N^o d'ordre : 3086

THÈSE

Présentée

DEVANT L'UNIVERSITÉ DE RENNES I

Pour obtenir

Le grade de : **DOCTEUR DE L'UNIVERSITÉ DE RENNES 1**
Mention : INFORMATIQUE

par

Benoît Gaudin

Équipe d'accueil : VerTeCS

École Doctorale : Matisse

Composante universitaire : Institut de Formation Supérieure en Informatique et Communication

Synthèse de contrôleurs sur des systèmes à événements discrets structurés

Rapportée par :

MM. Hassane ALLA
Stéphane LAFORTUNE

Soutenue le 15 novembre 2004 devant la commission d'examen composée de :

MM.	Hassane	ALLA	Rapporteur
	Philippe	DARONDEAU	Directeur
	Claude	JARD	
	Hervé	MARCHAND	Encadreur
	Laure	PETRUCCI	
	Olivier	RIDOUX	

Remerciements :

Je remercie tout d’abord Hervé Marchand, qui m’a encadré depuis le DEA dans le domaine de la synthèse de contrôleurs. Il a su me guider avec enthousiasme et me conseiller efficacement, tout en me laissant travailler très librement.

Je remercie aussi tous les membres de l’équipe VerTeCs, dirigée par Thierry Jéron, pour leur disponibilité et l’ambiance favorable qu’ils font régner au sein de ce projet.

Je remercie Hassane Alla et Stéphane Lafortune d’avoir accepté d’être rapporteurs, ainsi que Laure Petrucci, Claude Jard, et Olivier Ridoux de m’avoir fait l’honneur de participer au jury. Je remercie aussi mon directeur, Philippe Darondeau, pour ses remarques pertinentes.

Je remercie enfin ma famille et mes amis pour leur soutien et leur aide.

Table des matières

Introduction

Les systèmes informatiques ont été introduits et se sont développés afin de faciliter la modélisation et l'automatisation de certaines tâches. Il sont aujourd'hui omniprésents et se trouvent dans des systèmes de grande envergure (centrales nucléaires, réseaux de télécommunication), dans des systèmes de taille plus modeste (avionique, automobile, systèmes de production), ou encore des systèmes de petite taille comme des contrôleurs divers utilisés dans des produits de grande diffusion (cartes à puce, machines à café, téléphones portables, ...).

De tels systèmes peuvent posséder des natures très diverses. Certains, tels que les systèmes d'exploitation, sont *réactifs* et leurs comportements évoluent en fonction de leur environnement, alors que d'autres n'ont pour objectif que l'accomplissement d'une tâche très précise. Les systèmes *embarqués*, tels qu'un programme sur une carte à puce, sont quant à eux plongés dans un environnement possédant des contraintes particulières. De plus, les comportements de ces systèmes peuvent eux aussi posséder des natures très différentes. Certains évoluent naturellement de manière continue au cours du temps alors que d'autres évoluent à la fois de manière continue et discrète. Ainsi, le bras articulé d'un robot peut se déplacer entre différents points de l'espace. L'évolution entre deux points est continue, mais lorsqu'un de ces points est atteint, le bras s'arrête avant d'effectuer un autre mouvement continu. Un tel comportement est dit *hybride* : certains comportements du système s'effectuent de manière continue et il existe un ensemble discret d'états pertinents du système (les points d'arrêts). Le comportement d'un tel système peut même être abstrait, afin de n'en garder que la partie discrète. En effet, dans le cas du bras articulé, les mouvements possibles peuvent être représentés par l'ensemble des points d'arrêt ainsi que les transitions qui existent entre deux de ces points. Le système ainsi obtenu est alors dit *à événements discrets*. Différents modèles tels que les automates, les langages, ou encore les réseaux de Petri, ont été développés et permettent de modéliser de tels systèmes.

Il est fréquent que la sécurité des moyens et des hommes soit en partie gérée par des systèmes informatiques. Il faut alors non seulement que les systèmes considérés soient fiables, mais aussi que leurs comportements soient valides, au sens où ils doivent correspondre aux exigences introduites lors de la phase de conception. De fait, le développement de méthodes formelles qui garantissent leur correction et leur validité est devenu un objectif crucial de l'informatique. A cet effet, on peut distinguer les techniques de vérification de celles de contrôle. Les techniques de vérification permettent de valider/invalider le fait qu'un système ou un modèle satisfasse certaines propriétés. Les prouveurs, le model-checking, l'analyse statique, ou le test constituent par exemple de telles techniques.

Lorsque les systèmes sont déjà conçus et que la validité de leur comportement n'est pas connue, il est alors intéressant de pouvoir concevoir des contrôleurs contraignant le système afin

que ne se produisent que des comportements souhaités. Un contrôleur est alors vu comme un système qui interagit avec le système initial pour en contraindre les comportements. De plus, puisque les systèmes actuels sont de plus en plus complexes, assurer la validité des comportements à partir de contrôleurs ne peut pas en général être effectué par expertise. Une automatisation de cette phase, appelée *synthèse de contrôleurs* s'avère donc pertinente.

Ce sont Ramadge et Wonham [59, 79] qui à l'origine ont introduit la théorie de la synthèse de contrôleurs sur les systèmes à événements discrets. Le système, ainsi que l'ensemble des comportements valides (appelé *objectif de contrôle*) sont donnés chacun par un langage. À partir de ces deux langages, le but de la synthèse consiste alors à déterminer un sous ensemble des comportements du système qui appartienne aussi à l'objectif de contrôle[1]. Ce sous ensemble représente les comportements du système subissant l'action du contrôleur et caractérise ce dernier. Toutefois, cet ensemble de comportements ne peut être quelconque, car l'occurrence de certains événements ne peut être empêchée par le superviseur. De tels événements sont alors dits *incontrôlables*.

La théorie introduite par Ramadge et Wonham s'est beaucoup développée et a donné lieu à diverses extensions, mais souffre d'un problème de complexité lorsque les systèmes considérés sont complexes, au sens où ils sont finis mais possèdent un nombre important d'états. Or les systèmes complexes sont généralement obtenus par composition de sous systèmes plus simples, interagissant entre eux. Le mode de composition de ces sous systèmes induit une structure sur le système, et ce type de système est alors dit *structuré*.

Afin de répondre au problème de complexité lié à la synthèse de contrôleurs sur des systèmes complexes, différentes approches peuvent être envisagées. Une première consiste à déterminer une représentation pertinente et efficace des systèmes à contrôler et des objectifs de contrôle, afin que les algorithmes classiques de synthèse de contrôleurs puissent être appliqués. Les techniques symboliques utilisant une représentation des systèmes par des *diagrammes de décision* [10], offrent une solution à cette approche. Les techniques symboliques constituent une solution s'appuyant sur une implémentation efficace [31, 26, 29, 51]. Dans cette étude, nous nous sommes focalisés sur des méthodes dont l'efficacité est indépendante de l'implémentation. Ceci n'empêchant en rien de l'améliorer à l'aide de techniques symboliques dans la mesure où les différents algorithmes développés dans le cadre de cette thèse se prêtent relativement bien à une implémentation symbolique basée sur les diagrammes de décision.

Une seconde approche consiste à bénéficier de la connaissance de la structure du système ou de l'objectif, pour développer des techniques qui sont propres à la synthèse de systèmes structurés, et permettent un gain significatif en complexité. Ainsi, lorsque l'objectif de contrôle est donné par un ensemble d'objectifs devant être assurés, il est intéressant d'effectuer la synthèse pour chacun de ces objectifs plutôt que de calculer un unique objectif de contrôle et d'effectuer la synthèse ensuite [56]. Cette technique est appelée *synthèse modulaire* et tire parti de la structure de l'objectif. De plus, il se peut que le système possède lui-même une structure particulière. Dans le cadre de la *synthèse décentralisée* [17, 63], le système est composé de différents sites et le problème de contrôle revient alors à calculer un superviseur pour chaque site, de sorte que le système contrôlé global assure l'ensemble des propriétés souhaitées. De même, certains systèmes peuvent posséder une structure hiérarchique au sens où il existe des visions plus ou moins abstraites de celui-ci. La synthèse hiérarchique [78, 12] fournit des conditions suivant lesquelles il est possible de résoudre

[1] À noter que dans ces modèles, le temps n'intervient pas. L'analyse se situe donc à un niveau qualitatif.

le problème de la synthèse de contrôleurs sur une abstraction du système, généralement moins complexe que le système lui-même.

Dans cette étude, nous nous intéressons au problème de la synthèse de contrôleurs sur des systèmes structurés qui sont décrits à partir de sous-systèmes liés par composition : la concurrence et la hiérarchie. Il est en effet fréquent de rencontrer des systèmes à la fois spécifiés à partir de sous-systèmes agissant en parallèle, et spécifiés par "couche" en donnant des représentations du système à différents niveaux d'abstraction. Étant donné un système à contrôler et un objectif de contrôle, le but de la synthèse consiste ici à déterminer un superviseur qui restreigne le moins possible les comportements du système et assure que les comportements du système ainsi contrôlé appartiennent à l'objectif de contrôle. La démarche consiste à tenter de résoudre ce problème en tenant compte de la structure du système pour factoriser les calculs du superviseur, sans avoir à expliciter le système à contrôler. Puisque la structure du système nous intéresse davantage que le modèle lui-même, nous avons choisi de représenter les systèmes par des langages réguliers (ou par les automates associés à ces langages). Ces modèles ainsi que les résultats classiques de synthèse de contrôleurs sont présentés dans le chapitre 1. Dans les chapitres 2 et 3 nous traitons différents cas de synthèses de contrôleurs sur des systèmes structurés.

- Le chapitre 2 se focalise sur le cas où le système est modélisé par une composition parallèle de langages et l'objectif de contrôle par un langage. Une méthode permettant, sous certaines hypothèses, de synthétiser un superviseur assurant l'objectif y est décrite. En tirant parti de la structure concurrente du système, un superviseur global peut ainsi être calculé, uniquement à partir de calculs locaux. Plus précisément, étant donné un objectif de contrôle, le but consiste à calculer un superviseur maximal (i.e le plus permissif) assurant cet objectif, sans construire explicitement le système à contrôler. Des approximations du système sont dérivées à partir des sous-systèmes qui le composent, et une propriété appelée *contrôlabilité partielle*, devant être vérifiée par l'objectif de contrôle sur ces approximations, est introduite. Assurer la contrôlabilité partielle de l'objectif de contrôle sur chacune des approximations permet, sous certaines hypothèses, d'en déduire un superviseur maximal assurant l'objectif de contrôle sur le système global. De plus, les calculs effectués ont une faible complexité et ne nécessitent pas de construire explicitement le système, évitant ainsi l'explosion combinatoire inhérente aux systèmes concurrents. Cette méthode offre ainsi un gain significatif en complexité, et s'applique pour des objectifs ne possédant pas nécessairement une structure similaire à celle du système. Notons que cette approche diffère du cas décentralisé puisqu'ici un superviseur global, décrit de manière modulaire, est obtenu, plutôt qu'un ensemble de superviseurs interagissant avec chacun des sous systèmes.
- En revanche, dans le chapitre 3, nous nous intéressons aux états des systèmes concurrents modélisés par une composition parallèle d'automates. Les états constituent donc une notion importante de ce chapitre et les objectifs de contrôle considérés sont représentés par un ensemble d'états que l'on souhaite interdire. La méthode proposée permet la synthèse d'un superviseur, sans calculer explicitement (sous forme d'un unique automate) le système global et assure ainsi un gain significatif en complexité. De plus, les restrictions introduites dans le chapitre 2 sur l'objectif de contrôle ainsi que sur la nature contrôlable des événements, sont levées. Toutefois, l'évaluation du superviseur obtenu nécessite d'effectuer des calculs en-ligne. On montre alors dans ce chapitre que la structure du système permet de limiter la complexité de ces calculs. De plus, étant donné un système subissant l'action d'un

superviseur, on s'intéresse aux états depuis lesquels le système est bloqué au sens où aucun événement ne peut se produire depuis ces états. Nous présentons dans ce chapitre une méthode efficace permettant de synthétiser un superviseur, assurant à la fois l'objectif de contrôle, mais aussi qu'aucun état atteignable du système ainsi obtenu ne soit bloquant. Finalement nous étendons les résultats au cadre de la synthèse de contrôleurs des systèmes possédant une structure concurrente et hiérarchique. Comme dans la première partie du chapitre 3, les objectifs à assurer sont représentés par un ensemble d'états à interdire.

– Finalement, le chapitre 4 illustre les résultats donnés dans les chapitres précédents. Le prototype SYNTOOL constitue une mise en oeuvre de ces résultats et permet donc d'effectuer de la synthèse de contrôleurs sur des systèmes structurés.

Chapitre 1

Rappels sur la théorie du contrôle

1.1 Préliminaires sur les modèles

Dans cette section, on introduit les modèles et les concepts classiquement utilisés pour représenter et manipuler les systèmes à événements discrets. On pourra se reporter à [4, 35, 13] pour plus de détails.

1.1.1 Langages

1.1.1.1 Définitions

Un système à événements discrets peut être modélisé par l'ensemble des comportements qu'il est sensé produire. Ainsi, durant l'exécution du système, l'occurrence de certains événements peut être observée, et la séquence des événements qui se sont ainsi produits depuis l'instant initial forme un comportement possible du système. L'ensemble des événements qui peuvent se produire est supposé fini, est appelé *alphabet* du système, et est modélisé par un ensemble fini Σ.

Si σ_1 et σ_2 représentent deux événements, $\sigma_1.\sigma_2$ représente la concaténation de ces deux événements. Toutes les séquences finies d'événements peuvent alors être représentées par une concaténation d'événements $s = \sigma_1.\sigma_2 \dots \sigma_n$ appelée mot sur l'alphabet Σ. Par la suite, Σ^* représentera l'ensemble des séquences finies d'éléments de Σ. Un événement particulier, noté ϵ et appartenant à Σ^*, est introduit pour modéliser le fait qu'aucun événement ne se produit.

Étant donné un système sur un alphabet Σ, l'ensemble des comportements de ce système peut donc être modélisé par un sous ensemble non vide de Σ^*. Un tel sous ensemble est appelé langage sur l'alphabet Σ[1]. Étant donné un alphabet Σ, on note $\mathcal{L}(\Sigma)$ l'ensemble des langages sur Σ.

Pour un comportement $s \in L$ donné, il peut être utile de s'intéresser aux comportements s' qui peuvent se prolonger en s. Dans cette optique, on dit que s' est un préfixe de s s'il existe une séquence $s" \in \Sigma^*$ telle que $s's" = s$, ce qui sera noté $s' \leq s$. Étant donné un langage $L \subseteq \Sigma^*$, on note

$$\overline{L} = \{s \in \Sigma^* | \ \exists s' \in L, \ s \leq s'\} \tag{1.1}$$

[1]Le langage vide et le langage réduit à $\{\epsilon\}$ possèdent des interprétations différentes. En effet, le langage vide ne représente pas les comportements d'un système, alors que le langage $\{\epsilon\}$ représente les comportements d'un système dont l'unique comportement consiste à "ne rien faire".

la clôture par préfixe de L. Un langage L sera dit préfixe-clos si et seulement si $\overline{L} = L$. Intuitivement, un langage L est préfixe-clos si pour n'importe quel élément $s \in L$, tous les préfixes de s appartiennent aussi à L. Notons que Σ^* est préfixe-clos et que ϵ appartient à tout langage préfixe-clos. Le langage modélisant le comportement d'un système n'est pas nécessairement préfixe-clos. En fait, les mots présents dans le langage sont utilisés pour représenter les comportements ayant un intérêt particulier tandis que l'exécution partielle d'un comportement est en général décrite seulement par le préfixe d'un mot du langage (i.e. les préfixes d'un mot du langage ne sont pas pertinents pour caractériser le système).

De manière duale, pour $L \in \mathcal{L}(\Sigma)$, $\Sigma' \subseteq \Sigma$ et $s \in \Sigma^*$, on définit l'ensemble des suffixes de s dans L appartenant à Σ'^*, noté $L(s, \Sigma')$, par

$$L(s, \Sigma') = \{t \in \Sigma'^* | \; st \in L\}$$

Dans la suite, pour simplifier les notations, $L(s, \Sigma)$ sera noter $L(s)$.

1.1.1.2 Opérations sur les langages

Les systèmes complexes sont généralement obtenus à partir de plusieurs sous systèmes, qui interagissent les uns avec les autres. Ainsi, pour modéliser un tel système, il faut non seulement modéliser chacun des sous systèmes qui le composent, mais aussi le mode d'interaction entre ces sous systèmes. Une description modulaire des systèmes complexes est obtenue à l'aide de la *composition parallèle*.

Projection. Afin de définir formellement cette opération, nous introduisons dans un premier temps une opération utile sur les langages : la projection dite *naturelle* sur un sous alphabet. Soient Σ' et Σ deux alphabets tels que $\Sigma' \subseteq \Sigma$, alors $P_{\Sigma'} : \; \Sigma^* \longrightarrow \Sigma'^*$ la *projection naturelle* de Σ sur Σ' est définie par :

$$\begin{aligned} P_{\Sigma'}(\epsilon) &= \epsilon \\ P_{\Sigma'}(\sigma) &= \begin{cases} \sigma & \text{si } \sigma \in \Sigma' \\ \epsilon & \text{sinon} \end{cases} \\ P_{\Sigma'}(s\sigma) &= P_{\Sigma'}(s)P_{\Sigma'}(\sigma) \text{ pour } s \in \Sigma^*, \; \sigma \in \Sigma \end{aligned}$$

La notion de projection sur Σ' peut être étendue à un langage $L \in \mathcal{L}(\Sigma)$ par

$$P_{\Sigma'}(L) = \{s \in \Sigma'^* | \; \exists s' \in L, \; P_{\Sigma'}(s') = s\}$$

L' opération $P_{\Sigma'}$ consiste donc à enlever d'une séquence donnée du langage, tous les événements n'appartenant pas à Σ'. De manière duale, étant donnés deux alphabets Σ' et Σ, ainsi qu'un langage $L \subseteq \Sigma'^* \subseteq \Sigma^*$, la *projection inverse* de Σ' dans Σ, notée $P_{\Sigma'}^{-1}$ est définie par

$$P_{\Sigma'}^{-1}(L) = \{s \in \Sigma^* \mid P_{\Sigma'}(s) \in L\}.$$

Intuitivement, $P_{\Sigma'}^{-1}(L)$ est obtenu en insérant de toutes les manières possibles des séquences de $(\Sigma \setminus \Sigma')^*$ dans les mots de L.

Lemme 1 *Soient $\Sigma' \subseteq \Sigma$, $L_1,\ L_2 \subseteq \Sigma^*$ et $L'_1,\ L'_2 \subseteq \Sigma'^*$, la projection $P_{\Sigma'}$ et la projection inverse $P^{-1}_{\Sigma'}$ vérifient les propriétés suivantes :*

$$
\begin{aligned}
P_{\Sigma'}(\overline{L_1}) &= \overline{P_{\Sigma'}(L_1)} && (1.2)\\
P_{\Sigma'}(L_1 L_2) &= P_{\Sigma'}(L_1) P_{\Sigma'}(L_2) && (1.3)\\
P_{\Sigma'}(L_1 \cup L_2) &= P_{\Sigma'}(L_1) \cup P_{\Sigma'}(L_2) && (1.4)\\
P_{\Sigma'}(L_1 \cap L_2) &\subseteq P_{\Sigma'}(L_1) \cap P_{\Sigma'}(L_2) && (1.5)\\
P^{-1}_{\Sigma'}(L'_1 L'_2) &= P^{-1}_{\Sigma'}(L'_1) P^{-1}_{\Sigma'}(L'_2) && (1.6)\\
P^{-1}_{\Sigma'}(L'_1 \cup L'_2) &= P^{-1}_{\Sigma'}(L'_1) \cup P^{-1}_{\Sigma'}(L'_2) && (1.7)\\
P^{-1}_{\Sigma'}(L'_1 \cap L'_2) &= P^{-1}_{\Sigma'}(L'_1) \cap P^{-1}_{\Sigma'}(L'_2) && (1.8)
\end{aligned}
$$

Composition parallèle. La définition de la composition parallèle entre deux langages L_1 et L_2, respectivement sur les alphabets Σ_1 et Σ_2 est définie à partir des projections naturelles P_{Σ_i} : $\Sigma^* \longrightarrow \Sigma_i^*$ où $i = 1, 2$ et $\Sigma = \Sigma_1 \cup \Sigma_2$. Cette composition permet de modéliser les comportements d'un système à partir de sous systèmes.

Définition 1 *Soient $L_1 \in \mathcal{L}(\Sigma_1)$ et $L_2 \in \mathcal{L}(\Sigma_2)$. La composition parallèle entre L_1 et L_2 est un langage $L_1 \parallel L_2 \in \mathcal{L}(\Sigma_1 \cup \Sigma_2)$ défini par*

$$L_1 \parallel L_2 = P^{-1}_{\Sigma_1}(L_1) \cap P^{-1}_{\Sigma_2}(L_2) \tag{1.9}$$

De manière informelle, la composition parallèle de deux langages L_1 et L_2, respectivement sur des alphabets Σ_1 et Σ_2, représente un langage dont les éléments sont des entrelacements de séquences de L_1 et L_2. Plus précisément, après qu'une séquence de la composition se soit produite, les événements de $\Sigma_1 \setminus \Sigma_2$ peuvent se produire indépendemment des événements de $\Sigma_2 \setminus \Sigma_1$ (et inversement). En revanche, les événements de $\Sigma_1 \cap \Sigma_2$ ne peuvent se produire que si cela est autorisé par chacun des sous systèmes. Par conséquent, tous les sous systèmes partageant un événement doivent en permettre l'occurrence pour que celui-ci se produise dans la composition. Le langage obtenu par composition est formé des mots dont la projection sur Σ_1 (resp. Σ_2) est un mot du langage L_1 (resp. L_2). En d'autres termes,

$$L_1 \parallel L_2 = \{s \in \Sigma^* \mid P_1(s) \in L_1 \ \wedge \ P_2(s) \in L_2\}.$$

La composition parallèle s'étend aisément à un nombre fini quelconque de langages, par associativité. Les systèmes modélisés comme une composition parallèle de langages seront appelés *systèmes concurrents* dans la suite.

1.1.2 Machines à états finies

Les langages offrent un cadre théorique intéressant pour modéliser les systèmes à événements discrets. Cependant, bien que les langages permettent de modéliser un ensemble infini de comportements, certains langages peuvent toutefois être représentés de manière finie, offrant ainsi un moyen pratique de les manipuler. Les langages vérifiant cette propriété sont dits *réguliers* et sont représentables par une *machine à états finie*.

1.1.2.1 Définitions et sémantique

Définition 2 *Un automate (ou machine à états finie), abrégée FSM (pour Finite State Machine) G est un 5-tuple $(\Sigma, Q, q_0, Q_m, \delta)$ où*

- *Σ est l'alphabet fini des actions sur G.*
- *Q est l'ensemble fini des états de G, $q_o \in Q$ est l'état initial de G, Q_m représente l'ensemble des états finals (marqués) de G et*
- *δ est la fonction partielle de transition définie sur $\Sigma \times Q \longrightarrow Q$.*

δ représente ici une *fonction partielle* et non une *relation partielle* comme c'est généralement le cas. Cela induit que les automates considérés ici sont tous déterministes : $\forall q \in Q,\ \sigma \in \Sigma,$ si σ peut se produire depuis l'état q, alors il existe un unique état $q' \in Q$ tel que $\delta(\sigma, q) = q'$ Intuitivement, comme les langages, les automates modélisent les comportements d'un système à événements discrets : l'état q_0 de l'automate modélise l'état initial du système. Depuis cet état initial, les comportements possibles et les états atteints sont décrits par la fonction partielle de transition $\delta(\cdot)$. $\delta(\cdot)$ étant partiellement définie sur $\Sigma \times Q$, pour $(\sigma, q) \in \Sigma \times Q$, on note $\delta(\sigma, q)!$ pour signifier que $\delta(\cdot)$ est définie en (σ, q). De plus, pour faciliter les notations, on étend classiquement la fonction $\delta(\cdot)$ sur $\Sigma^* \times Q$ de manière récursive. Soient $s \in \Sigma^*$, $\sigma \in \Sigma$, et soit $q \in Q$.

$$\begin{aligned} \delta(\epsilon, q) &= q \\ \delta(s\sigma, q) &= \delta(\sigma, \delta(s, q)) \end{aligned}$$

L'ensemble des comportements possibles décrits par un automate est donné par le *langage généré* : le langage généré par l'automate $G = (\Sigma, Q, q_0, Q_m, \delta)$, noté $\mathcal{L}(G)$ est défini par

$$\mathcal{L}(G) = \{s \in \Sigma^* |\ \delta(s, q_0)!\} \tag{1.10}$$

Outre le langage généré, les états finals de G induisent aussi un langage dit *langage marqué*. Ce langage, noté $\mathcal{L}_m(G)$, est défini par

$$\mathcal{L}_m(G) = \{s \in \Sigma^* |\ \delta(s, q_0)! \wedge \delta(s, q_0) \in Q_m\} \tag{1.11}$$

Les séquences du langage généré représentent les comportements du système qui peuvent se produire durant son exécution. Le langage marqué d'un automate est l'ensemble des séquences du langage généré qui représentent une finalité (accomplissement d'une tâche).

Finalement, étant donné un automate G, le langage généré qui lui est associé est unique. En revanche, étant donné un langage régulier L, il existe en général plusieurs automates dont le langage généré est L.

1.1.2.2 Opérations sur les automates

Étant donné un automate G, il apparaît intéressant d'effectuer certains calculs sur ses états, afin d'avoir des renseignements sur les états qui permettent *d'atteindre/d'être atteints* à partir d'une séquence donnée.

Définition 3 *Soient $G = (\Sigma, Q, q_0, Q_m, \delta)$ un automate et $A \subseteq \Sigma$. On définit l'opérateur Pre_A^G pour tout $E \subseteq Q$ par*

$$Pre_A^G(E) = E \cup \{q \in Q |\ \exists \sigma \in A,\ \delta(\sigma, q) \in E\} \tag{1.12}$$

$Pre_A^G(E)$ représente donc l'ensemble des états desquels il est possible d'atteindre un des états de E par tirage d'un événement appartenant à A. On donne à présent quelques propriétés classiques de l'opérateur Pre_A^G.

Proposition 1 *Soient $G = (\Sigma, Q, q_0, Q_m, \delta)$ un automate, $A, A' \subseteq \Sigma$ et $E, E' \subseteq Q$.*

$$Pre_A^G(E \cup E') = Pre_A^G(E) \cup Pre_A^G(E') \tag{1.13}$$
$$Pre_{A\cup A'}^G(E) = Pre_A^G(E) \cup Pre_{A'}^G(E) \tag{1.14}$$
$$E \subseteq E' \Longrightarrow Pre_A^G(E) \subseteq Pre_A^G(E') \tag{1.15}$$

Définition 4 *Soient $G = (\Sigma, Q, q_0, Q_m, \delta)$ un automate et $A \subseteq \Sigma$. On définit l'opérateur $Post_A^G$ pour tout $E \subseteq Q$ par :*

$$Post_A^G(E) = E \cup \{q \in Q |\ \exists \sigma \in A, \exists q' \in E,\ \delta(\sigma, q') = q\}$$

$Post_A^G(E)$ représente donc l'ensemble des états qu'il est possible d'atteindre depuis un des états de E par tirage d'un événement appartenant à A.

Pour un entier n quelconque, on note $Pre_A^{G(n)}$ (resp. $Post_A^{G(n)}$) la composition $Pre_A^G \circ \cdots \circ Pre_A^G$ (resp. $Post_A^G \circ \cdots \circ Post_A^G$) n fois. On définit alors les opérateurs *Reach*$_A^G$ et *CoReach*$_A^G$ par :

$$\textit{Reach}_A^G(E) = \bigcup_{n\geq 0} Post_A^{G(n)}(E) \tag{1.16}$$

$$\textit{CoReach}_A^G(E) = \bigcup_{n\geq 0} Pre_A^{G(n)}(E) \tag{1.17}$$

Reach$_A^G(E)$ (resp. *CoReach*$_A^G(E)$) représente en fait le point fixe obtenu par applications successives de l'opérateur $Post_A^G$ (resp. Pre_A^G), à partir de l'ensemble E. De manière intuitive,

- *Reach*$_A^G(E)$ représente l'ensemble des états qu'il est possible d'atteindre depuis un des états de E par tirage d'une séquence d'événements appartenant à A. Lorsque A représente l'alphabet tout entier et que E correspond à $\{q_0\}$, alors les états de *Reach*$_A^G(E)$ seront dits *accessibles (ou atteignables) dans G*.
- *CoReach*$_A^G(E)$ représente l'ensemble des états desquels il est possible d'atteindre un des états de E par tirage d'une séquence d'événements appartenant à A. Lorsque A représente l'alphabet tout entier et que $E = Q_m$, alors les états de *CoReach*$_A^G(E)$ seront dits *coaccessibles (ou coatteignables) dans G* relativement à Q_m.

Remarque 1 *Par la suite, nous noterons de manière indifférente* *Reach*$_A^G(E)$ *(resp.* *CoReach*$_A^G(E)$*) et* $(Post_A^G)^*(E)$ *(resp.* $\left(Pre_A^G\right)^*(E)$*).*

Notons pour finir, que puisque les fonctions Pre_A^G et $Post_A^G$ sont croissantes, et qu'un automate possède un nombre fini d'états, *Reach*$_A^G$ et *CoReach*$_A^G$ sont bien définies et calculables.

Notion de blocage. Les états finaux d'un automate permettent de caractériser les comportements du système qui représentent l'accomplissement d'une tâche. Par conséquent, il apparaît important que ces états soient toujours atteignables au cours de l'exécution du système. De plus, les états d'un automate qui ne sont pas atteignables depuis son état initial ne jouent aucun rôle dans la description de l'évolution du système. Étant donné un automate $G = (\Sigma, Q, q_0, Q_m, \delta)$,

$$\mathit{Reach}_A^G(\{q_0\}) \cap \mathit{CoReach}_A^G(Q_m)$$

représente l'ensemble des états qui sont à la fois accessibles au cours de l'exécution et qui permettent également d'atteindre un état final du système (coaccessibles).

La restriction d'un automate G à l'ensemble de ces états qui sont à la fois atteignables et coatteignables est obtenue par application d'un opérateur appelé *Trim*. Ainsi, si on note $Q' = \mathit{Reach}_A^G(\{q_0\}) \cap \mathit{CoReach}_A^G(Q_m)$ et que $Q' \neq \emptyset$, alors Trim(G) est donné par l'automate $G = (\Sigma, Q', q_0, Q'_m, \delta')$ où $Q'_m = Q_m \cap Q'$ et δ' est la restriction de la fonction partielle de transition δ à $\Sigma \times Q'$. Lorsque tous les états d'un automate G sont accessibles et coaccessibles, alors $Trim(G) = G$. L'automate G sera alors dit *Trim*.

La notion d'automate *Trim* permet d'une part de ne pas considérer d'états "superflus" (i.e non atteignables depuis l'état initial), et est d'autre part intéressante pour caractériser les blocages d'un système. Basé sur les définitions précédentes, un automate G est *bloquant* si

- soit il existe un état atteignable q tel que $\delta(q) = \emptyset$ mais $q \notin Q_m$. Un tel état est alors dit en *deadlock*,
- ou il existe un ensemble d'états atteignables et non marqués de G qui forment une composante fortement connexe, telle que si le système atteint cet ensemble, il n'a plus aucun moyen d'atteindre un état marqué.

Maintenant, si G est *trim* par rapport à q_0 et Q_m, alors tous les états de G sont accessibles depuis l'état initial q_0 et co-accessibles pour les états finaux Q_m (i.e. $\forall q \in Q$, il existe $s, t \in \Sigma^*$, et $q_m \in Q_m$, tel que $\delta(s, q_o) = q$ and $\delta(t, q) = q_m$), ce qui induit que G est *non-bloquant*.

D'un point de vue langage, on a alors la propriété suivante :

Proposition 2 *Si G est trim, alors $\overline{\mathcal{L}_m(G)} = \mathcal{L}(G)$.*

Notons d'une part que la réciproque de cette proposition est fausse en générale, et d'autre part que si $\overline{\mathcal{L}_m(G)} = \mathcal{L}(G)$ alors tous les états atteignables depuis l'état initial permettent d'atteindre un état final $\mathit{Reach}_\Sigma^G(\{q_0\}) \subseteq \mathit{CoReach}_\Sigma^G(Q_m)$. Dans la suite, les automates considérés seront implicitement supposés *Trim*.

Composition de sous systèmes. Finalement, comme pour les langages, il est important de pouvoir décrire des systèmes complexes obtenus par composition de sous systèmes.

Avant de donner la définition de la composition parallèle d'automates, on présente l'opérateur In$(\cdot)$, qui indique quels sous systèmes d'un système concurrent partagent un événement. Pour

cela, on considère un ensemble $(G_i)_{1\leq i\leq n}$ d'automates modélisant les sous systèmes du système global. Soit $\sigma \in \Sigma = \cup_i \Sigma_i$ et $\mathcal{A}$ un ensemble fini et non vide de $\{1,\ldots,n\}$, alors

$$\textsc{In}(\sigma, \mathcal{A}) = \{i \in \mathcal{A} |\ \sigma \in \Sigma_i\} \tag{1.18}$$

Étant donné un événement de Σ, $\textsc{In}(\sigma, \mathcal{A})$ représente l'ensemble des indices des automates de $\{G_i\}_{i\in\mathcal{A}}$ qui partagent cet événement. Par la suite, lorsqu'il n'y aura pas d'ambiguïté, on notera $\textsc{In}(\sigma)$ (au lieu de $\textsc{In}(\sigma, \{1,\ldots,n\})$) l'ensemble des automates de $\{G_i\}_{1\leq i\leq n}$ partageant σ.

Définition 5 *Soient $\{G_i\}_{1\leq i\leq n}$ n automates avec pour tout $i \in \{1,\ldots,n\}$, $G_i = (\Sigma_i, Q_i, q_{o_i}, Q_{m_i}, \delta_i)$. La composition parallèle des automates G_i, notée $G_1 \parallel \cdots \parallel G_n$, est l'automate $(\Sigma, Q, q_o, Q_m, \delta)$, tel que $\Sigma = \bigcup_i \Sigma_i$, $Q = \times_i Q_i$, l'état initial est donné par $q_0 = (q_{0_1},\ldots,q_{0_n})$ et l'ensemble des états finals est donné par $Q_m = \times_i Q_{m_i}$. La fonction partielle de transition δ est définie sur $\Sigma \times Q$ de la manière suivante : pour $q = (q_1,\ldots,q_n) \in Q$ et $\sigma \in \Sigma$,*

$$\delta(\sigma, q)! \iff \forall i \in \textsc{In}(\sigma),\ \delta_i(\sigma, q_i)!$$

et si $\delta(\sigma, q)!$, alors $\delta(\sigma, q) = (q'_1,\ldots,q'_n)$ avec

$$(\forall i \in \textsc{In}(\sigma),\ q'_i = \delta_i(\sigma, q_i)) \text{ et } (\forall i \notin \textsc{In}(\sigma),\ q'_i = q_i)$$

Tout comme pour les langages, cette définition traduit d'une part le fait qu'un événement partagé n'est admissible dans G que s'il est admissible dans chacun des sous-systèmes qui le partagent. Cette définition traduit d'autre part qu'un événement local à un sous-système peut se produire dans G dès-lors qu'il est admissible dans ce sous-système.

Le lien entre la composition parallèle sur les automates (Def 5) et la composition parallèle entre langages définie au paragraphe 1.1.1.2 est donné par la proposition suivante.

Proposition 3 *Soit $(G_i)_{1\leq i\leq n}$ un ensemble d'automates, alors*

$$\mathcal{L}(G_1 \parallel \cdots \parallel G_n) = \mathcal{L}(G_1) \parallel \cdots \parallel \mathcal{L}(G_n) \tag{1.19}$$

$$\mathcal{L}_m(G_1 \parallel \cdots \parallel G_n) = \mathcal{L}_m(G_1) \parallel \cdots \parallel \mathcal{L}_m(G_n) \tag{1.20}$$

1.2 Synthèse de contrôleurs

Dans cette section, nous introduisons les concepts et résultats de base de la théorie de la synthèse de contrôleurs introduites par Ramadge et Wonham [59], ainsi que les extensions et résultats qui en ont découlé.

1.2.1 Supervision et contrôlabilité

Dans cette section, nous supposerons que le système à contrôler est modélisé par un automate G. Son comportement est donc donné par le langage $\mathcal{L}(G)$ et son comportement marqué par le langage $\mathcal{L}_m(G)$. Le comportement (marqué) de G peut s'avérer ne pas être entièrement satisfaisant dans le sens où il ne respecte pas certaines propriétés appellées *objectifs de contrôle*. il est

donc nécessaire de réduire ce comportement dans le but d'assurer ces objectifs. Cette restriction est réalisée par le biais d'un superviseur qui peut être vu comme une fonction qui, à partir de l'histoire du système (i.e, la trajectoire que le système a emprunté jusque là), va envoyer à celui-ci l'ensemble des événements qui doivent être interdits pour rester dans l'ensemble des comportements décrits par l'objectif. Contrôler un système consiste donc à lui ajouter des contraintes supplémentaires, induisant une réduction du comportement du système à un comportement souhaité.

Formellement, un superviseur S est une fonction $L \longrightarrow 2^{\Sigma}$ restreignant l'ensemble des comportements de G suivant le schéma donné par la figure 1.1.

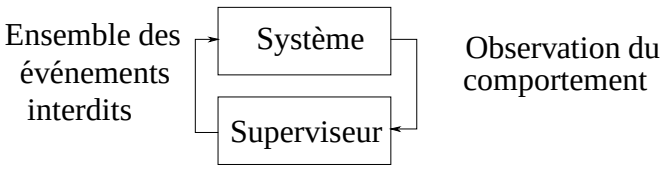


FIG. 1.1: Action du superviseur en boucle fermée

Le système composé de G sous l'action de S est appelé *système contrôlé* et noté S/G.
Les comportements du système contrôlé S/G sont modélisés par les langages $\mathcal{L}(S/G)$ et $\mathcal{L}_m(S/G)$, avec $\mathcal{L}(S/G)$ défini récursivement par

(1) $\epsilon \in \mathcal{L}(S/G)$
(2) $\forall s \in \mathcal{L}(S/G), \forall \sigma \in \Sigma$ tels que $s\sigma \in \mathcal{L}(G)$, $s\sigma \in \mathcal{L}(S/G) \Leftrightarrow \sigma \notin S(s)$

et $\mathcal{L}_m(S/G) = \mathcal{L}(S/G) \cap \mathcal{L}_m(G)$.

$\mathcal{L}_m(S/G)$ et $\mathcal{L}(S/G)$ sont simplement des sous-langages de $\mathcal{L}(G)$ et $\mathcal{L}_m(G)$ qui ne contiennent que les mots admis par le superviseur $\mathcal{S}$. Si $\overline{\mathcal{L}_m(\mathcal{S}/G)} = \mathcal{L}(\mathcal{S}/G)$, alors $\mathcal{S}$ est dit *non-bloquant*.

De plus, l'alphabet Σ est partitionné en deux ensembles disjoints Σ_c et Σ_{uc}, appelés respectivement ensemble des événements contrôlables et incontrôlables.

$$\Sigma = \Sigma_c \uplus \Sigma_{uc}$$

Intuitivement, les événements incontrôlables représentent les événements dont l'occurrence ne peut être empêchée par un superviseur. Ces événements peuvent servir à modéliser par exemple des événements de pannes, des données capteurs ou encore des *tick* d'horloge (Cf. [13] pour plus de détails).

Compte tenu du caractère incontrôlable de certains événements de Σ, le langage généré par un système contrôlé ne peut être quelconque. C'est dans cette optique que nous introduisons la notion de contrôlabilité d'un langage. Celle-ci permet de caractériser les langages générés par des systèmes contrôlés.

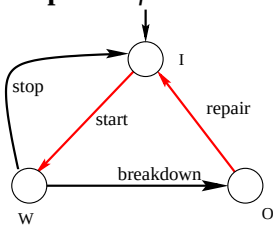
Définition 6 ([55]) *Soit $G = (\Sigma, Q, q_o, Q_m, \delta)$ un système modélisé par un automate, et $K \subseteq \mathcal{L}(G)$. Soit $\Sigma_{uc} \subseteq \Sigma$. K est dit contrôlable par rapport à Σ_{uc} et $\mathcal{L}(G)$ si*

$$\overline{K}\Sigma_{uc} \cap \mathcal{L}(G) \subseteq \overline{K}$$

Le langage vide est supposé contrôlable par convention.

Intuitivement, si $\mathcal{L}(G)$ représente le comportement non contrôlé d'un système, alors K est contrôlable par rapport à Σ_{uc} et $\mathcal{L}(G)$ si tout mot contenu dans $\overline{K}$ qui peut être complété par un événement incontrôlable $\sigma_{uc} \in \Sigma_{uc}$ en un comportement admissible de G, est encore dans le langage $\overline{K}$.

Exemple 1 *Afin d'illustrer la définition 6, considérons l'exemple suivant :*



Si on suppose que $\Sigma_c = \{start,\ repair\}$, $\Sigma_{uc} = \{stop,\ breackdown\}$, alors
- *$K = \{start, start\ breackdown\}$ n'est pas contrôlable*
- *$K = \{start, start\ stop, start\ breakdown\}$ est contrôlable*

Le théorème 1 traduit qu'il existe effectivement un lien fort entre les sous langages contrôlables du système et les superviseurs qui peuvent y être appliqués.

Théorème 1 ([59]) *Soient G un système à contrôler et $\emptyset \subset K \subseteq \mathcal{L}(G)$. Il existe un superviseur $\mathcal{S}$ agissant sur G tel que $\mathcal{L}(S/G) = \overline{K}$ si et seulement si K est contrôlable par rapport à Σ_{uc}, $\mathcal{L}(G)$.* ◇

Ainsi, lorsque K est vu comme un objectif de contrôle qui représente le sous ensemble des comportements désirés de G, alors si K est contrôlable par rapport à Σ_{uc} et $\mathcal{L}(G)$, il est possible de définir un superviseur restreignant exactement les comportements de G à ceux de K.

Dans la plupart des situtations, l'objectif de contrôle considéré K n'est pas contrôlable par rapport à Σ_{uc} et $\mathcal{L}(G)$. Dans ce cas, il apparaît intéressant de déterminer un sous langage de K qui vérifie cette propriété, induisant ainsi un superviseur S sur G. De plus, afin d'éviter de restreindre les comportements de G de manière inutile, le superviseur S doit permettre au système d'évoluer le plus librement possible. Ceci induit un critère de maximalité, à la fois sur les sous langages contrôlables du système, mais aussi sur les superviseurs qui peuvent s'y appliquer.

Langages contrôlables maximaux et superviseurs associés. Étant donné un langage préfixe-clos $L \subseteq \Sigma^*$ et $\Sigma_{uc} \subseteq \Sigma$, $\mathcal{C}(\Sigma_{uc}, L)$ représente l'ensemble des sous langages de L qui sont contrôlables par rapport à Σ_{uc} et L. Cet ensemble n'est pas vide dans la mesure où il contient L et le langage vide.

Maintenant, étant donné un langage $K \subseteq L$ (avec L préfixe-clos), comme évoqué dans le paragraphe précédent et illustré dans l'exemple 1, il se peut que K ne soit pas contrôlable par rapport à Σ_{uc} et L. Dans ce cas, on va s'intéresser aux sous (sur) langages de K qui sont contrôlables par

rapport à Σ_{uc} et L. On définit ainsi la classe des sous (sur) langages de K, contrôlables par rapport à Σ_{uc} et L :

$$\mathcal{C}(K, \Sigma_{uc}, L) = \left\{ M \subseteq K |\ \overline{M}\Sigma_{uc} \cap L \subseteq \overline{M} \right\} \tag{1.21}$$

$$\mathcal{C}^{\inf}(K, \Sigma_{uc}, L) = \left\{ M \subseteq \Sigma^* |\ K \subseteq M \subseteq L \wedge \overline{M} = M \wedge \overline{M}\Sigma_{uc} \cap L \subseteq \overline{M} \right\} \tag{1.22}$$

Proposition 4 ([79]) *Soit $K \subseteq L \subseteq \Sigma^*$, et $L = \overline{L}$*

- *L'ensemble $\mathcal{C}(\Sigma_{uc}, K, L)$ est non vide et stable par union.*
- *L'ensemble $\mathcal{C}^{inf}(K, \Sigma_{uc}, L)$ est non vide et stable par intersection* ◇

.

D'après la proposition 4, il existe un unique langage maximal, noté SupCont(K, Σ_{uc}, L) (ou $K^{\uparrow c}$ lorsqu'il n'y aura pas d'ambiguïté) dans $\mathcal{C}(K, \Sigma_{uc}, L)$, qui est contenu dans K et contrôlable par rapport à Σ_{uc} et L. Ce langage est défini par :

$$\text{SupCont}(K, \Sigma_{uc}, L) = \bigcup_{M \in \mathcal{C}(\Sigma_{uc}, K, L)} M \tag{1.23}$$

De même, d'après la proposition 4, il existe un unique langage minimal et préfixe-clos, noté InfCont(K, Σ_{uc}, L) (ou $K^{\downarrow c}$) qui contienne K et qui soit contrôlable par rapport à Σ_{uc} et L. De plus

$$\text{InfCont}(K, \Sigma_{uc}, L) = \bigcup_{M \in \mathcal{C}(\Sigma_{uc}, K, L), M = \overline{M}, K \subseteq M} M = \overline{K}.\Sigma_{uc}^* \cap L \tag{1.24}$$

De plus d'après le théorème 1, puisque SupCont(K, Σ_{uc}, L) est contrôlable, il existe donc des superviseurs qui restreignent le comportement du système à SupCont(K, Σ_{uc}, L). De tels superviseurs sont dits *maximaux,* car ils correspondent aux superviseurs restreignant le moins possible les comportements du système à ceux de l'objectif K.

Enfin, le résultat fournit par le lemme 2 se déduit aisément de la définition de la contrôlabilité et traduit que celle-ci est préservée lorsque l'on restreint les comportements du système à contrôler.

Lemme 2 *Soient K, L et L' des langages préfixes-clos sur Σ tels que $K \subseteq L' \subseteq L$. Soit $\Sigma_{uc} \subseteq \Sigma$.*

$$K \in \mathcal{C}(\Sigma_{uc}, L) \Longrightarrow K \in \mathcal{C}(\Sigma_{uc}, L')$$

D'après le lemme 2, si un langage K inclus dans un langage L' est contrôlable par rapport à un langage L contenant L', alors K est contrôlable par rapport à L'.

Lemme 3 ([13]) *Soient $K \subseteq L \in \mathcal{L}(\Sigma)$ préfixe-clos. Soit $\Sigma_{uc} \subseteq \Sigma$. SupCont$(K, \Sigma_{uc}, L)$ est également préfixe-clos.* ◇

Réalisation d'un superviseur. D'un point de vue théorique, si le langage $\mathcal{L}(G)$ modélise les comportements du système à contrôler G, alors un superviseur sur G est défini comme une fonction $S : \mathcal{L}(G) \rightarrow 2^{\Sigma}$. Toutefois, cette définition n'est en général pas exploitable d'un point de vue pratique. Il semble donc important de considérer une représentation adéquate d'un superviseur.

Lorsque le système G considéré est modélisé par un automate (ou que ses comportements sont modélisés par un langage régulier), et que l'objectif de contrôle est lui aussi modélisé par un automate H (tel que $\mathcal{L}(H) = K$), il est alors possible de représenter le superviseur S par un automate $\mathcal{R}_S$. Au cours de l'exécution de G, cet automate évolue en parallèle avec le système. En d'autres termes, le système contrôlé $S/G = G \parallel \mathcal{R}_S$. $\mathcal{R}_S$ restreint donc le comportement de G par synchronisation. Ainsi lors de l'évolution du système G, pour chaque état atteint, les événements tirables depuis cet état dans $\mathcal{R}_S$ sont ceux autorisés par le superviseur et donc tirables dans G.

Une telle représentation est appelée *réalisation du superviseur* dans [13] (et *superviseur* dans [79]) et correspond à ce qui doit être obtenu lorsqu'il est question de synthétiser un superviseur. En effet, un calcul hors-ligne d'une telle *réalisation du superviseur* assure que l'évaluation du superviseur S lui-même, qui devra être effectuée à l'exécution, ne nécessite alors plus qu'une lecture des événements à interdire/autoriser.

Étant donné deux automates G et H modélisant respectivement le système à contrôler et l'objectif de contrôle, des algorithmes sont fournis dans [79] et [13], et permettent de déterminer un automate dont le langage généré est celui du plus grand langage contrôlable inclus dans $\mathcal{L}(G)$ et $\mathcal{L}(H)$. Cet automate constitue une réalisation d'un superviseur maximal restreignant les comportements de G à ceux de l'objectif de contrôle, et est appelé *réalisation standard du superviseur*. Par conséquent, pour synthétiser un contrôleur S sur un système G, il est suffisant de déterminer une automate modélisant le langage du système contrôlé S/G.

Remarque 2 *S'il existe un unique comportement maximal du système, il n'en est pas de même de la réalisation du superviseur. Étant donné un système G et un objectif de contrôle, il existe en général plusieurs réalisations (i.e. plusieurs automates modélisant le superviseur). L'idée est que le superviseur incorpore des informations redondantes (par exemple sur les contraintes de transitions déjà définies par le système lui-même) et qu'il est donc possible de réduire cette information sans affecter le contrôle du système (modulo une relation appelé couverture de contrôle dans [76]). Ce problème a depuis été repris par [53] et [66]. Ces travaux peuvent avoir un impact important quant à l'implémentation réelle des superviseurs en e.g. PLC (programmable Logic Controllers) [28, 20, 42, 22].*

1.2.2 Problèmes classiques de synthèse

La restriction des comportements d'un système G par un superviseur S s'effectue suivant un objectif de contrôle. Dans la théorie de Ramadge et Wonham, celui-ci s'exprime par un langage. Différents problèmes de contrôle peuvent alors s'exprimer. Le problème basique consiste à restreindre les comportements de G afin que ceux du système contrôlé S/G soient inclus dans le langage représentant l'objectif de contrôle. Toutefois, une condition de non blocage sur S/G peut aussi être exigée. Nous rappelons ici qu'un système contrôlé S/G est non bloquant si

$$\overline{\mathcal{L}_m(S/G)} = \mathcal{L}(S/G)$$

Enfin, l'occurrence de certains événements de G peut ne pas être observable par le superviseur. Cette absence d'information nécessite de prendre des précautions lors de la synthèse du superviseur, et constitue un problème particulier dit *problème de contrôle sous observation partielle*.

1.2.2.1 Problème de base de la synthèse de contrôleurs

Le problème de base de la synthèse de contrôleurs (PBSC) est le suivant :

Problème : Étant donné un alphabet Σ, un ensemble d'événements incontrôlables Σ_{uc} et un langage $K \subseteq \mathcal{L}(G)$, où K est préfixe-clos, on veut déterminer un superviseur S tel que :
(1) $\mathcal{L}(S/G) \subseteq K$
(2) $\mathcal{L}(S/G)$ est maximal au sens où pour tout superviseur S' tel que $\mathcal{L}(S'/G) \subseteq K$, on ait

$$\mathcal{L}(S'/G) \subseteq \mathcal{L}(S/G)$$

•

Compte tenu de ce qui a été vu dans le paragraphe précédent, il suffit de déterminer un superviseur S tel que

$$\mathcal{L}(S/G) = \text{SupCont}(K, \Sigma_{uc}, \mathcal{L}(G)) \tag{1.25}$$

En particulier, un superviseur S, défini pour tout $s \in \text{SupCont}(K, \Sigma_{uc}, \mathcal{L}(G))$ par

$$S(s) = \{\sigma \in \Sigma |\ s\sigma \notin \text{SupCont}(K, \Sigma_{uc}, \mathcal{L}(G))\} \tag{1.26}$$

permet d'assurer la relation 1.25. Notons que les valeurs prises par S sur les séquences n'appartenant pas à $\text{SupCont}(K, \Sigma_{uc}, \mathcal{L}(G))$ n'ont pas d'influence sur la pertinence de S pour résoudre le PBSC. Par conséquent, $\text{SupCont}(K, \Sigma_{uc}, \mathcal{L}(G))$ permet de définir une fonction de supervision assurant l'objectif de contrôle.

Remarque 3 *Le PBSC est généralement énoncé pour un objectif de contrôle K tel que $K \subseteq \mathcal{L}(G)$. Toutefois, on peut noter que le PBSC peut s'étendre facilement au cas où $K \not\subseteq \mathcal{L}(G)$. En effet, si S est un superviseur solution du PBSC, alors $\mathcal{L}(S/G) \subseteq K$. Et puisque $\mathcal{L}(S/G) \subseteq \mathcal{L}(G)$, on en déduit que*
- *$\mathcal{L}(S/G) \subseteq K \cap \mathcal{L}(G)$*
- *$\mathcal{L}(S/G)$ est maximal au sens où pour tout superviseur S' tel que $\mathcal{L}(S'/G) \subseteq K \cap \mathcal{L}(G)$, on a $\mathcal{L}(S'/G) \subseteq \mathcal{L}(S/G)$.*

Par conséquent, on se ramène au problème précédent puisque $K \cap \mathcal{L}(G) \subseteq \mathcal{L}(G)$.

Algorithme et complexité D'un point de vue pratique, les langages modélisant les comportements du système à contrôler et de l'objectif sont modélisés par des automates. On note ainsi $G = (\Sigma, Q, q_0, Q_m, \delta)$ l'automate modélisant le système et $H = (\Sigma, Q', q'_0, Q'_m, \delta')$ l'automate modélisant l'objectif de contrôle (ici $Q_m = Q$ et $Q'_m = Q'$). On décrit à présent les différentes étapes du calcul permettant d'obtenir une réalisation d'un superviseur résolvant le PBSC (voir e.g. [38] pour plus de détails).

(a) Calcul de l'automate G' obtenu par composition entre G et H. On note $G' = (\Sigma, Q \times Q', (q_0, q'_0), Q_m \times Q'_m, \delta')$.

(b) Détermination de l'ensemble E des états G' qui violent le critère de contrôlabilité :

$$E = \{(q, q') \in Q \times Q' | \ \exists \sigma \in \Sigma_{uc}, \delta'((q, q'), \sigma)! \text{ et } \neg(\delta(q, \sigma)!)\}$$

(c) Détermination de l'ensemble E' des états menant à E dans G' par tirage d'une séquence d'événements incontrôlables :

$$E' = \mathit{CoReach}^{G'}_{\Sigma_{uc}}(E)$$

(d) Si $(q_0, q'_0) \in E'$, alors le PBSC n'a pas de solution. Sinon on considère l'automate $(\Sigma, (Q \times Q') \setminus E', (q_0, q'_0), Q_m \times Q'_m, \delta'')$ où pour tout $q \in (Q \times Q') \setminus E'$ et $\sigma \in \Sigma$,

$$\delta'(q, \sigma)! \text{ ssi } \delta'(q, \sigma)! \text{ et } \delta'(q, \sigma) \notin E'$$

et $\delta''(q, \sigma) = \delta'(q, \sigma)$ si $\delta''(q, \sigma)!$.

Cet automate est une réalisation d'un superviseur résolvant le PBSC et est appelé *réalisation standard*.

On s'intéresse maintenant à la complexité de l'algorithme précédent et on note N le nombre d'états de G et M le nombre d'états de H. Le nombre d'états de l'automate G' vaut alors $N.M$ dans le pire cas et représente aussi le nombre d'états de la *réalisation standard* du superviseur résolvant le PBSC dans le pire cas. De plus, le calcul de E' vaut alors $\mathcal{O}(|\Sigma|.N.M)$ dans le pire cas et représente la complexité en temps du calcul de cette réalisation.

1.2.2.2 Problème de la synthèse de contrôleurs non bloquant

Dans la plupart des situations, il semble également intéressant d'éviter les blocages dans le système contrôler. En d'autres termes, il est souhaitable de calculer un superviseur S, tel que $\mathcal{L}(S/G)$ soit non bloquant, ce qui signifie que $\mathcal{L}(S/G) = \overline{\mathcal{L}_m(S/G)}$ $\left(= \overline{\mathcal{L}(S/G) \cap \mathcal{L}_m(G)}\right)$. Dans un premier temps, nous introduisons la notion de L_m-clôture.

Définition 7 *Soient K et L_m deux langages sur Σ. K est L_m-clos si on a*

$$K = \overline{K} \cap L_m$$

À partir de cette notion, on peut donner une condition d'existence d'un superviseur telle que le langage généré par le système contrôlé soit non bloquant.

Théorème 2 ([59]) *On considère un système G ayant pour alphabet Σ, et on considère l'ensemble des événements incontrôlables $\Sigma_{uc} \subseteq \Sigma$. On considère le langage $K \subseteq \mathcal{L}_m(G)$, avec $K \neq \emptyset$. Il existe un* superviseur non bloquant *S sur G tel que*

$$\mathcal{L}_m(S/G) = K \text{ et } \mathcal{L}(S/G) = \overline{K}$$

si et seulement si les deux conditions suivantes sont vérifiées :
- *Contrôlabilité : $\overline{K}\Sigma_{uc} \cap \mathcal{L}(G) \subseteq \overline{K}$*
- *$\mathcal{L}_m(G)$-clôture : $K = \overline{K} \cap \mathcal{L}_m(G)$*

On introduit maintenant le problème de la synthèse de superviseurs non bloquants (PSCNB).

Problème : Étant donné un alphabet Σ, un ensemble d'événements incontrôlables Σ_{uc} et un langage $K \subseteq \mathcal{L}_m(G)$, où K est supposé être $\mathcal{L}_m(G)$-clos, on veut déterminer un superviseur non-bloquant S tel que :

(1) $\mathcal{L}_m(S/G) \subseteq K$

(2) $\mathcal{L}_m(S/G)$ est maximal au sens où pour tout superviseur non-bloquant S' tel que $\mathcal{L}_m(S'/G) \subseteq K$, on ait $\mathcal{L}_m(S'/G) \subseteq \mathcal{L}_m(S/G)$ •

Pour qu'un superviseur S soit solution de ce problème, il faut que $\mathcal{L}_m(S/G)$ soit à la fois contrôlable et $\mathcal{L}_m(G)$-clos. Le lemme qui suit a pour but de mettre en évidence que la préfixe-clôture d'un langage K est préservée par calcul du plus grand langage contrôlable rapport à un alphabet Σ_{uc} et un langage préfixe-clos L', qui soit inclus dans K.

Lemme 4 ([79]) *Soient $\Sigma_{uc} \subseteq \Sigma$ deux alphabets. Soient $K \subseteq L \subseteq \Sigma^*$. Si K est préfixe-clos (i.e $K = \overline{K}$), alors SupCont(K, Σ_{uc}, L') est préfixe-clos.*

Ainsi, la préfixe-clôture est stable par application de l'opérateur SupCont$(\bullet, \Sigma_{uc}, \mathcal{L}(G))$, i.e si $K \subseteq \mathcal{L}_m(G)$ est préfixe-clos, alors SupCont$(K, \Sigma_{uc}, \mathcal{L}(G))$ l'est aussi. Or on suppose dans la formulation du problème que K est $\mathcal{L}_m(G)$-clos. On en déduit donc qu'un superviseur S tel que $\mathcal{L}_m(S/G) =$ SupCont$(K, \Sigma_{uc}, \mathcal{L}(G))$ est solution du NBSCP.

Remarque 4 *Si K n'est pas $\mathcal{L}_m(G)$-clos, alors une solution maximale du NBSCP lorsque l'objectif de contrôle est $\overline{K} \cap \mathcal{L}_m(G)$ est aussi une solution maximale du NBSCP lorsque l'objectif de contrôle est K. En effet, $\overline{K} \cap \mathcal{L}_m(G)$ est le plus grand sous langage de K qui soit $\mathcal{L}_m(G)$-clos.*

Algorithme et complexité

Les langages modélisant les comportements du système à contrôler et de l'objectif de contrôle sont respectivement modélisés par les automates $G = (\Sigma, Q, q_0, Q_m, \delta)$ et $H = (\Sigma, Q', q'_0, Q'_m, \delta_H)$ (ici $Q_m = Q$). On décrit à présent les différentes étapes du calcul permettant d'obtenir une réalisation d'un superviseur résolvant le PSCNB. Tout d'abord, on applique le point (a) de l'algorithme du PSCB et l'automate obtenu est noté G'. Cet automate possède des états qui violent la propriété de contrôlabilité. Les étapes (b), (c) et (d) de l'algorithme du PSCB permettent de détecter et d'éliminer ces états. Toutefois, certains états de l'automate ainsi obtenu peuvent ne pas être coaccessibles. L'opérateur *Trim* peut alors lui être appliqué. Cependant de nouveaux états violant la contrôlabilité peuvent apparaître. L'algorithme du PSCNB consiste alors à alterner les étapes (b), (c), (d) de l'algorithme du PSCB avec l'application de l'opérateur *Trim*, jusqu'à stabilisation.

On s'intéresse maintenant à la complexité de l'algorithme précédent et on note N le nombre d'états de G et M le nombre d'états de H. Le nombre d'états de l'automate G' vaut alors $N.M$ dans le pire cas et représente aussi le nombre d'états de la *réalisation standard* du superviseur résolvant le PBSC dans le pire cas. De plus, la complexité en temps du calcul de E' vaut alors $\mathcal{O}(|\Sigma|.N.M)$ dans le pire cas. L'application de l'opérateur *Trim* s'effectuant en temps linéaire, la complexité de l'algorithme précédent vaut $\mathcal{O}(|\Sigma|.(N.M)^2)$ dans le pire cas.

1.2.2.3 Supervision sous observation partielle

Jusqu'à présent, nous avons supposé que le superviseur avait une vision parfaite du système. Celui-ci observait en effet tous les événements qui se produisaient dans le système. Toutefois dans de nombreux systèmes, de nombreux facteurs (absence de capteurs, caractère abstrait de la modélisation, délocalisation de certains sous-systèmes) font que cette situation s'avère être un cadre idéal. Pour pallier à ce problème, de nombreux travaux sur la supervision de systèmes à événement discrets sous observation partielle ont été réalisés [49, 62, 17, 16, 70, 68].

Dans ce cadre, le superviseur est non seulement incapable d'empêcher l'occurrence de certains événements (les incontrôlables), mais il est aussi incapable de tous les observer au cours de l'exécution. L'alphabet du système se décompose alors de la manière suivante : $\Sigma = \Sigma_o \cup \Sigma_{uo}$ où

- Σ_o représente l'ensemble des événements observables de G. Ce sont les événements qui sont visibles par le superviseur.
- Σ_{uo} représente l'ensemble des événements inobservables de G. Ce sont les événements dont l'occurrence ne peut être perçue par le superviseur (pannes non modélisées par des capteurs, événements internes, etc).

Le contrôle par boucle fermée sous observation partielle est donc maintenant décrit par la figure 1.2. Le superviseur ne peut prendre sa décision qu'en fonction des événements qu'il lui

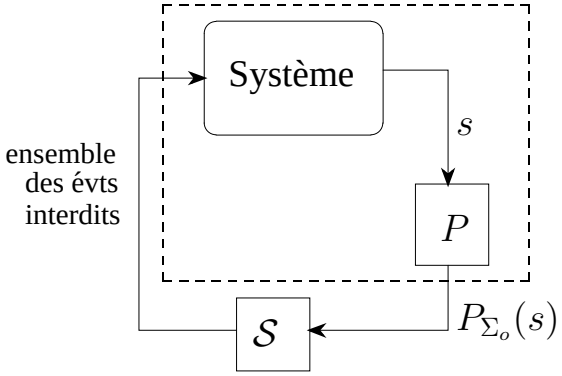


FIG. 1.2: Action du superviseur en boucle fermée sous observation partielle

est possible de voir. Par conséquent, le domaine d'un superviseur devient $P_{\Sigma_o}(L)$. De ce fait, le superviseur ne peut distinguer deux séquences s_1 et s_2 qui ont la même projection, i.e. $P_{\Sigma_o}(s_1) = P_{\Sigma_o}(s_2)$. Pour ces deux traces, le superviseur doit donc prendre la même décision de contrôle, alors même que cette décision d'interdiction de certains événements serait éventuellement différente sous observation totale. Cette particularité induit donc des contraintes sur le système contrôlé.

Afin de caractériser les langages générés par des systèmes contrôlés subissant l'action d'un superviseur agissant sous observation partielle, on introduit la notion d'observabilité [49].

Définition 8 *Soient K et L $(= \overline{L})$ deux langages sur un alphabet Σ et Σ', $\Sigma'' \subseteq \Sigma$. Soit $P_{\Sigma'} : (\Sigma')^* \longrightarrow \Sigma^*$ la projection naturelle de $(\Sigma')^*$ dans Σ^*. K est observable par rapport à $P_{\Sigma'}$, Σ'' et L si*

$\forall s \in \overline{K}$, $\forall \sigma \in \Sigma''$ *tels que* $s\sigma \in L$ *et* $\forall s' \in \overline{K}$ *tel que* $P_{\Sigma'}(s') = P_{\Sigma'}(s)$,

$$s'\sigma \in \overline{K} \Longrightarrow s\sigma \in \overline{K}$$

Un superviseur agissant sous observation partielle ne peut prendre en compte que le comportement visible du système pour agir. En particulier, son action doit être la même lorsqu'il observe un comportement pouvant provenir de deux comportements différents. Les comportements ainsi générés par le système contrôlé sont observables. Par conséquent, l'observabilité d'un langage est un critère important et nécessaire pour décrire les comportements d'un système contrôlé sous observation partielle [49, 59, 13].

Théorème 3 *Soit G le système à contrôler ayant pour alphabet Σ. Soient $\Sigma_{uc} \subseteq \Sigma$ les événements incontrôlables et $\Sigma_o \subseteq \Sigma$ les événements inobservables. Soit $K \subseteq \mathcal{L}_m(G)$, avec $K \neq \emptyset$ l'objectif à assurer. Il existe un superviseur non bloquant $S : P_{\Sigma_o}(\mathcal{L}(G)) \longrightarrow 2^{\Sigma}$ sur G tel que*

$$\mathcal{L}_m(S/G) = K \text{ et } \mathcal{L}(S/G) = \overline{K}$$

si et seulement si les conditions suivantes sont vérifiées :

- *Observabilité : K est observable par rapport à P_{Σ_o}, Σ_c, et $\mathcal{L}(G)$.*
- *Contrôlabilité : $\overline{K}\Sigma_{uc} \cap \mathcal{L}(G) \subseteq \overline{K}$.*
- *$\mathcal{L}_m(G)$-clôture : $K = \overline{K} \cap \mathcal{L}_m(G)$.*

On introduit maintenant le problème classique de la synthèse de superviseurs sous observation partielle.

Problème : Soit G le système à contrôler ayant pour alphabet Σ. Soient $\Sigma_{uc} \subseteq \Sigma$ les événements incontrôlables et $\Sigma_o \subseteq \Sigma$ les événements inobservables. Soit $K \subseteq \mathcal{L}_m(G)$, avec $K \neq \emptyset$ l'objectif à assurer tel que K est $\mathcal{L}_m(G)$-clos. Le problème est de synthétiser un superviseur non-bloquant $S : P_{\Sigma_o}(\mathcal{L}(G)) \longrightarrow 2^{\Sigma}$ tel que :

(1) $\mathcal{L}_m(S/G) \subseteq K$

(2) $\mathcal{L}_m(S/G)$ est maximal au sens où pour tout superviseur non-bloquant S' tel que $\mathcal{L}_m(S'/G) \subseteq K$, on ait $\mathcal{L}_m(S'/G) \subseteq \mathcal{L}_m(S/G)$. •

La différence entre le problème de la synthèse sous observation partielle et le NBSCP, réside donc dans la restriction du domaine de la fonction S à déterminer. Cette différence a des conséquences importantes puisque d'après le théorème 3, $\mathcal{L}(S/G)$ doit être maximal et observable par rapport à P_{Σ_o}, Σ_c et $\mathcal{L}(G)$. Or la condition d'observabilité n'est pas stable par union, ce qui implique la non existence d'une solution à ce problème, en général. Toutefois, lorsque $\text{SupCont}(K, \Sigma_{uc}, L)$ est observable par rapport à P_{Σ_o}, Σ_c et L, alors une fonction de supervision $S : P_{\Sigma_o}(L) \longrightarrow 2^{\Sigma}$ telle que $\mathcal{L}_m(S/G) = \text{SupCont}(K, \Sigma_{uc}, L)$ est une solution au problème de la synthèse sous observation partielle.

Afin de pallier à l'absence de maximalité due à l'observabilité, on introduit maintenant la notion de *normalité* [7, 39], qui s'avère être une condition plus forte que l'observabilité mais qui est stable par union.

Définition 9 *Soit K et L deux langages sur Σ avec L préfixe-clos. K est normal par rapport à P_{Σ_o} et L si*

$$\overline{K} = P_{\Sigma_o}^{-1}(P_{\Sigma_o}(\overline{K})) \cap L$$

Si un langage est normal par rapport à P_{Σ_o} et $\mathcal{L}(G)$, alors il est possible de montrer qu'il est observable par rapport à P_{Σ_o}, Σ_c et $\mathcal{L}(G)$ [7]. De plus, la condition de normalité est stable par union, donc il existe un plus grand langage normal par rapport à P_{Σ_o} et L qui soit inclus dans un langage K, et celui-ci peut être calculé. Par conséquent le calcul du plus grand langage qui soit à la fois contrôlable, normal et inclus dans un langage K peut être effectué.

Enfin, des conditions assurant l'équivalence entre observabilité et normalité sont connues. Ainsi, si $\Sigma_{uo} \subseteq \Sigma_{uc}$, alors la normalité par rapport à P_{Σ_o} et $\mathcal{L}(G)$ équivaut à l'observabilité par rapport à P_{Σ_o}, Σ_c et $\mathcal{L}(G)$. Par conséquent, on en déduit qu'il existe une solution au Problème de la Synthèse sous Observation Partielle lorsque $\Sigma_{uo} \subseteq \Sigma_{uc}$. Il suffit de calculer le plus grand langage qui soit à la fois contrôlable et normal [39].

Remarque 5 *Quelques remarques peuvent être faites concernant la résolution du problème de la synthèse sous observation partielle.*

- *Tout d'abord, il existe un algorithme polynomial permettant de tester si un langage est observable par rapport à P_{Σ_o} et L ([72]). Par conséquent, étant donné un objectif de contrôle modélisé par un automate, il est possible de déterminer si celui-ci est contrôlable et observable, et par conséquent d'obtenir une réalisation d'un superviseur résolvant le problème de la synthèse de contrôleurs sous observation partielle.*
- *Lorsque les événements inobservables sont incontrôlables, il y a équivalence entre les notions d'observabilité et de normalité. De plus, étant donné un langage généré par un automate, il est possible de calculer le plus grand langage normal inclus dans ce langage*[2] *([13]). Dès lors, il est possible de calculer (de manière itérative) une solution au problème de la synthèse sous observation partielle.*
- *Lorsque les événements inobservables ne sont pas tous incontrôlables, il n'y a pas d'équivalence entre les notions d'observabilité et de normalité. Dans ce cas, comme précédemment, étant donné un langage généré par un automate, il est possible de calculer le plus grand langage normal inclus dans ce langage. Toutefois, ceci ne permet pas de résoudre le problème de la synthèse de contrôleurs sous observation partielle. En effet, la solution ainsi obtenue n'est pas maximale en général. Il est cependant possible de calculer des langages observables au moins aussi permissifs que le plus grand langage normal et contrôlable ([68]).*
- *Enfin, notons qu'il existe une version plus générale du problème de contrôle sous observation partielle. Dans [5] par exemple, l'observation partielle est exprimée en termes de masquage : d'une part tous les événements ne peuvent être observés, et d'autre part certains événements ne peuvent être distingués lorsqu'ils sont observés.*

1.2.2.4 Approche états

Comme nous l'avons vu dans les sections précédentes, la synthèse de superviseur est effective lorsque les systèmes et les objectifs de contrôle sont modélisés par des automates. La méthode

[2] suivant une projection P_{Σ_o} et un langage L donnés

proposée consiste alors à réaliser le produit entre le système et l'objectif, puis à identifier un ensemble de "mauvais états" qui violent la condition de contrôlabilité. Ces états sont alors rendus, par contrôle, non atteignables par tirage d'une séquence d'événements incontrôlables. Puisque les systèmes sont en pratique représentés par un automate, il peut apparaître pertinent de considérer que l'ensemble des comportements que l'on souhaite assurer correspond à l'ensemble des comportements qui mènent à un ensemble d'états donnés. Dans ce cas, il paraît également judicieux de spécifier l'objectif de contrôle par l'ensemble des états dont on souhaite assurer l'invariance, plutôt que par l'ensemble des comportements qui permettent d'atteindre ces états. Cette approche, appelée *approche état*, bien qu'étant un cas particulier de l'approche classique, dite *approche langage*, a donné lieu à des études spécifiques initiées dans [58]. Le problème consiste à contrôler le système de manière à ce que celui-ci évolue dans un ensemble de bons états (ou de manière duale, n'atteigne pas un ensemble d'états interdits). En pratique, ce problème est souvent exprimé par l'intermédiaire de prédicats sur les états du systèmes. Le problème est alors de forcer le système à rester dans les états qui satisfont ces prédicats (pour l'invariance) ou satisfont la négation des ces prédicats (pour le problème d'interdiction) [58, 75, 82]. Dans la suite de cette section, nous nous focalisons sur le problème de l'interdiction d'états et nous supposerons que l'ensemble d'états à interdire est une donnée du problème.

Dans le cadre de l'interdiction d'états, il est possible de montrer que le superviseur n'a pas besoin de mémoire. En d'autres termes, l'état courant du système lui est suffisant pour prendre sa décision de contrôle. Dans ce cadre, un superviseur est donc simplement une fonction

$$S:\ Q \longrightarrow 2^{\Sigma_c}.$$

Intuitivement, pour tout état $q \in Q$, $S(q)$ représente un ensemble d'événements à interdire. Le problème du contrôle est alors le suivant :

Problème : Étant donné $E \subseteq Q$, on souhaite déterminer, lorsqu'il existe, un superviseur qui assure l'interdiction de E d'une manière maximale (i.e en permettant un maximum de comportements).

Afin de donner une solution au problème de l'interdiction d'états, on introduit l'opérateur $\mathcal{I}$ suivant.

Définition 10 *Soit G un automate et $E \subseteq Q$ un ensemble d'états de G. Alors $\mathcal{I}:\ 2^Q \longrightarrow 2^Q$ est la fonction définie pour tout E par :*

$$\mathcal{I}(E) = CoReach^G_{\Sigma_{uc}}(E) \tag{1.27}$$

Intuitivement, $\mathcal{I}(E)$ représente l'ensemble des états à partir desquels il est possible d'atteindre E en ne tirant que des événements incontrôlables. D'un point de vue contrôle, si l'un des ces états est atteint, alors il ne sera plus possible pour un superviseur d'empêcher que les états de E soient atteints.

On donne aussi quelques propriétés classiques de l'opérateur $\mathcal{I}(\cdot)$ qui se déduisent immédiatement des propriétés de l'opérateur $Pre^G_{\Sigma_{uc}}(\cdot)$ (voir section 1.1.2).

Proposition 5 *Soient E, E' deux ensembles d'états du système G. Alors*

$$E \subseteq E' \Longrightarrow \mathcal{I}(E) \subseteq \mathcal{I}(E') \tag{1.28}$$

$$\mathcal{I}(E) \cup \mathcal{I}(E') = \mathcal{I}(E \cup E') \tag{1.29}$$

$$\mathcal{I}(\mathcal{I}(E)) = \mathcal{I}(E) \tag{1.30}$$

Le lemme suivant donne alors une condition nécessaire et suffisante pour l'existence d'un superviseur assurant l'interdiction d'un ensemble d'états $E \subseteq Q$.

Lemme 5 *Il existe un superviseur assurant l'interdiction de $E \subseteq Q$ si et seulement si $q_0 \notin \mathcal{I}(E)$.*

Parmi les superviseurs assurant l'interdiction de E, certains sont maximaux au sens où ils autorisent un maximum de comportements de G. Pour $E \subseteq Q$, on définit sur Q le superviseur S_E par :

$$S_E(q) = \{\sigma \in \Sigma_c |\ \delta(q,\sigma)! \wedge \delta(q,\sigma) \in \mathcal{I}(E)\} \tag{1.31}$$

Le superviseur S_E possède deux propriétés intéressantes. Tout d'abord, si $q_0 \notin \mathcal{I}(E)$, alors S_E assure l'interdiction de E sur G. De plus, S_E est un superviseur maximal. Ces propriétés sont formalisées par la proposition 6.

Proposition 6 *Supposons que $q_0 \notin \mathcal{I}(E)$. Le superviseur S_E défini en 1.31 est maximal et assure l'interdiction de E dans G.*

Réalisation d'un superviseur

- Lorsque le nombre d'états du système est fini et que $q_0 \notin \mathcal{I}(E)$, une réalisation du superviseur peut être donnée par l'automate

  $$(\Sigma, Q \setminus \mathcal{I}(E), q_0, Q_m \setminus \mathcal{I}(E), \delta')$$

  avec $\delta' :\ \Sigma \times Q' \to Q'$ et pour $(\sigma, q \in \Sigma \times Q)$, on a $\delta'(\sigma,q)! \Leftrightarrow \delta(\sigma,q)!$ et si $\delta'(\sigma,q)!$ alors $\delta'(\sigma,q) = \delta(\sigma,q)$.
- Lorsque le nombre d'états du système est infini, une évaluation en ligne du superviseur est suggérée dans [79]. Le superviseur considéré est donné en (1.31) et l'évaluation consiste alors principalement à déterminer l'appartenance d'un état à $\mathcal{I}(E)$. Toutefois, cette évaluation n'est pas effective en général puisque $\mathcal{I}(E)$ peut être infini. Des solutions utilisant la structure particulière de certains systèmes ont été développées (dans [45] par exemple) et seront introduites plus en détails par la suite.

1.3 Problèmes liés à la complexité des systèmes et des superviseurs

La théorie du contrôle développée par Ramadge et Wonham est soumise en pratique à un problème d'efficacité. En effet, la composition entre automates provoque une multiplication du nombre d'états : le produit entre un automate possédant n états et un automate possédant p états est un automate dont le nombre d'états peut atteindre $n \times p$. Cette augmentation du nombre d'états par composition pose deux problèmes.

- D'une part, le superviseur est obtenu à partir de la composition du modèle du système et de l'objectif de contrôle. Or, il est souhaitable d'implémenter les superviseurs les plus petits possibles.

– D'autre part, les systèmes complexes sont généralement obtenus par composition de sous systèmes. Il se peut ainsi que l'explosion combinatoire induite par la composition des sous systèmes rende la résolution du problème irréalisable en pratique.

Ces deux contraintes ont ainsi données lieu à l'étude de deux types de problèmes :

(a) comment exprimer un superviseur résolvant le problème, à partir de systèmes de taille raisonnable ?

(b) comment résoudre le problème de synthèse lorsque le système initial est complexe, sans le construire explicitement ?

Des éléments de réponse ont été donnés pour résoudre ces deux problèmes : synthèse modulaire, décentralisée, hiérarchique pour la question (a) et synthèse sur des Réseaux de Petri, systèmes à événements discrets vectoriels, systèmes structurés (notamment concurrents et/ou hiérarchiques) pour la question (b).

1.3.1 Structuration du superviseur

Résoudre les problèmes classiques de synthèse de contrôleurs consiste à déterminer un superviseur assurant que les comportements du système contrôlé qu'il induit sont inclus dans un objectif de contrôle donné. Toutefois, le système à contrôler ou l'objectif de contrôle peuvent posséder des caractéristiques amenant à considérer des solutions adaptées.

Ainsi, lorsque l'objectif de contrôle est donné par une conjonction d'objectifs à assurer, et que le système est décentralisé, il semble intéressant de déterminer des solutions qui prennent en compte ces spécificités. Par exemple, les travaux sur la synthèse modulaire et la synthèse décentralisée tirent respectivement partie de la nature de l'objectif et du système pour déterminer plusieurs superviseurs, plutôt qu'un unique superviseur. De même, lorsque l'objectif de contrôle modélise un comportement abstrait du système, il paraît intéressant d'abstraire ce dernier et de déterminer un superviseur à partir de ces abstractions. Bien que de telles démarches imposent des restrictions, elles permettent d'assurer un gain significatif en complexité, concernant la taille des réalisations des superviseurs obtenus.

1.3.1.1 Superviseurs modulaires

Dans les situations concrètes, il est généralement difficile de spécifier un objectif de contrôle comme un seul langage. Les comportements souhaités du système sont généralement spécifiés de manière composite. Ainsi, le problème de la synthèse modulaire consiste à résoudre un problème classique de synthèse de contrôleurs pour un même système et plusieurs objectifs de contrôle devant être assurés sur ce système. La méthode classique consiste à déterminer un unique objectif de contrôle par composition de l'ensemble des objectifs, afin de se ramener au cas classique exposé précédemment. Toutefois, il faut noter que le calcul du langage objectif résultant de la composition peut s'avérer trop complexe pour pouvoir traiter le problème efficacement.

Dans [57, 80], Ramadge & Wonham ont mis en évidence que sous certaines conditions, il est possible de résoudre séparément chacun des sous problèmes (un par objectif) de sorte que la composition de chacune des solutions soit identique à la précédente. Cette seconde approche fournit des superviseurs en théorie plus "petits" que celui obtenu par la première approche.

Formellement, Soit G le système à contrôler ayant $\mathcal{L}(G)$ et $\mathcal{L}_m(G)$ comme comportements. Soient $K_1, K_2 \subseteq \mathcal{L}_m(G)$ deux langages représentant deux objectifs de contrôle que l'on souhaite assurer sur G. Le problème est de synthétiser un superviseur non-bloquant et maximal assurant $K_1 \cap K_2$. Comme nous venons de l'évoquer, pour des raisons de complexité, il est préférable de résoudre séparément le PSCNB avec pour objectifs K_1 et K_2.

Pour simplifier, on suppose que K_1 et K_2 sont $\mathcal{L}_m(G)$-clos[3]. Par conséquent, pour $i \in \{1, 2\}$, SupCont(K_i, Σ_{uc}, L) est solution du PSCNB dont l'objectif de contrôle est K_i. Malheureusement,

$$\text{SupCont}(K_1, \Sigma_{uc}, L) \cap \text{SupCont}(K_2, \Sigma_{uc}, L)$$

n'est en général pas une solution du PSCNB[4] dont l'objectif est $K_1 \cap K_2$. En fait, la proposition 7 fournit une condition nécessaire et suffisante pour que cette propriété soit vérifiée. Cette proposition fait intervenir la notion de langages *non conflictuels*.

Définition 11 *Soient L_1 et L_2 deux langages sur un alphabet Σ. L_1 et L_2 sont non conflictuels si*

$$\overline{L_1 \cap L_2} = \overline{L_1} \cap \overline{L_2} \tag{1.32}$$

De manière informelle, deux langages L_1 et L_2 sont non conflictuels si toute séquence qui est un préfixe appartenant à la fois à L_1 et à L_2 peut être complétée par le même suffixe dans L_1 et L_2. De plus, on peut noter que deux langages préfixes-clos sont non conflictuels.

Proposition 7 *Soient K_1 et K_2 deux langages sur un alphabet Σ.*

$$\textit{SupCont}(K_1, \Sigma_{uc}, L) \cap \textit{SupCont}(K_2, \Sigma_{uc}, L) = \textit{SupCont}(K_1 \cap K_2, \Sigma_{uc}, L) \tag{1.33}$$

si et seulement si SupCont(K_1, Σ_{uc}, L) et SupCont(K_2, Σ_{uc}, L) sont non conflictuels.

La proposition 7 offre une condition nécessaire et suffisante pour traiter le PSCNB dont l'objectif de contrôle est $K_1 \cap K_2$, en traitant les PSCNB dont les objectifs de contrôle sont respectivement K_1 et K_2. Cela offre des perspectives de gain concernant la mémoire nécessaire pour représenter un superviseur assurant $K_1 \cap K_2$. En effet, cette approche permet d'obtenir des superviseurs (S_i) tels que $\mathcal{L}(S_i/G) = \text{SupCont}(K_i, \Sigma_{uc}, L)$ et dont la réalisation est plus "petite" en générale que celle d'un superviseur S tel que $\mathcal{L}(S/G) = \text{SupCont}(K_1 \cap K_2, \Sigma_{uc}, L)$. De plus, la "disjonction" de ces superviseurs[5] permet d'obtenir les mêmes comportements que ceux fournis par un unique superviseur. Ainsi, la réalisation standard du superviseur global interdit un événement σ après une séquence s si et seulement si la réalisation d'au moins un des superviseurs S_i interdit σ après s.

Malheureusement, la vérification de la condition 1.33 nécessite de calculer $K_1 \cap K_2$, rendant dans le pire cas l'approche inefficace durant la phase de synthèse. Cependant, la complexité en mémoire de la réalisation des superviseurs que l'on obtient par cette approche, est en générale plus faible que celle de la réalisation d'un unique superviseur.

[3] Si tel n'est pas le cas, il suffit de considérer $K_i' = \overline{K_i} \cap \mathcal{L}_m(G)$ comme objectifs de contrôle

[4] On rappelle qu'un superviseur maximal assurant un objectif K sur L peut être caractérisé par SupCont(K, Σ_{uc}, L)

[5] les superviseurs sont supposés ici renvoyer un ensemble d'événements à interdire.

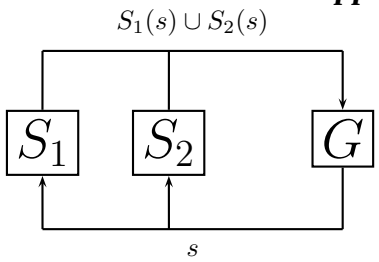


FIG. 1.3: Principe de la synthèse modulaire

1.3.1.2 Superviseurs décentralisés

Comme nous l'avons vu pour le problème de l'observation partielle, certains systèmes ne permettent pas de supposer que tous les événements qui s'y produisent sont observables. Ceci permet notamment de modéliser que le superviseur ne peut être couplé au système que par certains points d'entrées. De plus, il se peut que le système considéré soit par nature décentralisé et donc vu comme une collection de sites. Il semble alors pertinent de chercher à calculer des superviseurs pour chacun de ces sites, plutôt que de briser la nature décentralisée du système en le couplant avec un unique superviseur centralisé.

Étant donné un système dont on ne possède que différentes visions partielles de ses comportements, on cherche à déterminer un superviseur pour chacune de ces visions, de sorte qu'un objectif de contrôle global soit assuré. De manière formelle, on considère un système G dont les comportements sont modélisés par le langage généré $\mathcal{L}(G)$ et le langage marqué $\mathcal{L}_m(G)$ sur Σ. L'alphabet Σ est supposé contenir deux sous alphabets Σ_1 et Σ_2, et pour $i = 1, 2$, on note P_i les projections naturelles $P_i : \Sigma \longrightarrow \Sigma_i$. Étant donné un objectif de contrôle modélisé par un langage K sur Σ, le problème de la synthèse décentralisée consiste à déterminer, lorsqu'ils existent, deux superviseurs $S_1 : \Sigma_1 \longrightarrow 2^{\Sigma}$ et $S_2 : \Sigma_2 \longrightarrow 2^{\Sigma}$ tels que

$$\mathcal{L}((S_1 \vee S_2)/G) = \overline{K} \text{ et } \mathcal{L}_m((S_1 \vee S_2)/G) = K$$

où le superviseur $S_1 \vee S_2$ est défini pour tout $s \in L$ par $(S_1 \vee S_2)(s) = S_1(s) \cup S_2(s)$[6]. De nombreux travaux ont été menés sur ce sujet, parmi lesquels [17], [63], [48].

On peut remarquer que le problème de la synthèse décentralisée englobe celui de la synthèse sous observation partielle. Puisque le problème de la synthèse sous observation partielle n'admet pas de solution en général, il en est de même pour celui de la synthèse décentralisée. Dans le cadre de la synthèse sous observation partielle, la notion d'*observabilité* est introduite et représente la condition adéquate pour caractériser les solutions de ce problème. Dans le même esprit la notion de *coobservabilité* introduite dans [63] permet de caractériser les solutions du problème de contrôle décentralisé.

On considère ici un système G dont on possède différentes visions données par des systèmes G_1 et G_2 sur des alphabets respectifs Σ_1 et Σ_2. Pour $i = 1, 2$, Σ_i est partitionné suivant les

[6]On rappelle que les superviseurs fournissent ici un ensemble d'événements à interdire, plutôt qu'un ensemble d'événements autorisés.

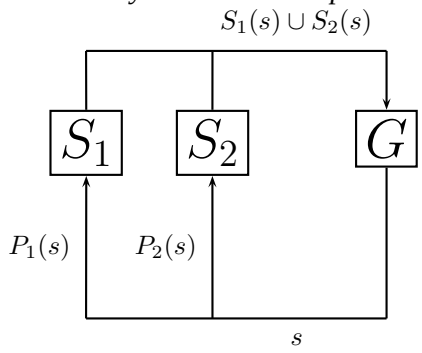


FIG. 1.4: Principe de la synthèse décentralisée

ensembles d'événements contrôlables $\Sigma_{i,c}$ et incontrôlables $\Sigma_{i,uc}$ d'une part, et les ensembles d'événements observables Σ_o et inobservables Σ_{uo} d'autre part. L'alphabet du système global G est noté Σ. Cet alphabet est partitionné suivant l'ensemble des événements contrôlables Σ_c et incontrôlables Σ_{uc} d'une part, et l'ensemble des événements observables Σ_o et inobservables Σ_{uo} d'autre part, tels que $\Sigma_c = \Sigma_{1,c} \cup \Sigma_{2,c}$ et $\Sigma_o = \Sigma_{1,o} \cup \Sigma_{2,o}$. On considère de plus les projections naturelles $P_1 : \Sigma^* \rightarrow \Sigma_{1,o}^*$ et $P_2 : \Sigma^* \rightarrow \Sigma_{2,o}^*$.

Définition 12 *Un langage K est dit coobservable par rapport à $\mathcal{L}(G)$, P_i et $\Sigma_{i,c}$ ($i = 1, 2$), si pour tout $s \in \overline{K}$ et tout $\sigma \in \Sigma_c$,*

$$(s\sigma \notin \overline{K}) \text{ et } (s\sigma \in \mathcal{L}(G)) \Rightarrow \exists i \in \{1,2\},\ P_i^{-1}(P_i(s)).\sigma \cap \overline{K} = \emptyset \text{ et } \sigma \in \Sigma_{i,c}$$

Intuitivement, la coobservabilité traduit le fait que si l'occurrence d'un événement doit être empêchée, alors au moins un des superviseurs qui en a la possibilité, le fera sans ambiguïté. Cette notion est la notion adéquate pour traiter le problème de la synthèse décentralisée. En effet, en reprenant les notations précédentes, il existe des superviseurs S_1 et S_2 tels que

$$\mathcal{L}((S_1 \vee S_2)/G) = \overline{K} \text{ et } \mathcal{L}_m((S_1 \vee S_2)/G) = K$$

si et seulement si

1 K est contrôlable par rapport à $\mathcal{L}(G)$ et Σ_{uc}.

2 K est coobservable par rapport à $\mathcal{L}(G)$, P_i et $\Sigma_{i,c}$ ($i = 1, 2$).

3 K est $\mathcal{L}_m(G)$-clos.

Finalement, un des intérêts d'obtenir une solution au problème du contrôle décentralisé réside dans la complexité des systèmes contrôlés obtenus. En effet, si l'objectif de contrôle vérifie les conditions précédentes, alors la réalisation de chaque superviseur S_i est en général moins complexe que celle d'un superviseur global.

1.3.1.3 Superviseurs hiérarchiques

Dans le même esprit, certains systèmes peuvent posséder une structure hiérarchique au sens où il en existe des visions plus ou moins abstraites. Le contrôle hiérarchique fournit des conditions suivant lesquelles il est possible de résoudre le problème de la synthèse de contrôleurs sur

une abstraction du système (obtenue par agrégation d'états), généralement moins complexe que le système lui-même [78, 12, 11, 15, 83]. Un des objectifs de cette étude consiste à utiliser l'architecture des systèmes à contrôler afin de diminuer la complexité des systèmes obtenus par contrôle. Dans [83], les auteurs introduisent le schéma de contrôle hiérarchique donné par la figure suivante.

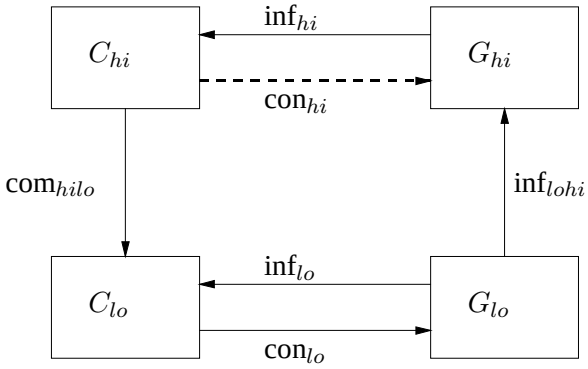


FIG. 1.5: diagramme informel sur la consistance hiérarchique

G_{lo} représente le système de bas niveau, qui est soumis au contrôle de C_{lo}, alors que G_{hi} représente une abstraction de G_{lo}. C_{hi} exerce un contrôle sur G_{hi}, à travers le canal con_{hi}, mais ceci n'est en fait que virtuel. En fait, C_{hi} communique avec C_{lo} par le canal com_{hilo} pour lui transmettre sa politique de contrôle. C_{lo} exerce alors son contrôle sur G_{lo} par con_{lo} qui dirige complètement son abstraction G_{hi} grâce au canal inf_{lohi}. Le retour d'information se faisant alors de G_{hi} vers C_{hi} par le canal inf_{hi}. G_{lo} peut donc être vu comme un modèle du système physique et G_{hi} comme une vision de plus haut niveau de ce système.

Plus formellement, L_{lo} désigne un langage sur un alphabet Σ et représente l'ensemble des comportements de G_{lo}. De même, L_{hi} désigne un langage sur un alphabet T et représente l'ensemble des comportements de G_{hi}. De plus, le canal inf_{lohi} est modélisé par une fonction $\theta : L_{lo} \rightarrow T^*$. Étant donné un objectif de contrôle de haut niveau modélisé par un langage $E_{hi} \subseteq T^*$, le but de cette étude est de déterminer sous quelles conditions le plus grand langage contrôlable inclus à la fois dans L_{hi} et E_{hi}, peut être assuré par contrôle du système G_{lo}. L'intérêt de cette approche est que sous ces conditions, il est possible de synthétiser un superviseur de haut niveau C_{hi} sur G_{hi}, sans manipuler le système de bas niveau G_{lo} qui est en général plus complexe.

Tout d'abord, une structure de contrôle associée à G_{lo} est induite de celle de G_{hi}. C'est à dire que l'alphabet T est supposé être partitionné en un ensemble d'événements contrôlables T_c et un ensemble d'événements incontrôlables T_{uc}, de sorte que l'interdiction d'événements contrôlables de haut niveau ait un sens au niveau le plus bas. Il est alors possible de définir $(E_{hi})^{\uparrow}$ qui représente le plus grand langage contrôlable inclus à la fois dans L_{hi} et E_{hi}. Dans [78], les auteurs s'intéressent aux systèmes hiérarchiques tels que l'image par θ de l'ensemble des sous langages contrôlables de L_{lo} soit égale à l'ensemble des sous langages contrôlables de G_{hi}. Cette propriété, appelée *consistance de contrôle* est en fait équivalente au fait que le système hiérarchique vérifie

$$\forall E_{hi} \subseteq T^*,\ \theta((\theta^{-1}(E_{hi}))^{\uparrow}) = (E_{hi})^{\uparrow}$$

Finalement, si G_{lo} et G_{hi} assure la *consistance de contrôle* et que E_{hi} représente un objectif de contrôle de haut niveau, alors il est possible de piloter G_{lo} par l'intermédiaire de C_{lo}, de sorte que le comportement de haut niveau obtenu soit $(E_{hi})^{\uparrow}$[7]. L'intérêt d'une telle approche réside dans la complexité du modèle utilisé. En effet, l'abstraction de haut niveau, G_{hi} est moins complexe que le système G_{lo} lui-même, et il est donc préférable de travailler sur cette abstraction.

1.3.2 Structuration des systèmes

Comme évoqué dans les paragraphes précédents, les approches décentralisée et hiérarchique sont liées à la nature du système à contrôler ou de l'objectif de contrôle. Elles permettent d'éviter de calculer une réalisation de superviseur possédant potentiellement un grand nombre d'états. Pour cela, plusieurs réalisations de moindre complexité sont envisagées dans le cadre modulaire et décentralisé d'une part, et une réalisation basée sur une abstraction du système dans le cadre hiérarchique d'autre part. Toutefois, bien que ces approches permettent une réduction significative des superviseurs synthétisés, elles ne permettent pas en général d'éviter les problèmes de complexités liés à la taille du système à contrôler.

Or, la plupart des systèmes réels sont obtenus par combinaison de sous systèmes, induisant ainsi une structure. Afin d'éviter une explosion combinatoire, en explicitant le système à contrôler de manière monolithique (i.e par un unique automate par exemple), il apparaît intéressant de travailler sur chacun des sous systèmes qui le composent. Ceci permet alors d'espérer un gain en complexité lors de la phase de synthèse de contrôleurs.

1.3.2.1 Réseaux de Pétri/Systèmes à Événements Discrets Vectoriels

On a vu dans le cadre de l'interdiction d'états, qu'une réalisation d'un superviseur ne peut être fournie en général, lorsque le nombre d'états du système est infini. Dans ce cas, les résultats du paragraphe 1.2.2.4 suggèrent une évaluation en ligne du superviseur donné en (1.31). Toutefois cette condition n'est pas suffisante pour permettre de résoudre effectivement le problème de l'interdiction d'états. En revanche, si le système possède certaines structures, alors les calculs peuvent devenir effectifs.

Ainsi, deux approches sur des types de systèmes très proches ont été développées. Les systèmes considérés sont les Systèmes à Événements Discrets Vectoriels (SEDV)/Réseaux de Petri, possédant une structure algébrique qui en permet une représentation finie. Les propriétés algébriques des SEDV/Réseaux de Petri permettent d'assurer l'effectivité des calculs évoqués précédemment. Des travaux sur les réseaux de Petri ont été développés dans [33, 81, 32, 34]. On donne à présent un aperçu de la théorie basique du contrôle sur les SEDV.

Dans [45, 46, 47, 14], Li a étudié le contrôle des *Systèmes à Événements Discrets Vectoriels (SEDV)*. Un SEDV est un SED dans lequel les états sont représentés par un vecteur d'entiers tandis que les transitions s'effectuent par des additions vectorielles sur les vecteurs d'états. La structure algébrique de ces systèmes est proche de celle des réseaux de Petri.

[7] Des résultats concernant des solutions non bloquantes sont aussi fournis dans [78].

Modélisation des SEDV : formellement, un SEDV est défini comme un quadruplet $(\Sigma,\ X,\ \delta,\ X_0)$, où Σ représente l'alphabet fini du système, X l'ensemble des états du système, X_0 l'ensemble des états initiaux du système, et δ la fonction partielle de transition.

Les états d'un SEDV sont décrits par des vecteurs d'entiers $x = (x_1, \ldots, x_n)$. L'espace d'états X est alors identifié à $(\mathbb{Z})^n$. La dynamique du système est décrite par la fonction partielle $\delta : \Sigma \times X \rightarrow X$ telle que $\delta(\sigma, x) = x + A_\sigma$, où $A_\sigma \in X$, est appelé le vecteur de déplacement de σ. Pour $\sigma \in \Sigma$ et $x \in X$, l'événement σ est tirable depuis x si et seulement si $\delta(\sigma,\ x) \geq 0$. Comme dans le cadre des machines à états finies, on écrira alors $\delta(\sigma, x)!$.

Une importante propriété du modèle est que l'état atteint après occurrence d'une suite d'événements ne dépend que du nombre d'occurrences de chaque événement dans cette suite et est indépendant de la manière dont les événements sont entrelacés. Plus précisément, soient $\Sigma = \{\sigma_1, \ldots, \sigma_m\}$ les événements d'un SEDV et $V_s = (V_s(1), \ldots, V_s(m))$ un vecteur d'entiers de dimension m défini pour une suite $s \in \Sigma^*$ par $V_s(i) =$ (nombre d'occurrences de σ_i dans la suite s). La fonction de transition sur un événement peut être étendue à une fonction de transition sur une suite d'événements de $\Sigma^* \times X$ dans X telle que pour tout $s \in \Sigma^*$ et pour tout état $x \in X$

$$\delta(s, x)! \Leftrightarrow (\forall t \in \overline{s})x + AV_t \geq 0$$

et

$$\delta(s, x) = x + AV_s$$

où $A = [A_{\sigma_1}, ..., A_{\sigma_m}] \in \mathbb{Z}^{n \times m}$ et $\overline{s}$ est l'ensemble de tous les préfixes de s.

Contrôle d'un SEDV Selon la théorie de Ramadge et Wonham, l'alphabet Σ du système est partitionné suivant un ensemble d'événements contrôlables Σ_c et incontrôlables Σ_{uc}. L'approche adoptée ici est basée sur les états et un superviseur sur un SEDV est alors une fonction $X \rightarrow 2^{\Sigma_c}$. Comme dans le paragraphe 1.2.2.4, l'objectif de contrôle est modélisé par un ensemble d'états dont on souhaite empêcher l'atteignabilité. Il s'agit donc d'un problème d'interdiction d'états, et l'ensemble d'états E à interdire est supposé ici être de la forme

$$E = \{x \in X | a.x \leq b\}$$

où $a \in (\mathbb{Z})^{1 \times n}$ et b est un entier. Par conséquent, l'ensemble d'états E à interdire est donné par une contrainte linéaire. En se référant au paragraphe 1.2.2.4, évaluer le superviseur standard assurant l'interdiction de E nécessite de pouvoir tester l'appartenance d'un état du système à l'ensemble $\mathcal{I}(E)$[8]. Puisque le système possède potentiellement un nombre infini d'états, il n'est pas possible en général d'effectuer ce test. Toutefois, Li et Wonham ont montré que si le système ne possède pas de boucle d'événements incontrôlables, alors le problème de tester l'appartenance d'un état à $\mathcal{I}(E)$ peut être ramené à un problème d'optimisation d'algèbre linéaire dont il est possible de déterminer une solution.

Par conséquent, les résultats donnés dans le paragraphe 1.2.2.4 sur l'approche états peuvent s'appliquer sur des systèmes possédant un nombre infini d'états. Cependant, pour rendre possible

[8] $\mathcal{I}(E)$ représente l'ensemble des états qui peuvent mener à E par tirage d'une séquence d'événements incontrôlables.

l'évaluation du superviseur standard assurant l'interdiction d'un ensemble d'états E, les systèmes considérés possèdent une forte structure algébrique. De plus, l'ensemble E possède lui-même une structure "adaptée" à celle du système.

1.3.2.2 Systèmes concurrents

Les systèmes concurrents sont des systèmes qui sont obtenus par combinaison de sous-systèmes interagissant entre eux. De tels systèmes sont généralement complexes au sens ou ils possèdent un grand nombre d'états.

Il apparaît alors intéressant de pouvoir effectuer la synthèse de contrôleurs sans calculer explicitement le système à contrôler et en tirant parti de sa structure. Dans ce cadre, étant donné un objectif de contrôle, le but consiste à dériver sur chacun des sous systèmes un superviseur assurant un certain objectif de contrôle, tel que la composition des sous systèmes contrôlés corresponde au système contrôlé global. Il apparaît qu'une telle approche ne fournit pas, en général, le résultat attendu. Des restrictions portant sur le système et/ou l'objectif de contrôle peuvent alors être envisagées pour parvenir à ce résultat. Parmi ces contraintes, on peut citer : la symétrie entre certains sous systèmes [60, 21], l'indépendance de certains sous systèmes [53], et surtout l'adéquation entre la structure de l'objectif et celle du système [77]. Ces travaux seront davantage introduits dans les chapitres suivants.

Chapitre 2

Contrôle de systèmes structurés : approche langage

Comme on l'a vu dans le chapitre 1, la complexité des systèmes actuels est trop importante pour que la théorie classique de la synthèse de contrôleurs développée par Ramadge et Wonham soit applicable en pratique. Deux problèmes principaux se posent : la taille des superviseurs synthétisés et la taille des systèmes à contrôler. Les travaux sur la synthèse décentralisée ([17], [63]), la synthèse modulaire ([57]), et la réduction de superviseurs [53] offrent des opportunités pour résoudre le problème lié à la taille des superviseurs synthétisés. Concernant la complexité trop importante des systèmes à contrôler, des travaux basés sur l'utilisation de la structure du système permettent sous certaines hypothèses de contourner le problème de complexité. En particulier, lorsque le système à contrôler est obtenu par composition de sous systèmes, les symétries du système ([21], [60]), l'indépendance entre sous systèmes ([19, 18, 20]), ou des objectifs adaptés à la structure du système ([77]) permettent de synthétiser des contrôleurs sur les sous systèmes plutôt que sur le système global, réduisant ainsi la complexité.

La problématique de ce chapitre consiste à apporter des solutions au *problème de base de la synthèse de contrôleurs* (PBSC) sur des systèmes de grande taille, modélisés à partir de sous systèmes. Plus précisément, on considère ici des systèmes concurrents, i.e obtenus par composition parallèle de sous systèmes.

Bien que dans la plupart des travaux précédemment cités, le problème traité sur ce modèle est plus général (le problème du non blocage de la solution est souvent traité), des contraintes sur le modèle ou la forme des objectifs sont introduites. L'objectif de l'étude effectuée dans ce chapitre consiste à obtenir une classe d'objectifs de contrôle assez générale pour laquelle la synthèse de contrôleurs peut être effectuée efficacement, en limitant les contraintes imposées au système. Pour cela, une approche permettant d'aborder le PBSC sur des systèmes concurrents est introduite dans ce chapitre.

Les approches adoptées dans ces circonstances sont le plus souvent dérivées de la synthèse décentralisée : l'objectif de contrôle est dérivé suivant la structure du système de manière à réduire le PBSC sur le système global en plusieurs PBSC sur chacun des sous systèmes. Par conséquent, c'est le système (ou plutôt sa structure) qui guide la démarche, et l'objectif de contrôle doit posséder les qualités lui permettant de s'adapter à celui-ci. Au contraire, dans l'approche pro-

posée dans ce chapitre, le PBSC sur le système global est ramené à plusieurs PBSC pour lesquels l'objectif de contrôle correspond à l'objectif sur le système global. En revanche, le système à contrôler pour chacun de ces PBSC est dérivé en utilisant la structure concurrente du système. Plus précisément, étant donné un objectif de contrôle, le but consiste à calculer un superviseur maximal (i.e le plus permissif) assurant cet objectif, sans construire explicitement le système à contrôler. Des approximations du système sont dérivées à partir des sous-systèmes qui le composent, et une propriété appelée *contrôlabilité partielle*, devant être vérifiée par l'objectif de contrôle sur ces approximations, est introduite. Assurer la contrôlabilité partielle de l'objectif de contrôle sur chacune des approximations permet, sous certaines hypothèses, d'en déduire un superviseur maximal assurant l'objectif de contrôle sur le système global. De plus, les calculs effectués ont une faible complexité et ne nécessitent pas de construire explicitement le système, évitant ainsi l'explosion combinatoire inhérente aux systèmes concurrents. Cette méthode offre ainsi un gain significatif en complexité, et s'applique pour des objectifs ne possédant pas nécessairement une structure similaire à celle du système.

Dans la suite de ce chapitre, nous commençons par introduire le formalisme utilisé pour modéliser les systèmes concurrents, ainsi qu'une présentation des travaux réalisés en synthèse de contrôleurs sur ce type de modèle. Dans un second temps, l'approche utilisée ici sera présentée ainsi que la notion centrale de ce chapitre : *la contrôlabilité partielle*. Étant donnés un système concurrent modélisé par un langage L sur un alphabet Σ, un ensemble d'événements incontrôlables Σ_{uc} et un objectif de contrôle K, la contrôlabilité partielle permet de calculer des sous langages de K contrôlables par rapport à Σ_{uc} et L. On verra alors que sous certaines hypothèses concernant l'objectif de contrôle K, il est possible d'obtenir le plus grand langage inclus dans K qui soit contrôlable par rapport à Σ_{uc} et L. De plus, une comparaison avec les travaux de [77] ainsi que la complexité obtenue par l'approche décrite ici seront fournies.

2.1 Modèle et Formulation du Problème

2.1.1 Modèle

Les systèmes étudiés ici sont dits *concurrents* au sens où ils sont obtenus par composition parallèle de sous systèmes. Dans la suite, on considère donc n systèmes modélisés par des automates $(G_i)_{1\leq i\leq n}$ avec $G_i = (\Sigma_i, Q_i, q_{0i}, \delta_i)$ pour $1 \leq i \leq n$. Le système à contrôler est alors modélisé par l'automate obtenu par composition parallèle des automates $(G_i)_{1\leq i\leq n}$. On note $G = (\Sigma, Q, q_0, \delta)$ cet automate et on a

$$G = G_1 \parallel \cdots \parallel G_n$$

Dans ce chapitre, plutôt que de manipuler les automates $(G_i)_{1\leq i\leq n}$ et G, on préférera utiliser les langages générés par ces automates. En particulier, on note

$$L_i = \mathcal{L}(G_i), \quad \text{pour } 1 \leq i \leq n$$
$$L = \mathcal{L}(G)$$

Basé sur cette notation, le comportement du système global G est donc donné par :

$$\mathcal{L}(G) = L = L_1 \parallel \cdots \parallel L_n = P_1^{-1}(L_1) \cap \cdots \cap P_n^{-1}(L_n)$$

où pour $i \in \{1, \ldots, n\}$, $P_i : \Sigma^* \longrightarrow \Sigma_i^*$ la projection naturelle de Σ sur Σ_i [1].

L'alphabet Σ de G est défini par $\Sigma = \bigcup_i \Sigma_i$. Finalement, l'ensemble des événements *partagés* par les systèmes $(G_i)_i$ (ou langages $(L_i)_i$[2]), noté Σ_s, est défini par l'ensemble

$$\Sigma_s = \bigcup_{i \neq j} (\Sigma_i \cap \Sigma_j) \tag{2.1}$$

Σ_s représente donc les événements de l'alphabet Σ de G qui sont partagés par au moins deux sous systèmes de G. Un événement qui n'est pas partagé (i.e appartenant à $\Sigma \setminus \Sigma_s$) est dit *local*.

Hypothèses de contrôlabilité. Pour chaque sous système G_i, certains événements de Σ_i sont incontrôlables $(\Sigma_{i,uc})$ alors que les autres sont contrôlables $(\Sigma_{i,c})$ et peuvent donc être empêchés sous l'action d'un superviseur.

La nature des événements de Σ est supposée être la même quel que soit le sous système G_i. En d'autres termes, un événement ne peut être contrôlable pour un des sous systèmes et incontrôlable pour un autre. Formellement,

$$\forall i, j \in \{1, \ldots, n\},\ \Sigma_{i,uc} \cap \Sigma_{j,c} = \emptyset \tag{2.2}$$

Les événements incontrôlables du système G sont alors donnés par $\Sigma_{uc} = \bigcup_i \Sigma_{i,uc}$, alors que les événements contrôlables de G sont donnés par $\Sigma_c = \bigcup_i \Sigma_{i,c}$.

On peut noter que l'hypothèse donnée par (2.2) est restrictive dès lors que l'on souhaite appliquer des superviseurs locaux à chacun des sous systèmes G_i. En effet, il est possible que l'occurrence de certains événements puisse être empêchée en agissant sur un sous système alors que cela est impossible depuis un autre sous système. Toutefois, la condition donnée par (2.2) est légitime lorsque le superviseur que l'on souhaite appliquer au système agit de manière global. En effet, dans ce cas, si l'occurrence d'un événement peut être empêchée depuis un des sous systèmes, elle peut être empêchée de manière globale, et l'événement correspondant peut alors être supposé contrôlable dans chacun des sous systèmes. Ainsi, les événements qui sont localement incontrôlables pour un sous système G_i sont ceux qui sont incontrôlables pour chacun des sous systèmes où ils interviennent.

2.1.2 Propriétés des systèmes concurrents

Dans ce paragraphe, on donne quelques résultats de base concernant les systèmes concurrents modélisés par une composition parallèle de langages. Puisque la composition parallèle de langages $(L_i)_i$, notée L, est définie comme l'intersection de la projection inverse de ces derniers, il semble notamment intéressant de faire le lien entre les séquences de L, L_i et $P_i^{-1}(L_i)$ pour $i \in \{1, \ldots, n\}$.

Le lemme 6 traduit le lien qui existe entre les événements de Σ_i qui se produisent dans le système modélisé par L et le sous système modélisé par L_i.

[1] voir paragraphe 1.1.1.2

[2] Par abus de langage, on parlera aussi des événements partagés par G ou L

Lemme 6 ([77]) *Soient $L = L_1 \parallel \cdots \parallel L_n$ avec $L_i \subseteq \Sigma_i^*$ préfixe-clos. Soit Σ_s l'ensemble des événements partagés de L. Soient $s \in L$, $i \in \{1, \ldots, n\}$ et $\sigma \in \Sigma_i \setminus \Sigma_s$. On a*

$$s\sigma \in L \Longleftrightarrow s\sigma \in P_i^{-1}(L_i)$$

L'implication $s\sigma \in L \Longrightarrow s\sigma \in P_i^{-1}(L_i)$ est toujours vérifiée par définition de la composition parallèle de langages. L'intérêt du lemme 6 réside donc dans l'implication inverse, qui traduit qu'un événement local au sous système G_i se produit indépendemment des autres sous systèmes.

Le lemme 7 donne une équivalence entre l'appartenance d'une séquence à la projection inverse d'un langage et celle de la même séquence subissant la projection d'un suffixe.

Lemme 7 *Soient $\Sigma_i \subseteq \Sigma$ deux alphabets et $L_i \in \mathcal{L}(\Sigma_i)$ un langage préfixe-clos. $P_i : \Sigma^* \longrightarrow \Sigma_i^*$ représente la projection naturelle de σ sur Σ_i. Soient $s \in P_i^{-1}(L_i)$ et $s' \in \Sigma^*$. On a*

$$ss' \in P_i^{-1}(L_i) \Longleftrightarrow sP_i(s') \in P_i^{-1}(L_i)$$

Démonstration :

$$\begin{array}{rcl} sP_i(s') \in P_i^{-1}(L_i) & \Longleftrightarrow & P_i(sP_i(s')) \in L_i \\ & \Longleftrightarrow & P_i(s).P_i(P_i(s')) \in L_i \\ & \Longleftrightarrow & P_i(s).P_i(s') \in L_i \ \ (\text{ car } P_i \circ P_i = P_i) \\ & \Longleftrightarrow & P_i(ss') \in L_i \\ & \Longleftrightarrow & ss' \in P_i^{-1}(L_i) \end{array}$$

◇

2.1.3 Synthèse de superviseurs sur des systèmes concurrents

La théorie de la synthèse de contrôleurs introduite par Ramadge et Wonham est confrontée à un problème d'efficacité algorithmique, lorsque les systèmes à contrôler sont complexes (i.e possèdent un grand nombre d'états). Les systèmes complexes sont généralement spécifiés par composition de sous systèmes. Il apparaît alors intéressant d'utiliser la structure d'un tel système pour tenter de réduire la complexité des calculs à effectuer. En d'autres termes, étant donné un objectif de contrôle modélisé par un langage K et un système concurrent G obtenu par composition parallèle de sous systèmes $(G_i)_{1 \leq i \leq n}$ modélisés par des langages $(\mathcal{L}_i)_{1 \leq i \leq n}$ sur lequel cet objectif doit être assuré, l'idée intuitive consiste à résoudre "localement" des problèmes de synthèse, afin de résoudre le problème global.

De manière générale, tous les travaux que nous avons pu trouver sur ce type de problème adoptent une approche décentralisée. L'idée consiste à calculer des superviseurs locaux pour chacun des sous-systèmes de manière à ce que la composition des sous-systèmes contrôlés soit égale au système contrôlé global. Toutefois on peut noter qu'il existe une autre approche dite centralisée donnant accès à un unique superviseur, synthétisé de manière modulaire en tirant parti de la structure du système. C'est cette dernière approche qui a été retenue dans le cadre de cette thèse. Avant de la présenter de manière plus approfondie, nous revenons dans un premier temps sur l'approche décentralisée.

2.1.3.1 L'approche décentralisée

De par leur nature concurrente, il est relativement intuitif d'utiliser une approche décentralisée pour la résolution de problèmes de contrôle sur les systèmes concurrents. Ainsi, dans [77], les auteurs s'intéressent à la résolution du PBSC sur des systèmes concurrents. L'approche adoptée est dérivée des travaux sur la synthèse décentralisée avec les sous systèmes G_i modélisant des visions partielles du système G (illustré par la figure 2.1.3.1).

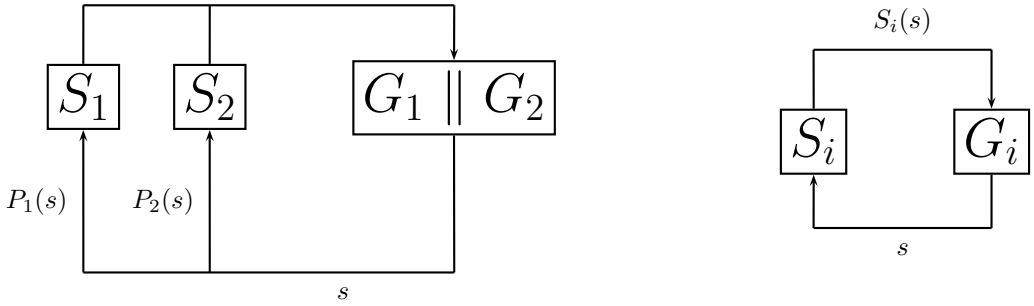


FIG. 2.1: Approche dérivée de la synthèse décentralisée.

Cependant, la structure particulière du système fournit une information supplémentaire. L'enjeu consiste alors à trouver des conditions selon lesquelles la composition parallèle des systèmes contrôlés S_i/G_i correspond au système contrôlé global le plus permissif assurant un objectif de contrôle donné. Malheureusement, cette approche ne permet pas en général de résoudre ce problème. Des conditions portant à la fois sur le système et sur l'objectif de contrôle sont nécessaires.

Puisque la démarche consiste à utiliser la structure du système lors de la synthèse, une contrainte portant sur les objectifs de contrôle considérés peut être exhibée. Les auteurs introduisent ainsi la notion de *langage séparable*, qui traduit qu'un langage peut être représenté par une composition parallèle de langages suivant un ensemble d'alphabets donné.

Définition 13 *Soient $\{\Sigma_i\}_{1\leq i\leq n}$ un ensemble d'alphabets et L un langage sur $\Sigma = \bigcup_i \Sigma_i$. L est séparable suivant $\{\Sigma_i\}_{1\leq i\leq n}$ si pour tout i, il existe des langages $L_i \subseteq \Sigma_i^*$ tels que*

$$\mathcal{L} = \|_{1\leq i\leq n}\, \mathcal{L}_i$$

De plus, les auteurs font l'hypothèse de contrôlabilité introduite en section 2.1.1 : si un événement est incontrôlable pour un sous système, alors il est incontrôlable pour chacun des sous systèmes. Sous cette hypothèse, la séparabilité de langages joue un rôle primordial pour assurer que la démarche adoptée est pertinente.

Théorème 4 *Soient un système concurrent $G = \|_{1\leq i\leq n}\, G_i$ et $K \subseteq \Sigma^*$ un objectif de contrôle. On note Σ l'alphabet de G, Σ_c et Σ_{uc} représentent respectivement l'ensemble des événements contrôlables et incontrôlables de G, et $\Sigma_s \subseteq \Sigma_c$ représente l'ensemble des événements partagés du système. Il existe des superviseurs locaux $(S_i)_{1\leq i\leq n}$ tels que*

$$\|_{1\leq i\leq n}\ (S_i/G_i) = \textit{SupCont}(\overline{K}\cap\mathcal{L}(G),\Sigma_{uc},\mathcal{L}(G))$$

si et seulement si $\textit{SupCont}(\overline{K}\cap\mathcal{L}(G),\Sigma_{uc},\mathcal{L}(G))$ *est séparable.* ◇

Le théorème 4 fournit une condition nécessaire et suffisante pour résoudre le PBSC sur le système global, à partir de calculs locaux. Cette condition correspond en fait à la séparabilité du plus grand langage contrôlable inclus dans $\overline{K}$ et $\mathcal{L}(G)$. Toutefois, vérifier cette condition nécessite de déterminer auparavant ce langage, ce qui revient à résoudre le problème initial. Le théorème 4 n'est donc pas complètement satisfaisant pour assurer un gain en complexité pour l'approche proposée. Cependant, les auteurs ont montré que si le l'objectif de contrôle est lui-même séparable, alors sous les hypothèses du théorème 4, SupCont$(\overline{K}\cap\mathcal{L}(G),\Sigma_{uc},\mathcal{L}(G))$ l'est aussi. Par conséquent, il suffit dans ce cas de vérifier que l'objectif de contrôle est séparable par rapport aux alphabets $\{\Sigma_i\}_{1\leq i\leq n}$ pour assurer la validité de la solution obtenue par des calculs locaux. La complexité d'une telle vérification est donnée par les auteurs et vaut $\mathcal{O}(m^{n+1})$ où m représente le nombre d'états de l'automate modélisant K.

Remarque 6 *Dans [36], les auteurs reprennent les résultats de [77] en affaiblissant les hypothèses du théorème 4. En effet, dans [77], les auteurs supposent d'une part que la nature contrôlable des événements est identique pour chacun des sous systèmes, et d'autre part que les événements incontrôlables ne sont pas partagés. En revanche, dans [36], bien que la première hypothèse soit conservée, les auteurs montrent que la seconde est superflue pour obtenir l'équivalence du théorème 4.*

Autres travaux Dans [18, 19, 20], [84, 3], les auteurs s'intéressent aussi à la synthèse de contrôleurs sur des systèmes concurrents modélisés par une composition parallèle de langages (automates). L'idée développée par les auteurs consiste à modéliser le système par une composition parallèle dont les opérandes sont des langages sur des alphabets deux à deux disjoints, se ramenant ainsi au cas des systèmes asynchrones (product system). Cette décomposition s'effectue selon l'alphabet de l'objectif de contrôle. Une telle approche trouve notamment son intérêt lorsque les objectifs de contrôle considérés ne concernent qu'un sous ensemble des sous systèmes du système concurrent. Lorsque le système est obtenu par composition asynchrone de sous systèmes et que l'objectif ne concerne qu'un seul de ces sous systèmes, la synthèse de contrôleurs peut alors s'effectuer localement.

De plus, les auteurs ont adapté l'approche au cas de la synthèse modulaire. L'objectif de contrôle K est alors modélisé par une intersection de langages $(K = \cap_i K_i)$. Dans ce cas, assurer que le superviseur obtenu soit maximal nécessite de vérifier que les superviseurs assurant les objectifs K_i génèrent des langages contrôlés qui soient non conflictuels[3]. Or la vérification de cette propriété nécessite en général d'effectuer la composition entre les différents systèmes contrôlés, rendant la phase de synthèse inefficace en pratique. La décomposition du système à contrôler sous la forme d'une composition asynchrone de sous systèmes (product system) permet de vérifier cette propriété de manière plus efficace. En effet, dans ce cas, il suffit de vérifier que les systèmes contrôlés des sous produits du système global sont non conflictuels, évitant ainsi le

[3] voir Définition 11.

calcul du produit global.

Dans [2, 1], les auteurs modélisent les systèmes concurrents G (sur un alphabet Σ) par une composition de sous systèmes $(G_i)_i$, contrainte par un système appelé *compensateur (compensator)* (G_c). Chaque sous système G_i est défini sur un alphabet Σ_i, G_c est défini sur l'alphabet Σ et le modèle ainsi obtenu est appelé *Interactive Discrete Event System (IDES)*.

$$G = (\|_i\ G_i) \parallel G_c$$

Étant donné un système quelconque, il est possible de calculer une représentation de celui-ci sous forme de IDES. En particulier, la représentation d'un système concurrent G sur un alphabet Σ obtenu par composition parallèle d'automates $(G_i)_i$ est immédiate sous forme de IDES :

$$\mathcal{L}(G) = (\|_i\ \mathcal{L}(G_i)) \parallel \Sigma^*$$

Les auteurs se sont intéressés aux problèmes de synthèse de contrôleurs sur des systèmes modélisés par des IDES. Étant donné un objectif de contrôle K, K est doté une structure d'IDES calquée sur celle du système :

$$K = (\|_{1\leq i\leq n}\ P_i(K)) \parallel K_c$$

où P_i est la projection naturelle de K suivant Σ_i et K_c est un compensateur. L'approche consiste alors à résoudre le problème composant par composant, i.e pour chaque $i \in \{1, \ldots, n\}$ les auteurs cherchent à résoudre le problème local de synthèse où G_i représente le système à contrôler et K_i l'objectif de contrôle. Les auteurs ont montré que si $\Sigma_s \subseteq \Sigma_c$, K est contrôlable par rapport à Σ_{uc} et $\|_i\ \mathcal{L}(G_i)$, et si $K_c\Sigma_{uc} \subseteq K_c$, alors

$$\text{SupCont}(K \cap \mathcal{L}(G), \mathcal{L}(G), \Sigma_{uc}) = (\|_{i\leq n}\ \text{SupCont}(P_i(K), \mathcal{L}(G_i), \Sigma_{i,uc})) \cap \text{SupCont}(K_c, \Sigma_{uc}, G_c)$$

Toutefois, la condition $K_c\Sigma_{uc} \subseteq K_c$ implique que les objectifs de contrôle doivent permettre à tout événement incontrôlable de se produire, quel que soit le comportement passé du système. Cette condition simplifie donc les difficultés introduites par la nature incontrôlable des événements et permet ainsi de résoudre le PBSC à partir de calculs locaux.

2.1.3.2 Approche centralisée : formulation du problème

Les différents travaux que nous venons d'évoquer se caractérisent par une décomposition de l'objectif de contrôle en fonction des différents composants qui modélisent le système. Ainsi, de manière à pouvoir calculer le système contrôlé global, l'objectif doit posséder certaines caractéristiques inhérentes à la structure du système. Notre approche est différente et s'apparente plus à une approche modulaire.

En reprenant les notations du paragraphe 2.1.1, plutôt que de dériver différents objectifs locaux suivant chacun des sous systèmes $(G_i)_{1\leq i\leq n}$, l'idée consiste à sur-approximer le système global G à partir de chacun des sous systèmes $(G_i)_{1\leq i\leq n}$, et à considérer l'objectif de contrôle initial sur chacune de ces sur-approximations.

Plus précisément, on note L le langage sur un alphabet donné Σ, modélisant les comportements du système G à contrôler. Le système ainsi considéré est concurrent et donné par la composition

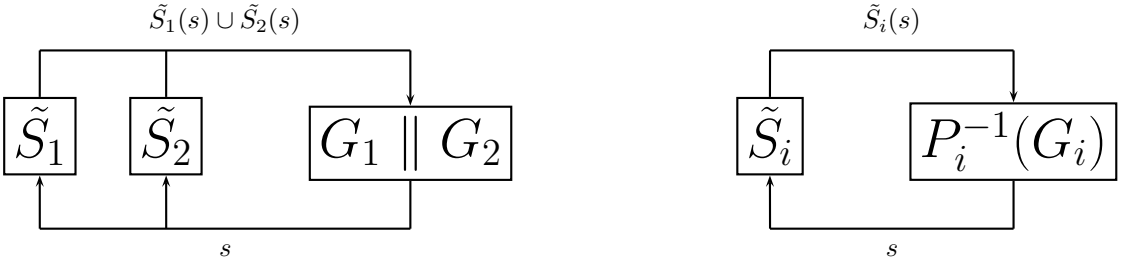


FIG. 2.2: Approche adoptée dans ce chapitre.

parallèle des n sous systèmes modélisés par des langages générés $(L_i)_{1\leq i\leq n}$ où L_i représente un langage sur un alphabet noté Σ_i ($\Sigma = \bigcup_i \Sigma_i$). Par conséquent,

$$L = L_1 \parallel \cdots \parallel L_n = P_1^{-1}(L_1) \cap \cdots \cap P_n^{-1}(L_n)$$

De plus, on considère un objectif de contrôle, modélisé par un langage $K \subseteq \Sigma^*$. Les langages $(K_i)_{1\leq i\leq n}$ sont alors introduits de la façon suivante :

$$\begin{aligned} \overline{K} \cap L &= \overline{K} \cap (P_1^{-1}(L_1) \cap \cdots \cap P_n^{-1}(L)) \\ &= \underbrace{(\overline{K} \cap P_1^{-1}(L_1))}_{K_1} \cap \cdots \cap \underbrace{(\overline{K} \cap P_n^{-1}(L_n))}_{K_n} \end{aligned}$$

Le problème auquel on s'intéresse consiste à calculer $(\overline{K} \cap L)^{\uparrow c}$, i.e la solution du PBSC pour un système modélisé par le langage L et un objectif modélisé par K. L'approche considérée ici a pour but de ramener ce problème à n sous problèmes algorithmiquement moins complexes à résoudre. Les systèmes considérés dans ces sous problèmes sont modélisés par les langages $(P_i^{-1}(L_i))_{1\leq i\leq n}$ et les objectifs de contrôle par les langages $(K_i)_{1\leq_1\leq n}$.

Remarque 7 *Les langages $(P_i^{-1}(L_i))_{1\leq i\leq n}$ sont simplement obtenus en ajoutant pour chaque $i \in \{1, \ldots, n\}$ des boucles d'événements de $\Sigma \setminus \Sigma_i$ à l'automate G_i. La complexité de ce calcul vaut donc $\mathcal{O}(n.N.|\Sigma|)$ dans le pire cas, où N représente le nombre d'états d'un automate G_i.*

2.2 Contrôlabilité Partielle

L'approche considérée dans ce chapitre a pour but de résoudre le PBSC (Problème de Base de la Synthèse de Contrôleurs, section 1.2.2) sur un système concurrent $G = \|_i\ G_i$ en prenant en compte sa structure. Pour cela, ce problème est ramené à la résolution de n problèmes où pour $i \in \{1, \ldots, n\}$, $P_i^{-1}(L_i)$ représente un système à contrôler et K_i un objectif de contrôle. Dans ce cadre, la pertinence de la nature des événements doit être relativisée, et comme le traduit l'exemple 2, la résolution du PBSC pour chacun des problèmes locaux n'est pas satisfaisante, ni en considérant que l'ensemble des événements incontrôlables est Σ_{uc}, ni en supposant qu'il s'agit de $\Sigma_{i,uc}$.

Exemple 2 *Considérons le système L constitué de deux sous systèmes L_1 et L_2 respectivement définis sur les alphabets $\Sigma_1 = \{a, b, d, u_1\}$ et $\Sigma_2 = \{a, b, d, u_2\}$.*

$$L_1 = \overline{\{dab, dau_1\}} \text{ et } L_2 = \overline{\{dab, dau_2\}}$$

avec $\Sigma_{i,uc} = \{u_i\}$.
On considère de plus un langage K sur $\Sigma = \Sigma_1 \cup \Sigma_2$.

$$K = \overline{\{dab, dau_1, dau_2\}}$$

et des langages K_1' et K_2' sur Σ.

$$K_1' = \{dab, dau_1\} \text{ et } K_2' = \{dab, dau_2\}$$

Il est facile de voir que pour $i = 1, 2$ K_i' est contrôlable par rapport à $\Sigma_{i,uc}$ et $P_i^{-1}(L_i)$. De plus $K_1' \cap K_2' = \{dab\}$, donc K_1' et K_2' sont non conflictuels. Or $K_1' \cap K_2'$ n'est pas contrôlable par rapport à $\Sigma_{uc} = \Sigma_{1,uc} \cup \Sigma_{2,uc}$ et $L_1 \parallel L_2$.

De plus, pour $i = 1, 2$, K_i' n'est pas contrôlable par rapport à Σ_{uc} et $P_i^{-1}(L_i)$, alors que $K_1' \cap K_2'$ est contrôlable par rapport à Σ_{uc} et $L_1 \parallel L_2$. Toutefois, le seul langage qui soit contrôlable par rapport à Σ_{uc} et $P_i^{-1}(L_i)$ est le langage vide. ◇

Pour résumer, de manière à obtenir la contrôlabilité globale par intersection de langages non conflictuels, il est insuffisant que ces langages soient localement contrôlables par rapport aux événements incontrôlables locaux. De plus, la contrôlabilité locale par rapport aux événements incontrôlables globaux apparaît extrêmement restrictive. La contrôlabilité partielle offre alors une alternative à ces deux problèmes.

Dans ce chapitre, on cherche à ramener la résolution du PBSC sur $G = \parallel G_i$ avec K pour objectif de contrôle, à la résolution de n problèmes où $(P_i^{-1}(L_i))_{1 \leq i \leq n}$ sont les modèles des systèmes et $(K_i)_{1 \leq i \leq n}$ les objectifs de contrôle. Comme évoqué précédemment et suggéré par l'exemple 2, la résolution locale des PBSC n'est pas satisfaisante en général. La contrôlabilité n'est en fait pas une notion localement adéquate dans ce cas.

Dans ce paragraphe, on donne la définition d'un langage partiellement contrôlable. Celle-ci est définie par rapport à un système modélisé par un langage préfixe-clos L sur un alphabet Σ, un langage préfixe-clos $M \in \mathcal{L}(\Sigma)$ inclus dans L, et deux ensembles Σ_{uc}' et Σ_{uc} d'événements tels que $\Sigma_{uc}' \subseteq \Sigma_{uc} \subseteq \Sigma$, où Σ_{uc} représente l'ensemble des événements incontrôlables du système.

Définition 14 *$\emptyset \subset M \subseteq L \subseteq \Sigma^*$, M, L préfixes-clos, $M' \subseteq M$, $\Sigma_{uc}' \subseteq \Sigma_{uc} \subseteq \Sigma$. M' est partiellement contrôlable par rapport à Σ_{uc}', Σ_{uc}, M et L si*

(i) M' est contrôlable par rapport à Σ_{uc}' et L.

(ii) M' est contrôlable par rapport à Σ_{uc} et M.

Les éléments de Σ_{uc}' et Σ_{uc} sont incontrôlables mais possèdent tout de même des statuts différents. En effet, la contrôlabilité d'un sous langage M' de M par rapport à Σ_{uc} et L n'est pas exigée par la définition 14. Seuls les événements de Σ_{uc}' semblent pertinents (point (i) de la définition 14) pour déterminer la contrôlabilité de M' par rapport au système. En revanche, l'exigence est plus forte concernant la contrôlabilité de M' par rapport à l'objectif M. On souhaite en effet que cette

contrôlabilité soit vérifiée pour l'ensemble Σ_{uc} des événements incontrôlables. Par conséquent, la contrôlabilité partielle utilise l'objectif de contrôle M et un sous ensemble des événements incontrôlables Σ'_{uc} pour affaiblir la notion de contrôlabilité d'un langage M' par rapport au système L.

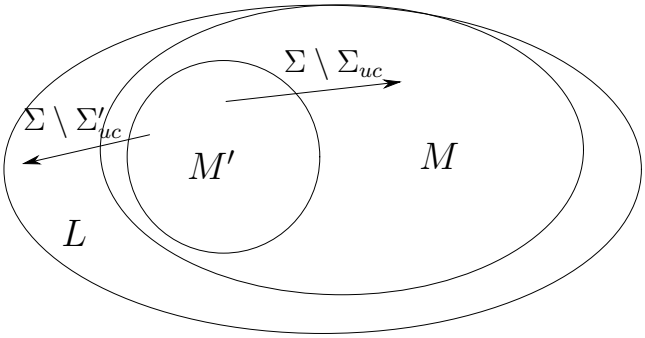
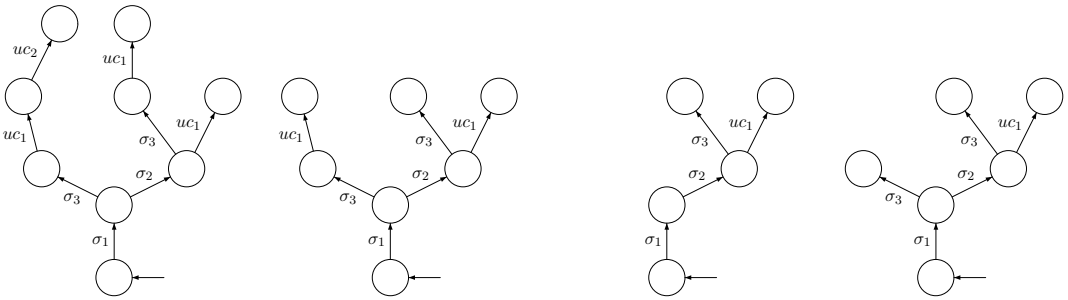


FIG. 2.3: Schématisation de la définition 14.

L'exemple 3 met en oeuvre la définition 14.

Exemple 3 *Considérons les langages L, M $(= M_1)$, M_2 et M_3 décrit par la Figure 2.4. Supposons que $\Sigma_{uc} = \{uc_1, uc_2\}$ et $\Sigma'_{uc} = \{uc_2\}$.*



(a) L (b) M $(= M_1)$ (c) M_2 (d) M_3

FIG. 2.4: Comportements de L, M, M_2, M_3.

- $M_1 = M$ *n'est pas partiellement contrôlable par rapport à Σ'_{uc}, Σ_{uc}, M et L, puisque M n'est pas contrôlable par rapport à Σ'_{uc} et L. En effet, $\sigma_1\sigma_3 uc_1 \in M_1$ et $\sigma_1\sigma_3 uc_1 uc_2 \in L$ avec $uc_2 \in \Sigma'_{uc}$, mais $\sigma_1\sigma_3 uc_1 uc_2 \notin M_1$. Cependant, $M_1 = M$ est contrôlable par rapport à Σ_{uc} et M.*
- *A contrario, M_2 est partiellement contrôlable par rapport à Σ'_{uc}, Σ_{uc}, M et L. En effet, d'une part M_2 est contrôlable par rapport à Σ'_{uc}, et L (notons que puisque $uc_1 \notin \Sigma'_{uc}$, le fait que $\sigma_1\sigma_2\sigma_3 uc_1$ n'appartienne pas à M_2 ne pose pas de problème de contrôlabilité). D'autre*

part, M_2 est contrôlable par rapport à Σ_{uc}, et M. Cependant, M_2 n'est pas contrôlable par rapport à Σ_{uc}, et L.

– *Finalement, M_3 est contrôlable par rapport à Σ'_{uc}, et L. Mais M_3 n'est pas partiellement contrôlable par rapport à Σ'_{uc}, Σ_{uc}, M et L. En effet, M_3 n'est pas contrôlable par rapport à Σ_{uc}, et M puisque $\sigma_1\sigma_3 \in M_3$, $\sigma_1\sigma_3 uc_1 \in M$ où $uc_1 \in \Sigma_{uc}$, mais $\sigma_1\sigma_3 uc_1 \notin M_3$.*

◇

Puisque le langage vide est contrôlable par convention, le langage vide est partiellement contrôlable par rapport à Σ'_{uc}, Σ_{uc}, M et L. On note alors

$$\mathcal{PC}(\Sigma'_{uc}, \Sigma_{uc}, M, L)$$

l'ensemble des langages partiellement contrôlables par rapport à Σ'_{uc}, Σ_{uc}, M et L. En général, un langage $M' \subseteq M$ n'est pas partiellement contrôlable par rapport à Σ'_{uc}, Σ_{uc}, M et L. En revanche, il existe toujours un plus grand sous langage de M' qui vérifie cette propriété.

Proposition 8 *$\emptyset \subset M \subseteq L \subseteq \Sigma^*$, M et L préfixes-clos, $\Sigma'_{uc} \subseteq \Sigma_{uc} \subseteq \Sigma$. Soit $M' \subseteq M$. Il existe un unique langage maximal inclus dans M' qui soit partiellement contrôlable par rapport Σ'_{uc}, Σ_{uc}, M et L. De plus, ce langage est*

$$\mathit{SupCont}(\mathit{SupCont}(M', \Sigma'_{uc}, L), \Sigma_{uc}, M)$$

Démonstration : Soit $M'' \subseteq M'$ un langage partiellement contrôlable par rapport à Σ'_{uc}, Σ_{uc}, M et L. Puisque $M'' \subseteq M'$ est contrôlable par rapport à Σ'_{uc} et L, on a $M'' \subseteq \text{SupCont}(M', \Sigma'_{uc}, L)$. Par monotonie de $\text{SupCont}(, \Sigma_{uc}, M)$,

$$\text{SupCont}(M'', \Sigma_{uc}, M) \subseteq \text{SupCont}(\text{SupCont}(M', \Sigma'_{uc}, L), \Sigma_{uc}, M) \qquad (\alpha)$$

Or M'' est contrôlable par rapport à Σ_{uc} et M. Par conséquent, $M'' = \text{SupCont}(M'', \Sigma_{uc}, M)$ et donc (α) devient

$$M'' \subseteq \text{SupCont}(\text{SupCont}(M', \Sigma'_{uc}, L), \Sigma_{uc}, M)$$

Par conséquent, il suffit de montrer que $\text{SupCont}(\text{SupCont}(M', \Sigma'_{uc}, L), \Sigma_{uc}, M)$ est partiellement contrôlable par rapport à Σ'_{uc}, Σ_{uc}, M et L pour conclure.

(i) Soient $s \in \overline{\text{SupCont}(\text{SupCont}(M', \Sigma'_{uc}, L), \Sigma_{uc}, M)}$ et $\sigma \in \Sigma'_{uc}$ tels que $s\sigma \in L$. Comme

$$\text{SupCont}(\text{SupCont}(M', \Sigma'_{uc}, L), \Sigma_{uc}, M) \subseteq \text{SupCont}(M', \Sigma'_{uc}, L)$$

On a

$$s\sigma \in \overline{\text{SupCont}(M', \Sigma'_{uc}, L)}.\Sigma'_{uc} \cap L$$

Donc par contrôlabilité de $\text{SupCont}(M', \Sigma'_{uc}, L)$,

$$s\sigma \in \overline{\text{SupCont}(M', \Sigma'_{uc}, L)}$$

Or $\text{SupCont}(M', \Sigma'_{uc}, L) \subseteq M$ donc $s\sigma \in M$. Puisque $\sigma \in \Sigma'_{uc}$ et $\Sigma'_{uc} \subseteq \Sigma_{uc}$, on obtient que

$$s\sigma \in \overline{\text{SupCont}(\text{SupCont}(M', \Sigma'_{uc}, L), \Sigma_{uc}, M)}.\Sigma_{uc} \cap M$$

Or $\text{SupCont}(\text{SupCont}(M', \Sigma'_{uc}, L), \Sigma_{uc}, M)$ est contrôlable par rapport à Σ_{uc} et M par définition. Donc

$$s\sigma \in \overline{\text{SupCont}(\text{SupCont}(M', \Sigma'_{uc}, L), \Sigma_{uc}, M)}$$

(ii) Par définition de SupCont$(\bullet, \Sigma_{uc}, M)$.

Par conséquent, SupCont(SupCont$(M', \Sigma'_{uc}, L), \Sigma_{uc}, M)$ est le plus grand langage inclus dans M' qui soit partiellement contrôlable par rapport à Σ'_{uc}, Σ_{uc}, M et L. ◇

Exemple 4 *En reprenant l'exemple 3, le plus grand sous langage de M qui soit partiellement contrôlable par rapport à Σ'_{uc}, Σ_{uc}, et L est le langage M_2.* ◇

La proposition 8 fournit non seulement un résultat d'existence du plus grand sous langage de M' appartenant à $\mathcal{PC}(\Sigma'_{uc}, \Sigma_{uc}, M, L)$, mais aussi une expression de celui-ci. Ce langage sera noté $(M')^{\uparrow pc}$ lorsqu'il n'y aura pas d'ambiguïté.

$(M')^{\uparrow pc}$ peut être calculé en appliquant deux fois l'algorithme de calcul associé à l'opérateur SupCont$(\bullet, \bullet, \bullet)$ en modifiant les paramètres. Par conséquent, la complexité de ce calcul vaut $\mathcal{O}(|\Sigma| N_M^2 . N_L^2)$, où N_M et N_L sont respectivement le nombre d'états des automates représentant les langages M et L.

Lemme 8 *Soient $\Sigma'_{uc} \subseteq \Sigma_{uc} \subseteq \Sigma$ trois alphabets, $M' \subseteq M \subseteq L$ trois langages sur l'alphabet Σ avec M et L préfixes-clos. Si M' est préfixe-clos, alors $(M')^{\uparrow pc}$ est aussi préfixe-clos.*

Démonstration : D'après la proposition 8,

$$(M')^{\uparrow pc} = \text{SupCont}(\text{SupCont}(M', \Sigma'_{uc}, L), \Sigma_{uc}, M)$$

Or d'après le lemme 3, $M'' = \text{SupCont}(M', \Sigma'_{uc}, L)$ est préfixe-clos puisque M' l'est par hypothèse. En appliquant alors à nouveau le lemme 3, SupCont(M'', Σ_{uc}, M) est préfixe-clos puisque M'' l'est. D'où le résultat. ◇

Dans la section suivante, la contrôlabilité partielle est appliquée dans le cadre de systèmes concurrents, afin de déterminer des langages contrôlables inclus dans un objectif de contrôle donné, sans avoir à expliciter le système.

2.3 Contrôle de systèmes concurrents

Dans ce paragraphe, la contrôlabilité partielle est appliquée aux systèmes concurrents. Les notations sont celles introduites dans le paragraphe 2.1. On rappelle que le système G à contrôler est donné par une composition parallèle d'automates

$$G = G_1 \parallel \cdots \parallel G_n$$

où pour $i \in \{1, \ldots, n\}$, L_i représentent le langage généré par G_i. Le langage généré de G est noté L. L'objectif de contrôle est donné par un langage K. On rappelle que

$$\forall i \leq n,\ K_i = \overline{K} \cap P_i^{-1}(L_i)$$

2.3.1 Sous langages contrôlables

Basé sur le concept de la contrôlabilité partielle introduite dans la section précédente, le théorème 5 fournit une méthode modulaire permettant de vérifier la contrôlabilité d'un sous langage de L inclus dans un objectif K.

Théorème 5 *En reprenant les notations précédentes, on considère* $(K'_i)_{1\leq i\leq n}$ *tels que* $\forall 1 \leq i \leq n$, $K'_i \subseteq K_i$ *. Si les langages* $(K'_i)_{1\leq i\leq n}$ *sont non conflictuels*[4] *alors*

$$\forall 1 \leq i \leq n,\ K'_i \in \mathcal{PC}(\Sigma_{i,uc}, \Sigma_{uc}, K_i, P_i^{-1}(L_i)) \Longrightarrow \bigcap_{1\leq i\leq n} K'_i \in \mathcal{C}(\Sigma_{uc}, L)$$

Démonstration : Soient $s \in \overline{\bigcap_i K'_i}$ et $\sigma \in \Sigma_{uc}$ tels que $s\sigma \in L$. Alors

$$s\sigma \in \overline{\bigcap_i K'_i}.\Sigma_{uc} \cap L$$

Donc puisque les langages $(K'_i)_{1\leq i\leq n}$ sont non conflictuels, on a

$$s\sigma \in \bigcap_i \overline{K'_i}.\Sigma_{uc} \cap L$$

Soit $i \in \{1, \ldots, n\}$ tel que $\sigma \in \Sigma_{i,uc}$. Puisque $L \subseteq P_i^{-1}(L_i)$, $s\sigma \in P_i^{-1}(L_i)$ et donc

$$s\sigma \in \overline{K'_i}.\Sigma_{i,uc} \cap P_i^{-1}(L_i)$$

Or $K'_i \in \mathcal{PC}(\Sigma_{i,uc}, \Sigma_{uc}, K_i, P_i^{-1}(L_i))$, donc K'_i est contrôlable par rapport à $\Sigma_{i,uc}$ et $P_i^{-1}(L_i)$. Par conséquent, $s\sigma \in \overline{K'_i}$. Or $K'_i \subseteq K_i$, donc $s\sigma \in K_i$. Par conséquent, $s\sigma \in \overline{K}$. Or $s\sigma \in L$, donc $\forall 1 \leq j \leq n$, $s\sigma \in P_j^{-1}(L_j)$. On en déduit que

$$\forall 1 \leq j \leq n,\ s\sigma \in \overline{K} \cap P_j^{-1}(L_j)\ (= K_j)$$

Puisque $s \in \bigcap_j \overline{K'_j}$, on en déduit que

$$\forall 1 \leq j \leq n,\ s\sigma \in \overline{K'_j}.\Sigma_{uc} \cap K_j$$

Or pour tout $1 \leq j \leq n$, K'_j est partiellement contrôlable par rapport à $\Sigma_{j,uc}$, Σ_{uc}, K_j et $P_j^{-1}(L_j)$, donc pour tout $1 \leq j \leq n$, K'_j est contrôlable par rapport à Σ_{uc} et K_j. Par conséquent, on a

$$\forall 1 \leq j \leq n,\ s\sigma \in \overline{K'_j}$$

ce qui signifie que $s\sigma \in \bigcap_j \overline{K'_j}$. Puisque les langages $(K'_j)_{1\leq j\leq n}$ sont non conflictuels, on a $s\sigma \in \overline{\bigcap_j K'_j}$. D'où le résultat. ◇

[4]On rappelle que deux langages K et K' sur un alphabet Σ sont non conflictuels si $\overline{K \cap K'} = \overline{K} \cap \overline{K'}$ (voir section 1.2.2).

Le théorème 5 fournit une condition suffisante pour déterminer localement si un langage de la forme $\bigcap_{i\leq n} K'_i$ est contrôlable, par rapport à Σ_{uc} et L.

Le théorème 5 montre l'incidence de la contrôlabilité partielle de langages par rapport à la contrôlabilité de l'intersection de ces langages. De plus, comme le traduit le corollaire 1, étant donné un langage préfixe-clos K, il est possible d'appliquer le théorème 5 pour obtenir un sous langage contrôlable à partir de calculs locaux.

Corollaire 1 *Soient $L = L_1 \parallel \cdots L_n$ un système concurrent tel que $L_i \subseteq \Sigma_i^*$ avec $\Sigma_i = \Sigma_{i,uc} \cup \Sigma_{i,c}$ et $\Sigma = \bigcup_i \Sigma_i$ et $\Sigma_{uc} = \cup_i \Sigma_{i,uc}$. Soit K un langage sur Σ. Pour $i \in \{1, \ldots, n\}$, on note*

- *$K_i = K \cap P_i^{-1}(L_i)$, et*
- *${K_i}^{\uparrow pc}$ le plus grand langage partiellement contrôlable par rapport à $\Sigma_{i,uc}$, Σ_{uc}, K_i et $P_i^{-1}(L_i)$.*

alors,

$$\bigcap_{1\leq i\leq n} {K_i}^{\uparrow pc} \in \mathcal{C}(\Sigma_{uc}, L)$$

Démonstration : On sait d'après le théorème 5 que si les langages $({K_i}^{\uparrow pc})_{1\leq i\leq n}$ sont non conflictuels, alors $\bigcap_{1\leq i\leq n} {K_i}^{\uparrow pc} \in \mathcal{C}(\Sigma_{uc}, L)$ [5]. Or pour tout $i \in \{1, \ldots, n\}$, K_i est préfixe-clos. Donc d'après le lemme 8, pour tout $i \in \{1, \ldots, n\}$, ${K_i}^{\uparrow pc}$ est préfixe-clos. On en déduit alors que les langages $({K_i}^{\uparrow pc})_{1\leq i\leq n}$ sont non conflictuels, et on obtient le résultat. ◇

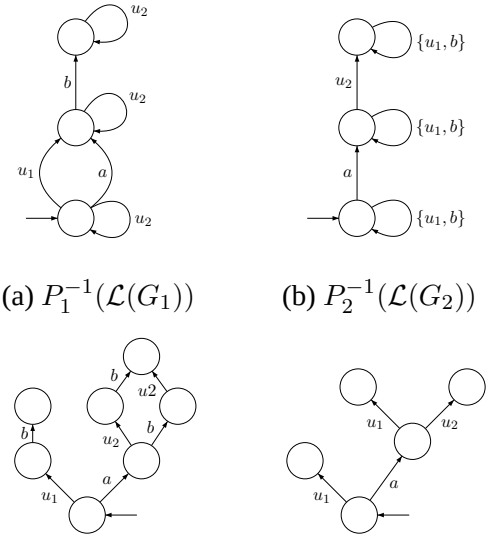
Lorsque K représente un objectif de contrôle, le corollaire 1 indique comment déterminer un ensemble de sous comportements de K dont l'intersection est contrôlable. Chacun de ces sous comportements induit un superviseur qui peut agir sur le système et le contrôle s'effectue ainsi suivant le schéma de la figure 2.1.3.2. Comme dans le cadre de la synthèse modulaire[6], il est préférable de calculer des réalisations des superviseurs locaux, plutôt qu'une unique réalisation d'un superviseur global assurant la même politique de contrôle. En effet, le calcul des réalisations de chacun des superviseurs locaux ne nécessite pas le calcul explicite du système et permet ainsi un gain significatif en complexité.

L'exemple 5 permet d'illustrer le corollaire 1

Exemple 5 *Soit $\mathcal{L}(G) = \mathcal{L}(G_1) \parallel \mathcal{L}(G_2)$, où $\mathcal{L}(G_1) = \overline{\{(a+u_1)b\}}$ et $\mathcal{L}(G_2) = \{a, au_2\}$. On considère $\Sigma_1 = \{a, u_1, b\}$, $\Sigma_2 = \{a, u_2\}$. Les Figures 2.5(a) et 2.5(b) représentent les automates générant $P_1^{-1}(\mathcal{L}(G_1))$ et $P_2^{-1}(\mathcal{L}(G_2))$, et la Figure 2.5(c) est un automate modélisant G (notons que les automates G_1 et G_2 sont facilement obtenus à partir des Figures 2.5(a) et 2.5(b) en enlevant les boucles d'événements). Finalement, dans cet exemple, Σ_s est réduit au singleton $\{a\}$. Supposons que l'objectif de contrôle soit donné par le langage K décrit en Figure 2.5(d).*
Afin d'appliquer le corollaire 1, on calcule les langages $K_i = K \cap P_i^{-1}(\mathcal{L}(G_i))$, $i = 1, 2$. Les automates générés K_1 et K_2 sont représentés par les Figures 2.6(a) et 2.6(b).
On calcule à présent $K_1^{\uparrow pc}$ (resp $K_2^{\uparrow pc}$), le plus grand langage de K_1 (resp. K_2) qui est partiellement contrôlable par rapport à $P_1^{-1}(\mathcal{L}(G_1)$, $\Sigma_{uc} = \{u_1, u_2\}$ et $\Sigma_{1,uc} = \{u_1\}$ (resp $\Sigma_{2,uc} = \{u_2\}$

[5] On rappelle que $\mathcal{C}(\Sigma_{uc}, L)$ représente l'ensemble des sous langages de L qui sont contrôlables par rapport à Σ_{uc} et L.

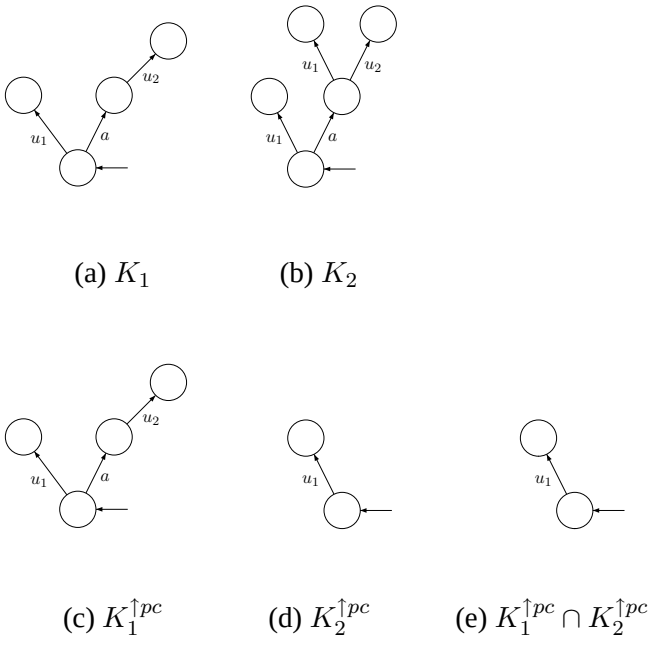
[6] Voir section 1.3.1.1.



(a) $P_1^{-1}(\mathcal{L}(G_1))$ (b) $P_2^{-1}(\mathcal{L}(G_2))$

(c) $\mathcal{L}(G)$ (d) K

FIG. 2.5: Comportements du système et objectif global.



(a) K_1 (b) K_2

(c) $K_1^{\uparrow pc}$ (d) $K_2^{\uparrow pc}$ (e) $K_1^{\uparrow pc} \cap K_2^{\uparrow pc}$

FIG. 2.6: Objectifs de contrôle locaux

et $P_2^{-1}(\mathcal{L}(G_2))$. L'intersection de ces deux langages donne le langage $K_1^{\uparrow pc} \cap K_2^{\uparrow pc}$ (Figure 2.6(e)), qui est contrôlable par rapport à $\mathcal{L}(G)$ et Σ_{uc}. ◇

Lorsque le langage K représente l'objectif de contrôle que l'on souhaite assurer, résoudre le PBSC consiste à déterminer le plus grand langage contrôlable par rapport à Σ_{uc} et L qui soit inclus dans $\overline{K} \cap L$. La démarche décrite dans ce chapitre ramène ce problème à des problèmes locaux. Le corollaire 1 offre une solution pour calculer un sous langage de $(\overline{K} \cap L)^{\uparrow c}$ qui soit contrôlable. Or cette propriété doit effectivement être vérifiée pour résoudre le PBSC. Toutefois, comme le montre l'exemple 6, la solution fournie par le corollaire 1 (i.e $\bigcap_i {K_i}^{\uparrow pc}$) n'est pas maximale en général et ne résout donc pas le PBSC. C'est pourquoi, on s'intéresse dans la suite aux conditions assurant cette maximalité.

Exemple 6 *On considère les langages $L_1 = \{a, auc, ab\}$, $L_2 = \{a, uc, ab\}$, $L = L_1 \parallel L_2 = \{a, ab\}$, et l'objectif de contrôle $K = \{a, uc\}$. On a*

$$\begin{aligned} K_1 &= \{a\} \\ K_2 &= \{a, uc\} \\ K_1^{\uparrow pc} &= \emptyset \\ K_2^{\uparrow pc} &= \{a, uc\} \\ K_1^{\uparrow pc} \cap K_2^{\uparrow pc} &= \emptyset \\ (\overline{K} \cap L)^{\uparrow c} &= \{a\} \end{aligned}$$

◇

Condition sur les événements partagés. Comme évoqué précédemment, on souhaite déterminer des conditions suivants lesquelles la solution fournie par le corollaire 1 (i.e $\bigcap_i {K_i}^{\uparrow pc}$) est maximale et résout le PBSC. Pour cela, on peut remarquer que les événements partagés et incontrôlables induisent des erreurs lorsque l'on travaille localement (voir calcul de $K_1^{\uparrow pc}$ dans l'exemple 6). Cela nous amène à supposer par la suite que les événements partagés du système G sont contrôlables (i.e $\Sigma_s \subseteq \Sigma_c$), ou de manière équivalente que les événements incontrôlables du système sont tous locaux.

Cette hypothèse s'avère primordiale lorsque l'on souhaite effectuer des calculs locaux. En effet, il est délicat de traiter localement les événements qui sont à la fois partagés et incontrôlables. L'aspect incontrôlable d'un événement sert à modéliser que celui-ci peut se produire de manière spontanée sans pouvoir être empêché. Or l'occurrence d'un événement partagé est conditionnée par la dynamique de chacun des sous systèmes auxquels il appartient. Ainsi, la condition $\Sigma_s \subseteq \Sigma_c$ permet d'éviter ce problème, puisque dans ce cas un événement incontrôlable est local et peut donc se produire indépendemment de la dynamique des autres sous systèmes.

Remarque 8 *L'hypothèse "$\Sigma_s \subseteq \Sigma_c$" est souvent rencontrée dans les problèmes de synthèse de superviseurs sur des systèmes concurrents. On la rencontre ainsi dans les travaux de [77] et [2]. De plus, on peut noter que cette condition est toujours vérifiée lorsque les sous systèmes ne partagent aucun événement (i.e $\Sigma_s = \emptyset$) comme dans les travaux de [53].*

En reprenant le résultat du corollaire 1, celui-ci nous indique que si le langage $\bigcap_{1\leq i\leq n} {K_i}^{\uparrow pc}$ est contrôlable par rapport Σ_{uc} et L. De plus,

$$\bigcap_i {K_i}^{\uparrow pc} \subseteq \bigcap_i K_i = \overline{K} \cap L$$

On en déduit alors que

$$\bigcap_{1\leq i\leq n} {K_i}^{\uparrow pc} \subseteq (\overline{K} \cap L)^{\uparrow c}$$

Or comme mentionné précédemment, on cherche des conditions suivant lesquelles l'égalité est vérifiée. Il est facile de voir que cette égalité est notamment valide si

$$\forall 1 \leq i \leq n,\ (\overline{K} \cap L)^{\uparrow c} \subseteq {K_i}^{\uparrow pc}$$

Or, par maximalité de ${K_i}^{\uparrow pc}$ l'inclusion précédente est valide si $(\overline{K} \cap L)^{\uparrow c}$ était partiellement contrôlable par rapport $\Sigma_{i,uc}$, Σ_{uc}, K_i et $P_i^{-1}(L_i)$. Le lemme 9 donne dans le cas où $\Sigma_s \subseteq \Sigma_c$, un premier résultat en ce sens.

Lemme 9 *Si $\Sigma_s \subseteq \Sigma_c$, alors $(\overline{K} \cap L)^{\uparrow c}$ est contrôlable par rapport à $\Sigma_{i,uc}$, $P_i^{-1}(L_i)$*

Démonstration : On suppose $\Sigma_s \subseteq \Sigma_c$. Soient $s \in \overline{(\overline{K} \cap L)^{\uparrow c}}$, $\sigma \in \Sigma_{i,uc}$ tels que $s\sigma \in P_i^{-1}(L_i)$. Il suffit de montrer que $s\sigma \in \overline{(\overline{K} \cap L)^{\uparrow c}}$. Or $\sigma \in \Sigma_{i,uc}$ et $\Sigma_s \subseteq \Sigma_c$, donc $\sigma \in \Sigma_i \setminus \Sigma_s$. De plus, $s \in P_i^{-1}(L_i)$, donc le lemme (6) s'applique, et on obtient $s\sigma \in L$. Par conséquent,

$$s\sigma \in \overline{(\overline{K} \cap L)^{\uparrow c}}.\Sigma_{i,uc} \cap L$$

Et puisque $\Sigma_{i,uc} \subseteq \Sigma_{uc}$, on a

$$s\sigma \in \overline{(\overline{K} \cap L)^{\uparrow c}}.\Sigma_{uc} \cap L$$

Or $\overline{(\overline{K} \cap L)^{\uparrow c}}$ est contrôlable par rapport à Σ_{uc} et L, donc $s\sigma \in \overline{(\overline{K} \cap L)^{\uparrow c}}$. D'où le résultat. $\diamond$

Par conséquent, le point (i) de la définition de la contrôlabilité partielle (Définition 14) est vérifié par $(\overline{K} \cap L)^{\uparrow c}$. En revanche, comme le montre l'exemple 7 le point (ii) de la définition 14 ne l'est pas toujours.

Exemple 7 *On considère les langages suivants :*

$$L_1 = \overline{\{a_1.uc_1\}},\ L_2 = \overline{\{a_2.uc_2\}},\ \textit{et}\ K = \{a_1.uc_2, a_1.uc_1.a_2.uc_2\}$$

On a

$$\overline{K} \cap L = (\overline{K} \cap L)^{\uparrow c} = \overline{\{a_1.uc_1.a_2.uc_2\}}$$

et

$$K_1 = \overline{K} \cap P_i^{-1}(L_1) = \overline{K} = \{a_1.uc_2, a_1.uc_1.a_2.uc_2\}$$

Dans ce cas, $(\overline{K} \cap L)^{\uparrow c}$ *n'est pas contrôlable par rapport à* $\Sigma_{1,uc}$ *et* K_1 *puisque*

$$a_1 \in \overline{(\overline{K} \cap L_m)^{\uparrow c}} \wedge uc_2 \in \Sigma_{uc} \wedge a_1.uc_2 \in K_1$$

et pourtant $a_1.uc_2 \notin \overline{(\overline{K} \cap L)^{\uparrow c}}$

◇

On peut notamment noter que le problème mis en évidence dans l'exemple 7 est lié à la séquence $a_1.uc_2$ qui n'est en fait pas une séquence de L. Le fait de travailler localement empêche de déterminer quelles sont exactement les séquences de L (les langages $(P_i^{-1}(L_i))_{1\leq i\leq n}$ sont en fait des sur-approximations de L). Assurer la contrôlabilité nous conduit alors "à ne pas prendre de risque", en traitant toutes les séquences de l'approximation comme si elle appartenait à L. Il apparaît alors naturel de s'intéresser aux conditions qui permettrait de lever l'ambiguïté concernant l'appartenance à L des séquences de K_i.

2.3.2 Objectifs de contrôle modélisant un sous comportement du système

Dans cette section, on reprend les notations introduites dans la paragraphe 2.1.3.2. On s'intéresse à un système concurrent G modélisé par une composition parallèle de sous systèmes $(G_i)_{1\leq i\leq n}$, i.e $G = \|_i\ G_i$. On supposera dans la suite que les événements partagés de G sont contrôlables, i.e $\Sigma_s \subseteq \Sigma_c$.

Le théorème 5 montre que la notion de *contrôlabilité partielle* offre un outil intéressant pour calculer des langages contrôlables sur des systèmes de grande taille. Le calcul de solutions maximales partiellement contrôlables et "locales" fournit (lorsque la solution est non vide) en effet un système contrôlé satisfaisant l'objectif de contrôle.

Toutefois, comme le montre l'exemple 6, la solution fournie par le théorème 5 n'est pas maximale en général. De plus, l'exemple 7 indique qu'une des causes de cette "non maximalité" est la méconnaissance de l'ensemble des comportements possibles du système lorsque l'on travaille localement.

C'est pourquoi, on considère dans cette section que l'objectif de contrôle est un ensemble de sous comportements du système global (i.e $K \subseteq L$). En effet, bien que cette hypothèse ne permette pas de connaître exactement l'ensemble des comportements possibles du système, elle s'avère suffisante pour que le sous langage contrôlable fournit par le corollaire 1 soit maximal. Ceci est d'autant plus intéressant que vérifier que $K \subseteq L$ peut se faire localement, évitant ainsi l'explosion combinatoire.

Lemme 10 *Soit* $L = L_1 \parallel \cdots \parallel L_n$ *un système concurrent tel que* $L_i \in \mathcal{L}(\Sigma_i)$ *et* $L \in \mathcal{L}(\Sigma)$ *avec* $\Sigma = \bigcup_i \Sigma_i$. *Soit* $K \in \mathcal{L}(\Sigma)$, *alors*

$$K \subseteq L \Longleftrightarrow \forall 1 \leq i \leq n,\ K \subseteq P_i^{-1}(L_i)$$

Démonstration : Par définition de la composition parallèle (définition 1), $L = \bigcap_{1\leq i\leq n} P_i^{-1}(L_i)$.
Par conséquent $K \subseteq L \Longleftrightarrow K \subseteq \bigcap_{1\leq i\leq n} P_i^{-1}(L_i)$.
Ce qui est équivalent à $\forall 1 \leq i \leq n,\ K \subseteq P_i^{-1}(L_i)$. ◇

Pour $i \in \{1, \ldots, n\}$, on note N le nombre d'états de l'automate G_i générant le langage L_i. L'automate G générant le langage L possède alors N^n états. Le langage $P_i^{-1}(L_i)$ peut être modélisé par un automate obtenu à partir de G_i en rajoutant des transitions correspondant à des boucles sur les états. Par conséquent, le nombre d'états de l'automate modélisant $P_i^{-1}(L_i)$ est aussi N. De plus, on note N_K le nombre d'états de l'automate générant K. Vérifier l'inclusion de K dans L à partir de G induit une complexité en $\mathcal{O}(|\Sigma|.N_K.N^n)$. Le lemme 10 fournit une équivalence qui permet d'effectuer cette vérification avec une complexité en $\mathcal{O}(n.|\Sigma|.N_K.N)$, évitant notamment le calcul explicite de l'automate G.

Le résultat principal de cette section est donné par le théorème 6. Ce théorème fournit des conditions suffisantes suivant lesquelles le PBSC peut être résolu à partir de calculs locaux.

Théorème 6 *Soit $L = L_1 \parallel \cdots \parallel L_n$ un système concurrent tel que $L_i \in \mathcal{L}(\Sigma_i)$ et $L \in \mathcal{L}(\Sigma)$ avec $\Sigma = \bigcup_i \Sigma_i$ et $\Sigma_s \subseteq \Sigma_c$. Soit $K \subseteq L$ un langage préfixe-clos. Pour $i \in \{1, \ldots, n\}$, on note*

- $K_i = \overline{K} \cap P_i^{-1}(L_i)$.
- ${K_i}^{\uparrow pc} =$ *SupCont*$(K_i, \Sigma_{i,uc}, P_i^{-1}(L_i)), \Sigma_{uc}, K_i)$ *le plus grand sous-langage de K_i partiellement contrôlable par rapport à $\Sigma_{i,uc}$, Σ_{uc}, K_i et $P_i^{-1}(L_i)$.*

Alors,

$$\bigcap_{1 \leq i \leq n} {K_i}^{\uparrow pc} = \left(\overline{K} \cap L\right)^{\uparrow c}$$

Démonstration : Tout d'abord, on a $\bigcap_{1 \leq i \leq n} {K_i}^{\uparrow pc} \subseteq \bigcap_{1 \leq i \leq n} K_i = \overline{K} \cap L$. De plus, d'après le corollaire 1, $\bigcap_{i \leq n} {K_i}^{\uparrow pc}$ est contrôlable par rapport à Σ_{uc} et L. Par maximalité de $\left(\overline{K} \cap L\right)^{\uparrow c}$

$$\bigcap_i {K_i}^{\uparrow pc} \subseteq \left(\overline{K} \cap L\right)^{\uparrow c}$$

Afin de montrer l'inclusion inverse, il suffit de montrer que $\forall 1 \leq i \leq n,\ \left(\overline{K} \cap L\right)^{\uparrow c} \subseteq {K_i}^{\uparrow pc}$. Or pour $i \in \{1, \ldots, n\}$

$$\left(\overline{K} \cap L\right)^{\uparrow c} \subseteq \overline{K} \cap L \subseteq \overline{K} \cap P_i^{-1}(L_i) = K_i$$

Donc par maximalité de ${K_i}^{\uparrow pc}$ (i.e ${K_i}^{\uparrow pc}$ est le plus grand sous langage de K_i qui est partiellement contrôlable par rapport à $\Sigma_{i,uc}$, Σ_{uc}, K_i, $P_i^{-1}(L_i)$), il suffit de montrer que $\left(\overline{K} \cap L\right)^{\uparrow c}$ est partiellement contrôlable par rapport à $\Sigma_{i,uc}$, Σ_{uc}, K_i, $P_i^{-1}(L_i)$. Pour cela, on va montrer que $\left(\overline{K} \cap L\right)^{\uparrow c}$ vérifie les deux points de la définition 14.

(i) D'après le lemme 9.

(ii) Par définition, $\left(\overline{K} \cap L\right)^{\uparrow c} \in \mathcal{C}(\Sigma_{uc}, L)$. Et puisque $\overline{K} \cap L^{\uparrow c} \subseteq K$ et $K \subseteq L$, on déduit du lemme 2 que $\left(\overline{K} \cap L\right)^{\uparrow c} \in \mathcal{C}(\Sigma_{uc}, K)$. Or $K = \overline{K} \cap P_i^{-1}(L_i) = K_i$ puisque K est préfixe-clos, inclus dans L, et que $L \subseteq P_i^{-1}(L_i)$. Donc $K_i = K$ et on obtient que $\left(\overline{K} \cap L\right)^{\uparrow c} \in \mathcal{C}(\Sigma_{uc}, K_i)$. D'où le résultat. ◇

Le théorème 6 donne donc une condition sur le langage représentant l'objectif de contrôle, qui permet de résoudre le PBSC en n'effectuant que des calculs locaux. Ceci est particulièrement intéressant lorsque Le système représenté par L est de taille trop importante pour être manipulé.

Complexité. En reprenant les notations du théorème 6, supposons que $\Sigma_s \subseteq \Sigma_c$, $K \subseteq L$. Dans ce cas, la solution du PBSC, qui vaut $(\overline{K} \cap L)^{\uparrow c}$ (si ce langage est non vide), est donnée par $\bigcap_i {K_i}^{\uparrow pc}$. Or d'après ce qui a été vu dans la section 2.2, pour $i \in \{1, \ldots, n\}$ le calcul de ${K_i}^{\uparrow pc}$ a une complexité dans le pire cas égale à $\mathcal{O}(|\Sigma|.N_K^2.N^2)$ où N_K et N représentent respectivement le nombre d'états des automates modélisant les langages K et L_i. Donc la complexité du calcul de l'ensemble des langages $({K_i}^{\uparrow pc})_{1 \leq i \leq n}$ vaut $\mathcal{O}(n.|\Sigma|.N_K^2.N^2)$. Cette complexité doit être comparée au calcul de $(\overline{K} \cap L)^{\uparrow c}$ effectué de manière classique, i.e $\mathcal{O}(|\Sigma|.N_K^2.(N^n)^2)$.

Exemple 8 *Afin d'illustrer le théorème 6, on s'intéresse à l'exemple d'une cellule de production (introduit dans [50] à partir de l'exemple Small Factory [79]. Le système considéré est composé de deux machines M_1 et M_2 échangeant des informations à travers un tampon (BUF).*

M_1 reçoit des entrées (événements) de l'environnement, les traite puis envoie des informations dans le tampon BUF. Ces informations sont ensuite utilisées par la machine M_2, qui les traite et renvoie alors des informations à l'environnement. Le tampon est supposé ne pouvoir contenir au plus qu'une seul unité d'information. Les machines M_1 et M_2, ainsi que le tampon BUF sont décrits par les automates donnés en figure 2.7.

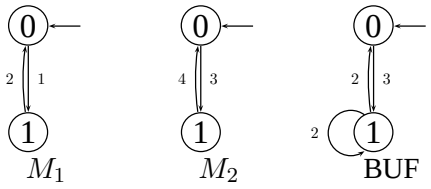


FIG. 2.7: Cellule de production

Le système à contrôler est représenté par la composition parallèle de ces automates. L'alphabet du système est représenté par l'ensemble $\Sigma = \{1, 2, 3, 4\}$. On note $\Sigma_1 = \{1, 2\}$, $\Sigma_2 = \{3, 4\}$ et $\Sigma_3 = \{2, 3\}$. De plus, on note P_i la projection naturelle de Σ sur Σ_i. Contrairement à l'exemple traité dans [50], seul l'événement 4 est supposé ici être incontrôlable. Ainsi, $\Sigma_{1,uc} = \emptyset$, $\Sigma_{2,uc} = \{4\}$, $\Sigma_{3,uc} = \emptyset$.

Bien que le tampon ne puisse contenir qu'une seule unité d'information, il se peut notamment que M_1 traite une information alors que le tampon n'est pas vide. Ceci augmente le risque de dépassement de capacité du tampon. Afin d'éviter ce problème, on souhaite décrire un objectif de contrôle assurant qu'une seule unité d'information soit présente dans le système global à tout moment de l'exécution. Cet objectif, noté K est donné par la figure 2.8 et traduit aussi que le tampon ne peut se vider s'il l'est déjà.

On cherche à appliquer le théorème 6. Pour cela, on donne les projections inverses des automates modélisant M_1, M_2 et BUF dans la figure 2.9.

On peut alors vérifier que $K \subseteq P_1^{-1}(M_1)$, $K \subseteq P_2^{-1}(M_2)$ et $K \subseteq P_3^{-1}(BUF)$, et en déduire que $K \subseteq L$. Puisque $K \subseteq L$, les langages $K_1 = \overline{K} \cap P_1^{-1}(M_1)$, $K_2 = \overline{K} \cap P_2^{-1}(M_2)$ et

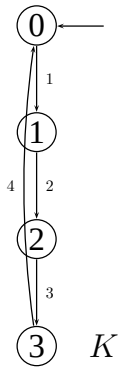


FIG. 2.8: Objectif de contrôle

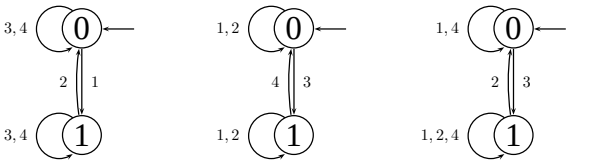


FIG. 2.9: Projections inverses

$K_3 = \overline{K} \cap P_3^{-1}(BUF)$ sont en fait égaux au langage K donné par la figure 2.8. De plus, on peut noter que les événements partagés du système sont contrôlables, et donc que les hypothèses du théorème 6 sont vérifiées.

Finalement, K_1 est partiellement contrôlable par rapport à $\Sigma_{1,uc}$, Σ_{uc}, K_1 et $P_1^{-1}(M_1)$. En effet, la contrôlabilité de K_1 par rapport à $\Sigma_{1,uc}$ et $P_1^{-1}(M_1)$ d'une part, et Σ_{uc} et K_1 d'autre part est triviale. De même K_2 (resp. K_3) est partiellement contrôlable par rapport à $\Sigma_{2,uc}$ et $P_2^{-1}(M_2)$ (resp. $\Sigma_{3,uc}$ et $P_3^{-1}(BUF)$).

Par conséquent, on déduit du théorème 6 que $\overline{K} \cap L\ (= \overline{K})$ est solution du PBSC où $M_1 \parallel M_2 \parallel BUF$ représente le système à contrôler et K l'objectif de contrôle.

◇

2.3.3 Objectif de contrôle localement cohérent

La section précédente fournit une solution au PBSC pour des systèmes concurrents. Toutefois cette solution n'est valide que si l'objectif de contrôle modélise un sous ensemble des comportements du système. Il est toutefois intéressant de pouvoir considérer des objectifs de contrôle modélisant des comportements non spécifiés pour le système. En effet, il semble délicat en général de ne spécifier que des comportements du système qui appartiennent aussi l'objectif de contrôle, notamment lorsque le système est concurrent et de grande taille.

Le problème étudié dans cette section consiste, étant donné un objectif modélisé par un langage K, à déterminer des conditions suivant lesquelles le PBSC peut être résolu en utilisant l'approche décrite dans la section 2.1.3.2. Notons que la condition $\Sigma_s \subseteq \Sigma_c$ est encore supposée valide. De plus, on rappelle les notations suivantes : pour $i \in \{1, \ldots, n\}$,

$$K_i = K \cap P_i^{-1}(L_i)$$

Notons enfin que puisque les langages $(K_i)_i$ sont préfixes-clos, ils sont non conflictuels. Par conséquent, le théorème 5 et le corollaire 1 sont ici valides et il est intéressant de déterminer des conditions assurant l'égalité dans le corollaire 1. Afin de cerner quels sont les obstacles qui s'opposent à la maximalité de la solution obtenue, considérons l'exemple 9.

Exemple 9

$$\begin{aligned}
L_1 &= \{a, uc_1, ab, uc_1b\} \\
L_2 &= \{a, auc_2\} \\
L &= \{a, uc_1, ab, auc_2, uc_1b, uc_1uc_2\} \\
K &= \{a, uc_1, auc_2, auc_1, uc_1uc_1, uc_1uc_2\} \\
P_1^{-1}(L_1) &= \overline{(uc_2)^*.a.(uc_2)^*.b.(uc_2)^*} \cup \overline{(uc_2)^*.uc_1.(uc_2)^*.b.(uc_2)^*} \\
P_2^{-1}(L_2) &= \overline{\{uc_1, b\}^*.a.\{uc_1, b\}^*.uc_2.\{uc_1, b\}^*} \\
K_1 &= \{a, uc_1\} \\
K_2 &= \{a, uc_1, auc_1, auc_2\} \\
K_1^{\uparrow pc} &= \{a, uc_1\} \\
K_2^{\uparrow pc} &= \{uc_1\} \\
K_1^{\uparrow pc} \cap K_2^{\uparrow pc} &= \{uc_1\} \\
(K \cap L)^{\uparrow c} &= \{a, uc_1, auc_2, uc_1uc_2\}
\end{aligned}$$

La mauvaise approximation locale dans cet exemple vient du fait que $P_2^{-1}(L_2)$ possède des traces n'appartenant pas à L et qui semblent ici rendre K_2 partiellement incontrôlable ($auc_1 \in K_2, auc_1uc_2 \in P_2^{-1}(L_2)$ et $auc_1uc_2 \notin K_2$). Pourtant auc_1 et auc_1uc_2 n'appartiennent pas à L. Ce type de traces ne devrait en fait pas être pris en compte. De plus, on peut noter que la propriété $\Sigma_s \subseteq \Sigma_c$ permet en fait de conclure à l'incohérence de l'objectif. En effet, l'objectif de contrôle représente non seulement l'ensemble des comportements que l'on désire que le système vérifie, mais aussi implicitement l'ensemble des comportements que l'on souhaite voir interdits par contrôle (il s'agit des comportements qui n'apparaissent pas dans l'objectif). En tenant compte de ces considérations, l'hypothèse $\Sigma_s \subseteq \Sigma_c$ doit mener à l'élaboration d'objectifs "cohérents".

Reprenons l'objectif de l'exemple précédent et supposons que L_1 et L_2 soient inconnus. En revanche, on suppose connaître Σ, Σ_1, Σ_2, Σ_s, Σ_c et on suppose que $\Sigma_s \subseteq \Sigma_c$. Cette simple connaissance du système permet de conclure que $K = \{a, uc_1, auc_2, auc_1, uc_1uc_1, uc_1uc_2\}$ pose problème pour résoudre le PBSC par des calculs locaux. Le problème provient de l'absence d'information concernant l'appartenance des séquences de l'objectif à l'ensemble des comportements possibles du système. Il est notamment tentant de conclure que les séquences $a.uc_1$ et $a.uc_2$ ne peuvent appartenir au système contrôlé puisque l'entrelacement des événements incontrôlables uc_1 et uc_2 n'est pas représenté par l'objectif. Or la séquence $a.uc_1$ ne modélise en fait pas un comportement du système. Par conséquent, l'absence d'entrelacement des événements uc_1 et uc_2

peut sembler légitime et ne provoque pas de violation de la condition de contrôlabilité. Malheureusement, ces considérations ne peuvent être invoquées lorsque les calculs s'effectuent localement, puisque le test d'appartenance au langage L nécessite une connaissance globale du système.

Notons enfin que les objectifs tels que K trouvent une interprétation : l'exemple précédent permet de modéliser que l'occurrence d'un événement incontrôlable (uc_1 ou uc_2) peut se produire si le système l'autorise. En revanche l'occurrence consécutive des événements uc_1 et uc_2 (ou inversement) n'est pas permise. ◇

On s'intéresse dans la suite à des conditions assurant le résultat du théorème 6, mais pour le cas où $K \not\subseteq L$. Une des vertus des conditions énoncées par la suite est qu'elles touchent une classe relativement large de langages. Une autre vertu est qu'elles peuvent être vérifiées avec un coût restreint, afin de ne pas perdre l'avantage de la méthode de synthèse modulaire proposée ici.

2.3.3.1 Cohérence locale

Dans cette section, une condition suffisante sur K permettant la résolution du PBSC par des calculs locaux est donnée. Cette condition, appelée *cohérence locale* est définie à partir de la notion de *cohérence*, dont la définition est donnée maintenant.

Définition 15 *Soient $\Sigma' \subseteq \Sigma$ deux alphabets et $M \subseteq \Sigma^*$ un langage préfixe-clos. On note $P_{\Sigma'} : \Sigma^* \longrightarrow \Sigma'^*$ la projection naturelle sur Σ'. Soient $\Sigma_{uc} \subseteq \Sigma$ et $\Sigma'_{uc} = \Sigma_{uc} \cap \Sigma'$. M est cohérent par rapport à $P_{\Sigma'}$, Σ_{uc} si*

$$\forall s \in M,\ \forall s' \in M(s, \Sigma_{uc}),\ \forall \sigma \in \Sigma'_{uc},\ P_{\Sigma'}(s')\sigma \in M(s, \Sigma'_{uc}) \Rightarrow s'\sigma \in M(s, \Sigma_{uc}) \tag{2.3}$$

On reprend les notations de la définition 15 et on raisonne à partir de la contraposée de l'expression (2.3). On s'intéresse à l'ensemble des séquences de Σ^*_{uc} qui prolongent une séquence de s dans M. Considérons une telle séquence, notée s'. Le langage M est cohérent par rapport à $P_{\Sigma'}$ et Σ_{uc} si lorsqu'un événement $\sigma \in \Sigma'_{uc}$ ne prolonge pas ss' dans M, alors $P_{\Sigma'}(s')\sigma$ ne prolonge pas s dans M.

Définition 16 *Soient $(\Sigma_i)_{1\leq i\leq n}$ n alphabets et n langages préfixes-clos $(L_i)_{1\leq i\leq n}$ n langages tels que $L_i \subseteq \Sigma_i^*$. On note $\Sigma = \bigcup_i \Sigma_i$, $\Sigma_{uc} = \bigcup_i \Sigma_{i,uc}$, $P_i : \Sigma \longrightarrow \Sigma_i$ et $L = \|_i\ L_i$. Soit un langage $K \subseteq \Sigma^*$. Pour $i \in \{1, \ldots, n\}$, on note $K_i = K \cap P_i^{-1}(L_i)$. K est localement cohérent par rapport à Σ_{uc} et L si*

$$\forall i \in \{1, \ldots, n\},\ K_i \text{ est cohérent par rapport à } P_i \text{ et } \Sigma_{uc} \tag{2.4}$$

La connaissance des événements incontrôlables et des sous systèmes où ils se produisent, implique en effet d'être cohérent lors de la spécification de l'objectif. En effet, même si l'on ne connaît pas explicitement les comportements du système à contrôler (ce qui est le cas lorsque celui-ci est décrit à partir d'une composition parallèle), certaines précautions peuvent être prises lorsque l'objectif de contrôle est spécifié. Notamment, lorsqu'après une séquence s du langage modélisant l'objectif, un événement incontrôlable $\sigma \in \Sigma_i$ (qui est local) est autorisé. En effet, dans ce cas toute séquence

d'événements incontrôlables s' n'appartenant pas à Σ_i, et autorisée après s, doit être prolongée par σ dans K_i, afin de respecter l'entrelacement entre événements locaux de sous systèmes différents. C'est ce que traduit la condition (2.4). Si la condition (2.4) n'est pas respectée, cela signifie que l'indépendance entre deux événements incontrôlables de deux sous systèmes n'est pas respectée. Cependant, cette indépendance peut ne pas être apparemment vérifiée par l'objectif K, alors que celui-ci est cohérent. En effet, si ss' n'est pas une séquence admise par le système, alors il est inutile de la spécifier dans l'objectif (mais on ne peut pas le savoir sans expliciter le système). En revanche si ss' est une séquence admise par le système, alors elle ne sera pas retenue comme valide lors de la phase de contrôle (ce que l'on peut vérifier sur l'objectif, sans construire le système).

Finalement, des problèmes surviennent lorsqu'après une séquence s, $\sigma \in \Sigma_i$ est autorisé dans K, ainsi qu'une séquence s' d'événements incontrôlables n'appartenant pas à Σ_i, qui n'est pas prolongée par σ dans K.

Un tel objectif peut traduire qu'après la séquence s dans K, σ et s' sont permis de manière exclusive. La cohérence de K n'est alors pas vérifiée si $s\sigma$ est une séquence de $P_i^{-1}(L_i)$ (et donc de K_i). Or comme le montre le théorème 7, la cohérence de l'objectif est intéressante pour déterminer une solution du problème de base de la synthèse de contrôleurs.

Théorème 7 *Soient $(\Sigma_i)_{1\leq i\leq n}$ n alphabets et n langages préfixes-clos $(L_i)_{1\leq i\leq n}$ tels que $L_i \subseteq \Sigma_i^*$. On note $\Sigma = \bigcup_i \Sigma_i$, $\Sigma_{uc} = \bigcup_i \Sigma_{i,uc}$, $P_i : \Sigma \longrightarrow \Sigma_i$ et $L = \|_i L_i$. On suppose que $\Sigma_s \subseteq \Sigma_c$. Soit un langage $K \subseteq \Sigma^*$. Pour $i \in \{1, \ldots, n\}$, on note $K_i = K \cap P_i^{-1}(L_i)$. Si K est localement cohérent par rapport à Σ_{uc} et L, alors*

$$\bigcap_{1 \leq i \leq n} K_i^{\uparrow pc} = (K \cap L)^{\uparrow c}$$

Démonstration : Soit $K \subseteq \Sigma^*$ un langage localement cohérent par rapport à Σ_{uc} et L. On veut montrer que

$$\bigcap_{1 \leq i \leq n} K_i^{\uparrow pc} = (K \cap L)^{\uparrow c}$$

Or $\bigcap_{1 \leq i \leq n} K_i^{\uparrow pc}$ est inclus dans K et est contrôlable par rapport à Σ_{uc} et L, d'après le théorème 5 (car les langages ${K_i}^{\uparrow pc})_i$ sont préfixes-clos, donc non conflictuels). On a donc

$$\bigcap_{1 \leq i \leq n} K_i^{\uparrow pc} \subseteq (K \cap L)^{\uparrow c}$$

Il suffit donc de montrer que $\forall 1 \leq i \leq n$, $(K \cap L)^{\uparrow c} \subseteq K_i^{\uparrow pc}$. On considère ainsi $i \in \{1, \ldots, n\}$ et on cherche à présent à montrer que $(K \cap L)^{\uparrow c} \subseteq K_i^{\uparrow pc}$. On considère à présent InfCont$((K \cap L)^{\uparrow c}, \Sigma_{uc}, K_i)$ et on rappelle que $(K \cap L)^{\uparrow c} \subseteq$ InfCont$((K \cap L)^{\uparrow c}, \Sigma_{uc}, K_i)$[7].

On cherche alors à montrer que InfCont$((K \cap L)^{\uparrow c}, \Sigma_{uc}, K_i)$ est partiellement contrôlable par rapport à $\Sigma_{i,uc}$, Σ_{uc}, K_i et $P_i^{-1}(L_i)$. En effet, dans ce cas, InfCont$((K \cap L)^{\uparrow c}, \Sigma_{uc}, K_i) \subseteq {K_i}^{\uparrow pc}$. De plus, comme $(K \cap L)^{\uparrow c} \subseteq$ InfCont$((K \cap L)^{\uparrow c}, \Sigma_{uc}, K_i)$ par définition, on obtiendrait alors que $(K \cap L)^{\uparrow c} \subseteq {K_i}^{\uparrow pc}$ et la preuve serait finie.

[7] On rappelle que InfCont$((K \cap L)^{\uparrow c}, \Sigma_{uc}, K_i)$ représente le plus petit langage préfixe-clos et contrôlable par rapport à $\Sigma_{i,uc}$ et K_i contenant $(K \cap L)^{\uparrow c}$

Montrons maintenant que InfCont$((K \cap L)^{\uparrow c}, \Sigma_{uc}, K_i)$ est partiellement contrôlable par rapport à $\Sigma_{i,uc}$, Σ_{uc}, K_i et $P_i^{-1}(L_i)$. Par définition, il suffit de montrer que ce langage est (i) contrôlable par rapport à $\Sigma_{i,uc}$ et $P_i^{-1}(L_i)$ et (ii) contrôlable par rapport à Σ_{uc} et K_i. Le point (ii) est évident par définition de InfCont$((K \cap L)^{\uparrow c}, \Sigma_{uc}, K_i)$. Il reste donc à montrer le point (i).

Soient $s \in$ InfCont$((K \cap L)^{\uparrow c}, \Sigma_{uc}, K_i)$ et $\sigma \in \Sigma_{i,uc}$ tels que $s\sigma \in P_i^{-1}(L_i)$. Donc

$$s\sigma \in \text{InfCont}((K \cap L)^{\uparrow c}, \Sigma_{uc}, K_i).\Sigma_{i,uc} \cap P_i^{-1}(L_i)$$

Or d'après 1.24, InfCont$((K \cap L)^{\uparrow c}, \Sigma_{uc}, K_i) = (K \cap L)^{\uparrow c}.\Sigma_{uc}^* \cap K_i$. Par conséquent, on considère $s' \in (K \cap L)^{\uparrow c}$ et $t \in \Sigma_{uc}^*$ tels que $s't = s$. Donc $s' \in L$ et $t\sigma \in \Sigma \setminus \Sigma_s$ (car $\Sigma_{uc} \cap \Sigma_s = \emptyset$ et $s't\sigma \in P_i^{-1}(L_i)$). Donc d'après le lemme 7, $s'P_i(t\sigma) \in L$. Or $(K \cap L)^{\uparrow c}$ est contrôlable par rapport à Σ_{uc} et L, donc $s'P_i(t\sigma) \in (K \cap L)^{\uparrow c}$. Puisque $(K \cap L)^{\uparrow c} \subseteq K_i$, on en déduit que

$$P_i(t\sigma) \in K_i(s')$$

Or $t \in \Sigma_{uc}^*$ et $\sigma \in \Sigma_{i,uc}$ donc $P_i(t)\sigma \in K_i(s', \Sigma_{i,uc})$. Puisque K est localement cohérent, on obtient que $t\sigma \in K_i(s', \Sigma_{uc})$ et donc $s\sigma\ (= s't\sigma) \in K_i$. Ce qui signifie que

$$s\sigma \in \text{InfCont}((K \cap L)^{\uparrow c}, \Sigma_{uc}, K_i).\Sigma_{uc} \cap K_i$$

Or InfCont$((K \cap L)^{\uparrow c}, \Sigma_{uc}, K_i)$ est contrôlable par rapport à Σ_{uc} et K_i, donc

$$s\sigma \in \text{InfCont}((K \cap L)^{\uparrow c}, \Sigma_{uc}, K_i)$$

On en déduit donc que InfCont$((K \cap L)^{\uparrow c}, \Sigma_{uc}, K_i)$ est contrôlable par rapport à Σ_{uc} et $P_i^{-1}(L_i)$ et on obtient ainsi le résultat. ◇

Comme le montre l'exemple 10, la réciproque du théorème 7 est fausse en général.

Exemple 10 *Soient $\Sigma_1 = \{a_1, uc_1\}$, $\Sigma_2 = \{a_2, uc_2\}$, et $\Sigma_{1,uc} = \{uc_1\}$ et $\Sigma_{2,uc}$. On considère les langages préfixe-clos L_1 et L_2 respectivement sur Σ_1 et Σ_2, définis par :*

$$L_1 = \overline{\{a_1.uc_1\}} \text{ et } L_2 = \overline{\{a_2.uc_2\}}$$

Soit $K = \overline{\{a_1.uc_2.uc_1, a_1.uc_1\}}$ un langage préfixe-clos sur $\Sigma = \Sigma_1 \cup \Sigma_2$.
On a

$$K_1 = K,\ K_2 = \overline{\{a_1.uc_1\}},\ K_1^{\uparrow pc} = K_1,\ K_2^{\uparrow pc} = K_2$$

Donc $K_1^{\uparrow pc} \cap K_2^{\uparrow pc} = \overline{\{a_1.uc_1\}}$ et est par conséquent contrôlable par rapport à Σ_{uc} et $L = L_1 \parallel L_2$. De plus, $K_1^{\uparrow pc} \cap K_2^{\uparrow pc} = K \cap L$, donc on a en fait $K_1^{\uparrow pc} \cap K_2^{\uparrow pc} = (K \cap L)^{\uparrow c}$.

Or la condition (2.4) n'est pas vérifiée. En effet, $a_1.uc_2 \in K_1$ et $a_1.P_1(uc_2).uc_1 \in K_1$ (car $P_1(uc_2) = \epsilon$). Pourtant, $a_1.uc_2.uc_1 \notin K_1$.

◇

2.3.3.2 Comparaison de la cohérence locale avec la séparabilité

Dans [77], les auteurs ont introduit la notion de séparabilité de langage. Lorsque cette condition est vérifiée par $(K \cap L)^{\uparrow c}$, alors il existe des superviseurs locaux assurant exactement $(K \cap L)^{\uparrow c}$. Ce résultat étant similaire à celui du théorème 7, il apparaît intéressant de comparer les notions de séparabilité de langages et de cohérence locale.

En reprenant les notations de la section 2.1.3.1, et étant donnés un ensemble d'alphabets $\Sigma_i = \Sigma_{i,c} \cup \Sigma_{i,uc}$ (avec $i \in \{1, \ldots, n\}$) et n langages $L_i \subseteq \Sigma_i^*$, la proposition 9 montre que la classe des langages vérifiant la condition (2.4) contient celle des langages séparables par rapport à $\{\Sigma_i\}_{1 \leq i \leq n}$.

Proposition 9 *Soient un ensemble d'alphabets $\Sigma_i = \Sigma_{i,c} \cup \Sigma_{i,uc}$ (avec $i \in \{1, \ldots, n\}$) et n langages $L_i \subseteq \Sigma_i^*$. On note $L = \|_i\ L_i$, $\Sigma = \bigcup_i \Sigma_i$, $\Sigma_{uc} = \bigcup_i \Sigma_{i,uc}$ et $\Sigma_c = \bigcup_i \Sigma_{i,c}$. De plus, on reprend les hypothèses de contrôlabilité de la section 2.1.1 (i.e la nature contrôlable des événements est identique pour chaque sous système) et on suppose que $\Sigma_s \subseteq \Sigma_c$.*

Si K est séparable par rapport $\{\Sigma_i\}_{1 \leq i \leq n}$, alors K est localement cohérent par rapport à Σ_{uc} et L.

Démonstration : Soit $K \subseteq \Sigma^*$. Supposons que K soit séparable par rapport à $\{\Sigma_i\}_{1 \leq i \leq n}$. On considère, pour $1 \leq i \leq n$, un langage $L_i' \subseteq \Sigma_i^*$ tel que

$$K = \|_{1 \leq i \leq n}\ L_i'$$

Soit $i = \{1, \ldots, n\}$. On note $K_i = K \cap P_i^{-1}(L_i)$. Soient $s \in K_i$, $s' \in K_i(s, \Sigma_{uc})$ et $\sigma \in \Sigma_{i,uc}$ tels que $P_i(s')\sigma \in K_i(s, \Sigma_{uc})$. Il suffit de montrer que $s'\sigma \in K_i(s, \Sigma_{uc})$. Or $s' \in K_i(s, \Sigma_{uc})$, donc $ss' \in K_i$. Donc $\forall j \neq i,\ P_j(ss') \in L_j'$. De plus, $\sigma \in \Sigma_i \setminus \Sigma_s$ (car $\Sigma_s \subseteq \Sigma_c$), donc $\forall j \neq i,\ P_j(ss'\sigma) = P_j(ss')$ et on a $P_j(ss'\sigma) \in L_j'$. Par conséquent, $\forall j \neq i,\ ss'\sigma \in P_j^{-1}(L_j')$. Il suffit finalement de montrer que $ss'\sigma \in P_i^{-1}(L_i')$.

Or $sP_i(s'\sigma) \in K_i$ puisque $P_i(s')\sigma \in K_i(s, \Sigma_{uc})$ et $P_i(s')\sigma = P_i(s'\sigma)$. Donc $sP_i(s'\sigma) \in K$ ce qui implique que $sP_i(s'\sigma) \in P_i^{-1}(L_i')$. Finalement, on déduit du lemme 7 que $ss'\sigma \in P_i^{-1}(L_i')$. D'où le résultat. ◇

On donne à présent un exemple montrant que sous l'hypothèse que $\Sigma_s \subseteq \Sigma_c$, la classe des langages séparables est strictement plus restreinte que celle des langages localement cohérents.

Exemple 11 *Soient $\Sigma_1 = \{a_1, uc_1\}$, $\Sigma_2 = \{a_2, uc_2\}$, et $\Sigma_{1,uc} = \{uc_1\}$ et $\Sigma_{2,uc} = \{uc_2\}$. On considère les langages préfixe-clos L_1 et L_2 respectivement sur Σ_1 et Σ_2, définis par :*

$$L_1 = \overline{\{a_1.uc_1\}} \text{ et } L_2 = \overline{\{a_2.uc_2\}}$$

Soit $K = \overline{\{a_1.uc_1, a_2.uc_2\}}$ un langage préfixe-clos sur $\Sigma = \Sigma_1 \cup \Sigma_2$. Pour $i = 1, 2$, $K_i = K$. En reprenant les notations de la définition 16,

s	$K_1(s, \Sigma_{uc})$	$P_1(s')uc_1$
ϵ	$\{\epsilon\}$	$\{\epsilon.uc_1\}$
a_1	$\{\epsilon, uc_1\}$	$\{\epsilon.uc_1, uc_1.uc_1\}$
$a_1.uc_1$	$\{\epsilon\}$	$\{\epsilon.uc_1\}$
$a_1.uc_1.a_2$	$\{\epsilon, uc_2\}$	$\{\epsilon.uc_1\}$
$a_1.uc_1.a_2.uc_2$	$\{\epsilon\}$	$\{\epsilon.uc_1\}$

Or, la seule valeur de (s, s') pour laquelle $P_1(s')uc_1 \in K_1(s, \Sigma_{uc})$ est : (a_1, ϵ). Et dans ce cas, $s'uc_1 = uc_1$, ce qui signifie que $s'.uc_1 \in K_1(s, \Sigma_{uc})$. On en déduit que K_1 est cohérent par rapport à P_1, Σ_{uc}. Puis en effectuant le même raisonnement pour $K_2(s, \Sigma_{uc})$, on en déduit que K est localement cohérent par rapport à Σ_{uc} et L.

Or K n'est pas séparable par rapport à $\{\Sigma_1, \Sigma_2\}$, car si K est de la forme $K = L'_1 \parallel L'_2$, alors $a_1 \in L'_1$ et $a_2 \in L'_2$. Par conséquent, $a_1.a_2$ appartiendrait à $L'_1 \parallel L'_2$. Pourtant $a_1.a_2$ n'appartient pas à K.

$\diamond$

La proposition 9 et l'exemple 11 montrent que la cohérence locale est une propriété strictement plus faible que la séparabilité. De plus, la vérification de la cohérence locale d'un langage possède une complexité inférieure à celle de la vérification de la séparabilité d'un langage. Les complexités relatives aux différentes notions introduites dans ce chapitre et à l'approche proposée sont discutées dans la section 2.3.3.3.

2.3.3.3 Complexité

Test de cohérence On considère un alphabet Σ, un langage préfixe-clos $M \in \mathcal{L}(\Sigma)$ et un automate $G = (\Sigma, Q, q_0, \delta)$ générant ce langage. On considère un sous ensemble Σ' de Σ et on note $\Sigma_{uc} \subseteq \Sigma$ l'ensemble des événements incontrôlables de G. De plus, on note $\Sigma'_{uc} = \Sigma_{uc} \cap \Sigma'$ et $P_{\Sigma'}$ la projection naturelle de Σ^* dans Σ'^*. On s'intéresse ici à la complexité engendrée lorsque l'on souhaite vérifier que M est cohérent par rapport à $P_{\Sigma'}$ et Σ_{uc}.

Pour cela, on va montrer que la cohérence de M par rapport à $P_{\Sigma'}$ et Σ_{uc} peut s'exprimer comme la contrôlabilité de $M(s, \Sigma_{uc})$ par rapport à Σ'_{uc} et $P_{\Sigma'}(M(s, \Sigma'_{uc}))$ pour tout $s \in M$. En effet, partant de la définition de la cohérence (définition 15),

$$
\begin{aligned}
& \forall s \in M, \forall s' \in M(s, \Sigma_{uc}), \forall \sigma \in \Sigma'_{uc},\ (P_{\Sigma'}(s')\sigma \in M(s, \Sigma'_{uc}) \Rightarrow s'\sigma \in M(s, \Sigma_{uc})) \\
\overset{(1)}{\iff}\ & \forall s \in M, \forall s' \in M(s, \Sigma_{uc}), \forall \sigma \in \Sigma'_{uc},\ (P_{\Sigma'}(s'\sigma) \in M(s, \Sigma'_{uc}) \Rightarrow s'\sigma \in M(s, \Sigma_{uc})) \\
\overset{(2)}{\iff}\ & \forall s \in M, \forall s' \in M(s, \Sigma_{uc}), \forall \sigma \in \Sigma'_{uc},\ \left(s'\sigma \in P^{-1}_{\Sigma'}(M(s, \Sigma'_{uc})) \Rightarrow s'\sigma \in M(s, \Sigma_{uc})\right) \\
\iff\ & \forall s \in M,\ \left(M(s, \Sigma_{uc}).\Sigma'_{uc} \cap P^{-1}_{\Sigma'}(M(s, \Sigma'_{uc})) \subseteq M(s, \Sigma_{uc})\right)
\end{aligned}
$$

L'équivalence (1) (resp. (2)) vient directement de la définition de $P_{\Sigma'}$ (resp. de $P^{-1}_{\Sigma'}$). L'expression droite de la dernière équivalence traduit que pour toute séquence s de M, $M(s, \Sigma_{uc})$ est contrôlable par rapport à Σ'_{uc} et $P^{-1}_{\Sigma'}(M(s, \Sigma'_{uc}))$.

On note N_M $(= |Q|)$ le nombre d'états de l'automate G. Un automate générant $M(s, \Sigma_{uc})$ et possédant dans le pire cas N_M états peut être obtenu. En effet, il suffit de calculer le produit entre

$G(q)$ (i.e G initialisé en $q\ (=\delta(q_0,s))$) et $G_{\Sigma_{uc}}$ ou $G_{\Sigma_{uc}}$ représente un automate générant Σ_{uc}^*. Or $G_{\Sigma_{uc}}$ peut être modélisé par un automate ne possédant qu'un seul état.

Finalement, $M(s,\Sigma'_{uc})$ peut aussi être généré par un automate possédant N_M états. Par conséquent, il en est de même pour $P_{\Sigma'}^{-1}(M(s,\Sigma'_{uc}))$. Ainsi, tester la cohérence de M par rapport à Σ_{uc} et $P_{\Sigma'}$ consiste pour tout $q\in Q$ à tester la contrôlabilité de $G(q)\cap G_{\Sigma_{uc}}$ (automate possédant N_M états dans le pire cas) par rapport à Σ'_{uc} et l'automate modélisant $P_{\Sigma'}^{-1}(M(s,\Sigma'_{uc}))$ (qui possède aussi N_M états dans le pire cas). Ainsi, la complexité d'une telle vérification vaut $\mathcal{O}(|\Sigma|.N_M^3)$.

Complexité globale de l'approche proposée Par conséquent, en reprenant les notations du théorème 7, tester la cohérence locale de K (généré par un automate possédant N_K états) par rapport à Σ_{uc} et L a une complexité en $\mathcal{O}(n.|\Sigma|.(N.N_K)^3)$. Cette complexité peut être notamment comparée à celle du calcul explicite d'un automate modélisant le langage L qui vaut $\mathcal{O}(|\Sigma|.N^n)$ dans le pire cas, où encore le test de séparabilité de K par rapport à $\{\Sigma_i\}_{1\leq i\leq n}$, qui vaut $\mathcal{O}(N_K^{n+1})$ [77] dans le pire cas.

Enfin, comme mentionné dans la section 2.2, pour $i\in\{1,\ldots,n\}$, le calcul de ${K_i}^{\uparrow pc}$ a une complexité en $\mathcal{O}(|\Sigma|.N_K^2.N^4)$, où N représente le nombre d'états de l'automate générant $P_i^{-1}(L_i)$ (N^4 vient du fait que le nombre d'états de l'automate K_i possède $N_K.N$ états dans le pire cas). Ainsi, la complexité globale de la méthode proposée dans cette section, lorsque les hypothèses du théorème 7 sont vérifiées, vaut $\mathcal{O}(n.|\Sigma|.N_K^2.(N_K+N))$ et est donc linéaire par rapport au nombre de composants du système. Notons que la méthode classique de synthèse (i.e en calculant G qui génère L et sans utiliser la structure concurrente du système) a une complexité en $\mathcal{O}(|\Sigma|.N_K.N^n)$ et est donc exponentielle par rapport au nombre de composants du système.

2.3.3.4 Exemple

Afin d'illustrer le théorème 7, on donne un exemple où le système à contrôler, noté G, est modélisé par la composition parallèle des automates G_1 et G_2 donnés en figure 2.10. On note $\Sigma_1=\{a,b,u,e\}$ (resp. $\Sigma_2=\{a',b',e'\}$) l'alphabet de G_1 (resp. G_2). De plus, seul u est supposé incontrôlable. Par conséquent, $\Sigma_{1,uc}=\{u\}$ et $\Sigma_{2,uc}=\emptyset$. Finalement, l'alphabet du système G est noté $\Sigma=\Sigma_1\cup\Sigma_2$ et pour $i=1,2$ on note P_i les projections naturelles de Σ sur Σ_i. L'ensemble des événements incontrôlables (resp. contrôlables) de G, noté Σ_{uc} (resp. Σ_c) est réduit à $\Sigma_{uc}=\{u\}$ (resp. $\Sigma\setminus\Sigma_{uc}$). On peut noter que $\Sigma_s\subseteq\Sigma_c$ puisqu'aucun événement n'est partagé par G_1 et G_2.

De plus, l'objectif de contrôle, noté K est aussi fournit en figure 2.10. On en déduit les langages $K_1=\overline{K}\cap P_1^{-1}(G_1)$ et $K_2=\overline{K}\cap P_2^{-1}(L_2)$ donnés en figure 2.11. Afin d'appliquer le théorème 7, il faut vérifier la cohérence locale de K.

Pour cela, pour $i=1,2$, il suffit de vérifier la cohérence de K_i par rapport à $\Sigma_{i,uc}$ et P_i. Puisque $\Sigma_{2,uc}=\emptyset$, la cohérence de K_2 par rapport à $\Sigma_{2,uc}$ et P_2 est immédiate. Pour montrer la cohérence de K_1 par rapport à $\Sigma_{1,uc}$ et P_1, il suffit de montrer que

$$\forall s\in K_1,\forall s'\in K_1(s,\Sigma_{uc}),\forall\sigma\in\Sigma_{1,uc},\ P_1(s')\sigma\in K_1(s,\Sigma_{1,uc})\Rightarrow s'\sigma\in K_1(s,\Sigma_{uc})$$

Par conséquent, à partir de l'automate générant K_1, il suffit de s'intéresser aux états depuis lesquels une séquence s' d'événements incontrôlables est tirable. Ainsi, seul l'état de K_1 atteint par

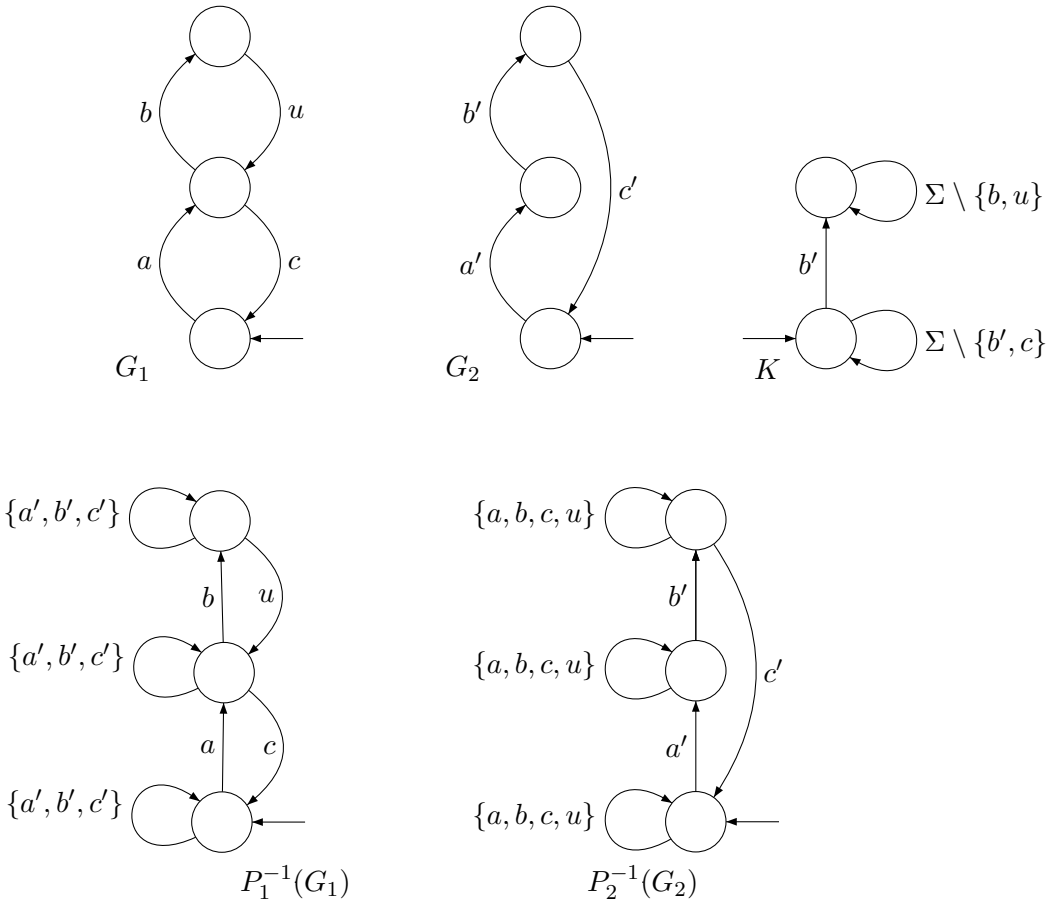


FIG. 2.10: Système à contrôler et objectif de contrôle

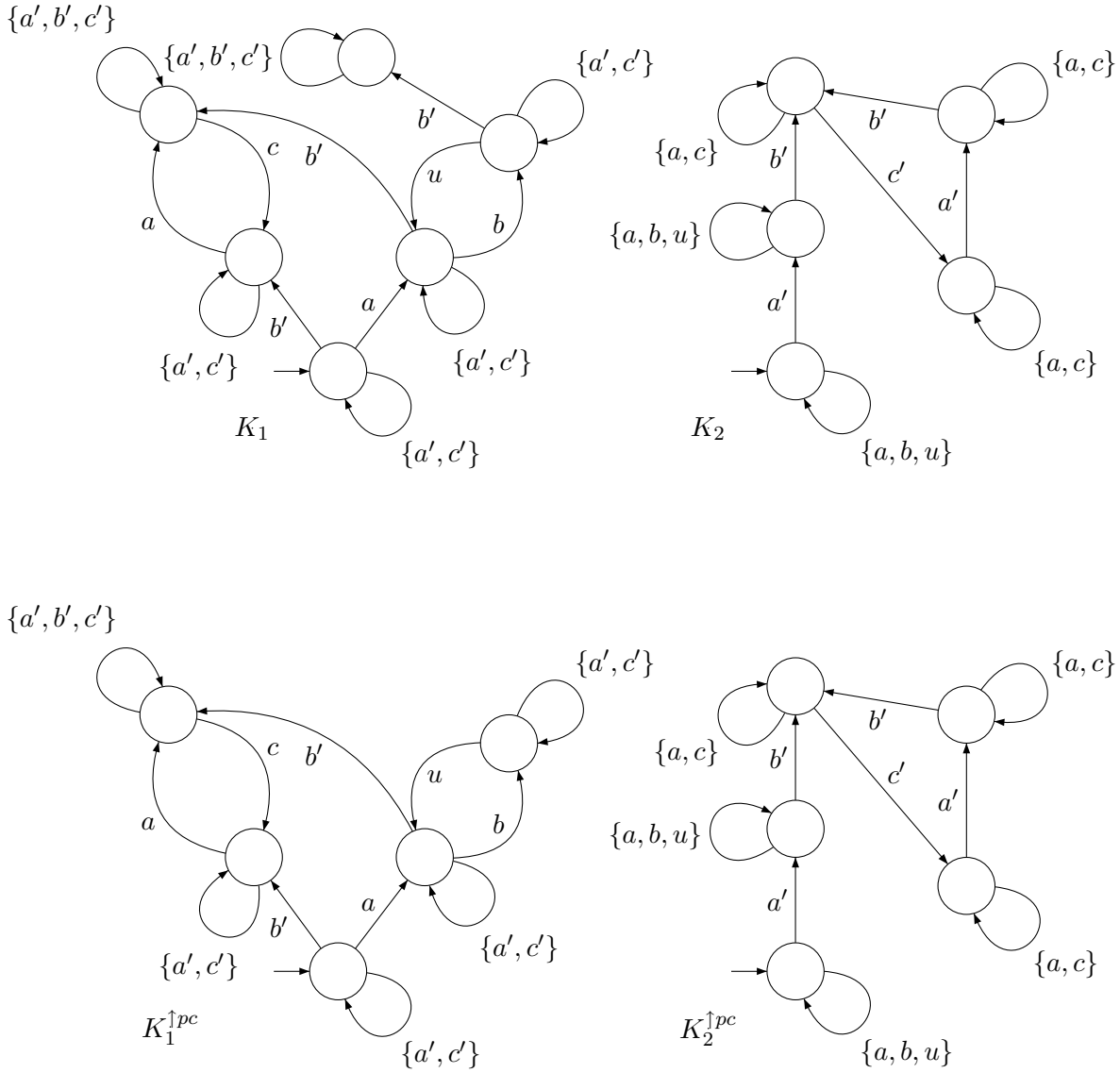


FIG. 2.11: Langages partiellement contrôlables

tirage de la séquence $s = ab$ est à considérer. De plus, $K_1(s, \Sigma_{uc}) = \{\epsilon, u\}$ et $\forall s' \in K_1(s, \Sigma_{uc})$, $\forall \sigma \in \Sigma_{1,uc}$,

$$P_1(s')\sigma \in K_1(s, \Sigma_{1,uc}) \Longrightarrow s' = \epsilon \text{ et } \sigma = u$$

Or $\epsilon.u(= u)$ appartient à $K_1(s, \Sigma_{uc})$, donc K_1 est cohérent par rapport à $\Sigma_{1,uc}$ et P_1 et le théorème 7 s'applique. Pour cela, pour $i = 1, 2$ on calcule le plus grand langage partiellement contrôlable par rapport à $\Sigma_{i,uc}$, Σ_{uc}, K_i et $P_i^{-1}(L_i)$ inclus dans K. Ce langage est noté ${K_i}^{\uparrow pc}$ et est donné par la figure 2.11. On peut noter que ${K_2}^{\uparrow pc} = K_2$ mais que ${K_1}^{\uparrow pc} \neq K_1$. En effet, K_1 n'est pas contrôlable par rapport à $\Sigma_{1,uc}$ et $P_1^{-1}(G_1)$ (la séquence $a.b.b'$ viole la condition de contrôlabilité). Finalement, d'après le théorème 7, ${K_1}^{\uparrow pc} \cap {K_2}^{\uparrow pc}$ est le plus grand langage contrôlable par rapport à Σ_{uc} et G qui soit inclus dans K.

Concernant la comparaison de ce résultat avec celui de [77], on peut noter que K n'est pas séparable par rapport à $\{\Sigma_1, \Sigma_2\}$ puisque l'entrelacement de b' et c n'est notamment pas assuré (c ne peut se produire si b' ne s'est pas produit auparavant). De plus, $(\overline{K} \cap L)^{\uparrow c}$ $(= {K_1}^{\uparrow pc} \cap {K_2}^{\uparrow pc})$ n'est pas non plus séparable par rapport à $\{\Sigma_1, \Sigma_2\}$ puisque ce langage admet la séquence $a.a'.b'.c$ et pas la séquence $a.c.a'.b'$.

2.4 Conclusion

Dans ce chapitre, nous nous sommes intéressés au *problème de base de la synthèse de contrôleurs* (PBSC) sur des systèmes concurrents. Le système concurrent G est obtenu par composition de sous systèmes G_i, modélisés par des langages. L'objectif de contrôle K représente lui aussi un langage. Afin d'éviter les problèmes de complexité liés au calcul explicite du système global G, une solution tirant parti de sa structure concurrente est envisagée. Toutefois, cette approche se démarque de l'approche décentralisée utilisée dans [77]. En effet, l'approche décentralisée consiste à déterminer des superviseurs locaux, agissant sur chacun des sous systèmes. En revanche, l'approche adoptée ici consiste à déterminer un superviseur centralisé, mais décrit de manière modulaire. Contrairement au cadre décentralisé, chaque composant de ce superviseur peut observer et agir sur les événements d'un sous système quelconque. Afin de pouvoir résoudre le problème de contrôle, nous avons été amenés à poser certaines hypothèses restrictives sur la nature contrôlable des événements. Tout d'abord, il est supposé que le caractère contrôlable des événements est le même quel que soit le sous système qui le partage. Par conséquent, un événement ne peut être contrôlable du point de vue d'un des sous systèmes, et incontrôlable du point de vue d'un autre. De plus, il est supposé qu'aucun événement partagé n'est incontrôlable. En d'autres termes, l'occurrence de n'importe quelle synchronisation du système peut être empêchée. Enfin, notons que ces conditions sont également supposées dans [77].

Comme dans le cadre décentralisé, le problème global (PBSC) est ramené à n sous problèmes locaux. Toutefois, pour le $i^{\text{ème}}$ sous problème, au lieu de chercher à contrôler directement le sous-système G_i, nous avons choisi de contrôler une sur-approximation de celui-ci, à savoir $P_i^{-1}(G_i)$, de manière à ce que celui-ci respecte l'objectif de contrôle général K. En fait $P_i^{-1}(G_i)$ peut être vu comme une approximation de G telle que le composant G_i peut le percevoir. Le problème du contrôle revient alors à déterminer n superviseurs S_i sur $P_i^{-1}(G_i)$ tels que $\bigcap_i \mathcal{L}(S_i/P_i^{-1}(G_i))$

soit solution du problème initial (PBSC sur le système global). Toutefois, résoudre le PBSC sur chacun de ces sous systèmes ne permet pas d'obtenir une solution globale satisfaisante. En fait, la contrôlabilité ne constitue pas la notion adéquate pour travailler localement. C'est pour lever ce problème que nous avons été amenés à introduire la notion de *contrôlabilité partielle*. L'idée est alors de restreindre chacune des approximations $P_i^{-1}(G_i)$ à l'objectif de contrôle initial K (ou du moins à un sous-ensemble de celui-ci). Tout comme la contrôlabilité, cette propriété est stable par union, ce qui implique, pour le $i^{\text{ème}}$ sous problème, l'existence d'un unique plus grand langage, noté ${K_i}^{\uparrow pc}$, partiellement contrôlable et inclus dans K et $P_i^{-1}(G_i)$. Le théorème 5 indique alors que $\bigcap_i {K_i}^{\uparrow pc}$ est contrôlable et inclus dans K. Toutefois, ce langage n'est pas maximal en général. Les deux dernières sections de ce chapitre répondent à ce problème et donnent des conditions portant sur l'objectif de contrôle, et assurant la maximalité.

- Une première condition correspond au cas où l'objectif de contrôle modélise un sous ensemble des comportements du système G. Dans ce cas, le théorème 6 $\bigcap_i {K_i}^{\uparrow pc}$ est une solution du PBSC global.
- Une seconde condition, appelée *cohérence locale* garantit aussi un tel résultat (Théorème 7). Notons que la cohérence locale ne nécessite pas que l'objectif de contrôle représente un sous ensemble des comportements du système. De plus la classe des langages vérifiant cette propriété est plus large que la classe des langages séparables introduites dans [77], utilisée pour caractériser la solution du problème dans le cadre d'une approche décentralisée.
  Notons que dans [24, 25], nous avions défini une condition sur l'objectif K plus forte appelée G-observabilité (dérivée de la condition d'observabilité introduite par [49]). Si cette condition est vérifiée par l'objectif, alors notre méthode permet de calculer $(\overline{K} \cap L)^{\uparrow c}$.

L'approche adoptée dans ce chapitre présente plusieurs intérêts, que l'on retrouvent notamment dans les travaux décrits dans [77]. Tout d'abord, un intérêt concernant la complexité des calculs et des systèmes manipulés : le système global n'est jamais explicité et tous les calculs sont effectués localement (i.e à partir de chacun des sous systèmes). De plus, vérifier que l'objectif de contrôle satisfait les conditions assurant que l'approche utilisée est valide, s'effectue uniquement à partir de calculs locaux. D'un autre côté, comme dans l'approche décentralisée, les calculs effectués sont réutilisables. Par conséquent, si un superviseur a été calculé et qu'un des sous systèmes doit être remplacé par un nouveau sous système, seuls les calculs concernant ce sous système doivent être effectués à nouveau.

Perspectives

- On peut se poser la question de savoir comment corriger l'objectif lorsque que celui-ci ne vérifie pas la condition de cohérence locale. En d'autres termes, il semble intéressant de regarder si il existe un plus grand (plus petit) langage cohérent inclus dans (contenant) le langage objectif initial.
- Dans le même esprit, la condition de cohérence locale ne s'avère être qu'une condition suffisante à l'obtention d'une solution maximale. Il serait donc intéressant de pouvoir exactement caractériser la classe des langages objectifs offrant une telle solution.
- Dans ce chapitre, seul le problème de base de la synthèse de contrôleurs a été étudié. Le résultat, i.e. le système contrôlé peut donc être bloquant. Il semble donc intéressant d'étendre notre méthodologie au cadre non-bloquant.
- Nous avons fait l'hypothèse que les événements partagés étaient contrôlables. Or dans [36],

les auteurs ont montré que cette hypothèse n'était par nécessaire avec une approche décentralisée pour montrer l'existence de superviseurs contrôlant le système global de manière à restreindre exactement celui-ci à l'objectif de contrôle. Même si au contraire de la méthode présentée dans ce chapitre, cette méthode n'est pas constructive, on peut se poser la question de savoir si l'hypothèse de localisation des événements incontrôlables à un composant particulier ne peut être levée.

– Enfin, dans un tout autre ordre d'idée, nous pensons que notre méthodologie peut être appliquée à la génération automatique de tests [6, 37] pour des systèmes concurrents. Ce travail est actuellement en cours de réalisation.

Chapitre 3

Contrôle de systèmes structurés : approche États

Introduction

Dans le chapitre précédent, nous nous sommes intéressés au problème du contrôle de systèmes concurrents pour des objectifs de contrôle exprimés sous forme de langages. Cependant, il se peut que certaines propriétés que l'on souhaite assurer par contrôle soient liées à la structure même du système, et plus particulièrement à la notion d'état. Ainsi, certains objectifs de contrôle s'expriment naturellement par un ensemble d'états dont on souhaite assurer l'interdiction/invariance par contrôle (voir section 1.2.2.4). Bien que le problème de l'interdiction d'un ensemble d'états puisse se ramener à celui d'un ensemble de comportements (l'ensemble des comportements qui mènent aux états n'appartenant pas à l'ensemble dont on souhaite assurer l'interdiction), il est intéressant de pouvoir traiter ce problème par une approche différente de celle présentée dans le chapitre précédent. En effet, supposons par exemple que l'on souhaite assurer l'interdiction d'un seul état q. Il faut dans ce cas pouvoir exprimer un automate, dont les comportements contiennent tous ceux du système mis à part ceux menant à q. Un tel automate s'obtient facilement, dès lors que l'on possède un unique automate G modélisant les comportements du système (il suffit de retirer de G l'ensemble des transitions dont q est "la cible"). Or comme précisé précédemment, l'obtention de G nécessite des calculs dont la complexité (notamment en mémoire) est trop importante pour être réalisés en pratique. Cette constatation nous amène à nous intéresser au problème spécifique de la synthèse de contrôleurs pour des objectifs portant sur les états d'un système concurrent.

Des travaux sur l'interdiction d'états pour des systèmes modélisés par un automate (possédant potentiellement un nombre infini d'états) ont déjà été réalisés dans ce cadre [58] (C.f. Section 1.2.2.4). Les superviseurs considérés sont définis sur l'ensemble des états du système plutôt que sur l'ensemble de ses comportements. Toutefois, le système possédant potentiellement un nombre infini d'états, la réalisation d'un superviseur[1] ne peut alors être effectuée en générale. La solution au problème de synthèse de contrôleurs dans ce cadre, consiste donc à déterminer

[1] La réalisation d'un superviseur est décrite en section 1.2.2.4.

un superviseur résolvant le problème posé, et à l'évaluer *en-ligne* (alors que la réalisation d'un superviseur est supposée être entièrement déterminée statiquement).

Toutefois l'évaluation en-ligne du superviseur, agissant sur un système possédant un nombre infini d'états, nécessite que la structure du système/modèle permette d'assurer l'effectivité des calculs. Ainsi dans [45], les auteurs ont introduit et traité le cas des VDES (Vector Discrete Event System) dont la structure algébrique permet de réaliser l'évaluation en ligne d'un superviseur alors que le nombre d'états du système est infini. Des résultats ont aussi été obtenus dans le cadre des réseaux de Petri [32], qui possèdent eux aussi une structure algébrique et un nombre potentiellement infini d'états. Notons enfin que dans les travaux précédemment cités, l'approche ne s'applique pas lorsque E représente un ensemble quelconque d'états. Il faut en effet que l'ensemble des états considérés "colle" à la structure du système.

Dans [54, 53], les auteurs se sont intéressés à la synthèse de contrôleurs sur des systèmes modélisés par une composition parallèle d'automates, mais agissant indépendemment les uns par rapport aux autres (systèmes modulaires). Dans ce cadre, les auteurs donnent une approche permettant de traiter un objectif de contrôle modélisé par un automate. Les calculs permettant d'effectuer la synthèse sont découpés en deux phases : une phase hors-ligne et une phase en-ligne. Une partie des calculs nécessaires à l'obtention de l'expression d'un superviseur est ainsi effectuée hors-ligne. Ces calculs possèdent une complexité très inférieure à celle du calcul explicite d'un automate modélisant le système global. Toutefois une évaluation en-ligne du superviseur, nécessitant des calculs est nécessaire. La structure compositionnelle du système, ainsi que l'hypothèse d'indépendance entre les sous systèmes qui le composent, permettent alors de limiter la complexité de ces calculs.

Dans ce chapitre, on s'intéresse dans un premier temps au problème de l'interdiction d'états sur un système concurrent, i.e. modélisé par une composition parallèle d'automates. Puisque ce type de système peut s'avérer très complexe et posséder un nombre d'états trop important pour être traité de manière monolithique, nous nous intéressons ici à une approche basée sur les techniques développées dans le cadre des systèmes infinis [58]. En particulier, l'évaluation en ligne du superviseur est préférée au calcul statique d'une réalisation de celui-ci. Toutefois, puisque les systèmes considérés sont finis, on souhaite minimiser les contraintes concernant l'objectif de contrôle et pouvoir interdire un ensemble quelconque d'états. En fait, l'approche proposée ici est aussi proche de celle décrite dans [54]. D'une part, parce que les systèmes considérés sont concurrents (ils sont en fait modulaires, i.e les sous systèmes ne partagent pas d'événements). D'autre part, parce que l'approche consistant à décomposer les calculs en deux phases (en-ligne et hors-ligne) est également appliquée. Cependant, les objectifs considérés dans [54] sont donnés sous forme de langages, exprimant ainsi un ensemble de comportements désirés, alors qu'on s'intéresse ici à des propriétés d'états. D'un autre coté, dans la plupart des problèmes de contrôle, les systèmes à événements discrets sont constitués d'un grand nombre de composants qui interagissent de manière concurrente. À cette composition parallèle, on peut rajouter un autre type de composition : la hiérarchie. Il est en effet fréquent de rencontrer des systèmes à la fois spécifiés à partir de sous-systèmes agissant en parallèle, et spécifiés par "couche" en donnant des représentations du système à différents niveaux d'abstraction. À cet effet, des techniques basées sur l'agrégation d'états ont été proposées dans e.g. [78] afin de réaliser du contrôle hiérarchique. Notre démarche ici est inverse. Le système est spécifié hiérarchiquement par un

modèle simplifié des STATECHARTS [30], les machines à états finis hiérarchiques (abrégé HSFM pour Hierarchical Finite State Machines). Le modèle que nous considérons peut être caractérisé par une collection de structures imbriquées $\langle K_1, \ldots, K_n \rangle$, où K_1 représente le plus haut niveau de la HFSM. À un niveau intermédiaire, la structure K_i est une HFSM, pour laquelle les états sont soit des "états ordinaires" soit des "macro-états b" qui sont constitués d'un ensemble de structures $(K_j)_{j \in J_b} \subseteq 2^{\langle K_{i+1}, \ldots, K_n \rangle}$, évoluant en parallèle. Chaque structure peut avoir plusieurs états initiaux (resp. finaux). Le comportement d'une structure est le suivant : lorsque le système transite dans un macro-état b, toutes les structures associées à b sont activées et initialisées dans un de leurs états initiaux. *A contrario*, la sortie d'un macro-état est synchronisée avec la fin des tâches associées aux différentes structures de b (i.e. chaque structure est dans un état final). Entre deux états (ordinaire ou macro), le comportement de la structure est similaire à celui d'un automate classique. En Section 3.4.1, nous proposons des algorithmes qui synthétisent des superviseurs assurant l'interdiction d'ensembles d'états du système (cf. [9, 27, 43] pour des travaux connexes). Le superviseur global est déterminé à partir de superviseurs locaux, bien que l'objectif initial porte sur le système global. Ce superviseur peut être vu comme un oracle qui active/déactive les superviseurs locaux en fonction de l'état courant du système. De plus, la structure du superviseur reflète celle du système, permettant ainsi d'améliorer la lisibilité et la compréhension des effets du contrôle, et de minimiser la place mémoire nécessaire pour stocker le superviseur. Finalement, la méthode proposée ici peut être vue comme une alternative, lorsque le système à contrôler est trop complexe pour être décrit par un automate, et que l'objectif de contrôle est trop général pour être traité uniquement à partir de calculs locaux.

Plus précisément, ce chapitre s'organise de la manière suivante :

- dans un premier temps, afin de faciliter la compréhension de l'approche, ainsi que son intérêt algorithmique, le cas où les sous systèmes G_i ne partagent que des événements contrôlables est d'abord étudié. La méthode utilisée est basée sur la décomposition de l'ensemble des états faiblement interdits en sous-ensemble faiblement interdits locaux à chaque sous-système (ce calcul est effectué hors-ligne) et en une recomposition de celui-ci en-ligne de manière à savoir quel est l'ensemble des événements qui doivent être interdits dans un état donné.
- Dans un second temps, nous prenons en compte les synchronisations incontrôlables entre les sous systèmes. Ceci nous amène à raffiner les calculs des ensembles des états faiblement interdits locaux.
- Une méthode utilisant la structure particulière du superviseur obtenu, permettant de résoudre les problèmes de vivacité est présentée. Enfin, la complexité algorithmique des calculs hors-ligne et en-ligne sera discutée. Il sera notamment mis en avant que sous les hypothèses introduites, la complexité de ces calculs dépend davantage de la "forme" de l'objectif E et du nombre d'événements partagés par les sous systèmes G_i, que du nombre d'états de G.
- Finalement, nous étendons ces résultats au cadre des systèmes à événements discrets modélisés par des automates hiérarchiques.

3.1 Modèle du système et problème de synthèse

3.1.1 Le modèle.

Tout comme dans le chapitre précédent, nous nous intéressons à des systèmes concurrents dont chacun des sous systèmes est modélisé par un automate. On considère ainsi n automates $(G_i)_{1\leq i\leq n}$ avec pour tout $i \in \{1,\dots,n\}$, $G_i = (\Sigma_i, Q_i, q_{0i}, Q_{mi}, \delta_i)$. Le système global est modélisé par la composition parallèle des automates $(G_i)_i$. On note $G = (\Sigma, Q, q_0, Q_m, \delta)$ l'automate modélisant le système global :

$$G = G_1 \parallel \dots \parallel G_n, \text{avec } \Sigma = \bigcup_{i\leq n} \Sigma_i$$

Par la suite, les états de G seront notés $q = (q_1, \dots, q_n)$, où $q_i \in Q_i$.

Statuts des événements. La synthèse de contrôleurs sur des systèmes concurrents fait intervenir des sous ensembles pertinents de l'alphabet du système : l'ensemble des événements partagés, contrôlables et incontrôlables.

Si $\mathcal{A}$ représente un ensemble d'automates, on note $\Sigma_s(\mathcal{A})$ l'ensemble des événements partagés par les automates de $\mathcal{A}$:

$$\Sigma_s(\mathcal{A}) = \bigcup_{G,G'\in\mathcal{A}} (\Sigma_G \cap \Sigma_{G'})$$

Lorsque $\mathcal{A}$ représentera sans ambiguïté l'ensemble des sous systèmes d'un système concurrent, $\Sigma_s(\mathcal{A})$ sera simplement noté Σ_s.
De plus, étant donné un ensemble d'automates $(G_i)_{i\leq n}$, on introduit la fonction IN définie par

$$\begin{array}{rl} \text{IN} : & \Sigma \to 2^n \\ & \sigma \to \{i \in [1..n] \,|\sigma \in \Sigma_i\} \end{array} \tag{3.1}$$

IN$(.)$ est une fonction qui pour chaque événement $\sigma \in \Sigma$ rend l'ensemble des sous systèmes qui partagent cet événement.

Pour chaque $1 \leq i \leq n$, l'alphabet de G_i se décompose en $\Sigma_i = \Sigma_{i,c} \bigcup \Sigma_{i,uc}$ où $\Sigma_{i,c}$ et $\Sigma_{i,uc}$ représentent respectivement l'ensemble des événements contrôlables et incontrôlables de G_i. Comme dans le chapitre 2, on suppose que

$$\forall i \neq j,\ \Sigma_{i,uc} \cap \Sigma_{ic} = \emptyset$$

En d'autres termes, un événement ne peut être contrôlable dans un sous système et incontrôlable dans un autre. Les motivations de cette restriction sont données dans la section 2.1.2 et restent valables dans ce chapitre. Basé sur cette hypothèse, les ensembles d'événements contrôlables et incontrôlables de $G = G_1 \parallel \dots \parallel G_n$ sont respectivement donnés par

$$\Sigma_c = \bigcup_i \Sigma_{i,c} \text{ et } \Sigma_{uc} = \bigcup_i \Sigma_{i,uc}.$$

3.1.2 Présentation du problème de contrôle

On s'intéresse au problème de l'interdiction d'états tel que décrit en section 1.2.2.4, sur des systèmes concurrents. Les systèmes concurrents possèdent une structure particulière. Les états d'un tel système sont notamment représentés par des tuples d'états de chacun des sous systèmes. Nous avons vu en Section 1.2.2.4 que le problème de l'interdiction d'un ensemble d'états E se réduit à celui du calcul de l'ensemble des états faiblement interdits $\mathcal{I}(E)$, qui représente l'ensemble des états du système pouvant mener à E par tirage d'une séquence d'événements incontrôlables.

$$\mathcal{I}(E) = CoReach^{G}_{\Sigma_{uc}}(E)$$

Un superviseur optimal assurant l'interdiction de E est alors simplement donné par

$$S_E(q) = \{\sigma \in \Sigma_c |\ \delta(q,\sigma)! \wedge \delta(q,\sigma) \in \mathcal{I}(E)\} \tag{3.2}$$

Le calcul de $\mathcal{I}(E)$ est théoriquement possible à réaliser directement sur le système global G dès lors que celui-ci peut être représenté efficacement par un unique automate (i.e que le nombre d'états de l'automate résultant de la composition entre les différents automates G_i, modélisant chacun des sous systèmes, n'est pas trop important pour que le problème puisse être traité en pratique), ce qui n'est en général pas le cas pour les systèmes concurrents. De plus l'expression du superviseur sous forme d'une fonction requiert une évaluation en-ligne pour déterminer l'ensemble des événements qu'il faut interdire dans un état donné (cette évaluation se réduit à l'évaluation de l'appartenance d'états à $\mathcal{I}(E)$). Cette évaluation est d'autant plus efficace que $\mathcal{I}(E)$ est efficacement représenté. C'est pourquoi, le but de cette section consiste à donner une approche utilisant la structure du système pour

- d'une part déterminer et calculer efficacement $\mathcal{I}(E)$ hors-ligne,
- d'autre part vérifier l'appartenance à $\mathcal{I}(E)$ de manière efficace lors de l'évaluation en-ligne du superviseur.

Par conséquent, dans ce qui suit, la problématique consiste à utiliser la structure du système à contrôler pour déterminer une représentation pertinente de $\mathcal{I}(E)$.

3.1.2.1 Modélisation des états interdits

Dans cette section, nous nous intéressons au problème de l'interdiction d'états pour des systèmes modélisés par des systèmes concurrents. Dans ce cadre, le but d'un superviseur sera de coordonner l'évolution de chacun des ces sous-systèmes les uns par rapport aux autres de manière à ce qu'ils n'évoluent pas dans une configuration dangereuse pour le système global. Prenons l'exemple d'un système composé d'une presse et d'un bras articulé, dont le but est de placer (enlever) un objet dans (de) la presse. Chacun des sous-systèmes peut être modélisé efficacement par un automate. On peut remarquer que les états du système global tels que "*la presse est fermée*" alors que "*le bras est étendu*" sont clairement à éviter au cours de l'exécution du système. Ainsi, dans la pratique, la plupart des objectifs de contrôle seront d'empêcher le système d'atteindre des configurations particulières du système global pouvant se décomposer localement sur chacun des sous-systèmes (dans l'exemple précédent : *la position étendue* pour le sous-système bras et *la position fermée*

pour le sous-système presse). Séparément, ils ne correspondent pas à des configurations dangereuses. Ce n'est que lorsque les sous-systèmes se trouvent simultanément dans ces configurations particulières que le système global est dans une situation dangereuse, donc à interdire.

L'exemple ci-dessus traduit le fait que certains problèmes s'expriment "naturellement" par des interdictions d'états de la forme

$$E = E_1 \times \cdots \times E_n \text{ avec } E_i \subseteq Q_i,\ \forall i \leq n$$

Un tel ensemble d'états sera appelé un pavé par la suite. Notons que la notion de pavé englobe celle de tuple (il suffit que chaque E_i soit un singleton). Toutefois, un pavé ne permet pas de modéliser un ensemble quelconque d'états du système G. Ceci conduit à s'intéresser par la suite à des ensembles d'états de la forme

$$E = \bigcup_{j \leq m} E^j,\ \text{avec } \forall j \leq m,\ E^j = E_1^j \times \cdots \times E_n^j \text{ et } \forall i \leq n,\ E_i^j \subseteq Q_i \tag{3.3}$$

De plus, une telle expression de l'ensemble d'états à interdire est légitime en général, comme le suggère l'exemple 12 introduit par [33].

Exemple 12 *Considérons l'exemple classique des chariots filoguidés introduits par [33]. Le système à contrôler modélise une chaîne de production manufacturière, composée de* 5 *stations de travail (WST1, …, WST5) interagissant par l'intermédiaire de 5 chariots filoguidés. Ces chariots se déplacent sur des rails et transportent du matériel entre deux stations de travail. Il existe des zones de conflits sur le réseau de circulation des chariots, où deux chariots ne peuvent se situer simultanément. La figure 12 représente schématiquement le système à contrôler. Les zones de conflit sont quant à elles modélisées par les rectangles noirs sur des parties du réseau.*

Le but du problème consiste à synthétiser un superviseur assurant que deux chariots ne se situent jamais simultanément dans une zone de conflit. Chaque composant du système peut être modélisé par un automate (AGV_i, $i = 1, \ldots, 5$ pour les chariots, et WST_i, $i = 1, \ldots, 5$ pour les stations de travail). Le système global peut alors être modélisé par la composition parallèle de ces automates et possède plus de 3.10^7 *états. De plus l'objectif de contrôle peut être modélisé par un ensemble d'états à interdire. Plus précisément, l'ensemble des états à interdire se modélise naturellement comme une union de pavés $E = \cup_{1 \leq i \leq 4} Zone_i$, avec*

$$\begin{aligned}
Zone_1 &= E_1^1 \times E_1^2 \times Q_{AGV_3} \times Q_{AGV_4} \times Q_{AGV_5} \times \Pi_{1 \leq i \leq j} Q_{WST_i} \\
Zone_2 &= Q_{AGV_1} \times E_2^2 \times E_2^3 \times Q_{AGV_4} \times Q_{AGV_5} \times \Pi_{1 \leq i \leq j} Q_{WST_i} \\
Zone_3 &= Q_{AGV_1} \times E_3^2 \times Q_{AGV_3} \times E_3^4 \times Q_{AGV_5} \times \Pi_{1 \leq i \leq j} Q_{WST_i} \\
Zone_4 &= Q_{AGV_1} \times Q_{AGV_2} \times Q_{AGV_2} \times E_4^4 \times E_4^5 \times \Pi_{1 \leq i \leq j} Q_{WST_i}
\end{aligned}$$

où E_j^i correspond à l'ensemble des états de chaque AGV_i modélisant le fait que le $i^{ème}$ AGV est situé dans la zone de confit j. $Zone_i$ est un pavé encodant l'ensemble des états du système global dans lesquels deux AGV sont situés en même temps dans la zone de conflit i, quelque soit la position des autres AGV. ◇

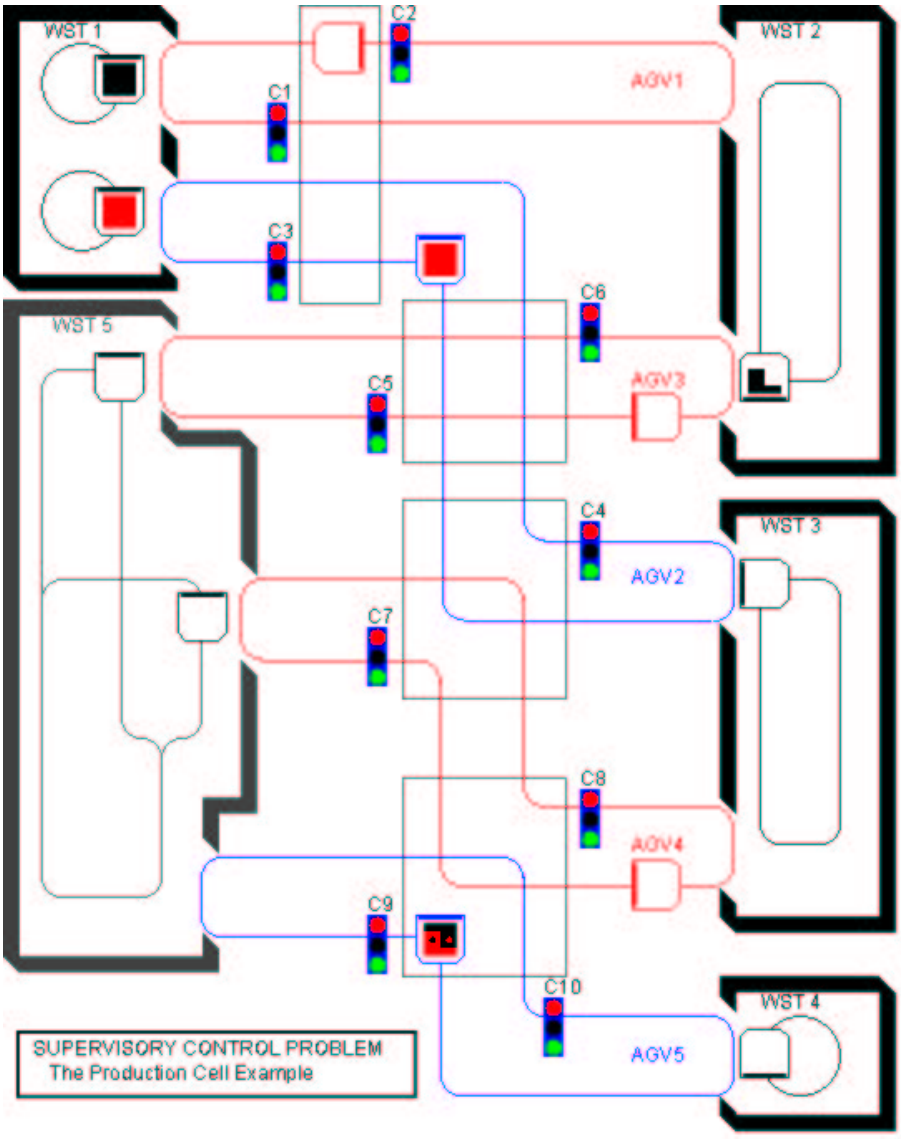


FIG. 3.1: Exemple des chariots filoguidés

Deux remarques peuvent être faites à ce niveau :

- On suppose que E est exprimé sous cette forme et qu'aucun traitement ne lui est appliqué (en particulier, on ne cherche pas à minimiser m).
- Les ensembles d'états de la forme exprimée par (3.3) permettent de représenter n'importe quel ensemble d'états du système global G.

Ainsi, l'objectif d'interdiction d'états considéré est général, au sens où n'importe quel ensemble d'états peut être considéré. De plus, comme on va le voir par la suite, l'approche proposée tire efficacement parti d'une expression de E sous la forme de (3.3). En particulier, on va voir que la complexité de cette approche est davantage liée au nombre de pavés m représentant E, plutôt qu'au nombre d'états du système ou même au nombre d'états représentés par E. Enfin, on peut noter que l'objectif de contrôle s'apparente à un problème d'exclusion mutuelle faisant intervenir plusieurs systèmes.

3.1.2.2 Propriétés de l'opérateur Pre sur des systèmes concurrents

La résolution du problème de l'interdiction d'états E pour un système G est principalement basée sur le calcul de l'ensemble des états faiblement interdits $\mathcal{I}(E)$. Or celui-ci s'obtient par un calcul de point fixe utilisant l'opérateur Pre_A^G. Le fait que nous travaillons sur un système concurrent $G = G_1 \parallel \ldots \parallel G_n$, nous amène dans un premier temps à regarder comment cet opérateur peut se "*distribuer*" par rapport à chacun des opérateurs $Pre_A^{G_i}$.

Proposition 10 *Soient $G = G_1 \parallel \ldots \parallel G_n$ un système concurrent et $(E_i)_{1 \leq i \leq n}$ des ensembles d'états tels que E_i appartiennent à G_i, alors*

$$Pre_{\{\sigma\}}^G(E_1 \times \cdots \times E_n) = \left(\mathop{\times}_{i \in \text{IN}(\sigma)} Pre_{\{\sigma\}}^{G_i}(E_i) \right) \times \left(\mathop{\times}_{k \notin \text{IN}(\sigma)} E_k \right) \tag{3.4}$$

et

$$Pre_A^G(E_1 \times \cdots \times E_n) = \bigcup_{\sigma \in A} Pre_{\{\sigma\}}^G(E_1 \times \cdots \times E_n) \tag{3.5}$$

De plus si $A \subseteq \Sigma \setminus \Sigma_s$, alors

$$\left(Pre_A^G\right)^*(E_1 \times \cdots \times E_n) = \left(Pre_{\Sigma_1 \cap A}^G\right)^*(E_1) \times \cdots \times \left(Pre_{\Sigma_n \cap A}^G\right)^*(E_n) \tag{3.6}$$

Remarque 9 *Notons que, dans la proposition 10, nous utilisons un abus de notation qui sera aussi appliqué dans la suite de ce chapitre. Pour simplifier l'écriture, l'expression à droite de l'égalité de l'équation (3.4) est factorisée, mais ne reflète plus l'ordre des éléments donné par la partie gauche de l'équation. On supposera par la suite qu'une réorganisation implicite de l'ordre des éléments est effectuée en partie droite.*

3.2 Problème de l'interdiction d'états

Dans le cadre de ce chapitre, nous nous intéressons au problème de l'interdiction d'états sur un système concurrent $G = G_1 \parallel \cdots \parallel G_n$. Dans un premier temps, nous reprenons les hypothèses concernant les événements partagés que nous avions dans le chapitre 2 et supposons que les événements partagés sont tous contrôlables. Cette hypothèse nous permet de fortement délocaliser les calculs de l'ensemble des états faiblement interdits sur chaque composant du système. Nous étendons par la suite nos résultats au cadre général.

3.2.1 Le cas $\Sigma_s \subseteq \Sigma_c$

Soit $G = G_1 \parallel \cdots \parallel G_n$ le système concurrent à contrôler, tel que ses événements incontrôlables ne sont pas partagés (i.e. $\Sigma_s \subseteq \Sigma_c$). Notre but est de résoudre le problème de l'interdiction d'états pour un ensemble de la forme

$$E = \cup_{1 \leq j \leq m} E^j \text{ avec } E^j_j = E^j_1 \times \cdots \times E^j_n \text{ et } \forall i,\ E^j_i \subseteq Q_i.$$

On rappelle que résoudre ce problème consiste à déterminer l'ensemble des états faiblement interdits $\mathcal{I}(E)$ puisque le superviseur décrit en Section 1.2.2.4 par l'expression 3.2 en est une solution. Plus précisément, on souhaite ici utiliser la structure du système pour déterminer l'appartenance à $\mathcal{I}(E)$ d'un état du système G, de manière plus efficace.

3.2.1.1 Calcul efficace de $\mathcal{I}(E)$ et du superviseur.

On s'intéresse tout d'abord au cas où l'ensemble des états à interdire est réduit à un pavé de la forme

$$E = E_1 \times \cdots \times E_n \text{ (avec } E_i \subseteq Q_i\text{)}$$

et on étendra ensuite les résultats à un ensemble quelconque E d'états tels que décrit en 3.3. La proposition 11 montre que l'opérateur $\mathcal{I}(\cdot)$ se distribue par rapport au produit cartésien représentant un pavé d'états d'un système concurrent.

Proposition 11 *Soit $G = G_1 \parallel \ldots \parallel G_n$ avec $G_i = (\Sigma_i, Q_i, q_{0_i}, Q_{m_i}, \delta_i)$, tel que $\Sigma_s \subseteq \Sigma_c$. Soit E un ensemble d'états de G de la forme $E = E_1 \times \cdots \times E_n$, avec $E_i \subseteq Q_i$. Alors,*

$$\mathcal{I}(E) = \mathcal{I}_1(E_1) \times \cdots \times \mathcal{I}_n(E_n)$$

où $\mathcal{I}_i(E_i)$ représente l'ensemble des états faiblement interdits de E_i dans G_i.

Démonstration :

$$\begin{aligned} \mathcal{I}(E) &= CoReach^G_{\Sigma_{uc}} = \left(Pre^G_{\Sigma_{uc}}\right)^*(E) \text{ d'après 1.27 et 1.17} \\ &= \left(Pre^G_{\Sigma_{uc}}\right)^*(E_1 \times \cdots \times E_n) \end{aligned}$$

Or $\Sigma_{uc} \subseteq \Sigma \setminus \Sigma_s$ et $\Sigma_{i,uc} = \Sigma_i \cap \Sigma_{uc}$, donc d'après la proposition 10 (formule (3.6))

$$\mathcal{I}(E) = {Pre^{G_1}_{\Sigma_{1,uc}}}^*(E_1) \times \cdots \times {Pre^{G_n}_{\Sigma_{n,uc}}}^*(E_n)$$

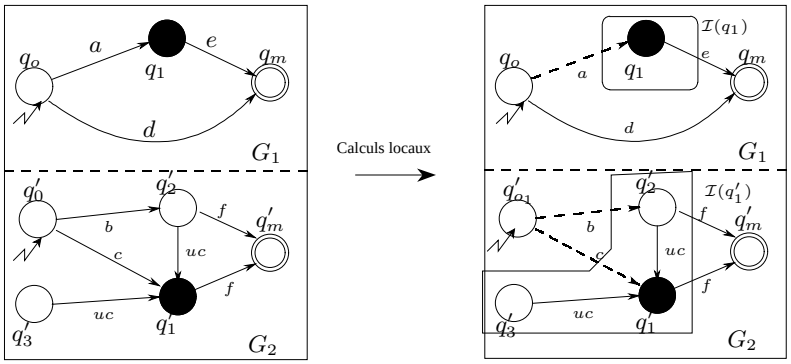
Finalement, par définition de $\mathcal{I}_i(\cdot)$,

$$\mathcal{I}(E) = \mathcal{I}_1(E_1) \times \cdots \times \mathcal{I}_n(E_n)$$

◇

La proposition précédente nous fournit un moyen efficace de calculer l'ensemble $\mathcal{I}(E)$, dans la mesure où ce calcul peut se réaliser localement sur chacun des sous systèmes (cf Section 3.2.1.2 pour l'étude de la complexité).

Exemple 13 *Considérons les automates G_1 et G_2 donnés par la figure 3.2. L'alphabet de G_1, noté Σ_1 vaut $\{a, e, d\}$ et l'alphabet de G_2 vaut $\{b, c, f, uc\}$. Tous les événements de l'alphabet de G_1 sont supposés contrôlables et uc_2 est le seul événement incontrôlable de G_2. L'objectif de contrôle consiste à interdire l'ensemble $E = \{(q_1, q'_1)\}$.*



(a) : $G = G_1 \parallel G_2$ (b) : Système contrôlé

FIG. 3.2: Un exemple simple

D'après la proposition 11, $\mathcal{I}(E) = \{q_1\} \times \{q'_3, q'_1, q'_2\}$. ◇

On va maintenant décrire l'ensemble $\mathcal{I}(E)$ pour le cas général où l'ensemble des configurations que l'on souhaite interdire est donné par une union de pavé.

Corollaire 2 *Soit $G = G_1 \parallel \ldots \parallel G_n$ avec $G_i = (\Sigma_i, Q_i, q_{0_i}, Q_{m_i}, \delta_i)$, tel que $\Sigma_s = \emptyset$. Soit $E = \bigcup_{1 \leq j \leq m} E_1^j \times \cdots \times E_n^j$, avec $E_i^j \subseteq Q_i$. Alors,*

$$\mathcal{I}(E) = \bigcup_{1 \leq j \leq m} \left(\prod_{1 \leq i \leq n} \mathcal{I}_i(E_i^j) \right) \tag{3.7}$$

La preuve de cette propriété est simplement basée sur le fait que $\mathcal{I}(E \cup E') = \mathcal{I}(E) \cup \mathcal{I}(E')$ (C.f. Section 1.2.2.4, Proposition 5). Pour déterminer l'ensemble $\mathcal{I}(E)$, il suffit donc de calculer sur chacun des sous systèmes les ensembles $(\mathcal{I}_i(E_i^j))_{j \leq m, i \leq n}$. Finalement, une expression pertinente d'un superviseur maximal assurant l'interdiction d'un ensemble d'états peut en être déduite.

Corollaire 3 *Soit $G = G_1 \parallel \ldots \parallel G_n$ avec $G_i = (\Sigma_i, Q_i, q_{0_i}, Q_{m_i}, \delta_i)$ et E un ensemble d'états de G. En reprenant les notations du corollaire 2, si $\Sigma_s \subseteq \Sigma_c$ alors le superviseur S défini pour tout $q \in Q$ par :*

$$S_E(q) = \{\sigma \in \Sigma_c |\ \delta(q,\sigma)! \wedge \delta(q,\sigma) \in \bigcup_{1 \leq j \leq m} (\mathcal{I}(E_1^j) \times \cdots \times \mathcal{I}(E_n^j))\} \tag{3.8}$$

assure l'interdiction de E et est maximal.

Comme évoqué précédemment, le superviseur S_E donné par (3.2) assure l'interdiction de E et est maximal. Le corollaire 3 provient donc directement de l'expression (3.2) et du corollaire 2.

Évaluation du superviseur : Voyons maintenant en quoi l'expression donnée par la formule (3.8) permet une évaluation en-ligne efficace de $S_E(q)$. Pour cela, il suffit tout d'abord pour chaque $\sigma \in \Sigma_c$ de déterminer $\delta(q,\sigma) = (q'_1, \ldots, q'_n)$ et il suffit alors de tester l'appartenance de $(q'_1, \ldots, q'_n)$ à $\mathcal{I}(E)$. Ceci revient d'après l'équation (3.7) à tester s'il existe $j \in \{1, \ldots, m\}$ tel que pour tout $i \in \{1, \ldots, n\}$, q'_i appartient à $\mathcal{I}_i(E_i^j)$.

La complexité précise de cette vérification est donnée dans le paragraphe suivant. Toutefois, on peut déjà remarquer que cette vérification ne nécessite que des calculs locaux, la rendant efficace.

Exemple 14 *De manière à illustrer note approche, considérons l'exemple du Chat/Souris introduit dans [59]. Le système représente un chat et une souris à l'intérieur d'un appartement comportant 5 pièces. Les mouvements du chat et de la souris peuvent être représentés par des automates décrits par la figure 14. Les états de ces automates représentent les pièces dans lesquelles l'animal est présent et sont respectivement notés C_i et M_i, pour $i = 1, \ldots, 5$ (les événements c_i et m_i modélisent les mouvements de l'animal d'une pièce à l'autre).*

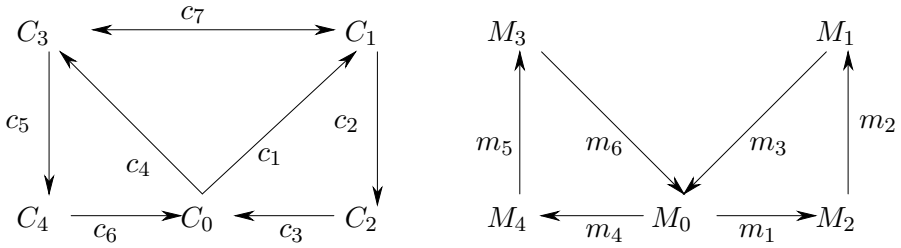


FIG. 3.3:

Le problème consiste alors à empêcher le chat et la souris d'être dans la même pièce. Ce problème se modélise naturellement comme un problème d'interdiction d'états où l'ensemble des états à interdire est donné par

$$E = \bigcup_{1 \leq j \leq 5} E^j \text{ avec } \forall 1 \leq j \leq 5,\ E^j = \{C_j\} \times \{M_j\}$$

Tous les mouvements du chat et de la souris peuvent être empêchés sauf les passages du chat entre les pièces 1 *et* 3 *(Ceci est possible par fermeture des portes entre les pièces). D'après la proposition 11,*

$$\begin{aligned}
\mathcal{I}(E^1) &= \{C_1, C_3\} \times \{M_1\} \\
\mathcal{I}(E^2) &= \{C_2\} \times \{M_2\} \\
\mathcal{I}(E^3) &= \{C_3, C_1\} \times \{M_3\} \\
\mathcal{I}(E^4) &= \{C_4\} \times \{M_4\} \\
\mathcal{I}(E^5) &= \{C_5\} \times \{M_5\}
\end{aligned}$$

et $\mathcal{I}(E) = \bigcup_{1 \leq j \leq 5} \mathcal{I}(E^j)$ *par définition. Or l'état initial du système est supposé être* (C_0, M_0) *et n'appartient pas à* $\mathcal{I}(E)$*. Par conséquent, le superviseur donné par (3.8) est une solution maximale au problème.* ◇

3.2.1.2 Étude de complexité.

Soit $G = G_1 \parallel \cdots \parallel G_n$ le système que l'on cherche à contrôler. On note N le nombre d'états d'un des sous systèmes G_i (on suppose que chaque G_i possède N états). Soit $E = \bigcup_{1 \leq j \leq m} E_1^j \times \cdots \times E_n^j$, avec $E_i^j \subseteq Q_i$, l'ensemble des états que l'on souhaite interdire par contrôle. D'après la corollaire 2,

$$\mathcal{I}(E) = \bigcup_{1 \leq j \leq m} \left(\prod_{1 \leq i \leq n} \mathcal{I}_i(E_i^j) \right)$$

- **Calculs effectués hors-ligne :** $\mathcal{I}(E)$ est décrit à partir d'une union de m pavés. Le calcul des composantes de ces pavés est effectué hors ligne et possède une complexité en $\mathcal{O}(m.n.N.|\Sigma|)$. On peut noter que la construction de l'automate G a pour complexité $\mathcal{O}(N^n.|\Sigma|)$ et que le calcul de $\mathcal{I}(E)$ effectué directement sur G a une complexité en $\mathcal{O}(N^n.|\Sigma|)$.
- **Calculs effectués en-ligne :** Toutefois, lors de l'exécution du système contrôlé, étant donné l'état courant q de ce système, la détermination de l'ensemble $S_E(q)$ décrit par 3.8 nécessite des calculs supplémentaires : pour chaque successeur de q, noté q', il faut vérifier l'appartenance de q' à $\mathcal{I}(E)$. Compte tenu de la forme particulière de $\mathcal{I}(E)$:

  $$\mathcal{I}(E) = \bigcup_j \mathcal{I}(E^j) \text{ et } \mathcal{I}(E^j) = \prod_i \mathcal{I}_i(E_i^j)$$

  on obtient que la complexité de l'évaluation en ligne de $S_E(q)$ vaut $\mathcal{O}(m.n.N.|\Sigma|)$.

Discussion. Plusieurs remarques peuvent être faites à ce niveau. Tout d'abord, si $m << N^n$, les calculs en ligne et hors ligne sont en général bien moins coûteux que ceux nécessaires à la construction de G. Ainsi, le superviseur le plus permissif assurant l'interdiction de E peut être explicité, alors même que le système G possède un nombre d'états trop important pour être représenté par un automate. Le nombre de machines impliquées dans le produit peut être raisonnablement considéré "petit". On en déduit que la phase de synthèse hors ligne $\mathcal{O}(m.n.N.|\Sigma|)$ n'est pas un obstacle à l'évaluation du superviseur, dès lors que le nombre de pavés m est aussi "petit".

De manière plus approfondie, on peut noter que le nombre d'états du système global n'influe pas de façon directe sur la complexité de l'approche adoptée. Le nombre de pavés représentant l'ensemble E des états à interdire, ainsi que le nombre de sous systèmes impliqués dans la composition sont des paramètres bien plus représentatifs. De plus, compte tenu des calculs effectués hors-ligne et en-ligne, seuls les ensembles $(\mathcal{I}(E_i^j))_{i\leq n, j\leq m}$ doivent être stockés. La capacité en mémoire nécessaire à un tel stockage vaut $n.m$ fois celle d'un ensemble à N éléments dans le pire cas, ce qui est inférieur à celle du stockage du système lui même qui possède N^n états dans le pire cas.

3.2.2 Cas général.

Dans la section précédente, nous avons donné une méthode efficace permettant le calcul d'un superviseur assurant l'interdiction d'un ensemble quelconque d'états pour des systèmes concurrents dont les événements incontrôlables ne sont pas partagés. Même s'il existe de nombreux systèmes répondant à ce critère, il est aussi intéressant dans la pratique de pouvoir traiter le cas où certains événements du systèmes permettent à des sous systèmes de se synchroniser de manière incontrôlable. Notons que dans le chapitre 2, les systèmes concurrents considérés ne possédaient que des événements partagés contrôlables. Dans cette section, bien que les objectifs de contrôle considérés soient moins généraux, les événements partagés seront supposés quelconques.

Nous considérons donc un système concurrent G, donné par composition parallèle de systèmes $(G_i)_{1\leq i\leq n}$, et un ensemble d'état E de G que l'on souhaite interdire par contrôle. Pour des raisons algorithmiques expliquées en section 3.1.2, on cherche à exprimer $\mathcal{I}(E)$ sous forme d'une union de pavés. Toutefois, puisque les événements partagés ne sont plus supposés contrôlables, le résultat fourni par la proposition 11 n'est plus valide. Cet aspect est illustré par l'exemple 15,

Exemple 15 *Considérons le système $G = G_1 \parallel G_2$, où G_1 et G_2 sont données en figure 15 et partagent les événements $\Sigma_s = \{\sigma_1, \sigma_2\}$. Les événements incontrôlables du système sont donnés par $\Sigma_{uc} = \{uc_1, uc_2, \sigma_2\}$. Le problème à résoudre est celui de l'interdiction de l'ensemble $E = (q_4, q_4')$. Or par définition de $\mathcal{I}(\cdot)$.*

$$\mathcal{I}(E) = \{(q_4, q_4'), (q_4, q_3'), (q_3, q_4'), (q_4, q_4'), (q_1, q_1')\}$$

Or

$$\mathcal{I}_1(\{q_4\}) \times \mathcal{I}_2(q_4') = \{q_4, q_3, q_1\} \times \{q_4', q_3', q_1'\}$$

Par conséquent, (q_1, q_3') appartient à $\mathcal{I}_1(\{q_4\}) \times \mathcal{I}_2(\{q_4'\})$ mais n'appartient pas à $\mathcal{I}(E)$. Ceci traduit que la proposition 11 n'est pas valide en général, lorsque les sous systèmes partagent des événements incontrôlables. $\diamond$

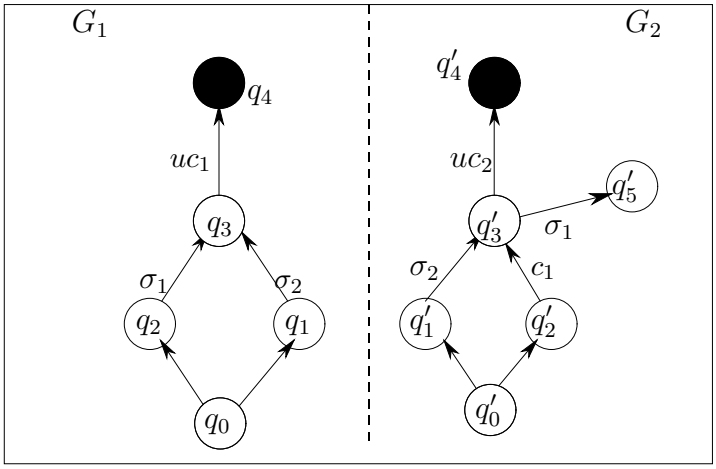


FIG. 3.4: Problème dû au partage d'événements

L'exemple précédent nous amène à raffiner le calcul des états faiblement interdits locaux $\mathcal{I}_i(\cdot)$ pour prendre en compte les événements partagés incontrôlables. C'est le but de la section suivante.

3.2.2.1 États interdits et événements locaux.

Dans cette section, on introduit un nouvel opérateur, noté $\mathcal{I}_{loc}(\cdot)$, qui "se distribue" par rapport aux composantes d'un pavé. Toutefois, si E représente un ensemble d'états à interdire, nous verrons que en général $\mathcal{I}_{loc}(E) \neq \mathcal{I}(E)$. L'importance du rôle joué par $\mathcal{I}(E)$ dans la résolution du problème de l'interdiction d'états nous amènera dans un premier temps à chercher des conditions suivant lesquelles $\mathcal{I}_{loc}(E) = \mathcal{I}(E)$.

Tout comme en Section 3.1.1, nous considérons un système concurrent modélisé par un automate $G = G_1 \parallel \cdots \parallel G_n$ avec $G_i = (\Sigma_i, Q_i, q_{0i}, Q_{mi}, \delta_i)$), et un ensemble d'états $E \subseteq Q$.

Comme le montre l'exemple 15, la proposition 11 n'est plus valide en général lorsque les sous systèmes du système à contrôler partagent des événements incontrôlables. Pour pallier à ce problème, nous introduisons l'opérateur $\mathcal{I}_{loc}(\cdot)$ défini sur les parties d'un ensemble d'états d'un automate.

Définition 17 *Soit $G = G_1 \parallel \ldots \parallel G_n$ et E un ensemble d'états de G, on note*

$$\mathcal{I}_{loc}(E) = CoReach^{G}_{\Sigma_{uc}\setminus\Sigma_s} = \left(Pre^{G}_{\Sigma_{uc}\setminus\Sigma_s}\right)^*(E) \tag{3.9}$$

et pour chaque ensemble d'états E_i de G_i, $\mathcal{I}_{i,loc}(E_i) = CoReach^{G_i}_{\Sigma_{i,uc}\setminus\Sigma_s} = {Pre^{G_i}_{\Sigma_{i,uc}\setminus\Sigma_s}}^*(E_i)$

$\mathcal{I}_{loc}(E)$ (resp. $\mathcal{I}_{i,loc}(E_i)$) représente donc l'ensemble des états qui peuvent mener à E (resp. E_i) dans G (resp. G_i) par tirage d'une séquence d'événements à la fois incontrôlables et locaux (i.e non partagés). Avec ces notations, on obtient le résultat suivant :

Proposition 12 *Soit $G = G_1 \parallel \cdots \parallel G_n$ avec $G_i = (\Sigma_i, Q_i, q_{0_i}, Q_{m_i}, \delta_i)$ et E un ensemble d'états de G tel que $E = E_1 \times \cdots \times E_n$, avec $E_i \subseteq Q_i$. Alors,*

$$\mathcal{I}_{loc}(E) = \mathcal{I}_{1,loc}(E_1) \times \cdots \times \mathcal{I}_{n,loc}(E_n)$$

Démonstration :

$$\begin{aligned} \mathcal{I}_{loc}(E) &= \left(Pre^G_{\Sigma_{uc}\setminus\Sigma_s}\right)^*(E) \\ &= \left(Pre^G_{\Sigma_{uc}\setminus\Sigma_s}\right)^*(E_1 \times \cdots \times E_n) \end{aligned}$$

Or $\Sigma_{uc} \subseteq \Sigma \setminus \Sigma_s$ et $\Sigma_{i,uc} = \Sigma_i \cap \Sigma_{uc}$, donc d'après la proposition 10 (formule (3.6)

$$\begin{aligned} \mathcal{I}_{loc}(E) &= \left(Pre^G_{\Sigma_{1,uc}\setminus\Sigma_s}\right)^*(E_1) \times \cdots \times \left(Pre^G_{\Sigma_{n,uc}\setminus\Sigma_s}\right)^*(E_n) \\ &= \mathcal{I}_{1,loc}(E_1) \times \cdots \times \mathcal{I}_{n,loc}(E_n) \end{aligned}$$

◇

La proposition 12 est donc l'analogue de la proposition 11. De plus compte tenu de la propriété 5, l'opérateur $\mathcal{I}_{loc}(\cdot)$ vérifie aussi

$$\forall E, E',\ \mathcal{I}_{loc}(E \cup E') = \mathcal{I}_{loc}(E) \cup \mathcal{I}_{loc}(E')$$

Par conséquent, le résultat de la proposition 12 peut s'étendre à une union quelconque de pavés de la façon suivante :

Corollaire 4 *En reprenant les notations de la proposition 12, si $E = \bigcup_j E^j$ avec $\forall j,\ E^j = E^j_1 \times \cdots \times E^j_n$ et $\forall i, j,\ E^j_i \subseteq Q_i$, alors*

$$\mathcal{I}_{loc}(E) = \bigcup_{j \le m} \left(\prod_{i \le n} \mathcal{I}_{loc}(E^j_i) \right)$$

$\mathcal{I}_{loc}(E)$ constitue en fait une sous approximation des états pouvant mener à E par tirage d'une séquence d'événements incontrôlables. On a en effet $\mathcal{I}_{loc}(E) \subseteq \mathcal{I}(E)$. L'égalité n'est en général pas valide. Il suffit en effet, qu'un état du système ne puisse atteindre l'ensemble E à interdire que par tirage d'une séquence d'événements incontrôlables contenant au moins un événement partagé. Par conséquent, on s'intéresse à présent aux conditions suivant lesquelles $\mathcal{I}_{loc}(E) = \mathcal{I}(E)$[2]. Ce qui nous amène à introduire l'opérateur $\mathcal{F}^\sigma_G(\cdot)$.

Définition 18 *Soit un système concurrent G sur un alphabet Σ, un événement $\sigma \in \Sigma$, et un ensemble d'états E de G,*

$$\mathcal{F}^\sigma_G(E) = Pre^G_{\{\sigma\}}(\mathcal{I}_{loc}(E)) \setminus \mathcal{I}_{loc}(E) \tag{3.10}$$

$\mathcal{F}^\sigma_G(E)$ représente l'ensemble des états de G qui peuvent mener à $\mathcal{I}_{loc}(E)$ par tirage de l'événement σ, mais qui n'appartiennent pas à $\mathcal{I}_{loc}(E)$.

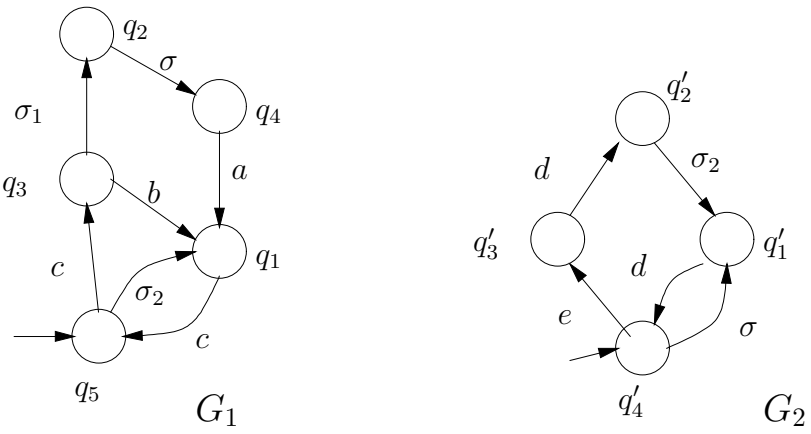


FIG. 3.5: Illustration d'ensembles locaux et globaux

Exemple 16 *Prenons l'exemple décrit par la Figure 3.5 pour illustrer ce que représente les ensembles introduits précédemment.*
Supposons que $E = \{(q_1, q'_1)\}$, $\Sigma_{1,uc} = \{a, b, \sigma_2\}$, $\Sigma_{2,uc} = \{\sigma_2\}$, *et* $\Sigma_s = \{\sigma, \sigma_1, \sigma_2\}$. *Avec ces hypothèses, on obtient les ensembles locaux*

$$\begin{cases} \mathcal{I}_{1,loc}(q_1) &= \{q_1, q_3, q_4\} \\ \mathcal{I}_{2,loc}(q'_1) &= \{q'_1\}. \end{cases}$$

En se basant sur la définition 18, on obtient

$$\begin{cases} \mathcal{F}_G^{\sigma}(\langle q_1, q'_1\rangle) &= \{(q_2, q'_4)\}, \\ \mathcal{F}_G^{\sigma_1}(\langle q_1, q'_1\rangle) &= \emptyset, \\ \mathcal{F}_G^{\sigma_2}(\langle q_1, q'_1\rangle) &= \{(q_5, q'_2)\}. \end{cases}$$

◇

En s'appuyant sur les ensembles $\mathcal{F}_G^{\sigma}(E)$, il est maintenant possible de définir une condition suivant laquelle $\mathcal{I}_{loc}(E) = \mathcal{I}(E)$.

Proposition 13 *Soit* $G = G_1 \parallel \cdots \parallel G_n$ *avec* $G_i = (\Sigma_i, Q_i, q_{0_i}, Q_{m_i}, \delta_i)$ *et* E *un ensemble d'états de* G. *Alors,*

$$\bigcup_{\sigma \in \Sigma_s \cap \Sigma_{uc}} \mathcal{F}_G^{\sigma}(E) = \emptyset \Longleftrightarrow \mathcal{I}(E) = \mathcal{I}_{loc}(E)$$

Démonstration : ($\Leftarrow$) par définition, $\bigcup_{\sigma \in \Sigma_s \cap \Sigma_{uc}} \mathcal{F}_G^{\sigma}(E) = Pre^G_{\Sigma_{uc} \cap \Sigma_s}(\mathcal{I}_{loc}(E)) \setminus \mathcal{I}_{loc}(E)$, or $\mathcal{I}_{loc}(E) = \mathcal{I}(E)$ (par hypothèse), d'où

$$\mathcal{F}_G^{\sigma}(E) = Pre^G_{\Sigma_{uc} \cap \Sigma_s}(\mathcal{I}(E)) \setminus \mathcal{I}(E)$$

[2]Compte tenu des définitions de $\mathcal{I}_{loc}(\cdot)$ et $\mathcal{I}(\cdot)$, on peut noter que $\Sigma_s \subseteq \Sigma_c$ est une condition suffisante pour obtenir ce résultat.

Or pour $\sigma \in \Sigma_{uc}$, on a $Pre_\sigma^G(\mathcal{I}(E)) = \mathcal{I}(E)$, d'où $\bigcup_{\sigma \in \Sigma_s \cap \Sigma_{uc}} \mathcal{F}_G^\sigma(E) = \emptyset$.
($\Rightarrow$) Par définition, on a

$$E \subseteq \mathcal{I}_{loc}(E) \subseteq \mathcal{I}(E) \qquad (\alpha)$$

Et par stabilité de l'opérateur $\mathcal{I}$, $\mathcal{I}(E) \subseteq \mathcal{I}(\mathcal{I}_{loc}(E)) \subseteq \mathcal{I}(\mathcal{I}(E))$. Or $\mathcal{I}(\mathcal{I}(E)) = \mathcal{I}(E)$; ce qui induit :

$$\mathcal{I}(\mathcal{I}_{loc}(E)) = \mathcal{I}(E) \qquad (\gamma)$$

De plus, $\bigcup_{\sigma \in \Sigma_s \cap \Sigma_{uc}} \mathcal{F}_G^\sigma(E) = \emptyset$ par hypothèse, ce qui signifie que

$$Pre^G_{\Sigma_{uc} \cap \Sigma_s}(\mathcal{I}_{loc}(E)) \setminus \mathcal{I}_{loc}(E) = \emptyset$$

c'est à dire

$$Pre^G_{\Sigma_{uc} \cap \Sigma_s}(\mathcal{I}_{loc}(E)) \subseteq \mathcal{I}_{loc}(E)$$

Or

$$Pre^G_{\Sigma_{uc} \setminus \Sigma_s}(\mathcal{I}_{loc}(E)) = \mathcal{I}_{loc}(E)$$

donc

$$Pre^G_{\Sigma_{uc} \cap \Sigma_s}(\mathcal{I}_{loc}(E)) \cup Pre^G_{\Sigma_{uc} \setminus \Sigma_s}(\mathcal{I}_{loc}(E)) \subseteq \mathcal{I}_{loc}(E)$$

donc d'après la propriété 1.14, on a

$$Pre^G_{\Sigma_{uc}}(\mathcal{I}_{loc}(E)) \subseteq \mathcal{I}_{loc}(E)$$

Par croissance de l'opérateur $Pre^G_{\Sigma_{uc}}$, on a

$$\forall n \geq 1,\ (Pre^G_{\Sigma_{uc}})^{(n)}(\mathcal{I}_{loc}(E)) \subseteq \mathcal{I}_{loc}(E)$$

Par passage à la limite, on obtient finalement que $\mathcal{I}(\mathcal{I}_{loc}(E)) \subseteq \mathcal{I}_{loc}(E)$ et on déduit alors de (γ) que

$$\mathcal{I}(E) \subseteq \mathcal{I}_{loc}(E)$$

ce qui, combiné avec (α), fournit le résultat. ◇

Par conséquent, la proposition 13 fournit une condition nécessaire et suffisante pour que $\mathcal{I}_{loc}(E) = \mathcal{I}(E)$. Lorsque cette condition est vérifiée, $\mathcal{I}(E)$ peut donc s'exprimer comme une union de pavés.

Corollaire 5 *Soit $G = (\Sigma, Q, q_0, Q_m, \delta)$ un système concurrent et E un ensemble d'états de G. En reprenant les notations de la proposition 13, si $\bigcup_{\sigma \in \Sigma_s \cap \Sigma_{uc}} \mathcal{F}_G^\sigma(E) = \emptyset$ alors le superviseur S défini pour tout $q \in Q$ par :*

$$S_E(q) = \{\sigma \in \Sigma_c |\ \delta(q,\sigma)! \wedge \delta(q,\sigma) \in \mathcal{I}_{loc}(E)\} \qquad (3.11)$$

assure l'interdiction de E et est maximal.

Le superviseur S_E donné par (3.11) assure l'interdiction de E et est maximal. Le corollaire 5 provient donc directement de l'expression (3.2) et de la proposition 13. Le résultat du corollaire 5 permet un affaiblissement des hypothèses du corollaire 3. Rappelons enfin qu'une telle expression du superviseur S_E possède les avantages énoncés en section 3.2 : efficacité algorithmique hors-ligne et en-ligne.

Discussion sur la complexité. La pertinence de la proposition 13 provient de l'intérêt algorithmique d'une expression de $\mathcal{I}(E)$ sous forme d'une union de pavés. En reprenant les notations de la proposition 13, l'ensemble $\mathcal{I}(E)$ peut donc ainsi être obtenu à partir des ensembles locaux $(\mathcal{I}_i(E_i^j))_{i,j}$, assurant une faible complexité des calculs effectués hors-ligne. De plus, comme expliqué dans la section 3.2.1.2, la complexité du test d'appartenance d'un état à $\mathcal{I}(E)$ est limitée lorsque $\mathcal{I}(E)$ s'exprime comme une union de pavés. Par conséquent, le résultat de la proposition 13 fournit un moyen efficace de calculer une expression de $\mathcal{I}(E)$, qui soit pertinente pour limiter la complexité d'une évaluation en-ligne du superviseur. Toutefois, la validité de la condition $\bigcup_{\Sigma_s \cap \Sigma_{uc}} \mathcal{F}_G^\sigma(E) = \emptyset$ doit être vérifiée pour appliquer la proposition 13.

Par définition de $\mathcal{F}_G^\sigma(E)$, vérifier la condition donnée par la proposition 13 consiste à vérifier que pour tout $\sigma \in \Sigma_{uc} \cap \Sigma_s$,

$$Pre^G_{\Sigma_{uc}\cap\Sigma_s}(\mathcal{I}_{loc}(E)) \subseteq \mathcal{I}_{loc}(E) \tag{3.12}$$

Or il se peut que $Pre^G_{\Sigma_{uc}\cap\Sigma_s}(\mathcal{I}_{loc}(E))$ possède un nombre d'états de l'ordre de grandeur de celui du système global. Dans ce cas, vérifier que l'inclusion 3.12 est valide peut s'avérer coûteux, voir irréalisable en pratique.

Toutefois, le pire cas n'est probablement pas représentatif. Des systèmes intéressants peuvent notamment être traités et n'entrent pas dans ce cadre (c.f l'exemple des chariots filoguidés). Ainsi, les systèmes dits *faiblement synchronisés* peuvent posséder un nombre important d'états et une occurrence très faible d'événements partagés. L'exemple des chariots filoguidés représente un tel système : le nombre d'événements partagés est faible comparativement au nombre d'états du système.

3.2.2.2 Gestion des événements partagés et incontrôlables.

Si $\bigcup_{\Sigma_{uc}\cap\Sigma_s} \mathcal{F}_G^\sigma(E) \neq \emptyset$, alors d'après la proposition 13, $\mathcal{I}(E) \neq \mathcal{I}_{loc}(E)$. Toutefois, il est possible de remarquer que les états faiblement interdits qui ne sont pas pris en compte sont ceux à partir desquels il existe une trajectoire menant à E qui contient au moins un événement partagé et incontrôlable. De manière à capturer ces états, nous introduisons maintenant la fonction Φ suivante :

Définition 19 *Soit $G = G_1 \parallel \cdots \parallel G_n$. On note $\Phi : 2^Q \to 2^Q$, la fonction définie pour tout ensemble d'états E de G par*

$$\Phi(E) = E \bigcup_{\sigma\in\Sigma_s\cap\Sigma_{uc}} \mathcal{F}_G^\sigma(E)$$

Les éléments de $\Phi(E)$ représentent soit des états de E, soit des états pouvant mener à $\mathcal{I}_{loc}(E)$ par tirage d'un événement incontrôlable et partagé. On considère maintenant la séquence $(\Phi^n(E))_{n\geq 1}$. Pour tout $n \geq 1$, les éléments de $\Phi^n(E)$ sont des états pouvant mener à E en tirant une séquence d'événements incontrôlables qui contiennent moins de n éléments de Σ_s. On note $\Phi^*(E)$ la limite de la séquence $(\Phi^n(E))_{n\geq 1}$. Cette limite existe toujours et est atteinte en un nombre fini d'itérations, puisque le nombre d'états du système est fini. Les deux lemmes suivants présentent des propriétés des opérateurs $\Phi(\cdot)$ et $\Phi^*(\cdot)$.

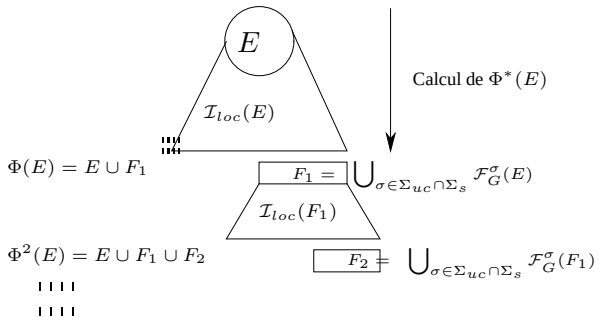


FIG. 3.6: Intuition du calcul de $\Phi^*(E)$

Lemme 11 *Soit $G = G_1 \parallel \cdots \parallel G_n$ et E un ensemble d'états de G. Alors,*

$$\bigcup_{\sigma \in \Sigma_s \cap \Sigma_{uc}} \mathcal{F}^\sigma(\Phi^*(E)) = \emptyset$$

Démonstration : Soit $E \subseteq Q$. Par définition de $\Phi^*(\cdot)$, on a $\Phi^*(E) = \Phi(\Phi^*(E))$. De plus, par définition de $\Phi(\cdot)$, on a

$$\Phi(\Phi^*(E)) = \Phi^*(E) \bigcup \left(\bigcup_{\sigma \in \Sigma_{uc} \cap \Sigma_s} \mathcal{F}_G^\sigma(\Phi^*(E)) \right)$$

Par conséquent, on en déduit que

$$\Phi^*(E) = \Phi^*(E) \bigcup \left(\bigcup_{\sigma \in \Sigma_{uc} \cap \Sigma_s} \mathcal{F}_G^\sigma(\Phi^*(E)) \right)$$

et donc que $\left(\bigcup_{\sigma \in \Sigma_{uc} \cap \Sigma_s} \mathcal{F}_G^\sigma(\Phi^*(E))\right) \subseteq \Phi^*(E)$. Or par définition de $\mathcal{F}_G^\sigma(E)$[3],

$$\left(\bigcup_{\sigma \in \Sigma_{uc} \cap \Sigma_s} \mathcal{F}_G^\sigma(\Phi^*(E)) \right) \cap \mathcal{I}_{loc}(\Phi^*(E)) = \emptyset$$

et puisque $\Phi^*(E) \subseteq \mathcal{I}_{loc}(\Phi^*(E))$, on en déduit que

$$\left(\bigcup_{\sigma \in \Sigma_{uc} \cap \Sigma_s} \mathcal{F}_G^\sigma(\Phi^*(E)) \right) \cap \Phi^*(E) = \emptyset$$

Et puisque $\left(\bigcup_{\sigma \in \Sigma_{uc} \cap \Sigma_s} \mathcal{F}_G^\sigma(\Phi^*(E))\right) \subseteq \Phi^*(E)$, on en déduit que

$$\left(\bigcup_{\sigma \in \Sigma_{uc} \cap \Sigma_s} \mathcal{F}_G^\sigma(\Phi^*(E)) \right) = \emptyset$$

◇

[3] Voir formule (3.10)

On cherche maintenant à comparer l'ensemble $\Phi^*(E)$ à l'ensemble $\mathcal{I}(E)$. Pour cela, on introduit d'abord le lemme 12 .

Lemme 12 *Soit $G = G_1 \parallel \cdots \parallel G_n$ et E un ensemble d'états de G. $\mathcal{I}(\Phi^*(E)) = \mathcal{I}(E)$*

Démonstration : $(\subseteq)$ Nous allons montrer par récurrence que $\forall n \geq 0$, $\Phi^n(E) \subseteq \mathcal{I}(E)$. D'abord, ce résultat est clair pour $n = 0$ (en effet, dans ce cas $\Phi^n(E) = E$). Considérons maintenant $n \geq 0$ et supposons que $\Phi^n(E) \subseteq \mathcal{I}(E)$. L'opérateur Φ étant stable pour l'inclusion,

$$\Phi^{n+1}(E) \subseteq \Phi(\mathcal{I}(E)) \qquad (\alpha)$$

par définition de $\Phi(\cdot)$,

$$\Phi(\mathcal{I}(E)) = \mathcal{I}(E) \bigcup \left(\bigcup_{\sigma \in \Sigma_{uc} \cap \Sigma_s} \mathcal{F}_G^{\sigma}(\mathcal{I}(E)) \right)$$

Or pour $\sigma \in \Sigma_{uc} \cap \Sigma_s$, $\mathcal{F}_G^{\sigma}(\mathcal{I}(E)) = Pre_{\{\sigma\}}^G(\mathcal{I}_{loc}(\mathcal{I}(E))) \setminus \mathcal{I}_{loc}(\mathcal{I}(E))$. Or par définition de $\mathcal{I}(\cdot)$ et $\mathcal{I}_{loc}(\cdot)$, on a $\mathcal{I}_{loc}(\mathcal{I}(E)) = \mathcal{I}(E)$. On en déduit alors que pour tout $\sigma \in \Sigma_{uc}$, $Pre_{\{\sigma\}}^G(\mathcal{I}(E)) \subseteq \mathcal{I}(E)$. Par conséquent, $\forall \sigma \in \Sigma_{uc} \cap \Sigma_s$, on a $\mathcal{F}_G^{\sigma}(\mathcal{I}(E)) = \emptyset$. On en déduit que $\Phi(\mathcal{I}(E)) = \mathcal{I}(E)$. On déduit alors de (α) que $\Phi^{n+1}(E) \subseteq \mathcal{I}(E)$. Par conséquent, $\forall n \geq 0$, $\Phi^n(E) \subseteq \mathcal{I}(E)$. On obtient donc que $\Phi^*(E) \subseteq \mathcal{I}(E)$. Finalement, par stabilité de l'opérateur $\mathcal{I}$, $\mathcal{I}(\Phi^*(E)) \subseteq \mathcal{I}(\mathcal{I}(E)) = \mathcal{I}(E)$
$(\supseteq)$ Puisque $E \subseteq \Phi^*(E)$, on obtient immédiatement $\mathcal{I}(E) \subseteq \mathcal{I}(\Phi^*(E))$. D'où le résultat. ◇

Finalement, on obtient que

Proposition 14 *Soit $G = G_1 \parallel \cdots \parallel G_n$ et E un ensemble d'états de G. $\mathcal{I}(E) = \mathcal{I}_{loc}(\Phi^*(E))$*

Démonstration : La preuve provient des lemmes 12 et 11 et la proposition 13. ◇

La proposition précédente nous donne un moyen de calculer $\mathcal{I}(E)$ sans avoir à expliciter le système global G. Ce calcul réside dans la détermination de l'ensemble $\Phi^*(E)$. Cet ensemble est obtenu par itération de l'opérateur $\Phi(\cdot)$ jusqu'à ce que la condition d'arrêt, fournie par la proposition 13 soit valide. Bien que le cardinal de $\Phi^*(E)$ puisse être de l'ordre de grandeur du nombre d'états du système dans le pire cas, ce résultat semble pertinent en général.

Corollaire 6 *Soit $G = (\Sigma, Q, q_0, Q_m, \delta)$ un système concurrent et E un ensemble d'états de G. En reprenant les notations de la proposition 14, le superviseur S défini pour tout $q \in Q$ par :*

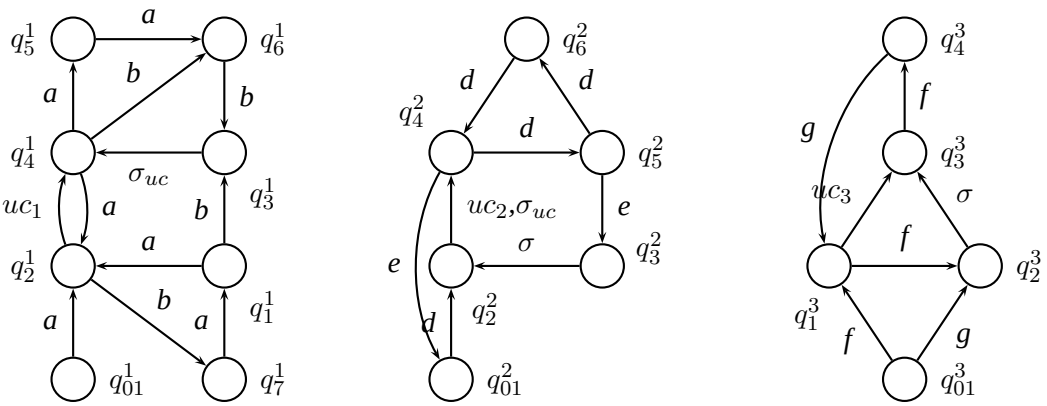
$$S_E(q) = \{\sigma \in \Sigma_c |\ \delta(q,\sigma)! \wedge \delta(q,\sigma) \in \mathcal{I}_{loc}(\Phi^*(E))\}$$

assure l'interdiction de E et est maximal. ◇

Le superviseur S_E donné par (3.2) assure l'interdiction de E et est maximal. Le corollaire 6 provient donc directement de l'expression (3.2) et de la proposition 14.

Enfin, comme évoqué dans les sections précédentes, l'intérêt algorithmique de la proposition 14 réside d'une part dans le gain en complexité des calculs effectués hors-ligne. D'autre part, l'expression de $\mathcal{I}(E)$ sous forme d'une union de pavés permet de limiter l'évaluation en-ligne du superviseur standard assurant l'interdiction de E.

Exemple 17 *Le système considéré G est donné par la composition parallèle des automates G_1, G_2, G_3 suivants. Les ensembles d'événements partagés et incontrôlables sont donnés par $\Sigma_s = \{\sigma, \sigma_{uc}\}$ et $\Sigma_{uc} = \{uc_1, uc_2, uc_3, \sigma_{uc}\}$.*



On souhaite ici assurer l'interdiction de l'ensemble $E = E^1 \cup E^2$ avec $E^1 = \{q_1^1, q_4^1\} \times \{q_4^2\} \times \{q_3^3\}$ et $E^2 = \{q_3^1, q_2^1\} \times \{q_2^2, q_3^2\} \times \{q_2^3, q_3^3\}$. Les calculs induits par l'approche décrite dans ce chapitre mènent aux résultats suivants :

	E_1^1	E_2^1	E_3^1	E_1^2	E_2^2	E_3^2
$\mathcal{I}_{loc}$	$\{q_1^1, q_2^1, q_4^1\}$	$\{q_2^2, q_4^2\}$	$\{q_1^3, q_3^3\}$	$\{q_2^1, q_3^1\}$	$\{q_2^2, q_3^2\}$	$\{q_1^3, q_2^3, q_3^3\}$
$\mathcal{F}_G^{\sigma_{uc}}$	$\{q_3^1\}$	$\{q_2^2\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$\mathcal{F}_G^{\sigma}$	$\emptyset$	$\{q_3^2\}$	$\{q_2^3\}$	$\emptyset$	$\{q_3^2\}$	$\{q_2^3\}$

Le calcul de $\Phi(E)$ donne alors $\Phi(E) = E^1 \cup E^2 \cup E^3$ avec $E^3 = \{q_3^1\} \times \{q_2^2\} \times \{q_3^3, q_1^3\}$. On obtient alors le tableau suivant :

	E_1^3	E_2^3	E_3^3
$\mathcal{I}_{loc}$	$\{q_3^1\}$	$\{q_2^2\}$	$\{q_1^3, q_3^3\}$
$\mathcal{F}_G^{\sigma_{uc}}$	$\emptyset$	$\emptyset$	$\emptyset$
$\mathcal{F}_G^{\sigma}$	$\emptyset$	$\{q_3^2\}$	$\{q_2^3\}$

Puisque $\Sigma_s \cap \Sigma_{uc} = \{\sigma_{uc}\}$, on en déduit que $\bigcup_{\Sigma_s \cap \Sigma_{uc}} \mathcal{F}_G^{\sigma}(E) = \emptyset$ et donc que $\Phi(E) = \Phi^*(E)$. Finalement, on déduit de la proposition 14 que

$$\mathcal{I}(E) = \mathcal{I}_{loc}(E^1 \cup E^2 \cup E^3) = \mathcal{I}_{loc}(E^1) \cup \mathcal{I}_{loc}(E^2) \cup \mathcal{I}_{loc}(E^3)$$

Donc

$$\begin{array}{rcl} \mathcal{I}(E) & = & \{\{q_1^1, q_2^1, q_4^1\} \times \{q_2^2, q_4^2\} \times \{q_1^3, q_3^3\} \\ & & \{q_2^1, q_3^1\} \times \{q_2^2, q_3^2\} \times \{q_1^3, q_2^3, q_3^3\} \\ & & \{q_3^1\} \times \{q_2^2\} \times \{q_1^3, q_3^3\}\} \end{array}$$

Étant donné un état q de G, l'évaluation de la solution S au problème de synthèse pour un objectif d'interdiction d'états, donnée par le corollaire 6, consiste pour chaque successeur q' de q dans G, à déterminer si $q' \in \mathcal{I}(E)$. Dans cet exemple, si $q = (q_7^1, q_2^2, q_3^3)$, alors les événements tirables

depuis q sont a, σ_{uc} et f, menant respectivement aux états $s_1 = (q_2^1, q_2^2, q_3^3)$, $s_2 = (q_7^1, q_4^2, q_3^3)$ et $s_3 = (q_7^1, q_2^2, q_4^3)$. On peut alors vérifier que l'événement a doit être empêché car $s_1 \in \mathcal{I}(E)$. En revanche, on peut vérifier que ni s_2, ni s_3 n'appartiennent à $\mathcal{I}(E)$. Par conséquent, $S(q) = \{a\}$.
$\diamond$

3.2.2.3 Complexité

On considère donc n automates $(G_i)_{1\leq i\leq n}$ (avec $G_i = (\Sigma_i, Q_i, q_{0_i}, \delta_i)$) dont la composition parallèle modélise le système à contrôler $G = (\Sigma, Q, q_0, \delta)$. Pour $i \in \{1, \ldots, n\}$, on note N_i le nombre d'états de G_i. De plus, on note $N = \max_i N_i$, et $s = \max_i |\Sigma_i|$. On considère un ensemble $E = \bigcup_{1\leq j\leq m}(E^j)$ avec

$$\forall j \in \{1, \ldots, m\},\ E^j = E_1^j \times \cdots \times E_n^j$$

et $E_i^j \subseteq Q_i$. Par conséquent, l'ensemble E est constitué de m pavés. On note alors $M = |\Phi^*(E)|$. Rappelons que la complexité du calcul explicite du système global vaut $\mathcal{O}(N^n.|\Sigma|)$. Les calculs induits par l'approche proposée ici se décomposent en deux phases : des calculs hors-ligne, et une évaluation "non triviale" en-ligne du superviseur.

- **Hors-ligne :**
  (a) Pour i et j fixés, le calcul de $\mathcal{I}_{loc}(E_i^j)$ et $(\mathcal{F}_G^\sigma(E_i^j))_{\{\sigma|\ i\in \mathrm{IN}(\sigma)\}}$ vaut $\mathcal{O}(N_i.|\Sigma_i|)$.
  (b) Le calcul effectué en (a) pour chaque $i \in \{1, \ldots, n\}$ et chaque $j \in \{1, \ldots, m\}$ vaut donc $\mathcal{O}(n.m.N.s)$.
  (c) La complexité de la vérification de la condition $\bigcup\limits_{1\leq i\leq n} \mathcal{F}_G^\sigma(E) = \emptyset$ et du calcul de $\Phi^*(E)$ peut être élevée si le nombre d'états de $\Phi^*(E)$ est très important. Toutefois, cette complexité est espérée raisonnable dans de nombreux cas.[4]
- **En-ligne :** Étant donné l'état courant q, on évalue pour chaque $\sigma \in \Sigma_c$ tel que $\delta(q, \sigma)!$, si $\delta(q, \sigma) \in \mathcal{I}_{loc}(\Phi^*(E))$. Or $\mathcal{I}_{loc}(\Phi^*(E))$ est constitué de M pavés. De plus, la complexité du test d'appartenance d'un élément à un ensemble à N éléments vaut $\mathcal{O}(\log(N))$ dans le pire cas. Enfin, ce test doit être effectué pour chaque composante q_i de l'état q. Par conséquent, la complexité du test d'appartenance de q à $\mathcal{I}_{loc}(\Phi^*(E))$ vaut

  $$\mathcal{O}(|\Sigma_c|.n.M.\log(N))$$

  dans le pire cas.

3.3 Évitement d'états en deadlock

Dans la section 3.2, nous avons étudié le problème de l'interdiction d'états pour un système concurrent. La résolution de ce problème permet d'assurer que certains états du système contrôlé ne soient pas atteignables sous contrôle. Toutefois, cette solution ne prend pas en compte les blocages du système. En effet, certains comportements du système peuvent modéliser une tâche à

[4] Voir notamment les exemples traités au chapitre 4.

accomplir. Cet ensemble de comportements est modélisé par le langage marqué du système (ou par les états marqués de l'automate qui modélise le système).

La résolution du *Problème de Synthèse de Contrôleurs Non Bloquants* (PSCNB) décrit en section 1.2.2, permet d'obtenir un système contrôlé dont les comportements peuvent toujours être complétés pour accomplir une tâche. Ce problème s'avère délicat à résoudre efficacement dans le cadre des systèmes concurrents dans la mesure où il nécessite le calcul explicite des comportements du système (ce que nous cherchons à éviter). En effet, l'ensemble des séquences du langage marqué du système[5] ne peut être en général déterminé à partir des séquences de chacun des sous systèmes. Puisque les systèmes considérés ici sont supposés être concurrents et de grande taille. Une modélisation explicite d'un tel système (sous forme d'un automate par exemple) n'est donc pas envisageable en pratique, de même que la détermination des comportements qui ne permettent pas d'accomplir une tâche.

Cette section tente d'apporter un début de réponse à ce problème en considérant un problème moins contraignant que celui du non-blocage : le problème de *l'évitement de deadlock*. En d'autres termes, le problème est de calculer/synthétiser un superviseur qui fasse en sorte que le système puisse toujours évoluer. Ceci équivaut à assurer que pour tout état du système sous contrôle, il existe au moins un événement qui soit admissible.

3.3.1 Problématique

Lorsque l'on souhaite contrôler les comportements d'un système, afin d'interdire un sous ensemble de ses états, et que ce système est concurrent et de grande taille, alors les résultats des sections précédentes du chapitre 3 peuvent s'appliquer. Toutefois, le résultat obtenu n'est pas toujours satisfaisant, notamment lorsque l'on souhaite qu'aucun état du système contrôlé ne soit bloquant. Le but de cette section consiste alors à apporter une solution à ce problème.

On reprend les notations introduites en section 3.1.1 et on considère ainsi un système concurrent modélisé par une composition parallèle d'automates, dont on supposera que tous les états sont accessibles depuis l'état intial. On note alors $G = (\Sigma, Q, q_0, \delta)$ le système à contrôler, modélisé par la composition parallèle d'automates trim $(G_i)_{1\leq i\leq n}$, où pour $i \in \{1, \ldots, n\}$, $G_i = (\Sigma_i, Q_i, q_{0i}, \delta_i)$. Les événements incontrôlables (resp. contrôlables) de G_i sont notés $\Sigma_{i,uc}$ (resp. $\Sigma_{i,c}$). Les conditions énoncées en section 3.1.1 sont aussi supposées valides dans cette section. En particulier, le caractère contrôlable d'un événement partagé est supposé être le même pour chacun des systèmes qui le partage. On note respectivement $\Sigma_{uc} = \bigcup_i \Sigma_{i,uc}$ et $\Sigma_c = \bigcup_i \Sigma_{i,c}$ l'ensemble des événements incontrôlables et contrôlables de G.

On considère aussi un ensemble d'états $E \subseteq Q$ à interdire. On suppose que E est représenté par une union de pavés :

$$E = \bigcup_{1\leq j\leq m} E^j \text{ avec } E^j = E_1^j \times \cdots \times E_n^j \text{ où } \forall i, j,\ E_i^j \subseteq Q_i \tag{3.13}$$

On rappelle que $\mathcal{I}(E)$ représente l'ensemble des états de G qui peuvent mener à E dans G par tirage d'une séquence d'événements incontrôlables. On suppose que $q_0 \notin \mathcal{I}(E)$ (ce qui assure

[5] Langage représentant les séquences menant à au moins un état marqué.

l'existence d'un superviseur interdisant l'ensemble E d'après la section 1.2.2.4) et on considère le superviseur S_E, assurant l'interdiction de E dans G, décrit en (3.2). La section 3.1.2 fournie une expression de $\mathcal{I}(E)$ permettant une évaluation efficace de S_E. Cette expression fait intervenir l'ensemble $\Phi^*(E)$ introduit en section 3.2.2.2. Pour simplifier, on suppose que $\Phi^*(E) = E$. On peut en fait toujours se ramener à ce cas, en calculant $\Phi^*(E)$ et en considérant $S_{\Phi^*(E)}$ (qui vaut en fait S_E d'après la section 3.2.2.2). Le système contrôlé, noté S_E/G peut être identifié à un automate

$$(\Sigma, Q, q_0, \delta_{S_E/G})$$

où $\delta_{S_E/G}$ est la fonction partielle de transition définie par

$$\delta_{S_E/G}(q,\sigma) = \begin{cases} \delta(q,\sigma) \text{ si } \delta(q,\sigma)! \wedge \sigma \notin S_E(q) \\ \text{Indéfini Sinon} \end{cases} \tag{3.14}$$

Cet automate représente en fait une réalisation du superviseur S_E. Toutefois, S_E est supposé ici être obtenu à partir des méthodes introduites dans les sections précédentes, et une telle réalisation ne sera par conséquent jamais construite en pratique. Afin de poser le problème qui nous intéresse de manière formelle, on introduit la notion *d'états en deadlock*, qui permet de définir les états d'un système desquels aucun événement ne peut être tiré.

Définition 20 *Soit $G = (\Sigma, Q, q_0, \delta)$ un automate et un état $q \in Q$ de G. q est dit en deadlock dans G si $\delta(q) = \emptyset$.*

De plus, il sera intéressant par la suite de distinguer les états qui sont en deadlock dans un système contrôlé, alors qu'ils ne l'étaient pas dans le système à contrôler. Ces états sont donc en deadlock car ils subissent l'action du superviseur, et sont appelés *états en deadlock par contrôle*.

Définition 21 *Soient $G = (\Sigma, Q, q_0, \delta)$ un automate, S un superviseur sur G, et $q \in Q$. q est en deadlock par contrôle dans S/G si $\delta_{S/G}(q) = \emptyset \wedge \delta(q) \neq \emptyset$.*

Les états en deadlock par contrôle dans un système contrôlé S/G représentent donc les états en deadlock dans S/G qui subissent l'action du superviseur S. En d'autres termes, il s'agit des états qui sont en deadlock dans S/G et qui ne l'étaient pas dans G.

Enfin, le problème de l'*évitement de deadlock* qui nous intéresse consiste à assurer qu'aucun état atteignable du système contrôlé ne soit en deadlock.

Problème : Problème de l'évitement de deadlock dans un système contrôlé : il s'agit de déterminer un superviseur maximal S

- interdisant d'une part un ensemble d'états E d'un système G,
- et assurant d'autre part qu'aucun état atteignable de S/G n'est en deadlock.

Ce problème peut être résolu classiquement de la façon suivante :

(1) Calcul de S interdisant l'ensemble E dans G.
(2) Détermination de l'ensemble D des états en deadlock dans S/G.
(3) Tant que $D \neq \emptyset$, retour au point (1) avec $E := E \cup D$ et $G := S/G$.

Par conséquent, la méthode de résolution consiste initialement à interdire E dans G à partir d'un superviseur S. Puis la méthode est alors itérative. Elle consiste à chaque étape à déterminer l'ensemble des états en deadlock dans le système contrôlé précédemment obtenu, puis à interdire cet ensemble d'états.

Le but de cette section est d'offrir une solution au problème de l'évitement de deadlock dans un système contrôlé, lorsque le système à contrôler est concurrent. Pour cela, la méthode présentée ci-dessus sera appliquée. Le point (1) de cette méthode peut être résolu à partir des résultats fournis dans les sections précédentes de ce chapitre. Un superviseur tel que décrit en (3.2) peut notamment être déterminé. Le problème consiste donc à donner une méthode permettant de résoudre le point (2), lorsque le système initialement considéré est concurrent. Ce point peut être aisément traité lorsque le système contrôlé, duquel on souhaite déterminer les deadlock, est donné de manière explicite. En revanche, la méthode de synthèse de contrôleurs utilisée dans les sections précédentes permet d'obtenir une expression d'un superviseur résolvant le problème, mais pas un modèle explicite (sous forme d'un automate par exemple) du système contrôlé. De plus, une telle expression de ce système n'est pas souhaitée ici car elle peut s'avérer irréalisable en pratique, lorsque le système concurrent initial est de grande taille.

Dans la suite de cette section, on s'intéresse donc à la détection des états en deadlock dans un système concurrent G qui subit le contrôle d'un superviseur S tel que décrit en (3.2). Pour cela, on cherche à caractériser les états en deadlock dans S/G, sans le calculer explicitement.

Compte tenu des différentes structures qui seront manipulées, la détermination des états en deadlock du système contrôlé s'effectuera ici suivant deux critères : soit ces états sont déjà en deadlock dans le système à contrôler, soit ils ne sont en deadlock qu'en subissant l'action du superviseur. Ces critères sont schématisés par la figure 3.7, et la définition 21 offre une formalisation des états en deadlock subissant l'action d'un superviseur.

Le système considéré pouvant être de grande taille, une solution utilisant la structure concurrente du système, et la forme particulière du superviseur est envisagée. Tout d'abord, des critères permettant de déterminer efficacement les états en deadlock d'un système concurrent ont été exhibés dans des travaux antérieurs ([1], [53]). Bien que certains de ces résultats soient utiles pour notre étude, le problème auquel on s'intéresse ici est différent. En effet, les systèmes à contrôler considérés sont concurrents, mais ce n'est en général pas le cas des systèmes concurrents subissant l'action d'un superviseur.

En se basant sur des propriétés données dans [1], la section 3.3.2 rappelle une méthode permettant de déterminer les états en deadlock dans un système concurrent, en utilisant sa structure. Ces résultats seront étendus par la suite (section 3.3.3) afin de prendre en compte l'action du superviseur sur un système concurrent.

3.3.2 Détection de deadlock dans un système concurrent

La structure concurrente d'un système peut être efficacement utilisée pour y détecter les états en deadlock. Dans [41], les auteurs introduisent une solution basée d'une part sur la décomposition des ensembles d'états des sous systèmes (dont on ne considère plus les événements partagés) en composantes fortement connexes. D'autre part, les auteurs considèrent un *graphe de synchronisations* qui modélise les séquences d'événements partagés du système. À partir de ces informations, les questions d'atteignabilité, de blocage, et de vivacité peuvent notamment être traitées efficace-

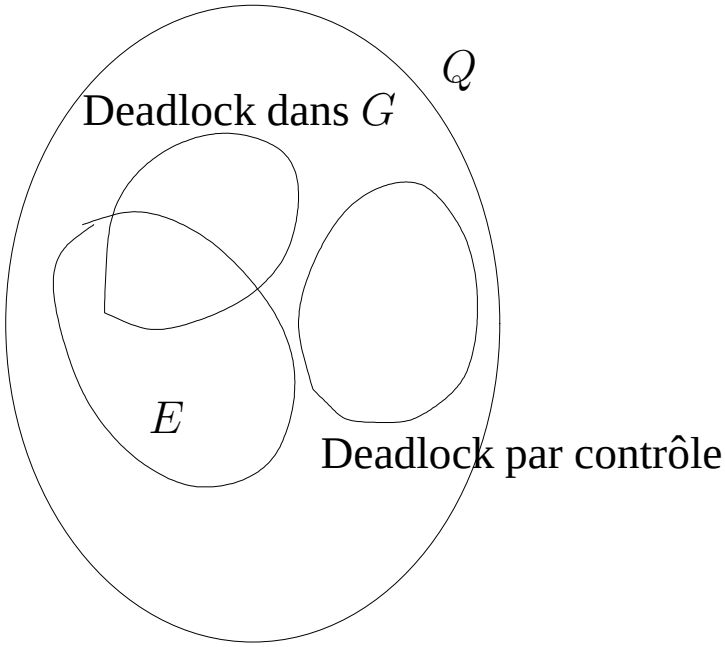


FIG. 3.7: Décomposition de l'ensemble des états en deadlock dans S/G

ment. Dans [53], l'absence de synchronisation entre les sous systèmes offre un critère intéressant pour caractériser les états en deadlock du système, en permettant une détection efficace. Dans [1], l'auteur donne lui aussi un critère permettant de réduire l'espace de recherche des états en deadlock, utilisant la structure concurrente du système et prenant en compte les synchronisations. L'exemple 18 illustre ce critère.

Exemple 18 *Dans cet exemple, le système concurrent G est obtenu par composition parallèle des sous systèmes G_1 et G_2 fournis en figure 3.8. L'automate G_1 est défini sur l'alphabet $\Sigma_1 = \{a, b, c, d, e\}$ et l'automate G_2 sur l'alphabet $\Sigma_2 = \{a', b, c, d', e'\}$. Les automates G_1 et G_2 sont trim, et donc localement non bloquants. En revanche, le système global G peut s'avérer bloquant. Ceci provient des synchronisations qui s'effectuent entre les sous systèmes.*

L'ensemble des événements partagés par G_1 et G_2 est noté Σ_s et vaut $\Sigma_s = \{b, c\}$. Dans cet exemple, seul les états accessibles de $G_1 \parallel G_2$ sont représentés. Parmi ces états, seul l'état (q_1, q_1') est en deadlock. Le fait que (q_1, q_1') soit en deadlock est dû à l'absence de synchronisation possible lorsque le système G_1 est dans l'état q_1 alors que le système G_2 est dans l'état q_1', ainsi qu'à l'absence d'événements locaux (i.e non partagés) tirables depuis chacun de ces deux états.

$\diamond$

L'exemple 18 met en évidence que les composantes des états en deadlock dans un système concurrent ne permettent que l'occurrence d'événements partagés (i.e appartenant à Σ_s). En d'autres termes, si un état q est en deadlock dans $G = \parallel G_i$, alors toute composante q_i de q dans $G_i = (\Sigma_i, Q_i, q_{0i}, \delta_i)$ ne permet que le tirage d'événements partagés. Par conséquent, étant donné un système concurrent G, on s'intéressera dans la suite à l'ensemble des états *potentiellement*

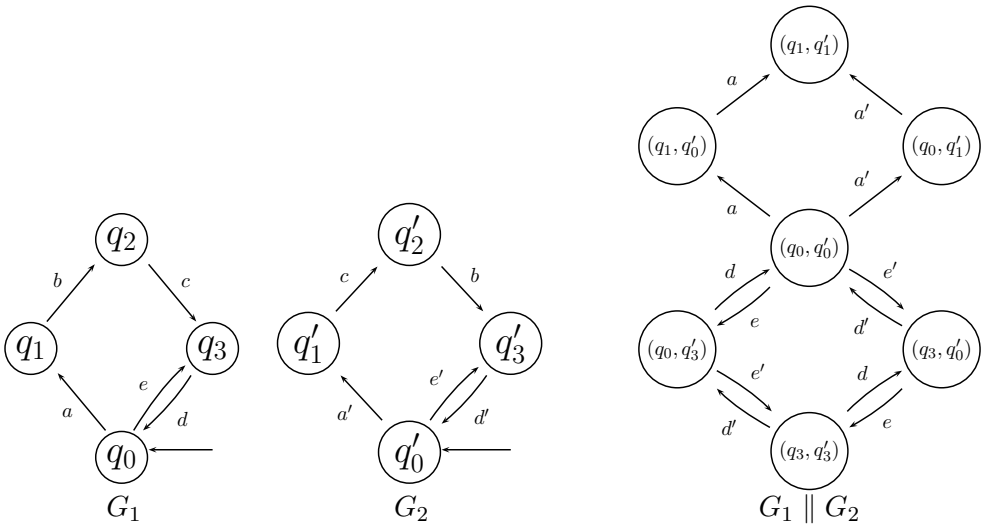


FIG. 3.8: Exemple de système concurrent possédant un état en deadlock.

bloquants $\mathrm{PB}(G)$ donné par :

$$\mathrm{PB}(G) = \{(q_1, \ldots, q_n) \in Q |\ \forall i, \delta_i(q_i) \subseteq \Sigma_s\} \tag{3.15}$$

L'intérêt de l'ensemble $\mathrm{PB}(G)$ est que le nombre de ses éléments peut être beaucoup plus restreint que celui de l'ensemble des états du système. Ce cas se produit notamment lorsque le système considéré est *faiblement synchronisé*[6]. En effet, l'occurrence d'événements partagés depuis un état du système est alors peu fréquente et l'ensemble $\mathrm{PB}(G)$ peut être beaucoup plus restreint que l'ensemble des états de G. Notons de plus que cette hypothèse intervient implicitement dans [53], pour assurer l'efficacité de l'approche proposée. En effet, dans [53], les systèmes considérés sont modulaires au sens où leurs sous systèmes ne partagent aucun événements, et sont par conséquent faiblement synchronisés.

De plus, on peut noter que $\mathrm{PB}(G)$ est un pavé. En effet, si pour $i \in \{1, \ldots, n\}$, on note $\mathrm{PB}_i(G_i)$ l'ensemble des états de G_i ne pouvant tirer que des événements partagés,

$$\mathrm{PB}_i(G_i) = \{q_i \in Q_i |\ \delta_i(q_i) \subseteq \Sigma_s\} \tag{3.16}$$

alors

$$\mathrm{PB}(G) = \mathrm{PB}_1(G_1) \times \cdots \times \mathrm{PB}_n(G_n)$$

Dans cette section, on s'intéresse au problème de *l'évitement de deadlock dans un système contrôlé* introduit dans le paragraphe précédent. Bien que les systèmes à contrôler sont ici supposés être concurrents, les systèmes contrôlés ne le sont pas en général.

Finalement, on supposera dans la suite que l'ensemble $\mathrm{PB}(G)$ est suffisamment restreint pour en permettre une exploration et ainsi détecter les états en deadlock dans le système concurrent G.

[6] Faiblement synchronisé signifie ici que le nombre d'événements partagés ainsi que leur occurrence sont faibles.

3.3.3 Détection de deadlock par contrôle dans un système concurrent contrôlé

Le paragraphe précédent décrit comment caractériser les états en deadlock dans un système concurrent, ce qui permet de les détecter sans parcourir l'ensemble des états du système. Or on souhaite en fait détecter efficacement l'ensemble des états en deadlock dans le système contrôlé. La figure 3.7 suggère que ces états peuvent être des états en deadlock dans le système (cas du paragraphe précédent), mais aussi des états en deadlock par contrôle.

C'est pourquoi, on s'intéresse dans ce paragraphe à la détection des états en deadlock par contrôle dans S/G. Bien que G soit concurrent, S/G ne l'est pas en général, et le critère fourni précédemment n'est plus applicable. Notre but consiste alors à donner un critère utilisant non seulement la structure concurrente de G, mais aussi la forme particulière du superviseur qui lui est appliqué, pour caractériser les états en deadlock par contrôle dans S/G.

Dans cette optique, le problème de détection des états en deadlock par contrôle dans S/G est à son tour décomposé. En effet, les états en deadlock par contrôle dans S/G le sont

- soit uniquement par inhibition d'événements partagés du système G,
- soit par inhibition d'au moins un événement local.

Ces deux cas sont à présent successivement étudiés afin de résoudre le problème de la détection des états en deadlock par contrôle dans S/G.

États en deadlock par contrôle uniquement par inhibition d'événements partagés. Ce cas se traite à partir du critère donné dans le paragraphe précédent. En effet, on va ici montrer que les états en deadlock par contrôle dans S/G, pour lesquels le superviseur S n'interdit que des événements partagés du système, appartiennent en fait à l'ensemble $\mathrm{PB}(G)$.

Lemme 13 *Soit G un système concurrent. On note Σ_s l'ensemble des événements partagés de G, et on considère un ensemble E d'états de G ainsi que le superviseur standard S assurant l'interdiction de E dans G, tel que décrit par (3.2). Si q est en deadlock par contrôle dans S/G tel que $S(q) \subseteq \Sigma_s$, alors $q \in PB(G)$.*

Le lemme 13 traduit donc qu'un état en deadlock par contrôle dans S/G uniquement par inhibition d'événements partagés appartient à l'ensemble $\mathrm{PB}(G)$. Par conséquent, l'ensemble $\mathrm{PB}(G)$ peut être utilisé pour restreindre l'espace de recherche de ce type d'états en deadlock.

Démonstration : On note $G = \|_{1 \leq i \leq n} G_i$ avec pour tout $i \in \{1, \ldots, n\}$, $G_i = (\Sigma_i, Q_i, q_{0i}, \delta_i)$. On considère un état $q = (q_1, \ldots, q_n)$ de G tel que pour tout $i \in \{1, \ldots, n\}$, $q_i \in Q_i$. On raisonne par l'absurde. On suppose pour cela que q est en deadlock par contrôle dans S/G, que $S(q) \subseteq \Sigma_s$ et que $q \notin \mathrm{PB}(G)$.

Puisque $q \notin \mathrm{PB}(G)$, il existe $i \in \{1, \ldots, n\}$ tel que $\delta_i(q_i) \not\subseteq \Sigma_s$. On considère un tel entier i et un événement $\sigma \in \Sigma_i$ tels que $\delta_i(q_i, \sigma)!$ et $\sigma \notin \Sigma_s$. Par conséquent, $\mathrm{IN}(\sigma) = \{i\}$ et donc $\forall i \in \mathrm{IN}(\sigma)$, on a $\delta_i(q_i, \sigma)!$. D'après la définition 5, on obtient que $\delta(q, \sigma)!$. Or $\sigma \notin \Sigma_s$ et $S(q) \subseteq \Sigma_s$ par hypothèse. Donc on déduit de (3.14) que $\delta_{S/G}(q, \sigma)!$ et donc que $\delta_{S/G}(q) \neq \emptyset$. Ceci contredit les hypothèses initiales et donne ainsi le résultat. $\diamond$

Finalement, si $\mathrm{PB}(G)$ peut être exploré pour déterminer les états en deadlock dans G, alors il peut l'être pour déterminer les états en deadlock par contrôle qui le sont uniquement par inhibition

d'événements partagés. De plus, si q est un état en deadlock dans un système à contrôler, alors q est aussi un état en deadlock dans le système contrôlé. C'est pourquoi, l'ensemble des états en deadlock dans S_E/G et appartenant à PB(G) contient non seulement les états en deadlock par contrôle uniquement par inhibition d'événements partagés, mais aussi les états en deadlock dans le système G.

Par conséquent, le problème de la détection des états en deadlock dans S_E/G se restreint à celui de la détection des états en deadlock dans PB(G) d'une part, et des états en deadlock par contrôle par inhibition d'au moins un événement local d'autre part.

États en deadlock par contrôle par inhibition d'au moins un événement local. Dans le paragraphe précédent, le lemme 13 permet de restreindre l'espace de recherche de certains états en deadlock par contrôle. Bien que ces états subissent l'action du superviseur, celui-ci n'intervient pas pour réduire cet espace de recherche. Dans ce paragraphe, on s'intéresse à la détection des états en deadlock par contrôle dans S/G, qui subissent l'action du superviseur et desquels au moins un événement local est inhibé. Dans ce cas, la structure particulière du superviseur standard est utilisée et intervient dans la réduction de l'espace de recherche de ce type d'états. Un résultat en ce sens est donné par le lemme 15. Afin de formaliser ce résultat, la notion de projection d'états et de pavés de systèmes concurrents est d'abord introduite.

Définition 22 *On considère un système concurrent $G = \|_{1\leq i\leq n} G_i$ avec pour tout $i \in \{1,\ldots,n\}$, $G_i = (\Sigma_i, Q_i, q_{0i}, \delta_i)$. Étant donnés un état $q = (q_1,\ldots,q_n) \in \times_{1\leq i\leq n} Q_i$ et un ensemble $\mathcal{A} \subseteq \{1,\ldots,n\}$, on définit la projection de q suivant $\mathcal{A}$, notée $p_{\mathcal{A}}(q)$, par*

$$p_{\mathcal{A}}(q) = \times_{i\in\mathcal{A}}\{q_i\} \tag{3.17}$$

Puisque l'opérateur d'union et de projection commutent, la définition donnée en 3.17 de la projection d'un état s'étend à une union de pavés E de la forme $E = \bigcup_{1\leq j\leq m} E_1^j \times \cdots \times E_n^j$ de la manière suivante :

$$p_{\mathcal{A}}(E) = \bigcup_{1\leq j\leq m} \left(\times_{i\in\mathcal{A}} E_i^j\right) \tag{3.18}$$

Par conséquent, la projection d'un état q (ou d'un pavé E) d'un système concurrent suivant un ensemble $\mathcal{A}$, consiste intuitivement à ne considérer que les composantes de q (ou celles de E) suivant cet ensemble. La projection représente donc un outil intéressant lorsque l'on souhaite se focaliser sur un sous ensemble des composantes d'un état (ou d'un pavé). Basé sur cette notion, on introduit le lemme 15, qui permet de restreindre l'espace de recherche de l'ensemble des états en deadlock par contrôle par inhibition d'au moins un événement local.

Proposition 15 *On reprend les notations précédemment introduites (G est concurrent, $E = \bigcup E^j$ (et $\Phi^*(E) = E$) est interdit et S est le superviseur standard décrit en (3.2)). Si q est en deadlock par contrôle dans S_E/G et s'il existe $1 \leq i \leq n$ et $\sigma \in \delta(q) \cap (\Sigma_i \setminus \Sigma_s)$ alors on a*

$$p_{\{i\}^c}(q) \in p_{\{i\}^c}(E)$$

où $\{i\}^c$ représente le complémentaire de $\{i\}$ dans $\{1,\ldots,n\}$.

Démonstration : On suppose que q est en deadlock par contrôle dans S/G et donc $\delta_{S/G}(q) = \emptyset$. On considère $i \in \{1, \ldots, n\}$ et $\sigma \in \Sigma_i \setminus \Sigma_s$ tel que $\delta(q, \sigma)!$. L'état $\delta(\sigma, q)$ est alors noté $q' = (q'_1, \ldots, q'_n)$. D'après la définition 5,

$$\forall j \neq i,\ q'_j = q_j \text{ et } q'_i = \delta_i(q_i, \sigma)$$

Puisque $\delta_{S/G}(q) = \emptyset$, on a $\delta(q) \subseteq S(q)$ et donc $\sigma \in S(q)$. On en déduit que $q' \in \mathcal{I}(E)$. Puisque $\Phi^*(E) = E$, on en déduit que $q' \in \mathcal{I}_{loc}(E)$. De plus, puisque $E = \bigcup E^j$, il existe $j \in \{1, \ldots, m\}$ tel que

$$q' \in \mathcal{I}_{loc}(E^j_1) \times \cdots \times \mathcal{I}_{loc}(E^j_m)$$

Finalement, en projetant sur $\text{IN}(\sigma)^c$, on obtient que $p_{\text{IN}(\sigma)^c}(q') \in p_{\text{IN}(\sigma)^c}(\mathcal{I}_{loc}(E^j))$. Finalement,

$$p_{\text{IN}(\sigma)^c}(q') \in \times_{k \in \text{IN}(\sigma)^c} \mathcal{I}_{loc}(E^j_k)$$

Par conséquent, $\forall k \notin \text{IN}(\sigma)$, $q'_k \in \mathcal{I}_{loc}(E^j_k)$. Puisque $\text{IN}(\sigma) = \{i\}$, on a $\forall k \notin \text{IN}(\sigma)$, $q'_k = q_k$ et on obtient que $\forall k \in \text{IN}(\sigma)^c$, $q_k \in \mathcal{I}_{loc}(E^j_k)$.

On souhaite à présent être plus précis et montrer que $\forall k \in \text{IN}(\sigma)^c$, on a $q_k \in E^j_k$. Pour cela, on suppose que $q_k \notin E^j_k$ et on cherche à contredire les hypothèses initiales.

Soit $k \in \text{IN}(\sigma)^c$. Si $q_k \notin E^j_k$, alors il existe $\sigma' \in \Sigma_{uc,k} \setminus \Sigma_s$ tel que $\delta_k(q_k, \sigma')$ existe. On obtient alors que $\forall k' \in \text{IN}(\sigma')$, $\delta_{k'}(q_k, \sigma')!$ et donc $\delta(q, \sigma')$ (définition 5). Or $\sigma' \in \Sigma_{uc}$ donc $\sigma' \notin S(q)$ et donc $\sigma' \in \delta_{S/G}(q)$ (d'après (3.2)). Ceci contredit que q est en deadlock par contrôle dans S/G et on a donc $\forall k \in \text{IN}(\sigma)^c$, $q_k \in E^j_k$. En d'autres termes,

$$p_{\{i\}^c}(q) \in p_{\{i\}^c}(E)$$

$\diamond$

Du point de vue de l'efficacité de la détection des états en deadlock par contrôle, il est intéressant de rechercher ces états en ne considérant que la projection des états de E sur des sous ensembles de $\{1, \ldots, n\}$ possédant $n - 1$ éléments. Il suffit alors de déterminer la composante manquante. Par conséquent le proposition 15 donne une condition forte sur $n - 1$ des composantes d'un état en deadlock par contrôle dans S_E/G et depuis lequel un événement local est tirable dans G. Ceci est intéressant car il apparaît important, dans un système dont la complexité augmente de manière exponentielle avec le nombre de sous systèmes qui le composent, de pouvoir limiter la complexité de la recherche des composantes des états en deadlock.

Le résultat de la proposition 15 est obtenu à partir de la structure concurrente de G d'une part, et de la structure du superviseur d'autre part. Plus précisément, le fait que l'ensemble E des états interdits et l'ensemble $\mathcal{I}(E)$ puissent être donnés sous la forme d'une union de pavés, sont à la base du résultat du proposition 15.

Finalement, l'ensemble des états en deadlock dans S/G peut être déterminé en réduisant l'espace de recherche. Cette réduction dépend de la nature des états en deadlock que l'on souhaite détecter. Les résultats obtenus dans les paragraphes précédents sont résumés par la figure 3.9.

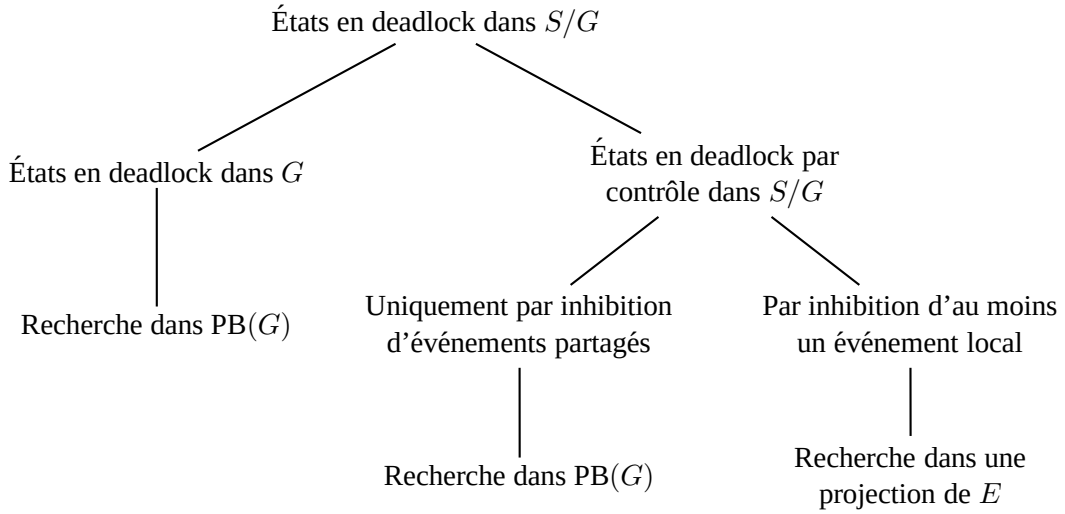


FIG. 3.9: Décomposition du problème de détection de deadlock dans S/G

Pour permettre un gain significatif en complexité lors de la phase de détection des deadlock, les résultats précédents tirent profit d'une part de la faible synchronisation qui peut caractériser le système, et d'autre part du faible nombre de pavés à interdire.

Toutefois, lorsque l'occurrence de synchronisations dans le système reste fréquente ou que le nombre d'états représentés par un pavé est important, la réduction de l'espace de recherche des états en deadlock peut ne pas être suffisante. Si on considère l'exemple des chariots filoguidés introduit en [33], seul 4 pavés sont à interdire. En revanche, chacun de ces pavés possède plusieurs centaines de milliers d'états, rendant insuffisante la réduction de l'espace de recherche proposée par la proposition 15. C'est pourquoi, dans la suite de cette section, on étend la notion de projection au système concurrent à contrôler et au système contrôlé correspondant. Le principe consiste à bénéficier de la structure particulière de ces systèmes pour affiner l'exploration de l'espace de recherche et éviter les problèmes liés à une forte synchronisation ou à des pavés possédant de nombreux éléments.

3.3.4 Raffinement

Dans ce paragraphe, on souhaite caractériser les états en deadlock dans un système concurrent sous contrôle, dans le but de restreindre l'espace de recherche de ces états. Cette caractérisation permet d'affiner l'exploration des espaces de recherche obtenus par les résultats précédents. De plus, cette caractérisation fait intervenir les structures particulières des systèmes à contrôler et contrôlés.

Plus précisément, l'approche consiste à travailler par approximations sur les composantes du tuple modélisant un état du système concurrent contrôlé. Tout d'abord, on rappelle que l'ensemble Q des états de $G = G_1 \parallel \cdots \parallel G_n$ est de la forme $Q = Q_1 \times \cdots \times Q_n$ et la forme de l'ensemble E des états à interdire est telle que décrite par l'équation (3.13).

On note S_E le superviseur assurant l'interdiction de E sur G tel que décrit en (3.2) et on note

Q_B, l'ensemble des états en deadlock par contrôle dans S_E/G. Plus précisément, on note

$$Q_B = \bigcup_j q_B^j, \text{ avec } \forall j,\ q_B^j = (q_{B,1}^j, \ldots, q_{B,n}^j) \in Q$$

L'idée de la méthode appliquée ici consiste dans un premier temps, à partir d'une (ou plusieurs) propriété caractérisant les états en deadlock, à déterminer un ensemble Q^1 tel que

$$\forall j, q_{B,1}^j \in Q^1$$

Q^1 représente donc une surapproximation de l'ensemble des composantes suivant G_1 des états en deadlock. Puis, en utilisant cette information, et toujours grâce à une propriété caractérisant les états en deadlock, on détermine $Q^2 \subseteq Q_1 \times Q_2$ tel que d'une part

$$\forall (q_1, q_2) \in Q_1 \times Q_2, (q_1, q_2) \in Q^2 \Longrightarrow q_1 \in Q^1$$

et d'autre part

$$\forall j,\ (q_{B,1}^j, q_{B,2}^j) \in Q^2$$

Par conséquent, Q^2 est une surapproximation des composantes suivant G_1 et G_2 des états en deadlock. De plus, Q^2 correspond à un ensemble de couples dont la première composante appartient à Q^1. On agit ainsi de manière itérative, utilisant à chaque étape une propriété caractérisant les états en deadlock par contrôle dans S_E/G, jusqu'à l'obtention de l'ensemble Q^n. Les propriétés utilisées à chaque étape doivent alors assurer que Q^n représente l'ensemble des états en deadlock dans $\mathcal{S}_E/G$.

On peut noter que dans une telle méthode, l'espace de recherche est affiné à chaque étape de l'itération. Par conséquent, l'explosion combinatoire des calculs, due à la mise en parallèle de sous systèmes, est réduite (voir inhibée dans les cas les plus favorables).

Dans le paragraphe suivant, on définit les notions de projection d'un ensemble, d'un système, d'un superviseur, et d'un système contrôlé. Ceci permettra de formaliser les sous produits du système, mais aussi de donner des caractérisations des états en deadlock par contrôle dans S_E/G, qui permettent d'appliquer la méthode décrite ci-dessus.

3.3.4.1 Projections de systèmes concurrents sous contrôle.

Dans ce paragraphe, on reprend les notations et hypothèses de la section 3.1.1. On considère donc n automates $(G_i)_{1\leq i\leq n}$, avec pour $i \in \{1, \ldots, n\}$, $G_i = (\Sigma_i, Q_i, q_{0i}, \delta_i)$. À partir des automates $(G_i)_{1\leq i\leq n}$, on introduit l'automate $G = (\Sigma, Q, q_0, \delta)$ obtenu par composition parallèle des $(G_i)_{1\leq i\leq n}$ et modélisant le système à contrôler. Pour un sous ensemble $\mathcal{A}$ de $\{1, \ldots, n\}$, on note $p_{\mathcal{A}}(\cdot)$ la projection suivant $\mathcal{A}$. Afin d'alléger les notations, $p_{\mathcal{A}}(\cdot)$ représente un opérateur agissant sur différents types d'objet. On va notamment introduire la notion de projection d'un système représenté par une composition parallèle de sous systèmes. Intuitivement, cette projection représente la composition parallèle d'un sous ensemble de ces sous systèmes.

Projection de systèmes concurrents. Comme évoqué précédemment, on souhaite caractériser les états en deadlock dans un système concurrent G sous contrôle d'un superviseur S, afin de pouvoir les détecter de manière efficace. Pour cela, une méthode itérative sur l'ensemble des sous systèmes de G tirant profit de cette caractérisation sera appliquée. Intuitivement, les ensembles de sous systèmes considérés à chaque étape de l'itération sont obtenus par projection de l'ensemble des sous systèmes de G. Toutefois, il est intéressant de munir cette projection d'une sémantique. L'idée consiste à définir la projection d'une composition parallèle d'automates $G_1 \parallel \ldots \parallel G_n$ suivant un ensemble $\mathcal{A}$ à partir de la composition parallèle de sous automates des $\{G_i\}_{i\in\mathcal{A}}$.

De plus, les événements locaux et partagés jouent un rôle important dans la résolution des problèmes de synthèse de contrôleurs. C'est pourquoi, il apparaît important de s'assurer que la nature des événements n'est pas altérée par projection.

Définition 23 *Soient $(G_i)_{1\leq i\leq n}$ un ensemble d'automates tels que pour $i \in \{1,\ldots,n\}$, $G_i = (\Sigma_i, Q_i, q_{0i}, \delta_i)$. On note $G = \|_i\ G_i$ et Σ_s l'ensemble des événements partagés de G. Soit $\emptyset \subset \mathcal{A} \subseteq \{1,\ldots,n\}$. On introduit l'ensemble d'événements $\Sigma_{s,\mathcal{A}}$ défini par*

$$\Sigma_{s,\mathcal{A}} = \Sigma_s(\{G_i\}_{\mathcal{A}}) \cap \Sigma_s(\{G_i\}_{\mathcal{A}^c}) = \{\sigma \in \Sigma_s |\ In(\sigma) \cap \mathcal{A} \neq \emptyset \text{ et } In(\sigma) \cap \mathcal{A}^c \neq \emptyset\} \quad (3.19)$$

où $\mathcal{A}^c$ représente l'ensemble complémentaire de $\mathcal{A}$ dans $\{1,\ldots,n\}$ et $\Sigma_s(\{G_i\}_{\mathcal{A}})$ représente l'ensemble des événements partagés par les automates $\{G_i\}_{\mathcal{A}}$ (voir définition 1.18).

Définition 24 *La projection de $G = G_1 \parallel \cdots \parallel G_n$ suivant $\mathcal{A}$, notée $p_{\mathcal{A}}(G)$, est définie par*

$$p_{\mathcal{A}}(G) = \|_{i\in\mathcal{A}}\ G_i^{\mathcal{A}} \quad (3.20)$$

avec $\Sigma_i^{\mathcal{A}} = \Sigma_i \setminus \Sigma_{s,\mathcal{A}}$, où $\Sigma_{s,\mathcal{A}}$ est donné par la définition 23, $\delta_i^{\mathcal{A}}$ est la restriction de δ_i à $Q_i \times \Sigma_i^{\mathcal{A}}$ et $G_i^{\mathcal{A}} = (\Sigma_i^{\mathcal{A}}, Q_i, q_{0_i}, \delta_i)$.

En d'autres termes, pour $i \in \mathcal{A}$, $G_i^{\mathcal{A}}$ correspond à l'automate G_i restreint à un sous ensemble de son alphabet. Par la suite, pour un tel système G, on notera $p_{\mathcal{A}}(\Sigma) = \bigcup_{i\in\mathcal{A}} \Sigma_i^{\mathcal{A}}$ et $p_{\mathcal{A}}(G) = (p_{\mathcal{A}}(\Sigma), p_{\mathcal{A}}(Q), p_{\mathcal{A}}(q_0), \delta_{p_{\mathcal{A}}(G)})$. De plus, $\Sigma_{s,\mathcal{A}}$ représente l'ensemble des événements partagés de G qui sont à la fois partagés par un automate G_i appartenant à $\mathcal{A}$ et un automate G_j n'appartenant pas à $\mathcal{A}$. Ces événements sont partagés dans le système initial G puisqu'ils appartiennent à au moins deux sous systèmes de G. Toutefois, la projection suivant un ensemble $\mathcal{A}$ peut induire qu'un tel événement n'appartienne plus qu'à un seul des automates résultant de la projection. La prise en considération de ce type d'événements permet d'assurer que la nature partagée/locale des événements n'est pas modifiée par projection.

Le lemme suivant permet de faire le lien entre les événements tirables depuis un état q d'un système concurrent G, et ceux tirables depuis la projection de q dans la projection de G correspondante, pour un ensemble $\mathcal{A}$ donné.

Lemme 14 *Soit $G = \|_{1\leq i\leq n}\ G_i$ un système concurrent. On note (Σ, Q, q_0, δ) l'automate modélisant G. Soient $\emptyset \subset \mathcal{A} \subseteq \{1,\ldots,n\}$, $q \in Q$ et $\sigma \in \Sigma$.*

$$\delta_{p_{\mathcal{A}}(G)}(p_{\mathcal{A}}(q), \sigma)! \Longrightarrow \delta(q,\sigma)! \wedge [\delta_{p_{\mathcal{A}}(G)}(p_{\mathcal{A}}(q), \sigma) = p_{\mathcal{A}}(\delta(q,\sigma))]$$

Démonstration : On suppose que $\delta_{p_{\mathcal{A}}(G)}(p_{\mathcal{A}}(q),\sigma)!$. Par conséquent, $\sigma \in \Sigma^{\mathcal{A}}$ et donc $\text{IN}(\sigma) \subseteq \mathcal{A}$. On en déduit que $\forall i \in \text{IN}(\sigma),\ \delta_i(q_i,\sigma)!$ et donc que $\delta(q,\sigma)!$.

On note maintenant $q' = \delta_{p_{\mathcal{A}}(G)}(p_{\mathcal{A}}(q),\sigma)$ et $q'' = \delta(q,\sigma)$. On a

$$(\forall i \in \text{IN}(\sigma),\ q'_i = \delta_i(q_i,\sigma)) \wedge (\forall i \notin \text{IN}(\sigma),\ q'_i = q_i)$$

Donc $\forall i \in \mathcal{A} \setminus \text{IN}(\sigma),\ q'_i = q_i$. Et puisque $\forall i \notin \text{IN}(\sigma),\ q''_i = q_i$, on en déduit que $\forall i \in \mathcal{A}, q'_i = q''_i$. Par conséquent, on a $q' = p_{\mathcal{A}}(q'')$. D'où le résultat. ◇

Le lemme 14 traduit d'une part que seuls les événements tirables depuis un état q de G peuvent être tirables depuis $p_{\mathcal{A}}(q)$ dans $p_{\mathcal{A}}(G)$. D'autre part, l'état atteint par tirage d'un événement σ depuis $p_{\mathcal{A}}(q)$ dans $p_{\mathcal{A}}(G)$ est la projection suivant $\mathcal{A}$ de l'état atteint depuis q par tirage de σ dans G.

De plus, il est possible de donner une caractérisation des états en deadlock dans un système concurrent à partir du lemme 14.

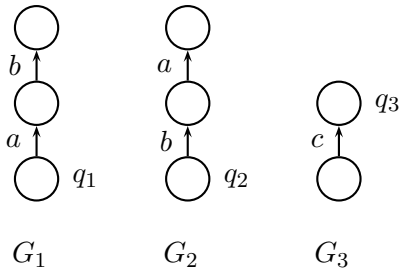
Corollaire 7 *Soit $G = \|_{1\leq i\leq n}\ G_i$ un système concurrent. On note (Σ, Q, q_0, δ) l'automate modélisant G. Soient $\emptyset \subset \mathcal{A} \subseteq \{1,\ldots,n\}$ et $q \in Q$. Si q est en deadlock dans G, alors $p_{\mathcal{A}}(q)$ est en deadlock dans $p_{\mathcal{A}}(G)$.*

Démonstration : Supposons que $p_{\mathcal{A}}(q)$ n'est pas en deadlock dans $p_{\mathcal{A}}(G)$ et considérons $\sigma \in p_{\mathcal{A}}(\Sigma)$ tel que $\delta_{p_{\mathcal{A}}(G)}(p_{\mathcal{A}}(q),\sigma)!$. D'après le lemme 14, $\delta(q,\sigma)!$ et par conséquent q n'est pas en deadlock dans G. ◇

Le corollaire 7 permet de raffiner l'exploration de l'espace de recherche des états en deadlock dans un système concurrent. En effet, les résultats de la section 3.3.2 permettent de réduire l'espace de recherche des états en deadlock dans un système concurrent. Toutefois, lorsque l'espace obtenu est de taille importante, il peut être intéressant de rechercher les composantes de ces états de manière incrémentale. Le corollaire 7 offre cette possibilité en indiquant que les composantes considérées doivent représenter un état en deadlock dans le système projeté correspondant.

Discussion sur la projection d'un système concurrent. On s'intéresse à présent à la pertinence de la définition 24 de la projection d'un système concurrent. En effet, cette définition permet une caractérisation des composantes des états en deadlock à travers le corollaire 7. En particulier, on va montrer par l'exemple 19 l'importance de prendre en considération les événements de $\Sigma_{s,\mathcal{A}}$ (i.e partagés par au moins deux automates dont un "appartient" à $\mathcal{A}$ et l'autre pas).

Exemple 19 *On considère trois automates G_1, G_2 et G_3, dont les alphabets respectifs sont $\Sigma_1 = \Sigma_2 = \{a,b\}$ et $\Sigma_3 = \{c\}$.*



Notons que l'état $q = (q_1, q_2, q_3)$ est en deadlock dans G. Si on considère à présent l'ensemble $\mathcal{A} = \{2, 3\}$, alors $p_{\mathcal{A}}(q) = (q_2, q_3)$ et d'après la définition 24, on a $\delta_{p_{\mathcal{A}}(G)}(p_{\mathcal{A}}(q)) = \emptyset$, ce qui reflète le corollaire 7.

La définition 24 prend en considération les événements de $\Sigma_{s,\mathcal{A}}$. Or une définition plus "naïve" de la projection d'un système concurrent pourrait amener à ne pas prendre ces événements en compte et à considérer que la projection de G suivant l'ensemble $\{2, 3\}$ vaut $G_2 \parallel G_3$. Or dans ce cas, l'événement b appartient à l'alphabet de cette projection et n'est pas partagé dans $G_1 \parallel G_2$. Par conséquent, $\delta_{G_1 \| G_2}((q_1, q_2)) = \{b\}$ $(\neq \emptyset)$.

Par conséquent, si les événements de $\Sigma_{s,\mathcal{A}}$ n'étaient pas pris en compte, le corollaire 7 ne serait pas valide car (q_1, q_2) représenterait la projection d'un état en deadlock (ici q) dans G, et ne serait lui même pas en deadlock dans la projection de G. ◇

Dans la suite, on souhaite étendre la notion de projection à un système contrôlé S/G, mais aussi faire le lien entre la sémantique de S/G et celle de ses projections.

Projection d'un superviseur. Dans ce paragraphe, on considère toujours un système concurrent G, mais aussi un superviseur standard S_E assurant l'interdiction d'un ensemble d'état $E(= \Phi^*(E))$ de G. À partir des définitions de la projection d'un système et d'un ensemble d'états de S_E/G, on introduit la notion de projection d'un superviseur et d'un système contrôlé. Le but consiste à utiliser la structure particulière du système pour rendre plus efficace l'exploration de l'espace de recherche des états en deadlock (et notamment en deadlock par contrôle).

Définition 25 *Soient G un système concurrent et E un ensemble d'états de G. Soit S_E le superviseur assurant l'interdiction de E dans G tel que décrit en (3.2).*

- *La projection de S_E suivant $\mathcal{A}$, notée $p_{\mathcal{A}}(S_E)$ est le superviseur défini pour tout $q \in p_{\mathcal{A}}(G)$ par*

$$p_{\mathcal{A}}(S_E)(q) = \{\sigma \in \Sigma_c \cap p_{\mathcal{A}}(\Sigma) |\ \delta_{p_{\mathcal{A}}(G)}(q, \sigma)! \wedge \delta_{p_{\mathcal{A}}(G)}(q, \sigma) \in p_{\mathcal{A}}(\mathcal{I}(E))\}$$

- *La projection du système contrôlé S_E/G suivant $\mathcal{A}$, notée $p_{\mathcal{A}}(S_E/G)$ est définie par*

$$p_{\mathcal{A}}(S_E/G) = p_{\mathcal{A}}(S_E)/p_{\mathcal{A}}(G) \tag{3.21}$$

Par conséquent, si G représente un système concurrent $G = \|_{1 \leq i \leq n} G_i$, E un ensemble d'états de G et $\mathcal{A}$ un sous ensemble de $\{1, \ldots, n\}$, alors la projection du superviseur S_E assurant l'interdiction de E dans G, notée $p_{\mathcal{A}}(S_E)$ est définie de manière naturelle comme le superviseur qui interdit

les événements qui mènent à la projection de $\mathcal{I}(E)$ (i.e $p_{\mathcal{A}}(\mathcal{I}(E))$), dans la projection de G (i.e $p_{\mathcal{A}}(G)$). De plus, la projection du système contrôlé S_E/G est définie comme le système contrôlé obtenu par l'action de $p_{\mathcal{A}}(S_E)$ sur $p_{\mathcal{A}}(G)$.

De plus, on peut noter que $p_{\mathcal{A}}(S_E)$ est principalement caractérisée par l'ensemble $p_{\mathcal{A}}(\mathcal{I}(E))$. Or cet ensemble peut être facilement déterminé dès lors que $\mathcal{I}(E)$ est décrit sous forme d'une union de pavé. En effet, on a vu en section 3.2.2 que $\mathcal{I}(E)$ pouvait s'exprimer comme une union de pavés (donnée par $\mathcal{I}_{loc}(\Phi^*(E))$). Or la projection d'une union de pavé (et donc le calcul de $p_{\mathcal{A}}(S_E)$) peut être déterminée sans effort, comme le montre l'égalité (3.18).

Pour un état q de G, le lemme 15 permet de faire le lien entre les ensembles d'événements interdits par S_E en q (i.e $S_E(q)$) et ceux interdits par $p_{\mathcal{A}}(S_E)$ en $p_{\mathcal{A}}(q)$ (i.e $p_{\mathcal{A}}(S_E)(p_{\mathcal{A}}(q))$).

Lemme 15 *Soit $G = \|_{1\leq i\leq n}\ G_i$ un système concurrent. On note (Σ, Q, q_0, δ) l'automate modélisant G. Soient $\emptyset \subset \mathcal{A} \subseteq \{1,\ldots,n\}$ et $q \in Q$. De plus on considère un ensemble d'états $E \subseteq Q$ et le superviseur S_E assurant l'interdiction de E dans G, tel que décrit en (3.2).*

$$S_E(q) \cap p_{\mathcal{A}}(\Sigma) \subseteq p_{\mathcal{A}}(S_E)(p_{\mathcal{A}}(q))$$

Démonstration : Soit $\sigma \in S_E(q) \cap p_{\mathcal{A}}(\Sigma)$. Puisque $\sigma \in p_{\mathcal{A}}(\Sigma)$, on a $\text{IN}(\sigma) \subseteq \mathcal{A}$ d'après la définition 24. De plus, par définition du superviseur standard interdisant E sur G, donné en 3.2,

$$\sigma \in S_E(q) \Longrightarrow \sigma \in \Sigma_c \wedge \delta(q,\sigma)! \wedge \delta(q,\sigma) \in \mathcal{I}(E)$$

Or

$$\begin{array}{rcl} \delta(q,\sigma)! & \Longrightarrow & \forall i \in \text{IN}(\sigma),\ \delta_i(q_i,\sigma)! \quad \text{(d'après la définition 5)} \\ & \Longrightarrow & \forall i \in \text{IN}(\sigma) \cap \mathcal{A},\ \delta_i(q_i,\sigma)! \quad (\text{car } \text{IN}(\sigma) \subseteq \mathcal{A}) \\ & \Longrightarrow & \delta_{p_{\mathcal{A}}(G)}(p_{\mathcal{A}}(q),\sigma)! \end{array}$$

Par conséquent, d'après le lemme 14, si $\delta_{p_{\mathcal{A}}(G)}(p_{\mathcal{A}}(q),\sigma)!$ alors

$$\delta(q,\sigma)! \wedge [\delta_{p_{\mathcal{A}}(G)}(p_{\mathcal{A}}(q),\sigma) = p_{\mathcal{A}}(\delta(q,\sigma))] \qquad (\alpha)$$

Or si $\delta(q,\sigma) \in \mathcal{I}(E)$, alors $p_{\mathcal{A}}(\delta(q,\sigma)) \in p_{\mathcal{A}}(\mathcal{I}(E))$. Donc d'après (α), si $\delta(q,\sigma) \in \mathcal{I}(E)$, alors $\delta_{p_{\mathcal{A}}(G)}(p_{\mathcal{A}}(q),\sigma) \in p_{\mathcal{A}}(\mathcal{I}(E))$. Finalement, on en déduit que

$$\sigma \in S_E(q) \Longrightarrow \Big(\delta_{p_{\mathcal{A}}(G)}(p_{\mathcal{A}}(q),\sigma)! \wedge \delta_{p_{\mathcal{A}}(G)}(p_{\mathcal{A}}(q),\sigma)) \in p_{\mathcal{A}}(\mathcal{I}(E))\Big)$$

Ce qui signifie que $\sigma \in p_{\mathcal{A}}(S_E)(p_{\mathcal{A}}(q))$. ◇

Par conséquent, le lemme 15 traduit que le comportement du système contrôlé par la projection d'un superviseur est plus restreint que celui obtenu par projection du système contrôlé par ce superviseur.

Cette propriété peut être utilisée pour améliorer la complexité de détection des états en deadlock par contrôle. Ceci se traduit par la proposition 16 qui fournit une caractérisation des états en deadlock dans un système concurrent contrôlé.

Proposition 16 *Si q est en deadlock dans $\mathcal{S}_E/G$, alors pour tout $\emptyset \subset \mathcal{A} \subseteq \{1, \ldots, n\}$, $p_{\mathcal{A}}(q)$ est en deadlock dans $p_{\mathcal{A}}(S_E/G)$.*

Démonstration : Soit $q \in Q$ un état en deadlock dans S_E/G. Cela signifie que $\delta_{S_E/G}(q) = \emptyset$. Par définition de S_E/G, on obtient que $\delta(q) \setminus S_E(q) = \emptyset$ ou encore

$$\delta(q) \subseteq S_E(q) \qquad (\alpha)$$

Par définition,

$$\delta_{p_{\mathcal{A}}(S_E/G)}(p_{\mathcal{A}}(q)) = \delta_{p_{\mathcal{A}}(G)}(p_{\mathcal{A}}(q)) \setminus p_{\mathcal{A}}(S_E)(p_{\mathcal{A}}(q))$$

Si $\delta_{p_{\mathcal{A}}(G)}(p_{\mathcal{A}}(q)) = \emptyset$, alors on a le résultat. Sinon, on considère $\sigma \in \delta_{p_{\mathcal{A}}(G)}(p_{\mathcal{A}}(q))$. Dans ce cas, $\delta_{p_{\mathcal{A}}(G)}(p_{\mathcal{A}}(q), \sigma)!$ et d'après le lemme 14 on obtient que $\delta(q, \sigma)!$ et donc que $\sigma \in \delta(q)$. Finalement, d'après (α), $\sigma \in S_E(q)$. En appliquant alors le lemme 15 (on a $\sigma \in \bigcup_i \Sigma_i^{\mathcal{A}} \cap \delta(q)$), on obtient que $\sigma \in p_{\mathcal{A}}(S_E)(p_{\mathcal{A}}(q))$. Finalement, on en déduit que

$$\delta_{p_{\mathcal{A}}(G)}(p_{\mathcal{A}}(q)) \subseteq p_{\mathcal{A}}(S_E)(p_{\mathcal{A}}(q))$$

d'où $\delta_{p_{\mathcal{A}}(S_E/G)}(p_{\mathcal{A}}(q)) = \emptyset$. ◇

On peut noter que la proposition 16 représente en fait une extension du corollaire 7. En effet, ce corollaire est obtenu lorsque $E = \emptyset$ qui modélise le cas où il n'y a pas d'état à interdire.

L'idée intuitive de la proposition 16 est la suivante : lorsque l'on considère la projection d'un état q en deadlock dans $\mathcal{S}_E/G$, cette projection est en deadlock dans la projection de $\mathcal{S}_E/G$. De plus, par analogie avec le corollaire 7, la proposition 16 offre un critère permettant d'explorer l'espace de recherche des états en deadlock dans S_E/G de manière incrémentale. L'espace de recherche ainsi parcouru peut être affiné à chaque itération, afin d'être réduit. Cette démarche sera explicitée dans le paragraphe suivant.

Discussion sur la projection d'un superviseur. On peut enfin s'interroger sur la pertinence de la définition de la projection d'un superviseur. En effet, la définition 25 assure la validité de la proposition 16, mais ce serait aussi le cas si la projection d'un superviseur était définie comme le superviseur "trivial" qui n'autorise l'occurrence d'aucun événement. Le lemme 16 met en évidence la pertinence de la projection d'un superviseur telle qu'introduite par la définition 25.

Lemme 16 *Soit $G = \|_{1 \leq i \leq n} G_i$ un système concurrent. On note (Σ, Q, q_0, δ) l'automate modélisant G. De plus on considère un ensemble d'états $E \subseteq Q$ et le superviseur S_E assurant l'interdiction de E dans G, tel que décrit en 3.2. On considère $\emptyset \subset \mathcal{A} \subseteq \{1, \ldots, n\}$, un état $q \in p_{\mathcal{A}}(Q)$ et $\sigma \in \Sigma$. Si $\sigma \in p_{\mathcal{A}}(S_E)(q)$ alors $\exists q' \in Q$ tel que $q = p_{\mathcal{A}}(q')$ et $\sigma \in S_E(q')$.*

De manière intuitive, les événements interdits par $p_{\mathcal{A}}(S_E)$ en $p_{\mathcal{A}}(q)$ sont des événements interdits par S_E depuis un état du système global G. Par conséquent, la projection d'un superviseur telle que décrite par la définition 25 n'interdit pas d'événement "superflu".

Démonstration : Soit $\sigma \in p_{\mathcal{A}}(S_E)(q)$. D'après la définition 25, $\delta_{p_{\mathcal{A}}(G)}(q, \sigma)!$ et $\delta_{p_{\mathcal{A}}(G)}(q, \sigma) \in p_{\mathcal{A}}(\mathcal{I}(E))$. On considère $q'' \in \mathcal{I}(E)$ tel que

$$\delta_{p_{\mathcal{A}}(G)}(q,\sigma) = p_{\mathcal{A}}(q'') \qquad (\alpha)$$

et on définit $q' = (q'_1, \ldots, q'_n) \in Q$ tel que

$$\forall i \in \mathcal{A},\ q'_i = q_i \text{ et } \forall i \notin \mathcal{A},\ q'_i = q''_i \qquad (\beta)$$

Puisque $\sigma \in p_{\mathcal{A}}(S_E)(q)$, on a $\sigma \in p_{\mathcal{A}}(\Sigma)$ et donc $\textsc{In}(\sigma) \subseteq \mathcal{A}$. On déduit alors de (α) que

$$\forall i \in \textsc{In}(\sigma),\ q''_i = \delta_i(q_i,\sigma) \text{ et } \forall i \notin \textsc{In}(\sigma),\ q''_i = q_i$$

On déduit alors de (β) que

$$\forall i \in \textsc{In}(\sigma),\ q''_i = \delta_i(q'_i,\sigma) \text{ et } \forall i \notin \textsc{In}(\sigma),\ q''_i = q'_i$$

ce qui signifie que $\delta(q',\sigma)!$ et $\delta(q',\sigma) = q''$. Or $q'' \in \mathcal{I}(E)$ et $\sigma \in \Sigma_c$ (puisque $\sigma \in p_{\mathcal{A}}(S_E)(q')$). Par conséquent, $\sigma \in S_E(q')$ et on déduit de (β) que $p_{\mathcal{A}}(q') = q$. D'où le résultat. $\diamond$

Dans le paragraphe suivant, une méthode appliquant les résultats de cette section est introduite et illustrée par un exemple.

3.3.5 Résolution du problème de l'évitement de deadlock.

Dans cette section, on cherche à caractériser les états en deadlock dans un système concurrent contrôlé. Pour cela, on va utiliser le lemme 13, les propositions 15 et 16 et les notions de projections introduites dans le paragraphe précédent.

On reprend les notations utilisées dans ce chapitre : on considère un système concurrent modélisé par une composition parallèle d'automates $(G_i)_{1\leq i\leq n}$ où pour $i \in \{1,\ldots,n\}$, $G_i = (\Sigma_i, Q_i, q_{0i}, \delta_i)$. On note $G = (\Sigma, Q, q_0, \delta)$ l'automate $G = \|_i\ G_i$. On considère un ensemble d'états $E \subseteq Q$ de G à interdire. On suppose d'une part que $E = \Phi^*(E)$, et d'autre part que E est de la forme

$$E = \bigcup_{1\leq j\leq m} E^j, \quad \text{avec } E^j = E^j_1 \times \cdots \times E^j_n \quad \text{et } \forall i,j,\ E^j_i \subseteq Q_i$$

De plus, on note S_E le superviseur assurant l'interdiction de E dans G tel que décrit en 3.2. On rappelle à présent les différentes étapes permettant de résoudre le problème de l'interdiction d'un ensemble d'états E d'un G système concurrent G, couplé au problème de l'évitement de deadlock du système contrôlé obtenu.

(1) Calcul de S_E interdisant l'ensemble E dans G.
(2) Détermination de l'ensemble D des états en deadlock dans S_E/G.
(3) Tant que $D \neq \emptyset$, retour au point (1) avec $E := E \cup D$ et $G := S_E/G$.

Tout d'abord, l'étape (1) peut être traitée suivant la méthode décrite dans la section 3.1.2. Par conséquent, le problème consiste ici à déterminer l'ensemble D des états en deadlock dans S_E/G.

À présent, nous allons décrire la méthode de détection de l'ensemble D des états en deadlock dans $\mathcal{S}_E/G$. Elle s'articule suivant deux étapes :

Étape1 : Recherche des états en deadlock dans S_E/G qui appartiennent à PB(G).
Étape2 : Recherche des états en deadlock par contrôle dans S_E/G, par inhibition d'au moins un événement local.

Ces deux étapes sont suffisantes pour déterminer exactement l'ensemble des états en deadlock dans S_E/G. En effet, d'après la section 3.3.2 et le lemme 13, l'étape 1 permet de déterminer les états en deadlock dans G, ainsi que les états en deadlock par contrôle uniquement par inhibition d'événements partagés. De plus, en reprenant le schéma de la figure 3.9, on en déduit que les états en deadlock dans S_E/G qui ne sont pas déterminés dans l'étape 1 correspondent aux états en deadlock par contrôle par inhibition d'au moins un événement local. Or ces états sont déterminés au cours de l'étape 2, ce qui assure que l'ensemble D des états en deadlock dans S_E/G est bien obtenu en appliquant ces deux étapes.

Voyons maintenant plus en détail, comment exploiter les résultats des paragraphes précédents afin de traiter ces étapes de manière efficace.

(Étape 1) Pour déterminer les états en deadlock dans S_E/G qui appartiennent à PB(G) :

(a) Déterminer l'ensemble D_1' des états qui sont en deadlock dans $p_{\{1\}}(S_E/G)$ et qui appartiennent à $p_{\{1\}}(\text{PB}(G))$.

$$D_1' = \{q_1 \in Q_1 |\ q_1 \in \text{PB}_1(G_1) \wedge \delta_{p_{\{1\}}(S_E/G)}(q_1) = \emptyset\}$$

PB$_1(G_1)$ est défini en (3.16) et vaut $p_{\{1\}}(\text{PB}(G))$. Ceci provient du fait que PB(G) est un pavé (voir paragraphe 3.3.2).

(b) Pour i allant de 2 à n, déterminer l'ensemble des états de $p_{\{1,\dots,i\}}(Q)$ $(= Q_1 \times \cdots \times Q_i)$ qui sont en deadlock dans $p_{\{1,\dots,i\}}(S_E/G)$, appartiennent à $p_{\{1,\dots,i\}}(\text{PB}(G))$, et dont les composantes suivant $\{1,\dots,i-1\}$ correspondent à un élément de D_{i-1}.

$$\begin{aligned} D_i' = \quad & \{(q_1,\dots,q_i) \in Q_1 \times \cdots \times Q_i | \\ & (q_1,\dots,q_j) \in \text{PB}_1(G_1) \times \cdots \times \text{PB}_n(G_n) \\ & \wedge \delta_{p_{\{1,\dots,i\}}(S_E/G)}((q_1,\dots,q_i)) = \emptyset \wedge (q_1,\dots,q_{i-1}) \in D_{i-1}'\} \end{aligned}$$

(Étape 2) Pour tout $1 \leq i \leq n$ on cherche à déterminer les états q en deadlock par contrôle par inhibition d'au moins un événement local de Σ_i.

(a) Initialisation.

(a1) Si $i = 1$ alors on parcourt Q_2 pour déterminer les états de $p_{\{2\}}(\mathcal{S}_E/G)$ qui sont en deadlock (proposition 16) et appartiennent à $p_{\{2\}}(E)$ (proposition 15). Le résultat est noté $D_{2,i}$.

(a2) Sinon on parcourt Q_1 pour déterminer les états de $p_{\{1\}}(\mathcal{S}_E/G)$ qui sont en deadlock (proposition 16) et appartiennent à $p_{\{1\}}(E)$ (proposition 15). Le résultat est noté $D_{1,i}$.

(b) Pour $k \neq i$ allant de 2 à n, on parcourt $D_{k-1,i} \times Q_k$ pour déterminer les états de $p_{\{1,\dots,k\}\setminus\{i\}}(S_E/G)$ qui sont en deadlock (proposition 16) et appartiennent à $p_{\{1,\dots,k\}\setminus\{i\}}(E)$ (proposition 15). Le résultat est noté $D_{k,i}$.

(c) Finalisation.

(c1) Finalement, si $i = n$ alors on parcourt $D_{n-1,n} \times Q_n$ pour déterminer l'ensemble $D_{n,n}$ des états de $\mathcal{S}_E/G$ qui y sont en deadlock.

(c2) Sinon, on parcourt $D_{n,i} \times Q_i$ pour déterminer l'ensemble $D_{i,i}$ des états de $\mathcal{S}_E/G$ qui y sont en deadlock.

Le résultat de l'étape 2 est noté D'' et vaut $\bigcup_i D_{i,i}$.

Le but de l'étape 1 est de déterminer les états en deadlock dans S_E/G qui appartiennent aussi à PB(G). À chaque itération de cette étape, les ensembles D'_i obtenus contiennent non seulement les projections des états en deadlock dans G, mais aussi les projections des états en deadlock par contrôle dans S_E/G, uniquement par inhibition d'événements partagés. Cette propriété est assurée par l'utilisation de la proposition 16 et du lemme 13. Finalement, D'_n représente l'ensemble des états en deadlock dans S_E/G qui appartiennent à PB(G).

Le but de l'étape 2 est de déterminer les états en deadlock par contrôle dans S_E/G pour lesquels au moins un événements local est inhibé par contrôle. À chaque itération de cette étape, les ensembles $D_{k,i}$ obtenus contiennent les projections des états en deadlock par contrôle dans S_E/G, pour lesquels au moins un événements local de Σ_i est inhibé par contrôle. Cette propriété est assurée par l'utilisation des propositions 15 et 16. Finalement, D'' représente l'ensemble des états en deadlock par contrôle dans S_E/G pour lesquels au moins un événements local est inhibé par contrôle.

En ce référant à la figure 3.9, on en déduit que $D = D'' \cup D'_n$ représente l'ensemble des états en deadlock dans S_E/G.

3.3.5.1 Exemple

On reprend maintenant l'exemple 17 et on s'intéresse à la synthèse d'un superviseur maximal S assurant l'interdiction de E et l'évitement de deadlock dans le système contrôlé obtenu. Tout d'abord, reprenons le système G en ajoutant des labels permettant d'identifier les états globaux interdits. Ces labels traduisent l'appartenance des états aux ensembles $(\mathcal{I}_{loc}(E^j))_{1\leq j\leq n}$. Ainsi, $(q_4^1, q_4^2, q_3^3) \in \mathcal{I}_{loc}(E^1)$ car chacune des composantes de cet état est labélisée par $\{1\}$. Les composantes des états appartenant à $\mathcal{I}_{loc}(E^j)$ sont donc labélisées par $\{j\}$.

De plus, on rappelle que les événements partagés de G sont donnés par $\Sigma_s = \{\Sigma_{uc}, \sigma\}$ et l'ensemble E d'états à interdire est $E = E^1 \cup E^2 \cup E^3$ avec $E^1 = \{q_1^1, q_4^1\} \times \{q_4^2\} \times \{q_3^3\}$, $E^2 = \{q_3^1, q_2^1\} \times \{q_2^2, q_3^2\} \times \{q_2^3, q_3^3\}$ et $E^3 = \{q_3^1\} \times \{q_2^2\} \times \{q_3^3, q_1^3\}$.

L'ensemble E d'états à interdire, considéré dans l'exemple 17, n'était composé que de E^1 et E^2. Pour simplifier, on considère ici que $E = E^1 \cup E^2 \cup E^3$ et on a ainsi $E = \Phi^*(E)$. Par conséquent, d'après la proposition 14, $\mathcal{I}(E) = \mathcal{I}_{loc}(E)$. Comme évoqué précédemment l'ensemble $\mathcal{I}_{loc}(E)$ est décrit à partir des labels associés aux états du système (figure 3.3.5.1). Enfin, on rappelle que le superviseur standard S_E assurant l'interdiction de E dans G est décrit en 3.2 et vaut pour tout état q de G,

$$S_E(q) = \{\sigma \in \Sigma_c |\ \delta(q,\sigma)! \wedge \delta(q,\sigma) \in \mathcal{I}(E)\}$$

Enjeu : on s'intéresse maintenant à la détection des états en deadlock dans S_E/G. Pour cela, on applique la démarche en deux étapes décrite dans le paragraphe précédent.

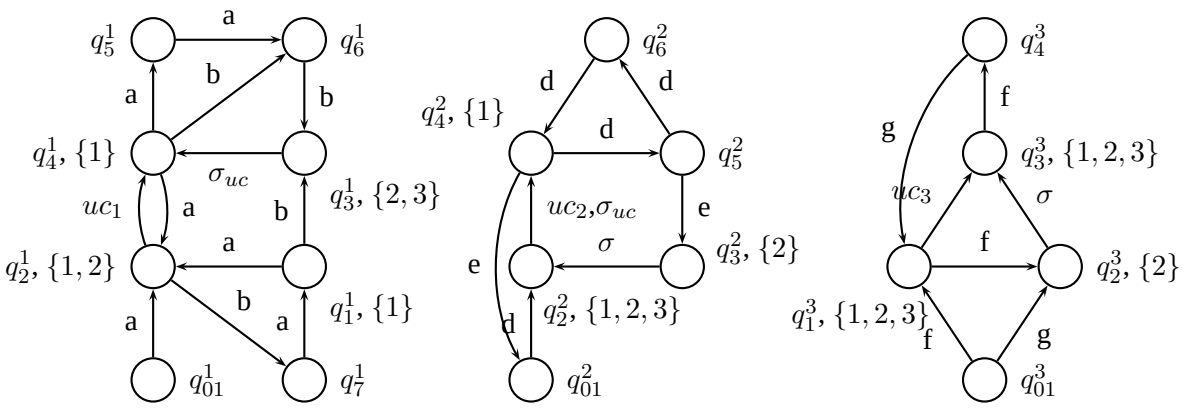


FIG. 3.10: Système concurrent G et représentation de $\mathcal{I}_{loc}(E)$ par des labels.

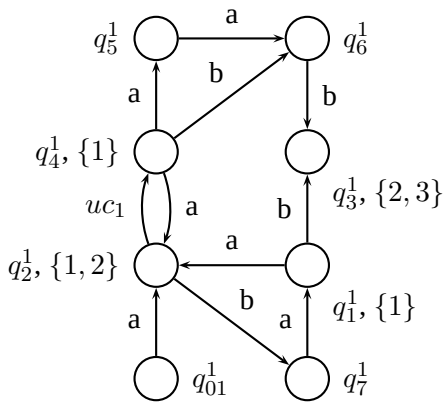


FIG. 3.11: $p_{\{1\}}(G)$

Étape 1 : On recherche les états en deadlock dans S_E/G qui appartiennent à PB(G). Pour cela, on explore une partie de l'ensemble PB(G) de manière incrémentale. Tout d'abord, on s'intéresse au point (a) de l'étape 1 et on considère ainsi $p_{\{1\}}(G)$, donné par la figure 3.11.

De plus, $p_{\{1\}}(S_E)$ est donné pour tout état q de $p_{\{1\}}(G)$ par

$$p_{\{1\}}(S_E)(q) = \{\sigma \in \Sigma_c \cap p_{\{1\}}(\Sigma) |\ \delta_{p_{\{1\}}(G)}(q,\sigma)! \wedge \delta_{p_{\{1\}}}(q,\sigma) \in p_{\{1\}}(\mathcal{I}(E))\}$$

Or on a vu que $\mathcal{I}(E) = \mathcal{I}_{loc}(E)$ donc

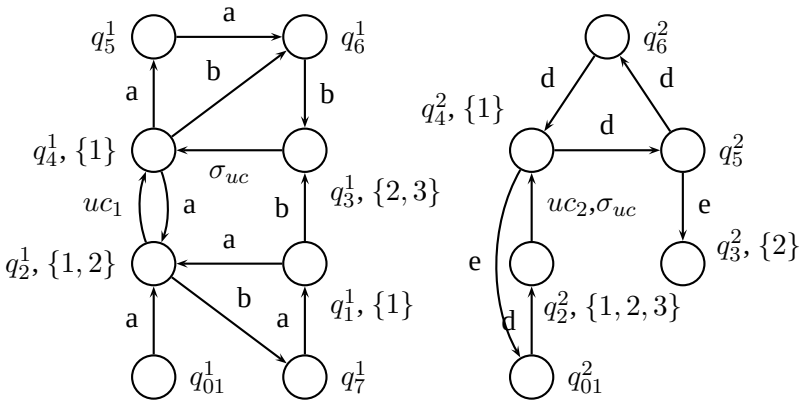
$$p_{\{1\}}(\mathcal{I}(E)) = p_{\{1\}}(\bigcup_{1\leq j\leq 3} \mathcal{I}_{1,loc}(E_1^j) \times \mathcal{I}_{2,loc}(E_2^j) \times \mathcal{I}_{3,loc}(E_3^j))$$

d'où $p_{\{1\}}(\mathcal{I}(E))\} = \{q_1^1, q_2^1, q_3^1, q_4^1\}$. On rappelle de plus que la projection du système contrôlé S_E/G suivant l'ensemble $\{1\}$, et noté $p_{\{1\}}(S_E/G)$, est donné par $p_{\{1\}}(S_E)/p_{\{1\}}(G)$. Le but

du point (a) de l'étape 1 est de calculer l'ensemble D'_1 des états qui sont en deadlock dans $p_{\{1\}}(S_E/G)$ et qui appartiennent à $\text{PB}_1(G_1)$. Ici, $\text{PB}_1(G_1) = \{q_3^1\}$ et on en déduit que

$$D'_1 = \{q_3^1\}$$

On considère à présent le point (b) de l'étape 1. Pour cela, on calcule $p_{\{1,2\}}(S_E/G)$ afin d'y rechercher les états qui y sont en deadlock, appartiennent à $p_{\{1,2\}}(\text{PB}(G))$ et dont la composante suivant $\{1\}$ appartient à D'_1. Le système projeté $p_{\{1,2\}}(G)$ est donné par



De plus, $p_{\{1,2\}}(\mathcal{I}(E)) = p_{\{1,2\}}(\mathcal{I}_{loc}(E))$ et vaut

$$\begin{aligned} p_{\{1,2\}}(\mathcal{I}_{loc}(E)) \quad = \quad & \{q_1^1, q_2^1, q_4^1\} \times \{q_2^2, q_4^2\} \\ & \cup \{q_2^1, q_3^1\} \times \{q_2^2, q_3^2\} \\ & \cup \{q_3^1\} \times \{q_2^2\} \end{aligned}$$

Notons que $p_{\{1,2\}}(\text{PB}(G)) = \text{PB}_1(G_1) \times \text{PB}_2(G_2) = \{q_3^1\} \times \{q_3^2\}$. Par conséquent, l'ensemble D'_2 des états qui sont en deadlock dans $p_{\{1,2\}}(S_E/G)$, appartiennent à $p_{\{1,2\}}(\text{PB}(G))$ et dont la composante suivant $\{1\}$ appartient à D'_1 vaut

$$D'_2 = \{(q_3^1, q_3^2)\}$$

Finalement, pour conclure l'étape 1, on détermine l'ensemble D'_3 des états en deadlock dans S_E/G qui appartiennent à $\text{PB}(G)$ et dont les composantes suivant $\{1, 2\}$ appartiennent à D'_2. Ici, $\text{PB}(G) = \{(q_3^1, q_3^2, q_2^3)\}$ et on en déduit que

$$D'_3 = \{(q_3^1, q_3^2, q_2^3)\}$$

Remarque 10 *Deux remarques peuvent être faites sur cet exemple.*

- *Ici $\text{PB}(G)$ ne possède qu'un seul état, par conséquent le calcul incrémental servant à déterminer D'_3 peut paraître superflu. Toutefois, cette approche peut s'avérer utile lorsque $\text{PB}(G)$ représente un nombre important d'états.*

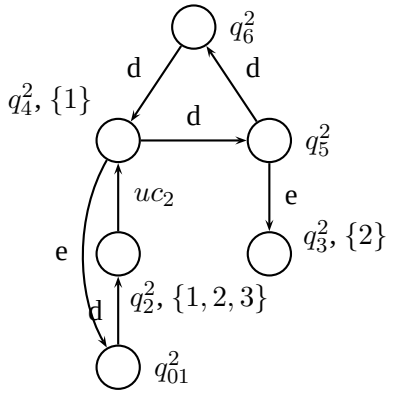
– *De plus,* (q_3^1, q_3^2, q_2^3) *est en deadlock dans* S_E/G *mais appartient en fait à* $\mathcal{I}(E)$*. Par conséquent, cet état n'est en fait pas accessible dans* S_E/G *et ne nécessite pas de traitement particulier par la suite.*

Étape 2 **:**

On cherche à présent à détecter les états en deadlock par contrôle dans S_E/G par inhibition d'au moins un événement local. Pour cela, on détermine les ensembles $D_{1,1}$, $D_{2,2}$ et $D_{3,3}$ correspondant aux états en deadlock par contrôle dans S_E/G par inhibition d'un événement local appartenant respectivement à Σ_1, Σ_2, Σ_3.

Calcul de $D_{1,1}$ **:**

Ce calcul s'effectue de manière incrémentale. Le point (a) de l'étape 2 correspond à une phase d'initialisation. En se référant à ce point, on considère $p_{\{2\}}(G)$ et $p_{\{2\}}(S_E)$.



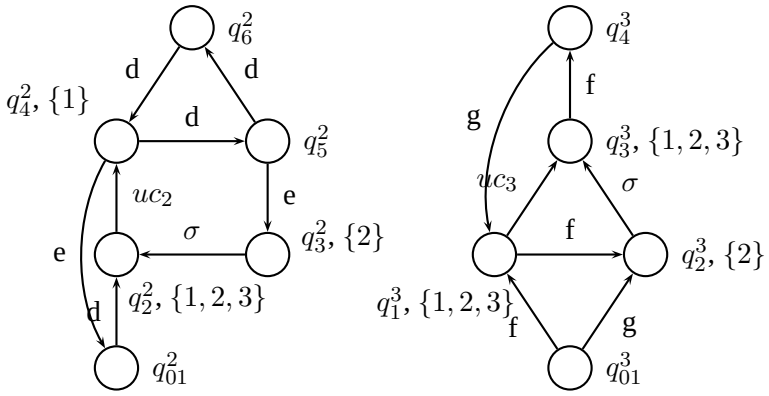
$p_{\{2\}}(S_E)$ est décrit à partir de $p_{\{2\}}(\mathcal{I}(E))$ qui vaut $p_{\{2\}}(\mathcal{I}_{loc}(E))$.

$$\begin{aligned} p_{\{2\}}(\mathcal{I}_{loc}(E)) &= \bigcup_{1\leq j\leq 3} \mathcal{I}_{2,loc}(E_2^j) \\ &= \{q_2^2, q_4^2, q_3^2\} \end{aligned}$$

On en déduit $p_{\{2\}}(S_E/G)$ $(= p_{\{2\}}(S_E)/p_{\{2\}}(G))$ et on cherche à déterminer l'ensemble $D_{2,1}$. Cet ensemble représente les états en deadlock dans $p_{\{2\}}(S_E/G)$ qui appartiennent à $p_{\{2\}}(E)$. Or $p_{\{2\}}(E) = \{q_2^2, q_4^2, q_3^2\}$. On obtient alors que

$$D_{2,1} = \{q_3^2\}$$

On itère le procédé et on s'intéresse au point (b) de l'étape 2. On considère ainsi les projections $p_{\{2,3\}}(G)$ et $p_{\{2,3\}}(S_E)$. $p_{\{2,3\}}(G)$ est donné par



De plus, $p_{\{2,3\}}(\mathcal{I}(E)) = p_{\{2,3\}}(\mathcal{I}_{loc}(E))$ et vaut

$$\begin{aligned} p_{\{2,3\}}(\mathcal{I}_{loc}(E)) &= \{q_2^2, q_4^2\} \times \{q_1^3, q_3^3\} \\ &\quad \cup \{q_2^2, q_3^2\} \times \{q_1^3, q_2^3, q_3^3\} \\ &\quad \cup \{q_2^2\} \times \{q_1^3, q_3^3\} \end{aligned}$$

On en déduit $p_{\{2,3\}}(S_E/G)$ $(= p_{\{2,3\}}(S_E)/p_{\{2,3\}}(G))$ et on cherche à déterminer l'ensemble $D_{3,1}$. Cet ensemble représente les états en deadlock dans $p_{\{2,3\}}(S_E/G)$ qui appartiennent à $p_{\{2,3\}}(E)$ et dont la composante suivant $\{2\}$ appartient à $D_{2,1}$. Or

$$\begin{aligned} p_{\{2,3\}}(E) &= \{q_2^2\} \times \{q_3^3\} \\ &\quad \cup \{q_2^2, q_3^2\} \times \{q_2^3, q_3^3\} \\ &\quad \cup \{q_2^2\} \times \{q_1^3, q_3^3\} \end{aligned}$$

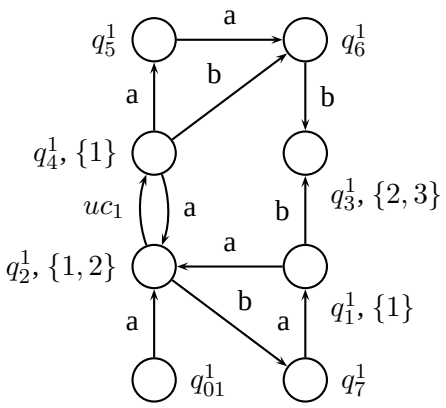
Il suffit donc de chercher les états en deadlock parmi $\{q_3^2\} \times \{q_2^3, q_3^3\}$. Seul (q_3^2, q_2^3) est en deadlock dans $p_{\{2,3\}}(S_E/G)$ et on obtient alors que

$$D_{3,1} = \{(q_3^2, q_2^3)\}$$

Finalement, on s'intéresse au point (c) de l'étape 2 et on cherche l'ensemble $D_{1,1}$ des états en deadlock dans S_E/G dont les composantes suivant $\{2,3\}$ appartiennent à $D_{3,1}$. Or aucun état de S_E/G ne vérifie cette propriété et donc $D_{1,1} = \emptyset$.

Calcul de $D_{2,2}$:

Ce calcul s'effectue de manière similaire à celui de $D_{1,1}$. Le point (a) de l'étape 2 correspond à une phase d'initialisation. En se référant à ce point, on considère $p_{\{1\}}(G)$ et $p_{\{1\}}(S_E)$.

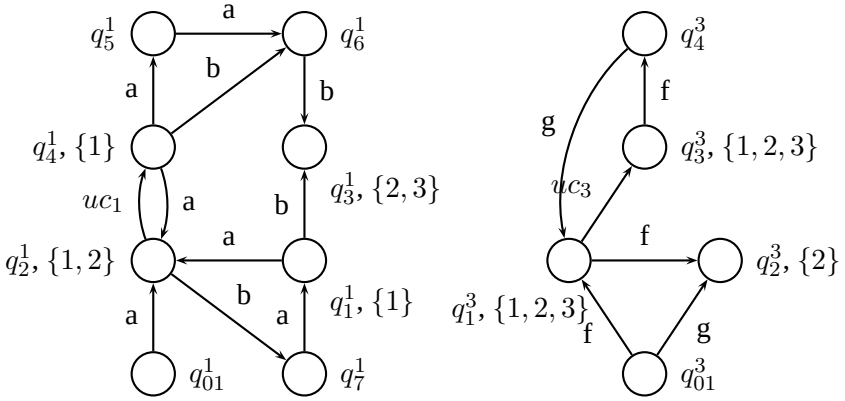


$p_{\{1\}}(S_E)$ est décrit à partir de $p_{\{1\}}(\mathcal{I}(E))$ qui vaut $p_{\{1\}}(\mathcal{I}_{loc}(E)) = \{q_1^1, q_2^1, q_3^1, q_4^1\}$.

On en déduit $p_{\{1\}}(S_E/G)$ $(= p_{\{1\}}(S_E)/p_{\{1\}}(G))$ et on cherche à déterminer l'ensemble $D_{1,2}$. Cet ensemble représente les états en deadlock dans $p_{\{1\}}(S_E/G)$ qui appartiennent à $p_{\{1\}}(E)$. Or $p_{\{1\}}(E) = \{q_1^1, q_2^1, q_3^1, q_4^1\}$. On obtient alors que

$$D_{1,2} = \{q_1^1, q_1^3\}$$

On itère le procédé et on s'intéresse au point (b) de l'étape 2. On considère ainsi les projections $p_{\{1,3\}}(G)$ et $p_{\{1,3\}}(S_E)$. $p_{\{1,3\}}(G)$ est donné par



De plus, $p_{\{1,3\}}(\mathcal{I}(E)) = p_{\{1,3\}}(\mathcal{I}_{loc}(E))$ et vaut

$$\begin{aligned} p_{\{1,3\}}(\mathcal{I}_{loc}(E)) \quad = \quad & \{q_1^1, q_2^1, q_4^1\} \times \{q_1^3, q_3^3\} \\ & \cup \{q_2^1, q_3^1\} \times \{q_1^3, q_2^3, q_3^3\} \\ & \cup \{q_3^1\} \times \{q_1^3, q_3^3\} \end{aligned}$$

On en déduit $p_{\{1,3\}}(S_E/G)$ $(= p_{\{1,3\}}(S_E)/p_{\{1,3\}}(G))$ et on cherche à déterminer l'ensemble $D_{3,2}$. Cet ensemble représente les états en deadlock dans $p_{\{1,3\}}(S_E/G)$ qui appartiennent à $p_{\{1,3\}}(E)$ et dont la composante suivant $\{1,3\}$ appartient à $D_{1,2}$. Or

$$\begin{aligned} p_{\{1,3\}}(E) &= \{q_1^1, q_4^1\} \times \{q_3^3\} \\ &\cup \{q_2^1, q_3^1\} \times \{q_2^3, q_3^3\} \\ &\cup \{q_3^1\} \times \{q_1^3, q_3^3\} \end{aligned}$$

Or aucun état ne vérifie ces propriétés et on obtient donc que $D_{3,2} = \emptyset$, et par conséquent que

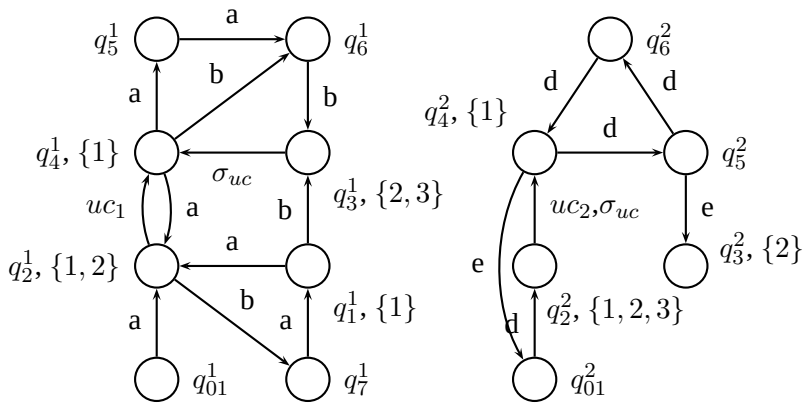
$$D_{2,2} = \emptyset$$

Calcul de $D_{3,3}$:

Ce calcul s'effectue de manière similaire à celui de $D_{1,1}$ et $D_{2,2}$. Le point (a) de l'étape 2 correspond à une phase d'initialisation. En se référant à ce point, on considère $p_{\{1\}}(G)$ et $p_{\{1\}}(S_E)$ qui ont été explicités pour le calcul de $D_{2,2}$. L'ensemble $D_{1,3}$ représente l'ensemble des états en deadlock dans $p_{\{1\}}(S_E/G)$ qui appartiennent à $p_{\{1\}}(E)$. Par conséquent, cet ensemble est identique à $D_{1,2}$ qui a été précédemment calculé. Par conséquent,

$$D_{1,3} = \{q_1^1, q_3^1\}$$

On s'intéresse au point (b) de l'étape 2. On considère ainsi les projections $p_{\{1,2\}}(G)$ et $p_{\{1,2\}}(S_E)$. $p_{\{1,2\}}(G)$ est donné par



De plus, $p_{\{1,2\}}(\mathcal{I}(E)) = p_{\{1,2\}}(\mathcal{I}_{loc}(E))$ et vaut

$$\begin{aligned} p_{\{1,2\}}(\mathcal{I}_{loc}(E)) &= \{q_1^1, q_2^1, q_4^1\} \times \{q_2^2, q_4^2\} \\ &\cup \{q_2^1, q_3^1\} \times \{q_2^2, q_3^2\} \\ &\cup \{q_3^1\} \times \{q_2^2\} \end{aligned}$$

On en déduit $p_{\{1,2\}}(S_E/G)$ $(= p_{\{1,2\}}(S_E)/p_{\{1,2\}}(G))$ et on cherche à déterminer l'ensemble $D_{2,3}$. Cet ensemble représente les états en deadlock dans $p_{\{1,2\}}(S_E/G)$ qui appartiennent à $p_{\{1,2\}}(E)$ et dont la composante suivant $\{1, 2\}$ appartient à $D_{1,3}$. Or

$$\begin{aligned} p_{\{1,2\}}(E) &= \{q_1^1, q_4^1\} \times \{q_4^2\} \\ &\quad \cup \{q_2^1, q_3^1\} \times \{q_2^2, q_3^2\} \\ &\quad \cup \{q_3^1\} \times \{q_2^2\} \end{aligned}$$

Il suffit donc de chercher les états en deadlock parmi

$$\{q_1^1\} \times \{q_4^2\} \cup \{q_3^1\} \times \{q_2^2, q_3^2\}$$

Seul (q_3^1, q_3^2) est en deadlock dans $p_{\{1,2\}}(S_E/G)$ et on obtient alors que

$$D_{2,3} = \{(q_3^1, q_3^2)\}$$

Finalement, on s'intéresse au point (c) de l'étape 2 et on cherche l'ensemble $D_{3,3}$ des états en deadlock dans S_E/G dont les composantes suivant $\{1, 2\}$ appartiennent à $D_{2,3}$. On obtient alors

$$D_{3,3} = \{(q_3^1, q_3^2, q_{01}^3), (q_3^1, q_3^2, q_2^3), (q_3^1, q_3^2, q_4^3)\}$$

L'ensemble des états en deadlock par contrôle par inhibition d'au moins un événement local, donné par l'étape 2, est noté D'' et vaut $\bigcup_i D_{i,i}$. Par conséquent,

$$D'' = \{(q_3^1, q_3^2, q_{01}^3), (q_3^1, q_3^2, q_2^3), (q_3^1, q_3^2, q_4^3)\}$$

Ensemble des états en deadlock dans S_E/G

L'ensemble D des états en deadlock dans S_E/G est constitué des états appartenant à $D'_3 = \{(q_3^1, q_3^2, q_2^3)\}$ ou appartenant à $D'' = \{(q_3^1, q_3^2, q_{01}^3), (q_3^1, q_3^2, q_2^3), (q_3^1, q_3^2, q_4^3)\}$. Par conséquent,

$$D = \{(q_3^1, q_3^2, q_{01}^3), (q_3^1, q_3^2, q_2^3), (q_3^1, q_3^2, q_4^3)\}$$

On cherche ici à déterminer un superviseur S assurant à la fois l'interdiction de l'ensemble d'états E, et qu'aucun état atteignable de S/G ne soit en deadlock. De plus, on souhaite que S soit le plus permissif possible. Pour cela, S est supposé être un superviseur standard interdisant un ensemble d'états contenant E mais aussi tout état en deadlock dans le système contrôlé S/G. C'est pourquoi, on s'intéresse ici au superviseur assurant l'interdiction de $E \cup D$. Si aucun état n'est en deadlock dans le système contrôlé correspondant, alors ce superviseur est solution de notre problème. Sinon, il faut réitérer la démarche précédente pour détecter les états en deadlock dans ce système contrôlé. Le nouvel ensemble d'états E à interdire vaut $E = E^1 \cup E^2 \cup E^3 \cup E^4 \cup E^5$ avec $E^4 = \{q_3^1\} \times \{q_3^2\} \times \{q_{01}^3\}$ et $E^5 = \{q_3^1\} \times \{q_3^2\} \times \{q_4^3\}$. E^4 et E^5 représentent les états de D qui n'étaient pas initialement interdits.

On peut enfin noter que $E^4 \cup E^5$ s'exprime naturellement comme un unique pavé $\{q_3^1\} \times \{q_3^2\} \times \{q_{01}^3, q_4^3\}$.

3.4 Extension au cas des machines hiérarchiques

Dans le chapitre 2 et la première partie de ce chapitre, nous nous sommes intéressés au contrôle de systèmes concurrents. En effet dans la plupart des problèmes de contrôle, les systèmes à événements discrets sont constitués d'un grand nombre de composants qui interagissent de manière concurrente. À cette composition parallèle, on peut rajouter un autre type de composition : la hiérarchie. Il est en effet fréquent de rencontrer des systèmes à la fois spécifiés à partir de sous-systèmes agissant en parallèle, et spécifiés par "couche", en donnant des représentations du système à différents niveaux d'abstraction.

Dans [8], les auteurs s'intéressent au problème de l'interdiction d'états sur un modèle hiérarchique appelé *Asynchronous Hierarchical State Machine*, dont les sous systèmes concurrents évoluent de manière asynchrone (sans partager d'événements). Dans [44] et [43], les auteurs considèrent un modèle hiérarchique à deux niveaux où le haut et le bas niveau interagissent à travers une interface. Plus récemment, dans [50], l'auteur s'intéresse à une modélisation hiérarchique des systèmes, dont l'implémentation par des *diagrammes de décision binaire* ([10]) utilise la structure du système pour optimiser les calculs.

Le but de cette section est de reprendre et étendre les travaux effectués dans [52] et [23], en prenant notamment en compte les synchronisations entre les sous systèmes concurrents, à l'aide des résultats des sections précédentes. Ainsi, le système considéré est spécifié hiérarchiquement par un modèle simplifié des STATECHARTS [30], les machines à états finis hiérarchiques (abrégé HSFM pour Hierarchical Finite State Machines). Le modèle que nous considérons permet de représenter des systèmes possédant deux niveaux de hiérarchie. Il peut être caractérisé par une structure de haut niveau K et une collection d'automates $(G_1, \ldots, G_n)$ permettant de modéliser le bas niveau du système. Au haut niveau K de la hiérarchie, les états sont soit des *états ordinaires* soit des *macro-états* b. Les macro-états permettent de modéliser la composition parallèle d'un ensemble d'automates (G_j). Intuitivement, le comportement d'une HFSM H est le suivant : lorsque le système évolue au haut niveau et que l'état de K correspondant n'est pas un macro-état, les comportements de la HFSM sont ceux de K. Et lorsqu'un macro-état b de K est atteint, le comportement de H devient celui de la composition parallèle des automates impliqués dans le macro-état b. La sortie d'un macro-état est synchronisée avec la fin des tâches associées à cette composition parallèle (i.e. chaque automate est dans un état final).

3.4.1 Le modèle hiérarchique

Précédemment, nous avons donné des résultats concernant le contrôle de systèmes concurrents. Nous allons maintenant étendre ces résultats au cas de Machines à États Finis Hiérarchiques (HFSM : Hierarchical Finite State Machines). Nous donnons ici les résultats pour des machines hiérarchiques à deux niveaux (une description plus générale, dans le cadre asynchrone, des HFSMs peut se trouver dans [52]).

Intuitivement, une HFSM est un automate dans lequel certains des états peuvent être raffinés en d'autres automates, induisant ainsi une hiérarchie. De plus, certains de ces automates peuvent apparaître à différents endroits et dans différents contextes. Pour illustrer cet aspect, prenons l'exemple d'une cellule flexible de production manufacturière. Les automates du niveau inférieur correspondent alors aux différentes machines, alors que l'automate de haut niveau permet d'agen-

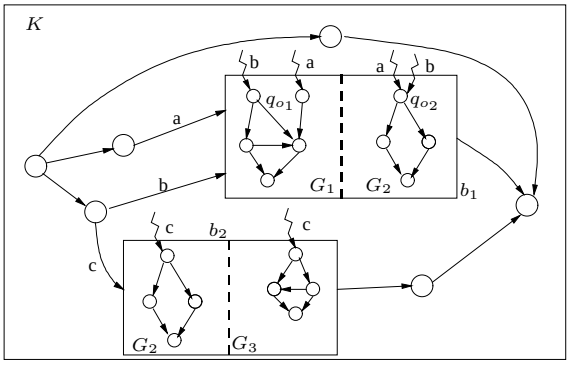


FIG. 3.12: Un exemple de HFSM

cer les machines pouvant être actives simultanément (e.g. G_1 et G_3, où G_2 et G_3 dans l'exemple de la figure 3.12). Le rôle d'un superviseur sera alors de coordonner les comportements de ces différentes machines de manière à éviter des comportements globaux dangereux.
Avant de donner la définition formelle d'une HFSM, nous avons besoin de définir la notion de structure modélisant le comportement haut niveau d'une HFSM.

Définition 26 *Une structure K est un tuple $\langle \Sigma, Q, \mathcal{B}, q_o, Q_m, \delta \rangle$, où Q est l'ensemble des états atomiques, $q_o \in Q$ est l'état initial et $Q_m \subseteq Q$ les états finals. $\mathcal{B}$ est l'ensemble des macro-états de K. δ est la fonction de transition partielle définie sur $\Sigma \times \{Q \cup \mathcal{B}\} \rightarrow \{Q \cup \mathcal{B}\}$.*

Par la suite K^A désignera l'automate $\langle \Sigma, Q \cup \mathcal{B}, q_o, Q_m, \delta \rangle$. K^A correspond à la structure K, considérant que tous ses états sont atomiques. Pour simplifier, on supposera dans la suite que K^A est déterministe et *Trim*[7]. Basé sur cette notion de structure, nous définissons maintenant les machines à états finis hiérarchiques.

Définition 27 *Une HFSM $\mathcal{K}$ est donnée par un tuple $(\langle K, G_1, \ldots, G_n \rangle, Y, I)$, où K est une structure (C.f. Def. 26) et $\forall 1 \leq i \leq n, G_i = \langle \Sigma_i, Q_i, Q_{o_i}, q_{m_i}, \delta_i \rangle$ est un automate déterministe et Trim. Y, I sont deux fonctions qui caractérisent la hiérarchie et la composition entre les automates.*

- *$Y : \mathcal{B} \longrightarrow 2^{\langle G_1, \ldots, G_n \rangle}$ est une fonction qui associe à un macro-état $b \in \mathcal{B}$, l'ensemble des automates G_i activés en b. On note par J_b l'ensemble des indices $\{j \leq n | \ G_j \in Y(b)\}$.*
- *I est une fonction telle que $\forall b \in \mathcal{B}$, $I(b)$ est une fonction définie sur $\Pi_{j \in J_b} Q_{o_j} \rightarrow 2^{\Sigma}$. Étant donnés un macro-état b et $q_o \in \Pi_{j \in J_b} Q_{o_j}$ un tuple d'états initiaux, $I(b)(q_o)$ est l'ensemble des événements admissibles qui font évoluer le système de son état courant dans q_o.*

K sera appelée *structure de haut niveau* de $\mathcal{K}$. Un exemple de HFSM est donné en Figure 3.12 (par exemple $Y(b_1) = \{G_1, G_2\}$ et $I(b_1)(\langle q_{o_1}, q_{o_2} \rangle) = \{b\}$).

Remarque 11 *Deux remarques peuvent être faites concernant les états initiaux et finaux des automates $G_1, \ldots, G_n$.*

- *On suppose ici que les automates impliqués dans les macro-états possèdent plusieurs états initiaux. Ceci permet de prendre en compte la dernière action effectuée au haut niveau*

[7] Voir section 1.1.2.2 pour davantage de détails.

pour définir une initialisation différente du système considéré lorsqu'un macro-état est atteint pendant l'exécution. Il aurait en effet été possible de considérer que des automates ne possédant qu'un seul état initial (comme c'est classiquement le cas). Toutefois, ceci impliquerait alors une duplication du nombre de macro-états, rendant moins compacte la représentation du système.

– *On suppose de plus que les automates impliqués dans les macro-états ne possèdent qu'un seul état final. Ceci constitue uniquement une simplification syntaxique dans la suite. En effet, la sémantique qui sera donnée à une HFSM ne fera pas intervenir les états finaux des automates impliqués dans un macro-état. De plus, les questions d'atteignabilité ne sont pas abordées ici.*

Hypothèses. On supposera dans la suite que les HFSM vérifient certaines propriétés :

(1) $\forall i$, t.q. $\Sigma \cap \Sigma_i = \emptyset$.

(2) Soit $b \in \mathcal{B}$. Soit $(G_j)_{j\in J_b} = Y(b)$ l'ensemble des automates impliqués dans le macro-état b. On suppose que

$$\delta^{-1}(b) \subseteq \bigcup_{q_o \in \Pi_{j\in J_b}(Q_{o_j})} (I(b)(q_o)) \text{ et } \forall q_o, q'_o \in \Pi_{j\in J_b}(Q_{o_j}), q_0 \neq q'_o \Rightarrow I(b)(q_o) \cap I(b)(q'_o) = \emptyset$$

L'hypothèse (1) permet notamment d'assurer que le modèle est déterministe. De plus, l'hypothèse (2) traduit que l'entrée dans un macro-état est déterministe et chaque événement menant à un macro-état b est pris en compte.

Le comportement de $\mathcal{K}$. Soit $\mathcal{K} = (\langle K, G_1, \ldots, G_n\rangle, Y, I)$ une HFSM. $\mathcal{K}$ est initialisée dans son état initial et tant qu'aucun macro-état de K n'est atteint, $\mathcal{K}$ se comporte comme l'automate K^A. Soit $b \in \mathcal{B}$ un macro-état de $\mathcal{K}$. Soit $q \in Q \cup \mathcal{B}$, t.q. $\delta(\sigma, q) = b$. On suppose maintenant que $\mathcal{K}$ se trouve dans l'état q. Lorsque σ est tiré, $\mathcal{K}$ évolue vers la configuration $q_o = \langle q_{o_{j_1}}, \ldots, q_{o_{j_{\|J_b\|}}}\rangle$, telle que $\sigma \in I(b)(q_o)$ et se comporte par la suite comme le système concurrent $\|_{j\in J_b}\ G_i$ initialisé en $q_o = \langle q_{o_{j_1}}, \ldots, q_{o_{j_{\|j_b\|}}}\rangle$. Pour "quitter" le macro-état b, chaque automate de $Y(b)$ doit avoir évolué dans son propre état final. À cet instant, les événements de $\delta(b)$ sont admissibles. En d'autres termes, il n'y a pas de préemption et la sortie d'un macro-état est synchronisée avec la fin des tâches associées aux différents automates de ce macro-état.

Notons que les comportements d'une HFSM $\mathcal{K} = (\langle K, G_1, \ldots, G_n\rangle, Y, I)$ peuvent être décrits par un automate, obtenu en substituant chaque macro-état b par l'automate correspondant au produit parallèle entre les automates donnés par $Y(b)$. Ainsi à chaque macro-état $b \in \mathcal{B}$, on associe son automate correspondant

$$K_b = \langle \Sigma_b, Q_b, Q_{o_b}, x_{f_b}, \delta_b\rangle = \|_{j\in J_b} G_j.$$

Finalement pour obtenir l'automate global à partir de K, chaque macro-état b de $\mathcal{B}$ est remplacé par son automate correspondant K_b (les états seront notés $[b, \langle q_1, \ldots, q_{\|J_b\|}\rangle]$), la connection entre les états initiaux de K_b et les états de K est alors réalisée en fonction de I (resp. pour l'état final). Par la suite, on notera $\mathcal{K}^F$ cet automate, dit automate associé à $\mathcal{K}$.

Définition 28 *L'automate associé à la HFSM $\mathcal{K}$ est noté $\mathcal{K}^F = \langle \Sigma^F, Q^F, q_o^F, Q_m^F, \delta^F \rangle$, où chaque composant est défini par :*

- $\Sigma^F = \Sigma \cup \bigcup_{b \in \mathcal{B}} \{\Sigma_b\}$, $q_o^F = q_o$, $Q_m^F = Q_m$ *et*
- $Q^F = Q \cup \{[b, \langle q_1, \ldots, q_{\|J_b\|} \rangle] \mid \forall b \in \mathcal{B}, \forall \langle q_1, \ldots, q_{\|J_b\|} \rangle \in Q_b\}$
- *La fonction partielle de transition δ^F est définie $\forall q, q' \in Q,\ \forall b, b' \in \mathcal{B},\ \forall \sigma \in \Sigma^F$ par :*
  - $\delta^F(\sigma, q) = q'$ *si* $\delta(q, \sigma)!$ *et* $\delta(\sigma, q) = q'$
  - $\delta^F(\sigma, q) = [b, q_{o_b}]$ *si* $\delta(q, \sigma)!$ *et* $\delta(\sigma, q) = b$ *et* $\sigma \in I(b)(q_{o_b})$.
  - $\delta^F(\sigma, [b, q_{f_b}]) = q$ *si* $\delta(q, b)!$ *et* $\delta(\sigma, b) = q$, *où* q_{f_b} *est l'état final de* G_b.
  - $\delta^F(\sigma, [b, q_{f_b}]) = [b', q_{o_{b'}}]$ *si* $\delta(q, b)!$ *et* $\delta(\sigma, b) = b'$, $\sigma \in I(b')(q_{o_{b'}})$ *et* q_{f_b} *est l'état final de* G_b.
  - $\delta^F(\sigma, [b, q_b]) = [b, \delta_b(\sigma, q_b)]$ *si* $\delta_b(\sigma, q_b)!$.
  - *Indéfinie sinon*

Dans cette section, on souhaite étendre les techniques introduites dans la section précédente au cadre des HFSM. Les systèmes considérés sont alors plus complexes que les systèmes concurrents, puisqu'une structure hiérarchique est ici ajoutée au modèle. Avant de donner des résultats sur la synthèse de contrôleurs sur des HFSM, on introduit des hypothèses sur la nature des événements.

Contrôlabilité des événements d'une HFSM. Une HFSM est composée de plusieurs sous systèmes modélisés par une structure K et des automates $\{G_i\}_{1 \leq i \leq n}$. Σ_c et Σ_{uc} représentent respectivement les ensembles d'événements contrôlables et incontrôlables de la structure K. Les alphabets $(\Sigma_i)_{1 \leq i \leq n}$ de chacun des automates $(G_i)_{1 \leq i \leq n}$ sont partitionnés en $\Sigma_i = \Sigma_{i,uc} \cup \Sigma_{i,c}$ où $\Sigma_{i,uc}$ représente l'ensemble des événements incontrôlables de G_i et $\Sigma_{i,c}$ l'ensemble de ses événements contrôlables. Comme dans les chapitres précédents, on suppose que la nature d'un événement est indépendante des sous systèmes auxquels il appartient :

$$\forall i \neq j,\ \Sigma_{i,c} \cap \Sigma_{j,uc} = \emptyset$$

Sous cette hypothèse, on définit alors les ensembles d'événements contrôlables (Σ_c^F) et incontrôlables (Σ_{uc}^F) de l'automate $\mathcal{K}^F$ (ou de la HFSM $\mathcal{K}$) par

$$\Sigma_c^F = \left(\bigcup_{1 \leq i \leq n} \Sigma_{i,c} \right) \cup \Sigma_{uc}, \quad \Sigma_{uc}^F = \left(\bigcup_{1 \leq i \leq n} \Sigma_{i,uc} \right) \cup \Sigma_c$$

3.4.2 Problème de l'interdiction d'états

Nous considérons ici le problème du contrôle d'une HFSM $\mathcal{K}$. Notre but est de calculer un superviseur interdisant un ensemble d'états de $\mathcal{K}$. Lorsque le système que l'on souhaite contrôler est concurrent, il est intéressant que l'ensemble des états à interdire soit donné par une union de pavés. Dans le cadre de HFSM, on s'intéresse aussi à des ensembles d'états possédant une forme particulière. On souhaite de plus que cet ensemble soit suffisamment général pour représenter un ensemble quelconque d'états de $\mathcal{K}^F$.

Configurations interdites. Une HFSM $\mathcal{K}$ est constituée de deux niveaux de hiérarchie. Le haut niveau est modélisé par un automate, alors que le bas niveau est modélisé par des compositions parallèles d'automates. Ainsi, on souhaite que les ensembles d'états de bas niveau de $\mathcal{K}$ soient représentés par des unions de pavés. Plus précisément, avec les notations de la définition 27, soit une HFSM $\mathcal{K} = (\langle K, G_1, \ldots, G_n\rangle, Y, I)$ telle que $K = (\Sigma, Q, \mathcal{B}, q_o, Q_m, \delta)$. Étant donné $b \in \mathcal{B}$, on note

$$E^b = \bigcup_{1 \leq j \leq m_b} E^{b,j}$$

avec pour $1 \leq j \leq m$,

$$E^{b,j} = E^{b,j}_{j_1} \times \cdots \times E^{b,j}_{j_{\|J_b\|}}$$

et $E^{b,j}_{j_i} \subseteq Q_{j_i}$ pour $j_i \in J_b$. Par simplicité, on note $[b, E^b]$ l'ensemble des configurations $[b, \langle q_{j_1}, \ldots, q_{j_{\|J_b\|}}\rangle]$ telles que $\langle q_{j_1}, \ldots, q_{j_{\|J_b\|}}\rangle \in E^b$. N'importe quel ensemble d'états de $\mathcal{K}^F$ peut être représenté par un ensemble de configurations E de $\mathcal{K}$ de la forme

$$E = E_0 \bigcup \left(\bigcup_{b \in \mathcal{B}} [b, E^b] \right) \tag{3.22}$$

où $E_0 \subseteq Q$. L'ensemble E_0 représente les configurations interdites au niveau supérieur de $\mathcal{K}$, alors que $[b, E^b]$ représente l'ensemble des configurations interdites dans le bas niveau de la HFSM (i.e. à l'intérieur du macro-état b).

Lorsque E représente l'ensemble des états à interdire pour $\mathcal{K}^F$, on note (comme dans les sections précédentes) $\mathcal{I}(E)$ l'ensemble des *états interdits* de $\mathcal{K}^F$ qui correspond à l'ensemble des états permettant d'atteindre E par tirage d'une séquence d'événements incontrôlables.

$$\mathcal{I}(E) = CoReach^{\mathcal{K}^F}_{\Sigma^F_{uc}}(E) \tag{3.23}$$

De plus, le superviseur S défini pour tout $q \in Q^F$ par

$$S(q) = \{\sigma \in \Sigma_c |\ \delta^F(q, \sigma)! \wedge \delta^F(q, \sigma) \in \mathcal{I}(E)\}$$

assure l'interdiction de E dans $\mathcal{K}^F$ et est maximal.

Comme dans le cadre des systèmes concurrents, traités dans le chapitre précédent, la méthode de résolution du PBSC présentée ici a pour but de fournir une représentation pertinente de $\mathcal{I}(E)$ sans avoir à explicitement construire l'automate associé. C'est à dire que l'on souhaite que cette représentation puisse d'une part être efficacement calculée, et d'autre part que l'appartenance d'un état de $\mathcal{K}^F$ à cet ensemble puisse être efficacement déterminée.

Opérateurs hiérarchiques. Dans un premier temps, nous avons besoin d'étendre les définitions d'ensembles d'états faiblement interdits de manière à prendre en compte la hiérarchie. L'idée est de ne pas interdire un macro-état dans sa globalité lorsqu'il mène à un état interdit, mais de considérer le fait qu'il a un comportement interne et donc qu'en "descendant" dans la hiérarchie, le contrôle peut s'effectuer au niveau inférieur. *A contrario*, considérons $b \in \mathcal{B}$ un macro-état de K. Par contrôle sur les structures internes à b, des configurations initiales de b peuvent devenir des configurations interdites. On peut donc être amené à empêcher l'atteignabilité de b par les événements faisant évoluer le système dans ces configurations initiales et donc à interdire partiellement le macro-état b. Dans ce but, pour $A \subseteq \delta^{-1}(b)$, on introduit $b_{|A}$ le "macro-état contrôlé" considérant qu'il n'est atteignable que par un événement appartenant à A et $\mathcal{B}_{|\mathcal{A}} = \{b_{|A} \mid b \in \mathcal{B} \text{ et } A \subseteq \delta^{-1}(b)\}$ l'ensemble correspondant des macro-états contrôlés. L'idée du contrôle sera donc d'empêcher dans K l'atteignabilité de $b_{|A}$ via des événements appartenant à l'ensemble A. Ce type particulier d'objectif sera utilisé pour empêcher certains états initiaux des structures de $Y(b)$ d'être atteignables.

Basé sur ces remarques, nous étendons les définitions d'ensembles d'états faiblement interdits de manière à prendre en compte les macro-états d'une structure.

Définition 29 *Soit $K = (\Sigma, Q, \mathcal{B}, q_o, Q_f, \delta)$ la structure de haut niveau d'une HFSM $\mathcal{K}$ et $e \in Q \cup \mathcal{B}_{|\mathcal{A}}$.*

- *Si $e \in Q$, alors*

$$\begin{aligned} \mathcal{I}_Q(e) &= \{q \in Q \mid \exists s \in \Sigma_{uc}^*,\ \delta(s,q) = e \text{ et } \forall s' \leq s,\ \delta(s',q) \notin \mathcal{B}\}. \\ \mathcal{I}_B(e) &= \{b \in \mathcal{B} \mid \exists \sigma \in \Sigma_{uc},\ \delta(\sigma,b) \in \mathcal{I}_Q(e)\} \end{aligned}$$

- *Si $e = b_{|A} \in \mathcal{B}_{|\mathcal{A}}$, alors*

$$\begin{aligned} \mathcal{I}_Q(b_{|A}) &= \{q \in Q \mid \exists s \in \Sigma_{uc}^*,\ \exists \sigma \in \Sigma_{uc} \cap A, \delta(s\sigma,q) = b \text{ et } \forall s' \leq s,\ \delta(s',q) \notin \mathcal{B}\} \\ \mathcal{I}_B(b_{|A}) &= \{b' \in \mathcal{B} \mid \exists \sigma \in \Sigma_{uc},\ (\delta(\sigma,b') \in \mathcal{I}_Q(b_{|A})) \text{ ou } (\sigma \in \Sigma_{uc} \cap A \text{ et } \delta(\sigma,b') = b)\} \end{aligned}$$

Finalement, étant donné $E \subseteq Q \cup \mathcal{B}_{|\mathcal{A}}$, $\mathcal{I}_Q(E) = \cup_{e \in E} \mathcal{I}_Q(e)$ et $\mathcal{I}_B(E) = \cup_{e \in E} \mathcal{I}_B(e)$

Intuitivement, si $e \in Q$, $\mathcal{I}(e)$ (resp. $\mathcal{I}_B(e)$) représente l'ensemble des états atomiques (resp. macro-états) de K à partir desquels e peut être atteint via une trajectoire incontrôlable qui ne traverse que des états atomiques de K. Si $e = b_{|A}$, la signification de $\mathcal{I}(e)$ et de $\mathcal{I}_B(e)$ est identique, excepté que le dernier événement de la trajectoire doit appartenir à A.
L'opérateur Ψ que nous définissons maintenant sera utile pour calculer l'ensemble global des états faiblement interdits en montant/descendant dans la hiérarchie. En effet, l'interdiction d'un état initial dans un macro-état fait que l'on doit remonter au niveau supérieur pour interdire les états menant à cet état initial. Inversement, il est nécessaire d'interdire l'état final d'un macro-état menant de manière incontrôlable à un état interdit du niveau supérieur. La fonction Ψ (c.f. Définition 30) permet de déterminer l'ensemble des configurations de la HFSM à interdire. Avant d'introduire cette définition, l'exemple suivant illustre de manière intuitive cet aspect :

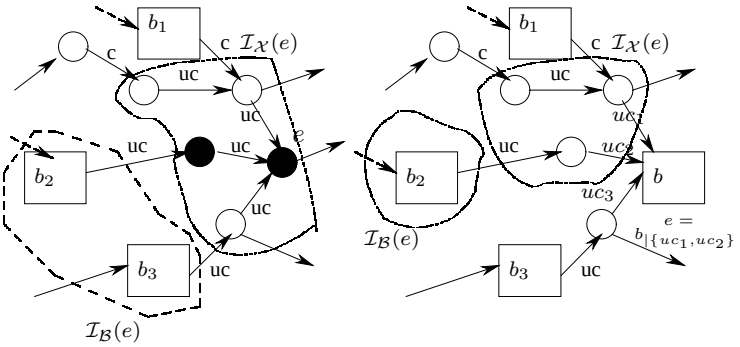
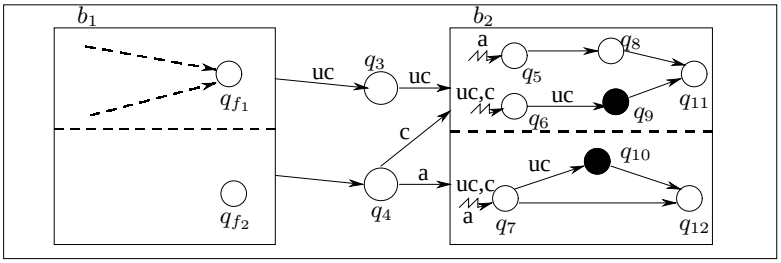


FIG. 3.13: Calcul de $\mathcal{I}(e)$,$\mathcal{I}_{\mathcal{B}}(e)$ sur un exemple

Exemple 20 *On suppose ici que l'on souhaite interdire la configuration* $[b_2, \langle q_9, q_{10}\rangle]$.



Les calculs locaux des états faiblement interdits nous amènent à considérer l'interdiction de $[b_2, \{q_6, q_9\} \times \{q_7, q_{10}\}]$. *On voit alors la nécessité d'empêcher le macro-état* b_2 *d'être atteignable suivant les événements qui mènent à l'état initial* $[b_2, (q_6, q_7)]$. *Compte tenu de la fonction* I, *on doit empêcher* b_2 *d'être atteint par tirage des événements* c *ou* uc. $b_{2|\{c,uc\}}$ *doit être interdit (Pt 2 Def. 30). De plus, compte tenu du caractère incontrôlable de* uc, *on a* $\mathcal{I}_Q(b_{2|\{c,uc\}}) = \{q_3\}$ *et* $\mathcal{I}_B(b_{2|\{c,uc\}}) = \{b_1\}$. *On en déduit que* q_3 *et* $[b_1, (q_{f_1}, q_{f_2})]$ *doivent être ajoutés à l'ensemble des configurations interdites (Pt 1 Def. 30). Par conséquent, étant donnée une configuration interdite, la fonction* Ψ *permet de déterminer les configurations ou éléments de* $\mathcal{B}_{|\mathcal{A}}$ *devant être interdits au niveau inférieur ou supérieur pour assurer l'objectif de contrôle à partir de calculs locaux.*

Définition 30 *Soit* $e \in Q^F \cup \mathcal{B}_{|\mathcal{A}}$. *Pour* $b \in \mathcal{B}$, *on note* Q_{ob} *(resp.* q_{f_b}*) l'ensemble des états initiaux (resp. l'état final) de la composition parallèle des automates donnés par* $Y(b)$ *(i.e.* $K_b = \|_{j\in J_b} G_i$*).*

1. *Si* $e \in Q \cup \mathcal{B}_{|\mathcal{A}}$, *alors* $\Psi(e) = \mathcal{I}_Q(e) \cup \{[b, q_{f_b}] \mid b \in \mathcal{I}_{\mathcal{B}}(e)\}$
2. *Si* $e = [b, (q_{j_1}, \ldots, q_{j_{\|J_b\|}})]$, *alors en notant* $\mathcal{I}^b = \mathcal{I}^b(\Phi^*(e))$, *où* $\mathcal{I}^b(\Phi^*(e))$ *correspond à l'ensemble des états faiblement interdits de* e *dans* $\|_{j\in J_b} G_i$ *calculé comme en Section 3.2.2.2,*

$$\Psi(e) = [b, \mathcal{I}^b] \cup b_{|\bigcup_{q_{o_b} \in \mathcal{I}^b \cap Q_{ob}} \{I(b)(q_{o_b})\}} \tag{3.24}$$

De plus, pour $E \subseteq Q^F \cup \mathcal{B}_{|\mathcal{A}}$, $\Psi(E) = \bigcup_{e \in E} \Psi(e)$.

Soit $e \in Q^F \cup \mathcal{B}_{|\mathcal{A}}$. Si $e \in Q \cup \mathcal{B}_{|\mathcal{A}}$, alors $\Psi(e)$ correspond à l'ensemble des configurations faiblement interdites issues de e auxquelles s'ajoutent les états finals des macro-états pouvant mener de manière incontrôlable à e. De cette manière, nous descendons dans la hiérarchie. Maintenant, si $e = [b, \langle q_{j_1}, \ldots, q_{j_{\|J_b\|}} \rangle]$, alors $\Psi(e)$ correspond à l'ensemble des configurations faiblement interdites de e auxquelles s'ajoutent éventuellement $b_{|A} \in \mathcal{B}_{|\mathcal{A}}$ tel que A représente l'ensemble des événements de $\delta^{-1}(b)$ menant à un état interdit de K_b. On peut remarquer que le calcul de $\Psi(e)$ nécessite de travailler avec la composition parallèle des automates impliqués dans le macro-états b. Il est alors possible de tirer profit des avantages algorithmiques offerts par les résultats du chapitre 3 sur les systèmes concurrents.

Calcul du superviseur.
Soient $\mathcal{K}$ une HFSM et $E = E_0 \bigcup \left(\bigcup_{b \in \mathcal{B}} [b, E^b] \right)$ un ensemble de configurations de $\mathcal{K}$ comme défini par l'équation 3.22. L'ensemble des événements faiblement interdits est alors calculé par l'intermédiaire du point fixe suivant :

$$\mathcal{E}_o = E \text{ et } \mathcal{E}_{i+1} = \Psi(\mathcal{E}_i) \tag{3.25}$$

Soit $\mathcal{I}_H(E)$ le résultat du calcul précédent. Compte tenu de la définition de Ψ et du fait que $\mathcal{K}$ possède un nombre fini de configurations, cette limite existe toujours et est obtenue en un nombre fini d'étapes. De plus les éléments retournés par Ψ appartiennent à Q ou à $\mathcal{B}_{|\mathcal{A}}$, ou sont des configurations de macro-états. Ainsi, il est possible de réorganiser l'ensemble $\mathcal{I}_H(E)$ de la manière suivante :

$$\mathcal{I}_H(E) = Q' \cup \mathcal{B}'_{|\mathcal{A}} \cup \bigcup_{b \in \mathcal{B}} [b, E'^b], \tag{3.26}$$

où $E'^b = \bigcup_i E'^{b,i}_{j_1} \times \cdots \times E'^{b,i}_{j_{\|J_b\|}}$ (i.e. une union de pavés comme décrit en Section 3), $Q' \subseteq Q$ et $\mathcal{B}'_{|\mathcal{A}} \subseteq \mathcal{B}_{|\mathcal{A}}$.

Pour les macro-états de $\mathcal{B}'_{|\mathcal{A}}$ cela signifie intuitivement qu'il est interdit d'entrer dans ces macro-états via les événements qui appartiennent à A. De plus, chaque macro-état b correspond à un système concurrent pour lequel l'ensemble des configurations E'^b doit être interdit. On peut remarquer que E'^b est donné par une union de pavés.

Le lien entre $\mathcal{I}_H(E)$ et les états de $\mathcal{K}^F$ menant à E de manière incontrôlable est donné par la proposition suivante.

Proposition 17 *Soit E un ensemble d'états d'une HFSM $\mathcal{K}$ à interdire. Alors,*

$$\mathcal{I}_H(E) \setminus \mathcal{B}_{|\mathcal{A}} = \mathcal{I}(E)$$

où $\mathcal{I}(E)$ est l'ensemble des états de $\mathcal{K}$ représentant les états interdits de E, dans l'automate associé $\mathcal{K}^F$ (calculé comme en definition 10).

En d'autres termes, l'ensemble $\mathcal{I}_H(E)$[8] correspond à l'ensemble des états interdits de $\mathcal{I}(E)$ du système "plat' $\mathcal{K}^F$. Toutefois, par rapport aux méthodes classiques, tous les calculs ont été réalisés localement sans avoir à construire explicitement le système.

Démonstration : Montrons dans un premier temps que $\mathcal{I}_H(E) \setminus \mathcal{B}_{|\mathcal{A}} \subseteq \mathcal{I}(E)$. La preuve procède par induction. À cet effet, montrons que $\forall i, \mathcal{E}_i \setminus \mathcal{B}_{|\mathcal{A}} \in I(E)$. À l'étape 0, il est évident que $E \subseteq \mathcal{I}(E)$. À l'étape i, soit $\mathcal{E}_i$ le résultat de (3.25) après i itérations. Supposons que $\mathcal{E}_k \setminus \mathcal{B}_{|\mathcal{A}} \subseteq \mathcal{I}(E)$, $\forall 0 \leq k \leq i$. Soit $\mathcal{E}_{i+1} = \Psi(\mathcal{E}_i)$. nous devons alors montrer que $\mathcal{E}_{i+1} \setminus \mathcal{B}_{|\mathcal{A}} \subseteq I(E)$. Pour cela, considérons $e' \in \mathcal{E}_{i+1} \setminus \mathcal{B}_{|\mathcal{A}}$ et $e \in \mathcal{E}_i$ tels que $e' \in \Psi(e)$.

– Si $e \in Q$, alors d'après la définition 30,

$$\Psi(e) = \mathcal{I}_Q(e) \cup \{[b, q_{f_b}] \mid b \in \mathcal{I}_{\mathcal{B}}(e)\}$$

Puisque $e' \in \Psi(e)$, deux cas de figures se présentent :

– soit $e' \in \mathcal{I}_Q(e)$ et dans ce cas, d'après la définition 29, il existe $s \in \Sigma_{uc}^*$ tel que $\delta(s, e') = e$ et pour tout $s' \leq s$, $\delta(s', e') \notin \mathcal{B}$. Par conséquent, d'après la définition 28, il existe $s \in \Sigma_{uc}^*$ tel que $\delta^F(s, e')!$ et $\delta^F(s, e') = e$. On déduit alors de la définition de $\mathcal{I}(E)$ (voir (3.23)) que $e' \in \mathcal{I}(e)$ et donc que $e' \in \mathcal{I}(E)$ (puisque $e \in \mathcal{I}(E)$ et $\mathcal{I}(\mathcal{I}(E)) = \mathcal{I}(E)$).

– soit $e' = [b, q_{fb}]$ avec $b \in \mathcal{I}_{\mathcal{B}}(e)$ et dans ce cas, d'après la définition 29, $\delta(\sigma, b)!$ et $\delta(\sigma, b) \in \mathcal{I}_Q(e)$. On déduit alors de la définition 28 que $\delta^F(\sigma, e')!$ et appartient à $\mathcal{I}_Q(e)$. On note $e'' = \delta^F(\sigma, e')$. Puisque $e'' \in \mathcal{I}_Q(e)$, d'après la définition 29, il existe $s \in \Sigma_{uc}^*$ tel que $\delta(s, e'') = e$ et pour tout $s' \leq s$, $\delta(s', e'') = e$. Par conséquent, d'après la définition 28, il existe $s \in \Sigma_{uc}^*$ tel que $\delta^F(s, e'')!$ et $\delta^F(s, e'') = e$. On déduit alors de la définition de $\mathcal{I}(E)$ (voir (3.23)) que $e' \in \mathcal{I}(e)$ et donc que $e'' \in \mathcal{I}(E)$ (puisque $e \in \mathcal{I}(E)$ et $\mathcal{I}(\mathcal{I}(E)) = \mathcal{I}(E)$). Or $e'' = \delta^F(\sigma, e')$ et $\sigma \in \Sigma_{uc}$ donc $e' \in \mathcal{I}(E)$.

– Si on suppose maintenant que $e = [b, e_b]$ avec $e_b \in Q^b$, alors en notant $\mathcal{I}^b = \mathcal{I}^b(\Phi^*(e))$,

$$\Psi(e) = [b, \mathcal{I}^b] \cup b_{|\bigcup_{q_{o_b} \in \mathcal{I}^b \cap Q_{ob}} \{I(b)(q_{o_b})\}}$$

Puisque par hypothèse $e' \notin \mathcal{B}_{|\mathcal{A}}$, on en déduit que $e' \in [b, \mathcal{I}^b]$ et on note $e' = [b, e'_b]$ avec $e'_b \in \mathcal{I}^b$. Par définition de $\mathcal{I}^b$, il existe $s \in \Sigma_{uc}^*$ tel que $\delta^b(s, e'_b) = e_b$. Puis, on déduit de la définition de δ^F que $\delta^F(s, e') = e$. Puisque $s \in \Sigma_{uc}$, on déduit alors de la définition de $\mathcal{I}(E)$ (voir (3.23)) que $e' \in \mathcal{I}(e)$ et donc que $e' \in \mathcal{I}(E)$ (puisque $e \in \mathcal{I}(E)$ et $\mathcal{I}(\mathcal{I}(E)) = \mathcal{I}(E)$).

– Finalement, si $e = b_{|A} \in \mathcal{B}_{|A}$, alors d'après la définition 30, on a

$$\Psi(e) = \mathcal{I}_Q(b_{|\mathcal{B}_{|\mathcal{A}}}) \cup \{[b', q_{fb'}] |\ b' \in \mathcal{I}_{\mathcal{B}(b_{\mathcal{A}})}\}$$

Deux cas sont alors à prendre en compte :

– si $e' \in \mathcal{I}_Q(b_{|\mathcal{A}})$, alors d'après la définition 29, il existe $s \in \Sigma_{uc}^*$ tel que $\delta(s, e') = b$ et pour tout $s' \leq s$, $\delta(s', e') \notin \mathcal{B}$. Par conséquent, d'après la définition 28, il existe $s \in \Sigma_{uc}^*$ et $\sigma \in \Sigma_{uc}$ tels que $\delta^F(s\sigma, e')!$ avec $\sigma \in A$ et

[8] auquel, l'ensemble $\mathcal{B}_{|\mathcal{A}}$ été enlevé, dans la mesure où cet ensemble correspond en fait à l'ensemble des états initiaux des macro-états. Et ces configurations sont déjà prises en compte dans $[b, E^b]$.

$$\delta^F(s\sigma, e') = [b, q_{0b}] \qquad (\alpha)$$

Or $b_{|\mathcal{A}} \in \mathcal{E}_i$ par hypothèse et $E \cap \mathcal{B}_{|\mathcal{A}} = \emptyset$ donc il existe $e'' \in \mathcal{E}_{i-1}$ tel que $b_{|\mathcal{A}} \in \Psi(e'')$. Par définition de Ψ (Def. 30), e'' est de la forme $e'' = [b, e''_b]$ avec $e''_b \in Q^b$ et $[b, q_{0b}] \in [b, \mathcal{I}^b(\Phi^*(e''_b))]$. Donc, on déduit des définitions de δ^F (Def. 28) et $\mathcal{I}^b(\Phi^*(e''_b))$ qu'il existe $s' \in \Sigma^*_{uc}$ tel que $\delta^F(s', [b, q_{0b}]) = (b, e''_b)$ $(= e'')$.

On déduit alors de (α) que $\delta^F(s\sigma s', e') = e''$. Or $e'' \in \mathcal{E}_{i-1}$ et donc $e'' \in \mathcal{I}(E)$ par hypothèse. On obtient donc finalement de la définition de $\mathcal{I}(E)$ (voir (3.23)) $e' \in \mathcal{I}(E)$.

– si $e' = [b', q_{fb'}]$ avec $b' \in \mathcal{I}_{|\mathcal{B}}(b_{|A})$, alors d'après les définitions 29 et 28, il existe $s \in \Sigma^*_{uc}$ et $\sigma \in A$ et $\sigma \in I(b)(q_{0b})$ tels que

$$\delta^F(s\sigma, e') = [b, q_{0b}] \qquad (\beta)$$

Or $b_{|A} \in \mathcal{E}_i$ et $E \cap \mathcal{B}_{|\mathcal{A}} = \emptyset$ donc il existe $e'' \in \mathcal{E}_{i-1}$ tel que $b_{|\mathcal{A}} \in \Psi(e'')$. Par définition de Ψ (Def. 30), e'' est de la forme $e'' = [b, e''_b]$ avec $e''_b \in Q^b$ et $[b, q_{0b}] \in [b, \mathcal{I}^b(\Phi^*(e''_b))]$. Donc, on déduit des définitions de δ^F (Def. 28) et $\mathcal{I}^b(\Phi^*(e''_b))$ qu'il existe $s' \in \Sigma^*_{uc}$ tel que $\delta^F(s', [b, q_{0b}]) = (b, e''_b)$ $(= e'')$.

On déduit alors de (β) que $\delta^F(s\sigma s', e') = e''$. Puisque $e'' \in \mathcal{I}(E)$, on en déduit que $e' \in \mathcal{I}(E)$.

Finalement, on obtient que $e' \in \mathcal{I}(E)$ et donc que

$$\mathcal{I}_H(E) \setminus \mathcal{B}_{|\mathcal{A}} \subseteq \mathcal{I}(E)$$

Montrons à présent l'inclusion inverse

$$\mathcal{I}(E) \subseteq \mathcal{I}_H(E) \setminus \mathcal{B}_{|\mathcal{A}}$$

Puisque $\mathcal{I}(E) \subseteq Q^F$, on a $\mathcal{I}(E) \cap \mathcal{B}_{|\mathcal{A}} = \emptyset$ et il suffit donc de montrer que $\mathcal{I}(E) \subseteq \mathcal{I}_H(E)$.

Pour cela, considérons $q \in \mathcal{I}(E)$. Alors il existe une séquence de configurations $q = q_1, q_2, \ldots, q_n$ dans Q^F et une séquence d'événements dans Σ^F_{uc} t.q. $q = q_1$, $\delta^F(\sigma_1, q_1) = q_2, \ldots, \delta^F(\sigma_i, q_i) = q_{i+1}, \ldots, \delta^F(\sigma_{n-1}, q_{n-1}) = q_n \in E$ avec $\sigma_i \in \Sigma^F_{uc}$[9]. Montrons maintenant par induction $\forall i$, $q_i \in \mathcal{I}_H(E)$.

Par construction, $q_n \in \mathcal{I}_H(E)$ (comme $q_n \in E$ et $E \subseteq \mathcal{I}_H(E)$). Supposons maintenant que $q_i, \ldots, q_n$ appartiennent à $\mathcal{I}_H(E)$.

– Si $q_i \in Q$, alors d'après la définition de δ^F,
  – soit $q_{i-1} \in Q$ et dans ce cas, $q_{i-1} \in \mathcal{I}_Q(q_i)$. Par conséquent, $q_{i-1} \in \Psi(q_i)$ et puisque par hypothèse $q_i \in \mathcal{I}_H(E)$ et que $\Psi(\mathcal{I}_H(E)) = \mathcal{I}_H(E)$, on en déduit que $q_{i-1} \in \mathcal{I}_H(E)$.
  – soit $q_{i-1} = [b, q_{fb}]$ et dans ce cas, $\delta(\sigma_{i-1}, b) = q_i$ et donc $b \in \mathcal{I}_{\mathcal{B}}(q_i)$ d'après la définition 29. On déduit alors de la définition 30 que $[b, q_{fb}] \in \Psi(q_i)$ et donc $q_{i-1} \in \Psi(q_i)$. Et puisque par hypothèse $q_i \in \mathcal{I}_H(E)$ et que $\Psi(\mathcal{I}_H(E)) = \mathcal{I}_H(E)$, on en déduit que $q_{i-1} \in \mathcal{I}_H(E)$.
– Si $q_i = [b, q_b]$, alors
  – soit $q_{i-1} \in Q$ et $q_b = q_{0b}$ et $\sigma_{i-1} \in I(b)(q_{0b})$, et dans ce cas, d'après la définition 30, en notant $\mathcal{I}^b = \mathcal{I}^b(\Phi^*(q_{0b}))$ et $A = \bigcup_{q_{0b} \in \mathcal{I}^b \cap Q_{0b}} I(b)(q_{0b})$, on a $b_{|A} \in \Psi(q_i)$.

[9] $\Sigma^F_{uc} = \Sigma_{uc} \cup \bigcup_i \Sigma_{i,uc}$, où $\Sigma_{i,uc}$ correspond aux événements incontrôlables de G_i.

Or d'après la définition 29, on a $q_{i-1} \in \mathcal{I}_Q(b_{|A})$. Donc d'après la définition 30 encore, on a $q_{i-1} \in \Psi(b_{|A})$ et donc $q_{i-1} \in \Psi(\Psi(q_i))$. Or $q_i \in \mathcal{I}_H(E)$ par hypothèse, et $\Psi(\mathcal{I}_H(E)) = \mathcal{I}_H(E)$ par définition, donc $q_{i-1} \in \mathcal{I}_H(E)$.

- soit $q_{i-1} = [b', q_{fb'}]$, $\delta(\sigma_{i-1}, b') = b$, $q_b = q_{0b}$ et $\sigma_{i-1} \in I(b)(q_{0b})$. En notant $\mathcal{I}^b = \mathcal{I}^b(\Phi^*(q_{0b}))$ et $A = \bigcup_{q_{0b} \in \mathcal{I}^b \cap Q_{0b}} I(b)(q_{0b})$, on a $b_{|A} \in \Psi(q_i)$.
  Or d'après la définition 29, on a $b' \in \mathcal{I}_{\mathcal{B}}(b_{|A})$. Donc d'après la définition 30 encore, on a $q_{i-1} \in \Psi(b_{|A})$ et donc $q_{i-1} \in \Psi(\Psi(q_i))$. Or $q_i \in \mathcal{I}_H(E)$ par hypothèse, et $\Psi(\mathcal{I}_H(E)) = \mathcal{I}_H(E)$ par définition, donc $q_{i-1} \in \mathcal{I}_H(E)$.
- soit $q_{i-1} = [b, q'_b]$, $\delta(\sigma_{i-1}, q'_b) = q_b$, et dans ce cas $q'_b \in \mathcal{I}^b(\Phi^*(q_i))$ et donc $[b, q'_b] \in \Psi(q_i)$ d'après la définition 30. Puisque $q_i \in \mathcal{I}_H(E)$ et que $\Psi(\mathcal{I}_H(E)) = \mathcal{I}_H(E)$ par définition de $\mathcal{I}_H$, on en déduit que $q_{i-1} \in \mathcal{I}_H(E)$.

Finalement, on obtient par récurrence que pour tout $i \in \{1, \ldots, n\}$, $q_i \in \mathcal{I}_H(E)$. On a donc en particulier que $q_1 \ (= q) \ \in \mathcal{I}_H(E)$. D'où le résultat.

◇

Par conséquent, interdire $\mathcal{I}_H(E)$ dans $\mathcal{K}$ équivaut à interdire E dans $\mathcal{K}^F$. Le superviseur assurant l'interdiction de E dans $\mathcal{K}$ peut donc être décrit à partir des superviseurs locaux assurant l'interdiction des éléments de $\mathcal{I}_H(E)$. Basé sur la décomposition (3.26) de $\mathcal{I}_H(E)$, un superviseur peut alors facilement être extrait :

1. $\forall b \in \mathcal{B}$, on calcule le superviseur $\mathcal{S}_b$ assurant l'interdiction E'^b dans $\|_{j \in J_b} G_j$, en utilisant les méthodes de la section 3.2. Notons que tous les calculs ont été réalisés durant la phase de montée/descente dans la hiérarchie grâce à l'opérateur Ψ.
2. Pour la structure K, on calcule un superviseur $\mathcal{S}_K$ défini par :

$$\mathcal{S}_K(e) \quad = \quad \{\sigma \in \Sigma_c | \delta(\sigma, e) \in Q' \vee \delta(\sigma, e) = b \text{ avec } b_{|A} \in \mathcal{B}'_{|\mathcal{A}} \wedge \sigma \in A\}$$

Corollaire 8 *Avec les notations précédentes, soit $\mathcal{S} = \langle \mathcal{S}_K, (\mathcal{S}_b)_{b \in \mathcal{B}} \rangle$ t.q.*

$$S_E(e) = \begin{cases} S_K(e) \textit{ si } e \in Q \\ S_K(b) \cup S_b(q_{f_b}) \textit{ si } e = [b, q_{f_b}] \textit{ (i.e. si } e \textit{ est final dans } b\textit{)} \\ S_b(q_{j_1}, \ldots, q_{j_{\|J_b\|}}) \textit{ si } e = [b, \langle q_{j_1}, \ldots, q_{j_{\|J_b\|}} \rangle] \textit{ mais pas final} \\ \emptyset \textit{ autrement} \end{cases} \tag{3.27}$$

Le superviseur $\mathcal{S}_E$ assure l'interdiction de E dans $\mathcal{K}$ et est maximal. ◇

Basé sur (3.27), la politique de contrôle du superviseur est la suivante :

- Quand la configuration courante de $\mathcal{K}$ est dans Q, alors le superviseur $\mathcal{S}_K$ est actif. $\mathcal{S}_K$ est également activé lorsque le système sous contrôle entre dans un état final d'un macro-état. on peut également remarquer que $\mathcal{S}_K$ prend également en compte les configurations initiales de macro-états qui sont interdites.
- Quand $\mathcal{K}$ entre dans un macro état b sous le contrôle de $\mathcal{S}_K$ (ce qui signifie que le système entre dans ce macro-état via des événements permis) alors le superviseur $\mathcal{S}_b$ devient actif. Il est désactivé dès lors que le système sous contrôle sort de ce macro-état.

Pour conclure cette section, remarquons que tout comme en Section 3.2, le superviseur a été synthétisé sans avoir calculé explicitement $\mathcal{K}^F$. En particulier, l'ensemble des configurations interdites a été calculé localement sur chaque $(G_i)_{1 \leq i \leq n}$ et sur la structure K.

3.5 Conclusion

Dans ce chapitre, nous nous sommes intéressés au problème de l'interdiction d'états sur des systèmes structurés (concurrents & hiérarchiques) avec pour priorité l'efficacité algorithmique dans le but de combattre la complexité inhérente à ce type de systèmes, en utilisant leur structure pour améliorer la phase de synthèse (voire la rendre réalisable). Pour cela, l'approche adoptée provient à la fois des travaux sur l'interdiction d'états introduits dans [58], mais aussi des travaux réalisés dans [54] avec comme soucis la minimisation des calculs en-ligne (en regard des calculs hors-ligne).

Dans le cadre du contrôle des systèmes concurrents, la synthèse de contrôleurs s'effectue alors en deux phases : une phase de calculs hors-ligne, et une phase de calcul en-ligne. Les objectifs de contrôle sont modélisés par un ensemble d'états à interdire. Cet ensemble est structuré (union de pavés), mais reste général au sens où n'importe quel ensemble d'états peut être exprimé sous cette forme. Les structures du système et de l'objectif sont alors utilisées pour limiter la complexité des calculs hors-ligne et en-ligne. La nature des événements partagés joue un rôle dans notre approche. Lorsque tous les événements partagés sont contrôlables, alors le calcul des ensembles d'états interdits se réduit au calcul d'ensembles locaux d'états interdits sur chacun des sous-systèmes. Cette décomposition du calcul a un fort impact sur la complexité des deux phases de calculs (hors-ligne et en-ligne) qui est alors inférieure à celle du calcul explicite du système. Lorsque le système à contrôler peut se synchroniser sur des événements incontrôlables, alors la complexité de la phase hors-ligne peut alors s'avérer coûteuse dans le pire cas. Toutefois, beaucoup de systèmes complexes sont faiblement synchronisés et la complexité du calcul hors-ligne peut s'avérer faible en pratique. De plus, la complexité de la phase de calculs en-ligne reste faible dans ce cas. Ces résultats ont également été étendus au cadre de systèmes hiérarchiques à deux niveaux.

Dans ce chapitre, nous avons également abordé les problèmes de blocages. Plus précisément, nous nous sommes intéressés au problème de l'évitement de deadlock dans un système contrôlé, obtenu par action d'un superviseur tel que décrit précédemment. Ce problème peut être ramené à celui de la détection des états en deadlock dans un système concurrent sous contrôle. Afin de tirer parti de la structure du système et du superviseur, nous avons été amenés à décomposer le problème en deux phases :

– détection des états initialement en deadlock dans le système.
– détection des états en deadlock dans le système contrôlé, subissant l'action du superviseur.

La première phase a été traitée dans d'autres travaux (par exemple [1]) et peut être notamment efficacement résolue sur des systèmes faiblement synchronisés. Une caractérisation pertinente des états en deadlock dans un système concurrent est ainsi fournie. La seconde phase tire parti des calculs effectués pour déterminer le superviseur. Une caractérisation pertinente des états en deadlock subissant l'action du superviseur est ainsi fournie.

Enfin, dans le but de limiter encore davantage l'espace de recherche des états précédemment caractérisés, une méthode de recherche incrémentale est proposée. Cette méthode tire profit de la structure concurrente du système à contrôler. Les notions de projections de systèmes concurrents et de superviseurs sont introduites. Ces projections permettent de considérer des compositions parallèles ne mettant en jeu qu'un sous ensemble des sous systèmes du système à contrôler. De plus, ces projections préservent les caractérisations précédemment introduites des états en dead-

lock dans le système contrôlé. Par conséquent, il est possible de combiner les caractérisations des états en deadlock et la notion de projection pour affiner l'espace de recherche de manière incrémentale (i.e en considérant initialement la projection correspondant à un seul sous système, puis en ajoutant à chaque étape un nouveau sous système dans la projection, jusqu'à ce que tous les sous systèmes soient considérés).

Perspectives

- Dans ce chapitre, nous avons développé une méthodologie permettant de détecter efficacement les états bloquants du système contrôlé. Il semblerait toutefois intéressant de pouvoir identifier des conditions suivant lesquelles le problème du non-blocage pourrait être traité dans sa globalité. Ceci passe notamment par une étude du problème d'atteignabilité (ou co-atteignabilité) modulaire dans un système concurrent.
- De même, il nous semblerait intéressant d'étendre nos techniques de synthèse à des modèles hiérarchiques possédant n niveaux. S'il est aisé de faire cette extension dans un cadre asynchrone (i.e. les différentes machines ne partagent aucun événement), le problème s'avère plus ardu dans le cadre général dans la mesure où des synchronisations peuvent s'opérer à différents niveaux dans la hiérarchie. Cela induit notamment un comportement du système global relativement difficile à appréhender. Une extension à des objectifs plus généraux (e.g. des langages) pourrait également être envisagée.
- Nous pourrions nous intéresser au contrôle de systèmes à événements discrets modélisés par des machines à états finis synchrones dans lesquelles l'occurrence de plusieurs événements peut être simultanée (c.f. [40, 51, 69, 61]). De tels systèmes se retrouvent dans les systèmes temps-réel tels les systèmes robotiques, automobiles ou avioniques. Ce type de système est conçu avec de multiples tâches cycliques, chacune dotée de modes multiples pour lesquels il est complexe de concevoir les gestionnaires de tâches qui contrôlent la commutation des activités de façon à assurer des propriétés de sûreté sur le système global (c.f. [64]).

Annexe A

SynTool

La théorie de la synthèse de contrôleurs sur des systèmes à événements discrets, introduite par Ramadge et Wonham [59], est décrite pour des systèmes modélisés par des langages sur un alphabet. Afin de rendre les calculs effectifs, il convient de ne considérer que des systèmes modélisables par un langage régulier. Dans ce cas, les systèmes considérés peuvent être représentés par des automates. Il est alors possible de manipuler les modèles de ces systèmes et de leur appliquer des algorithmes de synthèse de contrôleurs. Dans le cadre de cette thèse, nous avons considéré des systèmes structurés i.e. modélisés par plusieurs sous-systèmes interagissant entre-eux et avons développé des techniques qui consiste à tirer parti de la structure particulière du système, en effectuant autant que possible des calculs sur les sous systèmes, plutôt que sur le système global. Les algorithmes ainsi développés nous ont amené à développer un prototype pour valider notre approche.

A.1 Outils Existants

Il existe à l'heure actuelle de nombreux outils dédiés à la synthèse de contrôleurs, permettant d'une part de manipuler des automates (ainsi que les langages associés) et de synthétiser des superviseurs. La plupart d'entre eux ont une interface textuelle et travaille sur des systèmes monolithiques sans prendre en compte la structure du système. Parmi ces derniers, on peut citer :

TCT développé à l'université de Toronto (Canada) est le premier des outils dédié à la synthèse de contrôleurs [71].

DESCO développé à l'université de Chambers (Suède). Idem que TCT.

UMDES développé à l'université du Michigan (USA). La librairie UMDES regroupe les principaux algorithmes liés à la théorie du contrôle (ainsi que des algorithmes liés au diagnostic) [74].

J-DES développé à l'université de Pennsylvanie (USA). Tout comme UMDES il contient les algorithmes de base de la théorie du contrôle et possède une interface graphique très simple.

UKDES développé à l'université de Kentucky (USA) possède les mêmes fonctionnalités que UMDES ou TCT et une interface graphique pour manipuler les automates [73].

Les algorithmes classiques de synthèse de contrôleurs sont donc implémentés dans ces divers outils. Toutefois ces algorithmes ne prenent pas en considération la structure éventuelle du système. Par conséquent, les outils précédemment cités n'ont pas été conçus pour appliquer les éléments théoriques introduits dans les chapitres précédents. En particulier, les algorithmes implémentés nécessitent généralement de construire explicitement (éventuellement avec une structure de données pertinente) le système à contrôler.

Il existe également certains outils qui ont été conçus avec un souci d'efficacité algorithmique, dans le but de pouvoir traiter des systèmes de grandes tailles. Par exemple, STCT [82] développé à l'université de Toronto, SUPREMICA [3, 67] développé à l'université de Chambers et SIGALI [65, 51] développé à l'IRISA utilisent un codage efficace des automates grâce aux *diagrammes de décision binaire* (BDD). Les BDD, introduits par Bryant [10], constituent une structure de données intéressante pour représenter les automates. Il permettent notamment d'obtenir en pratique un gain en complexité, permettant de traiter des systèmes de grandes tailles.

Parmi tous ces outils, SUPREMICA est sans doute celui qui utilise au mieux d'une part une structure de donnée efficace pour coder les automates et d'autre part la structure des systèmes pour calculer les contrôleurs. Il permet en outre la simulation du système contrôlé. Toutefois, il se base sur une approche décentralisée et réorganise les sous-systèmes de manière à les rendre modulaires.

A.2 SYNTOOL

SYNTOOL est un prototype actuellement en cours de développement qui vise à implémenter les différents algorithmes présentés dans le cadre de cette thèse. Il s'agit d'un outil programmé en Java comprenant des librairies qui permettent de manipuler des automates ou des collections d'automates. De plus, ces librairies peuvent être utilisées à travers une interface graphique permettant de manipuler les graphes.

A.2.1 Manipulation et représentation des systèmes dans SYNTOOL.

Pour représenter des systèmes à événements discrets dans SYNTOOL, le modèle de base utilisé est l'automate. Un système concurrent est donné par une collection d'automates. Les automates peuvent être définis textuellement sous un format dit "*.vtc*" ou graphiquement à travers l'éditeur de graphes VGJ. De plus, VGJ a été étendu de sorte qu'il soit possible de générer un fichier au format ".vtc" depuis VGJ, mais aussi d'ouvrir un fichier ".vtc" avec VGJ. Le format ".vtc" permet de définir plusieurs automates à l'intérieur du même fichier. Le système décrit est alors concurrent et chacun des automates représente un de ses sous systèmes. La Table A.1 présente un exemple de fichier au format ".vtc" :

Sur cet exemple, le fichier définit un système concurrent composé de deux sous systèmes modélisés par des automates. Chacun de ces automates est décrit par

- l'ensemble de ces états,
- son alphabet,
- et l'ensemble de ses transitions.

Deux booléens sont associés à chaque état pour exprimer le caractère initial ou final de celui-ci. De même, deux booléens sont associés à chaque événement de l'alphabet afin d'exprimer le caractère contrôlable et observable de celui-ci.

Graphiquement, la figure A.1 représente dans VGJ le système concurrent décrit précédemment. Le caractère initial et final des états, ainsi que le caractère contrôlable des événements sont aussi définissables graphiquement, comme le montre les figures A.2 et A.3.

Remarque 12 *Il est également possible de saisir tant textuellement que graphiquement un automate hiérarchique. Toutefois, les algorithmes de synthèse m'ayant pas encore été implémentés nous ne présentons pas la syntaxe dans cette annexe.*

A.3 Différentes fonctionnalités de SYNTOOL

Outre l'aspect saisi d'un automate ou d'un système concurrent, différents algorithmes permettant de les manipuler ont également été implémentés. On retrouve ainsi

Fsm : graph_0
Etats :
Etat_0 0 true false
Etat_1 1 false false
Etat_2 2 false false
Etat_3 3 false false

Alphabet : Alphagraph_0
b true true 0.0
a true true 0.0
c true true 0.0

Transitions :
3 a 2
0 a 1
1 b 3
0 c 3
FinFsm

Fsm : graph_1
Etats :
Etat_0 0 true false
Etat_1 1 false false
Etat_2 2 false false

Alphabet : Alphagraph_1
a true true 0.0
e true true 0.0

Transitions :
0 a 1
1 e 2
FinFsm

TAB. A.1: Exemple de description d'un système concurrent

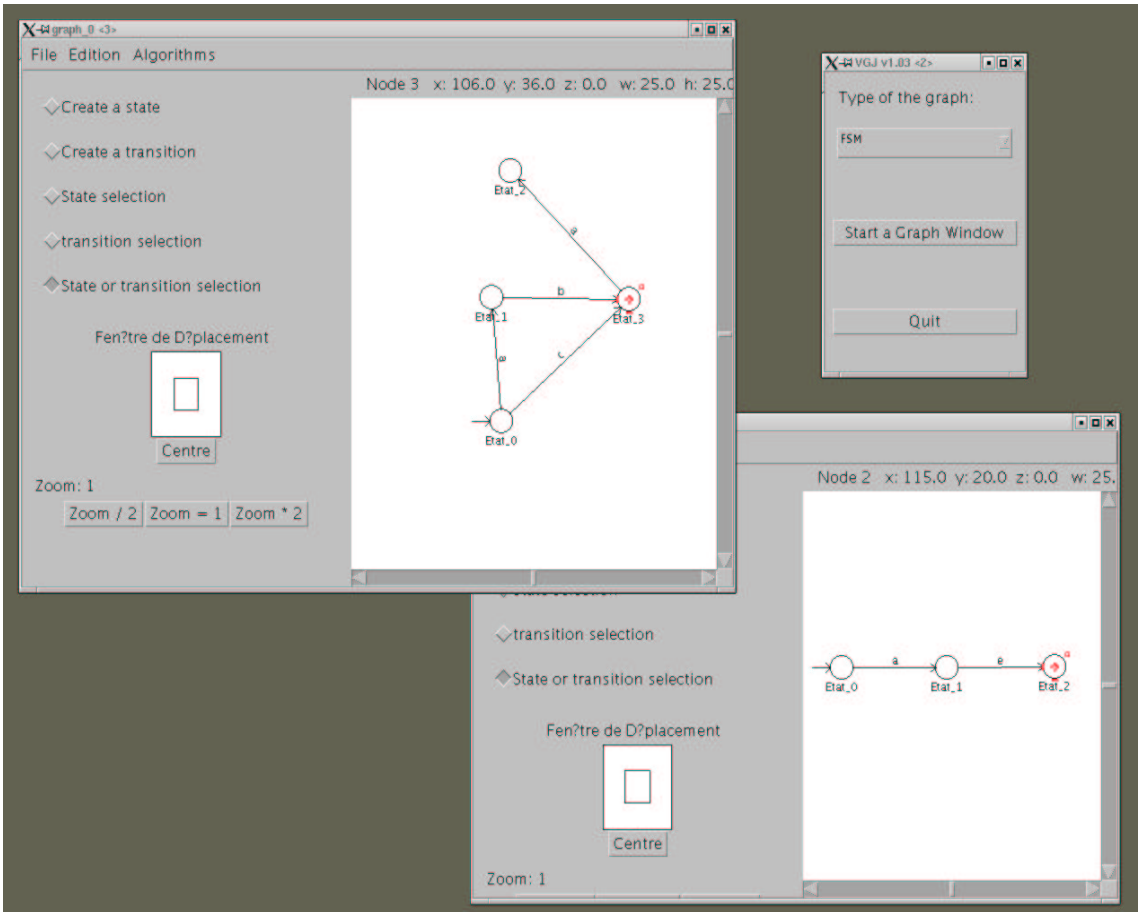


FIG. A.1: Exemple d'un système concurrent avec SynTool

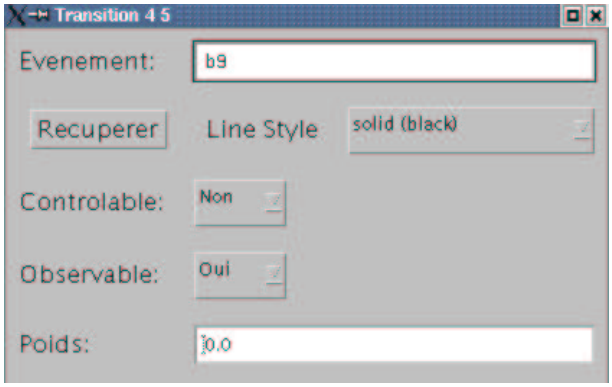


FIG. A.2: Définition des caractéristiques des événements.

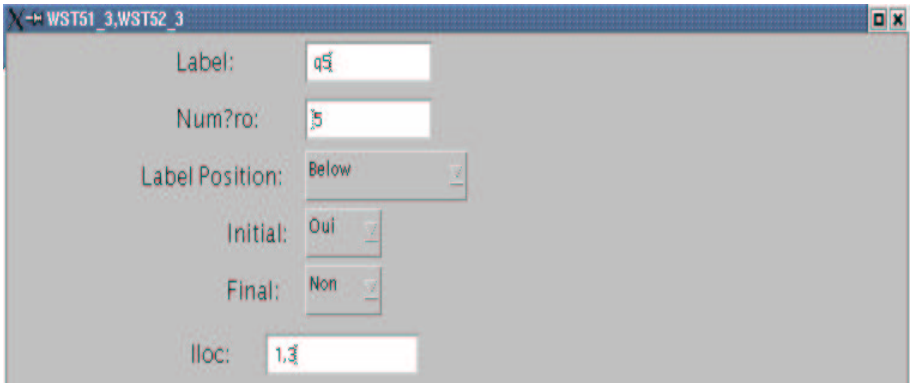


FIG. A.3: Définition des caractéristiques des états.

- des algorithmes de manipulations d'automates (Reachable, Coreachable, Trim, ainsi que les différents types de produits (mixte ou synchrone)),
- certains des algorithmes classiques de synthèse de contrôleurs pour des systèmes monolithiques résolvant
  - le problème de l'interdiction d'états
  - le problème général avec l'objectif donné sous forme d'un automate.
  - le problème du non-blocage

  Une simulation interactive du système contrôlé à également été implémentée.
- ainsi que des algorithmes de synthèse dédiés aux systèmes concurrents (c.f. Section suivante).

A.3.1 Problème de l'interdiction d'états pour un système concurrent.

Dans le chapitre 3, les opérateurs $\Phi(\cdot)$ et $\mathcal{I}_{loc}(\cdot)$ ont été introduits pour résoudre les problèmes de synthèse de contrôleurs lorsque les objectifs de contrôle représentent des propriétés de sûreté modélisées par un ensemble d'états à interdire. Ces opérateurs ont été implémentés dans SYNTOOL, ainsi que la méthode de détection de deadlock présentée dans la section 3.3. Notons que ces méthodes sont basées sur une décomposition de l'ensemble des états à interdire sous forme de pavés. Dans SynTool, les états (tuple) à interdire sont représentés en étiquetant chacune de leurs composantes par le même entier. Cet étiquetage peut s'effectuer graphiquement, en même temps que les caractéristiques des états (voir figure A.3). Enfin, puisque les résultats du chapitre 3 ne permettent pas de calculer explicitement le système contrôlé ni d'obtenir une réalisation standard du superviseur, un algorithme de simulation du système contrôlé a également été implémenté.

Le système représenté en Table A.2 est constitué de deux systèmes : Une presse et un bras mécanique. Le problème est d'empêcher que le bras soit étendu alors que la presse est fermée. Il convient donc d'interdire le couple (Fermée, tendu).

Pour cela, on spécifie dans le fichier que pour la FSM Presse l'état 2 (i.e. fermée) appartient au couple interdit 1 ($2 : \{1\}$). De même pour la FSM Bras l'état 2 (i.e. tendu) appartient au couple interdit 1 ($2 : \{1\}$).

Pour réaliser la synthèse, il suffit alors de faire appel à la fonction SIMULATIONWITHCON-

Fsm : Presse
Etats :
ouverte 1 true false
fermée 2 false true
Alphabet : alpha1
ouvrir true true 1
fermer true true 2
Transitions :
1 fermer 2
2 ouvrir 1
FinFsm

Fsm : Bras
Etats :
plié 1 true false
tendu 2 false true
Alphabet : alpha2
tendre true true 1
plier true true 2
Transitions :
1 tendre 2
2 plier 1
FinFsm

Forbidden :
Fsm : Presse
Etats :
2 : {1}

Fsm : Bras
Etats :
2 : {1}
FinForbidden
Fin

TAB. A.2: Exemple de description du système Bras/Presse

TROLER qui prend en paramètres un ensemble de FSMs (produit asynchrone ou composition parallèle) et réalise la simulation correspondante, en ayant demandé auparavant les tuples à interdire et réalisé les calculs correspondants (weakforbidden states spécifiques au produit). La simulation se fait de manière interactive.

À chaque étape, SYNTOOL fournit à l'utilisateur l'ensemble des événements qui n'ont pas été interdits par contrôle.

FIG. A.4: Simulation interactive

Après le choix de l'événement, le simulateur calcul le nouvel état global du système et affiche sur chacun des automates modélisant le système son état courant.

A.3.2 Synthèse générale pour des système concurrents

Dans le chapitre 2, la contrôlabilité partielle d'un langage (Définition 14) et la cohérence locale (Définition 15) constituent dans notre étude des notions centrales de la synthèse de contrôleurs, lorsque l'objectif de contrôle est une propriété de sûreté modélisée par un ensemble de comportements. Les algorithmes permettant la vérification de la cohérence locale d'un langage, ainsi que le calcul du plus grand langage partiellement contrôlable ont été implémentés. La simulation du système contrôlé n'a pour l'instant pas été implémentée.

De manière générale, étant donné un système concurrent $G = G_1 \parallel \cdots \parallel G_n$ et un objectif G_K, l'utilisateur doit créer un fichier contenant les n machines G_i ainsi que la machine G_K située en dernière position dans le fichier. En sortie, SYNTOOL, après avoir testé la cohérence lo-

cale, renvoie un fichier contenant les n machines modélisant les plus grand langage partiellement contrôlable de $K \cap P_i^{-1}(G_i)$.

A.4 Exemple : Système des chariots filoguidés

On considère dans cette section l'exemple classique des *chariots filoguidés* introduits par [33]. Le système à contrôler représente une chaîne de production manufacturière, composée de 5 stations de travail interagissant par l'intermédiaire de 5 chariots filoguidés. Ces chariots se déplacent sur des rails et transportent du matériel entre deux stations de travail.

Plus précisément, la figure A.5 représente un réseau de Petri modélisant le déplacement des ressources. Ces ressources sont constituées des chariots se déplaçant entre les stations de travail, et des ressources introduites dans le système afin d'y subir un traitement. Ainsi, les chariots sont notés $AGV1, \ldots, AGV5$ et pour $i \in \{1, \ldots, 5\}$, le parcourt du chariot $AGVi$ est modélisé par la partie du réseau notée $AGVi$. De même, les stations de travail sont notées $WST1, \ldots, WST5$ et pour $i \in \{1, \ldots, 5\}$, le parcours d'une ressource lorsqu'elle est traitée par la station $WSTi$, est modélisé par la partie du réseau notée "work station i" sur la figure A.5.

On décrit à présent le comportement du système global. Deux ressources sont introduites dans la station de travail $WST1$. Elles y subissent un traitement, puis une des ressources est prise en charge par le chariots $AGV1$ alors que l'autre est prise en charge par le chariot $AGV2$. Le chariot $AGV1$ mène la première ressource à la station de travail $WST2$ et le chariot $AGV2$ mène la seconde à $WST3$. Chacune de ces deux ressources subit alors un traitement dans la station correspondante, puis sont prises en charge par $AGV3$ pour la première, et $AGV4$ pour la seconde. Ces deux chariots mènent à la station $WST5$ qui transforme ces deux ressources en une unique troisième. Finalement, cette ressource est prise en charge par le chariot $AGV5$ pour être conduite à la station $AGV5$ où elle subit un dernier traitement avant de quitter le système.

Ce comportement est intuitivement représenté par le réseau de Petri donné en figure A.5. Ce réseau de Petri laisse clairement apparaître les sous systèmes qui composent le système global. Les résultats introduits dans les chapitres précédents étant basés sur une modélisation par des automates, on donne maintenant des automates modélisant le comportement de chacun des chariots filoguidés et des stations de travail.

Les comportements de la station de travail $WST1$ peuvent être décrits par la composition de deux automates, notée $WST11$ et $WST12$. De même, les comportements de la station de travail $WST5$ peuvent être décrits par la composition des deux automates $WST51$ et $WST52$ donnés par la figure A.8. En revanche, les comportements des autres stations de travail, et ceux des chariots filoguidés peuvent chacun être décrit par un unique automate (figures A.7, A.9, A.10 et, A.11).

Avec cette modélisation, le système global peut être considéré comme un système concurrent. Il est ici modélisé par la composition de douze automates, représentant chacun un sous système. Le nombre d'états d'un automate varie ici de deux à douze, et le nombre d'états du système global est évalué à environ trente millions. Par conséquent, il s'agit bien là d'un système complexe et il n'est pas envisageable en pratique de le représenter par un unique automate, qui pourrait être obtenue par l'évaluation de la composition parallèle de chacun des automates.

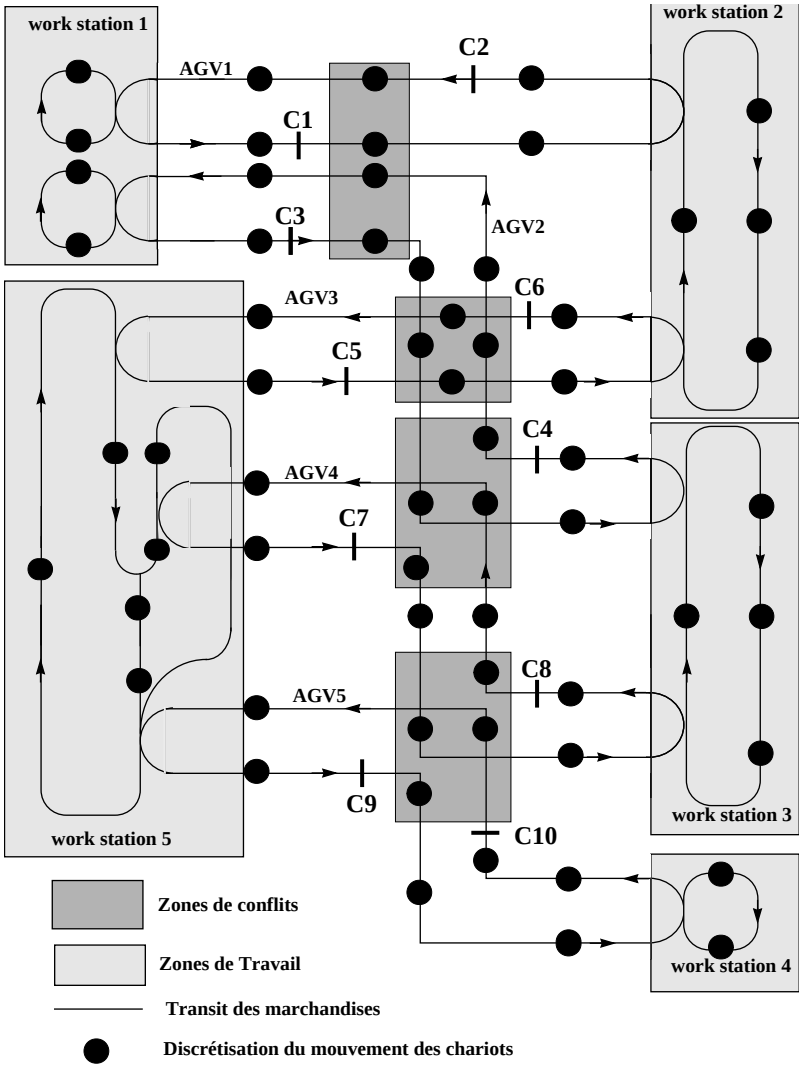


FIG. A.5: Vision globale du système

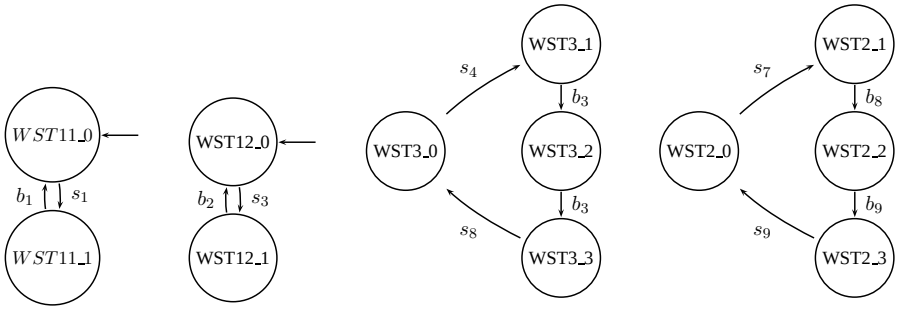


FIG. A.6: Station de travail WST11, WST12 et WST2

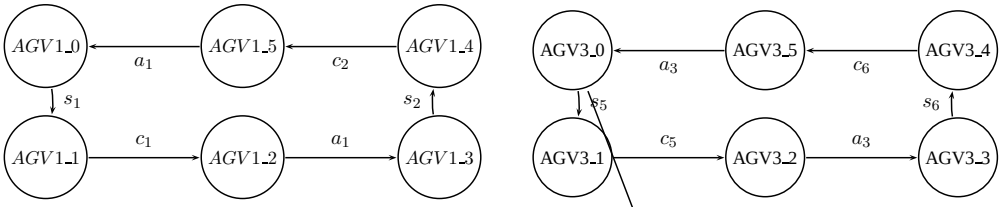


FIG. A.7: Chariot AGV1 & AVG3

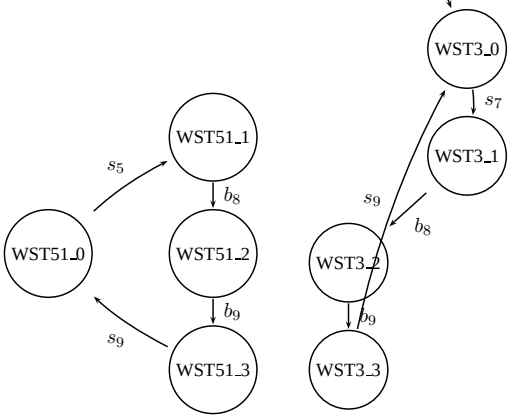


FIG. A.8: Stations de travail WST51 & WST52

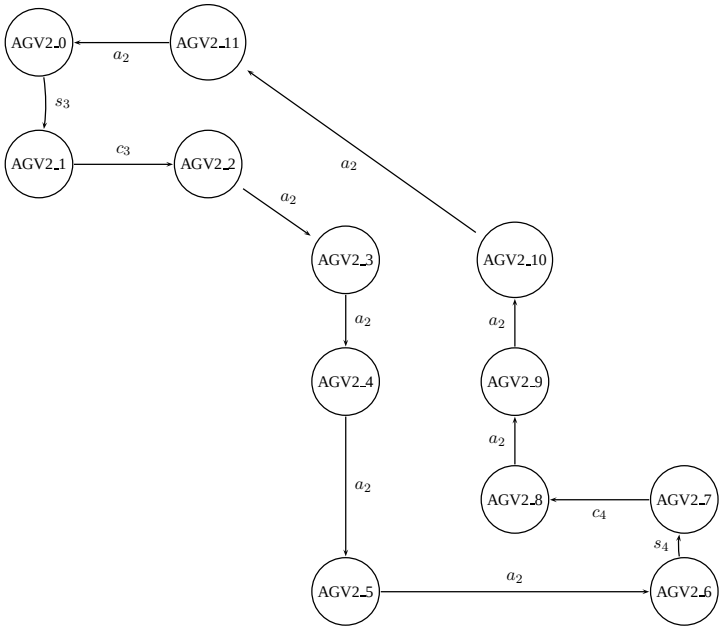


FIG. A.9: Chariot AGV2

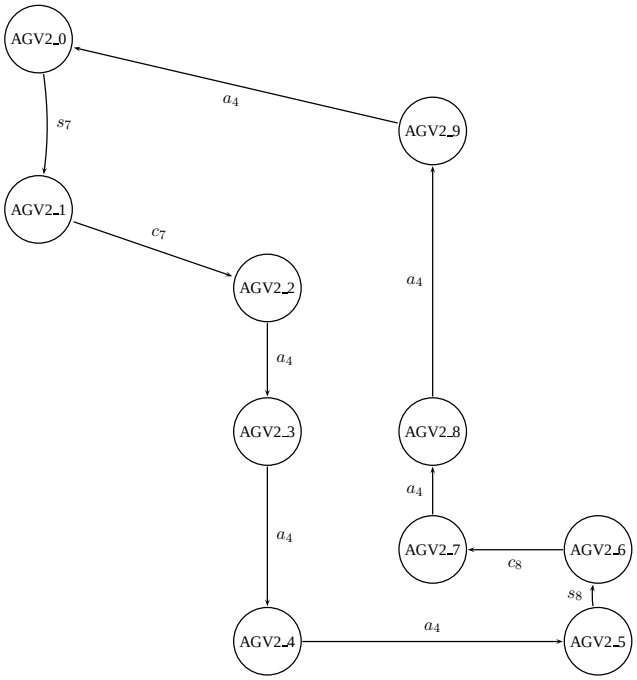


FIG. A.10: Chariot AGV4

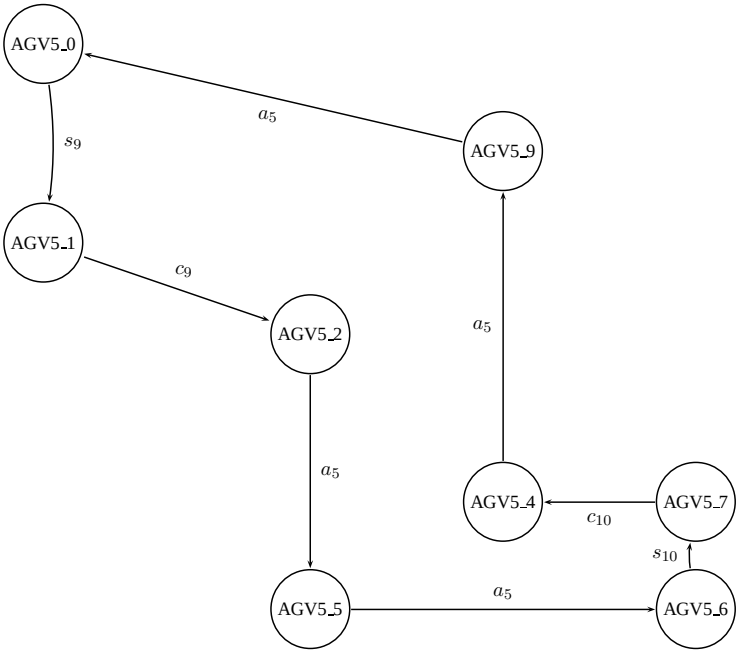


FIG. A.11: Chariot AGV5

A.5 Problèmes de synthèse de contrôleurs.

Comme évoqué précédemment, l'exemple des chariots filoguidés a déjà été traité dans le cadre de la synthèse de contrôleurs. L'objectif de contrôle classiquement associé à ces études consiste à interdire un ensemble d'états du système. Cet exemple est ici repris sous cette forme pour illustrer les résultats du chapitre 3 : calcul du superviseur en deux phases (hors-ligne et en-ligne) et obtention d'une solution assurant que le système contrôlé est sans état en deadlock. Dans un second temps, afin d'illustrer les résultats du chapitre 2, on s'intéressera à un objectif de contrôle portant sur les comportements du système global, et donné par un automate.

A.5.1 Interdiction d'états

Les trajectoires des chariots filoguidés peuvent créer des conflits. En effet, certains chariots possèdent des trajectoires similaires, rendant possible les collisions entre chariots. Ces zones de conflits sont identifiées et représentées par les zones grisées de la figure A.5. Il est donc nécessaire de contrôler le système afin que deux chariots ne puissent atteindre simultanément une de ces zones. L'objectif de contrôle peut s'exprimer par l'interdiction d'un ensemble d'états E, qui est naturellement donné comme l'union de quatre pavés.

$$E = \bigcup_{1 \leq j \leq 4} E^j$$

avec

$$\begin{aligned}
E^1 &= \{AGV1_2, AGV1_5\} \times \{AGV2_2, AGV2_11\} \times Q_{AGV3} \times Q_{AGV4} \times Q_{AGV5} \times_i Q_{WSTi} \\
E^2 &= \{AGV2_4, AGV2_9\} \times \{AGV3_2, AGV3_5\} \times Q_{AGV1} \times Q_{AGV4} \times Q_{AGV5} \times_i Q_{WSTi} \\
E^3 &= \{AGV2_5, AGV2_8\} \times \{AGV4_2, AGV4_9\} \times Q_{AGV1} \times Q_{AGV3} \times Q_{AGV5} \times_i Q_{WSTi} \\
E^4 &= \{AGV4_4, AGV4_7\} \times \{AGV5_2, AGV5_9\} \times Q_{AGV1} \times Q_{AGV2} \times Q_{AGV3} \times_i Q_{WSTi}
\end{aligned}$$

où Q_{AGVi} et Q_{WSTi} représentent les ensembles des états des automates $AGVi$ et $WSTi$. De plus, seuls les événements $\{c_i\}_{1 \leq i \leq 10}$ sont supposés contrôlables dans ce système.

$$\Sigma_c = \{c_i\}_{1 \leq i \leq 10}$$

Dans cet exemple, bien que tous les événements partagés soient incontrôlables, le calcul de $\Phi^*(E)$ ne nécessite qu'une itération (on a $E = \Phi^*(E)$). Le calcul hors-ligne du superviseur consiste alors à déterminer tous les ensembles $\mathcal{I}_{loc}(E_i^j)$ pour $j \in \{1, \ldots, 4\}$ et $i \in \{1, \ldots, 12\}$ et la simulation s'opère comme expliquée en Section A.3.1.

A.5.2 Objectif portant sur le comportement du système global

A présent, afin d'illustrer les résultats du chapitre 2, on considère un autre objectif de contrôle donné par un automate qui modélise l'ensemble des comportements souhaités. Par rapport à l'exemple précédent, on suppose de plus que les événements partagés sont contrôlables :

$$\Sigma_c = \{c_i\}_{1 \leq i \leq 10} \cup \{s_i\}_{1 \leq i \leq 9}$$

Cette hypothèse traduit qu'il est possible d'empêcher chaque chariot d'interagir avec des stations de travail. Dans ce cadre, l'objectif de contrôle K considéré est donné par l'automate représenté en figure A.12.

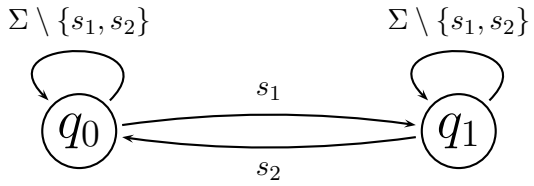


FIG. A.12: Objectif de contrôle portant sur les comportements du système

Cet objectif traduit le fait qu'il doit toujours y avoir une alternance entre l'interaction du chariot AGV1 et de la station WST1, et l'interaction du chariot AGV2 et de cette même station. De plus, l'interaction entre AGV1 et WST1 doit se produire la première.

SynTool permet de vérifier que l'objectif de contrôle K est *localement cohérent*, et permet aussi le calcul des plus grand langages partiellement contrôlables, induisant douze superviseurs dont la conjonction assure l'objectif de contrôle de façon maximale.

À titre d'exmple, nous donnons ici les automates modélisant les plus grand langages partiellement contrôlables par rapport à K et $AGV1$ (Figure A.13) et $WST51$ (Figure A.14).

A.6 Conclusion

SYNTOOL est un prototype dédié au contrôle de systèmes à évènements discrets structurés qui implémente en partie les divers algorithmes présentés dans les chapitre 2 et 3. SYNTOOL est doté d'une interface permettant à l'utilisateur de décrire graphiquement les différents automates modélisant le système à contrôler. Il permet en outre de résoudre certains problèmes de contrôle tels que le problème de l'interdiction d'états où le problème de base de la synthèse de contrôleurs. Dans la mesure où le système contrôlé n'est en général pas représentable par un automate, nous avons également développé dans le cadre de l'interdiction d'états un simulateur permettant de visualiser de manière interactive le comportement du système contrôlé.

Cet outil est actuellement en cours de développement. Outre l'aspect simulation du système contrôlé dans le cadre général, nous comptons également implémenter les algorithmes de synthèse et de simulation pour des systèmes hiérarchiques (sachant qu'il est déjà possible de saisir manuellement/graphiquement ce type de systèmes).

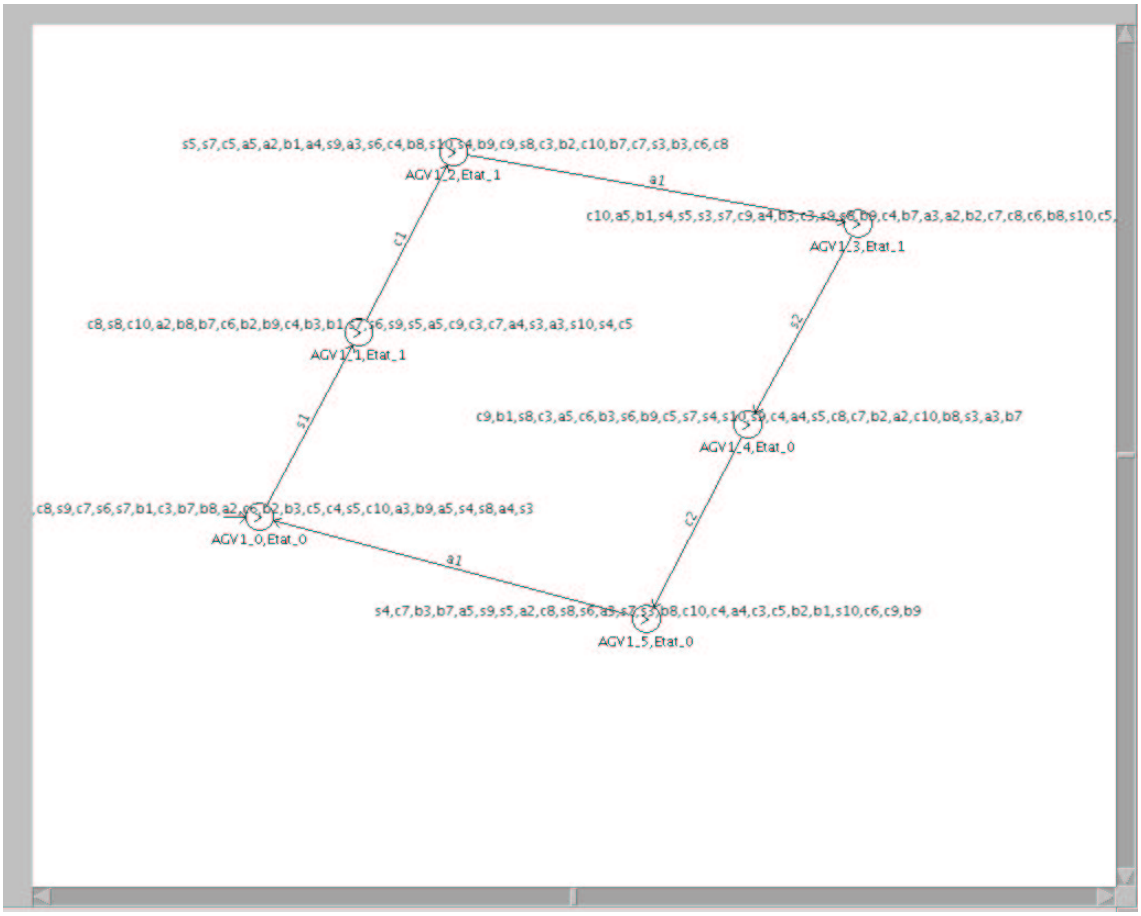


FIG. A.13: $K^{\uparrow pc}_{AGV1}$

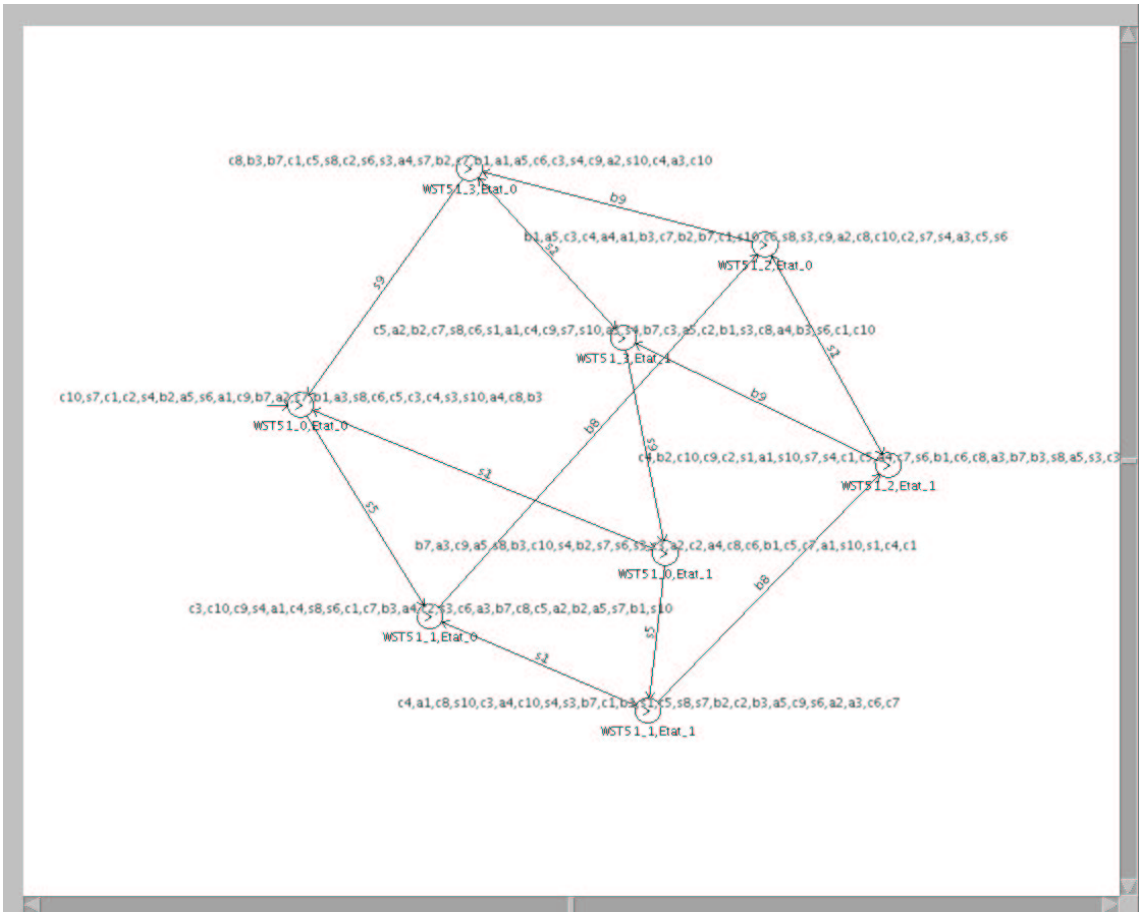


FIG. A.14: $K^{\uparrow pc}_{WST51}$

Bibliographie

[1] A. Abdelwahed. *Interacting Discrete event systems : modeling, verification an supervisory control*. PhD thesis, University of Toronto, 2002.

[2] S. Abdelwahed and W. Wonham. Supervisory control of interacting discrete event systems. In *41th IEEE Conference on Decision and Control*, pages 1175–1180, Las Vegas, USA, December 2002.

[3] K. Akesson, M. Fabian, H. Flordal, and A. Vahidi. Supremica - a tool for verification and synthesis of discrete event supervisors. In *11th Mediterranean Conference on Control and Automation*, Rhodos, Greece, 2003.

[4] A. Arnold. *Finite transition systems*. Prentice Hall, NJ, 1994.

[5] M. Barbeau, G. Custeau, and R. St-Denis. An algorithm for computing the mask value of the supremal normal sublanguage of a legal language. *IEEE Transactions on Automatic Control*, 40(4) :699–704, 1995.

[6] A. Belinfante, J. Feenstra, R. de Vries, J. Tretmans, N. Goga, L. Feijs, and S. Mauw. Formal test automation : a simple experiment. In *International Workshop on the Testing of Communicating Systems (IWTCS'99)*, pages 179–196, 1996.

[7] R. D. Brandt, V. K. Garg, R. Kumar, F. Lin, S. I. Marcus, and W. M. Wonham. Formulas for calculating supremal and normal sublanguages. *Systems and Control Letters*, 15(8) :111–117, 1990.

[8] Y. Brave and M. Heymann. Control of discrete event systems modeled as hierarchical state machines. In *Proceedings of the 30th IEEE Conference on Decision and Control*, pages 1499–1504, 1991.

[9] Y. Brave and M. Heymann. Control of discrete event systems modeled as hierarchical state machines. *IEEE Transacations on Automatic Control*, 38(12) :1803–1819, December 1993.

[10] R. E. Bryant. Symbolic boolean manipulation with ordered binary-decision diagrams. *ACM computing Surveys*, pages 293–318, September 1992.

[11] P. Caines, V. Gupta, and G. Shen. The hierarchical control of ST-finite-state machines. *Systems and Control Letters*, 32 :185–192, 1997.

[12] P. Caines and Y. Wei. The hierarchical lattice of a finite machine. *Systems & Control Letters*, 1996.

[13] C. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Kluwer Academic Publishers, 1999.

[14] S. Chen. *Control of discrete-event systems of vector and mixed strctural type*. PhD thesis, Dept. of Electrical and Computer Engineering, University of Toronto, Canada, 1996.

[15] Y.L. Chen and H.M. Hanish. Model aggregation for hierarchical control synthesis of discrete event systems. In *39th IEEE Conference on Decision and Control*, pages 418–423, Sydney, Australia, December 2000.

[16] H. Cho and S. J. Marcus. On supremal languages of classes of sublanguages that arise in supervisor synthesis problems with partial observation. *Mathematics of Control, Signals and Systems*, 2(2) :47–69, 1989.

[17] 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.

[18] M. H. De Gueiroz and J. Cury. Modular control of composed systems. In *Proceedings of the American Control Conference*, pages 4051–4055, Chicago, Illinois, June 2000.

[19] M.H. deQueiroz and J.E.R. Cury. Modular supervisory control of large scale discrete-event systems. In *Discrete Event Systems : Analysis and Control. Proc. WODES'00*, pages 103–110. Kluwer Academic, 2000.

[20] M.H. deQueiroz and J.E.R. Cury. Synthesis and implementation of local modular supervisory control for a manufacturing cell. In *Proceedings of the 6th International Workshop on Discrete Event Systems*, pages 377–382, October 2002.

[21] J. Eyzell and J. Cury. Exploiting symmetry in the synthesis of supervisors for discrete event systems. In *Proc. of the American Control Conference*, pages 244–248, Philadelphia, Pennsylvania, USA, June 1998.

[22] M. Fabian and A. Hellgren. Plc-based implementation of supervisory control for discrete event systems. In *37th IEEE Conference on Decision and Control*, Tampa, Florida, USA, December 1998.

[23] B. Gaudin and H. Marchand. Modular supervisory control of asynchronous and hierarchical finite state machines. In *European Control Conference, ECC 2003*, Cambridge, UK, September 2003.

[24] B. Gaudin and H. Marchand. Modular supervisory control of a class of concurrent discrete event systems. In *Workshop on Discrete Event Systems, WODES'04*, September 2004.

[25] B. Gaudin and H Marchand. Supervisory control of concurrent discrete event systems. Research Report 1593, IRISA, February 2004. available at http ://www.irisa.fr/vertecs/Publis/Ps/1593.ps.

[26] R. Germundsson. *Symbolic Systems - Theory, Computation and Applications*. PhD thesis, Linköping University, 1995.

[27] P. Gohari-Moghadam. *A linguistic Framework for controller hierarchical DES*. M.a.s.c. thesis, Dept. of Electl. & Comptr. Engrg., University of Toronto, April 1998.

[28] D. Gouyon, J.F. Petin, and Gouin A. Pragmatic approach for modular control synthesis and implementation. *International Journal of Production Research*, 42(14) :2839–2858, July 2004.

[29] J. Gunnarsson. *Symbolic Methods and Tools for Discrete Event Dynamic Systems*. PhD thesis, Linköping University, 1997.

[30] D. Harel. Statecharts : A visual formalism for complex systems. *Science of Computer Programming*, 8(3) :231–274, June 1987.

[31] G. Hoffmann and H. Wong-Toi. Symbolic synthesis of supervisory controllers. In *Proc. of 1992 American control Conference, Chicago,Il,USA*, pages 2789–2793, 1992.

[32] L. E. Holloway, X. Guan, and L. Zhang. A generalization of state avoaidance policies for controlled Petri nets. *IEEE Transactions on Automatic Control*, 41(6) :804–816, June 1996.

[33] L. E. Holloway and B. H. Krogh. Synthesis of feedback control logic for a class of controlled Petri nets. *IEEE Transactions on Automatic Control*, 35(5) :514–523, May 1990.

[34] L.E. Holloway, B.H. Krogh, and A. Giua. A survey of Petri net methods for controlled discrete event systems. *Discrete Event Dynamic Systems : Theory and Application*, 7 :151–190, 1997.

[35] J. E. Hopcroft and J. D. Ullman. *Introduction to automata theory, Languages, and computation*. Addison-wesley, Reading, MA., 1979.

[36] S. Jiang and R. Kumar. Decentralized control of discrete event systems with specializations to local control and concurrent systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 30(5) :653–660, October 2000.

[37] T. Jéron. *Contribution à la génération automatique de tests pour les systèmes réactifs*. habilitation à diriger des recherches, Université de Rennes 1, 2004.

[38] R. Kumar. *Supervisory synthesis Techniques for discrete event dynamical systems*. PhD thesis, University of Texas, 1991.

[39] R. Kumar, V. Garg, and Marcus S.I. On controllability and normality of discrete event dynamical systems. *Systems & Control Letters*, 17 :157–169, 1991.

[40] O. Kushnarenko and S. S. Pinchinat. Intensional approachs for symbolic methods. In *Electronic Notes in Theoretical Computer Science*, volume 18, 1998.

[41] C. lakos and L. Petrucci. Modular analysis of systems composed of semiautonomous subsystems. In *Proc. of fourth International Conference on Application of Concurrency to System Design*, pages 185–194, June 2004.

[42] R.J. Leduc. Plc implementation of a DES supervisor for manufacturing testbed : An implementation perspective. Master's thesis, Dept. of Elc. and Comp. Eng. University of Toronto, 1996.

[43] R.J. Leduc. *Hierarchical Interface Based Supervisory Control*. PhD thesis, Dept. of Elec. & Comp. Engrg., Univ. of Toronto, 2002.

[44] R.J. Leduc, W.M. Wonham, and M. Lawford. Hierarchical interface-based supervisory control : Parallel case. In *Proc. of the 39th Allerton Conf. on Comm., Contr., and Comp.*, pages 386–395, October 2001.

[45] Y. Li. *Control of vector discrete-events sytems*. PhD thesis, university of Toronto, July 1991.

[46] Y. Li and W.M. Wohnam. Control of discrete-event systems-part i : the base model. *IEEE Trans. Automatic Control*, 38(8) :1214–1227, August 1993.

[47] Y. Li and W.M. Wohnam. Control of discrete-event systems-part ii : controller synthesis. *IEEE Trans. Automatic Control*, 39(3) :512–531, March 1994.

[48] F. Lin and W. M. Wonham. Decentralized supervisory control of discrete event systems. *Information Sciences*, 44 :199–224, 1988.

[49] Y. Lin and W. M. Wonham. On observability of discrete-event systems. *Information Sciences*, 44(3) :173–198, 1988.

[50] C. Ma. *Non blocking supervisory control of state tree structures*. PhD thesis, Univeristy of Toronto, February 2004.

[51] H. Marchand, P. Bournai, M. Le Borgne, and P. Le Guernic. Synthesis of discrete-event controllers based on the signal environment. *Discrete Event Dynamic System : Theory and Applications*, 10(4) :347–368, October 2000.

[52] H. Marchand and B. Gaudin. Supervisory control problems of hierarchical finite state machines. In *41th IEEE Conference on Decision and Control*, Las Vegas, USA, December 2002.

[53] R.S. Minhas. *Complexity Reduction in Discrete Event Systems*. PhD thesis, Univeristy of Toronto, September 2002.

[54] R.S. Minhas and W.M. Wonham. Online supervision of discrete event systems. In *Proceedings of American Control Conference*, June 2003.

[55] P. J. Ramadge and W. M. Wonham. Supervision of discrete event processes. In *Proc. of 21st IEEE Conf. Decision and Control*, pages 1228–1229, Orlando, FL, December 1982.

[56] P. J. Ramadge and W. M. Wonham. Modular supervisory control of discrete event systems. In A. Bensoussan and J. L. Lions, editors, *Proc. of 7th Int. Conf. Analysis and Optimization of Syst.*, volume 83 of *LNCIS*, pages 202–214, Antibes, France, June 1986. Springer-Verlag , Berlin, Germany.

[57] P. J. Ramadge and W. M. Wonham. Modular supervisory control of discrete event systems. In A. Bensoussan and J. L. Lions, editors, *Proc. of 7th Int. Conf. Analysis and Optimization of Syst.*, volume 83 of *LNCIS*, pages 202–214, Antibes, France, June 1986. Springer-Verlag , Berlin, Germany.

[58] P. J. Ramadge and W. M. Wonham. Modular feedback logic for discrete event systems. *SIAM J. Control Optim.*, 25(5) :1202–1218, September 1987.

[59] P. J. Ramadge and W. M. Wonham. The control of discrete event systems. *Proceedings of the IEEE ; Special issue on Dynamics of Discrete Event Systems*, 77(1) :81–98, 1989.

[60] K. Rohloff and S. Lafortune. The control and verification of similar agents operating in a broadcast network environment. In *42nd IEEE Conference on Decision and Control*, Hawaii, USA, December 2003.

[61] I. Romanovski and P. Caines. Multi-agent product systems : analysis, synthesis and control. In *Proc of 6th Workshop on Discrete Event Systems, WODES 2002*, Zaragoza, Spain, October 2002.

[62] K. Rudie and W.M. Wonham. The infimal prefix closed and observable superlanguage of a given language. *Systems and Control Letters*, 15(5) :361–371, 1990.

[63] K. Rudie and W.M. Wonham. Think globally, act locally : decentralized supervisory control. *IEEE Transaction on Automatic Control*, 31(11) :1692–1708, November 1992.

[64] E. Rutten and H. Marchand. Task-level programming for control systems using discrete control synthesis. Research Report 4389, INRIA, February 2002.

[65] Sigali. Tool box for controller synthesis of reactive systems available at http ://www.irisa.fr/vertecs/logiciels/sigali.html.

[66] R. Su and W. Wonham. Supervisory reduction for discrete event systems. *Discrete Event Dynamic System : Theory and Applications*, 14(1) :31–53, January 2004.

[67] Supremica. A tool for verification and synthesis of discrete event supervisors available at http ://www.supremica.org/.

[68] S. Takai and T. Ushio. Effective computation of an $l_m(g)$-closed, controllable, and observable sublanguage arising in supervisory control. In *Proc of 6th Workshop on Discrete Event Systems, WODES 2002*, Zaragosa, Spain, October 2002.

[69] S. Takai and T. Ushio. Supervisory control of a class of concurrent discrete event systems. In *ATPN-Workshop on Discrete Event Systems Control*, June 2003.

[70] S. Takai, T. Ushio, and S. Kodama. Static-state feedback control of discrete-event systems under partial observation. *IEEE Transactions on Automatic Control*, 40(11) :1950–1954, 1995.

[71] #TCT. Various tools for the synthesis of discrete event supervisors available at http ://www.control.utoronto.ca/people/profs/wonham/wonham.html.

[72] J. N. Tsitsiklis. On the control of discrete event dynamical systems. *Mathematics of Control Signals and Systems*, 2(2) :95–107, 1989.

[73] UKDES. A graphical software tool for the design, analysis & control of discrete event systems. available at http ://clue.eng.iastate.edu/ rkumar/, section software.

[74] UMDES-LIB. Des group, university of michigan, ann arbor, mi, usa, available at http ://www.eecs.umich.edu/umdes/projects.html#lib, section umdes software library.

[75] T. Ushio. On controllable predicates and languages in discrete-event systems. In *Proc. of the* 28^{th} *Conference on Decision and Control*, pages 123–124, Tampa, Floride, December 1989.

[76] A. Vaz and W. Wonham. On supervisor reduction in discrete-event systems. *International Journal Control*, 44(2) :475–491, 1986.

[77] Y. Willner and M. Heymann. Supervisory control of concurrent discrete-event systems. *International Journal of Control*, 54(5) :1143–1169, 1991.

[78] K. C. Wong and W. M. Wonham. Hierarchical control of discrete-event systems. *Discrete Event Dynamic Systems*, 6 :241–273, 1996.

[79] W. M. Wonham. Notes on control of discrete-event systems. Technical Report ECE 1636F/1637S, Department of Electrical and Computer EngineeringUnivertsity of Toronto, July 2003.

[80] W. M. Wonham and P. J. Ramadge. Modular supervisory control of discrete event systems. *Mathematics of Control Signals and Systems*, 1 :13–30, 1988.

[81] K. Yamalidou, J. Moody, M. Lemmon, and P. Antsaklis. Feedback control of Petri nets based on place invariants. *Automatica*, 32(1) :15–28, 1996.

[82] Z.H. Zhang and W.M. Wonham. Stct : An efficient algorithm for supervisory control design. In *Symposium on Supervisory Control of Discrete Event Systems (SCODES2001)*, Paris (France), July 2001.

[83] H. Zhong and W.M. Wonham. On the consistency of hierarchical supervision in discrete-event systems. *IEEE Transactions on Automatic Control*, 35(10) :1125–1134, October 1990.

[84] K. Åkesson, Flordal, and M. Fabian. Exploiting modularity for synthesis and verification of supervisors. In *Proc. of the IFAC*, barcelona, Spain, July 2002.

Résumé :

L'étude concerne le contrôle de systèmes critiques, pour lesquels la correction est primordiale. Dans cette optique, la théorie du contrôle des systèmes à événements discrets, introduite par Ramadge et Wonham, a pour objectif de garantir par construction que le système respecte certaines propriétés. Toutefois, les méthodes proposées à ce jour souffrent d'un problème d'efficacité lorsque le système à contrôler est complexe. Cet obstacle constitue un frein à la propagation des techniques de synthèse pour des systèmes concrets. Les systèmes complexes sont le plus souvent spécifiés de manière compositionnelle. Pour pallier au problème d'explosion combinatoire induite par la composition, il semble alors intéressant de déterminer des méthodes propres à ces systèmes en tirant parti de leur structure, i.e. en effectuant autant que possible des calculs sur les sous systèmes, plutôt que sur le système global. Dans ce cadre, deux approches sont abordées.

La première partie est basée sur une approche langage. Nous présentons une méthodologie permettant de synthétiser un superviseur assurant l'objectif de contrôle en tirant parti de la structure du système. Des approximations du système sont dérivées à partir des sous-systèmes qui le composent, et une propriété appelée contrôlabilité partielle, devant être vérifiée par l'objectif de contrôle sur ces approximations, est introduite. Assurer la contrôlabilité partielle de l'objectif sur chacune des approximations permet, sous certaines hypothèses, d'en déduire un superviseur maximal assurant l'objectif de contrôle. Les conditions exhibées, portant sur l'objectif, s'avèrent moins restrictives que celles décrites jusqu'ici sous les mêmes hypothèses.

La deuxième partie est basée sur une approche à états. Le problème posé est de calculer un superviseur assurant l'invariance d'un ensemble d'états en décomposant le calcul sur chacun des sous systèmes et en regroupant les calculs réalisés localement pour inférer un superviseur centralisé (en reportant certains calculs à l'exécution). La structure particulière du superviseur ainsi obtenu permet non seulement de limiter ces calculs, mais aussi de détecter efficacement les états bloquants du système contrôlé. Nous étendons par la suite ces résultats au cadre des systèmes à événements discrets modélisés par des automates hiérarchiques à deux niveaux. Le haut niveau permet d'organiser le séquencement des systèmes de bas niveau, alors que les systèmes de bas niveau sont concurrents et modélisent des tâches s'exécutant en parallèle.

Mots clés : systèmes à événements discrets, synthèse de contrôleurs, systèmes concurrents, systèmes hiérarchiques.