ÉCOLE NORMALE SUPÉRIEURE DE CACHAN

THÈSE

présentée
pour obtenir

LE GRADE DE DOCTEUR EN SCIENCES
DE L'ÉCOLE NORMALE SUPÉRIEURE DE CACHAN
Spécialité INFORMATIQUE

PAR

Gérard CÉCÉ

Vérification, analyse et approximations symboliques des automates communicants

Soutenue le 22 Janvier 1998 devant le jury composé de :

MM.	André ARNOLD	Président
	Ahmed BOUAJJANI	Rapporteur
	Pierre WOLPER	Rapporteur
	Alain FINKEL	Directeur de thèse
	Thierry JÉRON	Examinateur
	Olivier ROUX	Examinateur

Laboratoire **S**pécification et **V**érification – URA 2236 CNRS;
ENS de Cachan; 61, Avenue du Président Wilson; 94235 CACHAN Cedex

Remerciements

Je remercie André ARNOLD, Thierry JÉRON et Olivier ROUX de m'avoir fait l'honneur de participer à ce jury.

Je remercie tout particulièrement Ahmed BOUAJJANI et Pierre WOLPER pour l'intérêt qu'ils ont porté à mon travail, et pour avoir accepté d'être les rapporteurs d'un manuscrit rédigé bien trop rapidement.

J'exprime ici ma profonde reconnaissance à Alain FINKEL pour m'avoir soutenu et guidé à travers les multiples facettes de l'une de ses passions : la Recherche.

Je remercie les anciens membres du LIFAC et ceux du LSV, qui lui a succédé, pour leur amitié et les années stimulantes que j'ai passées en leur compagnie. Merci en particulier à Ramâk, Catherine, Olivier, Serge, ...

Merci enfin à Christel et Dominique pour le temps qu'elles ont passé à relire ce manuscrit à la recherche de mes innombrables fautes de «frappes»...

À Claude et Cathy,
ainsi qu'à mes parents
et amis

Table des matières

Chapitre 1

Introduction

L'informatique est maintenant devenue incontournable pour de nombreux secteurs d'activité. Parmi ceux-ci, il y a bien sûr, et au premier chef, les télécommunications, mais aussi l'industrie, la finance et la médecine. Mais qu'il s'agisse du système de contrôle d'une centrale nucléaire ou d'un système de transactions boursières, une erreur de conception peut avoir des conséquences financières et humaines considérables. Ainsi, l'informatisation n'apporte un progrès réel que si elle est fiable. Il est donc nécessaire et crucial de s'assurer, au préalable, du bon fonctionnement des logiciels impliqués. Or, les algorithmes utilisés dans les systèmes actuels, sont de plus en plus complexes. Il est devenu indispensable que l'expert chargé de la validation de systèmes critiques soit secondé par des outils de vérification automatique ou quasi-automatique.

Une première difficulté surgit alors. L'étape préliminaire à la vérification automatique, et donc formelle, nécessite de modéliser à la fois le système à vérifier et les critères de bon fonctionnement de ce système. Bien entendu, les résultats de l'étape de vérification ne seront pertinents que si cette modélisation est «correcte». Malheureusement, cette notion de correction, étant à l'interface du «monde réel» et du «monde formel», n'est pas elle-même formalisable ! et ne peut donc pas être vérifiée. Tout de même, un degré de confiance peut être accordé à une modélisation, au vu de la méthodologie employée pour y aboutir [1].

L'obtention d'une spécification sûre passe par l'utilisation d'un modèle fournissant des possibilités de descriptions adaptées au système à étudier. Ainsi, on peut associer aux systèmes temps réel, le modèle des automates temporisés ou encore des systèmes hybrides ; aux systèmes impliquant de l'algorithmique distribuée ou répartie, le modèle des réseaux de Petri ou des automates com-

1. Le lecteur intéressé pourra consulter, par exemple, le projet scientifique du Laboratoire Spécification et Vérification sur ce thème : `http://www.lsv.ens-cachan.fr`.

municants. De façon duale, l'expression des propriétés que doit satisfaire un système passe par l'utilisation de logiques, et notamment les logiques temporelles telles CTL ou LTL.

La recherche en vérification automatique consiste alors à mettre en œuvre des algorithmes de vérifications tirant profit des particularités théoriques de chaque modèle, mais aussi des caractéristiques des systèmes réels qu'il peut modéliser.

C'est dans ce cadre, et pour le modèle des automates communicants, que se situe la contribution de cette thèse.

État de l'art

Le modèle des *automates communicants*, [Boc78], se compose d'un ensemble d'automates finis, représentant chacun un processus (i.e. un programme), et d'un ensemble de canaux, de tailles non bornées, à travers lesquels ces processus communiquent en s'envoyant des messages. Il est utilisé de façon plus ou moins explicite dans les langages de spécification d'algorithmes répartis comme ESTELLE ou SDL [Tur93].

Malheureusement, le modèle des automates communicants a la puissance des machines de Turing [BZ83], et cette puissance d'expressivité se paye par une impossibilité d'analyse. Il ne peut exister d'algorithmes permettant de vérifier, pour n'importe quel système d'automates communicants, telle ou telle propriété non triviale.

L'une des directions d'étude de ce modèle consiste alors à restreindre son expressivité pour gagner en possibilités d'analyse.

La contrainte la plus immédiate qui vient à l'esprit consiste à borner la taille des canaux de communication. Ce faisant, l'ensemble d'accessibilité du système ne peut plus être infini et quasiment toutes les propriétés auxquelles on s'intéresse deviennent décidables. Mais le problème est alors reporté sur le choix d'une «bonne» borne.

L'essentiel des résultats de décidabilité des dix premières années d'étude de ce modèle a porté sur la détection du caractère borné ou non des canaux d'un système [RG84, Fin86, GER87, FC87, Jér91].

Le problème de l'accessibilité [2] pour les systèmes d'automates communicants est abordé dans [Fin86] et [FC87] où il est résolu pour les réseaux à files *monogènes* ou *linéaires*. En fait, ce problème de l'accessibilité a eu un regain d'intérêt lorsqu'il a été montré décidable pour les systèmes pouvant perdre des messages, les *lossy systems* [AJ93]. Ce dernier résultat est d'autant plus intéressant qu'un article récent [IN97] montre que les pertes de messages peuvent être contrôlées par des probabilités, ce qui permet de mieux modéliser la réalité : tout message *peut* être perdu mais est *rarement* perdu.

Il est à noter, pour cette classe de systèmes pouvant perdre les messages que, contrairement aux réseaux de Petri, c'est le problème de l'accessibilité qui est plus «facile» que celui de la finitude de l'ensemble des états accessibles. Ce dernier problème est en effet conjecturé indécidable [CFP96].

Une autre direction d'étude du modèle des automates communicants consiste à garder sa puissance de description et à mettre en œuvre des algorithmes de vérification qui marcheront dans certains cas, mais sans que l'on puisse savoir a priori lesquels. Le pari qui est tenté est de considérer que la puissance de description du modèle n'est pas pleinement utilisée dans les applications pratiques. Par conséquent, on peut espérer que ces *semi-algorithmes* mis en œuvre seront «souvent» efficaces.

Jéron a ainsi défini un test détectant le caractère infini de l'ensemble des états accessibles [Jér91]. Peng et Purushothaman ont eux défini un semi-algorithme mettant en évidence l'absence de réceptions non spécifiées ou d'états de blocage [PP91].

Toujours dans le cadre de la mise en œuvre de semi-algorithmes, une approche intéressante a été initiée simultanément par Finkel et Marcé [FM96], d'une part, et Boigelot et Godefroid [BG96], d'autre part. Il s'agit d'approximer, de l'«intérieur», l'ensemble des états accessibles d'un système, avec la préoccupation constante de générer une infinité d'états dès que l'occasion se présente. Les résultats de [BG96] ont été récemment améliorés par [BGWW97] et [BH97]. Nous y reviendrons au Chapitre 5, lorsque nous proposerons nos résultats sur ce thème.

2. Ce problème est le suivant : étant donné un état global du système, existe-t-il une exécution y aboutissant.

Plan de la thèse

Au Chapitre 2, nous introduisons et définissons le modèle des automates communicants, ainsi que les outils mathématiques de base et les diverses notations qui seront utilisés tout au long de cette thèse.

Au Chapitre 3, nous introduisons le concept de systèmes utilisant des canaux non fiables. Outre le cas des *lossy systems* [AJ93], nous étudions deux autres types de défaillances, l'insertion et la duplication de messages, ainsi que les combinaisons de ces deux erreurs avec la perte de messages. Nos principaux résultats sont les suivants.

- Bien que l'ensemble d'accessibilité d'un *lossy-system* soit reconnaissable, il est impossible d'en calculer une représentation.
- Les erreurs d'insertion entraînent une plus forte diminution de la puissance du modèle que les erreurs de perte des messages. Et on peut calculer une représentation de l'ensemble d'accessibilité qui est reconnaissable.
- Les erreurs de duplication n'entraînent, elles, aucune perte de puissance du modèle. L'ensemble d'accessibilité reste non récursif. De même, lorsque les duplications de messages sont combinées avec des pertes de messages, les résultats de décidabilité sont les mêmes que s'il n'y avait eu que des pertes de messages. Dans ces deux cas, erreur de duplication et erreur de duplication + erreur de perte des messages, nous montrons que les duplications sont *récupérables*. Autrement dit, on peut «corriger» cette erreur.

Enfin, nous présentons une preuve du problème de l'accessibilité pour les *front lossy systems* (où les pertes ne peuvent avoir lieu qu'en tête des canaux). Ce problème était ouvert depuis [Fin94].

Au Chapitre 4, nous étudions la communication *half-duplex* entre deux automates. Ce mode exige qu'à tout moment, au moins un des canaux soit vide, c'est-à-dire, que les transmissions s'effectuent dans un seul sens à la fois (contrairement au mode *full-duplex*).

Nous montrons que l'hypothèse de communication half-duplex entre deux automates permet de calculer une représentation de l'ensemble, qui est alors reconnaissable, des états accessibles du système, ce qui permet de décider de la plupart des problèmes de vérification tels que l'accessibilité ou la finitude de l'ensemble d'accessibilité.

Par contre, nous montrons qu'un système composé de trois automates, tels que chaque paire d'automates soit half-duplex, peut simuler une machine de

Turing ; ce qui nous amène à proposer une «bonne» généralisation : les *automates à communication quasi-stable*. Cette nouvelle classe permet de considérer des systèmes de plus de deux automates tout en préservant les bonnes propriétés de vérification des systèmes half-duplex à deux automates.

Au Chapitre 5, nous revenons sur l'idée de [FM96] et [BG96] d'approximer et, dans certains cas, de calculer exactement, l'ensemble d'accessibilité d'un automate à files[3]. Cette approche consiste à définir, et à utiliser, des transitions particulières appelées *transitions symboliques*. Nous commençons par définir cette notion de transition symbolique et son dual, l'*état symbolique*, qui permet de représenter les états générés par les transitions symboliques.

Nous verrons qu'un premier type de transitions symboliques est décrit par les *langages d'actions de communication*, c'est-à-dire, des ensembles de séquences d'émissions et de réceptions. Puis nous caractériserons les conditions que doivent satisfaire les *méthodes de représentation* des ensembles d'états générés par les transitions symboliques, et ce, indépendamment des particularités de celles-ci. L'énoncé de la généralisation d'un théorème de [Pac87] nous permettra de conclure à l'optimalité de ces conditions.

Nous proposons alors une nouvelle classe de transitions symboliques, les *communications à émissions non consommées*. Cette classe est intéressante à plus d'un titre. Elle englobe les transitions symboliques étudiées dans [BG96] et permet d'améliorer des résultats précédents de décidabilité portant sur les systèmes d'automates disposés en anneau [RG84, PP92]. Nous proposons enfin une classe de transitions symboliques améliorant celles de [BGWW97].

Le Chapitre 6 conclut cette étude et propose quelques perspectives.

3. Un automate à files est une représentation compacte d'un système d'automates communicants (cf. Chapitre 2).

Chapitre 2

Préliminaires

2.1 Exemple de protocole

Nous pouvons considérer un protocole de communication comme un ensemble de procédures permettant à deux machines d'échanger des informations dans de «bonnes conditions», c'est-à-dire, en corrigeant les diverses erreurs qui peuvent être provoquées par les imperfections physiques des canaux de communication.

À titre d'exemple, considérons le protocole suivant énoncé dans un pseudo langage de spécification.

Émetteur	Récepteur
Tant que message à émettre	Tant que réception (*start*)
mot := nouveau message;	recevoir (mot);
émettre (*start*);	recevoir (*end*);
émettre (mot);	émettre (*accusé_R*);
émettre (*end*);	Fin Tant que;
recevoir (*accusé_R*);	
Fin Tant que;	

Le modèle des automates communicants consiste à abstraire, par un automate fini, le flot des actions de communication de chacun des processus du système. Ainsi le protocole précédent est modélisé par le système à deux automates de la Figure 2.1.

Chaque automate est constitué d'un ensemble d'*états locaux* dont l'un, l'*état initial*, est mis en évidence. Par convention, nous indiquons l'état initial par une courte flèche sans origine. Ainsi, celui de l'automate A_1 de la Figure 2.1 est 1. Entre les états, il y a des transitions étiquetées par des actions de

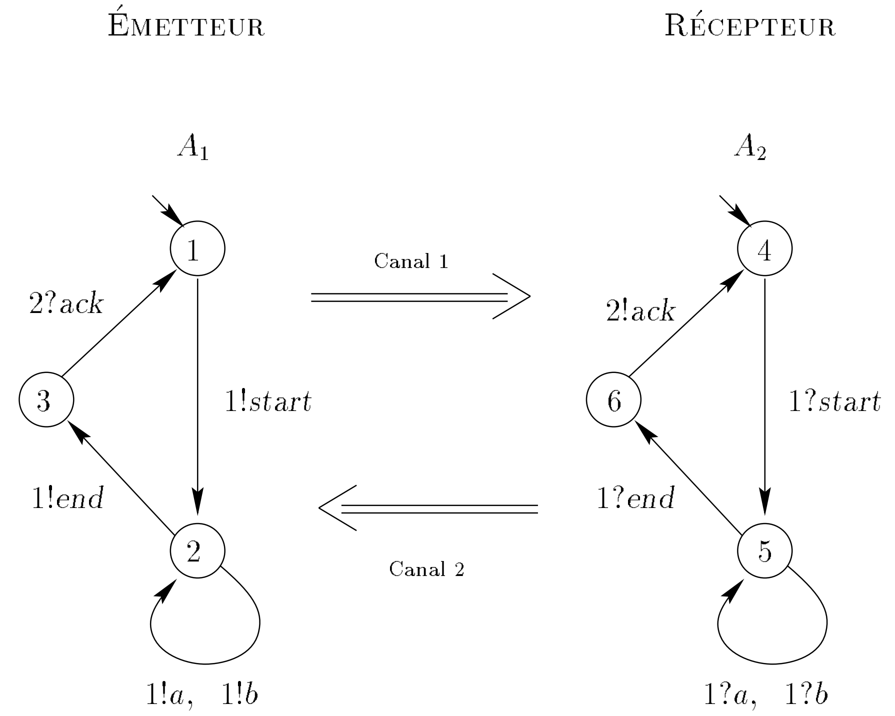


FIG. 2.1 – *Exemple d'un système de deux automates communicants*

communication, c'est-à-dire soit par une émission soit par une réception. Une émission est notée $c!a$ où c est le numéro du canal où se fera l'émission de la lettre a. La notation d'une réception est similaire sauf qu'un « ? » remplace le « ! ». La boucle étiquetée par « $1!a$, $1!b$ » remplace deux boucles, dont la première serait étiquetée par $1!a$ et la deuxième par $1!b$. Cette boucle signifie ici que le mot envoyé par l'émetteur est codé sur l'*alphabet* $\{a, b\}$. Les deux doubles flèches représentent le sens des deux canaux. Ces canaux sont *fifo*, c'est-à-dire qu'ils transmettent les messages dans leur ordre d'arrivée.

La *sémantique* (i.e. le fonctionnement) d'un système d'automates communicants est le passage, dans l'un des automates, d'un état local à un autre en effectuant l'action qui étiquette la transition utilisée. Par exemple, l'automate A_1 peut passer de son état local 1 à son état local 2 en émettant le symbole *start* dans le canal 1. Le franchissement des transitions dans l'un des automates s'effectue indépendamment du franchissement des transitions de l'autre automate. Ces deux automates fonctionnent ainsi en *parallèle*. Il y a tout de même une contrainte naturelle : une réception ne peut avoir lieu que si le message correspondant existe dans le canal où elle s'effectue !

Pour examiner le comportement d'un système d'automates communicants, nous devons pouvoir décrire, précisément, son *état global* à un moment donné, c'est-à-dire, l'état local courant de chacun des automates ainsi que le contenu des canaux. Le contenu de chaque canal est représenté par un mot sur l'alphabet des symboles (i.e. des messages) pouvant transiter par les canaux.

Par exemple, considérons une exécution où l'automate A_1 émet le message *start*, puis deux messages a et un message b, et où l'automate A_2 reçoit le message *start*. Cette exécution aboutit à l'état global $(2, 5; aab, \lambda)$, le symbole λ représentant le mot *vide*.

Dans ce qui suit, nous définissons le modèle ainsi que les notions dont nous aurons besoin pour pouvoir étudier : les différents comportements possibles d'un système ; l'ensemble de ses états accessibles, est-il fini ou infini ; les propriétés de ses exécutions, sont-elles finies ou infinies, correspondent-elles à ce à quoi on s'attend etc...

2.2 Automates finis et ensembles associés

Pour analyser les systèmes d'automates communicants, nous aurons besoin de représenter un ensemble, éventuellement infini, de contenus de canaux. Cette section présente à cette fin le modèle des automates finis. Ce modèle permet de représenter un certain type d'ensembles : les *ensembles rationnels*.

Un *alphabet* est un ensemble d'éléments appelés *lettres*. Un *mot* x sur un alphabet $\Sigma = \{a_1, \ldots, a_n\}$ est une séquence de lettres de cet alphabet. Par exemple, $x = informatique$ est un mot sur l'alphabet {a,e,f,i,m,n,o,q,r,t,u}. Étant donnés deux mots x et y, nous notons $x.y$ ou xy leur concaténation. Par exemple, si $x = informatique$ et $y = repartie$, alors $x.y = informatiquerepar tie$. Nous notons Σ^* l'ensemble de tous les mots sur l'alphabet Σ, et λ le mot vide. Un sous-ensemble de Σ^* est appelé *langage* de Σ^*.

Remarque 2.1 Dans tout ce document, nous supposerons que les alphabets sont finis.

Σ^* muni de l'opération « . » a donc une structure de monoïde dont l'élément neutre est λ. Ce monoïde est dit *monoïde libre* sur Σ.

La *longueur* d'un mot x, noté $|x|$, est le nombre de lettres de ce mot.

Définition 2.1 Un *automate fini* est tuple $A = (Q, q_o, \Sigma, \delta, F)$ tel que :

- Q est un ensemble fini d'états,
- $q_o \in Q$ est l'*état initial*,
- Σ est un alphabet fini,
- $\delta \subseteq Q \times \Sigma \times Q$ est un ensemble fini appelé *ensemble des transitions* de A ou *relation de transition* de A.

  Nous pouvons aussi considérer δ comme une application de $Q \times \Sigma$ vers 2^Q.
- $F \subseteq Q$ est l'ensemble des *états finaux* de A.

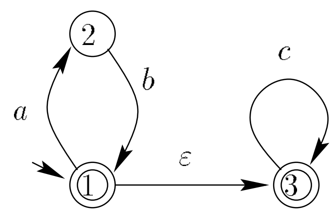
A est dit *déterministe* si pour tout $(q, a) \in Q \times \Sigma$, $\delta(q, a)$ est réduit à un unique élément.

Définition 2.2 Soit $A = (Q, q_o, \Sigma, \delta, F)$ un automate fini.

- Un mot $x = a_1 \ldots a_n$ de Σ^*, avec $(a_i)_{i=1,\ldots,n} \in \Sigma$, est dit *reconnu* par A, s'il existe une séquence de transitions $\Big((q_i, a_i, q_{i+1})\Big)_{i=0,\ldots,n} \in \delta$ telles que $q_0 = q_o$ et $q_n \in F$.
- L'ensemble des mots reconnus par A est noté $L(A)$.

Exemple: Le langage reconnu par l'automate, A, de la figure ci-dessous est $L(A) = \{(ab)^n c^m \mid n.m \in \mathbb{N}\}$. Les états finaux de A sont représentés par des doubles cercles et le symbole ε permet de passer de l'état 1 à l'état 3 sans prendre en compte de lettre.

A :



Définition 2.3 La classe des langages de Σ reconnus par un automate fini est dite *classe des reconnaissables* de Σ^* et est notée $Rec(\Sigma^*)$.

La définition suivante nous sera très utile, notamment au Chapitre 5.

Définition 2.4 Soit $A = (Q, q_o, \Sigma, \delta, F)$ un automate fini. Nous notons $A(q, q')$, avec $q, q' \in Q$, l'automate fini $A(q, q') = (Q, q, \Sigma, \delta, \{q'\})$.

Le théorème suivant est un classique de la théorie des langages.

Théorème 2.1 *La classe $Rec(\Sigma^*)$ est stable par concaténation, union, intersection et passage au complémentaire.*

L'ouvrage [HU79] peut être consulté pour les autres propriétés et notations concernant les langages reconnaissables.

2.2.1 Ensembles rationnels et reconnaissables de $(\Sigma^*)^p$

Les systèmes d'automates communicants mettent en œuvre plusieurs canaux. Par conséquent, nous devons nous intéresser aux tuples de mots et aux *ensembles* de tuples de mots sur un alphabet.

Soit Σ un alphabet, nous notons $(\Sigma^*)^p$ l'ensemble de tous les tuples de p mots sur Σ. Par exemple, (abb, dba) est un élément de $(\Sigma^*)^2$.

Soit $\vec{x} = (x_1, \ldots, x_p)$ et $\vec{y} = (y_1, \ldots, y_p)$ deux éléments de $(\Sigma^*)^p$. Nous notons $\vec{x}.\vec{y}$ ou tout simplement $\vec{x}\vec{y}$, l'élément de $(\Sigma^*)^p$ défini de la façon suivante :

$$\vec{x}.\vec{y} = (z_1, \ldots, z_p), \text{ avec pour } i \in \{1, \ldots, p\},\ z_i = x_i.y_i$$

$(\Sigma^*)^p$ muni de l'opération « . » a alors une structure de monoïde dont l'élément neutre est $\vec{\lambda} = (\lambda, \ldots, \lambda)$.

Ensembles rationnels de $(\Sigma^*)^p$

Définition 2.5 [Ber79] Soit M un monoïde. La classe, $Rat(M)$, des sous-ensembles rationnels de M est la plus petite classe $\mathcal{R}$ de sous-ensembles de M satisfaisant les propriétés suivantes :

- $\emptyset \in \mathcal{R}$,
- $\forall m \in M,\ \{m\} \in \mathcal{R}$,

- Si A, $B \in \mathcal{R}$, alors $A \cup B$, $A.B \in \mathcal{R}$
- Si $A \in \mathcal{R}$, alors $A^* \in \mathcal{R}$

avec $A.B = \{x.y \mid x \in A,\ y \in B\}$, $A^0 = \{1\}$, 1 étant l'élément neutre de M, et $A^* = \cup_{n \in \mathbb{N}} A^n$.

Un ensemble rationnel de $(\Sigma^*)^p$ est donc caractérisé par une expression rationnelle impliquant des éléments de $(\Sigma^*)^p$. Tout comme les ensembles reconnaissables de Σ^*, tout rationnel peut être caractérisé par un automate fini. Mais l'alphabet de cet automate sera $\{1, \ldots, n\} \times \Sigma$ au lieu de Σ, afin de pouvoir distinguer chacun des p canaux.

Notation: Soit u un mot sur l'alphabet $\{1, \ldots, p\} \times \Sigma$. Nous notons u_{Σ_i} la projection de u sur l'alphabet $\{i\} \times \Sigma$ et u_i, la séquence de lettres de Σ correspondant à cette projection. Nous avons par exemple :

$$\text{si } u = (1,a)(2,a)(1,b), \text{ alors } u_{\Sigma_1} = (1,a)(1,b),\ u_1 = ab \text{ et } u_2 = a$$

Définition 2.6 Soit $A = (Q, q_o, \{1, \ldots, p\} \times \Sigma, \delta, F)$ un automate fini. Nous notons $L_p(A)$ le sous-ensemble de $(\Sigma^*)^p$ défini de la façon suivante :

$$L_p(A) = \{\vec{w} \in (\Sigma^*)^p \mid \exists u \in L(A)\ \forall i \in \{1, \ldots, p\}\ w_i = u_i\}$$

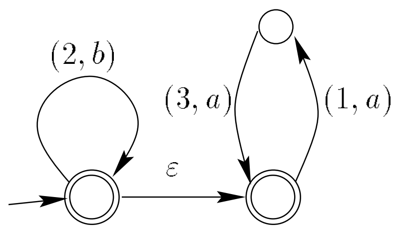
Remarque 2.2 Au vu de la définition précédente, nous pouvons remarquer que la projection d'un ensemble rationnel de $(\Sigma^*)^p$ sur l'une de ses coordonnées, est un ensemble reconnaissable de Σ^*.

Théorème 2.2 *[Niv68, Ber79] À partir de toute expression rationnelle définissant un sous-ensemble W de $(\Sigma^*)^p$, on peut construire un automate fini A sur l'alphabet $\{1, \ldots, p\} \times \Sigma$ tel que $W = L_p(A)$. Et vice-versa. Notons que l'automate A peut être déterminisé et minimisé.*

Exemple: Prenons $W = \{(a^n, b^m, a^n) \mid n, m \in \mathbb{N}\}$. Une expression rationnelle définissant W est

$$(\lambda, b, \lambda)^*.(a, \lambda, a)^*$$

et un automate fini correspondant est



Théorème 2.3 ([Ber79]) *La classe $Rat((\Sigma^*)^p)$, des ensembles rationnels de $(\Sigma^*)^p$, est stable par concaténation et union mais n'est pas stable par intersection ou passage au complémentaire.*

Exemple: Voici une intersection, non rationnelle, de deux rationnels de $(\Sigma^*)^2$:

$$\begin{aligned}(a,c)^*(b,\lambda)^* \cap (a,\lambda)^*(b,c)^* &= (a^n b^*, c^n) \cap (a^* b^m, c^m) \text{ avec } n, m \in \mathbb{N} \\ &= (a^n b^n, c^n) \text{ avec } n \in \mathbb{N}\end{aligned}$$

Or, $\{a^n b^n \mid n \in \mathbb{N}\}$ n'est pas un langage reconnaissable de Σ^*. Par conséquent, au vu de la Remarque 2.2, $(a,c)^*(b,\lambda)^* \cap (a,\lambda)^*(b,c)^*$ n'est pas un rationnel de $(\Sigma^*)^2$.

Par contre, nous verrons dans ce qui suit, que la classe des reconnaissables de $(\Sigma^*)^p$ possède les mêmes propriétés de clôture que dans Σ^*.

Ensembles reconnaissables de $(\Sigma^*)^p$

Pour définir les ensembles reconnaissables de $(\Sigma^*)^p$, nous partirons de la définition des ensembles reconnaissables d'un monoïde. Monoïde qui sera ici $(\Sigma^*)^p$.

Définition 2.7 Soit Σ un alphabet. W un sous ensemble de $(\Sigma^*)^p$ est dit reconnaissable, s'il existe un monoïde fini N, un sous ensemble P de N et h un morphisme de $(\Sigma^*)^p$ dans N tels que :

$$W = h^{-1}(P)$$

Une description plus pratique à manipuler est donnée par le théorème de Mezei.

Théorème 2.4 (Mezei [Ber79]) *Un sous ensemble W de $(\Sigma^*)^p$ est reconnaissable si et seulement si il existe une suite $(B_{ij})_{\substack{i=1,\ldots,n\\ j=1,\ldots,p}}$ d'ensembles reconnaissables de Σ^* telle que :*

$$W = \bigcup_{i=1}^{n} B_{i1} \times \ldots \times B_{ip}$$

Comme pour les ensembles reconnaissables de Σ^*, nous pouvons aussi «reconnaître» un ensemble reconnaissable de $(\Sigma^*)^p$ via un automate sur un alphabet d'éléments générateurs de $(\Sigma^*)^p$. L'alphabet de ce type le plus simple à utiliser est bien sûr le suivant :

$$\begin{array}{rcl} A^p_\Sigma & = & \Sigma \times \{\lambda\} \times \ldots \times \{\lambda\} \\ & \cup & \{\lambda\} \times \Sigma \times \{\lambda\} \times \ldots \times \{\lambda\} \\ & \vdots & \vdots \\ & \cup & \{\lambda\} \times \{\lambda\} \times \ldots \times \{\lambda\} \times \Sigma \end{array}$$

Puis, étant donné W un ensemble reconnaissable de $(\Sigma^*)^p$, on procède de la façon suivante. Soient N, P et h le monoïde, son sous-ensemble et le morphisme tels que $W = h^{-1}(P)$, on définit l'automate $(N, 1, A^p_\Sigma, \delta, P)$, avec 1 pour élément neutre de N et δ telle que pour $x \in A^p_\Sigma$ et $q \in N$, $\delta(q, x) = q.h(x)$. La sémantique est alors la suivante : un élément de $(\Sigma^*)^p$ est reconnu par cet automate si toutes les factorisations de cet élément via A^p_Σ sont reconnues, au sens classique, par cet automate. On peut vérifier que l'ensemble des éléments reconnus, de cette façon, par cet automate est bien W.

Toutefois, ce type d'automate a un nombre d'états inutilement élevé dû au fait qu'un élément de $(\Sigma^*)^p$ n'a pas une décomposition unique sur A^p_Σ. De ce fait, une autre caractérisation plus compacte est proposée par Boigelot et Godefroid [BG96], les *Queue Decision Diagram (QDD)*. Il s'agit simplement de reconnaître un élément de l'ensemble reconnaissable considéré, en procédant coordonnée par coordonnée, en commençant par la première et en finissant par la dernière. Ce faisant, on ne teste qu'une factorisation des éléments de $(\Sigma^*)^p$. On peut utiliser des automates déterministes et aussi les minimiser par la méthode classique [HU79].

Définition 2.8 [BG96] Un *QDD* A sur $(\Sigma^*)^p$ est un automate fini déterministe $(Q, q_o, \{1, \ldots, p\} \times \Sigma, \delta, F)$ tel que :

$$\forall u \in L(A),\ u = u_{\Sigma_1} \cdots u_{\Sigma_p}$$

u_{Σ_i} étant la projection de u sur l'alphabet $\{i\} \times \Sigma$, c'est-à-dire, u privé de tous les couples n'appartenant pas à $\{i\} \times \Sigma$.

La sémantique d'un QDD A est alors la suivante : un élément $\vec{w}$ de $(\Sigma^*)^p$ est dit reconnu par A si $\vec{w} \in L_p(A)$, $L_p(A)$ ayant été définie à la Définition 2.6.

Comme représentation des reconnaissables de $(\Sigma^*)^p$, nous avons tout de même préféré la caractérisation de Mezei, qui nous a paru syntaxiquement plus simple à manipuler que les QDD. En particulier, à chaque ensemble reconnaissable de $(\Sigma^*)^p$ défini par un produit de p ensembles reconnaissables de Σ^*, nous associons le tuple des automates finis correspondants (un pour chaque coordonnée).

Définition 2.9 Soit $B = (B_1, \ldots, B_p)$ un tuple de p automates finis, $B_i = (Q_i, q_{o_i}, \Sigma, \delta_i, F_i)$, définissant chacun un langage reconnaissable de Σ^*.

- Nous notons $L(B)$ l'ensemble reconnaissable de $(\Sigma^*)^p$ défini de la façon suivante :
$$L(B) = L(B_1) \times \cdots \times L(B_p)$$
- Soit $\vec{q}, \vec{q'} \in Q_1 \times \cdots \times Q_p$. Nous notons $B(\vec{q}, \vec{q'})$ le tuple, de p automates finis, défini de la façon suivante :
$$B(\vec{q}, \vec{q'}) = (B_1(q_1, q'_1), \ldots, B_p(q_p, q'_p))$$
- Nous notons Q_B et F_B les ensembles suivants
$$\begin{aligned} Q_B &= Q_1 \times \cdots \times Q_p \\ F_B &= F_1 \times \cdots \times F_p \end{aligned}$$

  et $\delta_B \subseteq Q_B \times (\Sigma^*)^p \times Q_B$ la relation de transition, satisfaisant la propriété suivante
$$(\vec{q}, \vec{w}, \vec{q'}) \in \delta_B \text{ si et seulement si pour } i \in \{1, \ldots, p\},\ q'_i \in \delta_i(q_i, w_i)$$

Le théorème suivant résume les propriétés des reconnaissables et des rationnels de $(\Sigma^*)^p$.

Théorème 2.5 *[Ber79] Les classes $Rec((\Sigma^*)^p)$ et $Rat((\Sigma^*)^p)$ ont les propriétés suivantes :*

- $Rec((\Sigma^*)^p) \subseteq Rat((\Sigma^*)^p)$,

– *$Rec((\Sigma^*)^p)$ est close par union, intersection, concaténation, et passage au complémentaire, mais n'est pas close par l'opérateur étoile.*

– *$Rat((\Sigma^*)^p)$ n'est pas close par intersection et par passage au complémentaire.*

2.3 Le modèle des automates communicants

2.3.1 Systèmes de transitions

La sémantique du modèle des automates communicants se définit par le système de transitions qu'il génère. Nous commençons donc par définir ce concept.

Définition 2.10 [Kel76, Arn92] Un système de transitions est un quadruplet $ST = (S, s_o, \Gamma, \Delta)$ tel que :

- S est ensemble d'*états*,
- $s_o \in S$ est l'*état initial*,
- Γ est un ensemble d'*actions*,
- $\Delta \subseteq S \times \Gamma \times S$ est une relation de transition.

  Δ est aussi considérée comme une fonction de $S \times \Gamma$ vers 2^S.

Définition 2.11 Soit ST un système de transition.

- Nous notons ℓ le morphisme de Δ^* dans Γ^* tel que $\ell((s, a, s')) = a$.
- L'état $s' \in S$ est dit *accessible* par l'état $s \in S$ en franchissant la transition $t \in \Delta$, ce que nous notons $s \xrightarrow{t}_{ST} s'$ ou $s \xrightarrow{\ell(t)}_{ST} s'$, si $t = (s, a, s')$ avec $a \in \Gamma$.
- Soient s et s' deux états de S. Nous notons $s \longrightarrow_{ST} s'$ s'il existe une transition $t \in \Delta$ telle que $s \xrightarrow{t}_{ST} s'$.

Remarque 2.3 Lorsqu'il n'y aura pas d'ambiguïté, nous noterons $\longrightarrow$ au lieu de $\longrightarrow_{ST}$.

Définition 2.12 Soit ST un système de transitions.

- Une *exécution* de ST est une séquence s_1, t_1, s_2, $t_2 \dots t_n$, s_{n+1} que nous notons aussi $s_1 \xrightarrow{t_1 \dots t_n} s_{n+1}$ ou $s_1 \xrightarrow{\ell(t_1 \dots t_n)} s_{n+1}$, telle que

$$\forall i \in \{1, \dots, n+1\}, \ \ s_i \xrightarrow{t_i} s_{i+1}$$

- Nous notons $\xrightarrow{*}$ la clôture réflexive et transitive de $\longrightarrow$.
- Un état $s' \in S$ est dit *accessible* à partir de l'état $s \in S$ si $s \xrightarrow{*} s'$, c'est-à-dire, s'il existe une exécution partant de s et aboutissant à s'.
- L'*ensemble des états accessibles* (ou encore *l'ensemble d'accessibilité*), $Acc(ST)$, de ST est l'ensemble des états accessibles à partir de l'état initial s_o. Autrement dit, nous avons

$$Acc(ST) = \{s \in S \mid s_o \xrightarrow{*} s\}$$

Définition 2.13 Soit ST un système de transition.

- L'arbre d'accessibilité de ST est un arbre étiqueté, $AAcc(ST)$, dont la racine est étiquetée par s_o et tel que : un nœud étiqueté par $s \in S$ a un fils étiqueté par $s' \in S$ et l'arc (s, s') est étiqueté par $\ell(t)$ si et seulement si $s \xrightarrow{t} s'$.
- Le graphe d'accessibilité de ST est le graphe étiqueté, $GAcc(ST)$, obtenu à partir de $AAcc(ST)$ en identifiant les nœuds ayant même étiquette.

Simulation et bisimulation

Les notions de *simulation* et de *bisimulation*, [Par81, Mil89], permettent de comparer les comportements de deux systèmes de transitions. Un système ST_1 *est simulé* par un autre système ST_2, si en partant de leur état initial respectif, toute action que peut réaliser ST_1 à partir de son état courant peut aussi être réalisée par ST_2 à partir de son propre état courant. La notion de bissimulation permet, elle, de définir l'équivalence de deux systèmes de transitions.

Définition 2.14 Soient $ST_1 = (S_1, s_{o_1}, \Gamma, \Delta_1)$ et $ST_2 = (S_2, s_{o_2}, \Gamma, \Delta_2)$ deux systèmes de transitions.

- Une relation $R \subseteq S_1 \times S_2$ est une *relation de simulation* de ST_1 par ST_2 si les conditions suivantes sont réalisées :

  1. $s_{o_1} \ R \ s_{o_2}$,

2. $s_1 \; R \; s_2, a \in \Gamma$ et $s_1 \xrightarrow{a}_{ST_1} s'_1 \Longrightarrow \exists s'_2 \in S_2,\, s_2 \xrightarrow{a}_{ST_2} s'_2$ et $s_2 \; R \; s'_2$

- Une relation $R \subseteq S_1 \times S_2$ est une *relation de bisimulation* entre ST_1 et ST_2 si R est une relation de simulation de ST_1 par ST_2 et aussi une relation de simulation de ST_2 par ST_1.

2.3.2 Systèmes d'automates communicants et automates à files

Définition 2.15 Un *automate communicant* est la donnée d'un sextuplet $A = (Q, q_o, \Sigma, \Pi, \delta)$ tel que :

- Q est un ensemble fini d'*états* de A,
- $q_o \in Q$ est l'*état initial*,
- Σ est un ensemble fini, appelé *alphabet des canaux*,
- Π est un ensemble fini, appelé *alphabet des actions internes*,
- $\delta \subseteq Q \times \Big((\mathbb{N} \times \{!, ?\} \times \Sigma) \cup \Pi\Big) \times Q$ est un ensemble fini de transitions.

  δ est aussi perçue comme une fonction de $Q \times \Big((\mathbb{N} \times \{!, ?\} \times \Sigma) \cup \Pi\Big)$ vers 2^Q.

Nous l'avons vu, une étiquette de transition de la forme $c!a$ (respectivement $c?a$), où $c \in \mathbb{N}$ et $a \in \Sigma$, correspond à l'émission du message a dans le canal c (respectivement à la réception du message a du canal c). Par contre, une étiquette de la forme π, avec $\pi \in \Pi$, correspond à une action interne qui n'a donc aucun effet sur les canaux.

Remarque 2.4 Comme δ est une relation finie, un automate communicant ne peut gérer qu'un nombre fini de canaux.

Définition 2.16 Un système de n automates communicants est un tuple $S = (A_1, \ldots, A_n)$, avec $A_i = (Q_i, q_{o_i}, \Sigma, \Pi, \delta_i)$ des automates communicants tels que, pour tout canal $c \in \mathbb{N}$, il existe au plus un automate pouvant y envoyer des messages et au plus un automate pouvant y recevoir des messages.

Nous notons *émet*$(c) \in \{1, \ldots, n\}$, avec $c \in \{1, \ldots, p\}$, l'indice de l'*unique* automate pouvant envoyer des messages dans le canal c.

Remarque 2.5 Soit p le nombre total de canaux différents impliqués dans un système $S = (A_1, \ldots, A_n)$ d'automates communicants. Nous pouvons, quitte à procéder à un renommage des canaux, considérer que l'identité des canaux va de 1 à p.

Définition 2.17 Soit $S = (A_1, \ldots, A_n)$ un système de n automates communicants $A_i = (Q_i, q_{o_i}, \Sigma, \Pi, \delta_i)$.

- Un *état global* (ou tout simplement *état*) de S est un élément de $Q_1 \times \ldots \times Q_n \times (\Sigma^*)^p$.
- L'ensemble des états de S est

$$EG(S) = Q_1 \times \ldots \times Q_n \times (\Sigma^*)^p$$

- $s_o = (q_{o_1}, \ldots, q_{o_n}; \lambda, \ldots, \lambda)$ est l'*état initial* de S.

Nous notons ℓ le morphisme de δ^* vers $\Big((\mathbb{N} \times \{!, ?\} \times \Sigma) \cup \Pi\Big)^*$ tel que $\ell((q, x, q')) = x$.

Définition 2.18 Soit $S = (A_1, \ldots, A_n)$ un système de n automates communicants $A_i = (Q_i, q_{o_i}, \Sigma, \Pi, \delta_i)$. Un état $s' = (q'_1, \ldots, q'_n; x'_1, \ldots, x'_p)$ est dit *accessible* d'un autre état $s = (q_1, \ldots, q_n; x_1, \ldots, x_p)$ par *l'exécution de la transition t*, ce que nous notons $s \rightarrow s'$, ou de façon redondante $s \xrightarrow{t} s'$ ou $s \xrightarrow{\ell(t)} s'$, si l'une des conditions suivantes est vérifiée :

1. (émission)

   Il existe $i \in \{1, \ldots n\}$, $c \in \{1, \ldots, p\} eta \in \Sigma$ tel que $t = (q_i, c!a, q'_i) \in \delta_i$, avec

   (a) $q'_k = q_k$, pour tout $k \neq i$
   (b) $x'_c = x_c.a$
   $x'_j = x_j$, pour tout $j \neq c$

2. (réception)

   Il existe $i \in \{1, \ldots n\}$, $c \in \{1, \ldots, p\} eta \in \Sigma$ tel que $t = (q_i, c?a, q'_i) \in \delta_i$, avec

   (a) $q'_k = q_k$, pour tout $k \neq i$
   (b) $a.x'_c = x_c$
   $x'_j = x_j$, pour tout $j \neq c$

3. (action interne)
   Il existe $i \in \{1, \dots n\}$ et $\pi \in \Pi$ tel que $t = (q_i, \pi, q'_i) \in \delta_i$, avec
   (a) $q'_k = q_k$, pour tout $k \neq i$
   (b) $x'_j = x_j$, pour tout j

À un système $S = (A_1, \dots, A_n)$ de n automates communicants $A_i = (Q_i, q_{o_i}, \Sigma, \Pi, \delta_i)$ et p canaux, nous pouvons associer le système de transitions

$$\Big(EG(S), s_o, (\{1, \dots, p\} \times \{!, ?\} \times \Sigma) \cup \Pi, \rightarrow\Big)$$

À partir de ce système de transitions, nous obtenons les notions classiques d'ensemble d'accessibilité, $Acc(S)$, d'arbre d'accessibilité, $AAcc(S)$, de graphe d'accessibilité, $GAcc(S)$, et d'exécution.

Nous notons $Acc(S, \vec{q})$ l'ensemble des contenus accessibles des canaux, à l'état de contrôle $\vec{q} \in Q_1 \times \dots \times Q_n$, et $Acc(S, \vec{q}, c)$ la projection de $Acc(S, \vec{q})$ sur le canal $c \in \{1, \dots, p\}$, ce qui nous donne :

$$\begin{array}{rcl} Acc(S, \vec{q}) & = & \{\vec{x} \in (\Sigma^*)^p \mid (\vec{q}; \vec{x}) \in Acc(S)\} \\ Acc(S, \vec{q}, c) & = & \{x_c \in (\Sigma^*)^p \mid (\vec{q}; \vec{x}) \in Acc(S)\} \end{array}$$

Automate à files

En faisant le produit des automates d'un système d'automates communicants, nous pouvons associer à ce système un autre système composé d'un *unique* automate.

Définition 2.19 Soit $S = (A_1, \dots, A_n)$ un système de n automates communicants $A_i = (Q_i, q_{o_i}, \Sigma, \Pi, \delta_i)$. Nous notons $A_S = (Q_S, \vec{q_o}, \Sigma, \Pi, \delta_S)$ l'automate communicant tel que :

- $Q_S = Q_1 \times \dots \times Q_n$,
- $\vec{q_o} = (q_{o_1}, \dots, q_{o_n})$
- $(\vec{q}, x, \vec{q'}) \in \delta_S$, si et seulement s'il existe $i \in \{1, \dots, n\}$ tel que $(q_i, x, q'_i) \in \delta_i$ et pour $j \neq i$, $q_j = q'_j$.

Il est immédiat de constater qu'un système $S = (A_1, \ldots, A_n)$ est équivalent au système (A_S). Par la suite, nous appellerons *automate à files* un système composé d'un unique automate communicant. Le système de transitions généré par un automate à files se définit immédiatement à partir de celui d'un système d'automates communicants. Toutefois, la définition suivante nous servira de référence, notamment au Chapitre 3.

Définition 2.20 Soit A un automate à p files. Un état $s' = (q'; x'_1, \ldots, x'_p)$ est dit *accessible* d'un autre état $s = (q; x_1, \ldots, x_p)$ par *l'exécution de la transition* t, ce que nous notons $s \rightarrow s'$, ou de façon redondante $s \xrightarrow{t} s'$ ou $s \xrightarrow{\ell(t)} s'$, si l'une des conditions suivantes est vérifiée :

1. (émission)

   Il existe $c \in \{1, \ldots, p\}$ et $a \in \Sigma$ tels que $t = (q, c!a, q') \in \delta$ avec

   $x'_c = x_c.a$

   $x'_j = x_j$, pour tout $j \neq c$

2. (réception)

   Il existe $c \in \{1, \ldots, p\}$ et $a \in \Sigma$ tels que $t = (q, c?a, j), q') \in \delta$ avec

   $a.x'_c = x_c$

   $x'_j = x_j$, pour tout $j \neq c$

3. (action interne)

   Il existe $\pi \in \Pi$ tel que $t = (q_i, \pi, q'_i) \in \delta_i$ et $x'_j = x_j$ pour tout $j \in \{1, \ldots, p\}$

2.3.3 Les propriétés

Nous présentons dans ce qui suit les problèmes classiquement étudiés lors de la vérification de systèmes d'automates communicants et nous rappelons les résultats concernant la décidabilité de ces problèmes.

Définition 2.21 Soient $S = (A_1, \ldots, A_n)$ un système de n automates communicants $A_i = (Q_i, q_{o_i}, \Sigma, \Pi, \delta_i)$, t une transition de S et $s = (\vec{q}; \vec{x}) \in EG(S)$ un de ses états.

- s est un *état stable* si $\vec{x} = \vec{\lambda}$.
- s est un *état de blocage* si s est un état stable et si toute transition partant de q_i, avec $i \in \{1, \ldots, n\}$, est une réception.

- s est un *état de réception non spécifiée* s'il existe $i \in \{1, \ldots, n\}$ tel que toute transition partant de q_i dans δ_i est étiquetée par une réception et $\Big(\delta_i(q_i, j?a) \neq \emptyset \Longrightarrow |x_j| \geq 1 \text{ et } x_j \notin a\Sigma^*\Big)$

  Un *état de réception non spécifiée*, correspond à un état dans lequel un automate est définitivement bloqué par son incapacité à recevoir les messages présents en tête de ses canaux.

- t est *quasi-vivante* s'il existe deux états accessibles s et s' tels que $s \xrightarrow{t} s'$.

Définition 2.22 Soit S un système d'automates communicants.

1. Le *problème de l'accessibilité* est de déterminer si un état donné de S est accessible.
2. Le *problème de la détection d'état de blocage* est de déterminer s'il existe des états accessibles qui soient des états de blocage.
3. Le *problème de la détection de réceptions non spécifiées* est de déterminer s'il existe des états accessibles qui soient des états de réceptions non spécifiées.
4. Le *problème de la quasi-vivacité* est de déterminer si une transition donnée de S est quasi-vivante.
5. Le *problème de la finitude de l'ensemble d'accessibilité* est de déterminer si $Acc(S)$ est fini.
6. Le *problème de la terminaison* est de déterminer si $AAcc(S)$ est fini.

Le théorème qui suit énonce l'impossibilité de vérifier ces problèmes.

Théorème 2.6 ([BZ83, FM97]) *Les systèmes d'automates communicants et les automates à files ont la puissance des machines de Turing.*

Par contre, dans [Pac82], Pachl montre que les problèmes énumérés à la définition précédente, à l'exception de ceux de la finitude de $Acc(S)$ ou $AAcc(S)$, sont décidables lorsque $Acc(S)$ est reconnaissable. Si, de plus, nous en avons une représentation *effective*, alors ils sont tous décidables.

Théorème 2.7 *Les problèmes suivants sont décidables pour un automate à files dont l'ensemble d'accessibilité est effectivement reconnaissable.*

- *l'accessibilité*

- *la détection de deadlocks*
- *la détection de réceptions non spécifiées*
- *la quasi-vivacité d'une transition*
- *la finitude de l'ensemble d'accessibilité*

2.3.4 Séquences de communication

Définition 2.23 Soit Σ un alphabet. Une *séquence de communication* sur $(\Sigma^*)^p$ est un mot sur l'alphabet $\{1, \ldots, p\} \times \{!, ?\} \times \Sigma$.

A partir d'une séquence de communication, nous aurons besoin d'extraire certaines informations. Pour cela, nous utilisons les notations suivantes.

Définition 2.24 Soit c, $c' \in \{1, \ldots, n\}$, $a \in \Sigma$ et $\mu \in \{c!a, c?a\}$:

- $(c!a)_! = c!a$, $(c?a)_! = \lambda$
- $(c?a)_? = c?a$, $(c!a)_? = \lambda$
- $(c!a)_{c'!} = a$, $(c?a)_{c'!} = \lambda$
- $(c?a)_{c'?} = a$, $(c!a)_{c'?} = \lambda$
- $\vec{\mu}_! = \begin{pmatrix} \mu_{1!} \\ \vdots \\ \mu_{p!} \end{pmatrix}$, $\vec{\mu}_? = \begin{pmatrix} \mu_{1?} \\ \vdots \\ \mu_{p?} \end{pmatrix}$

Par exemple, si $\mu = 1!a.1!b.2?c.1!a.1?b$ alors :

$$\begin{aligned} \mu_! &= 1!a.1!b.1!a \\ \mu_? &= 2?c.1?b \\ \mu_{1!} &= aba \end{aligned} \qquad \begin{aligned} \vec{\mu}_! &= \begin{pmatrix} aba \\ \lambda \end{pmatrix} \\ \vec{\mu}_? &= \begin{pmatrix} b \\ c \end{pmatrix} \end{aligned}$$

Chapitre 3

Vérification d'Automates Communicants Utilisant des Canaux Non Fiables

Nous avons vu au chapitre précédent que la vérification des automates communicants passe par la réduction de la puissance du modèle. En effet, celui-ci est trop «parfait» et permet de «tout» simuler. Cette puissance d'expression est manifestement apportée par la présence des canaux fifo non bornés. Par conséquent, si nous modifions les propriétés de ces canaux nous pouvons espérer dans certains cas une perte de puissance du modèle et donc l'existence d'algorithmes de vérification.

Les canaux ont pour rôle de transmettre, dans leur ordre d'émission, des messages différents. Commençons donc par perturber ces deux hypothèses. Si tout les messages ont la même identité, il n'y a plus lieu de faire l'effort de les transmettre intégralement. On peut les considérer comme de simples jetons et seul leur nombre par canal compte désormais ! Or, qui dit jetons dit réseaux de Petri, et donc de nombreuses possibilités *théoriques* de vérification [May84, Kos82, KM69]. En effet, on montre aisément que tout automate à files dont l'alphabet des messages se réduit à un unique élément est simulable par un réseau de Petri. La Figure 3.1 donne l'intuition de cette simulation. Nous pouvons ainsi décider de l'accessibilité, de l'utilité des transitions spécifiées (i.e. *quasi-vivacité* des transitions), de la finitude de l'ensemble d'accessibilité etc...

Une simulation du même type, par un réseau de Petri, est encore possible si l'on considère que l'ordre des messages n'a pas d'importance. Dans ce cas en effet, les canaux se comportent comme de simples *tas*. Ainsi, l'émission d'un message a dans un canal c va se traduire par l'ajout d'un jeton dans une place C_a correspondant aux messages de type a du canal c. La traduction est

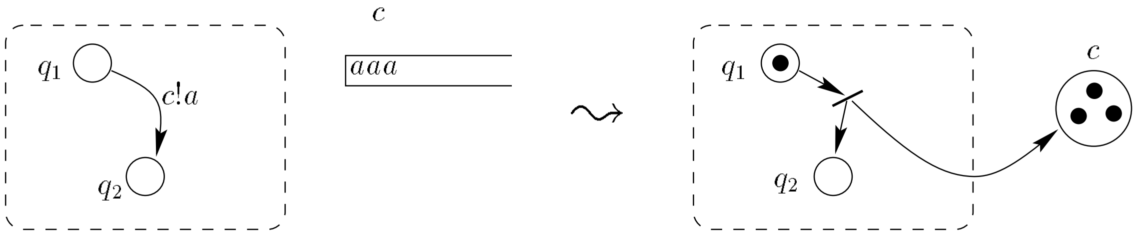


FIG. 3.1 – *Simulation d'une transition d'émission*

similaire pour les réceptions.

Un autre point de vue est de considérer les deux résultats précédents de décidabilité en terme de *défaillances* du système. La démarche devient ainsi la suivante. On modélise de façon classique un système, puis on l'analyse en considérant les diverses défaillances possibles, comme la perte d'identité des messages ou leur déséquencement (ou encore les deux à la fois). Un des intérêts de cette démarche est le suivant. L'ensemble des comportements (i.e. l'ensemble des exécutions) d'un système non fiable contient souvent celui du système fiable sous-jacent (cf. [GR84] pour l'exemple de deux automates communiquant via deux canaux dont l'un ne transmet qu'un type de message). Par conséquent, si la version non fiable d'un système ne souffre pas d'un certain type d'erreur, la version fiable de ce même système n'en souffrira pas non plus. Cette démarche nous permet alors de déduire des conditions *suffisantes* de bon fonctionnement.

Bien sûr, il s'agit de considérer judicieusement les défaillances. En effet, cela devient très facile, et à vrai dire sans grand intérêt, de vérifier les interactions entre un émetteur et un récepteur si, au préalable, on les a complètement «cassés». La défaillance qui en résulterait serait quelque peu trop brutale !

3.1 La non fiabilité facilite la vérification

Dans l'article [Fin94], Finkel introduit un nouveau type de défaillance : la perte de messages en tête des canaux. Les systèmes souffrant de cette défaillance sont maintenant dénommés *front lossy systems* [CFP96]. Finkel montre que ces systèmes sont *bien structurés* (au sens de [Fin90]), ce qui lui permet d'en déduire que le problème de la terminaison, à savoir est-ce que toutes les exécutions d'un système sont finies, est décidable. Deux autres auteurs, Abdulla et Jonsson [AJ93] ont la très bonne idée de considérer le cas où les pertes de messages peuvent avoir lieu n'importe où dans les canaux et non pas seulement en tête de ceux-ci ! Cette généralisation, les *lossy systems*, leur permet, au prix de la construction technique d'une relation de transition intermédiaire, de décider le problème de l'accessibilité.

Ce dernier résultat est important car l'un des buts des protocoles de communication est de pallier aux défaillances potentielles du média utilisé, parmi lesquelles il y a bien sûr la perte des messages. Auparavant cette perte était simulée, lors de la modélisation, par l'ajout d'un processus par canal (cf. Figure 3.2). Ce processus avait pour rôle de transmettre ou pas, et ceci de façon

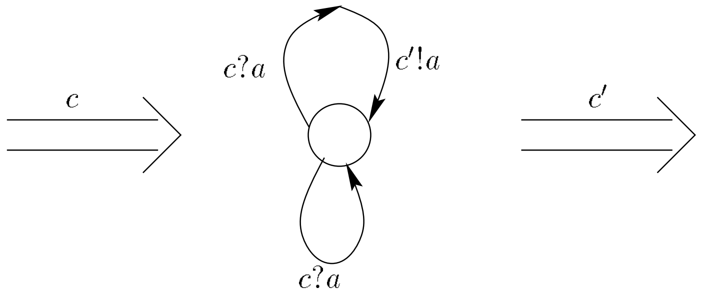


FIG. 3.2 – *Simulation de la perte de messages dans un canal. Tout message envoyé dans le canal c est soit réémis dans le canal c′ (boucle du haut), soit perdu (boucle du bas)*

non déterministe, les messages émis dans le canal en question. Mais cette astuce de modélisation n'a toutefois pas permis de décider de problèmes tels que l'accessibilité. Les résultats de [AJ93] permettent de se dispenser de cette astuce de modélisation en incluant la perte des messages directement dans la *sémantique* des automates communicants.

Les *lossy systems* ne sont pas complètement analysables, loin s'en faut. Dans [AJ94] il est montré que le model checking via la logique temporelle LTL et donc CTL* est indécidable.

Avec Finkel et Purushothaman [CFP94, CFP96], nous avons étudié outre les *lossy-systems*, deux autres types de défaillances des canaux : les insertions et les duplications de messages. Nous avons aussi considéré les diverses combinaisons de ces trois erreurs. Nos principaux résultats ont été les suivants :

– Bien que l'ensemble d'accessibilité d'un *lossy-system* soit reconnaissable et récursif, il est impossible d'en calculer une représentation *classique* sous la forme d'un QDD, [BG96], ou d'une union finie de produits de réguliers, [Ber79].

- Les erreurs d'insertion entraînent une plus forte diminution de la puissance du modèle que les erreurs de perte des messages. Et on peut calculer une représentation de l'ensemble d'accessibilité qui est reconnaissable.

- Les erreurs de duplication n'entraînent, elles, aucune perte de puissance du modèle. L'ensemble d'accessibilité reste non récursif. De même, lorsque les duplications de messages sont combinées avec des pertes de messages, les résultats de décidabilité sont les mêmes que s'il n'y avait eu que des pertes de messages.

Notre contribution personnelle dans [CFP94] a été la partie concernant les duplications seules ou combinées aux deux autres erreurs (perte et insertion de messages). Dans [CFP96] nous avons proposé une reformulation des preuves d'accessibilité pour les *lossy-systems* et les systèmes insérant des messages. Cette démarche s'est avérée simplificatrice et a été appliquée par d'autres auteurs aux *systèmes de transitions bien structurés*, [AČJT96, FS].

Ce chapitre reprend pour l'essentiel la présentation de [CFP96] mais avec les nuances suivantes.
Nous mettons l'accent sur les diverses simulations existant entre :

- la version *front-lossy* d'un système et sa version *lossy*.

- la version fiable d'un système et sa version dupliquant des messages.

- la version perdant des messages et la version perdant et dupliquant des messages d'un système.

Ce qui nous permet de déduire que les duplications de messages sont *récupérables*. Autrement dit, on peut *corriger* cette erreur.

Nous présentons aussi une preuve de décidabilité du problème de l'accessibilité pour les *front lossy systems*. Ce problème était ouvert depuis [Fin94].

3.1.1 La relation *sous-mot*

La clé de nombre de résultats concernant les systèmes non fiables, perdant ou ajoutant des messages, réside dans les bonnes propriétés de la relation *sous-mot*. Nous en faisons ici un rappel adapté aux automates à files.

Définition 3.1 Soient Σ un alphabet fini et x, y deux mots de Σ^*.

- Nous disons que x est *sous mot* de y, ce que nous notons $x \preceq y$, si

$$x = a_1 \dots a_n \text{ et } y = y_0 a_1 y_1 \dots a_n y_n \text{ avec } a_i \in \Sigma \text{ et } y_i \in \Sigma^*.$$

- Nous appelons *clôture supérieure* d'un ensemble E de mots, l'ensemble $clôture(E)$, de tous les mots supérieurs à au moins un élément de E. C'est à dire :

$$clôture(E) = \{y \in \Sigma^* | \exists x \in E,\ x \preceq y\}$$

Si x est sous mot de y, alors, on peut considérer que x a été obtenu en *perdant* des lettres de y; ou encore que y a été obtenu en *insérant* des lettres dans x. On montre aisément que la relation $\preceq$ est une relation d'ordre (i.e réflexive, transitive et anti-symétrique). Mais la propriété qui la rend vraiment intéressante est la suivante.

Théorème 3.1 (Higman [Hig52, Lot83]) *La relation $\preceq$ est un bel-ordre.*

Rappelons que cela signifie que de toute séquence infinie de mots, on peut extraire une sous séquence infinie croissante, ou encore que tout ensemble fini ou infini de mots a un nombre fini d'éléments minimaux par $\preceq$.

Tout en gardant ces propriétés, nous pouvons étendre de la façon qui suit la relation $\preceq$ aux états globaux d'un automate à files.

Définition 3.2 Soit $A =$ un automate à p files et $s = (q; \vec{w})$ et $s' = (q'; \vec{w'})$ deux de ses états. Nous disons que s est *inférieur à* s' (ou que s' est *supérieur à* s) par la relation $\preceq$, ce que nous notons $s \preceq s'$, si

$$q = q' \text{ et } \forall i \in \{1, \ldots, p\}\ w_i \preceq w'_i$$

Comme un produit de beaux-ordres demeure un bel-ordre, la relation $\preceq$ définie ci-dessus hérite des propriétés relatives aux beaux-ordres. Nous aurons aussi besoin de considérer tous les états pouvant être atteints par une succession d'un nombre quelconque d'insertions ou de pertes de messages. Par conséquent, nous étendons aussi la notion de *clôture* à un ensemble d'états.

Définition 3.3 Soit A un automate à files et $W \subseteq EG(A)$ un sous-ensemble d'états de A.

- Nous définissons $clôture(W)$ et $clôture_{inf}(W)$ de la façon suivante :

$$\begin{aligned} clôture(W) &= \{s' \in EG(A) \mid \exists s \in W, s \preceq s'\} \\ clôture_{inf}(W) &= \{s \in EG(A) \mid \exists s' \in W, s \preceq s'\} \end{aligned}$$

- Nous disons que W est *supérieurement clos* (ou plus simplement *clos*) si :

$$W = clôture(W)$$

Énonçons maintenant les deux lemmes qui nous permettrons, premièrement, de déduire la reconnaissabilité de l'ensemble d'accessibilité pour les *lossy systems* et les *insertion systems* et, deuxièmement, qui nous assureront l'arrêt des algorithmes que nous mettrons en œuvre.

Lemme 3.2 *Soit* $A = (Q, q_o, \Sigma, \Pi, \delta)$ *un automate à* p *files et* $W \subseteq \mathrm{EG}(A)$ *un sous-ensemble clos d'états de* A*. Alors* W *est reconnaissable et*

$$W = \bigcup_{\substack{(q; x_1, \ldots, x_p) \in min(W), \\ X_i = clôture(x_i)}} (q; X_1, \ldots, X_p)$$

Preuve : Si $x_i = a_1 \ldots a_p$, alors $clôture(x_i) = \Sigma^* a_1 \Sigma^* a_2 \Sigma^* \ldots \Sigma^* a_p \Sigma^*$. Par conséquent, chaque $(q; X_1, \ldots, X_p)$ est reconnaissable comme produit de langages reconnaissables [Lot83]. Or $min(W)$ est fini puisque $\preceq$ est un bel-ordre. Il s'en suit que W est lui même reconnaissable car une union finie de reconnaissables est reconnaissable [Lot83] ❑

Lemme 3.3 *Pour toute suite croissante* $\{W_k\}_{k \in \mathbb{N}}$ *d'ensembles clos d'états d'un automate à file (i.e.* $W_k \subseteq W_{k+1}$ *et* $W_k = clôture(W_k)$ *) il existe un indice* $l \in \mathbb{N}$ *tel que :*

1. $W_l = \bigcup_{k \in \mathbb{N}} W_k$.
2. $\forall k \geq l \ W_k = W_l$.

Preuve : Tous les W_i sont supérieurement clos. Il en est donc de même pour leur union $W = \bigcup_{k \in \mathbb{N}} W_k$. Par conséquent, grâce au lemme de Higman, $min(W)$ est fini. De plus, $min(W) \subseteq W$, et donc pour tout s de $min(W)$, il existe un W_{k_s} le contenant. Il suffit alors de prendre pour l le plus grand de ces k_s. L'application $clôture$ étant croissante, nous avons $W = clôture(min(W)) \subseteq clôture(W_l) = W_l$. Or, comme $W_l \subseteq W$, il s'ensuit que $W_l = \bigcup_{k \in \mathbb{N}} W_k$ ❑

3.2 La Perte de messages

Nous commençons notre étude des diverses défaillances, par le cas de la perte de messages. Nous procéderons de la manière suivante.

Après avoir modélisé le système de transition associé à ce type d'erreur, nous rappelons les résultats obtenus par Abdulla et Jonsson, notamment la décidabilité du problème de l'accessibilité et l'indécidabilité du problème de la satisfaction de formules de la logique temporelle LTL. Comme nous utiliserons certaines de leurs techniques de preuve, nous en présenterons un succinct rappel. Puis nous proposerons une nouvelle démarche permettant d'aborder le problème de l'accessibilité avec des arguments tirant meilleur parti de la structure théorique sous-jacente, les *systèmes de transitions bien structurés* [Fin90]. Cette démarche permet de simplifier les techniques de preuve de [AJ93] en évitant notamment l'utilisation de la relation *backward* partielle. Enfin nous montrons l'impossibilité du calcul d'une représentation effective de l'ensemble d'accessibilité.

Nous aborderons ensuite les *front lossy systems*, les automates à files perdant des messages mais uniquement en tête des canaux. Ce modèle a été étudié [Fin94] indépendamment des *lossy systems*. Ces deux modèles ont bien entendu de fortes similitudes, nous tâcherons de les mettre en évidence. Toutefois il serait faux de croire que tous les résultats relatifs aux *lossy systems* se transmettent aux *front lossy systems*. À ce jour, seul le problème de la terminaison était décidable [Fin94]. Nous montrons ici que le problème de l'accessibilité est lui aussi décidable.

3.2.1 Modèle associé à la perte de messages

Dans la littérature traitant du sujet, on définit les *automates à files perdant des messages* (i.e. les *lossy systems*) en modifiant une fois pour toute la relation de transition entre les états globaux. Or, nous nous attacherons ici à préciser les liens (simulation, bisimulation etc...) existant entre les différentes «versions» (i.e. à canaux parfaits, perdant les messages, dupliquant les messages...) d'un automate à files. Par conséquent, à chacune des défaillances que nous allons considérer, nous associerons un système de transitions différent, indicé par la défaillance en question. Et c'est sur ce système de transitions que nous énoncerons les résultats.

Ceci nous amène à la définition suivante dans le cas des automates à files pouvant perdre des messages.

Définition 3.4 Soit $A = (Q, q_o, \Sigma, \Pi, \delta)$ un automate à p files.

- La *fonction de transition perdant des messages*, $\longrightarrow_l$, (l pour *loss*) est définie de la façon suivante. Soit $s = (q; x_1, \ldots, x_p)$ et $s' = (q'; x'_1, \ldots, x'_p)$ deux états de A, on note $s \longrightarrow_l s'$, ou de façon redondante $s \xrightarrow{t}_l s'$ ou $s \xrightarrow{\ell(t)}_l s'$ si l'une des conditions de la Définition 2.20 ou celle qui suit est vérifiée :

4. (perte)
   Il existe $c \in \{1, \dots, p\}, a \in \Sigma$ et $w,\ w' \in \Sigma^*$ tels que $t = \tau_l$ avec
   $x_c = waw'$, $x'_c = ww'$,
   $x'_j = x_j$ pour tout $j \neq c$

Nous considérons que τ_l est une action inobservable et donc que $\ell(\tau_l) = \varepsilon$.

- Une *exécution perdant des messages* est une exécution basée sur la fonction de transition $\longrightarrow_l$.
- Un *automate à files perdant des messages* ou encore un *lossy system* est un automate à files dont on considère implicitement que la relation de transition entre états est $\longrightarrow_l$.
- Nous notons $S_l(A) = (EG(A), s_o, \rightarrow_l)$ le système de transitions associé à A considéré comme un automate perdant des messages.

Nous définissons de même $Acc_l(A)$ et $Acc_l(A, q)$ de façon identique à $Acc(A)$ et $Acc(A, q)$ sauf que nous considérons $\longrightarrow_l$ au lieu de $\longrightarrow$.

3.2.2 Résultats d'Abdulla et Jonsson

Le théorème qui suit résume l'essentiel des résultats obtenus par ces deux auteurs dans les articles [AJ93, AJ94].

Théorème 3.4 (Abdulla et Jonsson [AJ93, AJ94]) *Soit A un automate à files. Alors, pour $S_l(A)$, le système de transitions associé prenant en compte la perte de message, nous avons :*

1. *l'ensemble d'accessibilité est reconnaissable,*
2. *le problème de l'accessibilité est décidable,*
3. *le problème de la terminaison est décidable,*
4. *le model checking par la logique LTL est indécidable.*

Les techniques utilisées pour montrer ces résultats ont leur importance pour la suite de notre exposé. Nous en donnons ci-dessous les éléments significatifs.

Premier point, si un état s n'appartient pas à l'ensemble d'accessibilité, alors il en est de même pour tout état s' qui lui est supérieur. En effet, nous avons : $s \preceq s' \Longrightarrow s' \xrightarrow{*}_l s$. Par conséquent, si s' était accessible, s le serait aussi. Il s'en suit que le complémentaire, $\overline{Acc_l(A)}$, de l'ensemble d'accessibilité est supérieurement clos et donc reconnaissable par le Lemme 3.2. Or, le complémentaire d'un reconnaissable est reconnaissable [Ber79], l'ensemble $Acc_l(A)$ est donc reconnaissable.

En utilisant le résultat de Pachl [Pac87] relatif à la décidabilité de l'accessibilité dans le cas des automates à files possédant un ensemble d'accessibilité reconnaissable, le deuxième point est une conséquence directe du premier point. Cependant l'algorithme de Pachl passe par l'énumération des ensembles reconnaissables ! Ce qui est pour le moins peu efficace en pratique... Abdulla et Jonsson proposent donc une démarche plus directe (bien que sa complexité ne soit pas connue) et pour cela définissent une relation entre les états, $\leadsto$, qui est partiellement l'inverse de $\longrightarrow_l$. Ils montrent l'équivalence suivante : $s_0 \xrightarrow{*}_l s \Longleftrightarrow s \overset{*}{\leadsto} s_0$, l'état s_o étant l'état initial. De plus, grâce au lemme de Higman, ils montrent que l'arbre d'accessibilité de la relation $\leadsto$ est fini. Cela leur permet d'avoir ainsi une procédure effective.

Le troisième point, la décidabilité du problème de la terminaison, découle immédiatement du fait que les automates à files munis de la relation de transition $\longrightarrow_l$ et de l'ordre $\preceq$ sont bien-structurés [Fin90, Fin94]. Rappelons en le raisonnement. Par le lemme de Higman, s'il existe une exécution infinie, $s_o, s_1, \ldots, s_i, \ldots$, alors il existe deux états de cette exécution s_i et s_j tels que $i < j$, $s_i \xrightarrow{*}_l s_j$ et $s_i \preceq s_j$. Inversement, si deux états accessibles s et s' vérifient $s \xrightarrow{*}_l s'$ et $s \preceq s'$, alors il existe une exécution infinie. Il suffit en effet de répéter indéfiniment la séquence entre s et s', suivie de la perte de messages permettant de passer de s' à s. Par conséquent, pour décider de la terminaison, on construit l'arbre d'accessibilité en recherchant deux états satisfaisant les hypothèses précédentes. Si l'arbre est infini, on en trouvera et on pourra conclure à son infinité. Sinon on aura construit tout l'arbre et constaté qu'il est fini !

Le quatrième point, l'indécidabilité du *model-checking* de LTL sur les *lossy systems*, est fondé sur l'indécidabilité du problème RSP, *Recurrent State Problem* [AJ94], c'est-à-dire, de l'accès récurrent à un état de contrôle donné. Plus précisément :

Définition 3.5 [AJ94] Soit $A = (Q, q_o, \Sigma, \Pi, \delta)$ un automate à files et $q \in Q$ un de ses états de contrôle. Le problème, RSP (*Recurrent State Problem*), de l'accès récurrent à q consiste à déterminer s'il existe une exécution passant infiniment par q.

Théorème 3.5 ([AJ94]) *Le problème RSP est indécidable pour les automates à files perdant des messages.*

L'indécidabilité de LTL se déduit aisément du fait que RSP s'exprime en LTL par la formule suivante : $\Box\Diamond\hat{q}$, où $\hat{q}$ est le prédicat défini sur Q et vrai seulement en q.

3.2.3 Calcul effectif des prédécesseurs

Dans ce qui suit, nous proposons un résultat général qui permet de déduire de façon élégante la décidabilité de l'accessibilité. On évite, en effet, les détails techniques, relatif à la manipulation de la relation $\rightsquigarrow$, de la preuve d'Abdulla et Jonsson [AJ93]. De plus, la démarche employée ici permet de simplifier la preuve des résultats concernant les erreurs d'insertions de messages [CFP94].

Théorème 3.6 *Soit A un automate à files perdant des messages et $W \subseteq \mathrm{EG}(A)$ un ensemble d'états de A. Alors l'ensemble $\mathcal{P}(W) = \{s \in \mathrm{EG}(A) \mid \exists s' \in W, s \xrightarrow{*}_l s'\}$ de tous les prédécesseurs de W est reconnaissable. De plus, nous pouvons en calculer une représentation effective dès qu'une représentation effective de clôture(W) est calculable.*

Preuve : La quatrième condition de la Définition 3.4 nous permet de déduire que $\mathcal{P}(W)$ est supérieurement clos car : $(s \preceq s''$ et $s \xrightarrow{*}_l s') \Longrightarrow s'' \xrightarrow{*}_l s \xrightarrow{*}_l s'$. Par conséquent, $\mathcal{P}(W)$ est reconnaissable (Lemme 3.2). Définissons maintenant une suite d'ensembles d'états $\{P_k\}_{k\in\mathbb{N}}$ de la façon suivante :

$$\begin{aligned} P_0 &= W \\ P_{k+1} &= P_k \cup \{s \in EG(A) \mid \exists s' \in P_k, s \rightarrow_l s'\} \end{aligned}$$

Nous avons $\mathcal{P}(W) = \bigcup_{k\in\mathbb{N}} P_k$. Mais nous avons aussi, $\mathcal{P}(W) = \bigcup_{k\in\mathbb{N}} P'_k$ avec

$$\begin{aligned} P'_0 &= \mathit{clôture}(W) \\ P'_{k+1} &= \mathit{clôture}(P'_k \cup \{s \in EG(A) \mid \exists s' \in P'_k, s \rightarrow_l s'\}) \end{aligned}$$

Comme nous avons une représentation reconnaissable et effective de *clôture*(W), nous pouvons en calculer une pour chaque P'_{k+1} à partir de celle de P'_k. De plus, $\{P'_k\}_{k\in\mathbb{N}}$ est une séquence croissante d'ensembles supérieurement clos. Par conséquent, par le Lemme 3.3, il existe un indice l tel que $\mathcal{P}(W) = P'_l$. De plus, ce P'_l est atteint dès que $P'_l = P'_{l+1}$. Comme l'égalité entre ensembles reconnaissables est décidable, nous avons un critère d'arrêt de la construction des P'_k à P'_l ❑

Remarque 3.1 Dès que W est fini ou que nous en avons une représentation reconnaissable, nous pouvons calculer $\mathcal{P}(W)$, car dans ce cas, *clôture*(W) est calculable.

Le problème de l'accessibilité devient ainsi un corollaire immédiat du théorème précédent.

Corollaire 3.7 *Le problème de l'accessibilité est décidable pour les automates à files perdant des messages.*

Preuve : Soit s l'état global dont nous voulons vérifier l'accessibilité et s_o l'état initial du système. Nous avons :

$$s_0 \overset{*}{\rightarrow}_l s \Longleftrightarrow s_0 \in \mathcal{P}(\{s\})$$

Or, la seconde part de l'équivalence est décidable grâce au théorème précédent ❑

On peut remarquer que la démarche de la preuve du théorème précédent ne dépend pas explicitement de la relation sous mot, mais de ses propriétés : elle forme avec les automates à files des systèmes de transitions bien structurés (au sens de [Fin90]). Cette démarche que nous avons proposée dans [CFP96] a été généralisée aux *systèmes de transitions bien structurés* avec le résultat suivant :

Théorème 3.8 ([AČJT96, FS]) *Soient $(S, s_o, \rightarrow, \leq)$ un système de transitions bien structuré et W un sous-ensemble supérieurement clos de S tel que min(W) soit calculable. Alors $\mathcal{P}(W)$ l'ensemble de tous les prédécesseurs de W est supérieurement clos et sa représentation min$(\mathcal{P}(W))$ est calculable.*

3.2.4 Impossibilité d'une représentation effective de l'ensemble reconnaissable d'accessibilité

Nous avons vu que l'ensemble d'accessibilité d'un automate à files capables de perdre des messages est reconnaissable. De plus, comme son complémentaire est clos supérieurement, ce complémentaire est représentable par ses éléments minimaux qui sont en nombre fini (Théorème de Higman). Il est alors tentant de penser qu'il est possible de déterminer ces éléments minimaux étant donné que l'accessibilité est décidable ! Ceci pour pouvoir en déduire une représentation de l'ensemble d'accessibilité. Malheureusement, nous montrons dans ce qui suit que ce calcul est impossible !

Théorème 3.9 *Soit A un automate à files perdant des messages et c un de ses canaux. Déterminer si la taille utile (i.e. le nombre maximum de messages contenus dans un canal au cours des exécutions de A) du canal c est bornée est un problème indécidable.*

Preuve : Nous allons procéder par une réduction de ce problème au problème indécidable RSP (Définition 3.5 et Théorème 3.5). Soit A un automate à files perdant des messages et q un de ses état de contrôle. Modifions A de la façon suivante :

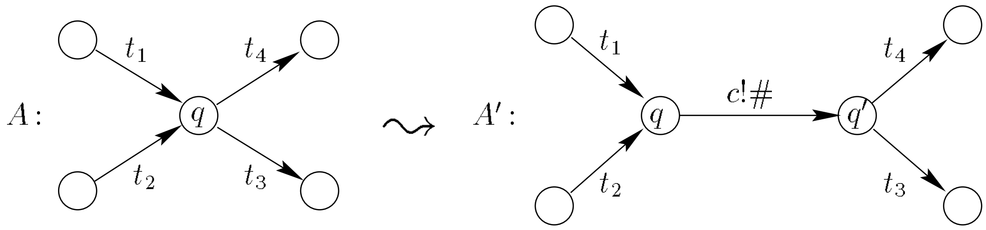


FIG. 3.3 – *Transformation de A*
c est un nouveau canal,
$\#$ une nouvelle lettre et
q' un nouvel état.

Et notons A' le nouvel automate ainsi obtenu. Les deux automates A et A' ont un fonctionnement quasi identique. En particulier, il existe, dans A, une exécution passant une infinité de fois par q si et seulement si il en est de même dans A'. De plus, le contenu du canal c de A' est de taille non bornée si et seulement si il existe une exécution passant une infinité de fois par q. Par conséquent, RSP est vrai pour l'automate à files A et son état de contrôle q si et seulement si le canal c de A' est de taille utile non bornée. Cette dernière remarque finit la réduction annoncée ❑

De ce théorème nous en déduisons le résultat annoncé au début de cette section.

Corollaire 3.10 *Il n'existe pas d'algorithme qui, étant donné un automate à files A, permettrait de calculer une représentation de l'ensemble reconnaissable d'accessibilité* $\mathrm{Acc}_l(A)$.

Preuve : Procédons par contradiction. Soit A un automate à files et c une de ses files. L'ensemble d'accessibilité $Acc_l(A)$ est reconnaissable. Supposons maintenant que nous en ayons une représentation sous la forme d'une union finie de produits de langages reconnaissables de Σ^*. L'une des coordonnées de chaque produit représente le contenu du canal c. Comme nous sommes face à des langages reconnaissables, il est aisé de savoir si ce contenu a un nombre fini d'éléments. Or, par le théorème précédent, ceci est impossible ❏

3.2.5 Les *front lossy systems*

Nous abordons maintenant le cas des *front lossy systems*, les automates à files perdant des messages uniquement en tête des files. Cette classe d'automates à files a été introduite par Finkel [Fin94] sous le nom de *completely specified protocols*. Les *front lossy systems* ont maintenant un intérêt un peu marginal car ils sont d'un maniement plus technique et d'un intérêt de modélisation moindre que les *lossy systems*. Cependant ils ont été un bon exemple d'illustration des concepts liés aux systèmes de transitions bien structurés [Fin90]. C'est ainsi par l'utilisation de ces concepts que Finkel a montré la décidabilité du problème de la terminaison, [Fin94]. De plus, l'utilisation des *front lossy systems* nous a permis de mieux aborder les démonstrations concernant la combinaison des erreurs de duplication et de perte de messages (cf. Section 3.5.3).

La suite de cette section consiste à définir le système de transition associé aux *front lossy systems*, puis à préciser les liens avec les *lossy systems*, enfin nous donnerons les résultats obtenus, dont celui de l'accessibilité.

Définition 3.6 Soit $A = (Q, q_o, \Sigma, \Pi, \delta)$ un automate à p files.

- La *fonction de transition perdant des messages en tête*, $\longrightarrow_{fl}$, (fl pour *front lossy*) est définie de la façon suivante. Soit $s = (q; x_1, \ldots, x_p)$ et $s' = (q'; x'_1, \ldots, x'_p)$ deux états de A, on note $s \longrightarrow_{fl} s'$, ou de façon redondante $s \xrightarrow{t}_{fl} s'$, si l'une des conditions de la Définition 2.20 ou celle qui suit est vérifiée :

  4. (perte en tête)
     Il existe $c \in \{1, \ldots, p\}, a \in \Sigma$, et $w \in \Sigma^*$ tel que $t = \tau_{fl}$ avec
     $x_c = aw$, $x'_c = w$,
     $x'_j = x_j$ pour tout $j \neq c$

- Une *exécution perdant des messages en tête* est une exécution basée sur la fonction de transition $\longrightarrow_{fl}$.

- Un *automate à files perdant des messages en tête* ou encore un *front lossy system* est un automate à files dont on considère que la relation de transition entre états est $\longrightarrow_{fl}$.

- Nous notons $S_{fl}(A) = (EG(A), s_o, \rightarrow_{fl})$ le système de transitions associé à A considéré comme un automate à files perdant des messages en tête.

Nous définissons de même $Acc_{fl}(A)$ et $Acc_{fl}(A,q)$ de façon identique à $Acc(A)$ et $Acc(A,q)$ sauf que nous considérons $\longrightarrow_{fl}$ au lieu de $\longrightarrow$.

Précisons maintenant les liens entre les *lossy systems* et les *front lossy systems*. Il va de soi que «tout ce que peut faire» la version *front lossy* d'un automate à files, sa version *lossy* peut le faire. Nous montrons dans le lemme qui suit que la réciproque est aussi vraie: lorsque les actions de perte sont considérées invisibles, toute exécution de la version *lossy* d'un automate à files est observationnellement simulable par sa version *front lossy*.

Lemme 3.11 *Soient $A = (Q, q_o, \Sigma, \Pi, \delta)$ un automate à files et $S_l(A) = (\mathrm{EG}(A), s_o, \rightarrow_l)$, $S_{fl}(A) = (\mathrm{EG}(A), s_o, \rightarrow_{fl})$ les systèmes de transitions associés en considérant respectivement les relation de transitions $\rightarrow_l$ et $\rightarrow_{fl}$. Alors, en considérant les actions τ_l et τ_{fl} comme étant invisibles, nous avons :*

- *$S_{fl}(A)$ est simulé par $S_l(A)$ avec l'égalité pour relation de simulation,*
- *$S_l(A)$ est simulé observationnellement par $S_{fl}(A)$ avec $\preceq$ pour relation de simulation.*

Preuve : Le premier point découle immédiatement du fait que $\rightarrow_{fl} \subseteq \rightarrow_l$. Pour le second point, considérons $s^1 = (q^1; \vec{w^1})$, $s^2 = (q^2; \vec{w^2})$, $s^3 = (q^3; \vec{w^3}) \in EG(A)$ et $t \in (\delta \cup \{\tau_l\})$ tels que: $s^1 \xrightarrow{t}_l s^2$ et $s^1 \preceq s^3$. Montrons qu'il existe $s^4 = (q^4; \vec{w^4}) \in EG(A)$ tel que $s^3 \overset{\ell(t)}{\Longrightarrow}_l s^4$ et $s^2 \preceq s^4$.

Suivant la nature de t, quatre cas se présentent :

$t = (q^1, c!a, q^2)$. Nous avons donc $\ell(t) = c!a$, $q^3 = q^1$, $\vec{w^2} = c!a(\vec{w^1}) = \vec{w^1}c!a(\vec{\lambda})$ et $\vec{w^1} \preceq \vec{w^3}$ et aussi: $(q^3; \vec{w^3}) \xrightarrow{\ell(t)}_l (q^2; \vec{w^3}c!a(\vec{\lambda}))$. Par conséquent, en prenant $q^4 = q^2$ et $\vec{w^4} = \vec{w^3}c!a(\vec{\lambda})$, on satisfait bien aux conditions demandées, car $\vec{w^1} \preceq \vec{w^3}$ implique $\vec{w^1}c!a(\vec{\lambda}) \preceq \vec{w^3}c!a(\vec{\lambda})$.

$t = (q^1, c?a, q^2)$. Nous avons donc $\ell(t) = c?a$, $q^3 = q^1$ et $\vec{w^1} \preceq \vec{w^3}$; soit en particulier pour le canal c: $w^3_c = y_o a_1 y_1 a_2 y_2 \ldots y_{n-1} a_n y_n$ avec $w^1_c = a_1 \ldots a_n$, $a_i \in \Sigma$, $y_i \in \Sigma^*$. Définissons $\vec{w^5}$ et $\vec{w^4}$ de la façon suivante: pour $i \neq c$ $w^5_i = w^4_i = w^3_i$, $w^5_c = a_1 y_1 a_2 y_2 \ldots y_{n-1} a_n y_n$ et $w^4_c = y_1 a_2 y_2 \ldots y_{n-1} a_n y_n$. Il est immédiat de constater que: $(q^1; \vec{w^3}) \xrightarrow{(\tau_{fl})^*}_{fl} (q^1; \vec{w^5}) \xrightarrow{c?a}_{fl} (q^2; \vec{w^4})$. Ce qui entraîne $s^3 \overset{\ell(t)}{\Longrightarrow}_{fl} s^4$ et $s^2 \preceq s^4$ avec $s^4 = (q^2, \vec{w^4})$.

$t = \tau_l$. Nous avons donc $\ell(t) = \varepsilon$, $s^2 \preceq s^1 \preceq s^3$. Prenons $s^4 = s^3$, nous obtenons $s^3 \overset{\varepsilon}{\Longrightarrow}_{fl} s^4$, c'est-à-dire, $s^3 \overset{\ell(\tau_{fl})}{\Longrightarrow}_{fl} s^4$ avec $s^2 \preceq s^4$.

$t = (q^1, \pi, q^2)$ avec π une action interne. Il suffit de prendre $s^4 = (q^2; \vec{w}_3)$ car π ne modifie en rien les canaux ❑

Ce lemme nous permet de déduire l'ensemble des liens existant entre les versions *front lossy* et *lossy* d'un automate à files.

Corollaire 3.12 *Soit A un automate à files. Alors :*

1. $\mathrm{Acc}_{fl}(A) \subseteq \mathrm{Acc}_l(A)$*,*
2. $\mathrm{Acc}_l(A) =$ *clôture*$_{inf}$ $(\mathrm{Acc}_{fl}(A))$*,*
3. $\mathrm{Acc}_{fl}(A)$ *est borné si et seulement si* $\mathrm{Acc}_l(A)$ *est borné,*
4. $(q; \vec{\lambda}) \in \mathrm{Acc}_l(A)$ *si et seulement si* $(q; \vec{\lambda}) \in \mathrm{Acc}_{fl}(A)$.

Preuve :

1. Ce premier point dérive immédiatement du fait que $\rightarrow_{fl} \subseteq \rightarrow_l$. Autrement dit, tout état atteint par une exécution perdant des messages en tête est aussi atteint par une exécution perdant des messages !

2. Comme la fonction *clôture*$_{inf}$ est croissante et que $Acc_l(A)$ est clos inférieurement, nous déduisons du premier point que :

$$\textit{clôture}_{inf}\,(Acc_{fl}(A)) \subseteq \textit{clôture}_{inf}\,(Acc_l(A)) = Acc_l(A)$$

De plus, par le deuxième point du lemme précédent, pour tout état $s \in Acc_l(A)$ il existe un état $s' \in Acc_{fl}(A)$ tel que $s \in$ *clôture*$_{inf}\,(s')$. D'où le fait que $Acc_l(A) \subseteq$ *clôture*$_{inf}\,(Acc_l(A))$. Des deux inclusions qui précèdent, on en déduit le deuxième point du corollaire.

3. Supposons que $Acc_{fl}(A)$ soit borné. Il en est alors de même de sa clôture inférieure et donc de $Acc_l(A)$. Inversement, si $Acc_l(A)$ est borné, le premier point du corollaire implique qu'il en est de même pour $Acc_{fl}(A)$. Ceci montre ainsi le troisième point.

4. Supposons maintenant que $(q; \vec{\lambda}) \in Acc_l(A)$ il existe alors par le deuxième point du lemme précédent un état $(q; \vec{w}) \in Acc_{fl}(A)$. Or, il est immédiat que $(q; \vec{w}) \xrightarrow{*}_{fl} (q; \vec{\lambda})$. Par conséquent, nous avons aussi : $(q; \vec{\lambda}) \in Acc_{fl}(A)$. Le fait que $(q; \vec{\lambda}) \in Acc_{fl}(A)$ implique $(q; \vec{\lambda}) \in Acc_l(A)$ est une conséquence immédiate du premier point de ce corollaire. Ces deux dernières implications finissent la démonstration du corollaire ❑

Décidabilité de l'accessibilité

Lors de la rédaction de [CFP96], le problème de l'accessibilité des *front lossy systems* avait résisté à nos investigations. Et notre intuition oscillait tour à tour entre la décidabilité et l'indécidabilité. Finalement, nous montrons ici que ce problème est bel et bien décidable.

L'idée a été de réaliser que, pour chaque canal, une exécution d'un *front lossy system* passait par deux phases. La première où tous les messages présents dans le canal seraient consommés ou perdus. La seconde où tous les messages émis ne seraient ni consommés ni perdus mais formeraient le contenu final du canal à la fin de l'exécution.

De plus, lors de la première phase, il n'est pas important que les messages perdus le soient de façon *front lossy* ou *lossy*. Enfin, lors de la seconde phase, le nombre des messages émis est borné par la taille du contenu final dont l'on veut justement tester l'accessibilité. Par conséquent, on peut stocker ces messages dans une structure finie que l'on pourra contrôler.

Nous allons donc construire, pour chaque instance du problème d'accessibilité (un automate à files et un état à tester) un autre automate à files dont les exécutions *lossy* vont simuler les premières phases des canaux et dont la partie finie (i.e. les états de contrôle) va stocker l'information pertinente de la deuxième phase de ces canaux.

Définition 3.7 Soient $A = (Q, q_o, \Sigma, \Pi, \delta)$ un automate à p files et $\vec{w} \in (\Sigma^*)^p$ un tuple de mots de Σ^*. Nous définissons l'automate à p files, $A_{\vec{w}} = (Q', q'_o, \Sigma, \Pi, \delta')$, de la façon suivante :

- $Q' = Q \times \{0,1\}^p \times \mathit{pref}(\vec{w})$, avec $\mathit{pref}(\vec{w})$ l'ensemble de tous les préfixes de $\vec{w}$,
- $q'_o = (q_o, \vec{0}, \vec{\lambda})$,
- δ' est le plus petit ensemble de transitions tel que :

$$\begin{array}{rcl}
b_c = 0,\ (q, c!a, q') \in \delta & \Longrightarrow & (q, \vec{b}, \vec{x}) \xrightarrow{c!a} (q', \vec{b}, \vec{x}) \in \delta' \\
b_c = 1,\ c!a(\vec{x}) \leq \vec{w},\ (q, c!a, q') \in \delta & \Longrightarrow & (q, \vec{b}, \vec{x}) \xrightarrow{\tau} (q', \vec{b}, c!a(\vec{x})) \in \delta' \\
(q, c?a, q') \in \delta & \Longrightarrow & (q, \vec{b}, \vec{x}) \xrightarrow{c?a} (q', \vec{b}, \vec{x}) \in \delta' \\
(q, \pi, q') \in \delta & \Longrightarrow & (q, \vec{b}, \vec{x}) \xrightarrow{\pi} (q', \vec{b}, \vec{x}) \in \delta' \\
b_c = 0,\ b'_c = 1, (j \neq c \Rightarrow b'_j = b_j) & \Longrightarrow & (q, \vec{b}, \vec{x}) \xrightarrow{\tau} (q, \vec{b'}, \vec{x}) \in \delta'
\end{array}$$

Le lemme qui suit fait le lien entre l'accessibilité *lossy* de $A_{\vec{w}}$, qui est donc décidable, et le problème de l'accessibilité *front lossy* de A.

Lemme 3.13 *Soit A un automate à files. Alors,*

$$(q;\vec{w}) \in \mathrm{Acc}_{fl}(A) \text{ si et seulement si } ((q,\vec{1},\vec{w});\vec{\lambda}) \in \mathrm{Acc}_{l}(A_{\vec{w}})$$

Preuve : Supposons que $(q;\vec{w}) \in Acc_{fl}(A)$. Il existe alors une exécution, σ, avec perte en tête des canaux, menant de l'état initial à $(q;\vec{w})$. Au cours de cette exécution, et pour chaque canal c tel que $w_c \neq \lambda$, il existe un « instant » i_c à partir duquel les messages émis dans c ne seront plus reçus ou perdus. C'est d'ailleurs cette séquence de messages émis qui forme w_c. A partir de σ, nous construisons une exécution σ' de $A_{\vec{w}}$, avec perte en tête des canaux, menant de l'état initial à $((q,\vec{1},\vec{w});\vec{\lambda})$. Cette construction se fait de la façon suivante. À toute transition de σ on fait correspondre son équivalent (au sens des implications de la Définition 3.7) dans σ', en considérant que $b_c = 0$ avant l'«instant» i_c et $b_c = 1$ après l'«instant» i_c. Ainsi, nous avons $((q,\vec{1},\vec{w});\vec{\lambda}) \in Acc_{fl}(A_{\vec{w}})$ et donc $((q,\vec{1},\vec{w});\vec{\lambda}) \in Acc_{l}(A_{\vec{w}})$ par le $4^{ième}$ point du Corollaire 3.12.

Inversement, si $((q,\vec{1},\vec{w});\vec{\lambda}) \in Acc_{l}(A_{\vec{w}})$, alors, par le Corollaire 3.12, nous avons $((q,\vec{1},\vec{w});\vec{\lambda}) \in Acc_{fl}(A_{\vec{w}})$. Ce qui signifie qu'il existe une exécution de $A_{\vec{w}}$ avec perte en tête des canaux σ' menant de l'état initial à $((q,\vec{1},\vec{w});\vec{\lambda})$. En utilisant les transitions équivalentes dans A on montre directement que $(q;\vec{w}) \in Acc_{fl}(A)$ ❑

Le théorème qui suit fait le bilan des résultats concernant les *front lossy systems*.

Théorème 3.14 *Soit A un automate à files et $S_{fl}(A) = (\mathrm{EG}(A), s_o, \rightarrow_{fl})$ le système de transitions associé en considérant la possibilité de perdre des messages en tête. Alors pour $S_{fl}(A)$,*

1. *le problème de la terminaison est décidable,*
2. *le model-checking par la logique LTL est indécidable,*
3. *le problème de l'accessibilité est décidable.*

Preuve :

1. Procédons par une réduction au problème de la terminaisons pour $S_l(A)$ car ce dernier est décidable (Théorème 3.4). Montrons donc qu'il existe une exécution infinie dans $S_{fl}(A)$ si et seulement si il en existe une dans $S_l(A)$.

Supposons qu'il existe une exécution infinie dans $S_{fl}(A)$. Comme $S_l(A)$ simule $S_{fl}(A)$, Lemme 3.11, il en existe aussi une dans $S_l(A)$.

Supposons maintenant qu'il existe une exécution infinie dans $S_l(A)$. Cette exécution comporte une infinité d'actions observables (actions de communication ou actions internes). En effet, les seules actions invisibles sont les pertes et une exécution ne peut comporter une infinité de pertes consécutives. Or, $S_{fl}(A)$ simule observationnellement $S_{fl}(A)$, Lemme 3.11. Par conséquent, il existe aussi dans $S_{fl}(A)$ une exécution infinie.

2. Il suffit ici d'utiliser les mêmes arguments que dans le point précédent pour réduire le problème RSP (voir Définition 3.5) pour $S_l(A)$ à son équivalent pour $S_{fl}(A)$. Ce problème étant indécidable pour $S_l(A)$, il en est de même pour $S_{fl}(A)$. Or, nous avons vu (paragraphe suivant la Définition 3.5) que l'indécidabilité de RSP entraîne celle de LTL.

3. Ce troisième point est une conséquence directe du lemme précédent : tester si $s = (q; \vec{w}) \in Acc_{fl}(A)$ revient à construire $A_{\vec{w}}$ puis à tester $((q; \vec{1}, \vec{w}); \lambda) \in Acc_l(A_{\vec{w}})$, ce qui est décidable, cf. Théorème 3.4 ❑

Remarque 3.2 La décidabilité du problème de la terminaison avait déjà été obtenue par Finkel. Ce résultat constitue d'ailleurs l'essentiel de l'article [Fin94]. Mais la démonstration obtenue ici, en se fondant sur le résultat équivalent et moins technique obtenu pour les *lossy systems*, est plus directe.

3.3 L'insertion de messages

Après avoir étudié les pertes de messages, nous portons notre attention au cours de cette section à l'erreur symétrique qui est l'insertion de messages. Nous verrons, entre autres, que l'accessibilité est décidable, que l'ensemble d'accessibilité est reconnaissable et qu'il est même possible d'en calculer une représentation effective. Ce qui n'est pas possible pour le cas de la perte de messages.

Définition 3.8 Soit $A = (Q, q_o, \Sigma, \Pi, \delta)$ un automate à p files.

- La *fonction de transition insérant des messages*, $\longrightarrow_i$, (i pour insertion) est définie de la façon suivante. Soit $s = (q; x_1, \ldots, x_p)$ et $s' = (q'; x'_1, \ldots, x'_p)$ deux états de A, on note $s \longrightarrow_i s'$, ou de façon redondante $s \xrightarrow{t}_i s'$, si l'une des conditions de la Définition 2.20 ou celle qui suit est vérifiée :

  4. (insertion)

Il existe $c \in \{1, \ldots, p\}, a \in \Sigma$, et $w, w' \in \Sigma^*$ tels que $t = \tau_i$ avec
$x_c = ww'$, $x'_c = waw'$,
$x'_j = x_j$, pour tout $j \neq c$

- Une *exécution insérant des messages* est une exécution basée sur la fonction de transition $\longrightarrow_i$.
- Un *automate à files insérant des messages* est un automate à files dont on considère implicitement que la relation de transition entre états est $\longrightarrow_i$.
- Nous notons $S_i(A) = (EG(A), s_o, \rightarrow_i)$ le système de transitions associé à A considéré comme un automate à files insérant des messages.

Nous définissons de même $Acc_i(A)$ $Acc_i(A, q)$ de façon identique à $Acc(A)$ $Acc(A, q)$ sauf que nous considérons $\longrightarrow_i$ au lieu de $\longrightarrow$.

3.3.1 Reconnaissabilité effective de l'ensemble d'accessibilité

Le théorème qui suit illustre (à comparer au Théorème 3.6) la symétrie entre la perte et l'insertion de messages.

Théorème 3.15 *Soit A un automate à files insérant des messages et $W \subseteq$ EG(A) un ensemble d'états de A. Alors l'ensemble $\mathcal{S}(W) = \{s' \in \mathrm{EG}(A) \mid \exists s \in W, s \xrightarrow{*}_i s'\}$ de tous les successeurs de W est reconnaissable. De plus, nous pouvons calculer une représentation de $\mathcal{S}(W)$ dès qu'une représentation de clôture(W) est calculable.*

Preuve : Nous procédons ici de façon symétrique à la preuve du Théorème 3.6 page 36.

La quatrième condition de la Définition 3.8 nous permet de déduire que $\mathcal{S}(W)$ est supérieurement clos car : $s \preceq s'' \Longrightarrow s \xrightarrow{*}_i s''$. Par conséquent, $\mathcal{S}(W)$ est reconnaissable (Lemme 3.2). Définissons maintenant une suite d'ensembles d'états $\{S_k\}_{k \in \mathbb{N}}$ de la façon suivante :

$$\begin{aligned} S_0 &= W \\ S_{k+1} &= S_k \cup \{s \in EG(A) \mid \exists s' \in S_k, s' \rightarrow_i s\} \end{aligned}$$

Nous avons $\mathcal{S}(W) = \bigcup_{k \in \mathbb{N}} S_k$ et aussi $\mathcal{S}(W) = \bigcup_{k \in \mathbb{N}} S'_k$, avec

$$\begin{aligned} S'_0 &= \mathit{clôture}(W) \\ S'_{k+1} &= \mathit{clôture}(S'_k \cup \{s \in EG(A) \mid \exists s' \in S'_k, s' \rightarrow_i s\}) \end{aligned}$$

Comme nous avons une représentation reconnaissable et effective de *clôture* (W), nous pouvons en calculer une pour chaque S'_{k+1} à partir de celle de S'_k. De plus, $\{S'_k\}_{k\in\mathbb{N}}$ est une séquence croissante d'ensembles supérieurement clos. Par conséquent, par le Lemme 3.3, il existe un indice l tel que $\mathcal{S}(W) = S'_l$. De plus, ce S'_l est atteint dès que $S'_l = S'_{l+1}$. Comme l'égalité entre ensembles reconnaissables est décidable, nous avons un critère d'arrêt de la construction des S'_k à S'_l ❏

Les propriétés de $\mathcal{S}(W)$ découlent donc de celles de $\mathcal{P}(W)$ énoncées au Théorème 3.6, page 36, et à la remarque qui lui suit.

Nous pouvons maintenant énoncer l'essentiel des résultats concernant les automates à files insérant des messages.

Théorème 3.16 *Soit A un automate à files et $S_i(A) = (\mathrm{EG}\,(A), s_o, \rightarrow_i)$ le système de transition associé, en considérant la possibilité d'insérer des messages. Alors pour $S_i(A)$,*

1. *l'ensemble d'accessibilité,* $\mathrm{Acc}_i(A)$, *est reconnaissable. De plus, il existe un algorithme calculant une représentation de cet ensemble,*

2. *le problème de l'accessibilité est décidable,*

3. *le problème de la terminaison est décidable,*

Preuve :

1. Ce résultat est une conséquence immédiate du théorème précédent et du fait que :
$$Acc_i(A) = \mathcal{S}(\{s_o\})$$

2. Conséquence immédiate du point précédent.

3. Trivial ! Il n'y a que des exécutions maximales infinies car à tout moment, une insertion peut être déclenchée.

3.4 La Duplication de messages

Nous avons vu que les erreurs de pertes et de duplications induisent une diminution de la puissance du modèle des automates à files. Par contre, nous verrons ici que les duplications de messages n'affectent pas la puissance du modèle. Bien entendu, la version dupliquant des messages d'un automate à files aura un comportement différent de celui de la version non dupliquante. Mais nous verrons qu'il est néanmoins possible de simuler une machine de Turing avec un automate à files dupliquant des messages. Mieux, nous énoncerons les modifications à apporter à un automate à files pour qu'il «résiste» aux duplications.

Définition 3.9 Soit $A = (Q, q_o, \Sigma, \Pi, \delta)$ un automate à p files.

- La *fonction de transition dupliquant des messages*, $\longrightarrow_d$, (d pour duplication) est définie de la façon suivante. Soit $s = (q; x_1, \ldots, x_p)$ et $s' = (q'; x'_1, \ldots, x'_p)$ deux états de A, on note $s \longrightarrow_d s'$, ou de façon redondante $s \xrightarrow{t}_d s'$ ou $s \xrightarrow{\ell(t)}_d s'$, si l'une des conditions de la Définition 2.20 ou celle qui suit est vérifiée :

  4. (duplication)
     Il existe $c \in \{1, \ldots, p\}, a \in \Sigma$, et $w, w' \in \Sigma^*$ tels que $t = \tau_d$ avec
     $x_c = waw'$, $x'_c = waaw'$,
     $x'_j = x_j$ pour tout $j \neq c$

  Nous considérons que τ_d est une action inobservable et donc que $\ell(\tau_d) = \varepsilon$.

- Une *exécution dupliquant des messages* est une exécution basée sur la fonction de transition $\longrightarrow_d$.

- Un *automate à files dupliquant des messages* est un automate à files dont on considère implicitement que la relation de transition entre états est $\longrightarrow_d$.

- Nous notons $S_d(A) = (EG(A), s_o, \rightarrow_d)$ le système de transitions associé à A considéré comme un automate à files dupliquant des messages.

Nous définissons de même $Acc_d(A)$ et $Acc_d(A, q)$ de façon identique à $Acc(A)$ et $Acc(A, q)$ sauf que nous considérons $\longrightarrow_d$ au lieu de $\longrightarrow$.

3.4.1 Rattrapage des duplications

Nous voulons donc rattraper les erreurs de duplications, c'est-à-dire, faire en sorte qu'elles ne perturbent pas le système. La difficulté réside dans le fait qu'il n'est pas possible a priori de distinguer dans un canal une séquence aa de deux occurrences consécutives du même message due au fonctionnement normal du système, et la même séquence due à une duplication du message a. Nous allons donc modifier l'automate de départ pour qu'après chaque émission

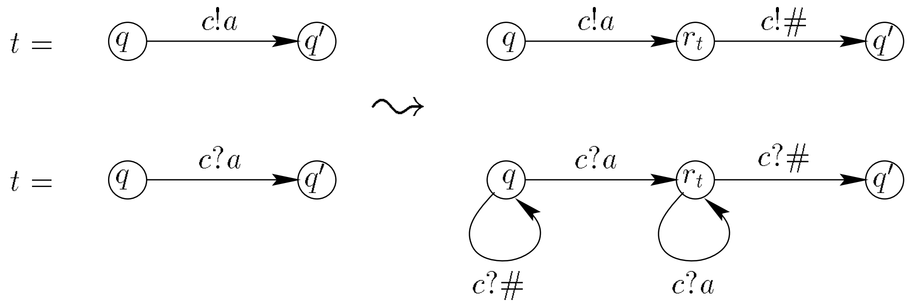


FIG. 3.4 – *Transformation de A en $R_d(A)$*

d'un message, il envoie aussi un *séparateur* (une nouvelle lettre n'appartenant pas à l'alphabet de départ, # par exemple). Inversement, à chaque réception, il faudra supprimer ce séparateur, tout en prenant en compte les duplications possibles des messages et ... des séparateurs !

La définition qui suit précise ces modifications et définit aussi une relation, que nous montrerons de bisimulation observationnelle, entre l'automate à files de départ et la modification de celui-ci.

Définition 3.10 Soit $A = (Q, q_o, \Sigma, \Pi, \delta)$ un automate à files.

- Nous notons $R_d(A)$, (R pour résistant, d pour duplication) l'automate à files obtenu à partir de A de la façon suivante.

  $R_d(A) = (Q', q_o, \Sigma', \Pi, \delta')$ avec :

  - $Q' = Q \cup \{r_t | \exists t \in \delta,\ t = (q, c!a, q')\ ou\ t = (q, c?a, q')\}$,
  - $\Sigma' = \Sigma \cup \{\#\}$,
  - δ' est le plus petit ensemble de transitions tel que :

$$\begin{array}{rcl} t = (q, c!a, q') \in \delta & \Longrightarrow & (q, c!a, r_t),\ (r_t, c!\#, q') \in \delta' \\ t = (q, c?a, q') \in \delta & \Longrightarrow & (q, c?\#, q), (q, c?a, r_t), \\ & & (r_t, c?a, r_t), (r_t, c?\#, q') \in \delta' \\ (q, \pi, q') \in \delta, \pi \in \Pi & \Longrightarrow & (q, \pi, q') \in \delta' \end{array}$$

  La Figure 3.4 résume l'essentiel des transformations de A pour obtenir $R_d(A)$.

- Nous notons R la relation incluse dans $EG(A) \times EG(R_d(A))$ telle que : $((q;\vec{w}),(q';\vec{w'})) \in R$ si et seulement si l'une des conditions suivantes est satisfaite (dans ce qui suit, nous considérons que A, et donc aussi $R_d(A)$, est un automate à p files) :

  - $q = q' \in Q$, $\forall i \in \{1,\dots,p\}$ $\exists (a_j)_{j=1,\dots,n} \in \Sigma$, $w_i = a_1 \dots a_n$, $w'_i \in \#^* a_1^+ \#^+ a_2^+ \#^+ \dots \#^+ a_n^+ \#^+$
  - $\exists t = (q^1, c!a, q^2) \in \delta$, $q = q^2$, $q' = r_t$, $\forall i \in \{1,\dots,p\}$ $\exists (a_j)_{j=1,\dots,n} \in \Sigma$, $w_i = a_1 \dots a_n$,

$$\begin{array}{lll} \text{si } i = c & \text{alors} & w'_i \in \#^* a_1^+ \#^+ a_2^+ \#^+ \dots \#^+ a_n^+ \\ \text{si } i \neq c & \text{alors} & w'_i \in \#^* a_1^+ \#^+ a_2^+ \#^+ \dots \#^+ a_n^+ \#^+ \end{array}$$

  - $\exists t = (q^1, c?a, q^2) \in \delta$, $q = q^2$, $q' = r_t$, $\forall i \in \{1,\dots,p\}$ $\exists (a_j)_{j=1,\dots,n} \in \Sigma$, $w_i = a_1 \dots a_n$,

$$\begin{array}{lll} \text{si } i = c & \text{alors} & w'_i \in a^* \#^+ a_1^+ \#^+ a_2^+ \#^+ \dots \#^+ a_n^+ \#^+ \\ \text{si } i \neq c & \text{alors} & w'_i \in \#^* a_1^+ \#^+ a_2^+ \#^+ \dots \#^+ a_n^+ \#^+ \end{array}$$

Considérons A un automate à files. Le théorème qui suit montre que $R_d(A)$ placé dans un environnement où les messages peuvent être dupliqués est observationnellement bisimulaire à A placé dans un environnement fiable. Ce qui revient à dire que $R_d(A)$ corrige les erreurs de duplication de messages !

Théorème 3.17 *Soient $A = (Q, q_o, \Sigma, \Pi, \delta)$ un automate à files, $S(A) = (\mathrm{EG}(A), s_o, \rightarrow)$ et $S_d(R_d(A)) = (\mathrm{EG}(R_d(A)), s_o, \rightarrow_d)$ les systèmes de transitions associés. Alors, en considérant les actions suivantes de $S_d(R_d(A))$ $c!\#$, $c?\#$ et celles , $c?a$, issues des transitions de type $(r_t, c?a, r_t)$ comme étant non-observables, $S(A)$ et $S_d(R_d(A))$ sont observationnellement bisimulaires.*

Preuve : Nous montrons que la relation R définie ci-dessus est une relation de bisimulation observationnelle.

Premièrement, montrons que R est une relation de simulation observationnelle de $S(A)$ par $S_d(R_d(A))$. Considérons $s^1 = (q^1;\vec{w^1})$, $s^2 = (q^2;\vec{w^2}) \in EG(A)$, $s^3 = (q^3;\vec{w^3}) \in EG(R_d(A))$ et $t \in \delta$ tels que : $s^1 \xrightarrow{t} s^2$ et $s^1\ R\ s^3$. Montrons qu'il existe $s^4 \in EG(R_d(A))$ tel que $s^3 \stackrel{\ell(t)}{\Longrightarrow}_d s^4$ et $s^2\ R\ s^4$. Suivant la définition de R, deux cas se présentent :

1. $q^3 = q^1 \in Q$. Il y a trois sous-cas suivant que t est une émission, une réception ou une action interne.

   - **Émission:** $t = (q^1, c!a, q^2)$.

Nous avons donc : $(q^3; \vec{w^3}) \xrightarrow{c!a}_d (r_t; c!a(\vec{w^3})) \xrightarrow{c!\#}_d (q^2; c!a\#(\vec{w^3}))$ ce qui signifie $(q^3; \vec{w^3}) \overset{c!a}{\Longrightarrow}_d (q^2; c!a\#(\vec{w^3}))$. Prenons $s^4 = (q^2; c!a\#(\vec{w^3}))$. Il est simple de vérifier que $q^1, q^2 \in Q$ et $(q^1; \vec{w^1})\ R\ (q^1; \vec{w^3})$ implique $(q^2; c!a(\vec{w^1}))\ R\ (q^2; c!a\#(\vec{w^3}))$ et donc $s^2\ R\ s^4$.

- **Réception:** $t = (q^1, c?a, q^2)$. Nous avons donc $w_c^1 = aa_2 \dots a_n$, $w_c^3 \in \#^* a^+ \#^+ a_2^+\ \#^+ \dots \#^+ a_n^+ \#^+$ avec $(a_i)_{j=2,\dots,n} \in \Sigma$. Dans $R_d(A)$, nous commençons par supprimer tous les $\#$ en tête de w_c^3. Ceci grâce à la transition $(q^1, c?\#, q^1)$. Puis nous exécutons la réception $c?a$ grâce à la transition $(q^1, c?a, r_t)$. Ensuite nous supprimons tous les a restant en tête de c grâce à la transition $(r_t, c?a, r_t)$. Enfin, grâce à la transition $(r_t, c?\#, q^2)$ nous recevons un des $\#$ qui ne manqueront pas de suivre . Soit en résumé :

  $$(q^3; \vec{w^3}) \xrightarrow{(c?\#)^*}_d (q^3; \vec{w^5}) \xrightarrow{c?a}_d (r_t; c?a(\vec{w^5})) \xrightarrow{(c?a)^*}_d (r_t; \vec{w^6}) \xrightarrow{c?\#}_d (q^2; \vec{w^7})$$

  Nous obtenons donc bien :

  $$(q^3; \vec{w^3}) \overset{c?a}{\Longrightarrow}_d (q^2; \vec{w^7}) \text{ avec } w_c^7 \in \#^* a_2^+ \#^+ \dots \#^+ a_n^+ \#^+$$

  Prenons $s^4 = (q^2; \vec{w^7})$. Il est simple de vérifier dans ces conditions que :

  $$(q^1; \vec{w^1})\ R\ (q^3; \vec{w^3}) \text{ implique } (q^2; c?a(\vec{w^1}))\ R\ (q^2; \vec{w^7}) \text{ et donc } s^2\ R\ s^4.$$

- **Action interne:** $t = (q^1, \pi, q^2)$ avec π une action interne. Il suffit de prendre $s^4 = (q^2; \vec{w_3})$ car π ne modifie en rien les canaux.

2. Supposons maintenant que $q^3 \neq q^1$. Il existe donc $r_{t'} \in Q'$ tel que $q^3 = r_{t'}$. Deux possibilités s'offrent à nous :

   - $(q^3, c!\#, q^1) \in \delta'$. Nous avons alors :

     $$(q^3; \vec{w^3}) \xrightarrow{c!\#}_d (q^1; c!\#(\vec{w^3})) \text{ et donc } (q^3; \vec{w^3}) \overset{\varepsilon}{\Longrightarrow}_d (q^1; c!\#(\vec{w^3}))$$

     Or

     $$(q^1; \vec{w^1})\ R\ (q^3; \vec{w^3}) \text{ implique } (q^1; \vec{w^1})\ R\ (q^1; c!\#(\vec{w^3}))$$

     et du coup, nous sommes ramenés au cas 1. En effet, toute action effectuée à partir de $(q^1; \vec{w^1})$ dans A pourra être observationnellement effectuée à partir de $(q^1; c!\#(\vec{w^3}))$ et donc de $(q^3; \vec{w^3})$ dans $R_d(A)$, le tout en conservant la mise en relation par R des états d'arrivées.

- $(q^3, c?\#, q^1) \in \delta'$. Dans ce cas, $s^1\ R\ s^3$ implique que nous avons $w_c^1 = a_1 \dots a_n$, $w_c^3 \in a^*\#^+a_1^+\#^+a_2^+\#^+ \dots \#^+a_n^+\#^+$ et $(r_{t'}, c?a, r_{t'}) \in \delta'$ avec $(a_i)_{i=1,\dots,n} \in \Sigma$. Nous commençons donc par consommer, grâce à la transition $(r_{t'}, c?a, r_{t'})$, les a au début du canal c. Puis nous recevons l'un des $\#$ qui suivent en exécutant la transition $(q^3, c?\#, q^1) \in \delta'$. Notons $\vec{w^5}$ le contenu des canaux ainsi obtenu à partir de $\vec{w^3}$. Nous avons donc $(q^3; \vec{w^3}) \overset{\varepsilon}{\Longrightarrow}_d (q^1; \vec{w^5}))$. Or, $(q^1; \vec{w^1})\ R\ (q^3; \vec{w^3})$ implique $(q^1; \vec{w^1})\ R\ (q^1; \vec{w^5})$ et, comme précédemment, nous sommes ramenés au cas 1.

Démontrer que R est une relation de simulation observationnelle de $S_d(R_d(A))$ par $S(A)$, s'effectue de façon similaire. On montre que :

$$\begin{array}{c} s^3, s^4 \in EG(R_d(A)),\ s^1 \in EG(A), \\ s^3 \overset{t}{\longrightarrow}_d s^4,\ s^1\ R\ s^3 \end{array} \Longrightarrow \exists s^2 \in EG(A),\ s^2\ R\ s^4 \text{ et } s^1 \overset{\ell(t)}{\Longrightarrow} s^2$$

Le cas où t est une émission ($c!a$ ou $c!\#$) une duplication ou une action interne est immédiat et suit directement la définition de la relation R. Considérons celui de la réception d'un message d'un canal c. Suivant la définition de R, deux cas se présentent :

1. $q^3 = q^1 \in Q$.

   Si w_c^3 commence par un $\#$, alors $t = (q^3, c?\#, q^3)$. Clairement, l'état obtenu après cette réception reste en relation avec s_1. Il suffit donc de prendre $s^2 = s^1$ et de n'effectuer aucune transition dans $S(A)$.

   Si w_c^3 commence par a une lettre de Σ, alors w_c^1 commence par la même lettre et par la construction de $R_d(A)$, il existe $t' \in \delta$, $q^2 \in Q$ tels que : $t = (q^3, c?a, q^4)$, $t' = (q^1, c?a, q^2)$ et $q^4 = r_{t'}$. Nous pouvons donc prendre $s^2 = (q^2; c?a(\vec{w^1}))$ et nous obtenons bien $s^1 \overset{c?a}{\Longrightarrow} s^2$ avec $s^2\ R\ s^4$.

2. Supposons maintenant que $q^3 \neq q^1$. Il existe donc $r_{t'} \in Q'$ tel que $q^3 = r_{t'}$. Deux possibilités s'offrent à nous :

   - $t = (q^3, c?\#, q^4) \in \delta'$. Vu la définition de R, $q^4 = q^1$ et l'état obtenu après cette réception reste en relation avec s_1. Il suffit donc de prendre $s^2 = s^1$ et de n'effectuer aucune transition dans $S(A)$.
   - $(q^3, c?a, q^3) \in \delta'$. Vu la définition de R, l'état obtenu après cette réception reste en relation avec s_1. Il suffit donc de prendre $s^2 = s^1$ et de n'effectuer aucune transition dans $S(A)$ ❑

3.4.2 Indécidabilité des automates à files dupliquant des messages

Nous savons qu'un automate à files peut simuler une machine de Turing [BZ83]. Par conséquent, le Théorème précédent implique qu'il en est de même pour les automates à files dupliquant des messages. Les deux résultats qui suivent précisent les propriétés des automates à files dupliquant des messages.

Lemme 3.18 *Soit $A = (Q, q_o, \Sigma, \Pi, \delta)$ un automate à p files et $q \in Q$ un de ses états de contrôle. Alors*

$$(q; \vec{w}) \in \mathrm{Acc}(A) \textit{ si et seulement si } (q; f(\vec{w})) \in \mathrm{Acc}_d(R_d(A))$$

avec $f : (\Sigma^*)^p \to ((\Sigma \cup \{\#\})^*)^p$ la fonction qui à $\vec{w}$ associe $f(\vec{w})$ tel que pour tout $c \in \{1, \ldots, p\}$ et $a_i \in \Sigma$, si $w_c = a_1a_2 \ldots a_n$, alors $f(w)_c = a_1\#a_2\# \ldots \#a_n\#$.

Preuve : Le premier sens de l'équivalence se démontre par une simple récurrence sur la longueur des exécutions menant aux états accessibles de $Acc(A)$. Le second sens de l'équivalence est une conséquence directe du lemme précédent, à savoir la simulation observationnelle de $S_d(R_d(A))$ par $S(A)$ via la relation R. En effet, cette simulation implique que pour tout état accessible $(q; f(\vec{w}))$ de $Acc_d(R_d(A))$ il existe un état accessible $(q'; \vec{w'})$, de $Acc(A)$ tel que $(q'; \vec{w'})\ R\ (q; f(\vec{w}))$. Ce qui implique : $q = q'$ car $q \in Q$, et $\vec{w'} = \vec{w}$ ❑

Théorème 3.19 *Soit A un automate à files et $S_d(A) = (\mathrm{EG}(A), s_o, \to_d)$ le système de transition associé en considérant la possibilité de dupliquer des messages. Alors pour $S_d(A)$:*

1. *le problème de l'accessibilité est indécidable,*
2. *le problème de la terminaison est décidable de même que celui de la finitude de $\mathrm{Acc}_d(A)$,*
3. *le model-checking par la logique LTL est indécidable.*

Preuve :

1. C'est une conséquence directe du lemme précédent. En effet, grâce à ce lemme, nous pouvons réduire le problème, indécidable, de l'accessibilité d'un automate à files classique (c'est-à-dire sans possibilité de duplication des messages) à celui de l'accessibilité d'un automate à files avec possibilité de dupliquer des messages.

2. Ces problèmes sont ici immédiats. En effet, nous avons les implications suivantes :

   – S'il existe un état de $Acc_d(A)$ dont l'une des files est non vide, alors $Acc_d(A)$ est infini et il existe dans $S_d(A)$ une exécution infinie.
   – Si $Acc_d(A)$ est infini ou s'il existe dans $S_d(A)$ une exécution infinie, alors il existe un état de $Acc_d(A)$ dont l'une des files est non vide.

   La première implication est bien entendu due aux duplications. La seconde est triviale.

3. C'est une conséquence du Théorème 3.17. En effet, soit q un état de Q, notons $\hat{q}$ le prédicat sur Q vrai seulement en q. Nous avons l'équivalence suivante :

$$S(A) \models \Box\Diamond\hat{q} \text{ si et seulement si } S_d(R_d(A)) \models \Box\Diamond\hat{q} \text{ et } \neg\Diamond\Box\hat{q}$$

   La partie $\neg\Diamond\Box\hat{q}$ a pour rôle d'éviter dans $S_d(R_d(A))$ une séquence infinie de duplications successives. Cela se traduirait en effet par une séquence infinie où l'état de contrôle ne changerait pas.

   Or, le premier membre est indécidable. Et comme $(\Box\Diamond\hat{q}$ et $\neg\Diamond\Box\hat{q})$ est une formule de LTL, on en déduit que LTL est indécidable pour les automates à files dupliquant des messages.

3.5 Combinaisons des erreurs

Après avoir étudié séparément les erreurs de pertes, d'insertions et de duplications de messages, nous porterons notre attention jusqu'à la fin de ce chapitre, sur les diverses combinaisons de ces trois types d'erreurs.

Nous verrons que la combinaison la plus dramatique est celle des pertes et des insertions. Non seulement, les files n'exercent plus aucune contrainte sur le fonctionnement de l'automate sous-jacent, mais, de plus, la quasi-totalité de l'espace des états est accessible. La conjonction des erreurs de duplications et d'insertions n'est bien entendu en rien différente des seules insertions. En effet, une duplication est une insertion !

Les résultats concernant les erreurs de duplications combinées aux erreurs de pertes seront moins immédiats à obtenir. Mais nous verrons là encore que les duplications ne modifient pas la puissance du modèle auxquels elles s'appliquent. En effet, les propriétés les plus significatives seront conservées lors du passage des automates à files perdant des messages aux automates à files perdant et dupliquant des messages.

3.5.1 Pertes et insertions

Définition 3.11 Soit $A = (Q, q_o, \Sigma, \Pi, \delta)$ un automate à p files.

- La *fonction de transition perdant et insérant des messages*, $\longrightarrow_{l,i}$, est définie de la façon suivante. Soit $s = (q; x_1, \ldots, x_p)$ et $s' = (q'; x'_1, \ldots, x'_p)$ deux états de A, on note $s \longrightarrow_{l,i} s'$, ou de façon redondante $s \xrightarrow{t}_{l,i} s'$, si l'une des conditions de la Définition 2.20 ou celles qui suivent sont vérifiées :

   4. (perte)
   Il existe $c \in \{1, \ldots, p\}, a \in \Sigma$, et $w,\ w' \in \Sigma^*$ tels que $t = \tau_l$ avec
   $x_c = waw'$, $x'_c = ww'$,
   $x'_j = x_j$ pour tout $j \neq c$
   5. (insertion)
   Il existe $c \in \{1, \ldots, p\}, a \in \Sigma$, et $w, w' \in \Sigma^*$ tels que $t = \tau_i$ avec
   $x_c = ww'$, $x'_c = waw'$,
   $x'_j = x_j$ pour tout $j \neq c$

- Une *exécution perdant et insérant des messages* est une exécution basée sur la fonction de transition $\longrightarrow_{l,i}$.

- Un *automate à files perdant et insérant des messages* est un automate à files dont on considère implicitement que la relation de transition entre états est $\longrightarrow_{l,i}$.

Nous définissons de même $Acc_{l,i}(A)$ et $Acc_{l,i}(A, q)$ de façon identique à $Acc(A)$ et $Acc(A, q)$ sauf que nous considérons $\longrightarrow_{l,i}$ au lieu de $\longrightarrow$.

Lemme 3.20 *Soit $A = (Q, q_o, \Sigma, \Pi, \delta)$ un automate à files et $S_{l,i}(A) = (\mathrm{EG}(A), s_o, \rightarrow_{l,i})$ le système de transitions associé en considérant la possibilité de dupliquer ou d'insérer des messages. Alors pour tout état local $q \in Q$ nous avons :*

$$\begin{array}{lcll} \mathrm{Acc}_{l,i}(A, q) & = & (\Sigma^*)^p & \textit{S'il existe une séquence de transitions de } q_o \textit{ à } q \\ \mathrm{Acc}_{l,i}(A, q) & = & \emptyset & \textit{Sinon} \end{array}$$

Théorème 3.21 *Soit A un automate à files et $S_{l,i}(A) = (\mathrm{EG}(A), s_o, \rightarrow_{l,i})$ le système de transitions associé en considérant la possibilité de perdre ou d'insérer des messages. Alors, pour $S_{l,i}(A)$, les problèmes classiques sont trivialement décidables.*

3.5.2 Duplications et Insertions

Définition 3.12 Soit $A = (Q, q_o, \Sigma, \Pi, \delta)$ un automate à p files.

- La *fonction de transition dupliquant et insérant des messages*, $\longrightarrow_{d,i}$, est définie de la façon suivante. Soit $s = (q; x_1, \ldots, x_p)$ et $s' = (q'; x'_1, \ldots, x'_p)$ deux états de A, on note $s \longrightarrow_{d,i} s'$, ou de façon redondante $s \xrightarrow{t}_{d,i} s'$, si l'une des conditions de la Définition 2.20 ou celle qui suit est vérifiée :

  4. (duplication)
     Il existe $c \in \{1, \ldots, p\}, a \in \Sigma$, et $w, w' \in \Sigma^*$ tels que $t = \tau_d$ avec
     $x_c = waw'$, $x'_c = waaw'$,
     $x'_j = x_j$ pour tout $j \neq c$
  5. (insertion)
     Il existe $c \in \{1, \ldots, p\}, a \in \Sigma$, et $w, w' \in \Sigma^*$ tels que $t = \tau_i$ avec
     $x_c = ww'$, $x'_c = waw'$,
     $x'_j = x_j$ pour tout $j \neq c$

- Une *exécution dupliquant et insérant des messages* est une exécution basée sur la fonction de transition $\longrightarrow_{d,i}$.

- Un *automate à files dupliquant et insérant des messages* est un automate à files dont on considère implicitement que la relation de transition entre états est $\longrightarrow_{d,i}$.

Nous définissons de même $Acc_{d,i}(A)$ et $Acc_{d,i}(A, q)$ de façon identique à $Acc(A)$ et $Acc(A, q)$ sauf que nous considérons $\longrightarrow_{d,i}$ au lieu de $\longrightarrow$.

Théorème 3.22 *Soit A un automate à files et $S_{d,i}(A) = (\mathrm{EG}(A), s_o, \rightarrow_{d,i})$ le système de transition associé en considérant la possibilité de dupliquer et d'insérer des messages. Alors pour $S_{d,i}(A)$,*

1. *l'ensemble d'accessibilité, $\mathrm{Acc}_{d,i}(A)$, est reconnaissable. De plus, il existe un algorithme calculant une représentation de cet ensemble,*

2. *le problème de l'accessibilité est décidable,*

3. *le problème de la terminaison est décidable,*

3.5.3 Duplications et Pertes

Nous verrons dans cette partie que les résultats de décidabilité concernant l'accessibilité se transmettent aisément du cas de la perte vers le cas de la perte + duplication. En effet, à partir d'un automate à files perdant et dupliquant les messages, on peut construire un autre automate à files perdant les messages, mais ne les dupliquant pas, et tel que l'ensemble d'accessibilité ne soit pas modifié. Les résultats d'indécidabilité ne se transmettent pas, eux, aussi aisément. Il faudra en donner de nouvelles preuves. Pour cela, nous passerons par une bisimulation observationnelle avec les *front lossy systems*.

Définition 3.13 Soit $A = (Q, q_o, \Sigma, \Pi, \delta)$ un automate à p files.

- La *fonction de transition perdant et dupliquant les messages*, $\longrightarrow_{l,d}$, est définie de la façon suivante. Soit $s = (q; x_1, \ldots, x_p)$ et $s' = (q'; x'_1, \ldots, x'_p)$ deux états de A, on note $s \longrightarrow_{l,d} s'$, ou de façon redondante $s \xrightarrow{t}_{l,d} s'$ ou $s \xrightarrow{\ell(t)}_{l,d} s'$, si l'une des conditions de la Définition 2.20 ou l'une de celles qui suivent est vérifiée :

  4. (perte)
     Il existe $c \in \{1, \ldots, p\}, a \in \Sigma$, et $w,\ w' \in \Sigma^*$ tels que $t = \tau_l$ avec
     $x_c = waw'$, $x'_c = ww'$,
     $x'_j = x_j$ pour tout $j \neq c$
  5. (duplication)
     Il existe $c \in \{1, \ldots, p\}, a \in \Sigma$, et $w, w' \in \Sigma^*$ tels que $t = \tau_d$ avec
     $x_c = waw'$, $x'_c = waaw'$,
     $x'_j = x_j$ pour tout $j \neq c$

  Nous considérons que $\tau_{l,d}$ est une action inobservable et donc que $\ell(\tau_{l,d}) = \varepsilon$.

- Une *exécution perdant et dupliquant les messages* est une exécution basée sur la fonction de transition $\longrightarrow_{l,d}$.

- Un *automate à files perdant et dupliquant les messages* est un automate à files dont on considère implicitement que la relation de transition entre états est $\longrightarrow_{l,d}$.

- Nous notons $S_{l,d}(A) = (EG(A), s_o, \rightarrow_{l,d})$ le système de transitions associé à A considéré comme un automate à files perdant et dupliquant les messages.

Nous définissons de même $Acc_{l,d}(A)$ et $Acc_{l,d}(A,q)$ de façon identique à $Acc(A)$ et $Acc(A,q)$ sauf que nous considérons $\longrightarrow_{l,d}$ au lieu de $\longrightarrow$.

Nous allons maintenant construire le «complété dupliquant» d'un automate à files, c'est-à-dire, l'automate à files qui, tout en gardant la sémantique classique, aura un ensemble d'accessibilité similaire à celui obtenu si des duplications étaient intervenues.

Définition 3.14 Soit $A = (Q, q_o, \Sigma, \Pi, \delta)$ un automate à files. Nous notons $D(A)$ l'automate obtenu à partir de A de la façon suivante.
$D(A) = (Q', q_o, \Sigma, \Pi, \delta')$ avec :

- $Q' = Q \cup \{r_t | \exists t \in \delta,\ t = (q, c!a, q')\}$
- δ' est le plus petit ensemble de transitions tel que :

$$\begin{aligned} t = (q, c!a, q') \in \delta &\implies (q, c!a, r_t),\ (r_t, c!a, r_t),\ (r_t, \tau, q') \in \delta' \\ (q, c?a, q') \in \delta &\implies (q, c?a, q') \in \delta' \\ (q, \pi, q') \in \delta, \pi \in \Pi &\implies (q, \pi, q') \in \delta' \end{aligned}$$

La Figure 3.5 résume la transformation essentielle de A pour obtenir $D(A)$.

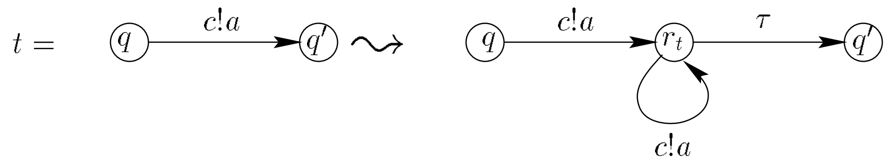


FIG. 3.5 – *Transformation de A en D(A)*

En considérant les pertes possibles de messages, nous obtenons le résultat suivant.

Lemme 3.23 *Soit A un automate files. Alors nous avons :*

$$\mathrm{Acc}_{l,d}(A) = \mathrm{Acc}_l(D(A))$$

Preuve : L'inclusion $Acc_{l,d}(A) \subseteq Acc_l(D(A))$ se montre par une simple récurrence sur la longueur des exécutions menant aux états de $Acc_{l,d}(A)$. La démarche est la suivante. Les émissions, réceptions et pertes de $S_{l,d}(A)$ sont simulées par leurs transitions correspondantes (au sens des implications de la Définition 3.14) dans $S_l(D(A))$. Par contre, pour simuler la duplication d'une lettre a, nous recherchons la transition de type $(q, c!a, r_t)$ ayant émis cette lettre dans $S_l(D(A))$. Il suffit alors d'effectuer la transition $(r_t, c!a, r_t)$ immédiatement après.

L'inclusion inverse se montre de façon similaire ❏

Corollaire 3.24 *Soit A un automate à files. Alors l'ensemble d'accessibilité* $\mathrm{Acc}_{l,d}(A)$ *est reconnaissable et l'accessibilité est un problème décidable pour* $S_{l,d}(A)$.

Preuve : L'ensemble $Acc_l(D(A))$ est reconnaissable, Théorème 3.4. Or, par le lemme précédent, $Acc_{l,d}(A) = Acc_l(D(A))$. Par conséquent, $Acc_{l,d}(A)$ est reconnaissable. On procède de même pour la décidabilité de l'accessibilité ❏

Abordons maintenant les résultats l'indécidabilité. Pour cela, nous montrons que nous pouvons supprimer l'impact des duplications et donc nous ramener au cas de la seule perte des messages.

Notre problème consiste à trouver un moyen permettant de distinguer les différentes origines d'une séquence aa dans un canal:

- fonctionnement normal du système,
- duplication de la lettre a,
- suppression d'une lettre intermédiaire. Par exemple, b dans aba.

Nous partons de la même idée développée à la Section 3.4, c'est-à-dire, l'introduction d'un séparateur entre les lettres afin de contrôler les duplications. Le problème est que ce séparateur peut lui aussi être perdu ! Qu'à cela ne tienne, nous pouvons considérer que la perte d'un séparateur est équivalent à la perte de la lettre qui suivait et donc que, lorsque deux lettres distinctes se suivront, seule la première devra être prise en compte. C'est cette solution qui a été adoptée dans [CFP96]. Nous arrivions alors à montrer qu'il y avait une certaine équivalence entre les exécutions d'un système perdant et dupliquant des messages et les exécutions d'un système se contentant de les perdre. Malheureusement la gestion des différents cas rendait la preuve fastidieuse et finalement imprécise.

Nous adoptons ici une autre démarche qui permet aussi d'obtenir un résultat plus fort : il est possible de *corriger* les erreurs de duplication apparaissant dans un environnement pouvant perdre et dupliquer les messages.

Nous gardons l'introduction du séparateur entre les lettres. Mais pour éviter certaines des complications dues à la perte possible de ce séparateur, nous ne considérons que les pertes en tête des canaux. Ce qui ne pose pas de problèmes car on peut se ramener au cas de la perte simple par une extension du Lemme 3.11 page 40.

Théorème 3.25 *Soit $A = (Q, q_o, \Sigma, \Pi, \delta)$ un automate à files, $S_{fl}(A) = (\mathrm{EG}\,(A\,), s_o, \rightarrow_{fl})$ et $S_{fl,d}(R_d(A)) = (\mathrm{EG}\,(R_d(A)\,), s_o, \rightarrow_{fl,d})$ les systèmes de transitions associés. Alors en considérant les actions c!# et c?# de $S_{fl,d}(R_d(A))$ comme étant non-observables, $S_{fl}(A)$ et $S_{fl,d}(R_d(A))$ sont observationnellement bisimulaires.*

Preuve :

Nous montrons que la relation R' définie ci-après est une relation de bisimulation observationnelle entre $S_{fl}(A)$ et $S_{fl,d}(R_d(A))$. R' est définie de façon identique à la relation R de la Définition 3.10, sauf que nous remplaçons la dernière condition par :

$\exists t = (q^1, c?a, q^2) \in \delta,\ q = q^2,\ q' = r_t,\ \forall i \in \{1, \ldots, p\}\ \exists (a_j)_{j=1,\ldots,n} \in \Sigma,\ w_i = a_1 \ldots a_n,$

$$\begin{array}{lll} \text{si } i = c & \text{alors} & \exists b \in \Sigma,\ w'_i \in b^* \#^+ a_1^+ \#^+ a_2^+ \#^+ \ldots \#^+ a_n^+ \#^+ \\ \text{si } i \neq c & \text{alors} & w'_i \in \#^* a_1^+ \#^+ a_2^+ \#^+ \ldots \#^+ a_n^+ \#^+ \end{array}$$

La différence porte sur le b : dans R, on demandait que b soit égal à a. Nous avons donc $R \subseteq R'$.

Premièrement, montrons que R' est une relation de simulation observationnelle de $S_{fl}(A)$ par $S_{fl,d}(R_d(A))$. Considérons $s^1 = (q^1; \vec{w^1})$, $s^2 = (q^2; \vec{w^2}) \in EG(A\,)$, $s^3 = (q^3; \vec{w^3}) \in EG(R_d(A)\,)$ et $t \in \delta$ tels que : $s^1 \xrightarrow{t}_{fl} s^2$ et $s^1\ R'\ s^3$. Montrons qu'il existe $s^4 \in EG(R_d(A)\,)$ tel que $s^3 \overset{\ell(t)}{\Longrightarrow}_{fl,d} s^4$ et $s^2\ R'\ s^4$. Il y a quatre cas suivant que t est une émission, une réception, une action interne ou une perte de message. Or, le Théorème 3.17 indique que $\rightarrow_d$ simule $\rightarrow$ par la relation R. Il en est donc de même pour la relation R'. Il ne reste ainsi que le cas de la perte à considérer. Nous supposons donc que $t = \tau_{fl}$ et que cette perte concerne le canal c.

Suivant la définition de R', deux cas se présentent :

- $q^3 = q^1 \in Q$.

  Supposons que ce soit le canal c qui subisse cette perte du message de tête. Nous avons donc : $w_c^1 = a_1 a_2 \ldots a_n$, $w_c^3 \in \#^* a_1^+ \#^+ a_2^+ \#^+ \ldots \#^+ a_n^+ \#^+$, $w_c^2 = a_1 \ldots a_n$ et $\ell(t) = \varepsilon$. Il suffit donc au sein de $S_{fl,d}(R_d(A))$ de perdre au début de w_c^3 les # et toutes les répétitions de a_1. Nous avons donc $(q^3; \vec{w^3}) \xrightarrow{(\tau_{fl})^*}_{fl,d} (q^3; \vec{w^4})$ et $(q^1; \vec{w^1})\ R'\ (q^3; \vec{w^4})$, avec $w_c^4 \in \#^+ a_2^+ \#^+ \ldots \#^+ a_n^+ \#^+$.

– Supposons maintenant que $q^3 \neq q^1$. Il existe donc $r_{t'} \in Q'$ tel que $q^3 = r_{t'}$. Deux possibilités s'offrent à nous :

  – $(q^3, c'!\#, q^1) \in \delta'$. Si $c' \neq c$, on effectue les mêmes opérations que précédemment. Si $c = c'$, nous avons $w_c^1 = a_1 a_2 \dots a_n$, et $w_c^3 \in \#^{k_1} a_1^{k_2} \#^+ \dots \#^+ a_n^+$. Comme c'est a_1 qui est perdu dans w_c^1 pour obtenir w_c^2, il suffit de perdre $\#^{k_1} a_1^{k_2}$ dans w_c^3 pour obtenir w_c^4. Et nous prenons $s^4 = (q^3; \vec{w^4})$. De là, on vérifie aisément les conditions concernant s^1, s^2, s^3 et s^4.

  – $(q^3, c'?\#, q^1) \in \delta'$. Si $c \neq c'$, on effectue le même traitement que dans le cas où $q^3 = q^1$. Supposons maintenant que $c = c'$. Nous avons alors, $w_c^1 = a_1 a_2 \dots a_n$, et $w_c^3 \in b^{k_1} \#^{k_2} a_1^+ \#^+ \dots \#^+ a_n^+ \#^+$. Comme c'est a_1 qui est perdu dans w_c^1 pour obtenir w_c^2, il suffit de perdre $b^{k_1} \#^{k_2}$ dans w_c^3 pour obtenir w_c^4. Et nous prenons $s^4 = (q^3; \vec{w^4})$. De là, on vérifie aisément les conditions concernant s^1, s^2, s^3 et s^4.

La démonstration que R' est une relation de simulation observationnelle de $S_{fl,d}(R_d(A))$ par $S_{fl}(A)$, s'effectue de façon similaire. On montre que :

$$\begin{array}{c} s^3, s^4 \in EG(R_d(A)),\ s^1 \in EG(A), \\ s^3 \xrightarrow{t}_{fl,d} s^4,\ s^1\ R'\ s^3 \end{array} \Longrightarrow \exists s^2 \in EG(A),\ s^2\ R'\ s^4 \text{ et } s^1 \overset{\ell(t)}{\Longrightarrow}_{fl} s^2$$

Pour les mêmes raisons que celles de la simulation précédente, nous n'avons à traiter que le cas où $t = \tau_{fl}$. Considérons que cette perte affecte le canal c.

Suivant la définition de R', deux cas se présentent :

1. $q^3 = q^1 \in Q$.
   Si $w_c^3 \in \#^{k_1} a^{k_2} (\Sigma \cup \{\#\})^*$ avec $a \in \Sigma$ et ($k_1 \geq 1$ ou $k_2 \geq 2$), on prend $s^2 = s^1$ et on n'effectue aucune transition dans $S_{fl}(A)$. Par contre, si $w_c^3 \in a\#(\Sigma \cup \{\#\})^*$, w_1 commence alors par un a. On prend donc pour w^2, w^1 privé de la première lettre du contenu du canal c, lettre que l'on considère comme ayant été perdue. De là, on vérifie aisément les conditions concernant s^1, s^2, s^3 et s^4.

2. Supposons maintenant que $q^3 \neq q^1$. Il existe donc $r_{t'} \in Q'$ tel que $q^3 = r_{t'}$. Deux possibilités s'offrent à nous :

   – $(q^3, c'?\#, q^1) \in \delta'$. Nous traitons ce sous-cas comme le cas 1.

   – $(q^3, c'!\#, q^1) \in \delta'$. Si $c \neq c'$, on effectue le même traitement que le cas 1. Supposons maintenant que $c = c'$. Si $w_c^3 \in b^{k_1} \#^{k_2} a (\Sigma \cup \{\#\})^*$ avec $a, b \in \Sigma$ et ($k_1 \geq 1$ ou $k_2 \geq 2$), alors on prend $s^2 = s^1$ et on n'effectue aucune transition dans $S_{fl}(A)$. Si par contre $w_c^3 \in \#a(\Sigma \cup \{\#\})^*$, w_1 commence par un a. On prend donc, pour w^2,

w^1 privé de la première lettre du contenu du canal c, lettre que l'on considère comme ayant été perdue. De là, on vérifie aisément les conditions concernant s^1, s^2, s^3 et s^4 ❑

Une première interprétation du Théorème précédent est que $R_d(A)$ reste insensible à la duplication de messages, même lorsque des pertes de messages peuvent intervenir en tête des canaux.

Théorème 3.26 *Soit A un automate à files et $S_{l,d}(A) = (\mathrm{EG}\,(A), s_o, \rightarrow_{l,d})$ le système de transition associé en considérant la possibilité de dupliquer des messages. Alors pour $S_{l,d}(A)$:*

1. *l'ensemble d'accessibilité $\mathrm{Acc}_{l,d}(A)$ est reconnaissable,*
2. *le problème de l'accessibilité est décidable,*
3. *le problème de la terminaison est décidable, de même que celui de la finitude de $\mathrm{Acc}_{l,d}(A)$,*
4. *le model-checking par la logique LTL est indécidable,*
5. *il n'existe pas d'algorithme permettant de calculer une représentation de $\mathrm{Acc}_{l,d}(A)$.*

Preuve :

1. et 2. Simple reprise du Corollaire 3.24.

3. Mêmes arguments que dans la preuve du Théorème 3.19.

4. Nous allons réduire le problème RSP, Définition 3.5 page 35, pour $S_{fl}(A)$ à la décidabilité d'une formule de LTL dans $S_{l,d}(R_d(A))$. Or, RSP étant indécidable dans $S_{fl}(A)$, Définition 3.5 et Théorème 3.14 page 43, il s'en suivra qu'il est aussi indécidable dans $S_{l,d}(R_d(A))$.

   Soit $q \in Q$ un état de contrôle de A. Montrons que nous avons l'équivalence suivante :

   $$S_{fl}(A) \models \Box\Diamond\hat{q} \ \text{ si et seulement si } \ S_{fl,d}(R_d(A)) \models \phi(q)$$

   avec $\phi(q) = \Box\Diamond(\hat{q} \text{ et } \neg\Box\hat{q})$.

La formule de gauche est l'expression sous forme d'une formule LTL du problème RSP. La formule de droite est quasiment la même, sauf que nous interdisons au système de rester indéfiniment à l'état de contrôle q. Ceci pour lui empêcher de procéder à une infinité de duplications sans changer d'état de contrôle. À partir de ces remarques, l'équivalence précédente est une conséquence directe du Théorème 3.25. Or, de même que pour tout automate à files A', $S_{fl}(A')$ et $S_l(R_d(A'))$ se simulent réciproquement, de même $S_{fl,d}(A')$ et $S_{l,d}(R_d(A'))$ se simulent réciproquement. C'est une simple extension du Lemme 3.11 page 40. Par conséquent, $S_{fl,d}(R_d(A))$ satisfait $\phi(q)$ si et seulement si $S_{l,d}(R_d(A))$ satisfait $\phi(q)$. Ce qui entraîne l'équivalence recherchée :

$$S_{fl}(A) \models \Box\Diamond\hat{q} \text{ si et seulement si } S_{l,d}(R_d(A)) \models \phi(q)$$

D'où l'indécidabilité de LTL pour $S_{l,d}$.

5. Nous procédons de façon similaire à la démonstration équivalente pour les *lossy systems*, Théorème 3.9 page 37. Soit A un automate à files, nous transformons A en A' de la façon indiquée par la Figure 3.3 du Théorème 3.9, sauf qu'au lieu d'émettre $c!\#$, nous émettons $c!\#.c!\$$. Nous avons alors l'équivalence suivante :

$$S_{l,d}(A') \models \phi(q) \text{ si et seulement si } Acc_{l,d}(A', q, c) = (\#^+\$^+)^*$$

Par conséquent, si on avait une représentation reconnaissable de $Acc_{l,d}(A')$, on pourrait par projection en obtenir une de $Acc_{l,d}(A', q, c)$ et donc décider du membre droit de l'équivalence précédente. Or, nous avons vu au point 4, que le membre gauche est indécidable. Par conséquent, on ne peut calculer une représentation effective de l'ensemble reconnaissable d'accessibilité d'un automate à files pouvant perdre et dupliquer les messages.

Chapitre 4

Automates à communication half-duplex

Au chapitre précédent, nous avons vu que la possibilité de pertes ou d'insertions de messages par les canaux de communication entraîne une diminution de la puissance d'expression du modèle des automates communicants. Cette diminution du pouvoir d'expression du modèle a alors été mise à profit pour mettre en œuvre des algorithmes de vérification.

Dans ce chapitre, nous étudions une autre hypothèse permettant la vérification. Cette hypothèse porte sur l'un des modes de communication entre deux entités : la communication *half-duplex*.

Rappelons-en la définition : une communication entre deux entités est dite half-duplex si à tout instant l'information ne circule que dans un seul sens. Bien sûr, pour qu'il y ait communication, il faut que l'entité qui reçoit l'information puisse répondre. Cela se fait sans contredire la règle. Il suffit tout simplement d'alterner le sens de la transmission des messages ; ceci, soit périodiquement, soit à la demande explicite de l'entité à qui c'est le tour d'émettre.

Ce mode de communication half-duplex résulte de contraintes propres au média utilisé. C'est le cas par exemple de la communication par ondes radio où l'émission et la réception se font sur la même fréquence. L'autre mode de communication où les deux entités peuvent s'adresser simultanément des messages est dit *full-duplex*.

Pour les systèmes constitués de deux automates ayant une communication half-duplex, nous montrons que l'ensemble d'accessibilité est reconnaissable et que nous pouvons, de plus, en calculer une représentation sous la forme d'une décomposition de Mezei ou d'un QDD (cf. Chapitre 2). Cette description effective de l'ensemble d'accessibilité nous permet de décider des problèmes de l'accessibilité, de la finitude de l'ensemble d'accessibilité et de la présence ou non d'erreurs de progression (*deadlocks* ou *réceptions non spécifiées*).

Ce résultat est d'autant plus intéressant que nous pouvons aussi décider si deux automates utilisent entre eux une communication half-duplex, ce qui est un préalable indispensable à la mise en œuvre des techniques proposées.

Nous nous intéressons aussi dans ce chapitre à des systèmes impliquant plus de deux automates. Or, la notion de communication half-duplex n'a pas été définie dans ce cas. Nous en proposons alors deux définitions. La première est de considérer comme half-duplex tout système où chaque paire d'automates a une communication half-duplex. Nous verrons qu'avec cette hypothèse, un système de trois automates suffit pour simuler une machine de Turing, ce qui interdit toute possibilité de vérification. La seconde définition possible est de considérer comme half-duplex tout système où un seul canal est utilisé à la fois. Nous verrons que cette hypothèse permet les mêmes possibilités de vérification que pour les systèmes half-duplex de deux automates.

Nous proposerons alors une «bonne» généralisation : les *automates à communication quasi-stable*. Cette nouvelle classe permet de considérer des systèmes de plus de deux automates, tout en préservant les bonnes propriétés de vérification des systèmes half-duplex à deux automates.

L'essentiel de ce chapitre est publié dans [CF97].

4.1 Les systèmes half-duplex de deux automates sont vérifiables

Après avoir défini les exécutions et les systèmes half-duplex, nous commencerons, dans cette section, par présenter le lemme essentiel de ce chapitre. Ce lemme nous permettra, dans un premier temps, d'exhiber, pour toute exécution half-duplex d'un système de deux automates, une exécution canonique aboutissant au même état. Nous verrons alors, dans un second temps, que nous pouvons décrire l'ensemble des états atteints par ces exécutions canoniques, et donc, par équivalence, l'ensemble d'accessibilité.

Remarque 4.1 À partir de maintenant, et pour le reste du chapitre, un système $S = (A_1, A_2)$ de deux automates gère implicitement deux canaux : le premier, qui va de A_1 à A_2, est le canal d'*émission* de A_1 et de *réception* de

A_2 ; le second, qui va de A_2 à A_1, est le canal d'émission de A_2 et de réception de A_1.

Nous posons aussi l'hypothèse supplémentaire qu'aucun des automates impliqués n'effectue d'action interne. Cette hypothèse, qui a pour seul but de simplifier les énoncés et les preuves, se retrouve aussi dans les articles [GMY84] et [Mou89] dont nous généralisons ici les résultats. Un automate communicant se définit alors par un quadruplet (Q, q_o, Σ, δ) et non plus par un quintuplet $(Q, q_o, \Sigma, \Pi, \delta)$, où Π serait l'alphabet des actions internes.

Définition 4.1 Soit $S = (A_1, A_2)$ un système à deux canaux.

- Un état $(q_1, q_2; x_1, x_2)$ de S est dit *half-duplex* si $x_1 = \lambda$ ou $x_2 = \lambda$.
- Une exécution σ de S est dite *half-duplex* si tous les états parcourus par σ sont half-duplex.
- S est dit *half-duplex* si tous ses états accessibles (et donc aussi ses exécutions) sont half-duplex.

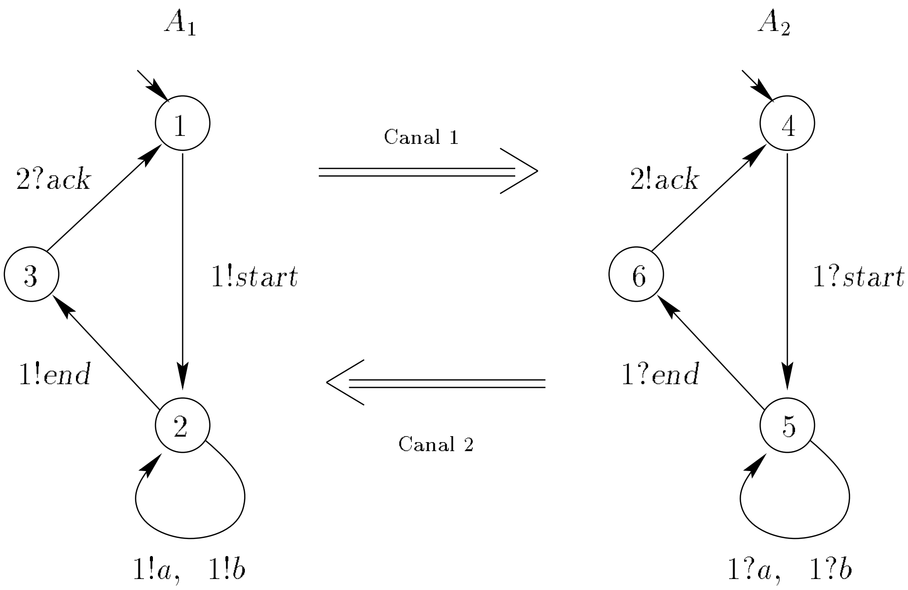


FIG. 4.1 – *Le protocole S_1*

Exemple: Considérons le système S_1 de la Figure 4.1. Nous avons :

$$\begin{aligned} Acc(S) \;=\Big\{ & (1,4;\lambda,\lambda),\ (2,4;start\,(a+b)^*,\lambda),\ (3,4;start\,(a+b)^*\,end,\lambda), \\ & (2,5;(a+b)^*,\lambda),\ (3,5;(a+b)^*\,end,\lambda),(3,6;\lambda,\lambda), \\ & (3,4;\lambda,ack)\Big\} \end{aligned}$$

Comme l'un des canaux est vide pour tous ses états accessibles, S_1 est half-duplex.

Voyons maintenant comment transformer une exécution half-duplex d'un système de deux automates en une exécution plus simple. Pour cela, considérons le système S_1 de la Figure 4.1 et l'exécution σ donnée ci-dessous qui aboutit à l'état $s = (2,5;ba,\lambda)$.

$$\begin{aligned} \sigma \;=\; & (1,1!start,2)\,(2,1!a,2)\,(4,1?start,5)\,(2,1!b,2) \\ & (2,1!b,2)\,(2,1!a,2)\,(5,1?a,5)\,(5,1?b,5) \end{aligned}$$

La suite des états parcourus par σ est la suivante :

$(1,4;\lambda,\lambda)$, $(2,4;start,\lambda)$, $(2,4;start\ a,\lambda)$, $(2,5;a,\lambda)$, $(2,5;ab,\lambda)$, $(2,5;abb,\lambda)$, $(2,5;abba,\lambda)$, $(2,5;bba,\lambda)$, $(2,5;ba,\lambda)$

On constate que le second canal de ces états est toujours vide. Par conséquent, σ est une exécution half-duplex.

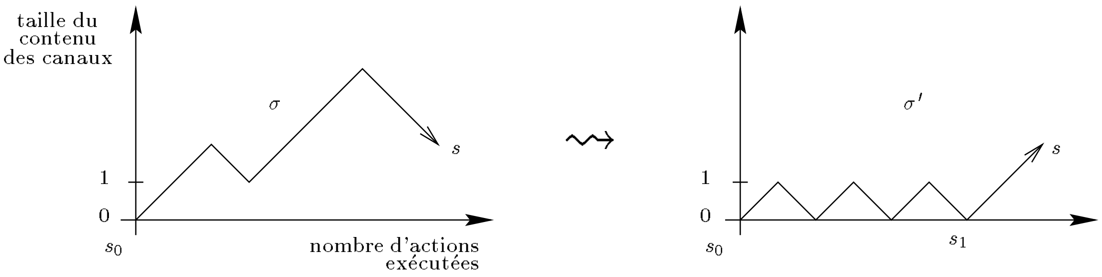
Cette exécution σ est un entrelacement d'une séquence de transition $\sigma_{|1}$ de l'automate A_1 et d'une séquence de transitions $\sigma_{|2}$ de l'automate A_2, avec :

$$\begin{aligned} \sigma_{|1} &= (\mathbf{1},\mathbf{1}!start,\mathbf{2})\,(\mathbf{2},\mathbf{1!a},\mathbf{2})\,(\mathbf{2},\mathbf{1!b},\mathbf{2})\,(\mathbf{2},\mathbf{1!b},\mathbf{2})\,(\mathbf{2},\mathbf{1!a},\mathbf{2}) \\ \sigma_{|2} &= (4,1?start,5)\,(5,1?a,5)\,(5,1?b,5) \end{aligned}$$

Nous allons maintenant «rejouer» ces deux séquences de transitions afin d'obtenir une nouvelle exécution σ' où chaque émission de l'un des automates est immédiatement suivie par la réception correspondante, si elle existe, de l'autre automate. Nous obtenons ainsi :

$$\begin{aligned} \sigma' \;=\; & (\mathbf{1},\mathbf{1}!start,\mathbf{2})\,(4,1?start,5)\,(\mathbf{2},\mathbf{1!a},\mathbf{2})\,(5,1?a,5)\,(\mathbf{2},\mathbf{1!b},\mathbf{2}) \\ & (5,1?b,5)\,(\mathbf{2},\mathbf{1!b},\mathbf{2})\,(\mathbf{2},\mathbf{1!a},\mathbf{2}) \end{aligned}$$

Cette nouvelle exécution est toujours half-duplex et aboutit au même état $s = (2,5;ba,\lambda)$.



(2.1) (2.2)

FIG. 4.2 – *Transformation des exécutions half-duplex*

La Figure 4.2 fait ressortir les caractéristiques de σ'. Cette exécution se décompose en deux parties :

- la première aboutit à un état stable (i.e. dont tous les canaux sont vides), noté s_1 sur la figure, dont tous les états intermédiaires contiennent au plus un message par canal.
- la seconde est exclusivement constituée d'émissions dans un seul des deux canaux.

Admettons pour l'instant que tout état d'un système half-duplex S puisse être atteint par une exécution décomposable de cette façon. Nous pouvons alors déduire un algorithme de calcul de l'ensemble d'accessibilité. Cet algorithme procède en deux étapes résumées ci-dessous :

- la première étape calcule l'ensemble *fini*, que nous notons $Acc_1(S)$, de tous les états accessibles par une exécution ne passant que par des états contenant au plus un message par canal.
- la seconde étape effectue, à partir de chacun des états obtenus à la première étape, l'ensemble des séquences d'émissions obtenues en parcourant un seul des deux automates.

C'est grâce à ce parcours d'un automate fini mentionné à la seconde étape, que nous obtenons un ensemble d'accessibilité reconnaissable.

Passons maintenant à l'approche formelle.

Définition 4.2 Soit S un système de deux automates communicants.

- Une exécution s_o, t_1, $s_1, \ldots, t_n$, s_n de S est dite *1-bornée* si tous les états $(s_i)_{i=1,\ldots,n}$ ont au plus 1 message par canal.

- Nous notons $Acc_1(S)$ l'ensemble de tous les états accessibles par une exécution 1-bornée.

Remarque 4.2 L'ensemble $Acc_1(S)$ d'un système S est fini et calculable.

Rappelons qu'étant donnée σ une séquence de transitions d'un système S, nous notons $\sigma_{|i}$ la projection de σ sur δ_i l'ensemble des transitions du $i^{\text{ième}}$ automate de S, $\ell(\sigma)$ la séquence d'actions de communications contenue dans σ et $(\ell(\sigma))_{j!}$ la séquence des messages émis par σ dans le $j^{\text{ième}}$ canal. Nous notons aussi $\delta_i^!$ l'ensemble des transitions de δ_i dont l'action est une émission. Nous sommes maintenant en mesure d'énoncer le lemme correspondant à la transformation de la Figure 4.2.

Lemme 4.1 *Soient $S = (A_1, A_2)$ un système de deux automates et σ une de ses exécutions half-duplex aboutissant à l'état $s = (q_1, q_2; x_1, x_2)$. Alors il existe une exécution half-duplex $\sigma^1\sigma^2$ telle que :*

1. $\forall i \in \{1,2\},\ \ \sigma_{|i} = (\sigma^1\sigma^2)_{|i}$
2. $s_o \xrightarrow{\sigma^1} s_1 \xrightarrow{\sigma^2} s$, *avec s_1 un état stable et σ^1 une exécution 1-bornée.*
3. *Il existe $i \in \{1,2\}$ tel que $\sigma^2 \in (\delta_i^!)^*$, $x_i = (\ell(\sigma^2))_{i!}$ et $x_{3-i} = \lambda$*

Preuve : Par récurrence sur la longueur de σ. Les conditions sont vérifiées pour $|\sigma| = 0$: il suffit de prendre $\sigma^1 = \sigma^2 = \varepsilon$. Supposons maintenant que le lemme soit vrai pour $|\sigma| \leq n$ et considérons que $|\sigma| = n+1$. Il existe alors une exécution σ' et une transition t telles que $\sigma = \sigma' t$. L'exécution σ étant half-duplex, il en est de même pour ses préfixes, dont σ'. Par conséquent, il existe une exécution half-duplex $\sigma^{1'}\sigma^{2'}$ telle que :

1. $\forall i \in \{1,2\},\ \ \sigma'_{|i} = (\sigma^{1'}\sigma^{2'})_{|i}$
2. $s_o \xrightarrow{\sigma^{1'}} s_1 \xrightarrow{\sigma^{2'}} s' = (q'_1, q'_2; x'_1, x'_2) \xrightarrow{t} s$, avec s_1 un état stable et $\sigma^{1'}$ une exécution 1-bornée.
3. Il existe $i \in \{1,2\}$ tel que $\sigma^{2'} \in (\delta_i^!)^*$, $x'_i = (\ell(\sigma^{2'}))_{i!}$ et $x'_{3-i} = \lambda$

Il y a deux cas, suivant que l'action de t est une émission ou une réception.

$t = (q'_j, j!a, q_j) \in \delta_j$, avec $j \in \{1,2\}$. Si $\sigma^{2'}$ est une séquence vide, nous pouvons prendre $\sigma^1 = \sigma^{1'}$ et $\sigma^2 = t$. Si, par contre, $\sigma^{2'}$ n'est pas une séquence vide, alors le 3[e] point de l'énumération précédente implique

que $|x'_i| \geq 1$ et $x'_{3-i} = \lambda$. En supposant que $j \neq i$, nous avons alors $x_j \neq \lambda$ et $x_{3-j} = x'_i$, avec $x'_i \neq \lambda$, ce qui contredit le fait que l'état s est half-duplex. Il s'ensuit que $j = i$ et nous pouvons choisir $\sigma^1 = \sigma^{1'}$ et $\sigma^2 = \sigma^{2'}t$.

$t =$ $(q'_j, (3-j)?a, q_j) \in \delta_j$, avec $j \in \{1,2\}$. Puisque $x'_{3-i} = \lambda$, la réception de t ne se fait pas sur le canal $(3-i)$ mais sur le canal i qui se doit d'être non vide ; ce qui signifie que $3 - j = i$ et $|x'_i| > 1$. L'état s_1 étant stable et $\sigma^{2'}$ uniquement constitué d'émissions, le message que reçoit t juste après l'exécution de $\sigma^{1'}$ correspond au premier message émis par $\sigma^{2'}$. Nous avons alors $\sigma^{2'} = t'\sigma^{3'}$, avec $t' \in \delta_i$ et $\sigma^{3'} \in (\delta_i^!)^*$. Grâce à l'asynchronisme des deux automates, nous pouvons exécuter à partir de l'état s_1 la séquence $t't\sigma^{3'}$, qui aboutit aussi à l'état s. Nous choisissons alors $\sigma = \sigma^{1'}t't$ et $\sigma^2 = \sigma^{3'}$ ❏

La définition qui suit tire profit de la forme des exécutions proposées par ce lemme, et particularise un ensemble d'états que nous montrerons égal à l'ensemble d'accessibilité pour un système half-duplex.

Définition 4.3 Soit $S = (A_1, A_2)$ un système de deux automates $A_i = (Q_i, q_{o_i}, \Sigma, \delta_i)$. Nous notons $H(S)$ l'ensemble d'états définis de la façon suivante.

$$H(S) = \bigcup_{\substack{(q'_1, q'_2; \lambda, \lambda) \in Acc_1(S) \\ (q_1, q_2) \in Q_1 \times Q_2}} \{q_1\} \times \{q_2\} \times L_1(q'_1, q_1) \times L_2(q'_2, q_2)$$

avec $L_i(q'_i, q_i)$ le langage régulier défini par l'automate fini $A'_i(q'_i, q_i) = (Q_i, q'_i, \Sigma, \delta'_i, \{q_i\})$ tel que q'_i soit l'état initial, q_i l'unique état final et $\delta'_i = \{(q, a, q') \mid (q, i!a, q') \in \delta_i\}$ la relation de transition.

Remarque 4.3 $H(S)$ est reconnaissable comme union finie de produits de langages reconnaissables (cf. Chapitre 2, [Ber79]). De plus, la description de $H(S)$ est effective puisque celles des $L_i(q'_i, q_i)$ sont effectives.

Lemme 4.2 *Soit $S = (A_1, A_2)$ un système half-duplex. Alors :*

$$\mathrm{Acc}(S) = H(S)$$

Preuve : L'inclusion $Acc(S) \subseteq H(S)$ est une conséquence directe du Lemme 4.1. En effet, soit $s = (q_1, q_2; x_1, x_2)$ un état accessible de S. Par le Lemme 4.1, il existe une exécution $\sigma^1\sigma^2$ de S telle que $s_o \xrightarrow{\sigma^1} (q'_1, q'_2; \lambda, \lambda) \xrightarrow{\sigma^2} s$, avec σ^1 1-bornée et σ^2 composée uniquement d'actions d'émissions dans l'un des canaux $i \in \{1, 2\}$. Par conséquent, $(q'_1, q'_2; \lambda, \lambda) \in Acc_1(S)$, $x_i \in L_i(q'_i, q_i)$ et avec $j = 3 - i$, $x_j = \lambda \in L_j(q'_j, q_j)$ car $q'_j = q_j$. Il s'ensuit que $s \in H(S)$.

Considérons maintenant l'inclusion inverse. Si $s = (q_1, q_2; x_1, x_2) \in H(S)$, alors, par le Lemme 4.1, il existe $s_1 = (q'_1, q'_2; \lambda, \lambda) \in Acc_1(S)$ et deux séquences de transitions, σ^1 dans A_1 et σ^2 dans A_2, composées uniquement d'actions d'émissions et telles que pour $i \in \{1, 2\}$, $q_i \in \delta_i(q'_i, \ell(\sigma^i))$ et $x_i = \big(\ell(\sigma^i)\big)_{i!}$. Par conséquent, nous pouvons construire à partir de s_1 l'exécution suivante :

$$s_1 = (q'_1, q'_2; \lambda, \lambda) \xrightarrow{\sigma^1} (q_1, q'_2; (\ell(\sigma^1))_{1!}, \lambda) \xrightarrow{\sigma^2} (q_1, q_2; (\ell(\sigma^1))_{1!}, (\ell(\sigma^2))_{2!}) = s$$

Le système S étant half-duplex, $(\ell(\sigma^1))_{1!}$ ou $(\ell(\sigma^2))_{2!}$ sont vides et donc l'une des séquences σ^1 ou σ^2 est nulle.

En résumé, nous avons $s_o \xrightarrow{*} s_1 \xrightarrow{*} s$ et donc $s \in Acc(S)$ ❏

Exemple: Considérons le système S_1 de la Figure 4.1. Nous pouvons vérifier que $Acc(S) = H(S)$. En particulier, l'ensemble $(2, 5; (a+b)^*, \lambda)$ est obtenu en prenant $q'_1 = q_1 = 2$ et $q'_2 = q_2 = 5$ dans la Définition 4.3.

Le dernier lemme nous permet d'aboutir au résultat annoncé.

Théorème 4.3 *L'ensemble d'accessibilité d'un système half-duplex $S = (A_1, A_2)$, avec $A_i = (Q_i, q_{o_i}, \Sigma, \delta_i)$, est reconnaissable et calculable en temps :*

$$O\Big((|\Sigma| + 1)^3 |Q_1 \times Q_2| (|Q_1| + |Q_2|)\Big)$$

Preuve : La reconnaissabilité effective de $Acc(S)$ est une conséquence immédiate de la remarque et du lemme précédents. Considérons la complexité. Selon la Définition 4.3, il nous suffit de déterminer les états stables de $Acc_1(S)$ puisque à partir de ces états, un simple parcours à travers les actions d'émissions de l'un des automates nous décrit les autres états accessibles.

Or, déterminer les états stables de $Acc_1(S)$ revient à parcourir le graphe formé par les états de S accessibles par une exécution 1-bornée. Ce graphe a au maximum $|Q_1 \times Q_2|.(|\Sigma| + 1)^2$ nœuds et chacun de ces nœuds a au maximum

$|\Sigma \times Q_1| + |\Sigma \times Q_2|$ successeurs. Ce parcours se fait alors (cf. [CLR90] par exemple) en :

$$O\Big((|\Sigma| + 1)^3 |Q_1 \times Q_2| (|Q_1| + |Q_2|)\Big)$$

❑

Corollaire 4.4 *Les problèmes suivants sont décidables pour les systèmes half-duplex.*

- *l'accessibilité,*
- *la détection des états de non progression,*
- *la finitude de l'ensemble d'accessibilité.*

Preuve : Conséquences immédiates du Théorème précédent et du Théorème 2.7 page 24 ❑

Graphe Symbolique d'accessibilité

Nous pouvons décrire l'ensemble d'accessibilité d'un système half-duplex. Or, comme cet ensemble peut être infini, nous pouvons considérer que c'est une *description symbolique*. La définition qui suit précise cette idée que nous développerons au chapitre suivant.

Définition 4.4 Soit $S = (A_1, A_2)$ un système de deux automates tels que $A_i = (Q_i, q_{o_i}, \Sigma, \delta_i)$.

- L'*ensemble symbolique d'accessibilité* de S, $ESAcc(S)$, est le suivant.

  $$ESAcc(S) = \Big\{(\vec{q}; Acc(S, \vec{q})\,) \mid \vec{q} \in Q_1 \times Q_2\Big\}$$

  Un élément de $ESAcc(S)$ est ainsi composé de deux parties. La première est un état de contrôle du système. La deuxième est l'ensemble des contenus accessibles des canaux pour cet état de contrôle.

- Le *graphe symbolique* de S, $GS(S)$, est un graphe étiqueté dont les nœuds sont les éléments de $ESAcc(S)$ et dont la relation de transitions est la suivante :

  $$(\vec{q}; Acc(S, \vec{q})\,) \xrightarrow{x} (\vec{q'}; Acc(S, \vec{q'}))$$

$$\text{si et seulement si}$$
$$Acc(S, \vec{q}) \neq \emptyset,\ Acc(S, \vec{q'}) \neq \emptyset \text{ et } (\vec{q}, x, \vec{q'}) \in \delta_S.$$

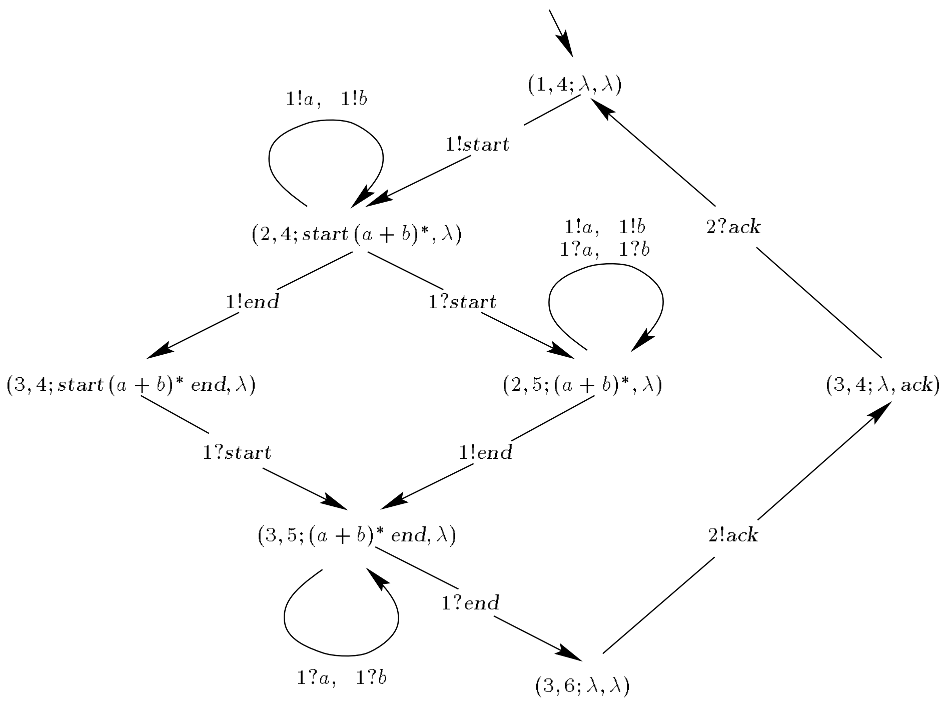


FIG. 4.3 – *Graphe symbolique d'accessibilité de* S_1

Exemple: La Figure 4.3 donne une représentation du graphe symbolique d'accessibilité de S_1.

Le graphe symbolique d'accessibilité d'un système est conceptuellement proche du *graphe minimal de couverture* d'un réseau de Petri [Fin93]. En effet, tout comme pour celui-ci, toute exécution d'un système S est un chemin dans $GS(S)$ et tout état accessible de S est *couvert* (ici est un élément de) par un nœud de ce graphe. Toutefois, le graphe symbolique d'accessibilité d'un système contient plus d'informations que le graphe minimal de couverture d'un réseau de Petri, car ses nœuds décrivent *exactement* l'ensemble d'accessibilité, ce qui n'est pas le cas pour le graphe minimal de couverture.

4.1.1 Décidabilité du caractère half-duplex

Certaines hypothèses rendent décidables des problèmes de vérification des automates communicants. Mais, malheureusement, ces hypothèses peuvent être elles mêmes indécidables ou conjecturées indécidables. C'est notamment le cas pour les systèmes monogènes [Fin82] ou linéaires [JJ93]. Il n'est donc pas possible de déterminer sur quels systèmes appliquer les algorithmes de vérification déduits.

Nous montrons maintenant que le cas des systèmes half-duplex est différent car le caractère half-duplex d'un système de deux automates est décidable.

Lemme 4.5 *Un système $S = (A_1, A_2)$, avec $A_i = (Q_i, q_{o_i}, \Sigma, \delta_i)$, n'est pas half-duplex si, et seulement si, il existe $s = (q_1, q_2; \lambda, \lambda) \in \mathrm{Acc}_1(S)$ et $a, b \in \Sigma$ tels que $\delta_1(q_1, 1!a) \neq \emptyset$ et $\delta_2(q_2, 2!b) \neq \emptyset$.*

Preuve : S n'est pas half-duplex si et seulement si un de ses états accessibles n'est pas half-duplex. Soit σ, l'une des plus courtes exécutions menant à un tel état s^3. Comme les canaux de s^3 ne sont pas vides, $|\sigma| > 0$. Par conséquent, il existe une exécution σ', une transition t et un état accessible s^2 tels que $\sigma = \sigma' t$ et $s_o \xrightarrow{\sigma'} s^2 \xrightarrow{t} s^3$. Comme σ est l'une des plus courtes exécutions non half-duplex, σ', qui est strictement plus courte est half-duplex. Et grâce au Lemme 4.1, il existe une exécution $\sigma^1\sigma^2$ telle que :

- $s_o \xrightarrow{\sigma^1} s_1 \xrightarrow{\sigma^2} s^2 \xrightarrow{t} s^3$, avec s_1 un état stable et σ^1 une exécution 1-bornée,
- il existe $i \in \{1, 2\}$ tel que $\sigma^2 \in (\delta_i^!)^*$, $x_i^2 = (\ell(\sigma^2))_{i!}$ et $x_{3-i}^2 = \lambda$, avec, pour $k \in \{1, 2, 3\}$, $s^k = (q_1^k, q_2^k; x_1^k, x_2^k)$

Comme les deux canaux de s^3 ne sont pas vides, contrairement au canal $(3 - i)$ de s^2, $|\sigma^2| > 0$, $t \in \delta_{3-i}^!$ et l'état origine de t est $q_{3-i}^2 = q_{3-i}^1$. Comme $|\sigma^2| > 0$ et $\sigma^2 \in (\delta_i^!)$, la première transition de σ^2 est une transition d'émission ayant q_i^1 pour état d'origine. Par conséquent, il existe une transition d'émission partant de q_1^1 et une autre partant de q_2^1. Nous pouvons donc prendre $s = s^1$.

Inversement, s'il existe un état $s = (q_1, q_2; \lambda, \lambda) \in Acc_1(S)$ tel qu'il existe une transition d'émission partant de q_1 et une autre partant de q_2, alors il suffit d'exécuter successivement ces deux transitions pour aboutir à un état accessible non half-duplex, ce qui implique que S est non half-duplex ❑

Théorème 4.6 *Le caractère half-duplex d'un système* $S = (A_1, A_2)$ *avec* $A_i = (Q_i, q_{o_i}, \Sigma, \delta_i)$*, est décidable en temps :*

$$O\Big((|\Sigma| + 1)^3 |Q_1 \times Q_2| (|Q_1| + |Q_2|)\Big)$$

Preuve : Conséquence directe du lemme précédent. Il suffit de parcourir $Acc_1(S)$ à la recherche d'un état satisfaisant les conditions de ce lemme. La preuve de la complexité de cette recherche est la même que celle du Théorème 4.3 page 70 ❑

4.1.2 Liens avec les systèmes définis dans [GMY84] et [Mou89]

Dans [GMY84], Gouda et al ont cherché à définir une classe simple de systèmes à deux automates ne présentant pas d'erreurs de progression (deadlocks ou réceptions non spécifiées). Nous rappellerons la définition de cette classe dans ce qui suit. Leur résultat est la décidabilité, pour cette classe, du problème de la finitude de l'ensemble d'accessibilité. Mountassir, [Mou89], a aussi abouti à ce résultat pour une classe similaire.

Nous montrons ici que la classe de systèmes définie par Gouda et al est incluse dans celle des systèmes half-duplex à deux automates. Ce qui entraîne, de fait, la décidabilité du problème de l'accessibilité et même la possibilité de calculer une description reconnaissable de l'ensemble d'accessibilité.

Définition 4.5 [GMY84]

- Nous notons Φ l'isomorphisme qui, à une séquence d'action de communication, i.e. un mot sur l'alphabet $\{1, 2\} \times \{!, ?\} \times \Sigma$, fait correspondre sa séquence symétrique ; c'est-à-dire :

  $$\Phi : \Big(\{1, 2\} \times \{!, ?\} \times \Sigma\Big)^* \longrightarrow \Big(\{1, 2\} \times \{!, ?\} \times \Sigma\Big)^*$$

  tel que, pour $i \in \{1, 2\}$ et $a \in \Sigma$, $\Phi(i!a) = i?a$ et $\Phi(i?a) = i!a$.

- Soit $A = (Q, q_o, \Sigma, \delta)$ un automate communicant. Nous notons $L(A)$ le langage *régulier* reconnu par l'automate $A' = (Q, q_o, \Sigma', \delta, F)$, avec $\Sigma' = \{1, 2\} \times \{!, ?\} \times \Sigma$ comme alphabet et $F = Q$ comme ensemble des états finaux.

- Un système $S = (A_1, A_2)$ est dit *compatible* si $L(A_2) = \Phi(L(A_1))$.

- Un système $S = (A_1, A_2)$, avec $A_i = (Q_i, q_{o_i}, \Sigma, \delta_i)$, est dit *symétrique* s'il existe une bijection γ de Q_1 dans Q_2 telle que :
  - $\gamma(q_{o_1}) = q_{o_2}$
  - $(q_1, x, q_1') \in \delta_1$ si et seulement si $\big(\gamma(q_1), \Phi(x), \gamma(q_1')\big) \in \delta_2$

Rappelons aussi la notion d'état local *mixte* d'un automate communicant.

Définition 4.6 Soit $A = (Q, q_o, \Sigma, \delta)$ un automate communicant et $q \in Q$ un de ses états locaux.

- q est dit *état d'émission* si les actions de toutes les transitions issues de q sont des émissions.
- q est dit *état de réception* si les actions de toutes les transitions issues de q sont des réceptions.
- q est dit *état mixte* s'il n'est ni un état d'émission ni un état de réception.

La classe définie dans [GMY84] est celle des systèmes compatibles $S = (A_1, A_2)$, où les A_i sont déterministes et sans état local mixte. La classe définie dans [Mou89] est similaire, sauf que les systèmes doivent être symétriques au lieu de compatibles.

Remarque 4.4 Tout système $S = (A_1, A_2)$ symétrique est aussi compatible. Par conséquent, la classe de [GMY84] est plus générale que celle de [Mou89].

Théorème 4.7 *Un système compatible $S = (A_1, A_2)$, où les A_i sont déterministes et sans état local mixte, est half-duplex.*

Preuve : Montrons par contradiction que, si S n'est pas half-duplex alors, A_2 possède un état local mixte.

Supposons que S n'est pas half-duplex. Par le Théorème 4.1, il existe une exécution 1-bornée aboutissant à un état stable $s = (q_1, q_2; \lambda, \lambda)$, telle que il existe $a_1, a_2 \in \Sigma$, $q_1' \in Q_1$, $q_2' \in Q_2$, $t_1 = (q_1, 1!a_1, q_1') \in \delta_1$ et $t_2 = (q_2, 2!a_2, q_2') \in \delta_2$. Soit σ l'une de ces exécutions les plus courtes, alors $\Phi\big(\ell(\sigma_{|1})\big) = \ell(\sigma_{|2})$. Reportons pour l'instant la démonstration de ce dernier point.

La séquence d'action $\ell(\sigma_{|1}).1!a_1$ fait partie de $L(A_1)$. Par conséquent, la compatibilité du système implique l'existence d'une séquence d'action μ de

$L(A_2)$ telle que $\mu.1?a \in L(A_2)$ et $\Phi\big(\ell(\sigma_{|1}).1!a_1\big) = \Phi\big(\ell(\sigma_{|1})\big).1?a_1 = \mu.1?a$. Or, $\Phi\big(\ell(\sigma_{|1})\big) = \ell(\sigma_{|2})$; par conséquent, $\ell(\sigma_{|2}).1?a \in L(A_2)$. Mais comme A_2 est déterministe, cela implique que $q_2 = \delta_2(q_{o_2}, \ell(\sigma_{|2}))$ et qu'il existe $t' = (q_2, 1?a, q_2'') \in \delta_2$, avec $q_2'' \in Q_2$. Par conséquent, l'état local q_2 est origine d'une transition d'émission t_2 et d'une transition de réception t''. Il est donc mixte. Ce qui contredit l'hypothèse de départ.

Il nous reste à montrer que $\Phi\big(\ell(\sigma_{|1})\big) = \ell(\sigma_{|2})$. Pour cela, prouvons une propriété plus générale :

$$\text{Soit } s^1 \text{ un état stable tel que } s_o \xrightarrow{\sigma^1} s^1 \xrightarrow{\sigma^2} s, \text{ avec } \sigma = \sigma^1\sigma^2,$$
$$\text{alors } \Phi\big(\ell({\sigma^2}_{|1})\big) = \ell({\sigma^2}_{|2}).$$

Par induction sur $|\sigma^2|$. Le cas de base où $|\sigma^2| = 0$ est immédiat. Supposons que la propriété est vraie pour $|\sigma^2| \leq n < |\sigma|$ et considérons le cas où $|\sigma^2| = n+1$. Comme s^1 est un état stable, la première transition de σ^2 est une transition d'émission. De plus, comme s est aussi un état stable, ce message émis devra être reçu ; σ^2 contient alors une autre transition qui est une transition de réception. Par conséquent, $|\sigma^2| \geq 2$ et nous avons :

$$s^1 \xrightarrow{t'} s^2 \xrightarrow{t''} s^3 \xrightarrow{\sigma^3} s, \text{ avec } \sigma^2 = t't''\sigma^3,\ \ t', t'' \in \delta,\ \ \sigma^3 \in \delta^*$$
$$\text{et pour } k \in \{1,2,3\},\ s^k = (q_1^k, q_2^k; x_1^k, x_2^k).$$

Supposons que $t' = (q_1^1, 1!a, q_1^2) \in \delta_1$. Alors, suivant la nature de t'', plusieurs cas sont à envisager.

$t'' = (q_1^2, 1!b, q_1^3) \in \delta_1$ n'est pas possible car ceci impliquerait que

$$s^3 = (q_1^3, q_2^1; ab, \lambda)$$

et contredirait l'hypothèse que σ est 1-bornée.

$t'' = (q_1^2, 2?b, q_1^3) \in \delta_1$ n'est pas possible car $x_2^2 = \lambda$.

$t'' = (q_2^1, 2!b, q_2^2) \in \delta_2$ n'est pas possible car nous aurions

$$s^1 = (q_1^1, q_2^1; \lambda, \lambda) \in Acc_1(S),\ \delta_1(q_1^1, 1!a) \neq \emptyset \text{ et } \delta_2(q_2^1, 2!b) \neq \emptyset,$$

ce qui contredirait l'hypothèse que σ est l'une des plus courtes exécutions menant à un tel état.

Par conséquent, $t'' = (q_2^1, 1?b, q_2^2) \in \delta_2$ avec $b = a$. Nous avons ainsi :

$$\Phi(\ell((t't'')_{|1})) = \Phi(\ell(t')) = 1?a = \ell(t'') = \ell((t't'')_{|2})$$

Or, par l'hypothèse d'induction, nous avons $\Phi(\ell({\sigma^3}_{|1})) = \ell({\sigma^3}_{|2})$ et donc

$$\Phi(\ell({\sigma^2}_{|1})) = \ell({\sigma^2}_{|2})$$

En prenant $\sigma^2 = \sigma$, nous obtenons bien le résultat escompté : $\Phi(\ell(\sigma_{|1})) = \ell(\sigma_{|2})$
❑

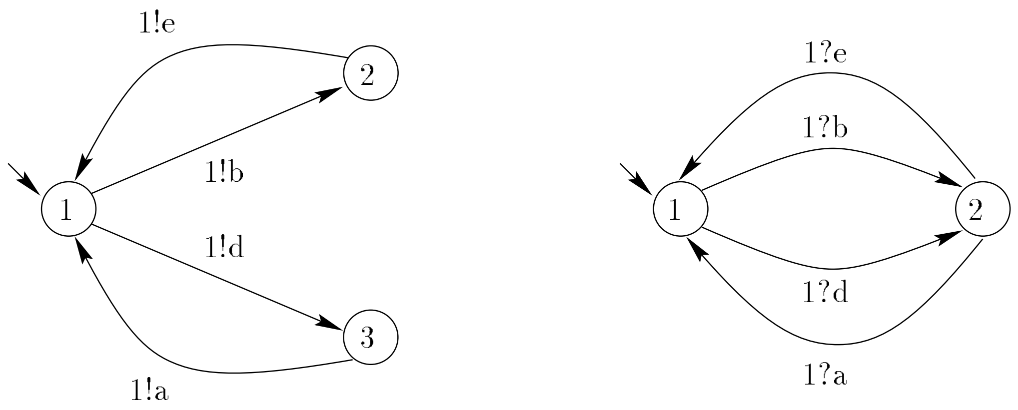


FIG. 4.4 – *Système half-duplex mais non compatible*

La classe des systèmes half-duplex englobe donc celle définie dans [GMY84]. La Figure 4.4 montre que cette inclusion est stricte.

4.2 Généralisation à plus de deux automates

À la section précédente, nous avons montré que l'ensemble d'accessibilité des systèmes half-duplex de deux automates est effectivement reconnaissable. Dans cette section, nous cherchons à étendre cette bonne propriété à des systèmes de plus de deux automates. Nous proposons deux généralisations de la notion de communication half-duplex : la *généralisation naturelle* et la *généralisation restrictive*. Puis nous définirons une classe analysable de systèmes, les systèmes *quasi-stables*.

Généralisation naturelle

Cette généralisation consiste à définir comme half-duplex tout système dont chaque paire d'automates a une communication half-duplex classique. Malheureusement, un tel système composé de trois automates peut simuler une machine de Turing.

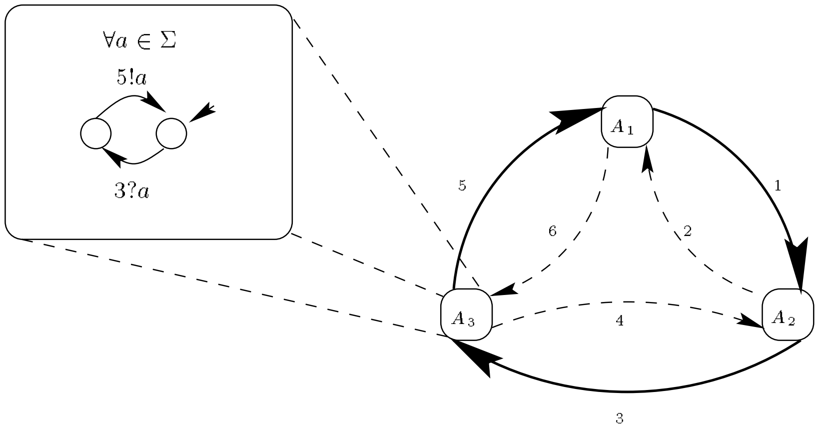


FIG. 4.5 – *Simulation d'une machine de Turing*

Pour cela, considérons la simulation, proposée dans [BZ83], d'une machine de Turing par deux automates communicants. Cette simulation fait intervenir deux automates A_1 et A_2 qui n'ont pas une communication half-duplex. Mais rajoutons un troisième automate A_3 qui aura pour simple rôle de faire transiter les messages allant de A_2 à A_1. La Figure 4.5 illustre cette transformation. Alors quitte à rajouter des canaux factices (ceux de numéro 2,4 et 6 dans la Figure 4.5), nous obtenons un système half-duplex à trois automates toujours capable de simuler une machine de Turing. Par conséquent, nous avons le résultat suivant.

Théorème 4.8 *Un système half-duplex de trois automates est capable de simuler une machine de Turing.*

Par conséquent, cette *généralisation naturelle* n'est pas analysable. Nous ne pouvons pas étendre les résultats valables pour les systèmes de deux automates.

Généralisation restrictive

Cette généralisation consiste à définir comme half-duplex tout système dont les états accessibles ont au plus un canal non vide. C'est le cas par exemple de plusieurs machines communiquant, par ondes radio, sur la même fréquence. À un instant donné, une seule machine a le droit d'émettre car, sinon, il y aurait brouillage des transmissions.

On peut montrer, en suivant la même démarche utilisée lors des sections précédentes, que cette classe de systèmes a les mêmes propriétés que les systèmes half-duplex de deux automates.

4.2.1 Automates utilisant des canaux Quasi-Stables

Une autre méthode de généralisation est de partir de la propriété clé des sections précédentes, c'est-à-dire, la transformation des exécutions par le Lemme 4.1.

Dans ce qui suit, rappelons que nous notons *émet*(j) l'indice de l'*unique* automate qui peut émettre dans le canal j, que $\delta^{j!}$ désigne l'ensemble des transitions émettant des messages dans le canal j, que $\delta^{j?}$ désigne l'ensemble des transitions recevant des messages du canal j et que $\sqcup\!\sqcup$ est l'opérateur de mélange des mots. C'est-à-dire :

$$x \sqcup\!\sqcup y = \{y_1 x_1 y_2 \dots y_n x_n y_{n+1} \mid x = x_1 \dots x_n,\ y = y_1 \dots y_{n+1}\}$$

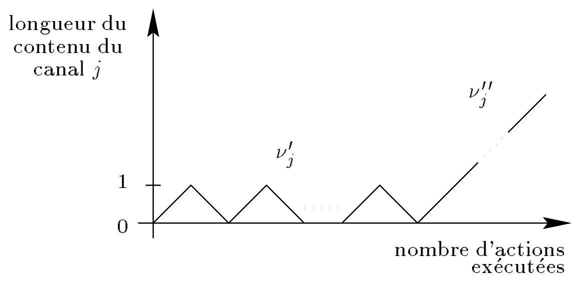


FIG. 4.6 – *Évolution de la taille du contenu d'un canal lors d'une exécution quasi-stable*

Définition 4.7 Soit S un système d'automates communicants avec p canaux.

- Une exécution σ est dite *quasi-stable* si :
  (a) $\sigma \in \nu'_1\nu''_1 \sqcup\!\sqcup \dots \sqcup\!\sqcup \nu'_p\nu''_p$, avec $\nu'_j \in (\delta^{j!}\delta^{j?})^*$ et $\nu''_j \in (\delta^{j!})$. La séquence ν'_j est dite *phase quasi-stable* de j et la séquence ν''_j sa *phase croissante*. La Figure 4.6 illustre l'évolution de la taille du contenu d'un canal lors de ces deux phases.
  (b) Pour chaque factorisation $\sigma = \sigma^1 t \sigma^2$, avec $t \in \nu''_j$, $\sigma^2{}_{|\acute{e}met(j)} \in (\delta^{j!})$. En d'autres termes, l'automate qui envoie des messages dans le canal j, durant la phase croissante de ce canal, n'exécute plus d'actions impliquant un autre canal.
- S est dit *quasi-stable*, ou ayant *des canaux quasi-stables*, si tous ses états accessibles peuvent être atteints par une exécution quasi-stable.

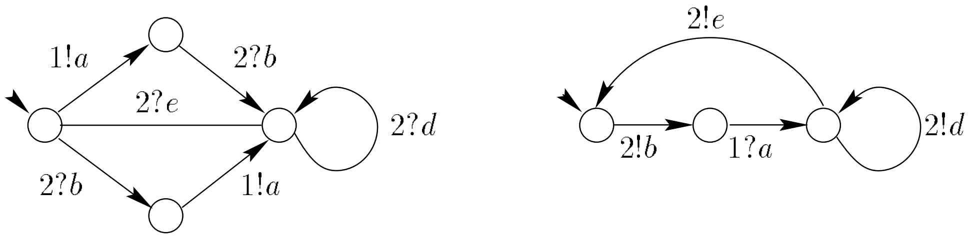


FIG. 4.7 – *Un système quasi-stable mais non half-duplex*

Remarque 4.5 En utilisant le Lemme 4.1 de la page 68, on peut vérifier qu'un système half-duplex de deux automates est quasi-stable. Toutefois, la Figure 4.7 montre que cette inclusion est stricte.

Les systèmes quasi-stables peuvent être caractérisés par une propriété plus effective, qui nous permettra de calculer leurs ensembles d'accessibilité.

Lemme 4.9 *Soit S un système de n automates communicants et p canaux. L'état $s = (\vec{q}; x_1, \ldots, x_p)$ est accessible par une exécution quasi-stable si et seulement si il peut être atteint par une exécution σ telle que :*

1. *$\sigma = \sigma^o \sigma^1 \sigma^2 \ldots \sigma^p$, avec σ^o 1-bornée et aboutissant à un état stable, et $\sigma^j \in (\delta^{j!})^*$.*

2. *$\forall j, k \in \{1, \ldots, p\},\ i \neq k,\ \Big(\sigma^j \neq \varepsilon \text{ et } \sigma^k \neq \varepsilon \Longrightarrow \text{émet}(j) \neq \text{émet}(k)\Big)$,*

3. *$\forall j \in \{1, \ldots, p\},\ x_j = \big(\ell(\sigma^j)\big)_{j!}$.*

Preuve : Le sens «si» est immédiat. Comme σ^o est 1-bornée, entre deux émissions concernant un canal donné dans σ^o, il y a forcément une réception concernant ce même canal. Et comme σ^o aboutit à un état stable, tout message émis doit être reçu. Nous avons donc $\sigma^o{}_{|\text{émet}(j)} \in (\delta^{j!}\delta^{j?})^*$. Par conséquent, il suffit de prendre $\nu'_j = \sigma^o{}_{|\text{émet}(j)}$ et $\nu''_j = \sigma^j$.

Pour le sens «seulement si», considérons $\nu = \nu' t' \nu'' t'' \nu'''$ une exécution quasi-stable telle que t' et t'' soient deux transitions de la phase croissante du canal j et que ν'' ne contienne aucune transition de $\delta^{j!}$. Les transitions t' et t'' sont exécutées par le même automate $A_{\text{émet}(j)}$. De plus, l'automate $A_{\text{émet}(j)}$ n'exécute aucune action dans ν'', Définition 4.7(b). Donc t' et t'' sont deux transitions consécutives dans $A_{\text{émet}(j)}$; c'est-à-dire, $t' = (q', x, q'')$, $t'' = (q'', x', q''') \in \delta_{\text{émet}(j)}$. Comme t' et t'' sont indépendantes des transitions de ν'' et que ν'' ne contient pas de réception du canal j, nous déduisons que la séquence de transitions $\nu' t' t'' \nu'' \nu'''$ est exécutable et aboutit au même état s. En réitérant une telle transformation, il est possible de transformer l'exécution ν en une exécution σ telle que $\sigma = \sigma^o \sigma^{j_1} \ldots \sigma^{j_p}$, avec σ^o 1-bornée, aboutissant à un état stable et σ^{j_r} regroupant toutes les transitions de la phase croissante du canal j. Comme toutes les séquences σ^{j_r} sont constituées de transitions d'émission effectuées par des automates différents, Définition 4.7(b), elles peuvent échanger leur position. Ce faisant, on satisfait les conditions requises par le lemme pour σ ❏

Définition 4.8 Soit $S = (A_1, \ldots, A_n)$ un système de n automates $A_i = (Q_i, q_{o_i}, \Sigma, \delta_i)$ et p canaux. Nous notons $Q(S)$ l'ensemble d'états défini de la façon suivante.

$$Q(S) = \bigcup_{(\vec{q'};\vec{\lambda}) \in Acc_1(S)} \quad \bigcup_{\substack{J \subseteq \{1, \ldots, p\} \\ \forall j,k \in J, \\ j \neq k \\ \Longrightarrow \\ émet(j) \neq émet(k)}} \left\{ (\vec{q}; \vec{x}) \middle| \begin{array}{l} j \in J \Rightarrow x_j \in L(j, q'_{émet(j)}, q_{émet(j)}) \\ j \notin J \Rightarrow x_j = \lambda \\ \forall i \notin \{émet(j) | j \in J\}, q'_i = q_i \end{array} \right\}$$

Avec $L(j, q'_{émet(j)}, q_{émet(j)})$ le langage régulier défini par l'automate fini

$$A'_j(q'_{émet(j)}, q_{émet(j)}) = (Q_{émet(j)}, q'_{émet(j)}, \Sigma, \delta'_j, \{q_{émet(j)}\})$$

avec $q'_{émet(j)}$ l'état initial, $q_{émet(j)}$ l'unique état final et $\delta'_j = \{(q, a, q') \mid (q, j!a, q') \in \delta_{émet(j)}\}$ la relation de transition.

$Q(S)$ est effectivement reconnaissable comme union finie de produits de langages effectivement reconnaissables (cf. Chapitre 2).

Lemme 4.10 *Soit S un système d'automates communicants. Alors,*

$$\mathrm{Acc}(S) = Q(S)$$

Preuve : Le Lemme 4.9 indique qu'un état accessible de S est atteignable à partir d'un état s' de $Acc_1(S)$, par une exécution dans laquelle chaque automate effectue uniquement des émissions dans un seul canal. La Définition 4.8 décrit exactement l'ensemble des états atteignables de cette façon.

Théorème 4.11 *L'ensemble d'accessibilité d'un système quasi-stable*

$$S = (A_1, \ldots, A_n), \text{ avec } A_i = (Q_i, q_{o_i}, \Sigma, \delta_i),$$

est reconnaissable et calculable en temps :

$$O\left((|\Sigma|+1)^{n+1}\left(\prod_{i=1}^{n}|Q_i|\right)\left(\sum_{i=1}^{p}|Q_i|\right)\right)$$

Preuve :
Similaire à celle du Théorème 4.3 ❑

De ce lemme, nous déduisons le résultat suivant.

Corollaire 4.12 *Les problèmes suivants sont décidables pour les systèmes quasi-stables.*

- *l'accessibilité,*
- *la détection des états de non progression,*
- *la finitude de l'ensemble d'accessibilité.*

Preuve : Conséquences immédiates du Théorème précédent et du Théorème 2.7 page 24 ❑

Tout comme pour les systèmes half-duplex à deux automates, le caractère quasi-stable d'un système est décidable. Mais la preuve nécessite des arguments différents.

Théorème 4.13 *Le caractère quasi-stable pour un système S d'automates communicants est décidable.*

Preuve : Pour avoir une démonstration rapide, nous faisons référence aux concepts et résultats du chapitre suivant.

À partir de la Définition 4.8 et du Lemme 4.10, décider si S est quasi-stable revient à décider si $Q(S) = Acc(S)$. Or, $Q(S)$ est un ensemble effectivement reconnaissable. Et les ensembles reconnaissables forment une classe de représentation d'états symboliques, Définition 5.4 page 94. De plus, $Q(S) \subseteq Acc(S)$. Par conséquent, il est décidable de savoir si $Q(S)$, qui forme un état symbolique, est égal à l'ensemble d'accessibilité, Lemme 5.2 page 93

❑

Chapitre 5

Vérification par Ensembles Reconnaissables et Transitions Symboliques

Nous savons que l'ensemble d'accessibilité d'un automate à files est non récursif. Il n'existe donc pas d'algorithme général permettant de le calculer. Toutefois, ce calcul est possible dans certains cas. C'est ce que nous avons vu au chapitre précédent pour les systèmes half-duplex à deux automates.

Nous verrons dans ce chapitre une approche permettant d'approximer, et dans certains cas de calculer exactement, l'ensemble d'accessibilité d'un automate à files. Cette approche consiste à définir et à utiliser des transitions abstraites : les *transitions symboliques*.

Pour illustrer notre problématique, considérons A l'automate à une file représenté ci-dessous :

A :

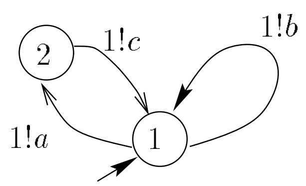


FIG. 5.1 – *Exemple introductif*

Il est clair que l'ensemble d'accessibilité de A est infini. En effet, en partant de l'état initial $(1;\lambda)$ on peut, par exemple, effectuer un nombre quelconque d'actions d'émission $1!b$ et générer ainsi une infinité d'états de la forme $(1;b^n)$ avec $n \in \mathbb{N}$. Par conséquent, si on veut appréhender tous les éléments de

$Acc(A)$, l'ensemble d'accessibilité de A, il faut *utiliser des techniques qui en un nombre fini d'étapes permettent de générer un nombre infini d'états.*

Ceci va consister dans le cas de A à considérer, en une unique étape, l'ensemble des exécutions possibles — et donc tous les états atteint par elles — partant de et aboutissant à l'état de contrôle 1. Ce qui revient à considérer l'ensemble des séquences d'actions appartenant au langage régulier $(1!a.1!d+1!b)^*$. Ceci fait, il nous restera à effectuer la transition permettant de passer de l'état de contrôle 1 à l'état de contrôle 2.

Nous pouvons donc écrire :

$$(1;\lambda) \overset{(1,(1!a.1!d+1!b)^*,1)}{\longrightarrow} (1;(ac+b)^*) \overset{(1,1!a,2)}{\longrightarrow} (2;(ac+b)^*a)$$

Et nous vérifions aisément que :

$$Acc(A) = \{(1;x)|x \in (ac+b)^*\} \cup \{(2;x)|x \in (ac+b)^*a\}$$

Ce que nous notons :

$$Acc(A) = (1;(ac+b)^*) \cup (2;(ac+b)^*a)$$

Comme nous sommes passés d'un état de contrôle — ici le 1 — à un autre état de contrôle — toujours le 1 — en effectuant, non pas une action ($1!a$, $1!d$ ou $1!b$), mais un ensemble infini d'actions de communications $(1!a.1!d+1!b)^*$, nous dirons que nous avons effectué une *transition symbolique*. De même, le résultat d'une transition symbolique n'est plus un simple couple, (*état de contrôle* ; *mot représentant le contenu du canal*), ce qui donnait par exemple $(1;\lambda)$ ou $(1;bb)$. Ce résultat doit désormais être représenté par un couple dont le premier élément demeure un état de contrôle mais dont le second élément est, lui, un *ensemble* de mots, par exemple : $(1;(ac+b)^*)$. Nous qualifierons d'*état symbolique* ce nouveau type de couple.

Deux problèmes se posent alors :

– le choix des transitions symboliques à utiliser et

– le choix de la méthode de représentation des états symboliques ainsi générés.

Ces deux problèmes sont liés. En effet, la méthode de représentation des états symboliques doit être stable par l'application des transitions symboliques que nous avons choisi d'utiliser. Par exemple, si nous représentons les contenus des canaux des état symboliques par des ensembles reconnaissables, il faut nous restreindre à n'utiliser que des transitions symboliques préservant cette reconnaissabilité.

Supposons que nous ayons réussi à identifier un ensemble intéressant de transitions symboliques. Leur utilisation va consister à les appliquer à partir de l'état initial du système. Et ceci jusqu'à ce que l'une des conditions suivantes soit vérifiée.

1. Les états symboliques générés représentent exactement l'ensemble d'accessibilité. Ceci est le cas positif.

2. La totalité de la mémoire impartie à l'utilisation de cet algorithme a été utilisée. Ceci est le cas négatif.

Même dans le deuxième cas, l'utilisation des transitions symboliques aura probablement permis de générer, et donc de vérifier, beaucoup plus d'états que la méthode classique d'énumération.

5.1 Transitions et états symboliques

L'idée sous-jacente aux transitions symboliques — générer une infinité d'états en un nombre fini d'étapes — est classique. Elle est en effet déjà présente dans l'algorithme de Karp et Miller [KM69], décidant du problème de la finitude de l'ensemble des marquages accessibles d'un réseau de Petri.

L'idée essentielle de cet algorithme est en effet de détecter deux marquages accessibles M_1 et M_2 tels que $M_1 \xrightarrow{\sigma} M_2$ et $M_1 \leq M_2$. Car dans ce cas, il est possible de représenter l'ensemble des marquages accessibles à partir de M_1 en effectuant un nombre quelconque de fois la séquence de transition σ. Cet ensemble représentable est infini dès que $M_1 < M_2$.

L'analyse des automates communiquant par des canaux non fiables est un autre exemple d'utilisation des transitions symboliques. Prenons le cas de l'insertion intempestive de messages dans les canaux. Nous avons vu qu'après chaque étape du calcul de l'ensemble d'accessibilité (cf. Théorèmes 3.6 et 3.15), nous prenions la clôture supérieure des états déjà calculés. Ce qui revient bien à générer une infinité d'états en une seule étape.

Toutefois, dans les deux exemples précédents, l'utilisation de transitions symboliques ad hoc permettait à coup sûr d'atteindre l'objectif poursuivit : générer une infinité de marquages accessibles dans le cas de l'algorithme de Karp et Miller ; calculer l'ensemble d'accessibilité dans le cas des automates communicants avec insertion de messages. Une autre approche est de caractériser des transitions symboliques pouvant s'appliquer à n'importe quel système d'automates communicants. Le but poursuivi est alors de générer et de représenter un maximum d'états du système, en espérant que ce maximum sera « souvent » l'ensemble total d'accessibilité.

C'est dans le cadre de cette démarche que Boigelot et Godefroid ont présenté leurs résultats [BG96]. Nous allons en résumer l'essentiel et pour cela, nous introduisons une notation sur laquelle nous reviendrons plus en détail à la section suivante.

Soit $W \subseteq (\Sigma^*)^p$ un ensemble de contenus de p canaux, et R un langage d'action de communication (c'est-à-dire un langage sur l'alphabet $\{1, \dots, p\} \times \{!, ?\} \times \Sigma$). Nous notons $R(W)$ le sous-ensemble de $(\Sigma^*)^p$ obtenu par application à W des séquences d'actions de communication [1] de R, c'est-à-dire :

$$R(W) = \{\vec{w'} \mid \exists \mu \in R,\ \vec{w} \xrightarrow{\mu} \vec{w'}\}$$

Par exemple, en considérant que nous sommes dans $(\Sigma^*)^2$, que $R = \{1?a.2!d\}$ et que $W = (b^*, c)$, nous avons :

$$1?a.2!d\ ((ab^*, c)) = (b^*, cd)$$

Boigelot et Godefroid ont caractérisé dans [BG96] trois types de séquences de communications pour lesquelles, si W est un reconnaissable de $(\Sigma^*)^p$ et μ l'une de ces séquences, alors $\mu^*(W)$ est reconnaissable et l'on peut en calculer une représentation.

En utilisant une combinaison de transitions classiques et de transitions symboliques, dont les séquences de communications sont de la forme μ^*, Boigelot et Godefroid ont pu calculer la totalité de l'ensemble d'accessibilité d'une vingtaine de protocoles de communication. Ce qui montre déjà l'intérêt pratique des transitions symboliques.

Ces résultats ont récemment été généralisés, d'une part, par Boigelot, Godefroid, Willems et Wolper [BGWW97], d'autre part, par Bouajjani et Haber-

1. Nous appelons séquence d'actions communication (ou plus simplement séquence de communication) un mot sur un alphabet de communication. Par exemple, $1?d.2!d$ est une séquence de communication.

mehl [BH97]. Ces deux généralisations concernent, tout comme dans [BG96], l'application de langages de communication de la forme μ^*.

Dans [BH97], aucune restriction n'est posée sur la nature de μ et, ce, quel que soit le nombre de canaux considérés. Par contre, la méthode choisie de représentation des états symboliques, les CQDD (pour Constraint Queue Decision Diagrams), n'est pas compatible avec les reconnaissables. Il existe ainsi des reconnaissables qui ne sont pas représentables par des CQDD, et inversement. Par exemple, l'ensemble d'accessibilité de l'automate à files de notre exemple introductif, Figure 5.1, ne peut pas être représenté par un CQDD. Et par conséquent, le semi-algorithme de [BH97] ne peut pas le calculer.

Dans [BGWW97], aucune restriction n'est portée sur μ dans le cas où l'automate à files considéré utilise une unique file. Si plusieurs files sont utilisées, des restrictions sont portées sur μ ; cela pour préserver la reconnaissabilité des états symboliques générés.

Ces deux articles [BH97] et [BGWW97] utilisent leurs transitions symboliques via un algorithme similaire à celui de [BG96]. Cet algorithme, comme nous l'avons vu plus haut, consiste en des applications successives de transitions classiques et de transitions symboliques où les langages d'actions sont de la forme μ^*.

Par conséquent, compte tenu du fait que, dans [BH97], aucune restriction n'est portée sur les séquences d'actions μ à itérer, le semi-algorithme issu de [BH97] a un champ d'application nettement plus grand que celui issu de [BGWW97].

Dans la suite de ce chapitre, nous commençons par définir formellement notre perception des transitions symboliques. En effet, bien que la notion soit utilisée dans les articles précédemment cités, elle n'a pas été clairement précisée. Nous verrons qu'un premier type de transitions symboliques est décrit par les *langages d'actions de communication*, c'est-à-dire des ensembles de séquences d'émissions et de réceptions à appliquer aux états symboliques. Puis nous caractériserons les propriétés que doivent satisfaire les *méthodes de représentation* des états symboliques, et ce, indépendamment des transitions symboliques utilisées.

Nous proposerons alors une nouvelle classe de transitions symboliques, les *communications à émissions non consommées*. Cette classe est intéressante à plus d'un titre. Elle englobe les transitions symboliques étudiées dans [BG96] et permet, nous le verrons, d'améliorer les résultats précédents de décidabilité portant sur les systèmes d'automates disposés en anneau [RG84, PP92]. Nous proposons enfin une classe de transitions symboliques améliorant celles de [BGWW97].

5.1.1 Transitions symboliques

Revenons à l'exemple introductif de la Figure 5.1. Nous sommes passés directement de l'état initial $(1; \lambda)$ à l'état symbolique $(1; (ac+b)^*)$. Nous pouvons donc considérer qu'une transition symbolique :

- spécifie un état de contrôle de départ et un autre état de contrôle d'arrivée,
- spécifie la transformation à effectuer sur l'état symbolique de départ pour obtenir l'état symbolique d'arrivée.

De plus, nous voulons qu'une transition symbolique corresponde à une « accélération » du calcul de l'ensemble d'accessibilité, mais qu'elle ne génère pas plus d'états que nécessaire (i.e. des états inaccessibles).

Nous formalisons cela par la définition suivante :

Définition 5.1 Soit $A = (Q, q_o, \Sigma, \Pi, \delta)$ un automate à p files, nous appelons *transition symbolique* un triplet (q, T, q') tel que :

1. $q, q' \in Q$
2. $\forall W \subseteq (\Sigma^*)^p \; \forall \vec{w} \in T(W) \; \exists \vec{w'} \in W, \; (q; w) \xrightarrow{*}_A (q'; w')$

   où T est une application de $2^{(\Sigma^*)^p}$ dans $2^{(\Sigma^*)^p}$ appelée *fonction de communication* de la transition symbolique (q, T, q').

La $2^{\text{ème}}$ condition a pour rôle de garantir que tout état accessible par une suite de transitions symboliques est aussi accessible par une exécution normale de l'automate à files.

Suite à notre exemple introductif, nous pouvons considérer qu'un premier type de fonctions de communication est défini à partir de langages d'actions de communication.

Définition 5.2 Soient Σ un alphabet de canal, Π un alphabet d'actions internes et p un entier naturel indiquant le nombre de canaux.

- Soient $c \in \{1, \dots, p\}$, et $\pi \in \Pi$. Nous notons $c!a$, $c?a$ et π les applications de $2^{(\Sigma^*)^p}$ dans $2^{(\Sigma^*)^p}$ telles que, pour tout $W \subseteq (\Sigma^*)^p$,

$$c!a(W) = \left\{ \vec{w'} \in (\Sigma^*)^p \mid \exists \vec{w} \in W, \; \begin{array}{ll} w'_i = w_i a & \text{si } i = c \\ w'_i = w_i & \text{sinon} \end{array} \right\},$$

$$c?a(W) = \left\{ \vec{w}' \in (\Sigma^*)^p \mid \exists \vec{w} \in W, \begin{array}{rl} aw'_i = w_i & \text{si } i = c \\ w'_i = w_i & \text{sinon} \end{array} \right\}$$
$$\pi(W) = W$$

- Soit $\mu = \mu_1 \ldots \mu_n$ un mot tel que $\mu_i \in \{1, \ldots, p\} \times \{!, ?\} \times \Sigma$ ou $\mu_i \in \Pi$. Nous notons aussi μ l'application de $2^{(\Sigma^*)^p}$ dans $2^{(\Sigma^*)^p}$ telle que

  $$\mu = \mu_n \circ \cdots \circ \mu_1$$

  C'est-à-dire :

  $$\forall W \subseteq (\Sigma^*)^p, \; \mu(W) = \mu_n(\mu_{n-1}(\cdots \mu_1(W) \cdots))$$

  μ est dite *séquence de communication.*

- Soit R un langage sur l'alphabet $\Big(\{1, \ldots, p\} \times \{!, ?\} \times \Sigma\Big) \cup \Pi$. Nous notons aussi R l'application de $2^{(\Sigma^*)^p}$ dans $2^{(\Sigma^*)^p}$ telle que

  $$\forall W \subseteq (\Sigma^*)^p, \; R(W) = \bigcup_{\mu \in R} \mu(W)$$

  R est dit *langage de communication* sur $(\Sigma^*)^p$.

Exemple:

$$\begin{array}{rcl} 1!d.1?a.1!b\,(aa) & = & adb \\ 1!d.1?a.1!b\,(ba) & = & \emptyset \\ & & \\ (1?a.2!b)^*\,((a^*b, \lambda)) & = & \{(a^{m-n}b, b^n) \mid m, n \in \mathbb{N}, \; m - n \geq 0\} \\ & = & (a^*b, b^*) \end{array}$$

Étant donné un automate à files, nous sommes surtout intéressés par les langages de communication qui correspondent à l'ensemble des chemins existant entre deux de ses états locaux et donc à des langages de communication *réguliers*. Remarquons qu'il ne manque que la donnée d'un ensemble d'états finaux à un automate à files pour obtenir un automate fini classique définissant un langage régulier de communication.

Définition 5.3 Soit $A = (Q, q_o, \Sigma, \Pi, \delta, F)$ un sextuplet tel que $(Q, q_o, \Sigma, \Pi, \delta)$ soit un automate à files et $F \subseteq Q$ un ensemble d'états finaux.

- A est dit *automate à files avec états finaux* (ou tout simplement *automate à files* lorsqu'il n'y a pas de confusion).

- $L(A)$ est le langage de communication engendré par A.

Notation: Pour éviter des surcharges d'écriture, nous noterons dans la suite A au lieu de $L(A)$ et $A(q,q')$ au lieu de $L(A(q,q'))$ pour A un automate à files quelconque. Nous identifions donc un automate à files avec états finaux avec le langage qu'il génère.

Automates à files vus en tant que fonctions

Les Définitions 5.2 et 5.3 nous permettent de réécrire la deuxième condition de la Définition 5.1 de la façon suivante :

$$\forall W \subseteq (\Sigma^*)^p,\ T(W) \subseteq A(q,q')(W)$$

$A(q,q')$ est un automate à files où q un état initial et q' un état final ont été précisés afin de le considérer comme un automate fini classique générant un langage de communication. Langage qui est dès lors perçu (via la Définition 5.2) comme définissant une fonction de $2^{(\Sigma^*)^p}$ dans $2^{(\Sigma^*)^p}$.

Ceci nous amène ainsi à passer de l'approche sémantique classique des automates à files comme générateur d'un système de transitions (le passage d'un état global à un autre ; ceci en partant de l'état initial) à leur perception comme des fonctions de $2^{(\Sigma^*)^p}$ dans $2^{(\Sigma^*)^p}$. Soit A l'automate à files de la

A :

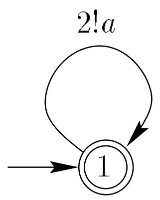


FIG. 5.2 – $L(A) = (2!a)^*$

Figure 5.2 générant le langage de communication $(2!a)^*$. Nous avons pour tout $W \subseteq (\Sigma^*)^3$:

$$A(W) = W.(\lambda, a^*, \lambda)$$

Le théorème qui suit est une conséquence directe de l'indécidabilité du modèle des automates à files.

Théorème 5.1 *Soient W un ensemble reconnaissable de Σ^* et R un langage régulier de communication. $R(W)$ est non récursif.*

Preuve : Soit $A = (Q, q_o, \Sigma, \Pi, \delta)$ un automate à une file. Son ensemble d'accessibilité, $Acc(A)$, vérifie l'égalité suivante :

$$Acc(A) = \bigcup_{q \in Q} \{q\} \times A(q_o, q)\ (\lambda)$$

Or, λ est trivialement un reconnaissable de Σ^* et les langages de communication définis par les automates à files $A(q_o, q)$ sont réguliers. Par conséquent, si pour tous les $q \in Q$, $A(q_o, q)\ (\lambda)$ était récursif, il en serait de même pour leur union $Acc(A)$. Or, nous savons, [BZ83], que $Acc(A)$ est non récursif ❑

Il s'en suit que nous ne pouvons pas «maîtriser», de façon générale, la fonction induite par un automate à files. Mais qu'à cela ne tienne, la caractérisation de transitions symboliques va justement consister à extraire d'un automate à files A une fonction f_A se *rapprochant* le plus de A et *couverte* par A ; c'est-à-dire, telle que $\forall W \subseteq (\Sigma^*)^p,\ f_A(W) \subseteq A(W)$.

5.1.2 États symboliques

Lorsque nous avons motivé le concept des transitions symboliques au début de ce chapitre, nous avons évoqué les liens devant exister entre les *transitions symboliques* et la méthode utilisée pour représenter les *états symboliques*. Nous allons maintenant préciser plus en détails les caractéristiques requises par ces méthodes de représentation. Puis nous nous inspirerons de l'élégant résultat de Pachl, [Pac82, Pac87], concernant les automates à files dont l'ensemble d'accessibilité est reconnaissable. Cela nous permettra de déduire, et ce indépendamment du calcul effectif (ou non) de l'ensemble d'accessibilité, que l'accessibilité est décidable pour tout automate à files dont l'ensemble d'accessibilité est représentable par une méthode satisfaisant nos hypothèses. Enfin, nous présenterons deux méthodes de représentation : les QDD de Boigelot et Godefroid [BG96], et les CQDD de Bouajjani et Habermehl [BH97].

Conditions requises

Un état symbolique est un couple constitué d'un état de contrôle q et d'un sous-ensemble W de $(\Sigma^*)^p$, où Σ est l'alphabet des contenus de canaux et p le nombre de canaux considérés. Or, nous voulons qu'un état symbolique soit une représentation *effective* d'une partie de l'ensemble d'accessibilité. Il faut donc que W soit décrit par une fonction caractéristique *récursive*.

Notation: Soit M une fonction récursive de $(\Sigma^*)^p$ vers $\{0,1\}$. Nous notons $L(M)$ l'ensemble représenté par M, c'est-à-dire :

$$L(M) = \{\vec{w} \in (\Sigma^*)^p \mid M(\vec{w}) = 1\}$$

Comme nous serons amenés à décrire plusieurs sous-ensembles de $(\Sigma^*)^p$, nous utiliserons pour cela une *méthode de représentation*, c'est-à-dire, une classe $\mathcal{C}$ de fonctions caractéristiques récursives. Nous précisons dans ce qui suit les hypothèses minimales à apporter à $\mathcal{C}$ pour que cette classe puisse représenter de façon satisfaisante des états symboliques.

Les transitions symboliques enrichissent la relation de transition d'un automate à files mais ne se substituent pas à elle. Ainsi, et notamment lorsque les conditions d'utilisation d'une transition symbolique ne seront pas réunies, nous devrons pouvoir utiliser les transitions classiques de l'automate à files considéré. Par conséquent, notre première hypothèse sera que $\mathcal{C}$ est stable par l'application des actions de communications classiques. Ce que nous formalisons par la formule suivante :

$$\begin{array}{l}\forall M \in \mathcal{C} \;\; \forall \mu \in \{1, \ldots, p\} \times \{!, ?\} \times \Sigma \;\; \exists M' \in \mathcal{C}, \\ M' \text{ est } \textit{effectivement calculable} \text{ et } \mu(L(M)) = L(M')\end{array}$$

Supposons maintenant, que nous ayons deux transitions symboliques (q, T_1, q') et (q, T_2, q') ayant même état de départ q et même état d'arrivée q'. Étant donné $W \subseteq (\Sigma^*)^p$, au lieu de considérer séparément

$$(q; W) \xrightarrow{T_1} (q'; T_1(W)) \text{ et } (q; W) \xrightarrow{T_2} (q'; T_2(W))$$

il est naturel de vouloir passer directement de $(q; W)$ à $(q'; T_1(W) \cup T_2(W))$. Notre prochaine hypothèse sera donc la stabilité de $\mathcal{C}$ par union. Ce que nous formalisons par :

$$\begin{array}{l}\forall M, M' \in \mathcal{C} \;\; \exists M'' \in \mathcal{C} \\ M'' \text{ est effectivement calculable et } L(M'') = L(M) \cup L(M')\end{array}$$

Notre dernière hypothèse concerne la possibilité de décider si un ensemble d'états symboliques est égal à l'ensemble d'accessibilité. Revenons à notre exemple introductif de la Figure 5.1 page 83. Nous avons abouti à l'exécution *symbolique* suivante :

$$(1;\lambda) \stackrel{(1,(1!a.1!d+1!b)^*,1)}{\longrightarrow} (1;(ad+b)^*) \stackrel{(1,1!a,2)}{\longrightarrow} (2;(ad+b)^*a)$$

Ce faisant, nous obtenons les deux états symboliques $E_1 = (1;(ac+b)^*)$ et $E_2 = (2;(ac+b)^*a)$. Appliquons maintenant l'ensemble des transitions de l'automate A à ces deux états symboliques. Nous vérifions que :

$$E_1 \stackrel{(1,1!b,1)}{\longrightarrow} (1;(ac+b)^*b) \subseteq E_1$$

$$E_1 \stackrel{(1,1!a,2)}{\longrightarrow} (2;(ac+b)^*a) \subseteq E_2$$

$$E_2 \stackrel{(2,1!d,1)}{\longrightarrow} (1;(ac+b)^*ad) \subseteq E_1$$

Par conséquent, quelle que soit la transition tirée, aucun nouvel état n'a été généré par rapport à $E_1 \cup E_2$. Or, comme $E_1 \cup E_2 \subseteq Acc(A)$, il découle du lemme qui suit que :

$$Acc(A) = E_1 \cup E_2$$

Lemme 5.2 *Soient $A = (Q, q_o, \Sigma, \Pi, \delta)$ un automate à p files et $(W_q)_{q\in Q}$ une suite de $|Q|$ sous-ensembles de $(\Sigma^*)^p$. Alors,*

$$\mathrm{Acc}(A) = \bigcup_{q\in Q} (q;W_q)$$

si et seulement si les trois conditions suivantes sont vérifiées :

1. $\vec{\lambda} \in W_{q_o}$
2. $\forall q \in Q,\ (q;W_q) \subseteq \mathrm{Acc}(A)$
3. $\forall q, q' \in Q\ \forall \mu \in (\{1,\ldots,p\} \times \{!,?\} \times \Sigma) \cup \Pi,$
   $(q,\mu,q') \in \delta \Longrightarrow \mu(W_q) \subseteq W_{q'}$

Preuve :

($\Rightarrow$) Ce sens est immédiat.

($\Leftarrow$) La deuxième condition implique : $\bigcup_{q \in Q} (q; W_q) \subseteq Acc(A)$. Pour montrer l'inclusion inverse, il nous suffit de montrer que, quelle que soit l'exécution σ telle que $(q_o, \vec{\lambda}) \xrightarrow{\sigma} (q, \vec{w})$, $\vec{w} \in W_q$. Procédons par récurrence sur la taille de σ.

- $|\sigma| = 0$ implique que $q = q_o$ et $\vec{w} = \vec{\lambda}$. Or, par la première condition, nous avons bien $\vec{\lambda} \in W_{q_o}$.
- $|\sigma| = n + 1$ avec $n \in \mathbb{N}$. Par hypothèse de récurrence, il existe $\sigma' \in \delta^*$, $q' \in Q$, $\vec{w'} \in W_{q'}$, $\mu \in \{1, \ldots, p\} \times \{!, ?\} \times \Sigma \cup \Pi$ tels que $\sigma = \sigma'.(q', \mu, q)$. Ce qui avec la troisième condition implique $\mu(W_{q'}) \subseteq W_q$ et donc $\vec{w} = \mu(\vec{w'}) \in W_q$ ❑

En général, les deux premières conditions sont vérifiées. En effet, l'utilisation des transitions symboliques s'effectue à partir de l'état initial, qui appartient donc d'office aux états symboliques générés par le semi-algorithme de calcul. De plus, notre définition des transitions symboliques (en fait, la 2e condition de la Définition 5.1 page 88) permet justement de préserver la seconde condition de ce lemme. Reste à pouvoir décider de la 3e condition du lemme. Ceci est possible dès que l'inclusion est décidable entre éléments de $\mathcal{C}$. Nous supposerons donc :

$$\forall M, M' \in \mathcal{C},$$
$$L(M) \subseteq L(M') \text{ est décidable}$$

Nous résumons ces diverses hypothèses par la définition qui suit.

Définition 5.4 Une classe $\mathcal{C}$ de fonctions caractéristiques récursives de sous-ensembles de $(\Sigma^*)^p$ est dite de *représentation d'états symboliques* si :
$\forall M, M' \in \mathcal{C}$

- $\forall \mu \in \{1, \ldots, p\} \times \{!, ?\} \times \Sigma, \;\; \exists M'' \in \mathcal{C}$,
  M'' est *effectivement calculable* et $\mu(L(M)) = L(M'')$
- $\exists M'' \in \mathcal{C}$,
  M'' est effectivement calculable et $L(M'') = L(M) \cup L(M')$
- $L(M) \subseteq L(M')$ est décidable.

Limites intrinsèques

Choisir une classe de représentation d'états symboliques délimite les possibilités de représentations de ces états. En effet, supposons que l'ensemble d'accessibilité d'un automate à files ne soit pas reconnaissable et que, par contre, la classe de représentation choisie définisse les reconnaissables. Il est clair, et ce, quelles que soient les transitions symboliques utilisées, qu'il ne sera pas possible de représenter la totalité de l'ensemble d'accessibilité.

De plus, même si l'ensemble d'accessibilité d'un automate à files peut être représenté par la classe choisie, il peut être impossible de mener à bien ce calcul. C'est le cas en particulier des automates à files pouvant perdre des messages. Leur ensemble d'accessibilité est reconnaissable mais il n'est pas possible d'en calculer une représentation sous la forme d'un QDD par exemple (premier item du Théorème 3.4 page 34 et Corollaire 3.10 page 38).

Toutefois, supposons fixé une classe de représentation $\mathcal{C}$ d'états symboliques et A un automates à files. La simple supposition qu'il existe des modèles de $\mathcal{C}$ (sans qu'il soit nécessaire de les caractériser effectivement ; seule leur existence est requise) permettant de représenter l'ensemble d'accessibilité de A, suffit à rendre décidable le problème de l'accessibilité pour A. C'est cette propriété que nous énonçons par le théorème suivant.

Théorème 5.3 *Soient $\mathcal{C}$ une classe de représentation d'états symboliques et $A = (Q, q_o, \Sigma, \Pi, \delta)$ un automate à p files. S'il existe une suite $(M_q)_{q \in Q} \in \mathcal{C}$ de modèles telle que $\mathrm{Acc}(A) = \bigcup_{q\in Q}(q; L(M_q))$ alors, le problème de l'accessibilité est décidable pour A.*

Preuve :

Soit $(q; \vec{w})$ un état global de A, nous mettons en œuvre deux semi-algorithmes pour décider si $(q; \vec{w}) \in Acc(A)$.

- le premier s'arrête si et seulement si $(q; \vec{w}) \in Acc(A)$,
- le second s'arrête si et seulement si $(q; \vec{w}) \notin Acc(A)$.

Par conséquent, l'algorithme réunissant ces deux semi-algorithmes s'arrête toujours, et répond correctement en fonction de celui qui s'est arrêté.

Le premier semi-algorithme consiste simplement à énumérer l'ensemble d'accessibilité jusqu'à trouver $(q, \vec{w})$. Par conséquent, il ne s'arrête que si $(q; \vec{w}) \in Acc(A)$.

Le second semi-algorithme énumère toutes les suites $(M_{q'})_{q' \in Q}$ de modèles de $\mathcal{C}$ jusqu'à en trouver une vérifiant les conditions suivantes :

1. $\vec{\lambda} \in L(M_{q_o})$

2. $\forall q', q'' \in Q\ \forall \mu \in \{1, \ldots, p\} \times \{!, ?\} \times \Sigma \cup \Pi$,
   $(q', \mu, q'') \in \delta \Longrightarrow \mu(L(M_{q'})) \subseteq L(M_{q''})$

3. $\vec{w} \notin L(M_q)$

Il reste à montrer que ce semi-algorithme s'arrête si et seulement si $(q, \vec{w}) \notin Acc(A)$.

– Si $(q, \vec{w}) \notin Acc(A)$, alors, par hypothèse, l'énumération arrivera à la suite de modèles telle que $Acc(A) = \bigcup_{q' \in Q}(q'; L(M_{q'}))$. Grâce au Lemme 5.2 cette suite vérifie les trois conditions. L'énumération s'arrêtera donc.

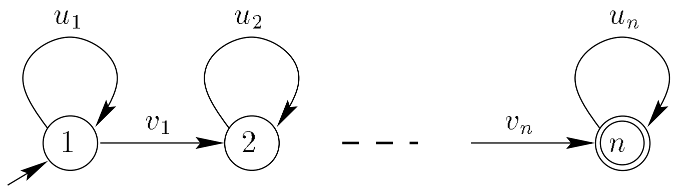
– Inversement, s'il existe une suite de modèles vérifiant ces trois conditions, nous montrons par contradiction que $(q, \vec{w}) \notin Acc(A)$. Cette dernière démonstration est en tous points similaire à celle du sens $(\Leftarrow)$ du Lemme 5.2 ❑

Ce théorème, quoique intéressant du point de vue théorique, ne peut pas être utilisé en pratique car il fait appel à une hypothèse indécidable. Néanmoins il indique une *certaine optimalité* de nos conditions sur les classes de représentations symboliques.

Les QDD et les CQDD

Les QDD [BG96] (QDD pour Queue Decision Diagrams) et les CQDD [BH97] (CQDD pour Constraint Queue Decision Diagrams) sont deux classes de représentation d'états symboliques. Les QDD, (cf. Chapitre 2) permettent de représenter les sous-ensembles reconnaissables de $(\Sigma^*)^p$. Les CQDD ont, eux, été introduits pour représenter certains sous-ensembles non reconnaissables et même non rationnels de $(\Sigma^*)^p$.

Un CQDD est un ensemble fini de couples, constitués chacun d'un tuple de p automates finis déterministes, et d'une contrainte linéaire sur les transitions de ces automates. Malheureusement, les automates finis d'un CQDD ne sont pas quelconques, ils doivent tous avoir la forme suivante :



$u_i \in \Sigma^*$, $v_i \in \Sigma^+$ pour $i = 1, \ldots, n$

La Figure 5.3 donne un exemple de CQDD et précise le sous-ensemble de $(\Sigma^*)^2$ qu'il représente. Pour qu'un couple (w_1, w_2) de $(\Sigma^*)^2$ soit reconnu par C, il faut que w_1 soit reconnu par le premier automate A_1 de C, que w_2 soit reconnu par le second automate A_2 de C et que la formule $x_{(2,d,2)} = 2x_{(4,b,4)}$ soit satisfaite par (w_1, w_2). C'est à dire que le chemin parcouru par w_1 dans A_1 passe 2 fois plus par la transition $(2, d, 2)$ que le chemin parcouru par w_2 dans A_2 passe par la transition $(4, b, 4)$.

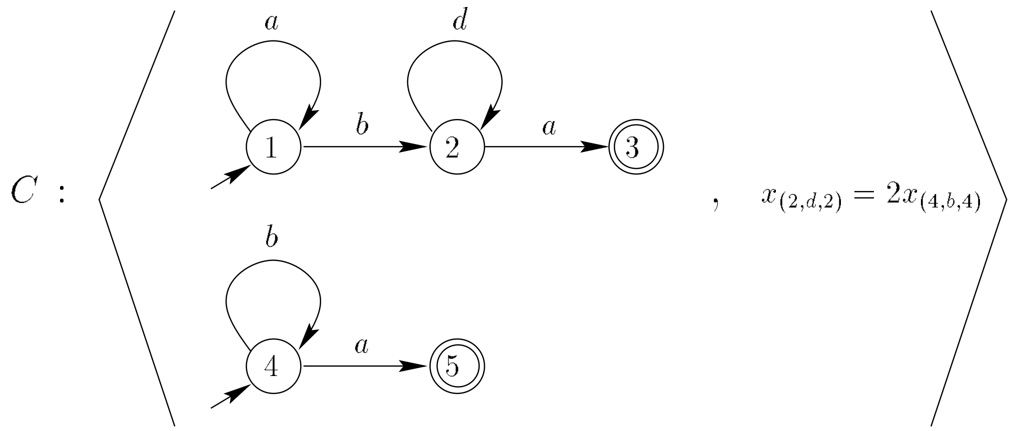


FIG. 5.3 – $L(C) = \{(a^m b d^{2n} a, b^n a) \mid m, n \in \mathbb{N}\}$

Bien que l'utilisation des CQDD rende le semi-algorithme de [BH97] plus général que celui de [BGWW97] (et donc aussi plus général que celui de [BG96]), nous portons notre attention dans les sections suivantes sur les transitions symboliques opérant sur des états symboliques définis à partir d'ensembles reconnaissables. Notre but premier était en effet de caractériser des transitions symboliques généralisant celles de [BG96] et permettant de calculer l'ensemble d'accessibilité des systèmes quasi-stables (cf. Chapitre 4). Or, l'automate à files de notre exemple introductif, Figure 5.1, est quasi-stable et son ensemble, reconnaissable, d'accessibilité ne peut être représenté par un CQDD.

De plus, en dépit du fait que les QDD permettent des représentations plus compactes, nous avons choisi la décomposition de Mezei (cf. Chapitre 2), pour représenter les reconnaissables de $(\Sigma^*)^p$. Cette décomposition des reconnaissables en union finie de produits de réguliers de Σ^* s'est en effet avérée d'un maniement plus aisé lors des preuves. Ceci dit, le passage d'un QDD à la décomposition de Mezei, et vice versa, ne pose pas de difficulté.

5.2 Communication à émissions non consommées

Nous avons vu qu'étant donné un langage régulier de communication R et W un reconnaissable de $(\Sigma^*)^p$, $R(W)$ était en général non récursif. Mais nous pouvons contraindre R de telle sorte que l'on puisse calculer $R(W)$ et qu'il soit de plus reconnaissable.

Nous commencerons par rappeler les résultats de [BG96]. Puis, nous définirons à partir d'un langage de communication R, une fonction de communication T_R que nous appellerons *fonction de communication à émissions non consommées de* R. Nous verrons que ces fonctions sont nettement plus générales que celles définies dans [BG96]. Enfin, nous utiliserons ces fonctions dans la section suivante pour démontrer de nouveaux résultats de décidabilité généralisant ceux de [RG84, PP92]. Les fonctions à émissions non consommées montreront, de nouveau, leur pertinence, en nous servant d'outil de base lorsque nous généraliserons les fonctions de communications caractérisées dans [BGWW97].

Théorème 5.4 (Boigelot et Godefroid [BG96]) *Soient W un ensemble reconnaissable de $(\Sigma^*)^p$ et les trois langages de communication suivants $R_1 = (c!a)^*$, $R_2 = (c?a)^*$ et $R_3 = (c!a.c'?b)^*$ avec $c \neq c'$. Alors les ensembles $R_1(W)$, $R_2(W)$, $R_3(W)$ sont effectivement reconnaissables.*

Les résultats concernant les langages R_1 et R_2 ci-dessus ne sont pas surprenants. Rappelons tout d'abord que nous identifions (c, a) à l'élément $(w_1, \ldots, w_p)$ de $(\Sigma^*)^p$ tel que $w_c = a$ et pour tout $c' \neq c$, $w_{c'} = \lambda$. Ce faisant, nous avons $R_1(W) = W.(c, a)^*$ et $R_2(W) = ((c, a)^*)^{-1}W$. Or, $(c, a)^*$ et W sont reconnaissables et les reconnaissables de $(\Sigma^*)^p$ sont clos par concaténation et dérivation gauche. Le résultat concernant la fonction R_3 est, lui, plus intéressant.

Nous pouvons mettre en évidence une caractéristique commune à ces trois langages de communication. En effet, au cours de l'application d'une séquence de communication μ de l'un de ces trois langages à l'un des éléments de W, les messages reçus par μ ne sont pas des messages précédemment émis par μ. Ce sont des messages qui étaient *déjà* présents dans W. Cela est vrai de R_1 car il n'implique pas de réceptions, de R_2 car il n'implique pas d'émissions, et de R_3 car ses réceptions concernent un canal différent de celui où se font les émissions.

Nous verrons que nous pouvons calculer l'effet sur un reconnaissable de tout langage régulier de communication satisfaisant à cette hypothèse de non consommation (i.e. réception) des messages émis par ce même langage.

Définition 5.5 Soit R un langage régulier de communication. Nous appelons *fonction à émissions non consommées de* R la fonction T_R de $2^{(\Sigma^*)^p}$ dans $2^{(\Sigma^*)^p}$ définie ci-dessous :

$$\forall W \subseteq (\Sigma^*)^p, \quad T_R(W) = \{\mu(\vec{w}) \mid \mu \in R, \vec{w} \in W, \vec{\mu_?} \leq \vec{w}\}$$

Rappelons que nous notons $\leq$ la relation *préfixe de* sur $(\Sigma^*)^p$ et que $\vec{\mu_?}$ est la projection sur leurs canaux respectifs des messages reçus par μ. Ainsi, nous avons :

$$\begin{pmatrix} abad \\ ba \end{pmatrix} \leq \begin{pmatrix} abadd \\ bab \end{pmatrix}$$

et si $\mu = 1?b.1!a.2?d.1!a.1!b.2?a$ alors

$$\vec{\mu_?} = \begin{pmatrix} \mu_{1?} \\ \mu_{2?} \end{pmatrix} = \begin{pmatrix} b \\ da \end{pmatrix} \text{ et de même, } \vec{\mu_!} = \begin{pmatrix} \mu_{1!} \\ \mu_{2!} \end{pmatrix} = \begin{pmatrix} aab \\ \lambda \end{pmatrix}$$

Enfin, nous étendons ces dernières notations aux langages de communication. Par conséquent, si R est un langage de communication sur $(\Sigma^*)^p$, nous avons :

$$\vec{R_?} = \{\vec{\mu_?} \mid \mu \in R\} \text{ et } \vec{R_!} = \{\vec{\mu_!} \mid \mu \in R\}$$

et donc $\vec{R_?}$, $\vec{R_!} \subseteq (\Sigma^*)^p$.

Notation: Soit A un automate à files définissant un langage de communication, nous noterons T_A au lieu de $T_{L(A)}$.

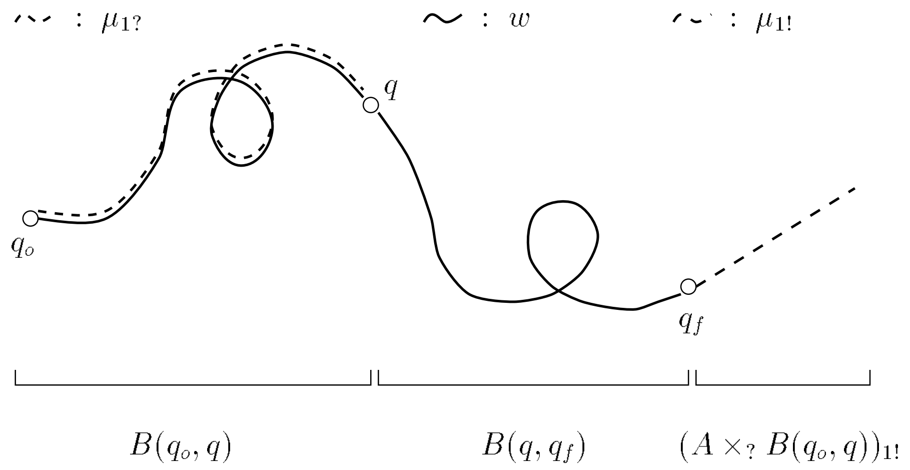


FIG. 5.4 – *fonctionnement de* $T_A(B)$

Au cours des paragraphes qui suivent, nous abordons informellement la construction de $T_A(W)$ pour A un automate à une file et W un reconnaissable de Σ^*. Pour cela, nous supposerons que W est défini par un automate fini déterministe $B = (Q, q_o, \Sigma, \delta, F)$.

Soient $w \in B$ et $\mu \in A$ tels que $\mu_{1?} \leq w$. Les trois propriétés qui suivent sont alors vérifiées :

- il existe un état final q_f de B tel que $q_f = \delta(q_o, w)$,
- il existe un état q de B tel que $q = \delta(q_o, \mu_{1?})$,
- $\mu(w) = (w - \mu_{1?})\mu_{1!}$ avec $(w - \mu_{1?}) \in B(q, q_f)$ car B est déterministe.

  ($B(q, q_f)$ est l'automate fini identique à B sauf que l'état initial est maintenant q et l'unique état final est q_f)

La Figure 5.4 illustre cette construction. Nous pouvons remarquer que, quel que soit le mot w de B dont le chemin correspondant passe par l'état q et aboutit à l'état q_f et que $\mu_{1?} \leq w$, $\mu(w) = (w - \mu_{1?}).\mu_{1!} \in T_A(B)$.

Ce qui nous donne :

$$B(q, q_f).\mu_{1!} \subseteq T_A(B)$$

Nous pouvons franchir une autre étape dans notre construction de $T_A(B)$ en remarquant que, quelle que soit la séquence $\mu \in A$ telle que $\mu_{1?} \in B(q_o, q)$,

$$B(q, q_f).\mu_{1!} \subseteq T_A(B)$$

Par conséquent, on a :

$$B(q,q_f)\,.\,\{\mu_{1!}\in\Sigma^* \mid \mu\in A \text{ et } \mu_{1?}\in B(q_o,q)\}\subseteq T_A(B)$$

Ce que nous pouvons écrire :

$$B(q,q_f)\,.\,\Big(A\times_? B(q_o,q)\Big)_{1!}\subseteq T_A(B)$$

avec $A\times_? B(q_o,q)=\{\mu\in A \mid \mu_{1?}\in B(q_o,q)\}$.

Il suffit alors de faire varier q et q_f. Nous montrerons que nous obtenons :

$$T_A(B)=\bigcup_{\substack{(q,q_f)\\ \in\\ Q_B\times F_B}} B(q,q_f)\,.\,\Big(A\times_? B(q_o,q)\Big)_{1!}$$

Revenons maintenant au cas général. Tout d'abord, nous aurons besoin de calculer $A\times_? B(q_o,q)$, c'est-à-dire, l'ensemble des séquences de communication de A dont la séquence des messages reçus appartient à $B(q_o,q)$.

Définition 5.6 Soient $B=(B_1,\ldots,B_p)$, avec $B_i=(Q_i,q_{o_i},\Sigma,\delta_i,F_i)$, un tuple de p automates finis définissant un reconnaissable de $(\Sigma^*)^p$ et $A=(Q_A,q_{o_A},\Sigma,\Pi,\delta_A,F_A)$ un automate à p files définissant un langage de communication sur $(\Sigma^*)^p$.

Nous notons $A\times_? B=(Q,q_o,\Sigma,\Pi,\delta_?,F)$ et $A\times_! B=(Q,q_o,\Sigma,\Pi,\delta_!,F)$ les automates à p files avec états finaux définis de la façon suivante :

- $Q=Q_A\times Q_B$
- $F=F_A\times F_B$
- $q_o=(q_{o_A},\vec{q}_{o_B})$
- $\delta_?$ est le plus petit ensemble de transitions tel que :

  1. $(q_A,c?a,q'_A)\in\delta_A,\ (q_c,a,q'_c)\in\delta_c,\ a\in\Sigma$
     $\Longrightarrow\ \forall i\in\{1,\ldots,p\},\ i\neq c,\ \forall q_i\in Q_i,$
     $\Big((q_A,q_1,\ldots,q_c,\ldots,q_p),c?a,(q'_A,q_1,\ldots,q'_c,\ldots,q_p)\Big)\in\delta_?$

  2. $(q_A,c!a,q'_A)\in\delta_A,\ a\in\Sigma$
     $\Longrightarrow\ \forall i\in\{1,\ldots,p\}\ \forall q_i\in Q_i,$
     $\Big((q_A,q_1,\ldots,q_p),c!a,(q'_A,q_1,\ldots,q_p)\Big)\in\delta_?$

3. $(q_A, \pi, q'_A) \in \delta_A,\ \pi \in \Pi$
$$\Longrightarrow \forall i \in \{1, \dots, p\}\ \forall q_i \in Q_i,$$
$$\Big((q_A, q_1, \dots, q_p), \pi, (q'_A, q_1, \dots, q_p)\Big) \in \delta_?$$

- $\delta_!$ est le plus petit ensemble de transitions tel que :

1. $(q_A, c!a, q'_A) \in \delta_A,\ (q_c, a, q'_c) \in \delta_c,\ a \in \Sigma$
$$\Longrightarrow \forall i \in \{1, \dots, p\},\ i \neq c,\ \forall q_i \in Q_i,$$
$$\Big((q_A, q_1, \dots, q_c, \dots, q_p), c!a, (q'_A, q_1, \dots, q'_c, \dots, q_p)\Big) \in \delta_!$$

2. $(q_A, c?a, q'_A) \in \delta_A,\ a \in \Sigma$
$$\Longrightarrow \forall i \in \{1, \dots, p\}\ \forall q_i \in Q_i,$$
$$\Big((q_A, q_1, \dots, q_p), c?a, (q'_A, q_1, \dots, q_p)\Big) \in \delta_!$$

3. $(q_A, \pi, q'_A) \in \delta_A,\ \pi \in \Pi$
$$\Longrightarrow \forall i \in \{1, \dots, p\}\ \forall q_i \in Q_i,$$
$$\Big((q_A, q_1, \dots, q_p), \pi, (q'_A, q_1, \dots, q_p)\Big) \in \delta_!$$

Avec $Q_B = Q_1 \times \ldots \times Q_p$ et $F_B = F_1 \times \ldots \times F_p$.

De cette construction, nous obtenons immédiatement les résultats suivants.

Lemme 5.5 *Soient A un automate à p files et $B = (B_1, \dots, B_p)$ un tuple de p automates finis définissant un ensemble reconnaissable de $(\Sigma^*)^p$. Alors,*

$$\begin{array}{rcl} A \times_? B & = & \{\mu \in A \,|\, \vec{\mu}_? \in B\} \\ A \times_! B & = & \{\mu \in A \,|\, \vec{\mu}_! \in B\} \end{array}$$

Preuve :

$A \times_? B$ résulte d'une synchronisation du même type que celle utilisée pour montrer que l'intersection de deux langages reconnaissables est reconnaissable.

Prouver ce lemme dans le cas de $A \times_? B$ revient donc à prouver la proposition suivante :

$$\forall \vec{q}_B \in Q_B\ \forall q_A \in Q_A\ \forall \mu \in A \left(\begin{array}{c} q_{o_A} \xrightarrow{\mu}_{\delta_A} q_A \text{ et } \vec{q}_{o_B} \xrightarrow{\vec{\mu}_?}_{\delta_B} \vec{q}_B \\ \Longleftrightarrow \\ (q_{o_A}, q_{o_B}) \xrightarrow{\mu}_{\delta_?} (q_A, \vec{q}_B) \end{array} \right)$$

Ce qui se fait par une simple récurrence sur la longueur de μ. Le cas de $A \times_! B$ se traite de façon similaire ❑

Remarque 5.1 Par la décomposition de Mezei, un sous-ensemble reconnaissable de $(\Sigma^*)^p$ est défini comme une union finie de tuples de la forme $B_k = (B_{k1}, \ldots, B_{kp})$ où les $(B_{ki})_{\substack{k=1,\ldots,n \\ i=1,\ldots,p}}$ sont des automates finis. Par conséquent, pour tout automate à files A définissant un langage de communication, nous pouvons noter $A \times_? W$ au lieu de $\bigcup_{k=1,\ldots,n} A \times_? B_k$.

Théorème 5.6 *Soient A un automate à p files et $B = (B_1, \ldots, B_p)$, avec $B_i = (Q_i, q_{o_i}, \Sigma, \delta_i, F_i)$, un tuple de p automates finis définissant un ensemble reconnaissable de $(\Sigma^*)^p$. Alors,*

$T_A(B)$ est un ensemble rationnel de $(\Sigma^*)^p$ et a la représentation suivante :

$$T_A(B) = \bigcup_{\substack{(\vec{q},\vec{q}_f) \\ \in \\ Q_B \times F_B}} B(\vec{q}, \vec{q}_f) \,.\, \overrightarrow{\Big(A \times_? B(\vec{q}_o, \vec{q})\Big)_!}$$

Avec $Q_B = Q_1 \times \ldots \times Q_p$ et $F_B = F_1 \times \ldots \times F_p$.

Preuve : Les ensembles $B(\vec{q}, \vec{q}_f)$ sont reconnaissables, les ensembles

$$\overrightarrow{\Big(A \times_? B(\vec{q}_o, \vec{q})\Big)_!}$$

sont, eux, rationnels. Or, dans $(\Sigma^*)^p$, la concaténation d'un ensemble reconnaissable et d'un ensemble rationnel est rationnelle [Ber79]. Par conséquent, le membre droit de l'égalité est rationnel comme union finie de rationnels.

Montrons maintenant que l'égalité est bien vérifiée. Nous avons :

$$T_A(B) = \{\mu(\vec{w}) \,|\, \mu \in A,\ \vec{w} \in B,\ \vec{\mu}_? \leq \vec{w}\}$$

Or, pour tout $\mu \in A$ et tout $\vec{w} \in B$ tels que $\mu(\vec{w})$ est défini, nous avons l'égalité $\vec{w}\vec{\mu}_! = \vec{\mu}_?\mu(\vec{w})$. Comme $\vec{\mu}_? \leq \vec{w}$, nous obtenons $\mu(\vec{w}) = (\vec{w} - \vec{\mu}_?)\vec{\mu}_!$. Par conséquent, il existe deux éléments $\vec{w'}$ et $\vec{w''}$ de $(\Sigma^*)^p$ tels que $\vec{w'}\vec{w''} = \vec{w}$,

$\vec{w'} = \vec{\mu}_?$ et donc $\mu(\vec{w}) = \vec{w''}\vec{\mu}_!$. Ce qui nous permet d'écrire :

$$
\begin{aligned}
T_A(B) &= \{\vec{w''}\vec{\mu}_! \,|\, \mu \in A,\ \exists \vec{w'},\ \vec{w'}\vec{w''} \in B,\ \vec{\mu}_? = \vec{w'}\} \\
&= \bigcup_{(\vec{q},\vec{q}_f)\in Q_B\times F_B} \{\vec{w''}\vec{\mu}_! \,|\, \mu \in A,\ \exists \vec{w'},\ \vec{w'} \in B(\vec{q}_o,\vec{q}), \vec{w''} \in B(\vec{q},\vec{q}_f),\ \vec{\mu}_? = \vec{w'}\} \\
&= \bigcup_{(\vec{q},\vec{q}_f)\in Q_B\times F_B} \{\vec{w''}\vec{\mu}_! \,|\, \mu \in A,\ \vec{\mu}_? \in B(\vec{q}_o,\vec{q}), \vec{w''} \in B(\vec{q},\vec{q}_f)\} \\
&= \bigcup_{(\vec{q},\vec{q}_f)\in Q_B\times F_B} B(\vec{q},\vec{q}_f)\,.\{\vec{\mu}_! \,|\, \mu \in A,\ \vec{\mu}_? \in B(\vec{q}_o,\vec{q})\} \\
&= \bigcup_{(\vec{q},\vec{q}_f)\in Q_B\times F_B} B(\vec{q},\vec{q}_f)\,.\overrightarrow{\Big(A \times_? B(\vec{q}_o,\vec{q})\Big)_!}
\end{aligned}
$$

❑

Corollaire 5.7 *Soient A un automate à files définissant un langage de communication sur $(\Sigma^*)^p$ et W un reconnaissable de $(\Sigma^*)^p$. Alors, $T_A(W)$ est effectivement rationnel.*

Preuve : Par la décomposition de Mezei, il existe une union finie de tuples de la forme $B_k = (B_{k1}, \dots, B_{kp})$ où les $(B_{ki})_{\substack{k=1,\dots,n\\ i=1,\dots,p}}$ sont des automates finis tels que $W = \bigcup_{k=1,\dots,n} B_k$. Par conséquent, nous avons :

$$T_A(W) = \bigcup_{k=1,\dots,n} T_A(B_k)$$

Or, via le théorème précédent, chaque $T_A(B_k)$ peut être calculé ❑

Corollaire 5.8 *Soit A un automate à files définissant un langage de communication sur $(\Sigma^*)^p$ tel que :*

$$\exists\, c \in \{1, \dots, p\}\ \forall c' \neq c,\ A_{c'!} = \lambda$$

Alors, quel que soit W reconnaissable de $(\Sigma^*)^p$, $T_A(W)$ est effectivement reconnaissable.

Preuve : Cette hypothèse implique que l'expression $\overrightarrow{\Big(A \times_? B(\vec{q}_o,\vec{q})\Big)_!}$ du Théorème 5.6 représente un rationnel de $(\Sigma^*)^p$ dont une seule coordonnée (celle de rang c) est différente du mot vide. Par conséquent, c'est un reconnaissable de $(\Sigma^*)^p$. Or, comme les $B(\vec{q}, \vec{q'})$ sont des reconnaissables, il s'en suit alors que $T_A(W)$ est reconnaissable comme union finie de produits de reconnaissables de $(\Sigma^*)^p$ ❏

En fait, comme nous ne voulons pas sortir du cadre des reconnaissables, nous ne pourrons véritablement utiliser les fonctions à émissions non consommées que sous la condition du corollaire précédent. Du point de vue pratique, si cette condition n'est pas vérifiée, nous pourrons la provoquer en choisissant un canal c et en ne considérant que les séquences de A n'émettant pas dans les canaux autres que c. Cela revient alors à considérer $A \times_! (c, \Sigma^*)$ au lieu de A avec $(c, \Sigma^*) = \{\lambda\} \times \cdots \times \{\lambda\} \times \Sigma^* \times \{\lambda\} \cdots \times \{\lambda\}$; Σ^* apparaissant au rang c.

Cette utilisation «pragmatique» des fonctions à émissions non consommées englobe néanmoins celles qui sont définies dans [BG96]. En effet, pour R égal à $(c!a)^*$ ou à $(c?a)^*$ ou à $(c!a.c'?b)^*$, nous avons constaté au début de cette section que $T_R(W) = R(W)$ car les émissions ne sont pas consommées. De plus, R vérifie la conditions du corollaire précédent : il y a des émissions dans au plus un canal.

Le théorème suivant limite l'application des fonctions de communication à émissions non consommées aux ensembles reconnaissables.

Théorème 5.9 *Soient W un ensemble reconnaissable de $(\Sigma^*)^p$, R et R' deux langages réguliers de communication. Alors*

$$T_{R'}(T_R(W)) \textit{ est un ensemble non récursif}$$

Preuve : Nous procéderons par une réduction au problème de correspondance de Post. Ce problème est bien connu pour être indécidable [Pos46, HU79]. Une instance de ce problème est la donnée d'une suite finie $(x_i, y_i)_{i=1,\dots,n}$ avec $x_i, y_i \in \Sigma^*$. Le problème est le suivant : existe-t-il une suite finie $(j_k)_{k=1,\dots,m}$ avec $j_k \in \{1, \dots, n\}$ telle que :

$$x_{i_1} x_{i_2} \dots x_{i_m} = y_{i_1} y_{i_2} \dots y_{i_m}$$

Considérons l'exemple du système $S = (A_1, A_2)$ d'automates communicants de la Figure 5.5.

Nous avons $Acc(S, (2,3)) = T_{A_2(3,3)}\Big(T_{A_1(1,2)}(\vec{\lambda})\Big)$. Or, décider que $\vec{\lambda} \in Acc(S, (2,3))$ reviendrait à décider que le problème de correspondance de Post sous-jacent a une solution. Ce qui n'est pas possible ❏

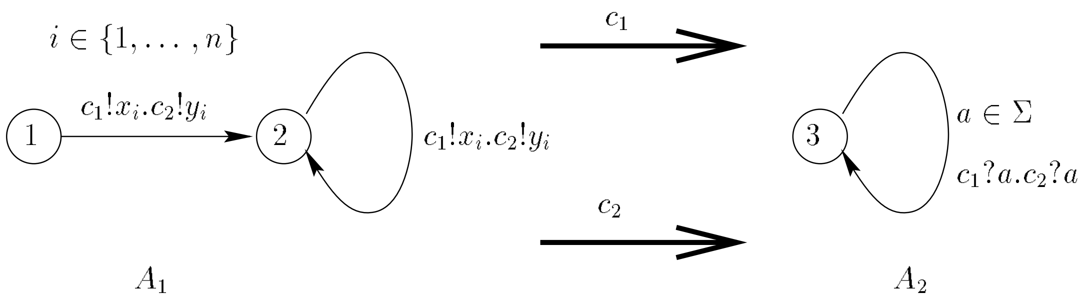


FIG. 5.5 – *Simulation du problème de correspondance de Post*

Nous avons donc tout intérêt à préserver la reconnaissabilité du résultat de l'application d'une fonction de communication à émissions non consommées à un reconnaissable.

5.3 Application des fonctions de communication à émissions non consommées

Les transitions symboliques sont principalement appelées à être utilisées par des semi-algorithmes d'approximation de l'ensemble d'accessibilité d'un automate à files. Mais nous montrons au cours de cette section que leur utilisation nous permet d'aboutir à un résultat intéressant de décidabilité ; et ce faisant, d'améliorer les résultats de [RG84, PP92].

Face à l'indécidabilité du modèle des automates communicants, les premières contraintes qui ont été proposées pour permettre leur vérification furent :

- l'utilisation d'un seul type de message par canal,
  (ce qui ramène aux réseaux de Petri)
- l'utilisation de canaux de tailles bornées
  (ce qui ramène à un système fini)

L'une des premières hypothèses intéressantes ne se ramenant pas, peu ou prou, aux réseaux de Petri ou aux systèmes finis a été énoncée par Rosier et Gouda [RG84]. Le théorème qui suit en rappelle la teneur.

Définition 5.7 Un système $S = (A_1, \dots, A_n)$ de n automates communicants est dit *en anneau* s'il implique n canaux et si chaque automate A_i ne gère que deux canaux : le canal i en réception et le canal $i+1$ (1 si $i = n$) en émission.

La figure 5.6 illustre la disposition relative des automates et des canaux pour un système en anneau de 5 automates communicants.

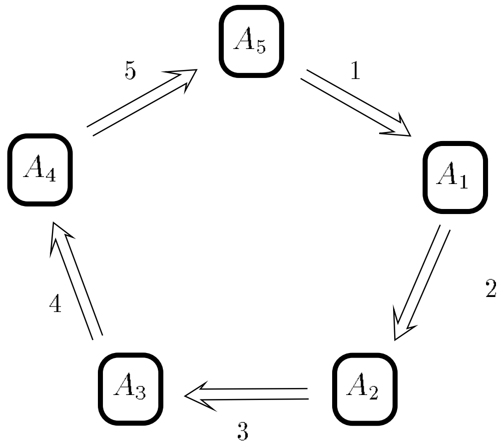


FIG. 5.6 – *Système en anneau de 5 automates*

Théorème 5.10 ([RG84]) *Soit $S = (A_1, A_2)$ un système en anneau de deux automates communicants. Si l'un des canaux est borné a posteriori ou ne transmet qu'un type de message, alors les problèmes suivants sont décidables pour S :*

- *la finitude de l'ensemble d'accessibilité,*
- *la présence de réceptions non spécifiées,*
- *la présence d'états de blocage.*

Rappelons qu'un état accessible est dit de réception non spécifiée si l'un des automates du système est bloqué par la présence dans chacun de ses canaux de réception d'un message qu'il ne peut recevoir. De même, un état accessible est dit de blocage si tous les canaux sont vides alors que tous les automates sont en attente d'un message. Précisons aussi que nous disons qu'un canal est borné à posteriori s'il s'avère que la taille des contenus de ce canal dans les états accessibles du système ne dépasse jamais une certaine borne. Mais cela ne signifie en aucun cas que nous fixons a priori une borne et que nous interdisons les émissions qui en provoqueraient le dépassement.

Un résultat similaire dans ses conclusions a été proposé par Peng et Purushothaman [PP92]. Il concerne les réseaux en anneau comportant plus de deux automates mais dont les canaux sont réduits à ne transmettre qu'un type de message.

Théorème 5.11 ([PP92]) *Soit $S = (A_1, \ldots, A_n)$ un système en anneau de n automates communicants dont l'alphabet des messages de canaux ne comporte qu'une lettre. Alors, l'ensemble d'accessibilité est effectivement semi-linéaire. Par conséquent, les problèmes suivants sont décidables :*

- *l'accessibilité,*
- *la finitude de l'ensemble des états accessibles.*

L'information essentielle de ce théorème est la semi-linéarité de l'ensemble d'accessibilité. En effet, le fait de réduire l'alphabet des canaux à un unique type de message réduit le système à un réseau de Petri. Et par conséquent, les deux problèmes cités sont décidables.

Nous nous proposons de montrer que l'ensemble d'accessibilité d'un système en anneau dont l'*un* des canaux est borné a posteriori ou réduit à ne contenir qu'un seul type de message est effectivement reconnaissable. Cela améliore le résultat de [RG84] car nous donnons une description effective de l'ensemble d'accessibilité (ce qui prouve du même coup le problème de l'accessibilité !) et, de plus, nous prenons en compte des systèmes anneaux impliquant un nombre quelconque d'automates. Cela améliore aussi le résultat de [PP92] car nous demandons à *un* seul canal d'être un compteur. De plus, notre résultat implique que, dans le cas ou tous les canaux sont réduits à des compteurs (hypothèse de [PP92]), l'ensemble d'accessibilité est plus que semi-linéaire, c'est-à-dire rationnel dans $\mathbb{N}^n$, il est un reconnaissable de $\mathbb{N}^n$.

Commençons par rappeler un théorème dont nous aurons besoin. Il est à noter que la propriété qu'il énonce est plus ou moins sous-jacente dans les démonstrations de [RG84] et [PP92]. Mais c'est l'utilisation des fonctions de communication à émissions non consommées qui nous permettra d'en tirer le maximum.

Théorème 5.12 ([Pac82]) *Soit $S = (A_1, \ldots, A_n)$ un système en anneau de n automates communicants. Alors, tout état accessible s peut être atteint par une exécution $s_o \xrightarrow{\sigma_o} s_1 \xrightarrow{\sigma_1} s_2 \xrightarrow{\sigma_2} \ldots \xrightarrow{\sigma_{n-1}} s_n = s$ telle que :*

1. *tous les états traversés par l'exécution σ_o sont de la forme $(\vec{q}; \vec{w})$ avec $\Sigma_{i=2}^{n} |w_i| \leq 1$,*
2. *$s_1 \in Q_1 \times \cdots \times Q_n \times \Sigma^* \times \{\lambda\}^{n-1}$, c'est-à-dire que tous les canaux de s_1 sont vides, sauf éventuellement le premier.*

3. *pour* $i \in \{1, \ldots, n-1\}$*, l' exécution* σ_i*, est effectuée par* A_i*.*

Preuve : Nous nous contentons ici de rappeler les grandes lignes de la démonstration. Soit σ une exécution aboutissant à l'état s. L'exécution σ est constituée par un entrelacement de n séquences locales, une pour chaque automate.

L'exécution σ_o est alors obtenue en rejouant ces séquences locales mais en donnant la plus forte priorité d'exécution à A_n, puis à A_{n-1} et ainsi de suite jusqu'à A_1 qui obtient la plus petite priorité pour rejouer sa séquence.

Ce faisant, on assure que l'automate A_i pour $i \in \{1, \ldots, n-1\}$ n'effectue une émission que si tous les automates A_k, $k \in \{i+1, \ldots, n\}$ sont bloqués et donc en attente d'un message dans leurs canaux d'entrées.

Lorsque A_i effectue son émission, au coup suivant l'automate A_{i+1} qui lui est prioritaire, peut alors exécuter sa réception en attente et ainsi consommer le message émis. Par conséquent, à aucun moment il n'y a plus d'un message au total dans l'ensemble des canaux autres que le premier.

La construction de σ_o est terminée lorsque la dernière action de la séquence locale de A_n est exécutée. Notons qu'à ce stade, seul le premier canal est non vide.

La séquence σ_i, pour $i \in \{1, \ldots, n-1\}$, est la fin de la séquence locale de A_i dans σ_o et qui n'a pas été exécutée dans σ_o ❑

Nous pouvons d'ores et déjà remarquer que les séquences σ_i pour $i \geq 1$ correspondent à des communications à émissions non consommées. En effet, au vu de la géométrie en anneau du système, les réceptions de chacune de ces séquences concernent un canal différent de celui où s'effectue les émissions. Par conséquent, si on peut rendre effectivement reconnaissable l'ensemble des états atteints par une exécution du type σ_o, on pourra utiliser des fonctions de communication à émissions non consommées pour propager cette reconnaissabilité et calculer ainsi l'ensemble d'accessibilité.

La définition et le théorème qui suivent formalisent cette intuition. Notons, que nous n'avons formulé aucune hypothèse sur les automates composant le système. Il peuvent donc comporter des boucles imbriquées, ce que ne peuvent gérer les fonctions de communications définies dans [BG96], [BGWW97] et [BH97].

Définition 5.8 Soit $S = (A_1, \ldots, A_n)$ un système en anneau de n automates communicants tels que $A_i = (Q_i, q_{o_i}, \Sigma, \Pi, \delta_i)$.

- Nous notons $H_1(S)$ l'ensemble des états accessibles s tels que :

  1. $s \in Q_1 \times \cdots \times Q_n \times \Sigma^* \times \{\lambda\}^{n-1}$. Autrement dit, seul le premier canal est non vide.
  2. s est atteint par une exécution telle que tous les états traversés au cours de cette exécution sont de la forme $(\vec{q}; \vec{w})$ avec $\Sigma_{i=\mathbf{2}}^{n} |w_i| \leq 1$.

- Soit $\vec{q} \in Q_1 \times \cdots \times Q_n$, nous notons $H_1(S, \vec{q})$ l'ensemble des contenus des canaux des éléments de $H_1(S)$ ayant pour état de contrôle $\vec{q}$:

$$H_1(S, \vec{q}) = \{\vec{w} \in (\Sigma^*)^p \mid (\vec{q}; \vec{w}) \in H_1(S)\}$$

Théorème 5.13 *Soient $S = (A_1, \ldots, A_n)$ un système en anneau de n automates communicants et $\vec{q}$ un état local de S. Nous avons :*

$$\mathrm{Acc}(S, \vec{q}) = \bigcup_{\substack{\vec{q'} \in Q_1 \times \cdots \times Q_n \\ q'_n = q_n}} T_{A_{n-1}(q'_{n-1}, q_{n-1})} \circ T_{A_{n-2}(q'_{n-2}, q_{n-2})} \circ \cdots \circ T_{A_1(q'_1, q_1)} \Big(H_1(S, \vec{q'})\Big)$$

De plus, si $H_1(S)$ est effectivement reconnaissable, $\mathrm{Acc}(S)$ *est effectivement reconnaissable.*

Preuve : $H_1(S)$ représente l'ensemble des états atteignables par une exécution σ_o du Théorème 5.12. De ce même théorème, on déduit que pour avoir l'ensemble des états accessibles, il faut, à partir des états de $H_1(S)$, effectuer une exécution locale à A_1, puis à A_2 et ainsi de suite jusqu'à effectuer une exécution locale à A_{n-1}. Ce faisant nous obtenons :

$$Acc(S, \vec{q}) = \bigcup_{\substack{\vec{q'} \in Q_1 \times \cdots \times Q_n \\ q'_n = q_n}} A_{n-1}(q'_{n-1}, q_{n-1}) \circ \cdots \circ A_1(q'_1, q_1) \Big(H_1(S, \vec{q'})\Big)$$

Or, pour $i \in \{1, \ldots, n-1\}$, $A_i(q'_i, q_i)$ implique des émissions dans un canal (le $i+1^{\text{ième}}$) différent de celui (le $i^{\text{ième}}$) où les réceptions sont effectuées. Il n'y a donc pas consommation des émissions. Par conséquent, $A_i(q'_i, q_i)$ et $T_{A_i(q'_i, q_i)}$ sont équivalentes en tant que fonctions de communication. De plus, les $A_i(q'_i, q_i)$ n'émettant chacun que dans un seul canal, ils satisfont à l'hypothèse du Corollaire 5.8. Par conséquent, leur application à des ensembles effectivement reconnaissables donne des ensembles effectivement reconnaissables.

Le résultat d'une application à émissions non-consommées étant effectivement reconnaissable à partir d'un ensemble effectivement reconnaissable, nous en déduisons que pour tout état local $\vec{q}$, $Acc(S, \vec{q})$ et donc $Acc(S)$ sont effectivement reconnaissables si $H_1(S)$ l'est ❑

Corollaire 5.14 *Soit $S = (A_1, \ldots, A_n)$ un système en anneau de n automates communicants tel que l'un des canaux est borné a posteriori ou réduit à ne transmettre qu'un type de message. Alors, l'ensemble d'accessibilité de S est effectivement reconnaissable.*

Preuve : Considérons tout d'abord le cas où la taille d'un des canaux est bornée par k. Le système étant en anneau, nous pouvons sans perte de généralité supposer qu'il s'agit du premier canal. Dans ce cas, il est clair par sa définition que $H_1(S)$ est borné et que l'on peut l'énumérer. Il est en effet constitué d'états atteints par des exécutions ne passant que par des états où la taille de tous les canaux est bornée (par k pour le premier et par 1 pour les autres). $H_1(S)$ étant un ensemble fini dont on connaît les éléments, il est effectivement reconnaissable. De là, on utilise le Théorème 5.13 pour conclure.

Supposons maintenant que l'un des canaux ne transmet qu'un type de message. Tout comme précédemment, nous pouvons considérer qu'il s'agit du premier canal. Nous allons construire A, un automate à une file équivalent à S en ce qui concerne les exécutions de type σ_o du Théorème 5.12. Soit δ_S la fonction de transition de l'automate à files correspondant au système S. Nous définissons $A = (Q, q_o, \Sigma, \Pi, \delta)$ tel que :

- $Q = Q_1 \times \cdots \times Q_n \times \Big((\{1, \ldots, n\} \times \Sigma) \cup \lambda \Big)$,
- $q_o = (q_{o_1}, \ldots, q_{o_n}, \lambda)$,
- δ est le plus petit ensemble de transitions tel que :
  - $\begin{array}{l} a \in \Sigma,\ c \neq 1 \\ (\vec{q}, c!a, \vec{q'}) \in \delta_S \end{array} \Longrightarrow ((\vec{q}, \lambda), \tau, (\vec{q'}, (c, a))) \in \delta$
  - $\begin{array}{l} a \in \Sigma,\ c \neq 1 \\ (\vec{q}, c?a, \vec{q'}) \in \delta_S \end{array} \Longrightarrow ((\vec{q}, (c, a)), \tau, (\vec{q'}, \lambda)) \in \delta$
  - $\begin{array}{l} a \in \Sigma \\ (\vec{q}, 1?a, \vec{q'}) \in \delta_S \end{array} \Longrightarrow ((\vec{q}, x), 1?a, (\vec{q'}, x)) \in \delta$

– $\begin{array}{l} a \in \Sigma \\ (\vec{q}, 1!a, \vec{q'}) \in \delta_S \end{array} \Longrightarrow ((\vec{q}, x), 1!a, (\vec{q'}, x)) \in \delta$

– $\begin{array}{l} \pi \in \Pi \\ (\vec{q}, \pi, \vec{q'}) \in \delta_S \end{array} \Longrightarrow ((\vec{q}, x), \pi, (\vec{q'}, x)) \in \delta$

Nous avons donc :

$$H_1(S, \vec{q}) = Acc(A, (\vec{q}, \lambda)) \times \{\lambda\}^{n-1}$$

Par hypothèse, l'automate A n'utilise qu'un type de message pour son unique canal, nous pouvons donc considérer que c'est un automate à un compteur. Il s'en suit que son ensemble d'accessibilité est effectivement reconnaissable [HP79]. Une autre façon de voir les choses est de considérer un automate à un compteur comme un automate à une pile. De là on peut utiliser l'astucieux algorithme de [FWW97] pour calculer l'ensemble reconnaissable de ses états accessibles. L'ensemble $Acc(A)$ étant effectivement reconnaissable, il en est de même pour $Acc(A, (\vec{q}, \lambda))$ quel que soit $\vec{q}$ et donc aussi pour $H_1(S, \vec{q})$. Nous utilisons alors le Théorème 5.12 pour conclure ❏

5.4 Émissions linéaires sur les mots

Nous avons défini à la section précédente la classe des fonctions de communication où les messages émis ne sont pas consommés. Puis nous l'avons restreintes pour rester dans le reconnaissable.

Nous nous proposons maintenant de considérer le cas où les messages émis peuvent être reçus. Ce qui correspond au fonctionnement normal d'un automate à files. Bien sûr, nous devrons tout de même nous contenter de cas particuliers car la puissance du modèle rend le cas général inabordable.

Au vu de nos résultats des sections précédentes, nous nous sommes convaincus que seules les émissions posaient problème. En effet, elles seules peuvent nous faire sortir du cadre des reconnaissables ; d'où l'hypothèse pragmatique que nous nous sommes imposée au Corollaire 5.8, page 104. L'un de nos objectifs dans ce qui suit sera donc de restreindre nos hypothèses aux émissions et de n'en faire aucune concernant les réceptions.

Notre premier point de départ a été l'un des résultats de Finkel et Marcé [FM96]. Ils ont en effet caractérisé les conditions devant porter sur une séquence

μ de communication et un mot w de Σ^* pour qu'elle soit infiniment exécutable sur w ; c'est-à-dire :

$$\forall n \in \mathbb{N},\ \mu^n(w) \neq \emptyset$$

Or, l'expression qu'ils donnent de $\mu^n(w)$ ne dépend que de sa taille et de la séquence des émissions de μ. Le point intéressant, c'est qu'une fois μ fixé, le calcul ne dépende que de la taille du résultat. Or, gérer une taille revient à gérer un compteur. Et nous avons eu l'occasion de le rappeler lors de la démonstration du Corollaire 5.14, l'application d'un automate à un compteur à un ensemble reconnaissable de $\mathbb{N}$ est effectivement reconnaissable [FWW97]. Nous chercherons donc dans ce qui suit à nous ramener à ce cas particulier d'automate à un compteur, où recevoir ce qui a été émis ne pose pas de difficulté.

Notre deuxième point de départ a été le principal résultat de [BGWW97].

Théorème 5.15 ([BGWW97]) *Soient μ une séquence de communication sur Σ^* et W un reconnaissable de Σ^*. Alors,*

$$\mu^*(W) \text{ est effectivement reconnaissable}$$

Nous avons cherché à appliquer notre propre approche pour généraliser ce dernier résultat, c'est-à-dire :

- aucune contrainte sur les réceptions,
- mise en évidence des diverses synchronisations nécessaires,
- utilisation des fonctions de communication à émissions non consommées,
- utilisation d'un automate à un compteur.

Nous avons abouti au résultat suivant.

Théorème 5.16 *Soient W un reconnaissable de Σ^* et A un automate à une file définissant un langage de communication sur Σ^* tel que :*

$$\exists (v_{kj})_{\substack{k=1,\dots,m \\ j=1,\dots,n}} \in \Sigma^*,\ A_{1!} \subseteq \bigcup_{k=1,\dots,m} v_{k1}^* v_{k2}^* \dots v_{kn}^*$$

Alors $A(W)$ est effectivement reconnaissable.

Nous pouvons remarquer qu'étant donnée μ une séquence de communication sur Σ^*, μ^* vérifie la condition du théorème car $(\mu^*)_{1!} \subseteq (\mu_{1!})^*$. Par conséquent, ce théorème englobe le résultat du Théorème 5.15.

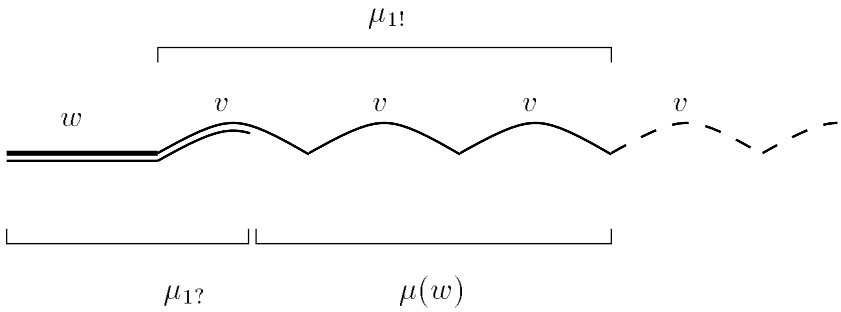


FIG. 5.7 – $\mu(w)$ *lorsque* $w \leq \mu_{1?}$

Avant d'en arriver à la démonstration formelle du Théorème 5.16 nous commencerons par une approche simplifiée. Pour cela, nous allons nous limiter au cas où $A = (Q, q_o, \Sigma, \Pi, \delta, F)$ est un automate à file définissant un langage de communication tel que $A_{1!} \subseteq v^*$ pour un $v \in \Sigma^*$ donné.

Soient $w \in W$ et $\mu \in A$ tels que $\mu(w) \neq \emptyset$. L'équation suivante est alors vérifiée :

$$\mu_{1?}\mu(w) = w\mu_{1!}$$

Deux cas sont possibles : $\mu_{1?} \leq w$ ou $w \leq \mu_{1?}$. Le premier cas correspond à une communication à émissions non consommées et sera pris en compte par $T_A(W)$ qui est ici reconnaissable car un seul canal est impliqué, Corollaire 5.8. Il nous reste à traiter le second cas qui est illustré par la Figure 5.7.

Nous pouvons remarquer que $\mu(w)$ est constitué d'un suffixe de v concaténé à un élément de v^*. Le mot v étant connu, il est aisé à partir de $|\mu(w)|$ et donc de $|w| + |\mu_{1!}| - |\mu_{1?}|$ d'en déduire $\mu(w)$.

Comme w est préfixe de $\mu_{1?}$, il existe μ' un préfixe de μ tel que $w = \mu'_{1?}$. Soit μ'' tel que $\mu'\mu'' = \mu$. Comme nous pouvons le constater sur la Figure 5.8, nous avons alors $\mu(w) = \mu''(\mu'_{1!})$. Or, $\mu'\mu'' \in A$ implique l'existence de deux états dont un final de l'automate A, $q \in Q$ et $q_f \in F$ tels que $\mu' \in A(q_o, q)$ et $\mu'' \in A(q, q_f)$.

Nous avons donc :

$$\forall \mu' \in A(q_o, q) \;\; \forall \mu'' \in A(q, q_f)$$
$$\mu'_{1?} \in W \implies \mu''(\mu'_{1!}) \in A(W)$$

Ce qui revient à :

$$A(q, q_f)\ \Big((A(q_o, q) \times_? W)_{1!}\Big) \subseteq A(W)$$

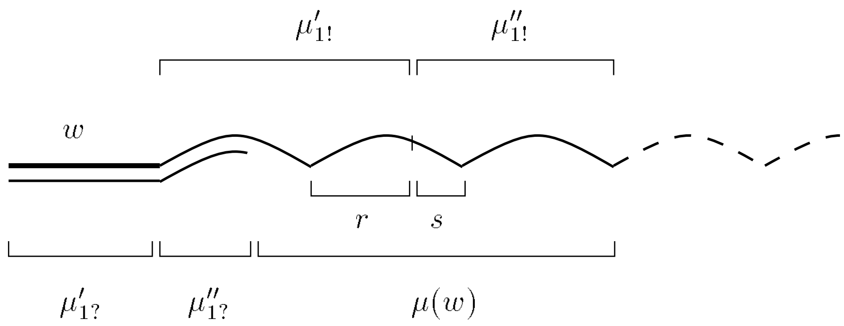


FIG. 5.8 – *Décomposition de μ en μ' et μ''*

Pour l'instant, nous ne pouvons pas calculer le membre gauche de cette inclusion. Nous allons donc affiner son expression. Pour cela, nous pouvons constater qu'il existe r et s, deux mots de Σ^*, tels que : $\mu'_{1!} \in (rs)^* r$ et $\mu''_{1!} \in s(rs)^*$. Ce qui nous permet d'écrire :

$$A(q, q_f) \times_! s(rs)^* \left(\left(\big(A(q_o, q) \times_? W \big) \times_! (rs)^* r \right)_{1!} \right) \subseteq A(W)$$

Il nous suffit alors de faire varier q, q_f, r et s pour obtenir la partie de $A(W)$ non prise en compte par $T_A(W)$. Ce faisant, il nous reste à montrer que l'on peut calculer l'expression à laquelle nous avons abouti. C'est ce que nous ferons lors du prochain lemme. Pour l'instant, nous pouvons énoncer le théorème suivant.

Théorème 5.17 *Soient $v \in \Sigma^*$, W un reconnaissable de Σ^* et $A = (Q, q_o, \Sigma, \Pi, \delta, F)$ un automate à une file définissant un langage de communication tel que $A_{1!} \subseteq v^*$. Alors $A(W)$ est effectivement reconnaissable et défini de la façon suivante :*

$$A(W) = \; T_A(W) \;\; \bigcup$$

$$\bigcup_{\substack{r,s \in \Sigma^* \\ rs = v \\ (q,q_f) \in Q \times F}} A(q, q_f) \times_! s(rs)^* \left(\left(\big(A(q_o, q) \times_? W \big) \times_! (rs)^* r \right)_{1!} \right)$$

Preuve :
Nous montrerons d'abord que l'égalité donnée est vérifiée, puis nous utiliserons le lemme qui suit pour montrer que nous sommes en mesure de calculer le membre droit de l'égalité.

Nous avons :

$$\begin{array}{rcll} A(W) = & & \{\mu(w) \mid \mu \in A,\ w \in W,\ \mu_{1?} \leq w\} & (I) \\ & \bigcup & \{\mu(w) \mid \mu \in A,\ w \in W,\ w \leq \mu_{1?}\} & (II) \end{array}$$

L'ensemble (I) est égal à $T_A(W)$. Montrons que l'ensemble (II) est égal à :

$$\bigcup_{\substack{r,s \in \Sigma^* \\ rs=v \\ (q,q_f) \in Q \times F}} A(q, q_f) \times_! s(rs)^* \left(\left(\left(A(q_o, q) \times_? W \right) \times_! (rs)^* r \right)_{1!} \right)$$

Nous procédons par une suite d'égalités :

$$\begin{array}{rcl} (II) & = & \{\mu(w) \mid \mu \in A,\ w \in W,\ w \leq \mu_{1?}\} \\ & = & \{(\mu'\mu'')(w) \mid \mu'\mu'' \in A,\ w \in W,\ \mu'_{1?} = w\} \\ & = & \{(\mu'\mu'')(\mu'_{1?}) \mid \mu'\mu'' \in A,\ \mu'_{1?} \in W\} \\ & = & \{\mu''(\mu'(\mu'_{1?})) \mid \mu'\mu'' \in A,\ \mu'_{1?} \in W\} \\ & = & \{\mu''(\mu'_{1!}) \mid \mu'\mu'' \in A,\ \mu'_{1?} \in W\} \\ & = & \bigcup_{\substack{\mu'\mu'' \in A \\ \mu'_{1?} \in W}} \{\mu''(\mu'_{1!})\} \end{array}$$

Or, par hypothèse, nous avons $A_{1!} \subseteq v^*$. Le cas où $(\mu'\mu'')_{1!} = \lambda$ correspond à des émissions non consommées (vu qu'il n'y a pas d'émission !) et est pris en compte par (I). Nous pouvons donc considérer ici que $A_{1!} \subseteq v^+$. Ce qui implique l'existence de deux mots r et s de Σ^* tels que : $rs = v$, $\mu'_{1!} \in (rs)^* s$ et $\mu''_{1!} \in s(rs)^*$. Nous obtenons alors :

$$(II) = \bigcup_{\substack{r,s\in\Sigma^* \\ rs=v}} \bigcup_{\substack{\mu'\mu''\in A \\ \mu'_{1?}\in W \\ \mu'_{1!}\in(rs)^*r \\ \mu''_{1!}\in s(rs)^*}} \{\mu''(\mu'_{1!})\}$$

$$= \bigcup_{\substack{r,s\in\Sigma^* \\ rs=v}} \bigcup_{(q,q_f)\in Q\times F} \bigcup_{\substack{\mu'\in A(q_o,q) \\ \mu'_{1?}\in W \\ \mu'_{1!}\in(rs)^*r \\ \mu''\in A(q,q_f) \\ \mu''_{1!}\in s(rs)^*}} \{\mu''(\mu'_{1!})\}$$

$$= \bigcup_{\substack{r,s\in\Sigma^* \\ rs=v \\ (q,q_f)\in Q\times F}} A(q,q_f)\times_! s(rs)^* \left(\left(\big(A(q_o,q)\times_? W\big)\times_! (rs)^* r\right)_{1!}\right)$$

Pour montrer que ce dernier membre droit de l'égalité est effectivement reconnaissable, il nous suffit de constater que :

$$\left(A(q,q_f)\times_! s(rs)^*\right)_{1!} \subseteq s(rs)^* \text{ et } \left(\big(A(q_o,q)\times_? W\big)\times_! (rs)^* r\right)_{1!} \subseteq (rs)^* r$$

et d'utiliser le lemme qui suit ❏

Nous aurons principalement besoin au lemme suivant d'extraire d'un reconnaissable de Σ^* l'ensemble des longueurs des ses éléments et de transformer un langage de communication sur Σ^* en un langage d'actions sur un compteur. D'où la définition suivante.

Définition 5.9

- Nous notons φ_{lg} le morphisme défini de la façon suivante :

  $\varphi_{lg} : \Sigma^* \longrightarrow (\mathbb{N}, +)$; $\forall a \in \Sigma,\ \varphi_{lg}(a) = 1$.

  $\varphi_{lg}(w)$ donne la longueur, en base 1, d'un mot $w \in \Sigma^*$.

- Nous notons φ_{cpt} le morphisme défini de la façon suivante :

$$\varphi_{cpt} : (\{1\}\times\{!,?\}\times\Sigma\cup\Pi)^* \longrightarrow (\{1\}\times\{!,?\}\times\{1\}\cup\Pi)^*$$

$$\begin{array}{lrcl} \forall a\in\Sigma, & \varphi_{cpt}(1!a) & = & 1!1 \\ \forall a\in\Sigma, & \varphi_{cpt}(1?a) & = & 1?1 \\ \forall \pi\in\Pi, & \varphi_{cpt}(\pi) & = & \pi \end{array}$$

- Soit A un automate à une file définissant un langage de communication, nous notons $\varphi_{cpt}(A)$ l'automate à un compteur (sans test à zéro !) obtenu à partir A en appliquant φ_{cpt} aux actions de communication de A.

- Soit $W \subseteq \Sigma^*$, nous notons *pref*(W) l'ensemble des préfixes de W et *suff*(W) l'ensemble de ses suffixes, c'est-à-dire :

$$\begin{array}{rcl} \mathit{pref}(W) & = & \{v \in \Sigma^* \mid \exists w \in W,\ v \leq w\} \\ \mathit{suff}(W) & = & \{v \in \Sigma^* \mid \exists w \in W\ \exists v' \in \Sigma^*,\ v'v = w\} \end{array}$$

Lemme 5.18 *Soient $W \subseteq (rs)^*r$ un langage reconnaissable avec $r, s \in \Sigma^*$ et A un automate à une file définissant un langage de communication sur Σ^* tel que $A_{1!} \subseteq s(rs)^*$. Alors $A(W)$ est effectivement reconnaissable et*

$$A(W) = \mathit{suff}\big((rs)^*\big) \bigcap \varphi_{lg}^{-1}\Bigg(\varphi_{cpt}\bigg(A \times_? \mathit{pref}\big((rs)^*\big)\bigg)\ \Big(\varphi_{lg}(W)\Big)\Bigg)$$

Preuve :

Soient les notations suivantes :

- $A' = A \times_? \mathit{pref}\big((rs)^*\big)$
- $A'' = \varphi_{cpt}(A')$
- $X = \varphi_{lg}(W)$
- $Y = A''(X)$.
- $\begin{array}{rcl} W' & = & \{w' \in \Sigma^* \mid w' \in \mathit{suff}\big((rs)^*\big),\ |w'| \in Y\} \\ & = & \mathit{suff}\big((rs)^*\big) \cap \varphi_{lg}^{-1}(Y) \end{array}$

Nous montrerons dans un premier temps que $A(W) = W'$ et dans un deuxième temps que W' est effectivement reconnaissable.

1. $A(W) = W'$

   Soient $\mu \in A$, $w \in W$ et $w' \in \Sigma^*$. Nous avons l'équivalence suivante :

$$\mu(w) = w' \iff \begin{array}{ll} \forall \mu' \leq \mu,\ \mu'_{1?} \leq w\mu'_{1!} & \text{(I)} \\ \mu_{1?}w' = w\mu_{1!} & \text{(II)} \end{array}$$

La condition (I) implique que $\mu_{1?} \leq WA_{1!}$ or $WA_{1!} \subseteq (rs)^*rs(rs)^* = (rs)^+$. Nous avons ainsi $\mu_{1?} \leq (rs)^*$ et par conséquent, $A(W) = A'(W)$.

Nous allons donc considérer à partir de maintenant que $\mu \in A'$. Soit $(rs)^\omega$ le mot infini constitué du mot rs répété infiniment. Nous avons $\mu_{1?} \leq (rs)^\omega$ et $w\mu_{1!} \leq (rs)^\omega$. La condition (I) est alors équivalente à :

$$\forall \mu' \leq \mu,\ |w| + |\mu'_{1!}| - |\mu'_{1?}| \geq 0$$

et donc aussi à :

$$\varphi_{cpt}(\mu)(|w|) = |w'|$$

Les conditions (I) et (II) sont donc équivalentes à

$$w' \in \mathit{suff}\big(w\mu_{1!}\big) \text{ et } |w'| = \varphi_{cpt}(\mu)\ (|w|)$$

Or, comme $w\mu_{1!} \in WA_{1!}$ et $WA_{1!} \subseteq (rs)^+$, nous obtenons

$$\mu(w) = w' \Longleftrightarrow w' \in \mathit{suff}\big((rs)^*\big) \text{ et } |w'| = \varphi_{cpt}(\mu)\ (|w|)$$

Ce qui signifie bien que $A(W) = W'$.

2. Montrons maintenant que W' est effectivement reconnaissable.

   Les ensembles $\mathit{pref}\big((rs)^*\big)$ et $\mathit{suff}\big((rs)^*\big)$ sont effectivement reconnaissables. Ainsi, pour obtenir un automate fini reconnaissant $\mathit{pref}\big((rs)^*\big)$, il suffit de considérer comme *acceptants* tous les états d'un automate reconnaissant $(rs)^*$. Inversement, pour obtenir un automate fini reconnaissant $\mathit{suff}\big((rs)^*\big)$, il suffit de considérer comme *initiaux* tous les états d'un automate reconnaissant $(rs)^*$.

Par conséquent, les automates à files A' et A'' sont parfaitement constructibles à partir de A.

L'ensemble W est un reconnaissable de Σ^* que nous supposons défini à partir d'un automate fini que nous notons A_W. Pour obtenir un automate définissant X, il suffit d'étiqueter toutes les transitions de A_W par 1. L'ensemble Y résulte, lui, de l'application d'un automate à un compteur sur un ensemble reconnaissable de $\mathbb{N}$. Il est donc effectivement reconnaissable.

Les reconnaissables étant *effectivement* stables par morphisme inverse, et par intersection, W' est effectivement reconnaissable. En fait, un automate reconnaissant W' peut être obtenu en synchronisant celui reconnaissant $suff\big((rs)^*\big)$ et celui reconnaissant Y ❑

Nous pouvons maintenant prouver le résultat du Théorème 5.16 annoncé au début de cette section .

Corollaire 5.19 *Soient W un reconnaissable de Σ^* et A un automate à une file définissant un langage de communication sur Σ^* tel que :*

$$\exists (v_{kj})_{\substack{k=1,\dots,m\\ j=1,\dots,n}} \in \Sigma^*,\ \ A_{1!} \subseteq \bigcup_{k=1,\dots,m} v_{k1}^* v_{k2}^* \dots v_{kn}^*$$

Alors $A(W)$ est effectivement reconnaissable.

Preuve :

L'opérateur $\times_!$ est distributif par rapport à l'union. Nous avons donc :

$$A(W) = \bigcup_{k=1,\dots,m} A \times_! v_{k1}^* v_{k2}^* \dots v_{kn}^* \ (W)$$

Considérons que $A = (Q, q_o, \Sigma, \Pi, \delta, F)$. Nous avons alors

$$A \times_! v_{k1}^* v_{k2}^* \dots v_{kn}^* = \bigcup_{\substack{q_1,\dots,q_{n-1}\in Q\\ q_n\in F}} \big(A(q_0,q_1) \times_! v_{k1}^*\big).\big(A(q_1,q_2) \times_! v_{k2}^*\big) \cdots \\ \cdots \big(A(q_{n-1},q_n) \times_! v_{kn}^*\big)$$

Soit la suite $(W_j)_{j=0,\dots,n} \subseteq \Sigma^*$ définie de la façon suivante.

$$\begin{array}{rcl} W_0 & = & W \\ W_j & = & A(q_{j-1},q_j) \times_! v_{kj}^* \ (W_{j-1}) \end{array}$$

En utilisant le Théorème 5.17, comme W est reconnaissable et que nous supposons en avoir une représentation, alors tous les W_j sont effectivement reconnaissables. De plus, par rapport à ce qui précède, nous avons

$$A \times_! v_{k1}^* v_{k2}^* \dots v_{kn}^* = W_n$$

Il s'en suit que $A(W)$ est effectivement reconnaissable comme union finie d'ensembles effectivement reconnaissables ❑

Chapitre 6

Conclusion et Perspectives

Nous avons étudié, au cours de cette thèse, le modèle des automates communicants qui fournit un support formel à la vérification des algorithmes répartis.

Ce modèle étant indécidable, nous avons considéré deux approches complémentaires. La première a été de s'inspirer des contraintes réelles de communication entre processus pour réduire sa puissance de description, afin d'obtenir des possibilités de vérification.

Ainsi, nous avons envisagé, dans une première partie, l'utilisation de canaux défaillants entre les automates. Trois types d'erreurs, et leurs combinaisons, ont été étudiées : la perte, l'insertion et la duplication des messages. Nous avons notamment montré, en faisant appel à des bisimulations, que l'erreur de duplication est toujours récupérable et que l'accessibilité est décidable pour les systèmes dont la perte de messages ne se fait qu'au début des canaux.

Dans une seconde partie, nous avons considéré le mode half-duplex de communication entre les automates. Dans le cas où seuls deux automates sont impliqués, nous obtenons une description effective de l'ensemble, reconnaissable, des états accessibles. Cette propriété n'est pas préservée lorsque plusieurs automates sont impliqués. Nous avons alors proposé une classe qui permet de considérer des systèmes de plus de deux automates tout en préservant les bonnes propriétés de vérification des systèmes half-duplex à deux automates.

La seconde approche que nous avons investie est de nature optimiste, mais néanmoins réaliste. Elle consiste à garder les possibilités de description du modèle, et à mettre en œuvre des semi-algorithmes de vérification. Le pari est qu'en pratique, toute la puissance du modèle n'est pas réellement utilisée ; ce qui permet aux semi-algorithmes mis en œuvre de s'arrêter correctement.

La dernière partie de notre étude a ainsi consisté à développer les concepts de transitions et d'états symboliques qui permettent d'approximer l'ensemble des états accessibles d'un systèmes. Ce faisant, nous avons amélioré les résultats de [RG84] et [PP92], ainsi que le principal résultat de [BGWW97].

Perspectives

Au niveau de l'étude des systèmes utilisant des canaux non fiables, le problème de l'indécidabilité de la finitude de l'ensemble d'accessibilité reste en suspens. Nous nous y sommes attaqué, mais sans succès pour l'instant. Nous pensons que sa résolution permettrait de conclure à une autre indécidabilité, celle de l'existence d'une bisimulation entre deux *lossy systems*.

Au niveau de l'étude des systèmes half-duplex, il serait intéressant de s'attaquer au problème de la décidabilité du model-checking via la logique temporelle LTL. Ce problème semble indécidable pour les systèmes utilisant des canaux quasi-stables.

Le thème de notre étude qui offre le plus de perspectives est sans aucun doute celui relatif aux transitions symboliques. Entre autres, il nous parait intéressant d'étudier la notion d'arbre réduit et symbolique d'accessibilité. L'intérêt par rapport au graphe symbolique que nous avons esquissé au Chapitre 4 est de pouvoir calculer une exécution menant à un état généré par l'utilisation successives de transitions symboliques. Enfin, nous envisageons d'appliquer ces notions à d'autres modèles que les automates communicants.

Bibliographie

[AČJT96] Parosh Aziz Abdulla, Kārlis Čerāns, Bengt Jonsson, and Yih-Kuen Tsay. General decidability theorems for infinite-state systems. In *Proceedings, 11*th *Annual IEEE Symposium on Logic in Computer Science* [IEE96], pages 313–321.

[AJ93] Parosh Aziz Adulla and Bengt Jonsson. Verifying programs with unreliable channels. In *Proceeding 8th IEEE Symp. on Logic in Computer Science*, 1993.

[AJ94] P. Abdulla and B. Jonsson. Undecidable verification problems for programs with unreliable channels. In *Proc. of ICALP*, volume 820 of *LNCS*, 1994.

[Arn92] A. Arnold. *Systèmes de transitions finis et sémantique des processus communicants.* Masson, 1992.

[Ber79] J. Berstel. *Transductions and Context-Free Languages.* B.G. Teubner Stuttgart, 1979.

[BG96] 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.

[BGWW97] Bernard Boigelot, Patrice Godefroid, Bernard Willems, and Pierre Wolper. The power of QDDs. In *Proceedings SAS'97*, September 1997.

[BH97] A. Bouajjani and P. Habermehl. Symbolic reachability analysis of FIFO-channel systems with nonregular sets of configurations. In *Proceedings of the 24th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume LNCS 1256, pages 560–570, July 1997.

[Boc78] G.V. Bochmann. Finite state description of communication protocols. In *Proc. of Comp. Network Protocols Symp.*, pages F3–1—F3–11, Liège, Belgique, February 1978.

[BZ83] D. Brand and P. Zafiropulo. On communicating finite-state machines. *JACM*, 30(2):323–342, April 1983.

[CF97] G. Cécé and A. Finkel. Programs with quasi-stable channels are effectively recognizable. In *Proc. of CAV'97, LNCS 1254, Israel*, pages 304–315, June 1997.

[CFP94] Gérard Cécé, Alain Finkel, and Iyer S. Purushothaman. Duplication, insertion and losiness errors in unreliable communication channel. In David Wile, editor, *Proc. of the 2nd ACM SIGSOFT' 94 Symposium on Foundations of Software Engineering*, volume 19, pages 35–43. ACM Press, December 1994.

[CFP96] Gérard Cécé, Alain Finkel, and Iyer S. Purushothaman. Unreliable channels are easier to verify than perfect channels. *Information and Computation*, 124(1):20–31, 1996.

[CLR90] Thomas Cormen, Charles Leiserson, and Ronald Rivest. *Introduction to algorithms*. The MIT Press, 1990.

[FC87] A. Finkel and A. Choquet. Simulation of linear fifo nets by petri nets having a structured set of terminal markings. In *Proc. of 8th European Workshop on Application and Theory of Petri Nets, Saragoza*, pages 95–112, 1987.

[Fin82] Alain Finkel. About monogeneous fifo nets. In *Proc. of 3rd European Workshop on Application and Theory of Petri Nets, Varenna, Italy*, 1982.

[Fin86] Alain Finkel. Structuration des systèmes de transitions ; application au contrôle du parallélisme par files fifo. In *Université d'Orsay*, Juin 1986. Thèse d'État.

[Fin90] A. Finkel. Reduction and covering of infinite reachability trees. *Information and Computation*, 89(2):144–170, 1990.

[Fin93] Alain Finkel. The minimal coverability graph algorithm. In *Advances in Petri Nets*, volume LNCS 674, pages 210–243. Springer-Verlag, 1993.

[Fin94] A. Finkel. Decidability of the termination problem for completely specified protocols. *Distributed Computing*, 7:129–135, 1994.

[FM96] A. Finkel and O. Marcé. Verification of infinite regular comunicating automata. Technical report, LIFAC, ENS de Cachan, April 1996.

[FM97] A. Finkel and P. McKenzie. Verifying identical communicating processes is undecidable. *Theoretical Computer Science*, 174:217–230, 1997.

[FS] Alain Finkel and Philippe Schnoebelen. well-structured transition systems everywhere ! soumis.

[FWW97] Alain Finkel, Bernard Willems, and Pierre Wolper. A direct symbolic approach to model checking pushdown systems. In Faron Moller, editor, *Infinity'97, 2^{nd} International Workshop on* Verification of Infinite State Systems, pages 30–40, Box 311, S-751 05 Uppsala, Sweden, July 97.

[GER87] M.G. Gurari Gouda, E.M., and L. E. Rosier. On deadlock detection in systems of communicating finite state machines. *Computer and Artificial Intelligence*, 6(3):209–228, 1987.

[GMY84] M. G. Gouda, E. G. Manning, and Y. T. Yu. On the progress of communication between two finite state machines. *Information and Control*, 63:200–216, 1984.

[GR84] M. G. Gouda and L. E. Rosier. Deciding progress for a classe of communication protocols. In *Proceeding of the 8th Annual Conference on Information Sciences and systems*, pages 663–667, Princeton University, 1984.

[Hig52] Graham Higman. Ordering by divisibility in abstract algebras. *Proceeding of the London Mathematical Society*, 2(7):326–336, September 1952.

[HP79] John Hopcroft and Jean-Jacques Pansiot. On the reachability problem for 5-dimensional vector addition systems. *TCS*, 8:135–159, 1979.

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

[IEE96] IEEE Computer Society Press. *Proceedings, 11^{th} Annual IEEE Symposium on Logic in Computer Science*, New Brunswick, New Jersey, 27–30 July 1996.

[IN97] Purush Iyer and Murali Narasimha. Probabilistic lossy channel systems. In Michel Bidoit and Max Dauchet, editors, *Proce. of TAPSOFT'97*, volume 1214 of *LNCS*, Lille, France, April 1997. Springer-Verlag.

[JJ93] T. Jéron and C. Jard. Testing for unboundedness of fifo channels. *Theoretical Computer Science*, 113:93–117, 1993.

[Jér91] Thierry Jéron. Testing for unboundedness of fifo channels. In *STACS'91*, pages 322–333, Hamburg, Germany, February 1991. Spinger-Verlag.

[Kel76] R.M. Keller. Formal verification of parallel programs. *Communications of the ACM*, 19(7):371–384, July 1976.

[KM69] R. M. Karp and R. E. Miller. Parallel program schemata. *Journal Computer System Science*, 4:147–195, 1969.

[Kos82] S. R. Kosaraju. Decidability of reachability in vector addition systems. In *Proceeding 14th Annual Symposium on Theory of Computing*, pages 267–281, 1982.

[Lot83] M. Lothaire. *Combinatorics on Words*, chapter 6: Subwords. Addison-Wesley, 1983.

[May84] E. W. Mayr. An algorithm for the general petri net reachability problem. *SIAM, Journal Computing*, 13(3):441–459, 1984.

[Mil89] Robin Milner. *Communication and concurency.* Prentice Hall, 1989.

[Mou89] H. Mountassir. Sur la progression de la communication entre processus communicants. Étude d'une classe particulière de machines déterministes. Technical report, UFR des Sciences et techniques, Laboratoire d'Informatique; 16 route de Gray, 25030 Besançon FRANCE, 1989.

[Niv68] Nivat. Transduction des langages de chomsky. *Ann. de l'Inst. Fourier*, 18:339–456, 1968.

[Pac82] Jan K. Pachl. Reachability problems for communicating finit state machines. Research Report CS-82-12, Department of Computer Science University of Waterloo, Waterloo, Ontario, Canada, May 1982.

[Pac87] J.K. Pachl. Protocol description and analysis based on a state transition model with channel expressions. In *Proc. of Protocol Specification, Testing and Verification, VII*, 1987.

[Par81] D. Park. Concurrency and automata on infinite sequences. In *5^{th} GI Conf. on Theoret. Comput. Sci., LNCS 104*, pages 167–183, 1981.

[Pos46] E. Post. A variant of a recursively unsolvable problem. *Bull. AMS*, 52:264–268, 1946.

[PP91] W. Peng and S. Purushothaman. Data flow analysis of communicating finite state machines. In *ACM Transactions on Programming Languages and Systems*, volume 13, pages 399–442, 1991.

[PP92] S. Purushothaman and Wuxu Peng. Analysis of a class of communicating finite state machines. *Acta Informatica*, 29:499–522, 1992.

[RG84] L. Rosier and M. Gouda. On deciding progress for a classe of communication protocols. In *Proc. of the 18*th *Annual Conference on Information Sciences and Systems*, pages 663–667, 1984. Une version complete constitue un rapport interne : TR-83-22, Dept. of computer Sciences, Univ. of Texas at Austin. Oct. 83.

[Tur93] Kenneth J. Turner, editor. *Using formal description techniques. A introduction to Estelle, Lotos and SDL.* Wiley, John & Sons Ltd., Baffins Lane, Chichester West Sussex PO19 1UD, England, 1993.