Vérification de propriétés sur les automates à file réactifs produits par compilation de programmes Électre

Mémoire de DEA

Grégoire Sutre

5 septembre 1997

Mon stage de DEA, dont le mémoire est ici présenté, s'est situé dans le cadre d'une collaboration entre le LSV (ENS Cachan) et l'IRCyN (Ecole Centrale de Nantes). J'ai été, durant ce stage qui s'est déroulé pour un tiers du temps à Nantes et pour deux tiers à Cachan, co-encadré par Alain Finkel (LSV) et Olivier Roux (IRCyN).

Table des matières

Table des figures

Introduction

Nous sommes concernés dans cette étude par la vérification de systèmes *réactifs asynchrones*. Parmi les systèmes réactifs, on trouve notamment les systèmes embarqués dans les transports (trains, avions...), pour lesquels, la vérification de propriétés de sûreté est souvent cruciale.

Nous étudions les systèmes réactifs écrits dans le *langage réactif déterministe asynchrone* Électre. Ce langage permet de décrire les *comportements* admissibles de *tâches* les unes par rapport aux autres, et sous l'arrivée d'*occurrences d'événement*. Le langage Électre est asynchrone car il est basé sur l'hypothèse de *non simultanéité* des occurrences d'événement.

Dans le cadre de la vérification, on associe à chaque programme un modèle, donné en général sous la forme d'un système de transitions. Il est ensuite possible de vérifier sur ce modèle des propriétés telles que l'absence de point de blocage, la satisfaction de conditions de sûreté ou de vivacité... Un modèle d'exécution pour les programmes Électre a déjà été défini dans [1]. Ce modèle, qu'on appelle dans la suite *automate à file réactif (AFR)*, consiste en un automate fini, auquel on adjoint une *file*, a priori **non bornée**, pour tenir compte de la mémorisation d'occurrences d'événement.

Nous nous sommes intéressés dans ce stage de DEA à la vérification de propriétés sur les AFRs. Dans [1], il est proposé une construction d'un AFR à file bornée, obtenue par produit synchronisé entre d'une part un automate fini, produit par le compilateur Électre [2, 3], correspondant à la partie contrôle de l'AFR, et d'autre part un automate fini simulant le comportement de la file bornée. Cette construction a été implémentée en 1996 en utilisant l'outil MEC. Ainsi, la vérification pour les AFRs à file bornée de propriétés *comportementales* exprimées en MEC est réalisable. D'autre part, des résultats de décidabilité concernant la vérification de propriétés *temporelles* sur les AFRs à file bornée ont été obtenus dans [4]. Ces résulats ont également été implémentés dans le logiciel Valet.

On constate cependant que la file d'un AFR n'est théoriquement pas bornée. D'autre part, l'automate simulant la file dans la construction de [1] a un nombre d'états exponentiel par rapport à la taille de la file considérée. On aimerait donc vérifier les AFRs directement, *en intégrant le fait que la file est potentiellement infinie*. Or la classe des *automates communicants*, qu'on appelle également *automates à file*, est étudiée depuis vingt ans. Bien que cette classe ait la puissance des machines de Turing [5, 6], des résultats récents de décidabilité et de représentation symbolique ont été établis pour des classes particulières d'automates

communicants [7, 8, 9]. Nous avons donc étudié dans notre stage de DEA, les possibilités d'analyse des AFRs, en nous inspirant des méthodes connues sur les automates communicants.

Nous indiquons à présent les résultats les plus significatifs auxquels nous avons abouti.

Représentation symbolique d'un AFR

Un AFR est construit à partir d'un automate fini, qu'on appelle *système de contrôle*, en le complétant par des transitions de *mémorisation* (ajout d'un événement à la file) et de *traitement d'occurrences mémorisées* (retrait d'un événement de la file). Nous n'entrons pas dans ici dans les détails de la construction, mais signalons que :

- la sémantique opérationnelle d'un AFR donne toujours la *priorité* au traitement d'une occurrence mémorisée. Ainsi, il y a dans un AFR R des séquences de "stabilisation" le long desquelles R ne peut que traiter des occurrences mémorisées.

- les transitions de mémorisation sont toujours des boucles : elles ont même état d'origine et d'arrivée. Ainsi, lorsqu'un AFR peut franchir une transition de mémorisation, il peut en franchir en fait une infinité.

Les deux remarques ci-dessus permettent d'établir des équivalences d'exécution : à toute exécution ρ, on associe une exécution ρ' ayant mêmes états d'origine et d'arrivée que ρ, et se terminant par une séquence de mémorisation suivie d'une séquence de stabilisation. On abouti ainsi au résultat suivant :

Pour tout AFR R, l'ensemble des états accessibles de R est reconnaissable et effectivement calculable.

Vérification de formules de logique temporelle

Décidabilité de la logique temporelle linéaire LTL

Nous nous sommes intéressés à la classe des formules de LTL sans l'opérateur next. Le résultat auquel nous avons abouti est le suivant :

Le model-checking des formules de LTL *sans next est décidable.*

On constate que cette classe de formules permet d'exprimer les mêmes propriétés globales sur un système que LTL. De ce fait, l'outil SPIN a choisi de ne pas intégrer l'opérateur next dans les formules de LTL qu'il permet de vérifier.

Afin de généraliser ce résultat et en nous inspirant de [10], nous avons considéré le produit synchronisé d'un AFR avec un automate de Büchi [11]. Cependant, le système de transitions obtenu, qui n'est pas un AFR, ne nous a permis de conclure à la décidabilité de LTL.

Décidabilité de la logique temporelle arborescente CTL

Nous nous sommes aussi intéressés à la classe des formules existentielles équitables de CTL sans next. Le résultat que nous avons obtenu est le suivant :

> *Le model-checking des formules existentielles équitables de* CTL *sans next est décidable.*

Ce résultat, qui peut s'exprimer de manière duale pour les formules universelles équitables de CTL sans next, permet notamment la vérification de propriétés de sûreté.

Nous avons également entrepris la généralisation de ce résultat, en cherchant une représentation symbolique de l'ensemble des états d'un AFR satisfaisant une formule de CTL. Cependant, nous avons rencontré les mêmes problèmes que dans [12] pour la gestion des cycles.

Organisation du document

Le mémoire est organisé comme suit : dans un premier chapitre, nous exposons brièvement les notions de base sur les mots et les systèmes de transitions. Ce chapitre permet de fixer les notations pour la suite du document.

Dans une première partie, nous présentons des rappels sur les automates communicants et sur Électre. Ainsi, le deuxième chapitre contient la description du modèle des automates communicants et de deux classes récentes d'automates communicants. Puis dans le troisième chapitre, nous décrivons succintement le langage Électre et nous rappelons en détails le modèle des AFRs.

Dans une deuxième partie, nous présentons les résultats auxquels nous avons abouti. Dans le quatrième chapitre, une normalisation du modèle des AFRs est effectuée, et nous donnons également quelques définitions complémentaires. Le cinquième chapitre contient des résultats sur les chemins des AFRs, dont les preuves sont assez techniques, et que nous utilisons dans la suite. Notre résultat principal est ensuite exposé dans le sixième chapitre, où on montre que l'ensemble des états accessibles d'un AFR est reconnaissable et effectivement calculable. Quelques détails sur l'implémentation que nous avons réalisée de ce résultat y sont également exposés. Enfin, dans le septième chapitre, nous expliquons les résultats de décidabilité que nous avons obtenu pour les fragments des logiques temporelles LTL et CTL considérés.

Chapitre 1

Préliminaires

Nous exposons dans ce chapitre les notations utilisées tout au long du document, ainsi que les notions de base sur les mots et les langages d'une part [13, 14], et sur les systèmes de transitions d'autre part [15]. Nous définissons finalement quelques problèmes d'intérêt qu'on retrouve habituellement dans la vérification de systèmes de transitions.

1.1 Mots, langages

Soit Σ un alphabet fini. Un mot sur Σ est une suite finie d'éléments de Σ. On note Σ^* l'ensemble des mots sur Σ, et λ le mot vide. Pour tout mot x, $|x|$ désigne la longueur de x et $x(i)$ la $i^{\text{ème}}$ lettre de x. Pour un mot $x \in \Sigma^*$ et une lettre $a \in \Sigma$, on note $|x|_a$ le nombre d'occurrences de a dans x. La concaténation des mots x et y s'écrit $x \cdot y$, ou plus simplement xy. Si x et y sont des mots, alors on dit que :

- x est un *facteur gauche* de y ssi il existe $z \in \Sigma^*$ tel que $y = xz$. On écrit alors $z = y - x$;
- x est un *facteur droit* de y ssi il existe $z \in \Sigma^*$ tel que $y = zx$.
- $x = x_1 \dots x_n$ est un *sous mot* de y ssi il existe $y_1, \dots, y_{n+1} \in \Sigma^*$ tels que $y = y_1 x_1 \dots y_n x_n y_{n+1}$;

L'ordre sous-mot est noté $|$, et l'ordre facteur gauche est noté $\leq$. Pour tout mot x, et pour tout $n \in \mathbb{N}$, on note $x_{|_n}$ le plus grand des facteurs gauche y de x tels que $|y| \leq n$. Si x est un mot, on note $Alph(x)$ l'ensemble $\{a \in \Sigma \ / \ |x|_a \neq 0\}$.

Un mot infini sur Σ est une suite infinie d'éléments de Σ. Si x est un mot infini et $a \in \Sigma$, on note également $|x|_a$ le nombre d'occurrences de a dans x. Signalons que si ce nombre est infini, on note alors $|x|_a = \infty$.

Rappelons qu'un langage $L \subseteq \Sigma^*$ est *rationnel* ssi il est reconnu par un automate fini d'alphabet d'entrée Σ. Un sous-ensemble E de $(\Sigma^*)^n$ est *reconnaissable* ssi E est une union finie de produits de langages rationnels sur Σ^*.

Soit $k = |\Sigma|$. On rappelle que $(\Sigma^*, \cdot)$ et $(\mathbb{N}^k, +)$ sont des monoïdes. Supposons que $\Sigma = \{a_1, \dots a_k\}$, et notons $(e_1, \dots e_k)$ la base canonique de $\mathbb{N}^k$.

Le morphisme de Parikh $\Psi : (\Sigma^*, \cdot) \longrightarrow (\mathbb{N}^k, +)$, est défini par : pour tout $i \in \{1, \ldots, k\}, \Psi(a_i) = e_i$. On remarque enfin que[1] :

- pour tout $x, y \in \Sigma^*$, $\Psi(x) \leq \Psi(y) \Longrightarrow Alph(x) \subseteq Alph(y)$;
- pour tout $x, y \in \Sigma^*$, $x|y \Longrightarrow \Psi(x) \leq \Psi(y)$.

1.2 Systèmes de transitions

1.2.1 Le modèle

Définition 1.1 *Un* système de transitions *est un quadruplet* (Q, q_0, A, δ) *où :*

- Q *est un ensemble d'états ;*
- $q_0 \in Q$ *est l'état initial ;*
- A *est un alphabet fini (ensemble des actions) ;*
- $\delta \subseteq Q \times A \times Q$ *est un ensemble de transitions.*

Si Q et δ sont finis, alors on parle de *système de transitions fini*. Un *chemin fini*, ou plus simplement un *chemin*, dans un système de transitions est une suite finie de transitions $(q_1, a_1, q_1'), (q_2, a_2, q_2'), \ldots, (q_m, a_m, q_m')$ telle que pour tout $i \in \{1, \ldots, m-1\}$, $q_i' = q_{i+1}$. On appelle *trace* d'un tel chemin le mot $a_1 \ldots a_m \in A^*$. Cette notion est étendue aux *chemins infinis*[2]. Une *exécution* d'un système de transitions est un chemin partant de l'état initial. Une transition (q, a, q') est aussi notée $q \xrightarrow{a} q'$, et cette notation est étendue aux chemins. Enfin, on écrit également :

- $q_1 \xrightarrow{a_1 \ldots a_m} q_{m+1}$ s'il existe $q_2, \ldots, q_m$ tels que $q_1 \xrightarrow{a_1} q_2 \ldots q_m \xrightarrow{a_m} q_{m+1}$;
- $q \xrightarrow{\sigma}$ s'il existe q' tel que $q \xrightarrow{\sigma} q'$.

Si $\rho_1 : q_1 \longrightarrow q_1'$ et $\rho_2 : q_2 \longrightarrow q_2'$ sont deux chemins tels que $q_1' = q_2$, alors on note $\rho_1 \cdot \rho_2$ le chemin $\rho_1 \cdot \rho_2 : q_1 \longrightarrow q_1' \longrightarrow q_2'$ construit en concaténant les chemins ρ_1 et ρ_2. Nous utiliserons également la notation suivante : si $\rho = q_0 \longrightarrow q_1 \longrightarrow q_2 \ldots q_m \longrightarrow q_{m+1} \ldots$ est un chemin infini, alors ρ^k désigne le chemin infini $q_k \longrightarrow q_{k+1} \longrightarrow q_{k+2} \ldots q_m \longrightarrow q_{m+1} \ldots$.

Définition 1.2 *Soit* $S = (Q, q_0, A, \delta)$ *un système de transitions.*

- *l'*ensemble des états accessibles de S, *noté* $\mathrm{RS}(S)$, *est l'ensemble* $\{s \in Q \ / \ \exists\, \sigma \in A^*, s_0 \xrightarrow{\sigma} s\}$;
- *l'*ensemble des séquences de S, *noté* $\mathcal{L}(S)$, *est l'ensemble* $\{\sigma \in A^* \ / \ q_0 \xrightarrow{\sigma}\}$;

1. $\leq$ désigne ici l'ordre usuel sur les vecteurs d'entiers.
2. la trace d'un chemin infini est donc un mot infini.

- *l'*arbre d'accessiblité de S, *noté* $\mathrm{RT}(S)$, *est l'arbre étiqueté dont la racine est étiquetée par* q_0 *et tel qu'un nœud étiqueté par* q *a un fils étiqueté par* q' *et l'arc* (q, q') *est étiqueté par* a *ssi* $q \xrightarrow{a} q'$ *;*

- *une transition* $t \in \delta$ *est* franchissable *ssi il existe* $s, s' \in \mathrm{RS}(S)$ *et* $a \in A$ *tels que* $t = (s, a, s')$.

Rappelons également qu'un *état de blocage* dans un système de transitions est un état d'où ne sort aucune transition.

1.2.2 Problèmes d'intérêt

Définition 1.3 *Si* S *est un système de transitions donné,*

1. *le* problème de l'accessiblité (RP) *consiste à déterminer si un état donné de* S *est accessible ;*

2. *le* problème du blocage (DP) *consiste à déterminer s'il existe un état de blocage accessible ;*

3. *le* problème de la terminaison (TP) *consiste à déterminer si l'arbre d'accessibilité* $\mathrm{RT}(S)$ *est fini ;*

4. *le* problème de la finitude de l'ensemble des états accessibles (FRSP) *consiste à déterminer si l'ensemble des états accessibles* $\mathrm{RS}(S)$ *est fini ;*

5. *le* problème de la représentation reconnaissable (RRP) *consiste, si* $\mathrm{RS}(S)$ *est reconnaissable, à déterminer un automate fini pour* $\mathrm{RS}(S)$.

Précisons que le problème RRP est un problème de *calcul*, contrairement aux autres problèmes qui sont des problèmes de *décision*. On fera cependant l'abus de langage suivant : si $\mathcal{C}$ est une classe de systèmes de transitions, telle que $\forall\, S \in \mathcal{C}, \mathrm{RS}(S)$ est reconnaissable, on dit que le RRP est *décidable* pour $\mathcal{C}$ ssi il existe une machine de Turing M qui, pour toute entrée $S \in \mathcal{C}$, calcule un automate fini pour $\mathrm{RS}(S)$.

Première partie

Automates communicants et Électre

Chapitre 2

Automates communicants

Nous décrivons dans ce chapitre le modèle des automates communicants. Comme ce modèle a la puissance des machines de Turing, des sous-classes particulières d'automates communicants ont été étudiées, et nous en présentons deux des plus récentes. Pour tout ce chapitre, Σ désigne un alphabet fini.

2.1 Le modèle

Nous présentons dans cette section le modèle des systèmes de machines communicantes à états finis et des automates communicants [5].

Définition 2.1 *Une* machine communicante à états finis (CFSM) *M est un système de transitions fini $(Q, q_0, \{+,-\} \times \Sigma \times \mathbb{N}, \delta)$.*

Intuitivement, $(+, a, i)$ dénote la réception de la lettre a du canal i, et $(-, a, i)$ dénote l'émission de a dans le canal i. Lorqu'il n'y a pas d'ambiguïté sur le canal, $(+, a, i)$ et $(-, a, i)$ sont plus simplement notés respectivement $+a$ et $-a$. On dit qu'un état q d'une CFSM est un *état de réception* ssi toutes les transitions sortantes de q sont étiquetées par une réception. L'exemple suivant, qu'on utilisera pour illustrer cette section, est tiré de [9].

Exemple 2.1 Pour l'ÉMETTEUR de la figure 2.1, on a :

- $Q = \{q_1, q_2, q_3\}$;
- $q_0 = 1$;
- $\Sigma = \{start, a, b, end, ack\}$;
- $\delta = \{(1,(-,start,1),2),(2,(-,a,1),2),(2,(-,b,1),2),(2,(-,end,1),3),$ $(3,(+,ack,1),1)\}$. □

Ces machines communicantes sont reliées entre elles par des canaux de communication pour former des systèmes :

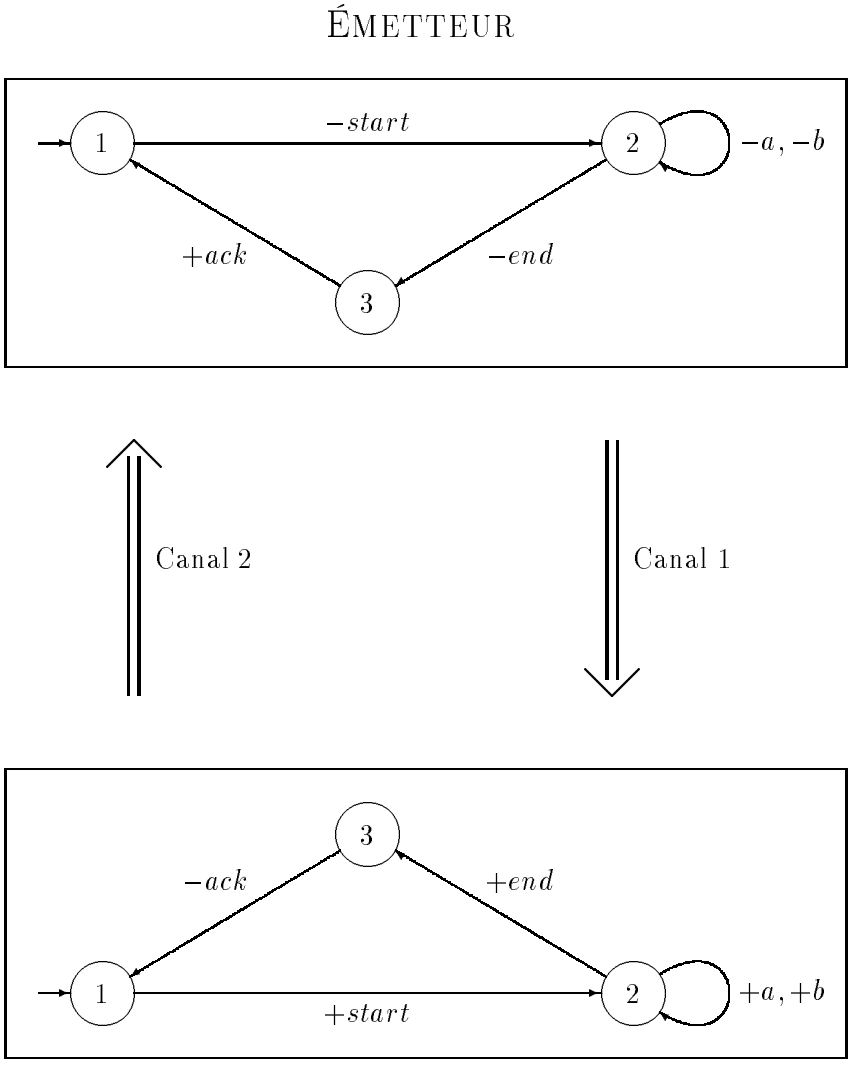


FIG. 2.1 – *Le système de CFSMs* S_1

Définition 2.2 *Un* système de n CFSMs à p canaux *est un* $(n+1)$*-uplet* $S = (M_1, \ldots, M_n, p)$ *avec* $M_i = (Q_i, q_{0i}, \{+,-\} \times \Sigma \times \{1, \ldots, p\}, \delta_i)$ *tel que :*

- *au plus une machine reçoit un message d'un canal donné :*

$$\forall\, j \in \{1, \ldots, p\}, \mathrm{Card}(\{i \in \{1, \ldots, n\} \;/\; \exists\, (q, (+, a', j), q') \in \delta_i\}) \leq 1$$

- *au plus une machine envoie un message dans un canal donné :*

$$\forall\, j \in \{1, \ldots, p\}, \mathrm{Card}(\{i \in \{1, \ldots, n\} \;/\; \exists\, (q, (-, a', j), q') \in \delta_i\}) \leq 1$$

Un système de 1 CFSM à p canaux est appelé un automate communicant à p canaux, et un automate communicant à 1 canal est appelé simplement un automate communicant. Clairement, tout système de n CFSMs à p canaux est bissimilaire à un automate communicant à p canaux (en construisant le produit cartésien des n CFSMs).

Exemple 2.2 Le système S_1 de la figure 2.1 forme un système de 2 CFSMs à 2 canaux. □

La sémantique opérationnelle d'un système de CFSMs, définie par le système de transitions associé, est donnée ci-après :

Définition 2.3 *Soit* $S = (M_1, \ldots, M_n, p)$ *un système de n CFSMs à p canaux avec* $M_i = (Q_i, q_{0i}, \{+,-\} \times \Sigma \times \{1, \ldots, p\}, \delta_i)$. $\widetilde{S}$, *le* système de transitions (Q, q_0, A, δ) associé à S *est défini par :*

- $Q = (Q_1 \times \ldots \times Q_n) \times (\Sigma^*)^p$ *est l'ensemble des états*[1] ;
- $q_0 = (q_{01}, \ldots, q_{0n}; \lambda, \ldots, \lambda)$ *est l'état initial;*
- $A = \{+,-\} \times \Sigma \times \{1, \ldots, p\}$ *est l'ensemble des actions;*
- $((q_1, \ldots, q_n; w_1, \ldots, w_p), t, (q'_1, \ldots, q'_n; w'_1, \ldots, w'_p)) \in \delta$ *ssi on se trouve dans l'un des deux cas suivants :*
  1. *il existe* $i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, p\}$ *et* $a \in \Sigma$ *tels que* $t = (+, a, j)$ *et* $(q_i, t, q'_i) \in \delta_i$, *avec :*
     - **(a)** $\forall\, k \neq i, q'_k = q_k$
     - **(b)** $a.x'_j = x_j$ *et* $\forall\, l \neq j, x'_l = x_l$
  2. *il existe* $i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, p\}$ *et* $a \in \Sigma$ *tels que* $t = (-, a, j)$ *et* $(q_i, t, q'_i) \in \delta_i$, *avec :*
     - **(a)** $\forall\, k \neq i, q'_k = q_k$
     - **(b)** $x'_j = x_j.a$ *et* $\forall\, l \neq j, x'_l = x_l$

1. un état $((q_1, \ldots, q_n), (w_1, \ldots, w_p))$ est plutôt noté $(q_1, \ldots, q_n; w_1, \ldots, w_p)$.

Dans la suite, si S est un système de CFSMs, on identifie S et $\widetilde{S}$. Expliquons cette définition : un état global du système consiste en l'état local de chaque machine d'une part, et en le contenu des canaux d'autre part. L'état initial du système est l'état où chaque machine est dans son état initial, et où les canaux sont vides. Enfin, le système franchit une transition t_S lorsqu'une machine M_i franchit une transition t alors que les autres machines restent dans le même état (condition (a)). Les canaux réagissent aux réceptions (condition 1.(b)) et aux émissions (condition 2.(b)) en se comportant comme des files FIFO. On étiquette alors la transition t_S, par l'étiquette de la transition t franchie par la machine M_i. On remarque qu'étant donné une transition t_S de S, on sait retrouver la machine M_i qui franchit une transition, et quelle transition M_i franchit [2].

Exemple 2.3 Considérons le système S_1 (figure 2.1). On a l'exécution suivante :

$$q_0 = (1,1;\lambda,\lambda) \xrightarrow{(-,start,1)} (2,1;start,\lambda) \xrightarrow{(-,a,1)} (2,1;(start)a,\lambda)$$

Pour ce système de CFSMs, on peut donner une représentation symbolique finie de l'ensemble des états accessibles [3] :

$$\begin{aligned} \mathrm{RS}(S_1) \quad = \quad & \{(1,1;\lambda,\lambda),(2,1;(start)\{a,b\}^*,(3,1;(start)\{a,b\}^*(end),\lambda), \\ & (2,2;\{a,b\}^*,\lambda),(3,2;\{a,b\}^*(end),\lambda),(3,3;\lambda,\lambda),(3,1;\lambda,ack)\} \end{aligned}$$

Comme on va le voir dans la section suivante, ce n'est pas possible dans le cas général, car on ne sait pas décider l'accessiblité pour les systèmes de CFSMs. □

Afin de projeter les séquences d'un système de CFSMs sur les réceptions et sur les émissions, nous aurons besoin dans la suite de la définition suivante :

Définition 2.4 *La projection sur les réceptions (resp. émissions) dans le canal $i \in \mathbb{N}$ est le morphisme p_i^+ (resp. p_i^-) de $(\{+,-\} \times \Sigma \times \mathbb{N})^*$ dans Σ^* défini par :*

$$p_i^+((op,a,j)) \quad (resp.\ p_i^-((op,a,j))) = \begin{cases} a \text{ si } op = + \quad (resp.\ op = -) \text{ et } j = i \\ \lambda \text{ sinon} \end{cases}$$

2.2 Vérification des automates communicants

Nous définissons dans cette section quelques problèmes d'intérêt qu'on retrouve habituellement dans la vérification de systèmes de CFSMs, qui est dans le cas général indécidable [5, 6].

Dans les systèmes de CFSMs, on considère une notion d'état de blocage particulière :

Définition 2.5 *Soit S un système de n CFSMs à p canaux, et $s = (q_1,\ldots,q_n; w_1,\ldots,w_p)$ un état de S.*

- *s est un* état de blocage *ssi $w_1 = \ldots = w_p = \lambda$ et pour tout $i \in \{1,\ldots,n\}$, q_i est un état de réception ;*

2. si $t_S = (q,(op,a,j),q')$, M_i est l'unique machine du système autorisée à faire l'opération op sur le canal j, et la transition franchie par M_i est $(q_i,(op,a,j),q_i')$.

3. S_1 est en fait un système half-duplex (cf. 2.3.2).

– *s est un* état de réception non spécifiée *ssi il existe $i \in \{1, \ldots, n\}$ tel que q_i est un état de réception et aucune transition sortante de q_i n'est étiquetée par $w_i(1)$.*

On constate que si le système se trouve dans un état de réception non spécifiée, une machine du système (la $i^{\text{ème}}$ dans la définition 2.5) ne pourra plus jamais franchir de transition. On introduit enfin une notion de finitude pour les canaux[4] :

Définition 2.6 *Soit S un système de n CFSMs à p canaux, et $i \in \{1, \ldots, p\}$. On dit que le canal i est* borné *ssi l'ensemble des contenus possibles du canal i :*

$$\{w_i \ / \ \exists \ (q_1, \ldots, q_n; w_1, \ldots, w_p) \in \mathrm{RS}(S)\}$$

est fini .

Outre les problèmes d'intérêt pour les systèmes de transitions (cf. définition 1.3), les problèmes suivants sont intéressants pour les systèmes de CFSMs :

Définition 2.7 (Problèmes d'intérêt pour les systèmes de CFSMs) *Si S est un système de CFSMs donné,*

1. *le* problème de la réception non spécifiée (URP) *consiste à déterminer s'il existe un état de réception non spécifiée accessible ;*
2. *le* problème de la borne faible (WBP) *consiste à déterminer si tous les canaux sont bornés ;*
3. *le* problème de la borne fort (SBP) *consiste à déterminer si un canal donné est borné ;*
4. *le* problème de la quasi-vivacité (QLP) *consiste à déterminer si une transition t donnée d'une machine du système est franchissable ;*
5. *le* problème de la vivacité (LP) *consiste à déterminer si une transition t donnée d'une machine du système est franchissable à partir de tout état $s \in \mathrm{RS}(S)$;*
6. *le* problème de l'état répété (RSP) *consiste à déterminer s'il existe une exécution visitant infiniment souvent un état local donné.*

On remarque que dans le cas des systèmes de CFSMs, le problème WBP est équivalent au problème FRSP. On constate également que si le RRP est décidable pour une classe d'automates communiquants $\mathcal{C}$, alors le RP, le DP, le FRSP, le URP, le QLP, et le SBP sont trivialement décidables. Enfin, le résultat suivant montre que la vérification générale est indécidable pour les systèmes de CFSMs.

Théorème 2.1 (Brand & Zafiropulo [5], Finkel & McKenzie [6]) *Les systèmes de CFSMs ont la puissance des machines de Turing.*

Comme les problèmes d'intérêt pour les automates communicants sont indécidables, on étudie des classes particulières d'automates communicants.

4. les canaux sont les éléments du modèle qui introduisent l'aspect infini dans les systèmes de CFSMs.

2.3 Classes d'automates communicants

Nous rappelons dans cette section des résultats récents de décidabilité concernant les automates communicants, en présentant deux classes particulières d'automates communicants. Ces classes se distinguent du modèle général en ajoutant des transitions de perte, de duplication ou d'insertion dans les canaux pour les automates communicants à canaux non-fiables, ou en imposant des conditions sur les contenus possibles des canaux pour les systèmes de CFSMs half-duplex. Enfin, la notion de système de CFSMs quasi-stable se base sur des équivalences d'exécutions.

2.3.1 Automates communicants à canaux non fiables

Nous présentons ici la classe des automates communicants à canaux non fiables, et leurs propriétés [7, 8].

On considère à présent des systèmes de CFSMs communiquant au moyen de canaux non fiables. Trois types d'erreurs dans les canaux peuvent survenir : perte, insertion ou duplication de messages. La modélisation de ces erreurs dans le cadre des systèmes de CFSMs est réalisée de la manière suivante :

Définition 2.8 *Soient $S = (M_1, \ldots, M_n, p)$ un système de n CFSMs à p canaux avec $M_i = (Q_i, q_{0i}, \{+,-\} \times \Sigma \times \{1, \ldots, p\}, \delta_i)$. On note $Q = (Q_1 \times \ldots \times Q_n) \times (\Sigma^*)^p$ l'ensemble des états de S. On définit les ensembles de transitions de* perte, *d'*insertion, *et de* duplication, *respectivement notés δ_L, δ_I, δ_D, par :*

- $\delta_L \subseteq Q \times \{L\} \times Q$, $\delta_I \subseteq Q \times \{I\} \times Q$, *et* $\delta_D \subseteq Q \times \{D\} \times Q$;
- $((q; w_1, \ldots, w_p), L, (q'; w'_1, \ldots, w'_p)) \in \delta_L$ *ssi* $q = q'$ *et il existe* $a \in \Sigma$, $i \in \{1, \ldots, p\}$ *et* $u_i, v_i \in \Sigma^*$ *tels que :*
  - **(a)** $\forall\, k \neq i, w'_k = w_k$
  - **(b)** $w_i = u_i a v_i$ *et* $w'_i = u_i v_i$
- $((q; w_1, \ldots, w_p), I, (q'; w'_1, \ldots, w'_p)) \in \delta_I$ *ssi* $q = q'$ *et il existe* $a \in \Sigma$, $i \in \{1, \ldots, p\}$ *et* $u_i, v_i \in \Sigma^*$ *tels que :*
  - **(a)** $\forall\, k \neq i, w'_k = w_k$
  - **(b)** $w_i = u_i v_i$ *et* $w'_i = u_i a v_i$
- $((q; w_1, \ldots, w_p), D, (q'; w'_1, \ldots, w'_p)) \in \delta_D$ *ssi* $q = q'$ *et il existe* $a \in \Sigma$, $i \in \{1, \ldots, p\}$ *et* $u_i, v_i \in \Sigma^*$ *tels que :*
  - **(a)** $\forall\, k \neq i, w'_k = w_k$
  - **(b)** $w_i = u_i a v_i$ *et* $w'_i = u_i a a v_i$

On peut maintenant définir les différentes classes d'automates communicants à canaux non fiables. Il existe six classes de systèmes de CFSMs à canaux non fiables, correspondant à des combinaisons d'erreurs de perte, d'insertion et de duplication. Un système de CFSMs à canaux non fiables est défini, en ce qui

concerne les éléments qui le composent, de la même façon qu'un système de CFSMs normal[5]. C'est donc par la sématique opérationnelle qu'on leur associe que les systèmes de CFSMs à canaux non fiables se distinguent du modèle classique.

Définition 2.9 *Soient $S = (M_1, \ldots, M_n, p)$ un système de n CFSMs à p canaux, et $\widetilde{S}$ le système de transitions normal*[6] *(Q, q_0, A, δ) associé à S.*

- *Le* système de transitions avec pertes associé à S, *noté $\widetilde{S}_L$, est défini par $\widetilde{S}_L = (Q, q_0, A, \delta \cup \delta_L)$;*
- *Le* système de transitions avec insertions associé à S, *noté $\widetilde{S}_I$, est défini par $\widetilde{S}_I = (Q, q_0, A, \delta \cup \delta_I)$;*
- *Le* système de transitions avec duplications associé à S, *noté $\widetilde{S}_D$, est défini par $\widetilde{S}_D = (Q, q_0, A, \delta \cup \delta_D)$;*
- *Le* système de transitions avec pertes et insertions associé à S, *noté $\widetilde{S}_{L\&I}$, est défini par $\widetilde{S}_{L\&I} = (Q, q_0, A, \delta \cup \delta_L \cup \delta_I)$;*
- *Le* système de transitions avec pertes et duplications associé à S, *noté $\widetilde{S}_{L\&D}$, est défini par $\widetilde{S}_{L\&D} = (Q, q_0, A, \delta \cup \delta_L \cup \delta_D)$;*
- *Le* système de transitions avec insertions et duplications associé à S, *noté $\widetilde{S}_{I\&D}$, est défini par $\widetilde{S}_{I\&D} = (Q, q_0, A, \delta \cup \delta_I \cup \delta_D)$;*

On parle alors de système de CFSMs normal, avec pertes, avec insertions, avec duplications, avec pertes et insertions, avec pertes et duplications, avec insertions et duplications respectivement pour indiquer que le système de transitions associé est normal, avec pertes, avec insertions, avec duplications, avec pertes et insertions, avec pertes et duplications, avec insertions et duplications respectivement.

Systèmes de CFSMs avec pertes

Dans [16], la notion de *protocole complètement spécifié* a été introduite. Un protocole complètement spécifié ne diffère d'un système de CFSMs avec pertes que parce qu'il n'autorise les pertes de messages qu'en début de canal. Bien que le modèle des protocoles complètement spécifiés soit plus restrictif que celui des systèmes de CFSMs avec pertes, il existe en fait une étroite relation entre les deux. En effet, si on considère un système de CSFMs avec pertes, la perte d'un message a lors d'une exécution σ n'est réellement prise en compte dans σ que lorsque a aurait dû se trouver en début de canal. Ainsi, quitte à :

- décaler les transitions de pertes de σ pour qu'elles se fassent le plus tard possible[7],

5. on se réfère ici à la définition 2.2.
6. on se réfère ici à la definition 2.3.
7. i.e. lorsque a se trouve en début de canal.

– supprimer les transitions de pertes de σ qui ne sont pas prises en compte dans σ,

on se ramène à une exécution $\widetilde{\sigma}$ où les pertes se font toujours en début de canal. Par conséquent, $\widetilde{\sigma}$ est une exécution d'un protocole complètement spécifié.

Nous présentons maintenant les résultats sur les systèmes de CFSMs avec pertes.

Théorème 2.2 (Abdulla & Jonsson [8]) *Les assertions suivantes sont vérifiées :*

1. *si S est un système de CFSMs avec pertes, alors* $\mathrm{RS}(S)$ *est reconnaissable ;*
2. *le TP est décidable pour les systèmes de CFSMs avec pertes ;*
3. *le RP et le DP sont décidables pour les systèmes de CFSMs avec pertes ;*
4. *le RSP et le model-checking de LTL sont indécidables pour les systèmes de CFSMs avec pertes.*

Rappelons brièvement les preuves des assertions 1 et 2. Soit S un système de CFSMs avec pertes. On introduit un ordre $\preceq$ sur $\mathrm{RS}(S)$ composé de l'égalité pour les états locaux d'une part, et de l'ordre sous-mot | sur les canaux d'autre part. Le système S muni de $\preceq$ est structuré, car $\preceq$ est un bel ordre monotone et décidable, et le TP est donc décidable [17]. On note que $\overline{\mathrm{RS}(S)}$ est clos supérieurement pour $\preceq$, donc reconnaissable. Comme la classe des ensembles reconnaissables est close par complémentaire [14], $\mathrm{RS}(S)$ est reconnaissable.

En ce qui concerne les assertions 3 et 4, on notera simplement que :

– le DP se réduit au RP. En effet, puisqu'on considère que les états de blocage ont des canaux vides, ces derniers sont en nombre fini.

– le RSP se réduit au model checking de LTL en considérant la formule $\Diamond\Box\neg p$, où p n'est vrai que dans l'état local instance du RSP.

Nous ne rappelons pas les preuves de décidabilité du RP et d'indécidabilité du RSP, qui sont plus techniques, et qu'on pourra trouver dans [7, 8].

Enfin, la preuve du théorème suivant est réalisée en réduisant le RSP au RRP pour les systèmes de CFSMs avec pertes.

Théorème 2.3 (Cécé, Finkel & Purushothaman Iyer [7]) *Le RRP est indécidable pour les systèmes de CFSMs avec pertes.*

Systèmes de CFSMs avec insertions

On constate que si S est un système de CFSMs avec insertions, alors $\mathrm{RS}(S)$ est clos supérieurement, et ainsi reconnaissable. On peut en fait construire une suite croissante stationnaire d'ensembles clos supérieurement et effectivement calculables, dont la limite est $\mathrm{RS}(S)$ [7]. Le RRP est donc décidable pour les

systèmes avec insertions, et par conséquent le RP est décidable. Signalons finalement que pour les systèmes avec insertions, le DP, le TP et le FRSP sont triviaux.

> *Le RRP et le RP sont décidables, et le DP, le TP et le FRSP sont triviaux pour les systèmes de CFSMs avec insertions* **[7]**.

Systèmes de CFSMs avec insertions, et pertes ou duplications

Si S est un système de CFSMs à p canaux avec pertes et insertions, alors pour tout état accessible $(q; w_1, \ldots w_p) \in RS(S)$, on a $(q; (\Sigma^*)^p) \subseteq \mathrm{RS}(S)$, car on peut perdre ce qu'on a dans le canal, et y insérer n'importe quel mot. Ainsi, il suffit de déterminer les états accessibles avec canaux vides pour en déduire trivialement une représentation reconnaissable de $\mathrm{RS}(S)$. D'autre part, pour les systèmes avec pertes et insertions, le DP, le TP et le FRSP sont triviaux.

> *Le RRP et le RP sont décidables, et le DP, le TP et le FRSP sont triviaux pour les systèmes de CFSMs avec pertes et insertions* **[7]**.

D'autre part, un système de CFSMs avec insertions et duplications est un système de CFSMs avec insertions, puisqu'une transition de duplication est une transition d'insertion. Il vient :

> *Le RRP et le RP sont décidables, et le DP, le TP et le FRSP sont triviaux pour les systèmes de CFSMs avec insertions et duplications* **[7]**.

Systèmes de CFSMs avec duplications

On considère maintenant des systèmes de CFSMs avec duplications. Cette classe de systèmes de transitions a la puissance des machines de Turing [7], car on peut en fait simuler un système de CFSMs normal par un système de CFSMs avec duplications, en codant les mots dans les canaux de manière à inhiber les duplications.

Le problème engendré par les erreurs de duplication est le suivant : étant donné un mot w dans un canal d'un système de CFSMs avec duplications S, si w s'écrit $uaav$, alors on ne sait pas si la présence des deux a est due au fonctionnement normal de S ou si elle est due à une transition de duplication. Mais on peut remédier à ce problème en envoyant une lettre $\# \notin \Sigma$ après chaque émission, car ainsi les $\#$ dans les canaux séparent des séquences de lettres identiques : si le contenu d'un canal est $a_1 \ldots a_1 \# \ldots \# a_2 \ldots a_2 \# \ldots \# \quad \ldots \quad a_n \ldots a_n \# \ldots \#$, alors :

- on sait que les séquences de a_j et de $\#$ sont dues à des transitions de duplication. Plus précisément, une séquence de a_j ou de $\#$ de longueur $k \geq 1$ a fait intervenir $k - 1$ transitions de duplication.

- le contenu de la file correspondant à une exécution normale est $a_1 \ldots a_n$.

En revanche, il faut également permettre la réception de # afin de ne pas bloquer sur un état de réception non spécifiée.

Plus formellement, étant donné un système $S = (M_1, \ldots, M_n, p)$ de n CFSMs à p canaux *normal*, avec $M_i = (Q_i, q_{0i}, \{+,-\} \times \Sigma \times \{1, \ldots, p\}, \delta_i)$, on construit le système $S^{\#} = (M_1^{\#}, \ldots, M_n^{\#}, p)$ de n CFSMs à p canaux *avec duplications* à partir de S de la manière suivante :

- $M_i^{\#} = (Q_i \cup \{r_t\}_{t \in \delta_i^-}, q_{0i}, \{+,-\} \times (\Sigma \cup \{\#\}) \times \{1, \ldots, p\}, \delta_i^{\#})$, où δ_i^- est l'ensemble des transitions d'émission de δ_i ;
- $\delta_i^{\#}$ est obtenu à partir de δ_i comme indiqué sur la figure 2.2.

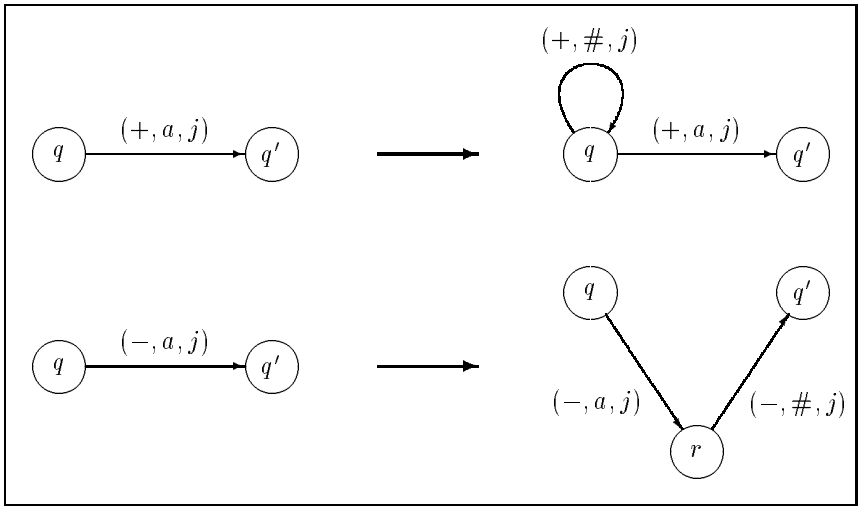


FIG. 2.2 – *Transitions de $M_i^{\#}$ à partir des transitions de M_i.*

Pour tout mot $w = a_1 \ldots a_n \in \Sigma^*$, on note $w^{\#}$ le mot $a_1 \# a_2 \# \ldots a_n \#$. Au vu des explications précédentes et d'après la définition de $S^{\#}$ à partir de S, on a pour tout système de CFSMs normal S :

$$(q; w) \in \mathrm{RS}(S) \iff (q; w^{\#}) \subseteq \mathrm{RS}(S^{\#})$$

Ainsi, si $\mathrm{RS}(S^{\#})$ est récursif, alors $\mathrm{RS}(S)$ est récursif. Comme $\mathrm{RS}(S)$ est récursivement énumérable, on obtient le théorème suivant :

Théorème 2.4 (Cécé, Finkel & Purushothaman Iyer [7]) *Les systèmes de CFSMs avec duplications ont la puissance des machines de Turing.*

Ainsi, aucune propriété non triviale (par exemple le RP, le DP, le TP, le RSP) n'est décidable pour les systèmes de CFSMs avec duplications.

Systèmes de CFSMs avec pertes et duplications

Afin d'obtenir les mêmes résultats de décidabilité pour les systèmes avec pertes et duplication que pour les systèmes avec pertes, on transforme un système de CFSMs avec pertes et duplication en un système de CFSMs avec pertes.

Soit un système $S = (M_1, \ldots, M_n, p)$ de n CFSMs à p canaux *avec pertes et duplications*, avec $M_i = (Q_i, q_{0i}, \{+,-\} \times \Sigma \times \{1, \ldots, p\}, \delta_i)$. On construit le système $S^D = (M_1^D, \ldots, M_n^D, p)$ de n CFSMs à p canaux *avec pertes* à partir de S de la manière suivante :

- $M_i^D = (Q_i \cup \{r_t\}_{t \in \delta_i^-}, q_{0i}, \{+,-\} \times (\Sigma \cup \{D\}) \times \{1, \ldots, p\}, \delta_i^D)$, où δ_i^- est l'ensemble des transitions d'émission de δ_i ;
- δ_i^D est obtenu à partir de δ_i en ne transformant que les transitions d'émission comme indiqué sur la figure 2.3.

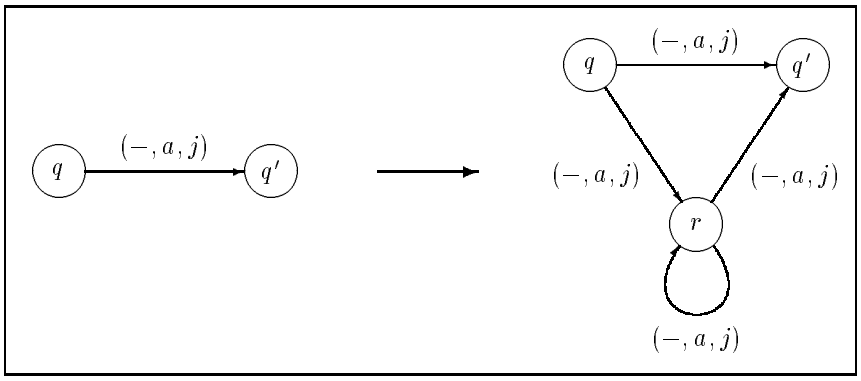


FIG. 2.3 – *Transitions de M_i^D à partir des transitions de M_i.*

D'après la définition de S^D à partir de S, il est clair que : $\mathrm{RS}(S) = \mathrm{RS}(S^D)$, puisque des passages par les états r dans S^D, en empruntant la boucle s'il le faut, peuvent remplacer des transitions de duplication dans S. Comme S^D est un système avec pertes, le RP, le DP et le TP sont décidables pour S^D.

Théorème 2.5 (Cécé, Finkel & Purushothaman Iyer [7]) *Le RP, le DP et le TP sont décidables pour les systèmes de CFSMs avec pertes et duplications.*

Enfin, en utilisant le même codage pour les systèmes de CFSMs avec pertes et duplications que pour les systèmes de CFSMs avec duplications, le RRP pour les systèmes de CFSMs avec pertes est réduit au RRP pour les systèmes de CFSMs avec pertes et duplications. D'autre part, en considérant la formule $\Diamond(\Box\neg p \ \vee \ \Box p)$, où p n'est vrai que dans l'état local instance du RSP, le RSP pour les systèmes de CFSMs avec pertes est réduit au model-checking de LTL pour les systèmes de CFSMs avec pertes et duplications.

Théorème 2.6 (Cécé, Finkel & Purushothaman Iyer [7]) *Le RRP et le model-checking de LTL sont indécidables pour les systèmes de CFSMs avec pertes et duplications.*

2.3.2 Systèmes de CFSMs half-duplex et quasi-stables

Nous présentons ici les systèmes de CFSMs half-duplex et quasi-stables, et leurs propriétés [9].

Systèmes de CFSMs half-duplex

Commençons par définir les systèmes de CFSMs half-duplex :

Définition 2.10 *Soit $S = (M_1, \ldots, M_n, p)$ un système de n CFSMs à p canaux avec $M_i = (Q_i, q_{0i}, \{+,-\} \times \Sigma \times \{1, \ldots, p\}, \delta_i)$. On dit que S est* half-duplex *ssi il vérifie les deux conditions suivantes :*

1. *pour tout $i \in \{1, \ldots, n\}$, on a :*

$$\{j \ / \ \exists\ (q, (+, a', j), q') \in \delta_i\} \bigcap \{j \ / \ \exists\ (q, (-, a', j), q') \in \delta_i\} = \emptyset$$

2. *pour tout $(q; w_1, \ldots, w_p) \in \mathrm{RS}(S)$, on a :*

$$\mathrm{Card}(\{j \in \{1, \ldots, p\} \ / \ w_j \neq \lambda\}) \leq 1$$

La condition 1 exprime que dans un système half-duplex, une même machine ne peut envoyer et recevoir de messages d'un même canal. Ainsi une machine ne peut se servir d'un canal comme d'une mémoire auxiliaire. Cependant cette condition n'est pas suffisante pour obtenir une classe de systèmes de transitions n'ayant pas le pouvoir des machines de Turing, car à l'aide d'une machine qui se contenterait de recopier les lettres d'un canal vers un autre canal, on pourrait simuler une machine de Turing [6]. C'est la raison pour laquelle la condition 2 est introduite. Cette dernière exprime que dans tout état accessible, un canal au plus est non vide. On remarque qu'un système de CFSMs comportant une machine qui recopie les lettres d'un canal vers un autre canal n'est pas half-duplex.

Définition 2.11 *Soit S un système de CFSMs half-duplex, et $s = (q; w)$ un état de S. On dit que s est stable ssi $w = (\lambda, \ldots, \lambda)$.*

Soit S est un système de CFSMs half-duplex. Si σ une exécution de S, alors σ est composée de sous-exécutions τ dans lesquelles un unique canal c_τ est non vide, reliées entre elles par des états stables[8]. Dans chacune de ces sous-exécutions τ, deux machines au plus sont susceptibles de franchir des transitions :

- la machine E_τ qui envoie dans le canal c_τ, s'il existe une machine qui envoie dans le canal c_τ, et alors E_τ ne peut franchir que des transitions d'émission dans τ ;
- la machine R_τ qui reçoit du canal c_τ, s'il existe une machine qui reçoit du canal c_τ, et alors R_τ ne peut franchir que des transitions de réception de τ.

On va donc pouvoir réordonner τ de manière à obtenir une exécution au cours de laquelle le canal c_τ évolue comme indiqué en figure 2.4. Si $s \xrightarrow{\tau} s'$, où s et s' sont des états stables, alors il existe deux exécutions τ_1, τ_2 et un état stable s_1 tels que $s \xrightarrow{\tau_1} s_1 \xrightarrow{\tau_2} s'$, et :

- c_τ est l'unique canal non vide de τ_1, et tous les états intermédiaires de τ_1 ont au plus une lettre dans le canal c_τ ;

8. notons que lorqu'on se trouve dans un état stable, comme tous les canaux sont vides, on est obligé de franchir une transition d'émission qui amène dans un état non stable.

– τ_2 n'est composée que de transitions d'émission dans le canal c_τ [9].

Par conséquent, on peut calculer une représentation reconnaissable de $\mathrm{RS}(S)$, en procédant comme suit :

– on calcule l'ensemble $\mathrm{RS}_{st}(S) = \{s \in \mathrm{RS}(S) \ / \ s \text{ est stable}\}$, ce qui est réalisable, car d'une part cet ensemble est fini, et d'autre part tout état stable accessible est accessible par une exécution dont la taille des canaux des états intermédiaires est bornée par 1.

– pour chaque état $(q; \lambda, \ldots, \lambda) \in \mathrm{RS}_{st}(S)$, on calcule l'ensemble reconnaissable $L(q)$ des contenus de canaux accessibles par un chemin ne réalisant que des émissions à partir de s.

Il vient finalement :

$$\mathrm{RS}(S) = \bigcup_{(q;\lambda,\ldots,\lambda)\in \mathrm{RS}_{st}(S)} \{(q; w) \ / \ w \in L(q)\}$$

Théorème 2.7 (Cécé & Finkel [9]) *Le RRP, le RP, le DP, le URP, le QLP, le FRSP et le SBP sont décidables pour les systèmes half-duplex.*

Théorème 2.8 (Cécé & Finkel [9]) *La propriété d'être half-duplex pour un système de CFSMs quelconque est décidable.*

Systèmes de CFSMs quasi-stables

Cependant, la propriété d'être half-duplex est assez contraignante. En particulier, deux systèmes de CFSMs half-duplex ne forment pas un système de CFSMs half-duplex, bien qu'on puisse, en les traitant séparément, obtenir une représentation reconnaissable de l'ensemble des états accessibles. La notion de système de CFSMs quasi-stable a donc été introduite dans [9] pour généraliser les systèmes de CFSMs half-duplex.

Donnons une définition informelle des systèmes quasi-stables (une définition précise se trouve dans [9]) :

– une exécution σ d'un système de CFSMs est *quasi-stable* ssi la taille de chaque canal au cours de σ évolue comme indiqué sur la figure 2.4.

– un système de CFSMs S est *quasi-stable* ssi :

1. S vérifie la condition 1 de la définition 2.10 ;
2. tout état $s \in \mathrm{RS}(S)$ est accessible par une exécution quasi-stable ;

En fait, les systèmes quasi-stables sont définis de manière à pouvoir généraliser le raisonnement réalisé pour les systèmes de CFSMs half-duplex, et obtenir ainsi les mêmes propriétés de décidabilité.

Théorème 2.9 (Cécé & Finkel [9]) *Si S est un système de CFSMs quasi-stable, alors* $\mathrm{RS}(S)$ *est reconnaissable et effectivement calculable. Ainsi, le RRP,*

9. on remarque que si τ aboutit à un état stable, alors τ_2 est vide.

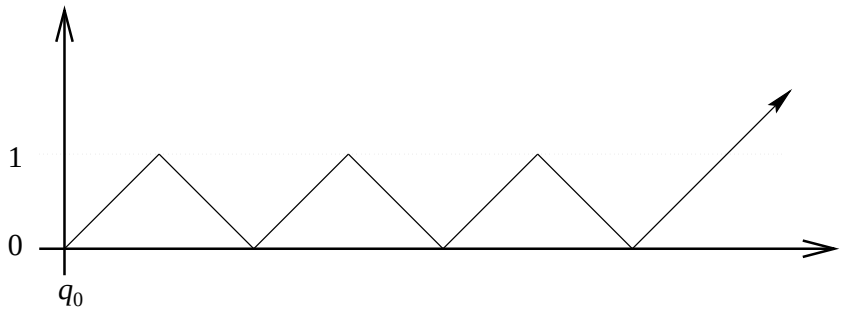


FIG. 2.4 – *Évolution de la taille d'un canal lors d'une exécution quasi-stable.*

le RP, le DP, le URP, le QLP, le FRSP, le SBP sont décidables pour les systèmes quasi-stables. La propriété d'être quasi-stable pour un système de CFSMs quelconque est également décidable.

Chapitre 3

Électre

Nous commençons dans ce chapitre par décrire le langage Électre, en attachant une importance particulière à la gestion de la mémorisation. Nous définissons ensuite le modèle formel sur lequel se base notre analyse de la deuxième partie de ce mémoire.

3.1 Le langage

Nous donnons dans cette section la syntaxe et la sémantique d'Électre [18, 3], en adoptant la même approche que dans [18].

3.1.1 Généralités

Le langage Électre est un langage réactif, c'est-à-dire permettant de décrire la partie contrôle d'un système : l'évolution de tâches (démarrage, interruption, reprise, terminaison) sous l'arrivée ou le déclenchement d'événements. En fait, en Électre, on ne décrit que cette partie contrôle du système, délaissant le codage séquentiel impératif des tâches au langage utilisé pour les implémenter. Le langage permet l'expression des comportements admissibles des tâches les unes par rapport aux autres (sequentiels, parallèles, exclusifs, répétitifs), ainsi que des tâches par rapport aux événements (préemption, activation).

Tout programme du langage Électre ne contient donc que deux types d'objets : le module et l'événement.

3.1.2 Les modules

Les tâches sont supposées sans point de synchronisation bloquant, et de durée d'exécution finie, mais non nulle. Elles correspondent aux objets du langage nommés *modules*. Si les modules ne peuvent pas être en attente d'événement, ils sont en revanche susceptibles d'émettre des événements. En particulier, la terminaison d'un module est un événement implicite[1] qui est pris en compte par le programme Électre.

1. pour un module A, cet événement est noté fin_A.

Au cours de l'exécution d'un module peuvent donc survenir des événements dont, par hypothèse d'asynchronisme, on observera et traitera les occurrences immédiatement.

Tout module, à un moment donné d'une exécution d'un programme Électre, est dans l'un des états suivants :

- **prêt** : le module a été lancé, et est actuellement en cours d'exécution ;
- **interrompu** : le module a été lancé, mais ensuite préempté avant sa terminaison ;
- **terminé** : le module n'a jamais été lancé ou, s'il l'a été, il s'est terminé.

Il existe dans le langage Électre un module prédéfini : **1** qui ne réalise aucune action et a une durée d'exécution infinie, i.e. ne termine jamais. Ce module joue un rôle important car il permet d'attendre une synchronisation tout en ne faisant rien.

3.1.3 Les événements

Les événements sont des signaux de type matériel ou logiciel, leur nature est indifférente. Leur rôle est de cadencer l'évolution de l'exécution des modules dans un programme Électre. Les événements ont une double interaction vis-à-vis des modules : il peuvent activer des modules et interrompre des modules.

De même que pour les modules, Électre associe à chaque événement un état. Trois états sont possibles pour les événements :

- **vivant** : l'événement a été traité et, ayant activé un module, il est vivant jusqu'à la terminaison du module activé ;
- **mémorisé** : l'événement est arrivé et, ne pouvant pas être pris en compte à ce moment, a été rangé dans la file. La possibilité pour un événement d'être mémorisé dépend de son éventuelle qualification (cf. 3.1.5) ;
- **consommé** : cet événement n'est pas survenu, ou n'a activé aucun module lors de sa prise en compte, ou le module activé s'est terminé.

En Électre, les occurrences d'événement sont non seulement asynchrones par rapport à l'exécution des modules, mais elles sont aussi, par hypothèse, asynchrones entre elles, et sont observées et traitées dès leur apparition. C'est pourquoi en Électre, on supposera que les observations des occurrences d'événement par le système se produisent toujours à des instants qui ne sont en aucun cas prédéfinis, qui peuvent être aussi proches que l'on veut, mais toujours différents : leur prise en compte sera donc toujours successive.

3.1.4 Les opérateurs

Les opérateurs sur les modules d'Électre permettent de décrire l'interaction entre modules et événements. Les structures de modules sont construites à partir des modules, ou d'autres structures de modules, et des opérateurs de composition de modules présentés ci-dessous. Elles sont généralement entourées par des crochets : [*structure de module*].

Séquentialité et parallélisme

Le comportement séquentiel de deux structures de modules est exprimé par leur concaténation. Le comportement parallèle de deux structures de modules est exprimé par l'opérateur '||'.

Exemple 3.1 Le programme **[A B].** exécute le module A, puis quand A s'est terminé exécute le module B, puis termine quand B s'est terminé. Le programme **[A || B].** exécute les modules A et B en parallèle, et termine lorsque les deux branches ont terminé. □

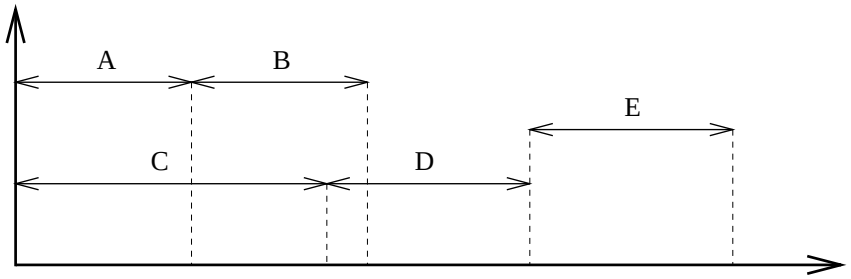


FIG. 3.1 – *Le programme* **[[[A B] || [C D]] E].**

Répétition

Le comportement répétitif d'une structure de modules est exprimé par l'opérateur '$\star$'.

Exemple 3.2 Le programme **[A B$\star$].** exécute le module A, puis quand A s'est terminé exécute infiniment le module B. Le programme **[A B]$\star$.** exécute infiniment la séquence **[A B]**. □

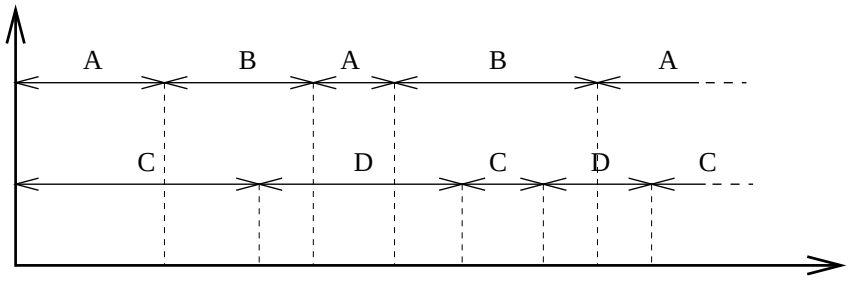


FIG. 3.2 – *Le programme* **[[A B]$\star$ || [C D]$\star$].**

Préemption et activation

L'opérateur de préemption signifie que, s'il y a une occurrence d'un événement et si le programme indique que cette occurrence doit être prise en compte,

alors l'exécution de la structure de module préemptée est interrompue. Il existe deux formes de préemption : la préemption nécessaire notée '/' et la préemption non nécessaire notée '↑'. Dans un programme Électre, un opérateur de préemption suit une structure de modules et précède un structure événementielle (cf. 3.1.4). Pour le moment, nous ne considérons que des structures événementielles simples, qui ne comportent qu'un seul événement activeur. Le mécanisme d'activation d'un module par l'arrivée d'un événement s'exprime par l'opérateur ':'.

Exemple 3.3 Un système qui réalise indéfiniment la tâche A tout en activant une tâche d'urgence $URGENT$ lors de l'apparition d'un événement *panne* indiquant une anomalie peut être décrit par le programme Électre :

$$\textbf{[A} \uparrow \textbf{\{panne: URGENT\}]}\star. \qquad \square$$

La notion de nécessité de la préemption évoquée plus haut indique si la progression dans le programme est soumise à l'occurrence de l'événement.

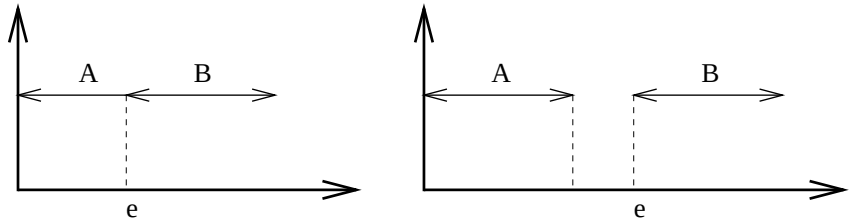


FIG. 3.3 – *Le programme* **[A / {e: B}].** *suivant les dates d'occurrences de e*

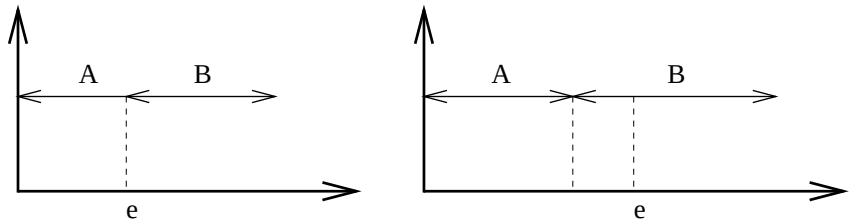


FIG. 3.4 – *Le programme* **[A ↑ {e: B}].** *suivant les dates d'occurrences de e*

On notera que sur l'exemple de droite de la figure 3.4, si l'événement e a la propriété de mémorisation, alors il est mémorisé lors de son occurrence.

Structure événementielle

Les structures événementielles qui se réduisaient précédemment à un simple événement peuvent être enrichies pour permettre d'exprimer l'activation en parallèle ou en disjonction exclusive des structures de modules suivant les occurrences d'événement. Ces structures sont délimitées par des accolades : {*structure événementielle*}. Les structures événementielles simples sont composées d'un

seul événement éventuellement activeur. Les opérateurs disponibles, permettant de composer des structures événementielles, sont :

- la *disjonction exlusive* notée '|' ;
- le *parallélisme* noté '||' ou '|||'[2]

Exemple 3.4 Les programmes **[A / {{e₁ : B} | {e₂}} : C]⋆.** et **[[A / {{e₁ : B} || {e₂ : C}}] / {e₃}]⋆.** utilisent des structures événementielles mettant en œuvre les opérateurs précédents. Des chronogrammes sont donnés en figures 3.5 et 3.6, où on suppose que les événements e_1 et e_2 ont la propriété de mémorisation. □

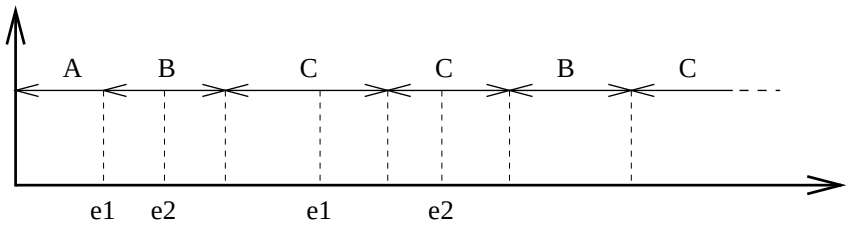


FIG. 3.5 – *Le programme* **[A / {{e₁ : B} | {e₂}} : C]⋆.**

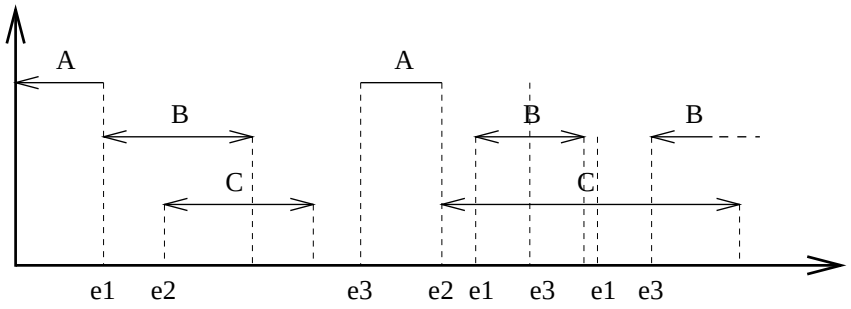


FIG. 3.6 – *Le programme* **[[A / {{e₁ : B} || {e₂ : C}}] / {e₃}]⋆.**

3.1.5 Propriétés des modules et des événements

Après avoir décrit les constructions essentielles du langage Électre, précisons à présent les propriétés qui peuvent être affectées aux objets du langage (modules et événements).

2. c'est la définition de la terminaison de la structure qui différencie ces deux opérateurs :
   - pour '||', la structure se termine lorsque les deux branches se sont effectivement exécutées et terminées ;
   - pour '|||', la structure se termine lorsque les branches effectivement démarrées se sont terminées.

Propriétés des modules

Pour les modules ces propriétés permettent de préciser les conditions de prise en compte d'une occurrence d'événement pendant l'exécution du module, ainsi que la reprise d'exécution d'un module aprés son interruption.

- la non-interruptibilité d'un module par une structure événementielle est notée par le qualificatif '!'. Si un module est ainsi qualifié, aucun événement ne pourra interrompre son exécution.
- par défaut, un module préempté dont l'exécution est ensuite reprise est relancé au point d'interruption. Afin de forcer systématiquement la reprise d'un module au début, on utilisera le qualificatif '>'.

Exemple 3.5 Les programmes **[A !B C] / {e: D}.** et **[>A / {{e: B}]⋆.** utilisent les qualificatifs de modules décrits précédemment. Des chronogrammes sont donnés en figures 3.7 et 3.8, où on suppose que l'événement e_1 a la propriété de mémorisation. □

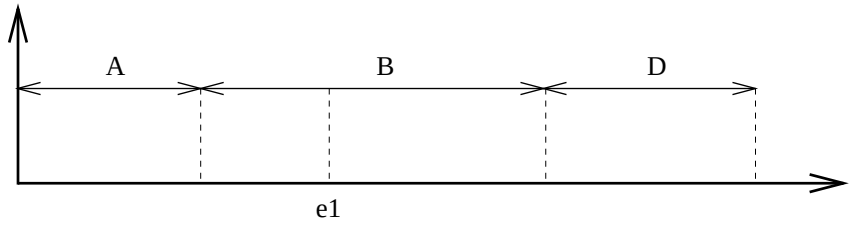


FIG. 3.7 – *Le programme* **[A !B C] / {e: D}.**

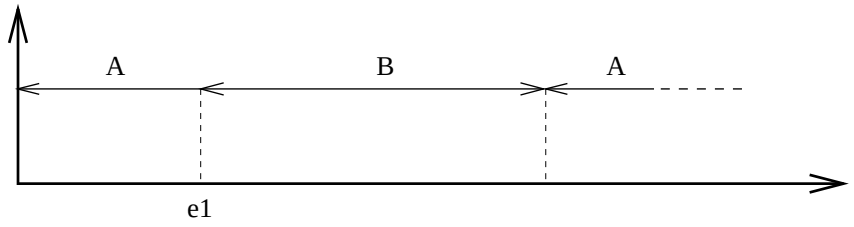


FIG. 3.8 – *Le programme* **[>A / {{e: B}]⋆.**

Propriétés des événements

Pour les événements, ces propriétés modifient la consommation et la mémorisation des occurrences. Par défaut, l'occurrence d'un événement n'est mémorisée qu'une fois (on dit que l'événement est *à mémorisation simple*), et elle est consommée à la terminaison de la structure de modules éventuellement activée. Les qualificatifs associés aux événements sont :

- la mémorisation multiple notée '#' : toutes les occurrences d'un tel événement sont mémorisées ;

– la consommation immédiate notée '$' : l'instant de consommation de l'occurrence de l'événement est identique à l'instant de prise en compte ;

– la fugacité notée '@' : pour ce type d'événement, il n'y a jamais de mémorisation de son occurrence et l'instant de consommation est confondu avec l'instant d'occurrence.

Précisons que les événements de fin de module (cf. 3.1.2) sont des événements fugaces. Enfin, on adoptera comme convention de ne pas donner plusieurs qualificatifs à un événement [3], même si le compilateur le permet, tout en le signalant. Ainsi dans un programme Électre, les ensembles d'événements fugaces, à mémorisation simple et à mémorisation multiple forment un partage de l'ensemble des événements.

Exemple 3.6 Les programmes **[A / {e: B}]⋆.** et **[A / {#e: B}]⋆.** utilisent les qualificatifs d'événement décrits précédemment. Des chronogrammes sont donnés en figures 3.9 et 3.10. □

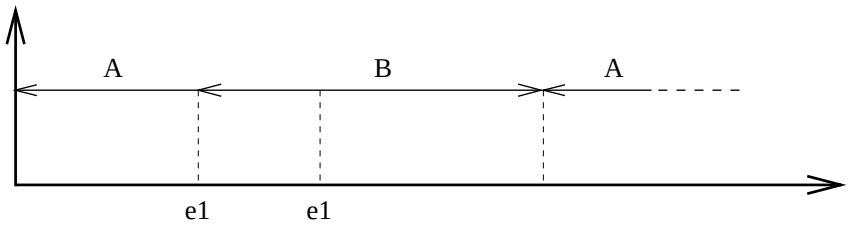


FIG. 3.9 – *Le programme* **[A / {e: B}]⋆.**

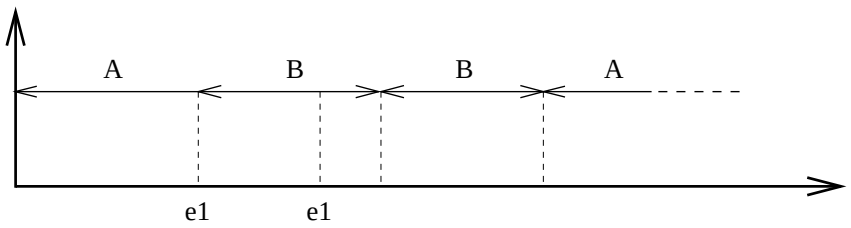


FIG. 3.10 – *Le programme* **[A / {#e: B}]⋆.**

3.1.6 Exemple : les lecteurs-écrivains

L'exemple décrit ici est tiré de [1], et nous l'utiliserons pour illustrer la section 3.3. Considérons donc le problème bien connu des lecteurs-écrivains : des lecteurs et des écrivains désirent accéder à un même livre. Les lecteurs peuvent lire le livre simultanément, mais lorqu'un écrivain modifie le livre, aucun autre processus (lecteur ou écrivain) ne peut accéder au livre. Nous réalisons la spécification

3. de toutes façons, quelle en serait la signification ?

en Électre de ce problème, pour deux lecteurs et deux écrivains, de la manière suivante :

- les modules $LIRE_1$ et $LIRE_2$ représentent les processus de lecture respectifs pour le lecteur 1 et le lecteur 2, et le module $ECRIRE$ représente le processus d'écriture utilisé par les deux écrivains ;
- les événements l_1, l_2, e_1 et e_2 représentent respectivement les demandes d'accès au livre pour les lecteurs et les écrivains.

et le programme correspondant est :

$$\mathbf{[1\ /\ \{\{@l_1 : LIRE_1 \parallel @l_2 : LIRE_2\} \mid e_1 : ECRIRE \mid e_2 : ECRIRE\}]}\star.$$

Bien sûr, on pourrait obtenir un système modélisant les lecteurs-écrivains avec d'autres propriétés en qualifiant les événements de manière différente.

3.2 Mémorisation et traitement des occurrences mémorisées

Notre travail porte plus spécifiquement sur le comportement des programmes Électre en la présence de mémorisation d'occurrences d'événements. En effet, comme on le verra plus en détail en 3.3, un programme Électre est compilé en un système de transitions fini pour la partie contrôle d'une part, auquel on adjoint une file FIFO pour la mémorisation des événements d'autre part. Ainsi l'aspect infini du système de transitions résultant provient de la file.

Comme on l'a vu précédemment, une occurrence d'événement mémorisable, i.e. non fugace, est rangée dans la file lorsqu'elle n'est pas traitable immédiatement. Précisons que pour l'ajout d'événements à mémorisation simple, la file se comporte comme suit :

- si l'événement est déjà présent dans la file, elle reste inchangée ;
- sinon, on l'ajoute à la file (dans l'ordre FIFO).

La sémantique intuitive d'Électre, en ce qui concerne le traitement des occurrences d'événement mémorisées est la suivante :

(a) une fois une occurrence d'événement mémorisée, elle doit être traitée *le plus tôt possible*

(b) les occurrences d'événement mémorisées doivent être traitées *dans le même ordre que celui de leurs occurrences.*

C'est la raison pour laquelle on utilise une file FIFO pour la mémorisation des occurrences d'événement. Cependant cette file FIFO n'a pas le comportement standard des canaux des automates communicants, et ce pour deux raisons :

- d'une part les événements à mémorisation simple ne peuvent avoir au plus qu'une occurrence mémorisée dans la file. Néanmoins, cette particularité

n'est pas très gênante, car de ce fait, les événements à mémorisation simple n'introduisent pas d'aspect infini dans notre système [4].

– d'autre part, lorsqu'on sort un événement de la file, on ne prend pas forcément le premier (au sens FIFO) de la file, mais on prend le toujours le premier qu'on est capable de traiter, parmi ceux présents dans la file. Cette sémantique particulière de la file est introduite afin de prendre en compte les aspects (a) et (b) de la sémantique de traitement des occurrences d'événement mémorisées.

En revanche, la mémorisation d'une occurrence d'événement se fait toujours par l'ajout standard (au sens FIFO) de l'événement à la file.

3.3 La compilation

Nous présentons dans cette section la compilation des programmes Électre, en décrivant tout d'abord comment le système de contrôle d'un programme Électre est obtenu [1, 2, 3, 18]. Ensuite ce système de contrôle est complété pour intégrer des transitions de mémorisation et de traitement d'occurrences d'événement mémorisées [1] : on obtient un système réactif asynchrone FIFO (AFR). Enfin, nous décrivons l'automate au format tef généré par la compilation, sur lequel se base l'implémentation de l'algorithme de calcul de la représentation symbolique de l'ensemble des états accessibles de l'AFR associé à un programme Électre (cf. 6.6) [3].

3.3.1 Le système de contrôle

La sémantique formelle d'Électre est décrite par des règles de réécriture conditionnelles exprimées sur la grammaire du langage [2, 3], lesquelles permettent également la compilation de tout programme Électre en un système de transitions fini : le *système de contrôle*, qui ne tient pas compte de la mémorisation des événements. Nous nous contentons ici d'expliquer de manière plus informelle comment le système de contrôle est obtenu [1].

Chaque état du système est modélisé par une paire $\langle B, C\rangle$ où :

– B est un *comportement* ;

– C est le *contexte* et représente l'ensemble des états des modules [5].

Un contexte est composé de trois sous-ensembles de l'ensemble des modules :

– $\mathcal{T}$: l'ensemble des modules qui sont *terminés* ;

– $\mathcal{P}$: l'ensemble des modules qui sont *prêts* ;

– $\mathcal{I}$: l'ensemble des modules qui sont *interrompus*.

4. lorsqu'on est en présence d'un programme sans événement à mémorisation multiple, le résultat de la compilation est un système de transitions fini, car la file est bornée [1].

5. rappelons que tout module est dans l'un des trois états suivants : prêt, interrompu ou terminé (cf. 3.1.2).

Pour chaque état $\langle B, C\rangle$ du système et pour chaque événement e traitable dans cet état[6], on calcule d'une part l'état $\langle B', C'\rangle$ du système après la prise en compte de l'occurrence de e, et d'autre part l'ensemble $\mathcal{A}$ des actions (lancer, interrompre, reprendre, ou tuer[7]) à opérer sur les modules pour les faire passer de l'état C à C'. On définit une *réaction* du système à l'événement e par la transition :

$$\langle B, C\rangle \xrightarrow{e/\mathcal{A}} \langle B', C'\rangle$$

De par l'hypothèse fondamentale d'asynchronisme en Électre, chaque transition du système de contrôle est étiquetée par un unique événement, car la perception de simultanéité est impossible, et par l'ensemble d'actions à réaliser sur les modules lors du franchissement de la transition.

Rappelons également qu'à chaque module est associé un événement de fin de module (cf. 3.1.2). Ainsi, dans chaque état $\langle B, C = [\mathcal{T}; \mathcal{P}; \mathcal{I}]\rangle$, tous les événements de fin des modules de $\mathcal{P}$ sont traitables et génèrent des transitions (avec ensemble d'actions vide).

Enfin, à l'état initial du système de contrôle, tous les modules sont dans l'état terminé, et les événements sont dans l'état consommé. Le calcul du système de contrôle est réalisé en partant de l'état initial, et en calculant de manière récursive l'ensemble des états directement accessibles pour chaque état nouvellement calculé. Il est prouvé dans [2] que ce processus termine ; ainsi le système de transitions obtenu et fini, et déterministe[8] par construction.

Exemple 3.7 Reprenons l'exemple des lecteurs-écrivains vu en 3.1.6, et décrivons le système de contrôle correspondant. Dans cet exemple, seul le module **1** peut être préempté, donc on aura toujours : $\mathcal{I} = \emptyset$. L'état initial du système est :

$$\langle B_0, C_0 = [\mathcal{T}_\prime = \{LIRE_1, LIRE_2, ECRIRE\}; \mathcal{P}_\prime = \emptyset]\rangle$$

Dans cet état, les quatre événements l_1, l_2, e_1 et e_2 peuvent être traités, et les quatres transitions (réactions) correspondantes sont :

$$\begin{aligned}
\langle B_0, C_0\rangle &\xrightarrow{l_1/\mathcal{A}_1} \langle B_1, C_1 = [\mathcal{T}_\prime = \{LIRE_2, ECRIRE\}; \mathcal{P}_\prime = \{LIRE_1\}]\rangle \\
\langle B_0, C_0\rangle &\xrightarrow{l_2/\mathcal{A}_2} \langle B_2, C_2 = [\mathcal{T}_\prime = \{LIRE_1, ECRIRE\}; \mathcal{P}_\prime = \{LIRE_2\}]\rangle \\
\langle B_0, C_0\rangle &\xrightarrow{e_1/\mathcal{A}_3} \langle B_3, C_3 = [\mathcal{T}_\prime = \{LIRE_1, LIRE_2\}; \mathcal{P}_\prime = \{ECRIRE\}]\rangle \\
\langle B_0, C_0\rangle &\xrightarrow{e_2/\mathcal{A}_4} \langle B_3, C_3 = [\mathcal{T}_\prime = \{LIRE_1, LIRE_2\}; \mathcal{P}_\prime = \{ECRIRE\}]\rangle
\end{aligned}$$

avec les ensembles d'actions : $\mathcal{A}_1 = \{\text{lancer } LIRE_1\}$, $\mathcal{A}_2 = \{\text{lancer } LIRE_2\}$, et $\mathcal{A}_3 = \mathcal{A}_4 = \{\text{lancer } ECRIRE\}$. Cette procédure peut être appliquée au nouvel état $\langle B_1, C_1\rangle$, et on obtient les deux transitions :

$$\begin{aligned}
\langle B_1, C_1\rangle &\xrightarrow{l_1/\emptyset} \langle B_1, C_1\rangle \\
\langle B_1, C_1\rangle &\xrightarrow{l_2/\mathcal{A}_5} \langle B_4, C_4 = [\mathcal{T}_\prime = \{LIRE_1, LIRE_2\}; \mathcal{P}_\prime = \{ECRIRE\}]\rangle
\end{aligned}$$

6. la mémorisation des événements sera prise en compte par l'AFR(cf. 3.3.2).
7. la reprise au début d'un module se fait en tuant le module, puis en le lançant.
8. il est donc utilisable en pratique.

avec $\mathcal{A}_5 = \{\text{lancer } LIRE_2\}$, car dans l'état $\langle B_1, C_1\rangle$, les événements e_1 et e_2 ne peuvent pas être traités. Enfin, comme dans l'état $\langle B_1, C_1\rangle$ le module $LIRE_1$ peut se terminer, on a également la transition :

$$\langle B_1, C_1\rangle \xrightarrow{fin_{LIRE_1}/\emptyset} \langle B_0, C_0\rangle$$

On procède de la même manière pour chaque nouvel état obtenu, et comme ce processus termine, on obtient le système de transitions fini de la figure 3.11. Enfin, on remarque sur cette figure l'importance de la composante comportement d'un état : bien que les états q_1 et q_6 aient le même contexte, l'effet d'une occurrence de l_2 est différent. □

Dans la suite nous utiliserons les notations suivantes : si E est l'ensemble des événements d'un programme Électre P :

- E_F est l'ensemble des événements fugaces de P (il comprend donc les événements de fin de module) ;
- E_M^S est l'ensemble des événements à mémorisation simple de P ;
- E_M^M est l'ensemble des événements à mémorisation multiple de P ;
- E_M, l'ensemble des événements mémorisables de P est défini par $E_M = E_M^S \cup E_M^M$.

Ainsi, d'après la convention sur les qualifications d'événements donnée en 3.1.5, on a : $E = E_F \cup E_M, E_F \cap E_M = \emptyset$ et $E_M^S \cap E_M^M = \emptyset$.

Résumons à présent les explications données ci-dessus sur la construction du système de contrôle associé à un programme Électre par une définition et un résultat sur lesquels se base toute la suite :

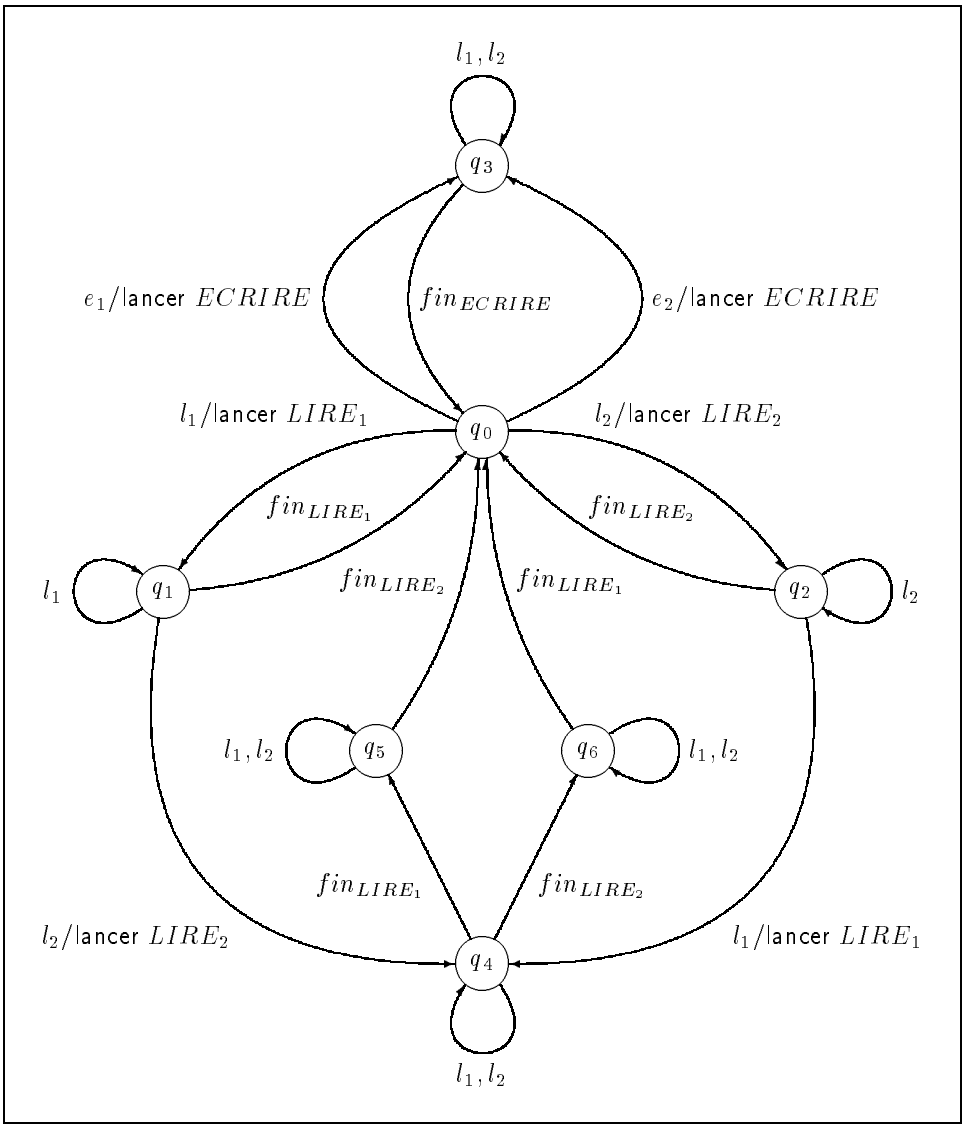
Définition 3.1 *Un* système de contrôle *est un système de transitions fini déterministe* $C = (Q, q_0, E \times 2^A, \delta)$, *vérifiant* $\mathrm{RS}(C) = Q$, *où :*

- *E est un ensemble fini d'événements dont on distingue :*
  - *les sous-ensembles E_F, E_M^S, et E_M^M formant un partage de E ;*
  - *le sous-ensemble E_M défini par $E_M = E_M^S \cup E_M^M$.*
- *A est un ensemble fini d'actions.*

Théorème 3.1 (Cassez & Roux [2]) *A tout programme Électre P est associé un système de contrôle $C_P = (Q, q_0, E \times 2^A, \delta)$, où :*

- *E est l'ensemble fini des événements du programme P ;*
- *A est l'ensemble fini d'actions :*

$$A = \bigcup_{M \text{ module de } P} \{\text{lancer } M, \text{interrompre } M, \text{reprendre } M, \text{tuer } M\}$$



	Contexte		
État	$\mathcal{T}$	$\mathcal{P}$	$\mathcal{I}$
q_0	$LIRE_1, LIRE_2, ECRIRE$	$\emptyset$	$\emptyset$
q_1	$LIRE_2, ECRIRE$	$LIRE_1$	$\emptyset$
q_2	$LIRE_1, ECRIRE$	$LIRE_2$	$\emptyset$
q_3	$LIRE_1, LIRE_2$	$ECRIRE$	$\emptyset$
q_4	$ECRIRE$	$LIRE_1, LIRE_2$	$\emptyset$
q_5	$LIRE_1, ECRIRE$	$LIRE_2$	$\emptyset$
q_6	$LIRE_2, ECRIRE$	$LIRE_1$	$\emptyset$

FIG. 3.11 – *Le système de contrôle pour les lecteurs-écrivains*

Le système de transitions C_P, est calculable pour tout programme Électre P, et respecte la sémantique du programme P, hormis la gestion des mémorisations.

Signalons finalement que, pour respecter les notations utilisées plus haut, une transition $(q, (e, \mathcal{A}), q')$ du système de contrôle est plus simplement notée $q \xrightarrow{e/\mathcal{A}} q'$.

3.3.2 L'automate à file réactif (AFR)

Le système de contrôle décrit en 3.3.1 ne gère ni la mémorisation des occurrences d'événement, ni le traitement des occurrences d'événement mémorisées. Cependant, comme nous l'avons déjà évoqué en 3.2, ce problème est résolu assez naturellement en adjoignant au système de contrôle une file FIFO, au *comportement non standard*. Cependant, le système de contrôle étant à présent formellement défini, nous pouvons préciser le comportement du système global, i.e. prenant en compte les mémorisations.

Dans la suite nous utiliserons les notations suivantes : soit $(Q, q_0, E \times 2^A, \delta)$ un système de contrôle associé à un programme Électre. Pour tout état $q \in Q$, on note :

- $Source(q)$ l'ensemble des événements $e \in E$ étiquetant une transition sortant de q ($Source(q)$ est l'ensemble des événements traitables dans l'état q) ;
- $Mémo(s)$ l'ensemble $E_M \setminus Source(q)$ ($Mémo(q)$ est l'ensemble des événements mémorisables dans l'état q).

Rappelons la sémantique d'Électre concernant la mémorisation des occurrences d'événement et le traitement des occurrences d'événement mémorisées :

- **mémorisation** : Si le système de contrôle est dans un état q au moment où une occurrence d'événement $e \in Mémo(q)$ survient, alors e est ajouté à la file ;
- **traitement** : Si le système de contrôle est dans un état q et qu'il existe une occurrence d'événement mémorisée $e \in Source(q)$, alors on sort de la file la première (au sens FIFO) occurrence mémorisée $e \in Source(q)$, et le système de contrôle la traite.

On complète dans un premier temps le système de contrôle associé à un programme Électre, en lui ajoutant des transitions de mémorisation et de traitement d'occurrence mémorisée :

Définition 3.2 *Soit $C_P = (Q, q_0, E \times 2^A, \delta)$ un système de contrôle associé à un programme Électre P. On appelle* système de contrôle complété associé à P, *noté $\overline{C_P}$, le système de transitions fini déterministe $(Q, q, (E \cup \{\oplus, \ominus\} \times E) \times 2^A, \bar{\delta})$ dont l'ensemble de transitions $\bar{\delta}$ est défini par :*

$$\begin{aligned} \bar{\delta} = \delta \quad & \cup \quad \{(q, (\oplus, e, \emptyset), q) \ / \ q \in Q \ et \ e \in Mémo(q)\} \\ & \cup \quad \{(q, (\ominus, e, a), q') \ / \ (q, (e, a), q') \in \delta \ et \ e \in E_M\} \end{aligned}$$

Comme pour le système de contrôle, on simplifie les notations des transitions du système de contrôle complété de la manière suivante :

- une transition $(q, (e, \mathcal{A}), q')$, correspondant au traitement immédiat d'une occurrence d'événement, est encore notée $q \xrightarrow{e/\mathcal{A}} q'$;
- une transition $(q, (\oplus, e, \emptyset), q)$, correspondant a la mémorisation d'une occurrence d'événement, est encore notée $q \xrightarrow{\ominus e/\emptyset} q$;
- une transition $(q, (\ominus, e, \mathcal{A}), q')$, correspondant au traitement d'une occurrence d'événement mémorisée, est encore notée $q \xrightarrow{\ominus e/\mathcal{A}} q'$.

ATTENTION Il ne faut pas confondre ces notations avec celles utilisées pour les automates communicants, car un $\oplus$ (resp. un $\ominus$) d'un système de contrôle complété correspond à un $-$ (resp. un $+$) d'un automate communicant.

Exemple 3.8 Pour l'exemple des lecteurs-écrivains vu en 3.1.6, et dont le système de contrôle a été décrit dans l'exemple 3.7, le système de contrôle complété est donné en figure 3.12. □

Nous pouvons à présent décrire la sémantique opérationnelle d'un système de contrôle complété, en imposant des conditions sur le franchissement des transitions, de sorte qu'il prenne en compte d'une part la mémorisation des occurrences d'événements, et d'autre part le traitement au plus tôt et dans l'ordre des occurrences mémorisées.

Définition 3.3 *Soit $C_P = (Q, q_0, E \times 2^A, \delta)$ un système de contrôle associé à un programme* Électre *P, et $\overline{C_P}$ sa complétion. On appelle* automate à file réactif (AFR) associé à P, *noté R_P, le système de transitions déterministe :*

$$(Q \times {E_M}^*, (q_0, \lambda), (E \cup \{\oplus, \ominus\} \times E) \times 2^A, \delta_R)$$

dont l'ensemble de transitions δ_R est défini par : $((q, w), a, (q', w')) \in \delta_R$ ssi on se trouve dans l'un des trois cas suivants :

1. $a = (e, \mathcal{A})$, $(q, (e, \mathcal{A}), q') \in \bar{\delta}$ *et* $w = w' \in Mémo(q)^*$;
2. $a = (\oplus, e, \emptyset)$, $(q, (\oplus, e, \emptyset), q') \in \bar{\delta}$ *et :*
   - $w \in Mémo(q)^*$;
   - $w' = we$ *si* $e \in E_M^M \cup (E_M^S \setminus Alph(w))$, *et* $w' = w$ *sinon*[9].
3. $a = (\ominus, e, \mathcal{A})$, $(q, (\ominus, e, \mathcal{A}), q') \in \bar{\delta}$ *et* $\exists\ w_1, w_2 \in {E_M}^*$ *tels que :*
   - $w = w_1 e w_2$ *et* $w' = w_1 w_2$;

9. on remarque que pour tout $e_m \in E_M$, les deux assertions suivantes sont équivalentes :

(i) $e_m \in E_M^S \setminus Alph(w)$
(ii) $\forall\, e \in E_M^S, |we_m|_e \leq 1$

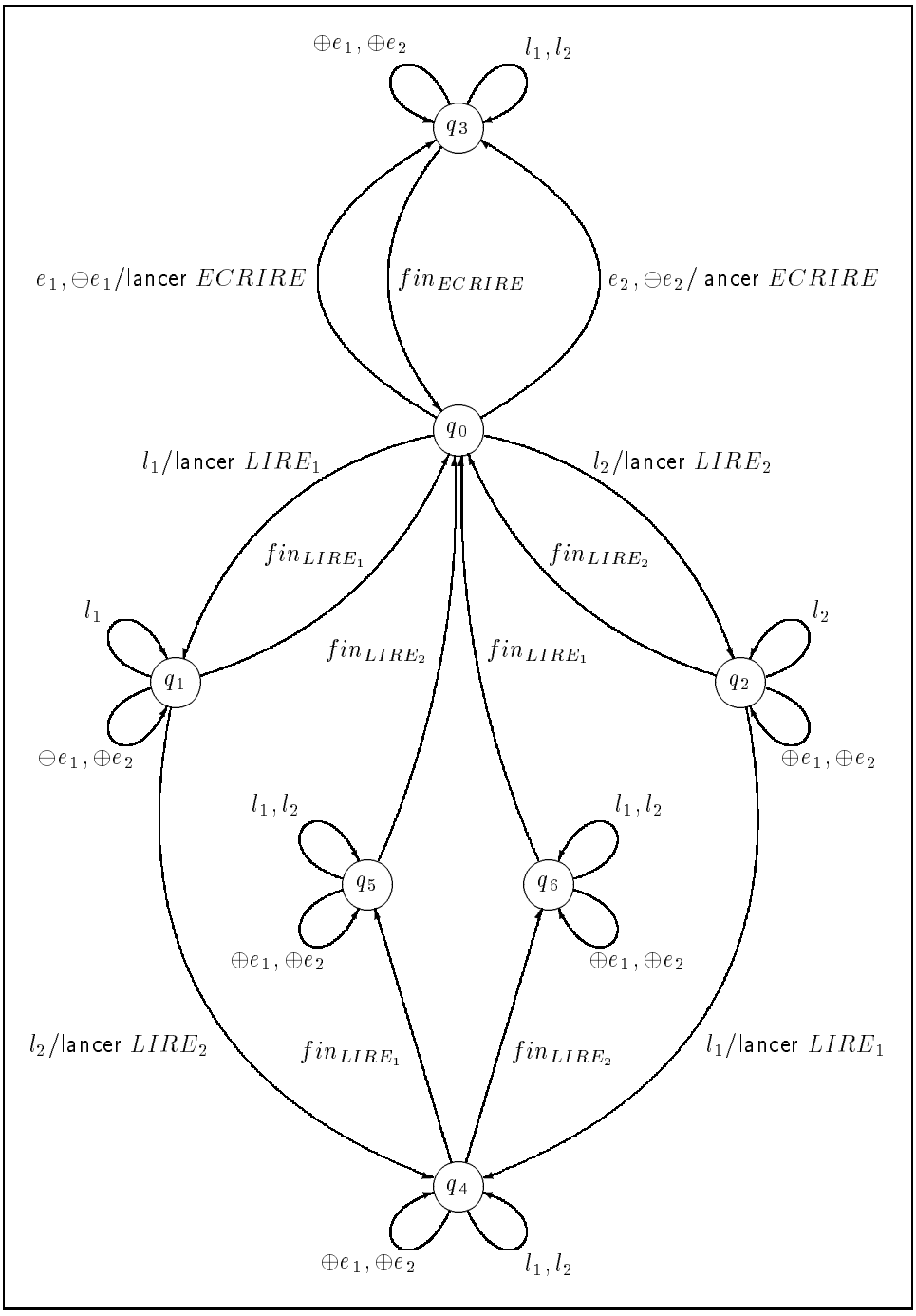


FIG. 3.12 – *Le système de contrôle complété pour les lecteurs-écrivains*

– $w_1 \in Mémo(q)^*$.

On appelle état de contrôle *de R tout état $q \in Q$.*

Commentons cette définition : un état global du système consiste en l'état de contrôle d'une part, et en le contenu de la file d'autre part. L'état initial du système est l'état de contrôle initial muni de la file vide. On remarque également que, d'après la définition 3.3, l'AFR associé à un programme Électre respecte la sémantique d'Électre pour la mémorisation des événements (condition 2), et pour le traitement des occurrences mémorisées (condition 3). Afin d'expliquer plus en détail les conditions de franchissement d'une transition du système de contrôle complété, on introduit la notion d'état stable :

Définition 3.4 *Un état (q, w) d'un AFR est* stable *ssi $w \in Mémo(q)^*$.*

Ainsi un état (q, w) d'un AFR est *instable* ssi il existe une occurrence événement e mémorisée (i.e. $e \in Alph(w)$) traitable dans l'état de contrôle q (i.e. $e \in Source(q)$). La définition 3.3 indique d'une part que lorqu'un AFR se trouve dans un état instable, alors il franchit une séquence de stabilisation pour arriver dans un état stable[10] et d'autre part qu'un AFR ne peut traiter immédiatement une occurrence d'événement, ou mémoriser une occurrence d'événement, que s'il se trouve dans un état stable.

On remarque également qu'étant donné un programme Électre P, et le système de contrôle associé C_P, d'une part le système de contrôle complété $\overline{C_P}$ associé à P est construit à partir de C_P, d'autre part l'AFR associé à P est construit à partir de $\overline{C_P}$. En fait la représentation qu'on donnera d'un AFR est celle du système de contrôle complété sur lequel il est construit, mais il faut bien distinguer ces deux notions : le système de contrôle complété donne l'ensemble des transitions qu'on est susceptible de franchir[11], alors que l'AFR donne l'ensemble des exécutions valides. Nous avons introduit ici le système de contrôle complété dans le seul but de clarifier la présentation.

Exemple 3.9 Toujours sur notre exemple des lecteurs-écrivains, considérons l'AFR donné par la figure 3.12. Une exécution valide est par exemple (on omet pour des raisons de place les ensembles d'actions) :

$$(q_0,\lambda) \xrightarrow{l_1} (q_1,\lambda) \xrightarrow{\oplus e_2} (q_1,e_2) \xrightarrow{l_2} (q_4,e_1) \xrightarrow{fin_{LIRE_1}} (q_6,e_2) \xrightarrow{\oplus e_1} (q_6,e_2e_1) \downarrow l_1$$

$$\ldots(q_3,e_1) \xleftarrow{l_2} (q_3,e_1) \xleftarrow{\ominus e_2} (q_0,e_2e_1) \xleftarrow{fin_{LIRE_2}} (q_6,e_2e_1)$$

Alors que l'exécution suivante du système de contrôle complété n'est pas valide dans l'AFR:

$$(q_0,\lambda) \xrightarrow{l_1} (q_1,\lambda) \xrightarrow{\oplus e_2} (q_1,e_2) \xrightarrow{fin_{LIRE_1}} (q_0,e_2) \xrightarrow{l_1} (q_2,e_2)\ldots$$

10. on remarquera que cette séquence est forcément finie puisque chaque transition de stabilisation enlève une occurrence d'événement à la file.

11. on verra en 6.6 qu'il peut exister des transitions dans le système de contrôle complété qui ne sont jamais franchies.

même si $q_0 \xrightarrow{l_1} q_1 \xrightarrow{\oplus e_2} q_1 \xrightarrow{fin_{LIRE_1}} q_0 \xrightarrow{l_1} q_2 \ldots$ est une exécution valide du système de contrôle complété. □

Signalons finalement que l'AFR associé à un programme Électre peut être construit comme le produit synchronisé du système de contrôle et d'un système de transitions infini $\mathcal{L}$ modélisant la file (chaque état de $\mathcal{L}$ correspond à un contenu de file) [1].

3.3.3 L'automate au format tef

La version 4 du compilateur Électre produit, à partir d'un fichier .ele contenant le source du programme Électre, un fichier .tef contenant l'AFR associé au format tef [3]. Ce format permet d'une part une optimisation de la place mémoire de stockage du fichier .tef, et d'autre part se rapproche de la sémantique intuitive d'Électre, et des besoins de l'exécutif Électre lors de la mise en pratique. Néanmoins, ceci présente un inconvénient en ce qui nous concerne : l'automate au format tef a une sémantique moins précise que celle de l'AFR. Aussi, dans notre implémentation de l'algorithme de calcul de la représentation des états accessibles d'un AFR (cf. 6.6), qui prend un automate au format tef représentant l'AFR en entrée, nous considérons que l'automate au format tef vérifie les conditions exposées ci-après.

Afin d'expliciter quelque peu ce format, nous présentons en figure 3.13 le début de l'automate au format tef pour notre exemple des lecteurs-écrivains (cf. 3.1.6), c'est-à-dire toutes les transitions partant de l'état initial et de l'état 4.

```
St0-&->St&0:A=[] D=[] F=[] ""
St&0-e1->St1:A=[ECRIRE] D=[] F=[] ""
St0-&e1->St1:A=[ECRIRE] D=[] F=[] "-e1"
St&0-e2->St2:A=[ECRIRE] D=[] F=[] ""
St0-&e2->St2:A=[ECRIRE] D=[] F=[] "-e2"
St&0-l1->St3:A=[LIRE1] D=[] F=[] ""
St&0-l2->St4:A=[LIRE2] D=[] F=[] ""

St4-&->St&4:A=[LIRE2] D=[] F=[] ""
St&4-e1->St&4:A=[LIRE2] D=[] F=[] "*e1"
St&4-e2->St&4:A=[LIRE2] D=[] F=[] "*e2"
St&4-endLIRE2->St0:A=[] D=[] F=[LIRE2] ""
St&4-l1->St5:A=[LIRE2 LIRE1] D=[] F=[] ""
```

FIG. 3.13 – *Début de l'automate au format* tef *pour les lecteurs-écrivains*

On différencie dans l'automate au format tef les états stables et les états instables. L'état stable correspondant à q_j est noté `etat&`j tandis que l'état instable lui correspondant est noté simplement `etat`j. De même, tout comme dans l'AFR, on distingue les transitions de traitement immédiat d'un événement e des

transitions de traitement d'une occurrence mémorisée de e. Une transition $\xrightarrow{e/\mathcal{A}}$ de l'AFR est notée `-`e`->` et une transition $\xrightarrow{\ominus e/\mathcal{A}}$ de l'AFR est notée `-&`e`->`. On remarque également qu'une transition de mémorisation $\xrightarrow{\oplus e/\emptyset}$ de l'AFR devient, dans l'automate au format **tef**, une transition de traitement immédiat `-`e`->` associée à une action de la file : `"+`e`"` si e est à mémorisation simple et `"*`e`"` si e est à mémorisation multiple. Enfin, on rajoute des transitions pour passer d'un état instable vers l'état stable correspondant : `-&->`.

La sémantique opérationnelle d'un automate au format **tef** est décrite par le franchissement de ses transitions :

- une transition de traitement est franchie dans les mêmes conditions que la transition correspondante de l'AFR. On remarque que :
  - une transition de traitement immédiat va d'un état stable vers un état instable ;
  - une transition de traitement d'occurrence mémorisée va d'un état instable vers un état instable ;
  - une transition de traitement immédiat correspondant à une mémorisation de l'AFR va d'un état stable vers le même état stable ;
- une transition `-&->` est franchie si l'état instable d'origine ne permet pas le traitement d'occurrence mémorisée.

Les notions de *stabilité* de l'AFR (cf. 3.3.2) et de l'automate au format **tef** coïncident donc.

Chaque ligne du fichier .**tef** décrivant l'automate au format **tef** est composée des trois informations suivantes :

- une transition de l'automate, par exemple : `St0-&e1->St1` ;
- les trois ensembles `A`, `D` et `F` dénotant respectivement [12]:
  - l'ensemble des modules actifs après la transition, i.e. l'ensemble $\mathcal{P}$ de l'état cible ;
  - l'ensemble des modules devant être relancés au début, ce qui permet de tenir compte du qualificatif '$>$' (`D` $\subseteq$ `A`) ;
  - l'ensemble des modules terminant lors de la transition. Cette information est en fait redondante car l'étiquette de la transition permet de déduire s'il y a et auquel cas lequel des modules termine (`F` est soit vide, soit un singleton).
- l'opération à effectuer sur la file :
  - `"-e"` pour le retrait de la première occurrence de l'événement e de la file ;

12. cette information remplace la donnée de $\mathcal{A}$ sur les transitions de l'AFR : en pratique, lors de l'exécution de l'automate **tef**, lorsqu'un événement extérieur arrive, on interrompt tous les modules, on déroule la séquence de transitions de l'automate **tef** jusqu'à l'obtention d'un état stable, puis on lance les modules de `A` en relançant ceux de `D` au début.

- `"+e"` pour l'ajout de l'événement à mémorisation simple e à la file[13] ;
- `"*e"` pour l'ajout de l'événement à mémorisation multiple e à la file.

La totalité de l'automate au format tef correspondant à notre exemple des lecteurs-écrivains est donnée en annexe B. On remarque qu'il comporte plus d'états que l'AFR donné en figure 3.12, mais les deux sont bissimilaires. Signalons enfin, comme nous l'avions évoqué au début de cette section, que le compilateur peut produire à partir d'un programme Électre correct un automate au format tef ne respectant pas les conditions exposées ci-dessus. Ainsi, plusieurs opérations sur la file peuvent être menées simultanément, comme par exemple `"-e2+e1"`, et il peut exister des transitions de mémorisation n'allant pas d'un état stable vers le même état stable. Cependant, ces cas que nous avons délibérément écartés sont le produit de l'optimisation du format, et ne remettent pas en cause les résultats théoriques qui vont suivre sur l'AFR.

13. rappelons qu'un événement à mémorisation simple est ajouté à la file s'il n'y est pas déjà, sinon il est perdu.

Deuxième partie

Analyse des automates à file réactifs

Chapitre 4

Normalisation des AFRs

Nous nous intéressons à présent à l'étude des propriétés des AFRs, et nous exposons dans cette partie les résultats auxquels nous avons abouti. Nous faisons dorénavant abstraction du langage Électre pour nous concentrer sur le modèle des AFRs. Nous considérons donc dans la suite des systèmes de contrôles vérifiant la définition 3.1. Rappelons qu'un AFR R est construit à partir d'un système de contrôle C, et qu'on le représente par le système de contrôle complété associé $\overline{C}$ (cf. 3.3.2). On note donc que les résultats exposés dans cette partie ne dépendent pas du langage Électre, mais s'appliquent bien aux AFRs.

4.1 Ajout d'une file initiale au modèle des AFRs

Le fait d'imposer une file initiale vide introduit une restriction dans le modèle des AFRs, contrairement aux automates communicants, où on peut toujours simuler un canal initial non vide par une succession de transitions d'émissions vers ce canal, partant d'un nouvel état initial, comme nous l'indiquons en figure 4.1. On note en effet que dans un AFR R associé à un système de contrôle

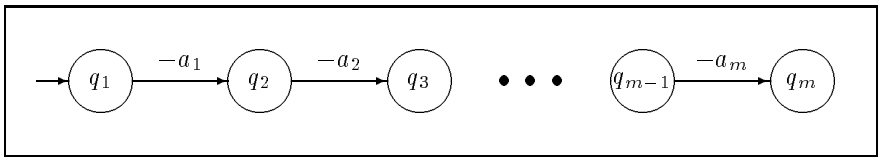


FIG. 4.1 – *Simulation d'un mot $a_1 \ldots a_n$ dans le canal à l'état initial.*

C, une transition de mémorisation de $\overline{C}$ va toujours d'un état de contrôle vers le même état de contrôle, si bien qu'on ne peut pas forcer R à franchir une succession donnée de transitions de mémorisation. C'est la raison pour laquelle nous faisons à présent la distinction entre un état initial avec file vide et un état initial avec file quelconque dans le modèle des AFRs, même si en pratique, le compilateur Électre génère un AFR dont l'état initial a une file vide. Aussi, si $C = (Q, q_0, E \times 2^A, \delta)$ est un système de contrôle, on utilise désormais les

dénominations suivantes :

- on dit que R est ***le*** *AFR à file initiale vide associé à* C ssi R vérifie la définition 3.3. En d'autres termes, ce qu'on appelait précédemment AFR s'appelle dorénavant AFR à file initiale vide.
- on dit que R est ***un*** *AFR quelconque associé à* C, ou plus simplement *un AFR associé à* C ssi $R = (Q_R, (q_{0_R}; w_R), A_R.\delta_R)$ avec :
  - $(Q_R, (q_{0_R}; \lambda), A_R.\delta_R)$ est l'AFR à file initiale vide associé à C ;
  - $w_R \in {E_M}^*$.

On notera qu'en revanche, le fait d'imposer pour état de contrôle initial d'un AFR l'état initial du système de contrôle associé n'est pas restrictif. En effet, si on souhaite avoir un état initial pour un AFR R différent de l'état initial du système de contrôle C sur lequel R est construit, il suffit de considérer un système de contrôle C' égal à C sauf pour son état initial qui correspond à l'état de contrôle initial de R, et duquel on ne garde que les états accessibles, pour rester dans les termes exacts de la définition 3.1.

D'autre part, on peut parler sans ambiguïté du système de contrôle associé à un AFR, ou du système de contrôle complété associé à un AFR.

Enfin, nous ne nous intéressons plus dorénavant aux actions réalisées sur les modules (lancer, interrompre, reprendre, ou tuer) lors du franchissement d'une transition. Aussi, pour toute la suite de ce chapitre, nous adoptons la **convention** suivante, qui ne modifie en rien les résultats et permet une présentation plus simple :

> **L'alphabet d'étiquetage d'un système de contrôle (complété ou non) ou d'un AFR ne comporte qu'une seule composante : celle correspondant aux événements.**

4.2 Définitions complémentaires

Les problèmes de vérification pour les AFRs auxquels nous nous intéressons, outre les problèmes d'intérêt pour les systèmes de transitions (cf. définition 1.3), sont les suivants :

Définition 4.1 (Problèmes d'intérêt pour les AFRs) *Si R est un AFR quelconque donné, et C le système de contrôle associé,*

1. *le* problème de la quasi-vivacité (QLP) *consiste à déterminer si une transition t donnée de $\overline{C}$ est franchissable ;*
2. *le* problème de l'état répété (RSP) *consiste à déterminer s'il existe une exécution visitant infiniment souvent un état de contrôle donné.*

Nous aurons besoin dans la suite des définitions suivantes :

Définition 4.2 *Soit $S = (Q, q_0, E \cup \{\oplus, \ominus\} \times E, \delta)$ un système de transitions. Soit σ une séquence de S et ρ un chemin de S étiqueté par σ_ρ. On dit que :*

- *σ est une* séquence de stabilisation *ssi $\sigma \in \{\ominus e \ / \ e \in E\}^*$;*

- σ *est une* séquence de mémorisation *ssi* $\sigma \in \{\oplus e \ / \ e \in E\}^*$;
- σ *est une* séquence sans mémorisation *ssi* $\sigma \in (E \cup \{\ominus e \ / \ e \in E\})^*$;
- ρ *est un* chemin de stabilisation, de mémorisation, sans mémorisation *respectivement ssi* σ_ρ *est une séquence de stabilisation, de mémorisation, sans mémorisation respectivement.*

Définition 4.3 *Soit* $S = (Q, q_0, E \cup \{\oplus, \ominus\} \times E, \delta)$ *un système de transitions. La* projection sur les mémorisations, *la* projection sur les traitements d'occurrences mémorisées, *la* projection sur le contrôle *et la* projection sur les traitements *sont les morphismes de* $(E \cup \{\oplus, \ominus\} \times E)^*$ *dans* E^* *respectivement notés* $p^\oplus$, $p^\ominus$, p^c *et* $p^{\ominus\&c}$, *définis par :*

- *pour tout* $e \in E$, $p^\ominus(\oplus e) = p^c(\oplus e) = p^{\ominus\&c}(\oplus e) = \lambda$ *et* $p^\oplus(\oplus e) = e$;
- *pour tout* $e \in E$, $p^\oplus(\ominus e) = p^c(\ominus e) = \lambda$ *et* $p^\ominus(\ominus e) = p^{\ominus\&c}(\ominus e) = e$;
- *pour tout* $e \in E$, $p^\oplus(e) = p^\ominus(e) = \lambda$ *et* $p^c(e) = p^{\ominus\&c}(e) = e$;

Les projections correspondantes $\pi^\oplus$, $\pi^\ominus$, π^c *et* $\pi^{\ominus\&c}$ *pour les chemins sont définies par : si* ρ *est un chemin de* S *de trace* σ, *alors* $\pi^\oplus(\rho) = p^\oplus(\sigma)$, $\pi^\ominus(\rho) = p^\ominus(\sigma)$, $\pi^c(\rho) = p^c(\sigma)$ *et* $\pi^{\ominus\&c}(\rho) = p^{\ominus\&c}(\sigma)$.

Définition 4.4 *Soit* R *un AFR quelconque, associé à un système de contrôle* $C = (Q, q_0, E, \delta)$. *On désigne par* δ_R *l'ensemble des transitions de* R. *L'*ensemble des transitions de pertes de R, *noté* $\mathrm{Pertes}(R)$ *est défini par :*

$$\mathrm{Pertes}(R) = \{((q;w),(\oplus,e),(q';w')) \in \delta_R \ / \ w' = w\}$$

On définit également les notions suivantes[1] *:*

- *on dit qu'un chemin* ρ *de* R *est un* chemin sans perte *ssi* ρ *s'écrit* $\rho = \tau_1 \ldots \tau_m$, *avec pour tout* $i \in \{1, \ldots, m\}$, $\tau_i \in \delta_R \setminus \mathrm{Pertes}(R)$;
- *on appelle projection sur les pertes, noté* π^p, *le morphisme de* ${\delta_R}^*$ *dans* E^* *défini par :*

$$\forall\, (s,(op,e),s') \in \delta_R, \pi^p((s,(op,e),s')) = \begin{cases} e \text{ si } (s,(op,e),s') \in \mathrm{Pertes}(R) \\ \lambda \text{ sinon} \end{cases}$$

Définition 4.5 *Soit* R *un AFR quelconque. L'ordre* $\preceq$ *sur* $\mathrm{RS}(R)$ *est défini par :* $(q;w) \preceq (q';w')$ *ssi* $q = q'$ *est* $w \leq w'$. *On définit également l'ordre* $|_\ominus$ *sur les chemins de* R *défini par* $\rho \,|_\ominus\, \rho'$ *ssi* $\pi^{\ominus\&c}(\rho) = \pi^{\ominus\&c}(\rho')$ *et* $\pi^\ominus(\rho) \,|\, \pi^\ominus(\rho')$.

Définition 4.6 *Soit* R *un AFR quelconque, d'état initial* $(q_0;w_0)$. *L'*arbre d'accessiblité sans mémorisation de R, *noté* $\mathrm{RT}^{\ominus\&c}(R)$, *est l'arbre étiqueté dont la racine* n_0 *est étiquetée par* $(q_0;w_0)$ *et tel qu'un nœud* n *étiqueté par* $(q;w)$ *a un fils étiqueté par* $(q';w')$ *et l'arc* $((q;w),(q';w'))$ *est étiqueté par* α *ssi :*

- $(q;w) \xrightarrow{\alpha} (q';w')$ *est une transition de* R *telle que* $\alpha \in E \cup \{\ominus e \ / \ e \in E\}$;

1. on considère ici, sans ambiguïté, les chemins de R comme des mots sur l'ensemble δ_R des transitions de R.

– *aucun nœud sur la branche de n_0 à n n'est étiqueté par $(q';w')$.*

Le graphe d'accessiblité sans mémorisation de R, *noté* $\mathrm{RG}^{\ominus\&c}(R)$, *est obtenu à partir de* $\mathrm{RT}^{\ominus\&c}(R)$ *en fusionnant les nœuds qui ont la même étiquette.*

Définition 4.7 *Soit R un AFR quelconque, d'état initial $(q_0;w_0)$. L'*ensemble des états accessibles sans mémorisation de R, *noté* $\mathrm{RS}^{\ominus\&c}(R)$, *est l'ensemble* $\{(q;w) \ / \ \exists\ \sigma$ *sans mémorisation*, $(q_0;w_0) \xrightarrow{\sigma} (q;w)\}$.

4.3 Remarques préliminaires

Nous commençons notre analyse des AFRs par une série de remarques qui vont permettre de bien fixer les notions et de mieux cerner les problèmes posés par ce modèle. Ces remarques, qui découlent directement des définitions, sont considérées acquises dans la suite.

Il est **IMPORTANT** de noter la constatation suivante :

Remarque 1 Puisqu'un système de contrôle C, un système de contrôle complété $\overline{C}$, ou un AFR R respectivement, est déterministe, un chemin fini ou infini dans C, $\overline{C}$, ou R respectivement, est uniquement déterminé par son état d'origine et sa trace.

Remarque 2 D'après la définition 3.3, on note que lors d'une transition de mémorisation d'un événement à mémorisation simple e d'un AFR, e est ajouté à la file ssi il n'y est pas déjà. Ainsi, pour tous les états accessibles d'un AFR à file initiale vide, la file contient au plus une occurrence de chaque événement à mémorisation simple.

Remarque 3 Soit R un AFR.

– Si ρ est un chemin sans mémorisation de R, alors ρ est un chemin sans perte de R ;

– les chemin sans mémorisation de R sont les chemins de $\mathrm{RG}^{\ominus\&c}(R)$.

Remarque 4 Soit R un AFR et ρ un chemin de R. Si $|\pi^{\ominus\&c}(\rho)| = 0$, alors ρ est un chemin de mémorisation.

Remarque 5 Si R est l'AFR à file initiale vide associé à un système de contrôle C, alors tous les nœuds de $\mathrm{RG}^{\ominus\&c}(R)$ sont étiquetés par un état de contrôle muni de la file vide. Ainsi, $\mathrm{RG}^{\ominus\&c}(R)$ considéré en tant que système de transitions est bissimilaire à C.

Remarque 6 Si R est un AFR, alors $\mathrm{RS}^{\ominus\&c}(R)$ est l'ensemble des étiquettes des nœuds de $\mathrm{RT}^{\ominus\&c}(R)$.

Remarque 7 Si ρ est un chemin d'un AFR R, tel que ρ n'est pas un chemin de stabilisation, alors ρ passe par un état stable.

Remarque 8 Soit R un AFR. Si $(q;w) \xrightarrow{\oplus e_m} (q';w')$ est un chemin de R, alors par définition :

- $q = q'$;
- si $\forall\, e \in E_M^S \cap \{e_m\}$, $|we_m|_e \leq 1$, alors $w' = we$.

En raisonnant par récurrence sur la taille des chemins, on obtient que pour tout chemin de mémorisation $(q;w) \xrightarrow{\oplus e_1 \ldots \oplus e_k} (q';w')$ de R, on a :

- $q = q'$;
- si $\forall\, e \in E_M^S \cap \{e_1, \ldots, e_k\}$, $|w \cdot (e_1 \ldots e_k)|_e \leq 1$, alors $w' = w \cdot (e_1 \ldots e_k)$.

Remarque 9 Soit R un AFR. Si $t = (q;w) \xrightarrow{\oplus e_m} (q';w')$ est une transition de R telle que $|w'|_{e_m} \geq 2$, alors :

- $w' = w$;
- t est une transition de perte.

En raisonnant par récurrence sur la taille des chemins, on obtient que pour tout chemin $\rho = (q;w) \longrightarrow (q';w')$ de R, et pour tout $e \in E_M^S$ tel que $|w'|_e \geq 2$:

- $|w|_e \geq |w'|_e$;
- toutes les transitions de ρ étiquetées par $\oplus e$ sont des transitions de perte, et donc $|\pi^{\oplus}(\rho)|_e = |\pi^{p}(\rho)|_e$.

Remarque 10 Soit $(q;w)$ est un état stable d'un AFR R. On a :

- pour tout $w_1|w$, (q, w_1) est stable ;
- si $e_1, \ldots, e_k \in \mathit{Mémo}(q)$: alors $(q;w) \xrightarrow{\oplus e_1 \ldots \oplus e_k}$ est un chemin dans R. Si de plus pour tout $e \in E_M^S \cap \{e_1, \ldots, e_k\}$, $|w \cdot (e_1 \ldots e_k)|_e \leq 1$, alors, d'après la remarque 8, $(q;w) \xrightarrow{\oplus e_1 \ldots \oplus e_k} (q; w \cdot (e_1 \ldots e_k))$.

Remarque 11 Soit $(q;w)$ est un état stable d'un AFR R, et $w' \in \mathit{Mémo}(q)^*$. D'après la remarque 10, on a :

$$\big\{(q;w) \in \mathrm{RS}(R) \text{ et } \forall\, e \in E_M^S \cap \mathit{Alph}(w'), |ww'|_e \leq 1\big\} \Longrightarrow (q;ww') \in \mathrm{RS}(R)$$

Remarque 12 Soit R un AFR associé à un système de contrôle $C = (Q, q_0, E, \delta)$.

- Si $(q;w) \xrightarrow{\ominus e_1 \ldots \ominus e_k} (q';w')$ est un chemin de stabilisation de R : alors d'une part $w'|w$, d'autre part $q \xrightarrow{e_1 \ldots e_k} q'$ est un chemin de C.
- Si $q \xrightarrow{e_1 \ldots e_k} q'$ est un chemin de C tel que $e_1 \ldots e_k \in {E_M}^*$: alors pour tout $w \in {E_M}^*$, $(q;(e_1 \ldots e_k) \cdot w) \xrightarrow{\ominus e_1 \ldots \ominus e_k} (q';w)$ est un chemin de C[2]. Par conséquent :

$$\forall\, w \in {E_M}^*, (q;(e_1 \ldots e_k) \cdot w) \in \mathrm{RS}(R) \Longrightarrow (q;w) \in \mathrm{RS}(R)$$

2. le lecteur peut se reporter à la proposition 5.4 qui généralise cette remarque.

Remarque 13 Si C est un système de contrôle, et R un AFR associé, alors à tout chemin $q_1 \xrightarrow{a_1} q_2 \ldots q_m \xrightarrow{a_m} q_{m+1}$ de C correspond le chemin dans R : $(q_1;\lambda) \xrightarrow{a_1} (q_2;\lambda) \ldots (q_m;\lambda) \xrightarrow{a_m} (q_{m+1};\lambda)$. En particulier, puisque tous les états du système de contrôle sont accessibles dans le système de contrôle (cf. définition 3.1), tous les états $(q;\lambda)$ d'un AFR à file initiale vide R, où q est un état de contrôle, sont accessibles dans R.

Remarque 14 Si $(q;w) \in \mathrm{RS}(R)$ est un état stable d'un AFR R, et s'il existe une transition de mémorisation d'un événement e dans l'état de contrôle q, i.e. s'il existe $e \in \mathit{Mémo}(q)$, alors on a le chemin suivant dans R : $(q;w) \xrightarrow{\oplus e} (q;w_1) \ldots (q;w_m) \xrightarrow{\oplus e} (q;w_{m+1})$, où :

- si $e \in E_M^M$, alors pour tout $i \geq 1$, $w_i = we^i$;
- si $e \in E_M^S$, alors pour tout $i \geq 2$, $w_i = w_1$, et $\begin{cases} w_1 = w \text{ si } e \in \mathit{Alph}(w) \\ w_1 = we \text{ si } e \notin \mathit{Alph}(w) \end{cases}$

Remarques concernant les problèmes de décision pour les AFRs

Remarque 15 Pour les AFRs, le DP et le TP sont triviaux, puisque s'il existe un événement mémorisable dans un AFR R, alors il n'y a pas d'état de blocage dans R. On note d'autre part qu'un AFR sans événement mémorisable est bissimilaire au système de contrôle qui lui est associé ; c'est donc un système de transitions fini, pour lequel tous les problèmes auxquels nous nous intéressons sont décidables.

Remarque 16 Le FRSP est trivialement décidable pour les AFR à file initiale vide : si R est un AFR à file initiale vide, alors $\mathrm{RS}(R)$ est infini ssi il existe, dans le système de contrôle complété associé à R, une transition de mémorisation d'un événement à mémorisation multiple. En effet, puisque tous les états de contrôle de R munis de la file vide sont accessibles, deux cas se présentent :

- s'il existe en l'état de contrôle q une transition de mémorisation d'un événement à mémorisation multiple e, alors puisque $(q;\lambda)$ est accessible dans R, on a $(q;e^*) \subseteq \mathrm{RS}(R)$ (cf. remarque 14).
- s'il n'existe pas dans R de transition de mémorisation d'un événement à mémorisation multiple, alors la file est bornée par le nombre d'événements à mémorisation simple de R, si bien que $\mathrm{RS}(R)$ est fini (cf. remarque 2).

Enfin, la décidabilité du RP, du FRSP, et du QLP pour les AFRs quelconques est une conséquence du résultat que nous exposons en 6.

Chapitre 5

Propriétés des chemins des AFRs

Nous montrons dans ce chapitre une série de résultats qui vont nous servir dans la suite. Nous nous sommes inspirés des équivalences de chemins dans les systèmes half-duplex pour établir ces résultats. Nous avons jugé utile de détailler la plupart des raisonnements, car le modèle des AFRs a une sémantique assez particulière.

5.1 Transformation des occurrences d'événement

Le lemme suivant indique qu'un AFR n'échappe pas à cette grande règle de la nature : rien ne se perd, rien ne se crée, tout se transforme...

Lemme 5.1.1 (de transformation) *Soit R un AFR. Si $\rho = (q;w) \xrightarrow{\sigma} (q';w')$ est un chemin de R, alors :*

$$\Psi(w') - \Psi(w) = \Psi(p^{\oplus}(\sigma)) - \Psi(p^{\ominus}(\sigma)) - \Psi(\pi^p(\rho))$$

Preuve Par récurrence sur la longueur de ρ. Pour la base, on note que si ρ est de longueur nulle, alors $w = w'$ et $\sigma = \lambda$. Pour l'induction, on se donne $n \in \mathbb{N}$ donné, et on suppose que le lemme est vrai lorsque ρ est de longueur n. Soit $\rho = (q;w) \xrightarrow{\sigma} (q';w')$ un chemin de R de longueur $n+1$. ρ s'écrit $(q;w) \xrightarrow{\sigma_1} (p;u) \xrightarrow{\alpha} (q';w')$, et on pose $\rho_1 = (q;w) \xrightarrow{\sigma_1} (p;u)$. Quatre cas se présentent :

- si $\alpha = e_t$: alors $u = w'$, $p^{\ominus}(\sigma) = p^{\ominus}(\sigma_1)$, $p^{\oplus}(\sigma) = p^{\oplus}(\sigma_1)$, et $\pi^p(\rho) = \pi^p(\rho_1)$. On conclut en utilisant l'hypothèse de récurrence sur ρ_1.
- si $\alpha = \ominus e_m$: alors u s'écrit $v_1 e_m v_2$, avec $v_1 \in Mémo(p)^*$, et $w' = v_1 v_2$. Par conséquent, $\Psi(w') = \Psi(u) - \Psi(e_m)$. D'autre part :
  - $p^{\ominus}(\sigma) = p^{\ominus}(\sigma_1) \cdot e_m$, donc $\Psi(p^{\ominus}(\sigma)) = \Psi(p^{\ominus}(\sigma_1)) + \Psi(e_m)$;
  - $p^{\oplus}(\sigma) = p^{\oplus}(\sigma_1)$, donc $\Psi(p^{\oplus}(\sigma)) = \Psi(p^{\oplus}(\sigma_1))$.
  - $\pi^p(\rho) = \pi^p(\rho_1)$, donc $\Psi(pi^p(\rho)) = \Psi(pi^p(\rho_1))$.

On conclut en utilisant l'hypothèse de récurrence sur ρ_1.

- si $\alpha = \oplus e_m$, avec $w' = ue_m$: alors $\Psi(w') = \Psi(u) + \Psi(e_m)$. D'autre part :
  - $p^{\oplus}(\sigma) = p^{\oplus}(\sigma_1) \cdot e_m$, donc $\Psi(p^{\oplus}(\sigma)) = \Psi(p^{\oplus}(\sigma_1)) + \Psi(e_m)$;
  - $p^{\ominus}(\sigma) = p^{\ominus}(\sigma_1)$, donc $\Psi(p^{\ominus}(\sigma)) = \Psi(p^{\ominus}(\sigma_1))$.
  - $\pi^p(\rho) = \pi^p(\rho_1)$, donc $\Psi(pi^p(\rho)) = \Psi(\pi^p(\rho_1))$.

  On conclut en utilisant l'hypothèse de récurrence sur ρ_1.
- si $\alpha = \oplus e_m$, avec $w' = u$: alors $\Psi(w') = \Psi(u)$. D'autre part :
  - $\pi^p(\rho) = \pi^p(\rho_1) \cdot e_m$, donc $\Psi(pi^p(\rho)) = \Psi(pi^p(\rho_1)) + \Psi(e_m)$.
  - $p^{\oplus}(\sigma) = p^{\oplus}(\sigma_1) \cdot e_m$, donc $\Psi(p^{\oplus}(\sigma)) = \Psi(p^{\oplus}(\sigma_1)) + \Psi(e_m)$;
  - $p^{\ominus}(\sigma) = p^{\ominus}(\sigma_1)$, donc $\Psi(p^{\ominus}(\sigma)) = \Psi(p^{\ominus}(\sigma_1))$.

  On conclut en utilisant l'hypothèse de récurrence sur ρ_1. ♦

Corollaire 5.1 *Soit R un AFR. Si $\rho = (q; w) \xrightarrow{\sigma} (q'; w')$ est un chemin de R, alors :*

$$\Psi(w') \geq \Psi(w) - \Psi(p^{\ominus}(\sigma))$$

Preuve On note que si ρ est un chemin de R, alors $\Psi(\pi^p(\rho)) \mid \Psi(\pi^{\oplus}(\rho))$, et par conséquent $\Psi(\pi^p(\rho)) \leq \Psi(\pi^{\oplus}(\rho))$. On conclut avec le lemme de transformation. ♦

Corollaire 5.2 *Soit R un AFR. Si $\rho = (q; w) \xrightarrow{\sigma} (q'; w')$ est un chemin sans mémorisation de R, alors :*

$$\Psi(w') = \Psi(w) - \Psi(p^{\ominus}(\sigma))$$

Preuve D'après la remarque 3, un chemin sans mémorisation de R est un chemin sans perte de R. On conclut avec le lemme de transformation ♦

Corollaire 5.3 *Soit R un AFR. Si $\rho = (q; w) \xrightarrow{\sigma} (q'; w')$ est un chemin de mémorisation de R, alors :*

$$\Psi(w') \geq \Psi(w)$$

Preuve On note que si ρ est un chemin de mémorisation de R, alors $\pi^{\ominus}(\rho) = \lambda$, et donc $\Psi(\pi^{\ominus}(\rho)) = o$. On conclut avec le corollaire 5.1 ♦

Corollaire 5.4 *Soit R un AFR. Si $\rho = (q; w) \longrightarrow (q'; w')$ est un chemin de stabilisation de R, alors :*

$$\Psi(w') + \Psi(\pi^{\ominus}(\rho)) = \Psi(w)$$

Preuve On note qu'un chemin de stabilisation de R est un chemin sans mémorisation de R. On conclut avec le corollaire 5.2. ♦

5.2 Monotonie de $\preceq$

Les résultats de cette section établissent en quelque sorte des équivalences de chemins, et montrent par ailleurs que l'ordre $\preceq$ est monotone pour les AFRs (voir [17]).

Lemme 5.2.1 *Soit R un AFR et $C = (Q, q_0, E, \delta)$ le système de contrôle associé à R. Soit $\rho = (q; w) \xrightarrow{\sigma} (q'; w')$ un chemin de R. Supposons que :*

- *ρ est sans mémorisation ;*
- *w s'écrit $w_1 e$ avec $w_1 \in {E_M}^*$ et $e \in E_M$;*
- *$e \in Alph(w')$.*

Alors w' s'écrit $w'_1 e$ et $\rho_1 = (q; w_1) \xrightarrow{\sigma} (q'; w'_1)$ est un chemin de R sans mémorisation.

Preuve Par récurrence sur la longueur de ρ. Pour la base, on note que si ρ est de longueur nulle, alors $w' = w$, et par conséquent, $w'_1 = w_1$. Pour l'induction, on se donne $n \in \mathbb{N}$ donné, et on suppose que le lemme est vrai lorsque ρ est de longueur n. Soit ρ un chemin de R sans mémorisation de longueur $n+1$. Deux cas se présentent :

- si ρ s'écrit $(q; w) \xrightarrow{\sigma_1} (p; u) \xrightarrow{e_t} (q'; w')$: alors $w' = u$, et donc $e \in Alph(u)$. On utilise l'hypothèse de récurrence, et il vient : u s'écrit $u_1 e$ et $(q; w_1) \xrightarrow{\sigma_1} (p; u_1)$ est un chemin de R. On remarque que $(p; u)$ est stable, et on obtient, d'aprés la remarque 10, que $(p; u_1)$ est stable. Ainsi $(q; w_1) \xrightarrow{\sigma_1} (p; u_1) \xrightarrow{e_t} (q'; u_1)$ est un chemin de R. On conclut en posant $w'_1 = u_1$.

- si ρ s'écrit $(q; w) \xrightarrow{\sigma_1} (p; u) \xrightarrow{\ominus e_m} (q'; w')$: alors u s'écrit $v_1 e_m v_2$, avec $v_1 \in Mémo(p)^*$, et $w' = v_1 v_2$. Comme $e \in Alph(w')$, $e \in Alph(u)$. On utilise l'hypothèse de récurrence, et il vient : u s'écrit $u_1 e$ et $(q; w_1) \xrightarrow{\sigma_1} (p; u_1)$ est un chemin de R.

  Montrons à présent, en raisonnant par l'absurde, que $v_2 \neq \lambda$: si $v_2 = \lambda$, alors $e_m = e$ et $w' = v_1$. Puisque $e_m \notin Mémo(p)$, Il vient $e \notin Mémo(p)$. Or $e \in Alph(w') \subseteq Alph(w') \subseteq Mémo(p)$, ce qui contredit l'hypothèse.

  Puisque $u = u_1 e$, $u = v_1 e_m v_2$ et $v_2 \neq \lambda$, on obtient que v_2 s'écrit $v_3 e$, et il vient $u_1 = v_1 e_m v_3$. Par conséquent $(q; w_1) \xrightarrow{\sigma_1} (p; u_1) \xrightarrow{\ominus e_m} (q'; v_1 v_3)$, et on conclut en notant que $w' = v_1 v_3 e$, et en posant $w'_1 = v_1 v_3$. ♦

Lemme 5.2.2 *Soit R un AFR et $C = (Q, q_0, E, \delta)$ le système de contrôle associé à R. Soit $\rho = (q; w) \xrightarrow{\sigma} (q'; w')$ un chemin de R. Supposons que :*

- *ρ est sans mémorisation ;*
- *w s'écrit $w_1 e$ avec $w_1 \in {E_M}^*$ et $e \in E_M$;*
- *$e \notin Alph(w')$.*

Alors ρ se décompose en $(q;w) \xrightarrow{\tau_1} (p_1;u_1e) \xrightarrow{\ominus e} (p_2;u_1) \xrightarrow{\tau_2} (q';w')$, avec $u_1 \in Mémo(p_1)^*$.

Preuve Puisque ρ est sans mémorisation, d'après le corollaire 5.2, on a $\Psi(p^{\ominus}(\sigma)) = \Psi(w) - \Psi(w')$. Or $e \in Alph(w)$ et $e \notin Alph(w')$, donc $e \in Alph(p^{\ominus}(\sigma))$. En isolant la dernière transition de ρ étiquetée par $\ominus e$, on obtient que ρ s'écrit $(q;w) \xrightarrow{\tau_1} (p_1;u) \xrightarrow{\ominus e} (p_2;v) \xrightarrow{\tau_2} (q';w')$, avec $e \notin Alph(p^{\ominus}(\tau_2))$. Comme $(p_1;u) \xrightarrow{\ominus e} (p_2;v)$, u s'écrit v_1ev_2, avec $v_1 \in Mémo(p)^*$, et $v = v_1v_2$. D'autre part, puisque $e \in Alph(u)$, d'après le lemme 5.2.2, u s'écrit u_1e.

Montrons à présent, en raisonnant par l'absurde, que $v_2 = \lambda$: si $v_2 \neq \lambda$, alors puisque $u = v_1ev_2$ et $u = u_1e$, on obtient que v_2 s'écrit v_3e. il vient $v = v_1v_3e$, d'où $e \in Alph(v)$. Or d'après le corollaire 5.2, $\Psi(p^{\ominus}(\tau_2)) = \Psi(v) - \Psi(w')$, et par ailleurs $e \notin Alph(w')$. Par conséquent $e \in Alph(p^{\ominus}(\tau_2))$, ce qui contredit l'hypothèse.

Finalement, $u = v_1e = u_1$, donc $u_1 = v_1 \in Mémo(p)^*$. ♦

Lemme 5.2.3 *Soit R un AFR et $C = (Q, q_0, E, \delta)$ le système de contrôle associé à R. Soit $\rho = (q;w) \longrightarrow (q';w')$ un chemin de R. Supposons que :*

- *ρ est sans mémorisation ;*
- *w s'écrit w_1e avec $w_1 \in {E_M}^*$ et $e \in E_M$.*

Alors il existe un chemin $\rho_1 = (q;w_1) \longrightarrow (q';w_1')$ dans R tel que :

- *$w_1' \leq w'$;*
- *ρ_1 est sans mémorisation, et $\rho_1 |_{\ominus} \rho$;*

Preuve Deux cas se présentent, suivant que la dernière occurrence mémorisée de e est pas traitée dans ρ, ou pas :

- si $e \in Alph(w')$: on note σ la trace de ρ, et d'après le lemme 5.2.1, d'une part w' s'écrit $w_1'e$, et $\rho_1 = (q;w_1) \xrightarrow{\sigma} (q';w_1')$ est un chemin de R sans mémorisation.
- si $e \notin Alph(w')$: d'après le lemme 5.2.2, ρ s'écrit $(q;w) \xrightarrow{\tau_1} (p_1;ue) \xrightarrow{\ominus e} (p_2;u) \xrightarrow{\tau_2} (q';w')$, avec $u \in Mémo(p_1)^*$. Du lemme 5.2.1, on déduit que $(q;w_1) \xrightarrow{\tau_1} (p_1;u)$ est un chemin de R. Comme $u \in Mémo(p_1)^*$, $(p_1;u) \xrightarrow{e} (p_2;u)$ est une transition de R, on pose finalement $w_1' = w'$, et $\rho_1 = (q;w_1) \xrightarrow{\tau_1} (p_1;u) \xrightarrow{e} (p_2;u) \xrightarrow{\tau_2} (q';w')$. On conlut en notant ρ_1 est un chemin de R sans mémorisation vérifiant $\rho_1 |_{\ominus} \rho$. ♦

Lemme 5.2.4 *Soit R un AFR et $\rho = (q;w) \longrightarrow (q';w')$ un chemin de R sans mémorisation. Pour tout $w_1 \leq w$, il existe un chemin $\rho_1 = (q;w_1) \longrightarrow (q';w_1')$ dans R tel que :*

- *$w_1' \leq w'$;*
- *ρ_1 est sans mémorisation, et $\rho_1 |_{\ominus} \rho$;*

Preuve Par récurrence sur $|w_1| - |w|$: la base est triviale car alors $w_1 = w$, et l'induction est montrée par le lemme 5.2.3. ♦

Proposition 5.1 *Soit R un AFR et $\rho = (q;w) \longrightarrow (q';w')$ un chemin de R. Pour tout $w_1 \leq w$, il existe un chemin $\rho_1 = (q;w_1) \longrightarrow (q';w'_1)$ dans R tel que :*

- $w'_1 \leq w'$;
- *ρ_1 est sans mémorisation, et $\rho_1 \mid_\ominus \rho$.*

Preuve Par récurrence sur le nombre de transitions de mémorisation de ρ. Pour la base, on note que si ρ n'a pas de transition de mémorisation, alors ρ est un chemin sans mémorisation, et le lemme est donc montré en prenant $w_1 = w'$ et $\rho_1 = \rho$. Pour l'induction, on se donne $n \in \mathbb{N}$ donné, et on suppose que le lemme est vrai lorsque ρ contient n transitions de mémorisations. Soit $\rho = (q;w) \longrightarrow (q';w')$ un chemin de R contenant $n+1$ transitions de mémorisations, et $w \leq w_1$. ρ s'écrit $(q;w) \xrightarrow{\sigma} (p_1;u) \xrightarrow{\oplus e} (p_1;ue) \xrightarrow{\tau} (q';w')$, avec $(q;w) \xrightarrow{\sigma} (p_1;u)$ sans mémorisation. De cette dernière assertion, et comme $w_1 \leq w$, on déduit, d'après le lemme 5.2.4, qu'il existe un chemin sans mémorisation $(q;w_1) \xrightarrow{\sigma_1} (p_1;u_1)$ dans R tel que :

- $u_1 \leq u$;
- $(q;w_1) \xrightarrow{\sigma_1} (p_1;u_1) \mid_\ominus (q;w) \xrightarrow{\sigma} (p_1;u)$.

Or, d'une part $u_1 \leq u \leq ue$, et d'autre part $(p_1;ue) \xrightarrow{\tau} (q';w')$ contient n transitions de mémorisation. On peut donc utiliser l'hypothèse de récurrence, et on obtient qu'il existe un chemin sans mémorisation $(p_1;u_1) \xrightarrow{\tau_1} (q;w'_1)$ dans R tel que :

- $w'_1 \leq w'$;
- $(p_1;u_1) \xrightarrow{\tau_1} (q;w'_1) \mid_\ominus (p_1;ue) \xrightarrow{\tau} (q';w')$

On conclut en posant $\rho_1 = (q;w_1) \xrightarrow{\sigma_1} (p_1;u_1) \xrightarrow{\tau_1} (q;w'_1)$, qui est un chemin sans mémorisation de R vérifiant $\rho_1 \mid_\ominus \rho$. ♦

5.3 Caractérisation du chemin sans mémorisation associé à un chemin donné

Les deux théorèmes que nous démontrons ici sont essentiels pour la suite, car ils permettent de restreindre les recherches de chemins, en se limitant à l'analyse des chemins sans mémorisation. Nous prouvons d'abord le résultat pour des chemins finis, puis nous l'étendons pour les chemins infinis. Le résultat est utilisé dans la suite pour le model-checking.

5.3.1 Chemin sans mémorisation associé à un chemin fini

Proposition 5.2 *Soit R un AFR et $\rho = (q; w) \longrightarrow$ un chemin de R. Pour tout $w_1 \leq w$, il existe un* ***unique*** *chemin $\rho_1 = (q; w_1) \longrightarrow$ dans R tel que ρ_1 est sans mémorisation, et $\pi^{\ominus\&c}(\rho_1) = \pi^{\ominus\&c}(\rho)$.*

Preuve D'après la proposition 5.1, on obtient l'existence. Il reste à montrer l'unicité. Pour tout chemin sans mémorisation ρ de trace σ, on a $\pi^{\ominus\&c}(\rho) = \sigma$. On conclut avec la remarque 1, d'après laquelle tout chemin est uniquement déterminé par son état origine et sa trace. ♦

Définition 5.1 *Soit R un AFR. Pour tout chemin $\rho = (q; w) \longrightarrow (q'; w')$ de R, et pour tout $w_1 \leq w$, on note $\Theta_R(\rho, w_1)$ l'unique chemin $\rho_1 = (q; w_1) \longrightarrow$ dans R tel que :*

1. *ρ_1 est sans mémorisation ;*
2. *$\pi^{\ominus\&c}(\rho_1) = \pi^{\ominus\&c}(\rho)$.*

Lemme 5.3.1 *Soit R un AFR et $\rho = (q; w) \longrightarrow (q'; w')$ un chemin de R. Pour tout $w_1 \leq w$, d'une part $\Theta_R(\rho, w_1)$ s'écrit $(q; w_1) \longrightarrow (q'; w'_1)$, avec $w'_1 \leq w'$, et d'autre part $\Theta_R(\rho, w_1) \mid_{\ominus} \rho$.*

Preuve Immédiate à partir de la proposition 5.1 et de la définition 5.1. ♦

Lemme 5.3.2 *Soit R un AFR et $\rho_p = (p; w_p) \longrightarrow (p'; w'_p)$, $\rho_q = (q; w_q) \longrightarrow (q'; w'_q)$ deux chemins de R. Supposons que $(p'; w'_p) = (q; w_q)$. Alors pour tout $w_1 \leq w_q$, si w'_1 désigne la file de l'état arrivée de $\Theta_R(\rho_p, w_1)$, alors $\Theta_R(\rho_p \rho_q, w_1) = \Theta_R(\rho_p, w_1)\Theta_R(\rho_q, w'_1)$.*

Preuve Supposons que $(p'; w'_p) = (q; w_q)$. Soit $w_1 \leq w_q$. On note que si w'_1 désigne la file de l'état arrivée de $\Theta_R(\rho_q, w_1)$, alors $\Theta_R(\rho_p, w_1)\Theta_R(\rho_p, w'_1)$ est un chemin de R vérifiant ls conditions 1 et 2 de la définition 5.1. Par conséquent $\Theta_R(\rho_p, w_1)$, alors $\Theta_R(\rho_p \rho_q, w_1) = \Theta_R(\rho_p, w_1)\Theta_R(\rho_q, w'_1)$. ♦

Lemme 5.3.3 *Soit R un AFR et $\rho = (q_1; w_1) \xrightarrow{\alpha_1} (q_2; w_2) \ldots (q_m; w_m) \xrightarrow{\alpha_m} (q_{m+1}; w_{m+1})$ un chemin de R. Si $|\pi^{\ominus\&c}(\rho)| \geq 1$, alors $\mu = \mathrm{Min}\{j \geq 1 \ / \ \alpha_j \notin \{\oplus e \ / \ e \in E\}\}$ est bien défini[1] et on a :*

- *$(q_1; w_1) \xrightarrow{\alpha_1} (q_2; w_2) \ldots (q_m; w_m) \xrightarrow{\alpha_m} (q_{\mu-1}; w_{\mu-1})$ est un chemin de mémorisation de R ;*
- *$\rho_\mu = (q_{\mu+1}; w_{\mu+1}) \xrightarrow{\alpha_{\mu+1}} (q_{\mu+1}; w_{\mu+1}) \ldots (q_m; w_m) \xrightarrow{\alpha_m} (q_{m+1}; w_{m+1})$ un chemin de R vérifiant $|\pi^{\ominus\&c}(\rho_\mu)| = |\pi^{\ominus\&c}(\rho)| - 1$.*

Preuve Supposons que $|\pi^{\ominus\&c}(\rho)| \geq 1$. Alors ρ n'est pas un chemin de mémorisation, ce qui implique $\{j \geq 1 \ / \ \alpha_j \notin \{\oplus e \ / \ e \in E\}\} \neq \emptyset$. Ainsi,

1. i.e. $\{j \geq 1 \ / \ \alpha_j \notin \{\oplus e \ / \ e \in E\}\} \neq \emptyset$.

$\mu = \text{Min}\{j \geq 1 \ / \ \alpha_j \notin \{\oplus e \ / \ e \in E\}\}$ est bien défini. Par définition même de μ, les deux assertions restant à prouver sont triviales. ♦

Théorème 5.1 *Soit R un AFR et $\rho = (q_1; w_1) \xrightarrow{\alpha_1} (q_2; w_2) \ldots (q_m; w_m) \xrightarrow{\alpha_m} (q_{m+1}; w_{m+1})$ un chemin de R. On pose $k = |\pi^{\ominus \& c}(\rho)|$, et on définit la suite $(\mu_1, \ldots, \mu_k)$ par*[2] *:*

- $\mu_1 = 1$;
- *pour tout* $i \in \{1, \ldots, k\}$, $\mu_{i+1} = \text{Min}\{j \geq \mu_i \ / \ \alpha_j \notin \{\oplus e \ / \ e \in E\}\}$.

On a :

1. $q_{\mu_{k+1}} = q_m$;
2. *pour tout* $i \in \{1, \ldots, k\}$, $\alpha_{\mu_i+1} \ldots \alpha_{\mu_{i+1}-1}$ *est une séquence de mémorisation ;*
3. $\alpha_{\mu_{k+1}+1} \ldots \alpha_m$ *est une séquence de mémorisation ;*
4. *pour tout* $\widetilde{w}_1 \leq w$, $\Theta_R(\rho, \widetilde{w}_1)$ *s'écrit :*

$$(q_{\mu_1}; \widetilde{w}_1) \xrightarrow{\widetilde{\alpha}_{\mu_1}} (q_{\mu_2}; \widetilde{w}_2) \ldots (q_{\mu_k}; \widetilde{w}_k) \xrightarrow{\widetilde{\alpha}_{\mu_k}} (q_{\mu_{k+1}}; \widetilde{w}_{k+1})$$

avec :

- *pour tout* $i \in \{1, \ldots, k+1\}$, $\widetilde{w}_i \leq w_{\mu_i}$;
- *pour tout* $i \in \{1, \ldots, k\}$,
  - *si* α_{μ_i} *s'écrit* $\ominus e_{\mu_i}$, *alors* $\widetilde{\alpha}_{\mu_i} = \ominus e_{\mu_i}$ *ou* $\widetilde{\alpha}_{\mu_i} = e_{\mu_i}$;
  - *sinon,* $\alpha_{\mu_i} \in E_M$, *et alors* $\widetilde{\alpha}_{\mu_i} = \alpha_{\mu_i}$.

Preuve On procède par récurrence sur $|\pi^{\ominus \& c}(\rho)|$. Pour la base, on note que si $k = 0$, alors d'une part la suite $(\mu_1, \ldots, \mu_k)$ est restreinte à un élément, et d'autre part, comme ρ est un chemin de mémorisation de R, pour tout $\widetilde{w}_1 \leq w$, $\Theta_R(\rho, \widetilde{w}_1)$ est le chemin vide. L'induction quant à elle est fstidieuse mais triviale, en considérant le chemin $\rho' = (q_1; w_1) \xrightarrow{\alpha_1} (q_2; w_2) \ldots (q_{\mu_k-1}; w_{\mu_k-1}) \xrightarrow{\alpha_{\mu_k-1}} (q_{\mu_k}; w_{\mu_k})$, et en utilisant :

- le lemme 5.3.3 et la remarque 4 pour les points 1, 2 et 3.
- les lemmes 5.3.1 5.3.2 pour le le quatrième point. ♦

5.3.2 Chemin sans mémorisation associé à un chemin infini

Proposition 5.3 *Soit R un AFR et $\rho = (q; w) \longrightarrow \ldots$ un chemin infini de R. Si $|\pi^{\ominus \& c}(\rho)| < \infty$, alors il existe deux chemins ρ_1 et ρ_m dans R tels que :*

- ρ_m *est un chemin de mémorisation ;*

2. d'après le lemme 5.3.3, c'est une bonne définition.

- $\rho = \rho_1 \rho_m$.

Preuve Supposons que $|\pi^{\ominus \& c}(\rho)| < \infty$. Alors ρ s'écrit $\rho_1 \rho_m$, avec $|\pi^{\ominus \& c}(\rho_m)| = 0$. Ainsi, ρ_m est un chemin de mémorisation. ♦

Théorème 5.2 *Soit R un AFR et $\rho = (q_1; w_1) \xrightarrow{\alpha_1} (q_2; w_2) \ldots (q_m; w_m) \xrightarrow{\alpha_m} (q_{m+1}; w_{m+1}) \ldots$ un chemin infini de R. On suppose que $|\pi^{\ominus \& c}(\rho)| = \infty$, et on définit la suite $(\mu_k)_{k \in \mathbb{N}^*}$ par:*

- $\mu_1 = 1$;
- *pour tout $i \in \mathbb{N}^*$, $\mu_{i+1} = \mathrm{Min}\{j \geq \mu_i \ / \ \alpha_j \notin \{\oplus e \ / \ e \in E\}\}$.*

On a:

1. *pour tout $i \in \mathbb{N}^*$, $\alpha_{\mu_i + 1} \ldots \alpha_{\mu_{i+1} - 1}$ est une séquence de mémorisation;*
2. *pour tout $\widetilde{w}_1 \leq w$, il existe un unique chemin $\rho_1 = (q; w_1) \longrightarrow$ dans R tel que:*

   (a) ρ_1 est sans mémorisation;
   (b) $\pi^{\ominus \& c}(\rho_1) = \pi^{\ominus \& c}(\rho)$.

   Ce chemin est noté $\Theta_R(\rho, \widetilde{w}_1)$, et il s'écrit:

$$(q_{\mu_1}; \widetilde{w}_1) \xrightarrow{\widetilde{\alpha}_{\mu_1}} (q_{\mu_2}; \widetilde{w}_2) \ldots (q_{\mu_k}; \widetilde{w}_k) \xrightarrow{\widetilde{\alpha}_{\mu_k}} (q_{\mu_{k+1}}; \widetilde{w}_{k+1}) \ldots$$

   avec:

   - *pour tout $i \in \mathbb{N}^*$, $\widetilde{w}_i \leq w_{\mu_i}$;*
   - *pour tout $i \in \mathbb{N}^*$,*
     - *si α_{μ_i} s'écrit $\ominus e_{\mu_i}$, alors $\widetilde{\alpha}_{\mu_i} = \ominus e_{\mu_i}$ ou $\widetilde{\alpha}_{\mu_i} = e_{\mu_i}$;*
     - *sinon, $\alpha_{\mu_i} \in E$, et alors $\widetilde{\alpha}_{\mu_i} = \alpha_{\mu_i}$.*

Preuve On donne juste l'idée de la preuve. Le premier point provient directement de la définition de $(\mu_k)_{k \in \mathbb{N}^*}$. Pour le deuxième point, l'unicité est obtenue de la même manière que dans la proposition 5.2, et l'existence, ainsi que la forme donnée, est construite par récurrence, en utilisant le théorème 5.1. ♦

5.4 Chemin de stabilisation associé à un chemin sans mémorisation

Le lemme et la proposition qui suivent généralisent la remarque 12.

Lemme 5.4.1 *Soit R un AFR et $C = (Q, q_0, E, \delta)$ le système de contrôle associé à R. Soit $\rho = (q; w) \xrightarrow{\sigma} (q'; w')$ un chemin sans mémorisation de R. Supposons que σ s'écrive $\sigma_1 \cdot \ominus e_1 \ldots \sigma_k \cdot \ominus e_k \cdot \sigma_{k+1}$, avec:*

- *pour tout $i \in \{1, \ldots, k+1\}$, $\sigma_i \in {E_M}^*$;*

– *pour tout* $i \in \{1, \ldots, k\}, e_i \in E_M$.

Alors pour tout $z \in {E_M}^*$, $(q; w \cdot \sigma_1 \ldots \sigma_{k+1} \cdot z) \xrightarrow{\ominus\sigma_1 \cdot \ominus e_1 \ldots \ominus\sigma_k \cdot \ominus e_k \cdot \ominus\sigma_{k+1}} (q'; w'z)$ *dans* R.

Preuve Par récurrence sur la longueur de ρ. Pour la base, on note que si ρ est de longueur nulle, alors $w'z = wz$, et $\sigma = \lambda$. Pour l'induction, on se donne $n \in \mathbb{N}$ donné, et on suppose que le lemme est vrai lorsque ρ est de longueur n. Soit $\rho = (q; w) \xrightarrow{\sigma} (q'; w')$ un chemin de R sans mémorisation, de longueur $n+1$, et tel que σ s'écrive $\sigma_1 \cdot \ominus e_1 \ldots \sigma_k \cdot \ominus e_k \cdot \sigma_{k+1}$, avec :

– pour tout $i \in \{1, \ldots, k+1\}$, $\sigma_i \in {E_M}^*$;

– pour tout $i \in \{1, \ldots, k\}, e_i \in E_M$.

Soit $z \in {E_M}^*$. Deux cas se présentent :

– si ρ s'écrit $(q; w) \xrightarrow{\tau} (p; w') \xrightarrow{e_t} (q'; u)$: alors $u = w'$, et :

  – σ_{k+1} s'écrit $\sigma'_{k+1} e_t$;
  – $\tau = \sigma_1 \cdot \ominus e_1 \ldots \sigma_k \cdot \ominus e_k \cdot \sigma'_{k+1}$.

  Par hypothèse de récurrence, et comme $e_t z \in {E_M}^*$, il vient :

$$(q; w \cdot \sigma_1 \ldots \sigma'_{k+1} \cdot e_t z) \xrightarrow{\ominus\sigma_1 \cdot \ominus e_1 \ldots \ominus\sigma_k \cdot \ominus e_k \cdot \ominus\sigma'_{k+1}} (p; w' e_t z)$$

  Puisque $(p; w') \xrightarrow{e_t} (q'; w')$ est une transition de R, $w' \in \mathit{Mémo}(p)^*$. Par conséquent, $(p; w' e_t z) \xrightarrow{\ominus e_t} (q'; w'z)$ est une transition de R. On conclut en notant que $\sigma_1 \ldots \sigma_{k+1} = \sigma_1 \ldots \sigma'_{k+1} e_t$.

– si ρ s'écrit $(q; w) \xrightarrow{\sigma_1} (p; u) \xrightarrow{\ominus e_m} (q'; w')$: alors

  – $\sigma_{k+1} = \lambda$, et $e_m = e_k$;
  – $\tau = \sigma_1 \cdot \ominus e_1 \ldots \sigma_{k-1} \ominus e_{k-1} \cdot \sigma_k$.

  Par hypothèse de récurrence, il vient :

$$(q; w \cdot \sigma_1 \ldots \sigma_k \cdot z) \xrightarrow{\ominus\sigma_1 \cdot \ominus e_1 \ldots \ominus\sigma_{k-1} \ominus e_{k-1} \cdot \ominus\sigma_k} (p; uz)$$

  Puisque $(p; u) \xrightarrow{\ominus e_k} (q'; w')$ est une transition de R, $(p; uz) \xrightarrow{\ominus e_k} (q'; w'z)$ est une transition de R. On conclut en notant que $\sigma_1 \ldots \sigma_{k+1} = \sigma_1 \ldots \sigma_k$.
  ♦

Proposition 5.4 *Soit* R *un AFR et* $C = (Q, q_0, E, \delta)$ *le système de contrôle associé à* R. *Soit* $\rho = (q; w) \xrightarrow{\sigma} (q'; w')$ *un chemin sans mémorisation de* R. *Si* $p^c(\sigma) \in {E_M}^*$, *alors pour tout* $z \in {E_M}^*$, $(q; w \cdot p^c(\sigma) \cdot z) \longrightarrow (q'; w'z)$ *dans* R.

Preuve On note que σ s'écrit $\sigma_1 \cdot \ominus e_1 \ldots \sigma_k \cdot \ominus e_k \cdot \sigma_{k+1}$, avec :

– pour tout $i \in \{1, \ldots, k+1\}$, $\sigma_i \in {E_M}^*$;

– pour tout $i \in \{1, \ldots, k\}, e_i \in E_M$.

Avec ces notations, $p^c(\sigma) = \sigma_1 \ldots \sigma_{k+1}$. On conclut avec le lemme 5.4.1. ♦

5.5 Finitude de l'arbre d'accessibilité sans mémorisation

Proposition 5.5 *Soit R un AFR. $\mathrm{RT}^{\ominus\&c}(R)$, l'arbre d'accessibilité sans mémorisation de R, est fini.*

Preuve Nous donnons une preuve quelque peu informelle, car le raisonnement est assez classique. Soit $\overline{C}$ le système de contrôle complété associé à R. Soit d_{max} le nombre maximal de transitions sortantes d'un état de $\overline{C}$. On vérifie que le degré de chaque nœud de $\mathrm{RT}^{\ominus\&c}(R)$ est borné par d_{max}. En appliquant le lemme de Koenig, il vient que $\mathrm{RT}^{\ominus\&c}(R)$ est infini ssi il admet une branche infinie. Or par définition, $\mathrm{RT}^{\ominus\&c}(R)$ ne peut admettre de branche infinie, parce que sur une telle branche, un même état de contrôle q reviendrait infiniment souvent. Mais alors, par définition de $\mathrm{RT}^{\ominus\&c}(R)$, on aurait une suite infinie de files distinctes pour cet état de contrôle q. Sachant que les files sont bornées par la taille de la file initiale, puisqu'on ne fait pas de mémorisation, on aboutirait à une contradiction. ♦

Corollaire 5.5 *Soit R un AFR. $\mathrm{RT}^{\ominus\&c}(R)$, l'arbre d'accessibilité sans mémorisation de R, est est calculable.*

Preuve Immédiate d'après la proposition 5.5. ♦

Corollaire 5.6 *Soit R un AFR. $\mathrm{RS}^{\ominus\&c}(R)$, l'ensemble des états accessibles sans mémorisation de R, est fini et calculable. L'appartenance à $\mathrm{RS}^{\ominus\&c}(R)$ est décidable.*

Preuve Immédiate d'après le corollaire 5.5 et la remarque 6. ♦

Corollaire 5.7 *Soit R un AFR. $\mathrm{RG}^{\ominus\&c}(R)$, le graphe d'accessibilité sans mémorisation de R, est fini et calculable.*

Preuve Immédiate d'après le corollaire 5.5 et la définition de $\mathrm{RG}^{\ominus\&c}(R)$. ♦

Chapitre 6

Calcul de l'ensemble reconnaissable des états accessibles

Nous montrons dans ce chapitre que l'ensemble des états accessibles d'un AFR est reconnaissable et effectivement calculable. Ce résultat, qui implique que le RRP est décidable pour les AFRs, nous permet de décider également le RP, le FRSP et le QLP. Pour simplifier la présentation, nous montrons d'abord ce résultat pour les AFRs à file initiale vide, puis nous l'étendons aux AFRs quelconques. Des exemples sont donnés avec les explications de l'implémentation à la fin du chapitre.

Nous utiliserons dans tout ce chapitre la définition suivante :

Définition 6.1 *Soit un R un AFR quelconque associé à un système de contrôle $C = (Q, q_0, E, \delta)$. Pour tout $q \in Q$, on appelle* langage de la file en l'état de contrôle q, *noté $\mathcal{L}_R(q)$ l'ensemble :*

$$\mathcal{L}_R(q) = \{w \in {E_M}^* \ / \ (q; w) \in \mathrm{RS}(R)\}$$

Dans la suite de ce chapitre, nous prouvons que pour tout AFR R :

- $\mathrm{RS}(R)$ est reconnaissable ;
- une représentation reconnaissable de $\mathrm{RS}(R)$ est effectivement calculable.

On note que si R est un AFR, alors $R(S)$ est reconnaissable ssi pour tout état de contrôle q de R, $\mathcal{L}_R(q)$ est rationnel. En effet, si Q désigne l'ensemble des états de contrôle de R[1], alors $\mathrm{RS}(R) = \cup_{q \in Q}\{(q; w) \ / \ w \in \mathcal{L}_R(q)\}$.

Donnons rapidement les idées clés de la preuve qui va suivre. Soit R un AFR quelconque.

1. Si $(q; w) \in \mathrm{RS}(R)$ est un état stable, alors $(q; wM\acute{e}mo(q)^*) \subset \mathrm{RS}(R)$. En particulier, si R est à file initiale vide, alors d'après la remarque 13, pour tout état de contrôle q de R, $(q; M\acute{e}mo(q)^*) \subset \mathrm{RS}(R)$.

1. on rappelle qu'alors Q est fini.

2. Si $(q;w) \in \mathrm{RS}(R)$ est un état instable, alors soit $(q;w)$ provient d'une séquence de stabilisation à partir de l'état intial de R, soit il existe un état stable $(q_{st};w_{st}) \in \mathrm{RS}(R)$ et un chemin $(q_{st};w_{st}) \xrightarrow{e_t} (q_1;w_{st}) \xrightarrow{\sigma} (q;w)$ dans R tel que σ soit une séquence de stabilisation, auquel cas $w \in Mémo(q_{st})^*$.

3. Les événements à mémorisation simple restent toujours en nombre borné dans la file[2], et font intervenir une analyse particulière pour les prendre en compte.

6.1 Résultats préliminaires

Établissons tout d'abord quelques résultats qui nous servirons dans la suite, et qui indiquent la façon dont nous allons utiliser la stabilité des états et les séquences de stabilisation.

Proposition 6.1 *Soit R un AFR et $\rho = (q;w) \longrightarrow (q';w')$ un chemin de R. Supposons que :*

- *$(q';w')$ est instable ;*
- *ρ n'est pas un chemin de stabilisation.*

Alors il existe un chemin $(q;w) \longrightarrow (q_{st};w_{st}) \xrightarrow{e_t} (q_1;w_{st}) \xrightarrow{\sigma} (q';w')$ dans R tel que :

- *$(q_{st};w_{st})$ est un état stable ;*
- *σ est une séquence de stabilisation.*

Preuve Supposons que $(q';w')$ est instable et que ρ n'est pas un chemin de stabilisation. D'après la remarque 7, ρ passe par un état stable. On en déduit, puisque $(q';w')$ est instable et en choisissant pour $(q_{st};w_{st})$ le dernier état stable de ρ, que ρ s'écrit $(q;w) \longrightarrow (q_{st};w_{st}) \xrightarrow{e_t} (q_1;w_{st}) \xrightarrow{\sigma} (q';w')$ avec :

- $(q_{st};w_{st})$ est stable ;
- σ est une séquence de stabilisation. ♦

Lemme 6.1.1 *Soit R un AFR, et $\rho = (q;w) \xrightarrow{e_t} (q_1;w) \xrightarrow{\sigma} (q';w')$ un chemin de R. Supposons que :*

- *$(q;w)$ est un état stable ;*
- *σ est une séquence de stabilisation.*

Alors on a :

- *$w' \in Mémo(q)^*$ et $p^{\ominus}(\sigma) \in Mémo(q)^*$.*

2. on vérifie, en utilisant la remarque 9, que si w_0 est la file initiale de R, alors pour tout $(q;w) \in \mathrm{RS}(R)$, et pour tout événement à mémorisation simple e de R, $|w|_e \leq |w_0|_e$.

– $q \xrightarrow{e_t} q_1 \xrightarrow{\sigma} q'$ *est un chemin de* C.

Preuve Supposons que $(q; w)$ est un état stable et que σ est une séquence de stabilisation. On déduit du corollaire 5.4 que :

– $\Psi(w') \leq \Psi(w)$. Donc $Alph(w') \subseteq Alph(w)$. Or $w' \in Mémo(q)^*$. Par conséquent, $w' \in Mémo(q)^*$.

– $\Psi(\pi^{\ominus}(\rho)) \leq \Psi(w)$. Donc $Alph(\pi^{\ominus}(\rho)) \subseteq Alph(w)$. Or $w \in Mémo(q)^*$. Par conséquent, $p^{\ominus}(\sigma) \in Mémo(q)^*$.

Enfin, par définition de la construction de R à partir de C, $q \xrightarrow{e_t} q_1 \xrightarrow{\sigma} q'$ est un chemin de C. ♦

6.2 AFR à file initiale vide — Cas où $E_M^S = \emptyset$

Soit R un AFR à file initiale vide associé à un système de contrôle $C = (Q, q_0, E, \delta)$. Nous supposons également que tous les événements mémorisables de E sont à mémorisation multiple, i.e. $E_M^S = \emptyset$.

Définition 6.2 *La relation* $\mathcal{M}_\lambda^M \subseteq Q \times Q$ *est définie par: pour tout* $(p, q) \in Q \times Q$, $p\mathcal{M}_\lambda^M q$ *ssi l'une des assertions suivantes est vraie :*

– $p = q$;

– *il existe un chemin* $p \xrightarrow{e} q_1 \xrightarrow{\sigma} q$ *dans* C *tel que* $\sigma \in Mémo(p)^*$.

On constate que pour tout $q \in Q$, l'ensemble $\{p \in Q \ / \ p\mathcal{M}_\lambda^M q\}$ est fini, puisque Q est fini. La relation $\mathcal{M}_\lambda^M$ est en fait définie de façon à pouvoir exprimer de manière reconnaissable $\mathrm{RS}(R)$, comme l'indique le théorème suivant.

Théorème 6.1 *Pour tout* $q \in Q$, $\mathcal{L}_R(q) = \bigcup_{p\mathcal{M}_\lambda^M q} Mémo(p)^*$.

Preuve Commençons par prouver $\subseteq$. Soit $q \in Q$, et $w \in \mathcal{L}_R(q)$. Deux cas se présentent :

– si $(q; w)$ est stable : alors par définition, $w \in Mémo(q)^*$, et d'autre part $q\mathcal{M}_\lambda^M q$.

– si $(q; w)$ est instable : comme $(q; w) \in \mathrm{RS}(R)$, il existe un chemin $\rho = (q_0, \lambda) \longrightarrow (q; w)$ dans R. Puisque (q_0, λ) est stable, ρ n'est pas un chemin de stabilisation, et comme $(q; w)$ est instable, on déduit de la proposition 6.1 qu'il existe un chemin $(q_{st}; w_{st}) \xrightarrow{e_t} (q_1; w_{st}) \xrightarrow{\sigma} (q'; w')$ dans R tel que :

  – $(q_{st}; w_{st})$ est un état stable ;
  – σ est une séquence de stabilisation.

  Puisque σ est une séquence de stabilisation, il vient, d'après le lemme 6.1.1, d'une part $w \in Mémo(q_{st})^*$, et d'autre part $q_{st}\mathcal{M}_\lambda^M q$.

Finalement :

$$w \in \bigcup_{p\mathcal{M}_\lambda^M q} Mémo(q)^*$$

Montrons à présent $\supseteq$. Soit $p, q \in Q$ tels que $p\mathcal{M}_\lambda^M q$, et soit $w \in Mémo(p)^*$. On note que d'après la remarque 13, $(p;\lambda) \in \mathrm{RS}(R)$. Deux cas se présentent :

- si $p = q$: alors $(q; w)$ est stable. Comme $(q;\lambda) \in \mathrm{RS}(R)$, il vient, d'après la remarque 11, $(q; w) \in \mathrm{RS}(R)$.
- si $p \neq q$: alors puisque $p\mathcal{M}_\lambda^M q$, il existe dans C un chemin $p \xrightarrow{e} q_1 \xrightarrow{e_1 \ldots e_k} q$ tel que $e_1 \ldots e_k \in Mémo(p)^*$. On a $(e_1 \ldots e_k) \cdot w \in Mémo(p)^*$, et $(p;\lambda) \in \mathrm{RS}(R)$ est un état stable. Par conséquent, d'après la remarque 11, $(p; (e_1 \ldots e_k) \cdot w) \in \mathrm{RS}(R)$. On obtient donc, d'après la remarque 12, $(q; w) \in \mathrm{RS}(R)$.

Finalement :

$$w \in \mathcal{L}_R(q) \qquad \blacklozenge$$

La proposition suivante permet de montrer que l'appartenance à $\mathcal{M}_\lambda^M$ est décidable.

Proposition 6.2 *Pour tout $(p,q) \in Q \times Q$, tel que $p \neq q$, $p\mathcal{M}_\lambda^M q$ ssi il existe un chemin **élémentaire** $p \xrightarrow{e} q_1 \xrightarrow{\sigma} q$ dans C tel que $\sigma \in Mémo(q)^*$.*

Preuve Le sens $\Longleftarrow$ étant trivial, on montre le sens $\Longrightarrow$. Supposons donc que $p\mathcal{M}_\lambda^M q$, avec $p \neq q$. Par définition de $\mathcal{M}_\lambda^M$, il existe un chemin $\rho = p \xrightarrow{e} q_1 \xrightarrow{\sigma} q$ dans C tel que $\sigma \in Mémo(q)^*$. On "élémentarise" le chemin ρ à l'aide de l'opérateur Ξ décrit ci-dessous.

Soit Ξ l'opérateur qui à tout chemin $c = p_0 \longrightarrow p_1 \longrightarrow \ldots p_{l-1} \longrightarrow p_l$ de C associe le chemin :

- $\Xi(c) = c$ si c est élémentaire ;
- $\Xi(c) = p_0 \longrightarrow \ldots p_{i_0} \longrightarrow p_{j_0} \longrightarrow \ldots p_l$, si c n'est pas élémentaire, avec i_0 et j_0 définis par[3] :
  - $i_0 = \mathrm{Min}\{i \in \{0, \ldots, l-1\} \ / \ \exists\ j \in \{i+1, \ldots, l\}, p_j = p_i\}$;
  - $j_0 = \mathrm{Max}\{j \in \{i_0+1, \ldots, l\} \ / \ p_j = p_{i_0}\}$.

On note que si c est un chemin, alors :

- $\Xi(c)$ a le même état origine et le même état d'arrivée que c ;
- la trace du chemin $\Xi(c)$ est un sous-mot de la trace de c.

Soit à présent (ρ_n) la suite de chemins définie par $\rho_0 = \rho$ et $\forall\, n \in \mathbb{N}, \rho_{n+1} = \Xi(\rho_n)$. Par définition de Ξ, (ρ_n) est stationnaire à partir d'un certain rang n_0, et posons $\widetilde{\rho} = \rho_{n_0}$. Puisque $p \neq q$, et comme $\widetilde{\rho}$ a p pour état origine et q pour état arrivée, on peut écrire $\widetilde{\rho} = p \xrightarrow{\widetilde{e}} q_1 \xrightarrow{\widetilde{\sigma}} q$. On a :

- $\widetilde{\rho}$ est élémentaire ;

3. on vérifie que c'est une bonne définition, i.e. les ensembles considérés ne sont pas vides.

– $\widetilde{e\sigma}|e\sigma$, et il vient donc $\widetilde{\sigma}|\sigma$. Par conséquent, $\widetilde{\sigma} \in Mémo(q)^*$. ♦

Corollaire 6.1 *Pour tout $(p,q) \in Q\times, Q$, $p\mathcal{M}_\lambda^M q$ est décidable.*

On aboutit finalement au théorème suivant :

Théorème 6.2 *L'ensemble des états accessibles* $\mathrm{RS}(R)$ *est reconnaissable, et il existe un algorithme calculant une expression rationnelle pour $\mathcal{L}_R(q)$, pour tout $q \in Q$.*

6.3 AFR à file initiale vide — Cas général

Soit R un AFR à file initiale vide associé à un système de contrôle $C = (Q, q_0, E, \delta)$. Nous considérons à présent que E_M^S est quelconque. On note que, d'après la remarque 2, pour tout état accessible $(q;w) \in \mathrm{RS}(R)$, w ne contient au plus qu'une occurrence de chaque événement à mémorisation simple. On introduit donc une nouvelle notation :

Définition 6.3 *Pour tout $F \subseteq E$, on note $F^\bullet$ l'ensemble :*

$$F^\bullet = \{w \in F^* \ / \ \forall\, e \in E_M^S, |w|_e \leq 1\}$$

On note que pour tout $F \subseteq E$, $F^\bullet$ est rationnel. La remarque 2 s'écrit :

Remarque 17 Pour tout $(q;w) \in \mathrm{RS}(R)$, pour tout $e \in E_M^S, |w|_e \leq 1$.

Par conséquent :

Proposition 6.3 $\forall\,(q;w) \in \mathrm{RS}(R), \forall\, F \subseteq E, (q;w) \in F^* \Longrightarrow (q;w) \in F^\bullet$

Avec cette nouvelle notation, on établit le lemme suivant :

Lemme 6.3.1 *Soit $\rho = (q;w) \xrightarrow{e_t} (q_1;w) \xrightarrow{\sigma} (q';w')$ un chemin de R. Supposons que :*

– *$(q;w) \in \mathrm{RS}(R)$ est un état stable ;*

– *σ est une séquence de stabilisation.*

Alors :

– *$p^\ominus(\sigma) \in Mémo(q)^\bullet$;*

– *$w' \in (Mémo(q) \setminus Alph(p^\ominus(\sigma)) \cap E_M^S)^\bullet$*

– *$q \xrightarrow{e_t} q_1 \xrightarrow{\sigma} q'$ est un chemin de C.*

Preuve Le premier point est immédiat à partir du lemme 6.1.1 et de la proposition 6.3. Le troisième point provient du lemme 6.1.1. Montrons donc le deuxième point. Comme $(q';w') \in \mathrm{RS}(R)$, d'après la proposition 6.3, il suffit de montrer que $Alph(w') \subseteq Mémo(q) \setminus Alph(p^{\ominus}(\sigma)) \cap E_M^S$. D'après le lemme 6.1.1, on a $w' \in Mémo(q)^*$, d'où $Alph(w') \subseteq Mémo(q)$. Il reste donc à montrer que $Alph(w') \cap Alph(p^{\ominus}(\sigma)) \cap E_M^S = \emptyset$. On le prouve en raisonnant par l'absurde : supposons qu'il existe $e \in Alph(w') \cap Alph(p^{\ominus}(\sigma)) \cap E_M^S$.

- Comme $e \in Alph(w')$, $|w'|_e \geq 1$;
- Comme $e \in Alph(p^{\ominus}(\sigma))$, $|p^{\ominus}(\sigma)|_e \geq 1$;
- Comme $(q;w) \in \mathrm{RS}(R)$, d'après la remarque 17, $|w|_e \leq 1$;

Or d'après le corollaire 5.4, on a $\Psi(w') + \Psi(\pi^{\ominus}(\sigma)) = \Psi(w)$, ce qui contredit l'hypothèse et termine la preuve. ♦

On considère une nouvelle relation pour prendre en compte les événements à mémorisation simple.

Définition 6.4 *La relation $\mathcal{M}_\lambda \subseteq Q \times 2^{E_M^S} \times Q$ est définie par: pour tout $(p,F,q) \in Q \times 2^{E_M^S} \times Q$, $(p,F,q) \in \mathcal{M}_\lambda$ ssi l'une des assertions suivantes est vraie :*

- *$p = q$ et $F = \emptyset$;*
- *$F \subseteq Mémo(p)$ et il existe un chemin $p \xrightarrow{e} q_1 \xrightarrow{\sigma} q$ dans C tel que $\sigma \in (Mémo(p)^M \cup F)^\bullet$.*

On constate que pour tout $q \in Q$, l'ensemble $\{(p,F) \in Q \times 2^{E_M^S} \ / \ (p,F,q) \in \mathcal{M}_\lambda\}$ est fini, puisque Q et E_M^S sont finis.

Théorème 6.3 *Pour tout $q \in Q$, $\mathcal{L}_R(q) = \bigcup_{(p,F,q) \in \mathcal{M}_\lambda} (Mémo(p) \setminus F)^\bullet$.*

Preuve Commençons par prouver $\subseteq$. Soit $q \in Q$, et $w \in \mathcal{L}_R(q)$. Deux cas se présentent :

- si $(q;w)$ est stable : alors par définition, $w \in Mémo(q)^*$. Ainsi, d'après la remarque 2, $w \in Mémo(q)^\bullet$. D'autre part $(q,\emptyset,q) \in \mathcal{M}_\lambda$.
- si $(q;w)$ est instable : comme $(q;w) \in \mathrm{RS}(R)$, il existe un chemin $\rho = (q_0,\lambda) \longrightarrow (q;w)$ dans R. Puisque (q_0,λ) est stable, ρ n'est pas un chemin de stabilisation, et comme $(q;w)$ est instable, on déduit de la proposition 6.1 qu'il existe un chemin $(q_0;\lambda) \longrightarrow (q_{st};w_{st}) \xrightarrow{e_t} (q_1;w_{st}) \xrightarrow{\sigma} (q';w')$ dans R tel que :
  - $(q_{st};w_{st})$ est un état stable. Ainsi, puisque $(q_0;\lambda) \longrightarrow (q_{st};w_{st})$, on a $(q_{st};w_{st}) \in \mathrm{RS}(R)$.
  - σ est une séquence de stabilisation.

On pose $F = Alph(p^{\ominus}(\sigma)) \cap E_M^S$. Puisque σ est une séquence de stabilisation, on en déduit, d'après le lemme 6.3.1 :

- $p^{\ominus}(\sigma) \in Mémo(q)^{\bullet}$, et donc, par définition de F, on obtient : $p^{\ominus}(\sigma) \in (Mémo(p)^M \cup F)^{\bullet}$. Par conséquent $(q_{st}, F, q) \in \mathcal{M}_\lambda$.
- $w' \in (Mémo(q) \setminus F)^{\bullet}$.

Finalement :

$$w \in \bigcup_{(p,F,q)\in\mathcal{M}_\lambda} (Mémo(p) \setminus F)^{\bullet}$$

Montrons à présent $\supseteq$. Soit $(p, F, q) \in \mathcal{M}_\lambda$, et soit $w \in (Mémo(p) \setminus F)^{\bullet}$. On note que d'après la remarque 13, $(p; \lambda) \in \mathrm{RS}(R)$. Deux cas se présentent :

- si $p = q$: alors $(q; w)$ est stable. Comme d'une part $(q; \lambda) \in \mathrm{RS}(R)$, et d'autre part pour tout $e \in E_M^S$, $|w|_e \leq 1$, il vient, d'après la remarque 11, $(q; w) \in \mathrm{RS}(R)$.
- si $p \neq q$: alors puisque $(p, F, q) \in \mathcal{M}_\lambda$, il existe dans C un chemin $p \xrightarrow{e} q_1 \xrightarrow{e_1 \ldots e_k} q$ tel que $e_1 \ldots e_k \in (Mémo(p)^M \cup F)^{\bullet}$. Or par hypothèse, $w \in (Mémo(p) \setminus F)^{\bullet}$, donc pour tout $e \in E_M^S$:
  - si $e \in \{e_1, \ldots, e_k\}$: alors $e \in F$, et donc $e \notin Alph(w)$. Par conséquent $|(e_1 \ldots e_k) \cdot w|_e = |(e_1 \ldots e_k)|_e \leq 1$;
  - si $e \notin \{e_1, \ldots, e_k\}$: alors $|(e_1 \ldots e_k) \cdot w|_e = |w|_e \leq 1$.

  et dans les deux cas $|(e_1 \ldots e_k) \cdot w|_e \leq 1$. On note que par définition $F \subseteq Mémo(p)$, et par conséquent $(e_1 \ldots e_k) \cdot w \in Mémo(p)^*$. D'autre part, $(p; \lambda) \in \mathrm{RS}(R)$ est un état stable. Par conséquent, d'après la remarque 11, $(p; (e_1 \ldots e_k) \cdot w) \in \mathrm{RS}(R)$. On aboutit donc, d'après la remarque 12, à $(q; w) \in \mathrm{RS}(R)$.

Finalement :

$$w \in \mathcal{L}_R(q) \qquad \blacklozenge$$

La proposition suivante permet de montrer que l'appartenance à $\mathcal{M}_\lambda$ est décidable.

Proposition 6.4 *Pour tout $(p, F, q) \in Q \times 2^{E_M^S} \times Q$, tel que $p \neq q$, $(p, F, q) \in \mathcal{M}_\lambda$ ssi $F \subseteq Mémo(p)$ et il existe un chemin **élémentaire** $p \xrightarrow{e} q_1 \xrightarrow{\sigma} q$ dans C tel que $\sigma \in (Mémo(p)^M \cup F)^{\bullet}$.*

Preuve Après avoir noté la constatation suivante, la preuve de la proposition 6.2 s'adapte clairement ici : $(Mémo(p)^M \cup F)^{\bullet}$ est stable par $sous - mot$:

$$\forall \sigma, \forall \widetilde{\sigma}, \{\sigma \in (Mémo(p)^M \cup F)^{\bullet} \text{ et } \widetilde{\sigma}|\sigma\} \Longrightarrow \widetilde{\sigma} \in (Mémo(p)^M \cup F)^{\bullet} \qquad \blacklozenge$$

Corollaire 6.2 *Pour tout $(p, F, q) \in Q \times 2^{E_M^S} \times Q$, $(p, F, q) \in \mathcal{M}_\lambda$ est décidable.*

On aboutit finalement au théorème suivant :

Théorème 6.4 *L'ensemble des états accessibles $\mathrm{RS}(R)$ est reconnaissable, et il existe un algorithme calculant une expression rationnelle pour $\mathcal{L}_R(q)$, pour tout $q \in Q$.*

6.4 AFR quelconque

La démarche que nous adoptons ici généralise la section précédente, mais dans le cas d'un AFR quelconque, on n'a pas la propriété fortement simplificatrice des deux sections précédentes, à savoir : pour tout état de contrôle q, $(q;\lambda)$ est accessible. Cependant, les états stables jouent cette fois-ci encore un rôle important, ainsi que les séquences sans mémorisation, comme le montrent les résultats préliminaires suivants.

6.4.1 Résultats préliminaires

Proposition 6.5 *Soit R un AFR et $\rho = (q;w) \xrightarrow{\sigma} (q';w')$ un chemin de R. Si $(q';w')$ est stable, alors il existe un chemin $(q;w) \xrightarrow{\sigma_1} (q';w_1) \xrightarrow{\sigma_2} (q';w')$ dans R tel que :*

- *$(q';w_1)$ est un état stable, et $w_1 \leq w'$;*
- *σ_1 est une séquence sans mémorisation ;*
- *σ_2 est une séquence de mémorisation.*

Preuve Supposons que $(q';w')$ est stable. D'après la proposition 5.1, il existe un chemin sans mémorisation $(q;w) \xrightarrow{\sigma_1} (q';w_1)$ dans R tel que d'une part $w_1 \leq w'$, et d'autre part $p^{\ominus}(\sigma_1) \mid p^{\ominus}(\sigma)$. On note que, d'après la remarque 10, $(q';w_1)$ est stable, et pose $z = w' - w_1$.

Montrons en raisonnant par l'absurde que $\forall\, e \in E_M^S \cap Alph(z), |w'|_e \leq 1$: si $\exists\ e \in E_M^S \cap Alph(z), |w'|_e \geq 2$, alors, d'après la remarque 9, $|\pi^{\oplus}(\rho)|_e = |\pi^{p}(\rho)|_e$. Par conséquent, d'après le lemme de conservation restreint à e, il vient : $|w|_e - |w'|_e = |p^{\ominus}(\sigma)|_e$. D'autre part, d'après le corollaire 5.2 restreint à e, $|w|_e - |w_1|_e = |p^{\ominus}(\sigma_1)|_e$. Or, comme $p^{\ominus}(\sigma_1) \mid p^{\ominus}(\sigma)$, il vient $|p^{\ominus}(\sigma_1)|_e \leq |p^{\ominus}(\sigma)|_e$. Ainsi, $|w_1|_e \geq |w'|_e$, et puisque $w_1 \leq w'$, $|w_1|_e \leq |w'|_e$. Par conséquent, $|w_1|_e = |w'|_e$, et $|z|_e = 0$, ce qui contredit l'hypothèse.

Ainsi, on obtient $\forall\, e \in E_M^S \cap Alph(z), |w_1 z|_e \leq 1$. Comme $z \in Mémo(q)^*$, on peut finalement utiliser la remarque 10, et on aboutit à l'existence d'un chemin de mémorisation $(q;w_1) \xrightarrow{\sigma_2} (q;w_1 z)$ dans R. On conclut en notant que $(q;w) \xrightarrow{\sigma_1} (q';w_1) \xrightarrow{\sigma_2} (q';w')$ est un chemin dans R. ♦

Proposition 6.6 *Soit R un AFR et $\rho = (q;w) \xrightarrow{\sigma} (q';w')$ un chemin de R. Supposons que :*

- *$(q';w')$ est instable ;*
- *ρ n'est pas un chemin de stabilisation.*

Alors il existe un chemin $(q;w) \xrightarrow{\sigma_1} (q_1;w_1) \xrightarrow{\sigma_2} (q_1;w_2) \xrightarrow{et} (q_2;w_2) \xrightarrow{\sigma_3} (q';w')$ dans R tel que :

- *$(q_1;w_2)$ est un état stable ;*
- *σ_1, σ_2 et σ_3 respectivement sont des séquences sans mémorisation, de mémorisation et de stabilisation respectivement.*

Preuve Supposons que $(q';w')$ est instable et que ρ n'est pas un chemin de stabilisation. D'après la proposition 6.1, ρ s'écrit $(q;w) \longrightarrow (q_1;w_2) \xrightarrow{e_t} (q_2;w_2) \xrightarrow{\sigma_3} (q';w')$ avec :

- $(q_1;w_2)$ est stable ;
- σ_3 est une séquence de stabilisation.

Comme $(q_1;w_2)$ est stable, on obtient d'après la proposition 6.5, qu'il existe un chemin $(q;w) \xrightarrow{\sigma_1} (q_1;w_1) \xrightarrow{\sigma_2} (q_1;w_2)$ dans R tel que :

- σ_1 est une séquence sans mémorisation ;
- σ_2 est une séquence de mémorisation. ♦

Lemme 6.4.1 *Soit R un AFR et $\rho = (q;w) \xrightarrow{e_t} (q_1;w) \xrightarrow{\sigma} (q';w')$ un chemin de R. Supposons que :*

- *$(q;w)$ est un état stable ;*
- *σ est une séquence de stabilisation.*

Alors pour tout $w_1 \leq w$ tel que $\forall e \in E_M^S \cap Alph(w - w_1), |w|_e \leq 1$, il existe un chemin sans mémorisation $\rho_1 = (q;w_1) \xrightarrow{e_t} (q_1;w_1) \xrightarrow{\sigma_1} (q';w_1')$ dans R vérifiant :

- *$p^c(\sigma_1) \in Mémo(q)^\bullet$;*
- *$Alph(p^c(\sigma_1)) \cap Alph(w_1) \cap E_M^S = \emptyset$;*
- *$w' \in w_1' \cdot (Mémo(q) \setminus (Alph(p^c(\sigma_1)) \cup Alph(w_1)) \cap E_M^S)^\bullet$.*

Preuve Supposons que :

- $(q;w)$ est un état stable ;
- σ est une séquence de stabilisation.

Soit $w_1 \leq w$ tel que $\forall e \in E_M^S \cap Alph(w - w_1), |w|_e \leq 1$. D'après la proposition 5.1, il existe un chemin sans mémorisation $(q_1;w_1) \xrightarrow{\sigma_1} (q';w_1')$ dans R tel que :

- $w_1' \leq w'$;
- $p^{\ominus}(\sigma_1) \mid p^{\ominus}(\sigma)$.
- $p^{\ominus \& c}(\sigma_1) = p^{\ominus \& c}(\sigma)$.

On pose $\rho_1 = (q;w_1) \xrightarrow{e_t} (q_1;w_1) \xrightarrow{\sigma_1} (q';w_1')$ et $z = w' - w_1'$. Clairement, ρ_1 est un chemin de R sans mémorisation. D'autre part, puisque σ est une séquence de stabilisation, il vient :

- $p^{\ominus \& c}(\sigma) = p^{\ominus}(\sigma)$;

– d'après le corollaire 5.4 : $\Psi(p^{\ominus}(\sigma)) \leq \Psi(w)$, et par ailleurs, comme $z|w'$, on a également $\Psi(z) \leq \Psi(w') \leq \Psi(w)$. Or, puisque $(q;w)$ est stable, $w \in Mémo(q)^*$. Par conséquent, $p^{\ominus\&c}(\sigma) \in Mémo(q)^*$ et $z \in Mémo(q)^*$.

Enfin, comme $p^c(\sigma_1) \,|\, p^{\ominus\&c}(\sigma_1) = p^{\ominus\&c}(\sigma)$, on obtient $p^c(\sigma_1) \in Mémo(q)^*$. Il reste donc à montrer que pour tout $e \in E_M^S$:

1. $|p^c(\sigma_1)|_e \leq 1$, et $|z|_e \leq 1$
2. $e \in Alph(p^c(\sigma_1)) \Longrightarrow e \notin Alph(z)$
3. $e \in Alph(w_1) \Longrightarrow \{e \notin Alph(p^c(\sigma_1)) \text{ et } e \notin Alph(z)\}$

On montre en fait que pour tout $e \in E_M^S$, $|p^c(\sigma_1)|_e + |z|_e = |w - w_1|_e$. Soit $e \in E_M^S$. En utilisant le lemme de transformation restreint à e, il vient :

– $|w|_e - |w'|_e = |p^{\ominus}(\sigma)|_e$;

– $|w_1|_e - |w_1'|_e = |p^{\ominus}(\sigma_1)|_e$;

Par ailleurs :

– $p^{\ominus\&c}(\sigma_1) = p^{\ominus\&c}(\sigma) = p^{\ominus}(\sigma)$;

– $|p^{\ominus\&c}(\sigma_1)|_e = |p^{\ominus}(\sigma_1)|_e + |p^c(\sigma_1)|_e$, par définition ;

On obtient donc, puisque $|z|_e = |w'|_e - |w_1'|_e$:

$$\begin{aligned} |p^c(\sigma_1)|_e &= |w|_e - |w'|_e - |w_1|_e + |w_1'|_e \\ |p^c(\sigma_1)|_e + |z|_e &= |w|_e - |w_1|_e = |w - w_1|_e \end{aligned}$$

On termine la preuve en notant que :

– par hypothèse, $|w - w_1|_e \leq 1$. En effet, si $e \in Alph(w - w_1)$, $|w|_e \leq 1$, et donc $|w - w_1|_e = |w|_e - |w_1|_e \leq 1$. Par ailleurs, si $e \notin Alph(w - w_1)$, $|w - w_1|_e = 0$. Il vient donc $|p^c(\sigma_1)|_e + |z|_e \leq 1$ ce qui prouve les assertions 1 et 2 ci-dessus.

– supposons que $e \in Alph(w_1)$. Montrons en raisonnant par l'absurde que $e \notin Alph(w - w_1)$. Si $e \in Alph(w - w_1)$, alors $|w|_e \leq 1$ et par conséquent $|w - w_1|_e = |w|_e - |w_1|_e \leq 1 - |w_1|_e$. Or comme $e \in Alph(w_1)$, $|w_1|_e \geq 1$. Ainsi $|w - w_1|_e = 0$ ce qui contredit l'hypothèse.

  Il vient finalement $|p^c(\sigma_1)|_e + |z|_e = |w - w_1|_e = 0$ ce qui prouve l'assertion 3 ci-dessus. ♦

6.4.2 Résultat principal

Soit R un AFR quelconque associé à un système de contrôle $C = (Q, q_0, E, \delta)$. On note w_0 la file initiale de R. Au vu du lemme 6.4.1, on introduit une nouvelle notation, pour exprimer de manière concise une représentation reconnaissable de $\mathrm{RS}(R)$:

Définition 6.5 *Pour tout $F \subseteq E$, et pour tout $x \in {E_M}^*$, on note $[x, F]^\bullet$ l'ensemble :*

$$[x, F]^\bullet = \{w \in F^* \;/\; \forall\, e \in E_M^S \cap Alph(w), |x\,w|_e \leq 1\}$$

Remarque 18 $\forall F \subseteq E \; \forall x \in {E_M}^*, [x, F]^\bullet = (F \setminus (Alph(x) \cap E_M^S))^\bullet$.

On note que pour tout $F \subseteq E$, et pour tout $x \in {E_M}^*$, $[x, F]^\bullet$ est rationnel. Le lemme suivant donne l'intérêt de cette notation :

Lemme 6.4.2 *Soit $(q; w)$ un état stable de R, et $w' \in {E_M}^*$. Les deux assertions suivantes sont équivalentes :*

1. *il existe une séquence de mémorisation σ telle que $(q; w) \xrightarrow{\sigma} (q; w')$;*
2. *$w' \in w \cdot [w, Mémo(q)]^\bullet$.*

Preuve La remarque 10 prouve $2 \Longrightarrow 1$. Montrons $1 \Longrightarrow 2$. Supposons qu'il existe une séquence de mémorisation σ telle que $(q; w) \xrightarrow{\sigma} (q; w')$. Alors $p^{\oplus}(\sigma) \in Mémo(q)^*$, et donc $w' \in w \cdot Mémo(q)^*$. Soit $z = w' - w$. Raisonnons par l'absurde, en supposant que $\exists \; e \in E_M^S \cap Alph(z), |wz|_e \geq 2$. Puisque σ est une séquence de mémorisation, en appliquant le corollaire 5.3 restreint à e, il vient : $|w'|_e \geq |w|_e$. D'après la remarque 9, on a $|w'|_e \leq |w|_e$, on obtient donc $|w'|_e = |w|_e$. Ainsi $|z|_e = 0$, donc $e \notin Alph(z)$, ce qui contredit l'hypothèse et termine la preuve. ♦

Définition 6.6 *La relation $\mathcal{M} \subseteq \mathrm{RS}^{\ominus \& c}(R) \times 2^{E_M^S} \times \mathrm{RS}^{\ominus \& c}(R)$ est définie par : pour tout $((p; w_p), F, (q; w_q)) \in \mathrm{RS}^{\ominus \& c}(R) \times 2^{E_M^S} \times \mathrm{RS}^{\ominus \& c}(R)$, $((p; w_p), F, (q; w_q)) \in \mathcal{M}$ ssi l'une des assertions suivantes est vraie :*

- *$(p; w_p) = (q; w_q)$ et :*
  - *$F = \emptyset$ si $(q; w_q)$ est stable ;*
  - *$F = Mémo(q)$ si $(q; w_q)$ est instable.*
- *$F \subseteq Mémo(p)$, $(p; w_p)$ est stable, $Alph(w_p) \cap F = \emptyset$ et il existe un chemin $(p; w_p) \xrightarrow{e} (q_1; w_p) \xrightarrow{\sigma} (q; w_q)$ dans R tel que :*
  - *σ est sans mémorisation ;*
  - *$p^c(\sigma) \in (Mémo(p)^M \cup F)^\bullet$.*

Le lemme suivant, qui utilise les notations introduites, est une conséquence directe du lemme 6.4.1.

Lemme 6.4.3 *Soit R un AFR et $\rho = (q; w) \xrightarrow{e_t} (q_1; w) \xrightarrow{\sigma} (q'; w')$ un chemin de R. Supposons que :*

- *$(q; w)$ est un état stable ;*
- *σ est une séquence de stabilisation.*

Soit $w_1 \leq w$ tel que :

- *$(q; w_1) \in \mathrm{RS}^{\ominus \& c}(R)$;*
- *$w \in w_1 \cdot [w_1, Mémo(q)]^\bullet$.*

Alors il existe $F \subseteq E_M^S$ *et* $w_1' \in {E_M}^*$ *vérifiant :*

- $((q; w_1), F, (q'; w_1')) \in \mathcal{M}$;
- $w \in w_1' \cdot [w_1, Mémo(q) \setminus F]^\bullet$.

Preuve Supposons que :

- $(q; w)$ est un état stable ;
- σ est une séquence de stabilisation.

Soit $w_1 \leq w$ tel que :

- $(q; w_1) \in \mathrm{RS}^{\ominus \& c}(R)$;
- $w \in w_1 \cdot [w_1, Mémo(q)]^\bullet$.

Par définition, $\forall e \in E_M^S \cap Alph(w - w_1), |w|_e \leq 1$. On obtient donc, d'après le lemme 6.4.1, l'existence d'un chemin sans mémorisation $\rho_1 = (q; w_1) \xrightarrow{e_t} (q_1; w_1) \xrightarrow{\sigma_1} (q'; w_1')$ dans R vérifiant :

- $p^c(\sigma_1) \in Mémo(q)^\bullet$;
- $Alph(p^c(\sigma_1)) \cap Alph(w_1) \cap E_M^S = \emptyset$;
- $w' \in w_1' \cdot (Mémo(q) \setminus (Alph(p^c(\sigma_1)) \cup Alph(w_1)) \cap E_M^S)^\bullet$.

On pose $F = Alph(p^c(\sigma_1)) \cap E_M^S$, et il vient :

- $(q; w_1) \xrightarrow{e_t} (q_1; w_1) \xrightarrow{\sigma_1} (q'; w_1')$ est un chemin sans mémorisation de R. Or $(q; w_1) \in \mathrm{RS}^{\ominus \& c}(R)$. Donc $(q_1; w_1) \in \mathrm{RS}^{\ominus \& c}(R)$ et $(q'; w_1') \in \mathrm{RS}^{\ominus \& c}(R)$.
- $p^c(\sigma_1) \in (Mémo(q)^M \cup F)^\bullet$;
- $(q; w_1)$ est stable, car (q, w) est stable et $w_1 \leq w$;
- $Alph(w_1) \cap F = \emptyset$.

Par conséquent $((q; w_1), F, (q', w_1')) \in \mathcal{M}$. D'autre part, comme $w' \in w_1' \cdot ((Mémo(q) \setminus F) \setminus (Alph(w_1) \cap E_M^S))^\bullet$, on obtient, d'après la remarque 18, $w' \in w_1' \cdot [w_1, (Mémo(q) \setminus F]^\bullet$. ♦

On constate que pour tout $q \in Q$, l'ensemble $\{((p; w_p), F, w_q) \in \mathrm{RS}^{\ominus \& c} \times 2^{E_M^S} \times {E_M}^* \ / \ ((p; w_p), F, (q; w_q)) \in \mathcal{M}\}$ est fini, puisque d'une part E_M^S et E_M sont finis, et d'autre part, d'après le corollaire 5.6, $\mathrm{RS}^{\ominus \& c}(R)$ est fini. Le théorème suivant prouve donc que $\mathrm{RS}(R)$ est reconnaissable.

Théorème 6.5 *Pour tout* $q \in Q$,

$$\mathcal{L}_R(q) = \bigcup_{((p;w_p),F,(q;w_q)) \in \mathcal{M}} w_q \cdot [w_p, Mémo(p) \setminus F]^\bullet$$

Preuve Commençons par prouver $\subseteq$. Soit $q \in Q$, et $w \in \mathcal{L}_R(q)$. Comme $(q;w) \in \mathrm{RS}(R)$, il existe un chemin $\rho = (q_0, w_0) \longrightarrow (q;w)$ dans R. Trois cas se présentent :

- si $(q;w)$ est stable : alors, d'après la proposition 6.5, il existe un chemin $(q_0;w_0) \xrightarrow{\sigma_1} (q;w_1) \xrightarrow{\sigma_2} (q;w)$ dans R tel que :
  - $(q;w_1)$ est un état stable, et $w_1 \leq w$;
  - σ_1 est une séquence sans mémorisation, et donc $(q;w_1) \in \mathrm{RS}^{\ominus\&c}(R)$. Par conséquent $((q;w_1), \emptyset, (q;w_1)) \in \mathcal{M}$
  - σ_2 est une séquence de mémorisation, et donc, d'après le lemme 6.4.2, $w \in w_1 \cdot [w_1, \mathit{Mémo}(q)]^\bullet$.
- si $(q;w)$ est instable et ρ est un chemin de stabilisation : alors ρ est un chemin sans mémorisation, et par conséquent $(q;w) \in \mathrm{RS}^{\ominus\&c}(R)$. On a donc :
  - $((q;w), \mathit{Mémo}(q), (q;w)) \in \mathcal{M}$;
  - $w \in w \cdot [w, \emptyset]^\bullet$.
- si $(q;w)$ est instable et ρ n'est pas un chemin de stabilisation : d'après la proposition 6.6, il existe un chemin $(q_0;w_0) \xrightarrow{\sigma_1} (q_1;w_1) \xrightarrow{\sigma_2} (q_1;w_2) \xrightarrow{e_t} (q_2;w_2) \xrightarrow{\sigma_3} (q;w)$ dans R tel que :
  - $(q_1;w_2)$ est un état stable ;
  - σ_1, σ_2 et σ_3 respectivement sont des séquences sans mémorisation, de mémorisation et de stabilisation respectivement.

  Ainsi, on a :
  - (q_1, w_2) est stable et σ_3 est une séquence de stabilisation ;
  - σ_2 est une séquence de mémorisation. Par conséquent, d'après le lemme 6.4.2, $w_1 \leq w_2$ et $w_2 \in w_1 \cdot [w_1, \mathit{Mémo}(q_1)]^\bullet$.
  - $(q;w_1) \in \mathrm{RS}^{\ominus\&c}(R)$;

  On conclut en utilisant le lemme 6.4.3.

Finalement :

$$w \in \bigcup_{((p;w_p),F,(q;w_q)) \in \mathcal{M}} w_q \cdot [w_p, \mathit{Mémo}(p) \setminus F]^\bullet$$

Montrons à présent $\supseteq$. Soit $((p;w_p), F, (q;w_q)) \in \mathcal{M}$, et soit $w \in w_q \cdot [w_p, \mathit{Mémo}(p) \setminus F]^\bullet$. On note que $(q;w_q) \in \mathrm{RS}^{\ominus\&c}(R) \subseteq \mathrm{RS}(R)$. On remarque également que $w_q \leq w$ et on note $z = w - w_q$. Trois cas se présentent :

- si $(p;w_p) = (q;w_q)$ et $F = \emptyset$: alors $(q;w_q)$ est stable. Comme $z \in [w_q, \mathit{Mémo}(q)]^\bullet$, on obtient, d'après le lemme 6.4.2, que $(q;w_q z)$ est accessible à partir de $(q;w_q)$. Par conséquent, $(q;w) \in \mathrm{RS}(R)$.

- si $(p; w_p) = (q; w_q)$ et $F \neq \emptyset$: alors $(q; w_q)$ est instable, $Mémo(q) \setminus F = \emptyset$. Par conséquent $w = w_q$ et $(q; w) \in \mathrm{RS}(R)$.

- si $(p; w_p) \neq (q; w_q)$: alors par définition $F \subseteq Mémo(p)$, $(p; w_p)$ est stable, $Alph(w_p) \cap F = \emptyset$ et il existe un chemin $(p; w_p) \xrightarrow{e} (q_1; w_p) \xrightarrow{\sigma} (q; w_q)$ dans R tel que :

  - σ est sans mémorisation ;
  - $p^c(\sigma) \in (Mémo(p)^M \cup F)^\bullet$.

  Or par hypothèse, $z \in [w_p, Mémo(p) \setminus F]^\bullet$, donc d'une part $Alph(z) \cap F = \emptyset$, et d'autre part $\forall e \in E_M^S \cap Alph(z), |w_p z|_e \leq 1$. Ainsi, pour tout $e \in E_M^S \cap Alph(p^c(\sigma) \cdot z)$:

  - si $e \in Alph(p^c(\sigma))$: comme $E_M^M \cap E_M^S = \emptyset$, il vient $e \in F$. Donc d'une part $e \notin Alph(z)$ et d'autre part $e \notin Alph(w_p)$. Par conséquent $|w_p \cdot p^c(\sigma) \cdot z|_e = |p^c(\sigma)|_e \leq 1$;
  - si $e \notin Alph(p^c(\sigma))$: alors $e \in Alph(z)$ et $|w_p \cdot p^c(\sigma) \cdot z|_e = |w_p z|_e \leq 1$.

  et dans les deux cas $|w_p \cdot p^c(\sigma) \cdot z|_e \leq 1$. Par ailleurs, $p^c(\sigma) \cdot z \in Mémo(p)^*$. Comme $(p; w_p) \in \mathrm{RS}^{\ominus \& c}(R) \subseteq \mathrm{RS}(R)$, on obtient d'après la remarque 11, $(p; w_p \cdot p^c(\sigma) \cdot z) \in \mathrm{RS}(R)$. Or $(p; w_p \cdot p^c(\sigma) \cdot z)$ est stable, donc $(q_1; w_p \cdot p^c(\sigma) \cdot z) \in \mathrm{RS}(R)$. On aboutit enfin, en utilisant la proposition 5.4, à $(q; w) \in \mathrm{RS}(R)$.

Finalement :

$$w \in \mathcal{L}_R(q) \qquad \blacklozenge$$

On sait que $\mathrm{RS}^{\ominus \& c}(R)$ est fini et calculable (cf. corollaire 5.6). La proposition suivante permet de montrer que l'appartenance à $\mathcal{M}$ est décidable et donc que $\mathcal{M}$ est fini et calculable.

Proposition 6.7 *Pour tout* $((p; w_p), F, (q; w_q)) \in \mathrm{RS}^{\ominus \& c}(R) \times 2^{E_M^S} \times \mathrm{RS}^{\ominus \& c}(R)$, *tel que* $(p; w_p) \neq (q; w_q)$, $((p; w_p), F, (q; w_q)) \in \mathcal{M}_\lambda$ *ssi* $F \subseteq Mémo(p)$, $(p; w_p)$ *est stable,* $Alph(w_p) \cap F = \emptyset$ *et il existe un chemin* ***élémentaire*** $(p; w_p) \xrightarrow{e} (q_1; w_p) \xrightarrow{\sigma} (q; w_q)$ *dans* R *tel que :*

- σ *est sans mémorisation ;*
- $p^c(\sigma) \in (Mémo(p)^M \cup F)^\bullet$.

Preuve Après avoir noté les deux constatations suivantes, la preuve de la proposition 6.2 s'adapte clairement ici :

- $(Mémo(p)^M \cup F)^\bullet$ est stable par $sous - mot$:

$$\forall \sigma, \forall \widetilde{\sigma}, \{\sigma \in (Mémo(p)^M \cup F)^\bullet \text{ et } \widetilde{\sigma}|\sigma\} \Longrightarrow \widetilde{\sigma} \in (Mémo(p)^M \cup F)^\bullet$$

- l'opérateur Ξ défini dans la preuve de la proposition 6.2 vérifie : si ρ est un chemin sans mémorisation, alors $\Xi(\rho)$ est un chemin sans mémorisation. $\blacklozenge$

Corollaire 6.3 *Pour tout* $((p;w_p),F,(q;w_q)) \in \mathrm{RS}^{\ominus\&c}(R) \times 2^{E_M^S} \times \mathrm{RS}^{\ominus\&c}(R)$, $((p;w_p),F,(q;w_q)) \in \mathcal{M}$ *est décidable.*

Corollaire 6.4 *L'ensemble* $\mathcal{M}$ *est fini et calculable.*

On aboutit finalement au théorème suivant :

Théorème 6.6 *L'ensemble des états accessibles* $\mathrm{RS}(R)$ *est reconnaissable, et il existe un algorithme calculant une expression rationnelle pour* $\mathcal{L}_R(q)$*, pour tout* $q \in Q$*.*

6.5 Conséquences de la décidabilité du RRP

On peut résumer les résultats théoriques de ce chapitre par le théorème suivant :

Théorème 6.7 *Les assertions suivantes sont vraies :*

- *Si* R *est un AFR quelconque, alors* $\mathrm{RS}(R)$ *est reconnaissable ;*
- *Le RRP est décidable pour les AFRs;*
- *Le RP, le FRSP et le QLP sont décidables pour les AFRs.*

Preuve La décidabilité du RP et du FRSP pour les AFRs provient directement des deux premiers points, dont la preuve est donnée par le théorème 6.6. Montrons que le QLP est décidable pour les AFRs. Soit donc R un AFR et $t = q \xrightarrow{\alpha} q'$ une transition du système de contrôle complété associé à R. On vérifie que les deux cas suivants se présentent :

- si $\alpha = \ominus e$ alors t est franchissable ssi $\mathcal{L}_R(q) \cap \mathit{Mémo}(q)^* e {E_M}^* \neq \emptyset$;
- sinon, t est franchissable ssi $\mathcal{L}_R(q) \cap \mathit{Mémo}(q)^* \neq \emptyset$.

On ramène ainsi la décision du QLP au test du vide sur l'intersection de deux rationnels dont on a une description effective. ♦

Enfin, pour conclure cette analyse, on note qu'on peut en fait simuler un AFR R quelconque par le graphe d'accessibilité sans mémorisation $\mathrm{RG}^{\ominus\&c}(R)$ associé, en rajoutant des transitions de mémorisation aux états stables, et en gardant la sémantique des AFRs, en ce qui concerne la mémorisation et le traitement des occurrences mémorisées. Cependant le système de transitions obtenu n'est pas un AFR, car il y a dans $\mathrm{RG}^{\ominus\&c}(R)$ des chemins de stabilisation sur lesquels l'AFR "n'interagit plus avec l'extérieur". Il faudrait alors fusionner les nœuds instables de tels chemins, pour obtenir un AFR R_λ à file initiale vide, et dont l'état initial serait le premier état stable obtenu à partir de l'état intial de R. On saurait alors retrouver la représentation reconnaissable de $\mathrm{RS}(R)$ à partir de celle de R_λ et de G, et la démarche utilisée se rapprocherait de la relation $\mathcal{M}$ définie en 6.4.

6.6 Implémentation et exemples

Nous exposons dans cette section l'implémentation que nous avons réalisée de l'algorithme calculant une représentation reconnaissable de l'ensemble des états accessibles. Après avoir indiqué l'intérêt de l'obtention automatique d'une telle représentation pour tout programme Électre, nous donnons quelques détails concernant l'algorithme implémenté, puis nous finissons par quelques exemples.

Puisque le compilateur Électre génère, à partir de tout programme Électre, un AFR à file initiale vide, nous ne considérons dans cette section que des AFRs à file initiale vide.

6.6.1 Motivations de l'implémentation

Nous avons déjà discuté en détail de l'intérêt théorique, pour la vérification, de l'obtention automatique d'une représentation reconnaissable de l'ensemble des états accessibles, qui permet de décider de nombreux problèmes de vérification. Indiquons à présent d'un point de vue plus pratique l'utilité d'une telle représentation, en tant qu'optimisation pour le compilateur Électre.

Comme nous l'avons vu en 3.3, la construction de l'AFR associé à un programme Électre procède en deux temps. Premièrement le système de contrôle est obtenu par réécritures successives du programme, et deuxièmement, on ajoute des transitions de mémorisation et de traitement des occurrences mémorisées pour obtenir le système de contrôle complété. Il est clair que toute transition de traitement immédiat d'une occurrence d'événement est franchissable dans le système de contrôle, ainsi que tout état de contrôle est accessible, par construction. Par ailleurs, comme on l'a déjà remarqué en 6, dans l'AFR à file initiale vide correspondant, toute transition de mémorisation est également franchissable, car il suffit d'accéder à l'état de contrôle correspondant avec une file vide. Néanmoins, on n'est pas assuré qu'une transition de traitement d'une occurrence mémorisée soit franchissable, et il peut s'avérer d'ailleurs en pratique que de nombreuses transitions de traitement d'une occurrence mémorisée soient infranchissables. Les deux exemples simples qui suivent permettent d'illustrer ce propos.

Exemple 6.1 Nous donnons en figure 6.1 l'AFR à initiale vide associé au programme Électre $QLP_1 = \mathbf{A\ [B \uparrow \{\{\#e\} \mid \{@f\colon C \uparrow \{\#e\}\}\}]\star.}$, et on trouvera l'automate au format tef correspondant en annexe C.1.1. Sur cet exemple, l'événement à mémorisation multiple e ne peut être mémorisé que dans l'état initial. Or dans l'état q_1, toutes les occurrences de e dans la file sont traitées, et par conséquent, e ne peut pas être dans la file lorsque l'AFR se trouve dans l'état de contrôle q_2. La transition $q_2 \xrightarrow{\ominus e} q_1$ est donc infranchissable.

Exemple 6.2 Nous donnons en figure 6.2 l'AFR à initiale vide associé au programme Électre $QLP_2 = \mathbf{A\ [B \uparrow \{e\colon C \uparrow \{e\}\}]\star.}$, et on trouvera l'automate au format tef correspondant en annexe C.2.1. Sur cet exemple, l'événement à mémorisation simple e ne peut être mémorisé que dans l'état initial. Or dans l'état q_1, toutes les occurrences de e dans la file sont traitées, puisqu'il ne peut

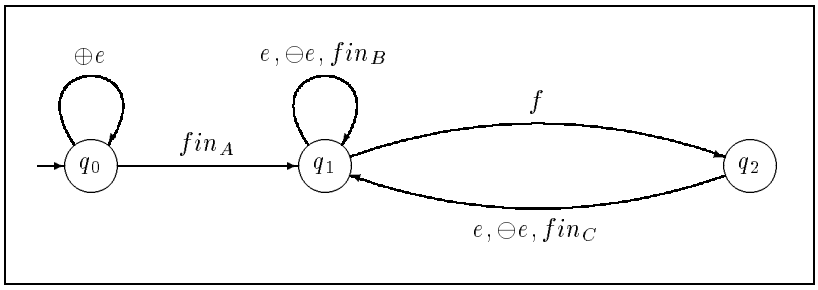


FIG. 6.1 – *L'AFR associé au programme* **A [B ↑ {{#e} | {@f: C ↑ {#e}}}]⋆.**

y en avoir au plus qu'une, et par conséquent, e ne peut pas être dans la file lorsque l'AFR se trouve dans l'état de contrôle q_2. La transition $q_2 \xrightarrow{\ominus e} q_1$ est donc infranchissable.

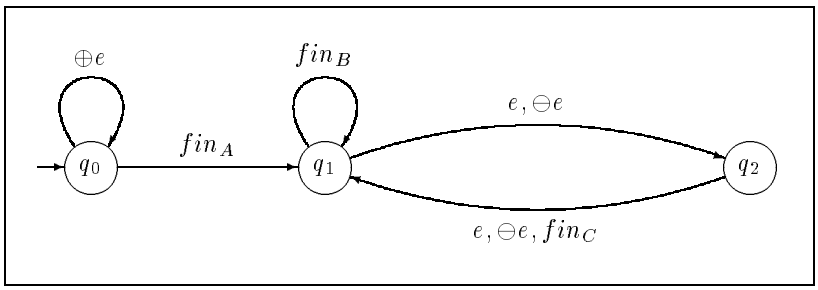


FIG. 6.2 – *L'AFR associé au programme* **A [B ↑ {e: C ↑ {e}}]⋆.**

Il faut noter que pour des programmes Électre de quelques dizaines de lignes, l'automate au format tef peut contenir plusieurs milliers d'états et plusieurs dizaines de milliers de transitions. On comprend donc aisément l'intérêt de pouvoir supprimer des transitions dans l'automate, celles dont on sait qu'elles ne pourront jamais être franchies, et qu'on peut détecter grâce à la décidabilité du QLP (cf. section 6.5). Il faudrait cependant effectuer des essais en pratique pour estimer le gain de cette optimisation.

6.6.2 Présentation générale de l'implémentation

L'algorithme implémenté

L'algorithme que nous avons implémenté n'est pas exactement celui qui est sous-jacent à la section 6.2, mais une version légèrement optimisée, où on calcule simultanément pour tout état de contrôle q, par itérations croissantes successives, une représentation reconnaissable du langage de la file $\mathcal{L}_R(q)$ en l'état de contrôle q.

Soit R un AFR à file initiale vide, et $\overline{C} = (Q, q, E \cup \{\oplus, \ominus\} \times E, \bar{\delta})$ le système de contrôle complété associé à R. Pour tout couple $(q, q') \in Q \times Q$, on note $I(q, q')$ l'ensemble $I(q, q') = \{e \in E \ / \ (q, e, q') \in \overline{\delta}\}$. D'autre part, à tout

état de contrôle $q \in Q$, on associe un attribut $\mathcal{A}_R(q)$ qui est modifié tout au long de l'algorithme. Cet attribut est un ensemble de sous-ensembles de E_M. Lorque l'algorithme termine, $\mathcal{A}_R(q)$ contient une représentation reconnaissable de $\mathcal{L}_R(q)$, sous la forme :

$$\forall\, q \in Q, \mathcal{L}_R(q) = \bigcup_{F \in \mathcal{A}_R(q)} F^\bullet \tag{6.1}$$

Comme tous les états de contrôle sont accessibles avec la file vide, on initialise les attributs par la procédure suivante :

INITILISATION $(\overline{C})$

1. **pour** chaque état $q \in Q$
2. **faire** $\mathcal{A}_R(q) = \{Mémo(q)\}$

Ensuite, on propage pour tout état de contrôle $q \in Q$ l'attribut initial $\mathcal{A}_R(q) = \{Mémo(q)\}$ vers les successeurs de q. Ceci correspond dans la définition [4] de $\mathcal{M}_\lambda$ au franchissement de la transition $p \xrightarrow{e} q_1$.

PROPAGATION $(\overline{C})$

1. **pour** chaque état $q \in Q$
2. **faire pour** chaque état $q' \in Q$ tel que $q \longrightarrow q'$
3. **faire** $\mathcal{A}_R(q') = \mathcal{A}_R(q') \cup \{Mémo(q)\}$

Enfin, chaque itération de la procédure ITÉRATION ci-dessous propage, pour tout état de contrôle $q \in Q$, chaque élément de $\mathcal{A}_R(q)$ vers les successeurs de q, en repectant la démarche suivante, qui correspond intuitivement à la définition de $\mathcal{M}_\lambda$. Soit q' un successeur de q. Pour tout $F \in \mathcal{A}_R(q)$:

- si on ne peut pas propager F, i.e. $F \cap I(q, q') = \emptyset$, alors on ne modifie pas $\mathcal{A}_R(q')$;
- si on peut propager F par un événement à mémorisation multiple, i.e. $F^M \cap I(q, q') \neq \emptyset$, alors on ajoute F à l'ensemble $\mathcal{A}_R(q')$;
- sinon : alors on ne peut propager F que par un événement à mémorisation simple, i.e. $F^S \cap I(q, q') = F \cap I(q, q') \neq \emptyset$, et on ajoute, pour tout $e \in F \cap I(q, q')$, $F \setminus \{e\}$ à l'ensemble $\mathcal{A}_R(q')$.

ITÉRATION $(\overline{C})$

1. **pour** chaque état $q \in Q$
2. **faire pour** chaque état $q' \in Q$ tel que $q \longrightarrow q'$
3. **faire pour** chaque ensemble $F \in \mathcal{A}_R(q)$

4. on se réfère ici à la définition 6.4.

4. **faire si** $F \cap I(q,q') \neq \emptyset$

5. **alors si** $F^M \cap I(q,q') \neq \emptyset$

6. **alors** $\mathcal{A}_R(q') = \mathcal{A}_R(q') \cup \{F\}$

7. **sinon** $\mathcal{A}_R(q') = \mathcal{A}_R(q') \cup \{F \setminus \{e\} \ / \ e \in F \cap I(q,q')\}$

Considérons une exécution de la procédure ITÉRATION. Si on note $\mathcal{A}_R^{old}(q)$ l'ancien attribut et $\mathcal{A}_R^{new}(q)$ le nouvel attribut de l'état de contrôle q, alors on remarque que $\mathcal{A}_R^{old}(q) \subseteq \mathcal{A}_R^{new}(q)$. Par conséquent, puisque le nombre d'ensembles $\mathcal{A}_R(q)$ possibles est fini, une succession d'appels à la procédure ITÉRATION finit par aboutir à un point fixe. Ainsi, la procédure REPRÉSENTATION ci-dessous termine toujours, et d'autre part, on vérifie que lorsqu'elle a terminé, l'équation (4.2) est satisfaite.

REPRÉSENTATION $(\overline{C})$

1. INITILISATION $(\overline{C})$
2. PROPAGATION $(\overline{C})$
3. **tant que** le point fixe n'est pas atteint
4. **faire** ITÉRATION $(\overline{C})$

Finalement, pour donner un ordre de grandeur de la complexité de l'algorithme, on note qu'il y a, dans la procédure REPRÉSENTATION, au plus $|Q|$ appels à la procédure ITÉRATION, car :

- on sait qu'on peut imposer de manière équivalente, dans la définition de $\mathcal{M}_\lambda$, au chemin $q_1 \xrightarrow{\sigma} q$ d'être élémentaire ;
- un appel à la procédure ITÉRATION propage les attributs vers chaque successeur, donc, intuitivement, le point fixe sera atteint lorsque tous les chemins élémentaires auront été parcourus ;
- un chemin élémentaire a pour longueur au plus $|Q|$.

Par ailleurs, si on note $\Delta = \{(q,q') \ / \ q \longrightarrow q'\}$:

- un appel à la procédure ITÉRATION se fait en $O(|\Delta|)$;
- Les procédures INITILISATION et PROPAGATION sont en $O(|\Delta|)$.

Donc la procédure REPRÉSENTATION calcule une représentation reconnaissable de $\mathrm{RS}(R)$ en $O(K(|E_M|) \cdot |Q| \cdot |\Delta|)$, où $K(|E_M|)$ dénote la complexité des opérations d'union réalisées sur les attributs.

Architecture globale du programme `tef_esr`

Le programme `tef_esr` qui calcule une représentation reconnaissable d'un AFR, à partir d'un automate au format tef, a été implémenté en Objective Caml. Nous avons choisi Objective Caml, car ce langage permet d'exprimer aisément les opérations sur les ensembles qui sont utilisées dans notre algorithme. Cependant, les performances du programme `tef_esr` pourraient très certainement être améliorées par l'utilisation de structures de données plus adaptées aux manipulations d'ensembles que les arbres binaires de recherche utilisés pour coder le module Set de Objective Caml.

Notre programme comprend deux parties distinctes, reliées par une même structure de données : le système de contrôle Électre.

- la première partie s'occupe de reconstruire le système de contrôle Électre à partir de l'automate au format tef. C'est cette partie qui posait le plus de problèmes, et qu'il reste encore éventuellement à affiner, car, comme nous l'avons déjà évoqué en 3.3.3, la sémantique de l'automate au format tef que nous avons considérée n'est pas complète.

- la deuxième partie calcule la représentation symbolique de l'AFR à file initiale vide asocié à un système de contrôle donné, d'après l'algorithme donné précédemment.

Ainsi, quitte à programmer une interface s'occupant de la première partie, le programme `tef_esr` est adaptable à tout langage réactif dont la compilation produit un AFR.

Signalons enfin que le programme `tef_esr` produit un fichier .esr à partir d'un fichier .tef.

6.6.3 Exemples

Les fichiers .esr des programmes QLP_1 et QLP_2 (cf. exemples 6.1 et 6.2) sont donnés en annexe C.1.2 et C.2.2.

Pour l'exemple des lecteurs-écrivains que nous avons considéré tout au long de ce mémoire[5], le fichier .esr correspondant est donné ci-après :

```
SymbolicRepresentation :

 St0 : { { e1 e2 } }
 St1 : { { e1 e2 } }
 St2 : { { e1 e2 } }
 St3 : { { e1 e2 } }
 St4 : { { e1 e2 } }
 St5 : { { e1 e2 } }
 St6 : { { e1 e2 } }
 St7 : { { e1 e2 } }
 St8 : { { e1 e2 } }
```

On rappelle que les événements l_1, l_2 sont fugaces, et que les événements e_1, e_2 sont à mémorisation simple dans cet exemple.

5. on pourra se reporter à la section 3.1.6, aux exemples 3.7 et 3.8, ou encore à l'annexe B.

Enfin, pour terminer cette section, nous donnons un exemple plus complet, faisant intervenir des priorités entre deux lecteurs. On considère à présent le programme Électre suivant :

$$IMPL = [[1 \ / \ \{\{l_1 : !\mathbf{LIRE}_1\} \parallel \{\#l_2 : \mathbf{LIRE}_2\}] \uparrow \{e : \mathbf{ECRIRE}\}]\star.$$

Le fichier IMPL.tef est donné en annexe C.3, et le fichier IMPL.esr produit par le programme `tef_esr` est donné ci-après :

```
SymbolicRepresentation :

 St0 : { { l1 l2 } }
 St1 : { { e l2 } { l1 l2 } }
 St2 : { { l2 } }
 St3 : { { l1 l2 } }
 St4 : { { l2 } }
 St5 : { { e l2 } }
 St6 : { { l1 l2 } }
 St7 : { { l2 } }
 St8 : { { l2 } }
```

Chapitre 7

Vérification comportementale et logiques temporelles

Nous présentons dans ce chapitre la vérification comportementale des AFRs, en montrant tout d'abord la décidabilité du RSP pour les AFRs. Nous utilisons ensuite les résultats établis afin d'étendre d'une part la décidabilité du RSP au model-checking d'un fragment de LTL, et de montrer d'autre part que le model-checking d'un fragment de CTL est également décidable. Nous finissons par la vérification de propriétés de famine pour Électre.

Dans tout ce chapitre, les chemins et les exécutions que nous considérons sont finis ou infinis. On remarque que dans un AFR ayant un événement mémorisable, il n'existe pas d'état de blocage, et par conséquent, tout état est origine d'un chemin infini. Nous ne considérons donc dans ce chapitre que des AFRs dont l'ensemble des événements E vérifie : $E_{_M} \neq \emptyset$. On rappelle que si R est un AFR d'ensemble d'événements E tel que $E_{_M} = \emptyset$, alors R est bissimilaire au système de contrôle associé à R (cf. remarque 15), et ainsi, la vérification comportementale est décidable.

Enfin, précisons que dans la suite, lorsqu'on dit qu'un chemin s'écrit $\rho_1 \cdot \rho_2$, alors implicitement, le chemin ρ_1 est fini et le chemin ρ_2 est infini.

7.1 Décidabilité du problème RSP

Avant d'aborder le RSP tel qu'il est défini en 4.2, notons que le problème suivant est décidable : étant donné un AFR R, et un état $(q;w)$ de R, existe-t-il une exécution de R visitant infiniment souvent $(q;w)$? Il suffit en effet de déterminer si $(q;w)$ est accessible dans R puis de déterminer si $(q;w)$ est accessible à partir d'un de ses successeurs [1].

Notons également que la représentation reconnaissable de l'ensemble des états accessibles d'un AFR permet aussi de décider le problème de l'accessibilité d'un état de contrôle donné.

Nous nous intéressons à présent au problème RSP. Comme pour la présentation du calcul d'une représentation reconnaissable des états accessibles d'un

1. on rappelle qu'on peut choisir non seulement une file initiale quelconque, mais aussi un état de contrôle initial quelconque (cf. chapitre 4).

AFR, nous établissons d'abord les résultats dans le cas des AFRs à file initiale vide, puis nous étendons ces résultats aux AFRs quelconques.

7.1.1 AFRs à file initiale vide

Si R est un AFR à file initiale vide associé à un système de contrôle C, on utilise un système de transitions intermédiaire $\widetilde{C}$, compris entre C et le système de contrôle complété $\overline{C}$ associé à R, dans lequel lorsqu'on franchit une boucle de mémorisation sur l'état $q \in Q$, alors on ne peut ensuite que boucler par des mémorisations sur l'état q. Plus formellement :

Définition 7.1 *Soit R un AFR à file initiale vide. Soit $C = (Q, q_0, E, \delta)$ le système de contrôle et $\overline{C} = (Q, q, E \cup \{\oplus, \ominus\} \times E, \bar{\delta})$ le système de contrôle complété associés à R. On définit le* système de contrôle intermédiaire associé à R, *noté $\widetilde{C}$, par $\widetilde{C} = (Q \cup \widetilde{Q}, q_0, E \cup \{\oplus, \ominus\} \times E, \delta \cup \widetilde{\delta})$ avec :*

- $\widetilde{Q} = \{\widetilde{q} \ / \ q \in Q\}$;
- $\widetilde{\delta} = \{(q, (\oplus, e), \widetilde{q}) \ / \ (q, (\oplus, e), q) \in \overline{\delta}\} \cup \{(\widetilde{q}, (\oplus, e), \widetilde{q}) \ / \ (q, (\oplus, e), q) \in \overline{\delta}\}$.

Remarque 19 La constatation suivante permet de s'assurer qu'on peut parler dans toute la suite de chemins infinis :

- tout état d'un système de contrôle complété est origine d'un chemin infini ;
- tout état d'un système de contrôle intermédiaire est origine d'un chemin infini.

Remarque 20 Soit R un AFR à file initiale vide. Soit $\overline{C}$ le système de contrôle complété et $\widetilde{C}$ le système de contrôle intermédiaire associés à R. Si on identifie les états q et $\widetilde{q}$ dans $\widetilde{C}$, alors il est clair que les chemins de $\widetilde{C}$ sont les chemins de $\overline{C}$ qui, s'ils contiennent une transition de mémorisation, alors bouclent ensuite par des mémorisations sur le même état.

Dans la suite, pour simplifier la présentation, **on identifie les états q et $\widetilde{q}$ d'un système de contrôle intermédiaire**. La remarque 20 s'écrit alors :

Remarque 21 Soit R un AFR à file initiale vide. Soit C le système de contrôle, $\overline{C}$ le système de contrôle complété et $\widetilde{C}$ le système de contrôle intermédiaire associés à R. D'une part, tout chemin de $\widetilde{C}$ est un chemin de $\overline{C}$. D'autre part, si ρ est un chemin de $\overline{C}$, alors ρ est un chemin de $\widetilde{C}$ ssi on se trouve dans l'un des deux cas suivants :

- ρ est un chemin de C ;
- ρ s'écrit $\rho_1 \rho_m$ avec :
  - ρ_1 est un chemin de C ;
  - ρ_m est un chemin de mémorisation de $\overline{C}$.

Si R est un AFR à file initiale vide, alors il existe des liens étroits entre R, le système de contrôle complété $\overline{C}$ associé à R et le système de contrôle intermédiaire $\widetilde{C}$ associé à R, comme le montre le lemme suivant :

Lemme 7.1.1 *Soit R un AFR à file initiale vide. Soit C le système de contrôle, $\overline{C}$ le système de contrôle complété et $\widetilde{C}$ le système de contrôle intermédiaire associés à R.*

1. *Si $\rho = (q_0;\lambda) \xrightarrow{\alpha_0} (q_1;w_1)\ldots(q_m;w_m) \xrightarrow{\alpha_m} (q_{m+1};w_{m+1})\ldots$ est une exécution de R, alors $q_0 \xrightarrow{\alpha_0} q_1\ldots q_m \xrightarrow{\alpha_m} q_{m+1}\ldots$ est une exécution de $\overline{C}$.*
2. *Si $\rho = q_0 \xrightarrow{\alpha_0} q_1\ldots q_m \xrightarrow{\alpha_m} q_{m+1}\ldots$ est une exécution de $\widetilde{C}$, alors il existe une exécution $(q_0;\lambda) \xrightarrow{\alpha_0} (q_1;w_1)\ldots(q_m;w_m) \xrightarrow{\alpha_m} (q_{m+1};w_{m+1})\ldots$ dans R.*

Preuve Le premier point est trivial, d'après la définition de la construction d'un AFR à partir d'un système de contrôle complété. Pour le deuxième point, on suppose que $\rho = q_0 \xrightarrow{\alpha_0} q_1\ldots q_m \xrightarrow{\alpha_m} q_{m+1}\ldots$ est une exécution de $\widetilde{C}$. D'après la remarque 21, deux cas se présentent :

- si ρ est un chemin de C : alors on conclut avec la remarque 13 ;
- si ρ s'écrit $\rho_1\rho_m$ avec :
  - ρ_1 est un chemin de C ;
  - ρ_m est un chemin de mémorisation de $\overline{C}$.

  Alors on peut écrire ρ_m sous la forme $\rho_m = q \xrightarrow{\oplus e_1} q \xrightarrow{\oplus e_2} q \xrightarrow{\oplus e_3} \ldots$. Par définition de $\overline{C}$, pour tout $i \in \mathbb{N}$, $e_i \in \mathit{Mémo}(q)$. Par conséquent, puisque $(q;\lambda)$ est stable, et en utilisant la remarque 10, $\rho_2 = (q;\lambda) \xrightarrow{\oplus e_1} (q;w_1) \xrightarrow{\oplus e_2} (q;w_2) \xrightarrow{\oplus e_3} (q;w_3)\ldots$ est un chemin de R. Enfin, si on note $\rho_{1\lambda}$ le chemin obtenu à partir de ρ_1 en gardant une file vide, comme indiqué en remarque 13, on conclut en posant $\rho = \rho_{1\lambda}\cdot\rho_2$ qui est un chemin de R. ♦

On introduit à présent un opérateur sur les chemins du système de contrôle complété, afin d'éliminer les boucles de mémorisations intermédiaires, et d'obtenir ainsi un chemin du système de contrôle intermédiaire.

Définition 7.2 *Soit R un AFR à file initiale vide. Soit $\overline{C}$ le système de contrôle complété associé à R. Pour tout chemin fini ou infini ρ de $\overline{C}$, on note $\Upsilon(\rho)$ le chemin de $\overline{C}$ défini par :*

- *si ρ est un chemin infini de mémorisation, alors $\Upsilon(\rho) = \rho$;*
- *sinon :*
  - *si ρ s'écrit $q \xrightarrow{\oplus e} q\cdot\rho'$, alors $\Upsilon(\rho) = \Upsilon(\rho')$;*
  - *si ρ s'écrit $q \xrightarrow{e} q'\cdot\rho'$, alors $\Upsilon(\rho) = q \xrightarrow{e} q'\cdot\Upsilon(\rho')$;*

- *si* ρ *s'écrit* $q \xrightarrow{\ominus e} q' \cdot \rho'$, *alors* $\Upsilon(\rho) = q \xrightarrow{\ominus e} q' \cdot \Upsilon(\rho')$.
- *sinon, alors* ρ *est de longueur nulle, et* $\Upsilon(\rho) = \rho$.

Remarque 22 Soit R un AFR à file initiale vide, $\overline{C}$ le système de contrôle complété associé à R, et ρ un chemin de $\overline{C}$. On a :

- $\Upsilon(\rho)$ est un chemin fini ssi ρ est un chemin fini. En effet, si ρ est fini, alors il est clair que $\Upsilon(\rho)$ est fini. Réciproquement, si ρ est infini, alors soit il se termine par une séquence de mémorisations, auquel cas $\Upsilon(\rho)$ aussi, soit il contient une infinité de transitions qui ne sont pas des transitions de mémorisation, auquel cas $\Upsilon(\rho)$ aussi.

- $\Upsilon(\rho)$ est un chemin de longueur nulle ssi ρ est un chemin fini de mémorisation. En effet, on note que si $\Upsilon(\rho)$ est un chemin de longueur nulle, alors d'après le point précédent, ρ est fini. On démontre ensuite le résultat par récurrence sur la taille de ρ.

- si ρ est un chemin sans mémorisation, alors $\Upsilon(\rho) = \rho$. On démontre le résultat en montrant par récurrence sur i que $\Upsilon(\rho)_{|_i} = \rho_{|_i}$.

- si ρ s'écrit $\rho_1 \cdot \rho_2$, alors $\Upsilon(\rho) = \Upsilon(\rho_1) \cdot \Upsilon(\rho_2)$. On démontre le résultat par récurrence sur la taille de ρ_1, qui, rappelons-le, est implicitement fini.

- $\Upsilon(\rho)$ a même état d'origine que ρ. Le résultat se démontre en notant que si ρ débute par une transition de mémorisation, alors ρ est soit une séquence infinie de mémorisations, auquel cas $\Upsilon(\rho) = \rho$, soit ρ s'écrit $\rho_1 \cdot q \xrightarrow{op} q' \cdot \rho_2$, avec :

  - ρ_1 est un chemin fini de mémorisations d'état origine q ;
  - $op \in (E \cup \{\ominus e \ / \ e \in E\})$. Par conséquent, d'après les points précédents, $\Upsilon(\rho) = q \xrightarrow{op} q' \cdot \Upsilon(\rho_2)$.

Lemme 7.1.2 *Soit* R *un AFR à file initiale vide. Soit* C *le système de contrôle,* $\overline{C}$ *le système de contrôle complété et* $\widetilde{C}$ *le système de contrôle intermédiaire associés à* R. *Pour tout chemin* ρ *de* $\overline{C}$, $\Upsilon(\rho)$ *est un chemin de* $\widetilde{C}$.

Preuve Soit ρ un chemin de $\overline{C}$. On montre le résultat en raisonnant par l'absurde : si $\Upsilon(\rho)$ n'est pas un chemin de $\widetilde{C}$, alors, d'après la remarque 21, $\Upsilon(\rho)$ n'est pas un chemin de C, et donc contient une transition de mémorisation. Par conséquent, il existe un chemin ρ_1 dans C tel que $\Upsilon(\rho)$ s'écrive $\rho_1 \cdot q_1 \xrightarrow{\oplus e} q_1 \cdot \rho_2$. D'après la remarque 21, ρ_2 n'est pas un chemin de mémorisation, et donc ρ_2 s'écrit $\rho_3 \cdot q_1 \xrightarrow{op} q_2 \cdot \rho_4$, avec :

- $op \in (E \cup \{\ominus e \ / \ e \in E\})$;

- ρ_3 est un chemin fini de mémorisation sur l'état q_1.

On utilise à présent la remarque 22. Comme ρ_1 est un chemin de C, $\Upsilon(\rho_1) = \rho_1$. Par conséquent, $\Upsilon(\rho) = \Upsilon(\rho_1) \cdot \Upsilon(\rho_3) \cdot \Upsilon(q_1 \xrightarrow{op} q_2) \cdot \Upsilon(\rho_4) = \rho_1 \cdot q_1 \xrightarrow{op} q_2 \cdot \Upsilon(\rho_4)$.

Or $\Upsilon(\rho) = \rho_1 \cdot q_1 \xrightarrow{\oplus e} q_1 \cdot \rho_2$, il vient donc $op = \oplus e$, ce qui contredit l'hypothèse et termine la preuve. ♦

Lemme 7.1.3 *Soit R un AFR à file initiale vide. Soit C le système de contrôle, $\overline{C}$ le système de contrôle complété et $\widetilde{C}$ le système de contrôle intermédiaire associés à R. Pour tout chemin ρ de $\overline{C}$, et pour tout état q de $\overline{C}$, ρ visite infiniment souvent l'état q ssi $\Upsilon(\rho)$ visite infiniment souvent l'état q.*

Preuve Soit ρ un chemin de $\overline{C}$, et q un état de $\overline{C}$. Si $\Upsilon(\rho)$ visite infiniment souvent l'état q, alors puisque $\Upsilon(\rho)$ est construit à partir de ρ en enlevant des transitions de mémorisation de ρ, ρ visite infiniment souvent l'état q. Pour montrer la réciproque, on suppose que ρ visite infiniment souvent l'état q. On se trouve dans l'un des deux cas suivants :

- si ρ s'écrit $\rho_1\rho_m$ tel que ρ_m est un chemin de mémorisation ou un chemin sans mémorisation, alors, d'après la remarque 22, $\Upsilon(\rho) = \Upsilon(\rho_1)\rho_m$. Puisque d'une part ρ visite infiniment souvent l'état q, et d'autre part ρ_1 est fini, ρ_m visite infiniment souvent l'état q. Par conséquent, $\Upsilon(\rho)$ visite infiniment souvent l'état q.

- sinon, ρ s'écrit $\rho_1 \cdot \rho_{1_m} \cdot \rho_2 \cdot \rho_{2_m} \cdot \rho_3 \cdot \rho_{3_m} \ldots$ avec pour tout $i \in \mathbb{N}^*$:

  - ρ_i est un chemin fini de longueur non nulle d'état arrivée q, ne contenant aucune aucune transition de mémorisation sur l'état q ;
  - ρ_{i_m} est un chemin fini de longueur non nulle de mémorisation sur l'état q.

  Ainsi, en procédant par récurrence d'après la remarque 22, on obtient que $\Upsilon(\rho) = \Upsilon(\rho_1) \cdot \Upsilon(\rho_2) \cdot \Upsilon(\rho_3) \ldots$. On note, d'après la même remarque, que :

  - pour tout $i \in \mathbb{N}^*$, $\Upsilon(\rho_i)$ est de longueur non nulle, car sinon : ρ_i serait un chemin de mémorisation. Puisque ρ_i a q pour état d'arrivée, ρ_i serait alors un chemin de longueur non nulle de mémorisation sur l'état q, or ρ_i est supposé ne contenir aucune transition de mémorisation sur l'état q.
  - pour tout $i \in \mathbb{N}^*$, si $i \geq 2$, alors $\Upsilon(\rho_i)$ a q pour état d'origine.

  Finalement, $\Upsilon(\rho)$ visite infiniment souvent l'état q. ♦

Proposition 7.1 *Soit R un AFR à file initiale vide. Soit $\overline{C}$ le système de contrôle complété et $\widetilde{C}$ le système de contrôle intermédiaire associés à R. Soit q un état de contrôle de R. Les trois assertions suivantes sont équivalentes:*

1. *il existe une exécution ρ de $\widetilde{C}$ visitant infiniment souvent l'état q ;*
2. *il existe une exécution ρ de R visitant infiniment souvent l'état de contrôle q ;*
3. *il existe une exécution ρ de $\overline{C}$ visitant infiniment souvent l'état q.*

Preuve Les implications $1 \Longrightarrow 2$ et $2 \Longrightarrow 3$ sont immédiates en utilisant le lemme 7.1.1. L'implication $3 \Longrightarrow 1$ provient des lemmes 7.1.2 et 7.1.3. ♦

Enfin, le système de contrôle complété associé à un AFR à file initiale vide est un système de transitions fini, pour lequel le RSP est trivialement décidable. On aboutit donc au théorème :

Théorème 7.1 *Le RSP est décidable pour les AFRs à file initiale vide.*

7.1.2 AFRs quelconques

Nous étendons à présent le raisonnement effectué pour les AFRs à file initiale vide aux AFRs quelconques. Le rôle joué par le système de contrôle dans le cas précédent est à présent joué par le graphe d'accessibilité sans mémorisation.

Remarque 23 Soit R un AFR associé à un système de contrôle $C = (Q, q_0, E, \delta)$. On note w_0 la file initiale de R. Le graphe d'accessibilité sans mémorisation de R, $\mathrm{RG}^{\ominus\&c}(R)$, peut être considéré comme un système de transitions fini $(Q_G, q_{0_G}, E \cup \{\oplus, \ominus\} \times E, \delta_G)$, où Q_G est l'ensemble des nœuds de $\mathrm{RG}^{\ominus\&c}(R)$. D'autre part, on identifie sans ambiguïté les nœuds de $\mathrm{RG}^{\ominus\&c}(R)$ avec leur étiquette, i.e. $Q_G \equiv \mathrm{RS}^{\ominus\&c}(R)$.

On note que si R est un AFR à file initiale vide, alors le graphe d'accessibilité sans mémorisation de R et le système de contrôle associé à R sont bissimilaires. On définit à présent le graphe d'accessibilité sans mémorisation complété et le graphe d'accessibilité sans mémorisation intermédiaire associés à un AFR.

Définition 7.3 *Soit R un AFR, et soit $\mathrm{RG}^{\ominus\&c}(R) = (Q_G, q_{0_G}, E \cup \{\oplus, \ominus\} \times E, \delta_G)$ le graphe d'accessibilité sans mémorisation de R. On note Q_R l'ensemble des états de R, et δ_R l'ensemble des transitions de R. On appelle* graphe d'accessibilité sans mémorisation complété associé à R, *noté $\overline{\mathrm{RG}}^{\ominus\&c}(R)$, le système de transitions fini $\overline{\mathrm{RG}}^{\ominus\&c}(R) = (Q_G, q_{0_G}, E \cup \{\oplus, \ominus\} \times E, \overline{\delta_G})$, dont l'ensemble de transitions $\overline{\delta_G}$ est défini par :*

$$\overline{\delta_G} = \delta_G \cup \{(q_G, (\oplus, e), q_G) \ / \ q_G \in Q_G \text{ et } \exists\, q' \in Q_R, (q_G, (\oplus, e), q') \in \delta_R\}$$

On note que $\overline{\mathrm{RG}}^{\ominus\&c}(R)$ est obtenu à partir de $\mathrm{RG}^{\ominus\&c}(R)$ en rajoutant des transitions de mémorisation $q_G \xrightarrow{\oplus e} q_G$ sur les états $q_G \in Q_G$ tels que :

- q_G est stable ;
- $q_G \xrightarrow{\oplus e}$ dans R.

On obtient donc la remarque suivante :

Remarque 24 Soit R un AFR, et $(q; w_1) \in \mathrm{RS}^{\ominus\&c}(R)$. S'il existe dans R un chemin $\rho = (q; w) \xrightarrow{\oplus e_1 \ldots \oplus e_k}$ tel que $w_1 \leq w$, alors pour tout $i \in \mathbb{N}$, $(q; w) \xrightarrow{\oplus e_i}$, et donc $(q; w_1) \xrightarrow{\oplus e_i}$ dans R. Ainsi, par définition de $\overline{\mathrm{RG}}^{\ominus\&c}(R)$, $(q; w_1) \xrightarrow{\oplus e_1}$ $(q; w_1) \ldots (q; w_1) \xrightarrow{\oplus e_k} (q; w_1)$ est un chemin de $\overline{\mathrm{RG}}^{\ominus\&c}(R)$.

Définition 7.4 *Soit R un AFR, et soit $\mathrm{RG}^{\ominus\&c}(R) = (Q_G, q_{0_G}, E \cup \{\oplus, \ominus\} \times E, \delta_G)$ le graphe d'accessibilité de R. On définit le* graphe d'accessibilité sans mémorisation intermédiaire associé à R, *noté $\widetilde{\mathrm{RG}}^{\ominus\&c}(R)$, par $\widetilde{\mathrm{RG}}^{\ominus\&c}(R) = (Q_G \cup \widetilde{Q_G}, q_{0_G}, E \cup \{\oplus, \ominus\} \times E, \delta_G \cup \widetilde{\delta_G})$ avec :*

- $\widetilde{Q_G} = \{\widetilde{q} \ / \ q_G \in Q_G\}$;
- $\widetilde{\delta_G} = \{(q_G, (\oplus, e), \widetilde{q_G}) \ / \ (q_G, (\oplus, e), q_G) \in \overline{\delta_G}\} \cup \{(\widetilde{q_G}, (\oplus, e), \widetilde{q_G}) \ / \ (q_G, (\oplus, e), q_G) \in \overline{\delta_G}\}$.

Remarque 25 La constatation suivante permet de s'assurer qu'on peut parler dans toute la suite de chemins infinis :

- tout état d'un graphe d'accessibilité sans mémorisation complété associé à un AFR est origine d'un chemin infini ;
- tout état d'un graphe d'accessibilité sans mémorisation intermédiaire associé à un AFR est origine d'un chemin infini.

Dans la suite, pour simplifier la présentation, **on identifie les états q_G et $\widetilde{q_G}$ d'un graphe d'accessibilité sans mémorisation intermédiaire**. La proposition suivante, qui généralise la remarque 21 et le lemme 7.1.1 du cas précédent, montrent les liens étroits qui existent entre R, $\overline{\mathrm{RG}}^{\ominus\&c}(R)$ et $\widetilde{\mathrm{RG}}^{\ominus\&c}(R)$.

Proposition 7.2 *Soit R un AFR d'état initial $(q_0; w_0)$. Les assertions suivantes sont vraies :*

1. *tout chemin de $\widetilde{\mathrm{RG}}^{\ominus\&c}(R)$ est un chemin de $\overline{\mathrm{RG}}^{\ominus\&c}(R)$;*
2. *si ρ est un chemin de $\overline{\mathrm{RG}}^{\ominus\&c}(R)$, alors ρ est un chemin de $\widetilde{\mathrm{RG}}^{\ominus\&c}(R)$ ssi on se trouve dans l'un des deux cas suivants :*
   - *ρ est un chemin de $\mathrm{RG}^{\ominus\&c}(R)$;*
   - *ρ s'écrit $\rho_1 \cdot \rho_m$ avec :*
     - *ρ_1 est un chemin de $\mathrm{RG}^{\ominus\&c}(R)$;*
     - *ρ_m est un chemin de mémorisation de $\overline{\mathrm{RG}}^{\ominus\&c}(R)$.*
3. *Si $\rho = (q_0; w_0) \xrightarrow{\alpha_0} (q_1; w_1) \ldots (q_m; w_m) \xrightarrow{\alpha_m} (q_{m+1}; w_{m+1}) \ldots$ est une exécution de R, alors il existe une exécution $q_{0_G} \xrightarrow{\alpha_0} q_{1_G} \ldots q_{m_G} \xrightarrow{\alpha_m} q_{m+1_G} \ldots$ dans $\overline{\mathrm{RG}}^{\ominus\&c}(R)$ telle que pour tout $i \in \mathbb{N}$, q_{i_G} a pour état de contrôle q_i.*
4. *Si $\rho = q_{0_G} \xrightarrow{\alpha_0} q_{1_G} \ldots q_{m_G} \xrightarrow{\alpha_m} q_{m+1_G} \ldots$ est une exécution de $\widetilde{\mathrm{RG}}^{\ominus\&c}(R)$, alors il existe une exécution $(q_0; w_0) \xrightarrow{\alpha_0} (q_1; w_1) \ldots (q_m; w_m) \xrightarrow{\alpha_m} (q_{m+1}; w_{m+1}) \ldots$ dans R telle que pour tout $i \in \mathbb{N}$, q_i est l'état de contrôle de q_{i_G}.*

Preuve Les points 1 et 2 sont immédiats à partir des définitions 7.4 et 7.3. Il est clair en effet que les chemins de $\widetilde{\mathrm{RG}}^{\ominus\&c}(R)$ sont les chemins de $\overline{\mathrm{RG}}^{\ominus\&c}(R)$ qui, s'il contiennent une transition de mémorisation, alors bouclent ensuite par des mémorisations sur le même état.

Montrons le troisième point. Supposons que $\rho = (q_0; w_0) \xrightarrow{\alpha_0} (q_1; w_1) \ldots (q_m; w_m) \xrightarrow{\alpha_m} (q_{m+1}; w_{m+1}) \ldots$ est une exécution de R. Deux cas se présentent :

- si $|\pi^{\ominus\&c}(\rho)| < \infty$, alors, d'après la proposition 5.3, il existe deux chemins ρ_1 et ρ_m dans R tels que :
  - ρ_m est un chemin de mémorisation ;
  - $\rho = \rho_1 \cdot \rho_m$.

  On conclut en utilisant le théorème 5.1 pour ρ_1, ainsi que la remarque 24.
- si $|\pi^{\ominus\&c}(\rho)| = \infty$, alors on conclut en utilisant le théorème 5.1, ainsi que la remarque 24.

Montrons le quatrième point. Supposons que $\rho = q_{0\,G} \xrightarrow{\alpha_0} q_{1\,G} \ldots q_{m\,G} \xrightarrow{\alpha_m} q_{m+1\,G} \ldots$ est une exécution de $\widetilde{\mathrm{RG}}^{\ominus\&c}(R)$. D'après le deuxième point, deux cas se présentent :

- ρ est un chemin de $\mathrm{RG}^{\ominus\&c}(R)$, et alors ρ est une exécution de R.
- ρ s'écrit $\rho_1 \cdot \rho_m$ avec :
  - ρ_1 est un chemin de $\mathrm{RG}^{\ominus\&c}(R)$, et donc ρ_1 est une exécution de R ;
  - ρ_m est un chemin de mémorisation de $\overline{\mathrm{RG}}^{\ominus\&c}(R)$. On peut écrire ρ_m sous la forme $\rho_m = q_G \xrightarrow{\oplus e_1} q_G \xrightarrow{\oplus e_2} q_G \xrightarrow{\oplus e_3} \ldots$. Par définition de $\overline{\mathrm{RG}}^{\ominus\&c}(R)$, pour tout $i \in \mathbb{N}$, $q_G \xrightarrow{\oplus e_i}$ est une transition de R. On en déduit finalement que, si q_G s'écrit $(q; w)$, alors d'une part $(q; w)$ est stable, et d'autre part pour tout $i \in \mathbb{N}$, $e_i \in \mathit{Mémo}(q)$. Par conséquent, d'après la remarque 10, $(q; w) \xrightarrow{\oplus e_1} (q; w_1) \xrightarrow{\oplus e_2} (q; w_2) \xrightarrow{\oplus e_3} (q; w_3) \ldots$ est un chemin de R. ♦

On étend également la définition de Υ :

Définition 7.5 *Soit R un AFR à file initiale vide. Pour tout chemin fini ou infini ρ de $\overline{\mathrm{RG}}^{\ominus\&c}(R)$, on note $\Upsilon(\rho)$ le chemin de $\overline{\mathrm{RG}}^{\ominus\&c}(R)$ défini par:*

- *si ρ est un chemin infini de mémorisation, alors $\Upsilon(\rho) = \rho$;*
- *sinon :*
  - *si ρ s'écrit $q_G \xrightarrow{\oplus e} q_G \cdot \rho'$, alors $\Upsilon(\rho) = \Upsilon(\rho')$;*
  - *si ρ s'écrit $q_G \xrightarrow{e} q'_G \cdot \rho'$, alors $\Upsilon(\rho) = q \xrightarrow{e} q' \cdot \Upsilon(\rho')$;*
  - *si ρ s'écrit $q_G \xrightarrow{\ominus e} q'_G \cdot \rho'$, alors $\Upsilon(\rho) = q_G \xrightarrow{\ominus e} q'_G \cdot \Upsilon(\rho')$.*
  - *sinon, alors ρ est de longueur nulle, et $\Upsilon(\rho) = \rho$.*

On note que la remarque 22 se généralise aisément, et que les lemmes 7.1.2 et 7.1.3 s'écrivent à présent :

Lemme 7.1.4 *Soit R un AFR. Pour tout chemin ρ de $\overline{\mathrm{RG}}^{\ominus\&c}(R)$, $\Upsilon(\rho)$ est un chemin de $\widetilde{\mathrm{RG}}^{\ominus\&c}(R)$.*

Preuve similaire à la preuve du lemme 7.1.2. ♦

Lemme 7.1.5 *Soit R un AFR. Pour tout chemin ρ de $\overline{\mathrm{RG}}^{\ominus\&c}(R)$, et pour tout état q de $\overline{\mathrm{RG}}^{\ominus\&c}(R)$, ρ visite infiniment souvent l'état q ssi $\Upsilon(\rho)$ visite infiniment souvent l'état q.*

Preuve similaire à la preuve du lemme 7.1.3. ♦

Proposition 7.3 *Soit R un AFR. Les trois assertions suivantes sont équivalentes :*

1. *il existe une exécution ρ de $\widetilde{\mathrm{RG}}^{\ominus\&c}(R)$ visitant infiniment souvent l'état de contrôle q ;*
2. *il existe une exécution ρ de R visitant infiniment souvent l'état de contrôle q ;*
3. *il existe une exécution ρ de $\overline{\mathrm{RG}}^{\ominus\&c}(R)$ visitant infiniment souvent l'état de contrôle q.*

Preuve Les implications $1 \Longrightarrow 2$ et $2 \Longrightarrow 3$ sont immédiates en utilisant la proposition 7.2. L'implication $3 \Longrightarrow 1$ provient des lemmes 7.1.4 et 7.1.5. ♦

Enfin, d'après le corollaire 5.7, le graphe d'accessibilité sans mémorisation complété associé à un AFR est un système de transitions fini et calculable, pour lequel le RSP est trivialement décidable. On aboutit donc au théorème :

Théorème 7.2 *Le RSP est décidable pour les AFRs.*

7.1.3 Remarques préliminaires à LTL et CTL

Nous effectuons ici d'une part un rapprochement avec un résultat de [1], et d'autre part nous donnons un aperçu quelque peu informel de notre démarche pour les deux sections suivantes.

Soit S et S' deux systèmes de transitions, d'ensembles d'états respectifs Q et Q'. Soit $\mathcal{X}$ un ensemble fini de *paramètres d'états* valables pour S et S', i.e. il existe deux applications $\chi : Q \longrightarrow \mathcal{X}$ et $\chi' : Q' \longrightarrow \mathcal{X}$ qui à chaque état de S, et de S' respectivement, associent un paramètre. Si ρ et ρ' sont des exécutions de S et S' respectivement, alors on note $\rho \simeq \rho'$ lorsque les états successifs de ρ et de ρ' ont mêmes paramètres, c'est-à-dire :

- ρ et ρ' ont même longueur $l \in \mathbb{N} \cup \{\infty\}$;
- pout tout $i \in \{0, \ldots, l\}$, $\chi(\rho(i)) = \chi'(\rho'(i))$.

On note enfin $S \sqsubseteq S'$ lorsque pour toute exécution ρ de S, il existe une exécution ρ' de S' telle que $\rho \simeq \rho'$. On remarque que $S \sqsubseteq S'$ exprime en fait la simulation "modulo $\mathcal{X}$" de S par S'.

On considère à présent une classe de formules $\mathcal{C}$ construite sur un ensemble de propositions atomiques AP. Pour l'interprétation des formules $\varphi \in \mathcal{C}$ sur S et S', à chaque paramètre $x \in \mathcal{X}$ est associé l'ensemble des propositions atomiques vraies pour x par une application $\mu : \mathcal{X} \longrightarrow 2^{AP}$. On dit que $\mathcal{C}$ est une *classe de formules intrinsèquement existentielles* ssi pour toute formule $\varphi \in \mathcal{C}$, et pour tout système de transitions S, S' :

$$\{S \sqsubseteq S'\} \implies \{S, \mu \models \varphi \implies S', \mu \models \varphi\}$$

Pour un AFR, pour un système de contrôle, éventuellement complété ou intermédiaire, et pour un graphe d'accessibilité sans mémorisation, éventuellement complété ou intermédiaire, associé à un AFR, l'ensemble $\mathcal{X}$ des paramètres d'états est l'ensemble des états de contrôles. Les paramétrages χ respectifs associent à chaque état l'état de contrôle correspondant.

Dans [1], les auteurs montrent notamment que si R est un AFR à file initiale vide[2] associé à un système de contrôle C, alors $C \sqsubseteq R \sqsubseteq \overline{C}$. Le lemme 7.1.1 que nous avons démontré donne un encadrement plus fin :

$$C \sqsubseteq \widetilde{C} \sqsubseteq R \sqsubseteq \overline{C}$$

Enfin, les points 3 et 4 de la proposition 7.2 que nous avons démontrée généralisent cet encadrement aux AFRs quelconques : si R est un AFR quelconque, alors :

$$\mathrm{RG}^{\ominus\&c}(R) \sqsubseteq \widetilde{\mathrm{RG}}^{\ominus\&c}(R) \sqsubseteq R \sqsubseteq \overline{\mathrm{RG}}^{\ominus\&c}(R)$$

Par conséquent, dans la suite, pour LTL comme pour CTL, nous cherchons à exploiter les propriétés des AFRs pour trouver une classe de formules intrinsèquement existentielles $\mathcal{C}$ telle que si R est un AFR, alors pour toute formule φ de $\mathcal{C}$:

$$\overline{\mathrm{RG}}^{\ominus\&c}(R), \mu \models \varphi \implies \widetilde{\mathrm{RG}}^{\ominus\&c}(R), \mu \models \varphi$$

On note en effet que pour une telle classe de formules $\mathcal{C}$, et pour une formule $\varphi \in \mathcal{C}$, on a :

$$\overline{\mathrm{RG}}^{\ominus\&c}(R), \mu \models \varphi \iff R, \mu \models \varphi \iff \widetilde{\mathrm{RG}}^{\ominus\&c}(R), \mu \models \varphi$$

On ramène ainsi le model-checking de $\mathcal{C}$ pour les AFRs au model-checking de $\mathcal{C}$ pour des systèmes de transitions finis.

7.2 Décidabilité d'un fragment de la logique temporelle linéaire LTL

Nous étendons à présent la démarche qui nous a permis de décider le RSP pour les AFRs à la décision d'un fragment de LTL, qui, permet notamment d'exprimer le RSP.

2. dans [1], un AFR à file initiale vide est appelé un "FIFO transition system".

On rappelle la syntaxe des formules de LTL construites sur un ensemble fini de propositions atomiques AP fixé [19, 15] :

$$\varphi ::= p \in AP \mid \neg\varphi \mid \ \circ\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \mathcal{U} \varphi_2$$

On a les abréviations habituelles suivantes : $\varphi_1 \vee \varphi_2 \equiv \neg(\neg\varphi_1 \wedge \neg\varphi_2)$, $\Diamond\varphi \equiv$ true $\mathcal{U}\, \varphi$, $\Box\varphi \equiv \neg\Diamond\neg\varphi$.

Rappelons que l'interprétation des formules de LTL est réalisée sur un *système de transitions fini paramétré* (S, μ), où :

- S est un système de transitions fini d'ensemble d'états Q ;
- $\mu : Q \longrightarrow 2^{AP}$ est un paramétrage.

Le paramétrage indique pour chaque état q d'un système de R l'ensemble des propositions atomiques vraies dans l'état q.

Nous interprétons également les formules de LTL sur un *AFR paramétré* (R, μ), où :

- R est un AFR d'ensemble d'états de contrôle Q ;
- $\mu : Q \longrightarrow 2^{AP}$ est un paramétrage.

Le paramétrage indique pour chaque état de contrôle q de R l'ensemble des propositions atomiques vraies dans l'état q.

Nous ne rappelons pas ici la sémantique des formules de LTL, qui pourra être trouvée par exemple dans [19, 15]. Nous rappelons seulement que les formules de LTL s'appliquent à des chemins, et qu'un système de transitions paramétré (S, μ) (resp. un AFR paramétré (R, μ)) satisfait une formule φ, noté $S, \mu \models \varphi$ (resp. $R, \mu \models \varphi$), ssi tous les chemins de S (resp. de R) satisfont φ.

Le théorème suivant donne la décidabilité du model-checking de LTL dans le cas des systèmes de transitions finis [19].

Théorème 7.3 *Soit (S, μ) un système de transitions fini paramétré, et φ une formule de* LTL. *$(S, \mu) \models \varphi$ est décidable.*

On note que la logique LTL exprime des propriétés sur les chemins qui sont indépendantes du système de transitions considéré d'une part, et des étiquettes des transitions d'autre part. LTL ne permet d'exprimer des propriétés que sur les suites d'états d'un chemin. C'est la raison pour laquelle dans la suite de cette étude de LTL, **si S est un système de transitions, et ρ_1, ρ_2 deux chemins de S, on note $\rho_1 \equiv \rho_2$ si et seulement si ρ_1 et ρ_2 ont mêmes suites d'états**. Avec cette notation, la remarque précédente s'exprime plus formellement par :

Remarque 26 Soit (S, μ) un système de transitions paramétré, et ρ_1, ρ_2 deux chemins de S. Pour toute formule φ de LTL:

$$\{\rho_1 \equiv \rho_2\} \implies \{S, \mu, \rho_1 \models \varphi \iff S, \mu, \rho_2 \models \varphi\}$$

Afin de pouvoir utiliser les résultats de la section précédente, on associe, pour tout AFR paramétré (R, μ), un paramétrage μ_G sur le système de transitions fini $\mathrm{RG}^{\ominus\&c}$.

Définition 7.6 *Pour tout AFR paramétré (R, μ), on note μ_G le paramétrage sur l'ensemble Q_G des états de $\mathrm{RG}^{\ominus\&c}(R)$ défini par:*

$$\forall\, (q; w) \in Q_G, \mu_G((q; w)) = \mu(q)$$

On remarque tout d'abord que de la section précédente vient le résultat suivant :

Proposition 7.4 *Soit (R, μ) un AFR paramétré, et φ une formule de* LTL. *Les deux assertions suivantes sont vraies:*

1. $\left\{\overline{\mathrm{RG}}^{\ominus\&c}(R), \mu_G \models \varphi\right\} \Longrightarrow \{R, \mu \models \varphi\}$
2. $\{R, \mu \models \varphi\} \Longrightarrow \left\{\widetilde{\mathrm{RG}}^{\ominus\&c}(R), \mu_G \models \varphi\right\}$

Preuve Immédiate en utilisant les points 3 et 4 de la proposition 7.2 et la définition 7.6. ♦

L'idée[3] est donc de chercher une classe $\mathcal{C}$ de formules de LTL telle que pour toute formule $\varphi \in \mathcal{C}$, et pour tout paramétrage μ_G :

$$\left\{\widetilde{\mathrm{RG}}^{\ominus\&c}(R), \mu_G \models \varphi\right\} \Longrightarrow \left\{\overline{\mathrm{RG}}^{\ominus\&c}(R), \mu_G \models \varphi\right\} \tag{7.1}$$

La classe que nous considérons est la restriction de LTL aux formules ne contenant pas d'occurrence de l'opérateur $\circ$.

Définition 7.7 *On appelle* $\mathrm{LTL}^{-\circ}$ *le sous-ensemble des formules de* LTL *dont la syntaxe est donnée par:*

$$\varphi ::= p \in AP \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \,\mathcal{U}\, \varphi_2$$

Toute formule de $\mathrm{LTL}^{-\circ}$ *est une formule de* LTL, *et est interprétée comme telle.*

L'opérateur sur les chemins que nous définissons ci-après va permettre d'exprimer une équivalence de chemins pour la classe de formules $\mathrm{LTL}^{-\circ}$, dans le cadre général des systèmes de transitions paramétrés. On en déduira ensuite que l'implication (4.1) est vérifiée pour la classe $\mathrm{LTL}^{-\circ}$.

Définition 7.8 *Soit (S, μ) un système de transitions paramétré. Pour tout chemin fini ou infini ρ de S, on note $\Gamma(\rho)$ le chemin de S défini par:*

– *si ρ s'écrit $q \longrightarrow q \longrightarrow q \ldots$, alors $\Gamma(\rho) = \rho$;*

3. par définition, les formules de LTL sont "intrinsèquement universelles", et par conséquent, la démarche que nous suivons ici est la démarche duale de celle que nous avons donnée dans la section 7.1.3.

– *sinon :*

  – *si* ρ *s'écrit* $q \longrightarrow q \cdot \rho'$, *alors* $\Gamma(\rho) = \Gamma(\rho')$.
  – *si* ρ *s'écrit* $q \longrightarrow q' \cdot \rho'$ *avec* $q' \neq q$, *alors* $\Gamma(\rho) = q \longrightarrow q' \cdot \Gamma(\rho')$.
  – *sinon, alors* ρ *est de longueur nulle, et* $\Gamma(\rho) = \rho$.

Remarque 27 Soit (S, μ) un système de transitions paramétré. Soit ρ un chemin de S. Les trois assertions suivantes se démontrent d'une manière semblable à la remarque 22 :

– $\Gamma(\rho)$ a même état d'origine que ρ ;

– si ρ est fini et s'écrit $q \longrightarrow q \longrightarrow q \ldots q \longrightarrow q$, alors $\Gamma(\rho) = q$.

– si ρ s'écrit $\rho_1 \cdot \rho_2$, alors $\Gamma(\rho) = \Gamma(\rho_1) \cdot \Gamma(\rho_2)$;

Le résultat suivant est en connection avec l'équivalence de systèmes de transitions par "stuttering" définie dans [20] :

Proposition 7.5 *Soit* (S, μ) *un système de transitions paramétré. Si* φ *est une formule de* $\mathrm{LTL}^{-\circ}$, *alors pour tout chemin* ρ *de* R, $\mu, \Gamma(\rho) \models \varphi \Longleftrightarrow \mu, \rho \models \varphi$.

Preuve Par induction sur la structure des formules de $\mathrm{LTL}^{-\circ}$. On peut également consulter [20]. ♦

Nous faisons à présent le lien avec la satisfaction des formules de $LTL^{-\circ}$ pour un AFR.

Lemme 7.2.1 *Soit* (R, μ) *un AFR paramétré. Pour tout chemin* ρ *de* $\overline{\mathrm{RG}}^{\ominus \& c}(R)$, *et pour toute formule* φ *de* $\mathrm{LTL}^{-\circ}$ *:*

$$\mu_G, \Gamma(\Upsilon(\rho)) \models \varphi \Longleftrightarrow \mu_G, \Gamma(\rho) \models \varphi$$

Preuve D'après la remarque 26, il suffit de montrer que pour tout chemin ρ de $\overline{\mathrm{RG}}^{\ominus \& c}(R)$, $\Gamma(\Upsilon(\rho)) \equiv \Gamma(\rho)$. On montre en fait par récurrence sur $i \in \mathbb{N}$, que pour tout $i \in \mathbb{N}$, pour tout chemin ρ de $\overline{\mathrm{RG}}^{\ominus \& c}(R)$, $\Gamma(\Upsilon(\rho))_{|_i} \equiv \Gamma(\rho)_{|_i}$. On note que la base est triviale, puisque pour tout chemin ρ de $\overline{\mathrm{RG}}^{\ominus \& c}(R)$, $\Gamma(\Upsilon(\rho))$ et $\Gamma(\rho)$ ont même état d'origine : l'état d'origine de ρ. Pour l'induction, on se donne $i \in \mathbb{N}$, et on suppose que pour tout chemin ρ de $\overline{\mathrm{RG}}^{\ominus \& c}(R)$, $\Gamma(\Upsilon(\rho))_{|_i} \equiv \Gamma(\rho)_{|_i}$. Soit ρ un chemin de $\overline{\mathrm{RG}}^{\ominus \& c}(R)$.

– si ρ s'écrit $q \longrightarrow q \longrightarrow q \ldots$, alors puisque d'une part $\Upsilon(\rho)$ ne fait qu'enlever éventuellement des transitions de mémorisation, et d'autre part, d'après la remarque 22, $\Upsilon(\rho)$ est infini, on obtient que $\Upsilon(\rho)$ s'écrit également $q \longrightarrow q \longrightarrow q \ldots$. Finalement $\Gamma(\Upsilon(\rho))_{|_{i+1}} \equiv \Gamma(\rho)_{|_{i+1}}$.

– sinon, ρ s'écrit $\rho_1 \cdot q \longrightarrow q' \cdot \rho_2$, avec :

  – $q' \neq q$;

– ρ_1 est un chemin fini s'écrivant $q \longrightarrow q \longrightarrow q \ldots q \longrightarrow q$.

Par conséquent, d'une part, $\Upsilon(\rho_1)$ est fini et s'écrit également $q \longrightarrow q \longrightarrow q \ldots q \longrightarrow q$, et d'autre par la transition $q \longrightarrow q'$ n'est pas une transition de mémorisation. On obtient donc $\Upsilon(\rho) = \Upsilon(\rho_1) \cdot q \longrightarrow q' \cdot \Upsilon(\rho_2)$. Ainsi, d'après la remarque 27 :

– $\Gamma(\Upsilon(\rho)) = \Gamma(\Upsilon(\rho_1)) \cdot q \longrightarrow q' \cdot \Gamma(\Upsilon(\rho_2)) = q \longrightarrow q' \cdot \Gamma(\Upsilon(\rho_2))$;

– $\Gamma(\rho) = q \longrightarrow q' \cdot \Gamma(\rho_2)$.

Or par hypothèse de récurrence, on a $\Gamma(\Upsilon(\rho_2))_{|_i} \equiv \Gamma(\rho_2)_{|_i}$. Finalement, on aboutit à :

$$\begin{aligned}
\Gamma(\Upsilon(\rho))_{|_{i+1}} &= \left[q \longrightarrow q' \cdot \Gamma(\Upsilon(\rho_2)) \right]_{|_{i+1}} \\
&= q \longrightarrow q' \cdot \Gamma(\Upsilon(\rho_2)_{|_i} \\
&\equiv q \longrightarrow q' \cdot (\Gamma(\rho_2)_{|_i} \\
&\equiv \Gamma(q \longrightarrow q' \cdot \rho_2)_{|_{i+1}} \\
&\equiv \Gamma(\rho)_{|_{i+1}} \qquad \blacklozenge
\end{aligned}$$

Proposition 7.6 *Soit R un AFR. Si φ est une formule de* $\mathrm{LTL}^{-\circ}$*, alors pour tout chemin ρ de $\overline{\mathrm{RG}}^{\ominus\&c}(R)$:*

$$\mu_G, \Upsilon(\rho) \models \varphi \Longrightarrow \mu_G, \rho \models \varphi$$

Preuve Soit ρ un chemin de $\overline{\mathrm{RG}}^{\ominus\&c}(R)$. Supposons que $\mu_G, \Upsilon(\rho) \models \varphi$. D'après la proposition 7.5, $\mu_G, \Gamma(\Upsilon(\rho)) \models \varphi$. Par conséquent, d'après le lemme 7.2.1, $\mu_G, \Gamma(\rho) \models \varphi$. Finalement, d'après la proposition 7.5, $\mu_G, \rho \models \varphi$. $\blacklozenge$

Proposition 7.7 *Soit (R, μ) un AFR paramétré, et φ est une formule de* $\mathrm{LTL}^{-\circ}$*. Les trois assertions suivantes sont équivalentes:*

1. $\overline{\mathrm{RG}}^{\ominus\&c}(R), \mu_G \models \varphi$;
2. $R, \mu \models \varphi$;
3. $\widetilde{\mathrm{RG}}^{\ominus\&c}(R), \mu_G \models \varphi$.

Preuve Les implications $1 \Longrightarrow 2$ et $2 \Longrightarrow 3$ sont prouvées par la proposition 7.4. L'implication $3 \Longrightarrow 1$ provient de la proposition 7.6 et du lemme 7.1.4. $\blacklozenge$

Enfin, d'après le corollaire 5.7, le graphe d'accessibilité sans mémorisation complété associé à un AFR est un système de transitions fini et calculable, pour lequel le model-checking de LTL est décidable. On aboutit donc au théorème :

Théorème 7.4 *Le model-checking de* $\mathrm{LTL}^{-\circ}$ *est décidable pour les AFRs.*

Remarque 28 On peut étendre le théorème 7.4 aux formules de LTL qui s'écrivent $\circ \ldots \circ \varphi$, où φ et une formule de LTL$^{-\circ}$. Si R est un AFR, il suffit en effet de développer l'arbre d'accessibilité de R sur une profondeur égale au nombre d'occurrence de $\circ$ dans une telle formule, puis de vérifier pour chaque feuille $(q; w)$ de l'arbre obtenu si l'AFR obtenu à partir de R en prenant comme état initial $(q; w)$ satisfait la formule de LTL$^{-\circ}$ φ.

Remarque 29 Dans cette remarque, on considère les formules, de manière assez classique, comme des arbres. On constate que pour toute formule φ_1 et φ_2 de LTL, on a :

$$\circ\neg\varphi_1 \equiv \neg \circ \varphi_1 \ , \ \circ(\varphi_1 \wedge \varphi_2) \equiv \circ\varphi_1 \wedge \circ\varphi_2 \ , \ \circ(\varphi_1 \mathcal{U} \varphi_2) \equiv \circ\varphi_1 \mathcal{U} \circ\varphi_2$$

Ainsi, pour toute formule φ de LTL comportant des occurrences de $\circ$, on peut "repousser" les $\circ$ jusque devant les propositions atomiques, i.e. les feuilles de l'arbre de φ, en utilisant les relations pécédentes. Si le même nombre de $\circ$ se trouvent devant chaque proposition atomique, alors on peut les remonter pour qu'ils se retrouvent tous en haut de l'arbre, et ainsi φ est équivalente à une formule de la forme $\circ \ldots \circ \varphi_1$, où φ_1 et une formule de LTL$^{-\circ}$, de sorte qu'on peut là aussi décider la satisfaction de φ.

Enfin, pour terminer notre analyse de la décidabilité du model-checking de LTL pour les AFRs, l'exemple qui suit montre que la démarche qui nous a permis d'obtenir la décidabilité de LTL$^{-\circ}$ n'est plus valable lorsqu'on considère les formules générales de LTL. En effet, comme l'illustre cet exemple, on peut avec l'opérateur $\circ$ spécifier qu'un chemin effectue un nombre voulu de mémorisations dans un état donné, auquel cas les chemins sont fortement contraints, puisque la sémantique des AFRs privilégie le traitement des occurrences mémorisées.

Exemple 7.1 On considère l'AFR R à file initiale vide de la figure 7.1. Soit μ le paramétrage défini par $\mu(q_0) = \{p_0\}, \mu(q_2) = \{p_2\}, \mu(q_1) = \mu(q_3) = \emptyset$. On pose $\varphi = \Box p_0 \vee \circ((p_0 \wedge \circ\neg p_0) \Rightarrow \Diamond p_2)$. On note que dans cet exemple, puisque R est un AFR à file initiale vide, le graphe d'accessibilité sans mémorisation de R est bissimilaire au système de contrôle C associé à R, et donc $\mu_G = \mu$. Cependant :

- $R, \mu \models \varphi$: soit ρ une exécution de R. Si $\mu, \rho \not\models \Box p_0$, alors ρ ne boucle par sur l'état q_0. Par conséquent, si de plus $\mu, \rho^1 \models (p_0 \wedge \circ\neg p_0)$, alors ρ s'écrit $(q_0; \lambda) \xrightarrow{\oplus e_m} (q_0; e_m) \xrightarrow{e_f} (q_1; e_m) \cdot \rho_2$. On note que $(q_1; e_m)$ est instable, et ainsi ρ_2 s'écrit $(q_1; e_m) \xrightarrow{\ominus e_m} (q_2; \lambda)$. Finalement $\mu, \rho^1 \models \Diamond p_2$.

- $\overline{C}, \mu \not\models \varphi$: le chemin $\rho = (q_0; \lambda) \xrightarrow{\oplus e_m} (q_0; e_m) \xrightarrow{e_f} (q_1; e_m) \xrightarrow{e_f} (q_3; e_m) \xrightarrow{e_f} (q_3; e_m) \ldots$ est un chemin de $\overline{C}$, et $\mu, \rho \not\models \varphi$. $\Box$

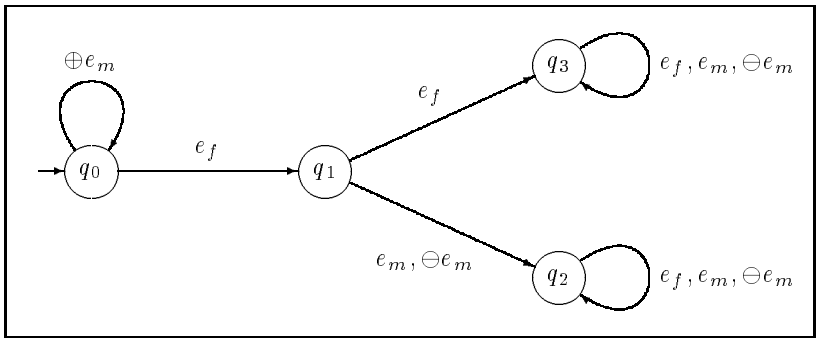


FIG. 7.1 – *Contre-exemple pour* LTL *et* CTL.

7.3 Décidabilité d'un fragment de la logique temporelle arborescente CTL

Nous nous intéressons à présent à la logique CTL. Le résultat que nous obtenons est partiel, et se base sur les points 3 et 4 de la proposition 7.2. En introduisant une notion d'équité sur les AFRs, nous obtenons des résultats de décidabilité pour une classe de formules intrinsèquement existentielles de CTL, et pour la classe duale de formules intrinsèquement universelles.

On rappelle la syntaxe des formules de CTL construites sur un ensemble fini de propositions atomiques AP fixé [19, 15] :

$$\varphi ::= p \in AP \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \exists_\circ \varphi \mid \exists\, \varphi_1 \mathcal{U}\, \varphi_2 \mid \forall \varphi_1 \mathcal{U}\, \varphi_2$$

On a les abréviations habituelles suivantes : $\varphi_1 \vee \varphi_2 \equiv \neg(\neg\varphi_1 \wedge \neg\varphi_2)$, $\forall_\circ \varphi \equiv \neg\exists_\circ\neg\varphi$. $\exists_\Diamond \varphi \equiv \exists\,\mathrm{true}\,\mathcal{U}\,\varphi$, $\forall_\Diamond \varphi \equiv \forall\,\mathrm{true}\,\mathcal{U}\,\varphi$, $\exists_\Box \varphi \equiv \neg\forall_\Diamond\neg\varphi$, $\forall_\Box \varphi \equiv \neg\exists_\Diamond\neg\varphi$.

Nous interprétons les formules de CTL sur un système de transitions fini paramétré (S, μ) ou sur un AFR paramétré (R, μ). Nous ne rappelons pas ici la sémantique des formules de CTL, qui pourra être trouvée par exemple dans [19, 15]. Nous rappelons seulement que les formules de CTL s'appliquent à des états, et qu'un système de transitions paramétré (S, μ) (resp. un AFR paramétré (R, μ)) satisfait une formule φ, noté $S, \mu \models \varphi$ (resp. $R, \mu \models \varphi$), ssi l'état initial de S (resp. de R) satisfait φ.

On utilise également l'opérateur de précédence $\mathcal{B}$, qui est défini par :

- $\exists\, \varphi_1 \mathcal{B}\, \varphi_2 \equiv \neg\,(\forall\neg\varphi_1 \mathcal{U}\, \neg\varphi_2)$;
- $\forall \varphi_1 \mathcal{B}\, \varphi_2 \equiv \neg\,(\exists\,\neg\varphi_1 \mathcal{U}\, \neg\varphi_2)$;

Avec cet opérateur, toute formule de CTL peut être mise sous forme normale négative en repoussant les négations jusqu'aux propositions atomiques. Plus formellement, on note CTL^N l'ensemble des formules dont la syntaxe est donnée par :

$$\begin{aligned}\varphi \quad ::= \quad & p \in AP \mid \neg p, p \in AP \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \exists_\circ \varphi \mid \forall_\circ \varphi \mid \\ & \exists\, \varphi_1 \mathcal{U}\, \varphi_2 \mid \forall \varphi_1 \mathcal{U}\, \varphi_2 \mid \exists\, \varphi_1 \mathcal{B}\, \varphi_2 \mid \forall \varphi_1 \mathcal{B}\, \varphi_2\end{aligned}$$

et on a :

Proposition 7.8 *Pour toute formule φ de CTL, il existe une formule calculable φ^N de* CTL^N *telle que $\varphi \equiv \varphi^N$.*

Le théorème suivant donne la décidabilité du model-checking de CTL^N dans le cas des systèmes de transitions finis [19].

Théorème 7.5 *Soit (S, μ) un système de transitions fini paramétré, et φ une formule de* CTL^N*. $(S, \mu) \models \varphi$ est décidable.*

Définition 7.9 *Une formule φ de* CTL^N *est dite* existentielle *(resp.* universelle*) ssi φ ne contient pas d'occurrence du quantificateur universel $\forall$ (resp. du quantificateur existentiel $\exists$).*

Nous limitons notre étude aux formules existentielles et aux formules universelles. Comme dans le cas de LTL, nous considérons de plus des formules ne contenant pas d'occurrence de l'opérateur $\circ$.

Définition 7.10 *On appelle* $\mathrm{CTL}^{\exists -\circ}$ *(resp.* $\mathrm{CTL}^{\forall -\circ}$*) le sous-ensemble des formules de* CTL^N *dont la syntaxe est donnée par :*

$$\begin{aligned} \varphi &::= p \in AP \mid \neg p, p \in AP \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \exists\, \varphi_1 \mathcal{U}\, \varphi_2 \mid \exists\, \varphi_1 \mathcal{B}\, \varphi_2 \\ (\textit{resp. } \varphi &::= p \in AP \mid \neg p, p \in AP \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \forall \varphi_1 \mathcal{U}\, \varphi_2 \mid \forall \varphi_1 \mathcal{B}\, \varphi_2) \end{aligned}$$

Toute formule de $\mathrm{CTL}^{\exists -\circ}$ *(resp.* $\mathrm{CTL}^{\forall -\circ}$*) est une formule de* CTL^N*, et est interprétée comme telle.*

Remarque 30 La classe de formules $\mathrm{CTL}^{\exists -\circ}$ est une classe de formules intrinsèquement existentielles (cf. section 7.1.3). La démonstration est réalisée par induction sur la structure des formules de $\mathrm{CTL}^{\exists -\circ}$. Suppsons que φ est une formule de $\mathrm{CTL}^{\exists -\circ}$. Soient S et S' deux systèmes de transitions tels que $S \sqsubseteq S'$. Si $S, \mu \models \varphi$, alors trois cas se présentent :

- si φ s'écrit p ou $\neg p$ avec $p \in AP$: alors le résultat est trivial ;
- si φ s'écrit $\varphi_1 \wedge \varphi_2$ ou $\varphi_1 \vee \varphi_2$: alors le résultat provient de l'hypothèse d'induction ;
- si φ s'écrit $\exists\, \varphi_1 \mathcal{U}\, \varphi_2$ ou $\exists\, \varphi_1 \mathcal{B}\, \varphi_2$: alors on utilise la relation $S \sqsubseteq S'$ et l'hypothèse d'induction pour conclure.

Remarque 31 Soit (R, μ) un AFR paramétré. On construit l'opérateur neg : $\mathrm{CTL}^N \longrightarrow \mathrm{CTL}^N$, de la manière suivante : si φ de CTL^N, alors $\neg\varphi$ est une formule de CTL. Soit φ' la formule de CTL obtenue à partir de $\neg\varphi$ en repoussant les négations jusqu'aux propositions atomiques. Puisque φ' est sous forme normale négative, φ' est une formule de CTL^N, et on pose $\mathrm{neg}(\varphi) = \varphi'$. Clairement, pour toute formule φ de CTL^N, $\mathrm{neg}(\varphi) \equiv_{\mathrm{CTL}} \neg\varphi$. De plus l'opérateur neg vérifie :

- si φ est une formule existentielle (resp. universelle), alors $\mathrm{neg}(\varphi)$ est une formule universelle (resp. existentielle) ;

– si φ est une formule qui ne contient pas d'occurrence de $\circ$, alors $\mathrm{neg}(\varphi)$ est une formule qui ne contient pas d'occurrence de $\circ$.

D'après la remarque précédente, nous pouvons restreindre notre étude à la décidabilité du model-checking des formules existentielles de CTL^N ne contenant pas d'occurrence de $\circ$. Afin d'obtenir des résultats de décidabilité pour le model-checking de $\mathrm{CTL}^{\exists-\circ}$, on introduit une notion d'équité sur les AFRs.

Définition 7.11 *Soit ρ un chemin d'un AFR (resp. d'un graphe d'accessibilité sans mémorisation complété associé à un AFR). On dit que ρ est un chemin* équitable *ssi on se trouve dans l'un des deux cas suivants :*

1. $|\pi^{\ominus\&c}(\rho)| = \infty$;
2. *ρ s'écrit $\rho_1 \cdot (q;w) \xrightarrow{\oplus e} (q;w') \cdot \rho_m$ (resp. $\rho_1 \cdot q \xrightarrow{\oplus e} q \cdot \rho_m$), avec :*
   - *ρ_1 est un chemin fini et ρ_m est un chemin de mémorisation ;*
   - *$Source(q) = \emptyset$.*

Intuitivement, un chemin équitable finit par boucler sur des mémorisations ssi il passe par un état à partir duquel il ne peut franchir que des transitions de mémorisation.

Définition 7.12 *Soit R un AFR. On appelle* AFR équitable associé à R, *noté* $\mathrm{Fair}(R)$, *le système de transitions obtenu à partir de R en ne gardant que les chemins équitables.*

Définition 7.13 *Soit R un AFR.*

– *On appelle* graphe d'accessibilité sans mémorisation complété équitable associé à R, *noté* $\mathrm{Fair}(\overline{\mathrm{RG}}^{\ominus\&c}(R))$, *le système de transitions obtenu à partir de $\overline{\mathrm{RG}}^{\ominus\&c}(R)$ en ne gardant que les chemins équitables.*

– *On appelle* graphe d'accessibilité sans mémorisation intermédiaire équitable associé à R, *noté* $\mathrm{Fair}(\widetilde{\mathrm{RG}}^{\ominus\&c}(R))$, *le système de transitions obtenu à partir de $\widetilde{\mathrm{RG}}^{\ominus\&c}(R)$ en ne gardant que les chemins équitables.*

Remarque 32 La constatation suivante permet de s'assurer qu'on peut parler dans toute la suite de chemins infinis. Soit R un AFR.

– tout état de $\mathrm{Fair}(R)$ est origine d'un chemin infini ;

– tout état de $\mathrm{Fair}(\overline{\mathrm{RG}}^{\ominus\&c}(R))$ est origine d'un chemin infini.

– tout état de $\mathrm{Fair}(\widetilde{\mathrm{RG}}^{\ominus\&c}(R))$ est origine d'un chemin infini.

Le lemme suivant donne une caratérisation précise du graphe d'accessibilité sans mémorisation intermédiaire équitable associé à un AFR.

Lemme 7.3.1 *Soit R un AFR, et soit $\mathrm{RG}^{\ominus\&c}(R) = (Q_G, q_{0_G}, E \cup \{\oplus,\ominus\} \times E, \delta_G)$ le graphe d'accessibilité de R. Pour tout $q_G = (q;w) \in Q_G$, on note $Source(q_G)$ l'ensemble $Source(q)$. Soit $\widehat{\mathrm{RG}}^{\ominus\&c}(R)$, le système de transitions fini défini par $\widehat{\mathrm{RG}}^{\ominus\&c}(R) = (Q_G, q_{0_G}, E \cup \{\oplus,\ominus\} \times E, \delta_G \cup \widehat{\delta_G})$, avec :*

$$\widehat{\delta_G} = \{(q_G, (\oplus, e), q_G) \ / \ Source(q_G) = \emptyset \text{ et } (q_G, (\oplus, e), q_G) \in \overline{\delta_G}\})$$

Le système de transitions $\widehat{\mathrm{RG}}^{\ominus\&c}(R)$ est bissimilaire à $\mathrm{Fair}(\widetilde{\mathrm{RG}}^{\ominus\&c}(R))$.

Preuve Immédiate d'après la définition de $\widetilde{\mathrm{RG}}^{\ominus\&c}(R)$ (cf. définition 7.4), et la définition des chemins équitables. ♦

Des liens étroits existent entre les sytèmes de transitions équitables définis ci-dessus, comme le montre le lemme qui suit.

Lemme 7.3.2 *Soit R un AFR d'état initial $(q_0; w_0)$.*

1. *Si $\rho = q_{0_G} \xrightarrow{\alpha_0} q_{1_G} \ldots q_{m_G} \xrightarrow{\alpha_m} q_{m+1_G} \ldots$ est une exécution de $\mathrm{Fair}(\widetilde{\mathrm{RG}}^{\ominus\&c}(R))$, alors il existe une exécution $(q_0;w_0) \xrightarrow{\alpha_0} (q_1;w_1) \ldots (q_m;w_m) \xrightarrow{\alpha_m} (q_{m+1};w_{m+1}) \ldots$ dans $\mathrm{Fair}(R)$ telle que pour tout $i \in \mathbb{N}$, q_i est l'état de contrôle de q_{i_G} ;*
2. *Si $\rho = (q_0;w_0) \xrightarrow{\alpha_0} (q_1;w_1) \ldots (q_m;w_m) \xrightarrow{\alpha_m} (q_{m+1};w_{m+1}) \ldots$ est une exécution de $\mathrm{Fair}(R)$, alors il existe une exécution $q_{0_G} \xrightarrow{\alpha_0} q_{1_G} \ldots q_{m_G} \xrightarrow{\alpha_m} q_{m+1_G} \ldots$ dans $\mathrm{Fair}(\overline{\mathrm{RG}}^{\ominus\&c}(R))$ telle que pour tout $i \in \mathbb{N}$, q_{i_G} a pour état de contrôle q_i.*

Preuve Immédiate d'après les points 3 et 4 de la proposition 7.2, et la définition des chemins équitables. ♦

Lemme 7.3.3 *Soit R un AFR. On a les encadrements[4] suivants :*

1. $\widetilde{\mathrm{RG}}^{\ominus\&c}(R) \sqsubseteq R \sqsubseteq \overline{\mathrm{RG}}^{\ominus\&c}(R)$
2. $\mathrm{Fair}(\widetilde{\mathrm{RG}}^{\ominus\&c}(R)) \sqsubseteq \mathrm{Fair}(R) \sqsubseteq \mathrm{Fair}(\overline{\mathrm{RG}}^{\ominus\&c}(R))$

Preuve Le premier point est immédiat à partir des points 3 et 4 de la proposition 7.2. Le deuxième point est immédiat à partir du lemme 7.3.2. ♦

Proposition 7.9 *Soit (R, μ) un AFR paramétré. Si φ est une formule de $\mathrm{CTL}^{\exists -\circ}$, alors :*

1. $\widetilde{\mathrm{RG}}^{\ominus\&c}(R), \mu \models \varphi \implies R, \mu \models \varphi$
2. $R, \mu \models \varphi \implies \overline{\mathrm{RG}}^{\ominus\&c}(R), \mu \models \varphi$

4. on rappelle que la relation $\sqsubseteq$ est définie dans la section 7.1.3.

3. $\mathrm{Fair}(\widetilde{\mathrm{RG}}^{\ominus\&c}(R)), \mu \models \varphi \implies \mathrm{Fair}(R), \mu \models \varphi$

4. $\mathrm{Fair}(R), \mu \models \varphi \implies \mathrm{Fair}(\overline{\mathrm{RG}}^{\ominus\&c}(R)), \mu \models \varphi$

Preuve Immédiate d'après le lemme 7.3.3 et la remarque 30. ♦

Proposition 7.10 *Soit (R, μ) un AFR paramétré, et φ une formule de* $\mathrm{CTL}^{\exists-\circ}$. *Pour tout état q_G du graphe d'accessibilité sans mémorisation associé à R :*

1. $\mathrm{Fair}(\overline{\mathrm{RG}}^{\ominus\&c}(R)), \mu, q_G \models \varphi \implies \mathrm{Fair}(\widetilde{\mathrm{RG}}^{\ominus\&c}(R)), \mu, q_G \models \varphi$;

2. *si φ ne contient pas d'occurrence de l'opérateur $\mathcal{B}$, alors* $\overline{\mathrm{RG}}^{\ominus\&c}(R), \mu, q_G \models \varphi \implies \widetilde{\mathrm{RG}}^{\ominus\&c}(R), \mu, q_G \models \varphi$.

Idée de la preuve Les deux points se prouvent par induction sur la structure des formules de $\mathrm{CTL}^{\exists-\circ}$. Si φ est une formule de $\mathrm{CTL}^{\exists-\circ}$, trois cas se présentent :

- si φ s'écrit p ou $\neg p$ avec $p \in AP$: alors le résultat est trivial ;
- si φ s'écrit $\varphi_1 \wedge \varphi_2$ ou $\varphi_1 \vee \varphi_2$: alors le résultat provient de l'hypothèse d'induction ;
- si φ s'écrit $\exists\, \varphi_1 \,\mathcal{U}\, \varphi_2$ ou $\exists\, \varphi_1 \,\mathcal{B}\, \varphi_2$: alors on utilise l'hypothèse d'induction et les définitions respectives de $\mathrm{Fair}(\overline{\mathrm{RG}}^{\ominus\&c}(R)), \mathrm{Fair}(\widetilde{\mathrm{RG}}^{\ominus\&c}(R)), \overline{\mathrm{RG}}^{\ominus\&c}(R)$ et $\widetilde{\mathrm{RG}}^{\ominus\&c}(R)$ pour conclure. ♦

Corollaire 7.1 *Soit (R, μ) un AFR paramétré, et φ une formule de* $\mathrm{CTL}^{\exists-\circ}$.

1. $\mathrm{Fair}(R), \mu \models \varphi \iff \mathrm{Fair}(\widetilde{\mathrm{RG}}^{\ominus\&c}(R)), \mu \models \varphi$;

2. *si φ ne contient pas d'occurrence de l'opérateur $\mathcal{B}$, alors* $R, \mu \models \varphi \iff \widetilde{\mathrm{RG}}^{\ominus\&c}(R), \mu \models \varphi$.

Preuve Immédiate d'après les propositions 7.9 et 7.10. ♦

Enfin, d'après le corollaire 5.7 et le lemme 7.3.1 :

- le graphe d'accessibilité sans mémorisation intermédiaire associé à un AFR est un système de transitions fini et calculable ;
- le graphe d'accessibilité sans mémorisation intermédiaire équitable associé à un AFR est un système de transitions fini et calculable.

Ainsi, d'après le théorème 7.5, on aboutit au théorème :

Théorème 7.6 *Soit (R, μ) un AFR paramétré.*

- *Si φ est une formule de* $\mathrm{CTL}^{\exists-\circ}$, *alors* $\mathrm{Fair}(R), \mu \models \varphi$ *est décidable.*
- *Si φ est une formule de* $\mathrm{CTL}^{\forall-\circ}$, *alors* $\mathrm{Fair}(R), \mu \models \varphi$ *est décidable ;*

- *Si φ est une formule de* $\mathrm{CTL}^{\exists-\circ}$ *sans occurrence de l'opérateur $\mathcal{B}$, alors $R, \mu \models \varphi$ est décidable ;*
- *Si φ est une formule de* $\mathrm{CTL}^{\forall-\circ}$ *sans occurrence de l'opérateur $\mathcal{B}$, alors $R, \mu \models \varphi$ est décidable.*

Nous indiquons sur un exemple, pour terminer cette section, pourquoi le résultat obtenu est aussi partiel.

Exemple 7.2 On considère l'AFR R à file initiale vide de la figure 7.1. Soit μ le paramétrage défini par $\mu(q_0) = \{p_0\}, \mu(q_1) = \{p_1\}, \mu(q_2) = \{p_2\}, \mu(q_3) = \{p_3\}$. On rappelle que dans cet exemple, puisque R est un AFR à file initiale vide, le graphe d'accessibilité sans mémorisation de R est bissimilaire au système de contrôle C associé à R.

Les deux exemples de formules ci-dessous illustrent les problèmes posés par l'opérateur $\circ$.

Soit la formule $\varphi = \exists_\circ(p_0 \wedge \exists_\diamond p_1)$. On considère le chemin $\rho = (q_0; e_m) \xrightarrow{e_f} (q_1; e_m) \xrightarrow{\ominus e_m} (q_2; \lambda) \xrightarrow{e_f} (q_2; \lambda) \ldots$, et il est donc clair qu'on a :

- $R, \mu, (q_0; e_m) \models p_0 \wedge \exists_\diamond p_1$;
- $\mathrm{Fair}(R), \mu, (q_0; e_m) \models p_0 \wedge \exists_\diamond p_1$.

Par conséquent, puisque que (q_0, e_m) est un état successeur de $(q_o; \lambda)$, il vient :

- $R, \mu \models \varphi$;
- $\mathrm{Fair}(R), \mu \models \varphi$.

Par contre, dans le système de contrôle intermédiaire, on n'a pas la transition $q_0 \xrightarrow{\oplus e_m} q_0$, mais la transition $q_0 \xrightarrow{\oplus e_m} \widetilde{q}_0$, et donc :

- $\widetilde{C}, \mu \not\models \varphi$;
- $\mathrm{Fair}(\widetilde{C}), \mu \not\models \varphi$.

On vérifie également que si φ' est la formule $\varphi' = \exists_\circ(p_0 \wedge \exists_\diamond p_3)$, alors :

- $R, \mu \not\models \varphi'$ et $\mathrm{Fair}(R), \mu \not\models \varphi'$, car si on mémorise en p_0, alors on ne peut plus ateindre l'état q_3 ;
- $\overline{C}, \mu \models \varphi'$ et $\mathrm{Fair}(\overline{C}), \mu \models \varphi'$.

Par ailleurs, la formule $\varphi'' = \exists_\Box(p_0 \wedge \exists_\diamond p_2)$ permet de comprendre la différence des résulats obtenus entre $\mathrm{Fair}(R)$ et R. En effet, en notant que le seul chemin correspondant à la formule $\exists_\Box p_0$ est la boucle de mémorisation sur p_0, il est clair qu'on a :

- $R, \mu \models \varphi''$ et $\mathrm{Fair}(R), \mu \not\models \varphi''$;
- $\widetilde{C}, \mu \not\models \varphi''$, $\mathrm{Fair}(\widetilde{C}), \mu \not\models \varphi'''$, $\overline{C}, \mu \models \varphi''$ et $\mathrm{Fair}(\overline{C}), \mu \not\models \varphi'''$.

et on note que par ailleurs, si on avait choisit p_3 à la place de p_2 dans φ'', alors on aurait eu $R, \mu \not\models \varphi''$ et $\overline{C}, \mu \models \varphi''$.

Enfin, si on considère la formule $\varphi''' = \exists_\diamond(p_1 \wedge \forall_\diamond p_3)$, on se rend compte que les imbrications de quantificateurs existentiels et universels posent également problème. On vérifie en effet que :

- $R, \mu \models \varphi'''$ et $\mathrm{Fair}(R), \mu \models \varphi'''$, puisque que $R, \mu, (q_1, e_m) \models p_1 \wedge \forall_\diamond p_3$, et $\mathrm{Fair}(R), \mu, (q_1, e_m) \models p_1 \wedge \forall_\diamond p_3$.
- $\widetilde{C}, \mu \not\models \varphi'''$, $\mathrm{Fair}(\widetilde{C}), \mu \not\models \varphi'''$, $\overline{C}, \mu \not\models \varphi'''$ et $\mathrm{Fair}(\overline{C}), \mu \not\models \varphi'''$. □

7.4 Décidabilité de deux propriétés de famine

Dans les logiques LTL et CTL, les propositions atomiques sont affectées aux états. Pour clore cette section sur la vérification comportementale des AFRs, nous donnons également un exemple de formule faisant intervenir des propositions atomiques affectées aux transitions. Cet exemple permet de se rapprocher des besoins intuitifs d'Électre en matière de vérification.

Une formule de famine en Électre exprime le fait qu'un événement e arrive infiniment souvent alors qu'aucune transition de traitement de e n'est franchie. Le problème qu'on se pose donc est défini ci-après.

Définition 7.14 *Soit R un AFR, et e un événement mémorisable de R. Le problème de* famine *consiste à déterminer s'il existe une exécution infinie ρ dans R telle que :*

- *ρ contient une infinité de transitions de mémorisation de e ;*
- *ρ ne contient pas de transition de traitement de e.*

D'après la sémantique de priorité de la file pour un AFR, on se rend compte que seule la première occurrence de e dans la file compte. Par conséquent, si R est un AFR, le *problème de famine* revient à déterminer s'il existe une exécution infinie ρ dans R telle que :

- ρ contient une transition de mémorisation de e ;
- ρ ne contient pas de transition de traitement de e.

On note que s'il existe un état stable $(q; w)$ accessible de R tel que $e \in \mathit{Mémo}(q)$, alors, le problème est trivialement résolu par la remarque 14. Réciproquement, si une exécution ρ contient une transition de mémorisation de e, alors il existe un état stable accessible de R tel que $e \in \mathit{Mémo}(q)$. La représentation reconnaissable de l'ensemble des états accessibles qu'on obtient en 6 permet alors de décider le problème de famine[5].

5. on note en effet que si un état $(q; w)$ est instable, alors pour tout $w' \geq w$, $(q; w')$ est instable. Par conséquent, on regarde pour chaque état de contrôle q s'il existe un élément minimal pour $\leq$ $w \in \mathcal{L}_R(q)$, tel que $(q; w)$ est stable. On constate enfin que d'après la forme donnée en 6 de $\mathcal{L}_R(q)$, les éléments minimaux de $\mathcal{L}_R(q)$ sont en nombre fini et calculables.

On s'intéresse donc à présent au cas plus réaliste vis-à-vis d'Électre où on impose de plus, dans l'enoncé du problème de famine, que ρ ne se termine pas par une infinité de mémorisations. Cette version du problème de famine est plus équitable, ce qui explique sa dénomination.

Définition 7.15 *Soit R un AFR, et e un événement mémorisable de R. Le problème de* famine équitable *consiste à déterminer s'il existe une exécution infinie ρ dans R telle que :*

- *ρ contient une transition de mémorisation de e ;*
- *ρ ne contient pas de transition de traitement de e ;*
- *ρ ne s'écrit pas $\rho_1 \cdot \rho_2$, avec ρ_2 chemin de mémorisation.*

Soit R un AFR. Supposons qu'il existe une exécution ρ vérifiant les trois conditions de la définition 7.15. Alors ρ s'écrit $\rho_1 \cdot (q;w) \xrightarrow{\oplus e} (q;w_1) \cdot \rho_2$. Par conséquent, w_1 contient une occurrence de e, et donc w_1 s'écrit $u_1 e u_2$. Soit S l'ensemble des états de contrôle q par lesquels passe ρ et vérifiant $e \in Source(q)$. Puisque e n'est jamais traité, dans chaque état de contrôle $q \in S$, il existe un événement $e_t \in Alph(u_1) \cap Source(q)$. Ainsi, chaque transition de traitement à partir d'un état $q \in S$ fait diminuer la taille de u_1 de 1. On en déduit que S est fini, et par conséquent ρ_2 s'écrit $\rho_3 \cdot \rho_4$, telle que ρ_4 est un chemin infini de R ne passant que par des états de contrôle q tels que $e \notin Source(q)$. Comme pour tout état de contrôle q par lequel passe ρ_4, $e \in Mémo(q)$, on en déduit que la transition de mémorisation de e aurait pu être faite dans le premier stable que rencontre[6] ρ_4. Finalement, on obtient qu'il existe une exécution $\rho' \cdot (q';w') \cdot \rho''$ dans R, telle que :

- $(q';w')$ est un état stable tel que $e \notin Source(q)$;
- ρ'' ne passe que par des états de contrôle q tels que $e \notin Source(q)$.

Et on note qu'en vertu des théorèmes 5.1 et 5.2, on peut supposer de plus que ρ' et ρ'' sont des chemins sans mémorisation.

Réciproquement, il est clair que s'il existe une exécution $\rho = \rho' \cdot (q';w') \cdot \rho''$ dans R, telle que :

- $(q';w')$ est un état stable tel que $e \notin Source(q)$;
- ρ'' ne passe que par des états de contrôle q tels que $e \notin Source(q)$;
- ρ' et ρ'' sont sans mémorisation.

alors $\rho' \cdot (q';w') \xrightarrow{\oplus e} (q';w'') \cdot \rho''$ est une exécution vérifiant les trois points de la définition 7.15.

6. on rappelle qu'un chemin de stabilisation est forcément fini (cf. 3.3.2).

On se ramène donc finalement à décider la formule de LTL : $\Diamond(p \wedge \Box q)$ sur le système de transitions paramétré $(\mathrm{RG}^{\ominus\&c}(R), \mu_G)$, avec :

$$\mu_G((q;w)) = \begin{cases} \{p,q\} \text{ si } (q;w) \text{ est stable et } e \notin Source(q) \\ \{p\} \text{ si } (q;w) \text{ est stable et } e \in Source(q) \\ \{q\} \text{ si } (q;w) \text{ est instable et } e \notin Source(q) \\ \emptyset \text{ si } (q;w) \text{ est instable et } e \in Source(q) \end{cases}$$

On note que pour un AFR, une formule de LTL peut exprimer le fait qu'un chemin ne passe que par des états de contrôle q vérifiant $e \notin Source(q)$, mais ne peut pas imposer qu'un chemin passe par un état stable, puisque le paramétrage affecte des propositions atomiques aux états de contrôle. C'est la raison pour laquelle on utilise une fois de plus le graphe d'accessibilité sans mémorisation pour conclure. Le théorème suivant résume les résultats auxquels on a abouti à propos des propriétés de famine :

Théorème 7.7 *Le problème de famine et le problème de famine équitable sont décidables pour les AFRs.*

Conclusion

Nous nous sommes intéressés dans notre stage de DEA à la vérification de propriétés sur les AFRs. Les différents résultats auxquels nous avons abouti montrent qu'en intégrant le fait que la file est potentiellement infinie, la vérification de certaines propriétés sur les AFRs reste décidable.

L'algorithme que nous avons obtenu, calculant une représentation reconnaissable de l'ensemble des états accessibles, permet notamment de décider le RP, le FRSP et le QLP. D'autre part, l'élimination des transitions infranchissables d'un AFR est une amélioration du compilateur Électre qu'il nous semble intéressant d'évaluer. Enfin, les résultats de décidabilité des fragments de logique temporelle LTL et CTL considérés permettent de vérifier des propriétés de sûreté et de vivacité sur les AFRs.

Par ailleurs, nous avons également analysé la détection des séquences infiniment itérables d'un AFR. Nous nous sommes inspirés des résultats existant pour les automtates communicants, dont on trouvera une présentation en annexe A. Nous avons d'une part réussi à adapter le test $\mathcal{U}$, et d'autre part obtenu un algorithme de décision pour le test $\mathcal{V}$.

Nous avons aussi envisagé l'adapatation des résultats obtenus pour les AFRs aux automates communicants. Il est en effet naturel, au vu des résultats précédents, de se poser la question suivante : la classe des automates communicants dont les transitions d'envoi dans un canal sont des boucles est-elle analysable ? Nous nous sommes en fait rendu compte que cette classe est une sous-classe des automates communicants avec pertes et duplications, pour laquelle le RRP reste indécidable.

Nous n'avons pas, dans cette étude, analysé les programmes Électre d'un point de vue temporisé. Des résultats existent déjà dans ce domaine, mais font l'hypothèse que la file est bornée [4]. Nous pensons qu'il serait intéressant d'étudier la temporisation des AFRs en intégrant le fait que la file est potentiellement infinie.

D'une manière plus générale, quels résultats peut-on obtenir sur la temporisation de systèmes de transitions admettant une représentation reconnaissable de leur ensemble d'états accessibles ?

Bibliographie

[1] F. Cassez and O. Roux. Fifo-transition systems to model reactive programs with event memorisation. 1995. submitted.

[2] F. Cassez and O. Roux. Compilation of the ELECTRE reactive language into finite transition systems. *Theoretical Computer Science*, 146, 1995.

[3] P. Argón. Sémantique opérationnelle d'ELECTRE par réécriture : la version 4 du compilateur. Technical report, Laboratoire d'Automatique de Nantes, CNRS (U.A. 823), École Centrale Nantes, Université de Nantes (FRANCE), 1995.

[4] V. Rusu. *Vérification temporelle de programmes* ELECTRE. PhD thesis, Laboratoire d'Automatique de Nantes, Ecole Centrale de Nantes (FRANCE), 1996.

[5] D. Brand and P. Zafiropulo. On communicating finite-state machines. *JACM*, 30(2):323–342, 1983.

[6] A. Finkel and P. McKenzie. Verifying identical communicating processes is undecidable. *Theoretical Computer Science*, 174:217–230, 1997.

[7] G. Cécé, A. Finkel, and S. Purushothaman Iyer. Unreliable channels are easier to verify than perfect channels. *Information and Computation*, 124(1):20–31, 1996.

[8] P. Abdulla and B. Jonsson. Undecidable verification problems for programs with unreliable channels. In *Proc. of ICALP*, volume 820. LNCS, 1994.

[9] G. Cécé and A. Finkel. Programs with quasi-stable channels are effectively recognizable. In *Proc. of* 9^{th} *CAV (June), ISRAEL*, 1997.

[10] A. Finkel, B. Willems, and P. Wolper. A direct symbolic approach to model checking pushdown systems. Technical report, Laboratoire Spécification et Vérification, 1997. submitted.

[11] M.Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *Proc. of the First Symposium on Logic in Computer Science*, pages 322–331, 1986.

[12] B. Boigelot and P. Godefroid. Symbolic verification of communication protocols with infinite state spaces using qdds. In *Proc. of* 8^{th} *CAV (August), USA*, volume 1102, pages 1–12. LNCS, 1996.

[13] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, 1979.

[14] J. Berstel. *Transductions and Context-Free Languages.* B.G. Teubner Stuttgart, 1979.

[15] A. Arnold. *Systèmes de transitions finis et sémantique des processus communiquants.* Masson, 1992.

[16] A. Finkel. Decidability of the termination problem for completely specified protocols. *Distributed Computing*, 7:129–135, 1994.

[17] A. Finkel. Reduction and covering of infinite reachability trees. *Information and Computation*, 89(2):144–170, 1990.

[18] O. Roux, D. Creusot, F. Cassez, and J.P. Elloy. Le langage réactif asynchrone ELECTRE. *Technique et Science Informatiques (TSI)*, 11(5):35–66, 1992.

[19] E.A. Emerson. *Handbook of Theoretical Computer Science*, chapter 16, pages 996–1072. Elsevier Science Publishers, 1990.

[20] M.C. Browne, E.M. Clarke, and O. Grümberg. Characterizing finite kripke structures in propositional temporal logic. *Theoretical Computer Science*, 59:115–131, 1988.

[21] T. Jéron and C. Jard. Testing for unboundedness of fifo channels. *Theoretical Computer Science*, 113:93–117, 1993.

[22] A. Finkel and O. Marcé. A minimal symbolic coverability graph for infinite-state comunicating automata. Technical report, Laboratoire Spécification et Vérification, 1997. submitted.

Annexe A

Séquences infiniment itérables d'un automate communicant

Nous présentons dans cette annexe les résultats concernant la détection de séquences infiniment itérables d'un système de CFSMs [21, 22]. Pour simplifier la présentation, nous considérons ici des automates communicants, mais les notions exposées se généralisent aux systèmes de CFSMs quelconques. Définissons la notion de séquence infiniment itérable :

Définition A.1 *Soit* $S = (Q, q_0, \{+,-\} \times \Sigma \times \{1\}, \delta, 1)$ *un automate communicant. Si* $(q;w) \xrightarrow{\sigma} (q';w')$ *est un chemin de* S*, alors on dit que la séquence* σ *est* infiniment itérable *ssi les deux conditions suivantes sont vérifiées :*

- $q = q'$;
- $\forall\, n \in \mathbb{N}^*, (q;w) \xrightarrow{\sigma^n}$.

Dans la suite, si σ est une séquence d'un automate communicant, on note $\mathrm{in}(\sigma)$ (resp. $\mathrm{out}(\sigma)$) la projection $p_1^+(\sigma)$ (resp. $p_1^-(\sigma)$) de σ sur les réceptions (resp. émissions). On remarque qu'une séquence σ infiniment itérable d'un automate communicant vérifie : $|\mathrm{out}(\sigma)| \geq |\mathrm{in}(\sigma)|$.

A.1 Le test $\mathcal{U}$

Le test $\mathcal{U}$ a été introduit dans [21] afin d'avoir une condition suffisante pour qu'un automate communicant S ait un ensemble d'états accessibles infini, i.e. que son canal ne soit pas borné. Rappelons que le FRSP est indécidable pour la classe des automates communicants.

L'idée des auteurs dans [21] était la suivante : on développe l'arbre d'accessibilité, et lorsqu'on retrouve le même état local q sur une branche : $(q, w_1) \xrightarrow{\sigma} (q; w_2)$, on teste si la séquence σ est infiniment itérable, à l'aide du test $\mathcal{U}$. Si on obtient que σ est infiniment itérable, et si de plus σ vérifie $|\mathrm{out}(\sigma)| > |\mathrm{in}(\sigma)|$, alors il est clair que le canal est non borné.

Donnons à présent la définition du test $\mathcal{U}$:

Définition A.2 *Soit $S = (Q, q_0, \{+,-\} \times \Sigma \times \{1\}, \delta, 1)$ un automate communicant. L'ensemble $\mathcal{U}^S \subseteq (Q \times \Sigma^*) \times (\{+,-\} \times \Sigma^*) \times (Q \times \Sigma^*)$ est défini par: $((q;w),\sigma,(q';w')) \in \mathcal{U}^S$ ssi les trois conditions suivantes sont satisfaites:*

- $(q;w) \xrightarrow{\sigma} (q';w')$ *un chemin de S ;*
- $q = q'$ *;*
- $w \cdot \mathrm{out}(\sigma) \leq w' \cdot \mathrm{out}(\sigma)$.

On va voir à présent que le test $\mathcal{U}$ est une condition suffisante pour qu'une séquence d'un automate communicant soit infiniment itérable [21]. Soit S un automate communicant et $((q;w),\sigma,(q';w')) \in \mathcal{U}^S$. Puisque $w \cdot \mathrm{out}(\sigma) \leq w' \cdot \mathrm{out}(\sigma)$, on a $w \leq w'$. On note $u = w' - w$, et on vérifie que u et $\mathrm{out}(\sigma)$ commutent[1]. Ainsi, $w' = w \cdot \mathrm{out}(\sigma) \cdot u$, et puisque σ est franchissable à partir de $(q;w)$, on montre par récurrence que pour tout $n \in \mathbb{N}^*$, $(q;w) \xrightarrow{\sigma^n} (q; w \cdot (\mathrm{out}(\sigma))^n \cdot u)$. Par conséquent, σ est infiniment itérable.

Théorème A.1 (Jéron & Jard) *Pour tout automate communicant S, pour tout $(s,\sigma,s') \in \mathcal{U}^S$, σ est infiniment itérable.*

A.2 Le test $\mathcal{V}$

Le test $\mathcal{V}$, introduit dans [22], est une généralisation du test $\mathcal{U}$ qui permet d'obtenir un semi-algorithme calculant un graphe de couverture symbolique pour les automates communicants. Le test $\mathcal{V}$ est défini comme la condition nécessaire suffisante pour qu'une séquence soit infiniment itérable. Formellement :

Définition A.3 *Soit $S = (Q, q_0, \{+,-\} \times \Sigma \times \{1\}, \delta, 1)$ un automate communicant. L'ensemble $\mathcal{V}^S \subseteq (Q \times \Sigma^*) \times (\{+,-\} \times Sigma^*) \times (Q \times \Sigma^*)$ est défini par: $((q;w),\sigma,(q';w')) \in \mathcal{V}^S$ ssi les trois conditions suivantes sont satisfaites:*

- $(q;w) \xrightarrow{\sigma} (q';w')$ *un chemin de S ;*
- $q = q'$ *;*
- $\forall\, n \in \mathbb{N}^*, (q;w) \xrightarrow{\sigma^n}$.

On remarque, d'après le théorème A.1, que pour tout automate communicant S, les ensembles $\mathcal{U}^S$ et $\mathcal{V}^S$ vérifient : $\mathcal{U}^S \subseteq \mathcal{V}^S$. En fait, l'inclusion inverse est également vraie :

Théorème A.2 (Finkel & Marcé [22]) *Pour tout automate communicant S, on a : $\mathcal{U}^S = \mathcal{V}^S$.*

1. comme $w \cdot \mathrm{out}(\sigma) \leq w' \cdot \mathrm{out}(\sigma)$, il vient $\mathrm{out}(\sigma) \leq u \cdot \mathrm{out}(\sigma)$. Soit $u' = u \cdot \mathrm{out}(\sigma) - \mathrm{out}(\sigma)$. On a : $|u| = |u'| = |\mathrm{out}(\sigma)| - |\mathrm{in}(\sigma)| \leq |\mathrm{out}(\sigma)|$. Or $wu = w' = w \cdot \mathrm{out}(\sigma) - \mathrm{in}(\sigma)$, donc u est un facteur droit de $w \cdot \mathrm{out}(\sigma)$, soit un facteur droit de $\mathrm{out}(\sigma)$, puisque $|u| \leq |\mathrm{out}(\sigma)|$. Comme d'une part u est de même taille que u', et d'autre part u' est un facteur droit de $\mathrm{out}(\sigma)$, on en déduit $u = u'$, et finalement $u \cdot \mathrm{out}(\sigma) = \mathrm{out}(\sigma) \cdot u$.

La preuve, qui est assez technique (elle est donnée en détail dans [22]), est basée sur la remarque suivante : si S est un automate communicant, et si $((q;w),\sigma,(q';w')) \in \mathcal{V}^S$, alors $\mathrm{in}(\sigma)^\omega \leq w \cdot \mathrm{out}(\sigma)^\omega$.

On dispose donc d'une condition nécessaire, suffisante et décidable pour détecter les séquences infiniment itérables d'un automate communicant.

Annexe B

Le fichier .tef de l'exemple des lecteurs-écrivains

```
**       ELECTRE compiler V5beta - Mar. 1997 - LAN/ECN        **
** Laboratoire d'Automatique de Nantes  (U.A. C.N.R.S. 823)   **
**         1, rue de la noe NANTES CEDEX 03 FRANCE            **
**                 electre@lan.ec-nantes.fr                   **

 Programme ELECTRE de depart :
[1/{{@l1:LIRE1|||@l2:LIRE2}|{#e1:ECRIRE|#e2:ECRIRE}}]*.

 Traduction en langage objet :
[1/{{@l1:LIRE1|||@l2:LIRE2}|{#e1:ECRIRE~#e1|#e2:ECRIRE~#e2}}]*([1/
          {{@l1:LIRE1|||@l2:LIRE2}|{#e1:ECRIRE~#e1|#e2:ECRIRE~#e2}}]).

Liste des modules :
  LIRE2 LIRE1 ECRIRE  .

Liste des evenements :
l2 l1 endLIRE2 endLIRE1 endECRIRE e2 e1.

Liste des evenements fugaces :
l2 l1.

Liste des evenements a memorisation simple :
  .

Liste des evenements a memorisation multiple :
e2 e1.

AUTOMATE CORRESPONDANT:
St&0 : A=[] D=[] F=[]

St0-&->St&0:A=[] D=[] F=[] ""
St&0-e1->St1:A=[ECRIRE] D=[] F=[] ""
St0-&e1->St1:A=[ECRIRE] D=[] F=[] "-e1"
St&0-e2->St2:A=[ECRIRE] D=[] F=[] ""
St0-&e2->St2:A=[ECRIRE] D=[] F=[] "-e2"
```

```
St&0-l1->St3:A=[LIRE1] D=[] F=[] ""
St&0-l2->St4:A=[LIRE2] D=[] F=[] ""

St4-&->St&4:A=[LIRE2] D=[] F=[] ""
St&4-e1->St&4:A=[LIRE2] D=[] F=[] "*e1"
St&4-e2->St&4:A=[LIRE2] D=[] F=[] "*e2"
St&4-endLIRE2->St0:A=[] D=[] F=[LIRE2] ""
St&4-l1->St5:A=[LIRE2 LIRE1] D=[] F=[] ""

St5-&->St&5:A=[LIRE2 LIRE1] D=[] F=[] ""
St&5-e1->St&5:A=[LIRE2 LIRE1] D=[] F=[] "*e1"
St&5-e2->St&5:A=[LIRE2 LIRE1] D=[] F=[] "*e2"
St&5-endLIRE1->St6:A=[LIRE2] D=[] F=[LIRE1] ""
St&5-endLIRE2->St7:A=[LIRE1] D=[] F=[LIRE2] ""

St7-&->St&7:A=[LIRE1] D=[] F=[] ""
St&7-e1->St&7:A=[LIRE1] D=[] F=[] "*e1"
St&7-e2->St&7:A=[LIRE1] D=[] F=[] "*e2"
St&7-endLIRE1->St0:A=[] D=[] F=[LIRE1] ""

St6-&->St&6:A=[LIRE2] D=[] F=[] ""
St&6-e1->St&6:A=[LIRE2] D=[] F=[] "*e1"
St&6-e2->St&6:A=[LIRE2] D=[] F=[] "*e2"
St&6-endLIRE2->St0:A=[] D=[] F=[LIRE2] ""

St3-&->St&3:A=[LIRE1] D=[] F=[] ""
St&3-e1->St&3:A=[LIRE1] D=[] F=[] "*e1"
St&3-e2->St&3:A=[LIRE1] D=[] F=[] "*e2"
St&3-endLIRE1->St0:A=[] D=[] F=[LIRE1] ""
St&3-l2->St8:A=[LIRE2 LIRE1] D=[] F=[] ""

St8-&->St&8:A=[LIRE2 LIRE1] D=[] F=[] ""
St&8-e1->St&8:A=[LIRE2 LIRE1] D=[] F=[] "*e1"
St&8-e2->St&8:A=[LIRE2 LIRE1] D=[] F=[] "*e2"
St&8-endLIRE1->St6:A=[LIRE2] D=[] F=[LIRE1] ""
St&8-endLIRE2->St7:A=[LIRE1] D=[] F=[LIRE2] ""

St2-&->St&2:A=[ECRIRE] D=[] F=[] ""
St&2-e1->St&2:A=[ECRIRE] D=[] F=[] "*e1"
St&2-e2->St&2:A=[ECRIRE] D=[] F=[] "*e2"
St&2-endECRIRE->St0:A=[] D=[] F=[ECRIRE] ""

St1-&->St&1:A=[ECRIRE] D=[] F=[] ""
St&1-e1->St&1:A=[ECRIRE] D=[] F=[] "*e1"
St&1-e2->St&1:A=[ECRIRE] D=[] F=[] "*e2"
St&1-endECRIRE->St0:A=[] D=[] F=[ECRIRE] ""

 nombre d'etats: 9 nombre de transitions:43

 etat initial = St0 ; etat terminal = {}
```

Annexe C

Les fichiers .tef et .esr des exemples de l'implémentation

C.1 Le programme QLP_1

C.1.1 Le fichier QLP1.tef

```
**       ELECTRE compiler V5beta - Mar. 1997 - LAN/ECN        **
** Laboratoire d'Automatique de Nantes  (U.A. C.N.R.S. 823)   **
**         1, rue de la noe NANTES CEDEX 03 FRANCE            **
**                 electre@lan.ec-nantes.fr                   **

 Programme ELECTRE de depart :
A[B^{{#e}|{@f:C^{#e}}}]*.

 Traduction en langage objet :
A;[B^{{#e}|{@f:C^{#e}}}]*([B^{{#e}|{@f:C^{#e}}}]).

Liste des modules :
 C B A .

Liste des evenements :
f endC endB endA e.

Liste des evenements fugaces :
f.

Liste des evenements a memorisation simple :
  .

Liste des evenements a memorisation multiple :
e.

AUTOMATE CORRESPONDANT:
St&0 : A=[A] D=[A] F=[]

St0-&->St&0:A=[A] D=[] F=[] ""
St&0-e->St&0:A=[A] D=[] F=[] "*e"
```

```
St&0-endA->St1:A=[B] D=[] F=[A] ""

St1-&->St&1:A=[B] D=[] F=[] ""
St&1-e->St1:A=[B] D=[] F=[] ""
St1-&e->St1:A=[B] D=[] F=[] "-e"
St&1-endB->St1:A=[B] D=[] F=[B] ""
St&1-f->St2:A=[C] D=[] F=[] ""

St2-&->St&2:A=[C] D=[] F=[] ""
St&2-e->St1:A=[B] D=[] F=[] ""
St2-&e->St1:A=[B] D=[] F=[] "-e"
St&2-endC->St1:A=[B] D=[] F=[C] ""

 nombre d'etats: 3 nombre de transitions:12

 etat initial = St0 ; etat terminal = {}
```

C.1.2 Le fichier QLP1.esr

```
SymbolicRepresentation :

 St0 : { { e } }
 St1 : { { e } }
 St2 : { { } }
```

C.2 Le programme QLP_2

C.2.1 Le fichier QLP2.tef

```
**       ELECTRE compiler V5beta - Mar. 1997 - LAN/ECN        **
** Laboratoire d'Automatique de Nantes  (U.A. C.N.R.S. 823)   **
**         1, rue de la noe NANTES CEDEX 03 FRANCE            **
**                 electre@lan.ec-nantes.fr                   **

 Programme ELECTRE de depart :
A[B^{e:C^{e}}]*.

 Traduction en langage objet :
A;[B^{e:C~e^{e}}]*([B^{e:C~e^{e}}]).

Liste des modules :
C B A .

Liste des evenements :
endC endB endA e.

Liste des evenements fugaces :
.

Liste des evenements a memorisation simple :
   e.
```

```
Liste des evenements a memorisation multiple :
.

AUTOMATE CORRESPONDANT:
St&0 : A=[A] D=[A] F=[]

St0-&->St&0:A=[A] D=[] F=[] ""
St&0-e->St&0:A=[A] D=[] F=[] "+e"
St&0-endA->St1:A=[B] D=[] F=[A] ""

St1-&->St&1:A=[B] D=[] F=[] ""
St&1-e->St2:A=[C] D=[] F=[] ""
St1-&e->St2:A=[C] D=[] F=[] "-e"
St&1-endB->St1:A=[B] D=[] F=[B] ""

St2-&->St&2:A=[C] D=[] F=[] ""
St&2-e->St1:A=[B] D=[] F=[] ""
St2-&e->St1:A=[B] D=[] F=[] "-e"
St&2-endC->St1:A=[B] D=[] F=[C] ""

 nombre d'etats: 3 nombre de transitions:11

 etat initial = St0 ; etat terminal = {}
```

C.2.2 Le fichier QLP2.esr

```
SymbolicRepresentation :

 St0 : { { e } }
 St1 : { { e } }
 St2 : { { } }
```

C.3 Le fichier IMPL.tef

```
**       ELECTRE compiler V5beta - Mar. 1997 - LAN/ECN        **
** Laboratoire d'Automatique de Nantes  (U.A. C.N.R.S. 823)   **
**         1, rue de la noe NANTES CEDEX 03 FRANCE            **
**                 electre@lan.ec-nantes.fr                   **

 Programme ELECTRE de depart :
[[1/{l1:!LIRE1|||#l2:LIRE2}]^{e:ECRIRE}]*.

 Traduction en langage objet :
[[1/{l1:!LIRE1~l1|||#l2:LIRE2~#l2}]^{e:ECRIRE~e}]*([[1/
                       {l1:!LIRE1~l1|||#l2:LIRE2~#l2}]^{e:ECRIRE~e}]).

Liste des modules :
  LIRE2 LIRE1 ECRIRE .

Liste des evenements :
l2 l1 endLIRE2 endLIRE1 endECRIRE e.
```

```
Liste des evenements fugaces :
.

Liste des evenements a memorisation simple :
l1    e.

Liste des evenements a memorisation multiple :
l2.

AUTOMATE CORRESPONDANT:
St&0 : A=[] D=[] F=[]

St0-&->St&0:A=[] D=[] F=[] ""
St&0-e->St1:A=[ECRIRE] D=[] F=[] ""
St0-&e->St1:A=[ECRIRE] D=[] F=[] "-e"
St&0-l1->St2:A=[LIRE1] D=[] F=[] ""
St0-&l1->St2:A=[LIRE1] D=[] F=[] "-l1"
St&0-l2->St3:A=[LIRE2] D=[] F=[] ""
St0-&l2->St3:A=[LIRE2] D=[] F=[] "-l2"

St3-&->St&3:A=[LIRE2] D=[] F=[] ""
St&3-l2->St&3:A=[LIRE2] D=[] F=[] "*l2"
St&3-e->St1:A=[ECRIRE] D=[] F=[] "*l2"
St3-&e->St1:A=[ECRIRE] D=[] F=[] "-e*l2"
St&3-endLIRE2->St0:A=[] D=[] F=[LIRE2] ""
St&3-l1->St4:A=[LIRE2 LIRE1] D=[] F=[] ""
St3-&l1->St4:A=[LIRE2 LIRE1] D=[] F=[] "-l1"

St4-&->St&4:A=[LIRE2 LIRE1] D=[] F=[] ""
St&4-l2->St&4:A=[LIRE2 LIRE1] D=[] F=[] "*l2"
St&4-e->St5:A=[LIRE1] D=[] F=[] ""
St4-&e->St5:A=[LIRE1] D=[] F=[] "-e"
St&4-endLIRE1->St6:A=[LIRE2] D=[] F=[LIRE1] ""
St&4-endLIRE2->St7:A=[LIRE1] D=[] F=[LIRE2] ""

St7-&->St&7:A=[LIRE1] D=[] F=[] ""
St&7-l2->St&7:A=[LIRE1] D=[] F=[] "*l2"
St&7-e->St5:A=[LIRE1] D=[] F=[] ""
St7-&e->St5:A=[LIRE1] D=[] F=[] "-e"
St&7-endLIRE1->St0:A=[] D=[] F=[LIRE1] ""

St6-&->St&6:A=[LIRE2] D=[] F=[] ""
St&6-l1->St&6:A=[LIRE2] D=[] F=[] "+l1"
St&6-l2->St&6:A=[LIRE2] D=[] F=[] "*l2"
St&6-e->St1:A=[ECRIRE] D=[] F=[] "*l2"
St6-&e->St1:A=[ECRIRE] D=[] F=[] "-e*l2"
St&6-endLIRE2->St0:A=[] D=[] F=[LIRE2] ""

St5-&->St&5:A=[LIRE1] D=[] F=[] ""
St&5-e->St&5:A=[LIRE1] D=[] F=[] "+e"
St&5-l2->St&5:A=[LIRE1] D=[] F=[] "*l2"
St&5-endLIRE1->St1:A=[ECRIRE] D=[] F=[LIRE1] ""
```

```
St2-&->St&2:A=[LIRE1] D=[] F=[] ""
St&2-e->St5:A=[LIRE1] D=[] F=[] ""
St2-&e->St5:A=[LIRE1] D=[] F=[] "-e"
St&2-endLIRE1->St0:A=[] D=[] F=[LIRE1] ""
St&2-l2->St8:A=[LIRE2 LIRE1] D=[] F=[] ""
St2-&l2->St8:A=[LIRE2 LIRE1] D=[] F=[] "-l2"

St8-&->St&8:A=[LIRE2 LIRE1] D=[] F=[] ""
St&8-l2->St&8:A=[LIRE2 LIRE1] D=[] F=[] "*l2"
St&8-e->St5:A=[LIRE1] D=[] F=[] ""
St8-&e->St5:A=[LIRE1] D=[] F=[] "-e"
St&8-endLIRE1->St6:A=[LIRE2] D=[] F=[LIRE1] ""
St&8-endLIRE2->St7:A=[LIRE1] D=[] F=[LIRE2] ""

St1-&->St&1:A=[ECRIRE] D=[] F=[] ""
St&1-l1->St&1:A=[ECRIRE] D=[] F=[] "+l1"
St&1-l2->St&1:A=[ECRIRE] D=[] F=[] "*l2"
St&1-endECRIRE->St0:A=[] D=[] F=[ECRIRE] ""

 nombre d'etats: 9 nombre de transitions:51

 etat initial = St0 ; etat terminal = {}
```