RAIRO. INFORMATIQUE THÉORIQUE

A. ALAIWAN

Langages persistants

RAIRO. Informatique théorique, tome 18, n° 3 (1984), p. 259-278

<http://www.numdam.org/item?id=ITA_1984__18_3_259_0>

R.A.I.R.O. Informatique théorique/Theoretical Informatics
(vol. 18, n° 3, 1984, p. 259 à 278)

LANGAGES PERSISTANTS (*)

par H. ALAIWAN (2)

Communiqué par A. ARNOLD

Résumé. — *Cet article traite de la notion de persistance (commutative) dans la théorie des langages. Cette notion est souvent rencontrée dans les systèmes à calcul parallèle non déterministe.*

Une définition formelle de la persistance (commutative) est présentée, ce qui donne lieu à une famille de langages. Sa stabilité par les opérateurs usuels est alors étudiée. Enfin, une procédure est proposée pour tester la persistance d'un langage dans le cas rationnel.

Abstract. — *This paper considers the notion of (commutative) persistency in language theory. This notion is often encountered in nondeterministic parallel computation systems.*

A formal definition of (commutative) persistency is given, that gives rise to a family of languages. The author then examines its stability under the usual operators and finally puts forward a procedure for testing the persistency of a language in the rational case.

INTRODUCTION

Soit à considérer le cas de deux usagers u_1 et u_2 ayant accès à une bibliothèque de données, contrôlée par un moniteur. Supposons que les usagers envoient simultanément une requête d'écriture ou de lecture. En l'absence de moniteur, cette situation de concurrence peut entraîner un malfonctionnement. Si u_1 accède aux données en premier, u_2 est alors mis en attente. Durant cette attente, il se peut qu'un troisième usager u_3 demande à son tour l'accès aux données. Ainsi u_2 et u_3 se retrouvent dans une situation de concurrence analogue à la précédente. Un moniteur équitable doit pouvoir donner l'accès à u_2 aussitôt que u_1 termine sa tâche, mais si la priorité de u_3 est plus forte, u_2 est mis en attente une fois de plus. Cette propriété d'« attente jusqu'à satisfaction » est connue sous le nom de « persistance ». Quant à l'évolution ultérieure de l'ensemble, elle peut dépendre de l'ordre de passage

(*) Reçu en février 1983, révisé en septembre 1983.
(1) Thomson-C.S.F., Domaine de Corbeville, B.P. n° 10, 91401 Orsay.

0399-0540/84/03 259 20/$ 4.00/

des usagers. Si ce n'est pas le cas, on dit alors que la persistance est « commutative ».

La notion de persistance a été étudiée dans le cadre des systèmes à calcul parallèle non déterministe [3, 5, 7, 8], tels que les schémas de programmes parallèles et les réseaux de Petri. Devant la difficulté des problèmes soulevés par cet indéterminisme, l'hypothèse de persistance a permis de restreindre ce dernier, puis de répondre à des questions sans solution dans le cas général. R. M. Karp et R. E. Miller [3] ont démontré qu'un schéma de programmes parallèles, persistant, commutatif et sans perte, est déterministe. Quant aux réseaux de Petri, la commutativité est assurée par la commutativité de l'addition. R. M. Keller [4] a montré que la vivacité d'un réseau de Petri persistant est décidable. La question de savoir si un réseau de Petri est à son tour persistant était restée ouverte jusqu'à ce que E. Mayr [6] en démontre la décidabilité. Enfin, les travaux de H. Yamasaki [9] vinrent agrandir nos connaissances dans ce domaine.

La motivation pratique de ce papier provient du besoin de mieux étudier la persistance indépendamment du modèle choisi. Notre objectif est donc de regrouper et d'unifier les résultats connus sur cette propriété, plus que d'essayer d'ajouter de nouveaux théorèmes. Comme les processus sont représentés par des langages, il est normal donc de choisir la théorie des langages comme approche. Ainsi, formellement, un langage L sur un alphabet A est dit persistant si et seulement si :

(1) il est fermé par facteur gauche, c'est-à-dire si $x \in L$ et $y < x$ alors $y \in L$;

(2) pour tout élément f de L [ou de $FG(L)$], et pour tout a, $b \in A$ $(a \neq b)$ on a l'implication :

$$f.a, f.b \in L \Rightarrow f.a.b, f.b.a \in L.$$

De plus, L est dit commutatif si et seulement si pour tout $h \in A^*$ on a l'équivalence :

$$f.a.b.h. \in L \Leftrightarrow f.b.a.h. \in L.$$

La condition (1) nous semble raisonnable puisque les comportements des processus réels présentent cette propriété.

Avec cette approche, on retrouve évidemment les résultats classiques sur la persistance, et on est conduit naturellement à étudier le comportement des langages persistants vis-à-vis des opérations usuelles de la théorie des langages. Quoique les réponses à certaines questions soient aisées, il existe néanmoins des problèmes sans réponses, comme la construction d'un langage persistant contenant un ensemble donné de langages...

La section 1 de ce papier est un bref rappel des définitions de base de la théorie des langages. On introduit dans la section 2, la définition d'un langage persistant. Après quelques résultats techniques, on propose une méthode pour vérifier que le langage d'un système de transition est persistant. Dans la section 3, on rappelle la notion de congruence à droite, afin de mieux définir dans la suite, la persistance commutative. On donne dans la suite, une condition nécessaire avec quelques propriétés se rattachant à cette condition. Dans une deuxième partie on expose une suite de résultats liés aux opérations classiques, puis on propose de nouveau une méthode pour vérifier la persistance commutative d'un langage associé à un système de transition. Enfin, on rappelle dans la section 4, la définition de la persistance faible ainsi que les principaux résultats concernant cette notion.

1 DÉFINITIONS

Dans ce paragraphe nous rappelons les concepts de base concernant la théorie des langages ainsi que la définition d'un système de transition.

1.1. Langage

Soit A un ensemble fini de symboles appelé *alphabet;* on note A^* le monoïde libre engendré par A.

Les éléments de A sont dits des *lettres*, de A^* des *mots.*

On note pour tout $f \in A^*$, $|f|$ *la longueur* de f. Le seul mot de longueur nulle est le *mot vide*, noté λ.

La relation $\leqq$ est définie sur $A^* \times A^*$ par :

$$f \leqq g \text{ si et seulement si } \exists h \in A^* \quad \text{tel que} \quad f.h = g.$$

On dit alors que f est *facteur gauche* de g.

Un *langage* sur un alphabet fini A, est un sous-ensemble de A^*. Pour un langage L, on note $FG(L)$ l'ensemble des facteurs gauches des éléments de L :

$$FG(L) = \{ h \in A^* / \exists g \in A^* \text{ et } h.g \in L \}.$$

On dit que L est *fermé par facteur gauche* quand il vérifie $L = FG(L)$.

On note, pour un mot f, $\mathrm{Ops}(f)$ l'ensemble des lettres de A « occurant » au moins une fois dans f :

$$\mathrm{Ops}(f) = \{ a \in A / f \in A^*.a.A^* \}.$$

Pour un langage L, on pose :

$$\mathrm{Ops}(L) = \bigcup_{f \in L} \mathrm{Ops}(f).$$

1.2. Système de transition

La définition qu'on va introduire, n'est autre que celle d'un automate classique [2]. En réalité, les systèmes de transition comportent outre les données relatives à un automate, un ensemble de configurations qui représente des comportements infinis [1].

Un *système de transition* $\mathscr{S}$ se compose :

- d'un ensemble C de configurations qui pourra être fini ou infini;
- d'un sous-ensemble D de C de configurations dites initiales;
- d'un sous-ensemble C_f de C de configurations dites finales;
- d'un ensemble fini d'actions réelles A;
- d'un ensemble de transitions T, sous-ensemble de $C \times A \times C$.

REMARQUES : Une transition $(c, a, c') \in T$ indique que $\mathscr{S}$ peut passer de la configuration c à la configuration c' en réalisant l'action a. Pour nous une transition s'opère en une unité de temps indivisible.

Introduisons maintenant d'autres notations bien commodes :

Pour une lettre $a \in A$:

$$c \xrightarrow{a} c' \quad \Leftrightarrow \quad (c, a, c') \in T.$$

Pour un mot $f = a.g$ et $f \in A.A^*$

$$c \xrightarrow{f} c' \quad \Leftrightarrow \quad \exists c'' \in C \qquad \text{tel que} \quad \begin{cases} c \xrightarrow{a} c'', \\ c'' \xrightarrow{g} c'. \end{cases}$$

Le langage (ou ensemble de comportements) d'un système de transition $\mathscr{S}$, noté $L(\mathscr{S})$ est l'ensemble de toutes les suites d'actions qui partent d'un élément de l'ensemble des configurations initiales et arrivent à un élément de l'ensemble des configurations finales :

$$L(\mathscr{S}) = \{ f \in A^* / \exists (c, c') \in D \times C_f \text{ tel que } c \xrightarrow{f} c' \}.$$

2. LANGAGES PERSISTANTS

On introduit dans cette section une nouvelle classe de langages sur un alphabet A, celle des langages *persistants*. Après un passage rapide sur les propriétés de cette classe, on donne en dernière partie un moyen de caractériser un système de transition dont le langage appartient à cette classe.

Dans la plupart des systèmes où il y a un conflit entre deux actions a et b, le choix d'exécuter a à la place de b, rend b non exécutable. Par exemple c'est le cas d'un consommateur se trouvant devant deux articles de même prix, mais n'ayant que le prix d'un seul. La propriété de persistance consiste à conserver la possibilité d'exécuter b après a ou inversement.

2.1. Définition

Un langage L est dit *persistant* ssi $L = FG(L)$ *et :*

$$\forall f \in L, \quad \forall a, b \in A, \qquad a \neq b,$$

alors ($f.a$ et $f.b \in L \Rightarrow f.a.b$ et $f.b.a \in L$).

On note $\mathscr{L}_p$ la classe des langages persistants.

2.2. Exemple

Soit à considérer le langage L suivant : $L = [a(b \cup ab)]^*$, alors $FG(L)$ est persistant. Soit $f \in FG(L)$ alors il existe deux suites d'entiers positifs $(n_i)_{i \in N}$ et $(m_i)_{i \in N}$ telles que :

$$f \in (ab)^{n_0}(a^2 b)^{m_0} \ldots (ab)^{n_i}(a^2 b)^{m_i}.(a \cup a^2 \cup \lambda).$$

Posons :

$$g = (ab)^{n_0}(a^2 b)^{m_0} \ldots (ab)^{n_i}(a^2 b)^{m_i}.$$

Si $f = g$, alors la seule action possible est a.

Si $f = g.a$ on a $f.a$ et $f.b \in FG(L)$. Or $g.(a.a.b)$ et $g.(a.b).a$ appartiennent aussi à $FG(L)$.

Enfin si $f = g.a^2$ alors la seule action possible est b.

2.3. Propriétés

Nous donnons maintenant une propriété qui justifie le terme de « persistance ».

Propriété de persistance : Soit L persistant et soient :

$$f \in L, \quad a \in A \quad \text{et} \quad g \in A^*$$

tels que :

$$f.a \quad \text{et} \quad f.g \in L \quad \text{et} \quad a \notin \text{Ops}(g) \quad \text{alors} \quad f.g.a \in L.$$

La propriété ci-dessus signifie que si à un moment donné l'action a est en conflit avec une autre action, disons b, et que l'on choisit d'exécuter b, alors a reste en conflit avec les actions éventuelles qui suivent b.

Preuve : Elle se fait par induction sur la longueur de g. □

La classe $\mathscr{L}_p$ n'est pas stable pour la plupart des opérations usuelles. La réunion de deux langages de $\mathscr{L}_p$ n'est pas un langage de $\mathscr{L}_p$ comme le montre le contre-exemple suivant :

$L_1 = (a \cup ab)^*$ et $L_2 = (a \cup ac)^*$. Il est facile de vérifier que L_1 et L_2 sont dans $\mathscr{L}_p$. Alors $L_1 \cup L_2 \notin \mathscr{L}_p$. En effet si on prend $f = a$, on a $f.b$ et $f.c \in L_1 \cup L_2$, mais $f.b.c.$ (ni $f.c.b.$) n'appartient pas à $L_1 \cup L_2$.

En revanche, l'intersection de deux langages de $\mathscr{L}_p$ est dans $\mathscr{L}_p$.

PROPOSITION 1 : *Soient L_1 et $L_2 \in \mathscr{L}_p$ alors $L_1 \cap L_2 \in \mathscr{L}_p$.*

Preuve : Sachant que $L_1 \cap L_2 \subseteq FG(L_1 \cap L_2) \subseteq FG(L_1) \cap FG(L_2)$ et que $L_i = FG(L_i)\ i = 1,2$ on a $L_1 \cap L_2 = FG(L_1 \cap L_2)$.

Soient $f \in L_1 \cap L_2$, a et $b \in A\ (a \neq b)\ tq$: $f.a$ et $f.b \in L_1 \cap L_2$ on a donc $f.a$ et $f.b \in L_i$, $i = 1,2$.

Par persistance, $f.a.b$ et $f.b.a \in L_i$, $i = 1,2$ donc $f.a.b$ et $f.b.a \in L_1 \cap L_2$.

C.Q.F.D. □

2.4. Système de transition persistant

Un système de transition est dit persistant ssi son langage $L(\mathscr{S})$ appartient à $\mathscr{L}_p$.

On s'intéresse dans cette partie à la caractérisation des systèmes persistants. Pour ceci, on va d'abord introduire une nouvelle notation.

On note $\delta(c) = \{ a \in A / \exists\, c' \in C\ tq\ c \xrightarrow{a} c' \}$.

On rappelle qu'un système $\mathscr{S}$ est déterministe quand pour tout couple (c, a) de $C \times A$ on a $\text{Card}\ (\{ c' \in C / c \xrightarrow{a} c' \}) \leqq 1$. $\mathscr{S}$ est dit réduit quand pour tout $c \in C$. il existe :

$$c_1 \in D, \quad c_2 \in C_f, \quad f \in A^* \quad \text{et} \quad g \in A^*\ tq\ c_1 \xrightarrow{f} c \xrightarrow{g} c_2 \quad [1].$$

On peut énoncer alors la proposition suivante :

PROPOSITION 2 : *Pour un système $\mathscr{S}$, déterministe et réduit, on a l'équivalence :*

$$\mathscr{S} \text{ est persistant} \Leftrightarrow \begin{cases} C_f = C, \\ \forall (c, a, c') \in T, \quad \delta(c) - \{a\} \subseteq \delta(c'). \end{cases}$$

Preuve : Il est facile de voir que la condition $C_f = C$ est nécessaire et suffisante pour que $L(\mathscr{S})$ soit fermé par facteur gauche. On suppose dans la suite que $L(\mathscr{S}) = FG(L(\mathscr{S}))$.

Démontrons l'implication de gauche à droite, et considérons une transition $c \xrightarrow{a} c'$.

Comme $\mathscr{S}$ est réduit, il existe $f, g \in A^*$, $c_1 \in D$ et $c_2 \in C_f$ tq :

$$c_1 \xrightarrow{f} c \xrightarrow{a} c' \xrightarrow{g} c_2.$$

Posons $L = L(\mathscr{S})$.

Si $\delta(c) - \{a\}$ est vide, alors l'inclusion est vraie.

Supposons que $\delta(c) - \{a\} \neq \emptyset$, et soit $b \in \delta(c) - \{a\}$. Donc on a $f.a$ et $f.b \in L$ (car $\mathscr{S}$ est réduit). Par persistance de L on a donc $f.a.b$ et $f.b.a \in L$. Comme $\mathscr{S}$ est déterministe, on a forcément $c' \xrightarrow{b} c''$, donc $b \in \delta(c')$ et par conséquent l'implication est démontrée.

Démontrons l'implication inverse $\Leftarrow$. Soient f, a, b définis comme dans le 2.1. On sait qu'il existe $c_1 \in D$, c et $c' \in C$ *tq :*

$$c_1 \xrightarrow{f} c \xrightarrow{a} c'.$$

Par hypothèse $\delta(c) - \{a\} \subseteq \delta(c')$, $b \in \delta(c) - \{a\}$ (car $\mathscr{S}$ est déterministe), donc on a aussi $c' \xrightarrow{b} c''$, *d'où* $f.a.b \in L$ (car $\mathscr{S}$ est réduit). On montre de même que $f.b.a \in L$.

C.Q.F.D. □

REMARQUES : Cette proposition ne s'étend pas aux systèmes non déterministes; la difficulté principale réside dans le fait que pour un mot $f \in L(\mathscr{S})$, il peut y avoir plusieurs $c \in D$ et $c' \in C$ *tq* $c \xrightarrow{f} c'$.

La proposition est évidemment valable même quand Card(C) est infini.

La complexité d'un algorithme basé sur la proposition 2 dépend de la structure de données choisie. En prenant une représentation par file à successeurs et une exploration de système en profondeur ou largeur, le coût est en $O(d^3 N)$, où $N = \text{Card}(C)$ et $d = \max_{c \in C} \text{Card}(\delta(c))$.

3. LANGAGES PERSISTANTS COMMUTATIFS

On introduit dans cette section une nouvelle classe de langages, contenue dans $\mathscr{L}_p$. Si on ajoute à la persistance l'hypothèse de commutativité, c'est-à-dire que l'évolution ultérieure après l'exécution de $a.b$ ou $b.a$ est identique, on obtient une classe plus riche en propriétés et qui représente un grand nombre de comportements courants. Avant de passer à la définition formelle, on donnera un bref rappel sur la congruence à droite, puis on étudiera les différentes propriétés. Enfin on proposera un algorithme pour caractériser les systèmes de transition dont le langage appartient à cette classe.

3.1. La congruence à droite

La relation de congruence à droite a déjà été introduite dans [2]. C'est une relation d'équivalence qui permet de grouper les mots de $FG(L)$ qui ont les mêmes facteurs droits dans L. Explicitement on a :

DÉFINITION : $\forall f, g \in A^*, f \sim_L g \overset{\text{def}}{\Leftrightarrow} \{h \in A^* / f.h \in L\} = \{h' \in A^* / g.h' \in L\}$.

On écrira '$\sim$' au lieu de '$\sim_L$' quand il n'y a aucune ambiguïté. Il est facile de voir que $\sim$ est une relation d'équivalence, de plus on a les propriétés suivantes :

(i) $f \sim g \Rightarrow f.a \sim g.a$;

(ii) $f \sim g$ et $f \in L \Rightarrow g \in L$.

3.2. Définition

Un langage L est dit *persistant commutatif* ssi $L = FG(L)$ *et* $\forall f \in L$, $\forall a, b \in A$ tels que $a \neq b$ on a l'implication :

$$(f.a \text{ et } f.b \in L) \Rightarrow \begin{cases} f.a.b \quad \text{et} \quad f.b.a \in L, & (1) \\ f.a.b \sim_L f.b.a. & (2) \end{cases}$$

La propriété (1) est celle de la persistance, la (2) est celle de la commutativité.

On note $\mathscr{L}_{pc}$ la classe de langages persistants commutatifs (ou *pc* en abrégé).

REMARQUE : Il est clair que $\mathscr{L}_{pc} \subseteq \mathscr{L}_p$. De plus cette inclusion est stricte, le langage de l'exemple 2.2 en est la preuve : il appartient à $\mathscr{L}_p$ mais pas à $\mathscr{L}_{pc}$

En effet si on prend $f=a$, on a $f.a.b$ et $f.b.a \in FG(L)$, or on a aussi $f.a.b.a.a \in FG(L)$, mais pas $f.b.a.a.a$.

3.3. Exemple

Soit à considérer le système de transition qui gère une file d'attente de taille N. On suppose que les actions « entrée » et « sortie » de la file sont exclusives.

Partant d'une file vide, la seule action possible est de « entrée », puis la file à la possibilité de « entrée » ou de sortie » dans les limites de remplissage.

Le système de transition $\mathscr{S}$ correspondant est le suivant (toutes les configurations sont finales) :

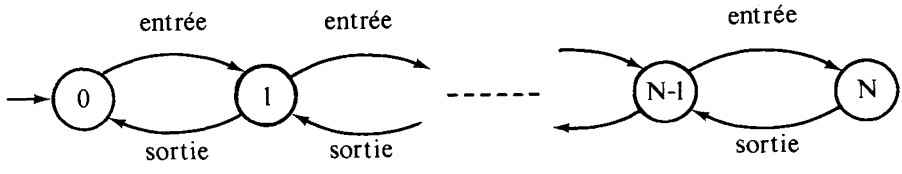

Figure 1

Chaque configuration correspond à un état de remplissage de la file. En prenant s pour « sortie » et e pour « entrée », le langage associé à $\mathscr{S}$ est égal à :

$$L(\mathscr{S}) = FG((e(e(\ldots(e(es)^* s)^* s)^* \ldots)^* s)^*).$$

n fois

Alors $L(\mathscr{S})$ est *pc*.

En effet, on a d'abord $L(\mathscr{S}) = \underset{f}{FG}(L(\mathscr{S}))$. Soit maintenant $f \in L(\mathscr{S})$. On sait qu'il existe $i \in [0, N]$ tel que $c_0 \to c_i$.

On distingue alors deux cas :

(*a*) $i=0$ ou $i=N$.

Dans ce cas, la seule action possible pour c_0 (resp. c_N) est e (resp. s).

(*b*) $0<i<N$.

On a donc $c_{i-1} \overset{s}{\leftarrow} c_i \overset{e}{\rightarrow} c_{i+1}$. Comme $i-1<N$, donc on peut exécuter l'action « entrée » d'où $c_{i-1} \overset{e}{\rightarrow} c_i$ et de même $c_{i+1} \overset{s}{\rightarrow} c_i$.

En conclusion on a démontré que $f.e.s$ et $f.s.e \in L(\mathscr{S})$, mais aussi que :

$$f.e.s \sim_L f.s.e \quad \text{car } c_i \overset{se}{\rightarrow} c_i \quad \text{et} \quad c_i \overset{es}{\rightarrow} c_i.$$

3.4. Confluence de la classe $\mathscr{L}_{pc}$

Les langages de $\mathscr{L}_{pc}$ possèdent la propriété de « confluence » [4], c'est-à-dire que pour tout f, $g \in L$, f et g sont prolongeables en des mots de L qui ont les mêmes lettres, mais dans un ordre différent.

On note $\| f \|$ pour $f \in A^*$, l'image de Parikh de f. On rappelle que si :

$$A=\{a_i\}, \qquad i=1, N, \qquad \text{alors} \quad \| f \| = (|f_{a_1}|, \ldots, |f_{a_N}|)$$

où $|f_{a_i}|$ désigne le nombre d'occurance de la lettre a_i dans f.

Démontrons d'abord le lemme suivant, pour $L \in \mathscr{L}_{pc}$.

LEMME 1 : *Soient* $z, z' \in A^*$, $a \in A$ *tq* $z.a \in L$ *et* $z.z' \in L$ *avec* $a \notin \mathrm{Ops}(z')$ *alors* $z.a.z' \in L$ *et* $z.a.z' \sim_L z.z'.a$.

Preuve : Elle se fait par induction sur la longueur de z'. Pour $|z'|=0$, l'assertion est vraie. Supposons que $z'=z''.t \in A^*.A$. On a $z.a$ et $z.z'' \in L$ avec $a \notin \mathrm{Ops}(z)$. Par hypothèse d'induction $z.a.z'' \in L$ et $z.a.z'' \sim_L z.z''.a$.

Or $z.z''.t=z.z' \in L$ et $z.z''.a$ aussi avec $a \neq t$, donc par hypothèse qu $L \in \mathscr{L}_{pc}$, on a $z.z''.a.t$ et $z.z''.t.a \in L$ avec $z.z''.a.t \sim_L z.z''.t.a$.

Comme $z.a.z'' \sim_L zz''.a$ et que $\sim_L$ est stable à droite, on a aussi $z.a.z''.t \sim_L z.z''.a.t$, d'où $z.a.z'.a \sim_L z.z'.a$.

C.Q.F.D. □

PROPOSITION 3 : (Confluence de Keller). *Soient* $L \in \mathscr{L}_{pc}$, f *et* $g \in L$ *alors il existe* f', $g' \in A^*$ *tq* $f.f' \in L$ *et* :

(i) $f.f' \sim g.g'$;

(ii) $\| f.f' \| = \| g.g'' \|$;

(iii) $\forall a \in A$, $|(ff')_a| = \sup(|f_a|, |g_a|)$.

Preuve : Elle se fait par induction sur la longueur du mot le plus court. On peut toujours supposer que c'est f.

(1) Si $f=\lambda$, il suffit de prendre $f'=g$ et $g'=\lambda$.

(2) Supposons que $f=z.t$ avec $t\in A$. Par hypothèse d'induction, la propriété est supposée vérifiée pour z et g, c'est-à-dire :

$$\exists z'',\quad g''\in A^* \; tq \; z.z''\sim g.g'' \qquad \text{et} \qquad \|z.z''\| = \|g.g''\|,$$

avec $|(zz'')_a| = \sup(|z_a|, |g_a|)$ pour tout a dans A.

On distingue alors deux cas, suivant que $t\in \mathrm{Ops}(z'')$ ou non.

(*a*) $t\notin \mathrm{Ops}(z'')$.

C'est le cas du lemme précédent. On a donc $z.t$ et $z.z''\in L$ avec $t\notin \mathrm{Ops}(z'')$, d'où $z.t.z''\sim z.z''.t$. Comme $z.z''\sim gg''$, on a $z.t.z''\sim z.z''.t\sim gg''.t$. L'égalité $\|z.t.z''\| = \|g.g''.t\|$ est immédiate.

D'autre part, comme $|z''_t|=0$ alors $|z_t| = |g_t|$; donc forcément on a les égalités $|z.t.z'')_t| = |(z.t)_t| = |z_t| + 1 = \sup(|(z.t)_t|, |g_t|)$.

Ainsi $g'=g''.t$ et $f'=z''$ (*fig.* 2) correspondent aux mots cherchés.

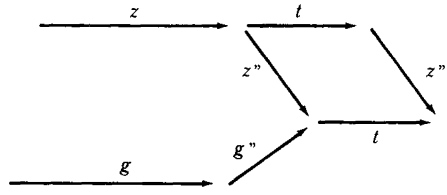

Figure 2

(*b*) $t\in \mathrm{Ops}(z'')$.

Il existe alors u, $v\in A^*$ tq : $z''=u.t.v$ avec $t\notin \mathrm{Ops}(u)$. D'après le lemme 1, on sait que $z.t.u\sim z.u.t$, or ceci implique $z.t.u.v\sim z.u.t.v$.

Il suffit alors de prendre $z'=u.v$ et $g'=g''$ (*fig.* 3).

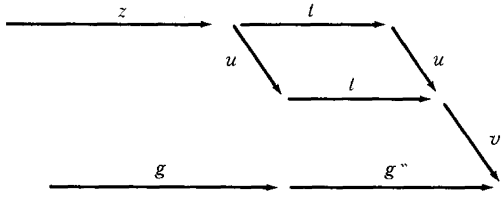

Figure 3

Par hypothèse, on a $|(z.u.t.v)_t| = \sup(|z_t|, |g_t|)$; or on a l'égalité $|(z.u.t.v)_t| = |(z.t.u.v)_t|$ et par conséquent la propriété (iii) est vérifiée.

L'égalité des images de Parikh est évidente.

C.Q.F.D. □

3.5. Opérations dans $\mathscr{L}_{pc}$

On s'intéresse dans cette partie aux opérations pour lesquelles la classe $\mathscr{L}_{pc}$ est stable.

PROPOSITION 4 : *L'intersection d'un nombre quelconque de langages de $\mathscr{L}_{pc}$ est un langage de $\mathscr{L}_{pc}$.*

Preuve : Considérons $L = \bigcap \{L_i / i \in I\}$ où pour tout $i \in I$, $L_i \in \mathscr{L}_{pc}$. Comme $FG(L) \subseteq \bigcap_{i \in I} FG(L_i) = \bigcap_{i \in I} L_i = L$ alors $L = FG(L)$.

Soient $f \in L$, a et $b \in A$ $(a \neq b)$ *tq* $f.a, f.b \in L$. Ainsi, $f.a$ et $f.b \in L_i\, i \in I$. Par persistance $f.a.b$ et $f.b.a \in \bigcap_{i \in I} L_i$. Enfin, la vérification de $f.a.b \sim_L f.b.a$ est facile. □

Ainsi, d'après ce qui précède, il existe un plus petit persistant commutatif contenant un langage donné.

$\mathscr{L}_{pc}$ n'est pas stable pour la réunion, comme le montre le contre-exemple suivant :

Soient : $L_1 = a^*$, $L_2 = b^*$ avec $a, b \in A$.

L_1 et L_2 sont manifestement dans $\mathscr{L}_{pc}$, mais $L_1 \cup L_2 \notin \mathscr{L}_{pc}$ car pour $f = \lambda$ on a $f.a$ et $f.b \in L_1 \cup L_2$ mais pas $f.a.b$.

On rappelle que le produit de deux langages L_1 et L_2 est l'ensemble :

$$L_1 . L_2 = \{x.y / x \in L_1 \text{ et } y \in L_2\}.$$

$\mathscr{L}_{pc}$ n'est pas stable pour le produit, comme le montre l'exemple suivant :

Soient : $L_1 = a^*$, $L_2 = b^*$ avec $a, b \in A$.

L_1 et L_2 sont manifestement dans $\mathscr{L}_{pc}$, mais $L_1 . L_1 \notin \mathscr{L}_{pc}$ car pour $f = \lambda$ on a $f.a$ et $f.b \in L_1 . L_2$ mais pas $f.b.a$.

On donne maintenant une propriété de « propagation », utile pour la représentation des langages de $\mathscr{L}_{pc}$.

PROPOSITION 5 : *Soient $L \in \mathscr{L}_{pc}$, $f \in L$ et $a \in A$ tq $f.a^* \subseteq L$. $\forall g \in L$ $(f \leqq g \Rightarrow g.a^* \subseteq L)$.*

La proposition précédente signifie que la présence d'une « étoile » au milieu du langage L, entraîne une propagation de cette étoile.

Preuve : Nous allons démontrer ceci par récurrence sur $|f^{-1}g|$, où $f^{-1}g$ désigne le facteur droit de f dans g.

(1) $|f^{-1}g|=0$ alors $f=g$ et la proposition est vraie;

(2) $|f^{-1}g|>0$. Posons $u=\sup\{x\in A^*/u\leqq f^{-1}g \text{ et } a\notin \mathrm{Ops}(u)\}$;

(*a*) si $u=\lambda$, ceci implique que :

$$a\leqq f^{-1}g \qquad \text{donc} \quad fa\leqq g \quad \text{et} \quad |(fa)^{-1}g|<|f^{-1}g|.$$

Par hypothèse de récurrence on a alors $ga^*\leqq L$.

(*b*) $u\neq\lambda$. Nous allons maintenant démontrer que $f.a^n.u\sim f.u.a^n \forall n\in N$.

La propriété est vraie à l'ordre 0. Supposons la vraie à l'ordre $n-1$, c'est-à-dire $f.a^{n-1}u\sim f.u.a^{n-1}$. Or $f.a^{n-1}.a\in L$ aussi, donc d'après la proposition 3 (*a*) on a $f.a^{n-1}.a.u\sim f.a^{n-1}.u.a$. Puis grâce à la fermeture à droite $f.a^{n-1}.a.u\sim f.u.a^{n-1}.a$, et ainsi $f.a^*.u\sim f.u.a^*$.

La suite est facile puisque $f.u.a^*\leqq L$ et $f.u\leqq g$ avec $|u|>0$. □

On rappelle que l'opérateur du shuffle classique, noté '⧢', est défini par : Soient f' et $g\in A^*$:

$$g \sqcup\!\sqcup f=\{z\in A^*/z=g_1.f_1\ldots g_nf_n \text{ et } f=f_1.f_2\ldots f_n,\ g=g_1.g_2\ldots g_n\}.$$

On note $L_1 \sqcup\!\sqcup L_2$ l'ensemble :

$$\{f \sqcup\!\sqcup g/f\in L_1 \text{ et } g\in L^2\}.$$

Le « shuffle » de deux langages de $\mathscr{L}_{pc}$ n'est pas toujours un élément de $\mathscr{L}_{pc}$.

En effet, si l'on considère les mots *a.b.c.c.* et *a.b.c.d*, et les langages *pc* $L=FG(a.b.c.c)$ et $L'=FG(a.b.c.d)$, et l'on pose $f=a.b.c$, alors on a $f.c$ et $f.d\in L \sqcup\!\sqcup L'$ mais pas $f.c.d$ ni $f.d.c$.

Par contre il existe une condition suffisante pour que la résultat soit dans $\mathscr{L}_{pc}$.

PROPOSITION 6 : *Soient* L_1 *et* $L_2\in\mathscr{L}_{pc}$ *tq* $\mathrm{Ops}(L_1)\cap\mathrm{Ops}(L_2)=\emptyset$. Alors $L_1 \sqcup\!\sqcup L_2\in\mathscr{L}_{pc}$.

Preuve : Il est facile de vérifier que $L_1 \sqcup\!\sqcup L_2=FG(L_1 \quad L_2)$.

Soient $z\in L_1 \sqcup\!\sqcup L_2$, $a\neq b$ *tq* $z.a$ et $z.b\in L_1 \sqcup\!\sqcup L_2$. Il existe donc deux suites $(f_i)i=1, n$ et $(g_i)i=1, n$ *tq* :

$$f=f_1.f_2\ldots f_n\in L_1, \qquad g=g_1.g_2\ldots g_n\in L_2 \quad \text{et} \quad z=g_1.f_1\ldots g_n.f_n.$$

Deux cas se présentent :

(*a*) $fa, f.b$ (resp. $ga, gb) \in L_1$ (resp. L_2). Par persistance commutative $f.a.b \sim_{L_1} f.b.a.$ (resp. $g.ab \sim_{L_2} gba$). Comme Ops $(L_1) \cap$ Ops $(L_2) = \emptyset$ on a forcément $g_1 f_1 \ldots g_n f_n a.b \sim_{L_1 \sqcup L_2} g_1 f_1 \ldots f_n.b.a.$

(*b*) $f.a \in L_1$ et $g.b \in L_2$ (l'inverse se traite de la même façon). En raison de la définition de $\sqcup$ on a $g_1 f_1 \ldots g_n f_n (ab \cup ba) \subseteq fa \sqcup gb \subseteq L_1 \sqcup L_2$. Puis, la commutativité est facile à démontrer à cause de l'unique décomposition des mots de $L_1 \sqcup L_2$ sur L_1 et L_2. □

L'image d'un élément de $\mathscr{L}_{pc}$ par un morphisme ψ, même alphabétique (i. e. : $\forall a \in A \,|\, \psi(a) | \leqq 1$), n'est pas forcément dans $\mathscr{L}_{pc}$.

Par exemple, prenons le langage L de $\mathscr{L}_{pc}$ reconnu par le système de transition de la figure 4*a*, où toutes les configurations sont finales. Si l'on considère le morphisme défini par $\Psi(a) = \Psi(b) = a$, $\Psi(c) = \Psi(d) = c$, $\Psi(e) = e$ et $\Psi(f) = f$, alors le langage $\Psi(L)$ est reconnu par le système de la figure 4*b* (toutes les configurations sont finales). Les mots $a.c.f$ et $a.c.e$ sont dans $\Psi(L)$ mais le mot $a.c.f.e$ ne l'est pas.

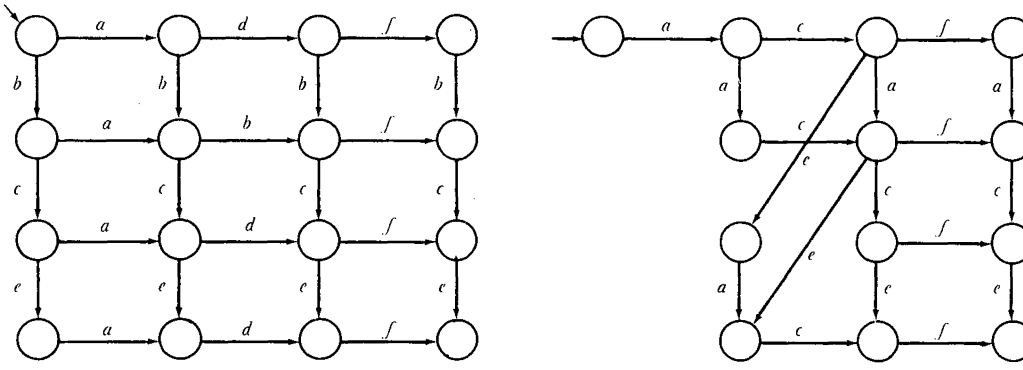

Figure 4

Par contre, $\mathscr{L}_{pc}$ est stable pour l'ensemble des projections.

PROPOSITION 7 : *Soient* $L \in \mathscr{L}_{pc}$ *et* ψ *une projection* (*i. e. :* $\forall a \in A : \psi(a) = a$ ou λ). *Alors* $\psi(L) \in \mathscr{L}_{pc}$.

Preuve : Démontrons d'abord que $\psi(L)$ est fermé par facteur gauche.

Soit $x \in FG(\psi(L))$. Il existe donc $f \in L$ *tq* $x \leqq \psi(f)$. Comme ψ est une projection, il existe $u \in A^*$ *tq* $u \leqq f$ et $x = \psi(u)$. Comme par ailleurs $FG(L) = L$ on a $x \in \psi(L)$; ainsi on a démontré l'inclusion $FG(\psi(L)) \subseteq \psi(L)$. L'inclusion inverse étant évidente, on a alors l'égalité.

Soit à démontrer les propriétés de persistance et de commutativité. On fera la preuve par induction sur $|\ker(\psi)|$ où $\ker(\psi) = \{ a \in A / \psi(a) = \lambda \}$.

(1) $|\mathrm{Ker}(\psi)| = 0$, donc ψ est l'identité et la proposition est vraie.

(2) $|\mathrm{Ker}(\psi)| > 0$. Soient $f \in A^*$, $a, b (a \neq b)$ tq $f.a$ et $f.b \in \psi(L)$. Soit $c \in \mathrm{Ker}(\psi)$ et ψ_c la projection de noyau $\{c\}$. On a alors $\psi = \psi_c \circ \varphi$ où φ est une projection de noyau $\mathrm{Ker}(\psi) - \{c\}$.

Par hypothèse d'induction $\varphi(L)$ est persistant commutatif. Posons $L' = \varphi(L)$.

Il existe x et $y \in A^*$ tq : $\psi_c(x.a) = f.a$ et $\psi_c(y.b) = f.b$ ou aussi $\psi_c(x).a = f.a$ et $\psi_c(y).b = f.b$.

Comme ψ_c est une projection de noyau $\{c\}$, il existe donc deux suites :

$$(c_i)\, i = 0, n \qquad \text{et} \qquad (c'_i)\, i = 0, n \quad \text{avec } c_i \text{ et } c'_i \in c^*$$

et tel que :

$$c_0 f_1 . c_1 f_2 \ldots f_n . c_n . a \in L', \qquad c'_0 . f_1 c'_1 f_2 \ldots f_n . c'_n . b \in L'$$

où :

$$f_1 \ldots f_n = f \qquad \text{et} \qquad f_i \in A, \quad i = 1, n$$

Soit $(u_i)\, i = 0, n$ la suite définie comme suit : $u_i = c_0 f_1 \ldots c_{i-1} . f_i . c_i$.

De même on définit $(u'_i)\, i = 0, n$ par : $u'_i = c'_0 f_i \ldots c'_{i-1} . f_i . c'_i$.

Démontrons maintenant le lemme suivant :

LEMME 2 : $\forall m \leqq n, \exists p, q \in \mathbb{N}$ tq $u_m . c^p \sim_{L'} u'_m c^q$.

Preuve : Elle se fait par induction sur m.

(*a*) Pour $m = 0$, $u_0 = c_0$ et $u'_0 = c'_0$. Il suffit alors de prendre, en supposant que $|c_0| \geqq |c'_0|$, $p = 0$ et $q = |c_0| - |c'_0|$.

(*b*) Par hypothèse d'induction, $u_{m-1} . c^{p'} \sim_{L'} u'_{m-1} . c^{q'}$ pour $m > 1$, p' et $q' \in \mathbb{N}$.

Or $u_{m-1} . f_m \in L'$. Donc d'après la proposition 3 (*a*) et l'équivalence précédente on a $u_{m-1} . f_m . c^{p'} \sim_{L'} u_{m-1} . c^{p'} . f_m$.

De même on a : $u'_{m-1} . f_m c^{q'} \sim_{L'} u'_{m-1} . c^{q'} . f_m$.

Or comme $u_{m-1} . c^{p'} \sim_{L'} u'_{m-1} . c^{q'} \Rightarrow u_{m-1} . f_m . c^{p'} \sim_{L'} u'_{m-1} . f_m . c^{q'}$.

Posons maintenant :

$$M = \sup (p', q', |c_m|, |c'_m|).$$

Alors $p = M - \sup(|c_m|, p')$ et $q = M - \sup(|c'_m|, q')$ répondent à la question. □

D'après le lemme 2, il existe alors :

$$p \text{ et } q \in \mathbb{N}\ tq : \quad u_n . c^p \sim_{L'} u'_n . c^q. \tag{1}$$

Sachant que $u_n . a \in L'$ èt que $a \notin \mathrm{Ops}(c^p)$, on a d'après la proposition 3 (a) : $u_n . c^p . a \in L'$. On démontre de même que $u'_n . c^q . b \in L'$.

D'après (1), on a alors $u_n . c^p . b \in L'$.

Par la pc de L', on a donc $u_n . c^p . a . b$ et $u_n . c^p . b . a \in L'$ avec :

$$u_n . c^p . a . b \sim_{L'} u_n . c^p . b . a. \tag{2}$$

A fortiori $\psi_c(u_n . c^p . a . b) = f_1 . . . f_n . ab = f . a . b \in \psi_c(L') = \psi(L)$ et idem pour $f . b . a$.

Il nous reste à démontrer que $f . a . b \sim_{\psi_c(L')} f . b . a$.

Posons $v = u_n . c^p . a . b$.

Soit $h \in A^*\ tq\ f . a . b . h \in \psi_c(L')$. Il existe alors x et y de $A^*\ tq$:

$$\psi_c(x) = f . a . b \qquad \text{et} \qquad \psi_c(x.y) = f . a . b . h.$$

En appliquant le lemme 2 à x et v, on montre qu'il existe deux entiers r et $s\ tq$:

$$v . c^r \sim_{L'} x . c^s. \tag{3}$$

On démontre par induction sur la longueur de y et à l'aide de la proposition 3, qu'il existe $w \in A^*$ et $k \in \mathbb{N}\ tq$:

$$x . c^s . w \sim_{L'} x . y . c^k \qquad \text{et} \qquad \psi_c(w) = \psi_c(y) = h. \tag{4}$$

D'après (3) et (4) on a $v . c^r . w \in L'$, ce qui implique avec (2) que $u_n . c^p . b . a . c^r . w \in L'$.

Or $\psi_c(u_n . c^p . b . a . c^r . w) = f . b . a . h$, ce qui entraîne $f . b . a . h \in \psi_c(L')$.

C.Q.F.D. □

La conséquence pratique de ce qui précède est que, pour vérifier qu'un langage $\notin \mathscr{L}_{pc}$, alors il suffit d'exhiber une projection sur un alphabet réduit qui ne soit pas dans $\mathscr{L}_{pc}$.

Ainsi on a l'équivalence suivante :

PROPOSITION 8 : *Soient deux langages* L_1, $L_2\ tq\ \mathrm{Ops}(L_1) \cap \mathrm{Ops}(L_2) = \emptyset$ *on a l'équivalence :* $L_1 \quad L_2 \in \mathscr{L}_{pc} \Leftrightarrow L_1$ *et* $L_2 \in \mathscr{L}_{pc}$.

Preuve : Conséquence des propositions 6 et 7. □

3.6. Système de transition persistant commutatif

Un système de transition $\mathscr{S}$ est dit persistant commutatif ssi son langage $L(\mathscr{S}) \in \mathscr{L}_{pc}$.

On s'intéresse dans cette partie à la caractérisation d'un système *pc*. On pourrait toujours appliquer l'algorithme de 2.4 pour déterminer la persistance. Néanmoins, la commutativité présente quelques difficultés à cause de la comparaison des ensembles des facteurs droits.

Pour parer à ces difficultés, on va passer par le système *minimal*.

Dans toute la suite de ce paragraphe, on ne considère que des systèmes de transition *déterministes* et ayant un nombre *fini* de configurations.

3.6.1. *Système de transition minimal*

La minimalité est entendue au sens du nombre de configurations. En effet, il existe une procédure classique [2] qui permet de réduire tout système de transition $\mathscr{S}$ en un système $\mathscr{S}'$ ayant un nombre minimal de configurations et tel que $L(\mathscr{S}) = L(\mathscr{S}')$. La réduction est justement basée sur la relation de congruence $\sim$.

Par exemple le système de la figure 5 donne après réduction le système de la figure 6.

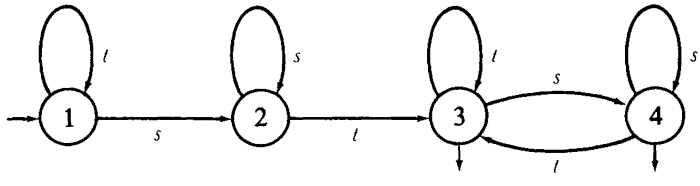

Figure 5

$$D = \{1\}; \qquad C_f = \{3, 4\},$$

Après minimisation on a :

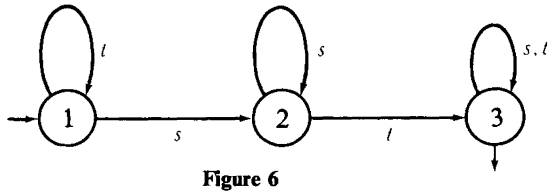

Figure 6

$$D = \{1\}; \qquad C_f = \{3\}.$$

PROPRIÉTÉ : Soit $\mathscr{S}$ un système minimal. Soient f, $g \in FG(L(\mathscr{S}))$ tq $f \sim g$. Alors $\exists c \in C\, tq : D \xrightarrow{f} c$ et $D \xrightarrow{g} c$.

3.6.2. *Décidabilité*

Soit $\mathscr{S}$ un système minimal. On a l'équivalence suivante :

PROPOSITION 9 : *$\mathscr{S}$ est persistant commutatif* $\Leftrightarrow C_f = C$ *et :*

$$\forall c \in C, \quad \forall a, b \in \delta(c), \qquad a \neq b, \quad \exists c' \in C\, tq (c \xrightarrow{ab} c' \text{ et } c \xrightarrow{ba} c').$$

Preuve : C'est une conséquence directe de la propriété du 3.6.1.

En effet, il est facile de voir que l'égalité $C_f = C$ est une condition nécessaire et suffisante pour la fermeture par facteur gauche.

D'autre part, si $\mathscr{S}$ est *pc*, alors $\forall c \in C$, il existe $f \in L(\mathscr{S})$ tq $D \xrightarrow{f} c$. Si a, $b \in A$ $(a \neq b)$ sont tels que $c \xrightarrow{a} c'$ *et* $c \xrightarrow{b} c''$, on a par hypothèse sur $\mathscr{S}$,

$$c \xrightarrow{a} c' \xrightarrow{b} c_1 \qquad \text{et} \qquad c \xrightarrow{b} c'' \xrightarrow{a} c_2.$$

Mais comme $f.a.b \sim f.b.a$ et d'après la propriété du 3.7.1 on a $c_1 = c_2$.

Réciproquement, supposons que les conditions de la partie droite de l'équivalence soient remplies, et soient f, a, b définis comme d'habitude.

Il existe $c \in C\, tq\, D \xrightarrow{f} c$. Donc par hypothèse :

$$\exists c' \in C \ tq \ c \xrightarrow{a.b} c' \qquad \text{et} \qquad c \xrightarrow{b.a} c',$$

ce qui achève la démonstration. □

Remarquons que l'exemple 3.3 répond à la condition de la proposition précédente avec la particularité d'avoir $c = c'$.

3.6.3. *Complexité*

Partant d'un système minimal $\mathscr{S}$, ayant N configurations, la complexité de l'algorithme proposé pour une représentation en file à successeurs est en $O(Nd^3)$ où $d = \max \text{Card } (\delta(c))\, c \in C$. Par contre, la procédure de minimisation d'un système est exponentielle.

L'algorithme énoncé n'est qu'une preuve de décidabilité. Il devrait y avoir une procédure directe et moins complexe pour décider la persistance commutative d'un système. Nous pensons néanmoins que la complexité est de toute façon exponentielle à cause de la commutativité.

4. LANGAGES FAIBLEMENT PERSISTANTS

L'objet de cette section est de montrer que le concept de persistance possède plusieurs niveaux. La notion de « persistance faible » a été introduite pour les réseaux de Petri [9].

DÉFINITION : Un langage L est dit *faiblement persistant* ssi :

$$\forall f \in FG(L), \quad \forall t \in A, \qquad \forall \alpha \in (A - \{t\})^*$$

on a :

$$f.t \quad \text{et} \quad f.\alpha.t \in FG(L) \quad \Rightarrow \quad \exists \beta \in A^* \; tq$$

(i) $\|\alpha\| = \|\beta\|$;

(ii) $f.\alpha.t \sim f.t.\beta$.

La différence entre la faible persistance et la persistance commutative, réside dans le fait que la possibilité d'exécuter t après l'exécution de t' avec laquelle t était en conflit, présente un certain retard.

PROPOSITION 10 [9] : *Soit $L \in \mathscr{L}_{pc}$, alors L est faiblement persistant.*

Preuve : Soient f, α, t définis comme dans la définition. D'après la proposition de confluence on a $f.\alpha.t \sim f.t.\alpha$, donc $\beta = \alpha$ répond à la question. □

La réciproque est fausse comme le montre le contre-exemple suivant : $L = \{a, b\}\, a, b \in A$. L est faiblement persistant mais n'appartient pas à $\mathscr{L}_{pc}$.

La propriété suivante ressemble à la confluence, mais elle est ni nécessaire ni suffisante de confluence.

PROPOSITION 11 [9] : *On a l'équivalence des assertions suivantes :*

(i) *L est faiblement persistant;*

(ii) $\forall \alpha, \beta \in A^*$ *où* $\|\alpha\| \leqq \|\beta\|$, $\forall f \in FG(L)\, tq\, f.\alpha$ *et* $f.\beta \in FG(L)$
$\Rightarrow \exists \gamma \in A^* \, tq \, \|\alpha.\gamma\| = \|\beta\|$ *et* $f.\alpha.\gamma \sim f.\beta$.

Preuve : Elle se fait par induction sur la longueur de α. □

La proposition 11 admet le corollaire immédiat suivant :

COROLLAIRE : *Soit L un langage faiblement persistant, alors pour tout mot* $\alpha, \beta \in FG(L)$ *on a* $\|\alpha\| = \|\beta\| \Rightarrow \alpha \sim \beta$.

5. CONCLUSION

Dans cet article, on a regardé successivement les notions de persistance, persistance commutative et faible persistance. Il n'y a aucune relation entre

la classe des langages persistants et la classe des faiblement persistants sauf que leur intersection contient les persistants commutatifs.

A l'inverse des autres classes, on a vu que $\mathscr{L}_{pc}$ possède plusieurs propriétés intéressantes. En revanche, elle est beaucoup plus difficile à manipuler, le problème de décidabilité étant une preuve. D'autre part, il existe un problème de représentation, car l'hypothèse de commutativité annihile le concept de conflit entre les actions.

Dans cet exposé, on a évité d'aborder le problème de la construction d'un plus petit langage persistant commutatif contenant un langage donné, ainsi que la construction d'un langage de $\mathscr{L}_{pc}$ contenant $L_1 \sqcup\!\sqcup L_2$ où L_1 et $L_2 \in \mathscr{L}_{pc}$ et $\mathrm{Ops}(L_1) \cap \mathrm{Ops}(L_2) \neq \emptyset$.

En réalité, ce problème est d'une complexité nettement supérieure aux propositions de cet article, et dont la résolution fera probablement l'objet de recherche ultérieure.

REMERCIEMENTS

Je tiens, ici, à remercier M. A. Arnold et le Comité de lecture de la R.A.I.R.O. pour leurs nombreuses suggestions et remarques sur cet article.

BIBLIOGRAPHIE

1. A. Arnold et M. Nivat, *Comportements de processus*, Rapport L.I.T.P. 82-12, février 1912.
2. S. Eilenberg, *Automata, Languages and Machines*, vol. A, Academic Press, 1974.
3. R. M. Karp et R. E. Miller, *Parallel Program Schemata*, J. Computer System Science, vol. 3, 1969, p. 147-195.
4. R. M. Keller, *Vector Replacement Systems; a Formalism for Modeling Asynchronous Systems; technical Report* 117, Pinceton Univ., 1972.
5. R. J. Lipton, R. E. Miller et L. Snyder, *Synchronization and Computing Capabilities of Linear Asynchronous Structures*, Proc. 16th Ann. Symp. on F.O.C.S., I.E.E.E. Computer Society, 1975, p. 19-22.
6. E. Mayr, *Persistence of Vector Replacement Systems is Decidable*, Acta Informatica, vol. 15, 1981, p. 309-318.
7. D. E. Muller et M. S. Bantky, *A Theory of Asynchronous Circuits*, Proc. Int. Symp. on Theory of Switching, Cambridge M. A.: Harvard, Univ. Press, 1959, p. 207-243.
8. B. Rosen, *Tree Manipulating Systems and Church Rosser Theorems*, J. Assoc. Comput. Mach., vol. 20, 1973, p. 160-187.
9. H. Yamasaki, *On Weak Persistency of Petri Nets*, Information Process Letters, vol. 3, n° 3, décembre 1981.