Inférence d'ω-langages à partir de préfixes

Colin de la Higuera, Jean-Christophe Janodet

EURISE, Université de Saint-Etienne
{cdlh, janodet}@univ-st-etienne.fr

Résumé

Les automates de Büchi sont utilisés pour reconnaître des langages de mots infinis. Ces langages permettent en particulier de modéliser le comportement de systèmes temps réel, ou les jeux infinis. Le problème d'inférence de ces automates peut se poser à partir de données infinies, ou plus raisonnablement à partir d'exemples finis, préfixes de mots infinis acceptés ou refusés. Nous décrivons les problèmes d'identification à la limite, et d'identification polynomiale à partir de préfixes fixés associés à plusieurs interprétations de ces préfixes, et nous démontrons la difficulté du problème général. Nous étudions ensuite une classe de langages, dits langages sûrs, dont les automates sont polynomialement identifiables à partir de préfixes fixés. Cette classe est maximale au sens où toute surclasse n'est plus identifiable.

Mots-clés : Inférence Grammaticale, mots infinis, ω-langages

1 INTRODUCTION

L'Inférence Grammaticale (Gold, 1978 ; de la Higuera, 1997 ; Parekh & Honavar, 2000) propose un certain nombre de techniques et de résultats théoriques pour traiter le problème général de l'apprentissage automatique de machines (automates ou grammaires) à partir de données structurées, le plus souvent des mots. Parmi les différents objets de la théorie des langages formels les automates finis déterministes ont été les plus étudiés ; certaines classes de grammaires ont également fait l'objet d'investigations. Par contre l'apprentissage des automates de mots infinis est une question peu étudiée.

L'étude de ces automates avait comme objectif premier de résoudre des problèmes de décidabilité issus de la logique mathématique (Büchi, 1960). D'autres travaux ont également montré le lien entre ces automates et la sémantique des logiques modales et temporelles (Vardi & Wolper, 1986). Sur le plan pratique, on les utilise aujourd'hui pour modéliser le comportement des systèmes réactifs critiques. On entend par réactif, tout logiciel en interaction permanente avec son environnement, et par critique, tout logiciel dont une

anomalie de fonctionnement peut avoir des conséquences beaucoup plus importantes que le bénéfice procuré par le service qu'il assure en absence d'anomalie. C'est le cas des pilotes automatiques des avions, des trains ou des centrales nucléaires.

Le développement de tels logiciels requiert des preuves de programmes. On souhaite en particulier qu'ils vérifient des propriétés dites de sûreté, qui expriment que quelque chose de mauvais ne se produira jamais pendant l'exécution du système. Des exemples courants de propriétés de sûreté sont l'exclusion mutuelle ou l'absence de blocage. Ces propriétés se décrivent formellement dans des logiques temporelles comme *PTL*, dont les modèles, des structures de Kripke, peuvent être vus comme des automates de Büchi (Vardi & Wolper, 1986). En conséquence, les automates de Büchi permettent de modéliser à la fois les systèmes critiques, et les propriétés logiques qu'ils doivent vérifier. Ce formalisme unique a permis de développer des algorithmes efficaces de preuve (*model checking*).

Néanmoins, les spécifications formelles des logiciels critiques, et plus encore, leurs propriétés, sont difficiles à écrire pour un non spécialiste des automates et des logiques temporelles. Prenons l'exemple d'un sas à deux portes permettant d'accéder à une chambre forte. On entre dans le sas par la porte 1 et on en sort par la porte 2 (ou vice-versa), mais la porte 2 ne doit pouvoir s'ouvrir que si la porte 1 a été fermée (et vice-versa). Ce système est représenté par l'automate ci-dessous :

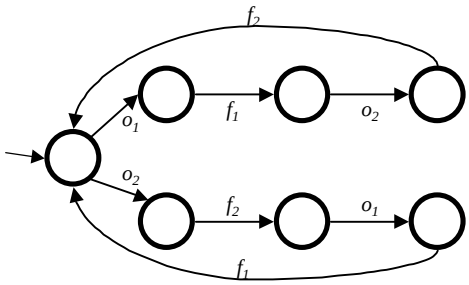

FIG. 1 — *Un sas à deux portes*

La propriété de sûreté « les portes 1 et 2 ne sont jamais ouvertes en même temps » s'écrit, en *PTL* : ❑(non $p_1 \wedge$ non p_2), où la propriété p_i est que "la porte *i* est ouverte".

Si un non spécialiste n'est pas capable de décrire un système et ses propriétés, il est par contre en mesure de donner des exemples de « bons » et de « mauvais » comportements du système. Ces exemples sont des séquences d'événements, comme $o_1\ f_1\ o_2\ f_2\ o_2\ f_2\ o_1\ f_1\ldots$ et $o_2\ f_2\ o_1\ f_1\ldots$, qui sont des « bons » comportements, ou $o_1\ o_2\ldots$ et $o_1\ f_1\ o_1\ldots$, qui sont des « mauvais » comportements. De même pour les propriétés logiques que doit vérifier le système. Notre objectif est donc d'apprendre automatiquement des automates de Büchi en recueillant uniquement des exemples positifs et négatifs.

Le problème de l'apprentissage d'automates de mots infinis pose un premier problème délicat : quelle que soit la façon de récupérer les données d'apprentissage (lot d'exemples, apprentissage en ligne, utilisation d'un oracle ou d'un enseignant), est-il raisonnable d'envisager des données qui seraient des mots infinis ? Rappelons qu'avec un alphabet de taille 2 l'ensemble des mots infinis est non dénombrable. Dans les travaux antérieurs consacrés à l'apprentissage d'automates de mots infinis le choix a été d'utiliser des données provenant du sous ensemble dénombrable des mots ultimement périodiques (de la forme uv^{ω}, u et v étant des mots finis). (Saoudi & Yokomori, 1993) définissent une classe de langages locaux, et montrent l'apprenabilité de ces langages à partir d'exemples positifs seulement ; (Maler & Pnueli, 1991) adaptent l'algorithme L^* d'(Angluin, 1978) et apprennent une classe particulière d'automates avec l'aide d'un nombre polynomial de requêtes d'équivalence et d'appartenance.

Néanmoins, nous souhaitons que l'apprentissage d'un automate se fasse à partir de données expérimentales recueillies auprès des utilisateurs potentiels d'un système. Les données seront donc nécessairement des mots finis. Et l'interprétation de ces mots peut varier. Un mot fini u peut être un préfixe positif, au sens où on saura dire que toutes ses continuations (infinies) sont bonnes, ou qu'une de ses continuations au moins l'est. Le même genre de définition peut exister dans le cas des préfixes négatifs.

Dans cet article nous nous intéressons donc à l'apprentissage de divers types de machines sur les mots infinis, à partir de préfixes. Dans la section 2 nous poserons les différentes définitions concernant les ω-langages, et en section 3 celles nécessaires à la compréhension des problèmes d'apprentissage posés. En section 4 nous établissons différents résultats d'apprenabilité, en montrant que pour la plupart des variantes, l'identification à la limite de la classe des langages ω-réguliers et ω-déterministes n'est pas garantie. Un résultat positif concernant l'identification polynomiale des langages sûrs est donné.

2 DEFINITIONS

2.1 Mots infinis et ω-langages

Nous suivrons les notations de (Thomas, 1990).

Un alphabet est un ensemble fini non vide de symboles. Pour un alphabet donné Σ, l'ensemble de toutes les chaînes finies sur Σ est Σ^*. Un langage L sur Σ est un sous-ensemble de Σ^*. On notera la chaîne vide par λ, les chaînes ou mots par les symboles $u, v, .., z$ et les lettres par a, b, c... $\mathbf{N}$ désignera l'ensemble des entiers positifs ou nuls.

Une suite infinie u (ou mot infini, ou encore ω-mot) sur Σ est une application $\mathbf{N}\rightarrow\Sigma$. Le mot u s'écrit $u(0)u(1)...u(n)...$, avec $u(i)\in\Sigma$. On note Σ^{ω} l'ensemble de tous les ω-mots sur Σ. Un ω-langage sur Σ est un ensemble de mots infinis, et donc un sous-ensemble de Σ^{ω}.

Soient L et K deux langages sur Σ. On définit :

$$L^{\omega}=\{u\in\Sigma^{\omega} / u=u_0u_1... : \forall i\in \mathbf{N}\ u_i\in L\}$$
$$KL^{\omega}=\{u\in\Sigma^{\omega} / u=u_1u_2 : u_1\in K \text{ et } u_2\in L^{\omega}\}$$

Un ω-langage L est ω-régulier *ssi* il existe deux suites finies de langages réguliers A_i et B_i tels que $L=\bigcup_{i=1}^{i=n} A_i B_i^{\omega}$. On note *Pref*(u) l'ensemble de tous les préfixes finis d'un mot infini u, et étant donné un ω-langage L, $Pref(L)=\bigcup_{u\in L} Pref(u)$.

2.2 Les automates de mots infinis

Les automates de Büchi, (1960) reconnaisent des langages de mots infinis. Ces langages permettent en particulier de modéliser le comportement de systèmes réactifs (Vardi & Wolper, 1986) ou les jeux infinis (Thomas, 1990).

Un automate de Büchi est un quintuplet $A=<Q, \Sigma, \delta, F, q_0>$ où Σ est un alphabet, Q est un ensemble fini d'états, avec q_0 l'état initial, δ: $Q\times\Sigma\rightarrow 2^Q$ une fonction de transition, et $F\subseteq Q$ l'ensemble des états marqués.

Le calcul de A sur un mot infini u est une application C_u : $\mathbf{N}\rightarrow Q$ vérifiant :

i. $C_u(0)= q_0$

ii. $\forall i\in \mathbf{N}$, $C_u(i+1)\in\delta(C_u(i), u(i))$

Un mot infini u est accepté par A *ssi* il existe un état de F qui apparaît infiniment souvent dans un calcul pour u. On note $L(A)$ l'ensemble de tous les mots acceptés A. On peut montrer qu'un langage est ω-régulier *ssi* il existe un automate de Büchi qui l'accepte (Thomas, 1990).

On dit qu'un automate est déterministe *ssi* $\forall q\in Q, \forall a\in\Sigma, | \delta(q, a) | \leq 1$.

Notons $\mathbf{Rég}_{\omega}(\Sigma)$, la famille des langages ω-réguliers et $\mathbf{Dét}_{\omega}(\Sigma)$, la famille des langages ω-réguliers acceptés par un automate de Büchi déterministe. Contrairement au cas des automates finis, $\mathbf{Dét}_{\omega}(\Sigma) \subset \mathbf{Rég}_{\omega}(\Sigma)$, mais $\mathbf{Dét}_{\omega}(\Sigma) \neq \mathbf{Rég}_{\omega}(\Sigma)$. En effet, considérons le langage $((a+b)^*a)^{\omega}$ des mots contenant un nombre infini de a. Ce langage est accepté par l'automate déterministe 2a, mais son complémentaire $(a+b)^*b^{\omega}$ n'est pas déterministe. Il est cependant accepté par l'automate de Büchi non déterministe 2b.

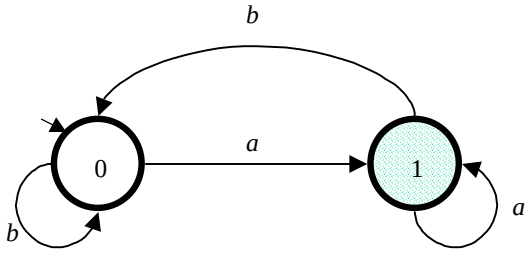

Fig. 2a — ***Automate de Büchi déterministe acceptant le langage ((a+b)*a)ω.***

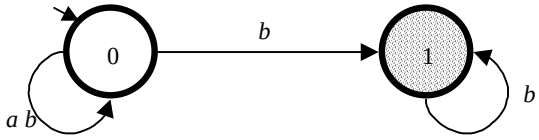

Fig. 2b — *Automate de Büchi non déterministe acceptant le langage* $(a+b)^*b^\omega$.

2.3 Langages ω_{ω}-sûrs et *DB*-machines

Un ω-langage est sûr (Alpern *et al.*, 1985) *ssi*

$\forall w \in \Sigma^\omega$, $(\forall u \in Pref(w), \exists v \in \Sigma^\omega : uv \in L) \Rightarrow w \in L$

c'est-à-dire,

$\forall w \in \Sigma^\omega$, $Pref(w) \subseteq Pref(L) \Rightarrow w \in L$

ou encore,

$\forall w \in \Sigma^\omega$, $w \notin L \Rightarrow (\exists u \in Pref(w) : \forall v \in \Sigma^\omega uv \notin L)$

On note $\mathbf{Sûr}_{\omega_{\omega}}(\Sigma)$ la famille des langages ω-réguliers sûrs.

Le langage b^*a^ω n'est pas un langage sûr : tout préfixe b^n du mot b^ω (qui n'est pas dans le langage) est préfixe d'un ω-mot $b^n a^\omega$ qui est dans le langage. Le langage $b^*a^\omega + b^\omega$ est sûr, lui. Il en découle que $\mathbf{Sûr}_{\omega_{\omega}}(\Sigma) \neq \mathbf{Dét}_{\omega_{\omega}}(\Sigma)$ et nous montrerons (théorème 1) que $\mathbf{Sûr}_{\omega_{\omega}}(\Sigma) \subset \mathbf{Dét}_{\omega_{\omega}}(\Sigma)$.

Une *DB*-machine est un automate de Büchi déterministe où $F=Q$.

Théorème 1. $L \in \mathbf{Sûr}_{\omega_{\omega}}(\Sigma)$ ssi L est accepté par une *DB*-machine.

Pour montrer ce théorème, nous posons les définitions suivantes :

Définition 1.

Soit $P \subseteq \Sigma^*$ et $M=<Q, \Sigma, \delta, F, q_0>$ un automate fini déterministe.

P est un langage préfixe régulier si et seulement si

i. P est régulier ;
ii. Tout préfixe de P est dans P ;
iii. Tout mot de P est le préfixe propre d'un mot de P : $\forall u \in P \; \exists a \in \Sigma : ua \in P$

M est un automate préfixe si et seulement si

i. Tout état est final ;
ii. Tout état est vivant : $\forall q \in Q, \exists a \in \Sigma : \delta(q, a) \in Q$.

Proposition 1.

i. Si L est un langage ω-régulier, $Pref(L)$ est un langage préfixe régulier ;
ii. Si P est un langage préfixe régulier, il existe un automate préfixe qui l'accepte ;
iii. Si $M=<Q, \Sigma, \delta, Q, q_0>$ est un automate préfixe, le langage L reconnu par la *DB*-machine $<Q, \Sigma, \delta, Q, q_0>$ est ω-régulier et $L(M)=Pref(L)$.

Preuve.

Notons que plusieurs langages ω-réguliers sont susceptibles d'avoir le même langage préfixe régulier.

i. L est ω-régulier, il existe donc deux suites finies A_i et B_i de langages réguliers où $L=\bigcup_{i=1}^{i=n} A_i B_i^{\omega}$. Or $Pref(\bigcup_{i=1}^{i=n} A_i B_i^{\omega}) = \bigcup_{i=1}^{i=n} Pref(A_i) \cup A_i B_i^{*} Pref(B_i)$ qui est régulier. Il est clair que ce langage est fermé par préfixe. Soient $P=Pref(L)$ et $u \in P$, il existe donc nécessairement $v \in \Sigma^{\omega}$ tel que $uv \in L$. Soit a la première lettre de v, ua est alors un préfixe de uv, et $ua \in P$.

ii. P est préfixe régulier. Il est donc, par définition, accepté par un automate fini déterministe *A* minimal et non nécessairement complet (autrement dit, l'automate minimal complet auquel on enlève le cas échéant l'état puits). Tout état de cet automate est nécessairement final, sinon, ou cet état n'est pas nécessaire (or l'automate est minimal) ou cet état l'est mais permet d'atteindre un état final. Un mot atteignant cet état serait un préfixe d'un mot de *P*, qui ne serait pas dans le langage, or *P* est préfixe. Soit un état non vivant q de *A*. Soit un mot u tel que $\delta(q_0, u)=q$. Par définition, il existe $a \in \Sigma$ tel que $ua \in P$. Donc $\delta(q, a) \in Q$ et q est vivant.

iii. Soit $A=<Q, \Sigma, \delta, Q, q_0>$ un automate préfixe, considérons la *DB*-machine correspondante $B=<Q, \Sigma, \delta, Q, q_0>$ et montrons que $Pref(L(B))=L(A)$. Soit $u \in Pref(L(B))$. Il existe $w \in \Sigma^{\omega}$ / $uw \in L(B)$. Nécessairement $\delta(q_0, u) \in Q$, et donc $u \in L(A)$. Inversement soit $u \in L(A)$; $\delta(q_0, u)=q$, et comme q est vivant (en itérant) on construit des mots v et w tels que $\delta(q, v)=q'$ et $\delta(q', w)=q'$. Dans ce cas $C_{uvw}\omega$ intersecte infiniment souvent avec l'état q'. $uvw^{\omega} \in L(B)$ et $u \in Pref(L(B))$.

Preuve du théorème 1.

Soit L un langage reconnu par une *DB*-machine $M=<Q, \Sigma, \delta, Q, q_0>$ et soit $w \in \Sigma^{\omega}$. Supposons que tout préfixe u_n de w puisse être prolongé en un mot (de L) reconnu par M. L'application $C_w : \mathbf{N} \rightarrow Q$ telle que $C_w(0)= q_0$ et $\forall i \in \mathbf{N}$, $C_w(i+1)=\delta(C_w(i), u_i(i))=\delta(C_w(i), w(i))$ est un calcul de M sur w. Comme tous les états de M sont marqués, ce calcul réussit, donc $w \in L$. Par conséquent, L est un langage ω-régulier sûr. Réciproquement, d'après la proposition 1, si L est un langage ω-régulier sûr, alors $Pref(L)$ est un langage préfixe reconnu par l'automate préfixe $P=<Q, \Sigma, \delta, Q, q_0>$. Nous prétendons que L est reconnu par la *DB*-machine $M=<Q, \Sigma, \delta, Q, q_0>$. En effet, d'après la proposition précédente, le langage $L(M)$ satisfait $Pref(L(M))= L(P)$ et d'après la première partie de cette preuve, $L(M)$ est un langage sûr (puisque M est une *DB*-machine). Donc L et $L(M)$ sont deux langages sûrs tels que $Pref(L)=Pref(L(M))= L(P)$. Supposons qu'il existe un mot w dans L et pas dans $L(M)$ (ou l'inverse). Comme $Pref(L)=Pref(L(M))$, tous les préfixes de w

sont dans *Pref*(*L*(*M*)), et comme *L*(*M*) est sûr, *w* est dans *L*(*M*), ce qui est impossible. Donc *L*= *L*(M*)*.

Corollaire 1. Si *L*, *L'*∈ $\mathbf{Sûr}_{\omega}(\Sigma)$, alors *Pref*(*L*)=*Pref*(*L'*) ⇔ *L*=*L'*.

Preuve.

Le sens ⇐ ont une conséquence immédiate de la définition de langage sûr. Quant au sens ⇒, c'est une conséquence de la preuve précédente.

3 L'APPRENTISSAGE A PARTIR DE PREFIXES

Une des principales difficultés consiste à expliciter ce que peut signifier : "*p* est un préfixe positif pour un ω-langage *L*", et inversement, "*n* est un préfixe négatif pour un ω-langage *L*". Le sens donné aux préfixes et le cas qui peut sembler intéressant va dépendre du contexte dans lequel est situé notre problème.

Définition 2.

i. *p* est un préfixe ∃-positif *ssi* $\exists u \in \Sigma^{\omega}$, $pu \in L$
ii. *p* est un préfixe ∀-positif *ssi* $\forall u \in \Sigma^{\omega}$, $pu \in L$
iii. *n* est un préfixe ∃-négatif *ssi* $\exists u \in \Sigma^{\omega}$, $nu \notin L$
iv. *n* est un préfixe ∀-négatif *ssi* $\forall u \in \Sigma^{\omega}$, $nu \notin L$

Etant donné un ω-langage *L*, nous noterons $P_{\forall}(L)$ l'ensemble des préfixes ∀-positifs de *L*, $P_{\exists}(L)$ l'ensemble des préfixes ∃-positifs de *L*, $N_{\forall}(L)$ l'ensemble des préfixes ∀-négatifs de *L*, et $N_{\exists}(L)$ l'ensemble des préfixes ∃-négatifs de *L*. Par exemple, pour l'automate de la figure 2a, $L=((a+b)^*a)^{\omega}$, $P_{\forall}(L)=N_{\forall}(L)=\emptyset$ et $P_{\exists}(L)=N_{\exists}(L)=\Sigma^*$.

On remarque aussi que pour tout ω-langage *L*, $P_{\exists}(L)=Pref(L)$ et

$P_{\exists}(L) \cap N_{\forall}(L) = P_{\forall}(L) \cap N_{\exists}(L) = \emptyset$ $\qquad$ $P_{\exists}(L) \cup N_{\forall}(L) = P_{\forall}(L) \cup N_{\exists}(L) = \Sigma^*$

$P_{\forall}(L) = N_{\forall}(\Sigma^{\omega} \setminus L)$ $\qquad$ $P_{\exists}(L) = N_{\exists}(\Sigma^{\omega} \setminus L)$

3.1 Sur les critères de convergence

Nous adaptons ici les définitions de (Gold, 1967) et (de la Higuera, 1997). D'autres paradigmes que l'identification à la limite sont connus, mais sont souvent ou liés à ceux-ci, ou plus difficiles à obtenir. Une comparaison entre différents modèles est proposée dans (Parekh & Honavar, 2000).

3.1.1 Sur les présentations

Nous adaptons les définitions usuelles de présentation de données au cas de l'apprentissage à partir de préfixes. Pour cela il sera nécessaire de considérer systématiquement une classe de langages $\mathcal{L}$ et une classe de représentations associées $\mathcal{R}$. La classe de représentations devra être assez puissante pour représenter tout langage de la classe, à savoir $\forall L \in \mathcal{L}$, $\exists R \in \mathcal{R}$: $L(R)=L$.

La taille d'une représentation (notée $|r|$) est polynomialement liée à la taille de son codage. Dans le cas d'un automate déterministe, la taille de l'alphabet étant constante, le nombre d'états est une mesure satisfaisante.

Définition 3.

Etant donné un ω-langage L et des quantificateurs $p, n \in \{\exists, \forall\}$, une présentation complète (par préfixes) selon (p, n) de L est définie comme une suite infinie de couples $<u_i, étiq(u_i)>$ où :

i. $étiq(u_i) = + \Rightarrow u_i \in P_p(L)$,
ii. $étiq(u_i) = - \Rightarrow u_i \in N_n(L)$,
iii. $m \in P_p(L) \Rightarrow \exists i \in \mathrm{N} : m = u_i$ et $étiq(u_i) = +$, et
iv. $m \in N_n(L) \Rightarrow \exists i \in \mathrm{N} : m = u_i$ et $étiq(u_i) = -$.

3.1.2 Identification à la limite d'ω-langages à partir de préfixes finis

Définition 4.

i. Etant donnée une présentation complète selon (p, n) d'un ω-langage L, et un algorithme d'identification A, on note $H_{A,k}$ l'hypothèse retournée par A après avoir vu les k premiers préfixes.
ii. Une classe d'ω-langages $\mathcal{L}$ est identifiable à la limite selon (p, n) *ssi* il existe un algorithme A qui, pour tout langage $L \in \mathcal{L}$ et toute présentation complète selon (p, n) de L, admet un rang n tel que $\forall k \geq n : H_{A,k} \equiv L$.

3.1.3 Identification à la limite polynomiale par préfixes fixés

Nous adaptons maintenant la définition d'identification polynomiale à partir de données fixées (Gold, 1978; de la Higuera, 1997) au cas de l'apprentissage à partir de préfixes. Cette définition offre l'avantage, par rapport à la précédente de correspondre un peu mieux à l'idée que l'on peut se faire d'un apprentissage possible en pratique ; en particulier par cette définition, les automates finis déterministes sont apprenables alors que les grammaires hors-contexte ne le sont pas.

Définition 5.

Une classe d'ω-langages $\mathcal{L}$ est polynomialement identifiable à la limite par préfixes fixés selon (p, n) pour la classe de représentations $\mathcal{R}$ si et seulement si il existe un algorithme A et deux polynômes $\alpha()$ et $\beta()$ tels que :

i. étant donné un ensemble de préfixes $<S+, S->$, de taille m[1], tel que $S+ \subseteq P_p(L)$ et $S- \subseteq N_n(L)$, A retourne h dans $\mathcal{R}$ consistant avec $<S+, S->$ en temps dans $\mathcal{O}(\alpha(m))$.
ii. pour toute représentation r de taille n d'un langage L de $\mathcal{L}$, il existe un ensemble caractéristique de préfixes $<CS+, CS->$ de taille au plus $\beta(n)$, pour lequel, à partir de $<S+, S->$ avec $CS+ \subseteq S+ \subseteq P_p(L)$ et $CS- \subseteq S- \subseteq N_n(L)$, A retourne une hypothèse h équivalente à r.

---

[1] La taille d'un ensemble S de mots finis est la somme des longueurs de tous les mots de S.

3.2 Les problèmes d'apprentissage d'ω-langages à partir de préfixes.

Nous avons maintenant défini les différents objets du problème. La question générale peut être posée en ces termes :

La classe $\mathcal{L}$ de langages ω-réguliers, représentés par $\mathcal{R}$, peut-elle être apprise selon le critère $\mathcal{C}$ sur des exemples interprétés selon *p*, *n* ?

Les classes $\mathcal{L}$ que nous envisageons sont celles définies dans la section 2.

Les classes de représentations associées seront *B*-Aut (les automates de Büchi) pour $\mathbf{Rég}_{\omega}(\Sigma)$, *DB*-Aut (les automates de Büchi déterministes) pour $\mathbf{Dét}_{\omega}(\Sigma)$ et *DB*-Mach (les *DB*-machines) pour $\mathbf{Sûr}_{\omega}(\Sigma)$.

Les critères sont ceux de l'identification à la limite et l'identification polynomiale à la limite par préfixes fixés.

Les exemples de préfixes positifs et négatifs peuvent être définis selon les différentes combinaisons des quantificateurs ∃ et ∀.

Un problème d'apprentissage sera donc entièrement spécifié par la donnée :

i. de la classe de langages avec la représentation choisie ;
ii. du critère de convergence ;
iii. de la signification des préfixes positifs et négatifs.

Un problème sera ainsi un triplet $<\mathcal{L}_{\mathcal{R}}$, *critère*, *signification*> où *critère* vaudra *idlim* (identification à la limite) ou *polyid* (identification polynomiale à la limite par préfixes fixés) et *signification* sera un couple (*p*, *n*) : *p* et *n* ∈{∃, ∀}.

Exemple.

Le problème $<\mathbf{Sûr}_{\omega}(\Sigma)_{DB\text{-Mach}}$, *idlim*, (∃, ∀)> est celui de l'identification à la limite de la classe $\mathbf{Sûr}_{\omega}(\Sigma)$ où les langages seront représentés par des *DB*-machines sur présentation de préfixes existentiels positifs et de préfixes universels négatifs (voir définition 2). Un tel problème a le statut "positif" si la classe est effectivement apprenable selon le critère voulu, "négatif" si elle ne l'est pas, et "inconnu" si la question n'est pas résolue.

4 RESULTATS

Le nombre de problèmes possibles est trop important pour qu'ils soient tous traités ici. Nous nous concentrons sur deux questions générales : celle de l'existence de réductions entre problèmes, et celle de l'identification à la limite des classes $\mathbf{Rég}_{\omega}(\Sigma)$ et $\mathbf{Dét}_{\omega}(\Sigma)$. Enfin, nous étudions la classe des langages sûrs, pour lesquels l'identification à la limite polynomiale par préfixes fixés est prouvée.

4.1 Propriétés générales

Nous donnons tout d'abord une propriété triviale de réduction entre problèmes. Elle établit que l'identification polynomiale ne peut être obtenue qu'à condition que l'identification à la limite soit assurée : Si $<\mathcal{L}_{\mathcal{R}}$, *idlim*, *sign*> a le statut négatif, alors $<\mathcal{L}_{\mathcal{R}}$, *polyid*, *sign*> aussi.

Une condition nécessaire à l'identification d'une classe de langages est que les langages de la classe soient deux à deux séparables par leurs préfixes :

Lemme 1.

Soit $\mathcal{L}$ une classe d'ω-langages et $\mathcal{R}$ une classe de représentations pour $\mathcal{L}$. S'il existe L_1 et L_2 dans $\mathcal{L}$ tels que $L_1 \neq L_2$, $P_p(L_1)=P_p(L_2)$ et $N_n(L_1)=N_n(L_2)$, alors $<\mathcal{L}_{\mathcal{R}}$, *idlim*, $(p, n)>$ a un statut négatif.

Preuve.

Supposons qu'un algorithme *A* identifie la classe $\mathcal{L}$ et considérons une présentation du langage L_1. Par définition, il existe un rang *k* à partir duquel *A* retourne systématiquement une représentation correcte de L_1. Mais dans ce cas L_2 ne peut pas être identifié.

Théorème 2.

Pour toute classe de représentations $\mathcal{R}$ et pour tout *p* et *n* dans {∃, ∀}, $<\mathbf{Rég}_{\omega}(\Sigma)_{\mathcal{R}}$, *idlim*, $(p, n)>$ et $<\mathbf{Dét}_{\omega}(\Sigma)_{\mathcal{R}}$, *idlim*, $(p, n)>$ ont le statut négatif.

Preuve.

Nous démontrons que ni la classe toute entière des langages ω-réguliers, ni celle des langages déterministes ne sont identifiables à la limite (et donc, de surcroît, qu'elles ne sont pas polynomialement identifiables à partir de préfixes fixés), à l'aide du même contre-exemple, de la figure 3. Les langages acceptés par les automates 3a et 3b sont respectivement $L_1=a^{\omega}+a^*ba^*b(a+b)^{\omega}$ et $L_2=a^*ba^*b(a+b)^{\omega}$. Quelque soit le choix des quantificateurs *p* et *n*, les langages P_p et N_n associés sont identiques dans les deux cas.

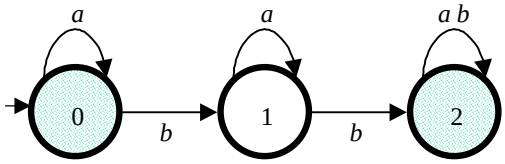

Fig. 3a — *Automate acceptant le langage* $a^{\omega}+a^*ba^*b(a+b)^{\omega}$.

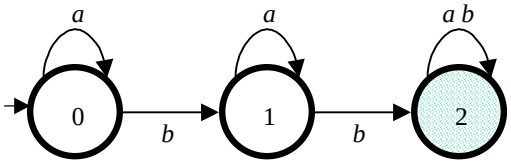

Fig. 3b — *Automate acceptant le langage* $a^*ba^*b(a+b)^{\omega}$.

Formellement :

$P_{\exists}(L_1)=P_{\exists}(L_2)=\Sigma^*$ $\quad N_{\exists}(L_1)=N_{\exists}(L_2)=a^*+a^*ba^*$

$P_{\forall}(L_1)=P_{\forall}(L_2)=a^*ba^*b(a+b)^*$ $\quad N_{\forall}(L_1)=N_{\forall}(L_2)=\varnothing$

4.2 Sur l'identification des langages sûrs

Le résultat précédent est particulièrement négatif, mais guère surprenant. Il est donc nécessaire, pour apprendre, de diminuer la classe de langages envisagée, et/ou de changer de critère d'apprentissage. Il n'est pas raisonnable de choisir un critère moins sévère que l'identification à la limite, c'est donc sur la classe d'ω-langages qu'il importe de travailler.

Théorème 3. < $\mathbf{Sûr}_{\wp\wp}(\Sigma)_{DB\text{-Mach}}$, *polyid*, (∃, ∀)> a le statut positif.

Pour montrer le théorème ci-dessus, une première idée « naïve » consiste à utiliser RPNI (Oncina & Garcia, 1992) sur une paire d'ensembles <*S*+, *S*-> de préfixes du langage cible *L*. Si <*S*+, *S*-> contient un ensemble caractéristique, alors RPNI va retourner un automate préfixe *A* acceptant le langage *Pref*(*L*). Cet automate, vu comme un automate de Büchi, reconnaît *L*, d'après la proposition 1. Par contre, si <*S*+, *S*-> ne contient pas d'ensemble caractéristique, alors RPNI va retourner un automate quelconque, consistant avec <*S*+, *S*->, mais qui n'est pas forcément un automate préfixe. Bien sur, cet automate, vu comme un automate de Büchi, reconnaît un ω-langage *W*, mais en général, *Pref*(*W*) n'est pas consistant avec <*S*+, *S*->. Une alternative est de transformer l'habituel arbre accepteur de préfixes augmenté (de la Higuera *et al.*, 1996) correspondant à <*S*+, *S*-> en un automate préfixe quand RPNI ne retourne pas de solution raisonnable, l'autre est d'adapter RPNI afin qu'il retourne systématiquement un bon automate. C'est ce que nous faisons à travers le lemme suivant que nous montrons à la fin de la section 4.2 :

Lemme 2.

La classe des langages réguliers préfixes, représentés par des automates préfixes, est polynomialement identifiable à la limite à partir de données fixées, par l'algorithme RPNI-préfixes.

Preuve du Théorème 3.

Montrons que les conditions de la définition 5 sont remplies :

i. Soit L un langage sûr. Sur toute paire d'ensembles $<S+, S->$ de préfixes pour L, l'algorithme RPNI-préfixes retourne un automate préfixe acceptant $S+$ et refusant $S-$, en temps polynomial. En temps constant cet automate est transformé en une *DB*-machine M (il suffit de changer le critère d'acceptation). De plus $S+ \subseteq Pref(L(M))$ et $S- \cap Pref(L(M)) = \emptyset$.

ii. Soit L un langage sûr, et soit M une *DB*-machine acceptant L. Soit A l'automate préfixe associé à M. Soit $<CS+, CS->$ l'ensemble caractéristique associé à A pour RPNI-préfixes. Soit maintenant $<S+, S->$ tel que $CS+ \subseteq S+$ et $CS- \subseteq S-$ et $S+ \subseteq L(A)$ et $S- \cap L(A) = \emptyset$. Notons que la taille de $<CS+, CS->$ est polynomiale en celle de A qui est la même que celle de M. Sur $<S+, S->$ RPNI-préfixes retourne un automate A' équivalent à A. Par construction la *DB*-machine M' associée à A' est telle que $Pref(L(M')) = L(A') = L(A) = Pref(L(M))$. Par le corollaire 1 on a $L(M) = L(M')$.

Théorème 4.

Si $\mathcal{L}$ contient strictement $\mathbf{Sûr}_{\omega}(\Sigma)$, $<\mathcal{L}_{DB\text{-Mach}}, idlim, (\exists, \forall)>$ a le statut négatif.

Preuve.

Soit $\mathcal{L}$ une classe contenant strictement $\mathbf{Sûr}_{\omega}(\Sigma)$, et L un langage dans $\mathcal{L}$ mais pas dans $\mathbf{Sûr}_{\omega}(\Sigma)$. $P_{\exists}(L)$ est un langage préfixe. Mais dans ce cas il existe un langage L' dans $\mathbf{Sûr}_{\omega}(\Sigma)$ tels que $P_{\exists}(L) = P_{\exists}(L')$ et $N_{\forall}(L) = N_{\forall}(L')$. Par application du lemme 1, il s'ensuit que $\mathcal{L}$ n'est pas identifiable.

Enfin, le théorème 3 nous permet de déduire un dernier résultat concernant l'apprentissage à partir de préfixes $(\forall, \exists)$. On dit qu'un ω-langage L est co-sûr *ssi* son complémentaire $\Sigma^{\omega} \setminus L$ est un langage sûr. On note $\mathbf{Co\text{-}Sûr}_{\omega}(\Sigma)$ la famille des langages ω-réguliers co-sûrs. Nous avons vu (théorème 1) que les langages sûrs sont acceptés par des *DB*-machines. De même, les langages co-sûrs sont reconnus par des co-*DB*-machines, c'est-à-dire des automates de Büchi complets avec un unique état puits qui est aussi l'unique état marqué.

Théorème 5.

$<\mathbf{Co\text{-}Sûr}_{\omega}(\Sigma)_{co\text{-}DB\text{-Mach}}, polyid, (\forall, \exists)>$ a le statut positif. De plus, pour toute classe $\mathcal{L}$ contenant strictement $\mathbf{Co\text{-}Sûr}_{\omega}(\Sigma)$, $<\mathcal{L}_{co\text{-}DB\text{-Mach}}, idlim, (\forall, \exists)>$ a le statut négatif.

Preuve.

Toute présentation complète par préfixes selon $(\forall, \exists)$ d'un langage co-sûr L est aussi une présentation complète selon $(\exists, \forall)$ du langage sûr $\Sigma^{\omega} \setminus L$. En effet, nous avons vu que $P_{\exists}(\Sigma^{\omega} \setminus L) = N_{\exists}(L)$ et $N_{\forall}(\Sigma^{\omega} \setminus L) = P_{\forall}(L)$. De plus, la construction d'une co-*DB*-machine à partir d'une *DB*-machine se fait en temps linéaire, en la complétant avec un état puits qui devient l'unique état marqué. D'après le théorème 3, le problème $<\mathbf{Sûr}_{\omega}(\Sigma)_{DB\text{-Mach}}, polyid, (\exists, \forall)>$ a

le statut positif, donc le problème <**Co-Sûr**$_{\omega}(\Sigma)_{\text{co-}DB\text{-Mach}}$, *polyid*, ($\forall$, $\exists$)> a le statut positif.

Nous terminons cette section par la **preuve du lemme 2**. Nous proposons l'adaptation suivante de l'algorithme RPNI (Oncina & Garcia, 1992), basé sur les notations de (de la Higuera *et al.*, 1996) :

Algorithme RPNI-préfixes

```
Input: S=<S+, S->
Output: un automate préfixe (défini par δ, F+, F-)
(* Initialisations *)
S+←S+∪Pref(S+); n←0; ∀a∈Σ, Testé(q0, a)←∅; F+←∅; F-←∅;
Tant qu'il y a des chaînes non marquées dans S+ ∪ S- faire
       <q, a, q'>←choisir_essai();
       Si Possible(δ(q, a)=q')
       alors δ(q, a)← q';
              Pour tout w non marqué dans S+ faire
                     Si δ(q0, w)=q"
                     alors marquer(w); F+←F+ ∪ {q"};
              Pour tout w non marqué dans S- faire
                     Si δ(q0, w)=q"
                     alors marquer(w); F-←F- ∪ {q"};
       sinon Testé(q, a)←Testé(q, a) ∪ {q'};
              Si|Testé(q,a)|=n+1 (*toute fusion est impossible*)
              alors (* création d'un nouvel état *)
                     n←n+1; Q←Q ∪ {qn}; δ(q, a)← qn;
                     Pour tout w non marqué dans S+ faire
                            Si δ(q0, w)=q"
                            alors marquer(w);F+←F+ ∪ {q"};
                     Pour tout w non marqué dans S- faire
                            Si δ(q0, w)=q"
                            alors marquer(w);F-←F- ∪ {q};
                     ∀a∈Σ, Testé(qn, a) ←∅ ;
Fin Tantque;
(* conversion en un automate préfixe consistant*)
Q←F+;
Pour tout q∈F+ tel que ∀a∈Σ, δ(q, a)∉Q
       choisir w minimal tel que ({u : δ(q0, u)=q}.Pref(w))∩S-=∅;
       Q←Q ∪ {q_i^w :0<i<|w|}; F+←F+ ∪ {q_i^w :0<i<|w|};
       δ(q, w(0))← q_1^w ;
       Pour tout i de 0 à|w| faire δ( q_{i-1}^w , w(i))← q_i^w ;
       Pour tout a dans Σ faire δ( q_{|w|}^w , a)← q_{|w|}^w ;
Fin.
```

`Fonction Choisir_essai:` retourne une transition menant à un état qui n'a pas été fusionnée et un état déjà validé.

`Fonction Possible(δ(q, a)=q'):` retourne `Vrai` si l'ajout à δ de la règle (q, a, q') n'entraîne pas une inconsistance, `Faux` sinon. L'inconsistance se teste sur l'automate courant dans lequel on ajoute la transition $\delta(q, a)=q'$ et peut avoir deux causes :

i. il existe deux chaînes uaw et vw telles que :
$\delta(q_0, u)=q$, $\delta(q_0, v)=q'$, $uaw \in S+$ et $uaw \notin S-$, et $vw \notin S-$ et $vw \in S+$.

ii. un état n'est plus vivant ; un état q est vivant si $\exists w \in \Sigma^{\omega}$ tel que $(\{u : \delta(q_0, u)=q\}.Pref(w)) \cap S- = \emptyset$. Ceci nous assure que l'automate courant (et donc le dernier) est transformable en automate préfixe.

La preuve de l'algorithme RPNI-préfixes ne peut être qu'esquissée ici, les principaux éléments de cette preuve sont :

i. L'algorithme retourne bien un automate préfixe par construction.
ii. Le test possible assure que tout état est vivant et donc que l'automate peut être transformé en automate préfixe.
iii. Le test "q est-il vivant" ne nécessite pas le calcul d'un mot infini : $S-$ est fini.
iv. Dans le cas où l'ensemble d'apprentissage contient un ensemble caractéristique, la transformation est inutile : l'automate obtenu est un AFD, par les propriétés de RPNI le langage est la cible, et donc préfixe.
v. Enfin, l'algorithme est bien polynomial.

Nous renvoyons le lecteur intéressé à (de la Higuera *et al.*, 1996) pour une preuve complète (dans le cas des AFDs, mais adaptable aisément au cas des automates préfixes).

5 CONCLUSION

Ce travail doit être considéré comme un premier travail concernant l'apprentissage ou l'identification d'automates de mots infinis. Un certain nombre de questions ouvertes et de nouvelles directions de recherche peuvent être proposées. Parmi celles-ci nous retenons :

- Le problème <**?**$_?$, *critère*, (∃, ∃)>. Il semble assez facile de démontrer que pour toutes les classes de langages vues ici, le statut sera négatif. Il est pertinent de trouver une classe de langages (sans doute assez restreinte) pour laquelle le statut serait positif.
- L'apprentissage à partir de requêtes d'appartenance sur les préfixes (et des requêtes d'équivalence).
- L'amélioration de l'algorithme d'inférence (RPNI) pour l'apprentissage des langages préfixes. L'algorithme proposé ici a le mérite de fonctionner en temps polynomial. Il n'est cependant ni aisé à implémenter, ni (probablement) performant.

- Enfin, la validation de cet algorithme sur des données réelles (produites par un système), reste à faire. Le type d'automates envisagés a la particularité d'avoir un alphabet assez important, mais peu de transitions sortantes par état. Dans ce contexte la simplification par typage de l'alphabet (Bernard & de la Higuera, 1999) est sans doute une piste à retenir.

REMERCIEMENTS

Les auteurs remercient Maurice Nivat pour leur avoir suggéré le problème, et Hugues Calbrix, pour certains éclaircissements concernant les ω-langages.

BIBLIOGRAPHIE

ALPERN B., DEMERS A.J. & SCHNEIDER F.B. (1985). Defining Liveness. *Information Processing Letters.* 21, p. 181-185.

ANGLUIN D. (1978). On the Complexity of Minimum Inference of Regular Sets. *Information and Control*. 39, p. 337–350.

BERNARD M. & DE LA HIGUERA C. (1999). GIFT : Grammatical Inference For Terms. *Proc. CAP'99*, Palaiseau.

BÜCHI J.R. (1960). On a decision method in restricted second order arithmetic. *Proc. Cong. Logic Method and Philos. Of Sci*., Stanford Univ. Press, California.

GOLD M.E. (1967). Language Identification in the Limit. *Information and Control.* 10-5, p. 447-474.

GOLD M.E. (1978). Complexity of Automaton Identification from Given Data, *Information and Control*. 37, p. 302-320.

DE LA HIGUERA C., ONCINA J. & VIDAL E (1996). Identification of DFA's: Data dependant versus Data-independant algorithms. *Proc. ICGI '96*, LNAI 1147.

DE LA HIGUERA C. (1997). Characteristic Sets for Polynomial Grammatical Inference. *Machine Learning.* 27, p. 125-138.

MALER O. & PNUELI A. (1991). On the learnability of Infinitary Regular Sets. *Proc. 4th COLT*, p. 128-136, Morgan Kauffman, San Mateo.

ONCINA J. & GARCIA P. (1992). Identifying Regular Languages in Polynomial Time, in *Advances in Structural and Syntactic Pattern Recognition*, H. Bunke ed., Series in Machine Perception and Artificial Intelligence. 5, p. 99-108.

PAREKH, R.J. & HONAVAR, V. (2000). On the relationship between Models for Learning in Helpful Environments. *Proc. ICGI 2000*, LNAI, 207-220.

SAOUDI A. & YOKOMORI T. (1993). Learning Local and Recognizable ω-languages and Monadic Logic. *Proc. EUROCOL 93*, J. Shawe-Taylor & M. Anthony eds., Oxford University Press. p. 157-169.

THOMAS W. (1990). Automata on infinite objects, *Handbook of Theoretical Computer Science* (Van Leewen ed.), p. 133-191.

VARDI M.Y. & WOLPER P. (1986). Automata-Theoretic Techniques in Modal Logics of Programs. *Journal of Computer and Systems Science.* 32, p. 183-221.