Inférence d'AFNs : restriction de l'espace de recherche aux automates non ambigus

François Coste , Daniel Fredouille

IRISA-INRIA,
Campus Universitaire de Beaulieu
35042 Rennes Cedex
http://www.irisa.fr/
{francois.coste|daniel.fredouille}@irisa.fr

Résumé

Dans le cadre de l'inférence d'automates non déterministes (AFNs), nous proposons de restreindre la recherche à l'inférence d'automates non ambigus (ANAs). Cette classe d'automate permet en effet, comme les AFNs, de représenter des langages réguliers avec des automates exponentiellement plus petits que ceux de la classe des automates déterministes (AFD). Par rapport aux AFNs, les ANAs peuvent être manipulés plus efficacement, plusieurs opérations non polynomiales sur les AFNs étant polynomiales sur les ANAs. Ce papier est une première étape dans l'inférence d'ANAs : nous décrivons ici l'espace de recherche correspondant à l'inférence d'ANAs par fusion d'états et montrons que celui-ci possède des propriétés prometteuses.

INTRODUCTION

Nous considérons dans cet article l'apprentissage de langage régulier sous forme d'automate à partir d'exemples et de contre exemples. En application du principe du rasoir d'Occam, le problème classique consiste à trouver le plus petit automate fini déterministe (AFD) compatible (i.e. : acceptant tous les exemples et rejetant tous les contre exemples). Ce problème est NP-complet (Gold, 1978). Cependant des algorithmes gloutons d'inférence *par fusions d'états* donnent de bons résultats si suffisamment d'exemples et contre exemples sont présents (Oncina & Garcia, 1992; Lang, 1992; Higuera (de la) *et al.*, 1996).

Bien que plus difficile que l'inférence d'AFD (de la Higuera, 1997), l'inférence d'automates finis non déterministes (AFN) semble intéressante. En effet, il est bien connu qu'il existe des langages tels que leur représentation sous forme d'AFDs demande un nombre exponentiel d'états comparé à une représentation sous forme d'AFNs. Réaliser l'inférence d'AFNs plutôt que d'AFDs peut permettre d'obtenir des solutions plus petites pour lesquelles nous espérons que le nombre d'exemples nécessaire à l'identification du langage cible soit réduit.

Un premier algorithme pour l'inférence d'AFNs nécessitant un oracle a été proposé en 1994 (Yokomori, 1994). Récemment (Denis *et al.*, 2000) ont proposé l'inférence d'une sous classe des AFNs : les AFERs. Cette sous classe a pour intérêt principal de posséder une forme canonique et ainsi d'avoir des propriétés fortes pour l'identification à partir d'exemples. Dans (Coste & Fredouille, 2000) nous avons présenté, dans le cadre des algorithmes par fusion d'états, une méthode de détection efficace de la compatibilité des exemples avec un langage inféré sous forme d'AFNs.

Nous proposons dans cet article d'étendre ce travail en restreignant (à l'aide de méthodes similaires) l'inférence d'AFNs à l'ensemble des AFNs non ambigus (ANAs). Un ANA est un automate pour lequel chaque mot du langage représenté ne possède qu'une acceptation, cette propriété permet de manipuler plus efficacement les ANAs que les AFNs (section 1). Ainsi, l'espace de recherche pour l'inférence d'AFNs (section 2.1) présente, quand il est restreint aux ANAs (section 2.2), des propriétés remarquables très similaires à celles obtenues pour les AFDs. Le parallèle entre ANAs et AFDs peut être poursuivit par la définition d'une opération de parcours de l'espace de recherche, similaire à celle utilisée dans les meilleurs algorithmes actuels d'inférence d'AFDs (section 2.3).

Pour des raisons de concision, les preuves des différents théorèmes présentés ne sont pas détaillées dans cet article, leur version détaillée fait l'objet d'un rapport technique (Coste & Fredouille, 2001).

1 NOTATIONS ET DÉFINITIONS

Soit Σ un alphabet fini, Σ^* est l'ensemble des mots sur Σ. On note ϵ le mot vide et $|u|$ la longueur d'un mot u de Σ^*. On dénotera également $|E|$ le cardinal d'un ensemble E.

Définition 1.1
AFN : *un* automate fini non déterministe *(AFN) est un quintuplet* $A = (\Sigma,Q,Q_0,\delta,F)$ *où* Σ *est l'alphabet d'entrée,* Q *est un ensemble fini d'états,* $Q_0 \subseteq Q$ *est l'ensemble des états initiaux,* δ *est la* fonction de transition *de l'automate définie de* $Q \times \Sigma$ *vers* 2^Q *et F est l'ensemble des états finaux.*

On généralise la fonction δ *aux mots et aux ensembles d'états, soit de* $2^Q \times \Sigma^*$ *vers* 2^Q *par :* $\forall Q' \subseteq Q, w \in \Sigma^*, a \in \Sigma, \delta(Q',a) = \bigcup_{q \in Q'} \delta(q,a)$, $\delta(Q',\epsilon) = Q'$ *et* $\delta(Q',wa) = \bigcup_{q \in Q'} \delta(\delta(q,w),a)$.

Tous les AFNs considérés dans cet article sont *émondés* (i.e. : $\forall q \in Q, \exists w \in \Sigma^*, \delta(q,w) \cap F \neq \emptyset$ et $\forall q \in Q, \exists q_0 \in Q_0, \exists w \in \Sigma^*, q \in \delta(q_0,w)$).

On note $|A| = |Q|$ la taille en nombre d'états d'un automate A. Le langage reconnu par un automate A, noté $L(A)$, est l'ensemble des mots w de Σ^* telles

que $\delta(Q_0,w) \cap F \neq \emptyset$.

Définition 1.2
AFD : *un* automate fini déterministe *(AFD) est un AFN* $A = (\Sigma,Q,Q_0,\delta,F)$ *tel que :* $|Q_0| = 1$ *et* $\forall q \in Q, \forall a \in \Sigma, |\delta(q,a)| \leq 1$.

Définition 1.3
Acceptation : *une* acceptation *pour un mot* $w = a_1 \dots a_{|w|}$ *dans un automate fini* $A = (\Sigma,Q,Q_0,\delta,F)$ *est une séquence de* $|w|+1$ *états* $(q_0,\dots,q_{|w|})$ *telle que* $q_0 \in Q_0$, $q_{|w|} \in F$ *et* $\forall i$, $0 \leq i < |w|-1$, $q_{i+1} \in \delta(q_i,a_{i+1})$. *L'ensemble des acceptations d'un mot* w *dans un automate* A *est noté* $Acc_A(w)$. *Si cet ensemble ne contient qu'une acceptation, on notera celle-ci* $acc_A(w)$.

Définition 1.4
Ambiguité et ANAs : *un automate* A *est dit* ambigu *si il existe au moins un mot* $w \in L(A)$ *pour lequel il existe plusieurs acceptations. Nous noterons* ANA *la classe des AFNs non ambigus*[1].

Notons que tout AFD est un ANA, et tout ANA est un AFN, i.e. : $AFD \subset ANA \subset AFN$. Il est possible de trouver des langages pour lesquels la taille de l'automate nécessaire pour les représenter sous forme d'ANA est exponentielle par rapport à la taille nécessaire pour les représenter sous forme d'AFN. Un saut exponentiel existe également entre ANAs et AFDs (Jiang & Ravikumar, 1993), par exemple les langages $\Sigma^*0\Sigma^n$ sont représentables par des ANAs de taille $n+2$ et des AFDs exponentiels en n.

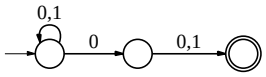

FIG. 1 – *Le langage* $\Sigma^*0\Sigma^n$ *pour* $n = 1$ *représenté par un ANA*

Les ANAs ne possèdent pas de forme canonique et étant donné un ANA, calculer un ANA équivalent de taille minimum est un problème NP-complet (Jiang & Ravikumar, 1993). Par contre, la propriété de non ambiguité permet de manipuler efficacement ces automates : en particulier, savoir si un AFN est un ANA, calculer un ANA reconnaissant l'intersection ou la différence du langage de deux ANAs, savoir si les langages de deux ANAs sont inclus sont des opérations polynomiales (Stearns & Hunt, 1985).

1. La non ambiguité des ANAs concerne ici l'unicité de l'analyse syntaxique, alors que dans les papiers précédents des auteurs (Coste, 2000; Coste & Fredouille, 2000) elle concernait l'unicité de classification. Ces propriétés sont bien sûr liées et peuvent être considérées simultanément. Cependant, nous nous limiterons ici, pour la simplicité de l'exposé, à l'inférence d'automates en ne faisant pas intervenir de notion de classification.

2 INFÉRENCE

2.1 Espace de recherche pour l'inférence d'automates

Nous présentons dans cette section l'espace de recherche pour les algorithmes par fusion d'états suivant (Dupont *et al.*, 1994). Les résultats présentés ici ont été trivialement étendus aux automates à plusieurs états initiaux.

Dans le cadre de l'apprentissage à partir de données fixées (Gold, 1978), nous supposons donné un ensemble d'apprentissage composé d'un *échantillon positif* S_+ d'exemples (i.e : de mots appartenant au langage à inférer) et d'un *échantillon négatif* S_- de contre exemples (i.e : de mots n'appartenant pas au langage à inférer).

L'identification d'un langage à partir d'un échantillon d'apprentissage n'est possible que si l'échantillon est suffisamment "représentatif" du langage à apprendre. Pour l'inférence d'automates, l'échantillon positif S_+ est généralement supposé *structurellement complet* relativement à l'automate cible. Cela signifie intuitivement que toutes les transitions et tous les états finaux et initiaux de l'automate cible sont supposés utilisés pour la reconnaissance de l'échantillon positif.

Cette hypothèse permet de restreindre l'espace de recherche à un treillis fini d'automates noté $Lat(\epsilon MCA(S_+))$. L'élément le plus spécifique de ce treillis, $\epsilon MCA(S_+)$, est l'automate canonique maximal[2] (figure 2). Il est construit en réalisant l'union de l'ensemble des automates acceptant chacun un mot de S_+. Il réalise ainsi un apprentissage par coeur de l'échantillon d'apprentissage.

Le treillis $Lat(\epsilon MCA(S_+))$ est formé de l'ensemble des automates obtenus par des *fusions d'états* de $\epsilon MCA(S_+)$. La fusion d'états consiste à unifier un ensemble d'états et permet de généraliser le langage reconnu. On dira qu'un automate A' est *dérivé* d'un automate A si A' est obtenu par fusion d'états de A. L'automate obtenu par fusion de tous les états du $\epsilon MCA(S_+)$, appelé *automate universel* (UA), accepte tous les mots de Σ^* (figure 2).

Chaque automate du treillis peut être représenté par une partition sur l'ensemble des états du $\epsilon MCA(S_+)$, les éléments de la partition regroupant les états fusionnés. Pour un automate A, on note A/π l'automate dérivé de A par rapport à la partition π. Soit π_1 et π_2 deux partitions sur un automate A, on note $\pi_1 \prec \pi_2$ quand la partition π_2 résulte de l'union de deux blocs de la partition π_1, par extension on note également $A/\pi_1 \prec A/\pi_2$.

Le principe des algorithmes gloutons par fusion d'états consiste à parcourir le treillis en profondeur d'abord, fusionnant à chaque pas deux états de l'automate courant (initialement $\epsilon MCA(S_+)$). 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 est alors un des automates les plus généraux du treillis acceptant tous les exemples et rejetant tous les contre exemples.

2. $\epsilon MCA(S_+)$ constitue l'extension triviale de $MCA(S_+)$ (Dupont *et al.*, 1994) lorsqu'on considère des automates à plusieurs états initiaux.

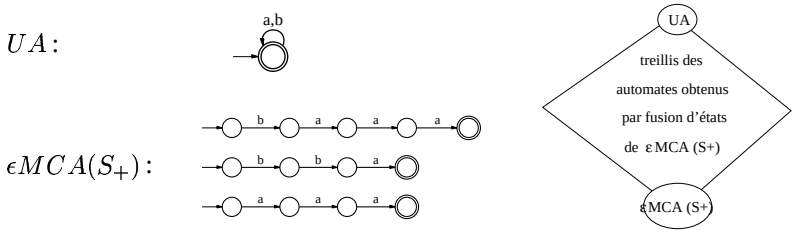

FIG. 2 – *Automate universel (UA), automate canonique maximum ($\epsilon MCA(S_+)$) pour $S_+ = \{aaa, bba, baaa\}$ et espace de recherche sous l'hypothèse de complétude structurelle*

2.2 Restriction de l'espace de recherche aux ANAs

Jusqu'à présent, l'apprentissage de langages réguliers sous forme d'automates non déterministes n'a été réalisé que sur la classe des AFERs ou sur la classe entière des AFNs. Ces automates, bien que potentiellement très compacts, sont difficilement manipulables (beaucoup d'opérations sur ces automates étant exponentielles). Les sont plus facilement manipulables et montrent de plus de bonnes propriétés pour l'apprentissage. En effet, il est toujours possible de trouver un mot dans la différence symétrique du langage de deux ANAs en temps polynomial en fonction du nombre d'états de ces ANAs (Stearns & Hunt, 1985) et les langages réguliers ont été montrés polynomialement prédictibles à partir de requêtes d'équivalence et d'appartenance par rapport au nombre d'états des ANAs (Bergadano & Varricchio, 1996). Ces propriétés et les liens entre les modèles d'apprentissage par oracle et à donnée fixée (Parekh & Honavar, 2000), nous laissent espérer obtenir des résultats d'apprenabilité à données fixées positifs pour les ANAs.

Nous proposons ici d'établir les bases de l'inférence d'ANAs en étudiant l'espace de recherche (section 2.2), ainsi qu'en proposant une opération de parcours de cet espace (section 2.3) similaire à la fusion pour déterminisation utilisée dans les meilleurs algorithmes actuels d'inférence d'AFDs à données fixées.

Théorème 2.1
L'ensemble des ANAs pour lesquels un échantillon positif S_+ est structurellement complet est inclus dans $Lat(\epsilon MCA(S_+))$.
Preuve : *d'après (Dupont* et al.*, 1994) l'ensemble des AFNs pour lesquels un échantillon positif S_+ est structurellement complet est $Lat(\epsilon MCA(S_+))$, les ANAs sont des AFNs, d'où l'inclusion.*□

Suivant ce théorème, nous pouvons considérer comme espace de recherche la restriction aux automates non ambigus de $Lat(\epsilon MCA(S_+))$. Notons que $\epsilon MCA(S_+)$ et l'automate universel sont des ANAs, les bornes de l'espace de recherche restent ainsi inchangées. Cependant, comme pour la restriction de l'espace de recherche aux AFDs, la restriction de l'espace de recherche aux ANAs peut diminuer le nombre de langages identifiables (voir exemples figure 3).

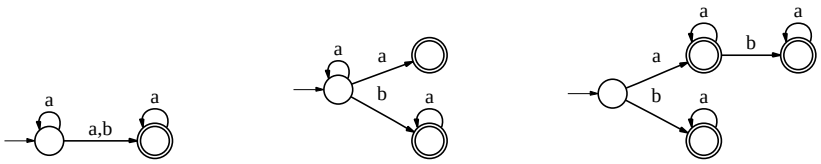

FIG. 3 – *Suivant la représentation choisie, sous l'hypothèse de complétude structurelle, on peut avoir besoin de plus d'exemples pour assurer que le langage est bien représenté dans l'espace de recherche. Cette figure montre un AFN, un ANA et un AFD représentant le même langage : $a^*(a|b)a^*$. L'AFN est dans $Lat(\epsilon MCA(\{aaa,b\}))$, l'ANA est dans $Lat(\epsilon MCA(\{aaa,b,ba\}))$ et l'AFD dans $Lat(\epsilon MCA(\{aaa,b,ba,aba\}))$.*

Nous pouvons alors obtenir sur l'espace de recherche des ANAs des propriétés similaires à celles obtenues pour les AFDs dans (Dupont *et al.*, 1994).

Théorème 2.2
Unicité d'un ANA dans le treillis : *soit A un ANA et S_+ un échantillon d'un langage régulier structurellement complet par rapport à A, alors il n'existe qu'un seul automate isomorphe à A dans $Lat(\epsilon MCA(S_+))$.*
Preuve : *nous savons que A est dans $Lat(\epsilon MCA(S_+))$ (théorème 2.1), donc qu'il existe une partition π telle que $A = \epsilon MCA(S_+)/\pi$. Il reste à montrer qu'elle est unique. Quelque soit un mot $w \in S_+$, $\epsilon MCA(S_+)$ et l'automate A étant des ANAs, il n'existe qu'une seule acceptation $acc_{\epsilon MCA(S_+)}(w)$ et qu'une seule acceptation $acc_A(w)$ dont on peut mettre les éléments en correspondance. Ces couples d'acceptation définissent de façon unique la partition.*□

Ce théorème montre que l'on peut espérer élaguer fortement l'espace de recherche en se limitant aux ANAs. En effet, on a pu constater lors des expériences de (Coste & Fredouille, 2000) que le treillis peut posséder un grand nombre de partitions représentant des AFNs isomorphes [3].

Théorème 2.3
Existence d'un chemin d'ANAs dans le treillis : *Soit S_+ un échantillon d'un langage régulier et soit A un ANA structurellement complet par rapport à S_+, alors il existe une séquence d'ANA $\epsilon MCA(S_+)/\pi_0, \ldots, \epsilon MCA(S_+)/\pi_n$ telle que :*

$$\epsilon MCA(S_+) = \epsilon MCA(S_+)/\pi_0 \prec \epsilon MCA(S_+)/\pi_1 \prec \ldots \prec \epsilon MCA(S_+)/\pi_n = A$$

Preuve : *la preuve consiste à construire une suite de partitions $\pi_{n-1}, \ldots, \pi_0$ par fissions successives à partir de la partition π_n. Ces partitions sont telles que chaque automate $\epsilon MCA(S_+)/\pi_i$ soit non ambigu. Cette suite de partition est construite en deux temps.*

3. Remarquons que le théorème 2.2 ne signifie pas que, pour un langage donné, on aura un seul ANA dans le treillis de taille minimal le reconnaissant (ce qui est d'ailleurs faux). Deux ANAs de même taille non isomorphes peuvent reconnaître le même langage.

Dans un premier temps, on réalise des fissions qui créent des états possédant des langages préfixes disjoints ce qui implique qu'aucune ambiguité n'est créée lors des fissions. Quand ce type de fission n'est plus possible, on montre que l'automate obtenu est en forme d'arbre. Il est ensuite simple d'extraire les branches du $\epsilon MCA(S_+)$ de cet arbre sans créer d'ambiguité. L'algorithme de construction des partitions est présenté dans (Coste & Fredouille, 2001).□

Remarquons que, de même que pour les automates déterministes, le théorème dual du théorème 2.3 qui consisterait à trouver une séquence d'ANAs entre un ANA quelconque de $Lat(\epsilon MCA(S_+))$ et l'automate universel n'est pas vérifié. Par exemple, aucune fusion de deux états sur l'ANA de la figure 4 ne donne un ANA, il n'est donc pas possible d'atteindre celui-ci depuis l'automate universel.

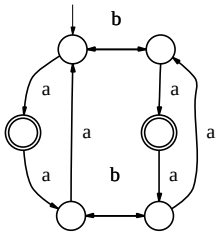

FIG. 4 – *ANA tel qu'aucune fusion de deux états ne donne un autre ANA*

Le théorème 2.3 signifie que l'on peut considérer des algorithmes se limitant aux fusions permettant d'obtenir des ANAs. Nous proposons dans la prochaine section d'adopter une approche similaire à celle ayant été montrée efficace pour l'inférence d'AFDs. Cette approche consiste à fusionner la paire d'états la plus prometteuse puis à effectuer des fusions complémentaires permettant de rester dans la classe d'automates inférée.

2.3 La fusion pour désambiguisation

Nous présentons dans cette section l'opération de *fusion pour désambiguisation* permettant d'obtenir à partir de tout AFN du treillis, un ANA du treillis dont le langage est égal ou contient celui de l'AFN original. Cette opération est l'équivalent pour les ANAs de la *fusion pour déterminisation* de RPNI (Oncina & Garcia, 1992). La fusion pour déterminisation est un outil puissant pour l'inférence d'automates déterministes. Elle est utilisée tant dans des algorithmes aux propriétés théoriques fortes (Oncina & Garcia, 1992), que dans des heuristiques performantes (Lang *et al.*, 1998).

Pour introduire la fusion pour désambiguisation, nous définissons différentes relations entre états d'un automate. Nous dirons que deux états q_1 et q_2 sont en *relation de préfixe commun*, noté $q_1 \parallel^P q_2$ (respectivement de *suffixe commun*, noté $q_1 \parallel_S q_2$) si il existe un mot w commun au langage préfixe [4] (respectivement au langage suffixe [5]) de ces états.

Si deux états sont simultanément en relation de préfixe commun et de suffixe commun, on dira qu'ils sont en relation d'*acceptation parallèle*, noté $q_1 \parallel q_2$.

4. Le langage préfixe d'un état q est l'ensemble des mots w tels que $q \in \delta(Q_0, w)$
5. Le langage suffixe d'un état q est l'ensemble des mots w tels que $\delta(q, w) \cap F \neq \emptyset$

Propriété 2.1
Un automate $A = (\Sigma,Q,Q_0,\delta,F)$ est ambigu ssi deux de ses états q_1 et q_2, $q_1 \neq q_2$ sont en relation d'acceptation parallèle.
Preuve : *A est ambigu ssi il existe un mot w possédant deux acceptations ; de par le mot w, tous les $i^{\text{èmes}}$ états ($0 \leq i \leq |w|$) de ces deux acceptations sont en relations d'acceptation parallèle. Comme les deux acceptations sont différentes, on aura au moins deux états différents en relation d'acceptation parallèle. De même, si on a deux états en relation d'acceptation parallèle, le mot w ayant crée la relation possède deux acceptations.* □

L'ensemble des relations considérées ici peut être calculé pour un automate A donné par des algorithmes similaires à ceux proposés dans (Coste, 2000; Coste & Fredouille, 2000). La complexité du calcul est au pire en $O(|\Sigma||A|^2 t_a^2)$ avec t_a le nombre de transition entrantes ou sortantes maximum d'un état par un symbole donné. Cet ensemble peut ensuite être maintenus de façon incrémentale après chaque fusion effectuée lors de l'inférence [6].

Ce maintient incrémental permet la réalisation efficace de la procédure de *fusion pour désambiguisation* (algorithme 1). Cette procédure consiste à fusionner toutes les paires d'états en relation d'acceptation parallèle. Chaque fusion entraînant possiblement la création de nouvelles relations, on arrêtera de fusionner lorsqu'il n'existera plus deux états différents en relation d'acceptation parallèle. Cette procédure calcule un nombre nécessaire et suffisant de fusions pour obtenir un ANA à partir d'un AFN (propriété 2.2).

Algorithm 1 Procédure de fusion pour désambiguisation

$A = (\Sigma,Q,Q_0,\delta,F)$; $\pi = \{\{q\}|q \in Q\}$
/* Les relations sont considérées sur l'automate A/π */
tant que $\exists B_1, B_2 \in \pi, B_1 \parallel B_2, B_1 \neq B_2$ **faire**
  $(\pi \leftarrow \pi \setminus B_1 \setminus B_2) \cup \{B_1 \cup B_2\}$
$A \leftarrow A/\pi$

Propriété 2.2
Soit A_1 un AFN, et A_2 l'ANA obtenu par fusion pour désambiguisation sur A_1, alors tout ANA de $Lat(A_1)$ est aussi dans $Lat(A_2)$.
Preuve : *On montre que cette propriété est un invariant conservé après chacune des fusions réalisées par la procédure de fusion pour désambiguisation.* □

La fusion pour désambiguisation permet de parcourir l'espace de recherche en limitant la recherche aux ANAs. L'algorithme 2, analogue à RPNI, présente

6. Les auteurs ont déjà présentés dans (Coste & Fredouille, 2000) la relation de préfixe commun et son maintient à chaque fusion. La relation de suffixe commun est présentée ici pour la première fois et peut être maintenue de façon totalement similaire à la relation d'incompatibilité présentée dans le même article. La relation d'acceptation parallèle n'a, quand à elle, pas besoin d'être calculée puisqu'on peut la déduire immédiatement des relations de préfixe commun et de suffixe commun.

un parcours en profondeur d'abord de l'espace de recherche pour l'inférence d'ANAs. La propriété 2.2 garantit que tout ANA du treillis est atteignable par ce type d'algorithme.

Algorithm 2 Inférence par fusion d'états en profondeur d'abord

$A = \epsilon MCA(S_+)$
tant que $choix\_2\_etats(A,q_1,q_2)$ **faire**
  $A' \leftarrow fusion(A,q_1,q_2)$
  $A' \leftarrow fusion\_pour\_desambiguisation(A')$
  si $L(A') \cap S_- = \emptyset$ **alors**
    $A \leftarrow A'$

On remarquera que la fusion pour déterminisation peut être définie comme la fusion de tous les états en relation de préfixe commun. La fusion pour désambiguisation, quand à elle, fusionne tous les états à la fois en relation de préfixe commun et de suffixe commun. La fusion pour désambiguisation n'effectue donc qu'un sous ensemble des fusions nécessitées par la fusion pour déterminisation, elle permet ainsi de parcourir plus finement l'espace de recherche.

CONCLUSIONS ET PERSPECTIVES

Dans le cadre de l'inférence d'AFNs, l'inférence d'ANAs semble être une approche prometteuse. Les ANAs sont manipulables plus facilement que les AFNs et l'espace de recherche pour les ANAs, défini dans ce papier, montre des propriétés similaires à celles de l'espace de recherche pour les AFDs. Nous espérons pouvoir ainsi transposer des techniques efficaces pour l'inférence d'AFDs à l'inférence d'ANAs - notamment en utilisant la fusion pour désambiguisation - tout en bénéficiant de représentations exponentiellement plus compactes pour certains langages.

Sur le plan théorique, l'apprenabilité des ANAs reste à être caractérisée. La difficulté réside dans le fait que, contrairement aux AFERs ou aux AFDs, les ANAs ne possèdent pas de forme canonique. Cela signifie qu'à moins de trouver une forme canonique pour les ANAs, l'identification ne peut porter que sur un langage cible au lieu d'un automate cible. Dans ce cadre, les résultats de (Bergadano & Varricchio, 1996) laissent espérer des résultats d'apprenabilité positifs à partir de données fixées sur les ANAs.

Sur le plan pratique, nous désirons appliquer l'inférence d'ANAs à des données réelles, d'origine biologique. Les ANAs nous semblent plus adaptés que les AFDs pour représenter ce type de données, notamment grâce à la possibilité d'exprimer simplement des concepts comme les "gaps". Les algorithmes de base ayant été implémentés, il faut maintenant trouver des heuristiques adaptées aux ANAs.

Grâce à la fusion pour désambiguisation, il parait ainsi possible d'adapter l'heuristique efficace pour l'inférence d'AFDs de (Lang *et al.*, 1998) à l'inférence d'ANAs.

RÉFÉRENCES

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

BERGADANO F. & VARRICCHIO S. (1996). Learning behaviors of automato from multiplicity and equivalence queries. *SIAM Journal on Computing*.

COSTE F. (2000). De l'inférence régulière à l'apprentissage d'automates classifieurs pour la discrimination de séquences. *CAp-00*.

COSTE F. & FREDOUILLE D. (2000). Efficient ambiguity detection in c-nfa, a step toward inference of non deterministic automata. *ICGI-00*, p. 25–38.

COSTE F. & FREDOUILLE D. (2001). Rapport techique, à paraître, une version de travail est disponible à http://www.irisa.fr/prive/dfredoui/techrep/demo.ps.gz.

DE LA HIGUERA C. (1997). Characteristic sets for polynomial grammatical inference. *Machine Learning*, **27**, 125–138.

DENIS F., LEMAY A. & TERLUTTE A. (2000). Learning regular languages using non deterministic finite automate. *ICGI-00*.

DUPONT P., MICLET L. & VIDAL E. (1994). What is the search space of the regular inference ? *ICGI'94, Grammatical inference and Applications*, p. 25–37. Springer Verlag.

GOLD E. M. (1978). Complexity of automaton identification from given data. *Information and Control*, **37**, 302 – 320.

JIANG T. & RAVIKUMAR B. (1993). Minimal nfa problems are hard. *SIAM Journal on Computing*.

LANG K. J. (1992). Random dfa's can be approximately learned from sparse uniform examples. *5th ACM workshop on Computation Learning Theorie*, p. 45 – 52.

LANG K. J., PEARLMUTTER B. A. & PRICE R. A. (1998). Results of the abbadingo one DFA learning competition and a new evidence-driven state merging algorithm. *Lecture Notes in Computer Science*, **1433**, 1–12.

ONCINA J. & GARCIA P. (1992). Inferring regular languages in polynomial update time. *Pattern Recognition and Image Analysis*, p. 49 – 61.

PAREKH R. & HONAVAR V. (2000). On the relationship between models for learning in helpful environments. *ICGI-00*, p. 207–220.

STEARNS R. & HUNT H. (1985). On the equivalence and containment problems for unambiguous regular expressions, regular grammars and finite automata. *SIAM Journal on Computing*.

YOKOMORI T. (1994). Learning non-deterministic finite automata from queries and counterexamples. *Machine Intelligence*, **13**, 169–189.