Titre: Title:	Synthèse de superviseurs pour des systèmes à événements discrets partiellement observés
Auteur: Author:	Mohamed Hichem Lamouchi
Date:	2000
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Lamouchi, M. H. (2000). Synthèse de superviseurs pour des systèmes à événements discrets partiellement observés [Master's thesis, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/8908/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/8908/
Directeurs de recherche: Advisors:	John G. Thistle
Programme: Program:	Unspecified

UNIVERSITÉ DE MONTRÉAL

SYNTHÈSE DE SUPERVISEURS POUR DES SYSTÈMES À ÉVÉNEMENTS DISCRETS PARTIELLEMENT OBSERVÉS

MOHAMED HICHEM LAMOUCHI
DÉPARTEMENT DU GÉNIE ÉLECTRIQUE ET DE GÉNIE INFORMATIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRE ÈS SCIENCES APPLIQUÉES (M.Sc.A.)
(SECTION AUTOMATION ET SYSTÈMES)
JUIN 2000

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé:

SYNTHÈSE DE SUPERVISEURS POUR DES SYSTÈMES À ÉVÉNEMENTS DISCRETS PARTIELLEMENT OBSERVÉS

présenté par: LAMOUCHI Mohamed Hichem

en vue de l'obtention du diplôme de: Maître ès science appliquées

a été dûment accepté par le jury d'examen constitué de:

M. MULLINS John, Ph.D., président
M. THISTLE John G., Ph.D., membre et directeur de recherche
Mme. BERGERON Anne, Ph.D., membre

À ma famille

REMERCIEMENTS

En premier lieu, je remercie mon directeur de recherche, Monsieur John Thistle, professeur agrégé au Département de génie électrique et de génie informatique, qui a suivi l'évolution de ce travail avec patience et compétence. Je le remercie pour sa disponibilité permanente et la pertinence de ses suggestions, et je lui suis reconnaissant pour son appui financier qui m'a permis de venir à Montréal pour faire ma maîtrise. Je tiens également à remercier tous les professeurs de la section Automation et Systèmes qui m'ont initié à l'automatique, ainsi que tous mes anciens professeurs à la Faculté des Sciences de Tunis qui ont contribué indirectement à la réalisation de ce travail. J'aimerais aussi remercier la secrétaire de la section Automation et Systèmes, Mme Marie-Claude Nadeau, pour sa gentillesse et son sourire. Plus généralement, je veux exprimer le plaisir que j'ai eu à travailler au sein du laboratoire de recherche en Automatique, et je voudrais remercier tous mes collègues qui ont contribué à créer une ambiance enrichissante et propice à la recherche. Finalement, je voudrais exprimer ma gratitude à ma famille pour le soutien et l'encouragement qu'elle m'a apporté tout au long de mes études.

RÉSUMÉ

Dans ce mémoire, on introduit le problème de supervision des séquences infinies générées par les systèmes à événements discrets partiellement observés et nous donnons une procédure pour la résolution de ce problème, dans le cas où le procédé et les spécifications sont représentés par des automates finis sur mots infinis et le masque d'observation par un automate fini de Moore. L'idée principale de notre construction est de ramener ce problème à celui de la supervision sous observations complètes, pour lequel on peut appliquer des résultats qui existent déjà. Ceci constitue une extension de travaux antérieurs puisque, d'une part, nous nous intéressons à la supervision et, aussi, parce que nous utilisons un masque d'observation plus général. Nous montrons également dans ce mémoire que le même problème, posé dans un contexte de supervision décentralisé devient non-décidable même si nous nous intéressons uniquement à la supervision des mots finis.

ABSTRACT

In this report, we introduce an infinite-string supervisory control problem for partially-observed discrete event systems. We also give a procedure for the effective resolution of this problem, for the case where the plant and specification langage are represented by finite ω-automata (automata on infinite strings), and the observation mask by a finite Moore automaton. The main idea of our procedure is the reduction of the problem to the case of complete observations, for which we can adapt some results from automata theory. This extends earlier control-theoretic results by taking account of infinite strings and also by introducing a more general observation mechanism. We also show that a natural extension to decentralized control is undecidable, even if the plant, the specification languages and the masks are represented by finite automata. The undecidability result carries over to the finite-string case.

TABLE DES MATIÈRES

LISTE DES FIGURES

INTRODUCTION

La théorie des systèmes à événement discrets est une théorie assez récente (elle a été introduite dans les années 80) qui a été fondée par P. J. Ramadge et W. M. Wonham [RW89] et enrichie par les travaux d'autres chercheurs [LW88b, GR88b, Zho92, TW94a, TW94b].

Dans le cadre de cette théorie, le terme Système à Événements Discrets (SÉD) désigne un système dynamique qui interagit avec son environnement externe au moyen d'un ensemble d'événements qui se produisent de façon asynchrone dans le temps, et des concepts liés à la commande d'un tel système sont étudiés d'un point de vue qualitatif (commandabilité, observabilité, commande décentralisée, commande hiérarchique, etc.).

Depuis son apparition, cette théorie a suscité un intérêt grandissant puisqu'elle a pu être appliquée à des domaines très variés (ateliers de fabrications, réseaux de communication, etc.) où il s'agissait de résoudre des problèmes très complexes de commande et de coordination.

Dans le cadre de ce mémoire, nous nous intérèssons à la supervision sous observations partielles, qui correspond au cas où le superviseur ne dispose pas d'une information complète concernant la dynamique du SÉD qu'il supervise. En effet il arrive que le superviseur soit incapable de distinguer entre deux événements, ou même qu'il soit incapable de détecter l'occurrence de certains événements, parce que ces derniers correspondent à des informations qui ne sont pas prises en compte dans la fonction de sortie du système. Par exemple si le générateur modélise une usine chimique, il

se peut que l'événement "la température du réservoir x a atteint un seuil critique" ne soit pas observé par le superviseur, parce que l'appareil de mesure correspondant à ce réservoir est tombé en panne ou n'existe pas. Cet exemple met en évidence les enjeux économiques de ce problème, en effet un superviseur qui serait capable de contrôler une telle usine permettrait des économies importantes puisqu'on ne serait pas obligé de mettre des appareils de mesure pour tous les réservoirs.

Le problème d'observations partielles se pose aussi dans un contexte de supervision répartie. Dans un tel schéma de commande, la tâche globale du procédé est décomposée en sous-tâches et il s'agit de concevoir un superviseur pour chaque sous-système. Évidemment chaque superviseur aura une vision locale du générateur global (il se peut même que ces superviseurs soient physiquement distants l'un de l'autre).

Évoquons finalement le cas de la supervision hiérarchique[Zho92], qui peut lui aussi poser le problème des observations partielles. Dans ce contexte il s'agit de transposer un problème de bas niveau (qui correspond au système réel), à un niveau d'abstraction plus élévé; on parle dans ce cas de modèle de haut niveau et il s'agit de résoudre le problème à ce niveau là. Bien entendu, une fois qu'un superviseur de haut niveau a été synthétisé, il faut en déduire un superviseur bas niveau pour le système réel. Il est évident qu'il y'a des événements de bas niveau qui ne seront pas (complètement ou partiellement) observés par le superviseur de haut niveau, puisqu'en faisant le passage d'un modèle d'implémentation à un modèle abstrait, il

y a des détails bas niveau qui ne seront pas pris en compte au haut niveau.

Vu donc l'importance du problème des observations partielles, beaucoup de travaux ont tenté de résoudre ce problème. La principale difficulté est qu'une solution optimale au problème n'existe pas. Pour contourner cette difficulté, on s'est contenté de chercher une solution sous-optimale. La principale limitation de ces travaux est la structure simpliste du mécanisme d'observation et, surtout, le fait que la plupart d'entre eux ne s'intéressent qu'aux séquences finies générées par un SÉD. Parmi ces travaux on peut citer ceux de Lin et Wonham [LW88b], ainsi que ceux de Cieslak, Desclaux, Fawaz et Varaiya [CDFV88].

Dans [LW88b] l'alphabet est partionné en symboles observables et symboles non-observables, le mécanisme d'observation est simplement la projection naturelle des mots finis définis sur l'alphabet global sur les mots finis définis sur l'alphabet des événements observables. Les séquences finies générées en boucle fermée sous la supervision d'un superviseur non-bloquant forment un langage observable, fermé et commandable. L'observabilité n'étant pas conservée sous réunion arbitraire, il n'existe donc pas une solution optimale au problème posé, qui correspond au superviseur non-bloquant le plus permissif qui permet de répondre à une spécification donnée. Il faut donc se contenter de chercher une solution sous-optimale qui correspond au plus grand sous-langage normal, fermé et commandable de l'intersection entre le langage de spécification et le langage généré par le SÉD. Dans [CDFV88], la structure du mécanisme d'observation a été enrichie en introduisant un nouvel al-

phabet qui correspond à ce que le superviseur peut observer, ceci pour tenir compte du cas où un événement peut être observé partiellement. La solution du problème de supervision que les auteurs ont considéré a été caractérisée comme étant un sous-langage de l'intersection entre le langage de spécification et le langage généré par le SÉD, qui soit invariant et controllable (ce qui correspond aux propriétés d'observabilité, de fermeture et de commandabilité de [LW88b]).

L'originalité de notre travail est que, d'une part, nous nous intéressons à la supervision sous observations partielles des séquences infinies générées par un SÉD, ce qui constitue une généralisation des travaux de Thistle et Wonham[TW94a,TW94b] qui ont introduit une extension de la théorie à un contexte de mots infinis. Leurs travaux permettent la synthèse d'un superviseur à observations totales qui, au moyen d'une famille d'entrées de commande et en se basant sur ce que le SÉD à déjà généré, limite les actions que le SÉD peut faire à la prochaine étape de telle façon qu'ultimement les séquences infinies d'événements générées par le système en boucle fermée soient contenues dans les ω-langages de spécifications. L'intérêt d'utiliser les ω-langages pour modéliser les SÉDs réside principalement dans le fait qu'ils offrent un modèle plus riche et qu'ils permettent d'analyser des propriétés de bon fonctionnement telle que la vivacité. Nous allons introduire un nouveau problème qui constitue une généralisation du problème introduit par Thistle et Wonham (qu'ils ont appelé SCP^{ω}) au cas des observations partielles et nous allons donner une procédure qui permet de résoudre ce problème. Notre approche de résolution se base sur l'idée

d'effectuer des étapes d'abstractions successives qui vont permettre de ramener le problème de départ à une variante du SCP^{ω}, et il suffit dans ce cas d'adapter la procédure de synthèse de Thistle et Wonham.

D'autre part notre travail constitue une extension de ce qui a été déjà fait pour la supervision des mots finis sous observations partielles, parce que nous allons introduire un nouveau mécanisme d'observation qui a la particularité d'être doté d'une "mémoire", dans le sens où l'observation d'un événement dépend de l'historique des événements générés avant lui.

Dans la dernière partie de notre mémoire, nous allons nous intéresser à la supervision répartie. Nous allons introduire un problème classique de supervision répartie et nous allons montrer que la question d'existence d'une solution à ce problème est non-décidable, en s'inspirant de travaux similaires concernant les machines alternantes, à joueurs multiples, information incomplète et espace borné[PR79] ainsi que la synthèse répartie des systèmes réactifs[?]. Ce résultat s'applique aussi bien à la supervision des mots infinis que celle des mots finis, problème qui a été déjà traité dans la littérature[CDFV88, LW88a, RW92].

Dans le prochain chapitre nous allons faire un survol de la théorie de Ramadge et Wonham et nous allons donner quelques résultats concernant la supervisions sous observations partielles et la supervision répartie. Le chapitre 2 sera consacré à l'extension de la théorie au cas des mots infinis; les résultats présentés sont principalement ceux de Thistle et Wonham. Dans le chapitre 3 nous allons définir le

problème auquel on s'intérèsse et donner les conditions d'existence d'une solution. Au chapitre 4, nous enrichissons la structure de notre SÉD pour tirer profit de toute l'information disponible, ce qui donne une nouvelle structure de SÉD, appelée SÉD contrôlé. Nous allons, entre autres, montrer que la supervision d'un SÉD est équivalente à celle du SÉD contrôlé qui lui est associé. Le chapitre 5 donne notre approche de résolution qui permet de synthétiser un superviseur pour le SÉD contrôlé et à partir duquel on peut déduire un superviseur pour le SÉD de départ. Finalement, dans le chapitre 6, nous donnons une extension du problème de supervision introduit dans le chapitre 3 au cas de la supervision répartie et nous montrons que ce nouveau problème est non-décidable. Nous terminons le mémoire avec une conclusion dans laquelle nous donnons des extensions possibles de nos résultats.

CHAPITRE 1

SYSTÈMES À ÉVÉNEMENTS DISCRETS : DÉFINITIONS ET NOTIONS DE BASE

1.1 Introduction

Un système à événements discrets est un système caractérisé par un ensemble d'états et un ensemble d'événements qui se produisent à des instants discrets. Ces événements font passer le système d'un état à l'état suivant. Ils correpondent soit à des actions que le procédé effectue, soit à des interactions entre le procédé et son environnement externe. Généralement on représente un tel système avec un automate fini sur mots finis, où les états de l'automate correspondent aux états de l'automate correspondent aux états du système et l'alphabet à l'ensemble des événements. La dynamique du système étant décrite par la structure de transition de l'automate. Les mots acceptés par cet automate représentent les séquences d'actions qui correspondent à un bon fonctionnement du système. Le problème typique qu'on se propose de résoudre dans le cadre de la théorie des SÉDs est de trouver un superviseur pour un SÉD donné qui, en empêchant à chaque étape de fonctionnement le système d'effectuer certaines actions, fait en sorte que les mots générés par le système en boucle fermée soient contenus dans un langage de spécification. En fait, un superviseur dans ce cas n'est autre qu'une fonction partielle qui à chaque état

de l'automate associe un ensemble d'actions que le procédé est autorisé à faire à ce niveau, c'est pour cette raison qu'on parle de superviseur par retour d'état.

Dans ce chapitre nous présentons la théorie des systèmes à événements discrets telle qu'introduite par P. J. Ramadge et W. M. Wonham. Nous commençons par quelques notions de base sur la théorie des automates et des langages formels qui serviront comme modèle logique des SÉDs. La théorie introduite par Ramadge et Wonham est présentée à la section 1.3. Le cas particulier des SÉDs partiellement observés est présenté à la section 1.4. L'autre problème auquel on s'intéresse dans ce mémoire, qui est celui de la supervision répartie, est introduit dans l'avant-dernière section. Nous terminons le chapitre par une petite conclusion.

1.2 La théorie des automates et des langages formels

1.2.1 Langages finis

Définition 1.1 *Un* alphabet *est un ensemble fini et non vide de symboles.*

Définition 1.2 *L'ensemble de toutes les séquences finies d'éléments d'un alphabet* Σ *est désigné par* Σ^+.

Soit $m = \sigma_1\sigma_2...\sigma_n$, $n \geq 1$, *où* $\sigma_i \in \Sigma$, *pour* $0 < i \leq k$, *une telle séquence. On dit alors que* m *est un* mot *sur l'alphabet* Σ; *la* longueur $|m|$ *du mot* m *est* n.

Définition 1.3 *Soit* $u = \sigma_1...\sigma_n$ *et* $v = \sigma'_1...\sigma'_m$ *deux éléments de* Σ^+. *La* concaténation *de* u *et* v *est le mot*

$$uv = \sigma_1...\sigma_n\sigma'_1...\sigma'_m.$$

Remarque 1.1 *L'élément neutre pour l'opération de concaténation est le mot vide* 1_Σ*:*

$$m.1_\Sigma = 1_\Sigma.m = m,\ \forall m \in \Sigma^+$$

Notation 1.1 *On désigne par* Σ^* *la réunion* $\Sigma^+ \cup \{1_\Sigma\}$.

Définition 1.4 *Soit* Σ *un alphabet et* u *et* v *deux mots définis sur* Σ. *On dit que* v *est un* facteur *de* u *s'il existe* $u_1, u_2 \in \Sigma^*$ *tels que*

$$u = u_1 v u_2$$

Si $u_1 = 1_\Sigma$ *(resp.* $u_2 = 1_\Sigma$*), alors* v *est un* préfixe *(resp.* suffixe*) de* u. *Si* $v \neq 1_\Sigma$ *et* $v \neq u$, *alors* v *est un* facteur propre *de* u.

Définition 1.5 *Soit* Σ *un alphabet et* L *un sous-ensemble de* Σ^*. *Alors on dit que* L *est un* langage *défini sur* Σ.

Remarque 1.2 *À partir de l'opération de concaténation des mots, il est possible de définir une opération similaire pour les langages :*

$$L_1 L_2 := \{uv \in \Sigma^* : u \in L_1 \& v \in L_2\} \text{ avec } L_1, L_2 \subseteq \Sigma^*$$

Définition 1.6 *L'*étoile, L^*, *de* $L \subseteq \Sigma^*$ *est définie par*

$$L^* := \bigcup_{n \geq 0} L^n$$

où

$$L^0 := \{1_\Sigma\}$$

et

$$L^{i+1} := L^i L, \text{ pour tout } i \geq 0.$$

Définition 1.7 *L'*étoile propre, L^+, *de* $L \subseteq \Sigma^*$ *est définie par*

$$L^+ := \bigcup_{n>0} L^n$$

1.2.2 Automates finis sur mots finis

On modélise le comportement d'un SÉD par un langage formel sur un certain alphabet (celui des événements) et on utilise les automates finis en tant que représentation finie de ces langages formels sous forme de machines à états.

Définition 1.8 *Un* automate *fini est un quintuplet*

$$\mathcal{A} = (\Sigma, X, \delta, x_0, T)$$

où

Σ *est un alphabet fini*

X *est un ensemble fini d'*états

$\delta : \Sigma \times X \longrightarrow 2^X$ *est une fonction partielle de transition*

$x_0 \in X$ *est l'*état initial, *et*

$T \subseteq X$ *est un ensemble d'*états terminaux .

Notation 1.2 *Pour tout* $\sigma \in \Sigma$, $x \in X$, $\delta(\sigma, x)!$ *veut dire que* δ *est définie pour l'élément* (σ, x).

Définition 1.9 *Si* $(\forall \sigma \in \Sigma)\ (\forall x \in X)\,[|\delta(\sigma, x)| \leq 1]$, *alors* $\mathcal{A}$ *est dit* déterministe. *Dans ce cas,* δ *devient une fonction (partielle):*

$$\delta : \Sigma \times X \longrightarrow X$$

Remarque 1.3 *On étend* $\delta : \Sigma \times X \longrightarrow 2^X$ *à une fonction*

$$\begin{aligned} \widehat{\delta} : \ \Sigma^* \times X &\longrightarrow 2^X \\ (1_\Sigma, x) &\longmapsto \{x\} \\ (s\sigma, x) &\longmapsto \bigcup_{x' \in \delta(s,x)} \delta(\sigma, x') \end{aligned}$$

Si $\mathcal{A}$ est déterministe, on peut écrire :

$$\begin{aligned} \widehat{\delta}: \ \Sigma^* \times X &\longrightarrow X \\ (1_\Sigma, x) &\longmapsto x \\ (s\sigma, x) &\longmapsto \delta\left(\sigma, \hat{\delta}(s,x)\right) \end{aligned}$$

Définition 1.10 *Le langage* reconnu *(ou* accepté*) par $\mathcal{A}$ est*

$$L(\mathcal{A}) = \left\{m \in \Sigma^* : \hat{\delta}(m, x_0) \cap T \neq \emptyset\right\}$$

Remarque 1.4 *$L(\mathcal{A})$ est l'ensemble des mots qui mènent (par un chemin quelconque) de x_0 à T.*

Remarque 1.5 *Si $\mathcal{A}$ est déterministe, alors*

$$L(\mathcal{A}) = \left\{m \in \Sigma^* : \hat{\delta}(m, x_0) \in T\right\}$$

Définition 1.11 *Soit $\mathcal{A} = (\Sigma, X, \delta, x_0, T)$ un automate fini. Le* composant accessible *de $\mathcal{A}$ est*

$$\mathcal{A}_c(\mathcal{A}) = (\Sigma, X_{ac}, \delta_{ac}, q_0, T_{ac})$$

où

$$X_{ac} = \{x \in X : (\exists m \in \Sigma^*)\,[x \in \delta(m, x_0)]\};$$

$$\delta_{ac} : \textit{La restriction de } \delta \textit{ à } \Sigma \times X, \textit{ et}$$

$$T_{ac} = T \cap X_{ac}.$$

Notons qu'on a $L(\mathcal{A}) = L(\mathcal{A}_c(\mathcal{A}))$.

Définition 1.12 $\mathcal{A}$ *est dit* accessible *si*

$$\mathcal{A} = \mathcal{A}_c(\mathcal{A}).$$

$\mathcal{A}$ *est dit* co-accessible *si*

$$(\forall m \in \Sigma^*)\,[\delta(m, x_0)! \Rightarrow (\exists m' \in \Sigma^*)\,[\delta(mm', x_0) \cap T \neq \emptyset]]$$

Définition 1.13 *Le* produit *de deux automates finis* $\mathcal{A}_i = (\Sigma, X_i, \delta_i, x_{0_i}, T_i)$, i=*1,2, est l'automate fini*

$$\mathcal{A}_1 \times \mathcal{A}_2 = (\Sigma, X_1 \times X_2, \delta_1 \times \delta_2, (x_{0_1}, x_{0_2}), T_1 \times T_2)$$

où

$$\delta_1 \times \delta_2 : \Sigma \times (X_1 \times X_2) \longrightarrow 2^{X_1 \times X_2}$$

$$(\sigma, (x_1, x_2)) \longmapsto \{(x_1', x_2') \in X_1 \times X_2 : x_1' \in \delta_1(\sigma, x_1)\ \&\ x_2' \in \delta_2(\sigma, x_2)\}$$

et on a $L(\mathcal{A}_1 \times \mathcal{A}_2) = L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$.

1.3 La théorie Ramadge-Wonham

1.3.1 Définition d'un SÉD

Définition 1.14 [RW89] *Un* Système à événements discrets *est un système dynamique défini par un ensemble fini d'états et caractérisé par une trajectoire dans l'espace d'état constante par morceaux. Les changement d'états, qui se produisent de façon asynchrone dans le temps, sont appelés* événements. *Ils sont étiquetés avec les symboles d'un certain alphabet (l'étiquette reflètant le phénomène physique qui a provoqué le changement d'état).*

Dans le cadre de cette théorie, on considère un modèle logique de SÉD, c'est-à-dire qu'on ne modélise pas le temps et qu'on s'intérèsse uniquement à l'ordre selon lequel les événements ont eu lieu. On modélise donc un SÉD par un ensemble de mots sur l'alphabet des événements, qui correspondent aux séquences d'événements que le procédé peut générer. On le représente sous forme de générateur de langages formels :

$$G = (\Sigma, Q, \delta, q_0, Q_b)$$

où:

Σ est un alphabet (représentant les événements).

Q est un ensemble d'états (peut-être infini).

$\delta : \Sigma \times Q \longrightarrow Q$ est une fonction de transition (généralement partielle).

$q_0 \in Q$ est l'état initial; et

$Q_b \subseteq Q$ est l'ensemble des états buts.

Remarque 1.6 *Dans tout le cadre de ce mémoire, on s'intéresse au cas où Q est fini.*

Comme dans le cas des automates, $\delta(\sigma, q)!$ signifie que $\delta(\sigma, q)$ est définie et on étend $\delta : \Sigma \times Q \longrightarrow Q$ en une fonction :

$$\begin{aligned} \delta : \ \Sigma^* \times Q &\longrightarrow Q \\ (1_\Sigma, q) &\longmapsto q \\ (k\sigma, q) &\longmapsto \delta(\sigma, \delta(k, q)) \end{aligned}$$

Le générateur est donc une machine qui démarre à l'état intial et génère des séquences d'événements en exécutant les transitions qui sont définies par sa fonction de transition. Ce générateur joue le rôle du procédé à superviser (cela peut être une chaîne de fabrication, une voiture automatisée,...etc.).

Définition 1.15 *Le* langage engendré *par G est :*

$$L(G) := \{m \in \Sigma^* : \delta(m, q_0)!\}$$

Le langage achevé *de G est:*

$$L_b(G) := \{m \in \Sigma^* : \delta(m, q_0) \in Q_b\}$$

Les mots de $L(G)$ correspondent à toutes les séquences possibles d'événements que le SÉD modélisé par G peut générer, alors que ceux de $L_b(G)$ représentent les séquences d'événements qui correspondent à des tâches complétées (c.-à.-d. celles qui correspondent à un bon fonctionnement du procédé).

Définition 1.16 *Pour tout langage $L \subseteq \Sigma^*$, $\bar{L}$ désigne l'ensemble des préfixes de tous les mots de L, soit:*

$$\bar{L} := \{m \in \Sigma^* : \exists n \in \Sigma^*, mn \in L\}.$$

On l'appelle la fermeture *de L.*

Définition 1.17 *Soient $L, K \subseteq \Sigma^*$ alors:*

1. *Si $L = \bar{L}$, alors L est dit* fermé.

2. *Si $L = \bar{L} \cap K$, alors L est dit* fermé par rapport *à K.*

Remarque 1.7 *$L(G)$ est fermé par définition.*

Définition 1.18 *On dit qu'un SÉD G est* non bloquant *si $L(G) = \bar{L}_b(G)$.*

De façon informelle, cette définition veut dire que tout mot engendré par G peut être étendu de façon à devenir un mot achevé de G. Ceci correpond à une propriété de non-blocage dans le sens où le système modélisé par G est capable de mener à bien toute action qu'il commence.

1.3.2 Le produit synchrone

Soient $L_1 \subseteq \Sigma_1^*$ et $L_2 \subseteq \Sigma_2^*$, avec $\Sigma = \Sigma_1 \cup \Sigma_2$.

On définit les *projections naturelles* suivantes, pour $i = 1, 2$:

$$\begin{aligned} P_i : \ \Sigma^* &\longrightarrow \Sigma_i^* \\ 1_\Sigma &\longmapsto 1_{\Sigma_i} \\ k\sigma &\longmapsto P_i(k)\sigma \quad \text{si } \sigma \in \Sigma_i \\ k\sigma &\longmapsto P_i(k) \quad \text{si } \sigma \notin \Sigma_i \end{aligned}$$

L'effet de P_i sur un mot s est tout simplement d'effacer tous les symboles qui n'appartiennent pas à Σ_i.

Définition 1.19 *Le* produit synchrone *de L_1 et L_2 est défini par :*

$$L_1 \| L_2 := P_1^{-1} L_1 \cap P_2^{-1} L_2$$

où

$$P_i^{-1} L_i := \{ s \in \Sigma^* : P_i(s) \in L_i \}$$

Lorsqu'il s'agit d'un système complexe, pour ne pas avoir à traiter un générateur dont la taille est excéssivement grande, on décompose un tel système en sous-systèmes modélisés chacun par un générateur local ayant une taille raisonnable. Ces sous-systèmes fonctionnent de manière concurentielle afin de réaliser la tâche globale du système. Ainsi, le langage engendré par le générateur global n'est autre que le produit synchrone des différents langages générés par les générateurs locaux.

1.3.3 Théorie de supervision

Pour introduire un mécanisme de commande, on suppose qu'il existe des événements qui peuvent être empêchés par un mécanisme de contrôle externe. On est capable ainsi de contrôler l'évolution du système en l'empêchant d'atteindre des états non désirables et en lui interdisant aux moments opportuns de générer les événements qui lui permettraient d'atteindre ces états.
Supposons donc que l'alphabet Σ (ou l'ensemble des événements) est divisé en deux parties disjointes :

- Σ_c : l'ensemble des événements commandables (qui peuvent être empêchés à tout moment par un superviseur), et
- Σ_u : l'ensemble des événements non commandables (par exemple une machine qui tombe en panne dans un atelier de fabrication, ou un client qui arrive dans une file d'attente,...).

Définition 1.20 *Soit* $G = (\Sigma, Q, \delta, q_0, Q_b)$ *un générateur, où* $\Sigma = \Sigma_u \dot{\cup} \Sigma_c$. *L'ensemble*

des entrées de commande *est donné par :*

$$C := \{\Gamma \subseteq \Sigma : \Gamma \supseteq \Sigma_u\}$$

La commande d'un SÉD consiste à observer l'évolution du système, et à choisir à chaque étape une entrée de commande qui permet de restreindre l'ensemble des événements que le générateur peut générer à cette étape (c.-à.-d. l'ensemble des actions que le procédé, modélisé par le SÉD, peut éffectuer à cet instant) afin que les séquences d'événements générées ainsi respectent certaines conditions correpondant à un bon fonctionnement du système. Le choix d'une entrée de commande pour une étape donnée, est basé sur l'observation du préfixe qui a été généré jusqu'à cet instant. On parle dans ce cas de *supervision* d'un SÉD. Le problème qui se pose donc, étant donné un procédé modélisé par un SÉD, est celui de synthétiser un superviseur qui fait en sorte que le système en boucle fermée respecte les règles de bon fonctionnement et qui sont fournies comme spécifications. Typiquement, on souhaiterait que ce superviseur soit le plus permissif possible.

Définition 1.21 *Un* superviseur *pour G est une fonction définie comme suit :*

$$V : L(G) \longrightarrow C$$

Notation 1.3 *On dénote le système en boucle fermée par V/G.*

Définition 1.22 *Le* langage engendré *par G sous la supervision de V, qu'on note*

$L(V/G)$, est défini comme suit :

(i) $1_\Sigma \in L(V/G)$

(ii) $\forall k \in \Sigma^*, \forall \sigma \in \Sigma$:

$$(k \in L(V/G) \text{ et } \sigma \in V(s) \text{ et } k\sigma \in L(G)) \Longleftrightarrow (k\sigma \in L(V/G))$$

Définition 1.23 *Le* langage achevé *par G sous la supervision de V, $L_b(V/G)$, est défini par:*

$$L_b(V/G) := L(V/G) \cap L_b(G)$$

Remarque 1.8 *$L_b(V/G)$ est l'ensemble des mots de $L_b(G)$ qui survivent à la supervision de V.*

Définition 1.24 *On dit qu'un superviseur V pour un SÉD G est* complet *si* $L(V/G) \subseteq \text{dom}(V)$.

Définition 1.25 *Soit G un SÉD. On dit qu'un superviseur V est* non-bloquant *pour G si et seulement si:*

$$L(V/G) = \overline{L_b(V/G)}$$

Comme nous l'avons déjà mentionné, l'objectif d'un superviseur est de faire en sorte que le comportement du système en boucle fermée respecte les contraintes imposées par les spécifications. Typiquement ces spécifications sont données sous forme de langages sur l'alphabet des événements qui représentent le comportement

en boucle fermée souhaité. La question qui se pose donc en définissant de telles spécifications, est de savoir s'il est possible de trouver un superviseur qui soit capable de forcer le procédé à générer uniquement des mots contenus dans le langage de spécification. En d'autres termes, étant donné un SÉD G dont le comportement (en boucle ouverte) est défini par L, est-il possible de trouver un superviseur tel que le comportement du système en boucle fermée soit $K \subseteq L$? Pour répondre à cette question, on utilise la propriété de commandabilité :

Définition 1.26 *Soit $G = (\Sigma, Q, \delta, q_0, Q_b)$ un générateur et soit $K \subseteq \Sigma^*$ un langage. Le langage K est dit* commandable *par rapport à G si et seulement si*

$$\overline{K}\Sigma_u \cap L(G) \subseteq \overline{K}$$

où Σ_u désigne l'ensemble des événements non commandables.

De façon informelle, K est dit commandable par rapport à G si l'appartenance à la fermeture de K est invariante sous la génération d'événements non commandables. En effet, comme ces événements ne peuvent pas être empêchés, alors pour tout bout de chemin qui a été généré à l'intérieur de K (c-à-d un élément de $\overline{K}$), si un événement non commandable est physiquement possible, alors son occurrence ne doit pas nous faire sortir de $\overline{K}$. Concernant la question qui a été évoquée plus haut, les fondateurs de cette théorie ont prouvé les résultats suivants [RW89]:

Théorème 1.1 *Soit $G = (\Sigma, Q, \delta, q_0, Q_b)$ un générateur et soit $K \subseteq L_b(G)$ un*

ensemble non vide. Alors il existe un superviseur V non bloquant pour G tel que $L_b(V/G) = K$ si et seulement si:

- *K est commandable par rapport à G, et*
- *K est fermé par rapport à $L_b(G)$.*

Corollaire 1.1 *Soit $G = (\Sigma, Q, \delta, q_0, Q_b)$ un générateur et soit $K \subseteq L_b(G)$ un ensemble non vide et fermé. Alors il existe un superviseur V pour G tel que $L(V/G) = K$ si et seulement si K est commandable par rapport à G.*

1.3.4 Le plus grand sous-langage commandable

Lorsqu'on essaye de synthétiser un superviseur pour un SÉD tel que le système en boucle fermée se comporte conformément à la spécification, on cherche typiquement à avoir le superviseur le plus permissif possible. Techniquement, étant donné un générateur G et une spécification $E \subseteq L_b(G)$, ceci revient à calculer le plus grand sous-langage de $E \cap L_b(G)$ commandable par rapport à G et fermé par rapport à $L_b(G)$. Ramadge et Whonham ont montré que cette solution existe toujours.

Définition 1.27 *Soit $G = (\Sigma, Q, \delta, q_0, Q_b)$ un générateur avec $\Sigma = \Sigma_u \dot{\cup} \Sigma_c$ et soit $E \subseteq \Sigma^*$. Définissons la classe de langages :*

$$C(E) = \{\, K \subseteq E : K \text{ est commandable par rapport à } G \,\}.$$

La structure de cette famille de langages est caractérisée par le résultat suivant:

Proposition 1.1 *$(C(E), \subseteq)$ est un ensemble ordonné, non vide et fermé sous réunions arbitraires. En particulier, $(C(E), \subseteq)$ contient sa propre borne supérieure, notée* $\sup C(E)$.

Définition 1.28 *Soit $E, L \subseteq \Sigma^*$. Alors E est dit* $L-$fermé *si*

$$\bar{E} \cap L \subseteq E$$

Proposition 1.2 *Soit $E \subseteq \Sigma^*$ $L_b(G)$-fermé. Alors* $\sup C(E \cap L_b(G))$ *est fermé par rapport à $L_b(G)$.*

D'après la proposition 1.1, il est clair que même si le langage de spécification lui même n'est pas commandable, il est possible de s'approcher (dans le sens de l'inclusion) de ce langage, en considérant le plus grand sous-langage commandable de ce langage. Comme il a été prouvé que cette approximation commandable est aussi fermée par rapport à $L_b(G)$ (d'après la proposition 1.2), on peut la considérer comme étant la solution optimale dans le sens ou elle correspond au superviseur le plus permissif possible.

1.4 Observations partielles

Jusqu'à maintenant, il été toujours supposé que le superviseur est capable d'observer tous les événements générés par le SÉD qu'il supervise. Malheureusement dans nombre de cas réels, ceci n'est pas toujours vérifié. Dans cette section nous allons faire

un survol de ce qui a été déjà fait concernant le problème de supervision sous observations partielles, notamment les travaux de Lin et Wonham [LW88b], ainsi que ceux de Cieslak, Desclaux, Fawaz et Varaiya [CDFV88].

1.4.1 Supervision avec observations partielles

Pour modéliser un SÉD partiellement observable on introduit un nouvel alphabet Σ_o qui correspond aux événements qui sont observables par le superviseur.

Définition 1.29 *Soit $V : L(G) \to C$ un superviseur pour G, et $P : \Sigma^* \longrightarrow \Sigma_o^*$ la projection naturelle qui éffface les événements non observables. Dénotons par $P \uparrow L(G)$ la restriction de P à $L(G)$. Alors V est dit un* superviseur sous observations partielles *si:*

$$ker\,(P \uparrow L(G)) \subseteq ker V$$

1.4.2 Langages observables

Afin de caractériser les langages sur Σ qui correspondent à un comportement en boucle fermée de G sous la supervision d'un superviseur avec observations partielles, il est utile d'introduire les ensembles suivants:

Définition 1.30 *Soit $K \subseteq \Sigma^*$, pour chaque mot $m \in \Sigma^*$, on définit:*

$$A_K(m) = \begin{cases} \{\sigma \in \Sigma : m\sigma \in \bar{K}\} & , m \in \bar{K} \\ \emptyset & , \textit{autrement} \end{cases}$$

$$NA_K(m) = \begin{cases} \{\sigma \in \Sigma : m\sigma \in L(G)\ \bar{K}\} & , m \in \bar{K} \\ \emptyset & , \textit{autrement} \end{cases}$$

L'ensemble $A_K(m)$ est l'ensemble des événements admissibles *(qui sont les événements qui permettent de rester dans $\bar{K}$). Alors que l'ensemble $NA_K s$ est l'ensemble des événements* non admissibles *(c.-à.-d. ceux qui nous font sortir de $\bar{K}$).*

Il est nécessaire aussi d'introduire la relation binaire suivante:

Définition 1.31 *Soit G un générateur et $K \subseteq \Sigma^*$. La relation binaire de K-*compatibilité, *notée $comp_K$, est définie sur Σ^* comme suit: $(m, m') \in comp_K$ ssi*

(i) $A_K(m) \cap NA_K(m') = \emptyset = A_K(m') \cap NA_K(m)$

(ii) $m \in \bar{K} \cap L_b(G)$ & $m' \in \bar{K} \cap L_b(G) \Rightarrow (m \in K \Leftrightarrow m' \in K)$

Définition 1.32 *Soit $P : \Sigma^* \to \Sigma_o^*$ la projection naturelle. Alors un langage $K \subseteq \Sigma^*$ est* observable *par rapport à (G, P), si*

$$ker P \subseteq comp_K$$

(c.-à.-d. $(\forall m, m' \in \Sigma^*)\ P(m) = P(m') \Rightarrow (m, m') \in comp_K$)

On a alors le résultat fondamental suivant:

Théorème 1.2 *Soit $K \subseteq L_b(G)$ non vide. Alors il existe un superviseur sous observations partielles non bloquant V pour G tel que $L_b(V/G) = K$ ssi*

(i) K est commandable par rapport à G;

(ii) K est observable par rapport à (G, P); et

(iii) K est fermé par rapport à $L_b(G)$

Corollaire 1.2 *Soit $K \subseteq L(G)$ non vide. Alors il existe un superviseur sous observations partielles non bloquant V pour G tel que $L(V/G) = K$ ssi*

(i) K est commandable par rapport à G;

(ii) K est observable par rapport à (G, P); et

(iii) K est fermé.

L'ensemble des langage observables par rapport à (G, P) n'est pas fermé sous réunion. Donc une solution optimale au problème n'existe pas (c.-à.-d. un plus grand sous-langage commandable, observable et fermé). Il faut donc se contenter d'une solution sous-optimale qui est le plus grand sous-langage commandable, normal et fermé.

1.4.3 Langages normaux

Remarque 1.9 *Soit $P : \Sigma^* \to \Sigma_o^*$ la projection naturelle définie dans la section précédente. On étend P comme suit:*

$$\begin{aligned} P: \ 2^{\Sigma^*} &\to 2^{\Sigma_o^*} \\ M &\mapsto \{P(m) : m \in M\} \end{aligned}$$

Définition 1.33 *Soit $N \subseteq M \subseteq \Sigma^*$, on dit que N est (M,P)−*normal *si*

$$N = M \cap P^{-1}(P(N))$$

où

$$P^{-1}: \ 2^{\Sigma_0^*} \ \rightarrow \ 2^{\Sigma^*}$$

$$L \ \mapsto \ \{m \in \Sigma^* : P(m) \in L\}$$

Cette propriété de normalité par rapport à (M,P) signifie que pour vérifier si un mot l de M appartient à N ou non, il suffit de vérifier que la projection de l appartient à $P(N)$.

Proposition 1.3 *Soit $K \subseteq L(G)$ fermé (ou $K \subseteq L_b(G)$ fermé par rapport à $L_b(G)$). Si $\bar{K}$ est $(L(G),P)$-normal, alors K est (G,P)-observable.*

Notation 1.4 *Soit $E, M \subseteq \Sigma^*$. Alors*

$$N(E,M) = \{K \subseteq E : K \ est \ (M,P) - normal\}$$

Cet ensemble présente des propriétés algébriques intéressantes comme le montre la proposition suivante:

Proposition 1.4 *$N(E,M)$ est fermé sous intersections et réunions arbitraires.*

On introduit aussi l'ensemble suivant :

Notation 1.5 *Soit $E, M \subseteq \Sigma^*$. Alors*

$$\bar{N}(E,M) = \left\{K \subseteq E : \bar{K} \text{ est } (M,P) - normal\right\}$$

Proposition 1.5 *Soit $E \subseteq M$. Alors $\bar{N}(E,M)$ est fermé sous réunions arbitraires.*

Notation 1.6 *Définissons pour tout langage $E \subseteq \Sigma^*$ la classe de langages*

$$F_b(E) := \{K \subseteq E : K \text{ est fermé par rapport à } L_b(G)\}.$$

Et désignons par $F_bC\bar{N}(E)$ l'intersection

$$F_b(E) \cap C(E) \cap \bar{N}(E, L(G))$$

Proposition 1.6 *$F_bC\bar{N}(E)$ est fermé sous réunions arbitraires. On a alors*

$$\sup F_bC\bar{N}(E, L(G)) \subseteq E$$

Ceci veut dire que si on remplace la condition d'observabilité par rapport à (G,P) par la condition plus exigente de $(L(G),P)$-normalité, alors il existe un superviseur "le moins restrictif possible" ce qui donne lieu au langage achevé en boucle fermée $\sup F_bC\bar{N}(E)$.

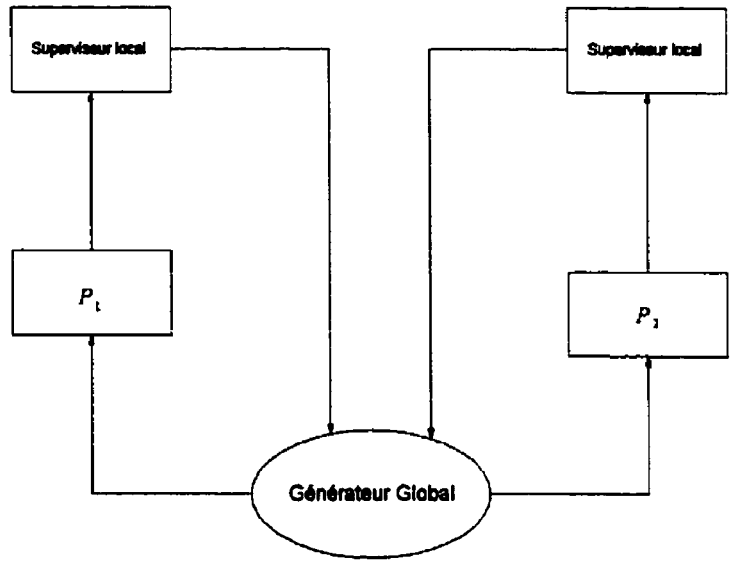


Figure 1.1: Supervision répartie

1.5 Supervision répartie

Une architecture de supervision répartie est caractérisée par la présence de plusieurs superviseurs possèdant chacun une vision locale du générateur (voir figure 1.1). Ces superviseurs peuvent même être physiquement distants l'un de l'autre. Dans une telle configuration, la tâche globale du système est décomposée en sous-tâches et chaque module de supervision (c.-à.-d. chaque superviseur local) s'occupe du bon déroulement d'une sous-tâche. L'avantage de la supervision répartie est de faciliter la résolution du problème global en le décomposant en sous-problèmes moins complexes, et d'éviter par la même occasion l'explosion combinatoire induite par la synthèse monolithique. Néanmoins ceci peut poser un problème de conflit et de synchronisation entre les superviseurs locaux. Un autre problème posé par la

supervision répartie est la question d'optimalité.

1.5.1 Superviseur réparti

On suppose qu'on se donne une famille de *sous-alphabets locaux*, non vides et pas nécessairement disjoints deux à deux $\{\Sigma_i \subset \Sigma, i \in I\}$. Soit P_i, $i \in I$ les projections naturelles asssociées à ces sous-alphabets:

$$P_i : \ 2^{\Sigma^*} \ \rightarrow \ 2^{\Sigma_i^*}$$

$$\sigma \ \mapsto \begin{cases} \sigma & , \text{si } \sigma \in \Sigma_i \\ 1_{\Sigma_i} & , \text{si } \sigma \notin \Sigma_i \end{cases}$$

Le comportement local du système tel que observé par le superviseur V_i est $P_i(L(G))$.

Dénotons par $\Sigma_{ci} = \Sigma_c \cap \Sigma_i$ et $\Sigma_{ni} = \Sigma_n \cap \Sigma_i$ respectivement les événements locaux commandables et non commandables par V_i.

Définition 1.34 *Un* superviseur local V_i *correspondant à l'alphabet* Σ_i *est une fonction :*

$$V_i : P_i(L(G)) \rightarrow C_i$$

où $C_i = \{\Gamma \subseteq \Sigma_i : \Gamma \supseteq \Sigma_{ni}\}$

Remarque 1.10 *L'action du superviseur local* V_i *sur le générateur global* G *est*

équivalente à celle du superviseur global suivant:

$$\begin{aligned} \tilde{V}_i : \ L(G) &\to C \\ m &\mapsto V_i(P_i(m)) \cup (\Sigma \setminus \Sigma_i) \end{aligned}$$

où $C = \{\Gamma \subseteq \Sigma : \Gamma \supseteq \Sigma_n\}$

Définition 1.35 *La* conjonction *de n superviseurs locaux* $V_i : P_i(L(G)) \to C_i$, $i = 1, 2, ..., n$, *est donnée par:*

$$\begin{aligned} \bigwedge_{i\in\{1,2,...,n\}} V_i : \ L(G) &\to C \\ m &\mapsto \bigcap_{i\in\{1,2,...,n\}} \tilde{V}_i(m) \end{aligned}$$

Définition 1.36 *Un superviseur est dit* réparti *s'il consiste en une conjonction de superviseurs locaux correspondant aux sous-alphabets locaux.*

Bien entendu ce n'est toujours pas possible d'associer à n'importe quel sous-langage $K \subseteq L(G)$ un superviseur répartie V tel que $L(V/G) = K$.

Notation 1.7 *Pour tout* $\sigma \in \Sigma_c$, *on associe l'ensemble suivant:*

$$I_\sigma := \{i : \sigma \in \Sigma_i\}$$

Définition 1.37 *Un langage* K *est dit* $\{P_i\}$-observable *si pour tout* $\sigma \in \Sigma_c$ *et* $\{m_j : j \in I_\sigma\} \subseteq K$, *et* $m \in K$ *si*

(i) $m\sigma \in L(G)$

(ii) $P_j(m) = P_j(m_j), \forall j \in I_\sigma$

(iii) $m_j\sigma \in K, \forall j \in I_\sigma$

impliquent

$$m\sigma \in K$$

Concernant l'existence d'un supervieur réparti, signalons le résultat suivant[CDFV88]:

Proposition 1.7 *Soit $K \subseteq L(G)$ non vide. Alors il existe un superviseur réparti pour G tel que $L(V/G) = K$ ssi K est fermé, commandable par rapport à G et $\{P_i\}$-observable pour $i \in I$.*

1.6 Conclusion

Dans ce chapitre nous avons fait un survol de la théorie des systèmes à événements discrets telle qu'introduite par Ramadge et Wonham. Nous avons présenté également des extensions de cette théorie au cas des observations partielles et celui de la supervision répartie. Une autre extension de cette théorie est celle qui porte sur la commande des séquences infinies d'événements générées par un SÉD, laquelle sera présentée au chapitre suivant.

CHAPITRE 2

COMMANDE DES SÉQUENCES INFINIS GÉNÉRÉES PAR LES SYSTÈMES À ÉVÉNEMENTS DISCRETS

2.1 Introduction

La théorie introduite par P. J. Ramadge et W. M. Wonham ne s'intéresse qu'aux séquences finies d'événements générées par un SÉD. C'est pourquoi les spécifications ne permettent que l'expression de propriétés de "bon fonctionnement" qui consiste typiquement à s'assurer que le système en boucle fermée ne génère pas des événements indésirables. Malheureusement de telles spécifications ne permettent pas d'exprimer des propriétés de "vivacité" qui consistent à s'assurer qu'ultimement le système en boucle fermée va générer certains événements "souhaitables". Pour pallier ce problème, ainsi que pour pouvoir étudier des cas où le procédé ne s'arrête jamais de fonctionner (comme c'est le cas par exemple d'un réseau téléphonique), il a fallu étendre la théorie afin qu'elle puisse prendre en compte les mots infinis générés par de tels SÉDs. Ceci est le fruit de travaux de plusieurs chercheurs, notamment P. J. Ramadge [Ram89], R. Kumar, V. Garg, S.I. Marcus [KGM91], Golaszewski [GR88a], J. G. Thistle et W. M. Wonham [TW88, Thi91, TW94a, TW94b, TW91, ?, ?]. Comme il s'agit d'étudier des séquences infinies, un SÉD est modélisé maintenant par un langage de mots infinis (c'est-à-dire un ω-langage) sur l'alphabet des événements.

Pour pouvoir se doter des outils nécessaires à la résolution des nouveaux problèmes dus au fait qu'on ne s'intérèsse plus uniquement aux séquences finies d'événements générées en boucle fermée, il faut se restreindre au cas des langages ω-réguliers, c.-à.-d. les ω-langages qui peuvent être représentés par un automate fini sur mots infinis (ou encore ω-automate). Ceci permet de représenter le procédé, le superviseur, le système en boucle fermée ainsi que les spécifications par des ω-automates. Un ω-automate consiste essentiellement en un ensemble fini d'états et une structure de transitions. L'acceptation d'un mot est liée à l'ensemble des états visités infiniment souvent lors de la génération du mot. Cette classe particulière d'automates a été introduite dans les années 60 par Büchi [?] pour résoudre des problèmes liés à des logiques mathématiques. Depuis ces automates ont été utilisés dans plusieurs domaines, notammnent l'informatique et la logique [?]. Leur utilisation dans le domaine de l'automatique et plus précisémment dans le cadre de la théorie des SÉDs a permis de faire le lien entre des problèmes de supervision et des problèmes classiques en informatique et en logique. En effet, un résultat de base, que nous allons rappeler dans ce chapitre et qui concerne la caractérisation du sous-ensemble de commandabilité par une formule de point fixe[TW94a], constitue une solution au fameux problème de Church [?] (il s'agit du problème de construction d'un ω-automate tel que le langage ω-régulier reconnu par cet automate satisfait une formule d'une certaine logique) et une solution également au problème du vide pour automates sur arbres infinis [?]. Le problème fondamental qu'il faut résoudre consiste typiquement à synthétiser un

superviseur pour un SÉD tel que le comportement en boucle fermée du système soit compris (dans le sens de l'inclusion) entre deux ω-langages de spécifications. Les travaux de J. G Thistle et W. M. Wonham ont permis de ramener ce problème à celui du controle d'un ω-automate de façon à ce qu'il satisfait sa propre condition d'acceptation (à condition que les mots générés respectent des conditions de vivacité), et ont permis également de résoudre ce problème en utilisant encore une fois le calcul de point fixe. Dans la section 2 nous rappeleons des notions de base sur la théorie des ω-langages. Les ω-automates sont présentés à la section suivante. La quatrième section constitue un survol de l'extension de la théorie Ramadge-Wonham au cadre des mots infinis, l'accent est mis notamment sur les travaux de J. G. Thistle et W. M. Wonham. Nous terminons avec une conclusion dans laquelle nous expliquons le problème auquel nous nous intéréssons dans la première partie de ce mémoire.

2.2 Les ω-langages

Dans cette section nous allons rappeler quelques notions de bases sur les mots infinis[TW94b] (pour plus de détail voir [?]).

Définition 2.1 *Soit Σ un alphabet fini. Alors*

Σ^ω *: dénote l'ensemble de toutes les séquences infinies sur* Σ.

Σ^∞ *: dénote l'ensemble de toutes les séquences sur* Σ *($\Sigma^\infty = \Sigma^\omega \cup \Sigma^*$).*

Définition 2.2 *Soit Σ un alphabet fini, un ω-langage est un sous ensemble de Σ^ω.*

Remarque 2.1 *Comme dans le cas des mots finis, on définit un produit de concaténation entre deux mots $f \in \Sigma^*$ et $i \in \Sigma^\infty$, qui induit un opérateur de concaténation entre un langage $F \subseteq \Sigma^*$ et un ω-langage $I \subseteq \Sigma^\omega$, qu'on appelle* produit *de F et I.*

Définition 2.3 *Soit F un langage et I un langage ou un ω-langage. On définit le* quotient I/F *comme suit:*

$$I/F = \{m \in \Sigma^\omega : (\exists f \in F)(fm \in I)\}$$

Définition 2.4 (ω-fermeture de Kleene) *Soit $F \subseteq \Sigma^*$, la ω-fermeture de Kleene, qu'on note F^ω est le produit infini de F avec lui même.*

Définition 2.5 *On définit la fonction suivante:*

$$\begin{aligned} \text{pre} : \ 2^{\Sigma^\infty} &\to 2^{\Sigma^*} \\ L &\mapsto \{k \in \Sigma^* : (\exists s \in \Sigma^\infty)(ks \in L)\} \end{aligned}$$

Et pour tout $L \subseteq \Sigma^\omega$, on appelle pre$(L)$ *le* préfixe *de L.*

Définition 2.6 *Soit L un langage, la* limite *de L est*

$$\lim(L) = \text{pre}^{-1}(L) \cap \Sigma^\omega$$

où $\text{pre}^{-1}(L) = \{m \in \Sigma^\infty : \text{pre}(m) \subseteq L\}$

Remarque 2.2 *La limite d'un langage L donne tous les mots infinis dont les préfixes sont contenus dans L.*

Définition 2.7 *La* ω-fermeture *est définie comme étant la fonction suivante:*

$$\begin{aligned} \text{clo}: \ 2^{\Sigma^\omega} &\rightarrow 2^{\Sigma^\omega} \\ L &\mapsto \lim(\text{pre}(L)) \end{aligned}$$

qui associe à chaque ω-langage L, l'ensemble de tous les mots infinis dont les prefixes sont contenus dans celui de L, qu'on appelle la ω-fermeture *de L.*

Définition 2.8 *Soit L et K deux ω-langages, on dit que L est* ω-fermé *si* $\text{clo}(L) = L$. *On dit qu'il* ω-fermé par rapport à K *si* $\text{clo}(L) \cap K = L$

Remarque 2.3 *Les langages qui sont ω-fermés sont des langage qui sont entièrement déterminés par leurs préfixes.*

2.3 Les automates finis sur mots infinis

Les automates finis sur mots infinis (qu'on appelle encore ω-automates) ont une structure similaire à celle des automates finis sur mots finis, sauf que l'acceptation d'un mot ne dépend plus du dernier état visité mais plutôt de l'ensemble des états visités un nombre infini de fois. Il existe différents types de conditions d'acceptation auquelles sont associées différentes classes d'ω-automates. Dans cette section nous allons nous concentrer sur les trois classes d'automates qui nous intéréssent dans le cadre de ce travail, à savoir :

- les automates de Büchi,
- les automates de Rabin, et
- les automates de Streett.

Ces automates peuvent être déterministes, non-déterministes ou universels. Dans tout ce qui va suivre nous allons considérer les ω-automates comme des générateurs de langages formels plutôt que comme des machines qui lisent des séquences d'évévnements.

Définition 2.9 *Un ω-automate est un quintuplet:*

$$\mathcal{A} = (\Sigma, X, x_0, \delta, M)$$

Où Σ est un alphabet fini (c'est l'alphabet des événements), X est un ensemble fini d'états, $x_0 \in X$ est l'état initial, $\delta : X \times \Sigma \to 2^X$ est une fonction de transition (partielle), et M est une condition d'acceptation.

Définition 2.10 *Si $(\forall \sigma \in \Sigma)\ (\forall x \in X)\ [|\delta(\sigma, x)| \leq 1]$, alors $\mathcal{A}$ est dit* déterministe.

Remarque 2.4 *Comme dans le chapitre précédent, on étend δ en une fonction:*

$$\begin{aligned} \delta : \ \Sigma^* \times X &\to 2^X \\ (1_\Sigma, x) &\mapsto \{x\} \\ (m\sigma, x) &\mapsto \bigcup_{x' \in \delta(m,x)} \delta(\sigma, x') \end{aligned}$$

Remarque 2.5 *Comme d'habitude on utilise la notation $\delta(m,x)!$ pour signaler que $\delta(m,x)$ est définie.*

Définition 2.11 *Soit $\mathcal{A}$ un ω-automate, et soit $s \in \Sigma^{\infty}$. Un* chemin *de s dans $\mathcal{A}$ est une fonction totale $\pi : \mathrm{pre}(s) \to X$ qui vérifie les deux conditions suivantes:*

(i) $\pi(1_{\Sigma}) = x_0$

(ii) $\forall k \in \mathrm{pre}(s), \sigma \in \Sigma$:

$$k\sigma \in \mathrm{pre}(s) \Rightarrow \pi(k\sigma) \in \delta\left(\sigma, \pi(k)\right).$$

Elle associe au mot s une séquence possible d'états de $\mathcal{A}$ qu'on visite, en générant ce mot.

Remarque 2.6 *Dans certaines références on parle d'*exécution *de s par $\mathcal{A}$* [9, Saf88].

Définition 2.12 *Soit $\mathcal{A}$ un ω-automate, et soit $s \in \Sigma^{\infty}$. On dit que s est* généré *par $\mathcal{A}$ si s admet un chemin sur $\mathcal{A}$.*

Définition 2.13 *Soit $\mathcal{A}$ un ω-automate, on désigne par $L(\mathcal{A})$ l'ensemble des mots finis générés par $\mathcal{A}$.*

Définition 2.14 *Soit s un mot généré par un ω-automate $\mathcal{A}$,* l'ensemble de récurence *d'un chemin π de s dans $\mathcal{A}$ est l'ensemble suivant :*

$$\Omega_{\pi} := \left\{x \in X : \left|\pi^{-1}(x)\right| = \omega\right\}$$

qui contient l'ensemble des états qui sont visités un nombre infini de fois en parcourant le chemin $\pi(s)$.

La première classe d'automates qui a été introduite est celle des automates de Büchi :

Définition 2.15 *Un* automate de Büchi *sur l'alphabet* Σ *est un quintuplet:*

$$\mathcal{A} = (\Sigma, X, x_0, \delta, T)$$

Où Σ *est un alphabet fini,* X *est un ensemble fini d'états,* $x_0 \in X$ *est l'état initial,* $\delta : X \times \Sigma \to 2^X$ *définit la structure de transition, et* $T \subseteq X$ *est l'ensemble des états acceptants.*

Définition 2.16 *Soit* $\mathcal{A} = (\Sigma, X, x_0, \delta, T)$ *un automate de Büchi. On dit que* $\mathcal{A}$ *reconnaît le mot infini* s *s'il existe un chemin* π *de* s *dans* $\mathcal{A}$ *qui vérifie la condition suivante:*

$$\Omega_\pi \cap T \neq \emptyset$$

En d'autres termes un automate de Büchi accepte un mot infini s'il est capable de le générer en visitant infiniment souvent au moins un état de T.

Définition 2.17 *On désigne par* $S(\mathcal{A})$ *l'ensemble des mots infinis reconnus par* $\mathcal{A}$.

Définition 2.18 *Les langages* ω-réguliers *sont les langages qui sont reconnus par un automate de Büchi.*

Remarque 2.7 *Contrairement au cas des automates finis sur mots finis, les automates de Büchi déterministes n'ont pas le même pouvoir d'expression que leurs semblables non déterministes; ils reconnaissent plutôt une sous-classe de la classe des langages* ω*-réguliers.*

Une autre classe d'automates qui nous serait utile est celle des automates de Rabin :

Définition 2.19 *Un* automate de Rabin *sur l'alphabet* Σ *est un quintuplet:*

$$\mathcal{A} = (\Sigma, X, x_0, \delta, \{(R_p, I_p) : p \in P\})$$

Où Σ *est un alphabet fini,* X *est un ensemble fini d'états,* $x_0 \in X$ *est l'état initial,* $\delta : X \times \Sigma \to 2^X$ *définit la structure de transition, et* $\{(R_p, I_p) : p \in P\}$ *est une famille de "paires acceptantes", avec pour tout* $p \in P$ $(R_p, I_p) \in 2^X \times 2^X$.

Définition 2.20 *Soit* $\mathcal{A} = (\Sigma, X, q_0, \delta, \{(R_p, I_p) : p \in P\})$ *un automate de Rabin. On dit qu'un mot* $s \in \Sigma^\omega$ *généré par* $\mathcal{A}$ *est accepté par* $\mathcal{A}$ *si la condition suivante est vérifiée:*

$$\exists p \in P : \Omega_\pi \cap R_p \neq \emptyset \ \& \ \Omega_\pi \subseteq I_p$$

avec π *un chemin de* s *dans* $\mathcal{A}$.

Définition 2.21 *On désigna par $S(\mathcal{A})$ l'ensemble des mots infinis reconnus par $\mathcal{A}$.*

Informellement, un automate de Rabin accepte un mot infini s'il est capable de le générer en visitant R_p un nombre infini de fois et $X \setminus I_p$ un nombre fini de fois pour un $p \in P$.

Remarque 2.8 *La définition ci-dessus [TW94b] constitue une légère modification de la définition standard d'un automate de Rabin.*

La troisième classe à laquelle nous allons nous intérèsser est celle des automates de Streett, dont la structure est semblable à celle des automates de Rabin, mais dont la condition d'acceptation a la forme duale de la condition d'acceptation de Rabin:

Définition 2.22 *Un* automate de Streett *sur l'alphabet Σ est un quintuplet:*

$$\mathcal{A} = (\Sigma, X, x_0, \delta, \{(R_p, I_p) : p \in P\})$$

Où Σ est un alphabet fini, X est un ensemble fini d'états, $x_0 \in X$ est l'état initial, $\delta : X \times \Sigma \to 2^X$ définit la structure de transition, et $\{(R_p, I_p) : p \in P\}$ est une famille de paires acceptantes, avec pour tout $p \in P$, $(R_p, I_p) \in 2^X \times 2^X$.

Définition 2.23 *Soit $\mathcal{A} = (\Sigma, Q, q_0, \delta, M)$ un automate de Streett. On dit qu'un mot $s \in \Sigma^\omega$ généré par A est accepté par A si la condition suivante est vérifiée:*

$$\forall p \in P : \Omega_\pi \subseteq R_p \text{ ou } \Omega_\pi \cap I_p \neq \emptyset$$

avec π un chemin de s dans $\mathcal{A}$.

Définition 2.24 *On désigne par $S(\mathcal{A})$ l'ensemble des mots infinis reconnus par $\mathcal{A}$.*

Autrement dit, un automate de Streett accepte un mot infini s'il est capable de le générer en respectant la condition suivante pour tout $p \in P$: s'il visite $X \setminus R_p$ un nombre infini de fois alors il visite aussi I_p un nombre infini de fois.

Mentionons finalement la classe des $\forall$-automates qui vont nous servir à modéliser le comportement infini d'un SÉD partiellement observable.
Un $\forall$-automate (qu'on appelle aussi *automate universel*) est un automate d'états finis non déterministe, qui accepte une séquence infinie s si tous les chemins associés à s sont des chemins acceptants, c-à-d qui respectent la condition d'acceptation.

Définition 2.25 *Un* automate universel *de Rabin sur l'alphabet Σ est un quintuplet:*

$$\mathcal{A} = (\Sigma, X, x_0, \delta, \{(R_p, I_p) : p \in P\})$$

Où Σ, X, $x_0 \in X$, $\delta : X \times \Sigma \to 2^X$ et $\{(R_p, I_p) : p \in P\}$ jouent le même rôle que dans la structure d'un automate de Rabin.

Définition 2.26 *Soit $\mathcal{A} = (\Sigma, Q, q_0, \delta, \{(R_p, I_p) : p \in P\})$ un automate universel. On dit qu'un mot $s \in \Sigma^\omega$ généré par $\mathcal{A}$ est accepté par $\mathcal{A}$ si la condition suivante est vérifiée pour tout chemin π de s dans $\mathcal{A}$:*

$$\exists p \in P : \Omega_\pi \cap R_p \neq \emptyset \ \& \ \Omega_\pi \subseteq I_p$$

2.4 Les SÉDs sur mots infinis

Dans cette section nous allons présenter une extension du modèle de base au cas des mots infinis:

2.4.1 Modélisation du comportement infini d'un SÉD

Ces notions ont été introduites par Ramadge, pour plus détails voir [Ram89].

Définition 2.27 *Un* SÉD *est une paire* $G = (L, S) \in 2^{\Sigma^*} \times 2^{\Sigma^\omega}$. *On appelle* L *le* comportement fini *de* G, *ou encore le* $*$-comportement *et* S *le* comportement infini *de* G, *ou encore le* ω-comportment, *où* Σ *est l'alphabet des événements.*

Remarque 2.9 *On suppose que* $\mathrm{pre}(L) = L$, *et que* $\mathrm{pre}(S) \subseteq L$

Définition 2.28 *On dit qu'un SÉD* $G = (L, S)$ *est* non bloquant *si* $\mathrm{pre}(S) = L$

2.4.2 Superviseurs

Étant donné un SÉD G dont l'alphabet des événements est Σ, alors on définit une famile d'entrées de commande qui va permettre de contrôler l'évolution du sytème, on considère ici une généralisation de la notion classique de famille d'entrées de commande qui a été faite par Golaszewski et Ramadge [GR87].

Définition 2.29 *Soit* Σ *l'alphabet des événements, une* famille d'entrées de commande *est une famille non vide de sous-ensembles d'événements :* $C \subseteq 2^{\Sigma}$.

Remarque 2.10 *Tous les résultats que nous allons présenter ici (établis principalement par J. G. Thistle et W. M. Wonham[TW94a, TW94b]) sont valables dans le cas où C est fermée sous réunion arbitraire (c.-à.-d. $\Gamma_i, \Gamma_j \in C \Rightarrow \Gamma_i \cup \Gamma_j \in C$). Ceci ne constitue pas une perte de généralité, puisqu'on pourra toujours considérer des superviseurs non-déterministes.*

Définition 2.30 *Un* superviseur *pour un SÉD $G = (L, S)$, à qui est associée une famille d'entrées de commande C, est une fonction partielle $V : \Sigma^* \longrightarrow C$. Le système en boucle fermée, résultant de l'action du superviseur V sur le SÉD G, est donné par V/G, où*

(i) $L(V/G)$, *le langage* synthétisé *par V, est défini par :*

(a) $1_\Sigma \in L(V/G)$.

(b) Pour tout $k \in \Sigma^*$, $\sigma \in \Sigma$

$$k\sigma \in L(V/G) \iff k \in L(V/G) \cap \mathrm{dom}(V) \text{ et } k\sigma \in L(G) \text{ et } \sigma \in V(k)$$

(ii) $S(V/G)$, *le ω-langage* synthétisé *par V, est donné par*

$$S(V/G) = \lim(L(V/G)) \cap S(G).$$

Définition 2.31 *On dit que V est un superviseur* non bloquant *pour G si V/G est*

un SÉD non bloquant.

Définition 2.32 *On dit que V est un superviseur* complet *pour G si et seulement si $L(V/G) \subseteq \text{dom}(V)$.*

2.4.3 Caractérisation des langages générés en boucle fermée

Voici une généralisation de la notion de commandabilité du chapitre précédent[GR87]:

Définition 2.33 *Soit G un SÉD dont l'alphabet des événements est Σ, et C une famille d'entrées de commande. On dit qu'un langage $K \subseteq \Sigma^\infty$, vérifiant* $\text{pre}(K) \subseteq L(G)$, *est* $*$-commandable *par rapport à $L(G)$ ssi*

$$\forall m \in pre(K), \exists \Gamma \in C : \Gamma \cap \Sigma_{L(G)}(m) = \Sigma_K(m)$$

Remarque 2.11 *Informellement, un langage K est $*$-commandable si pour tout préfixe m de K les extensions qui sont physiquement possibles peuvent être restreintes au moyen d'une entrée de commande à celles qui permettent de rester dans l'ensemble des préfixes de K.*

Concernant l'existence d'un superviseur qui permet de satisfaire une spécification sur le comportement fini en boucle fermée citons le résultat suivant:

Proposition 2.1 (Golaszewski-Ramadge-Wonham) *Soit G un SÉD et $K \subseteq L(G)$ non vide, il existe un superviseur complet et non-bloquant pour G ssi K est $*$-commandable par rapport à $L(G)$ et fermé.*

Pour caractériser maintenant l'existence d'un superviseur qui permet de satisfaire une spécification sur le comportement infini en boucle fermée, nous allons rappeler la notion de ω-commandabilité introduite par Thistle et Wonham [TW94b].

Définition 2.34 *Soit G un SÉD, on définit alors l'application suivante, qu'on appelle le* préfixe de commandabilité*:*

$$\mathrm{pre}_G : \ 2^{S(G)} \ \leftarrow \ 2^{\mathrm{pre}(S(G))}$$

$$K \ \mapsto \left\{ \begin{array}{c} k \in \mathrm{pre}(K) : (\exists K' \subseteq K/k) \\ K' \neq \emptyset \\ K' \text{ est } *\text{-commandable p. r. à } L(G)/k \\ K' \text{ et } \omega\text{-fermé p.r. } S(G)/k \end{array} \right\}$$

Remarque 2.12 *Informellement, le préfixe de commandabilité associe à chaque sous-langage K de $S(G)$ l'ensemble de tous les ses préfixes dont les extensions finies peuvent être contrôlées pour appartenir à K.*

Définition 2.35 *Soit G un SÉD et $K \subseteq S(G)$, on dit que K est* ω-commandable *par rapport à G ssi K est $*$-commandable par rapport à $L(G)$ et $\mathrm{pre}(K) = \mathrm{pre}_G(K)$.*

Remarque 2.13 *Il a été prouvé dans [TW94b] que dans le cas où C est fermée sous inclusion, c.-à.-d. $\Gamma \in C$ et $\Gamma \subseteq \Gamma' \Rightarrow \Gamma' \in C$, $\mathrm{pre}(K) = \mathrm{pre}_G(K) \Rightarrow K$ est $*$-commandable par rapport à $L(G)$*

On a aussi le résultat suivant:

Proposition 2.2 *Soit G un SÉD et soit K un sous-langage de $S(G)$ ω-fermé par rapport à $S(G)$. Alors on a*

$$K \text{ est } \omega\text{-commandable p.r. à } G \Leftrightarrow K \text{ est } *\text{-commandable p.r. à } L(G)$$

Pour caractériser les ω-langages qui peuvent être des comportements infinis en boucle fermée, nous avons le résultat suivant[TW94b]:

Proposition 2.3 *Soit G un SÉD et $K \subseteq S(G)$ non vide, il existe un superviseur complet et non-bloquant pour G tel que $S(V/G) = K$ ssi K est ω-commandable par rapport à G et ω-fermé par rapport à $S(G)$.*

2.5 Supervision des SÉDs sur mots infinis

2.5.1 Problème à résoudre

Lorsqu'on s'intéresse au comportement infini des SÉDs, on est confronté au problème suivant (qui n'est autre qu'une généralisation du problème original de synthèse de superviseur défini dans la section précédente).

Problème 2.1 (SCP$^\omega$) *Soit G un SÉD, et soit $A, E \subseteq S(G)$ deux spécifications sur le langage légal en boucle fermée. Construire un superviseur V pour G complet et non-bloquant tel que le comportement infini en boucle fermée vérifie la condition suivante:*

$$A \subseteq S(V/G) \subseteq E$$

Remarque 2.14 *Noter qu'on ne cherche pas à calculer la stratégie de commande la moins restrictive parce que, comme on va le voir plus tard, le superviseur le plus permissif n'existe pas dans tous les cas.*

D'après la proposition 2.3, une solution au problème existe, s'il existe un ω-langage K ω-commandable par rapport à G et ω-fermé par rapport à $S(G)$, et qui vérifie en plus $A \subseteq K \subseteq E$.

Pour donner les conditions d'existence d'une solution au problème, il faut définir les classes de ω-langages suivantes :

Définition 2.36 *Étant donné un SÉD G et deux ω-langages A et E vérifiant $A \subseteq E \subseteq S(G)$ on définit les classes suivantes:*

$$\mathcal{C}^{\omega}(E) := \{K \subseteq S(G) : K \subseteq E \subseteq S(G) \text{ et } K \text{ est } \omega\text{-commandable p.r. } G\}$$

$$\mathcal{F}^{\omega}(A) := \{K \subseteq S(G) : A \subseteq K \subseteq S(G) \text{ et } K \text{ est } \omega\text{-fermé p.r. } S(G)\}$$

Ces deux classes de langages ont des propriétés algébriques distinctes comme le montre la proposition suivante:

Proposition 2.4 *Pour un SÉD G et deux ω-langages A et E vérifiant $A \subseteq E \subseteq S(G)$, on a :*

(i) $\mathcal{C}^{\omega}(E)$ est fermée sous réunion arbitraires, et on a:

$$\sup \mathcal{C}^{\omega}(E) = \lim (\sup \mathcal{C}\mathcal{F}^{*} (\mathrm{pre}_{G}(E))) \cap E$$

(ii) $\mathcal{F}^\omega(A)$ *est fermée sous intersections arbitraires, et on a:*

$$\inf \mathcal{F}^\omega(A) = \mathrm{clo}(A) \cap S(G)$$

où $\mathcal{CF}^*$ *est la classe de langages définie, pour tout sous-langage* F *de* $L(G)$ *par:*

$$\mathcal{CF}^*(F) := \{K \subseteq F : K \text{ est } *\text{-commandable p.r. à } L(G) \text{ et fermé}\}$$

Remarque 2.15 *On sait que* $\mathcal{CF}^*(F)$ *est fermé sous réunion arbitraire d'après le chapitre précédent.*

L'existence d'une solution au SCP$^\omega$ est caractérisée par le théorème suivant[TW94b]:

Théorème 2.1 *SCP*$^\omega$ *admet une solution ssi* $\sup \mathcal{C}^\omega(E) \neq \emptyset$ *et*

$$\inf \mathcal{F}^\omega(A) \subseteq \sup \mathcal{C}^\omega(E)$$

Contrairement au cas des mots finis, le superviseur le plus permissif possible qui permet de répondre à des spécifications sur mots infinis n'existe pas pour n'importe quelles spécificaions. L'existence d'une telle solution optimale est caractérisée par le corollaire suivant:

Corollaire 2.1 *Supposons que SCP*$^\omega$ *admet une solution. Alors il admet une solution optimale ssi* $\sup \mathcal{C}^\omega(E) \in \mathcal{F}^\omega(A)$, *auquel cas* $\sup \mathcal{C}^\omega(E)$ *serait l'unique solution optimale.*

Pour garantir l'existence d'une solution optimale pour le problème, il faut se restreindre à un classe peu intéressante de spécifications:

Corollaire 2.2 *Si E est ω-fermé p.r. à $S(G)$, alors* $\sup C^{\omega}(E)$ *est aussi ω-fermé p.r. à $S(G)$.*

Remarque 2.16 *Le fait de considérer une spécification E qui soit ω-fermée par rapport à $S(G)$ limite l'intérêt de l'utilisation des ω-langages parce que de telles spécifications sont complètement determinées par la contrainte définie sur* $\mathrm{pre}(E)$.

2.5.2 Synthèse de superviseurs

Nous présentons dans cette section une approche développée par J. G. Thistle et W. M. Wonham dans [TW94b] pour la synthèse d'un superviseur pour un SÉD G qui fait en sorte que le comportement en boucle fermée soit compris entre les deux langages de spécifications A et E. Le point de départ de cette procédure est la représentation par des automates finis du SÉD G et des spécifications A et E, on suppose donc que :

- G est représenté par un automate de Büchi,
- E est représenté par un automate de Rabin, et
- pre(A) est représenté par un automate fini sur mots finis (comme on va le voir plus tard, tout ce qui nous intéresse de A est l'ensemble de ses préfixes).

Avant d'essayer de résoudre le SCP$^\omega$ il faut vérifier qu'il admet une solution. Ceci revient au calcul de $\inf \mathcal{F}^\omega(A)$ et de $\sup \mathcal{C}^\omega(E)$ (d'après le théoreme 2.1) et d'après la proposition 2.4 ceci nécessite le calcul de $\text{pre}_G(E)$.

2.5.2.1 Calcul de $\text{pre}_G(E)$

Nous allons supposer que $L(G)$, $S(G)$ et E sont représentés par la même structure de transition (c'est toujours possible de construire un tel automate "hybride")

Définition 2.37 *Un automate de* Rabin-Büchi *est un 6-tuple*

$$(\Sigma, X, \delta, x_0, \{(R_p, I_p) : p \in P\}, R)$$

Où Σ est un alphabet fini, Q est un ensemble fini d'états, $q_0 \in Q$ est l'état initial, $\delta : Q \times \Sigma \to Q$ est une fonction de transition (partielle), $\{(R_p, I_p) : p \in P\}$ est une condition d'acceptation de Rabin, et R est une condition d'acceptation de Büchi.

On suppose donc qu'on dispose d'un automate de Rabin-Büchi $(\Sigma, X, \delta, x_0, \{(R_p, I_p) : p \in P\}, R)$ tel que :

- l'automate de Rabin $(\Sigma, X, \delta, x_0, \{(R_p, I_p) : p \in P\})$ représente E,
- l'automate de Büchi $(\Sigma, X, \delta, x_0, R)$ reconnait $S(G)$,
- l'automate sur mots finis $(\Sigma, X, \delta, x_0, X)$ reconnait $L(G)$.

On considère le sous-ensemble d'états suivant:

$$C^{\mathcal{A}} := \left\{ x \in X : E_x \subseteq S_x, (\exists E'_x \subseteq E) \left[\begin{array}{c} E'_x \neq \emptyset \text{ est } *\text{-commandable p.r. à } L(G_x) \\ \text{et} \\ \omega\text{-fermé p.r. à } S(G_x) \end{array} \right] \right\}$$

où, pour tout $x \in X$: E_x est le langage reconnu par $(\Sigma, X, \delta, x, \{(R_p, I_p) : p \in P\})$,

$L(G_x)$ est le langage reconnu par $(\Sigma, X, \delta, x, X)$ et

$S(G_x)$ est le langage reconnu par $(\Sigma, X, \delta, x_0, R)$.

Remarque 2.17 *$C^{\mathcal{A}}$ représente l'ensemble des états de $\mathcal{A}$ à partir desquels on est capable de contrôler l'automate $\mathcal{A}$ de façon à ce qu'il satisfasse sa condition d'acceptation de Rabin et qu'en même temps les mots infinis générés satisfassent la condition d'acceptation de Büchi.*

Dans [TW94b] les auteurs ont prouvé que pour tout $k \in \text{pre}(E)$:

$$k \in \text{pre}_G(E) \Leftrightarrow \delta(k, x_0) \in C^{\mathcal{A}}$$

Ce qui veut dire que le problème de calcul de $pr_G(E)$ se ramène à celui du calcul de $C^{\mathcal{A}}$.

Concernant maintenant le calcul de $C^{\mathcal{A}}$, il faut recourir à la technique du calcul de point fixe. En effet, une démarche similaire à celle de [TW94a] permet de caractériser $C^{\mathcal{A}}$ sous forme d'un point fixe d'un certain opérateur sur l'ensemble de états X

(pour plus de détails se réferer à [TW91]).

2.5.2.2 Calcul de $\sup \mathcal{C}^{\omega}(E)$

D'après la proposition 2.4 pour calculer $\sup \mathcal{C}^{\omega}(E)$ il faut calculer $\sup \mathcal{CF}^{*}(\mathrm{pre}_G(E))$, puis calculer l'intersection de sa limite avec E. Pour le calcul de $\sup \mathcal{CF}^{*}(\mathrm{pre}_G(E))$, on peut utiliser la proposition suivante [TW94b]:

Proposition 2.5 *Soit $\mathcal{A} = (\Sigma, X, \delta, x_0, \{(R_p, I_p) : p \in P\}, R)$ l'automate de Rabin-Büchi qui représente $L(G)$, $S(G)$ et E. Alors l'automate sur mots finis :*

$$\left(\Sigma, X, \delta^{\mathcal{A}}, x_0, C^{\mathcal{A}}\right)$$

reconnaît $\sup \mathcal{CF}^{*}(\mathrm{pre}_G(E))$. *Où*

$\delta^{\mathcal{A}}$ est définie comme suit:

$$\delta^{\mathcal{A}} : \Sigma \times X \to X$$

$$(\sigma, x) \mapsto \begin{cases} \delta(\sigma, x) & \text{si } x \in C^{\mathcal{A}} \text{ et } \sigma \in \Gamma^{\mathcal{A}}(x), \\ \text{non définie} & \text{sinon} \end{cases}$$

Avec $\Gamma^{\mathcal{A}}$ est une fonction définie par:

$$\Gamma^{\mathcal{A}} : C^{\mathcal{A}} \to C$$

$$x \mapsto \cup\left\{\Gamma \in C : (\forall \sigma \in \Gamma)\left[\delta(\sigma, x)! \Rightarrow \delta(\sigma, x) \in C^{\mathcal{A}}\right]\right\}$$

Théorème 2.2 *Soit* $\mathcal{A} = (\Sigma, X, \delta, x_0, \{(R_p, I_p) : p \in P\}, R)$ *l'automate de Rabin-Büchi qui représente* $L(G)$, $S(G)$ *et* E. *Alors l'automate, déterministe, de Rabin suivant :*

$$\left(\Sigma, X, \delta^{\mathcal{A}}, x_0, \{(R_p, I_p) : p \in P\}\right)$$

reconnaît $\sup \mathcal{C}^{\omega}(E)$

2.5.2.3 Test de solvabilité du SCP$^{\omega}$

Nous sommes capable à ce stade de tester si l'inclusion:

$$\inf \mathcal{F}^{\omega}(A) \subseteq \sup \mathcal{C}^{\omega}(E)$$

est vraie ou non.

Supposons maintenant qu'on dispose:

- d'un automate fini sur mots finis $\mathcal{A}_{\inf} = (\Sigma, X, \delta, x_0, F)$ qui reconnaît pre(A) (il est toujours possible de construire un tel automate à partir d'un ω-automate qui reconnaît A),
- d'un automate $\mathcal{A}_S = (\Sigma, X', \delta', x'_0, R')$ (déterministe) de Büchi qui reconnaît $S(G)$, et
- d'un autommate de Rabin déterministe $\mathcal{A}_{\sup} = (\Sigma, X'', \delta'', x''_0, \{(R_p, I_p) : p \in P\})$ qui reconnaît $\sup \mathcal{C}^{\omega}(E)$.

A partir de ces automates, il est possible de construire un automate de Streett qui reconnait le langage $\inf \mathcal{F}^{\omega}(A) \setminus \sup \mathcal{C}^{\omega}(E)$:

$$\mathcal{A}_{diff} = \left(\Sigma, X''', \delta''', x_0''', \{(\emptyset, X \times R \times X''), (\emptyset, F \times X' \times X'')\} \bigcup \left\{\left(I_p''', R_p'''\right) : p \in P\right\}\right)$$

Où

$$X''' = X \times X' \times X''$$

$$\delta''' = \delta \times \delta' \times \delta''$$

$$x_0''' = (x_0, x_0', x_0'')$$

$$I_p''' = X \times X' \times R_p$$

$$R_p''' = X \times X' \times I_p.$$

Une fois cet automate construit, il suffit de voir s'il reconnaît le langage vide [?] pour vérifier la validité de l'inclusion.

2.5.2.4 Synthèse d'une solution

Une fois qu'on s'est assuré de l'existence d'une solution, on peut utiliser la méthode suivante, qui s'inspire de la démonstration du théorème 2.1 (voir [TW94b]), pour le calcul de cette solution:

- étape 1 : À partir de l'automate $\mathcal{A}_{sup}$, on construit l'automate $(\Sigma, X'', \delta'', x_0'', F'')$ qui reconnaît $\mathrm{pre}(\sup \mathcal{C}^{\omega}(E))$.

• étape 2 : On définit ensuite la fonction suivante:

$$\begin{aligned} \phi_0 : \quad F'' &\rightarrow C \\ x' &\mapsto \{\sigma \in \Sigma : \delta''(\sigma, x') \in F''\} \end{aligned}$$

• étape 3 : On construit ensuite la fonction $\phi : X'' \rightarrow C$ qui vérifie la condition suivante : Pour tout $k \in \text{pre}(\sup \mathcal{C}^{\omega}(E))$, le superviseur

$$\begin{aligned} f_k : \quad \Sigma^* &\rightarrow C \\ l &\mapsto \phi(\delta''(k, x_0)) \end{aligned}$$

est un superiseur complet non-bloquant pour $(L(G)/k, S(G)/k)$ qui synthétise un sous-ensemble de E/k.

• étape 4 : Il est possible de prouver que le superviseur suivant :

$$f : \Sigma^* \rightarrow C$$

$$l \mapsto \begin{cases} f_0(l) & \text{si } l \in \text{pre}(\mathcal{A}) \\ f_k(l/k) & \text{si } k \text{ est l'élément minimal de pre}(l) \setminus \text{pre}(\mathcal{A}) \end{cases}$$

Où

$$\begin{aligned} f_0 : \quad \Sigma^* &\rightarrow C \\ k &\mapsto \phi_0(\delta''(k, x_0)) \end{aligned}$$

constitue une solution au SCP$^{\omega}$.

2.6 Conclusion

Dans ce chapitre, nous avons fait un survol d'une théorie qui s'intérèsse au comportement infini des SÉDs. Nous avons surtout présenté des éléments de solution qui permettent de résoudre un problème de supervision typique lorsqu'on étudie le comportement asymptotique d'un SÉD. Même si nous nous sommes restreints au cas où le procédé à contrôler est modélisé par un automate de Büchi, il faut signaler que des cas plus généraux ont été étudiés [?,?]. Nous n'avons pas mentionné ces cas parce que, dans ce qui suit, nous allons considérer uniquement des procédés qui sont représentés par des automates de Büchi ayant des conditions d'acceptation dégénérées. Plus précisément une grande partie de notre travail consiste à étendre les limites du cadre formel que nous venons de voir pour englober le cas des SÉD partiellement observés. +

CHAPITRE 3

SUPERVISION DES SÉQUENCES INFINIES D'ÉVÉNEMENTS GÉNÉRÉES PAR LES SÉDS PARTIELLEMENT OBSERVÉS

3.1 Introduction

Dans ce chapitre nous allons nous intéresser à la commande des mots infinis dans le cas où le superviseur est incapable d'observer tout ce que le procédé génère. Dans une telle situation, un masque d'observation est placé entre le superviseur et le procédé. Ce mécanisme d'observation reçoit, en entrée, ce que le procédé génère et fournit, en sortie, ce que le superviseur observe de ce qui a été généré. Le fonctionnement du système en boucle fermée est le suivant : le procédé évolue dans le temps en générant une séquence (infinie) d'événements; à chaque instant le superviseur limite l'ensemble des actions que le système peut effectuer à cette étape, en se basant sur ce qu'il observe du comportement passé du procédé. On peut schématiser ceci avec la figure 3.1.

Les résultats que nous allons donner dans ce chapitre, ainsi que dans le chapitre suivant, constituent une généralisation de ce qui a été fait conçernant la supervision des séquences infinies puisque nous allons généraliser des résultats qui ne s'appliquaient qu'au cas des observations complètes à un cas d'observations partielles. Ces résultats constituent aussi une généralisation de ce qui a été déjà fait

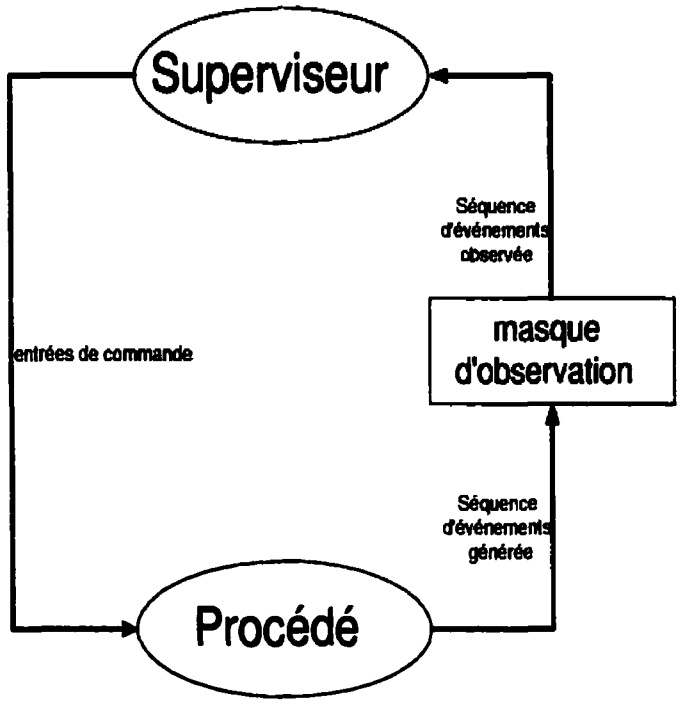


Figure 3.1: Système en boucle fermée dans un contexte d'observations partielles

concernant la supervision du comportement fini des SÉDs dans le sens où nous allons utiliser un mécanisme d'observation ayant une structure qui présente les deux particularités suivantes.

D'abord on ne partitionne pas l'alphabet Σ des événements en deux sous-ensembles, événements observables et événements non observables. On définit plutôt un nouvel alphabet Σ_o qui correspond à ce que le superviseur observe. Ce nouvel alphabet peut contenir des événements qui n'appartiennent pas à l'alphabet original, ils correspondent à des événements réels qui sont partiellement observables; c.-à.-d. que le superviseur peut détecter l'occurrence de ces événements, mais il est incapable de les observer complètement. Le fait d'avoir enrichi l'alphabet des événements réels

avec ces symboles "artificiels" permet aussi de modéliser le cas où deux symboles donnent lieu à une même observation. D'autre part nous doterons ce mécanisme d'observation d'une "mémoire" qui fera en sorte que l'observabilité d'un événement dépendra de ce qui a été généré auparavant.

Comme on peut le deviner, étant donné un SÉD G on ne peut pas construire pour n'importe quel ω-langage K un superviseur à observations partielles pour G tel que le ω-langage synthétisé en boucle fermée soit égal à K. Pour caractériser l'ensemble des ω-langages qui correspondent à un langage généré en boucle fermée, nous allons généraliser des notions tels que la commandabilité et la normalité pour le cas des ω-langages.

La supervision sous observations partielles présente des similarités avec la supervision hiérarchique dans le sens où le mécanisme d'observation peut être considéré comme un opérateur qui nous fait passer d'un modèle d'implémentation (le système réel) à un modèle abstrait (le système tel que observé par le superviseur), c'est pour cette raison que tout au long de ce mémoire nous allons parler de modèle de "haut niveau" et de modèle de "bas-niveau".

Nous commençons à la section 3.2 par présenter le mécanisme d'observation. Nous donnons aussi dans cette section la définition d'un superviseur sous observations partielles et nous caractérisons les comportements fini et infini générés en boucle fermée. La section 3.3 est dédiée à une généralisation aux ω-langages de quelques propriétés fondamentales. Dans l'avant-dernière section nous posons le problème

de synthèse de superviseurs pour le comportement infini des SÉDs partiellement observés et nous donnons un résultat qui caractérise l'existence d'une solution au problème. Nous clôturons le chapitre avec une conclusion dans laquelle nous expliquons brièvement notre approche de résolution.

3.2 SÉDs partiellement observés

Comme d'habitude on désigne l'alphabet des événements par Σ, et on suppose qu'on définit sur Σ une famille non vide d'entrées de commande $C \subseteq 2^{\Sigma}$ qui vérifie les deux conditions suivantes :

- C est fermée par rapport à la réunion:

$$\Gamma_i, \Gamma_j \in C \Longrightarrow \Gamma_i \cup \Gamma_j \in C$$

- $\forall \sigma \in \Sigma, \exists \Gamma \in C : \sigma \in \Gamma$.

3.2.1 Masque d'observation

Dans des cas réels, il arrive que le procédé génère un événement que le superviseur ne peut pas observer. Il arrive aussi que deux événements différents donnent lieu à une même observation. On modélise une telle situation en introduisant un nouvel alphabet Σ_o qui correspond aux événements que le superviseur peut observer et en définissant un masque d'observation à travers lequel le superviseur observe le procédé.

Définition 3.1 *Soit G un SÉD défini sur l'alphabet Σ, et soit Σ_o l'alphabet des observations. Un* masque d'observation *pour G est une fonction M ayant la structure suivante:*

$$\begin{aligned} M: \quad L(G) &\longrightarrow \Sigma_o^* \\ 1_\Sigma &\longmapsto 1_{\Sigma_o} \\ \sigma_1\sigma_2...\sigma_n &\longmapsto M(\sigma_1\sigma_2...\sigma_{n-1})\gamma \end{aligned}$$

ou

$$\sigma_1\sigma_2...\sigma_n \longmapsto M(\sigma_1\sigma_2...\sigma_{n-1})$$

avec

$n \geq 1$,

et

$\gamma \in \Sigma_o$ *est l'*observation *de l'événement σ_n lorsque celui-ci est généré à la suite de la séquence $\sigma_1\sigma_2...\sigma_{n-1}$.*

Si $\gamma = \sigma_n$, on dit que σ_n est complètement observable *après le mot $\sigma_1\sigma_2...\sigma_{n-1}$. Sinon ($\gamma \neq \sigma_n$) on dit que σ_n est* partiellement observable *après la génération de $\sigma_1\sigma_2...\sigma_{n-1}$.*

Lorsque c'est la dernière clause qui s'applique, on dit que σ_n est inobservable *après le mot $\sigma_1\sigma_2...\sigma_{n-1}$.*

Notons que le masque d'observation tel que défini est un mécanisme à mémoire dans le sens où l'observation d'un événement dépend de ce qui a été généré dans le passé.

Remarque 3.1 *On étend M en une fonction sur l'ensemble des sous-langages de $L(G)$ de la façon suivante :*

$$M(K) := \{M(s) : s \in K\}, \forall K \subseteq L(G).$$

On étend aussi M en une fonction sur l'ensemble des ω-languages qui sont inclus dans $S(G)$ comme suit:

$$M(K) := \lim (M(\text{pre}(K))), \forall K \subseteq S(G).$$

Notation 3.1 *Si K est un singleton $\{s\}$, on écrit $M(s)$.*

Notre mécanisme d'observation, tel que défini, présente deux propriétés intéressantes à savoir le fait qu'il commute avec le préfixe ainsi qu'avec la limite des languages :

Proposition 3.1 *Soit G un SÉD défini sur l'alphabet Σ et M son masque d'obervation. Alors on a pour tout $K \subseteq L(G)$, $M(\text{pre}(K)) = \text{pre}(M(K))$.*

Démonstration: *Soit $k \in M(\text{pre}(K))$. Alors il existe $k' \in \text{pre}(K)$ tel que $M(k') = k$.*

$$
\begin{aligned}
k' \in \mathrm{pre}(K) &\Rightarrow \exists m' \in \Sigma^* \textit{ tel que } k'm' \in K \\
&\Rightarrow M(k'm') \in M(K) \\
&\Rightarrow M(k')m \in M(K) \\
&\Rightarrow km \in M(K) \\
&\Rightarrow k \in \mathrm{pre}(M(K))
\end{aligned}
$$

$\Longrightarrow M(\mathrm{pre}(K)) \subseteq \mathrm{pre}(M(K))$

D'autre part, soit $l \in \mathrm{pre}(M(K))$, *alors on a*

$$
\begin{aligned}
l \in \mathrm{pre}(M(K)) &\Rightarrow \exists l' \in \Sigma^* \textit{ tel que } ll' \in M(K) \\
&\Rightarrow \exists p \in K \textit{ tel que } M(p) = ll' \\
&\Rightarrow \exists p_1 \leq p \textit{ tel que } M(p_1) = p \\
&\Rightarrow l \in M(\mathrm{pre}(K))
\end{aligned}
$$

$\Longrightarrow \mathrm{pre}(M(K)) \subseteq M(\mathrm{pre}(K))$

$\Longrightarrow \mathrm{pre}(M(K)) = M(\mathrm{pre}(K))$.

◇

Proposition 3.2 *Soit* G *un SÉD défini sur l'alphabet* Σ *et* M *son masque d'obervation. Alors on a, pour tout* $K \subseteq L(G)$, $\lim(M(K)) = M(\lim(K))$.

Démonstration: *Commençons par prouver que* $M(\lim(K)) \subseteq \lim(M(K))$.

En effet nous avons :

$$\begin{aligned} M(\lim(K)) &= \lim(M(\mathrm{pre}(\lim(K)))) \\ &\subseteq \lim(M(K)) \quad \textit{(puisque } \mathrm{pre}(\lim(K)) \subseteq K \textit{ par définition même de } \lim\textit{)} \end{aligned}$$

Pour montrer l'autre inclusion, soit $s \in \lim(M(K))$. *Alors on a* $\mathrm{pre}(s) \subseteq M(K)$. *Par conséquent il existe* $K' \subseteq K$, *non vide, tel que* $M(K') = \mathrm{pre}(s)$.
Soit $t \in \lim(K') \subseteq \lim(K)$ *(un tel mot existe parce que* $\lim(K') \neq \emptyset$ *d'après le lemme de König). Alors, on a :*

$$\begin{aligned} M(t) &= \lim(M(\mathrm{pre}(t))) \\ &\in \lim(M(K')) \\ &\in \lim(\mathrm{pre}(s)) \\ &\in \mathrm{clo}(s) \\ &= \{s\} \end{aligned}$$

Par conséquent, $M(t) = s \Longrightarrow s \in M(\lim(K))$.

◇

Le masque M est représenté par un automate (fini) de Moore:

$$\mathcal{A}_M = (\Sigma, X_M, \delta_M, x_{0_M}, \Sigma_o, \lambda)$$

où Σ est l'alphabet de transition,

X_M est l'ensemble des états,

$\delta_M : \Sigma \times X_M \longrightarrow X_M$ est la fonction de transition,

x_{0_M} est l'état initial,

Σ_o est l'alphabet de sortie et

$\lambda : X_M \longrightarrow \Sigma_o$ la fonction de sortie.

L'automate $\mathcal{A}_M$ représente le masque M comme suit :

$$\begin{aligned} M(m) &= M(\sigma_1 ... \sigma_n) \\ &= \lambda(x_1) ... \lambda(x_n) \end{aligned}$$

où $x_1 = \delta_M(\sigma_1, x_{0_M})$ et $x_i = \delta_M(\sigma_i, x_{i-1})$ pour $2 \leq i \leq n$.

Donc, pour déterminer l'observation d'un mot fini m, il suffit de trouver le chemin dans l'automate $\mathcal{A}_M$ qui respecte la structure du mot m et de concaténer les symboles de sortie associés aux états de ce chemin.

Remarque 3.2 *Puisque le masque d'observation qu'on considère est un masque à mémoire, il faut que l'automate de Moore qui le représente vérifie la condition suivante:*

$$M(m_1) \neq M(m_2) \Rightarrow \delta_M(m_1, x_{0_M}) \neq \delta_M(m_2, x_{0_M}).$$

3.2.2 Supervision sous observations partielles

Pour tenir compte du fait qu'on se place dans un contexte d'observations incomplètes, nous allons modifier la définition de superviseur du chapitre précédent:

Définition 3.2 *Soit G un SÉD partiellement observé défini sur l'alphabet Σ, pour lequel est associé un masque d'observation M. Soit C une famille d'entrées de commande définie sur l'alphabet Σ. Un* superviseur sous observations partielles *pour (G, M) est une fonction partielle $V : \Sigma_o^* \longrightarrow C$.*

Le système en boucle fermée obtenu suite à l'action de V sur G, noté V/G, est donné par :

(i) $L(V/G)$, *le* language synthétisé *par V, est défini récursivement comme suit :*

(a) $1_\Sigma \in L(V/G)$.

(b) Pour tout $k \in (\Sigma)^*$, $\sigma \in \Sigma$

$$k\sigma \in L(V/G) \iff k \in L(V/G) \text{ et } M(k) \in dom(V) \text{ et } k\sigma \in L(G) \text{ et } \sigma \in V(M(k))$$

(ii) $S(V/G)$, *le* ω-language synthétisé *par V, est donné par*

$$S(V/G) = \lim(L(V/G)) \cap S(G)$$

Définition 3.3 *Soit G un SÉD partiellement observé défini sur l'alphabet Σ auquel est associé un masque M. On dit qu'un superviseur V pour G est* complet *si*

$$M(L(V/G)) \subseteq dom(V)$$

On dit que V est non-bloquant *si V/G est non bloquant.*

3.3 Propriétés des ω-langages

Dans cette section nous allons donner des propriétés de ω-langages qui vont permettre de caractériser la classe des ω-langages qui correspondent à un comportement en boucle fermée dans un contexte d'observations partielles.

La première notion que nous allons généraliser est celle de normalité :

Définition 3.4 *Étant donné un SÉD G pour lequel est défini un masque d'observation M, on dit qu'un ω-langage $K \subseteq S(G)$ est* normal *par rapport à M, si* $\mathrm{pre}(K)$ *vérifie:*

$$M^{-1}(M(\mathrm{pre}(K))) = \mathrm{pre}(K)$$

Définition 3.5 *Soit G un SÉD défini sur l'alphabet Σ partiellement observé à travers le masque M et soit C une famille d'entrées de commande associée à G. On dit qu'un ω-langage $K \subseteq S(G)$ est* commandable par retour de sortie *par rapport à (G, M) si :*

$$\forall m \in \text{pre}(K), \exists \Gamma \in C : \forall m' \in M^{-1}(M(m)) \cap \text{pre}(K)$$

$$\Gamma \cap \Sigma_{L(G)}(m') = \Sigma_K(m')$$

Intuitivement, pour qu'un ω-langage soit généré en boucle fermée sous la supervision d'un superviseur à observations partielles il faut, qu'à chaque étape, l'entrée de commande que le superviseur génère en se basant sur l'observation d'une séquence finie k soit compatible avec toute autre séquence qui donne lieu à la même observation; cette propriété n'est autre que la commandabilité par retour de sortie qui généralise à la fois la notion de commandabilité ainsi que celle d'observabilité du chapitre 1.

3.4 Problème de synthèse de superviseurs sous observations partielles

3.4.1 Problème à résoudre

Dans un contexte d'observations partielles, on s'intéresse à la résolution du problème suivant :

Problème 3.1 *Étant donné un SÉD G défini sur l'alphabet Σ (qui modélise le procédé à contrôler) et un $\omega-$langage $E \subseteq \Sigma^\omega$ tel que $E \subseteq S(G)$ (qui représente la spécification sur le ω-langage généré en boucle fermée), construire un superviseur*

sous observations partielles V complet et non-bloquant tel que :

$$S(V/G) \subseteq E$$

Remarque 3.3 *Notons encore une fois qu'on ne s'intéresse pas à la solution optimale, car cette solution n'existe pas pour n'importe quel système et n'importe quelle spécification.*

3.4.2 Existence d'une solution

Étant donné un ω-langage $K \subseteq S(G)$, il n'est pas certain que K peut correspondre à un fonctionnement en boucle fermée. Pour pouvoir décider de l'existence d'un supervieur sous observations partielles qui fait en sorte que les séquences infinies générées sous la supervision de V soit exactement celles contenues dans K, on utilise le résultat suivant:

Proposition 3.3 *Pour tout $K \subseteq S(G)$, il existe un superviseur sous observations partielles complet et non-bloquant pour G qui synthétise K si et seulement si K est commandable par retour de sortie par rapport à (G, M) et ω-fermé par rapport à $S(G)$.*

Démonstration:

$\Longrightarrow$:

Soit $V:\Sigma_0^* \longrightarrow C$ *un superviseur complet et non-bloquant pour* G *tel que* $S(V/G) = K$

Montrons d'abord que K *est* ω*-fermé par rapport à* $S(G)$*:*

$$\begin{aligned}
\mathrm{clo}(K) \cap S(G) &= \lim(\mathrm{pre}(K)) \cap S(G) \\
&= \lim(\mathrm{pre}(S(V/G))) \cap S(G) \\
&= \lim(L(V/G)) \cap S(G) \ \textit{(}V \textit{ est non-bloquant)} \\
&= S(V/G) \\
&= K
\end{aligned}$$

Montrons maintenant que K *est commandable par retour de sortie par rapport à* (G, M)*:*

Soit $m \in \mathrm{pre}(K) = L(V/G)$*, comme* V *est complet,* $V(M(m))$ *existe. Posons* $\Gamma_i = V(M(m))$ *et montrons que* $\forall m' \in M^{-1}(M(m)) \cap \mathrm{pre}(K)$*, on a:*

$$\Gamma_i \cap \Sigma_{L(G)}(m') = \Sigma_K(m')$$

Soit $m' \in M^{-1}(M(m)) \cap \mathrm{pre}(K)$ *et soit* $\sigma \in \Gamma_i$ *tel que* $m'\sigma \in L(G)$*. Alors on a:*

$$\begin{aligned}
& m' \in \mathrm{pre}(K) \textit{ et } \sigma \in \Gamma_i \textit{ et } m'\sigma \in L(G) \\
\Longrightarrow\ & m' \in L(V/G) \textit{ et } M(m') \in \mathit{dom}(V) \textit{ et } \sigma \in V(M(m)) \textit{ et } m'\sigma \in L(G).
\end{aligned}$$

(Puisque $M(m') = M(m)$*, ce qui implique* $V(M(m')) = V(M(m))$*)*

$\Rightarrow m'\sigma \in L(V/G) = \text{pre}(K)$

$\Rightarrow \Gamma_i \cap \Sigma_{L(G)}(m') \subseteq \Sigma_K(m')$

Montrons l'autre inclusion: soit $\sigma \in \Sigma_K(m')$ *alors on a*

$$
\begin{aligned}
m'\sigma \in \text{pre}(K) \;&\Rightarrow\; m'\sigma \in L(V/G) \\
&\Rightarrow\; m' \in L(V/G) \text{ et } M(m') \in \text{dom}(V) \\
&\qquad \text{et } \sigma \in V(M(m')) = \Gamma_i \text{ et } m'\sigma \in L(G) \\
&\Rightarrow\; \sigma \in \Gamma_i \cap \Sigma_{L(G)}(m') \\
&\Rightarrow\; \Sigma_K(m') \subseteq \Gamma_i \cap \Sigma_{L(G)}(m') \\
&\Rightarrow\; \Gamma_i \cap \Sigma_{L(G)}(m') = \Sigma_K(m')
\end{aligned}
$$

$\Longrightarrow$ *K est commandable par retour de sortie par rapport à (G, M).*

$\Longleftarrow$ *:*

K est commandable par retour de sortie par rapport à $(G, M) \Longrightarrow \forall m \in$ $\text{pre}(K), \exists \Gamma_j \in C : \forall m' \in M^{-1}(M(m))$*, on a*

$$\Gamma_j \cap \Sigma_{L(G)}(m') = \Sigma_K(m')$$

On définit la fonction $\phi : M(\text{pre}(K)) \longrightarrow C$ *telle que* $\forall m \in \text{pre}(K)$ *et*

$\forall m' \in M^{-1}(M(m))$, *on a*

$$\Sigma_L(G)(m') \cap \phi(M(m)) = \Sigma_K(m')$$

et on considère le superviseur suivant

$$V : \Sigma_o^* \longrightarrow C$$

$$k \longmapsto \begin{cases} \phi(k) & \text{si } k \in M(\text{pre}(K)) \\ \text{non défini} & \text{sinon} \end{cases}$$

notre but est de montrer que $L(V/G) = \text{pre}(K)$, *ce qui revient à montrer que* $\forall l \in \Sigma^*$:

$$l \in L(V/G) \Leftrightarrow l \in \text{pre}(K)$$

Par récurrence sur la longueur du mot l: *Pour* $l = 1_\Sigma$ *l'équivalence est vérifiéee.*

Supposons qu'elle est vraie pour tout mot l *de longueur* $\leq n-1$ *et montrons que ça reste vrai pour tout mot* l *de longueur* n:

$l = k\sigma \in L(V/G)$

$\Rightarrow k \in L(V/G)$ *et* $k\sigma \in L(G)$ *et* $M(k) \in \text{dom}(V)$ *et* $\sigma \in V(M(k))$

$\Rightarrow k \in \text{pre}(K)$ *et* $k\sigma \in L(G)$ *et* $\sigma \in \phi(M(k))$ *(d'après l'hypothèse de récurrence.)*

$\Rightarrow \sigma \in \Sigma_{L(G)}(k) \cap \phi(M(k))$

$\Rightarrow \sigma \in \Sigma_{\mathrm{pre}(K)}(k)$

$\Rightarrow k\sigma \in \mathrm{pre}(K)$ *(d'après la commandabilité par retour de sortie.)*

Montrons l'autre implication :

$l = k\sigma \in \mathrm{pre}(K) \Rightarrow k \in \mathrm{pre}(K)$ *et* $\sigma \in \Sigma_K(k)$

$k \in \mathrm{pre}(K) \Rightarrow \exists \Gamma_j : \forall k' \in M^{-1}(M(k)) \cap \mathrm{pre}(K)$ *on a*

$$\Gamma_j \cap \Sigma_{L(G)}(k') = \Sigma_K(k')$$

$\sigma \in \Sigma_K(k) \Rightarrow \sigma \in \Gamma_j$ *et* $\sigma \in \Sigma_{L(G)}(k)$

On a donc $k \in L(V/G)$ *et* $k\sigma \in L(G)$ *et* $M(k) \in \mathrm{dom}(V)$ *et* $\sigma \in V(M(k))$.

Ceci implique $l \in L(V/G)$.

D'autre part on a

$$\begin{aligned} S(V/G) &= \lim(L(V/G)) \cap S(G) \\ &= \lim(\mathrm{pre}(K)) \cap S(G) \\ &= \mathrm{clo}(K) \cap S(G) \\ &= K. \end{aligned}$$

Montrons que V est complet:

$M(L(V/G)) = M(\mathrm{pre}(K)) = \mathrm{dom}(V)$.

Montrons que V est non-bloquant:

$L(V/G) = \mathrm{pre}(K) = \mathrm{pre}(S(V/G))$.

◇

3.5 Conclusion

Dans ce chapitre nous nous sommes placés dans un contexte d'observations partielles et nous avons redéfini la notion de superviseur pour tenir compte du manque d'information qui se produit dans un tel contexte. Dans le prochain chapitre nous allons introduire une nouvelle classe de SÉD appelée SÉD contrôlés dont la supervision est équivalente à celle des SÉDs. L'intérêt de cette opération est de ramener notre problème initial à un problème plus façile à résoudre.

CHAPITRE 4

SUPERVISION DES SÉDS CONTRÔLÉS

4.1 Introduction

Dans le chapitre précédent nous avons introduit le problème qu'on se propose de résoudre, à savoir la synthèse d'un superviseur sous observations partielles pour un SÉD G tel que le ω-langage généré en boucle fermée soit contenu dans un ω-langage de spécification. Notre approche de résolution consiste essentiellement en une série d'abstractions pour ramener le problème initial à un problème pour lequel des outils de résolution existent. Dans ce chapitre nous allons décrire une première étape dans le processus de résolution qui consiste à ramener le problème de supervision d'un SÉD G partiellement observé à celui de la supervision d'un nouveau SÉD défini à partir de G et que nous appellerons *SÉD contrôlé* associé à G. En effet, comme nous nous plaçons dans un contexte d'observations partielles, nous cherchons à exploiter toute information que le superviseur est capable de détecter. En particulier, nous supposons qu'à tout instant le superviseur est capable de mémoriser toutes les entrées de commande qu'il a générées jusque là. C'est pour avoir un mécanisme de supervision capable de mémoriser une telle information que nous allons enrichir l'alphabet des événements avec les entrées de commande qui ont été appliquées. On obtient alors le *SÉD contrôlé* G_C qui a la même dynamique que G mais qui est défini

sur l'alphabet enrichi. Nous allons d'abord associer à G_C un masque d'observation M_C qui sera l'équivalent de M pour G. Néanmoins pour simplifier les constructions que nous allons donner dans le chapitre suivant nous allons modifier la structure de M_C pour obtenir un nouveau masque $\widetilde{M}_C$. Nous allons montrer dans un premier temps que la supervision de (G, M) est équivalente à celle de (G_C, M_C), dans le sens où si on est capable de trouver un superviseur pour une paire, on est capable d'en trouver un pour l'autre paire.

Dans la section 4.2 nous décrivons la structure d'un SÉD contrôlé et le mécanisme d'observation M_C qui lui est associé. Nous allons valider le passage de (G, M) à (G_C, M_C) en montrant qu'à partir d'un superviseur pour (G, M) il est toujours possible de trouver un superviseur pour (G_C, M_C) qui génère le même langage en boucle fermé (à une projection près) et vice-versa. À la section 4.3 nous introduisons le nouveau masque $\widetilde{M}_C$ et nous validons le passage de (G_C, M_C) à $(G_C, \widetilde{M}_C)$ de la même façon que dans la section 4.2.

4.2 Les SÉDs contrôlés

4.2.1 SÉDs contrôlés partiellement observés

Comme d'habitude Σ dénote l'alphabet des événements, et $C \subseteq 2^{\Sigma}$ une famille non vide d'entrées de commande définie sur Σ.

Définition 4.1 *Soit G un générateur représenté par un automate de Büchi*

$$G = (\Sigma, X, \delta, x_0, T)\,.$$

Le SÉD contrôlé *associé à G est défini par :*

$$G_C = (\Sigma \times C, X, \delta_C, x_0, T)$$

où

δ_C *:*$(\Sigma \times C) \times X \longrightarrow X$ *est définie comme suit:*

$$\delta_C((\sigma, \Gamma_i), x) = \begin{cases} \delta(\sigma, x) & \text{si } \delta(\sigma, x) \text{ est définie et } \sigma \in \Gamma_i \\ \text{indéfinie} & \text{autrement} \end{cases}$$

Remarque 4.1 *À ce nouvel alphabet $\Sigma \times C$, on associe la famille suivante d'entrées de commande :*

$$C_C = \{\Gamma_{C_i}, 1 \leq i \leq n\}$$

avec

$$\Gamma_{C_i} = \{(\sigma, \Gamma_i) : \sigma \in \Gamma_i\}$$

et $\Gamma_i \in C$ (on dit que Γ_{C_i} est définie *à partir de Γ_i).*

Remarque 4.2 *Comme nous allons le montrer plus tard, concevoir un superviseur V pour G revient à synthétiser un superviseur V_C pour G_C. C'est pour cette raison*

qu'une étape importante vers la résolution du problème de synthèse de superviseur sous observations partielles pour un SÉD G sera de construire le SÉD contrôlé G_C qui lui est associé et de calculer un superviseur sous observations partielles pour G_C.

Comme on va chercher à superviser le SÉD contrôlé G_C, on lui associe aussi un mécanisme d'observation, l'alphabet des observations étant $\Sigma_o \times C$.

Définition 4.2 *Soit G un SÉD défini sur l'alphabet Σ pour lequel est associé l'alphabet des observations Σ_o et soit G_C le SÉD contrôlé associé à G. Un* masque d'observation *pour G_C est une application définie comme suit:*

$$
\begin{aligned}
M_C : \qquad L(G_C) &\longrightarrow (\Sigma_o \times C)^* \\
1_{\Sigma \times C} &\longmapsto 1_{\Sigma_o \times C} \\
(\sigma_1, \Gamma_1)...(\sigma_n, \Gamma_n) &\longmapsto M_C((\sigma_1, \Gamma_1)...(\sigma_{n-1}, \Gamma_{n-1}))(a, \Gamma_n) \\
&\text{ou} \\
(\sigma_1, \Gamma_1)...(\sigma_n, \Gamma_n) &\longmapsto M_C((\sigma_1, \Gamma_1)...(\sigma_{n-1}, \Gamma_{n-1}))
\end{aligned}
$$

Où $a \in \Sigma_o$ est le dernier symbole de $M(\sigma_1\sigma_2...\sigma_n)$ si σ_n est observable après la génération de $\sigma_1 \ldots \sigma_{n-1}$. La dernière clause s'applique lorsqu'on est incapable de détecter l'occurence de l'événement (σ_n, Γ_n) (on dit que (σ_n, Γ_n) est inobservable *après la génération de la séquence $(\sigma_1, \Gamma_1)...(\sigma_{n-1}, \Gamma_{n-1})$).*

Remarque 4.3 *On étend M_C à l'ensemble des sous-langages de $L(G_C)$ et à l'ensemble des ω-langages inclus dans $S(G_C)$ de le même façon qu'on l'a fait pour M au chapitre précédent.*

Remarque 4.4 *M_C présente les mêmes propriétés que M, à savoir il commute avec le préfixe ainsi qu'avec la limite des langages.*

Le masque M_C est aussi représenté par un automate de Moore:

$$\mathcal{A}_M = (\Sigma \times C, X_M, \delta_M, x_{0_M}, \Sigma_o, \lambda)$$

où $\Sigma \times C$ est l'alphabet de transition,

X_M est l'ensemble des états,

$\delta_M : (\Sigma \times C) \times X_M \longrightarrow X_M$ est la fonction de transition,

x_{0_M} est l'état initial,

Σ_o est l'alphabet de sortie et

$\lambda : X_M \longrightarrow \Sigma_o$ la fonction de sortie.

L'automate $\mathcal{A}_M$ représente le masque M_C comme suit :

$$\begin{aligned} M_C(s) &= M_C((\sigma_1, \Gamma_1)...(\sigma_n, \Gamma_n)) \\ &= (\lambda(x_1), \Gamma_1) ... (\lambda(x_n), \Gamma_n) \end{aligned}$$

où $x_1 = \delta_M((\sigma_1, \Gamma_1), x_{0_M})$ et $x_i = \delta_M((\sigma_i, \Gamma_i), x_{i-1})$ pour $2 \leq i \leq n$.

4.2.2 Lien entre la supervision de (G_C, M_C) et celle de (G, M)

La supervison de (G_C, M_C) est équivalente à celle de (G, M) dans le sens que si on dispose d'un superviseur pour l'une des paires on est capable d'en trouver un pour l'autre.

Dans un premier temps nous allons supposer que nous disposons d'un superviseur V_C pour (G_C, M_C) et nous allons montrer comment on peut en déduire un superviseur V pour (G, M) qui produit le même ω-langage généré en boucle fermée à une projection près.

Définition 4.3 *Soit* $\Sigma_C = \{(\sigma, \Gamma) \in \Sigma \times C : \sigma \in \Gamma\}$, *alors la fonction suivante*

$$\begin{aligned} P: \quad \Sigma_C^* &\to \Sigma^* \\ 1_{\Sigma_C} &\mapsto 1_\Sigma \\ m(\sigma, \Gamma_i) &\mapsto P(m)\sigma \end{aligned}$$

est une projection *de l'alphabet* Σ_C *sur l'alphabet* Σ.

Remarque 4.5 *On étend* P *aux langages comme suit :* $\forall L \subseteq \Sigma_C^*$,

$$P(L) = \{P(m) : m \in L\}$$

Notation 4.1 *On note* $P^{-1}(m) = \{k \in \Sigma_C^* : P(k) = m\}$.

Remarque 4.6 *On étend aussi P aux ω-langages comme suit : $\forall K \subseteq \Sigma_C^\omega$,*

$$P(K) = \lim(P(\text{pre}(K))) \cap S(G)$$

Proposition 4.1 *Soit G un SÉD défini sur l'alphabet Σ et G_C le SÉD contrôlé qui lui est associé. Soit P la projection définie auparavant. Alors on a:*

$$\begin{aligned} P(L(G_C)) &= L(G), \\ P^{-1}(L(G)) &= L(G_C), \\ P(S(G_C)) &= S(G) \text{ et} \\ P^{-1}(S(G)) &= S(G_C). \end{aligned}$$

Démonstration: *Le résultat est évident d'après la définition de G_C*

◇

Notation 4.2 *Pour tout $\Gamma_{C_i} \in C_C$, si Γ_{C_i} est définie à partir de l'entrée de commande Γ_i, alors on écrit*

$$P(\Gamma_{C_i}) = \Gamma_i$$

Définition 4.4 *Soit G un SÉD défini sur l'alphabet Σ et V_C un superviseur pour le*

SÉD contrôlé associé à G. On définit l'application suivante :

$$
\begin{aligned}
M_{V_C}: \; L(G) &\rightarrow (\Sigma_o \times C)^* \\
1_\Sigma &\mapsto 1_{\Sigma_o \times C} \\
k\sigma &\mapsto \begin{cases} M_{V_C}(k)(\sigma_o, \Gamma) & \text{si } \sigma \in \Gamma \text{ et } \sigma \text{ est observable} \\ M_{V_C}(k) & \text{si } \sigma \in \Gamma \text{ et } \sigma \text{ est inobservable} \\ \text{non définie} & \text{si } \sigma \notin \Gamma \end{cases}
\end{aligned}
$$

où

σ_o *est le dernier symbole de* $M(k\sigma)$ *si* σ *est observable*

et

$\Gamma = P(V_C(M_{V_C}(k)))$.

M_{V_C} *est appelée* l'observation contrôlée *de* k.

Lemme 4.1 *Soit* $k \in L(V/G)$ *et* $k' \in L(V_C/G_C)$ *tel que* $P(k') = k$. *Alors on a* $M_C(k') = M_{V_C}(k)$.

Démonstration:

Montrons le résultat par récurrence sur la longueur du mot k *(i.e. sur la longueur du mot* k'*).*

Pour $k = 1_\Sigma$ *le résultat est vrai, en effet*

$$M_{V_C}(1_\Sigma) = 1_{\Sigma_o \times C} = M_C(1_{\Sigma_C})$$

et on a $1_{\Sigma_C} = P^{-1}(1_\Sigma)$.

Supposons la propriété vraie pour tout mot de longueur $\leq n-1$ *et montrons qu'elle reste vraie pour tout mot de longueur* n*:*

Supposons que k *et* k' *s'écrivent sous la forme* $k = l\sigma$ *et* $k' = l'(\sigma, \Gamma_j)$.

Alors deux cas se présentent:

Cas 1 : σ ***est inobservable après*** l

Dans ce cas on a $M_{V_C}(k) = M_{V_C}(l)$. *D'autre part on a nécessairement* $M_C(k') = M_C(l')$.

Et d'après l'hypothèse de récurrence on a $M_{V_C}(l) = M_C(l')$.

$\Longrightarrow M_{V_C}(k) = M_C(k')$.

Cas 2 : σ ***est observable après*** l

$M_{V_C}(k) = M_{V_C}(l\sigma) = M_{V_C}(l)(\sigma_o, \Gamma_i)$ *avec* σ_o *est l'observation de* σ *et* $\Gamma_i = P(\Gamma_{C_i}) = P(V_C(M_{V_C}(l)))$.

D'autre part $k' = l'(\sigma, \Gamma_j)$ *avec* $l' \in P^{-1}(l)$ *et* $\Gamma_j = P(\Gamma_{C_j}) = P(V_C(M_C(l')))$.

D'après l'hypothèse de récurrence, on a $M_{V_C}(l) = M_C(l')$

$\Longrightarrow M_C(k') = M_C(l')(\sigma_o, \Gamma_j) = M_{V_C}(l)(\sigma_o, \Gamma_j) = M_{V_C}(l)(\sigma_o, \Gamma_i) = M_{V_C}(k)$.

Puisque $\Gamma_j = P\left(\Gamma_{C_j}\right) = P\left(V_C(M_C(l'))\right) = P\left(V_C\left(M_{V_C}(l)\right)\right) = \Gamma_i$.

◇

Proposition 4.2 *Soit G un SÉD défini sur l'alphabet Σ, pour lequel on associe un masque d'observation M, et V_C un superviseur pour (G_C, M_C) complet et non-bloquant, où G_C est le SÉD contrôlé associé à G et M_C son masque d'observation (défini à partir de M). Soit M_{V_C} le masque d'observation introduit dans la définition précédente. Alors la fonction suivante*

$$V = P \circ V_C \circ M_{V_C} \circ M^{-1}$$

définit un superviseur complet et non-bloquant pour (G, M) tel que :

$$L(V/G) = P\,(L(V_C/G_C))$$

et

$$S(V/G) = P\,(S(V_C/G_C))$$

Démonstration:

Commençons par montrer que V est bien définie :

Soit $m_1, m_2 \in \mathrm{dom}(V)$ telles que $m_1 = m_2$, montrons que $V(m_1) = V(m_2)$:

On a

$$V(m_1) = P\left(V_C\left(M_{V_C}(k_1)\right)\right)$$

et

$$V(m_2) = P\left(V_C\left(M_{V_C}(k_2)\right)\right)$$

avec $M(k_1) = M(k_2) = m_1 = m_2$ *(qu'on désigne par* m*).*

Pour montrer le résultat, nous allons montrer, par récurrence sur la longueur du mot m*, que* $M_{V_C}(k_1) = M_{V_C}(k_2)$.

Pour $m = 1_\Sigma$*, on a :*

$$M_{V_C}(k_1) = 1_{\Sigma_o \times C}$$

et

$$M_{V_C}(k_2) = 1_{\Sigma_o \times C}$$

Il est clair donc que

$$M_{V_C}(k_1) = M_{V_C}(k_2)$$

Supposons que la propriété soit vraie pour tout m *de longueur* $\leq n-1$*, et montrons qu'elle reste vraie pour tout mot* $m = a_1 \ldots a_n$ *de longueur* n*: si on suppose que* k_1 *et* k_2 *s'écrivent come suit :*

$$k_1 = \underbrace{\sigma_1 \ldots \sigma_{l_1}}_{k_1'} \sigma_{l_1+1} \ldots \sigma_{n_1}$$

et

$$k_2 = \underbrace{\sigma'_1 ... \sigma'_{l_2}}_{k'_2} \sigma'_{l_2+1} ... \sigma'_{n_2}$$

avec $n_1, n_2 \geq n$, $M(k'_1) = M(k'_2) = a_1 ... a_{n-1}$, *et* $M(k'_1 \sigma_{l_1+1}) = M\left(k'_2 \sigma'_{l_2+1}\right) = a_1 ... a_n$. *Alors d'après l'hypothèse de récurrence on a* $V_C(M_{V_C}(k'_1)) = V_C(M_{V_C}(k'_2)) = \Gamma_{C_i}$.

Posons $k''_1 = k'_1 \sigma_{k_1+1}$ *et* $k''_2 = k'_2 \sigma'_{k_2+1}$, *alors on a*

$$M_{V_C}(k''_1) = M_{V_C}(k'_1)(a_n, \Gamma_i)$$

et

$$M_{V_C}(k''_2) = M_{V_C}(k'_2)(a_n, \Gamma_i)$$

et

$$M_{V_C}(k_1) = M_{V_C}(k''_1)$$

et

$$M_{V_C}(k_2) = M_{V_C}(k''_2)$$

$\Rightarrow M_{V_C}(k_1) = M_{V_C}(k_2)$

$\Rightarrow V(m_1) = V(m_2)$.

Soit $m \in L(V/G)$, *montrons qu'il existe* $m' \in L(V_C/G_C)$ *tel que* $P(m') = m$.

Raisonnement par récurrence sur la longueur du mot m.

Pour le mot vide la propriété est vraie; supposons qu'elle soit vraie pour tout mot de longueur $n-1$ et montrons qu'elle reste vraie pour tout mot de longueur n:

$$m = k\sigma \in L(V/G) \Leftrightarrow k \in L(V/G) \text{ et } M(k) \in \mathrm{dom}(V) \text{ et } \sigma \in V(M(k)) \text{ et } k\sigma \in L(G)$$

D'abord $k \in L(V/G) \Rightarrow \exists k' \in L(V_C/G_C)$ *tel que* $P(k') = k$ *(d'après l'ypothèse de récurrence).*

D'autre part, $M(k) \in \mathrm{dom}(V) \Rightarrow M_{V_C}(M^{-1}(M(k))) \in \mathrm{dom}(V_C)$. *Puisque* $k \in M^{-1}(M(k))$, *on a alors* $M_{V_C}(k) \in \mathrm{dom}(V_C)$.

D'après le lemme, on a $M_C(k') = M_{V_C}(k)$, *ce qui implique* $M_C(k') \in \mathrm{dom}(V_C)$.

Soit $\Gamma_i = V(M(k))$, *montrons que* $(\sigma, \Gamma_i) \in V_C(M_C(k'))$:

$$\begin{aligned} V(M(k)) &= P(V_C(M_{V_C}(k))) \\ &= P(V_C(M_C(k'))) \\ &= \Gamma_i \end{aligned}$$

$\Rightarrow V_C(M_C(k')) = \Gamma_{C_i}$.

Il reste finalement à montrer que $k'(\sigma, \Gamma_i) \in L(G_C)$:

On a $k\sigma \in L(G) \Rightarrow P^{-1}(k\sigma) \subseteq L(G_C)$.

Or

$$P(k'(\sigma,\Gamma_i)) = P(k')\sigma$$
$$= k\sigma$$

$\Rightarrow k'(\sigma,\Gamma_i) \in P^{-1}(k\sigma)$

$\Rightarrow k'(\sigma,\Gamma_i) \in L(G_C)$

On a donc $m' = k'(\sigma,\Gamma_i) \in L(V_C/G_C)$ *et* $P(m') = m$.

$$\Longrightarrow L(V/G) \subseteq P(L(V_C/G_C)).$$

Montrons l'autre inclusion : soit m *un mot de* $L(V_C/G_C)$, *montrons que* $P(m) \in L(V/G)$.

Par récurrence sur la longueur du mot m:

Pour $m = 1_{\Sigma_C}$ *le résultat est vrai, puisque* $P(1_{\Sigma_C}) = 1_\Sigma \in L(V/G)$.

Supposons que le résultat soit vrai pour tout mot de longueur $\leq n-1$ *et montrons*

qu'il reste vrai pour tout mot de longueur n:

$m = k(\sigma,\Gamma_i) \in L(V_C/G_C) \Leftrightarrow k \in L(V_C/G_C)$ *et* $M_C(k) \in \mathrm{dom}(V_C)$ *et*
$(\sigma,\Gamma_i) \in V_C(M_C(k))$ *et* $k(\sigma,\Gamma_i) \in L(G_C)$

D'abord $k \in L(V_C/G_C) \Rightarrow k' = P(k) \in L(V/G)$ *(d'après l'hypothèse de récurrence).*

D'autre part, $M_C(k) \in \mathrm{dom}(V_C) \Rightarrow M(P(k)) \in \mathrm{dom}(V)$. *En effet,*

$\forall l \in M^{-1}(M(P(k)))$, *on a* $M_{V_C}(k') = M_{V_C}(l)$.

Et on a déjà prouvé que $M_C(k) = M_{V_C}(k') = M_{V_C}(l)$.

$\Rightarrow M_{V_C}(k') \in \mathrm{dom}(V_C)$

$\Rightarrow M_{V_C}(M^{-1}(M(k'))) \in \mathrm{dom}(V_C)$

$\Longrightarrow M(k') \in \mathrm{dom}(V)$.

D'autre part encore,

$$\begin{aligned} V(M(k')) &= P(V_C(M_{V_C}(k'))) \\ &= P(V_C(M_C(k))) \\ &= P(\Gamma_{C_i}) \\ &= \Gamma_i \end{aligned}$$

ce qui implique que $\sigma \in \Gamma_i$, *puisque* $(\sigma, \Gamma_i) \in \Gamma_{C_i}$.

Finalement on a,

$$\begin{aligned} k(\sigma, \Gamma_i) \in L(G_C) &\Rightarrow P(k)\sigma \in L(G) \\ &\Rightarrow k'\sigma \in L(G) \end{aligned}$$

$\Longrightarrow P(L(V_C/G_C)) \subseteq L(V/G)$

$\Longrightarrow P(L(V_C/G_C)) = L(V/G)$

Montrons maintenant que $S(V/G) = P(S(V_C/G_C))$*:*

$$\begin{aligned} S(V/G) &= \lim(L(V/G)) \cap S(G) \\ &= \lim(P(L(V_C/G_C))) \cap S(G) \\ &= \lim(P(\mathrm{pre}(S(V_C/G_C)))) \cap S(G) \quad (V_C \text{ est non-bloquant}) \\ &= P(S(V_C/G_C)) \end{aligned}$$

Montrons que V *est non-bloquant:*

On a

$$\begin{aligned} & S(V/G) = \lim(L(V/G)) \cap S(G) \\ \Rightarrow \quad & S(V/G) \subseteq \lim(L(V/G)) \\ \Rightarrow \quad & \mathrm{pre}(S(V/G)) \subseteq \mathrm{pre}(\lim(L(V/G))) \\ \Rightarrow \quad & \mathrm{pre}(S(V/G)) \subseteq L(V/G) \end{aligned}$$

Montrons maintenant que $L(V/G) \subseteq \mathrm{pre}(S(V/G))$.

Soit $m \in L(V/G)$, *puisque* $L(V/G) = P(L(V_C/G_C))$, *il existe* $m' \in L(V_C/G_C)$ *tel que* $P(m') = m$.

$$\begin{aligned} m' \in L(V_C/G_C) = \mathrm{pre}(S(V_C/G_C)) \quad &\Rightarrow \quad \exists s' \in (\Sigma \times C)^\omega \text{ tel que } m's' \in S(V_C/G_C) \\ &\Rightarrow \quad P(m's') \in P(S(V_C/G_C)) = S(V/G) \\ &\Rightarrow \quad P(m')\, s \in S(V/G) \\ &\Rightarrow \quad ms \in S(V/G) \\ &\Rightarrow \quad m \in \mathrm{pre}(S(V/G)) \end{aligned}$$

$\Longrightarrow L(V/G) \subseteq \text{pre}\,(S(V/G))$

$\Longrightarrow L(V/G) = \text{pre}\,(S(V/G))$.

Montrons que V est complet:

Soit $m \in M\,(L(V/G))$, montrons que $m \in \text{dom}(V)$:

$m \in M\,(L(V/G)) \Rightarrow \exists m' \in L(V/G)$ *tel que* $M(m') = m$

$m' \in L(V/G) \Rightarrow \exists m'' \in L(V_C/G_C)$ *tel que* $P\,(m'') = m'$

$m'' \in L(V_C/G_C) \Rightarrow M_C\,(m'') \in \text{dom}(V_C)$ *(V_C est complet)*

$\Rightarrow M_{V_C}(m') \in \text{dom}(V_C)$, *puisque* $M_C\,(m'') = M_{V_C}\,(m')$

$\Rightarrow \forall k \in M^{-1}(m)$, *on a* $M_{V_C}(k) \in \text{dom}(V_C)$, *puisque* $M_{V_C}(k) = M_{V_C}(m')$

$\Rightarrow m \in \text{dom}(V)$.

◇

Remarque 4.7 *Si on dispose d'un système tel que celui de la figure 4.1, alors on définit le masque M_{V_C} dont la structure est illustrée à la figure 4.2. On obtient finalement le système en boucle fermée de la figure 4.3.*

Nous allons maintenant montrer qu'à partir d'un superviseur V pour (G, M), il est possible de construire un superviseur V_C pour (G_C, M_C) qui fait en sorte que le ω-langage généré sous sa supervision soit équivalent à celui généré sous la supervision de V.

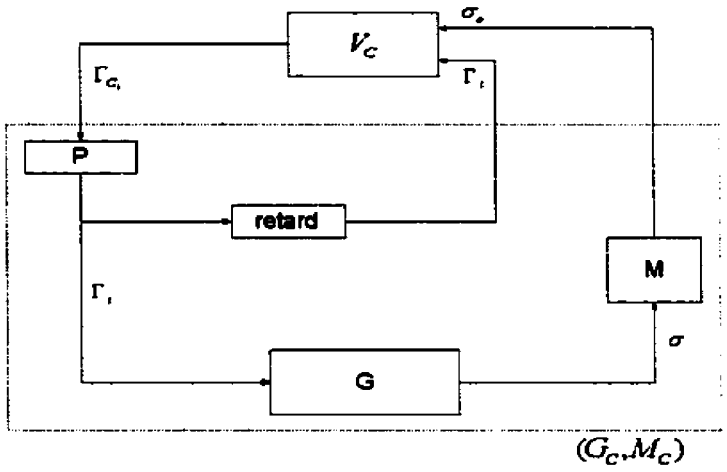


Figure 4.1: Système en boucle fermée V_C/G_C

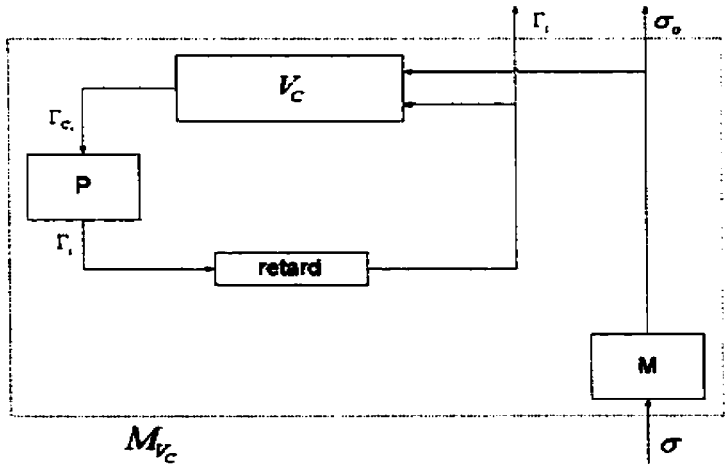


Figure 4.2: Structure du masque M_{V_C}

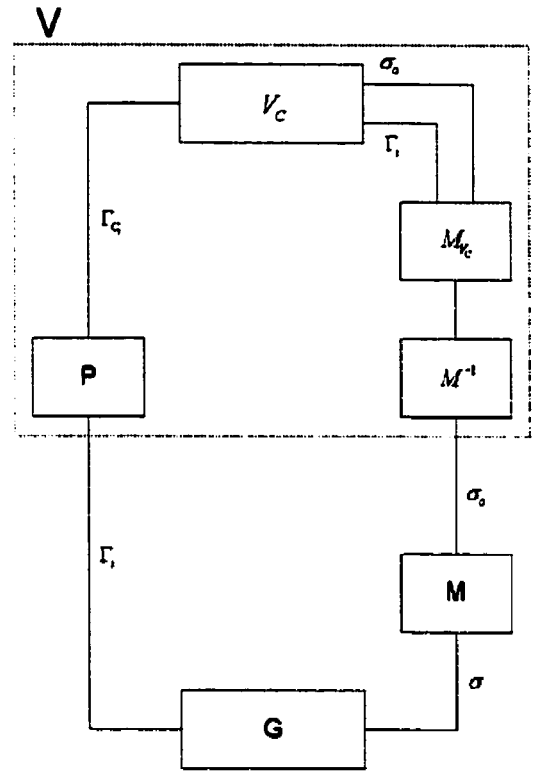


Figure 4.3: Système en boucle fermée V/G

Proposition 4.3 *Soit $V : \Sigma_o^* \longrightarrow C$ un superviseur complet et non-bloquant pour (G, M). On définit la fonction suivante:*

$$V_C : \ (\Sigma_o \times C)^* \ \rightarrow \ C_C$$

$$m \quad \mapsto \quad \begin{cases} \Gamma_{C_i} & \text{si } \Gamma_{C_i} \text{ existe} \\ \text{non défini} & \text{sinon} \end{cases}$$

avec Γ_{C_i} est l'entrée de commande définie à partir de $\Gamma_i = V\left(M\left(P\left(M_C^{-1}(m)\right)\right)\right)$.

On a alors V_C est un supervieur complet et non-bloquant pour (G_C, M_C) tel que

$$P\left(L\left(V_C/G_C\right)\right) = L(V/G)$$

et

$$P\left(S\left(V_C/G_C\right)\right) = S(V/G)$$

Démonstration:

Commençons par montrer que V_C est bien défini.

Soient $m_1, m_2 \in (\Sigma_o \times C)^*$ telles que $m_1 = m_2 = (\sigma_1, \Gamma_1)...(\sigma_n, \Gamma_n) = m$, montrons que $V_C(m_1) = V_C(m_2)$.

Soit $k_1 \in M_C^{-1}(m_1)$ et $k_2 \in M_C^{-1}(m_2)$, montrons par récurrence sur la longueur du mot m que $M\left(P(k_1)\right) = M\left(P(k_2)\right)$.

Pour $m = 1_{\Sigma_o \times C}$, la propriété est vraie. En effet, si on suppose que

k_1 et k_2 s'écrivent sous la forme $k_1 = (\sigma_1, \Gamma_1)...(\sigma_{n_1}, \Gamma_{n_1})$, et $k_2 = (\sigma'_1, \Gamma_1)...(\sigma'_{n_2}, \Gamma_{n_2})$.

Alors on a $P(k_1) = \sigma_1...\sigma_{n_1}$ et $P(k_2) = \sigma'_1...\sigma'_{n_2}$.

Puisque $M_C(k_1) = M_C(k_2) = 1_{\Sigma_o \times C}$, on a $M(P(k_1)) = 1_{\Sigma_o}$ et $M(P(k_2)) = 1_{\Sigma_o}$.

$\Longrightarrow M(P(k_1)) = M(P(k_2))$.

Supposons que le résultat soit vrai pour tous les mots de longueur $\leq n-1$ et montrons qu'il reste vrai pour tous les mots de longueur n:

Supposons que m s'écrit sous la forme $m = (a_1, \Gamma_1)...(a_n, \Gamma_n)$. Supposons aussi que $k_1 = \underbrace{(\sigma_1, \Gamma_1)...(\sigma_{l_1}, \Gamma_{l_1})}_{k'_1}(\sigma_{l_1+1}, \Gamma_{l_1+1})...(\sigma_{n_1}, \Gamma_{n_1})$ et $k_2 = \underbrace{(\sigma'_1, \Gamma'_1)...(\sigma'_{l_1}, \Gamma'_{l_2})}_{k'_2}(\sigma'_{l_2+1}, \Gamma'_{l_2+1})...(\sigma'_{n_2}, \Gamma'_{n_2})$ tel que $M_C(k_1) = M_C(k'_1)(a_n, \Gamma_n) = m_1$ et $M_C(k_2) = M_C(k'_2)(a_n, \Gamma_n) = m_2$.

Or $m_1 = m_2 \Rightarrow M_C(k'_1) = M_C(k'_2) = (a_1, \Gamma_1)...(a_{n-1}, \Gamma_{n-1})$.

D'après l'hypothèse de récurrence, on a $M(P(k'_1)) = M(P(k'_2))$. D'autre part $M(P(k_1)) = M\left(P(k'_1)\sigma_{l_1+1}P\left((\sigma_{l_1+1}, \Gamma_{l_1+1})...(\sigma_{n_1}, \Gamma_{n_1})\right)\right) = M(P(k'_1))a_n$, et $M(P(k_2)) = M\left(P(k'_2)\sigma_{l_2+1}P\left((\sigma_{l_2+1}, \Gamma_{l_2+1})...(\sigma_{n_2}, \Gamma_{n_2})\right)\right) = M(P(k'_2))a_n$.

$\Rightarrow M(P(k_1)) = M(P(k_2))$.

Montrons maintenant que $P\left(L\left(V_C/G_C\right)\right) = L(V/G)$.

Nous allons montrer dans un premier temps que $P\left(L\left(V_C/G_C\right)\right) \subseteq L(V/G)$.

Soit $m \in L\left(V_C/G_C\right)$, montrons par récurrence sur la longueur l du mot

m que $P(m) \in L(V/G)$.

Pour $l = 0$, *le résultat est vrai. En effet, on a* $P(1_{\Sigma_C}) = 1_\Sigma \in L(V/G)$.

Supposons que le résultat soit vrai pour tout mot de longueur $\leq n-1$ *et montrons que ça reste aussi vrai pour tout mot de longueur n:*

$$m = k(\sigma, \Gamma_i) \in L(V_C/G_C) \Leftrightarrow k \in L(V_C/G_C) \text{ et } M_C(k) \in \text{dom}(V_C) \text{ et } (\sigma, \Gamma_i) \in V_C(M_C(k)) \text{ et } k(\sigma, \Gamma_i) \in L(G_C)$$

D'abord $k \in L(V_C/G_C) \Longrightarrow P(k) = k' \in L(V/G)$ *(d'après l'hypothèse de récurrence).*

D'autre part

$$\begin{aligned} M_C(k) \in \text{dom}(V_C) &\Rightarrow M\left(P\left(M_C^{-1}(M_C(k))\right)\right) \in \text{dom}(V) \\ &\Rightarrow M(P(k)) \in \text{dom}(V) \\ &\Rightarrow M(k') \in \text{dom}(V) \end{aligned}$$

D'autre part encore

$$\begin{aligned} V_C(M_C(k)) = \Gamma_{C_i} &\Rightarrow V\left(M\left(P\left(M_C^{-1}(M_C(k))\right)\right)\right) = \Gamma_i \\ &\Rightarrow V(M(P(k))) = \Gamma_i \\ &\Rightarrow V(M(k')) = \Gamma_i \\ &\Rightarrow \sigma \in V(M(k')) \end{aligned}$$

Finalement, on a $k(\sigma, \Gamma_i) \in L(G_C) \Rightarrow k'\sigma \in L(G)$.

$\Longrightarrow k'\sigma \in L(V/G)$

$\Longrightarrow P\left(L\left(V_C/G_C\right)\right) \subseteq L(V/G)$.

Montrons l'autre inclusion:

Soit $m \in L(V/G)$, *montrons par récurrence sur la longueur du mot que* $m \in P\left(L\left(V_C/G_C\right)\right)$.

Pour $m = 1_\Sigma$, *le résultat est vrai, puisque* $m = P\left(1_{\Sigma_C}\right) \in P\left(L\left(V_C/G_C\right)\right)$.

Supposons que le résultat soit vrai pour tout mot de longueur $\leq n-1$ *et montrons que ça reste aussi vrai pour tout mot de longueur* n:

$$m = k\sigma \in L(V/G) \Leftrightarrow k \in L\left(V/G\right) \text{ et } M(k) \in \mathrm{dom}\left(V\right) \text{ et } \sigma \in V\left(M(k)\right) \text{ et } k\sigma \in L\left(G\right)$$

D'abord, d'après l'hypothèse de récurrence on a $k \in L(V/G) \Rightarrow \exists k' \in L\left(V_C/G_C\right)$ *telle que* $P\left(k'\right) = k$.

D'autre part

$$\begin{aligned} M(k) \in \mathrm{dom}\left(V\right) &\Rightarrow M\left(P(k')\right) \in \mathrm{dom}(V) \\ &\Rightarrow M\left(P\left(M_C^{-1}\left(M_C(k')\right)\right)\right) \in \mathrm{dom}(V) \\ &\Rightarrow M_C\left(k'\right) \in \mathrm{dom}(V_C) \end{aligned}$$

D'autre part encore $\sigma \in \Gamma_i = V\left(M(k)\right) \Rightarrow (\sigma, \Gamma_i) \in \Gamma_{C_i} = V_C\left(M_C\left(k'\right)\right)$.

Finalement on a $k\sigma \in L(G)$ *et* $P\left(k'(\sigma, \Gamma_i)\right) = k\sigma \Rightarrow k'(\sigma, \Gamma_i) \in L\left(G_C\right)$.

$\Longrightarrow k'(\sigma, \Gamma_i) \in L(V_C/G_C)$

$\Longrightarrow L(V/G) \subseteq P(L(V_C/G_C))$.

$\Longrightarrow L(V/G) = P(L(V_C/G_C))$.

Montrons maintenant que $S(V/G) = P(S(V_C/G_C))$*:*

$$\begin{aligned} P(S(V_C/G_C)) &= \lim(P(\mathrm{pre}(S(V_C/G_C)))) \cap S(G) \\ &= \lim(P(L(V_C/G_C))) \cap S(G) \\ &= \lim(L(V/G)) \cap S(G) \\ &= S(V/G) \end{aligned}$$

Montrons l'autre inclusion :

Soit $s \in S(V/G)$*, alors on a* $\mathrm{pre}(s) \subseteq L(V/G)$.

Posons $K = \{k \in L(V_C/G_C) : P(k) \in \mathrm{pre}(s)\}$. *Alors on a,* $\lim(K) \neq \emptyset$ *(d'après le lemme de König). Soit* $s' \in \lim(K)$*, alors on a* $s' \in \lim(L(V_C/G_C))$ *et*

$$\begin{aligned} P(s') &= \lim(P(\mathrm{pre}(s'))) \cap S(G) \\ &\subseteq \lim(P(k)) \cap S(G) \\ &\subseteq \lim(\mathrm{pre}(s)) \cap S(G) \\ &\subseteq \mathrm{clo}(\{s\}) \cap S(G) \\ &\in \{s\} \\ &= s \end{aligned}$$

Or $s \in S(G) \Rightarrow s' \in S(G_C)$

$\Rightarrow s' \in S(G_C) \cap \lim(L(V_C/G_C))$

$\Rightarrow s' \in S(V_C/G_C)$

$\Rightarrow s = P(s') \in P\left(S(V_C/G_C)\right)$

$\Rightarrow S(V/G) \subseteq P\left(S(V_C/G_C)\right)$

$\Rightarrow S(V/G) = P\left(S(V_C/G_C)\right).$

Montrons que V_C *est complet, c.-à.-d. montrons que* $M_C\left(L\left(V_C/G_C\right)\right) \subseteq$ $\mathrm{dom}\left(V_C\right)$*:*

Soit $k \in M_C\left(L\left(V_C/G_C\right)\right)$*, montrons que* $k \in \mathrm{dom}\left(V_C\right)$.

$k \in M_C\left(L\left(V_C/G_C\right)\right) \Rightarrow \exists k' \in L\left(V_C/G_C\right) : M_C\left(k'\right) = k$

$$
\begin{aligned}
k' \in L\left(V_C/G_C\right) &\Rightarrow P\left(k'\right) \in L(V/G) \\
&\Rightarrow M\left(P(k')\right) \in \mathrm{dom}(V) \ \textit{(V est complet)} \\
&\Rightarrow M\left(P\left(M_C^{-1}\left(M_C\left(k'\right)\right)\right)\right) \in \mathrm{dom}(V) \\
&\Rightarrow M\left(P\left(M_C^{-1}(k)\right)\right) \in \mathrm{dom}(V) \\
&\Rightarrow k \in \mathrm{dom}\left(V_C\right).
\end{aligned}
$$

Montrons finalement que V_C *est non-bloquant, c.-à.-d. montrons que* $L\left(V_C/G_C\right) = \mathrm{pre}\left(S\left(V_C/G_C\right)\right)$.

Il est évident que $\mathrm{pre}\left(S\left(V_C/G_C\right)\right) \subseteq L\left(V_C/G_C\right)$*, il nous reste à montrer l'autre inclusion.*

Soit $m \in L(V_C/G_C)$, *on a alors* $m' = P(m) \in L(V/G) = \text{pre}(S(V/G))$

(V est non-bloquant)

$\Rightarrow \exists s' = \sigma_1\sigma_2... \in \Sigma^\omega$ *tel que* $m's' \in S(V/G) = P(S(V_C/G_C))$.

$\Rightarrow ms \in S(V_C/G_C)$, *avec* $s = (\sigma_1, \Gamma_1)(\sigma_2, \Gamma_2)$ ... *et* $\Gamma_1 = V(M(m'))$,

$\Gamma_2 = V(M(m'\sigma_1))$...

$\Rightarrow m \in \text{pre}(S(V_C/G_C))$

$\Rightarrow L(V_C/G_C) \subseteq \text{pre}(S(V_C/G_C))$

$\Rightarrow L(V_C/G_C) = \text{pre}(S(V_C/G_C))$.

$\Rightarrow V_C$ *est non-bloquant.*

◇

4.3 Modélisation des événements non observables

Pour superviser le SÉD contrôlé G_C nous allons introduire un nouveau SÉD G_o que nous appellerons SÉD observé, qui correspondera à ce qu'on observe de la dynamique de G_C, et nous allons montrer qu'il est toujours possible de déduire un superviseur pour G_C à partir d'un superviseur pour G_o. L'avantage de cette opération est qu'on dispose des outils qui nous permettent de synthétiser un superviseur pour G_o.

Pour simplifier la structure de l'automate qui représente G_o, il s'est avéré utile de modifier la structure du mécanisme d'observation de G_C. Dans cette section nous

allons introduire ce nouveau masque qui présente la particularité que, même si un superviseur pour G_C est incapable de détecter l'occurrence d'un événement, il est quand même capable de mémoriser l'entrée de commande qu'il a appliquée et qui a permis l'occurrence de cet événement, ceci fera donc partie de l'information sur laquelle le superviseur se base pour choisir la prochaine entrée de commande.

Définition 4.5 *Soit G un SÉD défini sur l'alphabet Σ pour lequel est associé l'alphabet des observations Σ_o et soit G_C le SÉD contrôlé associé à G. On définit le masque d'observation suivant:*

$$
\begin{aligned}
\widetilde{M}_C : \qquad L(G_C) &\longrightarrow ((\Sigma_o \cup \{\epsilon\}) \times C)^* \\
1_{\Sigma \times C} &\longmapsto 1_{(\Sigma_o \cup \{\epsilon\}) \times C} \\
(\sigma_1, \Gamma_1)...(\sigma_n, \Gamma_n) &\longmapsto \widetilde{M}_C((\sigma_1, \Gamma_1)...(\sigma_{n-1}, \Gamma_{n-1}))(a, \Gamma_n) \\
&\text{ou} \\
(\sigma_1, \Gamma_1)...(\sigma_n, \Gamma_n) &\longmapsto \widetilde{M}_C((\sigma_1, \Gamma_1)...(\sigma_{n-1}, \Gamma_{n-1}))(\epsilon, \Gamma_n)
\end{aligned}
$$

Où $a \in \Sigma_o$ est le dernier symbole de $M(\sigma_1\sigma_2...\sigma_n)$ si σ_n est observable suite à la génération de $\sigma_1\sigma_2...\sigma_{n-1}$. La dernière clause s'applique lorsqu'on n'est pas capable de détecter l'occurrence de l'événement (σ_n, Γ_n) (on dit que (σ_n, Γ_n) est inobservable *après la génération de la séquence $(\sigma_1, \Gamma_1)...(\sigma_{n-1}, \Gamma_{n-1})$).*

Définition 4.6 *Soit G un SÉD, G_C le SÉD contrôlé associé à G et $\widetilde{M}_C$ le masque d'observation défini plus haut. Un superviseur sous observations partielles pour*

$\left(G_C, \widetilde{M}_C\right)$ *est une fonction partielle*

$$\tilde{V}_C : ((\Sigma_o \cup \{\epsilon\}) \times C)^* \longrightarrow C_C$$

qui vérifie la condition

$$M_C(m_1) = M_C(m_2) \Rightarrow \tilde{V}_C\left(\widetilde{M}_C(m_1)\right) = \tilde{V}_C\left(\widetilde{M}_C(m_2)\right)$$

(elle exprime le fait que le superviseur ne peut changer d'entrée de commande que suite à la génération d'événements observables).

Le système en boucle fermée V_C/G_C est défini de façon analogue à V/G dans le chapitre précédent.

Notation 4.3 *On note π_ϵ la projection naturelle de l'alphabet $(\Sigma_o \times \{\epsilon\}) \times C$ sur l'alphabet $\Sigma_o \times C$, c.-à.-d. :*

$$\begin{aligned}
\pi_\epsilon : \ ((\Sigma_o \cup \epsilon) \times C)^* &\leftarrow (\Sigma_o \times C)^* \\
1_{(\Sigma_o \cup \epsilon) \times C} &\mapsto 1_{\Sigma_o \times C} \\
k(\sigma, \Gamma) &\mapsto \pi_\epsilon(k)(\sigma, \Gamma) \text{ si } \sigma \neq \epsilon \\
k(\epsilon, \Gamma) &\mapsto \pi_\epsilon(k)
\end{aligned}$$

On l'étend aux langages et aux ω-langages comme suit : $\pi_\epsilon(K) = \{\pi_\epsilon(m) : m \in K\}$, pour $K \subseteq L(G_C)$ et $\pi_\epsilon(T) = \lim(\pi_\epsilon(\mathrm{pre}(T)))$, pour $T \subseteq S(G_C)$.

Les deux propositions suivantes montrent que la supervision de (G_C, M_C) est équivalente à celle de $\left(G_C, \widetilde{M}_C\right)$.

Proposition 4.4 *Soit* $V_C : (\Sigma_o \times C)^* \longrightarrow C_C$ *un superviseur complet et non-bloquant pour* (G_C, M_C). *On définit la fonction suivante:*

$$\begin{aligned} \tilde{V}_C : \ ((\Sigma_o \cup \{\epsilon\}) \times C)^* &\rightarrow C_C \\ m &\mapsto V_C(\pi_\epsilon(m)) \end{aligned}$$

on a alors $\tilde{V}_C$ *est un superviseur complet et non-bloquant pour* $\left(G_C, \widetilde{M}_C\right)$ *tel que*

$$L\left(\tilde{V}_C/G_C\right) = L(V_C/G_C)$$

et

$$S\left(\tilde{V}_C/G_C\right) = S(V_C/G_C)$$

Démonstration:

Montrons que $\tilde{V}_C$ *est un superviseur pour* $(G_C, \widetilde{M}_C)$:

Soit $m_1, m_2 \in (\Sigma \times C)^*$ *tel que* $M_C(m_1) = M_C(m_2)$. *Alors*

$$\begin{aligned}
\tilde{V}_C\left(\widetilde{M}_C(m_1)\right) &= V_C\left(\pi_\epsilon\left(\widetilde{M}_C(m_1)\right)\right) \\
&= V_C\left(M_C(m_1)\right) \\
&= V_C\left(M_C(m_2)\right) \\
&= V_C\left(\pi_\epsilon\left(\widetilde{M}_C(m_2)\right)\right) \\
&= \tilde{V}_C\left(\widetilde{M}_C(m_2)\right)
\end{aligned}$$

Montrons maintenant que $L\left(\tilde{V}_C/G_C\right) = L(V_C/G_C)$, *c.-à.-d. montrons que* $\forall m \in (\Sigma \times C)^*$

$$m \in L\left(\tilde{V}_C/G_C\right) \Leftrightarrow m \in L(V_C/G_C)$$

Par récurrence sur la longueur de m.

Pour $m = 1_{\Sigma \times C}$ *le résultat est vrai.*

Supposons que ça soit vrai pour tout mot de longueur $\leq n-1$ *et montrons que c'est encore vrai pour tout mot de longueur* n:

$$
\begin{aligned}
m = k(\sigma, \Gamma_i) \in L(V_C/G_C) \quad &\Leftrightarrow \quad k \in L(V_C/G_C) \text{ et } M_C(k) \in \mathrm{dom}(V_C) \text{ et} \\
&\quad (\sigma, \Gamma_i) \in V_C(M_C(k)) \text{ et } k(\sigma, \Gamma_i) \in L(G_C) \\
&\Leftrightarrow \quad k \in L(\tilde{V}_C/G_C) \text{ (d'après l'hypothèse de récurrence)} \\
&\quad \text{et } \widetilde{M}_C(k) \in \mathrm{dom}(\tilde{V}_C) \text{ puisque } \pi_\epsilon\left(\widetilde{M}_C(k)\right) = M_C(k), \\
&\quad \text{et } (\sigma, \Gamma_i) \in \tilde{V}_C(\widetilde{M}_C(k)) \text{ puisque} \\
&\quad \tilde{V}_C(\widetilde{M}_C(k)) = V_C(M_C(k)), \text{ et } k(\sigma, \Gamma_i) \in L(G_C) \\
&\Leftrightarrow \quad m \in L\left(\tilde{V}_C/G_C\right)
\end{aligned}
$$

D'autre part, on a:

$$
\begin{aligned}
S(\tilde{V}_C/G_C) &= \lim\left(L(\tilde{V}_C/G_C)\right) \cap S(G_C) \\
&= \lim\left(L(V_C/G_C)\right) \cap S(G_C) \\
&= S(V_C/G_C)
\end{aligned}
$$

D'autre part encore, on a:

$$
\begin{aligned}
L(\tilde{V}_C/G_C) &= L(V_C/G_C) \\
&= \mathrm{pre}(S(V_C/G_C)) \\
&= \mathrm{pre}\left(S(\tilde{V}_C/G_C)\right)
\end{aligned}
$$

$\Longrightarrow \tilde{V}_C$ *est non-bloquant.*

Finalement on a

$$
\begin{aligned}
& M_C(L(V_C/G_C)) && \subseteq \ \mathrm{dom}(V_C) \ \textit{(V_C est complet)} \\
\Rightarrow \ & \pi_\epsilon\left(\widetilde{M}_C\left(L(V_C/G_C)\right)\right) && \subseteq \ \mathrm{dom}(V_C) \\
\Rightarrow \ & \pi_\epsilon\left(\widetilde{M}_C\left(L(\tilde{V}_C/G_C)\right)\right) && \subseteq \ \mathrm{dom}(V_C) \\
\Rightarrow \ & \widetilde{M}_C\left(L(\tilde{V}_C/G_C)\right) && \subseteq \ \mathrm{dom}(\tilde{V}_C) \\
\Rightarrow \ & \tilde{V}_C \textit{ est complet}
\end{aligned}
$$

◇

Proposition 4.5 *Soit* $\tilde{V}_C : ((\Sigma_o \cup \{\epsilon\}) \times C)^* \longrightarrow C_C$ *un superviseur complet et non-bloquant pour* $(G_C, \widetilde{M}_C)$ *alors la fonction suivante:*

$$
\begin{aligned}
V_C : \ (\Sigma_o \times C)^* & \rightarrow \ C_C \\
m & \mapsto \ \tilde{V}_C\left(\widetilde{M}_C(k)\right)
\end{aligned}
$$

où $k \in \widetilde{M}_C^{-1}(m)$, *définit un superviseur complet et non-bloquant pour* (G_C, M_C) *tel que*

$$L(V_C/G_C) = L(\tilde{V}_C/G_C)$$

et

$$S(V_C/G_C) = S(\tilde{V}_C/G_C)$$

Démonstration:

Tout d'abord V_C est bien défini car pour tout $k_1, k_2 \in M_C^{-1}(m)$, on a

$$M_C(k_1) = M_C(k_2) = m \Rightarrow \tilde{V}_C\left(\widetilde{M}_C(k_1)\right) = \tilde{V}_C\left(\widetilde{M}_C(k_2)\right)$$

(par définition de $\tilde{V}_C$).

Montrons que $L(V_C/G_C) = L\left(\tilde{V}_C/G_C\right)$, c.-à.-d. montrons que $\forall m \in (\Sigma \times C)^*$

$$m \in L(V_C/G_C) \Leftrightarrow m \in L\left(\tilde{V}_C/G_C\right)$$

Par récurrence sur la longueur de m.

Pour $m = 1_{\Sigma \times C}$ le résultat est vrai.

Supposons que ça soit vrai pour tout mot de longueur $\leq n-1$ et montrons que c'est encore vrai pour tout mot de longueur n:

$$
\begin{aligned}
m = k(\sigma, \Gamma_i) \in L(V_C/G_C) \quad &\Leftrightarrow \quad k \in L(V_C/G_C) \text{ et } M_C(k) \in \mathrm{dom}(V_C) \text{ et} \\
&\quad (\sigma, \Gamma_i) \in V_C(M_C(k)) \text{ et } k(\sigma, \Gamma_i) \in L(G_C) \\
&\Leftrightarrow \quad k \in L(\tilde{V}_C/G_C) \text{ (d'après l'hypothèse de récurrence),} \\
&\quad \text{et } \widetilde{M}_C(k) \in \mathrm{dom}(\tilde{V}_C) \text{ et } (\sigma, \Gamma_i) \in \tilde{V}_C(\widetilde{M}_C(k)) \\
&\quad \text{puisque } \tilde{V}_C(\widetilde{M}_C(k)) = V_C\left(M_C(k)\right), \text{ et} \\
&\quad k(\sigma, \Gamma_i) \in L(G_C) \\
&\Leftrightarrow \quad k(\sigma, \Gamma_i) \in L\left(\tilde{V}_C/G_C\right)
\end{aligned}
$$

$\Longrightarrow L(V_C/G_C) = L\left(\tilde{V}_C/G_C\right)$.

Il s'en suit que $S(V_C/G_C) = S\left(\tilde{V}_C/G_C\right)$, *et que* V_C *est complet et non-bloquant.*

◇

4.4 Conclusion

Dans ce chapitre nous avons introduit le concept de SÉD contrôlé et nous lui avons associé un masque d'observation qui tient compte de toute l'information disponible. Le système à superviser ainsi obtenu présente une structure plus riche en informations, dans le sens qu'elle offre plus de renseignements au superviseur qui devra le contrôler. Nous avons également montré que le problème de supervision de $\left(G_C, \widetilde{M}_C\right)$ est équivalent à notre problème de départ. Donc nous sommes passés

d'un problème intial à un autre qui lui est équivalent et qui sera plus façile à résoudre. Dans le prochain chapitre, nous allons justement donner la solution à ce nouveau problème.

CHAPITRE 5

SYNTHÈSE DE SUPERVISEURS SOUS OBSERVATIONS PARTIELLES

5.1 Introduction

Notre problème de départ était de concevoir un superviseur sous observations partielles pour un SÉD G, observé à travers un masque M. Or nous avons montré que ce problème peut se ramener à celui de la supervision du SÉD contrôlé G_C associé à G, observé à travers le masque $\widetilde{M}_C$, et nous avons spécifié comment on peut déduire un supervieur pour (G, M) étant donné un superviseur pour $\left(G_C, \widetilde{M}_C\right)$. Dans ce chapitre nous allons donner une approche qui permet de synthétiser un superviseur sous observations partielles pour un SÉD contrôlé qui répond à une certaine spécification (définie à partir de la spécification sur le superviseur pour (G, M)).

Cette approche de résolution, qui suppose qu'aussi bien la spécification que le SÉD contrôlé sont représentés par des automates finis sur mots infinis, comporte neuf étapes qui permettent de ramener le problème initial à celui du contrôle d'un automate fini et déterministe de Rabin pour la satisfaction de sa propre condition d'acceptation pour lequel on peut appliquer l'algorithme de synthèse du chapitre 2 (qui utilise le calcul de points fixes). Une fois obtenu un contrôleur pour cet au-

tomate, il est aisé d'en déduire un supervieur sous observations partielles pour le SÉD contrôlé de départ. Cette approche sera présentée plus en détail à la section suivante. Ses trois étapes principales sont expliquées dans les sections 5.4, 5.5 et 5.6.

5.2 Approche de résolution

5.2.1 Données initiales

Nos données de départ sont :

- Un SÉD G partiellement observé qui vérifie la condition

$$S(G) = \lim(L(G))$$

et qui est représenté par un automate fini de Büchi, avec une condition d'acceptation triviale:

$$\mathcal{G} = (\Sigma, X_G, \delta_G, x_{0_G}, X_G)\,.$$

- Une spécification sur le ω-langage généré en boucle fermée qu'on suppose représentée par un automate universel de Rabin avec une seule paire de sous-ensembles, et une fonction de transition totale:

$$\mathcal{R} = (\Sigma, X_E, \delta_E, x_{0_E}, (R_E, I_E))\,.$$

Remarque 5.1 *Ceci ne constitue pas une perte de généralité car la classe des langages reconnus par de tels automates est égale à la classe des langages ω-réguliers.*

- Un masque d'observation

$$M : \Sigma^* \longrightarrow \Sigma_o^*$$

représenté par un automate fini de Moore:

$$\widehat{\mathcal{A}}_M = \left(\Sigma, X_M, \widehat{\delta}_M, \widehat{x}_{0_M}, \Sigma_o, \widehat{\lambda}\right)$$

5.2.2 Algorithme de synthèse

À partir des données initiales, on utilise la procédure suivante pour synthétiser un superviseur sous observations partielles pour G complet, non-bloquant et qui respecte la spécification E:

- Étape 1 : construire l'automate fini de Büchi

$$\mathcal{G}_C = (\Sigma \times C, X_G, \delta_{G_C}, x_{0_G}, X_G)$$

qui représente le SÉD contrôlé G_C, et l'automate universel

de Rabin

$$\mathcal{R}_C = (\Sigma \times C, X_E, \delta_{E_C}, x_{0_E}, (R_E, I_E)) .$$

qui représente la nouvelle spécification sur le superviseur de G_C (il s'agit de $E_C = P^{-1}(E) \cap L(G_C)$).

- Étape 2 : faire le produit de $\mathcal{G}_C$ et $\mathcal{R}_C$; le résultat de cette opération est un automate universel de Rabin qui représente à la fois le SÉD contrôlé G_C et la spécification E_C:

$$\mathcal{A}_C = (\Sigma \times C, X, \delta_C, x_0, (R, I)) .$$

On construit également l'automate de Moore

$$\mathcal{A}_M = (\Sigma \times C, X_M, \delta_M, x_{0_M}, \Sigma_o \cup \{\epsilon\}, \lambda)$$

qui représente le masque

$$\widetilde{M}_C : (\Sigma \times C)^* \longrightarrow ((\Sigma_o \cup \{\epsilon\}) \times C)^*$$

- Étape 3 : construire l'automate de haut niveau qui représente à la fois le SÉD observé G_o et la spécification qui lui est associée, à

savoir

$$E_o = \widetilde{M}_C(N)$$

où N est le plus grand sous-langage normal par rapport à $\widetilde{M}_C$ et inclus dans E_C. Le résultat de cette opération est l'automate universel de Rabin suivant:

$$\mathcal{A}_o = ((\Sigma_o \cup \{\epsilon\}) \times C, Y_o, \delta_o, y_0, (R_o, I_o)) .$$

- Étape 4 : calculer le produit entre $\mathcal{A}_o$ et l'automate $\mathcal{A}_i$, qui représente le plus grand sous-langage ϵ-invariant de $((\Sigma_o \cup \{\epsilon\}) \times C)^\omega$. Cette étape sera expliquée en détail à la section 5.5. Appelons l'automate de Rabin déterministe qui résulte de cette opération $\mathcal{A}_{o_i} = ((\Sigma_o \cup \{\epsilon\}) \times C, Y_{o_i}, \delta_{o_i}, y_{0_i}, (R_{o_i}, I_{o_i}))$.

- Étape 5 : Construire l'automate de Rabin déterministe

$$\mathcal{A}_d = ((\Sigma_o \cup \{\epsilon\}) \times C, Z, \delta_d, z_0, (R_d, I_d)) .$$

qui vérifie la propriété

$$\pi_\epsilon(L(\mathcal{A}_d)) = \pi_\epsilon(L(\mathcal{A}_{o_i}))$$

et

$$\pi_\epsilon(S(\mathcal{A}_d)) = \pi_\epsilon(S(\mathcal{A}_{a_i}))$$

- Étape 6 : calculer un superviseur f_d pour $\mathcal{A}_d$ qui fait en sorte que les mots infinis générés sous sa supervision vérifient la condition d'acceptation de $\mathcal{A}_d$ (Ceci constitue également un superviseur sous observations totales pour le SÉD observé représenté par $\mathcal{A}_d$) en utilisant les résultats de Thistle et Wonham (voir chapitre 2).
- Étape 7 : déduire à partir de V_o un superviseur $\widetilde{V}_C$ pour $\left(G_C, \widetilde{M}_C\right)$.
- Étape 8 : à partir de $\widetilde{V}_C$, calculer un superviseur V_C pour (G_C, M_C) en utilisant la méthode donnée au chapitre précédent.
- Étape 9 : à partir de V_C, calculer un superviseur V pour (G, M) en utilisant la méthode donnée au chapitre précédent.

5.3 Supervision de l'automate $\mathcal{A}_C$ dont la dynamique est observée à travers le masque $\widetilde{M}_C$

Nous avons vu dans la section précédente qu'à l'étape 2 on construit l'automate universel de Rabin $\mathcal{A}_C$ qui représente le SÉD contrôlé G_C et la spécification E_C. Dans cette section, nous allons justifier cette étape en montrant que la supervision de $\left(G_C, \widetilde{M}_C\right)$ qui respecte la spécification E_C est équivalente à la supervision de $\mathcal{A}_C$, observé à travers $\widetilde{M}_C$, pour la satisfaction de sa propre condition d'acceptation.

Définition 5.1 *Soit $\mathcal{A}_C$ un automate de Rabin universel défini sur l'alphabet $\Sigma \times C$. Un superviseur pour $\mathcal{A}_C$ sous observations partielles à travers le masque $\widetilde{M}_C$: $(\Sigma \times C)^* \longrightarrow ((\Sigma_o \cup \{\epsilon\}) \times C)^*$ est une application*

$$f : ((\Sigma_o \cup \{\epsilon\}) \times C)^* \longrightarrow C_C$$

qui vérifie la condition suivante :

$$\pi_\epsilon(m_1) = \pi_\epsilon(m_2) \Rightarrow f(m_1) = f(m_2)$$

avec C_C est la famille d'entrées de commande associée à $\Sigma \times C$ Le langage généré par $\mathcal{A}_C$ sous la supervision de f est défini récursivement comme suit:

(i) $1_{\Sigma \times C} \in L(f/\mathcal{A}_C)$

(ii) Pour tout $k \in (\Sigma \times C)^*$, et pour tout $(\sigma, \Gamma) \in \Sigma \times C$ on a :

$$k(\sigma,\Gamma) \in L(f/\mathcal{A}_C) \;\Leftrightarrow\; k \in L(f/\mathcal{A}_C) \text{ et } \widetilde{M}_C(k) \in \mathrm{dom}(f) \text{ et } (\sigma,\Gamma) \in f\left(\widetilde{M}_C(k)\right) \text{ et } k(\sigma,\Gamma) \in L(\mathcal{A}_C)$$

Le ω-langage généré par $\mathcal{A}_C$ sous la supervision de f est l'ensemble des éléments de $\lim(L(f/\mathcal{A}_C))$ qui respectent la condition d'acceptation de $\mathcal{A}_C$, c.-à.-d. $\lim(L(f/\mathcal{A}_C)) \cap S(\mathcal{A}_C)$.

Définition 5.2 *Un superviseur pour $\left(\mathcal{A}_C, \widetilde{M}_C\right)$ est non-bloquant si $L(f/\mathcal{A}_C) =$*

$\mathrm{pre}\,(S\,(f/\mathcal{A}_C))$. *On dit qu'il est complet si* $\widetilde{M}_C\,(L\,(f/\mathcal{A}_C)) \subseteq \mathrm{dom}(f)$.

Proposition 5.1 $\tilde{V}_C : ((\Sigma_o \cup \{\epsilon\}) \times C)^* \longrightarrow C_C$ *est un superviseur complet et non-bloquant pour* $\left(G_C, \widetilde{M}_C\right)$, *tel que* $S\left(\tilde{V}_C/G_C\right) \subseteq E_C$ *ssi il est un superviseur complet et non-bloquant pour* $\left(\mathcal{A}_C, \widetilde{M}_C\right)$ *tel que*

$$L\left(\tilde{V}_C/\mathcal{A}_C\right) = L\left(\tilde{V}_C/G_C\right)$$

et

$$S\left(\tilde{V}_C/\mathcal{A}_C\right) = S\left(\tilde{V}_C/G_C\right)$$

Démonstration:

D'abord il est clair que

$$\tilde{V}_C \text{ est un superviseur pour } \left(G_C, \widetilde{M}_C\right)$$

$$\Updownarrow$$

$$\tilde{V}_C \text{ est un superviseur pour } \left(\mathcal{A}_C, \widetilde{M}_C\right)$$

Montrons alors que $L\left(\tilde{V}_C/G_C\right) = L\left(\tilde{V}_C/\mathcal{A}_C\right)$. *Pour cela nous allons montrer par récurrence sur la longueur du mot* m *que,*

$$\forall m \in (\Sigma \times C)^*, m \in L\left(\tilde{V}_C/G_C\right) \Leftrightarrow m \in L\left(\tilde{V}_C/\mathcal{A}_C\right)$$

Pour $|m| = 0$ le résultat est vrai.

Supposons que c'est vrai pour tout mot de longueur $\leq n-1$ et montrons que ça reste vrai pour tout mot de longueur n.

$$\begin{aligned} m = k(\sigma,\Gamma) \in L\left(\tilde{V}_C/G_C\right) &\Leftrightarrow k \in L\left(\tilde{V}_C/G_C\right) \text{ et } \widetilde{M}_C(k) \in \mathrm{dom}(\tilde{V}_C) \text{ et} \\ &\quad (\sigma,\Gamma) \in \tilde{V}_C\left(\widetilde{M}_C(k)\right) \text{ et } k(\sigma,\Gamma) \in L(G_C) \\ &\Leftrightarrow k \in L\left(\tilde{V}_C/\mathcal{A}_C\right) \text{ et } \widetilde{M}_C(k) \in \mathrm{dom}(\tilde{V}_C) \text{ et} \\ &\quad (\sigma,\Gamma) \in \tilde{V}_C\left(\widetilde{M}_C(k)\right) \text{ et } k(\sigma,\Gamma) \in L(\mathcal{A}_C) \\ &\Leftrightarrow k(\sigma,\Gamma) \in L\left(\tilde{V}_C/\mathcal{A}_C\right) \end{aligned}$$

Il reste a montrer maintenant que $S\left(\tilde{V}_C/G_C\right) = S\left(\tilde{V}_C/\mathcal{A}_C\right)$.

$$\begin{aligned} s \in S\left(\tilde{V}_C/G_C\right) &\Leftrightarrow s \bullet \lim\left(L\left(\tilde{V}_C/G_C\right)\right) \text{ et } s \in E_C \\ &\Leftrightarrow s \in \lim\left(L\left(\tilde{V}_C/\mathcal{A}_C\right)\right) \text{ et } s \in E_C \\ &\Leftrightarrow s \in \lim\left(L\left(\tilde{V}_C/\mathcal{A}_C\right)\right) \\ &\quad \text{et } s \text{ vérifie la condition d'acceptation de } \mathcal{R}_C \\ &\Leftrightarrow s \in \lim\left(L\left(\tilde{V}_C/\mathcal{A}_C\right)\right) \\ &\quad \text{et } s \text{ vérifie la condition d'acceptation de } \mathcal{A}_C \\ &\Leftrightarrow s \in S\left(\tilde{V}_C/\mathcal{A}_C\right). \end{aligned}$$

◇

5.4 Construction de l'automate de haut niveau

5.4.1 SÉD observé

Étant donné un SÉD contrôlé G_C et un masque d'observation $\widetilde{M}_C$, une étape importante pour la résolution du problème de synthèse d'un superviseur $\tilde{V}_C$ pour $\left(G_C, \widetilde{M}_C\right)$ est la construction d'un ω-automate qui représente le SÉD *observé* G_o correspondant à l'image de G_C telle que observée par $\tilde{V}_C$. Il est obtenu en appliquant le masque d'observation $\widetilde{M}_C$ sur le SÉD contrôlé G_C.

Comme on part d'un automate qui représente à la fois G_C et E_C, la construction que nous allons donner produit un automate de Rabin qui représente à la fois le SÉD $G_o = \left(\widetilde{M}_C(L(G_C), \widetilde{M}_C(S(G_C))\right)$ et la spécification $E_o = \widetilde{M}_C(N)$, où N est le plus grand ω-langage inclu dans E_C et normal par rapport à $\widetilde{M}_C$.

5.4.2 Algorithme

Supposons que l'on dispose de l'automate de Rabin suivant:

$$\mathcal{A}_C = (\Sigma \times C, X, \delta_C, x_0, \{(R_p, I_p),\, p \in P\})$$

qui représente la SÉD observé contrôlé G_C et la spécification E_C.

Supposons aussi que $\widetilde{M}_C$ est représenté par l'automate de Moore suivant :

$$\mathcal{A}_M = (\Sigma \times C, X_M, \delta_M, x_{0_M}, \Sigma_o \cup \{\epsilon\}, \lambda)\,.$$

Voici la procédure pour la construction de G_o:

- étape 1 : Construire le "produit" des deux automates $\mathcal{A}_C$ et $\mathcal{A}_M$. Le résultat de cette opération est l'automate "hybride" suivant:

$$\mathcal{A}_C \times \mathcal{A}_M = \left(\Sigma \times C, Y, \delta_{CM}, (x_0, x_{0_M}), \left\{\left(R'_p, I'_p\right), p \in P\right\}, \Sigma_o \cup \{\epsilon\}, \lambda'\right)$$

avec

$$Y = X \times X_M,$$

$$\delta_{CM}: \quad (\Sigma \times C) \times Y \quad \rightarrow \quad 2^Y$$

$$((\sigma, \Gamma_i), (x_1, x_2)) \quad \mapsto \quad \left\{ \begin{array}{r} (x'_1, x'_2) \in Y: \quad x'_1 \in \delta_C\left((\sigma, \Gamma_i), x_1\right) \\ \& \\ x'_2 \in \delta_M\left((\sigma, \Gamma_i), x_2\right) \end{array} \right\}$$

et pour tout $p \in P$, $\left(R'_p, I'_p\right)$ est définie à partir de (R_p, I_p) comme suit:

$$R'_p = \; R_p \times X_M$$

et

$$I'_p = \; I_p \times X_M$$

Enfin λ' est définie à partir de λ comme suit:

$$\lambda': \quad Y \quad \longrightarrow \quad \Sigma_o \cup \{\epsilon\}$$

$$(x, x_M) \quad \longmapsto \quad \lambda(x_M)$$

• étape 2 : Calculer la composante accessible de $\mathcal{A}_C \times \mathcal{A}_M$; on obtient ainsi:

$$(\mathcal{A}_C \times \mathcal{A}_M)_{acc} = \left(\Sigma \times C, Y_{ac}, \delta_{CM_{ac}}, y_0, \left\{\left(R''_p, I''_p\right), p \in P\right\}, \Sigma_o \cup \{\epsilon\}, \lambda'_{ac}\right)$$

où

Y_{ac} : est l'ensemble des états accessibles,

$\delta_{CM_{ac}}$: la restriction de δ_{CM} à $(\Sigma \times C) \times Y_{ac}$,

$y_0 = (x_0, x_{0_M})$

$R''_p = R'_p \cap Y_{ac}, \forall p \in P$ et

$I''_p = I'_p \cap Y_{ac}, \forall p \in P.$

λ'_{ac} : la restriction de λ' à Y_{ac},

• étape 3 : Appliquer le masque d'observation sur $(\mathcal{A}_C \times \mathcal{A}_M)_{acc}$; le résultat de cette opération est l'automate de Rabin suivant:

$$\mathcal{A}_o = \left((\Sigma_o \cup \{\epsilon\}) \times C, Y_o, \delta_o, y_0, \left\{(R_{o_p}, I_{o_p}), p \in P\right\}\right)$$

avec

$Y_o = Y_{ac}$,

$\delta_o : (\Sigma_o \cup \{\epsilon\} \times C) \times Y_o \longrightarrow 2^{Y_o}$ est définie comme suit:

$\forall \sigma \in \Sigma_o \cup \{\epsilon\}$, $\forall \Gamma_i \in C$ et $\forall y \in Y_o$,

$$\delta_o((\sigma, \Gamma_i), y) = \begin{cases} \bigcup_{\sigma' \in \Sigma(\sigma, \Gamma_i, y)} \delta_{CM_{ac}}((\sigma', \Gamma_i), y) & \text{si } \Sigma(\sigma, \Gamma_i, y) \neq \emptyset \\ \emptyset & \text{sinon} \end{cases}$$

avec

$$\Sigma(\sigma, \Gamma_i, y) = \left\{ \begin{array}{c} \sigma' \in \Sigma : \delta_{CM_{ac}}((\sigma', \Gamma_i), y)! \\ \& \forall x \in \delta_{CM_{ac}}((\sigma', \Gamma_i), y),\ \lambda'_{ac}(x) = \sigma \end{array} \right\}.$$

$R_{o_p} = R''_p$ et $I_{o_p} = I''_p$, $\forall p \in P$.

Les deux propositions que nous allons donner montrent que le résultat de la procédure ci-dessus est un automate de Rabin qui représente le SÉD observé $\left(\widetilde{M}_C(L(G_C)), \widetilde{M}_C\right.$ et le langage de spécification $E_o = \widetilde{M}_C(N)$, N étant le plus grand ω-langage inclus dans E_C et normal par rapport à $\widetilde{M}_C$:

Proposition 5.2 *L'automate* $\mathcal{B}_o = (\Sigma_o \cup \{\epsilon\} \times C, Y_o, \delta_o, y_0, Y_o)$ *reconnaît le langage* $\widetilde{M}_C(L(G_C))$.

Démonstration:

Montrons d'abord que $L(\mathcal{B}_o) \subseteq \widetilde{M}_C(L(G_C))$*:*

Soit $m = (\sigma_1, \Gamma_1)...(\sigma_n, \Gamma_n) \in L(\mathcal{B}_o)$. *Alors on a* $\delta_o(m, y_0)!$, *ce qui implique* $\delta_o((\sigma_1, \Gamma_1), y_0)!$. *Alors il existe* y_i, $1 \leq i \leq n$, *tel que* $y_i \in \delta_o((\sigma_i, \Gamma_i), y_{i-1})$ *et* $\lambda'(y_i) = \sigma_i$.

Mais si $y_i \in \delta_o((\sigma_i, \Gamma_i), y_{i-1}) = \bigcup_{\sigma' \in \Sigma(\sigma_i, \Gamma_i, y_{i-1})} \delta_{CM_{ac}}((\sigma', \Gamma_i), y_{i-1})$, *alors il existe* $\sigma'_i \in \Sigma$ *telle que* $\delta_{CM_{ac}}((\sigma'_i, \Gamma_i), y_{i-1})$ *est définie et* $y_i \in \delta_{CM_{ac}}((\sigma'_i, \Gamma_i), y_{i-1})$.

Il existe donc une suite $\sigma'_1, \sigma'_2, ..., \sigma'_n$ *de symboles de* Σ *telle que :*

$y_i \in \delta_{MC}((\sigma'_i, \Gamma_i), y_{i-1})$, *pour* $1 \leq i \leq n$.

Ceci implique que $(\sigma', \Gamma_1)...(\sigma'_n, \Gamma_n) \in L((G_C \times \mathcal{A}_M)_{ac}) = L(G_C \times \mathcal{A}_M)$ *(voir chapitre 2). Par conséquent,*

$$\begin{aligned} & \delta_{CM}((\sigma'_1, \Gamma_1)...(\sigma'_n, \Gamma_n), (x_0, x_{0_M}))! \\ \Rightarrow\ & \delta_C((\sigma'_1, \Gamma_1)...(\sigma'_n, \Gamma_n), x_0)! \\ \Rightarrow\ & (\sigma'_1, \Gamma_1)...(\sigma'_n, \Gamma_n) \in L(G_C). \end{aligned}$$

D'autre part on pose $y_i = (x_i, x_{M_i})$ *pour tout* i, $1 \leq i \leq n$. *Alors* $(x_{M_0}, x_{M_1}, ..., x_{M_n})$ *est un chemin dans* $\mathcal{A}_M$ *qui correspond à* $(\sigma'_1, \Gamma_1)...(\sigma'_n, \Gamma_n)$ *puisque* $((x_0, x_{M_0}), ..., (x_n, x_{M_n}))$ *est un chemin dans* $G_C \times \mathcal{A}_M$ *qui correspond à la même séquence, et on a :*

$$\begin{aligned} M_C(\sigma', \Gamma_1)...(\sigma'_n, \Gamma_n)) &= (\lambda(x_{M_1}), \Gamma_1) ... (\lambda(x_{M_n}), \Gamma_n) \\ &= (\lambda'(x_1, x_{M_1}), \Gamma_1) ... (\lambda'(x_n, x_{M_n}), \Gamma_n) \\ &= (\lambda'_{ac}(x_1, x_{M_1}), \Gamma_1) ... (\lambda'_{ac}(x_n, x_{M_n}), \Gamma_n) \\ &= (\sigma_1, \Gamma_1)...(\sigma_n, \Gamma_n) \\ &= m \end{aligned}$$

Ceci implique que $m \in \widetilde{M}_C(L(G_C))$.

Montrons l'autre inclusion: soit $k \in L(G_C)$, *et montrons que* $\widetilde{M}_C(k) \in L(B_o)$.

Supposons que k s'écrit sous la forme $k = (\sigma_1, \Gamma_1)...(\sigma_n, \Gamma_n)$. Il existe donc $y_i \in Y_o = Y$, $1 \leq i \leq n$, tels que $y_i \in \delta_{CM}((\sigma_i, \Gamma_i), y_{i-1})$

$\Longrightarrow y_i \in \delta_o((\lambda'(y_i), \Gamma_i), y_{i-1})$

Alors le mot $(\lambda'(y_1), \Gamma_1)(\lambda'(y_2), \Gamma_2)...(\lambda'(y_n), \Gamma_n) \in L(\mathcal{B}_o)$. Mais ce mot n'est nul autre que $\widetilde{M}_C(k)$ (par la construction de $\mathcal{A}_C \times \mathcal{A}_M$).

◇

Remarque 5.2 *D'après la démonstration de la proposition précédente, il est possible de construire une correspondance*

$$Inv : ((\Sigma_o \cup \{\epsilon\}) \times C)^* \longrightarrow \Sigma \times C$$

qui à tout mot fini m de $L(G_o)$ associe une séquence m' de $L((\mathcal{A}_C \times \mathcal{A}_M)_{ac})$ telle que $\widetilde{M}_C(m') = m$, et pour tout $k \in \text{pre}(m')$, $\delta_{CM_{ac}}(k, y_0) \subseteq \delta_o\left(\widetilde{M}_C(k), y_0\right)$.

Corollaire 5.1 *L'automate de Büchi suivant:*

$$((\Sigma_o \cup \{\epsilon\}) \times C, Y_o, \delta_o, y_0, Y_o)$$

reconnaît $\widetilde{M}_C(S(G_C)) = \lim\left(\widetilde{M}_C(L(G_C))\right)$.

Proposition 5.3 *L'automate universel de Rabin*

$$\mathcal{A}_o = \left((\Sigma_o \cup \{\epsilon\}) \times C, Y_o, \delta_o, y_0, \left\{(R_{o_p}, I_{o_p}), p \in P\right\}\right)$$

reconnaît $\widetilde{M}_C(N)$, où N est le plus grand ω-langage normal par rapport à $\widetilde{M}_C$ et inclus dans $S(\mathcal{A}_C)$ (c.-à.-d. E_C).

Démonstration: *Il y a une correspondance injective entre les chemins (suites d'états) de $\mathcal{A}_o$ et ceux de $\mathcal{A}_C \times \mathcal{A}_M$-en effet, ce sont les mêmes chemins. Alors, si $s \in (\Sigma \times C)^\omega$,*

$s \in N$

$\Longleftrightarrow$ *les chemins correspondant à tous les mots qui "ressemblent" à s sont acceptants.*

$\Longleftrightarrow$ *tous les chemins dans $\mathcal{A}_o$ qui correspondent à $\widetilde{M}_C(s)$ sont acceptants.*

◇

On associe à $G_o = \left(\widetilde{M}_C(L(G_C)), \widetilde{M}_C(S(G_C))\right)$ la famille suivante d'entrées de commande

$$C_o = \{\Gamma_{o_i}, 1 \leq i \leq n\},$$

avec

$$\Gamma_{o_i} = \left\{ \begin{array}{c} (\sigma, \Gamma_i) : \sigma \in \Sigma_o \cup \{\epsilon\} \text{ et } \exists \sigma' \in \Gamma_i, x_M \in X_M \text{ tels que} \\ \delta_M((\sigma', \Gamma_i), x_M) \text{ est définie et } \lambda(\delta_M((\sigma', \Gamma_i), x_M)) = \sigma \end{array} \right\}$$

et

$$\Gamma_i \in C.$$

On écrit $\Gamma_{o_i} = \widetilde{M}_C(\Gamma_{C_i})$ où $\Gamma_{C_i} = \{(\sigma, \Gamma_i) : \sigma \in \Gamma_i\}$ et on dit que Γ_{o_i} est l'*image observée* de Γ_{C_i}.

Exemple 5.1 *On considère le SÉD G et la spécification E représentés par l'automate de Rabin suivant :*

$\mathcal{G} \times \mathcal{R}$:

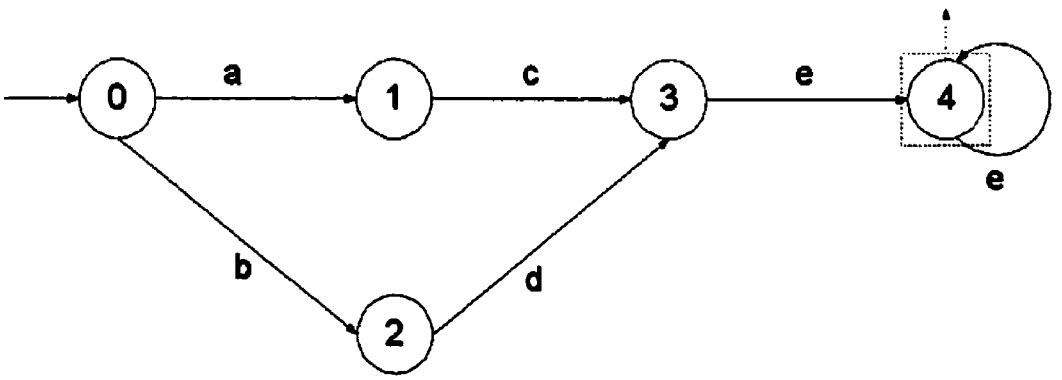


Figure 5.1: Automate de Rabin représentant le SÉD G et la spécification E

$\Sigma = \{a, b, c, d, e\}$ *et* $C = \{\Gamma_i,\ 1 \leq i \leq 7\}$

avec

$\Gamma_1 = \{a, b\}$

$\Gamma_2 = \{c, e\}$

$\Gamma_3 = \{d\}$

$\Gamma_4 = \{a, b, c, e\}$

$\Gamma_5 = \{a, b, d\}$

$\Gamma_6 = \{c, d, e\}$

$\Gamma_7 = \{a, b, c, d, e\}$

À partir de $\mathcal{G} \times \mathcal{R}$ on construit l'automate de Rabin $\mathcal{A}_C$ qui représente à la fois G_C et E_C :

$\mathcal{A}_C$:

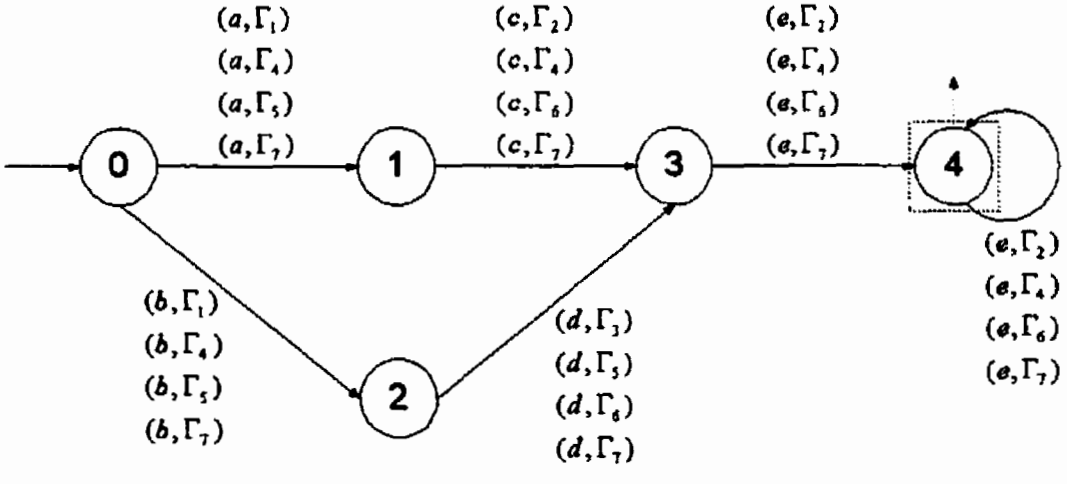


Figure 5.2: Automate de Rabin représentant G_C et E_C

La nouvelle famille d'entrées de commande est $C_C = \{\Gamma_{C_i}, 1 \leq i \leq 7\}$ avec

$$\Gamma_{C_1} = \{(a, \Gamma_1), (b, \Gamma_1)\}$$

$$\Gamma_{C_2} = \{(c, \Gamma_2), (e, \Gamma_2)\}$$

$$\Gamma_{C_3} = \{(d, \Gamma_3)\}$$

$$\Gamma_{C_4} = \{(a, \Gamma_4), (b, \Gamma_4), (c, \Gamma_4), (e, \Gamma_4)\}$$

$$\Gamma_{C_5} = \{(a, \Gamma_5), (b, \Gamma_5), (d, \Gamma_5)\}$$

$$\Gamma_{C_6} = \{(c, \Gamma_6), (d, \Gamma_6), (e, \Gamma_6)\}$$

$$\Gamma_{C_7} = \{(a, \Gamma_7), (b, \Gamma_7), (c, \Gamma_7), (d, \Gamma_7), (e, \Gamma_7)\}$$

On suppose que le masque d'observation de G est le suivant :

$$\begin{aligned} M : \quad a &\rightarrow \alpha \\ b &\rightarrow \alpha \\ ac &\rightarrow \alpha c \\ bd &\rightarrow \alpha d \\ ace &\rightarrow \alpha c \\ bde &\rightarrow \alpha d\tau \\ ac(e)^* &\rightarrow \alpha c \\ bd(e)^* &\rightarrow \alpha d(\tau)^* \end{aligned}$$

À partir de M on construit $\widetilde{M}_C$ représenté par l'automate de Moore suivant :

$\mathcal{A}_M$:

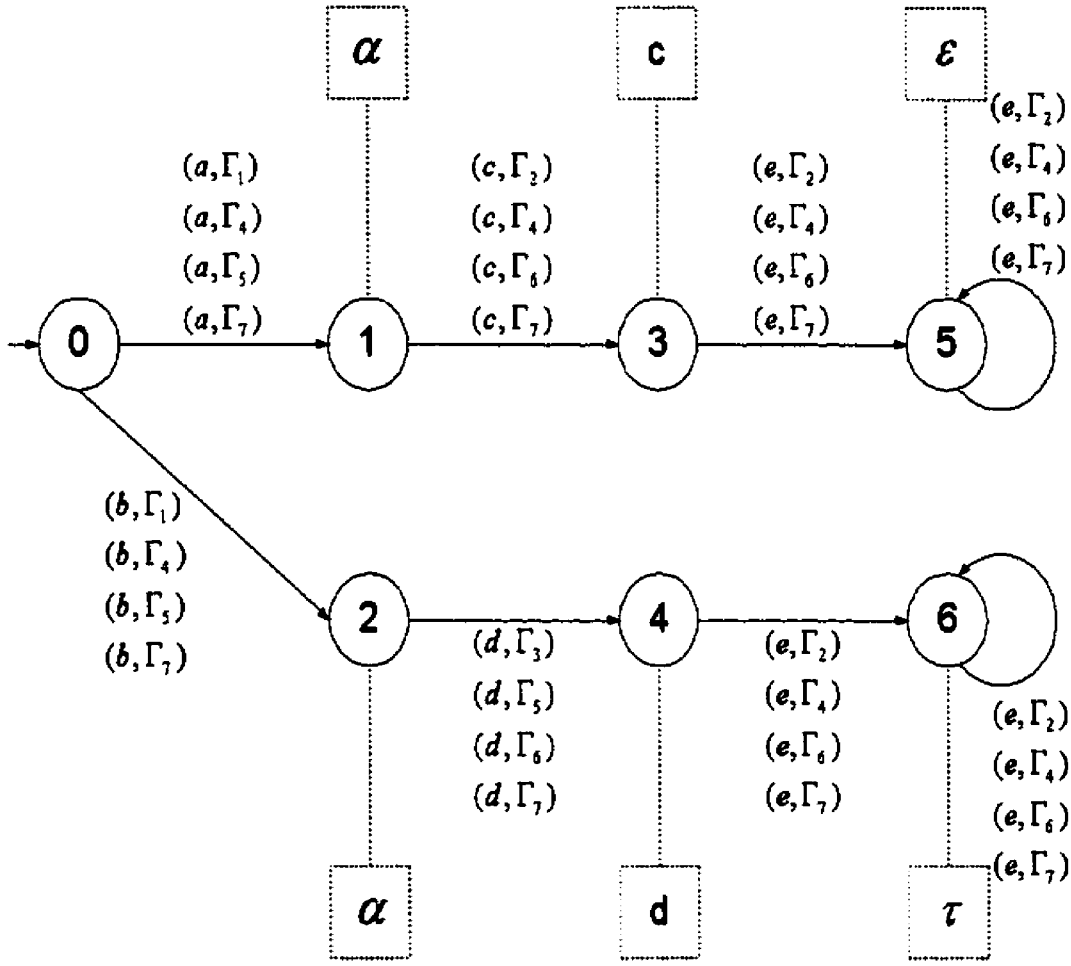


Figure 5.3: Automate de Moore représentant le masque $\widetilde{M}_C$

Si on applique l'algorithme donné dans cette section, on obtient comme résultat l'automate $\mathcal{A}_o$ de la figure 5.5.

$(\mathcal{A}_C \times \mathcal{A}_M)_{acc}$:

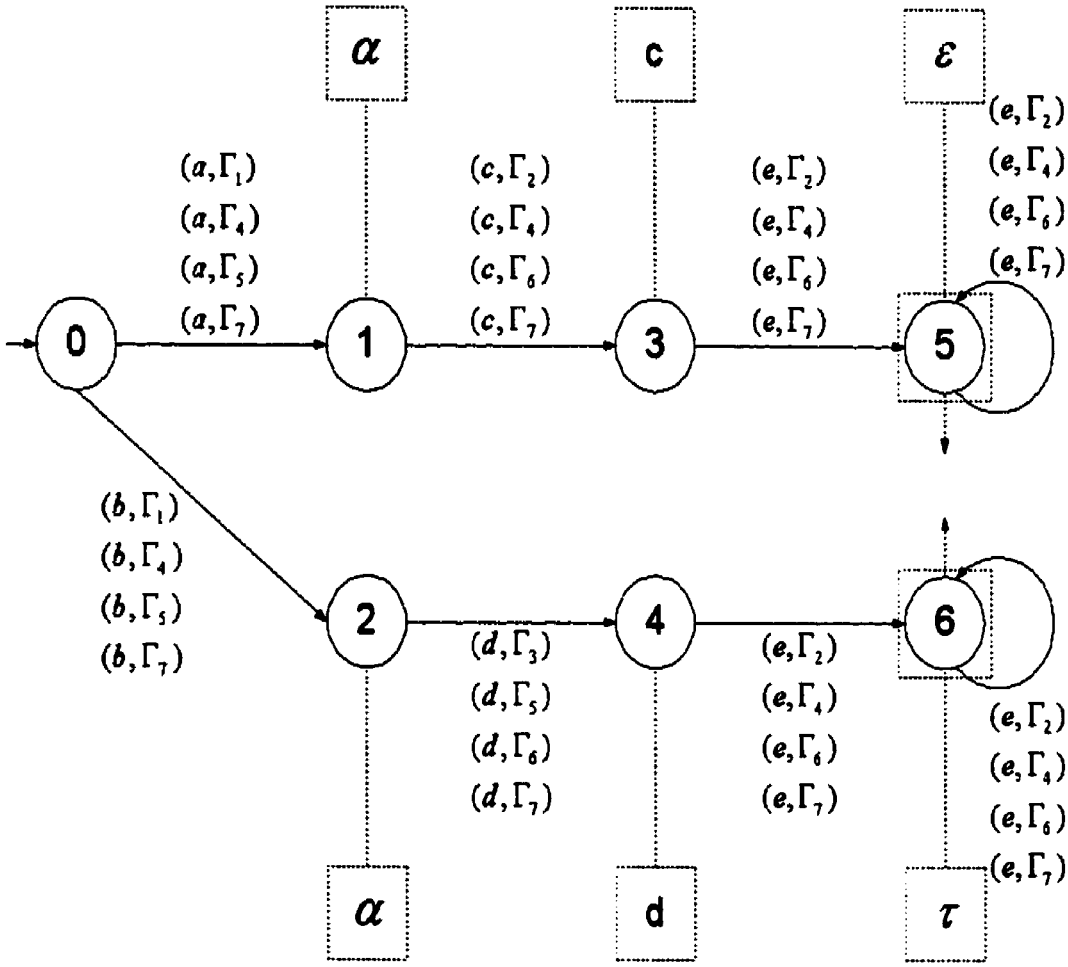


Figure 5.4: Automate "hybride" qui représente à la fois G_C, E_C et $\widetilde{M}_C$

$\mathcal{A}_o$:

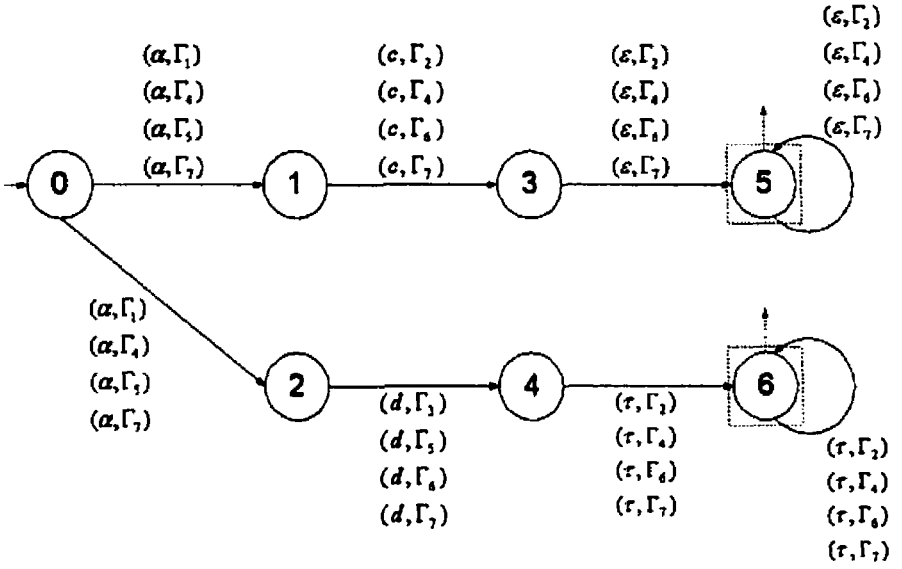


Figure 5.5: Automate universel de Rabin représentant le SÉD observé G_o et la spécification E_o

Comme nous l'avons déjà mentionné, la supervision de $\left(\mathcal{A}_C, \widetilde{M}_C\right)$ est équivalente à la supervision sous observations complètes de $\mathcal{A}_o$ (qui est elle même équivalente à la supervision de G_o sous observations complètes).

Définition 5.3 *Un superviseur sous observations complètes pour $\mathcal{A}_o$ est une fonction partielle $V_o : ((\Sigma_o \cup \{\epsilon\}) \times C)^* \longrightarrow C_o$ qui vérifie*

$$\pi_\epsilon(m_1) = \pi_\epsilon(m_2) \Longrightarrow V_o(m_1) = V_o(m_2)$$

Le langage généré en boucle fermée est défini comme suit :

(i) $1_{(\Sigma_o \cup \{\epsilon\}) \times C} \in L(V_o/\mathcal{A}_o)$

(ii) $\forall k \in ((\Sigma_o \cup \{\epsilon\}) \times C)^*$, *et* $\forall (\sigma, \Gamma) \in (\Sigma_o \cup \{\epsilon\}) \times C$ *on a :*

$$k(\sigma,\Gamma) \in L(V_o/\mathcal{A}_o) \;\Leftrightarrow\; k \in L(V_o/\mathcal{A}_o) \text{ et } k \in \mathrm{dom}(V_o)$$
$$(\sigma,\Gamma) \in V_o(k) \text{ et } k(\sigma,\Gamma) \in L(\mathcal{A}_o)$$

Le ω-langage généré en boucle fermée est

$$S(V_o/\mathcal{A}_o) = \lim(L(V_o/\mathcal{A}_o)) \cap S(\mathcal{A}_o)$$

On dit que V_o est complet si $L(V_o/\mathcal{A}_o) \subseteq \mathrm{dom}(V_o)$, et on dit qu'il est non-bloquant si $L(V_o/\mathcal{A}_o) = \mathrm{pre}(S(V_o/\mathcal{A}_o))$..

Proposition 5.4 *Soit $V_C : ((\Sigma_o \cup \{\epsilon\}) \times C)^* \longrightarrow C_C$ un superviseur sous observations partielles pour $\left(\mathcal{A}_C, \widetilde{M}_C\right)$ complet et non-bloquant. Alors,*

$$\begin{aligned} V_o : \; ((\Sigma_o \cup \{\epsilon\}) \times C)^* &\longrightarrow C_o \\ m &\longmapsto \widetilde{M}_C(V_C(m)) \end{aligned}$$

est un superviseur sous observations complètes pour $\mathcal{A}_o$, complet et non-bloquant, tel que:

$$L(V_o/\mathcal{A}_o) = \widetilde{M}_C(L(V_C/\mathcal{A}_C))$$

et

$$S(V_o/\mathcal{A}_o) = \widetilde{M}_C(S(V_C/\mathcal{A}_C))$$

Démonstration: *Il est clair que V_o est un superviseur sous observations complètes pour $\mathcal{A}_o$. En effet, pour tout $m_1, m_2 \in \mathrm{dom}(V_C)$, on a:*

$$\begin{aligned} \pi_\epsilon(m_1) = \pi_\epsilon(m_2) &\Rightarrow V_C(m_1) = V_C(m_2) \\ &\Rightarrow \widetilde{M}_C\left(V_C(m_1)\right) = \widetilde{M}_C\left(V_C(m_2)\right) \\ &\Rightarrow V_o(m_1) = V_o(m_2). \end{aligned}$$

Montrons maintenant que $L(V_o/\mathcal{A}_o) = \widetilde{M}_C\left(L(V_C/\mathcal{A}_C)\right)$. Pour cela nous allons montrer par récurrence sur la longueur l du mot m que:

$$m \in L(V_C/\mathcal{A}_C) \Leftrightarrow \widetilde{M}_C(m) \in L(V_o/\mathcal{A}_o)$$

Pour $l = 0$ le résultat est vrai, en effet $\widetilde{M}_C(1_{\Sigma \times C}) = 1_{(\Sigma_o \cup \{\epsilon\}) \times C} \in L(V_o/\mathcal{A}_o)$.

Supposons que la propriété soit vraie pour tout mot de longueur $\leq n-1$ et montrons qu'elle reste aussi vraie pour tout mot de longueur n.

$$\begin{aligned} m = k(\sigma, \Gamma) \in L(V_C/\mathcal{A}_C) &\Leftrightarrow k \in L(V_C/\mathcal{A}_C) \text{ et } \widetilde{M}_C(k) \in \mathrm{dom}(V_C) \text{ et} \\ &\quad (\sigma, \Gamma) \in V_C(\widetilde{M}_C(k)) \text{ et } k(\sigma, \Gamma) \in L(\mathcal{A}_C) \\ &\Leftrightarrow \widetilde{M}_C(k) \in L(V_o/\mathcal{A}_o) \text{ (d'après l'hypothèse de récurrence)} \\ &\quad \text{et } \widetilde{M}_C(k) \in \mathrm{dom}(V_o) \text{ et } \widetilde{M}_C(k(\sigma, \Gamma)) = \widetilde{M}_C(k)(a, \Gamma) \\ &\quad \in L(\mathcal{A}_o) \text{ et } (a, \Gamma) \in \widetilde{M}_C\left(V_C(\widetilde{M}_C(k))\right) = V_o(\widetilde{M}_C(k)) \\ &\Leftrightarrow \widetilde{M}_C(k)(a, \Gamma) \in L(V_o/\mathcal{A}_o) \\ &\Leftrightarrow \widetilde{M}_C(k(\sigma, \Gamma)) \in L(V_o/\mathcal{A}_o) \end{aligned}$$

D'autre part

$$\begin{aligned}\widetilde{M}_C\left(S(V_C/\mathcal{A}_C)\right) &= \lim\left(\widetilde{M}_C\left(\mathrm{pre}\left(S\left(V_C/\mathcal{A}_C\right)\right)\right)\right)\\ &= \lim\left(\widetilde{M}_C\left(L(V_C/\mathcal{A}_C)\right)\right)\\ &= \lim\left(L\left(V_o/\mathcal{A}_o\right)\right)\end{aligned}$$

Montrons que $S(V_C/\mathcal{A}_C)$ *est normal par rapport à* $\widetilde{M}_C$. *Pour cela il faut montrer que* $\widetilde{M}_C^{-1}\left(\widetilde{M}_C\left(L(V_C/\mathcal{A}_C)\right)\right) = L(V_C/\mathcal{A}_C)$, *puisque* $\mathrm{pre}(S(V_C/\mathcal{A}_C)) = L(V_C/\mathcal{A}_C)$. *On sait déjà que* $L(V_C/\mathcal{A}_C) \subseteq \widetilde{M}_C^{-1}\left(\widetilde{M}_C\left(L(V_C/\mathcal{A}_C)\right)\right)$, *il reste à montrer l'autre inclusion, en faisant un raisonnement par l'absurde.*

Supposons qu'il existe $m \in \widetilde{M}_C^{-1}\left(\widetilde{M}_C\left(L(V_C/\mathcal{A}_C)\right)\right)$, *tel que* $m \notin \mathrm{pre}\left(L(V_C/\mathcal{A}_C)\right)$. *et soit* $k_1 \leq m$ *le plus long préfixe de* m *tel que* $k_1 \in L(V_C/\mathcal{A}_C)$. *Alors il existe* $m' \in L(V_C/\mathcal{A}_C)$ *tel que* $\widetilde{M}_C(m) = \widetilde{M}_C(m')$.

Supposons que les deux mots s'écrivent comme suit : $m = k_1(\sigma_i, \Gamma_i)k_2$ *et* $m' = k_1'(\sigma_j, \Gamma_j)k_2'$ *avec* $|k_1| = |k_1'|$. *On a* $\widetilde{M}_C(m) = \widetilde{M}_C(m') \Rightarrow \Gamma_i = \Gamma_j$. *Or*

$$\begin{aligned}m' \in L(V_C/\mathcal{A}_C) &\Rightarrow k_1'(\sigma_j, \Gamma_j) \in L(V_C/\mathcal{A}_C)\\ &\Rightarrow k_1' \in L(V_C/\mathcal{A}_C) \text{ et } \widetilde{M}_C(k_1') \in \mathrm{dom}(V_C)\\ &\quad (\sigma_j, \Gamma_j) \in V_C(\widetilde{M}_C(k_1')) = \Gamma_{C_j} \text{ et}\\ &\quad k_1'(\sigma_j, \Gamma_j) \in L(\mathcal{A}_d)\end{aligned}$$

D'autre part $k_1 \in L(V_C/\mathcal{A}_C) \Rightarrow k_1 \in \mathrm{dom}(V_C)$ *(*V_C *est complet) et on a* $V_C\left(\widetilde{M}_C(k_1)\right) = V_C\left(\widetilde{M}_C(k_1')\right) = \Gamma_{C_j} \Rightarrow (\sigma_i, \Gamma_i) \in \Gamma_{C_j}$.

$\Rightarrow k_1(\sigma_i, \Gamma_i) \in L(V_C/\mathcal{A}_C)$: *Absurde.*

$\Rightarrow S(V_C/\mathcal{A}_C)$ *est normal par rapport à* $\widetilde{M}_C$.

$\Rightarrow S(V_C/\mathcal{A}_C) \subseteq N$

$\Rightarrow \widetilde{M}_C(S(V_C/\mathcal{A}_C)) \subseteq S(\mathcal{A}_o)$.

On obtient alors

$$\begin{aligned} S(V_o/\mathcal{A}_o) &= \lim(L(V_o/\mathcal{A}_o)) \cap S(\mathcal{A}_o) \\ &= \widetilde{M}_C(S(V_C/\mathcal{A}_C)) \cap S(\mathcal{A}_o) \\ &= \widetilde{M}_C(S(V_C/\mathcal{A}_C)) \end{aligned}$$

Montrons que V_o *est non-bloquant :*

$$\begin{aligned} L(V_o/\mathcal{A}_o) &= \widetilde{M}_C(L(V_C/\mathcal{A}_C)) \\ &= \widetilde{M}_C(\mathrm{pre}(S(V_C/\mathcal{A}_C))) \\ &= \mathrm{pre}\left(\widetilde{M}_C(S(V_C/\mathcal{A}_C))\right) \\ &= \mathrm{pre}(S(V_o/\mathcal{A}_o)) \end{aligned}$$

Montrons que V_o *est complet*

$$\begin{aligned} L(V_o/\mathcal{A}_o) &= \widetilde{M}_C(L(V_C/\mathcal{A}_C) \\ &\subseteq \mathrm{dom}(V_C) \ (V_C \text{ est complet}) \\ &\subseteq \mathrm{dom}(V_o) \ (\mathrm{dom}(V_o) = \mathrm{dom}(V_C)) \end{aligned}$$

◇

Proposition 5.5 *Soit* $V_o : ((\Sigma_o \cup \{\epsilon\}) \times C)^* \longrightarrow C_o$ *un superviseur sous observations complètes pour* $\mathcal{A}_o$. *Alors*

$$\begin{aligned} V_C : \ ((\Sigma_o \cup \{\epsilon\}) \times C)^* &\longrightarrow C_C \\ m &\mapsto \Gamma_{C_i} \end{aligned}$$

avec $\widetilde{M}_C(\Gamma_{C_i}) = V_o(m)$, *est un superviseur sous observations partielles pour* $\mathcal{A}_C$ *complet et non-bloquant tel que*

$$L(V_o/\mathcal{A}_o) = \widetilde{M}_C(L(V_C/\mathcal{A}_C))$$

et

$$S(V_o/\mathcal{A}_o) = \widetilde{M}_C(S(V_C/\mathcal{A}_C))$$

Démonstration:

Montrons d'abord que V_C *est bien défini:*

Soient $m_1, m_2 \in \mathrm{dom}(V_C)$ *tels que* $\pi_\epsilon(m_1) = \pi_\epsilon(m_2)$. *Alors d'après la définition de* V_o, *on a* $V_o(m_1) = V_o(m_2) \Rightarrow V_C(m_1) = V_C(m_2)$.

Montrons que $L(V_o/\mathcal{A}_o) = \widetilde{M}_C(L(V_C/\mathcal{A}_C))$. *Pour cela nous allons montrer par récurrence sur la longueur du mot que* $\forall m \in (\Sigma \times C)^*$, *on a*

$$m \in L(V_C/\mathcal{A}_C) \Leftrightarrow \widetilde{M}_C(m) \in L(V_o/\mathcal{A}_o)$$

Pour $|m| = 0$, le résultat est vrai. Supposons que le résultat est vrai pour tout mot de longueur $\leq n-1$ et montrons que ça reste aussi vrai pour tout mot de longueur n.

$$
\begin{aligned}
m = k(\sigma_n, \Gamma_n) \in L(V_C/\mathcal{A}_C) \quad \Rightarrow \quad & k \in L(V_C/\mathcal{A}_C) \text{ et } \widetilde{M}_C(k) \in \mathrm{dom}(V_C) \\
& \text{et } (\sigma_n, \Gamma_n) \in V_C\left(\widetilde{M}_C(k)\right) \text{ et} \\
& k(\sigma_n, \Gamma_n) \in L(\mathcal{A}_C) \\
\Rightarrow \quad & k' = \widetilde{M}_C(k) \in L(V_o/\mathcal{A}_o) \\
& \textit{(d'après l'hypothèse de récurrence)} \\
& \text{et } k' \in \mathrm{dom}(V_o) \text{ et } (a, \Gamma_n) \in V_o(k') \\
& \text{et } (a, \Gamma_n) \textit{ le dernier symbole de } \widetilde{M}_C(m) \\
& \text{et } \widetilde{M}_C(k(\sigma_n, \Gamma_n)) \in \widetilde{M}_C\left(L(\mathcal{A}_C)\right) = L\left(\mathcal{A}_o\right) \\
& k'(a, \Gamma_n) \in L(V_o/\mathcal{A}_o)
\end{aligned}
$$

On montre comme dans la preuve de la proposition 5.4 que $S(V_o/\mathcal{A}_o) = \widetilde{M}_C\left(S(V_C/\mathcal{A}_C)\right)$ et que V_o est complet et non-bloquant.

◇

5.5 L'ϵ-invariance

Définition 5.4 *Un ω-langage $K \subseteq ((\Sigma_o \cup \{\epsilon\}) \times C)^\omega$ est* ϵ-invariant *si* $\forall s \in \text{pre}(K)$ *de la forme* $s = (\sigma_1, \Gamma_1) \ldots (\sigma_n, \Gamma_n)(\epsilon, \Gamma_{n+1})$, *on a:*

$$s(\sigma, \Gamma_j) \notin \text{pre}(K),\ \forall \sigma \in \Sigma_o \cup \{\epsilon\} \text{ et } \forall \Gamma_j \in C \text{ tel que } \Gamma_j \neq \Gamma_{n+1}$$

L'automate de Rabin suivant

$$\mathcal{A}_i = ((\Sigma_o \cup \{\epsilon\}) \times C, X_i, \delta_i, x_{0_i}, (R_i, I_i))$$

Avec

$X_i = \{x_{0_i}, x_1, x_2, \ldots, x_n, x_p\}$, $n = |C| = |C_C| = |C_o|$.

$R_i = I_i = X_i \setminus \{x_p\}$.

$$\begin{aligned}
\delta_i :\ ((\Sigma_o \cup \{\epsilon\}) \times C) \times X_i &\longrightarrow X_i \\
(x_{0_i}, (\sigma, \Gamma_i)) &\longmapsto x_{0_i} \text{ avec } \sigma \in \Sigma_o \\
(x_{0_i}, (\epsilon, \Gamma_i)) &\longmapsto x_i \\
(x_i, (\epsilon, \Gamma_i)) &\longmapsto x_i \\
(x_i, (\sigma', \Gamma_j)) &\longmapsto x_p \text{ avec } \sigma' \in \Sigma_o \cup \{\epsilon\} \\
(x_i, (\sigma, \Gamma_i)) &\longmapsto x_{o_i} \text{ avec } \sigma \in \Sigma_o \\
(x_p, (\sigma, \Gamma_j)) &\longmapsto x_p \text{ avec } \sigma' \in \Sigma_o \cup \{\epsilon\} \text{ et } \Gamma_j \in C
\end{aligned}$$

reconnait le plus grand sous-langage ϵ-invariant de $((\Sigma_o \cup \{\epsilon\}) \times C)^\omega$.

Cette construction est illustrée à la figure 5.5 pour $C = \{\Gamma_1, \Gamma_2, \Gamma_3\}$.

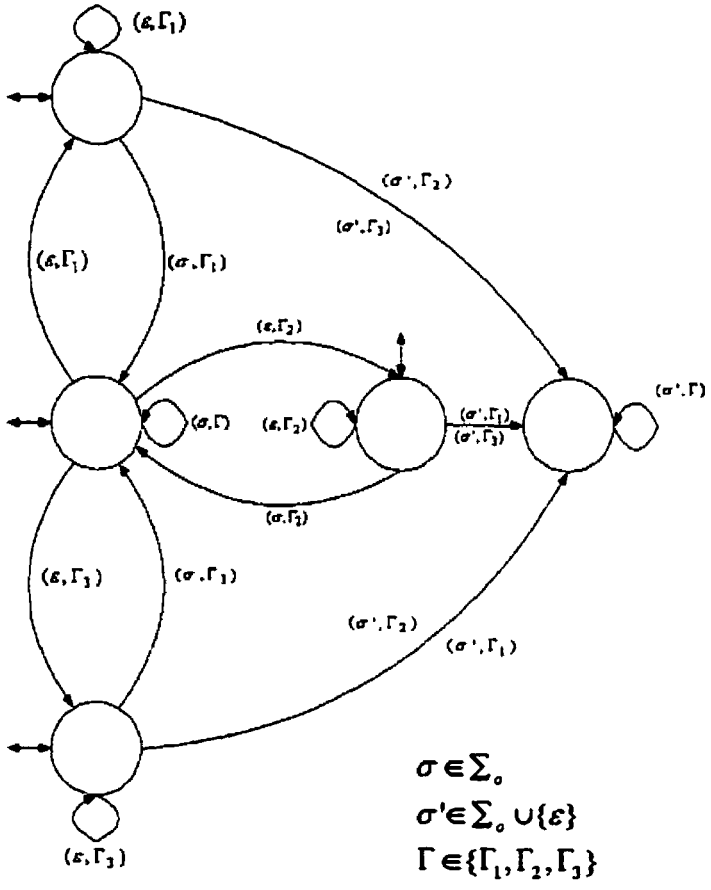


Figure 5.6: Exemple d'un automate qui vérifie la ϵ-invariance

À l'étape 4 on fait le produit entre l'automate $\mathcal{A}_o$ et l'automate $\mathcal{A}_i$. Le résultat de cette opération est l'automate $\mathcal{A}_{o_i}$ qui reconnaît le plus grand sous-langage de $S(\mathcal{A}_o)$ qui est ϵ-invariant. Évidemment, superviser $\mathcal{A}_o$ est équivalent à superviser $\mathcal{A}_{o_i}$ parce qu'un superviseur pour $\mathcal{A}_o$ synthétise en boucle fermée un ω-langage $S(V_o/\mathcal{A}_o)$ qui est nécessairement ϵ-invariant.

C'est pour cette raison qu'après l'étape 4, il suffit de contrôler $\mathcal{A}_{o_i}$ pour la satisfaction

de sa propre condition d'acceptation.

5.6 Pseudo-déterminisation de l'automate universel $\mathcal{A}_{o_i}$

Une fois l'automate $\mathcal{A}_{o_i}$ construit, nous chercherons à concevoir un superviseur qui contrôle cet automate pour la satisfaction de sa propre condition d'acceptation. D'après les travaux de Thistle et Wonham (voir chapitre 2), on sait qu'un outil de synthèse qui permet de concevoir un tel superviseur existe. Malheureusement ces résultats ne s'appliquent que pour le cas où l'automate qui représente le SÉD et la spécification est déterministe. Une étape fondamentale donc dans notre procédure de résolution du problème de supervision est la déterminisation de l'automate de Rabin universel $\mathcal{A}_{o_i}$, pour le transformer en un automate de Rabin déterministe. La construction que nous donnons s'insprire de l'algorithme de déterminisation pour les automates finis sur mots finis, et de l'algorithme de Safra[Saf88] pour la déterminisation d'un automate de Büchi non déterministe.

Mais le résultat de cette opération n'est pas un automate de Rabin déterministe qui reconnaît le même ω-langage que l'automate universel, mais plutôt un automate de Rabin déterministe qui reconnaît un ω-langage qui a la même projection par rapport à π_ϵ que le langage reconnu par l'automate universel. À chaque étape l'automate déterministe $\mathcal{A}_d$ calcule le sous ensemble des états de $\mathcal{A}_{o_i}$ qui sont atteignables par toutes les séquences finies qui ressemblent au préfixe qu'on vient de lire (dans le sens ou ils ont la même projection par rapport à π_ϵ).

En lisant un mot infini s on essaye d'identifier deux sortes de "breakpoint"s, des verts et des rouges, qui se produisent au niveau des états de $\mathcal{A}_d$. L'état initial est un breakpoint vert, le prochain breakpoint vert a lieu lorsque tous les éléments d'un état de $\mathcal{A}_d$, qui sont des états de $\mathcal{A}_{o_i}$, sont atteignables à partir des états de $\mathcal{A}_{o_i}$ du breakpoint vert précédent, que par des chemins qui passent par des états de R_{o_i}. Un breakpoint rouge a lieu dès qu'un élément d'un état de $\mathcal{A}_d$ est atteignable dans $\mathcal{A}_{o_i}$, à partir du breakpoint rouge précédent, sans passer par un état de I. Le mot est accepté si sa lecture par $\mathcal{A}_d$ donne lieu à un nombre infini de breakpoints verts et à un nombre fini de breakpoints rouges.

Remarque 5.3 *En fait tout ce qu'on fait en lisant un mot par $\mathcal{A}_d$ est de simuler sa lecture par $\mathcal{A}_{o_i}$.*

5.6.1 Algorithme

- Entrée :

  Un automate universel de Rabin avec une seule paire acceptante:

$$\mathcal{A}_{o_i} = ((\Sigma_o \cup \{\epsilon\}) \times C, Y_{o_i}, \delta_{o_i}, y_{0_i}, (R_{o_i}, I_{o_i}))$$

  On suppose qu'on peut associer à chaque élément de Y_o une couleur qui peut être rouge, vert ou blanc.

- Sortie :

Un automate de Rabin déterministe avec une seule paire acceptante:

$$\mathcal{A}_d = ((\Sigma_o \cup \{\epsilon\}) \times C, Z, \delta_d, z_0, (R_d, I_d))$$

Où

$Z = 2^{V_{o_i}} \times 2^{V_{o_i}}$,

$z_0 = (\{y_{0_i}\}, \emptyset)$,

$\delta_d : ((\Sigma_o \cup \{\epsilon\}) \times C) \times Z \longrightarrow Z$ est définie comme suit :

(i) $\forall z = (z^1, z^2)$, $\forall \Gamma_j \in C$, $\delta_d((\epsilon, \Gamma_j), z))!$ si $\exists y \in z^1$ tel que $\delta_{o_i}((\epsilon, \Gamma_j), y)!$ et on a

$$\delta_d((\epsilon, \Gamma_j), z)) = \left(\bigcup_{y \in z_\epsilon} \delta_{o_i}((\epsilon, \Gamma_j), y) \cup \left(z^1 \setminus z_\epsilon\right), z^2 \cup z_\epsilon \right)$$

avec

$$z_\epsilon = \{y \in z^1 : \delta_{o_i}((\epsilon, \Gamma_j), y)!\}$$

(ii) $\forall z = (z^1, z^2) \in Z$ et $\forall (\sigma, \Gamma_j) \in \Sigma_o \times C$, $\delta_d((\sigma, \Gamma_j), z))!$ si

$\forall y \in z^1$, $\delta_{o_i}((\epsilon, \Gamma_j), y)$ est non défini

ou

$\forall y \in z^1$ tel que $\delta_{o_i}((\epsilon, \Gamma_j), y)!$, on a

$$z'^1 \cup z'^2 = z^1 \cup z^2 \text{ (on dit que } z \text{ est maximal)}$$

avec

$$\left(z'^1, z'^2\right) := \delta_d\left((\epsilon, \Gamma_j), z\right)$$

et

$\exists y \in z^1$ tel que $\delta_{o_i}\left((\sigma, \Gamma_j), y\right)!$, auquel cas, on a

$$\delta\left((\sigma, \Gamma_j), z\right) = \left(\bigcup_{y \in z^1 \cup z^2} \delta_o\left((\sigma, \Gamma_j), y\right), \emptyset \right)$$

En plus, nous enrichissons l'ensemble des états de $\mathcal{A}_d$, en "colorant" les états de $\mathcal{A}_{o_i}$ en suivant les règles suivantes (on suppose que z' est le successeur de z pour $(\sigma, \Gamma) \in (\Sigma_o \cup \{\epsilon\}) \times C$, i.e. $z' = \delta_d\left((\sigma, \Gamma), z\right)$):

- Les éléments de z'^2 gardent les mêmes couleurs qu'ils avaient dans z (que ça soit dans z^1 ou z^2).

- Les éléments of z'^1 sont colorés de la façon suivante:

  (i) Si $\sigma \neq \epsilon$,

  - D'abord mettre tous les éléments à "blanc".

  - Si un élément de z'^1 est dans R_{o_i}, le mettre à "vert".

  - Si un élément de z'^1 a tous ses prédécesseurs qui sont "vert", le mettre lui aussi à "vert".

  - Si un élément de z'^1 n'est pas dans I_o, le mettre à "rouge".

  (ii) Si $\sigma = \bullet$

  - Les éléments $z^1 \setminus z_\epsilon$ gardent les mêmes couleurs qu'ils avaient dans z.

- Les éléments de

$$\bigcup_{y \in z_i} \delta_{o_i}((\epsilon, \Gamma_i), y)$$

sont colorés en suivant la règle du cas (i).

Comme nous l'avons déjà expliqué, en lisant une séquence infinie s, on essaye d'identifier des "breakpoints". L'état initial est un "breakpoint vert". Il est aussi un "breakpoint rouge" si $y_{o_i} \notin I_{o_i}$. Un "breakpoint rouge" a eu lieu dès qu'un état de $\mathcal{A}_{o_i}$ est coloré en "rouge", tandis qu'un "breakpoint vert" a eu lieu lorsque tous les éléments de l'état de $\mathcal{A}_d$ qu'on vient d'atteindre sont colorés en vert. Lorsqu'un "breakpoint" a eu lieu, on efface les couleurs des états de $\mathcal{A}_{o_i}$.

Intuitivement, un état de $\mathcal{A}_{o_i}$ est vert si tous les chemins qui mènent vers lui à partir du dernier "breakpoint vert", passent nécessairement par R_{o_i}. Il est marqué rouge s'il existe un chemin qui permet de l'atteindre à partir du dernier "breakpoint rouge", sans passer par I_{o_i}.

La séquence s est acceptée si l'automate déterministe passe par des "breakpoints verts" infiniment souvent et par des "breakpoints rouges" presque jamais (notons que c'est une condition de Rabin).

Les éléments de R_d sont les états de $\mathcal{A}_d$ qui correspondent à un "breakpoint vert" et ceux de $Z \setminus I_d$ sont ceux qui correspondent à un "breakpoint rouge".

5.6.2 Validation de l'algorithme

Lemme 5.1 *Soit $m \in L(\mathcal{A}_d)$, et posons $z = (z^1, z^2) = \delta_d(m, z_0)$. Alors, on a*

$$z^1 \cup z^2 \subseteq \bigcup_{k \in \pi_\epsilon^{-1}(\pi_\epsilon(m))} \delta_{o_i}(k, y_{o_i})$$

Démonstration:

Par récurrence sur la longueur du mot m.

Pour $m = 1_{(\Sigma_o \cup \{\epsilon\}) \times C}$ le résultat est vrai, en effet : $\delta_d(m, z_0) = z_0 = (\{y_{o_i}\}, \emptyset)$ et on a bien sûr $\{y_{o_i}\} \subseteq \bigcup_{k \in \pi_\epsilon^{-1}(\pi_\epsilon(m))} \delta_{o_i}(k, y_{o_i})$ puisque $\{y_{o_i}\} = \delta_{o_i}\left(1_{(\Sigma_o \cup \epsilon) \times C}, y_{o_i}\right) \subseteq \bigcup_{k \in \pi_\epsilon^{-1}(\pi_\epsilon(m))} \delta_{o_i}(k, y_{o_i})$.

Supposons que la propriété soit vraie pour tout mot de longueur $\leq n-1$ et montrons qu'elle reste vraie pour tout mot de longueur n.

Soit $m = (\sigma_1, \Gamma_1)(\sigma_2, \Gamma_2)...(\sigma_n, \Gamma_n)$, un mot de longueur n avec $\sigma_i \in (\Sigma_o \cup \{\epsilon\}) \times C$, pour tout $1 \leq i \leq n$. Posons $m_1 = (\sigma_1, \Gamma_1)(\sigma_2, \Gamma_2)...(\sigma_{n-1}, \Gamma_{n-1})$, alors on a $\delta_d\left(m_1(\sigma_n, \Gamma_n), z_0\right)! \Rightarrow z = \delta_d(m, z_0)!$ et $\delta_d\left((\sigma_n, \Gamma_n), z\right)!$.

Posons $z = (z^1, z^2)$, d'après l'hypothèse de récurrence $z^1 \cup z^2 \subseteq \bigcup_{k \in \pi_\epsilon^{-1}(\pi_\epsilon(m_1))} \delta_{o_i}(k, y_{o_i})$.

D'autre part, $\delta_d\left((\sigma_n, \Gamma_n), z\right)! \Rightarrow \exists\, x \in z^1 \cup z^2$ tel que $\delta_{o_i}\left((\sigma_n, \Gamma_n), x\right)!$.

$x \in z^1 \cup z^2 \subseteq \bigcup_{k \in \pi_\epsilon^{-1}(\pi_\epsilon(m_1))} \delta_{o_i}(k, y_{o_i}) \Rightarrow \exists k_x \in \pi_\epsilon^{-1}(\pi_\epsilon(m_1))$ tel que $x \in \delta_{o_i}(k_x, y_{o_i})$.

$\delta_{o_i}(k_x, y_{o_i})!$ *et* $x \in \delta_{o_i}(k_x, y_{o_i})$ *et* $\delta_{o_i}((\sigma_n, \Gamma_n), x) \Rightarrow \delta_{o_i}(k_x(\sigma_n, \Gamma_n), y_{o_i})!$

et $\delta_{o_i}((\sigma_n, \Gamma_n), x) \subseteq \delta_{o_i}(k_x(\sigma_n, \Gamma_n), y_{o_i})$

Avec

$$\begin{aligned}
\pi_\epsilon(k_x(\sigma_n, \Gamma_n)) &= \pi_\epsilon(k_x)\pi_\epsilon((\sigma_n, \Gamma_n)) \\
&= \pi_\epsilon(m_1)\pi_\epsilon((\sigma_n, \Gamma_n)) \\
&= \pi_\epsilon(m_1(\sigma_n, \Gamma_n)) \\
&= \pi_\epsilon(m)
\end{aligned}$$

◇

Proposition 5.6 *Soit $\mathcal{A}_{o_i}$ un automate universel de Rabin dont la condition d'acceptation contient une seule paire, et soit $\mathcal{A}_d$ l'automate de Rabin obtenu en appliquant l'algorithme décrit dans la section précédente à $\mathcal{A}_{o_i}$. Alors on a ,*

$$\pi_\epsilon(L(\mathcal{A}_{o_i})) = \pi_\epsilon(L(\mathcal{A}_d))$$

et

$$\pi_\epsilon(S(\mathcal{A}_{o_i})) = \pi_\epsilon(S(\mathcal{A}_d))$$

Démonstration:

Soit $m \in L(\mathcal{A}_{o_i})$, montrons, par récurrence sur la longueur de m que pour tout $y \in \delta_{o_i}(m, y_{o_i})$, il existe $m' \in L(\mathcal{A}_d)$, tel que $\pi_\epsilon(m) = \pi_\epsilon(m')$

et $y \in z^1 \cup z^2$ *ou* $z = (z^1, z^2) = \delta_d(m', z_0)$.

Pour $|m| = 0$ *le résultat est vrai, en effet pour* $m = 1_{(\Sigma_o \cup \{\epsilon\}) \times C}$, *on associe* $m' = 1_{(\Sigma_o \cup \{\epsilon\}) \times C}$. *Supposons que la propriété soit vraie pour tout mot de longueur* $\leq n-1$ *et montrons qu'elle reste aussi vraie pour tout mot de longueur* $\leq n$.

Soit $m = k(\sigma, \Gamma_n) \in L(\mathcal{A}_{o_i})$ *et soit* $y \in \delta_{o_i}(m, y_{0_i})$, *alors on a* $k \in L(\mathcal{A}_{o_i})$ *et* $\exists x \in \delta_{o_i}(k, y_{o_i})$ *tel que* $\delta_{o_i}((\sigma, \Gamma_n), x)!$ *et* $y \in \delta_{o_i}((\sigma, \Gamma_n), x)$.

$k \in L(\mathcal{A}_{o_i})$ *et* $\exists\ x \in \delta_{o_i}(k, y_{o_i}) \Rightarrow \exists k' \in L(\mathcal{A}_d)$ *tel que* $\pi_\epsilon(k) = \pi_\epsilon(k')$ *et* $x \in z^1 \cup z^2$ *avec* $z = \delta_d(k', z_0)$ *(d'après l'hypothèse de récurrence). À ce niveau nous devons faire la distinction entre deux cas:* ***Cas 1 :*** $\sigma \neq \epsilon$

Dans ce cas on a ou bien $\delta_d((\sigma, \Gamma_n), z)!$ *et alors il suffit de prendre* $m' = k'(\sigma, \Gamma_n)$.

Ou bien $\delta_d((\sigma, \Gamma_n), z)$ *n'est pas défini, alors il existe nécessairement* $p \geq 1$ *tel que* $\delta_d(\underbrace{(\epsilon, \Gamma_n)(\epsilon, \Gamma_n)...(\epsilon, \Gamma_n)}_{p\,fois}(\sigma, \Gamma_n), z)!$ *et dans ce cas on a* $m' = k'(\epsilon, \Gamma_n)(\epsilon, \Gamma_n)...(\epsilon, \Gamma_n)(\sigma, \Gamma_n) \in L(\mathcal{A}_d)$ *et on a* $y \in z_1^1 \cup z_1^2$, *où* $z_1 = (z_1^1, z_1^2) = \delta_d(m', z_0)$.

Cas 2 : $\sigma = \epsilon$

Dans ce cas on a $x \in z^1$, *ce qui implique que* $\delta_d((\epsilon, \Gamma_n), z)!$, *et il suffit de prendre* $m' = k'(\epsilon, \Gamma_n)$.

$\Longrightarrow \pi_\epsilon(L(\mathcal{A}_{o_i})) \subseteq \pi_\epsilon(L(\mathcal{A}_d))$.

Montrons l'autre inclusion :

Pour cela nous allons montrer par récurrence que pour tout $m \in L(\mathcal{A}_d)$, *il existe* $m' \in L(\mathcal{A}_{o_i})$ *tel que* $\pi_\epsilon(m) = \pi_\epsilon(m')$.

Pour $m = 1_{(\Sigma_o \cup \{\epsilon\}) \times C}$ *le résultat est vrai puisqu'il suffit de prendre* $m' = 1_{(\Sigma_o \cup \{\epsilon\}) \times C}$.

Supposons que la propriété soit vraie pour tout mot de longueur $\leq n-1$ *et montrons qu'elle reste vraie pour tout mot de longueur* n.

Supposons que m *s'écrit sous la forme* $m = (\sigma_1, \Gamma_1)(\sigma_2, \Gamma_2)...(\sigma_n, \Gamma_n)$ *et posons* $m_1 = (\sigma_1, \Gamma_1)(\sigma_2, \Gamma_2)...(\sigma_{n-1}, \Gamma_{n-1})$, *alors on a :*

$$\begin{aligned} m \in L(\mathcal{A}_d) &\Rightarrow m_1 \in L(\mathcal{A}_d) \\ &\Rightarrow \exists m'_1 \in L(\mathcal{A}_{o_i}) \text{ tel que } \pi_\epsilon(m'_1) = \pi_\epsilon(m_1) \\ &\quad \textit{(d'après l'hypothèse de récurrence)} \end{aligned}$$

Deux cas sont à étudier :

cas 1 : $\sigma_n = \epsilon$

Alors il suffit de prendre $m' = m'_1$ *et on a dans ce cas* $m' \in L(\mathcal{A}_{o_i})$, *et*

$$\begin{aligned} \pi_\epsilon(m') &= \pi_\epsilon(m'_1) \\ &= \pi_\epsilon(m_1) \\ &= \pi_\epsilon(m_1(\epsilon, \Gamma_n)) \\ &= \pi_\epsilon(m) \end{aligned}$$

cas 2 : $\sigma_n \neq \epsilon$

On a

$$\begin{aligned} m_1(\sigma_n, \Gamma_n) \in L(\mathcal{A}_d) &\Rightarrow \delta_d(m_1(\sigma_n, \Gamma_n), z_0)! \\ &\Rightarrow \delta_d(m_1, z_0)! \text{ et } \delta_d((\sigma_n, \Gamma_n), \delta_d(m_1, z_0))! \end{aligned}$$

Posons $z = (z^1, z^2) = \delta_d(m_1, z_0)$, *alors on a :*

$\delta_d((\sigma_n, \Gamma_n), z)! \Rightarrow \exists x \in z^1 \cup z^2$ *tel que* $\delta_{o_i}((\sigma_n, \Gamma_n), x)!$.

$x \in z^1 \cup z^2 \subseteq \bigcup_{k \in \pi_\epsilon^{-1}(\pi_\epsilon(m_1))} \delta_{o_i}(k, y_{o_i}) \Rightarrow \exists k \in \pi_\epsilon^{-1}(\pi_\epsilon(m_1))$ *tel que* $x \in \delta_{o_i}(k, y_{o_i})$.

$\delta_{o_i}((\sigma_n, \Gamma_n), x)!$ *et* $x \in \delta_{o_i}(k, y_{o_i}) \Rightarrow \delta_{o_i}(k(\sigma_n, \Gamma_n), y_{o_i})!$

$\Rightarrow k(\sigma_n, \Gamma_n) \in L(\mathcal{A}_{o_i})$ *et on a*

$$\begin{aligned}
\pi_\epsilon(k(\sigma_n, \Gamma_n)) &= \pi_\epsilon(k)(\sigma_n, \Gamma_n) \\
&= \pi_\epsilon(m_1)(\sigma_n, \Gamma_n) \\
&= \pi_\epsilon(m_1(\sigma_n, \Gamma_n)) \\
&= \pi_\epsilon(m)
\end{aligned}$$

$\Longrightarrow \pi_\epsilon(L(\mathcal{A}_d)) \subseteq \pi_\epsilon(L(\mathcal{A}_{o_i}))$

$\Longrightarrow \pi_\epsilon(L(\mathcal{A}_d)) = \pi_\epsilon(L(\mathcal{A}_{o_i}))$.

Montrons maintenant que $\pi_\epsilon(S(\mathcal{A}_d)) = \pi_\epsilon(S(\mathcal{A}_{o_i}))$*:*

$s \in S(\mathcal{A}_d)$ *ssi* $\exists \pi : \mathrm{pre}(s) \rightarrow z$ *un chemin acceptant dans* $\mathcal{A}_d$.

Posons $T = \{k \in L(\mathcal{A}_{o_i})$ *tel que* $\exists k' \in \mathrm{pre}(s)$ *tel que* $\pi_\epsilon(k) = \pi_\epsilon(k')\}$.

D'après le lemme de König, $\lim(T) \neq \emptyset$. *Soit* $s' \in \lim(T)$, *alors d'après la construction de* $\mathcal{A}_d$, s' *est nécessairement dans* $S(\mathcal{A}_{o_i})$ *et on a* $\pi_\epsilon(s') = \pi_\epsilon(s)$.

$\Rightarrow \pi_\epsilon(S(\mathcal{A}_d)) \subseteq \pi_\epsilon(S(\mathcal{A}_{o_i}))$.

Un raisonnement analogue permet de montrer que $\pi_\epsilon(S(\mathcal{A}_{o_i})) \subseteq \pi_\epsilon(S(\mathcal{A}_d))$.

$\Longrightarrow \pi_\epsilon(S(\mathcal{A}_d)) = \pi_\epsilon(S(\mathcal{A}_{o_i}))$.

$\diamond$

Proposition 5.7 *Soit* $f_d : ((\Sigma_o \cup \{\epsilon\}) \times C)^* \longrightarrow C_o$ *un superviseur pour* $\mathcal{A}_d$, *complet et non-bloquant. On définit*

$$\begin{aligned} f_{o_i} : \ ((\Sigma_o \cup \{\epsilon\}) \times C)^* &\longrightarrow C_o \\ m &\longmapsto f_d(m') \end{aligned}$$

avec $m' \in L(f_d/\mathcal{A}_d)$ *tel que* $\pi_\epsilon(m') = \pi_\epsilon(m)$.

Alors f_{o_i} *est un superviseur complet et non-bloquant pour* $\mathcal{A}_{o_i}$ *tel que*

$$\pi_\epsilon(L(f_{o_i}/\mathcal{A}_{o_i})) = \pi_\epsilon(L(f_d/\mathcal{A}_d))$$

et

$$\pi_\epsilon(S(f_{o_i}/\mathcal{A}_{o_i})) = \pi_\epsilon(S(f_d/\mathcal{A}_d))$$

Démonstration:

Commençons par montrer que f_{o_i} *est une fonction:*

Soient $m'_1, m'_2 \in L(f_d/\mathcal{A}_d)$ *tel que* $\pi_\epsilon(m'_1) = \pi_\epsilon(m'_2) = m$, *montrons que* $f_d(m'_1) = f_d(m'_2)$.

cas 1 : $m = 1_{(\Sigma_o \cup \{\epsilon\}) \times C}$

Dans ce cas on a $m'_1 = \underbrace{(\epsilon, \Gamma_1)(\epsilon, \Gamma_1)...(\epsilon, \Gamma_1)}_{n_1 \text{ fois}}$ *et* $m'_2 = \underbrace{(\epsilon, \Gamma_1)(\epsilon, \Gamma_1)...(\epsilon, \Gamma_1)}_{n_2 \text{ fois}}$.

Posons $\Gamma_{o_2} = f_d(m'_1)$ *et* $\Gamma_{o_3} = f_d(m'_2)$.

Alors, comme f_d *est non-bloquant, on est sur qu'il existe* $\sigma_1 \in \Sigma_o \cup \{\epsilon\}$ *tel*

que $m'_1(\sigma_1, \Gamma_2) \in L(f_d/\mathcal{A}_d)$ *et il existe* $\sigma_2 \in \Sigma_o \cup \{\epsilon\}$ *tel que* $m'_2(\sigma_2, \Gamma_3) \in L(f_d/\mathcal{A}_d)$. *Or* $L(f_d/\mathcal{A}_d) \subseteq L(\mathcal{A}_d)$ *qui est* ϵ*-invariant*

$\Longrightarrow \Gamma_3 = \Gamma_2 = \Gamma_1$

$\Longrightarrow f_d(m'_1) = f_d(m'_2)$.

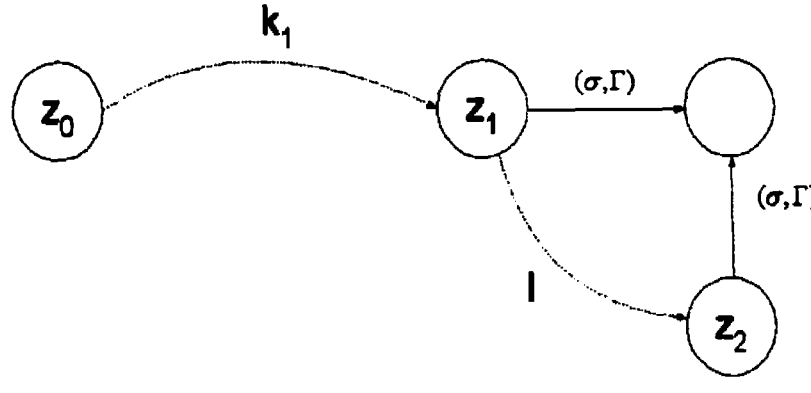
cas 2 : $m \neq 1_{(\Sigma_o \cup \{\epsilon\}) \times C}$

Soit (σ, Γ) *le dernier symbole de* m, *c.-à.-d. le dernier symbole observable de* m'_1 *et* m'_2. *Dans ce cas* m'_1 *et* m'_2 *s'écrivent sous la forme* $m'_1 = \underbrace{k_1(\sigma, \Gamma)}_{m''_1} k'_1$ *avec* $\pi_\epsilon(k'_1) = 1_{(\Sigma \cup \{\epsilon\}) \times C}$ *et* $m'_2 = \underbrace{k_2(\sigma, \Gamma)}_{m''_2} k'_2$ *avec* $\pi_\epsilon(k'_2) = 1_{(\Sigma \cup \{\epsilon\}) \times C}$.

Notons que $\pi_\epsilon(m''_1) = \pi_\epsilon(m''_2) = m$.

Nous allons montrer par récurrence sur la longueur de m *que* $\delta_d(m''_1, z_0) = \delta_d(m''_2, z_0)$.

Pour $|m| = 1$, *le résultat est vrai, en effet*



On suppose, sans perte de généralité, que $|k_1| \leq |k_2|$, $\Rightarrow k_2 = k_1 l$ *avec* $l = (\epsilon, \Gamma)...(\epsilon, \Gamma)$.

Appelons $z_1 = (z_1^1, z_1^2) = \delta_d(k, z_o)$. *Alors comme* $\delta((\epsilon, \Gamma), z)!$ *et* $\delta_d((\sigma, \Gamma), z)!$ *on a nécessairement* z_1 *est maximal. Appelons* $z_2 = (z_2^1, z_2^2) = \delta_d(l, z_1)$

$\Rightarrow z_2^1 \cup z_2^2 = z_1^1 \cup z_1^2$

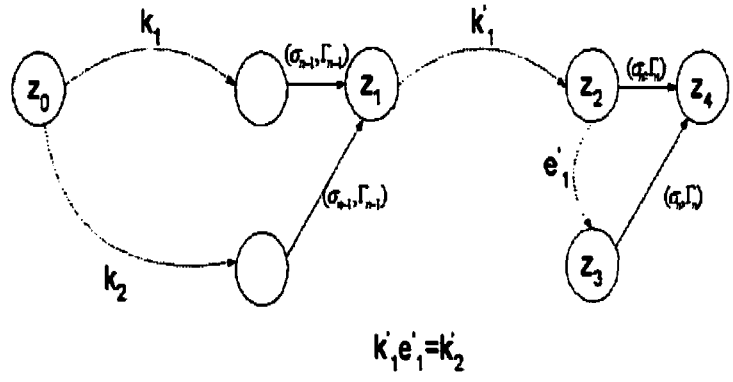
$\Rightarrow \delta_d((\sigma, \Gamma), z_1) = \delta_d((\sigma, \Gamma), z_2)$

$\Rightarrow \delta_d(m_1'', z_0) = \delta_d(m_2'', z_0)$.

Supposons que le résultat soit vrai pour tout mot de longueur $\leq n-1$ *et montrons que ça reste aussi vrai pour tout mot de longueur* n *(avec* $n \geq 2$*).*

Supposons que m *s'écrit sous la forme* $m = (\sigma_1, \Gamma_1)(\sigma_2, \Gamma_2)...(\sigma_n, \Gamma_n)$

Supposons aussi que m_1'' *et* m_2'' *s'écrivent sous la forme* $m_1'' = k_1(\sigma_{n-1}, \Gamma_{n-1})$ $k_1'(\sigma_n, \Gamma_n)$ *et* $m_2'' = k_2(\sigma_{n-1}, \Gamma_{n-1})k_2'(\sigma_n, \Gamma_n)$ *On a*



$$\begin{aligned} \pi_\epsilon(k_1(\sigma_{n-1}, \Gamma_{n-1})) &= \pi_\epsilon(k_2(\sigma_{n-1}, \Gamma_{n-1})) \\ &= (\sigma_1, \Gamma_1)...(\sigma_{n-1}, \Gamma_{n-1}) \end{aligned}$$

D'après l'hypothèse de récurrence, on a :

$$\begin{aligned}\delta_d(k_1(\sigma_{n-1},\Gamma_{n-1}),z_0) &= \delta_d(k_2(\sigma_{n-1},\Gamma_{n-1}),z_0)\\ &= z_1\end{aligned}$$

Posons $z_2 = \delta_d(k'_1, z_1)$ *et* $z_3 = \delta_d(k'_2, z_1)$. *Si on suppose que* $|k'_1| \leq |k'_2|$, *alors on a nécessairement* $k'_1 \leq k'_2$

$\Rightarrow z_2^1 \cup z_2^2 = z_3^1 \cup z_3^2$

$\Rightarrow \delta_d((\sigma_n,\Gamma_n),z_2) = \delta_d((\sigma_n,\Gamma_n),z_3)$

$\Rightarrow \delta_d(m''_1, z_0) = \delta_d(m''_2, z_0)$

$\Rightarrow f_d(m''_1) = f_d(m''_2)$ *(puisqu'on se restreint à la classe des superviseurs par retour d'états).*

Montrons maintenant que f_{o_i} *définit un superviseur sous observations complètes pour* $\mathcal{A}_{o_i}$.

Ceci revient à montrer que pour tout $m_1, m_2 \in \mathrm{dom}(f_{o_i})$, *on a la propriété suivante:*

$$\pi_\epsilon(m_1) = \pi_\epsilon(m_2) \Longrightarrow f_{o_i}(m_1) = f_{o_i}(m_2)$$

Soient $m'_1, m'_2 \in L(f_d/\mathcal{A}_d)$ *tels que* $\pi_\epsilon(m'_1) = \pi_\epsilon(m_1)$ *et* $\pi_\epsilon(m'_2) = \pi_\epsilon(m_2)$. *Alors on a* $\pi_\epsilon(m'_1) = \pi_\epsilon(m'_2)$, *ce qui implique que* $f_d(m'_1) =$

$f_d(m_2')$.

$\Rightarrow f_{o_i}(m_1) = f_{o_i}(m_2)$.

Montrons que $\pi_\epsilon(L(f_d/\mathcal{A}_d)) = \pi_\epsilon(L(f_{o_i}/\mathcal{A}_{o_i}))$.

Commençons par montrer que

$$\pi_\epsilon(L(f_d/\mathcal{A}_d)) \subseteq \pi_\epsilon(L(f_{o_i}/\mathcal{A}_{o_i}))$$

c.-à.-d. montrons que $\forall m \in L(f_d/\mathcal{A}_d)$, *il existe* $m' \in L(f_{o_i}/\mathcal{A}_{o_i})$ *tel que* $\pi_\epsilon(m) = \pi_\epsilon(m')$.

Démonstration par récurrence sur la longueur du mot m.

$|m| = 0$, *le résultat est vrai, en effet à* $m = 1_{(\Sigma_o \cup \{\epsilon\}) \times C}$, *on associe* $m' = 1_{(\Sigma_o \cup \{\epsilon\}) \times C}$.

Supposons que le résultat soit vrai pour tout mot de longueur $\leq n-1$ *et montrons que ça reste aussi vrai pour tout mot de longueur* $\leq n$.

$$m = k(\sigma_n, \Gamma_n) \in L(f_d/\mathcal{A}_d) \quad \Leftrightarrow \quad k \in L(f_d/\mathcal{A}_d) \text{ et } k \in \mathrm{dom}(f_d) \text{ et } (\sigma_n, \Gamma_n) \in f_d(k) \text{ et } k(\sigma_n, \Gamma_n) \in L(\mathcal{A}_d)$$

D'après l'hypothèse de récurrence, $k \in L(f_d/\mathcal{A}_d) \Rightarrow \exists k' \in L(f_{o_i}/\mathcal{A}_{o_i})$ *tel que* $\pi_\epsilon(k') = \pi_\epsilon(k)$.

Si $\sigma = \epsilon$, *alors il suffit de prendre* $m' = k'$. *Sinon, on a* $k \in \mathrm{dom}(f_d) \Rightarrow$

$k' \in \mathrm{dom}(f_{o_i})$. *On a aussi* $(\sigma_n, \Gamma_n) \in f_d(k) \Rightarrow (\sigma_n, \Gamma_n) \in f_{o_i}(k')$.

D'autre part $k(\sigma_n, \Gamma_n) \in L(\mathcal{A}_d) \Rightarrow \exists k'' \in L(\mathcal{A}_{o_i})$ *et* $\exists y \in \delta_{o_i}(k'', y_{0_i})$ *tel que* $\delta_{o_i}((\sigma_n, \Gamma_n), y)!$.

Montrons alors qu'on a nécessairement $k'' \in L(f_{o_i}/\mathcal{A}_{o_i})$.

Raisonnement par l'absurde : supposons que $k'' \notin L(f_{o_i}/\mathcal{A}_{o_i})$, *et soit* $p_1'' \leq k''$ *le plus long préfixe de* k'' *tel que* $p_1'' \in L(f_{o_i}/\mathcal{A}_{o_i})$.

Supposons que k'' *s'écrit sous la forme* $k'' = p_1''(\sigma_i, \Gamma_i)p_2''$.

D'autre part $\pi_\epsilon(k'') = \pi_\epsilon(k') \Rightarrow \exists p_1' \leq k'$ *tel que* $\pi_\epsilon(p_1') = \pi_\epsilon(p_1'')$.

$\Rightarrow p_1'' \in \mathrm{dom}(f_{o_i})$ *et* $f_{o_i}(p_1'') = f_{o_i}(p_1')$.

Donc $p''(\sigma_i, \Gamma_i) \notin L(f_{o_i}/\mathcal{A}_{o_i}) \Rightarrow (\sigma_i, \Gamma_i) \notin f_{o_i}(p_1'')$.

Deux cas sont à discuter :

cas 1 : $\sigma_i = \epsilon$

Supposons que $k' = p_1'p_2'$, *et soit* (σ_j, Γ_j) *le premier symbole observable de* p_2' *et posons* $\Gamma_{o_j} = f_{o_i}(p_1')$.

$\Rightarrow p_2'' = (\epsilon, \Gamma_i)(\epsilon, \Gamma_i)...(\epsilon, \Gamma_i)(\sigma_j, \Gamma_j)p_3''$

$\Rightarrow k''$ *correspond à un chemin où on n'a pas la propriété d'*ϵ*-invariance; absurde.*

cas 2 : $\sigma_i \neq \epsilon$

Dans ce cas on a nécessairement $\pi_\epsilon(k'') \neq \pi_\epsilon(k)$; *absurde aussi.*

Il est évident aussi que $k'' \in \mathrm{dom}(f_{o_i})$ *et que* $f_{o_i}(k'') = \Gamma_{o_n}$.

$\Rightarrow k''(\sigma_n, \Gamma_n) \in L(f_{o_i}/\mathcal{A}_{o_i})$

$\Rightarrow \pi_\epsilon(L(f_d/\mathcal{A}_d)) \subseteq \pi_\epsilon(L(f_{o_i}/\mathcal{A}_{o_i}))$.

Montrons l'autre inclusion :

Nous allons montrer que pour tout $m \in L(f_{o_i}/\mathcal{A}_{o_i})$, *il existe* $m' \in L(f_d/\mathcal{A}_d)$ *tel que* $\pi_\epsilon(m) = \pi_\epsilon(m')$. *Par récurrence sur la longueur de* m*:*

Pour $m = 1_{(\Sigma_o \cup \{\epsilon\}) \times C}$ *le résultat est vrai.*

Supposons que ça soit vrai pour tout mot de longueur $\leq n-1$ *et montrons que c'est aussi vrai pour tout mot de longueur* n.

$$m = k(\sigma_n, \Gamma_n) \in L(f_{o_i}/\mathcal{A}_{o_i}) \Leftrightarrow k \in L(f_{o_i}/\mathcal{A}_{o_i}) \text{ et } k \in \mathrm{dom}(f_{o_i}) \text{ et } (\sigma_n, \Gamma_n) \in f_{o_i}(k) \text{ et } k(\sigma_n, \Gamma_n) \in L(\mathcal{A}_{o_i})$$

Deux cas se présentent :

cas 1 : $\sigma_n \neq \epsilon$

$k \in L(f_{o_i}/\mathcal{A}_{o_i}) \Rightarrow \exists k' \in L(f_d/\mathcal{A}_d)$ *tel que* $\pi_\epsilon(k) = \pi_\epsilon(k')$.

D'autre part f_d *est complet* $\Rightarrow k' \in \mathrm{dom}(f_d)$.

Il est evident qu'on a aussi $(\sigma_n, \Gamma_n) \in f_d(k')$.

Finalement $k'(\sigma_n, \Gamma_n) \in L(\mathcal{A}_{o_i}) \Rightarrow \exists k'' \in L(\mathcal{A}_d)$ *tel que* $\delta_d((\sigma_n, \Gamma_n), \delta_d(k'', z_0))!$. *Alors il est possible de montrer, en faisant une discussion semblable à celle que nous avons fait pour montrer l'inclusion* $\pi_\epsilon(L(f_d/\mathcal{A}_d)) \subseteq$

$\pi_\epsilon(L(f_{o_i}/\mathcal{A}_{o_i}))$, *que* $k'' \in L(f_d/\mathcal{A}_d)$ *et dans ce cas on aura* $k''(\sigma_n, \Gamma_n) \in$ $L(f_d/\mathcal{A}_d)$, *avec* $\pi_\epsilon(k''(\sigma_n, \Gamma_n)) = \pi_\epsilon(m)$.

$\Rightarrow \pi_\epsilon(k(\sigma_n, \Gamma_n)) \in \pi_\epsilon(L(f_d/\mathcal{A}_d))$

cas 2 : $\sigma_n = \epsilon$

$\pi_\epsilon(m) = \pi_\epsilon(k) = \pi_\epsilon(k')$.

$\Rightarrow \pi_\epsilon(L(f_{o_i}/\mathcal{A}_{o_i})) \subseteq \pi_\epsilon(L(f_d/\mathcal{A}_d))$

$\Rightarrow \pi_\epsilon(L(f_{o_i}/\mathcal{A}_{o_i})) = \pi_\epsilon(L(f_d/\mathcal{A}_d))$

Montrons maintenant que $\pi_\epsilon(S(f_{o_i}/\mathcal{A}_{o_i})) = \pi_\epsilon(S(f_d/\mathcal{A}_d))$*:*

$$\begin{aligned}
\pi_\epsilon(S(f_{o_i}/\mathcal{A}_{o_i})) &\subseteq \pi_\epsilon(\lim(L(f_{o_i}/\mathcal{A}_{o_i}))) \\
&\subseteq \lim(\pi_\epsilon(L(f_{o_i}/\mathcal{A}_{o_i}))) \\
&\subseteq \lim(\pi_\epsilon(L(f_d/\mathcal{A}_d))) \\
&\subseteq \lim(\pi_\epsilon(\mathrm{pre}(S(f_d/\mathcal{A}_d)))) \\
&\subseteq \pi_\epsilon(S(f_d/\mathcal{A}_d))
\end{aligned}$$

Montrons l'autre inclusion :

$$\begin{aligned}
s \in S(f_d/\mathcal{A}_d) &\Rightarrow s \in S(\mathcal{A}_d) \\
&\Rightarrow \exists s' \in S(\mathcal{A}_{o_i}) \text{ tel que } \pi_\epsilon(s) = \pi_\epsilon(s').
\end{aligned}$$

Montrons que $s' \in \lim(L(f_{o_i}/\mathcal{A}_{o_i}))$, *c.-à.-d. montrons que* $\mathrm{pre}(s') \subseteq$ $L(f_{o_i}/\mathcal{A}_{o_i})$.

on a

$$s \in S(f_d/\mathcal{A}_d) \Rightarrow s \in \lim(L(f_d/\mathcal{A}_d))$$
$$\Rightarrow \text{pre}(s) \subseteq L(f_d/\mathcal{A}_d)$$
$$\Rightarrow \pi_\epsilon(\text{pre}(s)) \subseteq \pi_\epsilon(L(f_d/\mathcal{A}_d))$$

Or $\pi_\epsilon(s) = \pi_\epsilon(s')$

$\Rightarrow \pi_\epsilon(\text{pre}(s)) = \pi_\epsilon(\text{pre}(s'))$

$\Rightarrow \pi_\epsilon(\text{pre}(s')) \subseteq \pi_\epsilon(L(f_{o_i}/\mathcal{A}_{o_i}))$.

Ceci implique nécessairement que $\text{pre}(s') \subseteq L(f_{o_i}/\mathcal{A}_{o_i})$. *En effet, pour tout* $k \in \text{pre}(s')$, *on a :*

$$k \in \text{pre}(s') \Rightarrow \pi_\epsilon(k) \in \pi_\epsilon(\text{pre}(s'))$$
$$\Rightarrow \pi_\epsilon(k) \in \pi_\epsilon(L(f_{o_i}/\mathcal{A}_{o_i}))$$
$$\Rightarrow \exists k' \in L(f_{o_i}/\mathcal{A}_{o_i}) \text{ tel que } \pi_\epsilon(k') = \pi_\epsilon(k)$$
$$\Rightarrow k \in L(f_{o_i}/\mathcal{A}_{o_i}).$$

Donc on a $\text{pre}(s') \subseteq L(f_{o_i}/\mathcal{A}_{o_i})$

$\Rightarrow s' \in \lim(L(f_{o_i}/\mathcal{A}_{o_i}))$

$\Rightarrow s' \in S(f_{o_i}/\mathcal{A}_{o_i})$

$\Longrightarrow \pi_\epsilon(S(f_d/\mathcal{A}_d)) \subseteq \pi_\epsilon(S(f_{o_i}/\mathcal{A}_{o_i}))$.

$\Rightarrow \pi_\epsilon(S(f_{o_i}/\mathcal{A}_{o_i})) = \pi_\epsilon(S(f_d/\mathcal{A}_d))$.

Montrons que f_{o_i} *est complet :*

Soit $m \in L(f_{o_i}/\mathcal{A}_{o_i}) \Rightarrow \exists m' \in L(f_d/\mathcal{A}_d)$ *tel que* $\pi_\epsilon(m) = \pi_\epsilon(m')$.

Or $m' \in L(f_d/\mathcal{A}_d) \Rightarrow m' \in \mathrm{dom}(f_d)$

$\Rightarrow m \in \mathrm{dom}(f_{o_i})$.

Montrons que f_{o_i} *est non-bloquant:*

Il est évident que $\mathrm{pre}\,(S(f_{o_i}/\mathcal{A}_{o_i})) \subseteq L(f_{o_i}/\mathcal{A}_{o_i})$, *il reste à montrer l'autre inclusion.*

$m \in L(f_{o_i}/\mathcal{A}_{o_i}) \Rightarrow \exists m' \in L(f_d/\mathcal{A}_d)$ *tel que* $\pi_\epsilon(m) = \pi_\epsilon(m')$.

Il est toujours possible d'étendre m' *avec* $t' \in ((\Sigma_o \cup \{\epsilon\}) \times C)^\omega$ *tel que* $s' = m't' \in S(f_d/\mathcal{A}_d)$.

$\Rightarrow$ *Il est possible d'étendre* m *avec* $t \in ((\Sigma_o \cup \{\epsilon\}) \times C)^\omega$ *tel que* $\pi_\epsilon(mt) = \pi_\epsilon(m't')$.

$\Rightarrow mt \in S(f_d/\mathcal{A}_d)$

$\Rightarrow m \in \mathrm{pre}\,(S(f_d/\mathcal{A}_d))$.

$\Rightarrow f_d$ *est non-bloquant.*

◇

Proposition 5.8 *Soit* $f_{o_i} : ((\Sigma_o \cup \{\epsilon\}) \times C)^* \longrightarrow C_o$ *un superviseur pour* $\mathcal{A}_o$ *com-*

plet et non-bloquant. La fonction suivante

$$\begin{aligned} f_d : \ ((\Sigma_o \cup \{\epsilon\}) \times C)^* &\longrightarrow C_o \\ m &\longmapsto f_{o_i}(m') \end{aligned}$$

avec $m' \in L(f_o/\mathcal{A}_o)$ *tel que* $\pi_\epsilon(m') = \pi_\epsilon(m)$,

définit un superviseur complet et non-bloquant pour $\mathcal{A}_d$ *tel que*

$$\pi_\epsilon(L(f_d/\mathcal{A}_d)) = \pi_\epsilon(L(f_{o_i}/\mathcal{A}_{o_i}))$$

et

$$\pi_\epsilon(S(f_d/\mathcal{A}_d)) = \pi_\epsilon(S(f_{o_i}/\mathcal{A}_{o_i}))$$

Démonstration:

Il est clair que f_d *est une fonction parce que pour tout* $m_1, m_2 \in L(f_{o_i}/\mathcal{A}_{o_i})$ *tel que* $\pi_\epsilon(m_1) = \pi_\epsilon(m_2)$, *on a nécessairement* $f_{o_i}(m_1) = f_{o_i}(m_2)$, *par définition même de* f_{o_i}.

Montrons que $\pi_\epsilon(L(f_d/\mathcal{A}_d)) = \pi_\epsilon(L(f_{o_i}/\mathcal{A}_{o_i}))$.

Commençons par montrer que

$$\pi_\epsilon(L(f_{o_i}/\mathcal{A}_{o_i})) \subseteq \pi_\epsilon(L(f_d/\mathcal{A}_d))$$

c.-à.-d. montrons que $\forall m \in L(f_{o_i}/\mathcal{A}_{o_i})$, *il existe* $m' \in L(f_d/\mathcal{A}_d)$ *tel que* $\pi_\epsilon(m) = \pi_\epsilon(m')$.

Démonstration par récurrence sur la longueur du mot m.

$|m| = 0$, *le résultat est vrai, en effet à* $m = 1_{(\Sigma_o \cup \{\epsilon\}) \times C}$, *on associe* $m' = 1_{(\Sigma_o \cup \{\epsilon\}) \times C}$.

Supposons que c'est vrai pour tout mot de longueur $\leq n-1$ *et montrons que c'est aussi vrai pour tout mot de longueur* n.

$$m = k(\sigma_n, \Gamma_n) \in L(f_{o_i}/\mathcal{A}_{o_i}) \Leftrightarrow k \in L(f_{o_i}/\mathcal{A}_{o_i}) \text{ et } k \in \mathrm{dom}(f_{o_i}) \text{ et } (\sigma_n, \Gamma_n) \in f_{o_i}(k) \text{ et } k(\sigma_n, \Gamma_n) \in L(\mathcal{A}_{o_i})$$

$k \in L(f_{o_i}/\mathcal{A}_{o_i}) \Rightarrow \exists k' \in L(f_d/\mathcal{A}_d)$ *tel que* $\pi_\epsilon(k) = \pi_\epsilon(k')$ *(d'après l'hypothèse de récurrence).*

cas 1 : $\sigma_n = \epsilon$

On prends $m' = k'$ *et on a*

$$\begin{aligned} \pi_\epsilon(m') &= \pi_\epsilon(k') \\ &= \pi_\epsilon(k) \\ &= \pi_\epsilon(k(\epsilon, \Gamma_n)) \\ &= \pi_\epsilon(m) \end{aligned}$$

cas 2 : $\sigma_n \neq \epsilon$

$k \in \mathrm{dom}(f_{o_i}) \Rightarrow k' \in \mathrm{dom}(f_d)$, *par définition de* f_d.

On a aussi $(\sigma_n, \Gamma_n) \in f_{o_i}(k) \Rightarrow (\sigma_n, \Gamma_n) \in f_d(k')$.

D'autre part, $k'(\sigma_n, \Gamma_n) \in L(\mathcal{A}_{o_i}) \Rightarrow \exists k'' \in L(\mathcal{A}_d)$ *tel que* $\delta_d((\sigma_n, \Gamma_n), \delta_d(k'', z_0))!$.

Si $k'' = k'$, *alors on prend* $m' = k'(\sigma_n, \Gamma_n)$.

Sinon, $\pi_\epsilon(k) = \pi_\epsilon(k')$ *et* $k' \in L(f_d/\mathcal{A}_d)$ *et* $k'' \in L(\mathcal{A}_d)$, *impliquent* $k'' \in L(f_d/\mathcal{A}_d)$. *En effet, Supposons que* $k'' \notin L(f_d/\mathcal{A}_d)$ *et soit* $p \leq k''$ *le plus long préfixe de* k'' *qui appartient à* $L(f_d/\mathcal{A}_d)$.

Supposons que k'' *s'écrit sous la forme* $k'' = (\sigma_1, \Gamma_1)...(\sigma_m, \Gamma_m)$ *et que* p *s'écrit sous la forme* $p = (\sigma_1, \Gamma_1)...(\sigma_i, \Gamma_i)$. *Alors* $p(\sigma_{i+1}, \Gamma_{i+1}) \notin L(f_d/\mathcal{A}_d)$. *Alors on a deux possibilités :*

Sous-cas 1: $\sigma_{i+1} = \epsilon$

$p(\epsilon, \Gamma_{i+1}) \notin L(f_d/\mathcal{A}_d) \Rightarrow \Gamma_{o_{i+1}} \neq f_d(p)$. *Posons* $p' = (\sigma_{i+1}, \Gamma_{i+1})...(\sigma_m, \Gamma_m)$ *et soit* σ_j *le premier symbole observable de* p' *alors on a nécessairement* p' *qui s'écrit sous la forme* $p' = (\epsilon, \Gamma_{i+1})(\epsilon, \Gamma_{i+1})...(\epsilon, \Gamma_{i+1})(\sigma_j, \underbrace{\Gamma_{i+1}}_{\Gamma_j})$ $(\sigma_{j+1}, \Gamma_{j+1})...(\sigma_m, \Gamma_m)$.

D'autre part, $\pi_\epsilon(k'') = \pi_\epsilon(k')$, *donc* k' *s'écrit sous la forme* $k' = l(\sigma_j, \Gamma_{i+1})l'$, *puisque* $\sigma_j \notin \epsilon$, *et on a évidemment* $\pi_\epsilon(l) = \pi_\epsilon(p)$.

$\Rightarrow p \in \mathrm{dom}(f_d)$ *et on a* $f_d(p) = f_d(l) = \Gamma_{o_{i+1}}$. *Absurde puisqu'on a dit déjà que* $f_d(p) = f_d(l) \neq \Gamma_{o_{i+1}}$.

Sous-cas 2 : $\sigma_{i+1} \neq \epsilon$

Dans ce cas on $p(\sigma_{i+1}, \Gamma_{i+1}) \notin L(f_d/\mathcal{A}_d) \Rightarrow f_d(p) \neq \Gamma_{o_{i+1}}$.

Or $\pi_\epsilon(k'') = \pi_\epsilon(k')$, *donc on peut écrire* k' *sous la forme* $l(\sigma_{i+1}, \Gamma_{i+1})l'$ *avec* $\pi_\epsilon(l) = \pi_\epsilon(p)$.

$\Rightarrow f_d(l) = f_d(p) = \Gamma_{o_{i+1}}$. *Absurde.*

Donc on arrive dans les deux cas à une absurdité.

$\Rightarrow$ *Notre supposition est fausse.*

$\Rightarrow k'' \in L(f_d/\mathcal{A}_d)$.

En plus, on a $k'' \in \mathrm{dom}(f_d)$, *parce que* $\pi_\epsilon(k'') = \pi_\epsilon(k)$ *et* $k \in L(f_{o_i}/\mathcal{A}_{o_i})$.

Ceci implique aussi que $(\sigma_n, \Gamma_n) \in f_d(k'')$.

Comme $k''(\sigma_n, \Gamma_n) \in L(\mathcal{A}_d)$, *alors on a nécessairement* $m' = k''(\sigma_n, \Gamma_n) \in L(f_d/\mathcal{A}_d)$, *et on a bien entendu* $\pi_\epsilon(m') = \pi_\epsilon(m)$

$\Rightarrow \pi_\epsilon\left(L(f_{o_i}/\mathcal{A}_{o_i})\right) \subseteq \pi_\epsilon\left(L(f_d/\mathcal{A}_d)\right)$.

Montrons l'autre inclusion:

Soit $m \in L(f_d/\mathcal{A}_d)$, *montrons, par récurrence sur la longueur du mot* m, *qu'il existe* $m' \in L(f_{o_i}/\mathcal{A}_{o_i})$ *tel que* $\pi_\epsilon(m) = \pi_\epsilon(m')$.

Pour $m = 1_{(\Sigma_o \cup \{\epsilon\}) \times C}$, *le résultat est vrai.*

Supposons que le résultat soit vrai pour tout mot de longueur $\leq n-1$ *et montrons que ça reste aussi vrai pour tout mot de longueur* n.

$$m = k(\sigma_n, \Gamma_n) \in L(f_d/\mathcal{A}_d) \;\Leftrightarrow\; k \in L(f_d/\mathcal{A}_d) \text{ et } k \in \mathrm{dom}(f_d) \text{ et } (\sigma_n, \Gamma_n) \in f_d(k) \text{ et } k(\sigma_n, \Gamma_n) \in L(\mathcal{A}_d)$$

D'après l'hypothèse de récurrence, $k \in L(f_d/\mathcal{A}_d) \Rightarrow \exists k' \in L(f_{o_i}/\mathcal{A}_{o_i})$ *tel que* $\pi_\epsilon(k') = \pi_\epsilon(k)$.

Si $\sigma = \epsilon$, *alors il suffit de prendre* $m' = k'$. *Sinon, on a* $k' \in L(f_{o_i}/\mathcal{A}_{o_i}) \Rightarrow$ $k' \in \mathrm{dom}(f_{o_i})$, *puisque* f_{o_i} *est complet.*

On a aussi $(\sigma_n, \Gamma_n) \in f_d(k) \Rightarrow (\sigma_n, \Gamma_n) \in f_{o_i}(k')$.

D'autre part $k(\sigma_n, \Gamma_n) \in L(\mathcal{A}_d) \Rightarrow \exists k'' \in L(\mathcal{A}_{o_i})$ *et* $\exists y \in \delta_{o_i}(k', y_0)$ *tel que* $\delta_{o_i}((\sigma_n, \Gamma_n), y)$ *et* $\pi_\epsilon(k'') = \pi_\epsilon(k)$.

Alors on a, $k'' \in L(\mathcal{A}_{o_i})$, $k' \in L(f_{o_i}/\mathcal{A}_{o_i})$ *et* $\pi_\epsilon(k') = \pi_\epsilon(k'')$, $\Rightarrow k'' \in$ $L(f_{o_i}/\mathcal{A}_{o_i})$.

En plus il est clair que $k'' \in \mathrm{dom}(f_{o_i})$ *et que* $(\sigma_n, \Gamma_n) \in f_{o_i}(k'')$.

$\Rightarrow m' = k''(\sigma_n, \Gamma_n) \in L(f_{o_i}/\mathcal{A}_{o_i})$, *et on a* $\pi_\epsilon(m') = \pi_\epsilon(m)$

$\Rightarrow \pi_\epsilon(L(f_d/\mathcal{A}_d)) \subseteq \pi_\epsilon(L(f_{o_i}/\mathcal{A}_{o_i}))$

$\Rightarrow \pi_\epsilon(L(f_d/\mathcal{A}_d)) = \pi_\epsilon(L(f_{o_i}/\mathcal{A}_{o_i}))$.

Montrons que $\pi_\epsilon(S(f_{o_i}/\mathcal{A}_{o_i})) = \pi_\epsilon(S(f_d/\mathcal{A}_d))$:

$$s \in S(f_d/\mathcal{A}_d) \Rightarrow s \in S(\mathcal{A}_d)$$
$$\Rightarrow \exists s' \in S(\mathcal{A}_{o_i}) \text{ tel que } \pi_\epsilon(s) = \pi_\epsilon(s').$$

Montrons que $s' \in \lim(L(f_{o_i}/\mathcal{A}_{o_i}))$, *c.-à.-d. montrons que* $\mathrm{pre}(s') \subseteq$ $L(f_{o_i}/\mathcal{A}_{o_i})$.

on a

$$s \in S(f_d/\mathcal{A}_d) \Rightarrow s \in \lim(L(f_d/\mathcal{A}_d))$$
$$\Rightarrow \text{pre}(s) \subseteq L(f_d/\mathcal{A}_d)$$
$$\Rightarrow \pi_\epsilon(\text{pre}(s)) \subseteq \pi_\epsilon(L(f_d/\mathcal{A}_d))$$

Or $\pi_\epsilon(s) = \pi_\epsilon(s')$

$\Rightarrow \pi_\epsilon(\text{pre}(s)) = \pi_\epsilon(\text{pre}(s'))$

$\Rightarrow \pi_\epsilon(\text{pre}(s')) \subseteq \pi_\epsilon(L(f_{o_i}/\mathcal{A}_{o_i}))$.

Ceci implique nécessairement que $\text{pre}(s') \subseteq L(f_{o_i}/\mathcal{A}_{o_i})$. *En effet, pour tout* $k \in \text{pre}(s')$, *on a :*

$$k \in \text{pre}(s') \Rightarrow \pi_\epsilon(k) \in \pi_\epsilon(\text{pre}(s'))$$
$$\Rightarrow \pi_\epsilon(k) \in \pi_\epsilon(L(f_{o_i}/\mathcal{A}_{o_i}))$$
$$\Rightarrow \exists k' \in L(f_{o_i}/\mathcal{A}_{o_i}) \text{ tel que } \pi_\epsilon(k') = \pi_\epsilon(k)$$
$$\Rightarrow k \in L(f_{o_i}/\mathcal{A}_{o_i}).$$

Donc on a $\text{pre}(s') \subseteq L(f_{o_i}/\mathcal{A}_{o_i})$

$\Rightarrow s' \in \lim(L(f_{o_i}/\mathcal{A}_{o_i}))$

$\Rightarrow s' \in S(f_{o_i}/\mathcal{A}_{o_i})$

$\Longrightarrow \pi_\epsilon(S(f_d/\mathcal{A}_d)) \subseteq \pi_\epsilon(S(f_{o_i}/\mathcal{A}_{o_i}))$.

Montrons maintenant que $\pi_\epsilon(S(f_{o_i}/\mathcal{A}_{o_i})) \subseteq \pi_\epsilon(S(f_d/\mathcal{A}_d))$*:*

$$\begin{aligned}
\pi_\epsilon(S(f_{o_i}/\mathcal{A}_{o_i})) &\subseteq \pi_\epsilon(\lim(L(f_{o_i}/\mathcal{A}_{o_i}))) \\
&\subseteq \lim(\pi_\epsilon(L(f_{o_i}/\mathcal{A}_{o_i}))) \\
&\subseteq \lim(\pi_\epsilon(L(f_d/\mathcal{A}_d))) \\
&\subseteq \lim(\pi_\epsilon(\mathrm{pre}(S(f_d/\mathcal{A}_d)))) \\
&\subseteq \pi_\epsilon(S(f_d/\mathcal{A}_d))
\end{aligned}$$

$\Longrightarrow \pi_\epsilon(S(f_d/\mathcal{A}_d)) = \pi_\epsilon(S(f_{o_i}/\mathcal{A}_{o_i}))$.

Montrons aussi que f_d *est complet.*

Soit $m \in L(f_d/\mathcal{A}_d)$*, montrons que* $m \in \mathrm{dom}(f_d)$*:*

On a $m \in L(f_d/\mathcal{A}_d) \Rightarrow \exists m' \in L(f_{o_i}/\mathcal{A}_{o_i})$ *tel que* $\pi_\epsilon(m) = \pi_\epsilon(m')$.

Or $m' \in L(f_{o_i}/\mathcal{A}_{o_i}) \Rightarrow m' \in \mathrm{dom}(f_{o_i})$*, car* f_{o_i} *est complet.*

$\Rightarrow m \in \mathrm{dom}(f_d)$.

Finalement montrons que f_d *est non-bloquant.*

Il est évident que $\mathrm{pre}(S(f_d/\mathcal{A}_d)) \subseteq L(f_d/\mathcal{A}_d)$*, puisque* $S(f_d/\mathcal{A}_d) \subseteq \lim(L(f_d/\mathcal{A}_d))$.

Il reste à montrer que $L(f_d/\mathcal{A}_d) \subseteq \mathrm{pre}(S(f_d/\mathcal{A}_d))$.

Soit $k \in L(f_d/\mathcal{A}_d) \Rightarrow \exists k' \in L(f_{o_i}/\mathcal{A}_{o_i})$ *tel que* $\pi_\epsilon(k') = \pi_\epsilon(k)$.

Or f_{o_i} *est non-bloquant* $\Rightarrow \exists t' \in ((\Sigma_o \cup \{\epsilon\}) \times C)^\omega$ *tel que* $s' = k't' \in$

$S(f_{o_i}/\mathcal{A}_{o_i})$.

$\Rightarrow \exists t \in ((\Sigma_o \cup \{\epsilon\}) \times C)^\omega$ *tel que* $kt \in S(f_d/\mathcal{A}_d)$.

$\Rightarrow k \in \text{pre}\,(S(f_d/\mathcal{A}_d))$

$\Rightarrow L(f_d/\mathcal{A}_d) \subseteq \text{pre}\,(S(f_d/\mathcal{A}_d))$

$\Rightarrow L(f_d/\mathcal{A}_d) = \text{pre}\,(S(f_d/\mathcal{A}_d))$

$\Rightarrow f_d$ *est non-bloquant.*

$\diamond$

5.7 Conclusion

Dans ce chapitre nous avons donné une approche de résolution pour le problème de synthèse d'un superviseur complet et non-bloquant pour un SÉD contrôlé G_C partiellement observé à travers un masque $\widetilde{M}_C$. L'idée de base de notre approche était de faire une série d'abstractions qui a permis de ramener le problème initial à celui du contrôle d'un automate fini de Rabin déterminite, que nous avons désigné par $\mathcal{A}_d$, pour la satisfaction de sa propre condition d'acceptation. Ce dernier problème a été déjà résolu par Thistle et Wonham (voir chapitre 2). Il suffit donc d'utiliser ces travaux pour trouver un superviseur pour $\mathcal{A}_d$, et de déterminer à partir de celui-là une suite de superviseurs correspondant chacun à une étape d'abstraction; le dernier élément de cette suite étant un superviseur $\widetilde{V}_C$ pour $(G_C, \widetilde{M}_C)$. Bien entendu, ceci ne constitue pas une solution pour notre problème de départ, à savoir superviser

(G, M), mais il est possible de déduire à partir de $\tilde{V}_C$, un superviseur pour (G, M) complet, non-bloquant et qui respecte la spécification, en utilisant les résultats du chapitre précédent.

Malheureusement, contrairement au cas de la synthèse monolithique, pour laquelle nous avons pu trouver un algorithme de synthèse, le problème devient non-décidable lorsque nous nous intéressons au cas de la supervision répartie. Ce résultat surprenant fera l'objet du chapitre suivant.

CHAPITRE 6

SYNTHÈSE DE SUPERVISEURS RÉPARTIS

6.1 Introduction

Dans le chapitre 3, nous avons introduit le problème de synthèse de superviseur pour superviser les séquences infinies d'événements générées par un SÉD partiellement observé à travers un masque M. Les résultats des chapitres 4 et 5 donnent une procédure qui permet de résoudre le problème. Une extension de ce problème au cas de la supervision répartie le rend non-décidable.

En effet, dans ce chapitre nous allons introduire un nouveau problème qui est celui de la synthèse d'un superviseur réparti qui est composé de deux superviseurs locaux, et qui fait en sorte que les séquences infinies générées en boucle fermée sous sa supervision respectent les langages de spécification. Nous allons montrer que dans certains cas, ce problème peut être équivalent à celui de l'acceptation du mot vide par une machine de Turing. Ceci nous permet de montrer que la question d'existence d'une solution au problème est non-décidable, parce qu'il a été déjà établi que l'arrêt d'un machine de Turing sur une bande vide est non-décidable aussi. L'idée de transformer le problème d'arrêt d'une machine de Turing sur la bande vide en un problème d'existence d'une solution au problème de synthèse d'un superviseur réparti s'inspire des travaux de Peterson et Reif[PR79], qui ont adopté

une approche similaire pour montrer que le problème d'appartenance d'un mot fini à la classe des langages reconnus par les machines alternantes, à joueurs multiples et information incomplète et dont l'espace est borné, est non-décidable; ainsi que des travaux de Rosner[?] qui a adapté l'idée de Peterson et Reif pour montrer que le problème de synthèse de systèmes réactifs distribués est aussi non-décidable. Le résultat que nous allons établir dans ce chapitre concerne la supervision des mots infinis, mais une légère modification de la démonstration permet de trouver un résultat similaire pour la supervision des mots finis. Ces résultats sont présentés à la section suivante. À la dernière section nous discutons de leur impact.

6.2 Supervision répartie

Même si on se retrouve dans un contexte de mots infinis, nous allons considérer la même définition de conjonction de superviseurs que celle du chapitre 1.

Le problème auquel on s'intéresse est le suivant :

Problème 6.1 (Supervision Répartie des ω-langages : PSR$^\omega$) *Soit G, un SÉD défini sur l'alphabet Σ, A et E deux ω-langages qui vérifient la propriété suivante $A \subseteq E \subseteq S(G)$. Alors étant donné deux alphabets d'observation Σ_i, $i = 1, 2$ et deux masques $M_i : L(G) \longrightarrow \Sigma_i^*$, concevoir deux superviseurs locaux V_i, $i = 1, 2$ complets et non-bloquants tels que $\ker M_i \subseteq \ker V_i$, $i = 1, 2$, la conjonction $V_1 \wedge V_2$*

est non-bloquante, et

$$A \subseteq S(V_1 \wedge V_2/G) \subseteq E$$

Théorème 6.1 *L'existence d'une solution pour le PSR$^\omega$ est non décidable*

Démonstration: *Soit M une machine de Turing, définie sur un alphabet T. L'idée de la démonstration est de définir à partir de M un SÉD G et des spécifications A et E et de montrer que la question d'existence d'un superviseur réparti pour G qui permet de répondre aux spécifications est équivalente a l'acceptation par M du mot vide. Comme nous savons que le problème d'arrêt d'une machine de Turing sur une bande vide est non-décidable, nous pouvons en conclure que la question d'existence d'une solution pour ce problème de supervision particulier est non-décidable. Or ce dernier n'est autre qu'un cas particulier du PSR$^\omega$, il s'en suit que l'existence de solution pour le PSR$^\omega$ est aussi non-décidable.*

Définissons un alphabet $\mathcal{T}$ suffisamment riche pour encoder des configurations de M. Rappellons qu'une séquence de configurations de M est de la forme $\#C_0\#C_1\#...\#C_n\#,...$ où $C_i = (q, x)$, où q est l'état interne de M et $x \in T^*|T^+$ qui représente l'état de la bande (le symbole | dénote la position de la tête de lecture/écriture). Donc notre alphabet $\mathcal{T}$ doit contenir l'alphabet T, le symbole # et des symboles pour représenter les états internes de M. Définissons à partir de $\mathcal{T}$ deux alphabets disjoints $\mathcal{T}_1$ et $\mathcal{T}_2$, obtenus en ajoutant les indices 1 et 2 aux symboles

de $\mathcal{T}$. Posons

$$\Sigma_i := \mathcal{T}_i \dot{\cup} \{P_i, S_i\},\ i = 1, 2$$

avec $P_i, S_i \notin \mathcal{T}_i$.

Le système de commande auquel nous allons aboutir va être constitué d'un SÉD G et de deux superviseurs locaux qui vont essayer de générer chacun la séquence de configuration générée par M lorsqu'elle part avec une bande vide. Le générateur quant à lui va essayer de vérifier la validité des séquences générées en faisant certains tests implicites dans sa fonction de transition. Le symbole P_i sert à indiquer au superviseur V_i de donner le prochain symbole dans sa séquence, alors que S_i va servir à lui indiquer de sauter à la prochaine séquence. La fonction de transition de G doit respecter les règles suivantes:

- *Pour tout $i \in \{1, 2\}$, on a tout symbole de P_i ou S_i est nécessairement suivi par un symbole de Σ_i.*

- *Tout symbole dans $\mathcal{T}_1 \cup \mathcal{T}_2$ est nécessairement suivi par un symbole de $\{P_1, S_1, P_2, S_2\}$.*

La vérification que le SÉD fait sur les séquences données par les deux superviseurs n'est pas totale. En effet, il se contente juste de vérifier si certaines portions de chaque séquence, prises au hasard, correspondent bien à une configuration de M. Il vérifie aussi pour deux configurations successives C_j et C_{j+1} qu'on a $C_j \vdash C_{j+1}$ (c'est à dire que C_{j+1} est le successeur de C_j selon la structure de M). S'il détecte une

inconsistence il entre dans un état puits x_p à partir duquel il ne sort plus. Sinon, s'il détecte une configuration acceptante de M (i.e. q est un état marqué) alors il entre dans un autre état puits x_a duquel il ne sort plus. Alors on pose $I = R = \{x_a\}$, $A = \emptyset$ et $E = S(G)$.

Les masques d'observations sont naturellement $M_i : L(G) \longrightarrow \Sigma_i^*$, $i = 1, 2$, et les superviseurs locaux qu'il faut concevoir seront de la forme $V_i : M_i(L(G)) \longrightarrow C_i$, $i = 1, 2$. avec $C_i = \{\Gamma_i^j, 1 \leq j \leq n_i\}$ et les Γ_i^j qui sont sous la forme

$$\Gamma_i^j = a_j \cup \{S_i, P_i\} \cup \Sigma_j,\ a_j \in \mathcal{T}_i$$

Puisque chaque superviseur ne peut ni observer, ni contrôler ce qui est donné par l'autre supervieur, le seul moyen d'empêcher l'automate représentant G d'entrer dans l'état puits x_p est de trouver une stratégie qui permet aux deux superviseurs de générer la séquence de configurations qui correspond à celle générée par M en lisant une bande vide, et cette stratégie permet d'entrer dans l'état acceptant x_a si et seulement si M accepte le mot vide.

$\Longrightarrow$ L'existence de cette stratégie est non-décidable.

$\Longrightarrow$ l'existence d'une solution au PSR^ω est aussi non-décidable.

$\diamond$

Ce résultat est aussi valable dans un contexte de mots finis, comme le montre le

corollaire qui suit.

Problème 6.2 (Supervision Répartie des langages : PSR*) *Soit G, un SÉD défini sur l'alphabet Σ, A et E deux langages qui vérifient la propriété suivante $A \subseteq E \subseteq L(G)$. Alors, étant donné deux alphabets d'observation Σ_i, $i = 1,2$ et deux masques $M_i : L(G) \longrightarrow \Sigma_i^*$, concevoir deux superviseurs locaux V_i, $i = 1,2$ complets et non-bloquants tels que $\ker M_i \subseteq \ker V_i$, $i = 1,2$, la conjonction $V_1 \wedge V_2$ est non-bloquante, et*

$$A \subseteq L(V_1 \wedge V_2/G) \subseteq E$$

Corollaire 6.1 *L'existence d'une solution pour le PSR* est non-décidable.*

Démonstration: *Une approche semblable à celle utilisée pour démontrer le théorème 6.1 permet de démontrer aussi ce résultat, la seule différence réside dans la condition d'acceptation de l'automate fini sur mots finis qui représente G. Celui-ci aura une structure de transition identique à celle définie précédemment, avec un seul état marqué qui est x_a.*

◇

6.3 Conclusion

Dans ce chapitre nous avons montré que l'existence d'une solution à une classe de problèmes de supervision répartie est non-décidable. Ce résultat suggère qu'il faut se restreindre à des cas particuliers pour trouver des algorithmes de synthèse qui nous permettraient de trouver un superviseur réparti conforme à sa spécification. Comme ce résultat est aussi valable lorsque nous nous intéressons uniquement aux mots finis générés en boucle fermée, on peut conclure que toutes les procédures de synthèse qui ont été trouvées jusqu'à maintenant ne peuvent que correspondre à des cas particuliers et ne peuvent être étendues au cas général.

CONCLUSION

Dans ce mémoire, nous nous sommes intéressés à la supervision des séquences infinies générées par les SÉDs partiellement observés. Nous avons donné une construction qui permet de ramener le problème de supervision sous observations partielles à un problème de supervision sous observations complètes, et nous avons montré comment à partir d'une solution à ce dernier on peut déduire une solution à notre problème de départ. Le problème de supervision sous observations complètes a été résolu grâce aux travaux de Thistle et Wonham. Il a été démontré notamment qu'il n'admet pas de solution optimale qui correspond au superviseur le plus permissif qui permet de respecter la spécification. Ceci est aussi le cas du problème de supervision sous observations partielles, nous nous sommes contentés donc de chercher un superviseur quelconque qui permet de respecter la spécification sur le ω-langage généré en boucle fermée.

Afin de pouvoir résoudre le problème nous avons supposé que le procédé et la spécification sont représentés par des automates finis sur mots infinis et que le masque d'observation est représenté par un automate fini de Moore. En plus, nous avons supposé que l'automate de Rabin, qui représente le SÉD, possède une condition d'acceptation triviale, ceci afin de simplifier la construction de l'automate de Rabin déterministe, à partir de l'automate de Rabin universel.

Une perspective de travaux futurs serait donc d'essayer de se débarasser de cette hypothèse simplificatrice et de considérer un procédé qui est représenté par un au-

tomate de Rabin quelconque.

L'autre problème auquel nous nous sommes intéressés est celui de la supervision répartie. En effet, nous avons montré que l'existence d'une solution à ce problème est non-décidable, même si on s'intéresse uniquement à la supervision des mots finis. Ceci suggère donc d'essayer d'enrichir la structure du SÉD et des masques d'observations, pour pouvoir peut-être résoudre le problème dans un contexte moins général.

BIBLIOGRAPHIE

[CDC88] *Proc. of 27th IEEE Conf. Decision and Control*, Austin, TX, USA, December 1988.

[CDFV88] R. Cieslak, C. Desclaux, A. Fawaz, and P. Varaiya. Supervisory control of discrete-event processes with partial observations. *IEEE Trans. Autom. Control*, 33(3):249–260, March 1988.

[GR87] C.H. Golaszewski and P.J. Ramadge. Control of discrete event processes with forced events. In *Proc. of 26th IEEE Conf. Decision and Control*, pages 247–251, Los Angeles, CA, USA, December 1987.

[GR88a] C.H. Golaszewski and P.J. Ramadge. Mutual exclusion problems for discrete event systems with shared events. In *Proc. of 27th IEEE Conf. Decision and Control* [CDC88], pages 234–239.

[GR88b] C.H. Golaszewski and P.J. Ramadge. Supervisory control of discrete event processes with arbitrary controls. In M.J. Denham and A.J. Laub, editors, *Advanced Computing Concepts and Techniques in Control Engineering*, volume 47 of *Computer and Systems Sciences*, pages 459–469. Springer-Verlag, 1988.

[KGM91] R. Kumar, V. Garg, and S.I. Marcus. On ω-controllability and ω-normality of DEDS. In *Proc. of 1991 American Control Conf.*, Boston, MA, USA, June 1991.

[LW88a] F. Lin and W.M. Wonham. Decentralized control and coordination of discrete event systems. In *Proc. of 27th IEEE Conf. Decision and Control* [CDC88], pages 1125–1130.

[LW88b] F. Lin and W.M. Wonham. On observability of discrete-event systems. *Information Sciences*, 44(3):173–198, 1988.

[PR79] G.L. Peterson and J.H. Reif. Multiple-person alternation. In *Proc. of the 1979 IEEE Symp. on the Foundations of Computer Science*, 1979.

[Ram89] P.J. Ramadge. Some tractable supervisory control problems for discrete-event systems modeled by Büchi automata. *IEEE Trans. Autom. Control*, 34(1):10–19, January 1989.

[RW89] P.J. Ramadge and W.M. Wonham. The control of discrete event systems. *Proc. of the IEEE*, 77(1):81–98, January 1989.

[RW92] K. Rudie and W.M. Wonham. Think globally, act locally: Decentralized supervisory control. *IEEE Trans. Autom. Control*, 37(11):1692–1708, November 1992.

[Saf88] S. Safra. On the complexity of ω-automata. In *Proc. of the 1988 IEEE Symp. on the Foundations of Computer Science*, pages 319–327, 1988.

[Thi91] J.G. Thistle. *Control of Infinite Behavior of Discrete Event Systems.* PhD thesis, Dept. of El. Eng., Univ. of Toronto, Canada, January 1991.

[TW88] J.G. Thistle and W.M. Wonham. On the synthesis of supervisors subject to ω-language specifications. In *Proc. of 22nd Conf. on Information Sciences and Systems*, pages 440–444, Princeton, NJ, USA, March 1988.

[TW91] J.G. Thistle and W.M. Wonham. Control of ω-automata, Church's problem, and the emptiness problem for tree ω-automata. In E. Börger, G. Jáger, H. Kleine Büning, and M.M. Richter, editors, *Proc. of 5th Workshop on Computer Science Logic, CSL'91*, volume 626 of *LNCS*, pages 367–381, Bern, Switzerland, October 1991. Springer-Verlag, Berlin, Germany.

[TW94a] J.G. Thistle and W.M. Wonham. Control of infinite behaviour of finite automata. *SIAM J. Control Optim.*, 32(4): , July 1994.

[TW94b] J.G. Thistle and W.M. Wonham. Supervision of infinite behaviour of discrete-event systems. *SIAM J. Control Optim.*, 32(4): , July 1994.

[Zho92] H. Zhong. *Hierarchical Control of Discrete-Event Systems.* PhD thesis, Dept. of El. Eng., Univ. of Toronto, Canada, July 1992.