Systèmes à événements discrets
Synthèse d'un superviseur

Ines Klimann

mémoire de DEA

sous la direction de J.E. Pin et P. Weil

Résumé

Les systèmes à événements discrets apparaissent dans beaucoup de situations (production manufacturière, logistique, ...). Nous allons tenter de faire une étude de ces systèmes dans un cas réduit : quand on peut les représenter par des automates *finis*. Nous allons voir que dans ce cas, on peut connaître le comportement possible de tels systèmes en ne contrôlant qu'une partie des événements.

Table des matières

1 Introduction

Un système à événements discrets est un système qui évolue dans le temps à la suite d'événements physiques. Par exemple, un événement peut être le départ ou l'arrivée d'un client dans une queue, le fait qu'une machine tombe en panne,...

L'étude des systèmes à événements discrets ont des applications dans beaucoup de domaines différents (vérification de programmes, production manufacturière, logistique,...). Ceux-ci peuvent être modélisés de diverses manières. Nous allons tenter dans ce mémoire de décrire la modélisation par des automates (introduite par W.M. Wonham et P.J. Ramadge [RW87, RW89, WR87]) et nous nous intéresserons plus particulièrement aux superviseurs.

Dans la partie 2.1, nous rappellerons brièvement le vocabulaire de base lié aux automates et aux langages. Dans la partie 3, nous introduirons la notion de superviseur et de sous-langage contrôlable maximal. Nous donnerons alors deux algorithmes de calcul : le premier dans la section 3.4.1 qui correspond à l'approche intuitive et historique et qui apparaît dans [Gau95], le deuxième dans la section 3.4.2 qui est moins intuitif mais plus rapide et qui apparaît dans [BGK+90].

Enfin, nous montrerons dans la section 4 une approche logique pour la résolution de certains problèmes de supervision (intersection de langages, ...). Le vocabulaire utilisé dans cette partie a été traduit par mes soins à partir de travaux en Anglais ([WR88]), il se peut donc que cela ne corresponde pas à la terminologie consacrée.

Par rapport aux références données, j'ai modifié quelques démonstrations (théorème 2, exemple de la p.20 et théorème 6) et j'en ai explicitées d'autres (corollaire 1, lemme 1 et égalités p.17). Une partie de ces modifications est due au fait que les définitions de [Gau95] et celles de [BGK+90] ne sont pas exactement les mêmes, il a donc fallu adapter les résultats (en ce qui concerne les définitions, j'ai pris systématiquement celles de S. Gaubert [Gau95]).

*

* *

2 Principales notions utilisées

2.1 Rappels sur les automates et les langages

Cette partie est là principalement pour fixer les notations. Pour le vocabulaire manquant et les démonstrations, on pourra se reporter à [Eil74]. En particulier, on supposera connus la notion de langage rationnel et le

Théorème 1 (Kleene) *Les langages reconnaissables sont les langages rationnels.*

Notations 1 *Si A est un ensemble quelconque, on notera $\mathcal{P}(A)$ l'ensemble des parties de A et A^n l'ensemble des n-uplets de A.*

Exemple —
$\mathcal{P}(\Sigma^*)$ est l'ensemble des langages sur l'alphabet Σ et $\mathcal{P}(\Sigma^*)^n$ l'ensemble des n-uplets de langages sur Σ.

Définition 1 *Soit u et w, deux mots sur un même alphabet.*

On dira que u est préfixe *de w, et on notera $u \leq w$, s'il existe un mot v (éventuellement vide) tel que $w = uv$.*

Définition 2 *Si $L \subset \Sigma^*$ est un langage sur Σ, on appellera* clôture préfixielle *de L, notée $\overline{L}$, le plus petit langage contenant tous les préfixes des mots de L.*

$$\overline{L} = \{u \in \Sigma^* \mid \exists w \in L, u \leq w\}$$

Par la suite, on dira simplement qu'un langage est fermé *(ou* préfixiel*), sans préciser que c'est pour la fermeture préfixielle ($L = \overline{L}$).*

Exemple —
Si $\Sigma = \{a, b\}$ et $L = \{aba, bab\}$, alors $\overline{L} = \{a, b, ab, ba, aba, bab\}$.

Définition 3 *On appelle* automate (déterministe) *sur un alphabet Σ la donnée du quintuplet*

$$\mathcal{A} = (Q, \Sigma, \delta, q_0, Q_a),$$

*où Q est l'ensemble des états, $q_0 \in Q$ l'état initial, $\delta : \Sigma \times Q \to Q$ une fonction (*éventuellement partielle[1]*) appelée* fonction de transition*, et $Q_a \subset$*

1. Par *fonction partielle* $\varphi : A \to B$, on entend que φ peut n'être définie que sur un sous-ensemble strict de A.

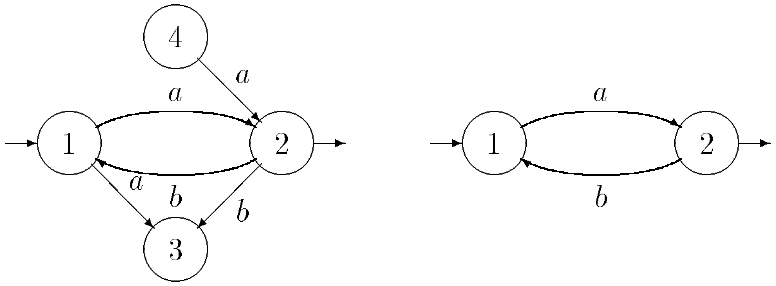

FIG. 1 – *Représentation graphique d'un automate et de son automate émondé*

Q l'ensemble des états d'acceptation *(ou* finaux[2]*). On peut éventuellement sous-entendre l'alphabet s'il n'y a pas de confusion possible et représenter alors ce même automate par le quadruplet*

$$\mathcal{A} = (Q, \delta, q_0, Q_a).$$

On dira que l'automate est complet *si la fonction de transition δ est définie sur* tout $\Sigma \times Q$*.*

Exemple —
$\Sigma = \{a, b\}$, $Q = \{1, 2\}$, $q_0 = 1$ et $Q_a = \{2\}$. Considérons la fonction de transition donnée par le tableau suivant, appelé *représentation matricielle* de l'automate :

$\nearrow$	1	2	3	4
1		a	a	
2	b		b	
3				
4		a		

On peut alors représenter graphiquement l'automate correspondant, comme dans la figure 1.

On a l'habitude de représenter les automates finis directement par ce type de graphiques, où l'état initial est représenté par une flèche entrante et les

2. En théorie des automates, on a l'habitude de dire états finaux, cependant, en ce qui concerne les systèmes à événements discrets, il est préférable de parler d'états d'acceptation, ce qui ne sous-entend aucun caractère final, c'est plutôt des états que l'on veut distinguer des autres.

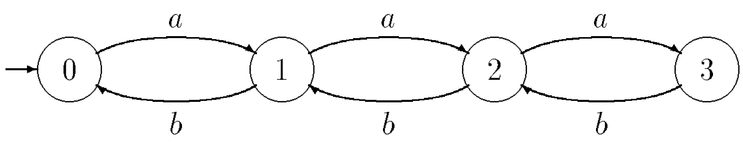

FIG. 2 – *Magasin de taille trois*

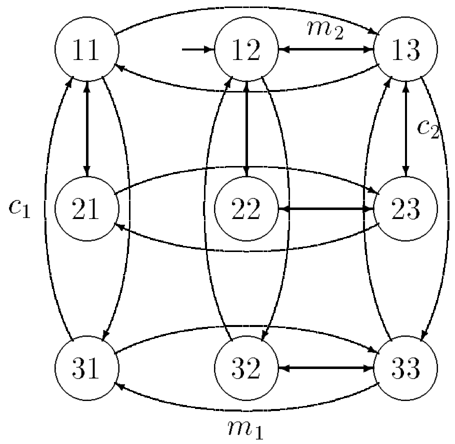

FIG. 3 – *Déplacements possibles d'un chat et d'une souris*

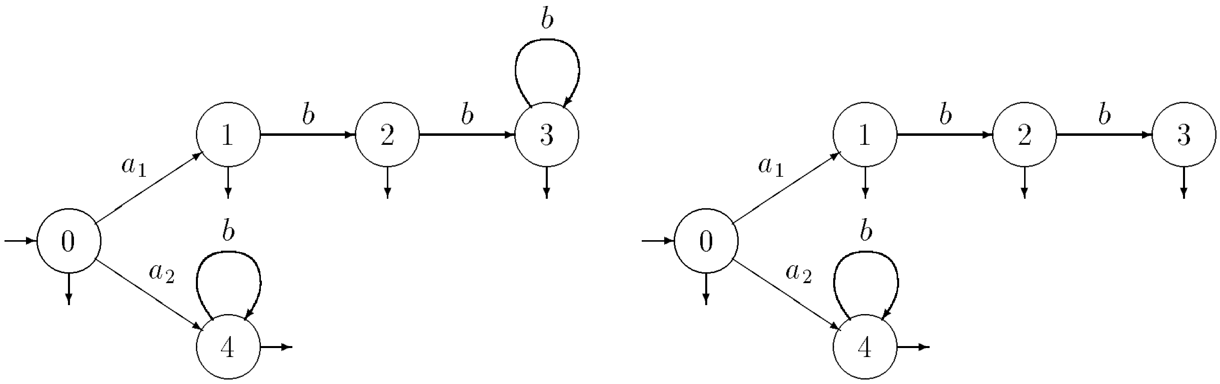

FIG. 4 – *automates $\mathcal{A}$ (à gauche) et $\mathcal{B}$ (à droite)*

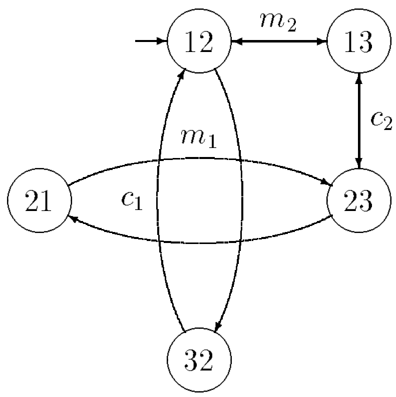

FIG. 5 – *Comportement admissible du chat et de la souris*

états finaux par des flèches sortantes, si l'état initial est aussi final, on le note par une flèche double.

On étend la fonction de transition δ à $\Sigma^* \times Q$ en posant :

$$\delta(e, q) = q, \quad \delta(w\sigma, q) = \delta(\sigma, \delta(w, q)),$$

où e désigne le mot vide. Dans l'exemple précédent, on a :

$$\delta(ba, 2) = \delta(a, \delta(b, 2)) = \delta(a, 1) = 2.$$

Dans la suite, on écrira $\delta(w, q)!$ pour "$\delta(w, q)$ est défini", ce qui graphiquement correspond à l'existence d'un chemin partant de q et portant les lettres de w comme labels successifs.

On dira qu'un mot w est *reconnu* par l'automate $\mathcal{A}$ si $\delta(w, q_0) \in Q_a$. L'ensemble

$$\mathcal{L}(\mathcal{A}) = \{w \in \Sigma^* \mid \delta(w, q_0) \in Q_a\}$$

est appelé *langage reconnu* par l'automate $\mathcal{A}$.

On peut aussi s'intéresser aux trajectoires, quel que soit leur point d'arrivée. Soit $\overline{\mathcal{A}} = (Q, \delta, q_0, Q)$, l'automate obtenu à partir de $\mathcal{A}$ en rendant tous les états finaux. On appellera alors *comportement* de l'automate $\mathcal{A}$, noté $\overline{\mathcal{L}}(\mathcal{A})$, le langage reconnu par l'automate $\overline{\mathcal{A}}$:

$$\overline{\mathcal{L}}(\mathcal{A}) \stackrel{\text{def}}{=} \mathcal{L}(\overline{\mathcal{A}}) = \{w \in \Sigma^* \mid \delta(w, q_0)!\}.$$

Exemple —

Reprenons l'exemple précédent. On a :

$$\mathcal{L}(\mathcal{A}) = \{(ab)^n, n \in \mathbb{N}\} \text{ et } \overline{\mathcal{L}}(\mathcal{A}) = \{(ab)^n, (ab)^n a, n \in \mathbb{N}\}.$$

Définition 4 *Pour tout langage rationnel L, il existe un unique automate complet et déterministe minimal par son nombre d'états qui reconnaît ce langage. On l'appelle l'*automate minimal *de L et on note $||L||$ son nombre d'états. $||L||$ est appelé* taille *du langage L. (cf. [HU79] pour la démonstration.)*

Notations 2 *Si w est un mot sur l'alphabet Σ, on notera $|w|$, la* longueur *de ce mot.*

Si L est un langage sur Σ, on notera L^c son complémentaire *dans Σ^*.*

Définition 5 *Si $\mathcal{A} = (Q, \delta, q_0, Q_a)$ est un automate et $q \in Q$ un état, on dira que*

q est accessible *s'il existe un chemin partant de q_0 et allant à q,*

q est co-accessible *s'il existe un chemin partant de q et allant à un état d'acceptation.*

On note Q_{ac} l'ensemble des états accessibles et Q_{co} l'ensemble des états co-accessibles.

On dira qu'un automate $\mathcal{A}$ est accessible *(resp.* co-accessible*) si tous ses états sont accessibles (resp. co-accessibles).*

On dira qu'un automate $\mathcal{A}$ est émondé *s'il est accessible et co-accessible.*

Définition 6 *Soit $\mathcal{A} = (Q, \delta, q_0, Q_a)$, un automate. Notons*

$$Q_{em} = Q_{ac} \cap Q_{co} \text{ et } \delta_{em} = \delta|_{\Sigma \times Q_{em}}.$$

Il existe alors un automate émondé particulier qui reconnaît le même langage $\mathcal{L}(\mathcal{A})$, noté $Em(\mathcal{A})$, défini par

$$Em(\mathcal{A}) = (Q_{em}, \delta_{em}, q_0, Q_a \cap Q_{em}).$$

(De même, on peut définir $Ac(\mathcal{A}) = (Q_{ac}, \delta_{ac}, q_0, Q_a \cap Q_{ac})$, où $\delta_{ac} = \delta|_{\Sigma \times Q_{ac}}$.)

Exemple —

Reprenons toujours le même exemple de la page 4. On a $Q_{ac} = \{1, 2, 3\}$ et $Q_{co} = \{1, 2, 4\}$, donc : $Q_{em} = \{1, 2\}$ et on obtient l'automate émondé de représentation matricielle

$$\begin{array}{c|cc} \nearrow & 1 & 2 \\ \hline 1 & & a \\ 2 & b & \end{array}$$

(voir figure 1)

Proposition 1 *Si $\mathcal{A}$ est co-accessible, alors*

$$\overline{\mathcal{L}}(\mathcal{A}) = \overline{\mathcal{L}(\mathcal{A})}.$$

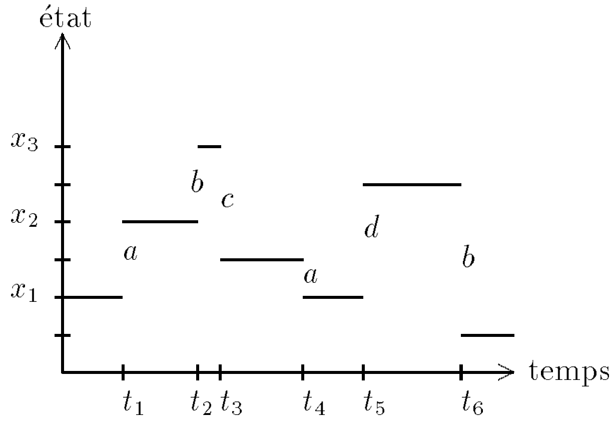

FIG. 6 – *Trajectoire d'un système à événements discrets*

2.2 Définition intuitive des systèmes à événements discrets

Avant de donner une liste d'exemples plus précis, donnons une idée générale de ce qu'est un système à événements discrets.

Un système à événements discrets est un système qui peut être caractérisé par son état. Celui-ci varie dans un espace discret. Le passage d'un état à un autre a lieu à la suite de l'exécution d'un événement. On suppose que dans un état donné l'exécution d'un événement est non ambiguë, c'est-à-dire qu'elle mène à un autre état *sans choix possible*. On ne s'occupe pas de savoir pourquoi ou comment les événements s'exécutent. On symbolise les événements par des lettres. Une suite finie d'événements s'appelle une tâche.

Dans l'exemple de la figure 6, le système est à l'état x_1 jusqu'à l'instant t_1 ; en t_1 se produit l'événement a et le système passe ̀l'état x_2 jusqu'en t_2, etc.

Les divers domaines dans lesquels apparaissent des systèmes à événements discrets et les divers aspects de leur comportement dominant dans chacun de ces domaines ont conduit à différents modèles. Une des simplifications habituelles est d'ignorer les temps d'apparition des événements et de ne considérer que l'ordre dans lequel ils apparaissent. C'est ce que nous allons faire ici.

Dans l'exemple de la figure 6, on peut donc réduire la trajectoire considérée au mot $abcadb\ldots$. On utilisera alors indifféremment les mots *lettre* et *événement* d'une part et *mot* et *tâche* d'autre part. On peut donc représenter le système à événements discrets par un automate déterministe, grâce à l'hypothèse faite sur la non ambiguïté de ses transitions.

Donnons quelques exemples de systèmes à événements discrets.

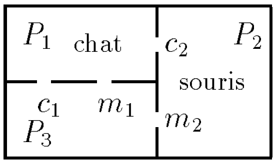

FIG. 7 – *Chat et souris*

Exemple — [Magasin de taille finie]

On considère un stock ayant une capacité de trois pièces. On a les deux événements

a : une pièce entre dans le stock

b : une pièce sort du stock

Et l'on considère que le stock est vide au départ.

Ce système est représenté par l'automate de la figure 2, où le nom des états correspond au nombre de pièces en stock.

Exemple — [Chat et souris]

On considère un chat et une souris qui se déplacent de façon imprévisible dans un ensemble de pièces communiquant par certaines trappes (cf. la figure 7). Chaque trappe est réservée à l'usage exclusif d'un seul des deux animaux : le chat peut emprunter les trappes c_1 et c_2 et la souris les trappes m_1 et m_2. On note c_1 l'événement "le chat passe par la trappe c_1", etc. L'ensemble des comportements possibles du chat et de la souris est représenté par l'automate de la figure 3, où l'état (i, j) signifie que le chat est dans la pièce i et la souris dans la pièce j. Par soucis de lisibilité de la figure, on n'a pas mis tous les labels : 2 flèches de même nature (courbe ou droite) et de même direction (horizontale ou verticale) portent bien sûr le même label.

On peut supposer que l'on ne désire pas que le chat mange la souris, une condition suffisante évidente est alors que les deux animaux ne soient jamais dans la même pièce. Grâce à la théorie développée dans les paragraphes suivants nous pourrons voir quelles sont les trappes à fermer et lesquelles on peut laisser ouvertes.

2.3 Notions liées aux algorithmes

Définition 7 *Soit $\mathcal{O} : \mathcal{P}(\Sigma^*)^n \to \mathcal{P}(\Sigma^*)$, un opérateur qui préserve la rationalité (i.e. si $L_1, L_2, \cdots L_n$ sont rationnels, $\mathcal{O}(L_1, L_2, \cdots L_n)$ aussi), on dira que l'opérateur $\mathcal{O}$ est* effectivement calculable *si on peut construire un algorithme (i.e. une méthode de calcul systématique) ayant $(L_1, L_2, \cdots L_n)$*

comme entrée (chaque langage étant donné par un automate) et un automate fini $\mathcal{A}$ reconnaissant $\mathcal{O}(L_1, L_2, \cdots L_n)$ en sortie.

Exemple — La fermeture préfixielle, le passage au complémentaire, l'union et l'intersection préservent la rationalité et sont effectivement calculables.

Proposition 2 *Soit deux automates finis $\mathcal{A}$ et $\mathcal{B}$ reconnaissant respectivement les langages K et L et de tailles respectives m et n. (La taille d'un automate est son nombre d'états.)*

On peut construire un automate reconnaissant K^c en un temps $\mathcal{O}(m)$ (on complète $\mathcal{A}$ en $\mathcal{O}(m)$, puis on prend le complémentaire de l'ensemble des états finaux, en un temps constant).

On peut construire un automate reconnaissant $K \cap L$ en un temps $\mathcal{O}(mn)$ (c'est démontré p.28, au cours de la démonstration du lemme 7). De même pour le langage $K - L$.

*

* *

3 Automates et systèmes à événements discrets

3.1 Supervision d'un automate

Soit Σ, l'ensemble des événements. On peut considérer la partition suivante :

$$\Sigma = \Sigma_{in} \cup \Sigma_c,$$

où Σ_{in} désigne l'ensemble des événements incontrôlables (une machine tombe en panne, un train déraille,...) et Σ_c l'ensemble des événements que l'on peut interdire ou autoriser (l'entrée en gare d'un train grâce à un feu rouge,...).

Définition 8 *On appelle* superviseur *une application*

$$f : \overline{\mathcal{L}}(\mathcal{A}) \to \mathcal{P}(\Sigma),$$

telle que $\forall w \in \overline{\mathcal{L}}(\mathcal{A}), f(w) \supset \Sigma_{in}$.

Intuitivement, $f(w)$ est l'ensemble des événements autorisés connaissant les événements passés qui sont donnés par w. L'obligation $f(w) \supset \Sigma_{in}$ signifie que les événements incontrôlables sont autorisés en permanence (c'est-à-dire qu'ils peuvent être exécutés à tout moment).

Définition 9 *On appelle* comportement d'un automate $\mathcal{A}$ soumis au superviseur f *le langage préfixiel* $\overline{\mathcal{L}}_f(\mathcal{A})$ *défini par :*

$$\begin{array}{l} e \in \overline{\mathcal{L}}_f(\mathcal{A}) \\ \forall w \in \Sigma^*, a \in \Sigma,\ (wa \in \overline{\mathcal{L}}_f(\mathcal{A}) \Leftrightarrow w \in \overline{\mathcal{L}}_f(\mathcal{A}), a \in f(w) \text{ et } wa \in \overline{\mathcal{L}}(\mathcal{A})). \end{array}$$

Le langage contrôlé par f *est alors défini par :*

$$\mathcal{L}_f(\mathcal{A}) = \overline{\mathcal{L}}_f(\mathcal{A}) \cap \mathcal{L}(\mathcal{A}).$$

C'est l'ensemble des mots du comportement supervisé qui aboutissent à un état final.

On dira que l'automate supervisé est non bloquant *si*

$$\overline{\mathcal{L}_f(\mathcal{A})} = \overline{\mathcal{L}}_f(\mathcal{A}),$$

i.e. si toute tâche commencée par le système supervisé peut être complété en une tâche achevée (mot de $\mathcal{L}_f(\mathcal{A})$*).*

On peut rendre cette définition effective en introduisant un automate qui jouera le rôle du superviseur et que nous appellerons *automate contrôleur*.

Définition 10 *Soit $S = (X, \xi, x_0, X_a)$ un automate et $\phi : X \to \mathcal{P}(\Sigma)$ une application telle que $\phi(x) \supset \Sigma_{in}$ pour tout x de X.*

On dira que $\mathcal{S} = (S, \phi)$ réalise le superviseur f *si pour tout mot w de $\overline{\mathcal{L}}_f(\mathcal{A})$, on a*

$$f(w) = \phi(\xi(w, x_0)).$$

En fait le contrôleur $\mathcal{S}$ lit le même mot que l'automate à contrôler $\mathcal{A}$ et on connaît directement l'ensemble des événements autorisés $f(w)$ grâce à l'état du contrôleur $\mathcal{S}$.

De façon plus précise, nous allons construire un automate $\mathcal{S}/\mathcal{A}$ représentant $\mathcal{A}$ supervisé par $\mathcal{S}$.

On considère l'ensemble des états $X \times Q$, l'état initial (x_0, q_0) et l'ensemble des états finaux $X_a \times Q_a$. On définit la fonction de transition $\xi \times \delta$ par

$$\xi \times \delta(\sigma, (x, q)) \quad = \quad \begin{cases} (\xi(\sigma, x), \delta(\sigma, q)) & \text{si } \sigma \in \phi(x), \\ \text{non défini} & \text{sinon.} \end{cases}$$

On définit alors $\mathcal{S}/\mathcal{A}$ par

$$\mathcal{S}/\mathcal{A} = \mathrm{Ac}(X \times Q, \xi \times \delta, (x_0, q_0), X_a \times Q_a).$$

On a $\overline{\mathcal{L}}(\mathcal{S}/\mathcal{A}) = \overline{\mathcal{L}}_f(\mathcal{A})$, donc grâce à la proposition 1, on peut conclure que l'automate supervisé $\mathcal{A}$ est non bloquant si et seulement si $\mathcal{S}/\mathcal{A}$ est co-accessible.

A l'avenir, on pourra parler de superviseur pour désigner un couple $\mathcal{S} = (S, \phi)$ qui vérifie la construction précédente. Nous utiliserons cette notion dans le paragraphe 4.

3.2 Contrôlabilité

Il est alors naturel de se demander, pour un langage K donné à quelles conditions il existe un superviseur f tel que K soit le langage contrôlé par f (i.e. $\mathcal{L}_f(\mathcal{A}) = K$). Pour cela, on introduit la notion de contrôlabilité :

Définition 11 *Le langage $K \subset \Sigma^*$ est dit* contrôlable *si*

$$\overline{K}\Sigma_{in} \cap \overline{\mathcal{L}}(\mathcal{A}) \subset \overline{K}.$$

Intuitivement, cela signifie que les événements incontrôlables ne sont pas illégaux (ou encore que tout préfixe de K suivi d'un événement incontrôlable admissible par le comportement de $\mathcal{A}$ reste préfixe de K).

Remarque —

La notion de contrôlabilité dépend du choix de Σ_{in} et de $\mathcal{A}$. On pourra donc être amené à dire, s'il y a confusion possible, K est contrôlable *par rapport à* Σ_{in} ou *par rapport à* $\mathcal{A}$.

La proposition suivante justifie le terme de contrôlable :

Théorème 2 *Soit $\mathcal{A}$, un automate. Pour tout $K \subset \overline{\mathcal{L}}(\mathcal{A})$ non vide, les conditions suivantes sont équivalentes :*

1. *Il existe un superviseur f tel que $\overline{\mathcal{L}}_f(\mathcal{A}) = K$,*
2. *K est une partie préfixielle contrôlable.*

Démonstration :

- $1 \Rightarrow 2$

  $\overline{\mathcal{L}}_f(\mathcal{A})$ est une partie préfixielle par définition. De plus, $\overline{\mathcal{L}}_f(\mathcal{A}) \subset \overline{\mathcal{L}}(\mathcal{A})$, donc $\overline{\mathcal{L}}_f(\mathcal{A})$ est contrôlable.

- $2 \Rightarrow 1$

  Soit $\mathcal{B} = (P, \xi, p_0, P)$, un automate *émondé* reconnaissant K (K est préfixiel, on peut donc prendre tous les états finaux). Considérons les deux sous-ensembles suivants de Σ :

  $$\begin{array}{rcl} \Sigma_p^0 & = & \{\sigma \mid \exists w \in K, \xi(w, p_0) = p, w\sigma \in \overline{\mathcal{L}}(\mathcal{A}) \text{ et } w\sigma \notin K\} \\ \Sigma_p^1 & = & \{\sigma \mid \exists w \in K, \xi(w, p_0) = p, w\sigma \in K\} \end{array}$$

  On a $\Sigma_p^0 \cap \Sigma_p^1 = \emptyset$, en effet si l'on suppose que $\sigma \in \Sigma_p^0 \cap \Sigma_p^1$, avec

  $$\begin{array}{lll} w^0 \in K, & \xi(w^0, p_0) = p, & w^0\sigma \notin K \\ w^1 \in K, & \xi(w^1, p_0) = p, & w^1\sigma \in K; \end{array}$$

  alors d'une part $\xi(\sigma, p) = \xi(\sigma, \xi(w^0, p_0)) = \xi(w^0\sigma, p_0)$ qui n'est pas défini car $w^0\sigma \notin K$ et tous les états sont finaux, d'autre part $\xi(\sigma, p) = \xi(\sigma, \xi(w^1, p_0)) = \xi(w^1\sigma, p_0)$ qui est bien défini, ce qui est impossible. Intuitivement, cette disjonction est évidente puisque Σ_p^0 représente les événements qui sortent de la cible K et qu'il faut donc interdire et Σ_p^1 ceux que l'on peut autoriser.

Montrons que $\Sigma_p^0 \cap \Sigma_{in} = \emptyset$, par l'absurde :

Soit $\sigma \in \Sigma_p^0 \cap \Sigma_{in}$:

$$\exists w \in K \mid \xi(w, p_0) = p, w\sigma \in \overline{\mathcal{L}}(\mathcal{A}) \text{ et } w\sigma \notin K.$$

Or

$$w\sigma \in K\Sigma_{in} \cap \overline{\mathcal{L}}(\mathcal{A}) \subset K$$

car K est une partie préfixielle contrôlable. D'où contradiction, donc : $\Sigma_p^0 \cap \Sigma_{in} = \emptyset \Rightarrow \Sigma_p^0 \subset \Sigma_c$, car (Σ_{in}, Σ_c) est une partition de Σ.

Soit $\phi : P \to \mathcal{P}(\Sigma^*)$, une application vérifiant $\phi(p) \cap \Sigma_p^0 = \emptyset$ et $\phi(p) \supset \Sigma_p^1 \cup \Sigma_{in}$. On définit alors le superviseur f par $f(w) = \phi(\xi(w, p_0))$.

Montrons que $\overline{\mathcal{L}}_f(\mathcal{A}) = K$, par récurrence sur $|w|$ où w est un mot de $\overline{\mathcal{L}}_f(\mathcal{A})$.

Si $|w| = 0$, nécessairement $w = e$ et $e \in \overline{\mathcal{L}}_f(\mathcal{A})$ et $e \in K$ car K est préfixiel. Supposons, par hypothèse de récurrence, que

$$\forall w \text{ tel que } |w| = n-1, w \in \overline{\mathcal{L}}_f(\mathcal{A})) \Leftrightarrow w \in K.$$

- Soit $u \in \overline{\mathcal{L}}_f(\mathcal{A})$, de longueur $n > 0$.
  On peut écrire $u = w\sigma$, où $w \in \overline{\mathcal{L}}_f(\mathcal{A})$, $\sigma \in f(w)$ et $u = w\sigma \in \overline{\mathcal{L}}(\mathcal{A})$. Comme $w \in \overline{\mathcal{L}}_f(\mathcal{A})$ et $|w| = |u| - 1 = n - 1$, on peut appliquer l'hypothèse de récurrence, donc $w \in K$.
  Si $\sigma \in \Sigma_{in}$, alors : $u = w\sigma \in K\Sigma_{in} \cap \overline{\mathcal{L}}(\mathcal{A}) \subset K$.
  Si $\sigma \in \Sigma_c$:
  on a $\sigma \in f(w) = \phi(\xi(w, p_0))$, posons $p = \xi(w, p_0)$, alors $\sigma \in \phi(p)$, donc $\sigma \notin \Sigma_p^0$.
  On a donc $w \in K$, $\xi(w, p_0) = p$ et $w\sigma \in \overline{\mathcal{L}}(\mathcal{A})$. Comme $\sigma \notin \Sigma_p^0$, nécessairement $w\sigma = u \in K$. La première inclusion est donc démontrée.
- Soit $u \in K$, de longueur $n > 0$.
  On peut écrire $u = w\sigma$, où σ est une lettre. Alors $w \in \overline{K} = K$ et $|w| = |u| - 1 = n - 1$, donc par hypothèse de récurrence $w \in \overline{\mathcal{L}}_f(\mathcal{A})$. De plus $p = \xi(w, p_0)$ est bien défini car $w \in K$. On a donc :

  $$w \in K,\ \xi(w, p_0) = p \text{ et } w\sigma \in K,$$

  c'est-à-dire exactement $\sigma \in \Sigma_p^1$, or on vient de voir que $\Sigma_p^1 \subset \phi(p) = \phi(\xi(w, p_0)) = f(w)$, donc $w\sigma \in \overline{\mathcal{L}}_f(\mathcal{A})$, par construction de cet ensemble.

□

On remarque que la démonstration du théorème (2 $\Rightarrow$ 1) donne une construction de superviseur (même si le choix de ϕ est à faire : il suffit de prendre $\phi(p) = \Sigma_p^1 \cup \Sigma_{in}$).

Ce théorème permet d'énoncer le résultat suivant :

Corollaire 1 *Il existe un superviseur f tel que la clôture du langage reconnu $\mathcal{L} = \mathcal{L}(\mathcal{A})$ soit le langage contrôlé par f. Si l'automate est co-accessible, ce corollaire signifie donc que le comportement de l'automate est contrôlable.*

Démonstration :

$\mathcal{L}$ est contrôlable, donc $\overline{\mathcal{L}}$ aussi. De plus $\overline{\mathcal{L}}$ est préfixiel, donc d'après le théorème 2, il existe un superviseur f tel que $\overline{\mathcal{L}}$ soit le langage contrôlé par f.

□

On a un autre lemme qui ressemble au théorème 2, mais qui suppose moins d'hypothèses sur le langage K considéré :

Lemme 1 *Soit $\mathcal{A}$ un automate non bloquant (i.e. co-accessible).*

Pour tout $K \subset \mathcal{L}(\mathcal{A})$ non vide, il y a équivalence entre les deux propositions suivantes :

1. *Il existe un superviseur f tel que l'automate supervisé soit non bloquant (i.e. $\overline{\mathcal{L}_f(\mathcal{A})} = \overline{\mathcal{L}}_f(\mathcal{A})$) et $\mathcal{L}_f(\mathcal{A}) = K$.*

2. *K est contrôlable et $\overline{K} \cap \mathcal{L}(\mathcal{A}) = K$.*

Démonstration :

– 1 $\Rightarrow$ 2

On a :

$$\begin{aligned}
\overline{K} \cap \mathcal{L}(\mathcal{A}) &= \overline{\mathcal{L}_f(\mathcal{A})} \cap \mathcal{L}(\mathcal{A}) \\
&= \overline{\mathcal{L}}_f(\mathcal{A}) \cap \mathcal{L}(\mathcal{A}) \\
&= \mathcal{L}_f(\mathcal{A}) \\
&= K,
\end{aligned}$$

et :

$$\begin{aligned}\overline{K}\Sigma_{in} \cap \overline{\mathcal{L}}(\mathcal{A}) &= \overline{\mathcal{L}_f(\mathcal{A})}\Sigma_{in} \cap \overline{\mathcal{L}}(\mathcal{A}) \\ &= \overline{\mathcal{L}}_f(\mathcal{A})\Sigma_{in} \cap \overline{\mathcal{L}}(\mathcal{A}) \\ &\subset \overline{\mathcal{L}}_f(\mathcal{A}) \\ &= \overline{\mathcal{L}_f(\mathcal{A})} \\ &= \overline{K},\end{aligned}$$

donc K est contrôlable.

– $2 \Rightarrow 1$

$\overline{K}$ est préfixielle et contrôlable, d'après le théorème 2, il existe donc un superviseur f tel que

$$\overline{\mathcal{L}}_f(\mathcal{A}) = \overline{K},$$

on a alors :

$$K = \overline{K} \cap \mathcal{L}(\mathcal{A}) = \overline{\mathcal{L}}_f(\mathcal{A}) \cap \mathcal{L}(\mathcal{A}) = \mathcal{L}_f(\mathcal{A}),$$

et :

$$\overline{\mathcal{L}}_f(\mathcal{A}) = \overline{K} = \overline{\mathcal{L}_f(\mathcal{A})},$$

i.e. l'automate supervisé est non bloquant.

□

Remarque —

La définition de contrôlabilité nous permet d'affirmer directement que la réunion d'une famille de langages contrôlables est contrôlable.

En revanche, l'intersection d'une famille de langages contrôlables n'est pas nécessairement contrôlable. Si on impose la condition supplémentaire que tous les langages soient fermés, alors l'intersection est contrôlable.

Exemple —

Soit $\Sigma = \{a, b, c\}$, avec $\Sigma_c = \{b, c\}$. Les langages $K_1 = \{e, ab\}$ et $K_2 = \{e, ac\}$ sont contrôlables, mais leur intersection $K_1 \cap K_2 = \{e\}$ ne l'est pas puisque a est incontrôlable et $a \notin \overline{K_1 \cap K_2}$, alors que $\overline{K}_1 \cap \overline{K}_2 = \{e, a\}$ est contrôlable.

Cependant, pour deux langages fermés contrôlables donnés, il n'est pas évident, connaissant un superviseur pour chacun, de trouver un superviseur pour l'intersection. C'est à cette question que nous tenterons de répondre dans la partie 4 de ce mémoire.

3.3 Sous-langage contrôlable maximal

3.3.1 Définition et calculabilité

Soit $K \subset \overline{\mathcal{L}}(\mathcal{A})$. Même si K n'est pas contrôlable, ce qui arrive fréquemment, il existe une approximation contrôlable de K que nous allons définir ci-dessous.

Soit $C(K)$, l'ensemble des parties contrôlables de K, i.e.

$$C(K) = \{J \subset K \mid \overline{J}\Sigma_{in} \cap \overline{\mathcal{L}}(\mathcal{A}) \subset \overline{J}\}.$$

Cet ensemble n'est pas vide car $\emptyset$ est contrôlable et il est stable par union (immédiat sur la définition), il admet donc un plus grand élément, noté $K^{\uparrow}$. On appelle cet élément *sous-langage contrôlable maximal* de K. Il représente le plus grand sous-ensemble de tâches de K qui peuvent être contrôlées.

Le problème de cette définition, vient du fait de la borne sup, qui ne permet pas le calcul. Nous allons donner une caractérisation de $K^{\uparrow}$, par point fixe.

Définissons l'opérateur

$$\Omega : \mathcal{P}(\Sigma^*) \to \mathcal{P}(\Sigma^*) : J \mapsto \sup\{T \subset K \mid \overline{T}\Sigma_{in} \cap \overline{\mathcal{L}}(\mathcal{A}) \subset \overline{J}\}.$$

On a alors les égalités :

$$\Omega(J) = \{t \in K \mid \overline{t}\Sigma_{in} \cap \overline{\mathcal{L}}(\mathcal{A}) \subset \overline{J}\} = K \cap \sup\{T \subset \Sigma^* \mid \overline{T}\Sigma_{in} \cap \overline{\mathcal{L}}(\mathcal{A}) \subset \overline{J}\}.$$

Démonstration :

Notons

$$\begin{aligned}
\Omega_1(J) &= \sup\{T \subset K \mid \overline{T}\Sigma_{in} \cap \overline{\mathcal{L}}(\mathcal{A}) \subset \overline{J}\},\\
\Omega_2(J) &= \{t \in K \mid \overline{t}\Sigma_{in} \cap \overline{\mathcal{L}}(\mathcal{A}) \subset \overline{J}\} \text{ et}\\
\Omega_3(J) &= K \cap \sup\{T \subset \Sigma^* \mid \overline{T}\Sigma_{in} \cap \overline{\mathcal{L}}(\mathcal{A}) \subset \overline{J}\}.
\end{aligned}$$

(Les bornes sup ont bien un sens dans les deux cas puisqu'il s'agit d'ensembles contenant $\emptyset$ et stables par union.)

- $\Omega_1 \subset \Omega_2$:

  Soit $T \subset K$ tel que $\overline{T}\Sigma_{in} \cap \overline{\mathcal{L}}(\mathcal{A}) \subset \overline{J}$ et soit $t \in T$.

  On a : $\overline{t}\Sigma_{in} \cap \overline{\mathcal{L}}(\mathcal{A}) \subset \overline{T}\Sigma_{in} \cap \overline{\mathcal{L}}(\mathcal{A}) \subset \overline{J}$.

- $\Omega_2 \subset \Omega_3$:

  Soit $t \in K$ tel que $\overline{t}\Sigma_{in} \cap \overline{\mathcal{L}}(\mathcal{A}) \subset \overline{J}$.

  Si $T = \{t\}$, on a :

  $$T \subset K \text{ et } \overline{T}\Sigma_{in} \cap \overline{\mathcal{L}}(\mathcal{A}) \subset \overline{J}.$$

– $\Omega_3 \subset \Omega_1$:
Soit $t \in \Omega_3$:

$$t \in K \text{ et } (\exists T \subset \Sigma^* \mid \overline{T}\Sigma_{in} \cap \overline{\mathcal{L}}(\mathcal{A}) \subset \overline{J} \text{ et } t \in T).$$

Soit $T' = T \cap K$:

$$t \in T' \text{ et } \overline{T'}\Sigma_{in} \cap \overline{\mathcal{L}}(\mathcal{A}) \subset \overline{T}\Sigma_{in} \cap \overline{\mathcal{L}}(\mathcal{A}) \subset \overline{J}.$$

□

Proposition 3 *$K^{\uparrow}$ est le plus grand point fixe de l'opérateur Ω, i.e.*

$$K^{\uparrow} = \Omega(K^{\uparrow}) \textit{ et } (J = \Omega(J) \Rightarrow K^{\uparrow} \supset J).$$

Démonstration :

Soit f et g, les deux applications croissantes[3] suivantes :

$f \ : \ \mathcal{P}(\Sigma^*) \to \mathcal{P}(\Sigma^*) \ : \ J \mapsto \overline{J}\Sigma_{in} \cap \overline{\mathcal{L}}(\mathcal{A})$

$g \ : \ \mathcal{P}(\Sigma^*) \to \mathcal{P}(\Sigma^*) \ : \ J \mapsto \overline{J}$

On remarque que f préserve les sup quelconques.

Introduisons les opérateurs

$$C(K, J) = \{T \subset K \mid f(T) \subset g(J)\} \text{ et}$$

$$\omega(K, J) = \sup C(K, J).$$

On a la relation

$$\Omega(J) = \sup\{T \subset K \mid f(T) \subset g(J)\} = \sup C(K, J) = \omega(K, J).$$

Avec ces nouvelles notations, la proposition que l'on veut démontrer s'écrit donc :

$$K^{\uparrow} = \omega(K, K^{\uparrow}) \tag{1}$$

et

$$J = \omega(K, J) \Rightarrow J \subset K^{\uparrow} \tag{2}$$

3. Par *croissant*, on entend : $J_1 \subset J_2 \ \Rightarrow \ f(J_1) \subset f(J_2)$.

On a $K^\uparrow \subset K$ car $K^\uparrow \in C(K)$ et le fait que f préserve les sup nous permet d'écrire

$$f(K^\uparrow) = \sup_{J \in C(K)} f(J) \subset \sup_{J \in C(K)} g(J) \subset g(K^\uparrow)$$

car $\forall J \in C(K), f(J) \subset g(J)$ et g croît. Donc $K^\uparrow \in C(K, K^\uparrow)$.

$$\Rightarrow K^\uparrow \subset \omega(K, K^\uparrow).$$

Par ailleurs, pour tout J, on a $\omega(K, J) \in C(K, J)$, soit en spécialisant en $J = K^\uparrow$: $f(\omega(K, K^\uparrow)) \subset g(K^\uparrow)$. Comme g croît, on a la double inégalité

$$f(\omega(K, K^\uparrow)) \subset g(K^\uparrow) \subset g(\omega(K, K^\uparrow)),$$

d'où $\omega(K, K^\uparrow) \in C(K)$

$$\Rightarrow \omega(K, K^\uparrow) \subset K^\uparrow.$$

Par double inégalité, on obtient l'équation 1.

Pour l'équation 2, il suffit de remarquer que si J est tel que $J = \omega(K, J)$, alors $J \in C(K, J)$, d'où $f(J) \subset g(J)$, i.e. $J \in C(K)$, d'où le résultat.

$\square$

On a alors l'idée pour obtenir $K^\uparrow$ de procéder de façon usuelle par un argument de point fixe : on construit une suite $(K_j)_{j \geq 0}$ de langages en posant

$$K_0 = K \text{ et } K_{j+1} = \Omega(K_j).$$

La suite (K_j) décroît, en effet $K_1 \subset K = K_0$ par définition et Ω est croissant (immédiat grâce à la caractérisation Ω_2, p.17), donc $K_2 = \Omega(K_1) \subset \Omega(K_0) = K_1$ et on conclut par récurrence.

On a donc :

$$K_\infty = \lim_{j \infty} K_j = \bigcap_{j \geq 0} K_j.$$

Comme Ω est un opérateur croissant et $K^\uparrow = \Omega(K^\uparrow) \subset K = K_0$, on montre par récurrence que $K^\uparrow \subset K_j$, pour tout $j \geq 0$, d'où :

$$K^\uparrow \subset \bigcap_{j \geq 0} K_j = K_\infty.$$

Le problème est de savoir s'il y a égalité, ce qui n'est pas le cas en général, comme nous l'illustrons dans l'exemple suivant.

Exemple —

Considérons l'alphabet $\Sigma = \{a, b, c\}$ et l'ensemble des événements incontrôlables $\Sigma_{in} = \{a, c\}$. Soit $\mathcal{A}$, un automate tel que

$$\overline{\mathcal{L}}(\mathcal{A}) = \overline{ab + acc^*bc^*},$$

et K le langage (non rationnel[4]) suivant :

$$K = ab + a\{c^{n+1}bc^m \mid n \geq 0, 0 \leq m \leq n\}.$$

Construisons la suite $(K_j)_{j\geq 0}$ définie par

$$K_0 = K \text{ et } K_{j+1} = \Omega(K_j) = \{t \in K \mid \overline{t}\Sigma_{in} \cap \overline{\mathcal{L}}(\mathcal{A}) \subset \overline{K}_j\}.$$

Montrons, par récurrence sur j que

$$K_j = ab + ac^j\{c^{n+1}bc^m \mid n \geq 0, 0 \leq m \leq n\}.$$

Démonstration :

Pour $j = 0$, c'est immédiat car $K_0 = K$. Supposons le résultat vrai pour un certain j.

On a vu que $\Omega(K_j) = K_{j+1} \subset K_j$. Soit alors $t \in K_j$.

– Si $t = ab$, alors $\Sigma_{in} = \{a, c, aa, ac, aba, abc\}$, donc

$$\overline{t}\Sigma_{in} \cap \overline{\mathcal{L}}(\mathcal{A}) = \{a, ac\} \subset \overline{K}_j,$$

d'où $ab \in K_{j+1}$.

– Si $t = ac^jcb$, alors $tc \in \overline{t}\Sigma_{in} \cap \overline{\mathcal{L}}(\mathcal{A})$ et $tc \notin \overline{K}_j$, donc $ac^jcb \notin K_{j+1}$.

– Si $t = ac^jc^{n+1}bc^m$ pour certains $n \geq 1$ et $0 \leq m \leq n$, alors

$$\overline{t}\Sigma_{in} = \{a, c, aa, ac, ac^pa, ac^pc, ac^{j+n+1}ba, ac^{j+n+1}bc, ac^{j+n+1}bc^qa,$$
$$ac^{j+n+1}bc^{q+1} \mid 1 \leq p \leq j+n+1, 1 \leq q \leq m\},$$

donc

$$\overline{t}\Sigma_{in} \cap \overline{\mathcal{L}}(\mathcal{A}) = \{a, ac, ac^pc, ac^jc^{n+1}bc^qc \mid 1 \leq p \leq j+n+1, 0 \leq q \leq m\} \subset \overline{K}_j,$$

4. On montre ce résultat grâce au lemme de pompage (cf. [HU79]). Intuitivement, on peut dire qu'un automate n'a pas de mémoire.

donc $t \in K_{j+1}$.

$\square$

Montrons que dans le cas où l'automate $\mathcal{A}$ est fini et le langage K reconnaissable, l'opérateur Ω est effectivement calculable.

Définissons $H : \mathcal{P}(\Sigma^*) \to \mathcal{P}(\Sigma^*)$ par

$$H(L) := \sup\{T \subset L \mid T = \overline{T}\}$$

($H(L)$ est le plus grand sous-langage préfixiel de L).

Et pour $\Sigma_0 \subset \Sigma$ fixé, définissons $B_0 : \mathcal{P}(\Sigma^*) \to \mathcal{P}(\Sigma^*)$ par

$$B_0(L) := \{w \in \Sigma^* \mid \forall \sigma \in \Sigma_0, w\sigma \in L\}.$$

Lemme 2 *Les opérations H et B_0 préservent la rationalité et sont effectivement calculables.*

Démonstration :

Soit $\mathcal{A} = (Q, \Sigma, \delta, q_0, Q_a)$, un automate fini reconnaissant L.

- Si $q_0 \notin Q_a$, alors pour tout $T \subset L$ non vide, on a : $1 \in \overline{T}$ et $1 \notin T$, donc $H(L) = \emptyset$.

  Si $q_0 \in Q_a$, posons alors

  $$P = Q_a,\ p_0 = q_0,\ P_a = P,$$

  et définissons $\xi : \Sigma \times P \to P$ par

  $$\xi(\sigma, p) = \begin{cases} \delta(\sigma, p) & \text{si } \delta(\sigma, p) \in Q_a \\ \text{non défini} & \text{sinon} \end{cases}$$

  Soit $\mathcal{B} := (P, \Sigma, \xi, p_0, P)$

  On a : $w \in \mathcal{L}(\mathcal{B}) \Leftrightarrow \forall v \in \overline{w}, \delta(v, q_0) \in Q_a \Leftrightarrow \overline{w} \in \mathcal{L}(\mathcal{A}) \Leftrightarrow w \in H(L)$,

  donc H est effectivement calculable.

– Soit $R_a = \{q \in Q \mid \forall \sigma \in \Sigma_0, \delta(\sigma, q) \in Q_a\}$ et $\mathcal{C} = (q, \Sigma, \delta, q_0, R_a)$
  On a alors : $w \in \mathcal{L}(\mathcal{C}) \Leftrightarrow \forall \sigma \in \Sigma_0, w\sigma \in \mathcal{L}(\mathcal{A}) \Leftrightarrow w \in B_0(L)$, donc B_0 est effectivement calculable.

□

Lemme 3 *Pour tout langage $L \subset \Sigma^*$, on a*

$$\Omega(L) = L \cap H \cdot B_0(\overline{L} \cup \overline{\mathcal{L}}(\mathcal{A})^c)$$

Démonstration :
Soit $L \subset \Sigma^*$

$$\Omega(L) = \{w \in K \mid \overline{w}\Sigma_0 \cap \overline{\mathcal{L}}(\mathcal{A}) \subset \overline{L}\}$$

On a alors

$$\begin{aligned}
w \in \Omega(L) \quad & \Leftrightarrow \quad w \in K \text{ et } \overline{w}\Sigma_0 \cap \overline{\mathcal{L}}(\mathcal{A}) \subset \overline{L}, \\
& \Leftrightarrow \quad w \in K \text{ et } \overline{w}\Sigma_0 \subset \overline{L} \cup \overline{\mathcal{L}}(A)^c, \\
& \Leftrightarrow \quad w \in K \text{ et } \overline{w} \subset B_0(\overline{L} \cup \overline{\mathcal{L}}(\mathcal{A})^c), \\
& \Leftrightarrow \quad w \in K \text{ et } w \in H \cdot B_0(\overline{L} \cup \overline{\mathcal{L}}(\mathcal{A})^c), \\
& \Leftrightarrow \quad w \in K \cap H \cdot B_0(\overline{L} \cup \overline{\mathcal{L}}(\mathcal{A})^c).
\end{aligned}$$

□

Ces deux lemmes permettent d'énoncer le

Théorème 3 *Si l'automate $\mathcal{A}$ est fini et le langage K reconnaissable, l'opérateur Ω est effectivement calculable.*

Démonstration :
immédiate grâce aux lemmes 2 et 3 et au fait que la clôture préfixielle, le passage au complémentaire, l'union et l'intersection sont effectivement calculables.

□

3.3.2 Taille

On peut se demander quelle est la taille du sous-langage contrôlable maximal par rapport à la taille du langage de départ. C'est ce que nous faisons dans cette partie.

Théorème 4 *Supposons $\mathcal{A}$ fini et K régulier,*
alors la suite (K_j) définie par :

$$K_0 = K \text{ et } \forall j \geq 0, K_{j+1} = \Omega(K_j)$$

converge en un nombre fini d'itérations vers $K^{\uparrow}$.
En outre :

$$||K^{\uparrow}|| \leq ||\overline{\mathcal{L}}(\mathcal{A})|| \cdot ||K|| + 1.$$

Nous allons d'abord montrer quelques résultats préliminaires.

Définition 12 *Si s et t sont deux mots sur Σ et $L \subset \Sigma^*$, un langage, on dira que s est L-*équivalent *à t et l'on notera $s \equiv_L t$ si*

$$\{s' \in \Sigma^* \mid ss' \in L\} = \{t' \in \Sigma^* \mid tt' \in L\},$$

c'est-à-dire si s et t peuvent être complétés par les mêmes mots dans L.
L'algorithme de Nérode (cf. [HU79]) donne alors immédiatement :

$$||L|| = card(\Sigma^* / \equiv_L). \tag{3}$$

Lemme 4 *Soit $A, B, C \subset \Sigma^*$ trois langages rationnels tels que :*

$$\forall s, t \in \Sigma^*, (s, t \in \overline{C} \ \wedge \ s \equiv_A t \ \wedge \ s \equiv_B t) \Rightarrow s \equiv_C t,$$

alors :

$$||C|| \leq ||A|| \cdot ||B|| + 1.$$

Démonstration :

Pour toute relation d'équivalence π sur Σ^*, notons $||\pi||$ son nombre de classes d'équivalence (cette notation est justifiée par l'égalité 3),

$$||\pi|| = \mathrm{card}(\Sigma^*/\pi).$$

On dira que π est plus fine qu'une autre relation ρ si

$$\forall s, t \in \Sigma^*, \ (s \equiv t \mod \pi) \Rightarrow (s \equiv t \mod \rho),$$

dans ce cas, on a : $||\pi|| \geq ||\rho||$.

Soit π, la relation d'équivalence sur Σ^* définie par :

$$s \equiv t \mod \pi \Leftrightarrow (s \equiv_A t \text{ et } s \equiv_B t).$$

Cette définition donne immédiatement l'inégalité $||\pi|| \leq ||A|| \cdot ||B||$.

Notons π_C, la restriction de π à $\overline{C} \subset \Sigma^*$ (i.e. $s \equiv t \mod \pi_C \Leftrightarrow (s \in \overline{C} \wedge t \in \overline{C} \wedge s \equiv t \mod \pi.)$), on a :

$$||\pi_C|| \leq ||\pi||.$$

De même, on note ρ la relation $\equiv_C$ et ρ_C la restriction de ρ à $\overline{C}$.

π_C est plus fine que ρ_C par hypothèse, donc :

$$||\rho_C|| \leq ||\pi_C||.$$

D'où

$$||\rho_C|| \leq ||\pi|| \leq ||A|| \cdot ||B||.$$

Or $\Sigma^* - \overline{C} \subset \Sigma^*$ contient au maximum une classe d'équivalence modulo ρ, donc :

$$||\rho|| \leq ||\rho_C|| + 1,$$

c'est-à-dire, en tenant compte de l'inégalité précédente :

$$||C|| \leq ||A|| \cdot ||B|| + 1.$$

□

Remarque —

Si $m \geq 1$ est un entier, l'ensemble

$$\mathcal{L}_m = \{L \mid L \subset \Sigma^* \text{ et } ||L|| \leq m\}$$

est fini. C'est immédiat si l'on remarque que sur un alphabet fini, on ne peut construire qu'un nombre fini d'automates ayant $\leq m$ états.

Lemme 5 *Soit $(L_j)_{j \in \mathbb{N}}$, une suite de langages rationnels sur Σ tels que :*

1. $L_0 \supset L_1 \supset L_2 \supset \dots,$

2. $\exists m < \infty \mid \forall j, ||L_j|| \leq m,$

alors, la suite (L_j) est stationnaire, i.e.

$$\exists k \mid \forall j \geq k, L_j = L_k.$$

Démonstration :

En reprenant les notations de la remarque précédente, $L_j \in \mathcal{L}_m$, pour tout j. $\mathcal{L}_m$ étant fini, il existe $L \in \mathcal{L}_m$ tel que la suite (L_j) prenne une infinité de fois la valeur L. Le résultat est alors obtenu par monotonie (hypothèse 1).

□

Lemme 6 *Si (K_j) est la suite de langages définie dans le théorème, on a :*

$$\forall j \geq 0, ||K_j|| \leq ||\overline{\mathcal{L}}(\mathcal{A})|| \cdot ||K|| + 1.$$

Démonstration :

Montrons que

$$\underbrace{(s,t \in \overline{K}_j \ \wedge \ s \equiv_K t \ \wedge \ s \equiv_{\overline{\mathcal{L}}(\mathcal{A})} t)}_{(H_j)} \quad \Rightarrow \quad \underbrace{s \equiv t \mod K_j}_{(C_j)}, \qquad (\mathcal{P}_j)$$

par récurrence sur $j \geq 0$.

Si $j = 0$, $K = K_0$, c'est donc évident.

hypothèse de récurrence : $(\mathcal{P}_{j-1})$ est vraie.

alors :

Remarquons que si s et t vérifient (H_j), alors s et t vérifient (H_{j-1}) car $K_j \subset K_{j-1}$, donc s et t vérifient (C_{j-1}) par hypothèse de récurrence.

Soit $s, t \in \Sigma^*$ tels que (H_j) soit vrai. Soit $v \in \Sigma^*$ tel que $sv \in K_j$. Alors :

$$sv \in K \text{ et } \overline{sv}\Sigma_{in} \cap \overline{\mathcal{L}}(\mathcal{A}) \subset \overline{K}_{j-1}. \tag{4}$$

Comme $s \equiv_K t$, on a : $tv \in K$.

Soit t', un préfixe de tv.

- si t' est préfixe de t, on a :

$$t'\Sigma_{in} \cap \overline{\mathcal{L}}(\mathcal{A}) \subset \overline{K}_{j-1},$$

  car $t \in \overline{K}_j$.

– si $t' = tu$, où u est un préfixe (non vide) de v :
  Soit $\sigma \in \Sigma_{in}$ tel que $tu\sigma \in \overline{\mathcal{L}}(\mathcal{A})$, alors

$$s \equiv_{\overline{\mathcal{L}}(\mathcal{A})} t \Rightarrow su\sigma \in \overline{\mathcal{L}}(\mathcal{A}),$$

  or su est un préfixe de sv, donc : $su\sigma \in \overline{K}_{j-1}$ d'après l'équation 4, i.e.

$$\exists w \in \Sigma^* \mid sv\sigma w \in K_{j-1}.$$

  Par hypothèse de récurrence et grâce à la remarque faite en début de démonstration $s \equiv t \mod K_{j-1}$, d'où

$$tu\sigma w \in K_{j-1} \Rightarrow tu\sigma \in \overline{K}_{j-1},$$

  donc

$$tu\Sigma_{in} \cap \overline{\mathcal{L}}(\mathcal{A}) = t'\Sigma_{in} \cap \overline{\mathcal{L}}(\mathcal{A}) \subset \overline{K}_{j-1}.$$

Par conséquent

$$\overline{tv}\Sigma_{in} \cap \overline{\mathcal{L}}(\mathcal{A}) \subset \overline{K}_{j-1} \text{ (et } tv \in K\text{), i.e. } tv \in K_j.$$

Finalement :

$$\forall v \in \Sigma^*,\ sv \in K_j \Rightarrow tv \in K_j.$$

Par symétrie, on a l'application inverse, d'où :

$$s \equiv_{K_j} t,$$

la récurrence est donc démontrée. Appliquons le lemme 4 aux langages $A = K$, $B = \overline{\mathcal{L}}(\mathcal{A})$ et $C = K_j$, on a alors

$$||K_j|| \leq ||\overline{\mathcal{L}}(\mathcal{A})|| \cdot ||K|| + 1.$$

□

Démonstration : (du théorème)

On a vu dans la partie 3.3.1 que la suite $(K_j)_j$ décroît. De plus, d'après le lemme 6 , $(||K_j||)_j$ est bornée. Le lemme 5 permet alors d'affirmer que la suite $(K_j)_j$ est stationnaire, notons K_∞ sa limite. Cette limite est bien entendu un point fixe pour l'opérateur Ω, donc : $K_\infty \subset K^\uparrow$. De plus, on a déjà vue l'inclusion inverse (cf. partie 3.3.1), donc

$$K^\uparrow = K_\infty.$$

D'où le résultat :

$$||K^\uparrow|| \leq ||\overline{\mathcal{L}}(\mathcal{A})|| \cdot ||K|| + 1.$$

□

3.4 Algorithmes de calcul

3.4.1 Un premier algorithme

Aux hypothèses faites dans le théorème 3 ($\mathcal{A}$ fini et K reconnaissable), on peut rajouter les hypothèses suivantes :

- $\mathcal{A}$ est émondé

  S. Gaubert [Gau95] ne fait pas cette hypothèse, mais cela simplifie et, étant donnée la complexité de l'algorithme, ce n'est pas gênant : on émonde en un temps linéaire par rapport à la taille de l'automate. Au besoin, on considère Em$\mathcal{A}$ à la place de $\mathcal{A}$.

- $K \subset \overline{\mathcal{L}}(\mathcal{A})$

  Ce qui est assez naturel puisque cela signifie que le langage cible est une restriction du comportement admissible du système ; au besoin, on peut s'y ramener en considérant $K \cap \overline{\mathcal{L}}(\mathcal{A})$ à la place de K.

- $\overline{\mathcal{L}}(\mathcal{A}) = \mathcal{L}(\mathcal{A})$

  Ce qui signifie que l'on se préoccupe de tous les événements.

Dans ces conditions, il existe un algorithme pour calculer $K^{\uparrow}$.

En effet, nous allons montrer que moyennant ces hypothèses, chaque langage K_j est rationnel et nous allons exhiber un algorithme donnant l'automate $\mathcal{C}_{j+1}$ à partir de l'automate $\mathcal{C}_j$ (où $\mathcal{C}_i$ désigne un automate reconnaissant le langage K_i). Nous montrerons alors que la suite $(\mathcal{C}_j)$ est stationnaire, de limite un automate $\mathcal{C}_\infty$ reconnaissant $K_\infty = K^{\uparrow}$.

(Le fait que l'on ait montré que $(K_j)_j$ est stationnaire dans la partie 3.3.2 n'implique pas que $(\mathcal{C}_j)$ le soit, puisqu'on ne fait aucune hypothèse sur le nombre d'états des $\mathcal{C}_j$.)

Notations 3 *Dans la suite, on notera $\Sigma^{\mathcal{A}}(q)$ l'ensemble des événements possibles à partir l'état q, dans l'automate $\mathcal{A}$.*

$$\Sigma^{\mathcal{A}}(q) = \{a \in \Sigma \mid \delta(a, q)!\}$$

Définition 13 *Si $\mathcal{A} = (Q, \delta, q_0, Q_a)$ et $\mathcal{B} = (X, \xi, x_0, X_a)$ sont deux automates émondé tels que $\overline{\mathcal{L}}(\mathcal{B}) \subset \overline{\mathcal{L}}(\mathcal{A})$, on dit que $\mathcal{B}$ est un* raffinement *de $\mathcal{A}$ si*

$$\forall s, t \in \overline{\mathcal{L}(\mathcal{B})}, \xi(s, x_0) = \xi(t, x_0) \;\Rightarrow\; \delta(s, q_0) = \delta(t, q_0).$$

On a alors une unique application $h : X \to Q$ telle que

$$\forall s \in \overline{\mathcal{L}(\mathcal{B})}, h(\xi(s, x_0)) = \delta(s, q_0).$$

L'intérêt des nouvelles hypothèses que nous avons effectuées est qu'elles entraînent le

Lemme 7 *Moyennant ces hypothèses, il existe un automate $\mathcal{C}_0$ raffinant $\mathcal{A}$ tel que $\mathcal{L}(\mathcal{C}_0) = K$.*

Démonstration :

Soit $\mathcal{B} = (Z, \alpha, z_0, Z_a)$, un automate fini reconnaissant K.

On définit le produit $\mathcal{A} \times \mathcal{B}$ par

$$\mathcal{A} \times \mathcal{B} := (Q \times Z, \delta \times \alpha, (q_0, z_0), Q_a \times Z_a),$$

où la fonction de transition est donnée par

$$(\delta \times \alpha)(\sigma, (q, z)) = (\delta(\sigma, q), \alpha(\sigma, z)).$$

Soit un mot w reconnu par $\mathcal{A} \times \mathcal{B}$, on a :

$$\begin{aligned}(\delta \times \alpha)(w, (q_0, z_0)) \in Q_a \times Z_a \quad & \Leftrightarrow \quad (\delta(w, q_0), \alpha(w, z_o)) \in Q_a \times Z_a \\ & \Leftrightarrow \quad (\delta(w, q_0) \in Q_a \text{ et } \alpha(w, z_0) \in Z_a) \\ & \Leftrightarrow \quad w \in \mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{B}). \qquad (5)\end{aligned}$$

Donc l'automate produit $\mathcal{A} \times \mathcal{B}$ reconnaît l'intersection des langages reconnus par $\mathcal{A}$ et $\mathcal{B}$. D'où :

$$\begin{aligned}\mathcal{L}(\mathcal{A} \times \mathcal{B}) &= \mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{B}) \\ &= \overline{\mathcal{L}}(\mathcal{A}) \cap K \qquad \text{d'après l'hypothèse } \mathcal{L}(\mathcal{A}) = \overline{\mathcal{L}}(\mathcal{A}). \\ &= K \qquad \text{car } K \subset \overline{\mathcal{L}}(\mathcal{A}).\end{aligned}$$

On ne change pas le langage reconnu par un automate en émondant celui-ci. Si on appelle $\mathcal{C}_0$, l'automate émondé de $\mathcal{A} \times \mathcal{B}$, alors $\mathcal{C}_0$ reconnaît K et est un raffinement de $\mathcal{A}$, en effet :

$$\delta \times \alpha(s, (q_0, z_0)) = \delta \times \alpha(t, (q_0, z_0)) \Rightarrow \delta(s, q_0) = \delta(t, q_0).$$

□

Revenons à la suite de langages (K_j) considérée plus haut.

Lemme 8 *Soit $\mathcal{A}$, un automate fini émondé et $\mathcal{C}_j = (X, \xi, x_0, X_a)$ un automate fini émondé raffinant $\mathcal{A}$ et reconnaissant le langage K_j. Alors*

$$w \in \Omega(K_j) \Leftrightarrow (w \in K_j \text{ et } (\forall u \in \overline{w}, x = \xi(u, x_0) \Rightarrow \Sigma^{\mathcal{A}}(h(x)) \cap \Sigma_{in} \subset \Sigma^{\mathcal{C}_j}(x))).$$

Démonstration :

Il y a équivalence entre les propriétés suivantes :

1. $w \in \Omega(K_j)$
2. $w \in K_j$ et $\overline{w}\Sigma_{in} \cap \overline{\mathcal{L}}(\mathcal{A}) \subset \overline{K_j}$
3. $w \in K_j$ et $\forall u \leq w, u\Sigma_{in} \cap \overline{\mathcal{L}}(\mathcal{A}) \subset \overline{K_j}$
4. $w \in K_j$ et $\forall u \leq w, \forall \sigma \in \Sigma_{in}, u\sigma \in \overline{\mathcal{L}}(\mathcal{A}) \Rightarrow u\sigma \in K_j$
5. $w \in K$ et $\forall u \leq w, \forall \sigma \in \Sigma^{\mathcal{A}}(\delta(u, q_0)) \cap \Sigma_{in}, u\sigma \in \overline{K_j}$
6. $w \in K_j$ et $\forall u \leq w, \Sigma^{\mathcal{A}}(\delta(u, q_0)) \cap \Sigma_{in} \subset \Sigma^{\mathcal{C}_j}(\xi(u, x_0))$

On montre que 1. $\Leftrightarrow$ 2., ... 5. $\Leftrightarrow$ 6., toutes ces équivalences s'obtiennent directement à partir des définitions.

Ce qui permet de conclure en posant

$$x = \xi(u, x_0) \text{ et } h(x) = \delta(u, q_o).$$

$\square$

On peut donc construire l'automate $\mathcal{C}_{j+1}$ reconnaissant $K_{j+1} = \Omega(K_j)$ en supprimant les états de $\mathcal{C}_j$ qui violent la contrainte

$$\Sigma^{\mathcal{A}}(h(x)) \cap \Sigma_{in} \subset \Sigma^{\mathcal{C}_j}(x). \tag{6}$$

Appelons X' l'ensemble des états de $\mathcal{C}_j$ qui vérifient cette contrainte et soit $X'_a = X_a \cap X'$. On définit la nouvelle fonction de transition

$$\xi'(\sigma, x) = \begin{cases} \xi(\sigma, x) & \text{si } \xi(\sigma, x) \in X', \\ \text{non défini} & \text{sinon.} \end{cases}$$

On définit alors $\mathcal{C}_{j+1}$ par :

$$\mathcal{C}_{j+1} = \begin{cases} \mathrm{Em}(X', \xi', x_0, X'_a) & \text{si } x_0 \in X', \\ \emptyset & \text{sinon.} \end{cases} \tag{7}$$

Grâce à cette construction, on peut énoncer le

Théorème 5 *Soit $\mathcal{C}_{j+1}$ défini par l'équation 7. On a :*

$$\overline{\mathcal{L}}(\mathcal{C}_{j+1}) = K_{j+1} = \Omega(K_j).$$

De plus, la suite $(\mathcal{C}_j)$ est stationnaire et sa limite $\mathcal{C}_\infty$ vérifie :

$$K_\infty = \overline{\mathcal{L}}(\mathcal{C}_\infty) = K^\uparrow.$$

On obtient donc en un nombre fini d'itérations un automate reconnaissant le sous-langage contrôlable maximal.

Démonstration :

On a, d'après le lemme 8

$$w \in \Omega(K_j) \quad \Leftrightarrow \quad w \in K_j \text{ et } \forall u \leq w, \Sigma^{\mathcal{A}}(\delta(u, x_0)) \cap \Sigma_{in} \subset \Sigma^{\mathcal{A}_j}(\xi(u, x_0))$$

Soit, d'après la définition de X', si on note $x = \xi(u, x_0)$ (alors $h(x) = \delta(u, x_0)$)) :

$$\begin{aligned} w \in \Omega(K_j) \quad &\Leftrightarrow \quad w \in K_j \text{ et } \forall u \leq w, x = \xi(u, x_0) \in X' \\ &\Leftrightarrow \quad w \in K_j \text{ et } \forall u \leq w, \xi'(u, x_0) \text{ est défini} \\ &\Leftrightarrow \quad \xi'(w, x_0) \text{ est défini.} \end{aligned}$$

La première égalité du théorème est donc démontrée.

$\mathcal{C}_{j+1}$ s'obtient en supprimant des états dans $\mathcal{C}_j$ et l'on part d'un automate fini, donc la suite $(\mathcal{C}_j)$ est stationnaire. Ceci permet tout de suite de conclure que K_∞ est un point fixe de l'opérateur Ω. Comme $K^\uparrow$ est le plus grand point fixe de cet opérateur, d'après la proposition 3, on a l'inclusion $K_\infty \subset K^\uparrow$. Par ailleurs, nous avons déjà vu l'inclusion inverse (p.19), on obtient donc finalement l'égalité entre les deux ensembles.

□

Calculons la complexité de cet algorithme. On émonde de façon progressive un automate raffinant $\mathcal{A}$ et reconnaissant K (soit au plus $||\overline{\mathcal{L}}(\mathcal{A})|| \cdot ||K||$ états), le nombre d'itérations est donc borné par le nombre d'arcs à émonder, soit au plus $(||\overline{\mathcal{L}}(\mathcal{A})|| \cdot ||K||)^2$.

Exemple — (tiré de [WR87], repris par [Gau95])

Soit $\Sigma = \{a_1, a_2, b\}$, $\Sigma_{in} = \{b\}$. On considère l'automate $\mathcal{A}$ reconnaissant le langage $\overline{\mathcal{L}}(\mathcal{A}) = \overline{(a_1 b^2 + a_2)b^*}$, et le langage $K = \overline{a_1 b^2} + a_2 b^*$ reconnu par l'automate $\mathcal{B}$ (cf. figure 4).

Pour $K_0 = K$, on a la table de transitions suivantes (l'ensemble vide sera noté par un blanc dans la table) :

x	0	1	2	3	4	$\Sigma^{\mathcal{A}}(h(x)) \cap \Sigma_{in}$	$\Sigma^{\mathcal{B}}(x)$
0		a_1			a_2		$a_1 + a_2$
1			b			b	b
2				b		b	b
3						b	
4					b	b	b

Ce tableau est divisé en trois sous-tableau, le premier n'est autre que la représentation matricielle de l'automate $\mathcal{B}$.

Pour l'état $x = 3$, on a : $\Sigma^{\mathcal{A}}(3) \cap \Sigma_{in} = \{b\}$, qui n'est pas inclus dans $\Sigma^{\mathcal{B}}(3)$, il faut donc éliminer l'état 3 de l'automate $\mathcal{B}$. On obtient alors le tableau suivant pour K_1 :

x	0	1	2	4	$\Sigma^{\mathcal{A}}(h(x)) \cap \Sigma_{in}$	$\Sigma^{\mathcal{B}}(x)$
0		a_1		a_2		$a_1 + a_2$
1			b		b	b
2					b	b
4				b	b	b

Donc $K_1 = \overline{a_1 b} + a_2 b^*$. On recommence en remarquant qu'il faut éliminer l'état 2 de ce nouvel automate, puis l'état 1. On obtient successivement :

x	0	1	4	$\Sigma^{\mathcal{A}}(h(x)) \cap \Sigma_{in}$	$\Sigma^{\mathcal{B}}(x)$
0		a_1	a_2		$a_1 + a_2$
1				b	b
4			b	b	b

Soit $K_2 = \overline{a_1} + a_2 b^*$.

x	0	4	$\Sigma^{\mathcal{A}}(h(x)) \cap \Sigma_{in}$	$\Sigma^{\mathcal{B}}(x)$
0		a_2		$a_1 + a_2$
4		b	b	b

D'où $K_3 = e + a_2 b^*$.

La contrainte 6 est vérifiée par K_3, on a donc $K^{\uparrow} = e + a_2 b^*$.

Exemple —

Reprenons l'exemple du chat et de la souris p.9

L'automate $\mathcal{B}$ représentant le comportement du système s'obtient en supprimant les états diagonaux, puis en émondant (cf. figure 5).

Supposons que l'on contrôle toutes les trappes sauf c_1 : le chat peut emprunter cette trappe de manière incontrôlable.

$$\Sigma_{in} = \{c_1\}$$

Pour l'automate de la figure 5, on obtient le tableau suivant :

x	12	13	21	23	32	$\Sigma^{\mathcal{A}}(h(x)) \cap \Sigma_{in}$	$\Sigma^{\mathcal{B}}(x)$
12		m_2			c_1	c_1	$m_2 + c_1$
13	m_2			c_2		c_1	$m_2 + c_2$
21				m_1			m_1
23		c_2	m_1				$m_1 + c_2$
32	c_1					c_1	c_1

L'état $x = 13$ ne vérifie pas la contrainte $\Sigma^{\mathcal{A}}(h(x)) \cap \Sigma_{in} \subset \Sigma^{\mathcal{B}}(x)$, il doit donc être supprimé. Les états 21 et 23 ne sont alors plus accessibles : on les supprime aussi. On obtient finalement $K^{\uparrow} = c_1^*$, c'est-à-dire que toutes les trappes contrôlables sont fermées, la souris reste enfermée dans la pièce P_2 et le chat circule librement entre les pièces P_1 et P_3.

3.4.2 Un algorithme plus rapide

Dans cette partie, on fait les mêmes hypothèse que dans la partie 3.4.1,

L'algorithme exposé dans la partie 3.4.1 correspond à l'approche naturelle (et historique), nous allons exposer dans cette partie un algorithme de nature différente et moins intuitive, mais plus rapide.

Pour cela, montrons d'abord le

Lemme 9 *Soit $J \subset \overline{\mathcal{L}}(\mathcal{A})$. On a l'équivalence*

$$J \text{ est contrôlable } \Leftrightarrow \overline{J}\Sigma_{in}^* \cap \overline{\mathcal{L}}(\mathcal{A}) \subset \overline{J}.$$

(Au lieu d'exécuter un *événement incontrôlable après un mot de J, on en exécute* un nombre fini quelconque.*)*

Démonstration :

- $\Leftarrow$) Supposons que $\overline{J}\Sigma_{in}^* \cap \overline{\mathcal{L}}(\mathcal{A}) \subset \overline{J}$. Alors

  $$\overline{J}\Sigma_{in} \cap \overline{\mathcal{L}}(\mathcal{A}) \subset \overline{J}\Sigma_{in}^* \cap \overline{\mathcal{L}}(\mathcal{A}) \subset \overline{J},$$

  donc J est contrôlable.

- $\Rightarrow$) Supposons J contrôlable.

  Montrons que

  $$\overline{J}\Sigma_{in}^j \cap \overline{\mathcal{L}}(\mathcal{A}) \subset \overline{J},$$

  par récurrence sur j.

  Si $j \in \{0, 1\}$, c'est évident.

  hypothèse de récurrence :

  $$\overline{J}\Sigma_{in}^j \cap \overline{\mathcal{L}}(\mathcal{A}) \subset \overline{J},$$

alors :

$$\begin{array}{lll}
\overline{J}\Sigma_{in}^{j+1} \cap & \overline{\mathcal{L}}(\mathcal{A}) \subset \overline{J}\Sigma_{in}^{j}\Sigma_{in} \cap \overline{\mathcal{L}}(\mathcal{A}) & \\
& \subset \overline{J}\Sigma_{in}^{j}\Sigma_{in} \cap \overline{\mathcal{L}}(\mathcal{A})\Sigma_{in} \cap \overline{\mathcal{L}}(\mathcal{A}) & \text{car } \overline{J}\Sigma_{in}^{j}\Sigma_{in} \cap \overline{\mathcal{L}}(\mathcal{A}) \subset \overline{\mathcal{L}}(\mathcal{A})\Sigma_{in} \\
& & (\text{si } w \in \overline{J}\Sigma_{in}^{j}\Sigma_{in} \cap \overline{\mathcal{L}}(\mathcal{A}), \\
& & \text{alors } w = ua, a \in \Sigma_{in}, \\
& & \text{et } u \in \overline{\overline{\mathcal{L}}(\mathcal{A})} = \overline{\mathcal{L}}(\mathcal{A})) \\
& \subset (\overline{J}\Sigma_{in}^{j} \cap \overline{\mathcal{L}}(\mathcal{A}))\Sigma_{in} \cap \overline{\mathcal{L}}(\mathcal{A}) & \\
& \subset \overline{J}\Sigma_{in} \cap \overline{\mathcal{L}}(\mathcal{A}) & \text{par hypothèse de récurrence} \\
& \subset \overline{J} & \text{car } J \text{ est contrôlable.}
\end{array}$$

□

Introduisons un nouvel opérateur

$$D_{in} : \Sigma^* \to \Sigma^*,$$

qui "coupe" le suffixe incontrôlable de chaque mot, i.e.

$D_{in}(w) = e$ si w ne contient pas d'événements contrôlable,

$D_{in}(w) =$ le plus long préfixe de w dont la dernière lettre est contrôlable, sinon.

Intuitivement, $\overline{\mathcal{L}}(\mathcal{A}) - K$ est l'ensemble des mots illégaux, et $D_{in}(\overline{\mathcal{L}}(\mathcal{A}) - K)$ l'ensemble des mots qui peuvent être complétés en mots illégaux, en leur ajoutant des événements incontrôlables. Ils doivent donc être enlevés de K, c'est le principe-même de ce second algorithme. Pour garantir la fermeture du langage obtenu, on enlève $D_{in}(\overline{\mathcal{L}}(\mathcal{A}) - K)\Sigma^*$.

Théorème 6 *On a l'égalité*

$$K^{\uparrow} = K - D_{in}(\overline{\mathcal{L}}(\mathcal{A}) - K)\Sigma^*$$

Démonstration :

($\supset$) Notons $H = K - D_{in}(\overline{\mathcal{L}}(\mathcal{A}) - K)\Sigma^*$.

H est un sous-ensemble de K. Il nous suffit donc de montrer que H est contrôlable. Commençons par montrer que H est fermé.

Soit $u \in \overline{H}$. Montrons que $u \in H$.

Soit $w \in H$ et $v \in \Sigma^*$ tels que $w = uv$.

w est un mot de K, donc u appartient à $\overline{K}$. Les hypothèses que nous avons faites nous disent que $K \subset \overline{\mathcal{L}}(\mathcal{A})$ et que $\overline{\mathcal{L}}(\mathcal{A})$ est fermé car égale à $\overline{\mathcal{L}(\mathcal{A})}$ du fait que l'on ait pris $\mathcal{A}$ co-accessible. Par conséquent, u est un élément de $\overline{\mathcal{L}}(\mathcal{A})$. Supposons que u ne soit pas un élément de K, alors $u \in \overline{\mathcal{L}}(\mathcal{A}) - K$, donc $D_{in}(u) \in D_{in}(\overline{\mathcal{L}}(\mathcal{A}) - K)$, or $D_{in}(u)$ est un préfixe de u, donc de w, on aurait donc $w \in D_{in}(\overline{\mathcal{L}}(\mathcal{A}) - K)\Sigma^*$, ce qui contredit le fait que $w \in H$, donc u est un élément de K.

Par ailleurs, si u était un élément de $D_{in}(\overline{\mathcal{L}}(\mathcal{A}) - K)\Sigma^*$, w le serait aussi, puisque u est préfixe de w, ce qui est impossible. Donc $u \in H$. Par conséquent H est fermé.

Montrons que H est contrôlable, par l'absurde : soit $s \in \Sigma^*$ et $\sigma \in \Sigma$ tels que

$$(s \in H) \wedge (\sigma \in \Sigma_{in}) \wedge (s\sigma \in \overline{\mathcal{L}}(\mathcal{A})) \wedge (s\sigma \notin H).$$

On a alors

$$(s \in H) \wedge (\sigma \in \Sigma_{in}) \wedge (s\sigma \in \overline{\mathcal{L}}(\mathcal{A})) \wedge ((s\sigma \notin K) \vee (s\sigma \in D_{in}(\overline{\mathcal{L}}(\mathcal{A}) - K)\Sigma^*)),$$

car $H = K - D_{in}(\overline{\mathcal{L}}(\mathcal{A}) - K)\Sigma^*$. D'où

$$(s \in H) \wedge (\sigma \in \Sigma_{in}) \wedge (s\sigma \in \overline{\mathcal{L}}(\mathcal{A})) \wedge ((s\sigma \in \overline{\mathcal{L}}(\mathcal{A}) - K) \vee (s\sigma \in D_{in}(\overline{\mathcal{L}}(\mathcal{A}) - K)\Sigma^*),$$

car $((s\sigma \in \overline{\mathcal{L}}(\mathcal{A})) \wedge (s\sigma \notin K)) \Rightarrow (s\sigma \in \overline{\mathcal{L}}(\mathcal{A}) - K)$. Donc

$$(s \in H) \wedge (\sigma \in \Sigma_{in}) \wedge (s\sigma \in \overline{\mathcal{L}}(\mathcal{A})) \wedge (s\sigma \in D_{in}(\overline{\mathcal{L}}(\mathcal{A}) - K)\Sigma^*),$$

car $D_{in}(w)$ est un préfixe de w, donc $\overline{\mathcal{L}}(\mathcal{A}) - K \subset D_{in}(\overline{\mathcal{L}}(\mathcal{A}) - K)\Sigma^*$. Finalement on a abouti à

$$(s \in H) \wedge (\sigma \in \Sigma_{in}) \wedge (s\sigma \in \overline{\mathcal{L}}(\mathcal{A})) \wedge (s \in D_{in}(\overline{\mathcal{L}}(\mathcal{A}) - K)\Sigma^*)$$

(car $s \in \Sigma_{in}$), ce qui est une contradiction : $H \cap D_{in}(\overline{\mathcal{L}}(\mathcal{A}) - K)\Sigma^* = \emptyset$.

($\subset$) On a déjà vu que $K^\uparrow \subset K$, on a donc juste besoin de montrer que

$$D_{in}(\overline{\mathcal{L}}(\mathcal{A}) - K)\Sigma^* \cap K^\uparrow = \emptyset.$$

Par l'absurde : soit $s \in D_{in}(\overline{\mathcal{L}}(\mathcal{A}) - K)$ et $t \in \Sigma^*$ tels que

$$(st \in D_{in}(\overline{\mathcal{L}}(\mathcal{A}) - K)\Sigma^*) \wedge (st \in K^\uparrow).$$

On a alors

$$(s \in D_{in}(\overline{\mathcal{L}}(\mathcal{A}) - K)) \wedge (s \in K^\uparrow),$$

car $K^\uparrow$ est fermé. La définition de D_{in} permet alors d'écrire

$$\exists u \in \Sigma_{in}^* \mid (su \in \overline{\mathcal{L}}(\mathcal{A}) - K) \wedge (s \in K^\uparrow).$$

Soit encore

$$(\exists u \in \Sigma_{in}^* \mid (su \notin K) \wedge (su \in \overline{\mathcal{L}}(\mathcal{A}))) \wedge (s \in K^\uparrow).$$

Finalement, le lemme précédent appliqué à $J = K^{\uparrow}$ donne :

$$\exists u \in \Sigma_{in}^{*} \mid (su \notin K) \ \wedge \ (su \in K^{\uparrow}),$$

ce qui est une contradiction car $K^{\uparrow} \subset K$.

$\square$

On peut alors utiliser l'expression que l'on vient de trouver pour $K^{\uparrow}$ pour développer un algorithme. Calculons la complexité de cet algorithme. Soit m et n le nombre d'états de l'automate minimal des langages $\overline{\mathcal{L}}(\mathcal{A})$ et K respectivement. Alors on peut construire l'automate minimal de $K^{\uparrow} = K - D_{in}(\overline{\mathcal{L}}(\mathcal{A}) - K)\Sigma^{*}$ en un temps $O(mn^2)$. En effet, on construit l'automate minimal qui reconnaît le langage $\overline{\mathcal{L}}(\mathcal{A}) - K$ en un temps $O(mn)$, donc l'automate minimal de $D_{in}(\overline{\mathcal{L}}(\mathcal{A}) - K)\Sigma^{*}$ aussi et finalement, l'automate minimal de $K - D_{in}(\overline{\mathcal{L}}(\mathcal{A}) - K)\Sigma^{*}$ est construit en un temps $O(mn^2)$. La complexité de l'algorithme qui calcule $K^{\uparrow}$ à partir de la formule est donc $O(mn^2)$

*

* *

4 Vers un lien avec la logique

Dans toute cette section $\mathcal{A} = (Q, \Sigma, \delta, q_0, Q_a)$ et nous étudierons la contrôlabilité par rapport à $\mathcal{A}$.

Nous allons voir dans cette section, comment la logique peut permettre de résoudre certains problèmes de supervision (comme l'intersection de deux langages contrôlables).

Pour cela, nous reprenons la notion d'automate contrôleur introduite au paragraphe 3.1.

Dans toute cette partie, nous ne nous intéresserons pas à l'ensemble des états d'acceptation pour les automates, ils existent, mais ne jouent aucun rôle. Aussi, nous noterons l'automate $\mathcal{A} = (Q, \Sigma, \delta, q_0, Q_a)$ par $\mathcal{A} = (Q, \Sigma, \delta, q_0)$.

4.1 Systèmes à événements discrets contrôlés

Soit un automate $\mathcal{A} = (Q, \Sigma, \delta, q_0)$ représentant un système à événements discrets, où $\Sigma = \Sigma_c \cup \Sigma_{in}$, comme d'habitude. On interprète $\mathcal{A}$ comme un processus qui commence à l'état q_0 et engendre une suite d'événements, autorisée par la fonction de transition δ. Chaque événement a lieu spontanément et instantanément.

Nous allons ajouter un mécanisme de contrôle à $\mathcal{A}$, qui va permettre de préciser pour chaque événement s'il est *autorisé* (i.e. il peut avoir lieu) ou *interdit* (i.e. il ne peut pas avoir lieu). L'ensemble des *modèles* pour $\mathcal{A}$ est défini par

$$\Gamma = \{\gamma \mid \gamma : \Sigma \to \{0, 1\} \text{ et } \forall \sigma \in \Sigma_{in}, \gamma(\sigma) = 1\}.$$

L'événement de label $\sigma \in \Sigma$ suit le modèle γ si $\gamma(\sigma) = 1$ et ne le suit pas $\gamma(\sigma) = 0$. Le système étendu

$$\mathcal{A}_c = (Q, \Gamma \times \Sigma, \delta_c, q_0),$$

avec

$$\delta_c((\gamma, \sigma), q) = \begin{cases} \delta(\sigma, q) & \text{si } \gamma(\sigma) = 1, \\ \text{non défini} & \text{sinon.} \end{cases}$$

est appelé *système à événements discrets contrôlé*. Ce processus est identique à celui de départ, mais le mécanisme de contrôle est maintenant explicite.

Nous allons définir sur Γ les opérations suivantes : la négation ($\neg$), la conjonction ($\wedge$) et la disjonction ($\vee$). Pour tous modèles $\gamma, \gamma_1, \gamma_2 \in \Gamma$ et toute lettre $\sigma \in \Sigma$, on pose :

$$
\begin{aligned}
(\neg\gamma)(\sigma) &= \begin{cases} \neg\gamma(\sigma) & \text{si } \sigma \in \Sigma_c, \\ 1 & \text{sinon,} \end{cases} \\
(\gamma_1 \wedge \gamma_2)(\sigma) &= \begin{cases} \gamma_1(\sigma) \wedge \gamma_2(\sigma) & \text{si } \sigma \in \Sigma_c, \\ 1 & \text{sinon,} \end{cases} \\
\gamma_1 \vee \gamma_2 &= \neg((\neg\gamma_1) \wedge (\neg\gamma_2)).
\end{aligned}
$$

$(\Gamma, \neg, \wedge, \vee)$ ainsi formé est une algèbre de Boole.

4.2 Un nouvel algorithme

4.2.1 Conjonction de superviseurs

Définition 14 *On dira qu'un superviseur $\mathcal{S} = (S, \phi)$ est* complet *pour le système étendu $\mathcal{A}_c$ si pour tout mot s et toute lettre σ, les conditions suivantes :*

1. $s \in \overline{\mathcal{L}}(\mathcal{S}/\mathcal{A})$,
2. $s\sigma \in \overline{\mathcal{L}}(\mathcal{A})$,
3. $\phi(\xi(s, x_0))(\sigma) = 1$

   entraînent la condition

4. $s\sigma \in \overline{\mathcal{L}}(\mathcal{S}/\mathcal{A})$.

Remarque —
Si $f : \overline{\mathcal{L}}(\mathcal{A}) \to \mathcal{P}(\Sigma) : w \mapsto \phi(\delta(w, q_0))$, on retrouve un superviseur comme on l'avait défini en un premier temps (partie 3).

Par abus de notations, on pose alors :

$$\mathcal{L}_c(\mathcal{S}/\mathcal{A}) = \mathcal{L}_f(\mathcal{A}) = \overline{\mathcal{L}}(\mathcal{S}/\mathcal{A}) \cap \mathcal{L}(\mathcal{A}),$$

qui est le *langage contrôlé* de $\mathcal{S}/\mathcal{A}$.

On dira que le superviseur $\mathcal{S}$ est non bloquant (pour le système contrôlé $\mathcal{A}_c$) si

$$\overline{\mathcal{L}_c(\mathcal{S}/\mathcal{A})} = \overline{\mathcal{L}}(\mathcal{S}/\mathcal{A}),$$

c'est-à-dire si tout mot du langage contrôlé peut être complété en un mot reconnu par $\mathcal{A}$. Cette définition est équivalente à la définition 9 qui avait été donnée pour la première notion de superviseur.

Définition 15 *Soit $\mathcal{S} = (S, \phi)$ et $\mathcal{T} = (T, \psi)$, deux superviseurs complets pour $\mathcal{A}_c$, avec $S = (X, \Sigma, \xi, x_0)$ et $T = (Y, \Sigma, \eta, y_0)$. La* conjonction *de $\mathcal{S}$ et de $\mathcal{T}$ est le superviseur*

$$\mathcal{S} \wedge \mathcal{T} = (S \times T, \phi * \psi),$$

où

$$\begin{aligned} S \times T &= Ac(X \times Y, \Sigma, \xi \times \eta, (x_0, y_0)) \\ \xi \times \eta(\sigma, (x, y)) &= \begin{cases} (\xi(\sigma, x), \eta(\sigma, x)) & si\ \xi(\sigma, x) \text{ et } \eta(\sigma, x) \ sont\ définis, \\ non\ défini & sinon, \end{cases} \\ \phi * \psi(x, y) &= \phi(x) \cap \psi(y). \end{aligned}$$

La proposition suivante montre que $\mathcal{S} \wedge \mathcal{T}$ est complet pour $\mathcal{A}_c$ et relie les langages contrôlés de $\mathcal{S} \wedge \mathcal{T}/\mathcal{A}$, $\mathcal{S}/\mathcal{A}$ et $\mathcal{T}/\mathcal{A}$.

Proposition 4 *Soit deux superviseurs $\mathcal{S}$ et $\mathcal{T}$ et $\mathcal{S} \wedge \mathcal{T}$ défini comme précédemment.*

On a alors :

1. $\overline{\mathcal{L}}(\mathcal{S} \wedge \mathcal{T}/\mathcal{A}) = \overline{\mathcal{L}}(\mathcal{S}/\mathcal{A}) \cap \overline{\mathcal{L}}(\mathcal{T}/\mathcal{A})$,
2. $\mathcal{L}_c(\mathcal{S} \wedge \mathcal{T}/\mathcal{A}) = \mathcal{L}_c(\mathcal{S}/\mathcal{A}) \cap \mathcal{L}_c(\mathcal{T}/\mathcal{A})$ *et*
3. $\mathcal{S} \wedge \mathcal{T}$ *est un superviseur complet pour $\mathcal{A}_c$.*

Démonstration :

1. Par récurrence sur la longueur des mots.

   $1 \in \overline{\mathcal{L}}(\mathcal{S} \wedge \mathcal{T}/\mathcal{A})$ et $1 \in \overline{\mathcal{L}}(\mathcal{S}/\mathcal{A}) \cap \overline{\mathcal{L}}(\mathcal{T}/\mathcal{A})$ car ce sont deux langages préfixiels.

   hypothèse de récurrence : u de longueur k appartient à $\overline{\mathcal{L}}(\mathcal{S} \wedge \mathcal{T}/\mathcal{A})$ si et seulement s'il appartient à $\overline{\mathcal{L}}(\mathcal{S}/\mathcal{A}) \cap \overline{\mathcal{L}}(\mathcal{T}/\mathcal{A})$.

   alors : soit $w = u\sigma$ avec $u \in \overline{\mathcal{L}}(\mathcal{S} \wedge \mathcal{T}/\mathcal{A})$ de longueur k et $\sigma \in \Sigma$, une lettre quelconque.

   L'hypothèse de récurrence entraîne que u appartient à $\overline{\mathcal{L}}(\mathcal{S}/\mathcal{A})$ et à $\overline{\mathcal{L}}(\mathcal{T}/\mathcal{A})$. Donc $q = \delta(u, q_0)$, $x = \xi(u, x_0)$ et $y = \eta(u, y_0)$ sont définis.

On a donc

$$\begin{array}{rcl} u\sigma \in \overline{\mathcal{L}}(\mathcal{S} \wedge \mathcal{T}/\mathcal{A}) & \Leftrightarrow & \xi \times \eta \times \delta_c(\sigma, (x,y), q) \text{ est défini}, \\ & \Leftrightarrow & \xi \times \eta(\sigma, (x,y)) \text{ et } \delta_c(\phi * \psi(x,y), \sigma, q) \text{ sont définis}, \\ & \Leftrightarrow & \left\{ \begin{array}{l} \xi(\sigma, x) \text{ et } \delta_c(\phi(x), \sigma, q) \text{ et} \\ \eta(\sigma, x) \text{ et } \delta_c(\psi(y), \sigma, q) \text{ sont définis}, \end{array} \right. \\ & \Leftrightarrow & u\sigma \in \overline{\mathcal{L}}(\mathcal{S}/\mathcal{A}) \text{ et } u\sigma \in \overline{\mathcal{L}}(\mathcal{T}/\mathcal{A}). \end{array}$$

2. On a les égalités :

$$\begin{array}{rcl} \mathcal{L}_c(\mathcal{S} \wedge \mathcal{T}/\mathcal{A}) & = & \overline{\mathcal{L}}(\mathcal{S} \wedge \mathcal{T}/\mathcal{A}) \cap \mathcal{L}(\mathcal{A}) \\ & = & \overline{\mathcal{L}}(\mathcal{S}/\mathcal{A}) \cap \mathcal{L}(\mathcal{A}) \cap \overline{\mathcal{L}}(\mathcal{T}/\mathcal{A}) \cap \mathcal{L}(\mathcal{A}) \\ & = & \mathcal{L}_c(\mathcal{S}/\mathcal{A}) \cap \mathcal{L}_c(\mathcal{T}/\mathcal{A}). \end{array}$$

3. Soit un mot $s \in \overline{\mathcal{L}}(\mathcal{S} \wedge \mathcal{T})$ et $\sigma \in \Sigma$ une lettre tels que $s\sigma \in \overline{\mathcal{L}}(\mathcal{A})$ et

$$\phi * \psi(\xi \times \eta(s, (x_0, y_0)))(\sigma) = 1.$$

La partie 1 de la proposition permet d'affirmer que $s \in \overline{\mathcal{L}}(\mathcal{S}/\mathcal{A})$ et la définition de $\phi * \psi$ que

$$\phi(\xi(s, x_0))(\sigma) = 1.$$

Comme $\mathcal{S}$ est complet pour $\mathcal{A}_c$, cela implique que $s\sigma \in \overline{\mathcal{L}}(\mathcal{S}/\mathcal{A})$. On montre de même que $s\sigma \in \overline{\mathcal{L}}(\mathcal{T}/\mathcal{A})$. La partie 1 de la proposition permet alors de conclure $s\sigma \in \overline{\mathcal{L}}(\mathcal{S} \wedge \mathcal{T}/\mathcal{A})$.

□

Cette proposition montre clairement que la famille des superviseurs complets pour le système contrôlé $\mathcal{A}_c$ est fermée par conjonction. De plus, la conjonction est une opération associative et commutative (immédiat sur la définition). Considérons le superviseur qui n'a qu'un état, tel que toutes les transitions sont définies (i.e. pour toute lettre σ, il existe une boucle de l'unique état vers lui-même libellée par σ) et où toutes les transitions sont autorisées : c'est un élément neutre pour la conjonction. La famille des superviseurs complets pour le système contrôlé $\mathcal{A}_c$ munie de la conjonction est donc un monoïde commutatif.

Cependant, le fait d'être non bloquant pour un superviseur n'est pas conservé par conjonction. En effet, on peut aisément imaginer que si deux superviseurs ont des contrôles contradictoires, leur conjonction ne sera pas non bloquante.

C'est la motivation de la

Définition 16 *Soit deux langages L_1 et L_2 sur Σ.*

L_1 et L_2 sont dits non conflictuels *si*

$$\overline{L_1 \cap L_2} = \overline{L_1} \cap \overline{L_2}.$$

Remarque —

On a toujours l'inclusion $\overline{L_1 \cap L_2} \subset \overline{L_1} \cap \overline{L_2}$.

Exemple — Deux langages fermés sont toujours non conflictuels.

Le fait d'être non bloquant pour $\mathcal{S} \wedge \mathcal{T}$ est caractérisé grâce à cette notion, comme nous le montrons dans la proposition suivante.

Proposition 5 *Soit $\mathcal{S}$ et $\mathcal{T}$, deux superviseurs non bloquants pour le système contrôlé $\mathcal{A}_c$.*

Le superviseur $\mathcal{S} \wedge \mathcal{T}$ est non bloquant si et seulement si les langages $\mathcal{L}_c(\mathcal{S}/\mathcal{A})$ et $\mathcal{L}_c(\mathcal{T}/\mathcal{A})$ sont non conflictuels.

Démonstration :

$$\begin{array}{rcl}
\mathcal{S} \wedge \mathcal{T} \text{ est non bloquant} & \Leftrightarrow & \overline{\mathcal{L}_c(\mathcal{S} \wedge \mathcal{T}/\mathcal{A})} = \overline{\mathcal{L}}(\mathcal{S} \wedge \mathcal{T}/\mathcal{A}), \\
& \Leftrightarrow & \overline{\mathcal{L}_c(\mathcal{S}/\mathcal{A}) \cap \mathcal{L}_c(\mathcal{T}/\mathcal{A})} = \overline{\mathcal{L}}(\mathcal{S}/\mathcal{A}) \cap \overline{\mathcal{L}}(\mathcal{T}/\mathcal{A}), \\
& \Leftrightarrow & \overline{\mathcal{L}_c(\mathcal{S}/\mathcal{A}) \cap \mathcal{L}_c(\mathcal{T}/\mathcal{A})} = \overline{\mathcal{L}_c(\mathcal{S}/\mathcal{A})} \cap \overline{\mathcal{L}_c(\mathcal{T}/\mathcal{A})}, \\
& \Leftrightarrow & \mathcal{L}_c(\mathcal{S}/\mathcal{A}) \text{ et } \mathcal{L}_c(\mathcal{T}/\mathcal{A}) \text{ sont non conflictuels.}
\end{array}$$

□

De façon similaire, on étend la notion de non conflictuels à une famille finie de superviseurs et on montre la proposition correspondante sur le superviseur conjonction de la famille.

4.2.2 Synthèse d'un superviseur

Dans la section 3.2, nous avons déjà vu des conditions suffisantes pour qu'un langage soit contrôlable. Dans cette section, nous allons en voir d'autres, exprimées en termes de logique et qui permettent d'aboutir à des algorithmes de synthèse de superviseurs différents.

Proposition 6 *Si L_1 et L_2 sont deux langages sur Σ non conflictuels et contrôlables, alors leur intersection $L_1 \cap L_2$ est contrôlable.*

Nous avions déjà vu une condition suffisante pour que l'intersection de deux contrôlables soit contrôlable p.16, mais celle-ci est plus fine.

Démonstration :

On a :

$$\begin{array}{rcl} \overline{L_1 \cap L_2}\Sigma_{in} \cap \overline{\mathcal{L}}(\mathcal{A}) & = & (\overline{L_1} \cap \overline{L_2})\Sigma_{in} \cap \overline{\mathcal{L}}(\mathcal{A}) \\ & = & \overline{L_1}\Sigma_{in} \cap \overline{\mathcal{L}}(\mathcal{A}) \cap \overline{L_2}\Sigma_{in} \cap \overline{\mathcal{L}}(\mathcal{A}) \\ & \subset & \overline{L_1} \cap \overline{L_2} \\ & = & \overline{L_1 \cap L_2}. \end{array}$$

□

Problème —

Deux langages $L_1, L_2 \in \overline{\mathcal{L}}(\mathcal{A})$ non vides et fermés étant donnés, synthétiser un superviseur $\mathcal{S}$ (pas nécessairement non bloquant) tel que $\overline{\mathcal{L}}(\mathcal{S}/\mathcal{A}) = L_1 \cap L_2$.

On a vu dans le théorème 2 qu'on peut résoudre ce problème si et seulement si $L_1 \cap L_2$ est contrôlable. La proposition 6 donne comme condition suffisante que L_1 et L_2 soient tous les deux contrôlables (car ce sont des langages fermés). Dans ce cas, il existe des superviseurs $\mathcal{S}_1$ et $\mathcal{S}_2$ complets tels que

$$\overline{\mathcal{L}}(\mathcal{S}_1/\mathcal{A}) = L_1 \text{ et } \overline{\mathcal{L}}(\mathcal{S}_2/\mathcal{A}) = L_2.$$

Et la proposition 4 permet alors de conclure

$$\overline{\mathcal{L}}(\mathcal{S}_1 \wedge \mathcal{S}_2/\mathcal{A}) = L_1 \cap L_2.$$

Il est intéressant de comparer la complexité de cette solution à la complexité de la solution obtenue par l'algorithme 3.4.2. Pour cela, considérons le cas où $\mathcal{A}$ est un automate fini de taille m et les langages L_1 et L_2 sont réguliers, leurs automates minimaux ayant un nombre d'états majoré par n.

Calculer $L_1 \cap L_2$ se fait en un temps $\mathcal{O}(n^2)$ et synthétiser un superviseur par l'algorithme 3.4.2 se fait alors en un temps $\mathcal{O}(mn^2)$. Avec cette nouvelle méthode, il faut vérifier que L_1 et L_2 sont contrôlables puis construire (si c'est possible) des superviseurs $\mathcal{S}_1$ et $\mathcal{S}_2$ tels que $\overline{\mathcal{L}}(\mathcal{S}_1/\mathcal{A}) = L_1$ et $\overline{\mathcal{L}}(\mathcal{S}_2/\mathcal{A}) = L_2$, ce qui se fait en un temps $\mathcal{O}(nm)$. Ce sont des bornes supérieures, mais cela laisse supposer que les méthodes issues de la logique offrent de meilleures complexités.

On pourra trouver dans [WR88] d'autres exemples d'utilisation de la logique pour résoudre des problèmes de supervision.

*

* *

5 Conclusion

L'introduction de la logique dans des situations de contrôle discret semble permettre de résoudre certains problèmes, avec une meilleure complexité que pour les algorithmes classiques.

Il serait intéressant de voir si l'on peut généraliser les algorithmes 3.4.1 et 3.4.2 à des classes de langages plus importantes grâce à la logique, ou même trouver des algorithmes totalement différents.

*

* *

Index

Références

[BGK+90] R.D. Brandt, V. Garg, R. Kumar, F. Lin, S.I. Marcus, and W.M. Wonham, *Formulas for calculing supremal controllable and normal sublanguages*, Systems and Control Letters **15** (Aug 1990), 111–117.

[Eil74] S. Eilenberg, *Automata, languages and machines*, vol. A, Academic Press, 1974.

[Gau95] S. Gaubert, *Introduction aux systèmes dynamiques à événements discrets*, Notes de Cours ENSTA, Nov 1995.

[HU79] J.E. Hopcroft and J.D. Ullman, *Introduction to automata theory, languages, and computation*, Addison-Wesley, 1979.

[KG95] R. Kumar and V.K. Garg, *Modeling and control of logical discrete event systems*, Kluwer Academic Publishers, 1995.

[RW87] P.J. Ramadge and W.M. Wonham, *Supervisory control of a class of discrete event processes*, SIAM J. Control and Optimization **25** (Jan 1987), no. 1, 206–230.

[RW89] P.J. Ramadge and W.M. Wonham, *The control of discret event systems*, IEEE Proceedings: Special issue on Discrete Event Systems **77** (Jan 1989), no. 1, 81–97.

[WR87] W.M. Wonham and P.J. Ramadge, *On the supremal controllable sublanguage of a given language*, SIAM J. Control and Optimization **25** (May 1987), no. 3, 637–659.

[WR88] W.M. Wonham and P.J. Ramadge, *Modular supervisory control of discrete event systems*, Math. Control, Sign. and Syst. **1** (Jan 1988), no. 1, 13–30.