière est une généralisation de la propriété 1.

Propriété 2. Les trajectoires qui caractérisent l'exécution acceptable de toutes les tâches de S vérifient la relation :

$$\bigcup \{(L \times X) \cap D_i \neq \phi : \forall D_i \in \mathbb{D}\}$$

□

B Synthèse du comportement acceptable d'un système

B.1 Formulation du problème Dans ce paragraphe, nous décrivons la synthèse de l'espace temporel représentant les évolutions possibles dans le comportement acceptable d'un système. Ceci peut être résumé de la manière suivante : soit un automate à chronomètres $\mathbb{A}$ qui représente un système S. Calculer l'automate $\mathbb{A}^*$, à partir de l'automate $\mathbb{A}$, tel que toutes les trajectoires possibles en $\mathbb{A}^*$ satisfont la propriété 2.

Problème 1. Synthèse pour la surveillance.
Soit $\mathbb{A} = (L, l_0, X, \Sigma, A, I, Dif)$ un automate à chronomètres et Q un sous-ensemble des trajectoires $L \times X$ qui vérifient la propriété 2 de $\mathbb{A}$. Le problème est de trouver l'automate $\mathbb{A}^*$ déduit de $\mathbb{A}$ tel que :

$$\text{chaque trajectoire } q \text{ dans } \mathbb{A}^* \Leftrightarrow q \in \mathsf{Q}.$$

□

Le calcul de $\mathbb{A}^*$ représentant la solution du problème 1, est basé sur les techniques d'analyse d'un automate à chronomètres : analyse en avant et en arrière. Ces techniques sont détaillées dans le chapitre 2. Cette construction de $\mathbb{A}^*$ implique deux phases. D'abord, nous calculons en utilisant la technique d'analyse en avant l'espace temporel atteignable de $\mathbb{A}$. Cet espace correspond aux évolutions possibles, y compris celles qui permettent d'atteindre l'état fautif. Dans la deuxième phase, une analyse en arrière sera appliquée sur $\mathbb{A}$ afin de synthétiser seulement les trajectoires qui satisfont la propriété 2.

B.2 Procedure de synthèse basée sur l'analyse d'atteignabilité Avant de donner la procédure de synthèse pour résoudre le problème 1 et représentée dans la figure 4.6.c, nous détaillons chacune de ses étapes.

1− ***L'analyse en avant***

Nous utilisons les opérateurs d'analyse en avant afin de calculer toutes les trajectoires possibles du système. Ceci consiste à calculer les espaces des chronomètres associés aux séjours du système dans chaque sommet du modèle. Il faut noter qu'un sommet du modèle peut être atteint avec des espaces d'horloges différents à son entrée, notamment lorsque le modèle comporte des cycles. Par conséquent l'espace des chronomètres associé à un sommet est égale à l'union des espaces d'horloges de toutes les visites possibles à ce sommet. Les opérateurs d'analyse en avant (ou *Succ* opérateurs) cherchent tous les états atteignables de cet automate à partir de son état initial, en restant dans les sommets tandis que le temps s'écoule ou en franchissant ses transitions. Notons par E l'espace temporel atteignable de l'automate $\mathbb{A}$, E_i l'espace des chronomètres associé au

sommet l_i, e_i l'espace des chronomètres à l'entrée du sommet l_i lors de la visite courante et E_i^a l'espace des chronomètres à l'entrée du sommet l_i pour toutes les visites possibles. L'espace E_i est construit récursivement de la manière suivante. Lorsqu'on atteint le sommet l_i avec un espace des chronomètres e_i, nous vérifions si cet espace est inclus dans l'espace des chronomètres E_i. Si $e_i \subseteq E_i$, nous abandonnons le calcul des espaces des chronomètres pour les successeurs du sommet l_i. Si $e_i \nsubseteq E_i$, nous calculons l'espace d'horloges associé au séjour dans le sommet l_i lors de la visite courante. Cet espace, noté e'_i, est obtenu en calculant le successeur continu de e_i : $(l_i, e'_i) = Succ_t(l_i, e_i)$. Ensuite, nous mettons à jour l'espace des chronomètres E_i en ajoutant l'espace des chronomètres e'_i associé à cette dernière visite : $E_i = E_i \cup e'_i$. Enfin, nous calculons l'espace des chronomètres à l'entrée de chaque sommet successeur de l_i. Si l_j désigne le sommet successeur de l_i suite au franchissement la transition a_{ij}, l'espace des chronomètres e_j à l'entrée du sommet l_j sera : $(l_j, e_j) = Succ_d(l_i, e'_i)$.

Nous présentons dans la suite la formalisation de l'algorithme calculant les états atteignable de l'automate $\mathbb{A}$.

Input : L'automate $\mathbb{A}$ ayant l'état initial $s_0 = (l_0, X := 0)$

Output : L'ensemble des états atteignables de $\mathbb{A}$ à partir de s_0 (les sommets atteignables L et l'espace temporel atteignable dans chaque sommet).

$Waiting \leftarrow Succ_t(s_0)$
$L \leftarrow l_0$
$Reach \leftarrow Succ_t(s_0)$
Tant que $Waiting \neq \phi$ **do**
    $s = pop(Waiting)$
    $S' = Succ_d(s)$ */* S' l'ensemble des successeurs discrets de toutes les transitions franchissables de s
    $L \leftarrow L \cup New\ reach\ locations(Succ_d(s))$
        Tant que $S' \neq \phi$
        $(l_j, e_j) = pop(S')$
            si $e_j \nsubseteq E_j$:
            $(l_j, e'_j) = Succ_t(l_j, e_j)$
            $E_j = E_j \cup e'_j$
            $Waiting \leftarrow Waiting \cup \{(l_j, e'_j)\}$
            $Reach \leftarrow Reach \cup \{(l_j, e'_j)\}$
            fin si
        fin Tant que
fin Tant que

Waiting est la liste des états à analyser et pour lesquels il faut calculer les successeurs discrets. La fonction *pop* choisit un élément de *Waiting* (ou de l'ensemble S'). La fonction *New reach locations* donne les nouveaux sommets atteignables lors de calcul du successeur discret.

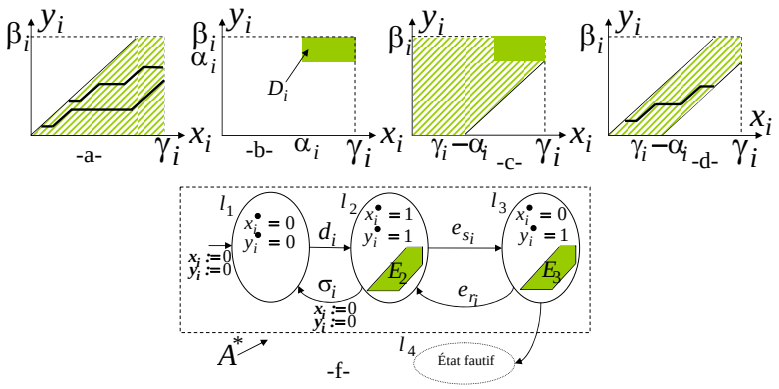

FIG. 4.9 – L'espace temporel dans l_2 et l_3 du modèle de $Tâche_i$ $\mathbb{A}_i$ présenté dans la figure 4.8.a : a- atteignable par l'analyse en avant, b- désiré, c- atteignable par l'analyse en arrière d- délimitation du comportement acceptable de $Tâche_i$ f- L'automate de surveillance de $Tâche_i$

Nous indiquons dans la suite par $E = Succ_{\mathbb{A}}(l_0, X := 0)$ l'espace temporel exprimant l'évolution possible de $\mathbb{A}$ et calculé par l'analyse en avant de $\mathbb{A}$ à partir de son état initial $s_0 = (l_0, X := 0)$.

Le logiciel de vérification formelle PHAVer [Frehse, 2005], [Asarin et al., 2006] que nous avons utilisé, fournit des commandes afin de calculer, pour un automate donné, l'espace temporel atteignable en utilisant la technique d'analyse en avant. L'implémentation de l'algorithme d'analyse en avant en PHAVer [Frehse, 2005], implique d'utiliser quelques fonctions après le calcul du successeur discret d'un élément de *Waiting*. Ces fonctions calculent la différence entre l'ensemble *Reach* et les nouveaux états représentés par s' (afin de décider si $e_j \nsubseteq E_j$) et les nouveaux sommets atteignables.
L'analyse d'atteignibilté dans l'automate à chronomètres n'est pas décidable [Cassez and Larsen, 2000]. Si l'analyse d'atteignabilité ne converge pas, nous proposons d'utiliser les techniques du logiciel de vérification formelle PHAVer afin de forcer la terminaison. Ces techniques sont détaillées en [Frehse, 2005] et [Asarin et al., 2006]. Dans cet outil, si les invariants de l'automate sont bornés, la terminaison se fait en utilisation des techniques de simplifications d'un polyèdre complexe (par exemple, limiter le nombre de bits utilisé pour représenter les contraintes d'un polyèdre et limiter le nombre des contraintes décrivant un polyèdre convexe [Frehse, 2005]). En conséquence, il n'y a qu'un nombre fini de contraintes possibles pour définir un polyèdre quelconque, et donc il n'y a qu'un nombre fini des résultats pour l'algorithme de l'atteignabilité. Il s'ensuit qu'on calcule une surapproximation conservative de l'espace d'état exact. Nous allons voir dans la suite l'effet des états ajoutés par la surapproximation sur le système de surveillance proposé (Si les opérateurs de simplification sont utilisés).
L'espace temporel exact obtenu en utilisant la technique d'analyse en avant dans

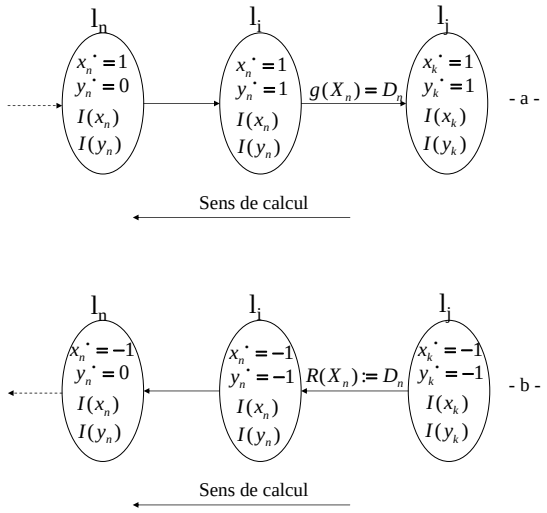

FIG. 4.10 – Principe de calcul

les sommets l_2 et l_3 de l'automate $\mathbb{A}_i$ de $Tâche_i$ (Fig. 4.8.a) est présenté dans la figure 4.9.a. Remarquez que les trajectoires spécifiées dans la propriété 1 représentent seulement une partie des trajectoires contenues dans l'espace temporel atteignable de $\mathbb{A}_i$.

2– *L'analyse en arrière*

Cette étape permet de synthétiser l'automate $\mathbb{A}^*$ mentionné dans le problème 1, à partir de l'automate $\mathbb{A}$. Nous indiquons dans la suite par E^* l'espace temporel exprimant l'évolution temporelle dans chaque sommet de $\mathbb{A}^*$. Le calcul de l'automate $\mathbb{A}^*$ et son espace E^*, nécessite l'analyse en arrière de $\mathbb{A}$ et l'utilisation des opérateurs d'analyse en arrière (ou Pre opérateurs). Il est clair que l'espace temporel E^* peut être obtenu de l'espace temporel E de $\mathbb{A}$, en éliminant les états à partir desquels les évolutions ne permettent pas d'atteindre l'espace désiré de chaque tâche interruptible. Autrement dit : on a besoin d'abord d'appliquer les opérateurs d'analyse en arrière aux gardes correspondant aux espaces désirés des tâches représentées dans l'automate $\mathbb{A}$. Ensuite, $E^*=E \cap (\bigcup Pre(D_i))$, $i \in \{1, ..., n\}$ où n le nombre des tâches interruptibles représentées dans $\mathbb{A}$. L'intuition d'appliquer les opérateurs d'analyse en arrière à une garde représentant l'espace désiré d'une tâche interruptible dans $\mathbb{A}$, est qu'on cherche tous les états qui conduisent à cet espace à partir de l'état initial de $\mathbb{A}$.

Nous expliquons le principe de cette étape d'analyse en considérant le schéma illustré dans la figure 4.10.a et représentant une partie de l'automate $\mathbb{A}$. L'objectif de

cette analyse est de déterminer l'espace E_i^* mémorisant les valeurs des chronomètres qui permettent seulement une évolution vers le sommet l_j. La transition $l_i \rightarrow l_j$ est franchissable si sa garde $g_{i,j} = g(X_n) = D_n$ est vérifiée. Cette garde implique les chronomètres $X_n = \{x_n, y_n\}$ surveillant la tâche ayant l'espace désiré D_n. Soit E_i l'espace temporel atteignable dans le sommet l_i, résultant de l'analyse en avant de $\mathbb{A}$.

• ***Analyse en arrière de l'automate direct :***

La démarche que nous proposons pour atteindre cet objectif consiste à effectuer les opérations suivantes :

1. Calculer les valeurs à partir desquelles on peut atteindre l'espace D_n en laissant le temps évoluer, pendant le séjour dans le sommet l_i. Ces valeurs sont contenues dans le prédécesseur continu de l'espace D_n, noté $Pre_t(l_i, D_n)$.
2. Calculer l'espace des horloges possible dans le sommet l_i, qui permet d'atteindre l'espace D_n en laissant le temps s'écouler pendant le séjour dans le sommet l_i. Cet espace est décrit par l'intersection de l'espace des valeurs possibles dans l_i, noté E_i et de l'espace calculé dans l'étape précédente : $E_i^* = E_i \cap Pre_t(l_i, D_n)$.
3. Déterminer les valeurs des chronomètres à l'entrée de l_i, avec lesquelles on peut atteindre le sommet l_i pour que l'évolution vers l'espace D_n soit possible. Ce sont contenues dans l'espace $e_i^* = E_i^* \cap E_i^a$; où E_i^a est l'espace à l'entrée de l_i résultant de l'analyse en avant de $\mathbb{A}$.
4. Si $E_i^a = e_i^*$, alors toutes les valeurs des horloges avec lesquelles on peut atteindre le sommet l_i par le franchissement de la transition $a_{n,i} : l_n \rightarrow l_i$ appartient à e_i^*. Dans ce cas, il n'est pas nécessaire de remonter cette transition. Par contre, si $E_i^a \neq e_i^*$, alors il y a des valeurs à l'entrée de l_i par le franchissement de $a_{n,i}$ qui peuvent engendrer une évolution non-désirée. Dans ce cas, il faut remonter cette transition $a_{n,i}$. On calcule l'ensemble des valeurs des chronomètres dans le sommet l_n qui vérifient la garde de la transition $a_{n,i}$ et à partir desquelles on atteint l'espace calculé dans l'étape 3 (c-à-d : e_i^*) par le franchissement de transition $a_{n,i}$. Cet espace, noté $S_{n,i}$, est le prédécesseur discret de l'espace e_i^* par le franchissement de $a_{n,i} : S_{n,i} = Pre_d(l_i, e_i^*)$.
5. L'espace des chronomètres dans l_n est formé par deux types de valeurs : celles qui appartiennent à l'espace $S_{n,i}$ et celles à partir desquelles on atteint l'espace $S_{n,i}$ en laissant le temps s'écouler. Cet espace est le prédécesseur continu de l'espace $S_{n,i}$, noté $Pre_t(S_{n,i})$. Par conséquent, on a : $E_n^* = E_n \cap Pre_t(S_{n,i})$. L'algorithme répète ensuite la démarche présentée dans les étapes 3 et 4, pour vérifier si l'on continue l'analyse en arrière ou non.

Soit $L_{init} = \{l_i, ..., l_j\}$ l'ensemble des sommets dans $\mathbb{A}$ où $\{\forall l_i \in L_{init} : \exists a_{i,x}$ tel que $g_{i,x} = D_i$ où $D_i \in \mathbb{D}\}$ et $a_{i,x}$ représente une transition sortante de l_i ayant une garde $g_{i,x}$.

On mémorise dans une pile P les sommets de l'ensemble L_{init} avec les espaces désirés à ses sorties. Soit (l_j, D_i) un élément de P, cet élément signifie que l'espace désiré à la sortie de l_j est D_i. Le calcul de l'espace E^* de $\mathbb{A}$ exige d'appliquer la procédure mentionnée ci-dessus pour tous les éléments dans la pile P. Ceci est afin de synthétiser l'espace représentant l'exécution acceptable de chaque tâche interruptible du système.

□

• ***Analyse en avant de l'automate inversé :***

Comme nous avons illustré dans le chapitre 2, il est possible de définir pour l'automate à chronomètres $\mathbb{A}$ un automate inversé $\mathbb{A}^{-1}$ tel que l'analyse en avant de $\mathbb{A}^{-1}$ est équivalent à l'analyse en arrière de $\mathbb{A}$ [Henzinger et al., 1998].
Pour s'en convaincre, nous allons présenter dans la suite l'analyse en avant de l'automate inversé $\mathbb{A}^{-1}$ présenté dans la figure 4.10.b. Cet automate correspond à l'inverse de l'automate présenté dans la figure 4.10.a. Nous gardons le même objectif pour lequel l'analyse en arrière a été appliqué sur l'automate présenté dans la figure 4.10.a.
Il est important de remarquer que la construction de l'automate $\mathbb{A}^{-1}$ consiste à inverser la causalité de l'automate et les dynamiques des chronomètres (la dynamique 1 est remplacée par la valeur -1). L'inversion permute la garde et la mise à zéro d'une transition. Les autres composants restent inchangés.
La démarche d'analyse sur $\mathbb{A}^{-1}$ que nous proposons consiste à effectuer les opérations suivantes :

1. Calculer l'évolution temporelle possible dans l_i à partir de l'espace à son entrée D_n. Autrement dit : les valeurs possibles des chronomètres pendant le séjour de l'automate dans l_i. Ces valeurs sont contenues dans le successeur contenues de D_n : $(l_i, E_i^{-1}) = Succ_t(l_i, D_n)$. Remarquez que l'espace temporel dans le sommet l_i résultant de cette étape d'analyse est équivalente à celui résultant de l'étape 1 de la procédure d'analyse en arrière de l'autmate $\mathbb{A}$.
2. Calculer l'espace E_i^* : $E_i^* = E_i \cap E_i^{-1}$.
3. Déterminer si la transition $a_{i,n} : l_i \rightarrow l_n$ est franchissable. Cette transition est franchissable si $E_i^* \cap g_{i,n}^{-1}$ est un polyèdre non vide. Ce polyèdre contient les valeurs pour lesquelles la transition $a_{i,n}$ est franchissable. $g_{i,n}^{-1}$ est la garde de la transition $a_{i,n}$ dans l'automate inversé. L'espace résultant de l'intersection est équivalent à l'espace à l'entrée de l_i (c-à-d : e_i^*) dans l'automate direct. Ceci est dû au fait : $g_{i,n}^{-1}$ $= R_{n,i}$ où $R_{n,i}$ l'affectation de la transition correspondante dans l'automate direct.
4. Déterminer les valeurs des chronomètres à l'entrée de l_n. Ces valeurs sont contenues dans la successeur discret de E_i^* : $(l_n, e_n^{-1}) = Succ_d(l_i, E_i^*)$. L'espace temporel dans le sommet l_n résultant de cette étape est équivalent à celui résultant de l'étape 4 de la procédure d'analyse en arrière de l'autmate $\mathbb{A}$, et contenant les valeurs dans

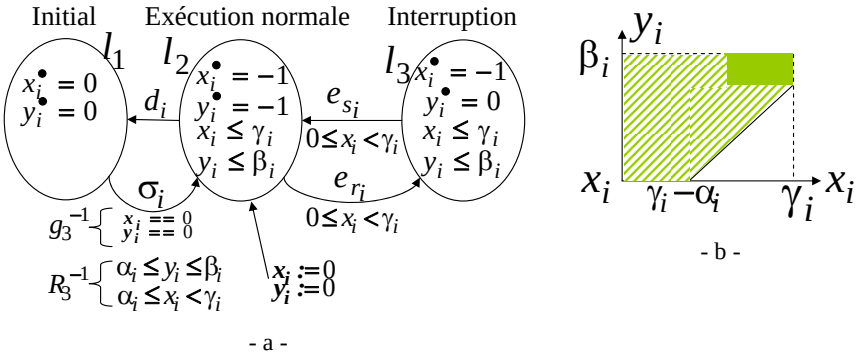

FIG. 4.11 – a- L'automate inversé $\mathbb{A}_i$ b- E_2^{-1} et E_3^{-1}.

l_n pour lesquelles le franchissement de la transition $a_{n,i}$ conduit à l'espace D_n.

5. Calculer les évolutions temporelles possibles pendant le séjour dans l_n. Elles sont contenues dans le successeur continu de e_n^{-1} : $(l_n, E_n^{-1}) = Succ_t(l_n, e_n^{-1})$. Cet espace est équivalent à celui résultant de l'étape 5 de la procédure d'analyse en arrière de l'automate $\mathbb{A}$. En conséquent, nous trouvons que : $E_n^* = E_n \cap E_n^{-1}$.

Nous proposons dans notre démarche pour calculer l'espace E^*, d'utiliser l'analyse en avant de l'automate inversé $\mathbb{A}^{-1}$. L'analyse en arrière de l'automate $\mathbb{A}$ pour un espace donné et une transition $\pi_i \in \Sigma \cup \mathbb{R}_{\geq 0}$, est équivalente à l'analyse en avant de l'automate $\mathbb{A}^{-1}$ pour le même espace et la même transition π_i [Henzinger et al., 1998]. □

L'état initial de l'automate inversé $\mathbb{A}^{-1}$, depuis lequel l'algorithme d'analyse en avant sera appliqué, est défini par la propriété suivante :

Propriété 3. L'état initial de l'automate $\mathbb{A}^{-1}$ est un état tel que $(l_i \times [X := 0])$ où $l_i \in L_{init}$. □

La figure 4.11.a présente l'automate inversé de $\mathbb{A}_i$ représentant la tâche interruptible $Tâche_i$ (Fig.4.8.a). En appliquant la propriété 3, l'état initial de l'automate inversé de $\mathbb{A}_i$ est $(l_2, (0, 0))$. L'analyse en avant de l'automate inversé à partir de cet état initial a pour résultat l'espace temporel E^{-1} (Fig. 4.11.b). L'intersection de cet espace avec celui résultant de l'analyse en avant de l'automate $\mathbb{A}_i$ d'une tâche est présentée dans la figure 4.9.d. Ceci est l'espace temporel caractérisant l'exécution acceptable de $Tâche_i$. Les trajectoires contenues dans cet espace (Fig. 4.9.d) montrent que la tâche atteint l'état fautif seulement du sommet l_3 ayant la dynamique $\dot{x}_i = 1$ and $\dot{y}_i = 0$. La figure 4.9.f présente l'automate de surveillance d'une tâche interruptible.

La procédure de calcul de $\mathbb{A}^*$ est la suivante :

- Analyser en avant l'automate $\mathbb{A}$: $E = Succ_{\mathbb{A}}(l_0, X := 0)$,
- Construire l'automate inversé de $\mathbb{A}$: $\mathbb{A}^{-1} = reverse(\mathbb{A})$,
- Définir l'état initial de $\mathbb{A}^{-1}$: $s'_0 = (l'_0, X := 0)$ où $l'_0 = l_i \in L_{init}$
- Analyser en avant l'automate $\mathbb{A}^{-1}$: $E^{-1} = Succ_{\mathbb{A}^{-1}}(s'_0)$,
- $E^* = E \cap E^{-1}$

Théorème 4.3.1. *L'automate $\mathbb{A}^*$ ayant l'espace temporel $E^* = E \cap E^{-1}$ contient l'ensemble des trajectoires qui représentent la solution du problème 1.* □

Démonstration. L'espace E^* résulte de l'intersection de l'espace E représentant toutes les trajectoires possibles et l'espace E^{-1}. Tout d'abord, nous allons prouver que l'espace E^{-1} contient les trajectoires représentées par la propriété 2. Ensuite, nous allons prouver que l'espace E^{-1} ne contient aucune trajectoire conduisant à l'état fautif. Nous allons voir également qu'aucune trajectoire éliminée par l'intersection ne représente une exécution acceptable d'une tâche du système. Par conséquent, $\mathbb{A}^*$ contient l'ensemble des trajectoires représentant l'exécution acceptable des tâches du système.

- L'espace E^{-1} contient les trajectoires représentées dans la propriété 2.

Les travaux présentés dans [Henzinger et al., 1998], ont montré que l'analyse en arrière d'une région dans un automate, pour une transition $\pi \in \Sigma \cup R_{\geq 0}$ est équivalente à l'analyse en avant de cette région dans l'automate inversé. Alors, on a :

$$\forall D_i \in \mathbb{D} : \text{ Pour chaque } D_i \subset E : Pre^{\pi}_{\mathbb{A}}(D_i) = Succ^{\pi}_{\mathbb{A}^{-1}}(D_i)$$

Donc grâce à cette propriété de l'automate $\mathbb{A}^{-1}$, nous exécutons progressivement l'analyse en arrière de tous les éléments de D par l'analyse en avant de l'automate $\mathbb{A}^{-1}$. Par conséquent, $E^* = E \cap E^{-1}$ contient les trajectoires représentées par la propriété 2.

- L'espace E^{-1} ne contient aucune trajectoire conduisant à l'état fautif.

Dans l'automate $\mathbb{A}^{-1}$, supposons qu'il y a une transition $l_r \rightarrow l_i$ où l_r le sommet correspondant à l'interruption de $Tâche_i$, c-à-d : les dynamiques des chronomètres dans l_r sont $\dot{x}_i = -1$ et $\dot{y}_i = 0$. Les valeurs des chronomètres X_i dans e_i^{-1} suite au franchissement de la transition $l_r \rightarrow l_i$ sont les suivantes : $e_i^{-1} = (x_i - y_i < \gamma_i - \alpha_i \wedge x_i < \gamma_i \wedge y_i \leq \beta_i)$. L'espace obtenu par l'analyse en avant de ces valeurs au sommet l_i ayant la dynamique $\dot{x}_i = \dot{y}_i = -1$, ne permet pas d'atteindre l'état fautif $F_i = (x_i = \gamma_i \wedge y_i < \alpha_i)$. Autrement dit : dans l'espace E_i^{-1}, on a $(L \times X) \cap F_i = \emptyset$. De même façon,

on trouve que dans l'espace E_r^{-1}, on a $(L \times X) \cap F_i = \emptyset$. Donc, E^{-1} contient l'espace temporel resultant de l'analyse en arrière de $\mathbb{D}$ et ne contient aucune trajectoire pouvant atteindre les états fautifs $\mathbb{F}$ où $\mathbb{F} = \{F_1, F_2, ..., F_n\}$.

Nous allons voir également qu'aucune trajectoire éliminée par l'intersection ne représente une exécution acceptable d'une tâche du système. Afin de prouver ce point, supposons que les valeurs à l'entrée de l_i ou l_r n'appartiennent pas à l'espace e_i^{-1}, c-à-d : elles appartiennent à l'espace $(x_i - y_i \geq \gamma_i - \alpha_i \wedge x_i < \gamma_i \wedge y_i \leq \beta_i)$. L'analyse en avant de ces valeurs au sommet l_i ou l_r permet d'atteindre l'état fautif $F_i = (x_i = \gamma_i \wedge y_i < \alpha_i)$. Par conséquent, l'analyse en avant de cet espace ne conduit pas à l'espace désiré D_i car nous avons $D_i \times F_i = \emptyset$. Autrement dit : on a, dans ce cas : $(L \times X) \cap D_i = \emptyset$. Donc, aucune trajectoire éliminée ne conduit à un espace désiré. □

B.3 Terminaison de l'algorithme d'analyse d'atteignabilité L'analyse d'atteignibilté dans l'automate à chronomètres n'est pas décidable [Cassez and Larsen, 2000], [Henzinger et al., 1998]. Si l'analyse d'atteignabilité ne converge pas, nous proposons d'utiliser les techniques du logiciel de vérification formelle PHAVer afin de forcer la terminaison. Ces techniques sont détaillées en [Frehse, 2005] et [Asarin et al., 2006]. Dans cet outil, si les invariants de l'automate sont bornés, la terminaison se fait en utilisation des techniques de simplifications d'un polyèdre complexe (par exemple, limiter le nombre de bits utilisés pour représenter les contraintes d'un polyèdre et limiter le nombre des contraintes décrivant un polyèdre convexe [Frehse, 2005]). En conséquence, il n'y a qu'un nombre fini de contraintes possibles pour définir un polyèdre quelconque, et donc il n'y a qu'un nombre fini des résultats pour l'algorithme de l'atteignabilité. Il s'ensuit qu'on calcule une surapproximation conservative de l'espace d'état exact. La limitation de la complexité d'un polyèdre dans PHAVer est paraméteré, donc nous pouvons réduire la surapproximation en utilisant des valeurs appropriées des ces paramètres. Cependant, PHAVer a été capable de calculer l'espace exact pour tous les exemples considérés.

Discussion 1. – La surapproximation n'est pas gênante pour le système de surveillance proposé.

Lorsqu'on calcule l'espace temporel de $T\hat{a}che_i$ présenté dans la figure 4.8.a, l'algorithme d'analyse d'atteignabilité converge sans utiliser les opérateurs de forçage de la terminaison. Supposons que ceci n'est pas le cas et utilisons ces opérateurs. Nous considérons cette hypothèse afin d'étudier l'effet des états ajoutés par la surapproximation sur le système de surveillance. Dans ce cas, des états supplémentaires seront ajoutés à l'espace d'état exact. La prédiction des états ajoutés est difficile. Dans les figures 4.12.a,b, les états S_1 et S_3 sont ajoutés afin de montrer une surapproximation possible de l'espace temporel exact. Ces états ajoutés à l'espace exact résultent de l'analyse en avant de $\mathbb{A}$ et $\mathbb{A}^{-1}$. L'estimation

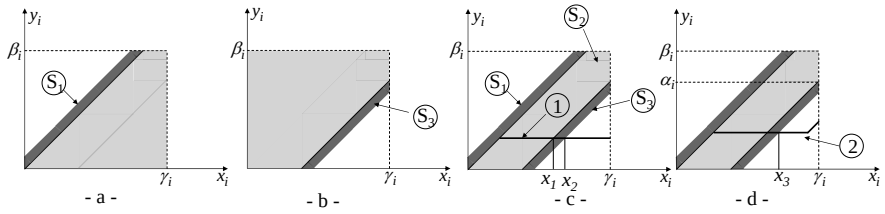

FIG. 4.12 – L'espace surapproximé de l'automate d'une tâche interruptible : a - par l'analyse en avant de $\mathbb{A}$. b- par l'analyse en avant de $\mathbb{A}^{-1}$. c, d- l'effet de la surapproximation sur le système de surveillance.

de S_1 et S_3 que nous avons faite considère le fait que les espaces E_2 et E_2^{-1} (respectivement E_3 et E_3^{-1}) ne peuvent pas dépasser le polyhedron délimité par l'invariant de l_2 et l_3. Par ailleurs, nous pouvons réduire la surapproximation en utilisant des valeurs appropriées des paramètres qui limitent la complexité d'un polyèdre dans PHAVer (qui forcent la terminaison).
L'espace temporel présenté dans la figure 4.12.c résulte de l'intersection des espaces présentés dans les figures 4.12.a,b. Cet espace temporel sera utilisé afin de surveiller $Tâche_i$. Dans les figure 4.12.c,d, nous montrons l'effet des états ajoutés par la surappoximation, sur le système de surveillance. Les états indiqués par S_1 ne sont pas effectivement atteignables tant que les chronomètres x_i et y_i sont initialisés au début de la tâche. La trajectoire 1 (Fig. 4.12.c), représente l'interruption de $Tâche_i$ à cause d'un défaut permanent. L'espace surapproximé détecte ce défaut à l'instant x_2 tandis que l'espace exact le détecte à l'instant x_1. On peux remarquer que cet instant est plus petit que γ_i auquel les méthodes existantes dans la littérature, détectent ce défaut. De même, le défaut représenté par la trajectoire 2 (Fig. 4.12.d) est détecté à l'instant $x_3 < \gamma_i$.

– La robustesse du système de surveillance contre le retard dû aux conditions d'exploitation des ressources exécutant une tâche interruptible.

La conception de notre méthode de surveillance est basée sur les hypothèses suivantes :
• le système est sujet seulement aux défauts interruptibles,
• la vitesse d'exécution d'une tâche est constante où la durée $I_{nor} = \alpha_i$ représente le temps nécessaire d'exécution la tâche à cette vitesse.
En gardant la première hypothèse, nous avons introduit la durée β_i dans l'intervalle I_{nor}. Cette durée représente une marge ajoutée à l'intervalle $I_{nor} = \alpha_i$. Le but est de tenir compte des retards qui peuvent être dûs aux conditions d'exploitation des ressources exécutant $Tâche_i$ (la surcharge ou le vieillissement des ressources, par exemple), pendant le séjour du système dans le sommet d'exécution normale. La valeur de β_i dépend de la tâche surveillée et est donnée par le concepteur du

système : $\alpha_i \leq \beta_i < \gamma_i$.
En ce qui concerne la durée γ_i, cette dernière considère le retard dû aux défauts interruptibles survenant aux ressources exécutant $Tâche_i$. Cette durée est aussi spécifiée par le concepteur ou l'ordonnanceur du système.
En bref, β_i considère le retard dûs aux conditions d'exploitation et γ_i tient compte le retard résultant d'occurrence des défauts interruptibles et intermittents.

– Prise en compte d'autres types des défauts.

Comme nous l'avons indiqué ci-dessus, l'étude présentée au cours de ce chapitre a été restreinte aux défauts interruptibles : intermittents et permanents, expliqués dans le chapitre 2. En effet, un système temps réel peut être susceptible à un autre type de défauts. Ces derniers changent la dynamique d'exécution de la tâche sans l'interrompre. Ils font ralentir ou accélérer l'exécution de la tâche. Nous avons remarqué dans le chapitre précédent que, les méthodes existantes détectent le défaut qui fait ralentir l'exécution à l'instant d'expiration la date au plus tard prévue d'exécution de la tâche.
Supposons qu'on veuille tenir compte de ce défaut pour que le système de surveillance devienne sensible à ce défaut. Autrement dit : nous voulons considérer les systèmes sur lesquels deux types de défauts peuvent survenir : les défauts interruptibles et les défauts ralentissant l'exécution des tâches du système. Considérons le cas où une tâche interruptible a été sujette à un défaut qui fait ralentir la tâche sans l'interrompre. Comment se comporte le système de surveillance ? L'arrêt est un cas de ralentissement particulier pour lequel on dispose d'un capteur qui le détecte. Or, il n'y a pas de capteurs pour le ralentissement. Par conséquent, l'automate de surveillance de $Tâche_i$ reste dans le sommet l_2. Supposons que, à l'instant β_i, l'événement σ_i, représentant la fin de tâche, n'apparaît pas, alors l'automate de surveillance de $Tâche_i$ détecte un défaut. Cela est dû à la violation de l'espace temporel E_2^*. Ce cas de fonctionnement est décrit par la trajectoire 1 dans la figure 4.13.a. Nous trouvons qu'il n'est pas normal qu'en présence d'un défaut de ralentissement la tâche est plus exigeante en terme de durée d'exécution qu'en présence d'un défaut d'interruption. Pour cela, il faut modifier le modèle d'une tâche interruptible tel que donné par la figure 4.13.b. Dans ce modèle, l'invariant, dans les sommets l_2 et l_3, devient $x_i \leq \gamma_i$ et $y_i \leq \gamma_i$. La garde g_3 de la transition $l_2 \xrightarrow{g_3} l_1$ devient $g_3 = (\alpha_i \leq x_i < \gamma_i \wedge \alpha_i \leq y_i \leq \gamma_i)$. La propriété représentant l'exécution acceptable de $Tâche_i$ devient :

$$\forall Tâche_i \in Tâche_{int}, \forall x_i, y_i \in X_i : x_i \in I_{acc} \wedge y_i \in I_{acc}$$

Remarquons que la propriété mentionnée ci-dessus considère le fait que les tolérances aux deux types des défauts sont identiques. Cela nous paraît logique car nous voulons que la tâche $Tâche_i$ soit exécutée avant de dépasser γ_i pour n'im-

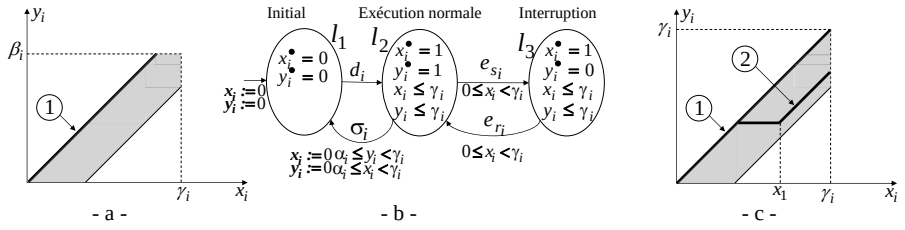

FIG. 4.13 – La prise en compte des défauts ralentissant la tâche : a- l'espace représentant le comportment acceptable dans le sommet l_2 de l'automate $\mathbb{A}_i^*$ b- le modèle de la tâche après la modifiaction c- l'espace représentant le comportement acceptable dans les sommets l_2 et l_3 da l'automate présenté dans la figure b.

porte quel défaut survenant.
De l'application de la procédure de synthèse à l'automate présenté dans la figure 4.13.b, résulte l'espace temporel présenté dans la figure 4.13.c. Cet espace est l'espace temporel dans les sommets l_2 et l_3, représentant l'exécution acceptable.
Supposons que la tâche subit un défaut qui fait ralentir l'exécution. L'automate reste dans le sommet représentant l'exécution l_2. L'espace temporel dans ce sommet détecte ce défaut à l'instant $x_i = \gamma_i$, la date au plus tard prévue d'apparition σ_i. Ce comportement est représenté par la trajectoire 1 dans la figure 4.13.c. La trajectoire 2 représente le fonctionnement suivant. La tâche a été sujette à un défaut interruptible. Ce dernier a disparu à l'instant x_1. Ensuite, la tâche a subi un autre défaut qui l'a fait ralentir et l'a empêché d'être exécutée dans sa durée acceptable. L'espace temporel détecte ce défaut à l'instant $x_i = \gamma_i$.
On peut conclure que le système de surveillance est sensible aux défauts qui font ralentir ces tâches.

4.4 Exemple illustratif : la surveillance d'un atelier manufacturier

4.4.1 Spécification du système

Pour illustrer les idées que nous avons développées dans ce chapitre, nous allons considérer l'exemple suivant. Un système manufacturier (Fig. 4.14.a) se compose d'un poste de travail, d'un robot et d'une station d'assemblage. Le poste de travail est constitué d'un poussoir pour charger les palettes, d'un convoyeur et d'un capteur de présence de fin de convoyeur.
La figure 4.14.b présente le contôleur de ce système manufacturier. Dans ce système, lorsque l'événement d se produit, le poussoir met une palette sur le convoyeur. Cet évé-

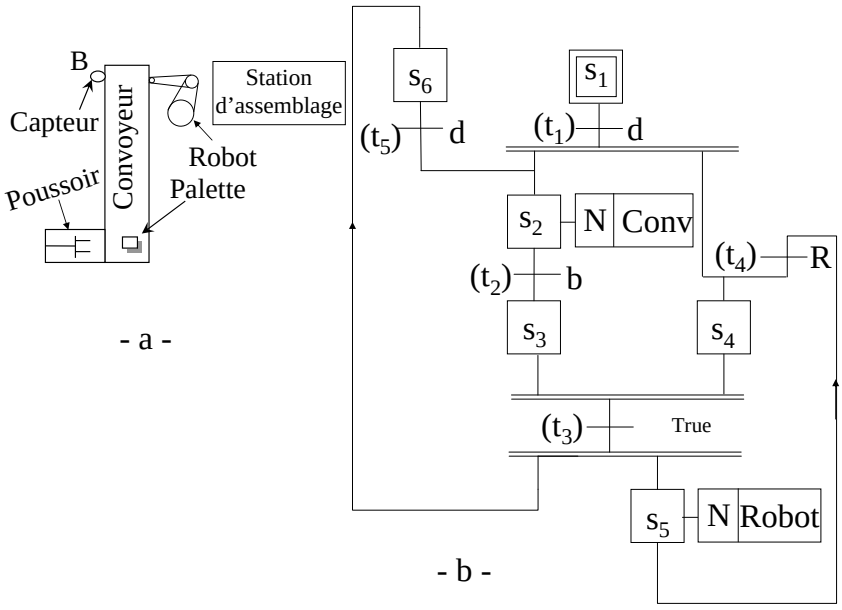

FIG. 4.14 – a- Un système manufacturier simple b- Le programme SFC (Sequential Function Chart) représentant la commande du système considéré.

nement d est généré automatiquement dans l'intervalle $[0, 2]$ $u.t$ après l'activation de l'étape s_1 ou s_6. Cet événement peut être généré aussi manuellement par l'opérateur 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. L'action mettant le convoyeur en marche a lieu dès que et tant que l'étape s_2 est active. Lorsque la palette atteint le point B, le capteur produit l'événement b (à cet instant l'étape s_3 est active). Si le robot n'est pas occupé (l'étape s_4 est active), la transition t_3 est franchie immédiatement et le robot commence à transférer la palette au poste d'assemblage. L'action mettant le robot en marche a lieu dès que et tant que l'étape s_5 est active. Lorsque le robot atteint la station d'assemblage, il met la palette dans la station et redevient à nouveau disponible pour transférer d'autres palettes. Un capteur produit l'événement R, représentant la fin de la tâche du robot lorsqu'il dépose la palette dans la station d'assemblage.

On peut remarquer que le système exécute deux tâches : $Tâche_1$ exécutée par le convoyeur et le poussoir et $Tâche_2$ exécutée par le robot.

Les informations concernant ces tâches sont présentées dans le tableau 4.1. Dans ce tableau, les signaux s_c et r_c représentent les sorties d'un capteur logique indiquant la dynamique d'exécution de la tâche du convoyeur. De même, les signaux s_R et r_R représentent les sorties d'un capteur logique implémenté sur le robot et indiquant la dynamique d'exécution de la tâche du robot.

La tâche	$Tâche_1$	$Tâche_2$
La durée d'exécution exacte $[\alpha_i, \beta_i]$ (u.t)	[3,4]	[2,3]
La durée d'exécution tolérée $[\alpha_i, \gamma_i)$ (u.t)	[3,5)	[2,4)
L'événement représentant le début	d	e
L'événement représentant la fin	b	R
L'événement représentant l'interruption	s_c	s_R
L'événement représentant la reprise	r_c	r_R

TAB. 4.1 – Les informations nécessaires pour construire le système de surveillance.

4.4.2 Construction du système de surveillance

Nous présentons dans la figure 4.15.a,b les automates à chronomètres $\mathbb{A}_1$ et $\mathbb{A}_2$ correspondant aux $tâche_1$ et $tâche_2$. L'événement e est l'événement toujours occurant. Dans l'automate $\mathbb{A}$ résultant de la composition de ces deux automate, l'événement e correspond à la validation de la transition t_3 dans la figure 4.14.b. Ceci correspond la situation suivante : le robot est disponible ($\mathbb{A}_2$ est dans le sommet l''_1) et il y a une palette au point B sur le convoyeur ($\mathbb{A}_1$ est dans le sommet l'_4). La composition synchrone de ces automates donne l'automate présenté dans la figure 4.16. À titre de simplification, les gardes et les invariants sont omis. Cet automate est composé de 12 sommets.

L'application de la procédure de synthèse proposée et présentée dans ce chapitre, donne les résultats présentés dans le Tableau 4.2. Ce tableau montre l'espace temporel dans chaque sommet de l'automate, resultant de l'analyse en avant et en arrière de l'automate. La figure 4.16 montre l'espace temporel, dans quelques sommets résultant de l'analyse en avant de l'automate. Nous avons exécuté l'analyse en arrière de l'automate (comme nous l'avons expliqué ci-dessus) en faisant l'analyse en avant de son automate inversé. L'état initial choisi de l'automate inversé est $(l_7, (x_1 = x_2 = x_3 = x_4 = y_2 = y_4 = 0))$. Les espaces temporels résultant de l'analyse en avant de $\mathbb{A}$ et $\mathbb{A}^{-1}$ sont donnés dans le tableau 4.2. Le calcul est fait en utilisant le logiciel de vérification formelle PHAVer.

L'automate $\mathbb{A}^*$, représentant l'automate délimitant le comportement acceptable du système, est présenté dans la figure 4.17. Dans cet automate, les inégalités algébriques délimitant les espaces temporels dans les sommets de l'automate $\mathbb{A}^*$ sont présentées dans le Tableau 4.2.

Sommet	Espace atteingable résultant de l'analyse en avant de $\mathbb{A}$	Espace atteingable résultant de l'analyse en avant de $\mathbb{A}^{-1}$	Espace résultant de l'intersection
l_1	$x_2 = x_4 = y_2 = y_4 == 0 \wedge 0 \leq x_1 \leq 2$	$x_4 = y_4 == 0 \wedge x_2 - y_2 < 2 \wedge 0 \leq x_1 \leq 2 \wedge 0 \leq x_2 < 5 \wedge 0 \leq y_2 \leq 4$	$x_2 = x_4 = y_2 = y_4 == 0 \wedge 0 \leq x_1 \leq 2$
l_2, l_3	$x_4 = y_4 == 0 \wedge 0 \leq x_2 - y_2 \wedge 0 \leq x_1 \leq 2 \wedge 0 \leq y_2 \leq 4 \wedge 0 \leq x_2 < 5$	$x_4 = y_4 == 0 \wedge x_2 - y_2 < 2 \wedge 0 \leq y_2 \leq 4 \wedge 0 \leq x_2 < 5$	$0 \leq x_2 - y_2 < 2 \wedge x_2 < 5 \wedge y_2 \leq 4 \wedge 0 \leq x_1 \leq 2$
l_4	$x_2 = x_4 = y_2 = y_4 == 0 \wedge 0 \leq x_1 \leq 2$	$x_4 = y_4 == 0 \wedge x_2 - y_2 < 2 \wedge 0 \leq x_2 < 5 \wedge 0 \leq y_2 \leq 4$	$x_2 = x_4 = y_2 = y_4 == 0 \wedge 0 \leq x_1 \leq 2$
l_5, l_6	$x_1 - x_4 == 0 \wedge x_2 = y_2 == 0 \wedge x_1 \leq 2 \wedge 0 \leq y_4 \wedge 0 \leq x_1 - y_4$	$0 \leq x_1 \leq 2 \wedge 0 \leq y_2 \leq 4 \wedge 0 \leq x_2 < 5 \wedge 0 \leq x_4 < 4 \wedge 0 \leq y_4 \leq 3 \wedge x_2 - y_2 < 2 \wedge x_4 - y_4 < 2$	$x_1 - x_4 == 0 \wedge x_2 = y_2 == 0 \wedge x_1 \leq 2 \wedge 0 \leq y_4 \wedge 0 \leq x_4 - y_4$
l_7, l_8, l_9, l_{11}	$x_1 + x_2 - x_4 == 0 \wedge 0 \leq x_2 - y_2 \wedge 0 \leq y_4 \leq 3 \wedge 0 \leq x_1 \leq 2 \wedge 0 \leq x_4 - y_4 \wedge x_4 < 4 \wedge y_2 \geq 0$	$0 \leq y_2 \leq 4 \wedge 0 \leq x_2 < 5 \wedge 0 \leq y_4 \leq 3 \wedge 0 \leq x_4 < 4 \wedge x_2 - y_2 < 2 \wedge x_4 - y_4 < 2$	$x_1 + x_2 - x_4 == 0 \wedge 0 \leq x_2 - y_2 < 2 \wedge 0 \leq x_4 - y_4 < 2 \wedge 0 \leq x_4 - x_2 \leq 2 \wedge y_2 \geq 0 \wedge x_4 < 4 \wedge 0 \leq y_4 \leq 3$
l_{10}, l_{12}	$x_2 = y_2 == 0 \wedge x_4 < 4 \wedge -x_1 + x_4 \geq 3 \wedge 0 \leq y_4 \leq 3 \wedge 0 \leq x_1$	$0 \leq y_2 \leq 4 \wedge 0 \leq y_4 \leq 3 \wedge 0 \leq x_2 < 5 \wedge 0 \leq x_4 < 4 \wedge x_4 - y_4 < 2 \wedge x_2 - y_2 < 2$	$x_2 = y_2 == 0 \wedge 0 \leq x_1 \wedge 0 \leq x_4 - y_4 < 2 \wedge 3 \leq x_4 - x_1 \wedge x_4 < 4 \wedge y_4 \leq 3$

TAB. 4.2 – L'espace temporel dans chaque sommet de l'automate $\mathbb{A}$ résultant de : l'analyse en avant de $\mathbb{A}$, l'analyse en avant de $\mathbb{A}^{-1}$ et l'intersection de ces espaces.

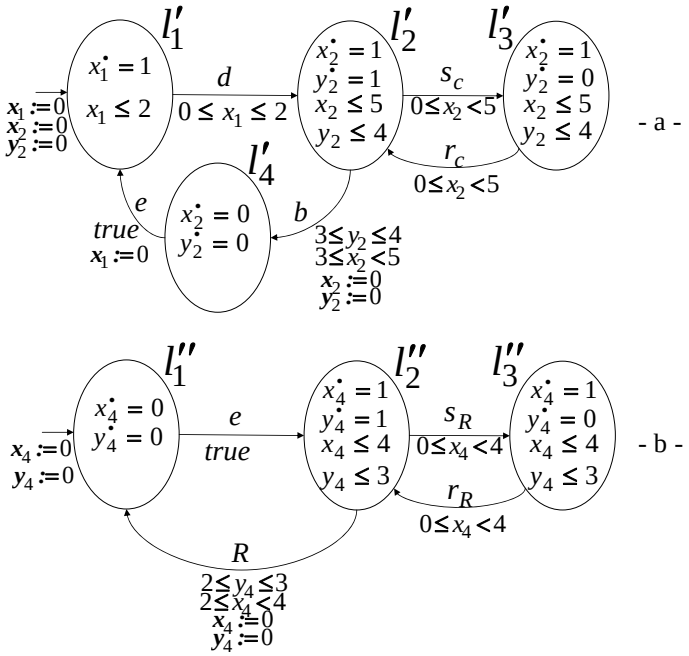

FIG. 4.15 – a- L'automate $\mathbb{A}_1$ représentant la tâche du convoyeur. b- L'automate $\mathbb{A}_2$ représentant la tâche du robot.

4.4.3 Implémentation et simulation de système de surveillance

L'automate $\mathbb{A}^*$ obtenu est synchronisé sur l'événement provenant du système commandé. Il est également temporisé sur les horloges internes. Un événement de l'automate correspond à un événement dans le système à surveiller. L'événement associé à une transition peut se produire tant que les valeurs des chronomètres vérifient l'espace temporel associé au sommet source. Ainsi, cet automate permettra de détecter un défaut suite à la violation de l'espace temporel associé à un sommet. Ceci peut être illustré par l'automate présenté dans la figure 4.17. Supposons que le système est dans la situation correspondante au sommet l_9. Lorsque l'événement b se produit alors que les valeurs des chronomètres x_2, x_4, y_2 et y_4 appartiennent à l'espace E_9^* et il aura lieu la commutation vers le sommet l_{12} : $l_9 \rightarrow l_{12}$. Dans le sommet l_9, une alarme sera déclenchée lorsque la relation suivante est vérifiée : x_2, x_4, y_2, $y_4 \not\models E_9^*$.
La figure 4.18 montre la structure du système de surveillance. Ce dernier, représenté par l'automate à chronomètres $\mathbb{A}^*$, reçoit les signaux provenant du système commandé. En cas de violation de l'espace temporel associé à un sommet, le système de surveillance donne une alarme.

Dans la figure 4.19, nous considérons quelques scénarios de fonctionnement du système.

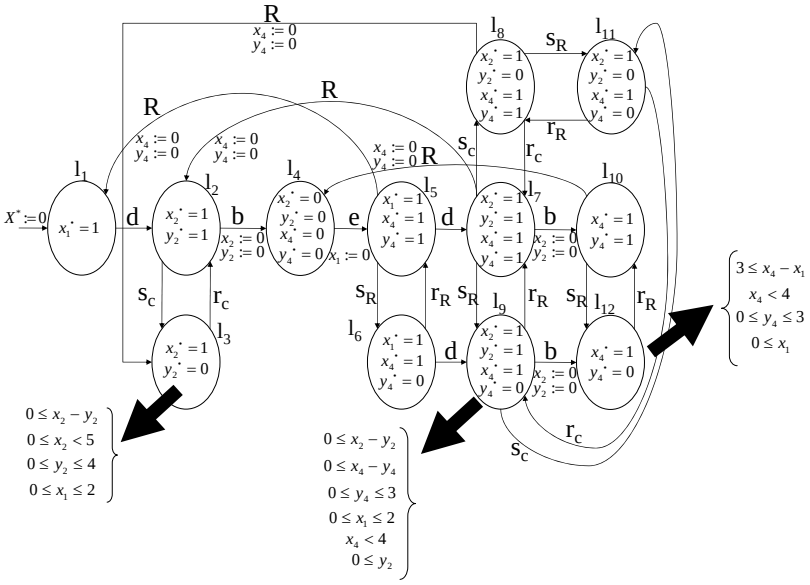

FIG. 4.16 – L'automates $\mathbb{A}$ du système considéré.

• ***Le premier scenario :***

Supposons qu'on a observé la séquence suivante d'événements : $d_{(0)} \rightarrow b_{(3)} \rightarrow (d, s_c)_{(4)} \rightarrow R_{(5)}$, présentée dans la figure 4.19.a. Dans cette séquence, on a observé l'événement s_c à l'instant $\tau = 4$ $u.t$. Physiquement ce scénario correspond au fait que, à l'instant $\tau = 4$ lorsque le poussoir est en train de poser une palette sur le convoyeur, il est bloqué en position sortie. Par conséquent, la palette a été bloquée à ce point par le poussoir. Dans ce scénario de fonctionnement, les évolutions des chronomètres x_2, x_4, y_2 et y_4 sont présentées dans la figure 4.19.b. Dans ces chronogramme, nous pouvons remarquer les chronomètres x_2 et y_2 sont initialisés à l'instant $\tau = 4$ (l'instant d'occurrence l'événement d).

L'automate $\mathbb{A}^*$ atteint le sommet l_3 du l_1 où les sommets visités au cours de ce scenario sont : $l_1 \xrightarrow{d_{(0)}} l_2 \xrightarrow{b_{(3)}} l_4 \xrightarrow{e_{(3)}} l_5 \xrightarrow{d_{(4)}} l_7 \xrightarrow{s_{c(4)}} l_8 \xrightarrow{R_{(5)}} l_3$.

L'inégalité $0 \le x_2 - y_2 < 2$ dans le sommet l_3 détecte un défaut à l'instant $\tau_1 = 6$ $u.t$. Les valeurs correspondantes de x_2 et y_2 sont respectivement $x_2(\tau_1) = 2$ $u.t$ et $y_2(\tau_1) = 0$. On peut expliquer le déclenchement de l'alarme par la raison suivante : afin d'accomplir la tâche de transfert la palette correctement jusqu'au point B, on a besoin d'avoir au moins la durée $\alpha_1 - y_2(\tau_1) = 3 - 0 = 3$ $u.t$. Donc, la valeur correspondante de x_2 est $x_2 = x_2(\tau_1) + (\alpha_1 - y_2(\tau_1)) = 2 + 3 = 5$ $t.u$. Cette duration d'exécution dépasse la maximale durée pour transférer la palette jusqu'au point B.

• ***le deuxième scenario :***

Supposons qu'on a observé la séquence suivante d'événements : $d_{(1)} \rightarrow s_{c(2)} \rightarrow r_{c(2.5)} \rightarrow$

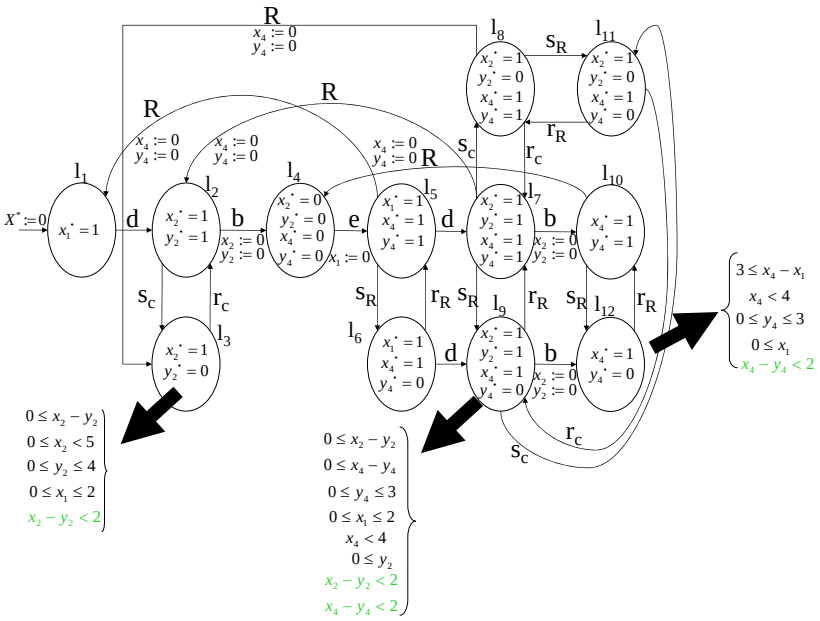

FIG. 4.17 – L'automate $\mathbb{A}^*$ du système considéré.

$s_{c(3)} \rightarrow r_{c(4)} \rightarrow b_{(5.5)} \rightarrow d_{(6.5)}$, présentée dans la figure 4.19.c. Dans cette séquence, la tâche de convoyeur a été sujette aux défauts intermittents deux fois. On a observé ce défaut, pour la première fois, à l'instant $\tau = 2$ $u.t$ et son recouvrement à l'instant $\tau = 2.5$ $u.t$. Ce défaut a été observé pour la deuxième fois à l'instant $\tau = 3$ $u.t$ et son recouvrement à l'instant $\tau = 4$ $u.t$. Ces interruptions peuvent se produire, par exemple, à cause d'un défaut intermittent dans l'alimentation électrique du moteur M. Les évolutions des chronomètres x_2, y_2 et x_1 sont présentées dans la figure 4.19.d.
L'automate $\mathbb{A}^*$ atteint le sommet l_7 du l_1 où les sommets visités au cours de ce scenario sont : $l_1 \xrightarrow{d_{(1)}} l_2 \xrightarrow{s_{c(2)}} l_3 \xrightarrow{r_{c(2.5)}} l_2 \xrightarrow{s_{c(3)}} l_3 \xrightarrow{r_{c(4)}} l_2 \xrightarrow{b_{(5.5)}} l_4 \xrightarrow{e_{(5.5)}} l_5 \xrightarrow{d_{(6.5)}} l_7$....

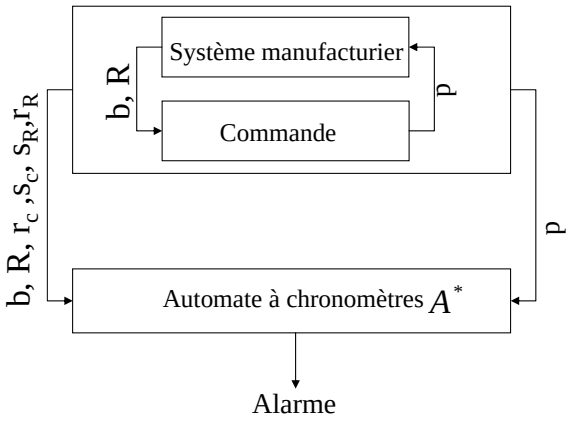

FIG. 4.18 – Structure du système de surveillance du système considéré

Notre système de surveillance ne détecte pas un défaut malgré que des interruptions se sont produit pendant l'exécution de la tâche. Ceci peut être expliqué comme suit : la tâche est exécutée correctement et l'événement b représentant la fin de $tâche_1$ se produit à l'instant $\tau_2 = 5.5$ $u.t$. Les valeurs correspondantes des chronomètres x_2 et y_2 sont respectivement $x_2(\tau_2) = 4.5$ et $y_2(\tau_2) = 3$. Ces valeurs vérifient les inégalités algébriques délimitant le fonctionnement acceptable dans les sommets visités au cours de ce scénario de fonctionnement.

• ***le troisième scenario :***
Supposons qu'on a observé la séquence suivante d'événements : $d_{(0)} \rightarrow b_{(3)} \rightarrow d_{(3.5)} \rightarrow s_{R(4)} \rightarrow r_{R(4.5)} \rightarrow s_{R(5)} \rightarrow b_{(6.5)}$, présentée dans la figure 4.19.e. Dans cette séquence, la tâche de robot est sujet aux défauts interruptibles pour deux fois. On a observé ce défaut, pour la première fois, à l'instant $\tau = 4$ $u.t$ et son recouvrement à l'instant $\tau = 4.5$ $u.t$. Ce défaut a été observé pour la deuxième fois à l'instant $\tau = 5$ $u.t$. Ces interruptions peuvent se produire, par exemple, à cause d'un défaut dans le bras manipulateur du robot. Le bras subit d'un défaut intermittent pour une durée 0.5 $u.t$. Ensuite, ce défaut a propagé à un défaut permanent apparu à l'instant $\tau = 5$ $u.t$. Dans ce scénario de fonctionnement, les évolutions des chronomètres x_2, y_2, x_4 et y_4 sont présentées dans la figure 4.19.f. Dans ces chronogramme, les chronomètres x_4 et y_4 sont initialisés à l'instant $\tau = 3$ (l'instant correspondant à l'occurrence successifs de b et e).
L'automate $\mathbb{A}^*$ atteint le sommet l_{12} du l_1 où les sommets visités au cours de ce scenario sont : $l_1 \xrightarrow{d_{(0)}} l_2 \xrightarrow{b_{(3)}} l_4 \xrightarrow{e_{(3)}} l_5 \xrightarrow{d_{(3.5)}} l_7 \xrightarrow{s_{R(4)}} l_9 \xrightarrow{r_{R(4.5)}} l_7 \xrightarrow{s_{R(5)}} l_9 \xrightarrow{b_{(6.5)}} l_{12}$....
L'inégalité $0 \leq x_4 - y_4 < 2$ dans le sommet l_{12} détecte un défaut à l'instant $\tau_3 = 6.5$ $u.t$. Les valeurs correspondantes de x_4 et y_4 sont respectivement $x_4(\tau_3) = 3.5$ $u.t$ et $y_4(\tau_4) = 1.5$ $u.t$. On peut expliquer le déclenchement de l'alarme par la raison suivante : afin d'accomplir la tâche de transfert la palette correctement jusqu'au poste d'assemblage, on a besoin d'avoir au moins la durée $\alpha_2 - y_4(\tau_3) = 2 - 1.5 = 0.5$ $u.t$. Donc, la valeur correspondante de x_4 sera $x_4 = x_4(\tau_3) + (\alpha_2 - y_4(\tau_3)) = 3.5 + 0.5 = 4$ $t.u$. Cette durée d'exécution dépasse la maximale durée pour transférer la palette jusqu'au poste d'assemblage.

4.5 Conclusion

Nous avons présenté au cours de ce chapitre le principe de notre approche de surveillance des SED. Tout d'abord, nous avons défini le comportement acceptable d'un système. Généralement, ce comportement est adopté dans les systèmes réels, qui sont sujets aux défauts intermittents afin d'augmenter leure disponibilité. Ensuite, nous avons proposé une procédure permettant de synthétiser un système de surveillance. Ce dernier détecte les défauts interruptibles : permanents et intermittents dans les systèmes

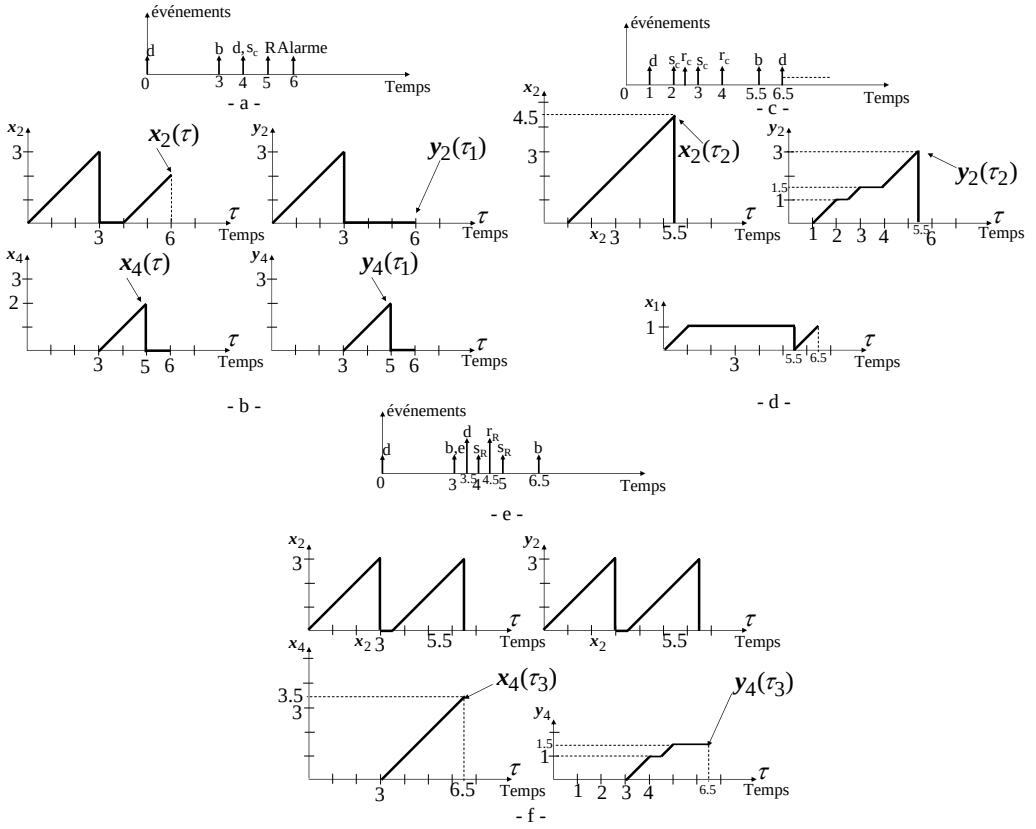

FIG. 4.19 – Quelques scénarios de fonctionnement dans le système de transfert considéré.

commandés. Cette détection a lieu au plus tôt par rapport à la date au plus tard prévue d'apparition d'un événement, ou la date au plus tard de terminaison d'une tâche.

La construction de système de surveillance se fait en deux étapes. Dans un premier temps, nous construisons le modèle du système à surveiller. Ceci est fait en faisant la composition synchrone des différents modèles des tâches constituant le système. Chaque tâche est modélisée par un automate à chronomètres. Le modèle global obtenu du système est également un automate à chronomètres. L'exécution acceptable des tâches d'un système est caractérisée par une propriété à vérifier dans l'automate représentant le comportement acceptable du système. Dans la deuxième étape, une procédure de synthèse, basée sur les techniques d'analyse d'atteignabilité de l'automate, est appliquée à ce modèle afin de calculer les trajectoires de l'automate vérifiant la propriété requise. L'automate résultant de la procédure de synthèse délimite le comportement acceptable du système qui correspond à l'exécution acceptable des tâches du système. Dans cet automate, les sommets représentent les différentes situations atteignables du système et les équations différentielles dans un sommet reflètent les dynamiques des tâches dans ce sommet : actives ou interrompues à cause des défauts interruptibles. L'évolution de cet automate se produit suite à l'observation d'un événement généré par le système

à surveiller. L'espace temporel dans un sommet caractérise les évolutions temporelles possibles du système dans la situation correspondante du système. Cet automate permettra de détecter un défaut suite à la violation de l'espace temporel dans un sommet.

La démarche de construction du système de surveillance s'effectue d'une part, hors ligne à travers la construction du système de surveillance et d'autre part, en ligne pour son implémentation. Nous avons présenté aussi au cours de ce chapitre la méthode proposée pour son implémentation en ligne.

Enfin, nous avons présenté un exemple spécifique qui illustre l'application de notre approche.

Nous allons présenter dans le chapitre 5 le modèle réseau de Petri à chronomètres Post- et Pré-initialisés. Ce modèle est proposé dans le cadre de notre travail, il permet de représenter en général le comportement des systèmes où les actions peuvent être interrompues et reprises. Ce modèle sera utilisé pour représenter le comportement des systèmes sujet aux défauts interruptibles. Notre objectif est de simplifier la modélisation du système et de profiter des caractéristiques générales des modèles RdP.

Chapitre 5

LES RÉSEAUX DE PETRI À CHRONOMÈTRES POST- et PRÉ-INITIALISÉS

5.1 Introduction

Nous avons considéré dans le chapitre 4, le problème de la surveillance des systèmes commandés sujets aux défauts interruptibles. L'outil de modélisation utilisé a été l'automate à chronomètres. La démarche a été la suivante. Nous modélisons le système à surveiller d'une manière modulaire. Nous modélisons d'abord les différentes tâches du système par des automates à chronomètres. Ensuite, nous obtenons le modèle global par la composition synchrone de ces automates. Enfin, une procédure de synthèse est appliquée au modèle obtenu. Ceci a permis de délimiter les trajectoires représentant l'exécution acceptable du système.
Modéliser le comportement du système susceptible aux défauts interruptibles par des automates à chronomètres n'est pas toujours simple. De plus, certains cas des systèmes à surveiller ne sont pas exprimables avec les automates, comme le cas des systèmes à instance-multiple. On a vu dans l'exemple présenté dans la figure 3.9.a que le modèle automate ne peut pas prendre en compte de manière simple la présence de plusieurs palettes au convoyeur B. Ceci rejoint la notion de multi-sensibilisation. Une des solutions retenues dans la littérature pour modéliser le comportement normal (sans interruption) d'un procédé à multi-instance est d'utiliser les RdP temporels avec la notion multi-sensibilisation. Cependant, les RdP temporels [Merlin, 1974], [Berthomieu and Diaz, 1991] ne sont en général pas suffisants pour modéliser les applications en présence du comportement interruptible. Ils doivent pour cela être étendus afin de pouvoir modéliser une tâche interrompue et reprise au même endroit un peu plus tard. Ceci nous conduit à proposer une extension de RdP temporels, appelée "***Réseaux de Petri à chronomètres Post- et Pré-initialisés***". Ce modèle garde les caractéristiques principales des RdP. Il a la capacité à représenter d'une manière intuitive et naturelle les principaux mécanismes des systèmes à événements discrets : parallélisme, synchronisation et partage des ressources. Ce modèle est un modèle de spécification formelle qui allient les avantages d'une description graphique puissante et d'une sémantique formelle.

Le modèle que nous proposons permet aussi de représenter les comportement des systèmes dits "préemptifs". Ces systèmes sont répandus dans les systèmes temps-réel et surtout dans le domaine de l'ordonnancement. Ils sont généralement composés de plusieurs tâches qui interagissent. Une tâche peut être suspendue puis reprise au même endroit un peu plus tard. Le modèle "***Réseaux de Petri à chronomètres Post- et Pré-initialisés***" est donc un modèle général qui va au-delà de la modélisation du comportement d'un système sujet aux défauts interruptibles 4. Dans la suite et à titre de simplification, nous indiquons ce modèle réseaux de Petri à chronomètres avec l'abbreviation ***RdP à chronomètres***.
Un certain nombre d'auteurs ont proposé d'étendre les RdP temporels afin de prendre

en compte les aspects liés à l'interruption et à la reprise d'actions : les Scheduling-TPN [Lime and Roux, 2004], les Preemptive-TPN [Bucci et al., 2004], les réseaux de Petri temporels à hyperarcs inhibiteurs ($IHTPN$) [Roux and Lime, 2004] et les réseaux de Petri Temporels à chronomètres ($SWTPN$) [Berthomieu et al., 2005]. Les deux premiers ajoutent des ressources et des priorités au modèle RdP temporels, alors que les $IHTPN$ introduisent des arcs inhibiteurs qui contrôlent la progression du temps dans les transitions. Les $SWTPN$ étendent les RdP temprels avec des arcs activateurs qui contrôlent la progression du temps dans les transitions. Nous allons présenter dans ce chapitre une comparaison entre le RdP à chronomètres et d'autres modèles réseaux de Petri, à savoir RdP temporels, Scheduling-TPN, Preemptive-TPN et $IHTPN$. La comparaison se fait en termes de comportements modélisables.
Une contribution importante dans notre modèle repose sur les concepts de *Pré-* et *Post-initialisation* des horloges. Dans le modèle RdP temporels, seulement le concept de *Pré-initialisation* est utilisé. Les horloges sont initialisées lorsque les transitions correspondantes à ces horloges sont nouvellement sensibilisées. Dans le modèle RdP à chronomètres, nous introduisons le concept de *Post-initialisation* où les horloges sont initialisées lorsque les transitions correspondantes à ces horloges sont franchies. Ce mécanisme d'initialisation des variables associées aux transitions est utilisé dans [Bobbio et al., 2000] pour les RdP stochastiques. Il est ainsi possible de modéliser la suspension et la reprise d'action. La différence entre notre modèle et celui basé sur un modèle RdP stochastiques vient du fait qu'on n'a pas de connaissance précise sur la manière de distribuer le temps associé à une transition dans un modèle RdP temporels. Autrement dit : lorsque la transition t_i à laquelle est associé l'intervalle $[a_i, b_i]$ est validée, son franchissement aura lieu quelque part dans l'intervalle $[a_i, b_i]$. Dans un modèle RdP stochastiques, on a une loi pour déterminer l'instant de franchissement de t_i. Cette différence entre les deux modèles conduit à des techniques d'analyse différentes.

Nous considérons le problème de calcul de l'espace d'états pour le nouveau modèle RdP à chronomètres, celui-ci est indécidable comme toutes les extension des RdP temporels mentionnées ci-dessus. Dans ce but, nous proposons une méthode basée sur la traduction de RdP à chronomètres en SWA. Ensuite, une analyse en avant sera appliquée sur l'automate à chronomètres ainsi obtenu en utilisant le model-checker PHAVer.
C'est à partir des avantages de modélisation en utilisant les RdP que, nous construisons le système de surveillance présenté dans le chapitre précédent à partir d'un modèle RdP à chronomètres. Concrètement, nous partons d'un modèle du système sous la forme d'un RdP à chronomètres. Nous traduisons ensuite ce modèle en automate à chronomètres SWA. Puis, la procedure de synthèse présentée dans le chapitre précédent est appliquée à cet automate afin de calculer l'automate de surveillance.

Le chapitre est organisé comme suit : nous présentons d'abord le réseau de Petri à chronomètres, sa syntaxe et sa sémantique. Une comparaison entre le RdP à chronomètres et d'autres modèles réseaux de Petri est présentée. Ensuite, l'algorithme de traduction d'un RdP à chronomètres en un SWA et le calcul de l'espace des états de RdP à chronomètres seront exposés. La modélisation par un RdP à chronomètres du système de surveillance que nous avons présenté dans le chapitre 4, sera aussi décrite en detail. Puis, deux exemples seront considérés. Le premier exemple est donné afin d'illustrer la modélisation du système de surveillance en utilisant un RdP à chronomètres. Le deuxième exemple montre d'autres possibilités de modélisation offerte par le modèle RdP à chronomètres. Enfin dans la conclusion, nous synthétiserons notre contribution.

5.2 Réseaux de Petri à chronomètres Post- et Pré-initialisés

5.2.1 Présentation informelle de RdP à chronomètres

Les réseaux de Petri à chronomètres étendent les RdP temporels [Berthomieu and Diaz, 1991] en incluant dans sa sémantique le comportement des systèmes interruptibles. Cela implique la suspension et la reprise de l'exécution des tâches, lors des interruptions. Le temps s'arrête pour les tâches interrompues, donc ce modèle s'appuie sur le concept de chronomètre, horloge pour laquelle le temps peut être arrêté et redémarré. Dans un RdP à chronomètres, il y a deux types de transitions : interruptibles et non-interruptibles. Un mécanisme d'initialisation des chronomètres appelé *Post-initialisation* est utilisé. Ce mécanisme repose sur le franchissement de la transition interruptible correspondante. Le franchissement d'une transition interruptible met à zéro le chronomètre associé à cette transition, tandis que le franchissement d'autre transition qui désensibilise la transition interruptible, suspend ce chronomètre. Il reprend lorsque la transition interruptible est sensibilisée de nouveau.

Exemple : Considérons l'exemple d'une tâche interruptible ayant une durée d'exécution $[\alpha, \beta]$ (sans compter les interruption). Une interruption peut se produire pendant l'intervalle $[0, \delta]$ après le début de tâche où $\delta \leq \beta$. La durée d'une interruption appartient à l'intervalle $[0, \gamma]$. La tâche reprend après chaque interruption au même endroit. L'interruption peut se produire plusieurs fois pendant l'exécution de la tâche. La durée entre la reprise de la tâche et l'interruption suivante est aussi $[0, \delta]$. Le RdP à chronomètres de cette tâche est présenté dans la figure 5.1.a. Dans ce modèle, la place P_2 représente l'exécution de la tâche tandis que la place P_3 représente l'état d'interruption de la tâche. Les transitions t_3 et t_4 représentent respectivement l'occurrence de l'interruption et la reprise de la tâche. La transition t_2 étant une transition interruptible représente l'exécution de la tâche. Afin de distinguer la transition interruptible des autres, nous la dessinons en

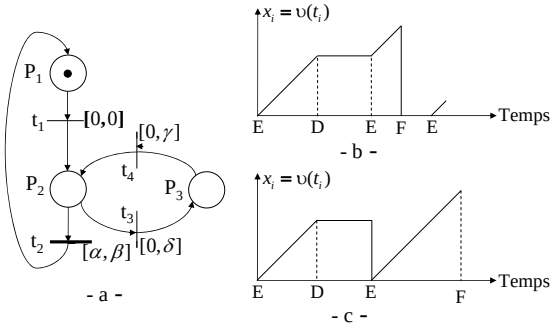

FIG. 5.1 – a- RdP à chronomètres d'une tâche interruptible b- Évolution du chronomètre associé à une transition interruptible t_i c- Évolution du chronomètre associé à une transition non-interruptible

gras. Lorsqu'une interruption a lieu, le jeton bascule de P_2 à P_3 et x_2 devient inactif. Lorsque l'interruption se termine, la transition t_4 est franchie. La tâche reprend au même endroit et le chronomètre x_2 devient actif à nouveau et récupère sa valeur atteinte et sauvegardée lors de l'interruption. Le franchissement de t_2 met à zéro x_2. A l'état initial du RdP à chronomètres, tous les chronomètres utilisés sont mis à zéro.
Nous présentons dans la suite la syntaxe et la sémantique du modèle réseau de Petri à chronomètres.

5.2.2 Syntaxe et Sémantique de RdP à chronomètres

Définition 5.2.1. *(Syntaxe d'un RdP à chronomètres) Un RdP à chronomètres est un 6-uplet* $\langle P, T, {}^\bullet(.), (.)^\bullet, M_0, I_s \rangle$*, tel que :*

- *P est un ensemble fini et non vide de places ;*
- *T est un ensemble fini et non vide de transitions. L'ensemble $T = T_{int} \cup T_{no-int}$ est composé de deux sous-ensembles disjoints : les transitions interruptibles T_{int} et non-interruptibles T_{no-int} ;*
- *${}^\bullet(.)$, $(.)^\bullet$ sont respectivement les fonctions d'incidence amont et aval ;*
- *$M_0 \in \mathbb{N}^{card|P|}$ est le marquage initial du réseau ;*
- *I_s est une fonction associant à chaque transition un intervalle donnant son instant de franchissement au plus tôt $EFT(t_i) \in \mathbb{Q}^+$ et au plus tard $LFT(t_i) \in \mathbb{Q}^+ \cup \{\infty\}$.*

□

Un marquage M du réseau est un élément de $\mathbb{N}^{card|P|}$ tel que $\forall p \in P$, $M(p)$ est le nombre de jetons dans la place p.
Une transition t est dite sensibilisée par le marquage M si $M \geq {}^\bullet(t)$. Nous notons $t_i \in enabled(M)$.
Nous noterons $v(t_i)$ la valeur d'un chronomètre associée à t_i : $v(t_i) \in (\mathbb{R}^+)^{card|T|}$.

Lorsqu'une transition t_i est franchie, le marquage M' est déduit de celui de M en retirant des jetons de chaque place en amont de t_i, et en ajoutant des jetons à chaque place en aval de t_i : $M_{temp} = M - {}^\bullet(t_i)$ et $M' = M_{temp} + (t_i)^\bullet$. □
Les transitions qui sont sensibilisées par le marquage temporaire M_{temp} et M, sont dites persistantes sensibilisées. Les transitions non-interruptibles qui sont sensibilisées par M' mais ne sont pas sensibilisées par M_{temp}, sont dites nouvellement sensibilisées. Nous noterons une transition t_i nouvellement sensibilisée par le franchissement de la transition t_k à partir du marquage M, comme $\uparrow enabled(t_i, M, t_k)$. □
Une transition interruptible est dite premièrement sensibilisée par M' si : $\forall t_i \in T_{int}$, t_i n'est pas sensibilisée par M_{temp} et $\upsilon(t_i) = 0$ au M_{temp} mais t_i est sensibilisée par M'. Rappelons que $\upsilon(t_i) \leftarrow 0$ à chaque fois t_i est franchie. □
Une transition $t_i \in T_{int}$ est dite suspendue au marquage M, ce que nous noterons $susp(M)$, $t_i \in T_{int} : t_i \in susp(M)$ si ${}^\bullet(t_i) > M \wedge \upsilon(t_i) > 0$. □
La valeur d'un chronomètre associée à $t_i \in T$ est :

- $\forall t_i \in T_{int}$, $\upsilon(t_i)$ est le temps écoulé depuis l'instant de sa première sensibilisation, pendant lequel la transition reste active (sensibilisée).
- $\forall t_i \in T_{no-int}$, $\upsilon(t_i)$ est le temps écoulé depuis l'instant pour lequel la transition t_i a été nouvellement sensibilisée.

La figure 5.1.b présente l' évolution d'un chronomètre associé à une transitions interruptible tandis que la figure 5.1.c montre l'évolution du chronomètre associé à une transition non-interruptible. Dans ces figures, les instants indiqués par E, D et F représentent respectivement l'instant de la sensibilisation, de la désensibilisation et du franchissement de la transition considérée.
Une transition t_i est dite franchissable par le marquage M, ce que nous noterons $t_i \in firable(M)$, si $t_i \in enabled(M) \wedge EFT(t_i) \leq \upsilon(t_i) \leq LFT(t_i)$. □
Pré-initialisation :Le mécanisme d'initialisation d'une transition non-interruptible $t_i \in T_{no-int}$ est dit Pré-initialisation car son chronomètre est mis à zéro lorsque t_i est nouvellement sensibilisée, c-à-d : avant l'utilisation de t_i. □
Post-initialisation : Le mécanisme d'initialisation d'une transition interruptible $t_i \in T_{int}$ est dit Post-initialisation car son chronomètre est mis à zéro par le franchissement de t_i, c-à-d : après l'utilisation de t_i. □

Nous définissons la sémantique des réseaux de Petri à chronomètres sous la forme d'un système de transitions temporisé TTS.

Définition 5.2.2. *(Sémantique d'un RdP à chronomètres) La sémantique d'un réseau de Petri à chronomètres est définie sous la forme d'un système de transitions temporisé TTS $S_N = (Q, q_0, \rightarrow)$ tel que :*
• $Q = \mathbb{N}^p \times (\mathbb{R}^+)^{card|T|}$.
• $q_0 = (M_0, \overline{0})$. *A cet état, tous les chronomètres sont mis à zéro* $\forall t_i \in T : \upsilon(t_i) := 0$.
• $\rightarrow \in Q \times (T \cup \mathbb{R}) \times Q$ *est la relation de transition incluant des transitions continues*

et des transitions discrètes :

• *la relation de transition continue est définie* $\forall d \in \mathbb{R}^+$ *par :*

$$(M,\upsilon) \xrightarrow{d} (M,\upsilon') \text{ ssi } \forall t_i \in T$$
$$\begin{cases} \upsilon'(t_i) = \begin{cases} \upsilon(t_i) + d \text{ si } t_i \in enabled(M) \\ \upsilon(t_i) \text{ sinon} \end{cases} \\ M \geq {}^\bullet(t_i) \Rightarrow \upsilon' \leq LFT(t_i) \end{cases}$$

• *la relation de transition discrète est définie* $\forall t_i \in T$ *par :*

$$(M,\upsilon) \xrightarrow{t_i} (M',\upsilon') \text{ ssi } \forall t_i \in T$$
$$\begin{cases} t_i \in firable(M), \\ M' = M - {}^\bullet(t_i) + (t_i)^\bullet, \\ \upsilon'(t_i) := 0 \text{ si } t_i \in T_{int}(Post - initialisation) \\ \forall t_k \in T, \upsilon'(t_k) = \begin{cases} \bullet\ 0 \text{ si } t_k \in\uparrow enabled(t_k, M, t_i) \\ \bullet\ \upsilon(t_k) \text{ sinon} \end{cases} \end{cases}$$

□

Dans la relation de transition discrète, les *Post-* et *Pré-initialisation* sont présentées. Dans la première, le chronomètre est mis à zéro par le franchissement de la transition interruptible correspondante, tandis que dans la pré-initialisation le chronomètre est mis à zéro lorsque la transition non-interruptible est nouvellement sensibilisée.
Le cas où la transition t_k a été suspendue dans un état précédent M'' (qui précède M et M'), puis elle reste suspendue dans l'état M ; n'est pas présenté explicitement dans la définition 5.2.2. Supposons que la valeur de t_k à l'instant de la suspension est θ. Donc, La valeur de t_k dans le marquage M' est celle de t_k à l'instant de la suspension : $\upsilon(t_k) = \upsilon'(t_k) = \theta$.

Franchissement multiple et sensibilisation multiple : Le franchissement simultané de plusieurs transitions et la sensibilisation multiple d'une transition peuvent être envisagées. Mais pour avoir une présentation simple, nous nous limitons ici aux franchissements simples (un seul franchissement à la fois) et à une seule sensibilisation d'une transition.

Définition 5.2.3. *Un réseau de Petri à chronomètres synchronisé est un 8-uplet* $\langle P, T, \Sigma, {}^\bullet(.), (.)^\bullet, M_0, I_s, \Lambda \rangle$*, tel que* $\langle P, T, {}^\bullet(.), (.)^\bullet, M_0, I_s \rangle$ *est un RdP à chronomètres,* Σ *est un ensemble fini d'événements et* $\Lambda : T \rightarrow \Sigma$*.*

□

Nous désignons, par RdP à chronomètres, implicitement des RdP à chronomètres synchronisés avec $\Sigma = \{\epsilon\}$ et $\Lambda(t) = \epsilon$, où ϵ est l'événement vide. C'est la convention que nous utilisons dans ce manuscrit. Par conséquent, nous pouvons étendre très naturellement la définition 5.2.2 et les concepts mentionnés au dessus, pour considérer les RdP à

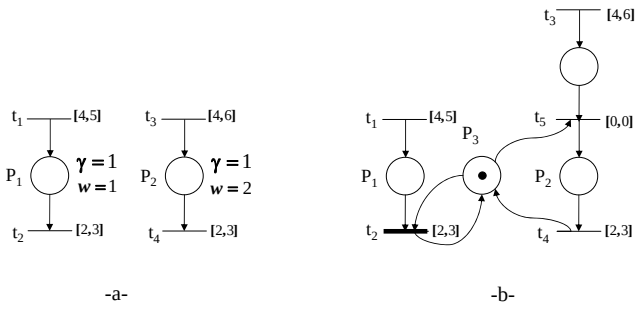

FIG. 5.2 – Modélisation de deux tâches avec des priorités fixes différentes sur un processeur par : a- Scheduling-TPN b- RdP à chronomètres.

chronomètres synchronisés.
Dans le système de surveillance que nous présenterons au cours de ce chapitre, un événement dans le RdP à chronomètres synchronisé correspond à un événement en provenance de système commandé. Dans la suite, chaque transition est associée par un nom, un événement et un intervalle temporel. L'événement ϵ peut être omis dans le modèle RdP à chronomètres lorsqu'une transition est associée par cet événement.
Dans l'étape d'analyse d'atteignibilité d'un RdP à chronomètres, il est important de noter que les événements sont traités comme des étiquettes.

5.3 Comparaison entre le RdP à chronomètres et d'autres modèles réseaux de Petri

Dans ce qui suit, nous comparons le RdP à chronomètres présenté ici avec d'autres modèles réseaux de Petri, à savoir Scheduling-TPN, Preemptive-TPN et $IHTPN$. Avant d'entrer dans le détail de cette comparaison, nous voulons indiquer deux points : tout d'abord, la comparaison se fait en terme de comportements modélisables. Le deuxième point est qu'un RdP temporels peut être décrit par un RdP à chronomètres. Ceci est dû à un RdP temporels est un RdP à chronomètres sans transitions interruptibles. Donc, un RdP temporels est strictement une sous-classe des RdPs à chronomètres.

5.3.1 Relation entre Scheduling-TPN ou Preemptive-TPN et RdP à chronomètres

Les Scheduling-TPN [Lime and Roux, 2004] étendent les RdP temporels en associant aux places deux nouveaux attributs, les ressources et les priorités. Les preemptive-TPN [Bucci et al., 2004] associent les mêmes attributs aux transitions au lieu des places.

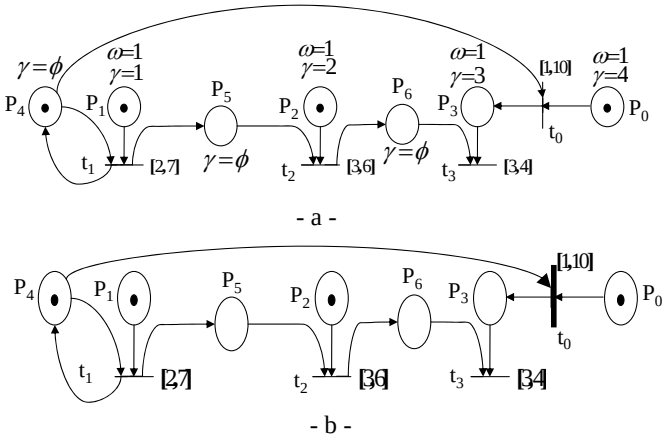

FIG. 5.3 – Modélisation d'une relation d'inhibition circulaire : a- Scheduling-TPN a- RdP à chronomètres

Les Scheduling-TPN et Preemptive-TPN sont particulièrement bien adaptés à la modélisation des systèmes préemptifs pour des objectifs d'ordonnancement. La façon d'ordonnancer des tâches réparties sur différents processeurs est prise en compte pour une priorité fixe.

Un Scheduling-TPN ou Preemptive-TPN peut être modélisé par un RdP à chronomètres. Supposons qu'il y a deux tâches différentes $tâche_1$ et $tâche_2$ représentées par les places P_1 et P_2. Ces deux tâches sont associées au même processeur $\gamma = 1$. La tâche $tâche_1$ a une priorité ($w = 1$) inférieure à celle de $tâche_2$ ($w = 2$) (Fig. 5.2.a). Lorsque les places P_1 et P_2 sont marquées, c'est le chronomètre associé à la transition t_4 qui est actif tandis que celui associé à la transition t_2 est suspendu. Le RdP à chronomètres modélisant ces deux tâches est représenté sur Figure 5.2.b. Ce modèle représente la disponibilité du processeur par la place p_3. Le fait que la priorité des tâches sur le processeur est différente, est modélisé par des arcs entre p_3 et les transitions t_2, t_4 et t_5 où la transition t_2 est une transition interruptible. Dans ce modèle, supposons que le processeur est en train d'exécuter la tâche $tâche_1$ et la transition t_3 est franchie. Dans ce cas, la transition t_5 est franchie immédiatement, la transition t_2 devient inactive et t_4 devient active. Ceci exprime que le fait que processeur n'exécute plus $tâche_1$ et exécute $tâche_2$. Lorsque $tâche_2$ se termine, la transition t_4 est franchie, et la transition t_2 revient active exprimant que le processeur exécute à nouveau $tâche_1$ au même endroit

Les Scheduling-TPN et Preemptive-TPN sont dédiés au problème d'ordonnancement des tâches pour une relation de priorité fixe, tandis que le problème de la suspension et la reprise d'une tâche est plus général. Supposons que nous avons un système composé de quatre tâches $tâche_0$, $tâche_1$, $tâche_2$ et $tâche_3$. Ces tâches s'exécutent par des ressources différentes. La spécification du système précise que :

• À l'état initial du système, les tâches $tâche_0$, $tâche_1$ s'exécutent. Le début de $tâche_2$ dépend à la fin de $tâche_1$. Le début de $tâche_3$ dépend à la fin de $tâche_2$ et $tâche_0$.
• Si $tâche_0$ se termine avant $tâche_1$, les autres tâches sont bloquées même si leurs ressources sont disponibles.
• Si $tâche_1$ se termine avant $tâche_0$, les tâches seront exécutées comme le premier point de specification précise.
Supposons qu'on veuille modéliser ce système en utilisant le modèle Scheduling-TPN (Fig. 5.3.a). Les tâches $tâche_0$, $tâche_1$, $tâche_2$ et $tâche_3$ sont présentées respectivement par P_0, P_1, P_2 et P_3. Les transitions t_0, t_1, t_2 et t_3 représentent les fins de ces tâches. Les attributs γ associé aux places est différents car les tâches sont exécutées par des différents ressources. La relation entre des tâches décrite ci-dessus est aussi présentée. Dans ce modèle, nous avons :
• D'un côté, la fin de la tâche $tâche_0$ présentée par la place P_0 bloque les tâches $tâche_1$, $tâche_2$ et $tâche_3$. L'arrêt de ces tâches a lieu même si leurs ressources sont disponibles.
• De l'autre côté, le franchissement de t_1 avant de t_0 met à zéro la valeur $\upsilon(t_0)$ (t_0 devient nouvellement sensibilisée). Ceci signifie que l'état atteint de $tâche_0$ lors de la dernière validation de t_0 est perdu lorsque $tâche_0$ est fini.
Donc, le Scheduling-TPN n'est pas capable de modéliser cette relation de priorité tandis que le modèle RdP à chronomètres l'est. Dans la figure 5.3.b, nous présentons le modèle RdP à chronomètres. Ce modèle garde la même représentation des places et des transitions de modèle présenté dans la figure 5.3.a. Mais, la transition t_0 devient interruptible. Ceci est afin de résoudre le problème d'initialisation de t_0 par le franchissement de t_1.

Pour conclure cette comparaison, les Scheduling-TPN et Preemptive-TPN sont dédiés au problème d'ordonnancement des tâches pour une relation de priorité fixe, tandis que le problème de la suspension et la reprise d'une tâche est plus général. A titre d'un exemple, ces modèles ne sont pas capables de modéliser une relation circulaire de la priorité pouvant arrêter définitivement l'écoulement de temps des transitions, tandis que les ressources nécessaires associées aux transitions ou aux places sont disponibles.
Par conséquent, les Scheduling-TPN et Preemptive-TPN sont strictement des sous-classes du RdP à chronomètres.

5.3.2 Relation entre IHTPN et RdP à chronomètres

Le modèle $IHTPN$ [Roux and Lime, 2004] étend le RdP temporels en contrôlant la progression du temps dans les transitions. Le franchissement d'une transition peut être interrompu, s'il y a une place non vide connectée à cette transition par un arc inhibiteur. Les RdP temporels à chronomètres $SWTPN$ [Berthomieu et al., 2005] étendent aussi les RdPs temporels avec un type d'arcs inhibiteurs. Ces arcs sont appelés les arcs activateurs.

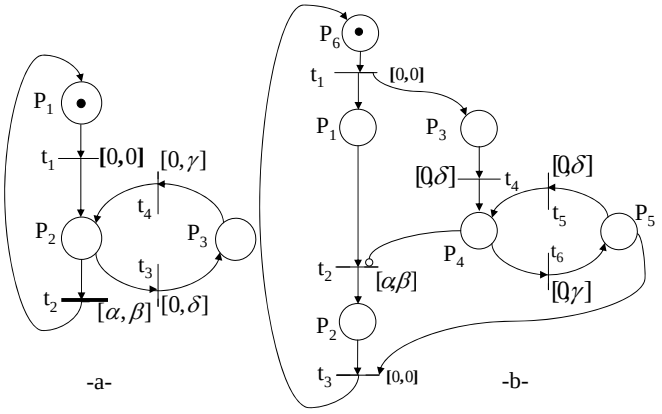

FIG. 5.4 – a- RdP à chronomètres d'une tâche interruptible b- Modélisation de la même tâche interruptible par $IHTPN$

La figure 5.4.b présente le modèle $IHTPN$ modélisant la tâche déjà présentée dans la figure 5.1.a. Dans la figure 5.4.b, les transitions t_2 et t_4 seront sensibilisées par le franchissement de t_1. Le chronomètre associé à t_2 compte la durée effective de l'exécution. Ce chronomètre devient inactif lorsque P_4 est marquée. La sensibilisation de t_4 signifie que l'interruption peut arriver à n'importe quel instant pendant l'intervalle $[0, \delta]$ après le franchissement de t_1. Lorsqu'une interruption se produit par le franchissement de t_4, t_6 sera validée et son chronomètre compte la durée d'une interruption. L'interruption et la reprise peuvent arriver plusieurs fois pendant l'exécution. Ceci est présenté par le franchissement successivement de t_5 et t_6. Lorsque la valeur $\upsilon(t_2)$ appartient à l'intervalle $[\alpha, \beta]$ et P_5 est marquée (la tâche n'est pas interrompue), les transitions t_2 et t_3 sont franchies et la place P_6 devient de nouveau marquée.
Nous pouvons remarquer que notre extension RdP à chronomètres offre un formalisme graphique classique où la seule modification apportée concerne l'initialisation des horloges qui est un mécanisme simple à appréhender. Cela permet de représenter facilement les systèmes interruptibles.

5.4 Analyse temporelle de RdP à chronomètres

Pour l'analyse temporelle des RdP à chronomètres, nous nous intéressons aux RdP à chronomètres bornés pour lesquels les bornes des intervalles temporels associés aux transitions sont des valeurs rationnelles et le RdP associé est borné. Les problèmes de l'accessibilité d'un marquage ou d'un état, et le caractère borné, sont indécidables pour les RdP temporels. Il s'ensuit que ces problèmes sont également indécidables pour les RdPs à chronomètres. Ces problèmes sont décidables pour les RdP temporels bornés [Berthomieu and Diaz, 1991]. La question posée est de savoir si ces problèmes sont décidables ou non pour les RdP à chronomètres bornés.

Théorème 5.4.1. *Les problèmes de l'accessibilité d'un marquage, d'un état, du caractère borné sont indécidables pour les RdP à chronomètres bornés.*

□

Démonstration. Un Scheduling-TPN, comme nous l'avon vu, peut être décrit par un RdP à chronomètres. Un Scheduling-TPN est donc un cas particulier de RdP à chronomètres. Les propriétés de l'accessibilité d'un marquage, du caractère borné sont indécidables pour les Scheduling-TPN bornés [Berthomieu et al., 2005]. Par conséquent ces propriétés sont indécidables pour les RdPs à chronomètres bornés. □

Nous proposons ici une méthode d'analyse temporelle de RdP à chronomètres bornés. Cette méthode est basée sur l'analyse d'atteignabilité de l'automate à chronomètres équivalent au modèle RdP à chronomètres. Nous montrons, dans la suite, comment traduire un RdP à chronomètres borné en un automate à chronomètres SWA. Le calcul de l'automate SWA se fait en deux étapes. Dans la première, nous allons calculer le graphe des marquages du RdP à chronomètres. A partir de ce graphe, nous déduirons syntaxiquement l'automate à chronomètres du RdP à chronomètres. Le sommet initial est défini par le marquage initial et les chronomètres sont associés aux transitions. Un chronomètre x_i associé à la transition t_i correspond à la valeur $v(t_i)$ présentée dans le paragraphe 5.2. Un algorithme d'analyse en avant sera ensuite appliqué sur cet automate. Cet algorithme commence de l'état initial et explore toutes les évolutions possibles du RdP à chronomètres soit par le franchissement des transitions ou en laissant le temps s'écouler. Cette méthode est une adaptation du graphe des régions dans l'automate temporisé [Alur and Dill, 1994].

5.4.1 Construction de l'automate SWA d'un RdP à chronomètres

A Étiqueter le graphe des marquages du RdP à chronomètres comme un SWA

Dans ce paragraphe, nous montrons comment construire l'automate à chronomètres à partir du graphe des marquages de RdP à chronomètres. Supposons que G est le graphe des marquages du RdP à chronomètres. Nous rappelons que le graphe des marquages est défini par le couple $G = (M, A)$, où :

- M est l'ensemble des marquages possibles du RdP à chronomètres : M_0,...M_p. Chaque marquage caractérise un état de validation ou de suspension des transitions ;
- A est l'ensemble des arcs entre les nœuds du graphe des marquages : a_0,....a_q.

L'automate à chronomètres sera obtenu en associant à chaque marquage des équations différentielles qui expriment la dynamique des chronomètres dans chaque marquage, et un invariant. A chaque arc sont associées une garde et des affectations. Comme nous l'avons supposé le nombre de franchissements simultanés est limité à un.
Nous indiquons l'automate à chronomètres obtenu à partir du graphe des marquages G par SWA_G. Dans cet automate, nous indiquons aussi chaque sommet par son marquage, c-à-d : l_0....l_k correspondent à M_0,....M_k.

Dynamique des chronomètres : Un ensemble d'équations différentielles sous la forme $\dot{x} = c$ où $c = \{0, 1\}$ est associé à chaque marquage M_k.

$$\forall t_i \in enabled(M_k) : \dot{x_i} = 1 \text{ et } \forall t_j \in susp(M_k) : \dot{x_j} = 0.$$

Les transitions inactives qui ne participent pas à l'évolution de l'automate, ne sont pas considérées. La transition t_m est inactive si :

$$(t_m \in T_{int} \wedge t_m \notin enabled(M_k) \wedge v(t_m) = x_m = 0) \text{ ou } (t_m \in T_{no-int} \wedge t_m \notin enabled(M_k)).$$

Invariant : Un invariant est associé à chaque marquage M_k. Supposons que X_k est l'ensemble des chronomètres associés à des transitions actives ou suspendues dans le marquage M_k d'un RdP à chronomètres. L'invariant associé à M_k est défini par :

$$\forall x_i \in X_k : I(M_k) = \{x_i \leq LFT(t_i) \mid t_i \in enabled(M_k) \text{ ou } susp(M_k)\}$$

Garde et affectations d'un arc : Chaque arc a_k du graphe G correspond au franchissement d'une transition t_i du RdP à chronomètres. Donc, nous étiquetons $a_k : M \xrightarrow{t_i} M_k$ par :

- Le nom de transition t_i ou son événement,
- La garde : $EFT(t_i) \leq x_i \leq LFT(t_i)$
- $\forall t_k \in \uparrow enabled(t_k, M, t_i)$ où M et le marquage d'état d'origine : nous ajoutons l'affectation $x_k \leftarrow 0$. Si $t_i \in T_{int}$, nous ajoutons aussi l'affectation $x_i \leftarrow 0$.

Exemple :
La figure 5.5 présente un RdP à chronomètres et son automate à chronomètres correspondant. Dans cet automate, les sommets l_1, l_2 et l_3 correspondent respectivement aux marquages (P_1, P_2), (P_1, P_3) et (P_1, P_4). L'invariant du sommet l_1 est obtenu à partir des bornes supérieures des intervalles de franchissement des transitions sensibilisées

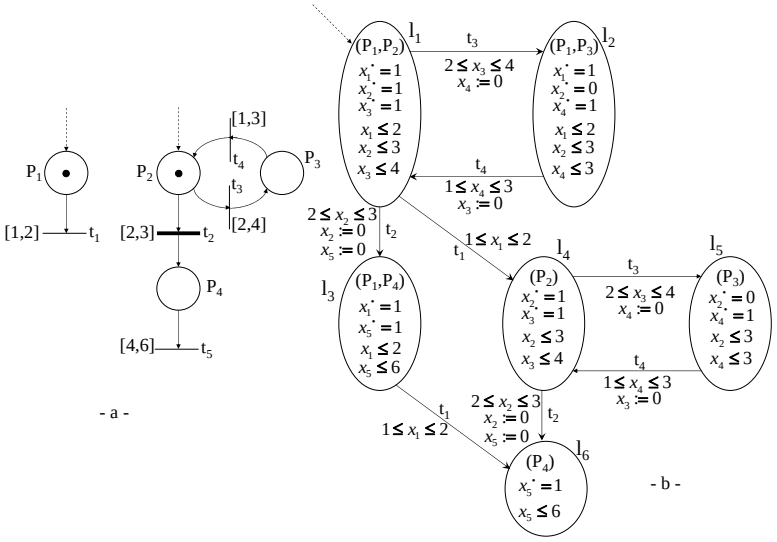

FIG. 5.5 – a- Un RdP à chronomètres b- Sa traduction en SWA

par le marquage (P_1, P_2) (c'est-à-dire : les transition t_1, t_2 et t_3). La dynamique des chronomètres dans le sommet l_1 exprime l'état des transitions dans ce marquage, donc $\dot{x_1} = \dot{x_2} = \dot{x_3} = 1$. Dans le sommet l_2, la transition t_2 est suspendue donc on a :$\dot{x_2} = 0$. La garde de chaque transition correspond aux bornes de l'intervalle de franchissement de la transition correspondante du réseau. Le franchissement de t_3 met à zéro l'horloge x_4 car la transition t_4 est nouvellement sensibilisée. Le franchissement de t_2 met à zéro le chronomètres x_2 car la transition t_2 est interruptible.

B Calcul les états atteignables de SWA$_G$

L'analyse temporelle de SWA$_G$ a pour objectif de trouver les états atteignables de SWA$_G$. Par la suite, nous présentons une itération de l'algorithme que nous proposons pour calculer des états atteignables à partir de $s_0 = (l_0, X := 0)$ où l_0 est le sommet initial. Soit SWA$_{Reach}$ l'automate contenant les états atteignables de SWA$_G$. Soit L l'ensemble des sommets atteignables de l'automate SWA$_G$. Autrement dit : L est l'ensemble des sommets de SWA$_{Reach}$. Cet ensemble L est initialisé à l_0 ($L \leftarrow l_0$). L'ensemble $Reach$ représente l'ensemble des états de SWA$_{Reach}$.

- Calculer l'évolution temporelle possible des chronomètres actifs dans l_0. Autrement dit : les valeurs possibles des chronomètres pendant le séjour de l'automate dans l_0. Ces valeurs sont contenues dans le successeur continu de s_0 : $(l_0, e'_0) = Succ_t(s_0)$. Mettre à jour l'ensemble $Reach$: $Reach \leftarrow Reach \cup (l_0, e'_0)$.
- Déterminer les transitions franchissables depuis l_0. Supposons que $a_{0,k}$ est une transition de l_0 à l_k. À la transition $a_{0,k}$ est associée l'étiquette t_i. Cette transition est franchissable si : $e'_0 \cap \{EFT(t_i) \leq x_i \leq EFT(t_i)\}$ est un polyèdre non vide. Ajouter à l'ensemble L le sommet l_k : $L = \{l_0, l_k\}$.

– Pour chaque transition franchissable $a_{0,k}$, déterminer les valeurs des chronomètres à l'entrée de l_k. Soit la transition $a_{0,k}$ correspondante au franchissement de la transition t_i, donc

$$S_{0,k} = e'_0 \cap (EFT(t_i) \leq x_i \leq LFT(t_i))$$
$$(l_k, e_k) = Succ_d(l_0, e'_0) = S_{0,k} \wedge [X_e := 0]$$

où X_e est l'ensemble des chronomètres qui sont mis à zéro par le franchissement de $a_{0,k}$. e_k est le polyèdre pour lequel le sommet l_k est atteignable.

– Calculer les évolutions temporelles possibles pendant le séjour dans l_k :

$$(l_k, e'_k) = Succ_t(l_k, e_k)$$

Mettre à jour l'ensemble $Reach$: $Reach \leftarrow Reach \cup (l_k, e'_k)$.

La généralisation de cet algorithme a été présenté au chapitre précédent dans le paragraphe 4.3.2.

Définition 5.4.1. *L'automate à chronomètres SWA_{Reach} obtenu en appliquant l'algorithme d'analyse en avant sur SWA_G est défini par :*

– *$L = \{l_0, ..., l_k\}$, est l'ensemble des marquages atteignable du $SWPN$.*
– *$l_0 = M_0$ est le marquage initial,*
– *X, est l'ensemble de tous les chronomètres associés aux transitions du RdP à chronomètres,*
– *$\Sigma = \{t_1, ..., t_q\}$, est l'ensemble des étiquettes, c-à-d : les noms ou les événements associés aux transitions de RdP à chronomètres,*
– *A, est un ensemble fini des transitions franchissables entre les sommets atteignables,*
– *$I : L \rightarrow C(X)$.*
– *$Dif \in (\{0,1\}^X)^L$ associe à chaque sommet l'activité des chronomètres associés aux transitions dans le marquage correspondant à ce sommet.*

□

C Terminaison de l'algorithme d'analyse d'atteignabilité

Nous avons proposé ci-dessus un semi-algorithme afin d'analyser temporellement un RdP à chronomètres. Si cet algorithme d'analyse ne converge pas, nous proposons d'utiliser les techniques du logiciel de vérification formelle PHAVer afin de forcer la terminaison. Si les invariants de l'automate sont bornés, la terminaison se fait en utilisation des techniques de simplifications d'un polyèdre complexe (par exemple, limiter le nombre de bits utilisé pour représenter les contraintes d'un polyèdre et limiter le nombre des contraintes décrivant un polyèdre convexe [Frehse, 2005]). En conséquence, il n'y a qu'un nombre fini de contraintes possibles pour définir un polyèdre quelconque, et donc il n'y a qu'un nombre fini des résultats pour l'algorithme de l'atteignabilité. Il s'ensuit qu'on calcule une surapproximation conservative de l'espace d'état.

5.4.2 Bisimulation entre RdP à chronomètre et son automate à chronomètres

A Calcul exact de l'espace d'états

Lorsque le calcul se termine, nous allons prouver que l'automate à chronomètres produit et le RdP à chronomètres initial sont temporellement bisimilaires, ce qui prouvera la correction de la traduction. La preuve de bisimulation est inspirée des travaux présentés dans [Gardey et al., 2006]. Ces travaux montrent la bisimualtion entre un RdP temporels et l'automate temporisé obtenu de ce modèle.

Définition 5.4.2. *Soient N un RdP à chronomètres et $\mathbb{A}$ son automate à chronomètres présenté dans la définition 5.4.1. Soient Q_N l'ensemble des états de N et Q_A l'ensemble des états de $\mathbb{A}$. Soit $R \subset Q_N \times Q_A$ une relation entre un état de N et un état de l'automate à chronomètres A tel que :*

$$\begin{cases} \forall (M, \upsilon_N) \in Q_N \\ \forall (l, \upsilon_A) \in Q_A \end{cases}, (M, \upsilon_N) R (l, \upsilon_A) \Leftrightarrow \begin{cases} M = \mathbb{M}(l) \\ \upsilon_N = \upsilon_A \end{cases}$$

où $\mathbb{M}$ est la fonction donnant le marquage associé à un état l de $\mathbb{A}$. □

Théorème 5.4.2. *$\forall$ (M, υ_N), (l, υ_A) tel que $(M, \upsilon_N) R (l, \upsilon_A)$:*

$$\bullet (M, \upsilon_N) \xrightarrow{t_i} (M', \upsilon'_N) \Leftrightarrow \begin{cases} (l, \upsilon_A) \xrightarrow{t_i} (l', \upsilon'_A) \\ (M', \upsilon'_N) R (l', \upsilon'_A) \end{cases}$$

$$\bullet (M, \upsilon_N) \xrightarrow{d} (M, \upsilon'_N) \Leftrightarrow \begin{cases} (l, \upsilon_A) \xrightarrow{d} (l, \upsilon'_A) \\ (M, \upsilon'_N) R (l, \upsilon'_A) \end{cases}$$

R est une relation de bisimulation.

Démonstration. Soient $s_N = (M, \upsilon_N) \in Q_N$, $s_A = (l, \upsilon_A) \in Q_A$, $(M, \upsilon_N) R (l, \upsilon_A)$.

• Transition continue – écoulement de temps.
- Supposons que le RdP à chronomètres N puisse laisser le temps $d \in \mathbb{R}^+$ s'écouler : $s_N \xrightarrow{d} s'_N$ où $s'_N = (M, \upsilon'_N)$ et $\upsilon'_N(t_j) = \upsilon_N(t_j) + d$. Cela signifie que $\forall t_j \in enabled(M) \Rightarrow \upsilon_N(t_j) + d \leq LFT(t_j)$. Par construction, l'invariant du sommet l est la conjonction de temps au plus tard de franchissement des transitions sensibilisées et interrompues $I(l) = \vee\{x_j \leq LFT(t_j)\} : \forall t_j \in enabled(M)$ or $susp(M)$. Donc, $\forall \tau \in [0, d] : I(t_j)(\upsilon_A(t_j) + \tau) = vrai$. Puisque le RdP à chronomètres N reste dans le même

marquage, et le SWA $\mathbb{A}$ dans le même sommet, l'activité des transitions et les conditions sur $\dot{x}$ ne change pas. Par ailleurs, on a $(M, \upsilon_N)R(l, \upsilon_A)$. Par conséquent, $(M, \upsilon_N + d) = (l, \upsilon_A + d) \Longrightarrow (M, \upsilon_N + d)R(l, \upsilon_A + d)$.
- Symétriquement, supposons que le SWA $\mathbb{A}$ puisse laisser le temps $\tau \in [0, d]$ s'écouler : $s_A \xrightarrow{\tau} s'_A$ où $s'_A = (l, \upsilon'_A)$ et $\upsilon'_A = \upsilon_A + \tau$. Cela signifie que $I(t_j)(\upsilon_A + \tau) \leq LFT(t_j)$ est vrai. Selon la sémantique du RdP à chronomètres, une transition continue peut se produire ssi $\forall t_j \in enabled(M) : \upsilon_N + d \leq LFT(t_j)$. Ce qui signifie que N peut laisser le temps d s'écouler. Puisque, on a $(M, \upsilon_N)R(l, \upsilon_A)$, alors $(M, \upsilon_N + d)R(l, \upsilon_A + d)$ pour $\tau = d$.
En conséquent, R est une relation de bisimilation pour les transitions continues.

• Transitions discrètes – franchissement d'une transition.
- Supposons que le RdP à chronomètres N puisse franchir la transition $t_j \in T : s_N \xrightarrow{t_j} s'_N$. Nous allons montrer que la transition correspondante du SWA a est aussi franchissable. Une transition t_j peut être franchie si $t_j \in enabled(M) \wedge EFT(t_j) \leq \upsilon_N(t_j) \leq LFT(t_j)$. Le nouveau marquage est $M' = M - {}^\bullet(t_j) + (t_j)^\bullet$ et les nouvelles valeurs des chronomètres sont : $\upsilon'_N(t_k) = 0$ si $t_k \in \uparrow enabled(t_k, M, t_j)$, $\upsilon'_N(t_j) = 0$ si $t_j \in T_{int}$, $\upsilon'_N(t_m) = \upsilon_N(t_m)$ pour les autres transitions. Si la transition t_m est suspendue au marquage M'' (qui précède M et M') où la valeur du chronomètres associé à t_m à l'instant de l'interruption est $\upsilon''_N(t_m) = \theta$ et t_m reste suspendue au M, donc $\upsilon'_N(t_m) = \theta$. Il y a une transition de A $a = (l, \delta, t_j, R, l')$ où $\mathbb{M}(l)$=M et $\mathbb{M}(l')$=M'. Par construction, la garde δ est : $EFT(t_j) \leq x_j \leq LFT(t_j)$. Ainsi, lorsque t_j est franchissable : $\delta(\upsilon_A) = vrai$ et le SWA $\mathbb{A}$ peut prendre la transition a. Par construction, les chronomètres qui sont mis à zéro par le franchissement de a sont les mêmes chronomètres affectés par le franchissement de t_j en N. La dynamique des chronomètres au sommet l' est également définie en accord avec les activités des transition au marquage M'. $x_k = 1$ si la transition t_k est nouvellement validée ou persistante active de M. Or, $x_k = 0$ si t_k est suspendue par le franchissement de t_j ou elle persiste suspendue de M. D'ailleurs, $I(l')(\upsilon'_A) = vrai$ car les chronomètres actifs et suspendus sont représentés dans l'invariant du l' par construction. Nous constatons que la transition a en l'automate $\mathbb{A}$ est possible et $(M', \upsilon'_N)R(l', \upsilon'_A)$.
- Symétriquement, supposons que le SWA $\mathbb{A}$ puisse franchir la transition $a : s_A \xrightarrow{t_j} s'_A$. Nous allons prouver que la transition t_j de N est aussi franchissable.
Tant que $a = (l, \delta, t_j, R, l')$ de SWA_{Reach} est franchissable, cela signifie que la transition t_j est sensibilisée par le marquage M correspondant au sommet l et que $\delta(\upsilon_a)$ est vraie. Donc, par definition de δ, $EFT(t_j) \leq x_j \leq LFT(t_j)$ et donc $EFT(t_j) \leq \upsilon_N(t_j) \leq LFT(t_j)$, ce qui signifie que t_j est franchissable pour le RdP à chronomètres N. Comme le précédent point, $(M', \upsilon'_N)R(l', \upsilon'_A)$ par construction.
En conséquent, R est une relation de bisimilation pour les transitions discrètes.

Cela termine la preuve que R est une relation de bisimulation. □

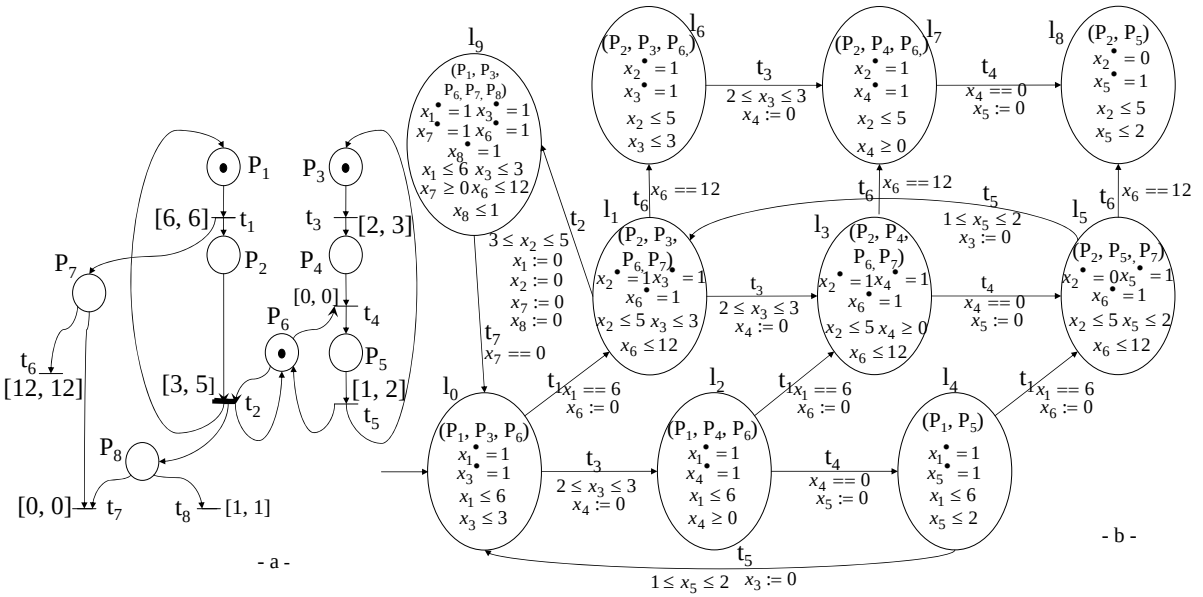

FIG. 5.6 – a- Un RdP à chronomètres b- Sa traduction partielle en SWA

B Calcul approximatif de l'espace d'états

La bisimulation est la même que celle de l'automate exact et la preuve également. Pour s'en convaincre, supposons qu'il existe dans le sommet (l) une transition sortante $a = (l, \delta, t_j, R, l')$ car t_j est franchissable dans l'espace surapproximé du (l) alors qu'il ne l'est pas dans l'espace exact correspondant. Si nous supposons qu'avant d'atteindre le sommet (l), le comportement de l'automate était correct (exact), alors à l'instant précis de l'entrée dans le sommet (l), l'automate est dans un état $a = (l, \upsilon_A)$ qui est en relation avec un état (M, υ_N) du RdP à chronomètres par R. D'une part, puisque t_j n'est pas en fait franchissable, cela signifie qu'il existe une autre transition t_i qui doit être franchie avant elle : $EFT(t_j) - \upsilon_N(t_j) > LFT(t_i) - \upsilon_N(t_i)$. Donc, par definition de R, $EFT(t_j) - \upsilon_A(t_j) > LFT(t_i) - \upsilon_N(t_i)$. Puisque, par definition d'un RdP à chronomètres, $LFT(t_i) \geq \upsilon_N(t_i)$, cela nous donne $EFT(t_j) - \upsilon_A(t_j) > 0$. D'autre part, la transition a est franchissable si $EFT(t_j) - \upsilon_A(t_j) \leq 0$. Avec ce qui précède, nous pouvons conclure que la garde est fausse et l' n'est pas donc accessible.

Cela est illustré par le réseau de la figure 5.6.a et la partie de son automate présentée dans la figure 5.6.b. L'espace temporel atteignable obtenu en l_5 par PHAVer (avec l'utilisation des opérateurs de forçage de la terminaison) est : $-7x_2 - 5x_5 + 5x_6 \geq -10 \wedge -x_2 + x_6 \geq 0 \wedge 0 \leq x_2 \leq 5 \wedge 0 \leq x_5 \leq 2 \wedge x_6 \leq 12 \wedge 2x_2 + x_5 - x_6 \geq -1$. Dans cet espace, la garde $x_6 = 12$ vérifie cet espace et le sommet l_8 est atteignable.

Nous pouvons constater que dans les sommets l_1, l_3 et l_5 la transition t_6 n'est effectivement pas franchissable, car la transition t_7 (qui désensibilise t_6 dans le modèle RdP à chronomètres) sera validée au plus tard après le franchissement de t_1 pour une valeur maximale de x_6 est 11. Cette valeur de x_6 correspond à la séquence suivante de franchisse-

ment à partir de l_3. Dans cette séquence, un état atteignable après le franchissement d'une transition est indiqué par $(l_i, (x_2, x_4, x_5, x_6))$ où les valeurs des chronomètres sont celles de l'instant d'atteindre le sommet l_i. La séquence est : $(l_3, (0, , 0, 2, 0)) \xrightarrow{t_4} (l_5, (0, 0, 0, 0)) \xrightarrow{t_5} (l_1, (0, 0, 2, 2)) \xrightarrow{t_3} (l_3, (2, 0, 2, 4)) \xrightarrow{t_4} (l_5, (2, 0, 0, 4)) \xrightarrow{t_5} (l_1, (2, 0, 2, 6)) \xrightarrow{t_3} (l_3, (4, 0, 2, 8)) \xrightarrow{t_4} (l_5, (4, 0, 0, 8)) \xrightarrow{t_5} (l_1, (4, 0, 2, 10)) \xrightarrow{t_2} (l_9, (0, 0, 2, 11))$.
Nous voyons que les sommets l_6, l_7 et l_8 qui correspondent au franchissement de la transition t_6 ne sont en fait pas accessibles. Ceci est dû au fait que la valeur de x_2 dans les sommets l_1, l_3 et l_5 est toujours inférieure ou égale à 11. Donc, la garde $x_6 = 12$ ne sera jamais vérifiée. Pour s'en convaincre complètement, prenons la séquence de l'automate à laquelle nous nous étions intéressés. Supposons que l'automate est à l'entrée du sommet l_5 avec les valeurs : $x_2 = 4$, $x_4 = 0$, $x_5 = 0$ et $x_6 = 8$. Nous pouvons laisser le temps s'écouler τ et il est clair que $\forall \tau \in [1, 2] : x_5 + \tau \in [1, 2]$ et $max\{x_6 + \tau\} < 12$. Donc la garde $x_6 = 12$ ne sera jamais vérifiée.
Nous avons vu que le calcul à l'aide des techniques existantes en PHAVer pour forcer le convergence, peut ajouter des états au SWA_{Reach}. Ces états supplémentaires ajoutés par la surapproximation ne sont pas aussi accessibles par le RdP à chronomètres initial puisque les gardes et les invariants sont calculés de manière syntaxique. Par conséquent, le comportement de SWA_{Reach} est exactement le même (au sense de la bisimulation temporelle) que celui du RdP à chronomètres initial.

5.5 Modélisation du système de surveillance proposé par le RdP à chronomètres

Nous avons présenté dans le chapitre 4, la modélisation du comportement du système sujet aux défauts interruptibles par la composition des automates représentant des différentes tâches du système. Ce comportement peut être aussi modélisé par un RdP à chronomètres. Notre objectif est d'exploiter les avantages que la modélisation par le RdP apporte. Ces avantages sont détaillés dans l'introduction de ce chapitre.
Avant d'entrer dans le détail de la modélisation, la notion de lecture d'un marquage sera introduite.

Lecture d'un marquage : Dans la figure 5.7.a, le franchissement de t_i est conditionné par le marquage de P_j, sans toutefois modifier la valeur de l'horloge associée à la transition non-interruptible t_j. On dit que l'on effectue une "lecture" du marquage P_j. Nous noterons cette opération de lecture par $\uparrow Read(P_j, M, t_i)$ qui signifie que la place P_j est lue par le franchissement de t_i à partir du marquage M. Suite au franchissement de t_i, nous proposons de ne pas changer la valeur de $\upsilon(t_j)$ (ou x_j) après le franchissement. Notre argument est que la désensibilisation et la sensibilisation de t_j sont simultanées, donc il

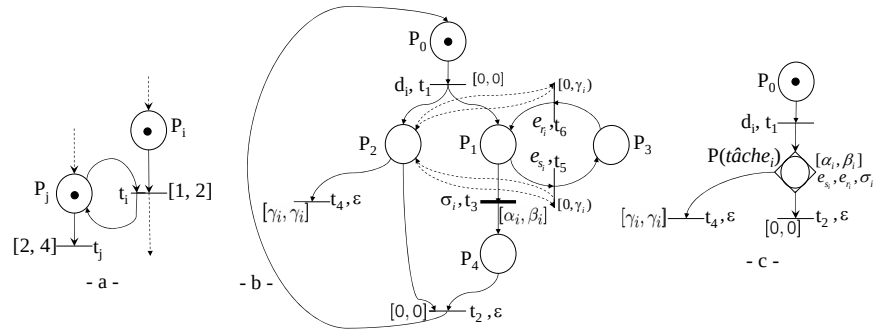

FIG. 5.7 – a- Lecture d'un marquage b- Modèle RdP à chronomètres d'une tâche interruptible c- Représentation de la tâche interruptible par la macro-place d'une tâche.

n'y a pas de temps à prendre en compte entre la la désensibilisation et la sensibilisation de la transition.
Nous noterons par $t_j \in \uparrow Read(P_j, M, t_i)$ le fait d'exprimer que la valeur de $\upsilon(t_j)$ ou x_j n'est pas changé par le lecture de P_j lorsque la transition t_i est franchie. En effet, cette sémantique donne des nouvelles possibilités de modélisation comme nous allons le voir.

Définition 5.5.1. *Soit un modèle RdP à chronomètres dans lequel on a une transition* $t_j \in T_{no-int}$, $P_j \in {}^\bullet(t_j)$, $P_j \in {}^\bullet(t_i)$ *et* $P_j \in (t_i)^\bullet$.
Soit l'évolution discrète suivante de ce modèle : $(M, \upsilon) \xrightarrow{t_i} (M', \upsilon')$ *où* $t_j \in enabled(M)$, $t_j \in enabled(M')$. *Nous avons aussi* $M(P_j) > 0$, $M'(P_j) > 0$, $M_{temp} = M - {}^\bullet(t_i)$ *et* $M_{temp}(P_j) = 0$. *Le concepteur peut choisir :*
(1)- soit la transition t_j *est nouvellement sensibilisée au* M', *c-à-d :* $t_j \in \uparrow enabled(t_j, M, t_i)$. *Dans ce cas,* $\upsilon'(t_j) = 0$.
(2)- ou la place P_j *est lue lors de franchissement de* t_i *sans modifier la valeur du chronomètre associé à* t_j, *c-à-d :* $t_j \in \uparrow Read(P_j, M, t_i)$. *Dans ce cas,* $\upsilon'(t_j) = \upsilon(t_j)$.

□

Dans la conception du modèle de surveillance, on va retenir les sémantiques du RdP à chronomètres, y compris la sémantique de lecture. On verra que cette notion nous sera très utile pour mesurer une durée totale de tâche. Dans la suite, nous allons introduire d'abord la modélisation d'une tâche interruptible. Ensuite, la modélisation d'un système sujet aux défauts interruptibles sera présentée.

5.5.1 Modélisation d'une tâche interruptible par un RdP à chronomètres

La figure 5.7.b représente le modèle RdP à chronomètres d'une tâche interruptible. Dans ce modèle, les transitions t_1 et t_2 représentent respectivement le début et la fin de

la tâche pendant la durée acceptable. Les transitions t_5 et t_6 représentent respectivement l'interruption et la reprise de la tâche, dûes à un défaut interruptible. Les événements des transitions t_1, t_5, t_6 et t_3 sont respectivement d_i, e_{s_i}, e_{r_i} et σ_i. Ces événements correspondent aux événements provenant du système à surveiller, indiquant le début, l'interruption, la reprise et la fin de la tâche. L'événement associé aux t_2 et t_4 est ϵ.
Les transitions t_3, t_4 et t_5 seront validées par le franchissement de t_1. Le chronomètre x_3 associée à la transition interruptible t_3 mesure la durée effective d'exécution tandis que le chronomètre x_4 mesure la durée totale d'exécution. La validation de t_5 par le franchissement de t_1 signifie que l'interruption peut se produire après le début de la tâche. L'interruption peut apparaître à n'importe quel instant pendant l'exécution et avant l'expiration de la durée totale d'exécution. Ceci signifie que le franchissement de t_5 lit le marquage P_2 afin de vérifier que l'intervalle acceptable n'est pas dépassé. Remarquons que la place P_2 est marquée tant que le chronomètre associé à t_4 ne dépasse pas γ_i (c-à-d : $x_4 < \gamma_i$). La lecture du marquage de P_2 est présenté par les arcs entre P_2 et t_5. Lorsqu'une interruption se produit par le franchissement de t_5, x_3 devient inactif. La transition t_6 sera validée et son chronomètre comptera la durée de l'interruption. La reprise doit se produire avant que la durée totale d'exécution n'atteigne la valeur γ_i. Ceci signifie que le franchissement de t_6 doit lire aussi le marquage P_2 afin de vérifier que l'intervalle acceptable n'est pas dépassé. Cette lecture est modélisée par les arcs entre P_2 et t_6. Lorsque la valeur de x_3 appartient à l'intervalle $[\alpha_i, \beta_i]$ et la place P_2 est marquée (autrement dit : la durée acceptable d'exécution n'est pas dépassée), les transitions t_3 et t_2 sont franchies immédiatement. Le franchissement de t_2 signifie que la tâche est exécutée pendant la durée acceptable. Le franchissement de t_4 représente un défaut car la durée acceptable est dépassée ($x_4 = \gamma_i$) mais la tâche n'est pas exécutée ($x_3 < \alpha_i$).

• ***Notation :*** Nous proposons la notation de macro-place d'une tâche interruptible. Cette notation a pour objectif de faciliter la description des systèmes complexes. Nous présentons une macro-place d'une tâche par un cercle entouré d'un losange et associé aux paramètres de la tâche : $[\alpha, \beta]$ et les événements représentant l'interruption, la reprise et la fin de la tâche (Fig. 5.7.c). Le détail décrivant une tâche interruptible est appelé l'expansion de la macro-place.

5.5.2 Modélisation d'un système sujet aux défauts interruptibles par un RdP à chronomètres

Dans ce paragraphe, nous allons présenter la modélisation d'un système S par un RdP à chronomètres. Comme nous avons vu dans le chapitre 4, la modélisation du système se fait par ses tâches. Le système S est susceptible aux défauts interruptibles, donc

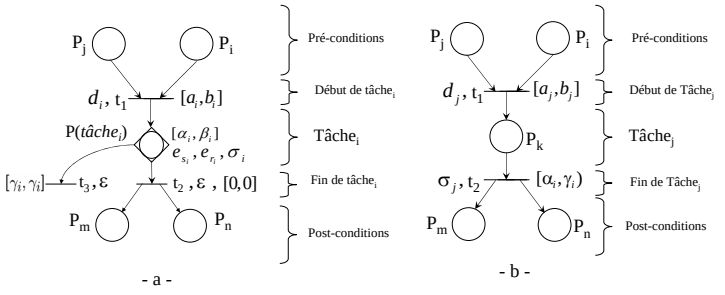

FIG. 5.8 – a- Modèle RdP à chronomètres d'une tâche interruptible b- Modèle RdP à chronomètres d'une tâche non-interruptible.

il contient des tâches interruptibles et non-interruptibles. Nous rappelons qu'une tâche est non-interruptible si l'on ne peut observer sa dynamique d'exécution. L'ensemble des tâches interruptibles dans S est indiqué par $Tâche_{int}$.

Le modèle d'une tâche interruptible $Tâche_i \in Tâche_{int}$ est présenté dans la figure 5.8.a. La tâche est délimitée par deux transitions t_1 et t_2. La macro-place $P(tâche_i)$ représente le comportement de $Tâche_i$. Le modèle de système S conditionne la validité de l'événement représentant le début de $Tâche_i$. Ces derniers correspondent des conditions de franchissement de t_1. Elles concernent les places d'entrées (disponibilité des ressources par exemple) de t_1. À cette transition peut être associé un intervalle temporel $[a_i, b_i]$. Il représente une durée prévue de franchissement t_1 après la validité des conditions de début de $Tâche_i$. Lorsque la tâche se termine, la transition t_2 est franchie. Son franchissement entraîne le marquage des places P_m et P_n en général appelées post-conditions. Ces places représentent la conséquence de $Tâche_i$ (la validité de l'événement représentant le début d'une autre tâche, par exemple).

Le modèle d'une tâche non-interruptible $Tâche_j$ est présenté dans la figure 5.8.b. La tâche est délimitée aussi par deux transitions t_1 et t_2. La place P_k représente l'exécution de $Tâche_j$. Le modèle du système S conditionne la validité de l'événement représentant le début de $Tâche_j$ (franchissement t_1). A cette transition peut être associée un intervalle temporel $[a_j, b_j]$. Il représente une durée prévue de franchissement après la validité des conditions du début de $Tâche_j$. Lorsque $Tâche_j$ se termine, la transition t_2 est franchie. Son franchissement entraîne le marquage des places P_m et P_n. Ces places représentent la conséquence de $Tâche_j$. La transition t_2 peut représenter aussi le début d'une autre tâche interruptible ou non-interruptible. Les événements d_j et σ_j représentent le début et la fin de $Tâche_j$.

5.5.3 Traduction le RdP à chronomètres du système de surveillance en automate à chronomètres

Dans ce paragraphe, nous nous intéressons à la traduction du modèle RdP à chronomètres représentant un système susceptible aux défauts interruptibles en automate à chronomètres SWA. Une procédure de traduction a été proposée dans la section 5.4. L'application de cette procédure au modèle RdP à chronomètres défini ci-dessus donne un automate contenant autant de chronomètres que le modèle RdP à chronomètres.
Dans un modèle RdP à chronomètres d'une tâche interruptible, nous avons remarqué que les chronomètres associés aux transitions sont couplés. En effet, il n'a y que deux horloges indépendantes. Notre objectif est de ne garder que ces deux horloges. Il est donc nécessaire d'exprimer toutes les gardes en fonction de ces deux horloges. De ce fait, nous proposons de traduire le modèle RdP à chronomètres en SWA de telle façon qu'on exprime les évolutions de l'automate en utilisant seulement les chronomètres indépendants représentant les durées d'exécution effective et totale des tâches. Cela permettra de diminuer le nombre d'horloges utilisées et en conséquence le coût de l'analyse en utilisant PHAVer sera diminué. L'espace temporel résultant dans chaque sommet nous convient aussi pour atteindre l'objectif de notre méthode de surveillance, exposée dans le chapitre 4.
La procédure de traduction tiendra compte les relations entre les chronomètres associés aux transitions d'un modèle RdP à chronomètres d'une tâche interruptible. Ces relations seront présentées dans la suite de ce paragraphe. Comme nous le verrons, cette traduction n'influence pas la bisimulation entre les valeurs des chronomètres représentant les durées effectives et totales des tâches dans les modèles RdP à chronomètres et SWA.

Dans la suite, nous allons mettre en évidence les relations entre les chronomètres représentant l'interruption, la reprise et les durées effective et totale d'une tâche interruptible modélisée par un RdP à chronomètres.

A Propriétés du modèle RdP à chronomètres d'une tâche interruptible

La figure 5.9.a présente le graphe des marquages du modèle RdP à chronomètres de la tâche interruptible (Fig. 5.7.b). L'application de la procédure de traduction présentée dans le paragraphe 5.4 donne l'automate présenté dans la figure 5.9.b. Dans ce dernier, le sommet non stable correspondant au marquage M_4 est omis. La transition t_2 conduisant au marquage M_1 est fusionnée avec la transition t_3 dans une seule transition. Par ailleurs, l'automate possède les propriétés suivantes.

Propriété 4. Les transitions de l'automate représentant l'interruption et la reprise ont la garde $g = 0 \leq x_4 < \gamma_i$.

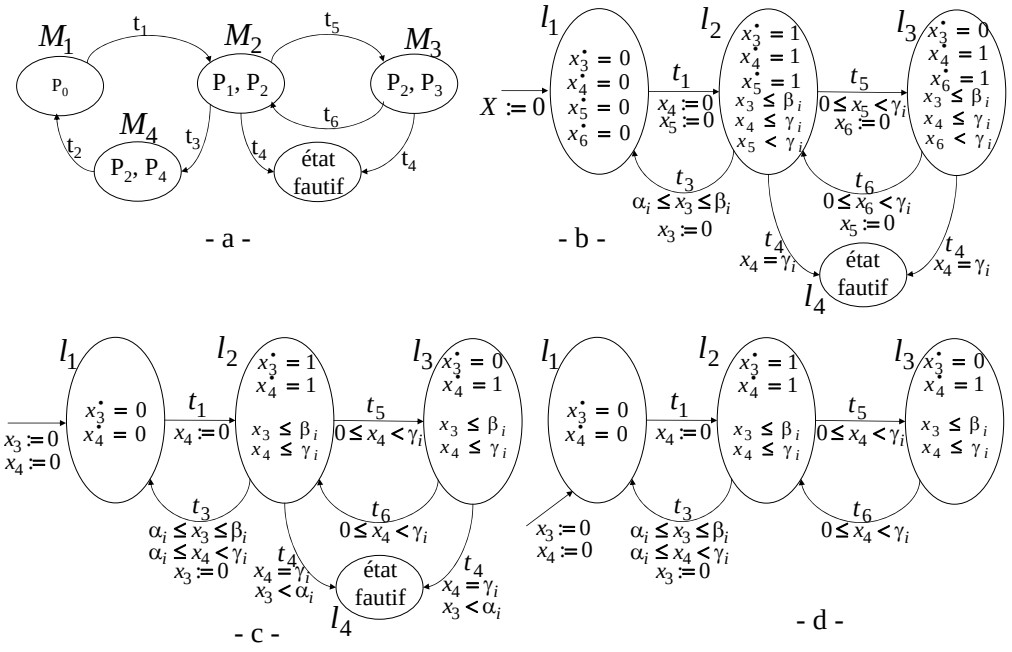

FIG. 5.9 – a- Le graphe des marquage du réseau de la figure 5.7.b b,c,d- L'automate SWA correspondant au modèle RdP à chronomètres de la tâche interruptible.

□

La figure 5.10 présente les transitions de l'automate représentant l'interruption et la reprise d'une tâche avant et après l'application de la propriété 4.

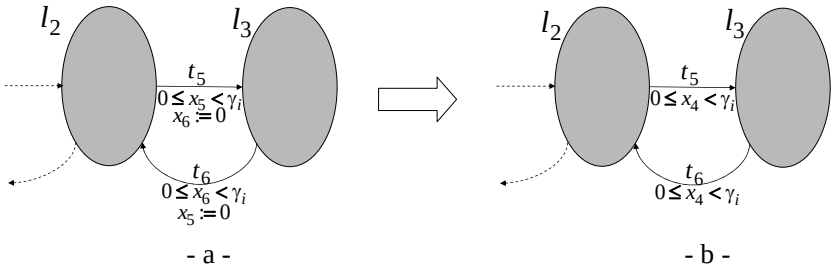

FIG. 5.10 – Les transitions de l'automate représentant l'interruption et la reprise d'une tâche avant et après l'application de la propriété 4.

Démonstration. En basant sur le graphe des marquages présenté dans la figure 5.9.a, nous remarquons le suivant :
- La transition t_5 peut être franchie du marquage M_2 correspondant au sommet l_2 si :

$$t_5 \in firable(M_2) : M(P_1) > 0 \wedge M(P_2) > 0 \wedge 0 \leq x_5 < \gamma_i$$

- La transition t_6 peut être franchie du marquage M_3 correspondant au sommet l_3 si :

$$t_6 \in firable(M_3) : M(P_2) > 0 \wedge M(P_3) > 0 \wedge 0 \leq x_6 < \gamma_i$$

En sachant que la relation suivante est toujours vraie : $M(P_2) > 0 \Leftrightarrow 0 \leq x_4 < \gamma_i$, nous avons :

$$t_5 \in firable(M_2) : M(P_1) > 0 \wedge 0 \leq x_4 < \gamma_i \wedge 0 \leq x_5 < \gamma_i \tag{5.5.1}$$

$$t_6 \in firable(M_3) : 0 \leq x_4 < \gamma_i \wedge M(P_3) > 0 \wedge 0 \leq x_6 < \gamma_i \tag{5.5.2}$$

Supposons que le modèle bascule entre les marquages M_2 et M_3 correspondent aux sommets l_2 et l_3, avant que les transitions t_3 et t_4 ne soient franchies. Dans ce cas, on a :

$$M(P_2)(\tau) = M(P_1)(\tau) + M(P_3)(\tau)$$

$M(P_1)(\tau)$, $M(P_2)(\tau)$ et $M(P_3)(\tau)$ correspondant respectivement aux marquages des places P_1, P_2 et P_3 à un instant τ. Cette relation peut être exprimée comme :

$$\dot{x_4}(\tau) = \dot{x_5}(\tau) + \dot{x_6}(\tau) \tag{5.5.3}$$

• Intégrons la relation 5.5.3 dans l'intervalle $[0, \tau_1]$ où les transitions t_5 et t_6 sont franchies une fois pendant cet intervalle. L'instant $\tau = 0$ correspond à l'instant de franchissement de t_1. L'intégration dans l'intervalle $[0, \tau_1]$ aboutit à la relation suivante : $x_4(\tau) = x_5(\tau) + x_6(\tau)$ où $x_4(0) = x_5(0) = x_6(0) = 0$.
En sachant que la relation suivante est toujours vraie : $0 \leq x_4(\tau) < \gamma_i$, on obtient : $x_5(\tau) + x_6(\tau) < \gamma_i$. Par conséquent, les relations 5.5.1 et 5.5.2 deviennent :

$$0 \leq x_4 < \gamma_i \Rightarrow 0 \leq x_5 < \gamma_i \text{ et } 0 \leq x_4 < \gamma_i \Rightarrow 0 \leq x_6 < \gamma_i \tag{5.5.4}$$

• Supposons qu'on veuille intégrer la relation 5.5.3 dans l'intervalle $[\tau_1, \tau_2]$. Dans ce dernier, les transitions t_5 et t_6 sont franchies une deuxième fois avant que les t_3 et t_4 soient franchies. L'intégration aboutit la relation suivante : $x_4(\tau) = x_5(\tau) + x_6(\tau) + x_4(\tau_1)$.
En sachant que la relation suivante est toujours vraie : $0 \leq x_4(\tau) < \gamma_i$, on obtient :

$$0 \leq x_4 < \gamma_i \Rightarrow 0 \leq x_5 < \gamma_i \text{ et } 0 \leq x_4 < \gamma_i \Rightarrow 0 \leq x_6 < \gamma_i \tag{5.5.5}$$

A partir des relations 5.5.4 et 5.5.4, on trouve la condition $0 \leq x_4 < \gamma_i$ dans les relations 5.5.1 et 5.5.2 est plus restrictive que les conditions $0 \leq x_5 < \gamma_i$ et $0 \leq x_6 < \gamma_i$. Alors, ces relations 5.5.1 et 5.5.2 deviennent :

$$t_5 \in firable(M_2) : 0 \leq x_4 < \gamma_i \wedge M(P_1) > 0 \wedge M(P_2) > 0 \tag{5.5.6}$$

$$t_6 \in firable(M_3) : 0 \leq x_4 < \gamma_i \wedge M(P_2) > 0 \wedge M(P_3) > 0 \tag{5.5.7}$$

Ces deux dernières relations prouvent la propriété. □

Propriété 5. • La transition de l'automate représentant l'exécution de la tâche a la garde : $\alpha_i \leq x_3 \leq \beta_i \wedge \alpha_i \leq x_4 < \gamma_i$.
• La transition de l'automate représentant un défaut a la garde : $x_4 = \gamma_i \wedge \alpha_i > x_3$.
□

La figure 5.11 présente les transitions de l'automate représentant la fin et l'occurrence d'un défaut d'une tâche avant et après l'application de la propriété 5.

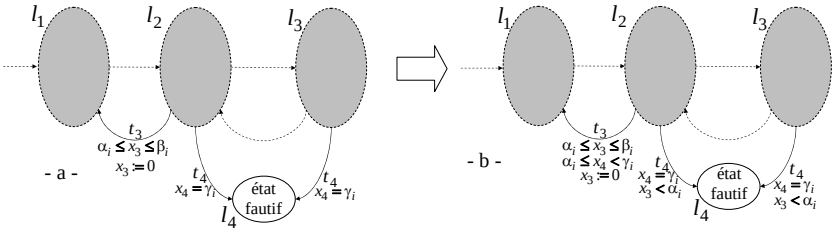

FIG. 5.11 – Les transitions de l'automate représentant la fin et l'occurrence d'un défaut d'une tâche avant et après l'application de la propriété 5.

Démonstration. • En se basant sur le graphe des marquages présenté dans la figure 5.9.a, nous remarquons que la transition t_3 est franchissable à partir du marquage M_2 correspondant au sommet l_2 si :

$$t_3 \in firable(M_2) : t_3 \in enabled(M_2) \wedge \alpha_i \leq x_3 \leq \beta_i \tag{5.5.8}$$

Comme nous avons vu dans la preuve de la propriété 4, $M(P_2)(\tau) = M(P_1)(\tau) + M(P_3)(\tau)$. Cette relation peut être exprimée comme :

$$\dot{x_4}(\tau) = \dot{x_3}(\tau) + \dot{x_6}(\tau) \tag{5.5.9}$$

Intégrons la relation 5.5.9 dans l'intervalle $[0, \tau_1]$. Dans cet intervalle, les transitions t_5 et t_6 sont franchies une fois. L'instant $\tau = 0$ correspond à l'instant de franchir t_1. L'intégration dans l'intervalle $[0, \tau_1]$ aboutit la relation suivante :

$$x_4(\tau) = x_3(\tau) + x_6(\tau) \tag{5.5.10}$$

où $x_4(0) = x_3(0) = x_6(0) = 0$.
En sachant que la transition t_3 est franchissable ssi

$$t_3 \in firable(M_2) : t_3 \in enabled(M_2) \wedge \alpha_i \leq x_3 \leq \beta_i$$

Il s'ensuit de ces deux relations que :

$$\forall \tau : \text{ si } x_3 \geq \alpha_i \Rightarrow x_4 \geq \alpha_i \tag{5.5.11}$$

En sachant que la relation suivante est toujours vraie : $t_3 \in enabled(M_2) \Leftrightarrow M(P_1) > 0 \wedge M(P_2) > 0$, nous avons :

$$t_3 \in enabled(M_2) \Leftrightarrow x_4 < \gamma_i \tag{5.5.12}$$

Les relations 5.5.8, 5.5.11 et 5.5.12 prouvent le premier point dans la propriété.
• Dans le marquage M_2 correspondant au sommet l_2, nous avons :

$$t_4 \in firable(M_2) \wedge t_3 \notin firable(M_2) \Leftrightarrow x_4 = \gamma_i \wedge x_3 < \alpha_i \tag{5.5.13}$$

Dans le marquage M_3, le fait que $t_3 \in Susp(M_3)$ signifie que $x_3 \notin [\alpha_i, \beta_i]$ à l'instant de l'interruption. Par conséquent, nous avons dans le marquage M_3 :

$$t_4 \in firable(M_3) \wedge t_3 \in Susp(M_3) \Leftrightarrow x_4 = \gamma_i \wedge x_3 < \alpha_i \tag{5.5.14}$$

Les relations 5.5.13 et 5.5.14 prouvent le deuxième point dans la propriété. □

L'application des propriétés 4 et 5 sur le graphe des marquages présenté dans la figure 5.9.a, aboutit à l'automate $\mathbb{A}$ présenté dans la figure 5.9.c. Dans cet automate :
(1) Les évolutions de l'automate sont exprimées seulement par les chronomètres x_3 et x_4.
(2) L'arc transitoire correspondant au franchissement de t_2 est fusionné avec l'arc étiqueté par t_3.

B Procédure de traduction le RdP à chronomètres du système de surveillance en automate de surveillance

Soit N un modèle RdP à chronomètres représentant un système S. Notre objectif est de construire l'automate $\mathbb{A}$ correspondant au modèle N. La construction tient compte les relations liant les chronomètres associés aux transitions utilisées pour modéliser une tâche interruptible. Avant d'introduire la procédure de traduction, nous définissons les ensembles suivants :
• T_{re} est un ensemble des transitions ayant des événements exprimant l'interruption et la reprise des tâches.
• T_{eff} est un ensemble des transitions ayant des événements exprimant la fin des tâches.
• T_{tot} est un ensemble des transitions qui mesurent les durées totales d'exécution des différentes tâches de l'ensemble $Tâche_{int}$.
L'ensemble de chronomètres X_{eff}, X_{tot} et X_{re} correspondent respectivement l'ensemble des transitions T_{eff}, T_{tot} et T_{re}.
• La fonction $fin : T_{eff} \rightarrow T_{tot}$ qui associe à chaque transition $t_i \in T_{eff}$ la transition mesurant la durée totale d'exécution. Pour la tâche interruptible présentée dans la figure 5.7.b, on a : $fin(t_3) = t_4$.
L'automate $\mathbb{A}$ peut être obtenu du RdP à chronomètres au moyen de la procédure suivante :

1. Calculer le graphe des marquages : $G = (M, A)$.
2. Calculer les dynamiques des transitions actives et suspendues dans chaque marquage. Soit un marquage $M_k \in M$.

$$\forall t_i \in enabled(M_k) \wedge t_i \notin T_{re} : \dot{x} = 1.$$
$$\forall t_j \in susp(M_k) : \dot{x} = 0.$$

3. Ajouter l'invariant dans chaque marquage. L'invariant associé à $M_k \in M$ est défini par : $I(M_k) = \{x_i \leq LFT(t_i) | t_i \in susp(M_k) \vee (t_i \in enabled(M_k) \wedge t_i \notin T_{re})\}$
4. Chaque arc $a_k \in A$ du graphe G correspond au franchissement d'une transition t_i du RdP à chronomètres, entre les marquages $M_i \rightarrow M_j$. Donc, nous étiquetons a_k par :
   - L'événement de l'arc : $\sigma_i \in \Sigma$.
   - La garde : si $t_i \in T \setminus \{T_{re} \cup T_{eff}\}$, la garde sera : $EFT(t_i) \leq x_i \leq LFT(t_i)$.
   Si $t_i \in T_{re} : \exists t_j \in\uparrow Read(P_i, M_i, t_i)$ où $t_j \in T_{tot}$, $M_i(P_i) > 0$ et $M_j(P_i) > 0$, la garde sera : $0 \leq x_j < LFT(t_j)$.
   Si $t_i \in T_{eff} : \exists t_j = fin(t_i)$, donc la garde sera : $EFT(t_i) \leq x_i \leq LFT(t_i) \wedge EFT(t_i) \leq x_j < LFT(t_j)$.
   - L'affectation : $\forall t_k \in\uparrow enabled(t_k, M, t_i) \wedge t_k \notin T_{re}$ nous ajoutons $x_k \leftarrow 0$. si $t_i \in T_{int}$, nous ajoutons aussi l'affectation $x_i \leftarrow 0$.

La procédure de construction ne tient pas compte des transitions inactives. Une transition t_m est inactive si :
$(t_m \in T_{int} \wedge t_m \notin enabled(M_k) \wedge x_m = 0)$ ou $(t_m \in T_{no-int} \wedge t_m \notin enabled(M_k))$.

Remarque 5.5.1. Dans la construction d'un modèle global du système, le franchissement de la transition mesurant la durée totale et l'arc correspondant sont supprimés au niveau de chaque tâche. Cela permettra de définir un état de défaut global et unique.

La considération de cette remarque pendant la construction de l'automate de la tâche interruptible (Fig. 5.7.b), aboutit l'automate $\mathbb{A}$ de cette tâche (Fig. 5.9.d).

Nous allons prouver la correction de la traduction en montrant que l'automate à chronomètres produit $\mathbb{A}$ et le RdP à chronomètres initial sont temporellement bisimilaires.

Définition 5.5.2. *Soient N un RdP à chronomètres modélisant un système sujet aux défauts interruptibles et $\mathbb{A}$ son automate à chronomètres obtenu par la procédure de construction présentée dans le paragraphe 5.5.3.B. Soit Q_N l'ensemble des états de N tel que $\forall q_N(M, \upsilon_N) \in Q_N : \upsilon_N(X_{sur})$ représentent seulement les valeurs des chronomètres $X_{sur} = X \setminus X_{re}$. Soit $Q_{\mathbb{A}}$ l'ensemble des états de $\mathbb{A}$. Soit $R \subset Q_N \times Q_{\mathbb{A}}$ une relation entre un état de N et un état de l'automate à chronomètres $\mathbb{A}$ tel que :*

$$\begin{cases} \forall (M, \upsilon_N(X_{sur})) \in Q_N \\ \forall (l, \upsilon_{\mathbb{A}}) \in Q_{\mathbb{A}} \end{cases}, (M, \upsilon_N(X_{sur}))R(l, \upsilon_{\mathbb{A}}) \Leftrightarrow \begin{cases} M = \mathbb{M}(l) \\ \upsilon_N(X_{sur}) = \upsilon_{\mathbb{A}} \end{cases}$$

où $\mathbb{M}$ *est la fonction donnant le marquage associé à un état* l *de* $\mathbb{A}$.

□

Théorème 5.5.1. $\forall\ (M, \upsilon_N(X_{sur}))$, $(l, \upsilon_{\mathbb{A}})$ *tel que* $(M, \upsilon_N(X_{sur}))R(l, \upsilon_{\mathbb{A}})$:

$$\bullet(M, \upsilon_N(X_{sur})) \xrightarrow{t_i} (M', \upsilon'_N(X_{sur})) \Leftrightarrow \begin{cases} (l, \upsilon_{\mathbb{A}}) \xrightarrow{t_i} (l', \upsilon'_{\mathbb{A}}) \\ (M', \upsilon'_N(X_{sur}))R(l', \upsilon'_{\mathbb{A}}) \end{cases}$$

$$\bullet(M, \upsilon_N(X_{sur})) \xrightarrow{d} (M, \upsilon'_N(X_{sur})) \Leftrightarrow \begin{cases} (l, \upsilon_{\mathbb{A}}) \xrightarrow{d} (l, \upsilon'_{\mathbb{A}}) \\ (M, \upsilon'_N(X_{sur}))R(l, \upsilon'_{\mathbb{A}}) \end{cases}$$

R est une relation de bisimulation.

□

Ce théorème peut être prouvé directement en basant sur la preuve du théorème 5.4.2 et les propriétés 4 et 5 d'un tâche interruptible.

5.6 Exemples illustratifs

Dans cette section, nous considérons deux différents exemples. Le premier concerne le système de surveillance proposé dans le cadre de ce mémoire. Il illustre les idées que nous développons dans ce chapitre sur le système de surveillance. Le deuxième exemple présente une autre application du modèle RdP à chronomètres avec sa méthode d'analyse temporelle. L'objectif de cet exemple est de montrer d'autres possibilités de modélisation offertes par le RdP à chronomètres. Dans cet exemple, nous utilisons le modèle RdP à chronomètres pour représenter un processus de production ayant des ressources partagées. Ces dernières exécutent plusieurs tâches avec des priorités différentes. Une ressource partagée peut arrêter l'exécution d'une tâche afin d'exécuter une autre tâche ayant une priorité supérieure. La méthode d'analyse temporelle du modèle RdP à chronomètres est utilisée afin d'évaluer les performances du processus considéré.

5.6.1 Surveillance d'un atelier manufacturier simple

L'exemple considéré est constitué de l'atelier manufacturier présenté dans la figure 4.14. La figure 5.12 présente le modèle RdP à chronomètres du système considéré et les expansions des macro-places représentant les tâches $tâche_1$ et $tâche_2$.

Dans le tableau 5.1, nous indiquons le chronomètre associé à chaque transition. Ces chronomètres sont indiqués par des appellations identiques à celles que nous avons utilisées dans le paragraphe 4.4. Le but est d'avoir un espace temporel comparable avec celui

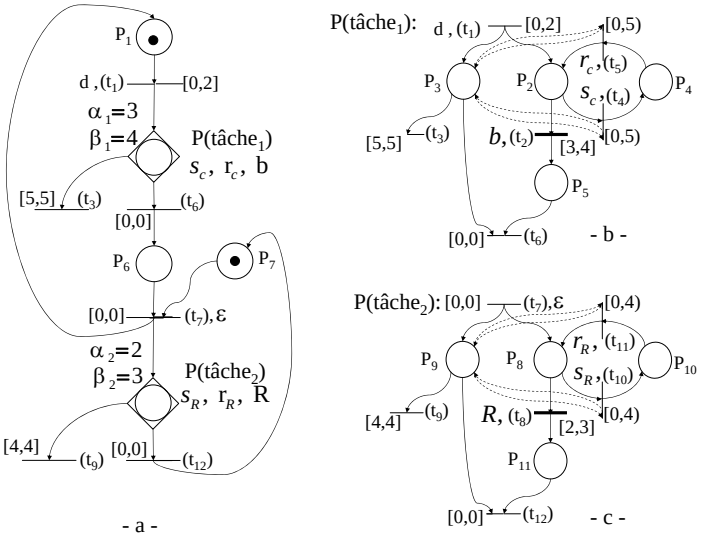

FIG. 5.12 – a- RdP à chronomètres modélisant le comportement du système manufacturier b,c- les expansions des macro-places P($tâche_1$) et P($tâche_2$).

Transition	Chronomètre	Transition	Chronomètre	Transition	Chronomètre
t_1	x_1	t_2	y_2	t_3	x_2
t_7	x_3	t_8	y_4	t_9	x_4

TAB. 5.1 – Les informations nécessaires pour construire le système de surveillance.

obtenu en suivant la méthode de la composition des automates.

La traduction du modèle RdP à chronomètres du système (Fig. 5.12) en suivant la procédure proposée dans ce chapitre donne l'automate à chronomètres présenté dans la figure 5.13. Dans cet automate, les sommets non-stables ont été supprimés. Ces sommets sont atteignables par le franchissement des transitions t_2 et t_8. Ils sont non stables car à l'instant d'entrer à ces sommets les gardes des transitions correspondant aux transitions t_6 et t_{12} sont vérifiées. Par conséquent, l'automate quitte ces sommets immédiatement. Dans l'automate présenté dans la figure 5.13, le franchissement de la transition ayant l'événement b correspond aux franchissements successifs de t_2 et t_6 dans le modèle RdP à chronomètres.
L'application de la procédure de synthèse présentée dans le chapitre 4 à cet automate nous permet de délimiter l'espace temporel caractérisant le comportement acceptable du système. Le tableau 5.2 présente l'espace temporel synthétisé dans chaque sommet de l'automate. Dans ce tableau, nous avons :
• La colonne 2 présente l'espace temporel résultant de l'application de la procédure de synthèse sur l'automate obtenu par la composition des automates des tâches.
• La colonne 3 présente l'espace temporel résultant de l'application de la procédure de

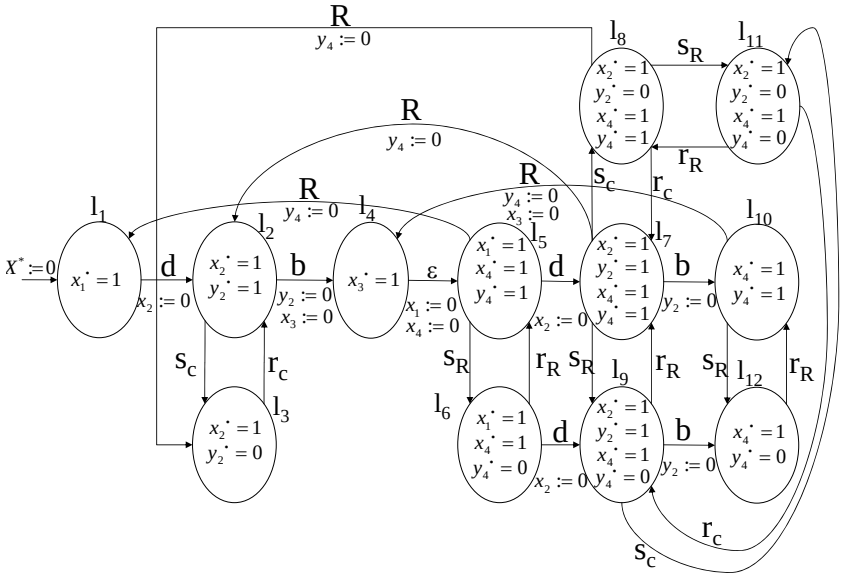

FIG. 5.13 – Les automates $\mathbb{A}^*$ et $\mathbb{A}^*$ du système considéré.

synthèse sur l'automate obtenu par la traduction du modèle RdP à chronomètres du système.

Nous pouvons remarquer qu'il y a des différences entre les espaces synthétisés présentés dans les colonnes (2) et (3) du tableau 5.2. Cette différence résulte du fait suivant :
Dans l'automate obtenu par la composition des automates des tâches, les chronomètres x_2 et y_2 sont initialisés en même temps par le franchissement de la transition ayant l'événement b. Dans l'automate obtenu du modèle RdP à chronomètres, le chronomètre y_2 est initialisé à la fin de $tâche_1$ (à l'instant de franchir t_2) et le chronomètre x_2 est initialisé à son début (à l'instant de franchir t_1). De même, les chronomètres x_4 et y_4 sont initialisés en même temps par le franchissement de la transition ayant l'événement R. Dans l'automate obtenu du modèle RdP à chronomètres, le chronomètre y_4 est initialisé à la fin de $tâche_2$ (à l'instant de franchir t_8) et le chronomètre x_4 est initialisé à son début (à l'instant de franchir t_7). Ceci est la cause de la différence entre les espaces synthétisés (1) et (2).

Il est important de noter que dans les deux automates (résultant de la composition des automates des tâches et celui obtenu du modèle RdP à chronomètres) les chronomètres x_2 et y_2 (respectivement x_4 et y_4) utilisés afin de surveiller $tâche_1$ (respectivement $tâche_2$) ont une valeur 0 à l'instant de franchir la transition représentant le début de la tâche. Par conséquent dans les sommets des deux automates dans lesquels $tâche_1$ (respectivement $tâche_2$) est active ou suspendue, les projections des espaces temporels sur

Sommet	l'espace synthétisé (1)	l'espace synthétisé (2)
l_1	$x_2 = x_4 = y_2 = y_4 == 0 \wedge 0 \leq x_1 \leq 2$	$x_3 = y_2 = y_4 == 0 \wedge -2x_2+5x_4 \geq 0 \wedge -x_2 + 2x_4 > -1 \wedge -x_4 + x_1 \geq 0 \wedge 0 \leq x_1 \leq 2 \wedge 2x_2 - 3x_4 \geq 0$
l_2, l_3	$y_4 = x_4 == 0 \wedge 0 \leq x_2 - y_2 < 2 \wedge x_2 < 5 \wedge y_2 \leq 4 \wedge 0 \leq x_1 \leq 2$	$x_3 = y_4 == 0 \wedge x_2 < 5 \wedge 0 \leq x_2 - y_2 < 2 \wedge 0 \leq x_4 < 4 \wedge 0 \leq y_2 \leq 4 \wedge 0 \leq x_1 \leq 2 \wedge x_2 - x_4 + x_1 \leq 0$
l_4	$x_2 = x_4 = y_2 = y_4 == 0 \wedge 0 \leq x_1 \leq 2$	$x_3 = y_2 = y_4 == 0 \wedge 3 \leq x_2 < 5 \wedge 0 \leq x_4 < 4 \wedge 0 \leq x_1 \leq 2$
l_5, l_6	$x_1 - x_4 == 0 \wedge x_2 = y_2 == 0 \wedge x_1 \leq 2 \wedge 0 \leq y_4 \wedge 0 \leq x_4 - y_4$	$x_1 - x_4 == 0 \wedge x_3 = y_2 == 0 \wedge 3 \leq x_2 < 5 \wedge x_4 \leq 2 \wedge 0 \leq y_4 \wedge 0 \leq x_4 - y_4 < 2$
l_7, l_8, l_9, l_{11}	$x_1 + x_2 - x_4 == 0 \wedge 0 \leq x_2 - y_2 < 2 \wedge 0 \leq x_4 - y_4 < 2 \wedge 0 \leq x_4 - x_2 \leq 2 \wedge y_2 \geq 0 \wedge x_4 < 4 \wedge 0 \leq y_4 \leq 3$	$x_1 + x_2 - x_4 == 0 \wedge x_3 == 0 \wedge 0 \leq x_2 - y_2 < 2 \wedge 0 \leq x_4 - y_4 < 2 \wedge 0 \leq y_4 \leq 3 \wedge y_2 \geq 0 \wedge 0 \leq x_4 - x_2 \leq 2$
l_{10}, l_{12}	$x_2 = y_2 == 0 \wedge 0 \leq x_1 \wedge 0 \leq x_4 - y_4 < 2 \wedge 3 \leq x_4 - x_1 \wedge x_4 < 4 \wedge y_4 \leq 3$	$x_3 = y_2 == 0 \wedge -x_2 + x_4 - x_1 \geq 0 \wedge 3 \leq x_2 \wedge y_4 \leq 3 \wedge x_4 - y_4 < 2 \wedge 0 \leq x_1 \wedge x_4 < 4$

TAB. 5.2 – L'espace temporel dans les sommets de l'automate $\mathbb{A}^*$.

(x_2, y_2) (respectivement (x_4, y_4)) sont identiques. Ceci peut être remarqué directement en comparant les espaces temporels dans les sommets l_7, l_8, l_9 et l_{11}, présentés dans le tableau 5.2.

5.6.2 Évalution le performance d'un système de production

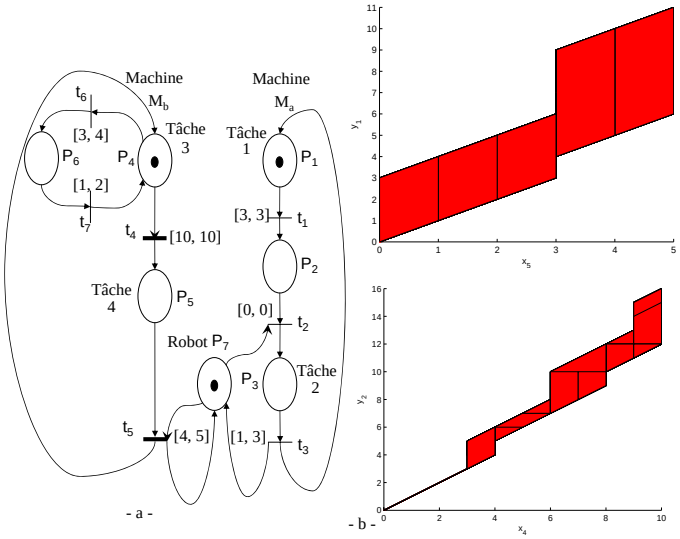

FIG. 5.14 – a- RdP à chronomètres modélisant le processus de production b- les projection de l'espace temporel atteignable sur les axes (x_4, y_2) et (x_5, y_1)

• ***Système de production considéré et son modèle*** :
On considère un processus de production de deux types de produits S_1 et S_2. Les différentes tâches du processus s'exécutent sur les machines M_a, M_b et un robot de manipulation représente une ressource partagée. Les tâches Tâches 1 et Tâche 2 sur S_1 s'exécutent en séquence respectivement sur la machine M_a et le robot. Les durées d'exécution sont respectivement de $[3, 3]$ $u.t$ pour Tâche 1 et $[1, 3]$ $u.t$ pour Tâche 2. Les tâches Tâches 3 et Tâche 3 sur S_2 s'exécutent en séquence respectivement sur la machine M_b et le robot. Tâche 3 s'exécutant sur la machine M_b pour S_2 est une tâche interruptible. Sa durée d'exécution (sans compter les interruptions) est de $[10, 10]$ $u.t$. L'interruption peut arriver après le début de la tâche dans l'intervalle $[3, 4]$ $u.t$. La durée d'une interruption est de $[1, 2]$ $u.t$. La durée entre la reprise de la tâche et l'interruption suivante est $[3, 4]$ $u.t$. Tâche 4 s'exécutant sur S_2 par le robot a une durée d'exécution $[4, 5]$ $u.t$. Tâche 2 a une priorité supérieure à celle de Tâche 4 par rapport au robot. Lorsque le robot exécute Tâche 4 pour S_2 et la machine M_a termine Tâche 1 pour S_1, le robot exécute immédiatement Tâche 2 pour S_1 en interrompant Tâche 4 si celle-ci était en cours d'exécution. Lorsqu'il termine, Tâche 4 reprend au même endroit où elle a été interrompue.

La Figure 5.14.a présente le modèle RdP à chronomètres représentant le processus de production considéré. La figure 5.15 montre l'automate à chronomètres correspondant au modèle RdP à chronomètres où les sommets instables correspondant à la transition t_2 ont été supprimés. Cet automate peut être exploité de plusieurs manières, on peut par exemple déterminer les performances du système de production.

• ***Évalution des performances du système considéré :***

Il nous a paru intéressant de calculer les durées maximales des Tâche 3 et Tâche 4 car elles sont interruptibles. Pour cette raison, nous introduisons dans l'automate les variables y_1 et y_2. La variable y_1 est activée par le franchissement de t_4. Elle est désactivée par le franchissement de t_5. y_1 calcule la durée d'exécution totale de Tâche 4. La variable y_2 est active dans les sommets qui suivent l'arc libellé par t_5 et désactive dans les sommets qui suivent l'arc libellé par t_4. Elle mesure la durée d'exécution de Tâche 3. Les projections de l'espace atteignable calculé sur les axes (x_4, y_2) et (x_5, y_1) sont présentées dans la figure 5.14.b. Nous trouvons pour l'espace atteignable de (x_5, y_1) que la durée maximale afin que Tâche 4 puisse être exécutée est $y_1 = 11$. Cette valeur est bien sûre supérieure à 5 à cause de la priorité de Tâche 2 supérieure à celle de Tâche 4 On peut voir également sur cette figure que Tâche 4 a été interrompue 1 fois pendant l'exécution. Nous trouvons aussi à partir de l'espace atteignable de (x_2, y_2), que la machine M_b prend une durée $12 \leq y_2 \leq 16$ afin d'exécuter Tâche 3. Cette figure exprime aussi que Tâche 3 peut être interrompue 2 ou 3 fois pendant l'exécution.
On a pu ainsi mettre en évidence un comportement du système que l'on peut difficilement déterminer directement à partir du RdP à chronomètres.

5.7 Conclusion

Nous avons proposé un formalisme permettant de modéliser une sous-classe des systèmes temps-réel appelé les systèmes interruptibles. L'extension proposée conserve les propriétés fondamentales d'un réseau de Petri tout en apportant des possibilités nouvelles et intéressantes de modélisation de l'interruption et la reprise d'une activité. Ce nouveau modèle étend le concept d'horloge au concept de chronomètre. Un nouveau mécanisme d'initialisation des chronomètres appelé "Post-initialisation" est introduit. Une comparaison du modèle RdP à chronomètres avec d'autres modèles RdP utilisés à modéliser les systèmes interruptibles est aussi présentée. La comparaison se fait en termes de comportements modélisables.
Nous avons présenté aussi une méthode pour l'analyse temporelle des RdPs à chronomètres. Elle consiste dans un premier temps à traduire le RdP à chronomètres borné en automate à chronomètres. Dans un second temps, nous appliquons un semi-algorithme

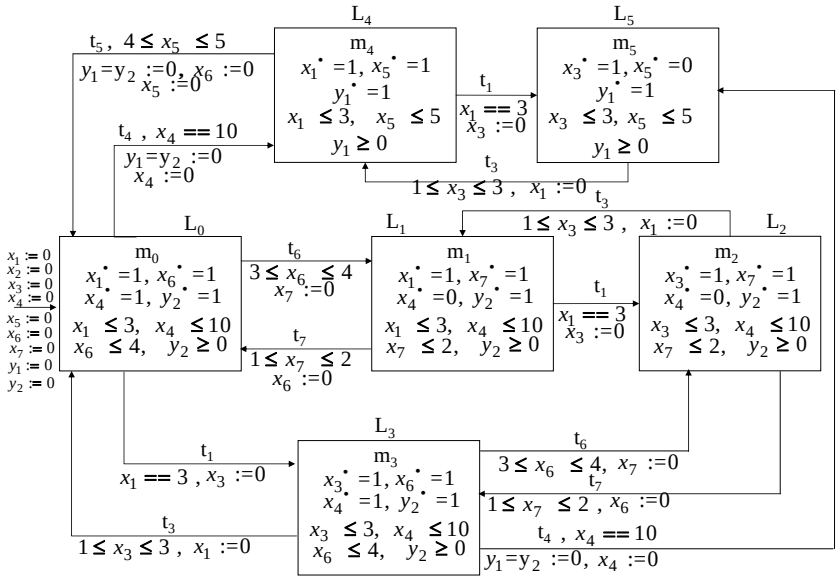

FIG. 5.15 – L'automate à chronomètres du réseau de la figure 5.14.

sur l'automate ainsi obtenu, basé sur les techniques d'analyse en avant connues dans les automates hybrides et en utilisant le logiciel de vérification formelle PHAVer. Si cet algorithme d'analyse ne converge pas, nous avons proposé d'utiliser les techniques du logiciel de vérification formelle PHAVer afin de forcer la terminaison. Il s'ensuit qu'on calcule une surapproximation conservative de l'espace d'état. Cette analyse permet de calculer l'espace d'états temporel de l'automate à chronomètres qui est bisimilaire au réseau de Petri à chronomètres initial.
Ensuite, nous avons combiné les possibilités du RdP à chronomètres et la puissance d'analyse de SWA pour concevoir notre système de surveillance présenté dans le chapitre 4. Nous avons utilisé les RdPs à chronomètres afin de modéliser les systèmes sujet aux défauts interruptibles. Le modèle RdP à chronomètres d'une tâche interruptible a certaines propriétés. Grâce à ces propriétés, nous avons traduit le modèle RdP à chronomètres du système en un automate. Ce dernier exprime les évolutions du système en utilisant seulement les chronomètres mesurant les durées effectives et totales des tâches. Enfin, nous avons considéré deux applications du RdP à chronomètres. Dans la première il s'agit d'un système de surveillance d'un atelier manufacturier. La deuxième application concerne la modélisation d'un système de production ayant des ressources partagées. Les tâches exécutées sur ces ressources ont des priorités différentes. De par la différence de priorités entre les tâches exécutées par la même ressource, la durée totale d'exécution de ces tâches est différente de leure durée d'exécution effective. Des calculs d'indices de performances sont effectués sur l'automate et interprétés sur le modèle réseau de Petri.

Nous pouvons remarquer que le travail présenté jusqu'à l'instant concerne la méthode de conception du système de surveillance. Notre objectif est de concevoir un système de surveillance implémentable facilement dans le milieu industriel. Pour cela, nous allons

présenter dans le chapitre 6 la méthode d'implémentation du système de surveillance ou plutôt de l'automate de surveillance en utilisant l'automate programmable industriel (API). Nous allons également tester notre approche de surveillance en utilisant un environnement de simulation Matlab.

Chapitre 6

IMPLÉMENTATION ET VALIDATION EXPÉRIMENTALE DU SYSTÈME DE SURVEILLANCE EN UTILISANT L'AUTOMATE PROGRAMMABLE INDUSTRIEL

6.1 Introduction

La surveillance d'un système en appliquant notre méthode s'effectue, hors ligne en construisant le système de surveillance et, en ligne pour son implémentation. Nous pouvons remarquer que la plupart des travaux présentés jusqu'à présent, concernent la conception de systèmes de surveillance et s'intéressent peu à la mise en œuvre. Pour cela, nous allons présenter dans ce chapitre la méthode d'implémentation du système de surveillance en utilisant l'automate programmable industriel (API). Dans cet objectif, nous présentons la méthode de traduction structurelle de l'automate de surveillance en programme SFC "Sequential function Chart" [International Electrotechnical Commission, 2003]. Ce dernier est un outil de programmation de l'API très efficace. Associé aux langages de programmation de l'automate, il permet d'obtenir une programmation claire et lisible. Par conséquent, nous ne parlons pas, dans la suite, de modèle de SFC mais de programme SFC implémentable dans l'API.

Les travaux présentés dans les chapitres 4, 5 montrent que le comportement acceptable d'un système est synthétisé par un automate à chronomètres. Dans ce modèle, les sommets représentent les différentes situations atteignables du système. Les équations différentielles dans un sommet reflètent les dynamiques des tâches dans la situation correspondante du système : actives ou interrompues à cause des défauts interruptibles. L'espace temporel dans un sommet caractérise les évolutions temporelles possibles du système dans la situation correspondante du système. L'automate peut rester dans un sommet tant que les valeurs des chronomètres vérifient l'espace temporel de ce sommet. Le programme SFC correspondant à cet automate représente un sommet par une étape. Une transition dans l'automate peut être représentée aussi par une transition dans le programme SFC. La question qu'on peut poser est : comment les informations temporelles peuvent être représentées dans un programme SFC ?
Généralement dans un programme SFC, le temps est assuré par l'utilisation de temporisateurs et des variables temporelles associées aux étapes. Comme nous le verrons, ni les temporisateurs ni les variables temporelles associées aux étapes ne sont capables de représenter le comportement d'un chronomètre de l'automate de surveillance. Ce problème affecte à son tour la méthode de représentation de l'espace temporel dans chaque sommet. Cependant, nous allons montrer que ce problème peut être résolu en utilisant des variables intermédiaires et en associant des actions aux étapes de programme SFC.

Après la construction du programme de surveillance, l'étape suivante est de tester notre approche avant de l'implémenter. Donc dans la deuxième partie de ce chapitre, un test sera appliqué au le système de surveillance d'un procédé industriel. Pour ce faire, nous allons construire l'automate et le programme SFC representant son

comportement acceptable. Ensuite, la validation expérimentale sera appliquée au système de surveillance (l'automate et le programme SFC). Elle consiste à simuler le système de surveillance du procédé considéré en réagissant à des signaux binaires. Ces derniers représentent les événements provenant du procédé et sa commande. Les outils de simulation Checkmate [Silva et al., 2001], Stateflow [The MathWorks, 2004] et OPC de l'environnement de simulation Matlab seront utilisés.

Avant d'entrer dans le détail du passage d'un automate représentant le comportement acceptable d'un système, en programme SFC, nous présentons d'abord un rappel du SFC et sa syntaxe.

6.2 Outil de programmation de l'API : SFC

Un automate programmable industriel (API) est une machine électronique programmable utilisée pour piloter des systèmes automatisés. Sa flexibilité explique son large domaine d'utilisation. Dans notre cas, le but de l'API est de surveiller des procédés physiques en générant des alarmes en réponse à la violation du comportement acceptable du système surveillé. Les entrées de l'API sont les signaux de capteurs du système et les ordres en provenance de système de commande. Un API est programmé à l'aide de langages spécialisés. La norme CEI 61131-3 définit quatre langages correspondant aux familles de langages les plus utilisés pour la programmation des API. Les langages sont : liste d'instructions (Instruction List,IL), texte structuré (Structured Text, ST), langages graphiques de schémas à relais (Ladder Diagrams, LD), et de diagrammes de blocs fonctionnels (Functional Block Diagrams, FBD).
La norme CEI 61131-3 définit aussi le SFC (Sequential Functional Chart) comme un outil de programmation. Cet outil n'est pas considéré comme un langage mais comme des éléments de structuration d'une application. Cela vient du fait que le SFC ne fournit pas une syntaxe complète pour la description des actions. Celles-ci doivent être spécifiées à l'aide d'autres langages de programmation.
Dans la suite de ce paragraphe, nous allons d'abord décrire l'outil SFC et ensuite présenter en bref la relation entre le SFC et les modèles formels.

6.2.1 Syntaxe

L'outil SFC défini dans la norme CEI 61131-3 est fondé sur le Grafcet [David, 1995], [Zaytoon, 2002]. Ce dernier est défini dans la norme CEI 60848 [Commission, 1988].
Le SFC repose sur les notions d'étape et de transition (Fig. 6.1.a) que nous détaillons par la suite :

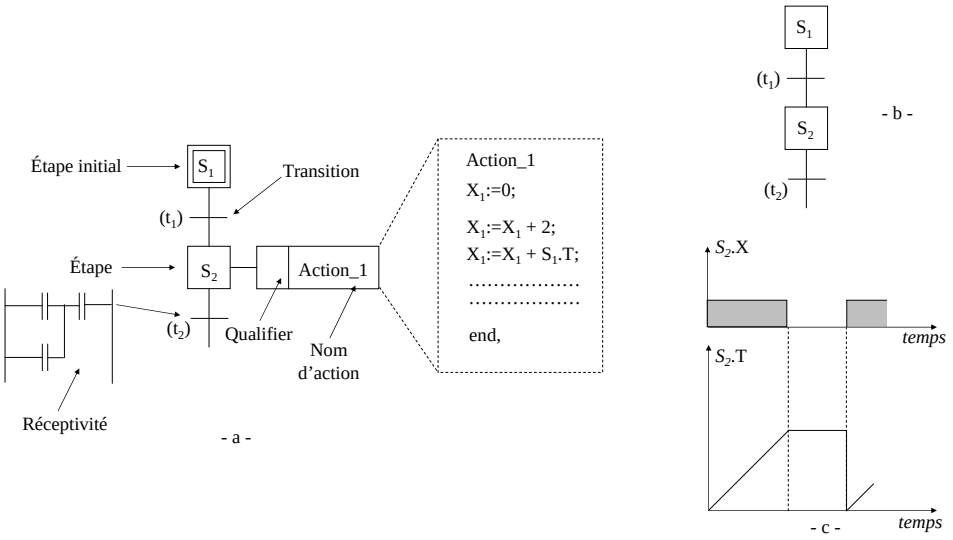

FIG. 6.1 – a- Les composants d'un programme SFC. b- Un programme SFC c- L'évolution des variables associées à l'étape S_2 du programme.

• *Étape :* Une étape correspond à une situation dans laquelle tout ou une partie du système est invariant, par rapport à ses entrées et ses sorties. Chaque étape porte un nom unique. La situation initiale d'un programme est caractérisée par l'étape initiale. Chaque SFC ne peut comporter qu'une seule étape initiale. Dans notre travail, nous indiquons l'ensemble des étapes du programme SFC par : $\{s_1, s_2, ...s_i, ...s_n\}$ où n le nombre des étapes du programme.

• *Action* : Une ou plusieurs *actions* peuvent être associées à une étape. Une *action* représente un ensemble d'opérations à réaliser lorsque l'étape associée est activée. Une action est caractérisée par : un *qualificatif*, un *nom* et *un élément du programme*. Le qualificatif d'action spécifie le type d'action (action continue N, action impulsionnelle à l'activation de l'étape P_1 et action impulsionnelle à la désactivation de l'étape P_0, action mémorisée). Nous nous limitons ici à présenter les actions de type N, P_0 et P_1. Elles seront utilisées dans la suite de notre travail. L'action continue ayant le qualificatif N est scrutée et exécutée tant que l'étape associée est active. L'action ayant le qualificatif P_1 est exécutée une seule fois lorsque l'étape associée passe de l'état inactif à l'état actif (sur front montant de l'étape). L'action ayant le qualificatif P_0 est exécutée une seule fois lorsque l'étape associée passe de l'état actif à l'état inactif (sur front descendant de l'étape). L'élément d'action est un ensemble d'instructions écrit en IL, ST ou SFC qui décrit ce que fait l'action. Dans la figure 6.1.a, l'élément de l'action associée à l'étape s_2 et ayant le nom $Action_1$, est écrit en utilisant la langage ST.

• *Transition* : Une transition indique la possibilité d'évolution entre les étapes. Chaque transition porte un nom. On associe à chaque transition une condition logique appelée *réceptivité*. Cette condition sera écrite en utilisant les langages suivants : Instruction List (IL), soit en Structured Text (ST) ou soit en Ladder Diagram (LD). Dans la figure 6.1.a,

la réceptivité de la transition t_2 est écrite en Ladder Diagram (LD).
• *Variables associées à une étape* : Généralement, *une étape* peut être active ou inactive. Une variable booléenne *nom-étape*.X, dont *nom-étape* est le nom d'une étape, représente l'état actuel de l'étape (active ou inactive). Elle a la valeur vrai lorsque l'étape est active, et la valeur faux lorsqu'elle est inactive. Considérons le programme présenté dans la figure 6.1.b. Supposons qu'on veuille suivre l'évolution de la variable booléenne $s_2.X$ associée à l'étape S_2. La figure 6.1.c présente l'évolution de cette variable correspondant à la séquence suivante de franchissement des transitions : $t_1 \rightarrow t_2 ..... \rightarrow t_1$. Également, le temps écoulé depuis l'activation d'une étape est représenté par le *nom-étape*.T. Lorsqu'une étape est désactivée, cette variable reste à sa dernière valeur. Lorsque l'étape est nouvellement activée, la valeur est remise à 0. Considérons à nouveau le programme présenté dans la figure 6.1.b. Supposons qu'on veuille suivre aussi l'évolution de la variable temporelle $s_2.T$. Lorsque cette étape est active, cette variable compte le temps écoulé. Lorsque l'étape est désactivée à l'instant de franchir t_2, cette variable reste à sa dernière valeur. Lorsque l'étape est nouvellement activée par le franchissement de t_1, la valeur est remise à zéro et recommence à compter le temps.
Les variables associées aux étapes peuvent être utilisées en mode lecture, dans une réceptivité ou/et dans une action.
• *Hiérarchie entre programmes SFC* :
Les programmes SFC hiérarchisés forment une structure de type maître esclave dans laquelle le programme maître donne des ordres à un ou plusieurs programmes SFC esclaves. La figure 6.2 présente deux programmes : un maître SFC_1 et un esclave SFC_2. Lorsque le programme SFC_1 entre dans l'étape s_2, le programme SFC_2 sera exécuté et les actions a_2 et a_3 seront aussi exécutées. Si la réceptivité g_{11} associée à la transition t_{11} est vérifiée, le programme SFC_2 bascule vers l'étape s_{12} et ainsi de suite. Dans le programme SFC_1, lorsque la réceptivité g_2 associée à la transition t_2 est vérifiée, le SFC_1 bascule vers l'étape s_1 et le SFC_2 est bloquée. En effet, la norme CEI 61131-3 a une ambiguïté concernant l'étape du programme SFC_2 qui est activée lorsque l'étape s_2 dans SFC_1 est active. Une des solutions possibles de cette ambiguïté est la suivante : l'action dans un programme SFC_1 peut avoir une histoire. Par conséquent, il est possible, d'une part, d'entrer toujours dans la dernière étape active (l'action a une histoire). D'autre part, entrer toujours dans l'étape initiale (l'action n'a pas une histoire).

6.2.2 Transformation un programme SFC en modèle formel

L'automate à chronomètres est un modèle formel, tandis que le SFC est un outil informel. La relation entre un programme SFC et les modèles formels a été déjà considérée dans la littérature. Des nombreux travaux de recherches visent à améliorer la sûreté des programmes pour l'API par les méthodes formelles. Parmi ces travaux, nous citons [Stursberg et al., 2005], [Smet et al., 2000], [Couffin and Lesage, 2000], [Frensel

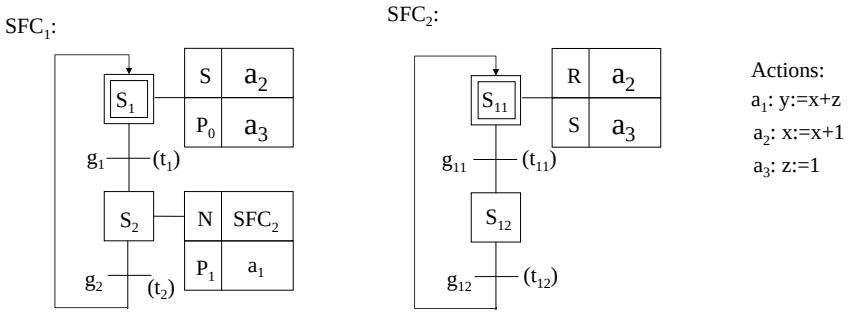

FIG. 6.2 – Concept de l'hiérarchie.

and Bruiijn, 1998], [Remelhe et al., 2004], [Engell et al., 2005], [Akesson and Skoldstam, 2007]. Ces travaux visent la vérification des programmes de l'API par la technique du model-checking. Ils traduisent le programme SFC en un modèle formel, à savoir, l'automate à états finis et l'automate temporisé. L'objectif est d'utiliser les techniques et les outils de ces modèles formels afin de vérifier quelques propriétés du programme SFC.
La transformation d'un programme SFC en automate temporisé nécessite que le programme SFC possède une sémantique claire. Parmi les travaux de recherches définissant une sémantique formelle d'un programme SFC, nous citons [Bauer et al., 2004].

Une transformation d'un programme SFC en automate temporisé a été proposée dans [Remelhe et al., 2004], [Stursberg and Lohmann, 2005], [O. Stursberg and Engell, 2005], [Engell et al., 2005]. L'idée de la procédure de transformation est de diviser le programme SFC en plusieurs unités. Une unité est composée d'une séquence étapes-transitions du programme SFC. Chaque unité est représentée par un automate temporisé. La structure sommet-transition de l'automate est dérivée directement de la séquence étape-transition dans l'unité correspondante du programme SFC. Dans une unité, la réceptivité d'une transition est représentées par les événements (étiquettes) associés aux transitions de l'automate correspondant. Par conséquent, l'étiquette d'une transition de l'automate implique soit des événements en provenance du processus commandé ou des variables internes du programme SFC. Les actions associées aux étapes d'une unité du programme est aussi représentée dans l'automate temporisé correspondant à cette unité. Des horloges et des conditions temporelles (gardes et invariants) sont utilisées dans l'automate temporisé lorsque les qualificatifs des actions dépendent du temps. Les automates temporisés résultant sont synchronisés par les événements.
En effet, les travaux mentionnés ci-dessus visent à vérifier les programmes SFC, donc il est normal qu'ils considèrent aussi le comportement de l'API exécutant les programmes. Dans ce but, le travail présenté par [Remelhe et al., 2004] considère dans la procédure de transformation décrite ci-dessus, le cycle de l'API exécutant le programme SFC. Un

automate temporisé appelé "coordinateur" est construit. Cet automate change l'état à chaque cycle de l'API. Par conséquent, une trajectoire dans l'automate représentant le programme SFC complet contient une séquence des états pour lesquels l'automate appelé coordinateur change son état mais non pas les automates représentant les unités dans le programme SFC. Cela cause un problème dans le modèle-checking par rapport au temps de calcul et à la consommation de mémoire. C'est pourquoi les travaux présentés dans [O. Stursberg and Engell, 2005] et [Engell et al., 2005] considèrent l'aspect comportementale de l'API par un automate temporisé appelé automate de déclenchement. Lorsque ce dernier reçoit un événement du modèle de processus, il émet aux automates temporisés (correspondant aux unités du programme SFC) l'événement de déclenchement associé, dans l'intervalle $[0, T_c]$. La durée T_c représente le temps d'un cycle de l'API. De même dans le cas d'écriture dans le processus, il envoie l'ordre au modèle de processus dans l'intervalle $[0, T_c]$ après la réception de l'ordre des modèles correspondant au programme SFC.

Notre travail se situe dans le sense inverse par rapport aux travaux décrits ci-dessus. Nous traduisons l'automate à chronomètres spécifiant le comportement acceptable du système en programme SFC. Le but de ce dernier est d'implémenter le système de surveillance par un API. Le programme SFC correspondant à l'automate à chronomètres mentionnés ci-dessus représente aussi un sommet par une étape. Une transition dans l'automate peut être représentée aussi par une transition dans le programme SFC. La réceptivité d'une transition du SFC est l'événement de la transition correspondante de l'automate. Nous avons considéré que la durée d'un cycle est assez petite tel que la réception et l'écriture sont instantanées. Les questions posées dans les paragraphes suivants sont : comment les informations temporelles peuvent être représentées dans le SFC ? Est-ce que le comportement du programme SFC résultant se conforme à celui de l'automate à chronomètres ?.

6.3 Passage de l'automate de surveillance au programme de surveillance $\mathbf{SFC}_M$

Nous avons vu dans le chapitre 4, la méthode de synthèse de l'automate à chronomètres qui représente le comportement acceptable d'un système. Cet automate a été noté par $\mathbb{A}^*$. Ce dernier est synchronisé sur les événements provenant du procédé commandé. Il est également temporisé par les horloges internes. Ceci signifie que l'événement associé à une transition peut se produire tant que les valeurs des chronomètres vérifient l'espace temporel associé au sommet source. A son occurrence l'automate bascule vers un autre sommet. Ainsi, cet automate permettra de détecter un défaut suite à la violation de l'espace temporel associé à un sommet. Nous pouvons remarquer que l'automate $\mathbb{A}^*$ ne

peut pas être implémenté directement. Ceci est dû à deux raisons :
- l'absence de l'état fautif dans $\mathbb{A}^*$,
- l'absence des transitions conduisant vers l'état fautif et traduisant le fait de violer l'espace temporel d'un sommet.
Afin de rendre l'automate $\mathbb{A}^*$ implémentable, nous définissons l'automate de surveillance $\mathbb{A}_M$.
Nous nous intéressons à implémenter le système de surveillance en utilisant un API. Donc, nous proposons traduire l'automate de surveillance en programme SFC. Un algorithme de traduction structurelle de l'autmate de surveillance $\mathbb{A}_M$ en programme SFC sera présenté.

6.3.1 Automate de surveillance $\mathbb{A}_M$

L'automate de surveillance $\mathbb{A}_M$ sera obtenu à partir de $\mathbb{A}^*$. La méthode de construction de $\mathbb{A}^*$ a été présentée dans les chapitres 4 et 5. Dans la suite, nous définissons formellement l'automate $\mathbb{A}^*$.

Définition 6.3.1. *L'automate à chronomètres représentant le comportement acceptable d'un système est un 7-uplet* $\mathbb{A}^* = (L^*, l_0^*, X^*, \Sigma^*, A^*, Sp, Div)$ *où :*
- $L^* = \{l_1, ..., l_n\}$ *est un ensemble fini de sommets atteignables et n est leur nombre ;*
- l_0^* *est le sommet initial ;*
- X^* *est un ensemble fini de chronomètres à valeurs réelles positives ;*
- Σ^*[1] *est un ensemble fini d'événements ;*
- A^* *est un ensemble fini de transitions entre* L^*. $a = (l_i, l_j, \sigma, R) \in A^*$ *est une transition liant le sommet* l_i *au sommet* l_j *avec l'événement* σ *et l'ensemble de chronomètres à remettre à zéro R.*
- $Spc : l \rightarrow 2^{\mathbb{P}(x)}$ *est la fonction assignant une combinaisons booléenne des inégalités de X à chaque sommet* $l \in L^*$. *Cette combinaison des propositions indiquée par* $\mathbb{P}(x)(l_i)$ *exprime l'espace synthétisé* E_i^* *dans le sommet* l_i.
- Dif *associe à chaque sommet l'activité des chronomètres dans ce sommet.*

□

L'automate $\mathbb{A}_M$ peut être obtenu de $\mathbb{A}^*$ en ajoutant le sommet l_f représentant l'état de défaut et les transitions conduisant à ce sommet. Les figures 6.3.a,b présentent respectivement les automates $\mathbb{A}_i^*$ et $\mathbb{A}_{M_i}$ d'une tâche interruptible $tâche_i$. Par construction, l'automate $\mathbb{A}_{M_i}$ est obtenu de $\mathbb{A}_i^*$ en ajoutant le sommet l_f qui représente l'état fautif l_f et les transitions $l_1 \xrightarrow{g_{1f}} l_f$ et $l_3 \xrightarrow{g_{2f}} l_1$. Les gardes de ces transitions g_{1f} et g_{2f} impliquent respectivement la négation des espaces temporelles E_2^* et E_3^* (c-à-d : $\overline{E_2^*}$ et $\overline{E_3^*}$).

[1] Σ^* *représente alphabétique associé à l'automate* $\mathbb{A}^*$ *et non pas un ensemble de mots.*

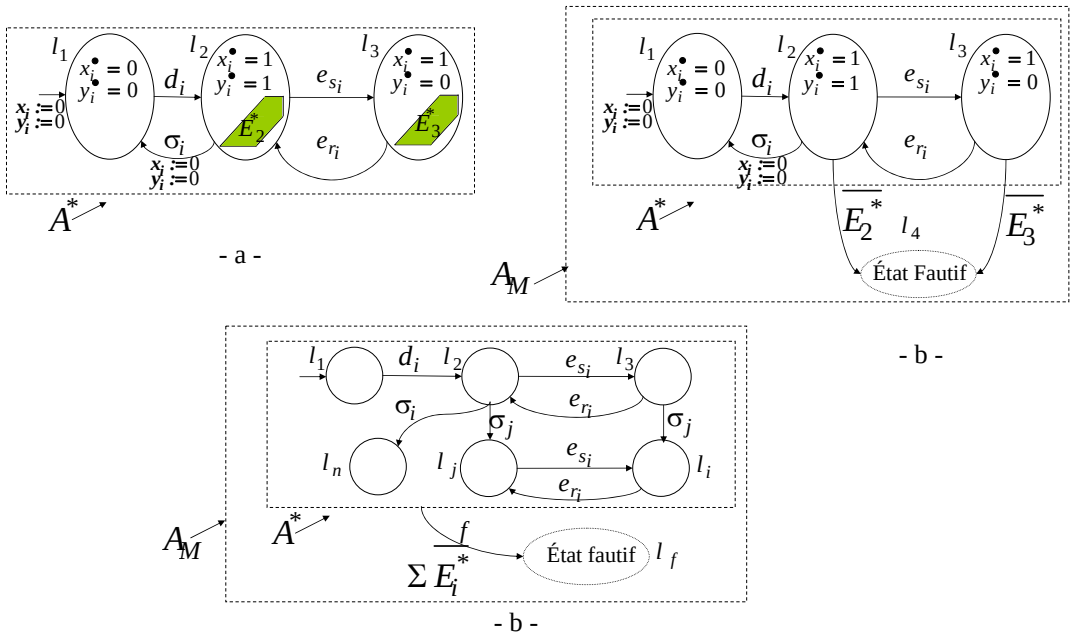

FIG. 6.3 – a- L'automate représentant le comportement acceptable d'une tâche interruptible. b- L'automate de surveillance d'une tâche interruptible c- L'automate de surveillance d'un système susceptible aux défauts interruptibles.

Formellement, l'automate de surveillance $\mathbb{A}_M$ peut être défini par la suivante :

Définition 6.3.2. *Soit* $\mathbb{A}^* = (L^*, l_0^*, X^*, \Sigma^*, A^*, Sp, Div)$ *l'automate représentant le comportement acceptable de système, défini par la definition 6.3.1. L'automate de surveillance est défini par :* $\mathbb{A}_M = (L_m, l_0^*, X^*, \Sigma_m = \Sigma^* \cup f, A_m, Sp, Div)$. *Dans cet automate,* $L_m = L^* \cup l_f$ *où* l_f *est la sommet fautif,* f *est l'événement représentant le défaut et* $A_m = A^* \cup A_f$. $\forall l_i \in L^*$ *:* $\exists a = (l_i, l_f, f, g_f) \in A_f$ *où* l_i, l_f *sont respectivement leurs sommets de source et de destination et* $g_f = \overline{E_i^*}$ *est leur garde.*

□

Propriété 6. Par construction, $\mathbb{A}_M$ a la propriété suivante :

$$\forall (l_i, \upsilon_i) \notin (l_i, E_i^*) : \exists a = (l_i, l_f, f, g_f) \in A_f \text{ tel que } \upsilon_i \vDash g_f$$

□

Cette propriété signifie que la transition $a \in A^*$ peut être franchie ssi $\forall \upsilon_i : \upsilon_i \in E_i^*$. Autrement dit : $a \in A^*$ peut être franchie tant que $\upsilon_i \vDash \mathbb{P}(x)(l_i)$. Si $\upsilon_i \notin E_i^*$ (c'est-à-dire : $\upsilon_i \nvDash \mathbb{P}(x)(l_i)$ ou $\upsilon_i \vDash \overline{\mathbb{P}(x)(l_i)}$), il existe une transition $a \in A_f$ qui est franchie immédiatement.
La figure 6.3.c présente l'automate de surveillance $\mathbb{A}_M$ d'un système.

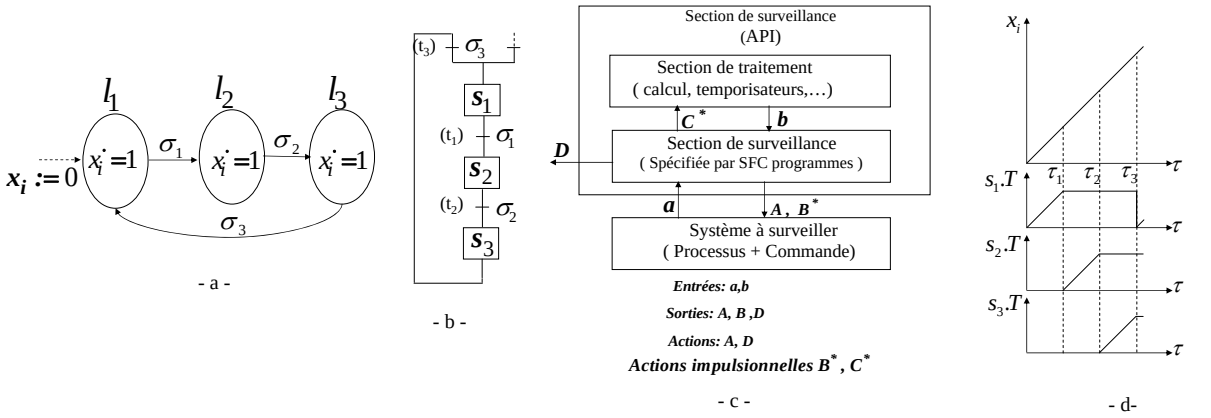

FIG. 6.4 – a- Une partie de l'automate de surveillance b- Programme SFC correspondant c- Automate Programmable Industriel d- Évolution des variables temporelles associées aux étapes.

6.3.2 Programme de surveillance SFC$_M$

Dans cette section, nous montrons comment construire le programme de surveillance SFC$_M$ à partir de l'automate de surveillance $\mathbb{A}_M$. Dans la construction de SFC$_M$, nous présentons le principe du passage de $\mathbb{A}_M$ en SFC$_M$. Nous considérons aussi les problèmes de la représentation des chronomètres par des variables temporelles associées aux étapes, de la surveillance de l'espace temporel et des sorties de SFC$_M$.

A Principe de traduction de l'automate de surveillance $\mathbb{A}_M$ en programme de surveillance SFC$_M$

L'idée principale de traduction de l'automate de surveillance $\mathbb{A}_M$ en programme SFC de surveillance est la suivante :

- Chaque sommet de l'autmate est représenté par une étape. Soit $L_m = \{l_1, ..., l_{n+1}\}$ l'ensemble de sommets de $\mathbb{A}_M$. L'ensemble d'étapes correspondant à ces sommets est $\{s_1, ..., s_{n+1}\}$.
- Chaque transition entre deux sommets est représentée par une transition liant les deux étapes correspondant à ces deux sommets.
- La réceptivité d'une transition est l'événement associé à la transition correspondante de l'automate.

Afin de considérer le comportement temporel de l'automate dans le SFC$_M$, nous utilisons les variables temporelles associées aux étapes de SFC$_M$. Cependant, le problème réside dans le fait que le chronomètre dans l'automate $\mathbb{A}_M$ dépasse les possibilités offertes par ces variables. Afin d'expliquer ce problème, considérons la partie de l'automate $\mathbb{A}_M$ présentée dans la figure 6.4.a. Le programme SFC correspondant à cet automate est présenté

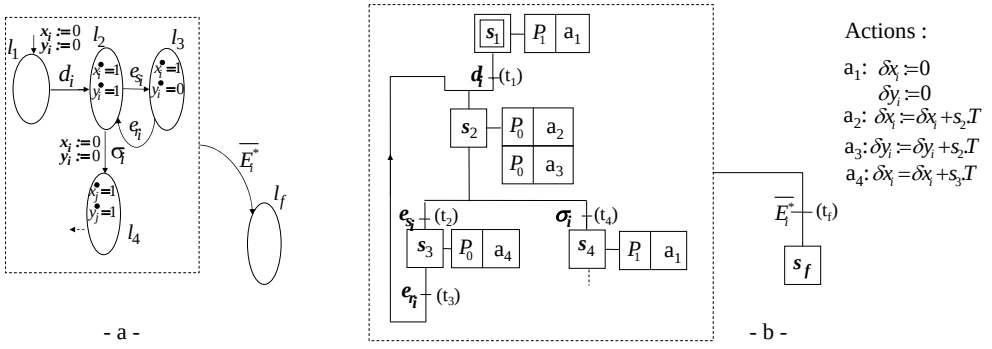

FIG. 6.5 – a- Partie de l'automate de surveillance b- Programme de surveillance SFC_M.

dans la figure 6.4.b. Les étapes s_1, s_2, s_3 correspondent aux sommets l_1, l_2 et l_3. Les transitions t_1, t_2 et t_3 correspondent aux transitions entre ces sommets. La réceptivité d'une transition est l'événement de la transition correspondante. Les événements (réceptivités des transitions du SFC_M) représentent l'entrée a de l'API (Fig. 6.4.c). Les variables $s_1.T$, $s_2.T$ et $s_3.T$ représentent respectivement le temps de séjour dans les étapes : s_1, s_2 et s_3. Supposons qu'on veuille suivre les évolutions de ces variables pour le scénario de franchissement suivant : $s_1 \xrightarrow{\sigma_1}_{\tau_1} s_2 \xrightarrow{\sigma_2}_{\tau_2} s_3 \xrightarrow{\sigma_3}_{\tau_3} s_1$. La variable associée à l'étape s_1 avance avec le passage de temps jusqu'à l'instant de passage de s_1 à s_2. A cet instant, la variable associée à s_1 s'arrête et la variable associée à s_2 commence à compter le temps et ainsi de suite. Les évolutions des variables $s_1.T$, $s_2.T$ et $s_3.T$ sont présentées dans la figure 6.4.d. Dans l'automate, le chronomètre x_i est actif dans les sommets l_1, l_2 et l_3. Donc la valeur de x_i avance avec le passage de temps. Lorsque l'événement σ_1 se produit l'automate évolue du sommet l_1 au l_2. Ce franchissement ne change pas la valeur de x_i. La figure 6.4.d montre l'évolution temporelle du chronomètre x_i. Nous remarquons qu'un chronomètre ne peut pas être représenté seulement par les variables associées aux étapes. Afin de résoudre ce problème, nous proposons d'utiliser avec chaque chronomètre une variable intermédiaire et d'associer aux étapes des actions.

B Représentation des chronomètres de $\mathbb{A}_M$ dans le SFC_M

Sans perte de généralité, considérons l'automate présenté dans la figure 6.5.a. Le programme SFC lui correspondant est composé des étapes s_1, s_2, s_3 et s_4. Le sommet fautif l_f est représenté par l'étape s_f. Il contient aussi les transitions t_1, t_2, t_3 et t_4 ayant respectivement les réceptivités d_i, e_{s_i}, e_{r_i} et σ_i. La représentation du chronomètre x_i dans le programme de surveillance se fait en basant sur le fait que la valeur du chronomètre x_i à un instant donné dans le sommet, (l_2 ou l_3), est la somme de deux durées :

1. la durée représentant la valeur de x_i à l'entrée du sommet ;
2. le temps écoulé depuis l'instant d'entrée dans le sommet jusqu'à l'instant actuel.

Il est clair que la deuxième durée correspond à la valeur de la variable temporelle associée à l'étape active (c'est-à-dire : $s_2.T$ ou $s_3.T$). Afin d'exprimer la première durée, une variable intermédiaire est introduite dans le programme de surveillance. Cette variable est notée δ_{x_i} et appelée "*variable de décalage*". Par conséquent, la valeur de x_i à chaque instant dans le sommet l_2 (respectivement l_3) correspond à la valeur de : $s_2.T + \delta_{x_i}$ (respectivement $s_3.T + \delta_{x_i}$) dans le programme de surveillance. La question qu'on peut relever est : de savoir comment on peut avoir les valeurs de x_i aux instants de franchissements ? Autrement dit : comment on peut calculer la valeur de δ_{x_i} correspondant à x_i aux instants de franchissements entre les sommets l_2 et l_3 ?.
La variable δ_{x_i} est initialisée lorsque le chronomètre x_i est mis à zéro dans l'automate. Supposons que l'automate suit le scénario suivant : $l_2 \xrightarrow{e_{s_i}}_{\tau_1} l_3 \xrightarrow{e_{r_i}}_{\tau_2} l_2 \xrightarrow{e_{s_i}}_{\tau_3} l_3$. Dans ce scénario, τ_1, τ_2 et τ_3 sont les instants de franchissement des transitions. Le programme SFC_M suit ce scénario et il bascule entre les étapes s_2, s_3 et s_2. La figure 6.6.a montre les chronogrammes représentant l'évolution des variables booléennes et temporelles associées aux étapes s_2 et s_3.
Pour que le programme de surveillance SFC_M puisse simuler le comportement temporel de l'automate considéré, il doit calculer les valeurs $x_i(\tau_1)$, $x_i(\tau_2)$ et $x_i(\tau_3)$ (Fig. 6.6.b). La figure 6.5.b présente le programme proposé qui calcule ces valeurs. Nous expliquons les actions de ce programme ci-dessous
• A l'entrée de l_1, la valeur de x_i est zéro, donc à l'étape s_1 sera associée à une action a_1 qui initialise la variable δ_{x_i}. Cette action aura lieu à l'activation de s_1.
• L'action a_2 associée à l'étape s_2 calcule la valeur de x_i à l'instant de franchissement de s_2 à s_3 (c-à-d : $x_i(\tau_1)$). Cette action aura lieu à la désactivation de s_2. Elle met à jour la variable δ_{x_i} en ajoutant la durée de séjour dans l'étape s_2.
• L'action a_4 associée à l'étape s_3 calcule la valeur de x_i à l'instant de franchissement de s_3 à s_2 (c-à-d : $x_i(\tau_2)$).
De même, nous trouvons l'action a_2 qui calcule la valeur $x_i(\tau_3)$. Le chronogramme représentant l'évolution de la variable δ_{x_i} est donné dans la figure 6.6.c. La valeur de δ_{x_i} dans les étapes s_2 et s_3 correspond à la valeur de x_i, à chaque fois l'automate atteint le sommet l_2 ou l_3. Par conséquent, la somme de δ_{x_i} et la variable temporelle associée à l'étape active ($s_2.T$ ou $s_3.T$), correspond à la valeur de x_i. Ceci est vrai à n'importe quel instant pendant l'évolution de l'automate dans les sommets l_2 et l_3 (Fig. 6.6.d).

Le chronogramme qui représente l'évolution du chronomètre y_i est présenté dans la figure 6.6.e. Le programme calculera les valeurs $y_1(\tau_1)$ et $y_1(\tau_3)$. Les actions du programme sont :
• L'action a_1 initialise le chronomètre y_i à l'entrée du sommet l_1.
• L'action a_3 associée à l'étape s_2 calcule la valeur de y_i à l'instant de franchissement de s_2 à s_3 (c-à-d : $y_i(\tau_1)$). Cette action aura lieu à la désactivation de s_2. Elle met à jour la variable δ_{y_i} en ajoutant la durée de séjour dans l'étape s_2.

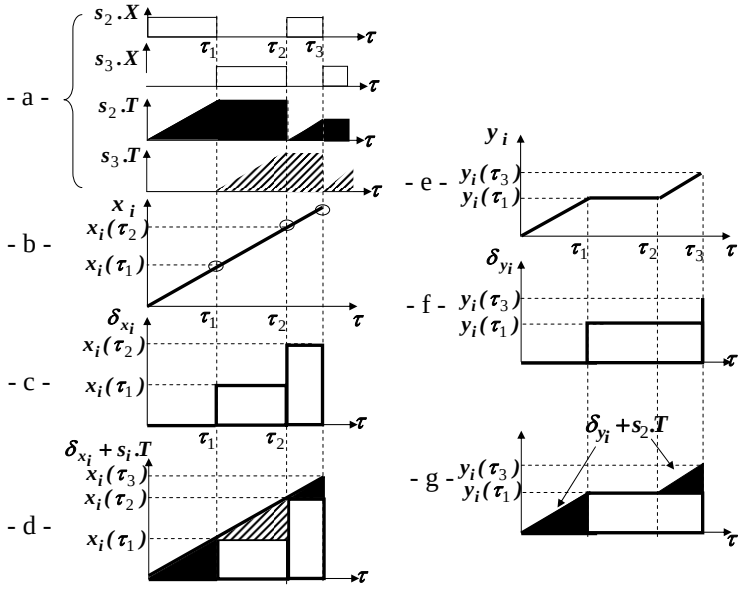

FIG. 6.6 – Représentation du comportement des chronomètres dans un programme SFC

Il est à noter que la dynamique de y_i est zéro au sommet l_3, par conséquent $y_i(\tau_1) = y_i(\tau_2)$. Pour cela, le programme SFC_M ne met pas à jour la variable δ_{y_i} à l'instant τ_2. Le chronogramme 6.6.f montre l'évolution de δ_{y_i}. La représentation de la valeur du chronomètre y_i dans le SFC_M dépend de sa dynamique dans l'automate (Fig. 6.6.g). Si le chronomètre y_i est actif (dans le sommet l_2), donc la valeur de y_i correspond à la somme de $\delta_{y_i} + s_2.T$. Si le chronomètre est inactif (dans le sommet l_3), la variable δ_{y_i} correspond seule à y_i. Les actions a_1, a_2, a_3 et a_4 de SFC_M représentent l'action C^* dans la figure 6.4.c.

C Initialisation des chronomètres

Dans la figure 6.5.a, les chronomètres x_i et y_i sont initialisés par le franchissement de la transition $l_2 \rightarrow l_4$. Ce fait sera représenté dans le programme de surveillance SFC_M par l'initialisation des variables de décalage qui leur correspondent (c'est-à-dire : les variables δ_{x_i} et δ_{y_i}). L'action initialisant ces variables est associée à l'étape correspondant au sommet de destination de la transition de l'automate. Autrement dit : le programme initialise δ_{x_i} et δ_{y_i} à l'instant d'entrée à l'étape s_4.
Les valeurs des chronomètres à l'état initial de l'automate ont une valeur nulle. Donc, l'étape initiale du SFC_M sera associée une action qui initialise toutes les variables de décalage utilisées dans le programme.
Nous pouvons remarquer que l'initialisation des variables de décalage se fait par les étapes et non pas par les transitions. Autrement dit : la variable de décalage concernée par l'initialisation est mise à zéro à l'instant d'entrée à l'étape de destination de la transition.

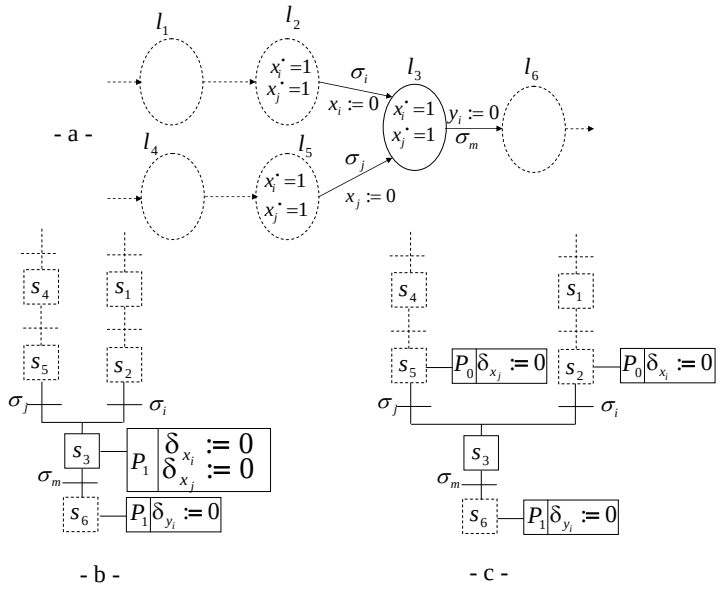

FIG. 6.7 – Premier cas complex d'initialisation des chronomètres.

Considérons le cas présenté dans la figure 6.7.a : un sommet dans l'automate de surveillance a deux transitions à son entrée. Les chronomètres initialisés par ces deux transition sont différents. De l'application de la règle de traduction donnée ci-dessus il résulte le programme mentionné dans la figure 6.7.b. L'activation de l'étape s_3 correspondant au sommet l_3 met à zéro les variables de décalage correspondant aux chronomètres initialisés par les deux transitions. L'initialisation est faite sans la considération de la manière dont le programme atteint cette étape. Cela représente une erreur de traduction. Afin de résoudre ce problème, nous proposons d'initialiser les variables des décalages à la désactivation de l'étape source. La variable δ_{x_i} (respectivement δ_{x_j}) est initialisée à l'instant désactivation l'étape s_2 (respectivement l'étape s_5).
Une autre cas envisageable dans l'automate de surveillance celui présenté dans la figure 6.8.a. Dans ce cas, l'application de la proposition décrite dans la figure 6.7.c donne le programme présenté dans la figure 6.8.b. Ce programme initialise la variable de décalage δ_{x_i} à chaque fois l'étape s_5 est désactivée, y compris le cas de désactivation par le franchissement de transition $s_5 \rightarrow s_4$. Cela représente un erreur de traduction. Dans ce cas, nous proposons de réagir sur l'automate et de doubler le sommet l_3 de la manière présentée dans la figure 6.8.c. Le programme SFC_M équivalent est montré dans la figure 6.8.d .

D Surveillance de l'espace temporel

L'automate détecte un défaut lorsque les valeurs des chronomètres vérifient la garde d'une transition $a_f \in A_f$. Cette garde implique la négation de l'espace temporel

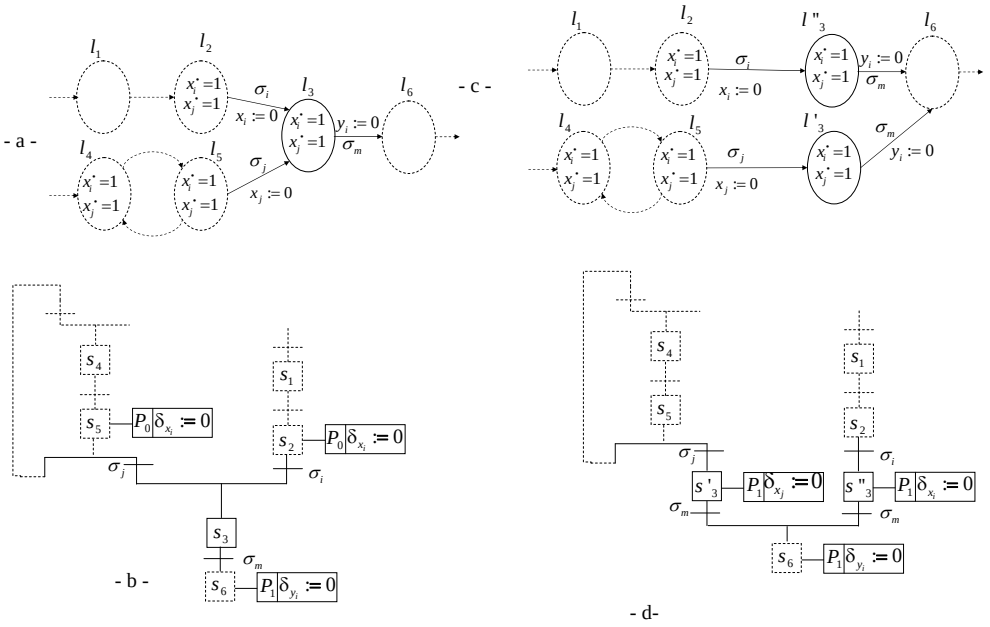

FIG. 6.8 – Deuxième cas complexe d'initialisation des chronomètres

associant au sommet source. La traduction structurelle de l'automate $\mathbb{A}_M$ (fig. 6.5.a) donne le programme SFC_M présenté dans la figure 6.5.b. Dans ce dernier, une étape de défaut s_f correspond au sommet de défaut l_f. Il contient aussi des transitions correspondant aux arcs de défauts A_f. Les réceptivités de ces transitions impliquent la négation des espaces temporels associant aux étapes de source.

Afin de simplifier le programme, nous avons proposé d'utiliser le concept de l'hiérarchie. Donc, nous proposons de remplacer SFC_M par deux programme : esclave SFC_1 et maître SFC_2. Le programme SFC_1 sera obtenu en suivant la démarche que nous venons d'expliquer. L'étape initiale s_{11} de SFC_2 signifie que le fonctionnement est acceptable et l'étape s_{12} correspond à l'état fautif l_f. La transition t_{21} sera franchie si un espace temporel est violé. La réceptivité de cette transition implique l'union de la négation de tous les espaces temporels associés aux sommets de l'automate de surveillance.
Supposons que E_1^*, E_2^*,...,E_i^*,...,E_n^* les espaces temporels dans les sommets l_1, l_2,...,l_i,...,l_n. Les étapes correspondantes dans le programme SFC_1 sont s_1, s_2,...,s_i,...,s_n. La réceptivité de la transition t_{21} du programme SFC_2 présenté dans la figure 6.9.b est : [($s_1.X$ et $\overline{E_1^*}$) ou ($s_2.X$ et $\overline{E_2^*}$) ou .... ou ($s_i.X$ et $\overline{E_i^*}$) ou ...($s_n.X$ et $\overline{E_n^*}$)].
Le prédicat de cette réceptivité représente l'entrée b dans la figure 6.4.b.
Supposons que $a_i \leq x_i - y_i < b_i$ est une des inégalités délimitant E_i^* dans le sommet l_i où a_i, b_i sont des constantes. Cette inégalité peut être exprimée dans la réceptivité de la transition t_{21} selon les dynamiques des chronomètres x_i et y_i dans le sommet l_i comme le suivant :

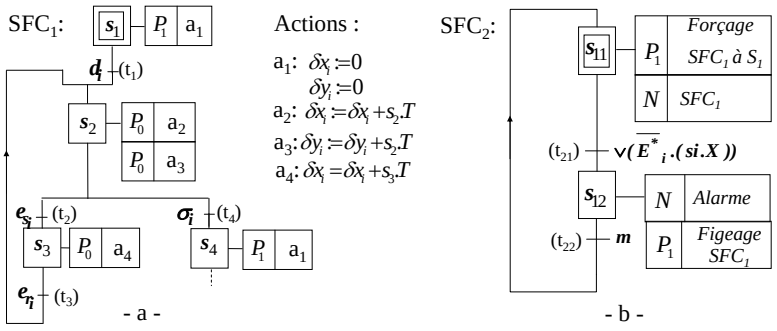

FIG. 6.9 – a- SFC_1 esclave b- SFC_2 maître

• les dynamiques $\dot{x}_i = \dot{y}_i = 1 : a_i > (\delta_{x_i} + s_i.T) - (\delta_{y_i} + s_i.T) \geq b_i$,
• les dynamiques $\dot{x}_i = 1$ et $\dot{y}_i = 0 : a_i > (\delta_{x_i} + s_i.T) - (\delta_{y_i}) \geq b_i$.

E Sorties du programme SFC de surveillance

Nous avons vu dans le chapitre 2 que la surveillance des procédés industriels consiste à générer des alarmes à partir des informations délivrées par des capteurs. Elle est limitée aux fonctions qui collectent des informations et font des inférences, sans agir réellement, ni sur le procédé, ni sur le système de commande. Les travaux présentés dans [Nourelfath, 2000] considèrent le problème de réactivité aux défauts. Lorsqu'un défaut est détecté, le fonctionnement normal est figé et remplacé par un autre type de fonctionnement appelé "dégradé". Ce dernier reste actif jusqu'à l'occurrence de recouvrement. Ces actions de réactivité sont représentées par les actions A et B^* dans la figure 6.4.b

Nous voulons construire un programme de surveillance qui prend en compte la réactivité aux défauts. Cette dernière considérée ici représente notre choix. D'autres procédures peuvent être considérées. L'effet de réactivité sur le système de surveillance sera considéré en associant les étapes de SFC_2 par des actions. En effet, lorsqu'un défaut survient il demande un arrêt complet du système. La procédure de remise en route consiste très souvent à réinitialiser la commande, la partie opérative et également le système de surveillance. Afin de prendre en compte l'effet de cette réactivité sur notre système de surveillance, à l'étape de défaut s_{12} dans le programme SFC_2 seront associées les actions suivantes :
• Le déclenchement d'une alarme. Dans la figure 6.4.c représentant la structure de l'API, cette action est représentée par l'action D. Le qualificatif de cette action est N (action continue).
• Le figeage du programme SFC_1 dans sa situation actuelle. Le qualificatif de cette action est P_1 (impulsionnelle à l'activation de l'étape s_{12}).

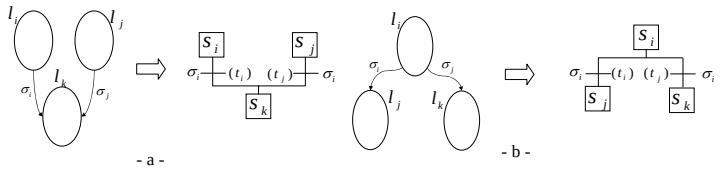

FIG. 6.10 – Configurations des transitions dans le programme SFC_1 : a- Convergence en ou b- Divergence en ou

A l'étape initiale s_{11} de SFC_2 sera associée une action qui force SFC_1 à activer l'étape initiale s_1. Cette action implusionnelle aura lieu lorsque l'événement m représentant la fin de l'opération de réparation, se produit.

6.3.3 Traduction structurelle de l'automate de surveillance $\mathbb{A}_M$ en programme de surveillance SFC_M

Le programme SFC_2 présenté dans la figure 6.9.b, surveille en permanence la concordance entre les valeurs des chronomètres et l'espace temporel représentant le comportement acceptable. Ce programme est générique. C'est seulement la réceptivité de la transition t_{21} qui dépend du système à surveiller. La programmation de cette réceptivité est expliquée ci-dessus. Le programme SFC_1 résulte de la traduction structurelle de $\mathbb{A}^*$. Avant de présenter les règles de traduction, nous présentons les notations suivantes : soit t_i une transition dans le programme SFC_1. Les notations ${}^\bullet(t_i)$ et $(t_i)^\bullet$ représentent respectivement l'étapes à l'entrée et à la sortie de la transition t_i dans le programme SFC_1.
Les règles de traduction de $\mathbb{A}^*$ en programme SFC_1 sont les suivantes.

1. S'il y a une structure telle que décrite dans la figure 6.8.a , appliquer à l'automate $\mathbb{A}^*$ la transformation présentée dans la figure 6.8.c.
2. Créer une étape en correspondance à chaque sommet de $\mathbb{A}^*$.
3. Créer une transition en correspondance à chaque transition (arc) de l'automate. Soit $a_i = (l_i, l_j, \sigma_i, R(X_i)) \in A^*$ une transition de l'automate. Il existe une transition correspondante dans le SFC_1 telle que :
   – sa réceptivité est l'événement de l'arc, c'est-à-dire : σ_i,
   – ${}^\bullet(t_i)= s_i$ et $(t_i)^\bullet=s_j$ où s_i et s_j sont les étapes correspondant aux l_i et l_j.
   – Ajouter à l'étape $s_j = (t_i)^\bullet$ une action ayant le qualificatif P_1. Cette action met à zéro les variables de décalage correspondant aux chronomètres dans l'ensemble $R(X_i)$. Si les chronomètres initialisés par a_i vérifie le cas d'initialisation décrit dans la figure 6.7.a , appliquer la transformation présentée dans la figure 6.7.c. Autrement dit : ajouter à l'étape $s_i = {}^\bullet(t_i)$ une action ayant le qualificatif P_0.

Cette action met à zéro les variables de décalage correspondant aux chronomètres dans l'ensemble $R(X_i)$.

– Soit x_i un chronomètre ayant la dynamique $\dot{x_i} = 1$ dans l_i. Ajouter à l'étape $s_i = {}^\bullet(t_i)$ l'action $\delta_{x_i} = \delta_{x_i} + s_i.T$. Le qualificatif de cette actio est P_0.

4. Soit t_i et t_j deux transitions dans SFC_1 ayant la même étape à ses sorties, c'est-à-dire : $s_k = (t_i)^\bullet = (t_j)^\bullet$. Soit $s_i = {}^\bullet(t_i)$ et $s_j = {}^\bullet(t_j)$ les étapes à l'entrée de t_i et t_j. Les étapes s_i, s_j et s_k sont liées par une jonction de type : "Convergence en ou" (Fig. 6.10.a).
5. Soit t_i et t_j deux transitions dans SFC_1 ayant la même étape à ses entrées, c'est-à-dire : $s_i = {}^\bullet(t_i) = {}^\bullet(t_j)$, Soit $s_j = (t_i)^\bullet$ et $s_k = (t_j)^\bullet$ les étapes à la sortie de t_i et t_j. Les étapes s_i, s_j et s_k sont liées par une jonction de type : "Divergence en ou" (Fig. 6.10.b).

6.4 Validation expérimentale de l'approche de surveillance

Afin de montrer les avantages et les performances de l'approche de surveillance proposée dans ce mémoire, nous allons la valider expérimentalement. Dans ce but, nous étudierons un processus industriel plus complexe. L'objectif de cet exemple est d'une part, de montrer l'applicabilité de l'approche de surveillance en considérant un autre domaine d'application. D'autre part, cet exemple diffère de l'exemple du système manufacturier traité dans les chapitres précédents du fait qu'il contient plusieurs tâches qui s'exécutent en parallèle.

Les validations expérimentales se font à partir de l'automate de surveillance et du programme SFC correspondant :

• La première validation est de simuler l'automate de surveillance de l'application considérée. L'outil de simulation est basé sur les Stateflow et Checkmate dans l'environment Matlab. Nous générons des signaux binaires représentant les événements du procédé et sa commande. Ces signaux sont envoyés à l'automate de surveillance codé par Checkmate et Stateflow de Matlab. L'automate évolue en réagissant à ces événements. L'information sur la décision de système de surveillance, calculée par cet outil, permet de prendre une décision soit sur le comportement acceptable, soit sur la déclaration d'un défaut.

• La deuxième validation est d'utiliser le programme de surveillance développé sous SFC dans un API avec Matlab. En effet, tout code Matlab peut être considéré comme client OPC. Une connexion à un serveur OPC[2] à partir de Matlab permet alors

[2]La spécification OPC est une spécification technique qui définit un jeu d'interfaces standard basées

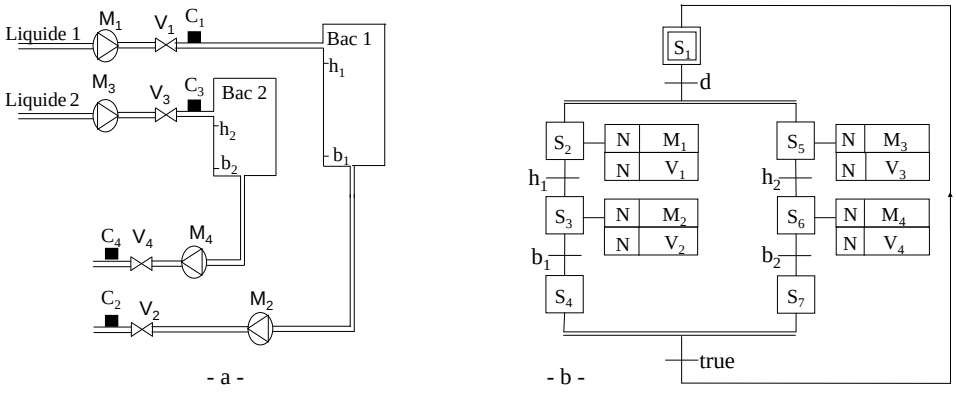

FIG. 6.11 – a- Le système d'alimenatation en liquide b- Le système de commande du processus.

d'accéder aux variables de l'API et d'effectuer une lecture/écriture sur les différentes variables. Ainsi, les événements générés par Matlab (représentant des scénarios de fonctionnement) peuvent être envoyés au module de surveillance codé par SFC dans l'automate. Les informations sur la décision de surveillance, calculées par cet outil, peuvent être récupérées de l'API. Par conséquent, nous pouvons visualiser la décision du programme de surveillance sur le comportement de système (acceptable/déclaration d'un défaut).
En effet, nous allons appliquer les mêmes scénarios de fonctionnement (séquences de signaux binaires) dans les deux validations. Cela nous permet de juger la correction de traduction de l'automate de surveillance en programme SFC de manière expérimentale.

Dans la suite de cette section, nous présentons, d'abord, l'application considérée. Ensuite, l'automate de surveillance et son programme SFC correspondant seront construits. Enfin, la validation expérimentale sera présentée.

6.4.1 Présentation de l'exemple

Un système d'alimentation en liquide (Fig. 6.11.a) se compose de deux bacs, 4 pompes et 4 vannes. Ce système est conçu afin de remplir et de vider les deux bacs 1 et 2. La pompe M_1 envoie la liquide 1 à travers la vanne V_1 dans le bac 1. Le vidage de ce bac se fait en utilisant la pompe M_2 à travers la vanne V_2. La pompe et la vanne peuvent être respectivement en Marche/Arrêt ou Ouvert/Fermé. La pompe M_1 remplit le bac 1 jusqu'à ce que le liquide atteigne le niveau h_1, c-à-d : l'événement h_1 se produit par le

sur la technologie OLE/COM de Microsoft. Il y a plusieurs specifications OPC comme OPC Data Access. Il est utilisé pour déplacer des données en temps réel d'automates vers d'autres clients (pare exemple : Matlab). Le serveur utilisé est OFS de Schneider-Electric.

La tâche	remplissage bac 1	vidage bac 1	remplissage bac 2	vidage bac 2
La durée d'exécution exacte $[\alpha_i, \beta_i]$ (u.t)	[3,4]	[5,6]	[4,5]	[3,5]
La durée d'exécution tolérée $[\alpha_i, \gamma_i)$ (u.t)	[3,5)	[5,8)	[4,7)	[3,6)
L'événement représentant le début	d	h_1	d	h_2
L'événement représentant la fin	h_1	b_1	h_2	b_2
L'événement représentant l'interruption	s_{h_1}	s_{b_1}	s_{h_2}	s_{b_2}
L'événement représentant la reprise	r_{h_1}	r_{b_1}	r_{h_2}	r_{b_2}

TAB. 6.1 – Les informations nécessaires pour construire le système de surveillance.

capteur placé à ce niveau. A ce moment, M_2 et V_2 commencent la tâche de vidage du bac 1. Cette tâche se termine lorsque la liquide atteint le niveau b_1. A ce moment, la pompe M_2 et la valve V_2 s'arrêtent. De même, les tâches de remplissage et de vidage de bac 2 sont exécutées en utilisant les pompes M_3 et M_4 à travers les valves V_3 et V_4. La figure 6.11.b présente le système de commande du procédé considéré. L'événement d représente l'ordre déclenchant les tâches de remplissage des deux bac exécutées par les pompes M_1 et M_3 et les vannes V_1 et V_3. Cet événement d peut être généré automatiquement dans l'intervalle $[0, 2]$ $u.t$ après l'activation de l'étape initiale s_1 (Fig. 6.11.b). Il peut être généré aussi manuellement par l'opérateur dans l'intervalle mentionné.
Les informations nécessaires pour construire le système de surveillance sont présentées dans le tableau 6.1. Les capteurs logiques C_1, C_2, C_3 et C_4 présentés dans la figure 6.11.a et placés en série avec les pompes et les vannes, indiquent la présence ou l'absence d'écoulement du liquide. Les sorties de C_1 correspondent aux événements s_{h_1} et r_{h_1}. Ces événements indiquent respectivement l'interruption et la reprise de la tâche de remplissage bac 1. De même, les sorties des autres capteurs sont données dans le tableau 6.1.

6.4.2 Construction du système de surveillance

La construction du système de surveillance nécessite de modéliser le procédé par un automate à chronomètres. Ensuite, la procédure de synthèse sera appliquée à cet automate afin d'en déduire l'automate représentant le comportement acceptable du système. Dans les chapitre précédents, nous avons vu que le modèle du système peut être obtenu soit par la composition synchrone des modèles d'automate à chronomètres des différentes tâches ou par la traduction du modèle RdP à chronomètres représentant le comportement du procédé. En ce qui concerne notre application, nous avons choisi de modéliser le comportement du procédé par un RdP à chronomètres. Le RdP permet de modéliser de manière simple et lisible des systèmes complexes.

A Modélisation du comportement du système par un RdP à chronomètres

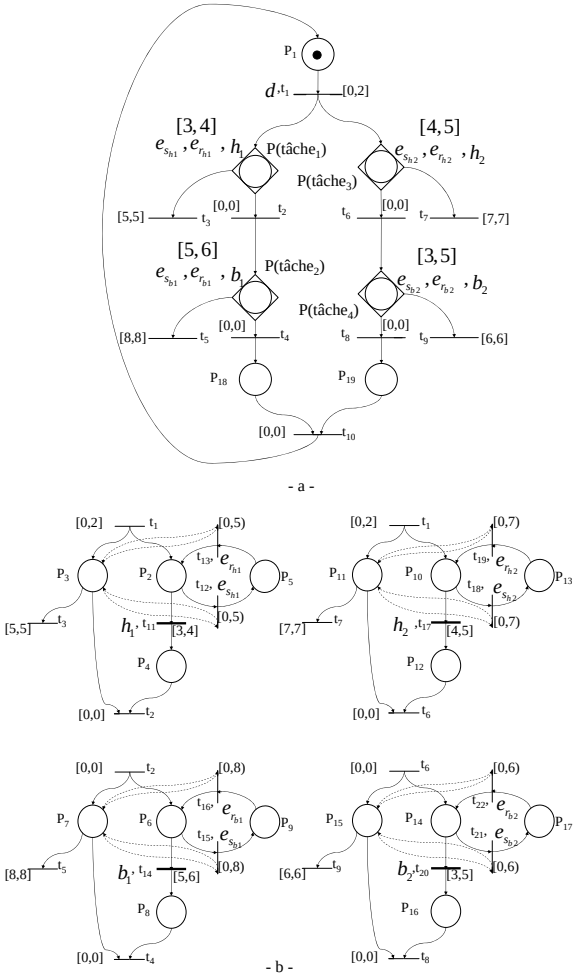

FIG. 6.12 – a- Le modèle RdP à chronomètres du système considéré b- Les expansions des tâches $Tâche_1$, $Tâche_2$, $Tâche_3$ et $Tâche_4$.

La figure 6.12 présente le modèle RdP à chronomètres représentant le comportement de procédé sujet aux défauts interruptibles. Les expansions des tâches sont aussi présentées dans la même figure. Les tâches $Tâche_1$ et $Tâche_2$ corespondent respectivement aux tâches de remplissage et vidage du bac 1. Par ailleurs, les tâches $Tâche_3$ et $Tâche_4$ corespondent respectivement aux tâches de remplissage et vidage de bac 2. Le tableau 6.2 donne les appellations des chronomètres associés aux transitions de modèle RdP à chronomètres présenté dans la figure 6.12.

Transition	Chronomètre	Transition	Chronomètre	Transition	Chronomètre
t_1	x_1	t_3	x_2	t_{11}	y_2
t_5	x_3	t_{14}	y_3	t_7	x_4
t_{17}	y_4	t_9	x_5	t_{20}	y_5

TAB. 6.2 – Chronomètres associés aux transitions de modèle RdP à chronomètres présenté dans la figure 6.12.

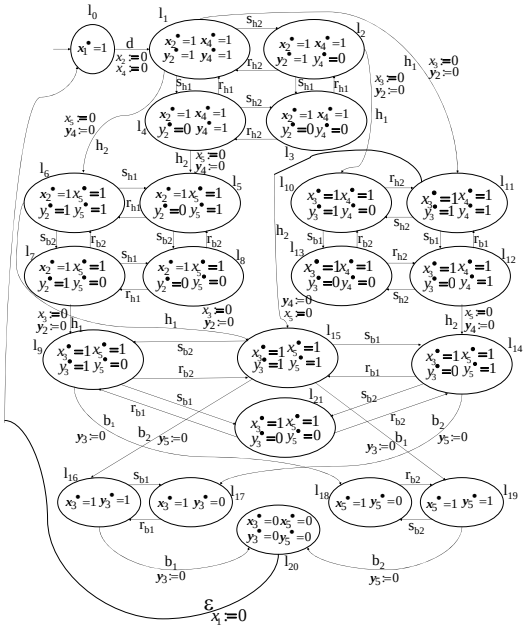

FIG. 6.13 – L'automate représentant le comportement acceptable du procédé considéré.

B Construction de l'automate de surveillance

Nous présentons dans la suite les étapes nécessaires afin de construire l'automate de surveillance :

1. Construire l'automate à chronomètres correspondant au modèle RdP à chronomètres.
   - Calculer le graphe des marquages du modèle RdP à chronomètres du procédé présenté dans la figure 6.12.
   - Appliquer la procédure de construction présentée dans le paragraphe 5.5.3.B à ce graphe afin d'avoir l'automate correspondant au modèle RdP à chronomètres.
2. Appliquer la procédure de synthèse présentée dans le paragraphe 4.3.2.B à l'automate résultant de l'étape précédente afin de déduire l'automate représentant le comportement acceptable du procédé.

Il est important de noter que, pendant la construction du graphe des marquages et l'automate à chronomètres, les points suivants sont considérés :
• L'évolution de l'automate a été exprimée seulement par les chronomètres mesurant les durées effectives et totales des tâches. Dans ce modèle RdP à chronomètres, nous avons les ensembles suivants des transitions :
- l'ensemble des transitions ayant des événements exprimant l'interruption et la reprise des tâches est : $T_{re} = \{t_{12}, t_{13}, t_{15}, t_{16}, t_{18}, t_{19}, t_{21}, t_{22}\}$.
- l'ensemble des transitions ayant des événements exprimant la fin des tâches est $T_{eff} = \{t_{11}, t_{14}, t_{17}, t_{20}\}$.
- l'ensemble des transitions qui mesurent les durées des exécutions totales des différentes tâches est $T_{tot} = \{t_3, t_5, t_7, t_9\}$.
• Les marquages non-stables ont été supprimés. Ces marquages sont atteignables par le franchissement des transitions t_{11}, t_{14}, t_{17} et t_{20}. Par conséquent, le franchissement de la transition ayant l'événement h_1 (respectivement h_2) correspond aux franchissements successifs de t_{11} et t_2 (respectivement t_{17} et t_6). Ainsi, le franchissement de la transition ayant l'événement b_1 (respectivement b_2) correspond aux franchissements successifs de t_{14} et t_4 (respectivement t_{20} et t_8) dans le modèle RdP à chronomètres.
• Le franchissement de la transition mesurant la durée totale et l'arc correspondant sont supprimés au niveau de chaque tâche. Cela permettra de définir un état de défaut global et unique.

L'automate à chronomètres correspondant au modèle RdP à chronomètres contient 26 sommets. L'application de la procédure de synthèse présentée dans le paragraphe 4.3.2.B à cet automate aboutit à ce que les sommets correspondant aux marquages (P_2, P_3, P_{19}), (P_3, P_5, P_{19}), (P_{10}, P_{11}, P_{18}) et (P_{13}, P_{11}, P_{18}) ne sont pas atteignables. Ceci est dû aux contraintes temporelles. Ces sommet correspondent aux situations suivantes :
- $Tâche_1$ est active ou interrompue et les tâches $Tâche_3$ et $Tâche_4$ sont terminées.
- $Tâche_3$ est active ou interrompue et les tâches $Tâche_1$ et $Tâche_2$ sont terminées.
L'automate correspondant au comportement acceptable du procédé est présenté dans la figure 6.13. Les inégalités algébriques qui délimitent l'espace temporel dans ses sommets sont présentées dans le tableau 6.3. Cet espace temporel a été calculé en appliquant la procédure de synthèse proposée et en utilisant le logiciel de vérification formelle PHAVer.

C Programme de Surveillance du procédé

La figure 6.14 présente les programmes SFC_1 et SFC_2 du procédé considéré. Le programme SFC_1 est obtenu en appliquant la procédure proposée ci-dessus dans le paragraphe 6.3.3. Ce programme est composé de 22 étapes. Il utilise les variables de décalage suivantes : δ_{x_1}, δ_{x_2}, δ_{y_2}, δ_{x_3}, δ_{y_3}, δ_{x_4}, δ_{y_4}, δ_{x_5} et δ_{y_5}. Les actions utilisées dans le programme

l_0	$0 \leq x_1 \leq 2 \wedge y_2 = y_3 = y_4 = y_5 == 0 \wedge 0 \leq 8x_2 - 3x_3 \wedge 0 \leq 7x_2 - 3x_4 \wedge x_2 < 5 \wedge x_3 < 8 \wedge x_4 < 7 \wedge x_5 < 6.$
l_1, l_2, l_3, l_4	$x_2 - x_4 == 0 \wedge y_3 = y_5 == 0 \wedge 0 \leq x_1 \leq 2 \wedge 0 \leq x_2 - y_2 < 2 \wedge 0 \leq x_4 - y_4 < 3 \wedge 0 \leq y_2 \leq 4 \wedge 0 \leq -3x_3 + 8x_5 \wedge x_3 < 8 \wedge x_5 < 6 \wedge 0 \leq y_4 \wedge x_2 < 5 \wedge 0 \leq 6x_3 - 5x_5.$
l_5, l_6, l_7, l_8	$x_2 - x_4 - x_5 == 0 \wedge y_3 = y_4 == 0 \wedge 0 \leq x_1 \leq 2 \wedge x_2 \leq 5 \wedge x_2 - y_2 < 2 \wedge y_2 \leq 4 \wedge 0 \leq x_3 < 8 \wedge 0 \leq y_5 \wedge x_4 \geq 4 \wedge 0 \leq x_5 - y_5.$
l_{10}, l_{11}, l_{12}, l_{13}	$y_2 = y_5 == 0 \wedge x_2 + x_3 - x_4 == 0 \wedge 0 \leq x_1 \leq 2 \wedge y_4 \leq 5 \wedge 0 \leq x_5 < 6 \wedge 0 \leq x_4 - y_4 < 3 \wedge 0 \leq y_3 \wedge 0 \leq x_3 - y_3 < 3 \wedge 3 \leq x_2 < 5 \wedge x_4 < 7.$
l_9, l_{14}, l_{15}, l_{21}	$y_2 = y_4 == 0 \wedge x_2 + x_3 - x_4 - x_5 == 0 \wedge 0 \leq x_1 \leq 2 \wedge 3 \leq x_2 < 5 \wedge 0 \leq x_5 - y_5 < 3 \wedge 0 \leq y_3 \leq 6 \wedge 4 \leq x_4 < 7 \wedge 0 \leq y_5 \leq 5 \wedge 0 \leq x_3 - y_3 < 3 \wedge x_3 < 8 \wedge x_5 < 6.$
l_{18}, l_{19}	$y_2 = y_3 = y_4 == 0 \wedge 0 \leq x_1 \leq 2 \wedge 3 \leq x_2 < 5 \wedge 5 \leq x_3 < 8 \wedge 4 \leq x_4 < 7 \wedge 0 \leq x_5 - y_5 < 3 \wedge 0 \leq y_5 \leq 5 \wedge 0 \leq x_4 + x_5 - x_2 - x_3 \wedge x_5 < 6.$
l_{16}, l_{17}	$y_2 = y_4 = y_5 == 0 \wedge 0 \leq x_1 \leq 2 \wedge 3 \leq x_2 < 5 \wedge x_3 < 8 \wedge 4 \leq x_4 < 7 \wedge 0 \leq x_3 - y_3 < 3 \wedge y_3 \leq 6 \wedge 3 \leq x_5 < 6 \wedge x_2 + x_3 - y_3 < 7 \wedge 0 \leq x_2 + x_3 - x_4 - x_5.$
l_{20}	$y_2 = y_3 = y_4 = y_5 == 0 \wedge 0 \leq x_1 \leq 2 \wedge 3 \leq x_2 < 5 \wedge 0 \leq x_5 < 6 \wedge 5 \leq x_3 < 8 \wedge 4 \leq x_4 < 7.$

TAB. 6.3 – L'espace temporel synthétisé dans les sommets de l'automate de surveillance.

SFC_1 pour initialiser et mettre à jour les variables intermédiaires sont données dans le tableau 6.4. À titre d'exemple, prenons la transition suivante de l'automate $l_1 \xrightarrow{h_1} l_{11}$. Le franchissement de la transition correspondante dans le programme SFC_1 $s_1 \rightarrow s_{11}$, ayant la réceptivité h_1, exécute les actions suivantes :
- à l'instant de désactivation l'étape s_1, les variables de décalage correspondant aux chronomètres persistants actifs (x_4 et y_4) sont mises à jour. Cela se fait par l'ajout des valeurs de δ_{x_4} et δ_{y4}, à la durée de l'activation de l'étape s_1 avant de franchir $s_1 \rightarrow s_{11}$.
- à l'instant de désactivation l'étape s_1, il y a mise à zéro de la variable de décalage correspondant au chronomètres x_3 initialisés par la transition $l_1 \rightarrow l_{11}$. Ce cas d'initialisation est similaire à celui présenté dans la figure 6.8.
- à l'instant de l'activation l'étape s_{11}, il y a mise à zéro de la variable de décalage correspondant au chronomètres y_2 initialisés par la transition $l_1 \rightarrow l_{11}$.
Une partie de la réceptivité de la transition $s_{22} \rightarrow s_{23}$ du programme SFC_2 est la suivante :
.... ou $[s_{11}.X$ et $\{(\delta_{x_2} + (\delta_{x_3} + s_{11}.T) - (\delta_{x_4} + s_{11}.T) \neq 0)$ ou $((\delta_{y_4} + s_{11}.T) > 5)$ ou $((\delta_{x_4} + s_{11}.T) \geq 7)$ ou $(\delta_{y_3} + s_{11}.T < 0)$ ou $(0 > (\delta_{x_4} + s_{11}.T) - (\delta_{y_4} + s_{11}.T) \geq 3)$ ou $(0 > (\delta_{x_3} + s_{11}.T) - (\delta_{y_3} + s_{11}.T) \geq 3)\}]$ ou ..... ou $[s_{21}.X$ et $\{(0 > (\delta_{x_3} + s_{21}.T) - \delta_{y_3} \geq 3)$ ou $(0 > (\delta_{x_5} + s_{21}.T) - \delta_{y_5} \geq 3)$ ou $(0 > (\delta_{x_5} + s_{21}.T) \geq 6)$ ou $0 > \delta_{y_3} > 6$ ou $0 > \delta_{y_5} > 5$ ou $(\delta_{x_3} + s_{21}.T) \geq 8\}]$ .....

D Simulation du système de surveillance

Nous présentons dans ce paragraphe la simulation du système de surveillance de procédé d'alimentation en liquide (Fig. 6.11). Notre objectif de simuler le système de surveillance, à savoir l'automate de surveillance et le programme de surveillance, est d'une part de valider le système de surveillance et d'autre part, de montrer la correction de la translation de l'automate de surveillance en programme de surveillance.

D.1 Validation de l'automate de surveillance Nous avons créé un outil de simulation basé sur un simulateur réagissant à des signaux binaires et permettant la détection de défauts. Nous avons opté pour les outils de simulation Stateflow et CheckMate du logiciel Matlab/Simulink. Stateflow est un outil de développement et de conception graphique puissant. Il permet notamment de simuler les systèmes réactifs complexes basés sur la théorie des machines à états finis. Checkmate a été développé à l'université Carnegie Mellon. Il permet de représenter une classe des systèmes hybrides dans lesquels les franchissement des transitions peuvent se produire lorsque les variables continues dépassent des seuils spécifiés. Cet outil appelé "Checkmate" permet d'associer à chaque sommet d'un automate à états finis défini par le Stateflow, les dynamiques des variables continues et un polyèdre. Il génère un événement binaire indiquant si les valeurs des variables continues, dans un sommet appartiennent à la région décrite par

le polyèdre. Cet événement peut être utilisé par l'automate à états finis défini par le Stateflow pour franchir une transition.
Destiné à la base pour représenter l'automate hybride, nous avons utilisé les Stateflow et CheckMate dans le cadre de notre application pour la simulation du système de surveillance du procédé d'alimentation en liquide. La figure 6.15.a décrit le simulateur de l'automate de surveillance. Nous générons des signaux binaires représentant les événements du procédé et sa commande (quelques scénarios possibles de fonctionnement). Ces signaux sont envoyés à l'automate de surveillance codé par les Stateflow et CheckMate de Matlab. Stateflow décrit les sommets de l'automate de surveillance et les conditions de franchissement. Ces conditions sont les événements en provenance du procédé commandé ou les événements représentant les violation de l'espace temporel associé au sommet. Ces derniers événements sont générés par le Checkmate à l'instant de la violation d'in espace temporel associé à un sommet de l'automate.

Nous présentons dans le figure 6.15.b un scénario de fonctionnement possible. Les pompes M_2 et M_4 sont interrompues pendant 3 $u.t$. Par conséquent, la tâche de vidage du bac 1 est interrompue pendant 3 $u.t$. Cette interruption peut être, par exemple, la conséquence d'une perturbation dans l'alimentation électrique des pompes M_2 et M_4. Elle peut se produire à cause d'un blocage dans la conduite. Ce blocage peut avoir comme origine des particules solides dans le liquide.
La séquence d'événements observés dans ce scénario est la suivante : $d_{(0.1)} \rightarrow h_{1(3)} \rightarrow h_{2(5)} \rightarrow s_{b_1(6)} \rightarrow s_{b_2(6.1)}$. Pour ce scénario de fonctionnement, les évolutions des chronomètres x_2, x_3, x_4, x_5, y_2, y_3, y_4 et y_5 sont présentées dans la figure 6.16. Cette dernière montre aussi qu'un défaut est détecté à l'instant $\tau = 9$ $u.t$.
L'automate $\mathbb{A}_M$ atteint le sommet l_{21} du l_0 où les sommets visités au cours de ce scenario sont : $l_0 \xrightarrow{d_{(0.1)}} l_1 \xrightarrow{h_{1(3)}} l_{11} \xrightarrow{h_{2(5)}} l_{15} \xrightarrow{s_{b_1(6)}} l_{14} \xrightarrow{s_{b_2(6.1)}} l_{21}$.
L'inégalités $0 \leq x_3 - y_3 < 3$ dans le sommet l_{21} détecte un défaut à l'instant $\tau = 9$ $u.t$ (Fig. 6.16). Les valeurs correspondantes de x_3 et y_3 sont respectivement $x_3(\tau) = 6$ $u.t$ et $y_3(\tau) = 3$ (Fig. 6.16). On peut expliquer le déclenchement de l'alarme par la raison suivante : afin d'accomplir la tâche de vidage de bac 1 correctement, on a besoin d'avoir au moins la durée $\alpha_3 - y_3(\tau) = 5 - 3 = 2$ $u.t$. Donc, la valeur correspondante de x_3 est $x_3 = x_3(\tau) + (\alpha_3 - y_3(\tau)) = 6 + 2 = 8$ $t.u$. Cette durée d'exécution dépasse la durée maximale pour vider le bac 1.

D.2 Validation du programme SFC de surveillance Le schéma de simulateur décrit ci-dessus est présenté dans la figure 6.15.c. Nous avons introduit le programme de surveillance codé par le SFC dans l'API. Les scénarios de fonctionnement sont générés en utilisant Matlab. Nous accédons à l'automate par l'intermédiaire d'un réseau Ethernet. Nous nous connectons à l'automate par l'intermédiaire du serveur OPC installé au réseau. Cette connexion à un serveur OPC à partir de Matlab

permet alors d'accéder aux variables de l'API et d'effectuer une lecture/écriture sur les différentes variables. L'outil de Matlab utilisé permettant d'effectuer la lecture et l'écriture vers et depuis l'automate, est OPC librairie de Simulink. Ainsi, les événements générés par Matlab peuvent être envoyés au programme de surveillance dans l'automate. La décision de surveillance, calculée par le programme SFC, peut être récupérée de l'API.

Le scénario présenté dans la figure 6.15.b est envoyé au programme SFC. La figure 6.17.a montre les variables de décalage δ_{x_3}, δ_{y_3}. Les évolutions des différentes variables temporelles associées aux étapes visitées sont aussi présentées dans la figure 6.17.b. A partir de ces variables, le programme SFC_1 a simulé les évolutions des chronomètres x_3 et y_3. La valeur de x_3 correspond à :
- la variable δ_{x_3} pendant l'activation des étapes s_0, s_1. Cette variable a une valeur nulle pendant cette durée. Remarquons que, dans les sommets l_0 et l_1, nous avons : $\dot{x_3} = 0$ et $x_3 = 0$ (Fig. 6.16).
- la somme de δ_{x_3} et les variables temporelles $s_{11}.T$, $s_{15}.T$, $s_{14}.T$ et $S_{21}.T$ pendant l'activation des étapes s_{11}, s_{15}, s_{14} et s_{21}. Remarquons que, dans les sommets correspondants, nous avons $\dot{x_3} = 1$.
De même, la valeur du y_3 correspond à :
- la variable δ_{y_3} pendant l'activation des étapes s_0, s_1. Cette variable a une valeur nulle pendant cette durée.
- la somme de δ_{y_3} et les variables temporelles $s_{11}.T$ et $s_{15}.T$ pendant l'activation des étapes S_{11} et s_{15}.
- la variable δ_{y_3} pendant l'activation des étapes $s_{14}.T$ et $S_{21}.T$. Cette variable a une valeur 3 pendant cette durée.
La figure 6.17.c montre les évolutions des variables du SFC_1 correspondantes aux horloges x_3 et y_3.

Nous présentons, dans la figure 6.18, une comparaison entre les évolutions des chronomètres dans l'automate de surveillance et leurs correspondant dans le programme de surveillance du procédé d'alimentation en liquide. Le défaut est détecté par le programme à l'instant $\tau \approx 9\ u.t.$

Pour conclure, nous constatons la capacité du programme de surveillance à suivre les évolutions de l'automate de surveillance. Cela nous conduit aussi à constater la correction de la procédure de traduction de l'automate de surveillance en programme SFC.

6.5 Conclusion

Nous avons présenté au cours de ce chapitre une méthode de traduction de l'automate de surveillance représentant le comportement acceptable en programme SFC. Ce dernier, appelé "programme de surveillance" sera implémenté dans l'API. Le programme de surveillance représente un sommet de l'automate de surveillance par une étape. Une transition de l'automate peut être représentée par une transition dans le programme SFC. La réceptivité d'une transition est l'événement associé à la transition de l'automate. Chaque chronomètre dans l'automate est représenté par une variable intermédiaire. La représentation de l'évolution d'un chronomètre se fait en utilisant sa variable intermédiaire, des variables temporelles associées aux étapes de programme SFC et des actions. Ces dernières initialisent les variables intermédiaires lorsque les chronomètres correspondant dans l'automate sont initialisés. Elles mettent à jour les variable intermédiaires à l'instant de franchissement d'une étape à une autre. Ceci s'est fait pour les variables correspondant aux chronomètres suspendus ou persistant actifs dans le nouveau sommet. La méthode de représentation de l'espace temporel dans chaque sommet est présentée. Elle prend en compte les dynamiques des chronomètres dans chaque sommet. Une procédure de traduction structurelle de l'automate de surveillance en programme SFC a été présentée.

Afin de montrer les avantages et les performances de l'approche de surveillance proposée dans ce mémoire, nous avons étudié un processus industriel. L'automate de surveillance de cette application est construit. Les validations expérimentales de cette application ont été faites à partir de l'automate de surveillance et de son programme de surveillance. Des séquences des signaux binaires représentant les événements du procédé sont générées. Ces séquences sont utilisées à la fois pendant la validation de l'automate de surveillance et du programme de surveillance. Dans un premier temps, l'automate évolue en réagissant à ces séquences d'événements. Les décisions de l'automate sur le comportement sont enregistrées (acceptable ou déclaration d'un défaut). Dans un deuxième temps, le programme de surveillance développé avec le SFC dans un API est soumis aux mêmes séquences d'événements représentant des scénarios de fonctionnement du procédé considéré. La décision du programme de surveillance sur les comportements représentés par les scénarios considérés est aussi enregistrée. La comparaison entre les réactions de l'automate de surveillance et du programme de surveillance sur les scénarios de fonctionnement, nous a permis de constater la correction de la procédure de traduction proposée.

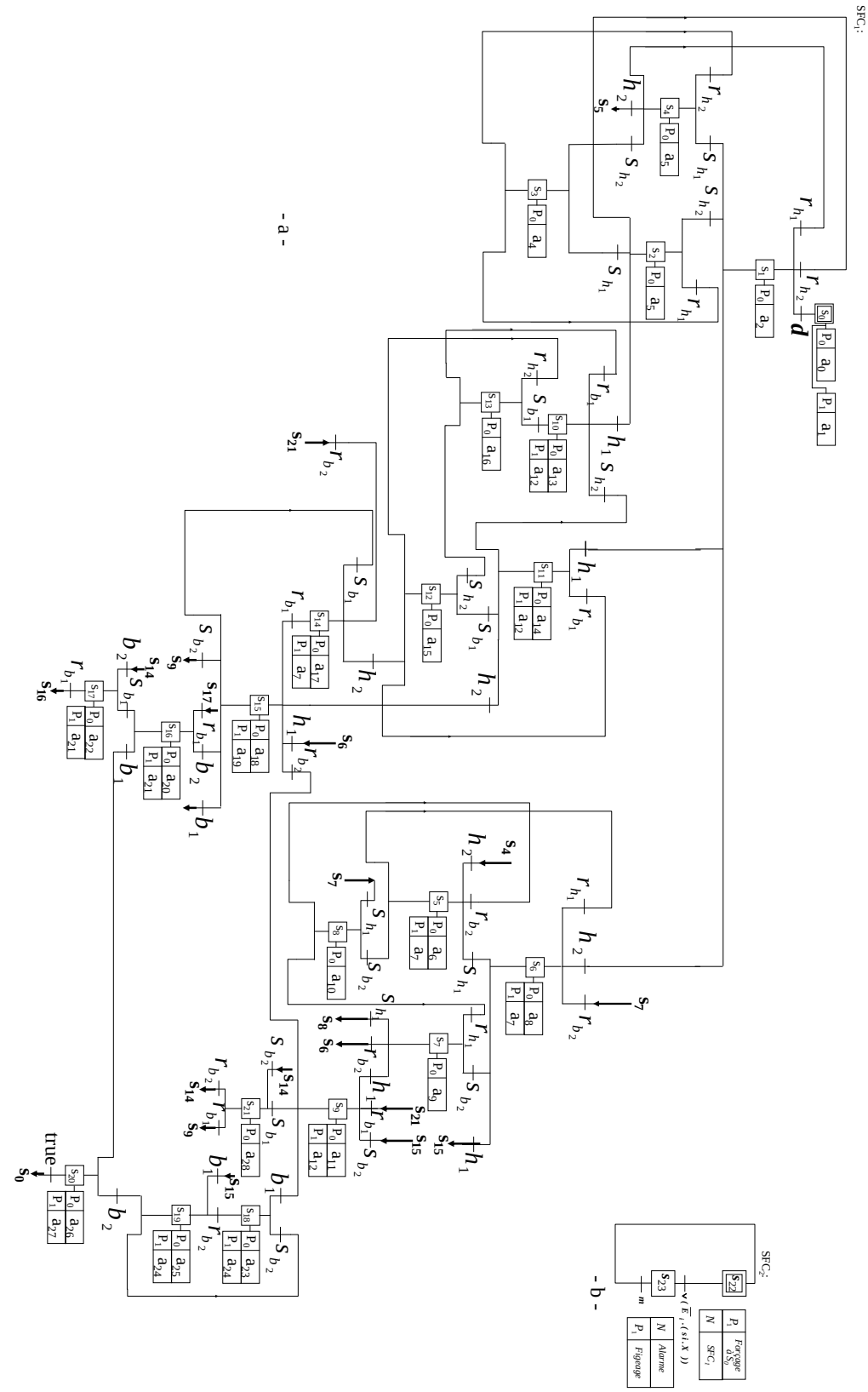

FIG. 6.14 – Programmes de surveillance du procédé considéré.

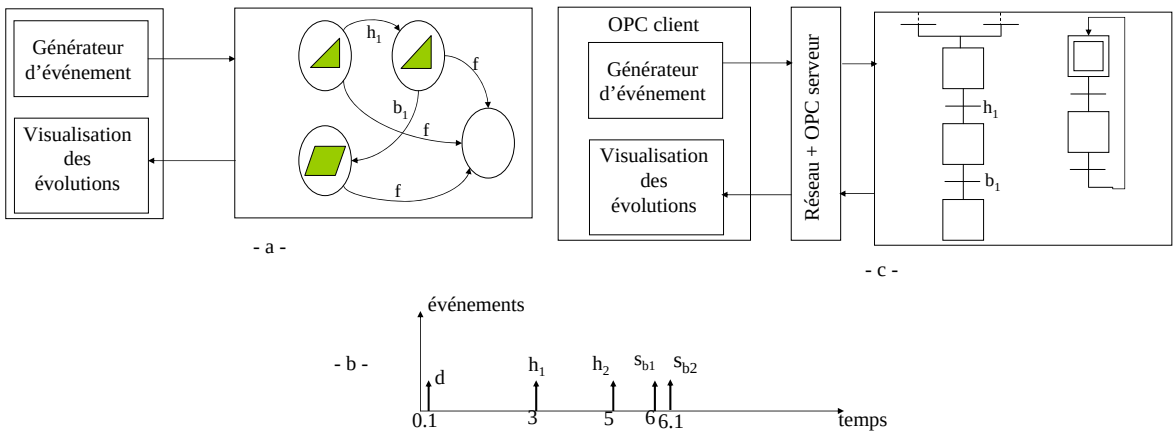

FIG. 6.15 – a- Schéma de simulateur basé à l'outil Checkmate b- une séquence possible d'événements c- Schéma de simulateur basé à l'outil SFC.

Nom d'action	Action
a_0	$\delta_X := 0$
a_1	$\delta_{x_2} = \delta_{x_4} := 0$
a_2	$(\delta_{x_2}, \delta_{y_2}, \delta_{x_4}, \delta_{y_4}) = (\delta_{x_2}, \delta_{y_2}, \delta_{x_4}, \delta_{y_4}) + s_1.T$, $\delta_{x_3} = \delta_{x_5} := 0$
a_3	$(\delta_{x_2}, \delta_{y_2}, \delta_{x_4}) = (\delta_{x_2}, \delta_{y_2}, \delta_{x_4}) + s_2.T$, $\delta_{x_3} := 0$
a_4	$(\delta_{x_2}, \delta_{x_4}) = (\delta_{x_2}, \delta_{x_4}) + s_3.T$
a_5	$(\delta_{x_2}, \delta_{y_4}, \delta_{x_4}) = (\delta_{x_2}, \delta_{y_4}, \delta_{x_4}) + s_4.T$, $\delta_{x_5} := 0$
a_6	$(\delta_{x_2}, \delta_{x_5}, \delta_{y_5}) = (\delta_{x_2}, \delta_{x_5}, \delta_{y_5}) + s_5.T$
a_7	$\delta_{y_4} := 0$
a_8	$(\delta_{x_2}, \delta_{x_5}, \delta_{y_2}, \delta_{y_5}) = (\delta_{x_2}, \delta_{x_5}, \delta_{y_2}, \delta_{y_5}) + s_6.T$, $\delta_{x_3} := 0$
a_9	$(\delta_{x_2}, \delta_{x_5}, \delta_{y_2}) = (\delta_{x_2}, \delta_{x_5}, \delta_{y_2}) + s_7.T$, $\delta_{x_3} := 0$
a_{10}	$(\delta_{x_2}, \delta_{x_5}) = (\delta_{x_2}, \delta_{x_5}) + s_8.T$
a_{11}	$(\delta_{x_3}, \delta_{x_5}, \delta_{y_3}, \delta_{y_5}) = (\delta_{x_3}, \delta_{x_5}, \delta_{y_3}, \delta_{y_5}) + s_9.T$
a_{12}	$\delta_{y_2} := 0$
a_{13}	$(\delta_{x_3}, \delta_{x_4}, \delta_{y_3}) = (\delta_{x_3}, \delta_{x_4}, \delta_{y_3}) + s_{10}.T$
a_{14}	$(\delta_{x_3}, \delta_{x_4}, \delta_{y_3}, \delta_{y_4}) = (\delta_{x_3}, \delta_{x_4}, \delta_{y_3}, \delta_{y_4}) + s_{11}.T$, $\delta_{x_5} := 0$
a_{15}	$(\delta_{x_3}, \delta_{x_4}, \delta_{y_4}) = (\delta_{x_3}, \delta_{x_4}, \delta_{y_4}) + s_{12}.T$, $\delta_{x_5} := 0$
a_{16}	$(\delta_{x_3}, \delta_{x_4}) = (\delta_{x_3}, \delta_{x_4}) + s_{13}.T$
a_{17}	$(\delta_{x_3}, \delta_{x_5}, \delta_{y_5}) = (\delta_{x_3}, \delta_{x_5}, \delta_{y_5}) + s_{14}.T$
a_{18}	$(\delta_{x_3}, \delta_{x_5}, \delta_{y_3}, \delta_{y_5}) = (\delta_{x_3}, \delta_{x_5}, \delta_{y_3}, \delta_{y_5}) + s_{15}.T$
a_{19}	$\delta_{y_2} := 0$, $\delta_{y_4} := 0$
a_{20}	$(\delta_{x_3}, \delta_{y_3}) = (\delta_{x_3}, \delta_{y_3}) + s_{16}.T$
a_{21}	$\delta_{y_5} := 0$
a_{22}	$\delta_{x_3} := \delta_{x_3} + s_{17}.T$
a_{23}	$\delta_{x_5} := \delta_{x_5} + s_{18}.T$
a_{24}	$\delta_{y_3} := 0$
a_{25}	$(\delta_{x_5}, \delta_{y_5}) = (\delta_{x_5}, \delta_{y_5}) + s_{19}.T$
a_{26}	$\delta_{y_3} := 0$, $\delta_{y_5} := 0$
a_{27}	$\delta_{x_1} := 0$
a_{28}	$(\delta_{x_3}, \delta_{x_5}) = (\delta_{x_3}, \delta_{x_5}) + s_{21}.T$

TAB. 6.4 – Les actions du programme SFC_1 présenté dans la figure 6.14.

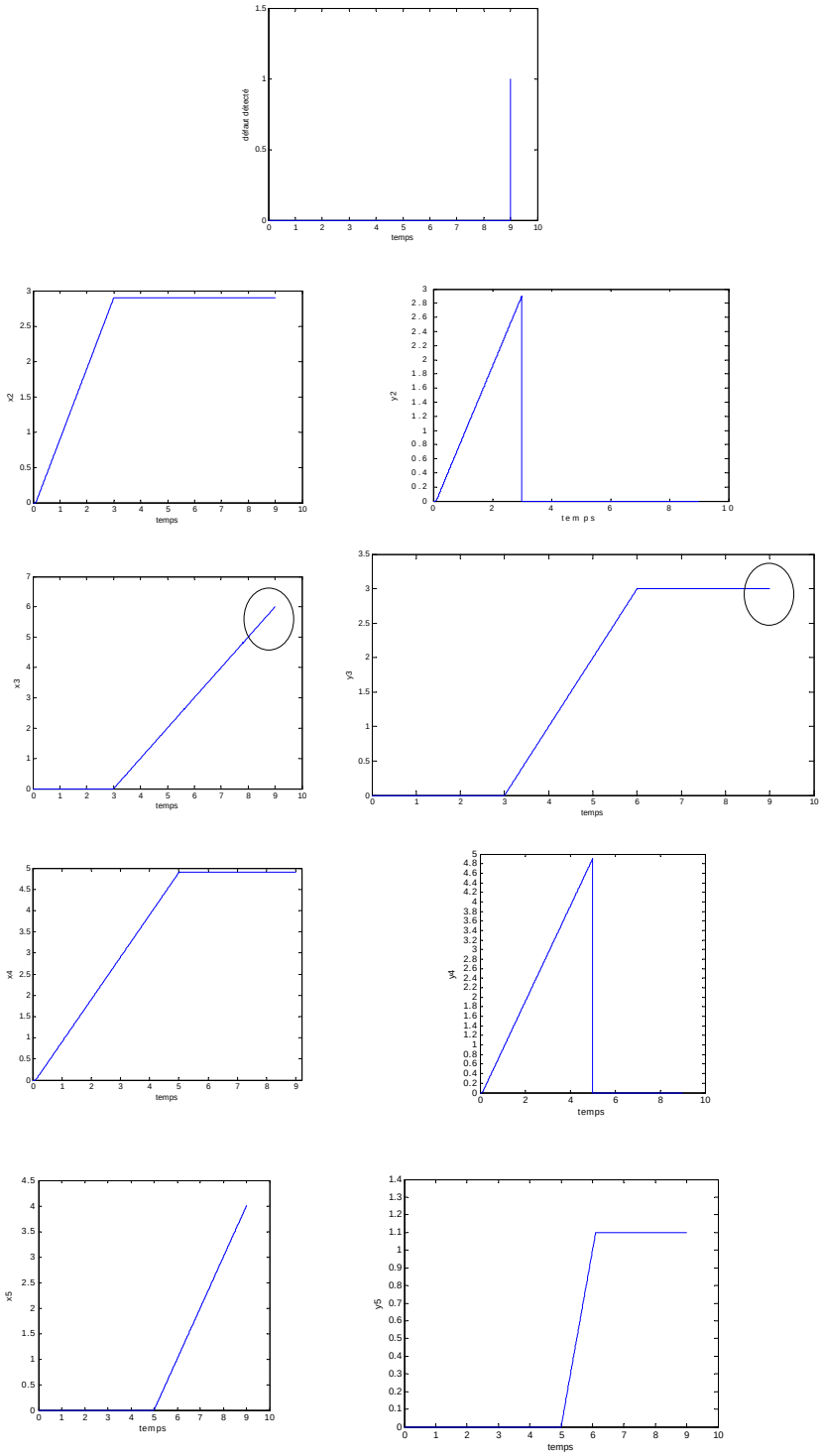

FIG. 6.16 – Évolution des chronomètres de l'automate de surveillance pour le scénario considéré.

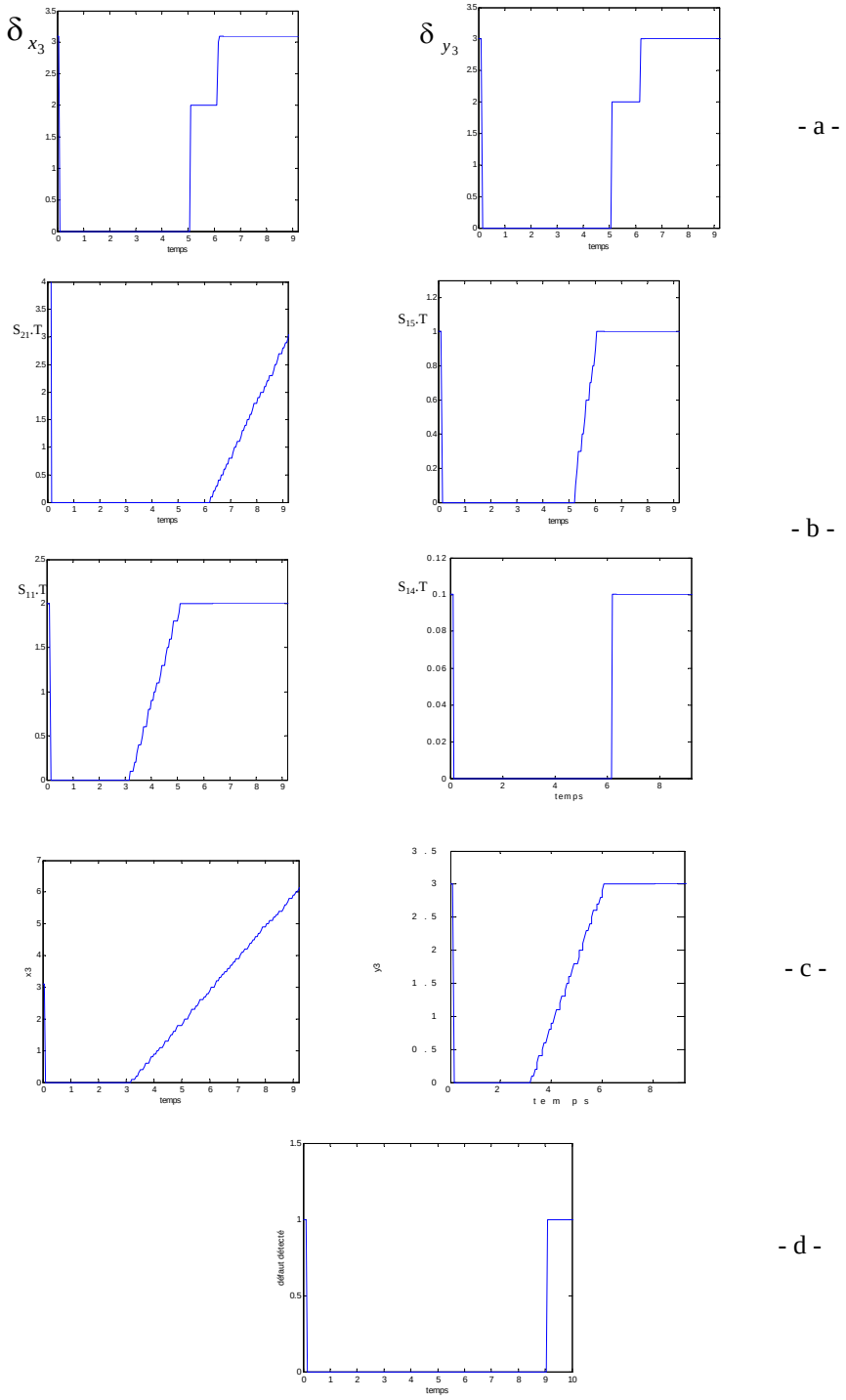

FIG. 6.17 – Évolution des variables du programme de surveillance pour le scénario considéré.

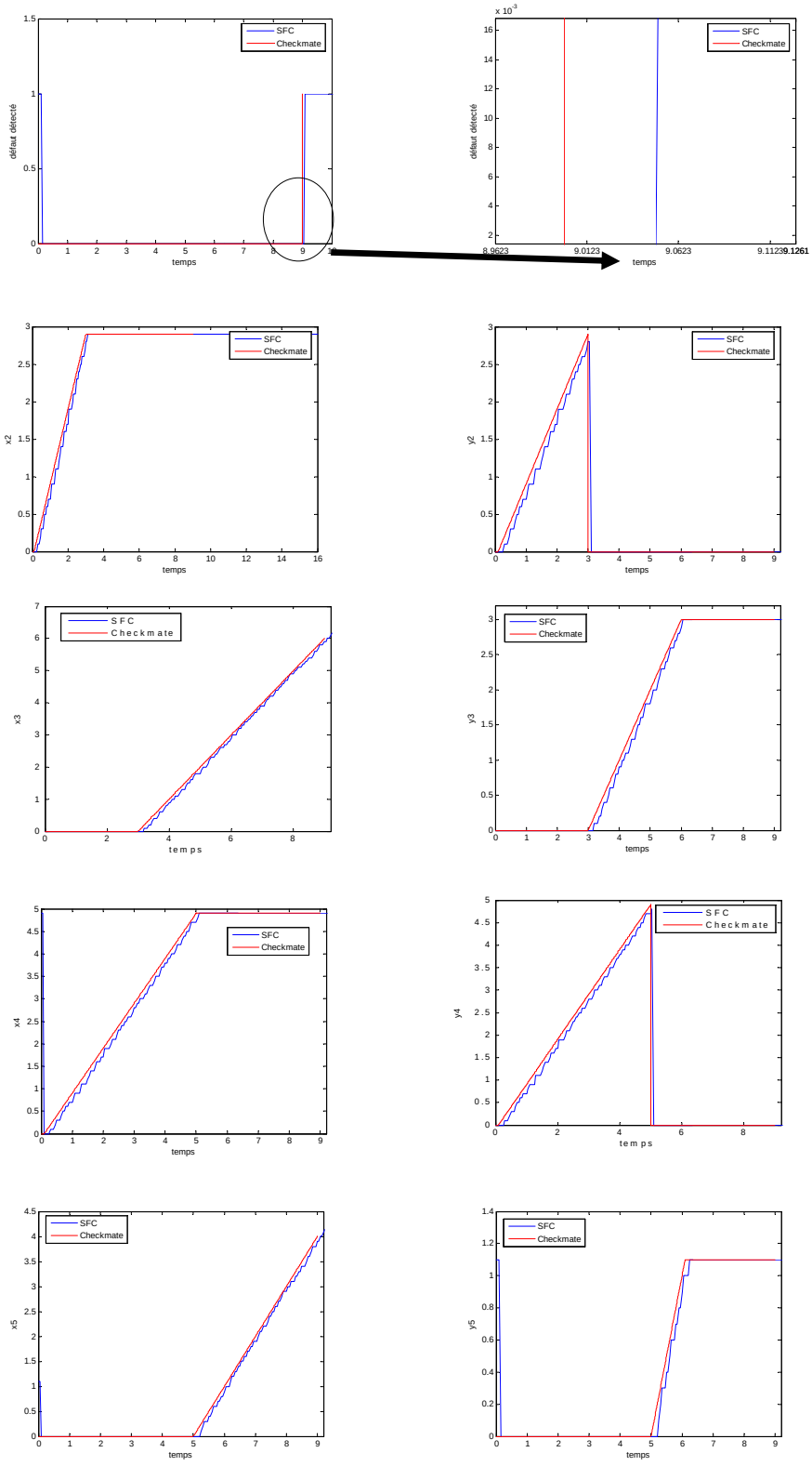

FIG. 6.18 – Comparaison entre l'évolution des chronomètres dans l'automate de surveillance et leurs correspondants dans le programme SFC.

Chapitre 7

CONCLUSION GENERALE ET PERSPECTIVES

Les travaux que nous avons présentés dans ce document considèrent le problème de surveillance des systèmes à événements discrets commandés. Ces derniers ont un caractère temporel où le temps apparaît comme paramètre de fonctionnement. La méthode de surveillance proposée a pour objectif de détecter au plus tôt les défauts interruptibles pouvant survenir dans ce type des systèmes. La construction de ce système de surveillance a pris en compte les éléments décrits ci-dessous.
Tout d'abord, les spécifications temporelles relatives aux événements qui peuvent avoir lieu dans un système à événements discrets représentent les durées d'exécutions des tâches du système.
Ensuite, une tâche du système à surveiller a un état intermédiaire entre l'état de fonctionnement normal et l'état de défaut. Suite à l'occurrence d'un défaut intermittent, l'état du système peut basculer entre les états normaux et intermédiaires. Nous avons appelé comportement acceptable, le comportement atteint par le système.
Enfin, l'étude a été restreinte aux défauts intermittents et permanents, dits "interruptibles". La dynamique d'exécution des tâches du système, suite à l'occurrence d'un défaut interruptible est observable. Autrement dit : le basculement de l'état du système entre l'état normal et l'état intermédiaire est observable.
Le mécanisme développé dans notre approche a pour principe d'une part, d'exploiter les contraintes temporelles dont on peut disposer sur le comportement acceptable du système ; et d'autre part, d'utiliser les techniques d'analyse d'atteignabilité de l'automate à chronomètres représentant le comportement du système sujet aux défauts interruptible. En effet, le modèle de système sujet aux défauts interruptibles est obtenu par la composition synchrone des différents modèles des tâches constituant le système. Chaque tâche est modélisée par un automate à chronomètres ayant deux chronomètres. Ces derniers mesurent les durées d'exécution effective et totale. Le modèle global obtenu du système est également un automate à chronomètres. Le fait que les tâches du système sont exécutées correctement est décrit sous la forme d'une propriété de l'automate. Ensuite, une procédure de synthèse est appliquée à l'automate afin de délimiter l'espace temporel dans chaque sommet vérifiant cette propriété. Autrement dit, nous avons déduit de l'automate représentant le comportement du système, l'automate délimitant seulement l'exécution acceptable des tâches du système. Dans cet automate, les sommets représentent les différentes situations atteignables du système et les équations différentielles dans un sommet reflètent les dynamiques des tâches dans ce sommet : actives ou interrompues à cause des défauts interruptibles. L'espace temporel dans un sommet caractérise les évolutions temporelles possibles du système dans la situation correspondante du système. Ceci signifie que le système peut rester dans une situation tant que les valeurs des chronomètres vérifient l'espace temporel du sommet correspondant à cette situation. Cet automate est synchronisé sur l'événement provenant du système commandé. Il est également temporisé sur les horloges internes. Une action de l'automate correspond à un événement dans le système à surveiller. Elle peut être soit un signal du capteur ou un ordre provenant du système de commande. L'événement

associé à une transition peut se produire tant que les valeurs des chronomètres vérifient l'espace temporel associé au sommet source. Ainsi, cet automate permettra de détecter un défaut au plus tôt, suite à la violation de l'espace temporel associé à un sommet. Nous entendons par le terme "*détection au plus tôt*" une détection de l'absence ou du retard d'un événement sans attendre l'expiration de la date au plus tard prévue de son apparition.
Malgré la restriction que nous avons faite, nous avons vu que le système de surveillance peut être modifié afin de prendre en compte des défauts qui font ralentir l'exécution des tâches du système.

La deuxième partie des travaux que nous avons présentés dans ce mémoire s'inscrit dans les travaux de la communauté de modélisation en utilisant le modèle Réseau de Petri.
En effet, modéliser le comportement du système susceptible aux défauts interruptibles par des automates à chronomètres n'est pas toujours simple. De plus, certains cas des systèmes à surveiller ne sont pas exprimables avec les automates, comme le cas des systèmes à instance-multiple. Ceci rejoint la notion de multi-sensibilisation. Une des solutions retenues dans la littérature pour modéliser le comportement sans interruption d'un système à multi-instance est d'utiliser les réseaux de Petri temporels avec la notion multi-sensibilisation. Cependant, les réseaux de Petri temporels ne sont en général pas suffisants pour modéliser les applications en présence du comportement interruptible. Ceci nous a conduit à proposer une extension du réseau de Petri temporel permettant de modéliser les systèmes interruptibles. L'extension proposée conserve les propriétés fondamentales d'un réseau de Petri tout en apportant des possibilités nouvelles et intéressantes de modélisation de l'interruption et la reprise d'une activité. Ce nouveau modèle étend le concept d'horloge au concept de chronomètre. Un nouveau mécanisme d'initialisation des chronomètres appelé "Post-initialisation" est introduit. Pour cela, ce nouveau modèle est appelé Réseaux de Petri à chronomètres Post- et Pré-initialisés.
Nous avons également présenté une méthode pour l'analyse temporelle des réseaux de Petri à chronomètres. Elle consiste, dans un premier temps, à traduire le réseau de Petri à chronomètres borné en automate à chronomètres. Dans un second temps, nous appliquons un semi-algorithme sur l'automate ainsi obtenu, basé sur les techniques d'analyse en avant connues dans les automates hybrides et en utilisant le logiciel de vérification formelle PHAVer. Si cet algorithme d'analyse ne converge pas, nous avons proposé d'utiliser les techniques du logiciel de vérification formelle PHAVer afin de forcer la terminaison. Il s'ensuit qu'on calcule une surapproximation conservative de l'espace d'état. Cette analyse permet de calculer l'espace d'états temporel de l'automate à chronomètres qui est bisimilaire au réseau de Petri à chronomètres initial.
Ensuite, nous avons combiné les possibilités du réseau de Petri à chronomètres et la puissance d'analyse de l'automate à chronomètres pour concevoir notre système de

surveillance. Nous avons utilisé les réseaux de Petri à chronomètres afin de modéliser les systèmes sujet aux défauts interruptibles. Le modèle réseau de Petri à chronomètres d'une tâche interruptible a certaines propriétés. Grâce à ces propriétés, nous avons traduit le modèle réseau de Petri à chronomètres du système en un automate. Ce dernier exprime les évolutions du système en utilisant seulement les chronomètres mesurant les durées effectives et totales des tâches.
Le modèle réseau de Petri à chronomètres proposé est général. Il est capable non seulement de modéliser le comportement des systèmes sujets aux défauts interruptibles mais également les activités peuvent être suspendues puis reprises au même endroit un peu plus tard. Ce fait est expliqué par une application. Dans cette dernière, nous avons modélisé un système de production ayant des ressources partagées. Les tâches exécutées sur ces ressources ont des priorités différentes. De par la différence de priorités entre les tâches exécutées par la même ressource, la durée totale d'exécution de ces tâches est différente de sa durée d'exécution effective. Des calculs d'indices de performances ont été effectués sur l'automate et interprétés sur le modèle réseau de Petri.

La construction du système de surveillance, soit par la modélisation directe de l'automate à chronomètres soit par le réseau de Petri à chronomètres, est réalisée hors ligne. L'implémentation en ligne de notre approche de surveillance consiste à traduire l'automate de surveillance en programme SFC implémentable dans les automates programmables industriels (API). Ces derniers sont largement utilisés dans le milieu industriel pour l'automatisation des installations industrielles. Dans ce but, une procédure de traduction structurelle de l'automate de surveillance en programme SFC est introduite. Une validation expérimentale de l'automate de surveillance et de son programme SFC correspondant d'un procédé est présentée. Des séquences des signaux binaires représentant les événements du procédé sont générées. Ces séquences sont utilisées à la fois pendant la validation de l'automate de surveillance et du programme de surveillance. L'évolution de l'automate et de son programme SFC, en régissant à ces séquence d'événements est enregistrée. La comparaison entre les réactions de l'automate de surveillance et du programme de surveillance sur les scénarios de fonctionnement, nous ont permis de constater la correction de la procédure de traduction proposée.

Dans la proposition que nous avons faite dans ce travail de thèse, nous avons modélisé le comportement de système en utilisant des horloges à deux états : opérationnelles ou arrêtées. Cela est dû au fait que nous avons restreint notre approche aux seuls défauts interruptibles. Parmi les perspectives que nous envisageons, il y a l'extension des résultats obtenus aux systèmes hybrides linéaires. Cela permettra de construire un système de surveillance capable de détecter les défauts qui modifient la dynamique des tâches de systèmes (accélération, ralentissement), l'interruption étant un cas particulier. En effet, les systèmes hybrides linéaires permettent de décrire, à la fois, les aspects événementiels

et continus. La dynamique continue peut prendre n'importe quelle valeur réelle.
Une deuxième extension est d'orienter l'approche vers une structure décentralisée. Ceci permet d'éviter le problème d'explosion combinatoire résultant de l'utilisation de la structure centralisée. Il faut, pour réaliser une structure décentralisée de surveillance, développer un coordinateur qui va gérer les différents problèmes d'ambigüté entre les systèmes de surveillances locaux et qui va prendre la décision finale.
Une autre perspective à plus long terme de cette démarche, est d'établir une procédure de reconfiguration de la commande après la détection de défaut. Ce travail demande une classification des défauts selon leur importance. Une solution, pour la reconfiguration de la commande à partir de notre approche, est d'analyser le défaut et d'indiquer la situation du système à l'utilisateur. Ce dernier doit avoir la possibilité de choisir la séquence d'événements lui permettant de retourner vers le fonctionnement acceptable.

Bibliographie

Akesson, K. and Skoldstam, M. (2007). Towards a framework for integrated supervisory and logic control. In *1st IFAC Workshop on Dependable Control of Discrete Systems (DCDS'07)*, volume -, pages –.

Allahham, A. and Alla, H. (2006). Monitoring of timed discrete events systems : Application to manufacturing systems. In *The 32nd Annual conference of IEEE Industrial Electronics Society*, pages 3609–3614.

Allahham, A. and Alla, H. (2007a). Design and implementation of a monitoring system using grafcet. In *4th International Conference on Informatics in Control, Automation and Robotics, Angers, France*.

Allahham, A. and Alla, H. (2007b). Monitoring of a class of timed discrete events systems. In *IEEE International Conference On Robotics and Automation, Rome, Italy*, pages 3609–3614.

Altisen, K. (2001). *Application de la synthèse de contrôleur à l'ordonnancement de systèmes temps-réel*. PhD thesis, Institut National Polytechnique de Grenoble.

Alur, R., Courcoubetis, C., Halbwachs, N., Henzingerd, T., Hod, P., Nicollin, X., Olivero, A., Sifakis, J., and Yovine, S. (1995). The algoritmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1).

Alur, R. and Dill, D. (1994). A theory of timed automata. *Theoretical computer Science*, (126) :183–235.

Asarin, E., Dang, T., Frehse, G., Girard, A., Guernic, C. L., and Maler, O. (2006). Recent progress in continuous and hybrid reachability analysis. In *Proceedings of the IEEE International Symposium on Computer-Aided Control Systems Design*, Technische Universität München, Munich, Germany.

Bauer, N., Huuck, R., Lukoschus, B., and Engell, S. (2004). A unifying semantics for sequential function charts. *Intergration of software specification techniques for application in engineering*, LNCS-3147.

Bengtsson, J. and Yi, W. (2004). Timed automata : Semantics, algorithms, tools. In *Lectures on Concurrency and Petri Nets*, volume LNCS 3098. Springer-Verlag.

Berthomieu, B. (2001). La méthode des classes d'états pour l'analyse des réseaux temporels - mise en œuvre, extension à la multi-sensibilisation. In *Proceedings de Modélisation des Systèmes Réactifs*, Toulouse, France.

Berthomieu, B. and Diaz, M. (1991). Modeling and verification of time dependent systems using time petri nets. *IEEE transactions on software engineering*, 17(3) :259Ű–273.

Berthomieu, B., Lime, D., Roux, O. H., and Vernadat (2005). Problèmes d'accessibilité et espaces d'états abstraits des réseaux de petri temporels à chrnomètres. *Journal européen des systèmes automatisés*, 39 :223–238.

Berthomieu, B., Ribet, P. O., and Vernadat, F. (2004). The tool tina - construction of abstract state spaces for petri nets and time petri nets. *International Jounal of Production Research*, 42(14) :2741–2756.

Bobbio, A., Puliafito, A., and Telek, M. (2000). A modeling framework to implement preemption policies in non-Markovian SPNs. *IEEE Transactions on Software Engineering*, 26(1) :36–54.

Boufaied, A. (2003). *Contribution à la surveillance distribuée des systèmes à événements discrets complexes*. PhD thesis, L'université Paul Sabatier de Toulouse.

Brandin, B. A. and Wonham, W. M. (1994). Supervisory control of timed discrete-event systems. *IEEE transaction on Automatic Control*, 39(2) :329–342.

Bucci, G., Fedeli, A., Sassoli, L., and Vicario, E. (2004). Timed state space analysis of real-time preemptive systems. *IEEE Transactions on Software engineering*, 30(2) :97–111.

Cassez, F. and Larsen, K. (2000). The impressive power of stopwatch. Number 1877, pages 38–152. Lecture Notes in Computer Science, Springer-Verlag.

Chen, J. and Patton, R. (1999). *Robust model-based fault diagnosis for dynamic systems*. Kluwer Academic Publishers.

Chen, Y.-L. and Provan, G. (1997). Modeling and diagnosis of timed discrete event systems- a factory automation example. In *The American Control Conference, New Mexico*, pages 31–36.

Chow, E. Y. and Willsky, A. S. (1984). Analytical redundancy and the designe of robust failure detection system. *IEEE transaction on Automatic and Control*, AC-29(7) :603–614.

Cocquempot, V., Mezyani, T. E., and Staroswieckiy, M. (2004). Fault detection and isolation for hybrid systems using structured parity residuals. In *Proceeding of Asian Control Conference, ASCC'04*.

Cocquempot, V., Staroswiecki, M., and Mezyani, T. E. (2003). Switching time estimation and fault detection for hybrid systems using structured parity residuals. In *Proceeding of IFAC SAFEPROCESS'03*.

Combacau, M. (1991). *Commande et surveillance des systèmes à événements discrets complexes : applications aux ateliers flexibles*. PhD thesis, Université Paul Sabatier.

Combacau, M. (2001). Approche discrète de la surveillance des systèmes de production. In *Matrîse des risques et sûrteté de fonctionnement des systèmes de production*, page Chapitre 11. Hermès.

Combacau, M., Berrut, P., Charbonnaud, F., and Khatab, A. (2000a). Réflexions sur la terminologie : Surveillance - supervision. In *Groupement pour la recherche en Produtique, Systèmes de Production Sûrs de Fonctionnement*.

Combacau, M., Berrut, P., Charbonnaud, F., Khatab, A., and Zamai, E. (2000b). Supervision and ponitoring of production systems. In *Proceeding of Second Conference on Management and Control of Production and Logistics, MCPL'2000*.

Commission, I. E. (1988). Iec 60848 : Grafcet specifiaction language for sequential function charts.

Correcher, A., Garcia, E., Morant, F., Quiles, E., and Blasco-Gimenez, R. (2003). Intermittent failure diagnosis in indstrial processes. In *IEEE International Symposium on Industrial Electronics*, volume 2, pages 723–728.

Couffin, S. and Lesage, J. J. (2000). Formal verification of the sequential part of plc programs. In *Proceeding of the 5th Workshop on Discrete Event Systems, WODES'2000, Ghent (Belgium)*, pages 247–254.

David, R. (1995). Grafcet : A powerful tool for specification of logic controllers. *IEEE transactions on control, systems technology*, 3(3).

David, R. and Alla, H. (2005). *Discrete, Continuous and Hybrid Petri Nets*. Springer.

Derbel, H., Yeddes, M., Hadj-Alouane, N. B., and Alla, H. (2006). Diagnosis of a class of timed discrete event systems. In *Proceedings of the 8th International Workshop on Discrete event systems, WODES'06*, Michigan, USA. published by IEEE (ISBN 1-4244-0053-8 and IEEE catalog number 06EX1259).

Engell, S., Lohmann, S., and Stursberg, O. (2005). Verification of embedded supervisory controllers considering hybrid plant dynamics. *International Journal of Software Engineering and Knowledge Engineering*, 15(2).

Frank, P. (1996). Analytical and qualitative model-based fault diagnosis - a survey and some new results. *European Journal of Control*, 2 :6–28.

Frank, P. M. (1990). Fault diagnosis in dynamic systems using anaytical and knowledge based redundancy - a survey and some new rsults. *Automatica*, 26(3) :459–474.

Frehse, G. (2005). Phaver : Algorithmic verification of hybrid systems past hytech. In *Proceedings of the Fifth International Workshop on Hybrid Systems : Computation and Control*, pages 258–273.

Frensel, G. and Bruiijn, P. M. (1998). From modelling control systems using grafcet to analyzing systems using hybrid automata. In *American Control Conference*, volume 2, pages 704–705.

Gardey, G., Roux, O. H., and Roux, O. F. (2006). State space computation and analysis of time Petri nets. *Theory and Practice of Logic Programming (TPLP). Special Issue on Specification Analysis and Verification of Reactive Systems*, 6(3) :301–320.

Ghazel, M., Toguéni, A., and Bigang, M. (2005). A monitoring approach for discrete events systems based on a timed perti net model. *Proceedings of 16th IFAC World Congress*.

Henzinger, T. (1996). The theory of hybrid automata. In *Proceeding of the 11th Annual IEEE Symposium on Logic in Computer Science, LICS'96*, pages 278–292.

Henzinger, T., Ho, P.-H., and Wong-Toi, H. (1997). Hytech : A model checker for hybrid systems. In *Software Tools for Technology Transfer*, pages 110–122.

Henzinger, T. A., Kopke, P. W., Puri, A., and Varaiya, V. (1998). What's decidable about hybrid automata ? *Journal of Computer Sciences*, 57.

Holloway, L. E. and Krogh, B. H. (1991). Monitoring behavioral evolution for on-line fault-detection. In *Proceedings of the IFAC SAFEPROCESS'91 conference*, Baden.

Huang, Z., Chandra, V., Jiang, S., and Kumar, R. (1996). Modeling discrete event systems with faults using a rules based modeling formalism. *Mathematical Modeling of Systems*, 1(1).

International Electrotechnical Commission, C. N. . (2003). Programmable controllers - programming languages, iec 61131-3.

Isermann, R. (1984). Process fault detection based on modelling and estimation methods - a survey. *Automatica*, 20(4) :387–404.

Jiang, S., Kumar, R., and Garcia, H. E. (2003). Diagnosis of repeated/intermittent failure in discrete event systems. *IEEE Transaction On Robotic and Automation*, 19(2) :310–323.

Klein, S., Lesage, J. J., and Litz, L. (2005). Identification comportementale des systèmes logiques en vue de leur surveillance. In *Numéro spécial du Jouranl européen des systèmes automatisés pour les actes du colloque français MSR'05*, volume 39, pages 111–126, Grenble, France.

Koutsoukos, X., He, K. X., Lemmon, M., and Antsaklis, P. (1998). Timed petri nets in hybrid systems ; stability and supervisory control. *Discrete Event Dynamic Systems*, 8(2) :137–173.

Lime, D. and Roux, O. H. (2004). A translation based method for the timed analysis of scheduling extended time petri nets. In *25th IEEE International Real time Systems Symposuim*, pages 187–196, Lisbon, Portugal.

Lunze, J. (2000). *Diagnosis of quantised qystems by means of timed discrete-event representations*. Lecture notes in computer science, Springer Berlin.

Merlin, P. M. (1974). *A study of the recoverability of computing systems*. PhD thesis, Department of Information and Computer Science, University of California, Irvine.

Murata, T. (1989). Petri nets : properties, analysis and applications. *Proceedings of the IEEE*, 77(4) :541–580.

Nourelfath, M. (1997). *Extension de la théroie de la supervision à la surveillance et à la commande des systèmes à événements discrets : application à la sécurité opérationelle des systèmes de production*. PhD thesis, L'Institut National des Sciences Appliquées de Lyon, INSA.

Nourelfath, M. (2000). The real-time monitoring of an experimental manufacturing cell. In *Canadian Conference on Electrical and Computer Engineering*, volume 1, pages 297–301.

O. Stursberg, S. L. and Engell, S. (2005). Improving dependability of logic controllers by algorithmic verification. In *Proceeding of the 16th IFAC World Congress*, volume -, pages –.

Pandalai, D. N. and Holloway, L. E. (1996). Condition templates : Improved distributed models for automated fault monitoring of manufacturing systems. In *Proceeding of the 1996 IEEE International Conference on Robotic and Automation*.

Pandalai, D. N. and Holloway, L. E. (2000). Template languages for fault monitoring of timed discrete event processes. *IEEE Transactions On Automatic Control*, 45(5) :868–882.

Philippot, A. (2006). *Contribution au diagnostic décentralisé des systèmes à événements discrets : Application aux systèmes manufacturiers*. PhD thesis, L'Université de Reims Champagne Ardenne.

Rayhane, H. (2004). *Surveillance des systèmes de production automatisés : Détection et Diagnostic*. PhD thesis, Institut National Polytechnique de Grenoble, INPG.

Remelhe, M., Lohmann, S., Stursberg, O., Engell, S., and Bauer, N. (2004). Algorithmic verifiaction of ligic controllers given as sequential function charts. In *IEEE International Symposium on Computer Aided Control Systems Design*, volume -, pages 53–58, Taipei.

Roux, O. H. (2005). *Vérification des réseaux de Petri temporels et à chronomètres*. PhD thesis, Université de Nantes.

Roux, O. H. and Lime, D. (2004). Time petri nets with inhibitor hyperarcs. formal semantics and state space computation. In *25th International Conference on theory and application of Petri nets*, pages 371–390, Bologna, Italy.

R.Vallett (1995). Petri nets for control and monitoring : specification, verification, implementation. *workshop on Analysis and design of Event-Driven Operations in Process Systems*.

Sahraoui, A., Atabakhche, H., Courvoisrier, A., and Valette, R. (1987). Joining petri nets and knowledge based systems for monitoring purposes. *IEEE, International conference robotics*, pages 1160–1165.

Sampath, M., Senqupta, R., Lafortune, S., Sinnamohideen, K., and Teneketzis, C. (1995). Diagnosability of discrete-event systems. *IEEE Transaction on Automatic Control*, 40(9) :1555–1575.

Sampath, M., Senqupta, R., Lafortune, S., Sinnamohideen, K., and Teneketzis, C. (1996). Failure diagnosis using discrete- event models. *IEEE Transaction on Control Systems Technology*, 4(2) :105–124.

Sampath, M., Sungupta, R., Lafortune, S., Sinnamohideen, K., and Teneketize, D. (1994). Diagnosability of discrete event systems. In *Proceedings of the 11th International Conference Analysis Optimization of Systems : Discrete event systems*, Sophia-Antipolis, France.

SAVA, A. (2001). *Sur la synthèse de la commande des systèmes à événements discrets temporisés*. PhD thesis, Institute National Polytechnique de Grenoble, INPG.

Silva, B. I., Richeson1, K., Krogh, B., and Chutinan, A. (2001). Modeling and verifying hybrid dynamic systems using checkmate. *Journal européen des systèmes automatisés*, 35(4).

Smet, O. D., Couffin, S., Rossi, O., Canet, G., Lesage, J. J., Schnoebelen, P., and Papini, H. (2000). Safe programming of plc using formal verification methodes. In *Proceeding of the 4th International conference on Industrial Control Programming ICP'2000, Utrech, The Netherlands*, pages 73–78. PLCOpen, Zaltbommel, The Netherlands.

Srinivasan, V. and Jafari, M. (1993). Fault detection/monitoring using time petri nets. *IEEE Transactions on Systems, Man and Cybernetics*, 23 :1155–1162.

Stursberg, O. and Lohmann, S. (2005). Analysis of logic controllers by transformation of sfc into timed automata. In *Proceeding of the 44th IEEE Conference on Decision and Control*, volume -, pages 7720–7725, Seville, Spain.

Stursberg, O., Lohmann, S., and Engell, S. (2005). Improving dependability of logic controllers by algorithmic verification. In *Proceeding of 16th IFAC World Congress.*

Sujit, R. D. and Holloway, L. E. (2000). Characterizing a confidence space for discrete event timing for fault monitoring using discrete sensing and actuation signals. *IEEE Transaction on Systems, Man and Cybernetics*, 30(1) :52–66.

The MathWorks, w. (2004). Stateflow 6 : Design and simulate event-driven systems.

Toguyeni, A. (1992). *Surveillance et diagnostic en ligne dans les systèmes flexibles de l'industrie manufacturière.* PhD thesis, Université Paul Sabatier, Toulouse.

Tripakis, S. (2002). Fault diagnosis for timed automata. *Proceeding 7th International Symposium on Formal Techniques in Real-Time and Fault Tolerant Systems (FTRTFT'02)*, 2791 of Lecture Notes in Computer Science :205–224.

Valette, R., Cardoso, J., and Dubois, D. (1989). Monitoring manufacturing systems by means of petri nets with imprecise marking. In *Proceedings of IEEE International Symposium on Intelligent Control*, pages 233–238, Albany, USA.

Zad, S. H., Kwong, R. H., and Wonham, W. M. (2003). Fault diagnosis in discrete-event systems : Framework and model reduction. *IEEE Transactions On Automatic Control*, 48(7) :1199–1212.

Zad, S. H., Kwong, R. H., and Wonham, W. M. (2005). Fault diagnosis in discrete-events systems : Incorporating timing information. *IEEE transaction on automatic control*, 50(7).

Zaytoon, J. (2002). On the recent advances in grafcet. *Production Planning and Control*, 13(1).

Zwingelstein, G. (1995). *Diagnostic des défaillances.* Traité des nouvelles Technologies, série Diagnostic et Maintenance, Hèrmes.

Surveillance des systèmes à événements discrets commandés : Conception et implémentation en utilisant l'automate programmable industriel

Résumé : Ce mémoire de thèse présente une approche pour la surveillance des systèmes à événements discrets commandés. L'étude se restreint aux défauts interruptibles : intermittents et permanents. Le comportement acceptable de ces systèmes est introduit afin d'accroître la disponibilité des systèmes commandés. Ce comportement présente une tolérance aux défauts intermittents. Une démarche de construction du système de surveillance est présentée. Nous modélisons, dans un premier temps, le comportement du système sujet aux défauts interruptibles par un automate à chronomètres. Nous appliquons, dans un deuxième temps, une procédure de synthèse à cet automate. Cette procédure est basée aux opérateurs d'analyse en avant et en arrière de l'automate.
Un nouveau modèle appelé réseaux de Petri à chronomètres post- et Pré-initialisés est également présenté. Ce modèle général du RdP a été utilisé, dans le cadre de ce mémoire, pour modéliser le comportement des systèmes sujets aux défauts interruptibles.
L'implémentation de notre méthode de surveillance se fait par un automate programmable industriel. Pour ce faire, le système de surveillance étant sous la forme d'un automate à chronomètres est traduit structurellement en programme SFC.

Mots clefs : Systèmes à événements discrets, Surveillance, automate à chronomètres, Analyse d'atteignabilité, Réseaux de Petri temporels, SFC.

Monitoring of the controlled discrete events systems : Conception and implementation by using programmable logic controller

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. We model, firstly, the behavior of the systems by a stopwatch automaton. Secondly, we apply to this automaton a synthesis procedure. This procedure is based on the forward and backward reachability analysis of the stopwatch automaton.
A new model called Post- and Pre-initialized stopwatch Petri nets is also presented. This model is used to model the behavior of systems subjected to the interrupting faults.
The implementation of our monitoring approach is done by using a Programmable Logic Controller. The monitoring system is translated structurally into a SFC program.

Key words : Discrete events systems, Monitoring, Stopwatch automata, Reachability analysis, Timed Petri nets, SFC.

Discipline : Automatique-Productique

Laboratoire GIPSA-lab - ENSE^3 - BP 46, 38402 Saint-Martin d'Hères, FRANCE.