De l'inférence régulière à l'apprentissage d'automates classifieurs pour la discrimination de séquences

François Coste

Aïda, IRISA
Campus Universitaire de Beaulieu
35042 Rennes Cedex
francois.coste@irisa.fr

RÉSUMÉ. *Nous proposons d'utiliser et d'étendre les techniques de l'inférence régulière à partir d'exemples positifs et négatifs, pour la discrimination de séquences. Nous discernons ainsi deux paradigmes d'apprentissage. D'une part, l'apprentissage d'une fonction d'acceptation par généralisation de l'échantillon positif seul et d'autre part l'apprentissage d'une fonction de classification, intégrant la notion d'indétermination de classification, par généralisation simultanée de l'échantillon positif et négatif. La dernière approche peut être généralisée à plus de deux classes. La représentation alors utilisée nous permet de proposer des techniques efficaces pour assurer l'unicité de la classification des séquences au cours de l'apprentissage.*

ABSTRACT. *We propose to extend the framework of regular inference from positive and negative samples for the task of sequences discrimination. We distinguish learning an acceptance function by generalizing the positive sample between learning a classification function by generalizing also the negative sample simultaneously. This last approach may be generalized to several classes and allows to express uncertainty in the classification. Thanks to the chosen representation, we are then able to propose efficient techniques to insure an unique classification of each sequence during the inference process.*

MOTS-CLÉS : *Inférence grammaticale, discrimination de séquences, automates*

KEYWORDS: *Grammatical inference, sequence discrimination, automata*

1. Introduction

La motivation de cette étude est l'apprentissage par induction pour la discrimination de séquences. On peut définir ce problème comme l'identification, ou au moins l'approximation, d'une fonction de classification inconnue γ, à partir d'un ensemble d'exemples de classifications de la forme $(x,\gamma(x))$, tel que x est la description d'un objet à classer et $\gamma(x)$ est la classe de l'objet. Alors que la plupart des méthodes d'apprentissage utilisent une description des objets x sous la forme d'un vecteur fini de valeurs d'attributs, l'originalité consiste ici à considérer la classification d'objets décrits par une séquence de symboles, ce qui nous permet d'envisager l'apprentissage de concepts intégrant la notion de *séquentialité*. Parmi le grand nombre d'applications concernées par cet apprentissage, on peut par exemple citer des tâches aussi différentes que : reconnaître, étiqueter ou traduire une suite de mots d'un langage, analyser une séquence d'ADN, exécuter ou planifier un enchaînement d'actions, interpréter une séquence d'alarmes, surveiller un électroencéphalogramme, reconnaître une écriture manuscrite ou détecter une intrusion dans un système informatique.

Utiliser une représentation sous forme de séquences de symboles nous permet également d'utiliser les outils de la théorie des langages. Dans ce cadre, une séquence de symboles est alors appelée un *mot*. Un ensemble de mots constitue un *langage* et peut être représenté par un ensemble fini de règles de production appelé *grammaire*. Nous proposons alors d'utiliser et d'étendre les techniques de l'*inférence grammaticale*, dont l'objet est précisément l'apprentissage par induction d'une grammaire à partir de mots appartenant au langage, pour l'apprentissage inductif de classification de séquences. La classe de grammaires considérée dans cette étude est celle des grammaires régulières. Cette classe offre le meilleur compromis entre généralité d'expression et effectivité des calculs. Elle est la plus haute dans la classification de Chomsky qui soit polynomialement identifiable à partir de données fixées [Hig 96]. Pratiquement, cette classe est suffisamment expressive pour décrire des phénomènes non triviaux de la vie réelle tout en restant apprenable en un temps raisonnable à partir d'exemples. L'ensemble des langages pouvant être engendrés par les grammaires régulières sont dits *réguliers* et peuvent être représentés de façon équivalente par les *automates à états finis*. Nous présentons dans la section suivante les travaux et résultats classiques en *inférence régulière* pour l'apprentissage d'automates, avant d'introduire et étudier dans les sections suivantes son extension à l'apprentissage d'automates classifieurs.

2. Inférence d'automates

Nous présentons dans cette section l'approche, classique en inférence régulière, par fusions d'états, suivant les travaux de [DUP 94]. Nous commençons par rappeler les définitions classiques sur les automates.

Définition 1 *Un* automate fini *est un quintuplet* (Q,Σ,δ,q_0,F) *où* Q *est un ensemble fini d'états,* Σ *est un alphabet,* δ *une fonction de transition de* $Q \times \Sigma \rightarrow 2^Q$ *(qui peut être étendue à* $Q \times \Sigma^* \rightarrow 2^Q$*),* q_0 *est l'état initial et* $F \subseteq Q$ *l'ensemble des états d'acceptation, encore appelés états finals.*

Si pour tout état $q \in Q$ et tout $a \in \Sigma$, $\delta(q,a)$ a au plus un élément, on dit que l'automate est *déterministe*. Le langage régulier accepté par un automate fini A est défini par : $L(A) = \{\sigma \in \Sigma^* \ / \ \delta(q_0,\sigma) \cap F \neq \emptyset\}$. Pour tout langage régulier L, il existe un automate déterministe minimal unique, appelé *automate canonique* de L, noté $A(L)$.

Étant donné un ensemble I_+ de mots appartenant au langage à inférer (échantillon d'apprentissage), l'espace de recherche est généralement défini sous l'hypothèse de complétude structurelle :

Définition 2 *[DUP 94] Un échantillon I^+ est* structurellement complet *relativement à un automate A, s'il existe une acceptation de I^+ par A telle que toute transition de A soit exercée et tout état final de A soit utilisé comme état d'acceptation.*

Le plus grand automate pour un échantillon d'apprentissage I_+, sous l'hypothèse de complétude structurelle, s'appelle l'automate canonique maximal ($MCA(I_+)$ [1]). Il est construit en réalisant l'union de l'ensemble des automates acceptant chacun un mot de l'échantillon d'apprentissage. Cet automate réalise un apprentissage par coeur de l'échantillon d'apprentissage. L'espace de recherche sous l'hypothèse de complétude structurelle de l'échantillon d'apprentissage I_+ peut alors être défini comme le treillis formé de l'ensemble des automates obtenus par fusions d'états de $MCA(I_+)$, noté Lat($MCA(I_+)$) [DUP 94]. La fusion d'états consiste à unifier un ensemble d'états d'un automate et permet de généraliser le langage reconnu. L'automate obtenu par fusion de tous les états de $MCA(I_+)$ est appelé *automate universel*, noté UA [2] et accepte tous les mots de Σ^*.

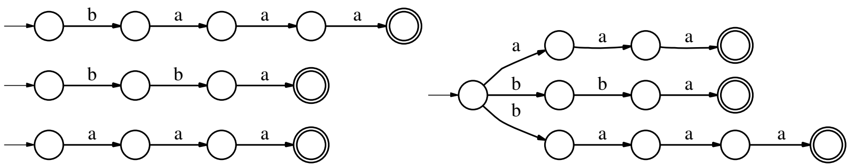


FIG. 1 –. *Automates des mots de I_+ et $MCA(I_+)$ pour $I_+ = \{aaa,bba,baaa\}$*

L'identification à la limite des langages réguliers à partir d'exemples positif seulement étant impossible [GOL 67] à cause du problème de surgénéralisation, plusieurs approches ont été adoptées pour limiter la généralisation : limiter la représentation des langages à une sous-classe des automates, utiliser un critère statistique pour stopper les fusions ou utiliser des contre-exemples.

Nous intéressant à la discrimination de séquences, nous adoptons la dernière démarche et supposons désormais la disponibilité d'un échantillon négatif I_- composé de mots n'appartenant pas au langage à inférer. Un automate est dit *compatible* avec les échantillons d'apprentissage positif I_+ et I_- s'il accepte tous les mots de I_+ et

1. Maximal Canonical Automaton.
2. Universal Automaton

aucun de I_-. L'opération de fusion étant une opération de généralisation du langage reconnu, si un automate accepte un contre exemple, tous les automates dérivés acceptent également ce contre exemple. L'ensemble des automates compatibles est par conséquent le sous ensemble des automates du treillis qui sont plus spécifiques que les automates d'un ensemble frontière, appelé *Border Set* et constitué de l'ensemble des automates compatibles les plus généraux (figure 2).

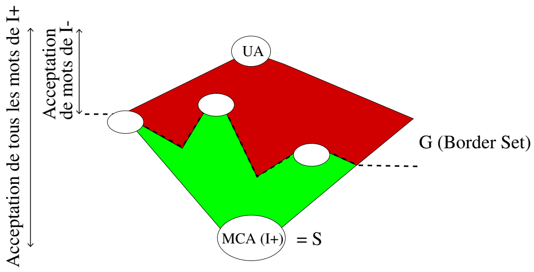


FIG. 2 –. *Espace de recherche sous l'hypothèse de complétude structurelle*

Le principe des algorithmes classiques par fusion d'états consiste à fusionner à chaque pas deux états de $MCA(I_+)$. Si l'automate résultant accepte un exemple négatif, la fusion est défaite et une autre paire d'états est choisie. L'algorithme s'arrête lorsque plus aucune fusion n'est possible, l'automate obtenu appartient alors au *Border Set*. Le prototype de ce type d'algorithmes est RPNI proposé par Oncina et Garcia [ONC 92] et indépendamment par Lang [LAN 92] dans lequel, l'ordre de fusion des états est effectué suivant l'ordre standard. Il permet l'identification à la limite de la classe des langages réguliers [ONC 92, DUP 96] et a été montré expérimentalement efficace pour l'identification approximative des automates déterministes [LAN 92].

3. Inférence d'automates classifieurs

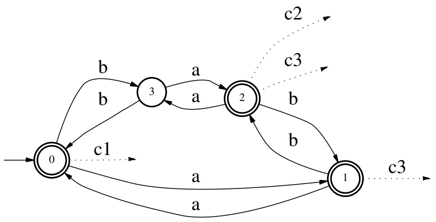
Dans l'approche classique en inférence régulière, on peut remarquer une disymétrie dans l'utilisation de l'échantillon positif et de l'échantillon négatif. Pourtant, les langages réguliers étant clos par complémentation, apprendre un langage revient à apprendre également le langage complémentaire. Les exemples négatifs ajoutés pour assurer l'identification à la limite suivant le théorème de Gold [GOL 78] peuvent ainsi être vus comme un moyen de limiter la surgénéralisation mais aussi comme des exemples d'apprentissage pour le langage complémentaire. Le problème peut alors être considéré comme un problème d'apprentissage d'un langage L_+ et d'un langage L_- à partir d'échantillons de chacun des langages I_+ et I_-.

La première formalisation de l'utilisation symétrique des exemples positifs et négatifs est faite par Alquezar et Sanfeliu par l'introduction des *unbiaised finite state automata* [ALQ 95]. Cependant cette technique a aussi été utilisée par Oncina dans des implémentations de RPNI, pour éviter d'avoir à *parser* les exemples négatifs pour vérifier la compatibilité de l'automate obtenu par une fusion. L'utilisation simultanée des exemples positifs et négatifs est également employée dans les algorithmes avec les heuristiques de fusion d'état "data driven" de [Hi 96] et [LAN 98]. Enfin des travaux sur l'identification de "machines de Moore" [BIE 72] peuvent également être envisagés sous cet angle.

Nous proposons dans cette section d'étudier et formaliser cette approche dans le cadre plus général de l'apprentissage simultané d'un ensemble de langages réguliers à partir d'exemples de chacun des langages, en utilisant une représentation des langages par automate classifieur [COS 99]. Les automates classifieurs constituent une extension des automates permettant de représenter simultanément un ensemble de langages réguliers par l'ajout d'une fonction d'émission aux états finals :

Définition 3 *Un automate classifieur A à k classes d'acceptation (également appelé automate k-classifieur et noté k-FSA), est un sextuplet $(\Sigma,\Gamma,Q,q_0,\delta,\rho)$ où Σ et Γ sont respectivement les alphabets d'entrée et de classification, Γ comportant k symboles appelés* classes*, Q est un* ensemble fini d'états*, q_0 est* l'état initial*, δ est une fonction de $Q \times \Sigma$ dans $\mathcal{P}(Q)$ appelée* fonction de transition*, ρ est une fonction de Q dans $\mathcal{P}(\Gamma)$ appelée* fonction d'émission.

La classification d'un mot de Σ^* par A est donnée par la fonction de classification *γ_A de $Q \times \Sigma^*$ dans $\mathcal{P}(\Gamma)$ définie par : $\gamma_A(q,w) = \bigcup_{q' \in \delta(q,w)} \rho(q')$*



3-FSA classant les mots avec un nombre pair de a et un nombre pair de b dans la classe $c1$, avec un nombre impair de a et un nombre impair de b dans la classe $c2$ et avec un nombre impair de a dans la classe $c3$

FIG. 3 –. *Automate classifieur*

La classification d'une séquence sera dite *indéterminée* si la fonction d'émission n'est pas définie pour le dernier état atteint. Pour la discrimination de séquences, on souhaite que la classification de chaque mot soit unique. On appelera automate classifieur *univoque* tout automate classifieur affectant à chaque mot au plus une classe.

L'inférence d'automates classifieurs univoques constitue l'extension naturelle de l'inférence d'automates compatibles avec un échantillon positif et négatif. Il est alors facile d'étendre les résultats obtenus en inférence régulière pour définir l'espace de recherche des automates classifieurs. Nous supposons alors que l'échantillon d'apprentissage est donné sous la forme $\mathcal{I} = \{I_k\}_{k\in\Gamma}$, où chaque sous échantillon d'apprentissage I_k, $k \in \Gamma$, contient les mots classés dans la classe k de l'échantillon. Nous étendons la définition de la complétude structurelle aux automates classifieurs :

Définition 4 *Un échantillon $\mathcal{I}$ est dit structurellement complet relativement à un k-FSA A s'il existe une acceptation des mots de $\mathcal{I}$ par A telle que toute transition de A est exercée et toute émission de A est exercée.*

Remarquons ici que, sous cette nouvelle définition de la complétude structurelle, tous les mots présents dans l'échantillon d'aprentissage participent à la définition des transitions et des états d'acceptation. Un seul exemple peut ainsi définir une transition utile à la reconnaissance de mots de classes différentes. Le nombre d'exemples nécessaire à la complétude structurelle est alors potentiellement beaucoup plus petit que si l'on avait dû considérer, par exemple, la complétude structurelle de chacun des sous-échantilllons I_k par rapport aux automates canoniques $A(L_k)$ correspondants.

L'espace de recherche sous l' hypothèse de complétude structurelle est alors le treillis des automates classifieurs dérivés de l'arbre canonique maximal $MCA(\mathcal{I})$, construit sur l'ensemble des sous-échantillons d'apprentissage. La figure 4 présente l'espace de recherche correspondant, la généralisation étant ici limitée par la propriété d'univocité au lieu de la compatibilité.

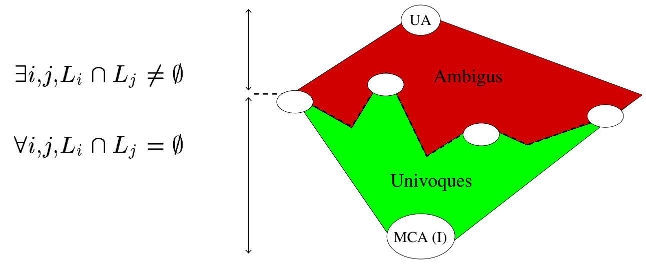


FIG. 4 –. *Espace de recherche des automates classifieurs*

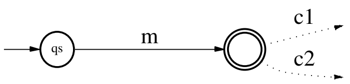
L'inférence d'automates classifieurs peut alors être effectué par extension des algorithmes par fusions développés pour l'inférence régulière classique. Il faut cependant noter le changement de paradigme entre l'apprentissage d'accepteurs de séquences et l'apprentissage de classifieurs de séquences. Dans le dernier cas, les algorithmes peuvent être spécialisés pour prendre en compte l'indétermination de classification. Pour la situation d'apprentissage à partir d'échantillons positifs et négatifs, il faut également évaluer les conséquences du changement d'espace de recherche. Dans le premier cas seul l'échantillon positif est généralisé alors que dans le second la généralisation est effectuée simultanément sur les deux échantillons. Le choix de l'une ou l'autre des approches dépendra donc de la nature des exemples négatifs disponibles.

4. Univocité

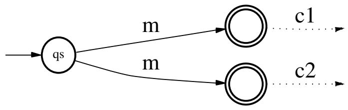
L'utilisation de la représentation par automate classifieur permet de déterminer efficacement si les langages acceptés ont une intersection non nulle, sans avoir de *parsing* à faire. Un tel automate classifieur sera dit *ambigu*, par opposition à univoque. Deux types de situations d'ambiguïté peuvent exister. Un k-FSA est ambigu si :

– La fonction d'émission d'un état retourne au moins deux valeurs (cas a)

– Il existe deux chemins étiquetés par le même mot m permettant d'atteindre deux états d'émission **définie et différente** (cas b).



(cas a)



(cas b)

Nous dirons que deux classifications γ_1 et γ_2 de $\mathcal{P}(\Gamma)$ définies et différentes sont *incompatibles*, noté $\gamma_1 \not\sim \gamma_2$ et défini par $(\gamma_1 \neq \gamma_2) \wedge (\gamma_1 \neq \emptyset) \wedge (\gamma_2 \neq \emptyset)$. Sinon, les classifications seront dites compatibles, noté $\gamma_1 \sim \gamma_2$.

La définition des classifications incompatibles permet de définir les états incompatibles. Deux états q_1 et q_2 sont incompatibles ($q_1 \not\sim q_2$) s'ils permettent d'atteindre par le même mot des états dont les classifications sont incompatibles :

$$q_1 \not\sim q_2 \Leftrightarrow \exists m \in \Sigma^*, \gamma(q_1, m) \not\sim \gamma(q_2, m)$$

On peut alors vérifier qu'un automate classifieur éventuellement non déterministe est ambigu si et seulement si l'un de ses états est incompatible avec lui-même. Nous proposons un algorithme de marquage des états incompatibles (algorithme 1) inspiré de l'algorithme de détection des paires d'états distingables présenté par Hopcroft et Ullman (fig 3.8, p78 [HOP 80]) pour la minimisation d'automates. Les automates considérés ici n'étant pas forcément déterministes, le sens de propagation du marquage est ici inversé par rapport à l'algorithme original pour éviter l'utilisation de listes et avoir une complexité en $O(n^2)$ pour les automates à structure d'arbre (comme $MCA(\mathcal{I})$).

Algorithme 1 États incompatibles et ambiguïté d'un k-FSA.

États_Incompatibles_NDet(A = $(\Sigma, \Gamma, Q, q_0, \delta, \rho)$):
/* Recherche de l'ensemble des états incompatibles de A */
/* Et détection de l'ambiguïté de A */
$\mathcal{E}_I \leftarrow \emptyset$ /* ensemble des états incompatibles */
pour tout $\{q_i, q_j\} \in Q \times Q$, $\rho(q_i) \not\sim \rho(q_j)$ **faire**
    si $\{q_i, q_j\} \notin \mathcal{E}_I$ **alors**
        Marque_Incompatible_Et_Propage_En_Arriere(q_i, q_j)
retourner $\mathcal{E}_I$

Marque_Incompatible_Et_Propage_En_Arriere(q_1, q_2):
/* Détection Ambiguïté */
si $q_1 = q_2$ **alors**
    Retourner Exception("automate ambigu (état q_1)")
/* Mémorisation incompatibilité */
$\mathcal{E}_I \leftarrow \mathcal{E}_I \cup \{q_1, q_2\}$
/* Propagation */
pour tout $a \in \Sigma$ **faire**
    pour tout $\{p_1, p_2\} \in \delta^{-1}(q_1, a) \times \delta^{-1}(q_2, a)$ **faire**
        si $\{p_1, p_2\} \notin \mathcal{E}_I$ **alors**
            Marque_Incompatible_Et_Propage_En_Arriere(p_1, p_2)

La complexité dans le pire des cas de l'algorithme 1 est en $O(|\Sigma|n^4)$: $O(n^2)$ appels à la fonction *Marque_Incompatible_Et_Propage_En_Arriere()*, dont le corps est en $O(|\Sigma|n^2)$. On peut néanmoins affiner ce résultat en fonction du nombre maximal t_a de transitions entrantes en un état par le même symbole. Le corps de *Marque_Incompatible _Et_Propage_En_Arriere()* est alors en $O(|\Sigma|t_a^2)$ et la complexité totale en $O(|\Sigma|t_a^2 n^2)$, ce qui permet de situer plus pratiquement la complexité de l'algorithme entre $O(|\Sigma|n^2)$ et $O(|\Sigma|n^4)$ suivant la valeur de t_a.

L'algorithme 1 permet de détecter si un automate classifieur est ambigu. Il peut être utilisé à chaque pas de l'inférence pour vérifier que l'automate courant est univoque. Cependant, l'ensemble des paires incompatibles $\mathcal{E}_I$ croissant par l'opération de fusion, dans un processus d'inférence par fusions d'états, il est plus économique de n'utiliser cet algorithme que pour l'initialisation de l'ensemble des paires d'états incompatibles sur $MCA(\mathcal{I})$ et de maintenir ensuite de façon incrémentale cet ensemble par l'utilisation de fonctions de propagation locale des conséquences des fusions effectuées. D'autre part, la fusion de toute paire d'états incompatibles produisant un automate ambigu (l'unification des deux états permet d'accéder par le même mot à des classifications incompatibles), le maintien de l'ensemble des paires incompatibles $\mathcal{E}_I$ permet également de disposer d'une information préalable sur les fusions à ne pas faire, qui peut être exploitée par des heuristiques. Nous étudions plus particulièrement dans la section suivante le cas de l'inférence d'automates classifieurs déterministes.

5. Univocité et déterminisme

En inférence grammaticale, de par la définition de l'automate canonique et par application du principe du rasoir d'Occam, le problème généralement considéré est l'inférence du plus petit automate déterministe compatible. Pour les automates classifieurs, la condition de compatibilité est remplacée par celle d'univocité, qui peut être traitée comme dans la section précédente. Nous proposons dans cette section de tirer parti de l'ajout de la condition de déterminisme pour proposer une caractérisation par système de contraintes de l'ensemble des automates classifieurs déterministes univoques.

Pour ne parcourir que les automates déterministes du treillis, un opération de *fusion pour déterminisation* est effectuée après chaque opération classique de fusion. Cette opération consiste à fusionner tous les états atteints par le même symbole à partir d'un même état. Formellement, pour un k-FSA $(\Sigma,\Gamma,Q,q_0,\delta,\rho)$, elle consiste en la procédure suivante :

tant que $\exists q \in Q, \exists a \in \Sigma, \exists q_1, q_2 \in \delta(q,a), q_1 \neq q_2$ **faire** fusionner(q_1,q_2)

La fusion pour déterminisation effectue une déterminisation dans le treillis : l'automate résultant des éventuelles fusions est la plus petite borne supérieure déterministe de l'automate original dans le treillis. Cependant, il faut garder à l'esprit que cet automate est éventuellement plus général que l'automate d'origine et que cette opération ne doit donc pas être confondue avec la déterminisation classique [AHO 72] qui produit une machine déterministe équivalente mais éventuellement hors du treillis.

Le résultat de l'opération de fusion pour déterminisation sur $MCA(\mathcal{I})$ permet d'obtenir un automate appelé *arbre accepteur des préfixes de* $\mathcal{I}$, $(PTA(\mathcal{I})$ [3]$)$. L'arbre accepteur des préfixes est par construction déterministe. Il accepte exactement $\mathcal{I}$ et réalise donc lui aussi un apprentissage par coeur de l'échantillon d'apprentissage. L'espace de recherche des automates classifieurs déterministes sous l'hypothèse de complétude structurelle peut alors être défini comme étant le treillis $Lat(PTA(\mathcal{I}))$ [DUP 94] (figure 5). Notons que cet espace de recherche ne contient pas que des automates déterministes et que la fusion pour déterminisation devra encore être utilisée dans cet espace.

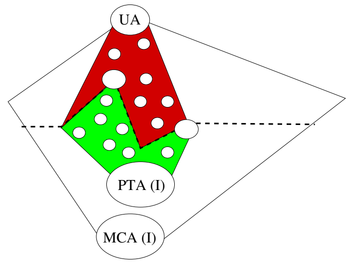


FIG. 5 –. *k-FSA déterministes*

Pour étudier plus particulièrement la dynamique des fusions et des incompatibilités de fusion, nous introduisons un espace de recherche implicite que nous appelons *espace des fusions*. Les attributs considérés sont alors les paires d'états de $PTA(\mathcal{I})$ et peuvent prendre deux valeurs : $\simeq$ si les états sont *fusionnés*, $\not\simeq$ si les états sont *différenciés*, i.e. dans des blocs différents de la partition.

L'espace de recherche peut alors être représenté par un demi tableau supérieur indicé par les états de $PTA(\mathcal{I})$. Nous appelerons *affectation* un choix d'attributs pour des paires d'états. Une affectation sera dite *complète* si une valeur a été attribuée à tous les attributs.

	q_1	q_2	q_3	q_4
q_1	$(\simeq)$	$\not\simeq$	$\simeq$	?
q_2		$(\simeq)$	$\not\simeq$	$\not\simeq$
q_3			$(\simeq)$	?
q_4				$(\simeq)$

FIG. 6 –. *Affectation*

En remarquant que le sens de propagation des fusions pour déterminisation est opposé à celui de la propagation du marquage des états incompatibles, on peut alors exprimer par un système de contraintes dans l'espace des fusions, l'ensemble des automates déterministes univoques du treillis, uniquement en fonction de $PTA(\mathcal{I})$:

Théoreme 1 *Soit $\mathcal{I}$ un échantillon d'apprentissage d'un ensemble de langages réguliers univoques $\mathcal{L}$. L'ensemble des automates classifieurs univoques, pour lesquels $\mathcal{I}$ est structurellement complet, est l'ensemble des automates classifieurs dérivés de $PTA(\mathcal{I}) = (\Sigma,\Gamma,Q,q_0,\delta,\rho)$ suivant les affectations complètes satisfaisant le système de contraintes :*

$$\left\{\begin{array}{ll} \forall q_1,q_2,q_3 \in Q \ : \ q_1 \simeq q_2 \wedge q_2 \simeq q_3 \Rightarrow q_1 \simeq q_3 & (1) \\ \forall q_1,q_2 \in Q \ : \ q_1 \simeq q_2 \Rightarrow \rho(q_1) \sim \rho(q_2) & (2) \\ \forall q_1,q_2,q_1',q_2' \in Q, \forall a \in \Sigma, q_1' \in \delta(q_1,a), q_2' \in \delta(q_2,a) \ : \ q_1 \simeq q_2 \Rightarrow q_1' \simeq q_2' & (3) \end{array}\right.$$

La preuve de ce théorème est détaillée dans [COS 99]. Les relations (1), (2) et (3) expriment respectivement la transitivité des fusions, la compatibilité de classification de deux états fusionnés et la propagation des fusions pour déterminisation. La version contraposée du système de contraintes permet d'étudier les différenciations :

$$\left\{\begin{array}{ll} \forall q_1,q_2,q_3 \in Q \ : \ q_1 \not\simeq q_3 \Rightarrow q_1 \not\simeq q_2 \vee q_2 \not\simeq q_3 & (1') \\ \forall q_1,q_2 \in Q \ : \ \rho(q_1) \not\sim \rho(q_2) \Rightarrow q_1 \not\simeq q_2 & (2') \\ \forall q_1,q_2,q_1',q_2' \in Q, \forall a \in \Sigma, q_1' \in \delta(q_1,a), q_2' \in \delta(q_2,a) \ : \ q_1' \not\simeq q_2' \Rightarrow q_1 \not\simeq q_2 & (3') \end{array}\right.$$

3. Prefix Tree Acceptor

Cette forme met en évidence un ensemble de différenciations initiales pour incompatibilité d'émission par la relation (2') qui peuvent être propagée par *modus ponens* suivant la relation (3'). Cet ensemble de différenciations initial est l'ensemble des états incompatibles $\mathcal{E}_I$ de $PTA(\mathcal{I})$ (voir section 4). Cet ensemble $\mathcal{E}_I$ peut être maintenu à l'aide du système de contraintes en fonction des choix de fusions effectués. Alors que seules les fusions sont généralement considérées dans les algorithmes d'inférence, l'approche présentée ici permet un nouveau type d'algorithme prenant aussi en compte les fusions à ne pas faire (paires d'états *différenciées*). Nous présentons ici le schéma d'énumération complète des solutions (algorithme 2) sur lequel peuvent se greffer les différentes stratégies d'exploration de l'espace pour la recherche de solutions optimales.

Algorithme 2 Schéma d'énumération des solutions

```
EnumSol()
si Affectation Consistante alors
   si Affectation Complète alors
      Solution
   sinon
      Choix d'une paire d'états : q1,q2
      Affecte et propage q1 ≃ q2
      EnumSol()
      Affecte et propage q1 ≄ q2
      EnumSol()
```

Notons enfin que ces algorithmes peuvent aussi être appliqués à l'inférence d'automates classiques à partir d'exemples et de contre exemples. Le choix des fusions et différenciations peut alors être limité aux paires d'états utilisés pour l'acceptation des exemples positifs, tout en considérant la propagation des contraintes sur l'ensemble des états de $PTA(\mathcal{I})$.

6. Conclusion et perspectives

En utilisant et en étendant les techniques de l'inférence régulière à partir d'exemples positifs et négatifs, cette étude permet de discerner deux paradigmes d'apprentissage pour la discrimination de séquences. D'une part, l'apprentissage d'une fonction d'acceptation par généralisation de l'échantillon positif seul et d'autre part l'apprentissage d'une fonction de classification (intégrant la notion d'indétermination de classification) par généralisation simultanée de l'échantillon positif et négatif. Le choix de l'une ou l'autre de ces approches doit être guidé, pour des applications réelles, par l'information disponible sur la nature des contre-exemples et l'intérêt de disposer, ou non, d'une classification des séquences dans la classe négative.

Par ailleurs, pour les deux approches, et quelque soit le nombre de classes, la représentation par automates classifieurs permet de définir formellement la notion d'univocité, grâce à laquelle l'unicité de la classification des séquences pendant le processus d'inférence par fusions peut être assurée efficacement. L'inférence d'automates classifieurs nous paraît donc particulièrement adaptée à l'apprentissage d'automates pour la discrimination de séquences. Des expérimentations effectuées sur des données artificielles, générées aléatoirement, montrent l'existence d'heuristiques efficaces pour la recherche exacte de solutions de taille minimale [COS 00]. Pour cette inférence, il existe également un très bon algorithme glouton basé sur l'heuristique "*Evidence Driven*" [LAN 98]. Disposant à présent d'algorithmes efficaces, il est désormais nécessaire de confronter cette approche à des problèmes d'apprentissage réels pour pouvoir évaluer la qualité de l'inférence effectuée et introduire ce critère dans le processus d'inférence.

7. Bibliographie

[Hi 96] HIGUERA (DE LA) C., ONCINA J., VIDAL E., « Identification of DFA : data-dependent versus data-independant algorithms », *Grammatical inference Learning Syntax from Sentences, ICGI'96*, , 1996, p. 311-325, Springer Verlag.

[AHO 72] AHO A., ULLMAN J., *The Theory of Parsing, Translation and compiling, Vol 1 : Parsing*, Englewood Cliffs, Prentice-Hall, 1972.

[ALQ 95] ALQUÉZAR R., SANFELIU A., « Incremental grammatical inference from positive and negative data using unbiased finite state automata », Shape, Structure and Pattern Recognition*, Proc. Int. Workshop on Structural and Syntactic Pattern Recognition, SSPR'94, Nahariya (Israel)*, 1995, p. 291–300.

[BIE 72] BIERMANN A. W., FELDMANN J. A., « On the synthesis of finite-state machines from samples of their behaviour », *IEEE Transactions on Computeurs C 21*, , 1972, p. 592 - 597.

[COS 99] COSTE F., « State merging inference of finite state classifiers », rapport n° INRIA/RR-3695, 1999, IRISA.

[COS 00] COSTE F., « Apprentissage d'automates classifieurs en inférence grammaticale », PhD thesis, Université de Rennes 1, 2000.

[DUP 94] DUPONT P., MICLET L., VIDAL E., « What is the Search Space of the Regular Inference ? », *ICGI'94, Grammatical inference and Applications*, , 1994, p. 25-37, Springer Verlag.

[DUP 96] DUPONT P., « Utilisation et apprentissage de modèles de langages pour la reconnaissance de la parole continue », PhD thesis, Ecole Nationale Supérieure des Télécommunications, 1996.

[GOL 67] GOLD E. M., « Langage Indentification in the Limit », *Information and control*, vol. 10, n° 5, 1967, p. 447 - 474.

[GOL 78] GOLD E. M., « Complexity of Automaton Identification from given Data », *Information and Control*, vol. 37, 1978, p. 302 - 320.

[Hig 96] HIGUERA (DE LA) C., « Ensembles caractéristiques en inférence grammaticale », *Rapport interne : Lirmm*, , 1996.

[HOP 80] HOPCROFT J., ULLMAN J., *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, N. Reading, MA, 1980.

[LAN 92] LANG K. J., « Random DFA's can be approximately Learned from Sparse Uniform Examples », *5th ACM workshop on Computation Learning Theorie*, , 1992, p. 45 - 52.

[LAN 98] LANG K. J., PEARLMUTTER B. A., PRICE R. A., « Results of the Abbadingo One DFA Learning Competition and a New Evidence-Driven State Merging Algorithm », *Lecture Notes in Computer Science*, vol. 1433, 1998, p. 1–12.

[ONC 92] ONCINA J., GARCIA P., « Inferring Regular Languages in Polynomial Update Time », *Pattern Recognition and Image Analysis*, , 1992, p. 49 - 61.