Chapitre 5

Introduction

On a considéré dans l'introduction de ce cours deux classes de syst`emes.

(a) Les syst`emes déterministes présentant des phénom`enes de synchronisation et de saturation, et dont le comportement *temporel* se représente par des équations linéaires récurrentes sur un dioïde réel (alg`ebre (max, +) ou (min, +)) Voir par exemple le limitateur de débit, les processus d'assemblage ...

(b) Des syst`emes plus généraux présentant en sus des phénom`enes de concurrence (par exemple, deux utilisateurs partageant une ressource). Ces syst`emes ne peuvent plus être considérés comme "déterministes", dans la mesure o`u plusieurs événements peuvent se produire (plusieurs trajectoires partent d'un même état initial). Cependant, on a vu sur des cas particuliers que l'ensemble des trajectoires possibles était déterminé comme l'ensemble des chemins d'un graphe, et était solution d'équations linéaires sur un dioïde de langages.

5.1 Langages et Systèmes à Evénements Discrets

5.1.1 Spécification d'un Système a Evenements Discrets par un langage

Soit Σ un alphabet *fini*. Chaque lettre de Σ représente un événement. Une suite d'événements correspond `a un mot $w = a_1 \dots a_n \in \Sigma^*$, ce qui se lit : d'abord l'événement

a_1 puis l'événement $a_2, \ldots$. Le mot vide (unité pour le produit de concaténation) sera noté e. On rappelle que u est un *préfixe* de w s'il existe v tel que $uv = w$, ce que l'on note $u \leq w$. L'ordre $\leq$ est qualifié de *préfixiel*. Une partie $X \subset \Sigma^*$ est dite *préfixielle* si

$$w \in X \text{ et } u \leq w \Rightarrow u \in X \ ,$$

(X est stable par préfixe). Etant donné une partie $X \subset \Sigma^*$, on note

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

la *clôture préfixielle* de X. On a évidemment $\overline{\overline{X}} = \overline{X}$, $\overline{X} \supset X$, $X = \overline{X} \Leftrightarrow X$ est une partie préfixielle, ce qui justifie le terme "clôture"[1]. On spécifie le comportement d'un SED par une partie $L \subset \Sigma^*$ (on dit aussi un *langage*). Si $w = a_1 \ldots a_n \in L$, cela signifie que la suite d'événements $a_1 \ldots a_n$ est compatible avec le fonctionnement du syst`eme. Il est clair que tout préfixe de w est également admissible. *On spécifiera donc un système à événement discrets par une partie préfixielle L de Σ^*.*

EXEMPLE 5.1.1.1. Soit un poinçonneur automatique de tickets de métro. Soit l'alphabet $\Sigma = \{a, b, c\}$. On associe aux lettres a, b, c les événements suivants :

a : ticket introduit
b : ticket accepté
c : ticket refusé

On se convainc que l'ensemble des trajectoires possible est représenté par le langage suivant :

$$\begin{aligned} L &= \overline{(ab \oplus ac)^*} = \overline{ab \oplus ac \oplus (ab)^2 \oplus (ac)^2 \oplus abac \oplus acab \oplus \ldots} \\ &= a \oplus ab \oplus ac \oplus aba \oplus aca \oplus abab \oplus \ldots \\ &= (e \oplus a)(ab \oplus ac)^* \end{aligned} \tag{5.1}$$

Selon l'usage, on a identifié dans ce qui préc`ede la lettre x et le singleton $\{x\} \subset \Sigma^*$, et l'on a noté l'union $\oplus$. Ainsi, $a \oplus b$ désigne le langage $\{a\} \cup \{b\} = \{a, b\} \subset \Sigma^*$. La notation additive est justifiée par le fait que $(\mathcal{P}(\Sigma^*), \cup, .)$ constitue un dioïde (c'est le dioïde des langages sur Σ). On peut déj`a noter au vu de (5.1) que le langage décrivant le syst`eme n'est pas quelconque, mais constitue une *partie rationnelle* de Σ^*, i.e. que L s'écrit comme une expression finie faisant intervenir des parties finies et les opération $\oplus, \otimes, *$ (cf. Chap. 2,2.2.4.7).

Notre probl`eme est ici de manier effectivement (i.e. d'un point de vue calculatoire) des langages a priori infinis. On peut dans certains cas comme ci-dessus décrire un langage infini `a l'aide d'une expression rationnelle. On peut aussi le décrire comme l'ensemble des mots reconnus par un automate. Nous précisons ce point de vue dans le numéro suivant.

5.1.2 Rappels sur les automates

DEFINITION: On appelle automate (d´eterministe) sur l'alphabet Σ un quadruplet

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

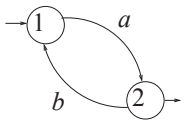
o`u Q est un ensemble (ensemble des états), $q_0 \in Q$ un état initial, $\delta : \Sigma \times Q \to Q$ une fonction en générale partielle (i.e. non définie pour certaines valeurs) qualifiée de *fonction de transition*, et $Q_a \subset Q$ un ensemble d'états dits *finaux* ou encore appelés *états d'acceptation*.

---

[1] $\circ f = \quad \geq$

Dans la suite, les automates ne seront pas supposés finis, sauf spécialement lorsqu'il s'agira de calcul effectif.

La mani`ere la plus simple de spécifier un automate est graphique : l'ensemble $Q$ des états est représenté par l'ensemble des sommets d'un graphe orienté. On dessine une fl`eche $q \to q'$ valuée par la lettre $a \in \Sigma$ ssi $\delta(a, q) = q'$. La fonction de transition δ peut être donnée sous forme de tableau. L'état initial est repéré par une fl`eche entrante. Les états finaux par des fl`eches sortantes. Par exemple, prenons $\Sigma = \{a, b\}$, $Q = \{1, 2\}$, $q_0 = 1$ et $Q_a = \{2\}$. donn´ee Soit la fonction de transition δ par le tableau `a droite de la Figure 1 : par exemple,



	1	2
1		a
2	b	

fonction de transition

FIG.1: Repr´esentation graphique d'un automate et repr´esentation matricielle

la premi`ere ligne signifie que $\delta(a, 1) = 2$ et que $\delta(b, 1)$ n'est pas défini. On obtient alors le graphe `a gauche de la Figure.

On étend la fonction de transition en une fonction partielle sur $\Sigma^* \times Q$ en posant :

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

(rappelons que e désigne le mot vide). Par exemple, pour l'automate 5.1, $\delta(ba, 2) = \delta(a, \delta(b, 2)) = \delta(a, 1) = 2$. Dans la suite, on écrira $\delta(w, q)!$ pour "$\delta(w, q)$ est définie". Graphiquement, si $w = a_1 \dots a_n$, $\delta(w, q)$ est défini ssi il existe un chemin dans le graphe partant de q et portant les labels successifs $a_1 \dots a_n$. Si ce chemin conduit au sommet q', évidemment, $\delta(w, q) = q'$. On dira que le mot w est *reconnu* par l'automate $\mathcal{A}$ si $\delta(w, q_0) \in Q_a$. Nous noterons

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

le *langage reconnu* par l'automate (i.e. l'ensemble des mots envoyant l'état initial sur un état final).

Du point de vue des syst`emes `a événements discrets, un automate représente un syst`eme dynamique, et l'on peut s'interesser `a toutes ses trajectoires, y compris celles qui n'arrivent pas dans un état final. Soit donc $\overline{\mathcal{A}} = \{Q, \delta, q_0, Q\}$ l'automate obtenu `a partir de $\mathcal{A}$ en rendant tous les états finaux. On s'intéressera particuli`erement au langage suivant

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

que nous qualifierons de *comportement de l'automate*[2] $\mathcal{A}$. On a clairement

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

EXEMPLE 5.1.2.2. Pour l'automate de la Figure 5.1, un mot w est reconnu ssi il conduit de l'état 1 `a l'état 2, i.e. ssi il commence par un nombre quelconque de circuits $ab$ et s'ach`eve par une lettre a, soit

5.

$$\mathcal{L}(\mathcal{A}) = a \oplus aba \oplus ababa \oplus \ldots = (ab)^* a \ .$$

De même, $\overline{\mathcal{L}}(\mathcal{A})$ désigne l'ensemble des chemins partant de 1 et arrivant dans un état quelconque, soit

$$\overline{\mathcal{L}}(\mathcal{A}) = e \oplus a \oplus ab \oplus aba \oplus abab \oplus \ldots = (ab)^*(e \oplus a) \ .$$

On définit les ensembles des états respectivement *accessibles* et *co-accessibles* par

$$Q_{\text{ac}} = \{q \in Q \mid \exists w \in \Sigma^*,\ \delta(w, q_0) = q\},$$
$$Q_{\text{co}} = \{q \in Q \mid \exists w \in \Sigma^*,\ \delta(w, q) \in Q_a\} \ .$$

L'automate est accessible ssi tous ses états le sont. Définition analogue pour la co-accessibilité. Par exemple, l'automate de la Figure 5.1 est accessible et co-accessible. L'automate est dit *émondé* si $Q = Q_{\text{ac}} = Q_{\text{co}}$. Soit

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

On définit l'automate

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

qui est clairement émondé et reconnaît le même langage que $\mathcal{A}$. On a trivialement :

FAIT 5.1.2.3. *Si $\mathcal{A}$ est co-accessible, alors $\overline{\mathcal{L}}(\mathcal{A}) = \overline{\mathcal{L}(\mathcal{A})}$.*

Autrement dit, sous l'hypoth`ese de co-accessibilité, $\overline{\mathcal{L}}(\mathcal{A})$ n'est autre que l'ensemble des mots qui se prolongent en des mots reconnus.

EXEMPLE 5.1.2.4. L'automate de la Figure 5.1 est émondé. Pour l'automate $\mathcal{A}$ de la Figure 5.2, on a

$$Q_{\text{ac}} = \{1, 2, 3\},\ \ Q_{\text{co}} = \{1, 2, 4\}$$

On a représenté `a droite de la Figure l'automate émondé $\text{Em}(\mathcal{A})$. On voit facilement que

$$\mathcal{L}(\mathcal{A}) = ba^* = \mathcal{L}(\text{Em}(\mathcal{A})), \quad \overline{\mathcal{L}}(\mathcal{A}) = e \oplus ba^* \oplus aa^* \neq \overline{\mathcal{L}(\mathcal{A})} = \overline{\mathcal{L}}(\text{Em}(\mathcal{A})) = e \oplus ba^* \ .$$

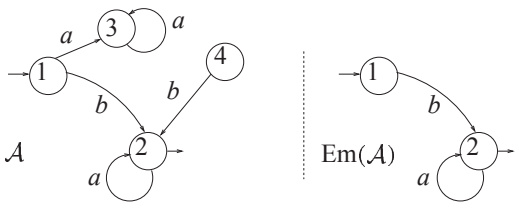


FIG.2 : Un automate et l'automate emond´´ e associ´e

EXEMPLE 2. On a repr´esent´e sur la Figure 5.3 l'automate reconnaissant le langage du poinc¸onneur de m´etro. Le lecteur pourrait se demander quel est le lien entre les langages rationnels et les langages reconnus par des automates. On dira qu'un langage est r´egulier s'il est reconnu par un automate fini (i.e. avec un nombre fini d'´etats). Nous citons pour m´emoire le plus c´el`ebre des th´eor`emes de la th´eorie des langages formels :

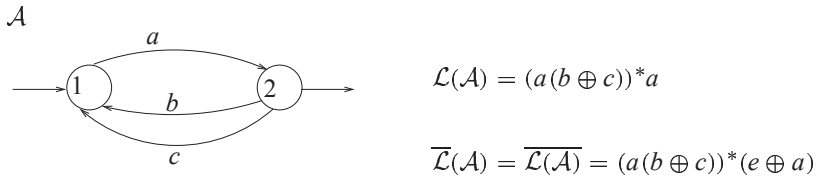


FIG. 3: J'suis l'poinc¸onneur des Lilas

THEOR´ EME` (KLEENE). Soit Σ *un alphabet fini. Un langage est régulier ssi il est* rationnel.

Nous ne prouverons pas compl`etement ce théor`eme. Notons simplement que l'implication régulier $\Rightarrow$ rationnel a déj`a été prouvé au chapitre 2 lorsque l'on a montré que l'ensemble des chemins entre deux sommets d'un graphe (fini) est une partie rationnelle. Le langage reconnu par l'automate n'est autre que l'ensemble des chemins du sommet initial aux sommets finaux. Pour la réciproque, il suffit de vérifier que l'ensemble des langages réguliers est stable par union, produit et étoile, ce qui revient `a fabriquer des automates finis reconnaissant $A \oplus B$, $AB$ et $A^*$ `a partir d'automates reconnaissant A et B. Nous renvoyons le lecteur `a [33] pour la preuve.

5.1.3 Supervision d'un automate

On consid`ere une partition de l'ensemble des événements :

$$\Sigma = \Sigma_{\text{in}} \cup \Sigma_{\text{c}} \ .$$

Σ_{in} désigne l'ensemble des événements "incontrôlables" (une machine tombe en panne, un client arrive, ...) Σ_{c} désigne l'ensemble des événements que l'on peut interdire ou bien autoriser. Par exemple, si l'événement $a \in \Sigma_{\text{c}}$ représente l'entrée d'un train sur une portion de voie, on peut interdire l'événement a au moyen d'un feu rouge.

DÉFINITION 5.1.3.1 (SUPERVISEUR). On appelle *superviseur* une application

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

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

$f(w)$ s'interpr`ete comme *l'ensemble des événements autorisés* connaissant les événements passés donnés par le mot $w$. On impose $f(w) \supset \Sigma_{\text{in}}$ (les événements incontrôlables sont autorisés de mani`ere permanente).

DÉFINITION 5.1.3.2. On appelle comportement d'un automate $\mathcal{A}$ soumis au superviseur f le langage préfixiel $\overline{\mathcal{L}}_f(\mathcal{A})$ de Σ^* défini récursivement comme suit :

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

Le *langage contrôlé* par f est par définition :

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

i.e. l'ensemble des mots du comportement supervisé qui aboutissent dans un état final. L'automate supervisé est dit *non bloquante* si $\overline{\mathcal{L}_f(\mathcal{A})} = \overline{\mathcal{L}}_f(\mathcal{A})$.

La condition de non blocage exprime que toute tâche commencée par le syst`eme supervisé, i.e. tout mot de $\overline{\mathcal{L}}_f(\mathcal{A})$ doit pouvoir se completer en une tâche achevée (mot de $\mathcal{L}_f(\mathcal{A})$).

Pour rendre effective la définition ci-dessus d'un superviseur, on peut introduire un automate "contrôleur" réalisant cette supervision. Soit un automate $\mathcal{S} = (X, \xi, x_0, X_a)$, et $\phi : X \to \mathcal{P}(\Sigma)$ une application telle que $\forall x \in X,\ \phi(x) \supset \Sigma_{\text{in}}$. On dira que $\mathcal{S}$ réalise le superviseur f si pour tout mot $w \in \overline{\mathcal{L}}_f(\mathcal{A})$:

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

Autrement dit, on fait lire au contrôleur $\mathcal{S}$ le même mot que l'automate `a contrôler $\mathcal{A}$, et l'ensemble $f(w)$ des événements autorisés est un feedback de l'état du contrôleur $\mathcal{S}$. De mani`ere plus précise, considérons l'ensemble des états (resp. l'état initial, resp. les états finaux) $X \times Q$ (resp. (x_0, q_0), resp. (X_a, Q_a)). Soit la fonction de transition $\xi \times \delta$ définie par

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

On notera $\mathcal{S}/\mathcal{A}$ l'automate représentant "$\mathcal{A}$ supervisé par $\mathcal{S}$", soit

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

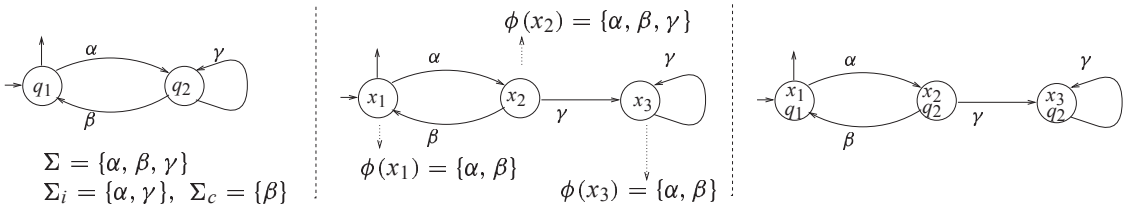
oú Ac dénote la partie accessible d'un automate. Il résulte du fait 5.1.2.3 que l'automate supervisé est non bloquant ssi $\mathcal{S}/\mathcal{A}$ est co-accessible.

EXEMPLE .3. Soit l'automate de la Figure 5.4,(a). Consid´erons le $\mathcal{S}$ de la superviseur Figure 4,(b). On a repr´esent´e $\mathcal{S}/\mathcal{A}$ en (c). On a

$$\mathcal{L}(\mathcal{A}) = (\alpha\gamma^*\beta)^*, \quad \overline{\mathcal{L}}(\mathcal{A}) = \overline{\mathcal{L}(\mathcal{A})} \ .$$

$$\overline{\mathcal{L}}_f(\mathcal{A}) = \overline{\mathcal{L}}(\mathcal{S}/\mathcal{A}) = (\alpha\beta)^*(e \oplus \alpha\gamma^*) \ .$$

$$\mathcal{L}_f(\mathcal{A}) = (\alpha\beta)^* \ .$$



(a) Automate $\mathcal{A}$ (b) Contr^oleur $\mathcal{S}$ (c) Automate supervis´e $\mathcal{S}/\mathcal{A}$

FIG. 4: Un automate supervis´e

5.2 Synthèse d'un superviseur

5.2.1 Contrôlabilité

Etant donné un automate $\mathcal{A}$ et un langage K, on se demande `a quelle condition il existe un superviseur f tel que $\mathcal{L}_f(\mathcal{A}) = K$. On introduit pour cela la notion suivante.

DÉFINITION 5.2.1.1 (CONTRÔLABILITÉ). Le langage $K \subset \Sigma^*$ est dit contrôlable si

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

Autrement dit, tout préfixe de K suivi d'un événement incontrôlable admissible pour le comportement de $\mathcal{A}$ reste préfixe de K, o`u "les événements incontrollables ne sont pas illégaux". La proposition fondamentale qui suit justifie le terme de "contrôlable".

PROPOSITION 5.2. Soit $\mathcal{A}$ *un automate. Pour tout* $K \subset \overline{\mathcal{L}}(\mathcal{A})$ *non vide, les deux assertions 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. (i)⇒(ii) : Résulte évidemment de la définition de $\overline{\mathcal{L}}_f(\mathcal{A})$ (cf. (5.5)).

(ii)⇒(i). Soit $\mathcal{B} = (X, \xi, x_0, X)$ un automate émondé reconnaissant K (comme K est préfixiel, on peut prendre tous les états finaux). Définissons, pour $x \in X$,

$$\begin{aligned} \Sigma_x^0 &= \{\sigma \mid \exists s \in K,\ \xi(s, x_0) = x,\ s\sigma \in \overline{\mathcal{L}}(\mathcal{A}) \text{ et } s\sigma \notin K\} \\ \Sigma_x^1 &= \{\sigma \mid \exists s \in K,\ \xi(s, x_0) = x \text{ et } s\sigma \in K\} \end{aligned}$$

De mani`ere intuitive, Σ_x^0 représente les événements qu'il faut prohiber (car il sortent de la cible K) et Σ_x^1 est l'ensemble des événements que l'on peut autoriser.

Comme l'automate est émondé, l'état x est accessible depuis x_0, donc la condition $\exists s, \xi(s, x_0) = x$ est toujours remplie. Notons tout d'abord que $\Sigma_x^0 \cap \Sigma_x^1 = \emptyset$. Supposons en effet $\sigma \in \Sigma_x^0 \cap \Sigma_x^1$ avec

$$\begin{aligned} &s^0 \in K, \xi(s^0, x_0) = x,\ s^0\sigma \notin K \\ &s^1 \in K, \xi(s^1, x_0) = x,\ s^1\sigma \in K \ . \end{aligned}$$

Alors, $\xi(\sigma, x) = \xi(\sigma, \xi(s^0, x_0)) = \xi(s^0\sigma, x_0)$ qui n'est pas défini (car $s^0\sigma \notin K$), mais par ailleurs, $\xi(\sigma, x) = \xi(\sigma, \xi(s^1, x_0)) = \xi(s^1\sigma, x_0)$ qui est bien défini : contradiction. K étant contrôlable, on a $\Sigma_x^0 \subset \Sigma_c$. Soit $\phi : X \to \Sigma$ une application vérifiant $\phi(x) \cap \Sigma_x^0 = \emptyset$, $\phi(x) \supset \Sigma_x^1 \cup \Sigma_{\text{in}}$. On définit le superviseur f par $f(w) = \phi(\xi(w, x_0))$. On va montrer que $\overline{\mathcal{L}}_f(\mathcal{A}) = K$. Clairement,

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

Posons pour un langage M :

$$M^{(j)} = M \cap \Sigma^{(j)} = \{w \in M \mid |w| = j\}$$

(ensemble des mots de longueur j). On montre par récurrence sur j que $\overline{\mathcal{L}}_f(\mathcal{A})^{(j)} = K^{(j)}$. Cas $j = 1$: si $\sigma \in K$, on a $\sigma \in \Sigma_{x_0}^1$, donc $\sigma \in \overline{\mathcal{L}}_f(\mathcal{A})$. Supposons par récurrence que

$$\overline{\mathcal{L}}_f^{(j)}(\mathcal{A}) = K^{(j)} \ .$$

Soit $s\sigma \in K^{(j+1)}$ avec $s \in K^{(j)} = \overline{\mathcal{L}}_f^{(j)}$. Comme $x = \xi(s, x_0)$ est défini, $s\sigma \in K$ entraîne que $\sigma \in \Sigma_x^1$, donc $\sigma \in f(s)$, d'o`u $s\sigma \in \overline{\mathcal{L}}_f(\mathcal{A})$ et $\overline{\mathcal{L}}_f^{(j+1)}(\mathcal{A}) \supset K^{(j+1)}$. L'autre inclusion étant claire d'apr`es (Lf), le r´esultat est □

acquis.

Le probl`eme de supervision pour le langage reconnu $\mathcal{L}$ se résoud de mani`ere analogue.

PROPOSITION .3. Soit $\mathcal{A}$ *un automate non-bloquant (i.e. co-accessible). Pour tout* $K \subset \mathcal{L}(\mathcal{A})$ *non vide, les deux assertions suivantes sont équivalentes :*

1. *il existe un superviseur* f *tel que l'autamate supervisé soit non bloquant et* $\mathcal{L}_f(\mathcal{A}) = K$
2. K *est contrôlable et* $\overline{K} \cap \mathcal{L}(\mathcal{A}) = K$.

La preuve, similaire, est laissée au lecteur.

EXEMPLE 5.2.1.4. Considérons l'automate représentant un magasin de stockage d'une capacité de trois unités `a gauche de la Figure 5.5 (cf. 1,1.1). Supposons que les deux derni`eres places de stock soient indisponibles, et que l'on veuille limiter la capacité du magasin `a une unité. Le langage K admissible est alors la partie préfixielle reconnue par l'automate $\mathcal{B}$ de la Figure 5.5, soit

$$K = (ab)^*(e \oplus a) = \overline{K} \ .$$

Supposons l'événement a contrôlable et b incontrôlable :

$$\Sigma_{\text{in}} = \{b\}, \quad \Sigma_{\text{c}} = \{a\} \ .$$

On a

$$\overline{K}\Sigma_i \cap \overline{\mathcal{L}}(\mathcal{A}) = (ab)^*(e \oplus a)b \cap \overline{\mathcal{L}}(\mathcal{A}) \ .$$

On a $(ab)^*ab \subset \overline{K}$ et $(ab)^*b \cap \overline{\mathcal{L}}(\mathcal{A}) = \emptyset$, donc

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

ce qui montre que le langage K est contrôlable. Prenons comme superviseur l'automate $\mathcal{B} = (\{0', 1'\}, \xi, 0', \{0', 1'\})$, avec

$$\phi(0') = \{a, b\}, \quad \phi(1) = \{b\}$$

et posons $f(w) = \phi(\xi(w, x_0))$ On a alors $\overline{\mathcal{L}}_f(\mathcal{A}) = K$. Supposons maintenant $\Sigma_{\text{in}} =$

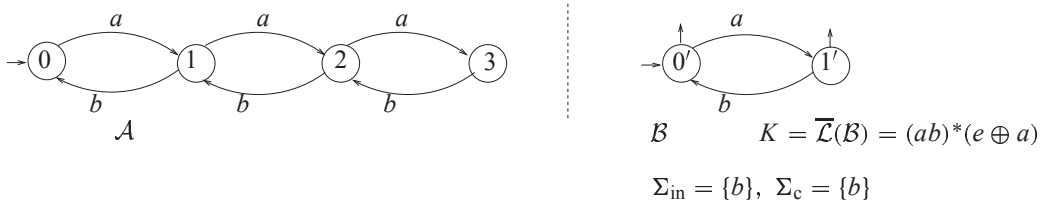


FIG. 5.5: Le langage $K = \overline{\mathcal{L}}(\mathcal{B})$ est contrôlable

$\{a\}, \ \Sigma_{\text{c}} = \{b\}$. On a alors

$$\overline{K}\Sigma_{\text{in}} \cap \overline{\mathcal{L}}(\mathcal{A}) = (ab)^*(e \oplus a)a \cap \overline{\mathcal{L}}(\mathcal{A}) \supset \{a^2\}$$

et $a^2 \notin \overline{K}$, ce qui montre que le syst`eme n'est plus contrôlable. Tout cela est intuitivement tr`es clair : l'événement a revient `a stocker une pi`ece supplémentaire. Si l'on ne contrôle pas a, on ne peut pas limiter la capacité du stock.

5.2.2 Sous-langage contrôlable maximal

Soit $K \subset \overline{\mathcal{L}}(\mathcal{A})$. Même si K n'est pas contrôlable, il existe une "approximation contrôlable" naturelle de K. Il s'agit du sous langage contrôlable maximal, dont nous montrons ci apr`es l'existence. Soit $C(K)$ l'ensemble des parties de K contrôlables, i.e.

$$C(K) = \{J \subset K \mid \overline{J}\Sigma_{\text{in}} \cap \overline{\mathcal{L}}(\mathcal{A}) \subset \overline{J}\} \ . \tag{5.8}$$

Comme $\emptyset$ est contrôlable, $C(K)$ est non vide. Au vu de la condition (5.6), il est immédiat que cet ensemble est stable par union (éventuellement infinie). Donc il admet un plus grand élément. On notera

$$K^{\uparrow} = \sup C(K) = \max C(K)$$

ce plus grand élément (sous-langage contrôlable maximal). Il faut bien voir que la définition de $K^{\uparrow}$ comme borne sup ne permet pas de le calculer. On va donner ci-apr`es une caractérisation alternative de $K^{\uparrow}$ comme point fixe d'un opérateur, ce qui permettra de le calculer par une technique élémentaire d'itération de point fixe. Définissons l'application $\Omega : \mathcal{P}(\Sigma^*) \to \mathcal{P}(\Sigma^*)$ par

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

On obtient facilement les expressions équivalentes de Ω :

$$\Omega(J) = \{t \in K \mid \overline{t}\Sigma_{\text{in}} \cap \overline{\mathcal{L}}(\mathcal{A}) \subset \overline{J}\} = K \cap \sup\{T \subset \Sigma^* \mid \overline{T}\Sigma_{\text{in}} \cap \overline{\mathcal{L}}(\mathcal{A}) \subset \overline{J}\} \ . \tag{5.9}$$

PROPOSITION 5.2.2.1. *On a $K^{\uparrow} = \Omega(K^{\uparrow})$. En outre,*

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

En d'autres termes, $K^{\uparrow}$ est le plus grand point fixe de l'opérateur Ω.

Démonstration. . En fait, il s'agit d'un lemme tout `a fait général sur les structures ordonnées. Soit $\mathcal{E}$ un treillis complet, et soient $f, g : \mathcal{E} \to \mathcal{E}$ deux applications croissantes, f préservant les sup quelconques :

$$f(\bigvee_{u \in \mathcal{U}} u) = \bigvee_{u \in \mathcal{U}} f(u) \quad \text{pour tout } \mathcal{U} \subset \mathcal{E} \tag{5.10}$$

($\mathcal{U}$ éventuellement infinie). Définissons de mani`ere plus générale :

$$\mathcal{C}(K) = \{J \subset K \mid f(J) \leq g(J)\}, \quad K^{\uparrow} \stackrel{\text{def}}{=} \sup \mathcal{C}(K) \ ,$$

ainsi que

$$\mathcal{C}(K, J) = \{M \subset K \mid f(M) \leq g(J)\} \ , \omega(K, J) \stackrel{\text{def}}{=} \sup \mathcal{C}(K, J) \ .$$

Il suffit clairement de montrer que

$$J = \omega(K, J) \quad \Rightarrow \quad J \leq K^{\uparrow} \tag{5.11}$$

$$K^{\uparrow} \quad = \quad \omega(K, K^{\uparrow}) \tag{5.12}$$

Il résulte de (5.10) que $K^{\uparrow} \in \mathcal{C}(K, K^{\uparrow})$, donc

$$K^{\uparrow} \leq \omega(K, K^{\uparrow}) \ . \tag{5.13}$$

D'autre part, en spécialisant $\omega(K, J) \in \mathcal{C}(K, J)$ `a $J = K^{\uparrow}$, il vient

$$f(\omega(K, K^{\uparrow})) \leq g(K^{\uparrow}) \leq g(\omega(K, K^{\uparrow}))$$

(par (5.13) et par croissance de g). Ainsi, $\omega(K, K^{\uparrow}) \in \mathcal{C}(K)$, d'o`u l'on tire $\omega(K, K^{\uparrow}) \leq K^{\uparrow}$. On a montré (5.12). Enfin, $J = \omega(K, J)$ entraîne $J \in \mathcal{C}(K)$, d'o`u $J \leq K^{\uparrow}$. □

A partir de la Proposition 5.2.2.1, il est naturel d'appliquer un argument de point fixe et de définir la suite :

$$K_0 = K, \ K_{j+1} = \Omega(K_j) \ . \tag{5.14}$$

En vertu de la décroissance de la suite $\{K_n\}$, on a l'existence de

$$K_{\infty} = \lim_{n \to \infty} K_n = \bigcap_{n \geq 0} K_n$$

et l'on se demande bien sûr si K_{∞} est le point fixe requis. On a $K^{\uparrow} = \Omega(K^{\uparrow}) \subset K = K_0$. Donc par récurrence $K^{\uparrow} = \Omega(K^{\uparrow}) \subset K_j$ et donc

$$K^{\uparrow} \subset \bigcap_{j \geq 0} K_j = K_{\infty} \ . \tag{5.15}$$

En général, K_{∞} n'a aucune raison d'être un point fixe et cette inclusion est stricte. Nous donnons une condition suffisante dans la section suivante.

5.2.3 Synthèse d'un automate reconnaissant $K^{\uparrow}$

Nous montrons que la convergence de la suite K_n vers le sous-langage contrôlable maximal est acquise sous les hypoth`eses suivantes

5.2.3.1 *(Hypothèses)*.

1. $\mathcal{A}$ est fini et K est un langage régulier (i.e. reconnu par un automate fini)
2. $K \subset \overline{\mathcal{L}}(\mathcal{A})$
3. $\overline{\mathcal{L}}(\mathcal{A}) = \mathcal{L}(\mathcal{A})$.

L'hypoth`ese (i) est la plus simple qui permette de manipuler effectivement des langages. L'hypoth`ese (ii) signifie que le langage "cible" K est une restriction du comportement admissible du syst`eme $\overline{\mathcal{L}}(\mathcal{A})$, cas auquel il est naturel de ce ramener au besoin en remplaçant K par $K \cap \overline{\mathcal{L}}(\mathcal{A})$. La condition (iii) enfin signifie qu'on se préoccupe de tous les événements (et pas seulement des mots acceptés). Cette condition est aussi souvent réalisée. Le lecteur pourra se reporter aux exemples de la section suivante pour juger de la classe de probl`eme prise en compte. Moyennant ces hypoth`eses, on va montrer que chacun des langages K_j de la suite définie ci-dessus et convergeant vers K_{∞} est régulier. On va construire pour cela `a chaque étape un automate $\mathcal{C}_j$ reconnaissant $K_j$, et donner la r`egle produisant $\mathcal{C}_{j+1}$ a partir de $\mathcal{C}_j$. On montrera que la suite $\mathcal{C}_j$ converge en un nombre fini d'itérations vers un automate $\mathcal{C}_{\infty}$ reconnaissant $K_{\infty} = K^{\uparrow}$.

Dans la suite, on notera

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

(c'est l'ensemble des événements possibles dans l'état q). Soient

$$\mathcal{A} = (Q, \delta, q_0, Q_a) \text{ et } \mathcal{B} = (X, \xi, x_0, X_a)$$

deux automates émondés 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

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

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

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

Par exemple, l'automate de la Figure 5.6,(b) raffine celui de la Figure 5.6,(a), la fonction h étant tabulée sur la Figure. L'intérêt des l'hypoth`eses 5.2.3.1 est que l'on peut fabriquer un

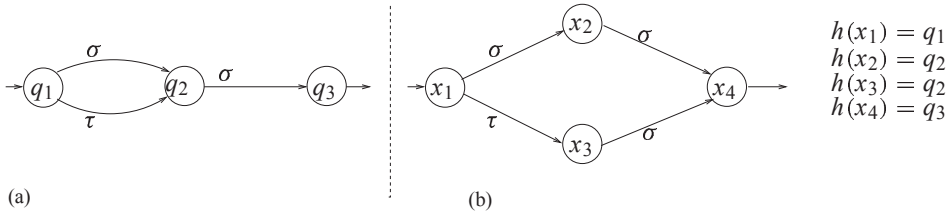


FIG. 6: Un automate et un raffinement

automate raffinant $\mathcal{A}$ et reconnaissant K

LEMME 5.2.3.2. *Moyennant l'hypothèse 5.2.3.1, il existe un automate fini $\mathcal{C}_0$ raffinant $\mathcal{A}$ tel que $\mathcal{L}(\mathcal{C}_0) = K$.*

Démonstration. K étant régulier, il existe un automate fini $\mathcal{B} = (Z, \alpha, z_0, Z_a)$ tel que $\mathcal{L}(\mathcal{B}) = K$. On définit le produit $\mathcal{A} \times \mathcal{B}$ des automates $\mathcal{A}$ et $\mathcal{B}$ par

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

o`u

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

Il est clair que $\mathcal{A} \times \mathcal{B}$ reconnaît l'intersection des langages reconnus par $\mathcal{A}$ et par $\mathcal{B}$. Comme $K \subset \overline{\mathcal{L}}(\mathcal{A})$, on a

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

On ne change pas le langage reconnu par un automate en l'émondant. En posant

$$\mathcal{C}_0 = \mathrm{Em}(\mathcal{A} \times \mathcal{B}) \ ,$$

on a un automate reconnaissant K et raffinant $\mathcal{A}$. En effet, trivialement

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

□

On consid`ere `a nouveau la suite de langages $\{K_j\}$ définie plus haut.

LEMME 5.2.3.3. *Soit $\mathcal{A}$ un automate fini émondé et $\mathcal{C}_j$ un automate fini émondé raffinant $\mathcal{A}$ tel que $K_j = \mathcal{L}(\mathcal{C}_j)$. Alors $w \in \Omega(K_j)$ ssi $w \in K_j$ et*

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

Démonstration. Les propriétés suivantes sont équivalentes

$$
\begin{array}{l}
w \in \Omega(K_j) \\
w \in K_j \text{ et } \overline{w}\Sigma_{\text{in}} \cap \overline{\mathcal{L}}(\mathcal{A}) \subset \overline{K_j} \\
w \in K_j \text{ et } \forall u \leq w,\ u\Sigma_{\text{in}} \cap \overline{\mathcal{L}}(\mathcal{A}) \subset \overline{K_j} \\
w \in K_j \text{et } \forall u \leq w,\ \forall \sigma \in \Sigma_{\text{in}},\ u\sigma \in \overline{\mathcal{L}}(\mathcal{A}) \Rightarrow u\sigma \in \overline{K_i} \\
w \in K_j \text{ et } \forall u \leq w,\ \forall \sigma \in \Sigma^{\mathcal{A}}(\delta(u, q_0)) \cap \Sigma_{\text{in}},\ u\sigma \in \overline{K_j} \\
w \in K_j \text{ et } \forall u \leq w,\ \Sigma^{\mathcal{A}}(\delta(u, q_0)) \cap \Sigma_{\text{in}} \subset \Sigma^{\mathcal{C}_j}(\xi(u, x_0))
\end{array}
$$

d'o`u la conclusion. □

Ainsi, on peut 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 suivante :

$$\Sigma^{\mathcal{A}}(h(x)) \cap \Sigma_{\text{in}} \subset \Sigma^{\mathcal{C}_j}(x) \quad . \tag{5.16}$$

Soient précisément

$$X' = \{x \in X \mid \ \Sigma^{\mathcal{A}}(h(x)) \cap \Sigma_{\text{in}} \subset \Sigma^{\mathcal{C}_j}(x)\} \ ,$$

$$X'_a = X_a \cap X' \ ,$$

et soit la nouvelle fonction de transition

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

On définit alors

$$\mathcal{C}_{j+1} = \begin{cases} \text{Em}(X', \xi', x_0, X'_a) & \text{si } x_0 \in X' \\ \emptyset & \text{sinon.} \end{cases} \tag{5.17}$$

THÉORÈME 5.2.3.4. *Soit $\mathcal{C}_{j+1}$ comme en (5.17). Alors*

$$\overline{\mathcal{L}}(\mathcal{C}_{j+1}) = K_{j+1} = \Omega(K_j) \quad . \tag{5.18}$$

En outre, il existe un indice i tel que $\mathcal{C}_i = \mathcal{C}_{i+1}$ et l'on a

$$\overline{\mathcal{L}}(\mathcal{C}_i) = K^{\uparrow} \quad .$$

Autrement dit, on obtient en un nombre fini d'itérations un automate reconnaissant le sous-langage contrôlable maximal.

Démonstration. du Théor`eme. L'assertion (5.18) résulte immédiatement du Lemme 5.2.3.3. Montrons la convergence. Comme $\mathcal{C}_{j+1}$ s'obtient en supprimant des états de $\mathcal{C}_j$, la suite $\mathcal{C}_j$ converge en un nombre fini d'itérations (éventuellement vers l'automate vide), soit pour un certain i, $\mathcal{C}_i = \mathcal{C}_{i+1}$ et donc

$$K_i = \overline{\mathcal{L}}(\mathcal{C}_i) = \overline{\mathcal{L}}(\mathcal{C}_{i+1}) = K_\infty = \Omega(K_\infty) \quad .$$

K_∞ est donc point fixe de Ω, et comme $K^{\uparrow}$ est le point fixe maximal (Proposition 5.2.2.1), $K_\infty \subset K^{\uparrow}$. D'autre part, on a observé dans la section précédente que l'on a toujours $K^{\uparrow} \subset K_\infty$. □

5.3 Exemples

Nous donnons quelques exemples d'application de l'algorithme décrit ci-dessus.

5.3.1 Exemple 1

Soit $\Sigma = \{\alpha_1, \alpha_2, \beta\}$, $\Sigma_{in} = \{\beta\}$. $M = \overline{(\alpha_1\beta^2 \oplus \alpha_2)\beta^*}$, $K = \overline{\alpha_1\beta^2} \oplus \alpha_2\beta^*$. On a représenté sur la Figure 5.7 un automate $\mathcal{A}$ tel que $M = \overline{\mathcal{L}}(\mathcal{A})$. Le mot $\alpha_1\beta^2$ appartient `a K,

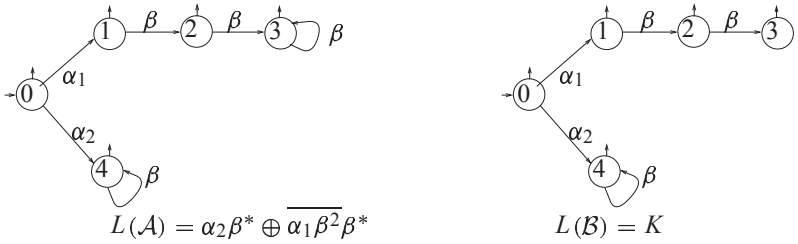


FIG. 5.7: Automates reconnaissant M et K

mais $\alpha_1\beta^3 \notin K$. Comme β est incontrôlable, cela entraîne que $\alpha_1\beta^2 \notin K_1$. De la sorte, il est intuitivement clair que $K^{\uparrow} = e \oplus \alpha_2\beta^*$. On peut le vérifier plus formellement `a partir de la construction de point fixe décrite plus haut. On a pour K_0 la table des transitions suivante :

	0	1	2	3	4	$\Sigma(h(x)) \cap \Sigma_{in}$	$\Sigma(x)$
0		α_1			α_2		$\alpha_1\alpha_2$
1			β			β	β
2				β		β	β
3						β	
4					β	β	β

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

	0	1	2	4	$\Sigma(h(x)) \cap \Sigma_{in}$	$\Sigma(x)$
0		α_1		α_2		$\alpha_1\alpha_2$
1			β		β	β
2			β		β	β
4				β	β	β

$$K_1 = \overline{\alpha_1\beta} \oplus \alpha_2\beta^* \ .$$

En poursuivant, on obtient

K_2 :

	0	1	4	$\Sigma(h(x)) \cap \Sigma_{in}$	$\Sigma(x)$
0		α_1	α_2		$\alpha_1\alpha_2$
1				β	β
4			β	β	β

$$K_2 = \overline{\alpha_1} \oplus \alpha_2\beta^* \ .$$

K_3 :

	0	4	$\Sigma(h(x)) \cap \Sigma_{in}$	$\Sigma(x)$
0		α_2		$\alpha_1\alpha_2$
4		β	β	β

$$K_3 = e \oplus \alpha_2 \beta^* \ .$$

La contrainte (5.16) est maintenant satisfaite pour tous les états : on a donc $K^\uparrow = e \oplus \alpha_2 \beta^*$.

5.3.2 Chat et souris dans un labyrinthe

On reprend l'automate de la Figure 1.2 du Chapitre 1 représentant les mouvements possibles du chat et de la souris. L'automate $\mathcal{B}$ représentant le comportement admissible du syst`eme s'obtient en supprimant tous les états diagonaux (o`u le chat et la souris se trouvent dans la même pi`ece), soit l'automate de la Figure 5.8. Considérons tout d'abord le cas fort

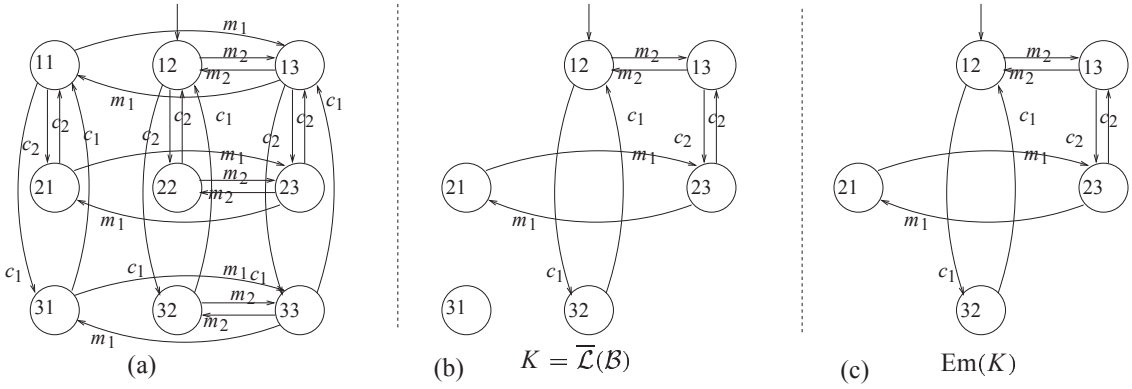


FIG. 8: Comportement admissible du chat et de la souris (c)

simple o`u $\Sigma_{in} = \emptyset$ : cela signifie que l'on contrôle toutes les trappes. Il suffit alors d'interdire les transitions conduisant aux états diagonaux d`es qu'on se trouve dans les états les précédant immédiatement. Soit ξ la fonction de transition de l'automate (c). Cela revient `a définir un contrôle de la forme

$$f(w) = \phi(\xi(w, i_0 j_0)), \quad i_0 j_0 = 12 \text{ (état initial)},$$

o`u le retour d'état ϕ ($\phi(q)$ décrit l'ensemble des transitions autorisées au point $q = ij$) est donnée par le tableau suivant :

12	$\phi(q) = \{c_1, m_2\}$
13	$\phi(q) = \{m_2, c_2\}$
21	$\phi(q) = \{m_1\}$
23	$\phi(q) = \{m_1\}$
32	$\phi(q) = \{c_1\}$

Autrement dit, on interdit `a la souris tout mouvement qui la ferait rentrer dans une pi`ece o`u le chat se trouve déj`a et de même pour le chat. Un cas plus intéressant est celui o`u $\Sigma_{in} = \{c_1\}$. Le chat peut alors de mani`ere incontrôlable emprunter la trappe q. Calculons le sous-langage contrôlable maximal. Pour l'automate de la Figure 5.8, on a le tableau suivant.

	12	13	21	23	32	$\Sigma(h(x)) \cap \Sigma_{in}$	$\Sigma(x)$
12		m_2			c_1	c_1	$m_1 \oplus c_1$
13	m_2			c_2		c_1	$m_1 \oplus c_2$
21	m_2			m_1			m_1
23		c_2	m_1				$m_1 \oplus c_2$
32	m_2					c_1	c_1

La contrainte $\Sigma(h(x)) \cap \Sigma_{\text{in}} \subset \Sigma(x)$ est violée pour $x = 13$. Lorsque l'on supprime cet état de l'automate $\mathcal{A}$, on obtient l'automate $\mathcal{C}$ `a gauche de la Figure 5.9, pour lequel les états 21 et 23 ne sont plus accessibles. L'automate émondé correspondant est représenté `a droite de la Figure. On vérifie en dressant un tableau analogue que la contrainte est maintenant satisfaite pour tous les états. Cette automate reconnaît donc le sous-langage commandable maximal de K. Ce langage est réduit `a $\not{\epsilon}$. Autrement dit, le chat peut circuler librement entre les pi`eces 1 et 3, ce qui est normal, l'événement c_1 étant incontrôlable. En outre, la souris est cantonnée dans la pi`ece 2, comme on aurait pu le voir de mani`ere tout `a fait élémentaire.

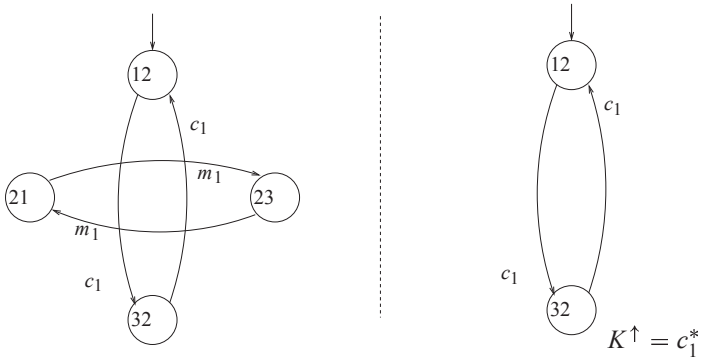


FIG. 5.9: Calcul de $K^\uparrow$

Commentaires et r´ef´erences

La th´eorie de la supervision des automates est due `a l'´ecole de Ramadge et Wonham. Cet expos´e est largement inspir´e, que le lecteur pourra utilement lire. Signalons quelques probl`emes et quelques directions d'approfondissement.

Dans le cas o`u les hypoth`eses du théor`eme 5.2.3.4 ne sont pas vérifiées, on peut se demander si la suite K_j converge encore vers $K^\uparrow$. La convergence n'a pas lieu en général pour des langages non réguliers [69]. Par contre, les conditions $\mathcal{L}(\mathcal{A}) = \overline{\mathcal{L}}(\mathcal{A})$ et $K \subset \overline{\mathcal{L}}(\mathcal{A})$ du Théor`eme 5.2.3.4 qui sont commodes pour définir la suite d'automates reconnaissant les K_j peuvent être relaxées comme suit. Lorsque M est un langage régulier, on note $\|M\|$ le nombre minimal d'états d'un automate reconnaissant M.

THÉORÈME 5.3.2.1. *(Ramadge et Wonham [69]) Supposons $\mathcal{A}$ fini et K régulier. La suite $\{K_j\}$ définie en (5.14) converge en un nombre fini d'itérations vers $K^\uparrow$. En outre*

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

On s'est limité dans ce chapitre au minimum concernant la supervision d'automates. Il y a une méthode de nature différente qui permet de calculer le langage controlable maximal : au lieu de calculer une suite d'automates, on soustrait `a $K$ le langage des mots indésirables que l'on caractérise comme une sorte de résiduel. Nous avons privilégié ici par simplicité la premi`ere approche, historiquement la plus ancienne et tr`es naturelle, mais il faut bien voir que la complexité de l'approche via la résiduation est meilleure. En effet, la premi`ere approche émonde de mani`ere progressive un automate raffinant $\mathcal{A}$ et reconnaissant $K$ (soit au plus $\|\overline{\mathcal{L}}(\mathcal{A})\|\|K\|$ états), donc le nombre d'itérations est borné par le nombre d'arcs `a émonder, soit au plus $(\|\overline{\mathcal{L}}(\mathcal{A})\|\|K\|)^2$. Le second algorithme exposé dans [16] valable pour une partie K préfixielle a une complexité de $O(\|\overline{\mathcal{L}}(\mathcal{A})\|\|K\|^2)$.

Dans le cadre, de cette introduction, nous avons, pour faire bref, laiss´e de cot´e les probl`emes de controleˆ en observation partielle. La th´eorie pr´esent´ee ici s’´etendre de mani`ere el´´ egante, pour peu que certaines conditions naturelles d’observabilit´e ou de clotureˆ du lan- gage l´egal K par rapport aux observations (normalit´e) soient v´erifi´ees. On a ignor´e aussi le principal probl`eme pratique qui est li´e `a la taille des automates : lorsque l’on couple un syst`eme `a n etat´ avec un syst`eme `a p etats,´ on obtient en g´en´eral un automate produit (`a np etats).´ Pour cette raison, les questions de contrˆole d´ecentralis´e sont essentielles. Il s’agit d’impl´ementer uniquement des controleursˆ locaux sans avoir `a manipuler des produits d’automates. Certaines conditions de d´ecentralisation de nature alg´ebrique ont et´´ e mises en evidence.´ Nous renvoyons le lecteur intéresse.