Inférence grammaticale régulière : caractérisation de l'ensemble des solutions canoniques

François Coste, Jacques Nicolas

IRISA Campus de Beaulieu
35042 RENNES Cedex
fcoste@irisa.fr, jnicolas@irisa.fr

RÉSUMÉ. *Nous intéressant à l'ensemble des solutions en inférence grammaticale régulière, nous proposons une représentation compacte de l'espace de recherche et une caractérisation dans cet espace des solutions canoniques. Nous pouvons alors reformuler le problème en termes de k-coloriage de graphe, ce qui nous permet de proposer un nouvel algorithme de production des solutions. L'implémentation de cet algorithme et son utilisation sur un jeu de test du domaine permettent une première validation expérimentale de l'approche envisagée.*

MOTS CLÉS : *Inférence grammaticale régulière, espace des versions, automate canonique, k-coloriage de graphe.*

1 Introduction

Au sein de la famille des méthodes d'inférence inductive de séquences [AS83], l'inférence grammaticale est un processus d'apprentissage des règles de composition de suites de lettres à partir d'exemples.

Dans cet article, nous nous intéressons à l'inférence régulière, c'est à dire à l'apprentissage d'une grammaire régulière (sous la forme d'un automate fini A) à partir de deux ensembles de phrases I_+ et I_- (respectivement les échantillons positifs et négatifs d'inférence) tels que les éléments de I_+ puissent être analysés par la grammaire, mais non ceux de I_-. Parmi les propriétés importantes de l'inférence régulière, il est à noter que si l'on ne recherche que des automates pour lesquels I_+ possède la propriété raisonnable de complétude structurelle (cf. définition 7) , l'espace de recherche est un ensemble fini pouvant être partiellement ordonné.

Dans ce cadre, avec I_- non vide, les principales méthodes actuelles consistent à cheminer dans cet ensemble pour trouver la meilleure solution compatible avec I_- : RPNI [OG92] parcours en profondeur d'abord, BRIG [MdG94] parcours en largeur d'abord limitée par un faisceau, GIG [Dup94] progression par mutations et croisements pour optimisation par algorithme génétique. Ces algorithmes ne produisent qu'un seul automate, par des méthodes incomplètes. RPNI peut néanmoins garantir que la solution fournie est effectivement l'automate recherché, moyennant des conditions supplémentaires sur I_+ et I_-.

Notre approche aborde le problème dans le cadre des espaces de versions proposé par Mitchell [Mit82] en considérant *l'ensemble* des automates solutions. Un algorithme de construction de l'ensemble des solutions s'inscrivant dans cette approche a déjà été implémenté, RIG [MdG94], mais il a été abandonné à cause de sa complexité pratique exponentielle au profit d'une version limitée par un faisceau, BRIG [MdG94], plus efficace, mais de nature heuristique. Des travaux préliminaires ont également été effectués pour les grammaires hors contexte [VB87], [Gio93], [Gio95]. La combinatoire du problème provient à la fois de la taille de l'espace de recherche et de la taille de l'ensemble des solutions lui-même. Pour y échapper, plusieurs auteurs proposent de travailler sur une représentation implicite de l'espace des solutions ([Hir92], [Nic93], [Seb94]). Dans cet esprit, nous proposons une méthode de représentation compacte de l'ensemble solution, basée sur l'ensemble des fusions admissibles. Nous restreignant alors comme RPNI à la recherche d'automates canoniques, nous exploitons dans ce cadre les conséquences de l'opération de fusion pour déterminisation.

Nous présentons brièvement dans la prochaine section les notations et résultats classiques en inférence grammaticale, indispensables à la compréhension de l'article. Notre proposition de caractérisation et de constructions des automates solution dans l'espace des fusions est développée §3. Nous terminons par une expérimentation sur un jeu de tests [Dup94] largement employé dans le domaine. Pour la première fois, nous obtenons l'ensemble des solutions canoniques pour les problèmes correspondants, tout en conservant une complexité pratique satisfaisante dans la plupart des cas.

2 Inférence grammaticale régulière

2.1 Automates et langages réguliers

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

Définition 2 *Un* **langage est régulier** *s'il existe un automate fini qui l'accepte. Le langage accepté par un automate fini* A*, est défini par :* $L(A) = \{\sigma \in \Sigma^*, \delta(q_0, \sigma) \cap F \neq$

$\emptyset\}$. *L'acceptation d'une phrase* σ *correspond donc à l'existence d'un chemin dans l'automate entre* q_0 *et un état de* F.

Définition 3 *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**.

Définition 4 *Pour tout langage régulier* L, *il existe un automate déterministe minimal (possédant le moins d'états) unique, appelé* **automate canonique** *de* L, *noté* $A(L)$. *On appelle* **automate universel**, *noté* UA [1], *l'automate acceptant le langage* Σ^*.

2.2 Dérivation d'automates

Définition 5 *Soit* $(Q, \Sigma, \delta, q_0, F)$ *un automate fini et* $\pi = (B_0, B_1, \ldots, B_r)$ *une partition de* Q.
*L'***automate dérivé de** A **pour la partition** π *est* $A' = (\pi, \Sigma, \Delta, B_0, R)$ *où :*

- $q_0 \in B_0$,
- $R = \{B_i \in \pi, \exists q \in B_i \text{ tq } q \in F\}$,
- $[B_j \in \Delta(B_i, a)] \Leftrightarrow [\exists q \in B_i, \exists q' \in B_j \text{ tq } q' \in \delta(q, a)]$.

Nous noterons A/π l'automate dérivé qui résulte de la *fusion des états* de l'automate A, selon une partition π de ses états.

Propriété 1 *[FB75] Si un automate* A/π_j *dérive d'un automate* A/π_i, *alors le langage accepté par* A/π_i *est inclus dans celui accepté par* A/π_j.

Définition 6 *L'ensemble partiellement ordonné des automates dérivés d'un automate* A *est un treillis, que nous désignerons par* $Lat(A)$. *Les automates* A *et* UA *en sont respectivement les éléments nul et universel.*

2.3 Espace de recherche

Définition 7 *[DME94] 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.*
- *tout état final de* A *soit utilisé comme état d'acceptation.*

Cette notion est importante. En effet, sous l'hypothèse que l'on ne recherche que les automates pour lesquels I_+ est structurellement complet, on dispose de résultats forts permettant de limiter l'espace de recherche. Avant de citer les trois théorèmes correspondants, nous rappelons les définitions d'automate canonique maximal et d'arbre accepteur des préfixes.

[1] Universal Automaton.

Définition 8 *A partir d'un échantillon I, on peut construire* **l' automate canonique maximal**, *noté $MCA(I)^2$ qui est l'automate comportant le plus grand nombre d'états tel que $L(MCA) = I$, et I est structurellement complet pour $MCA(I)$.*

Définition 9 **L'arbre accepteur des préfixes** *[Mic79], noté $PTA(I)^3$, est l'automate obtenu en fusionnant les états partageant les mêmes préfixes [4] du $MCA(I)$.*

Théorème 1 *[Mic79][DME94] Soit I_+ un échantillon positif d'un langage régulier L et soit $A(L)$ n'importe quel automate acceptant exactement L. Si I_+ est structurellement complet relativement à $A(L)$ alors $A(L)$ peut être obtenu par dérivation du $MCA(I_+)$.*

Ce qui permet de spécifier l'espace de recherche :

Théorème 2 *[DME94][Dup96] Soit* A *l'ensemble des automates pour lesquels un échantillon positif I_+ est structurellement complet.*
L'ensemble A *est $Lat(MCA(I_+))$.*

Si l'on ne recherche que des automates canoniques, le théorème suivant, d'après [DME94], permet de réduire encore l'espace de recherche.

Théorème 3 *Soit I_+ un échantillon positif et soit Λ l'ensemble des langages réguliers tels que $\forall L \in \Lambda$, I_+ est structurellement complet pour $A(L)$, où $A(L)$ désigne l'automate canonique de L.*
On a alors $\forall L \in \Lambda$, $A(L) \in Lat(PTA(I_+))$.

Pour ne considérer que les automates déterministes de $Lat(PTA(I_+))$, on peut utiliser la procédure de *fusion pour déterminisation*[5] utilisée dans RPNI [OG92] [Dup96] pour parcourir le treillis. Nous notons $f_1 \rightarrow^* f_2$ la relation liée à la fusion pour déterminisation qui signifie que si l'on effectue la fusion f_1, alors, par la procédure de fusion pour déterminisation, on effectue également la fusion f_2. On a $\{q_1, \ldots, q_m\} \rightarrow^* \{q'_1, \ldots, q'_n\}$ si pour tout couple (q'_k, q'_l), il existe un mot w et un couple (q_i, q_j) avec $\delta(q_i, w) = q'_k$ et $\delta(q_j, w) = q'_l$.

3 Espace des fusions

Les résultats de la section précédente montrent que sous l'hypothèse que l'on ne recherche que des automates pour lesquels l'échantillon positif I_+ est structurellement complet, l'inférence régulière peut être considérée comme un problème de recherche d'un automate satisfaisant certains critères dans un treillis.

[2] Maximal Canonical Automaton.
[3] Prefix Tree Acceptor.
[4] u est un préfixe de l'état q si $q \in \delta(q_0, u)$.
[5] à ne pas confondre avec l'algorithme classique de déterminisation d'un AFN qui a pour objet de produire un AFD produisant le même langage [AU72], alors que la fusion pour déterminisation peut produire un sur-langage.

Les automates du treillis pouvant être obtenus par dérivation de l'automate nul du treillis A_0 ($MCA(I_+)$ ou $PTA(I_+)$), le problème de l'inférence régulière peut être reformulé sous forme de recherche de partitions π telles que l'automate A_0/π satisfasse les critères. Nous pouvons reformuler encore une fois le problème, cette fois ci en terme de fusions : quelles sont les fusions à effectuer sur l'automate A_0 pour que l'automate résultant satisfasse les critères ?

Si les critères à satisfaire peuvent être traduits en termes de partition ou de fusion, il peut être plus avantageux de travailler dans l'espace de recherche associé. Nous appelons *espace des fusions*, l'espace de recherche associé aux fusions pour l'inférence régulière (i.e. l'ensemble des fusions d'états possibles sur l'automate nul du treillis A_0). La section 3.1 traite de la représentation de cet espace et des ensembles solution par rapport aux critères les plus couramment employés. Dans la section 3.2, nous nous intéressons plus particulièrement aux relations entre fusion d'états et ensemble des solutions. Dans la dernière section 3.3, nous nous intéressons à la génération des solutions.

3.1 Représentation

3.1.1 Fusions et partitions

Une fusion $f = \{q_1, \ldots, q_r\}$ est caractérisée par l'ensemble des états fusionnés.

Un ensemble de fusions deux à deux disjointes $\{f_0, \ldots, f_p\}$ désigne une partition π de l'ensemble des états, tout état n'appartenant à aucune fusion étant considéré comme un singleton de la partition. On note $\pi = (f_0, \ldots, f_p)$.

En particulier, on peut identifier l'automate résultant d'une fusion f à l'automate dérivé par rapport à la partition π telle que $\pi = (f)$, cet automate sera noté $A_0/(f)$ ou $A_0/(\{q_1, \ldots, q_r\})$.

3.1.2 Ensembles solution

En inférence régulière, deux critères de sélection des automates solution sont principalement employés :

- la compatibilité : les automates solution ne doivent pas accepter d'exemple négatif et doivent accepter tous les exemples positifs ;
- la taille de l'automate en nombre d'états (sa généralité) : on cherche les automates les plus petits possible (ceux qui acceptent les langages les plus généraux).

Du fait de la propriété 1 sur l'inclusion des langages, l'ensemble des automates compatibles avec I_+ et I_- peut être représenté par un ensemble frontière. Si, de plus, le deuxième critère doit être vérifié, alors l'ensemble solution est cet ensemble frontière.

Pour l'espace des fusions, on peut introduire les ensembles solution suivants :

- f_{OK} (resp. f_{NOK}): ensemble des fusions compatibles (resp. incompatibles) avec I_+ et I_- ;
- F_{OK} (resp. F_{NOK}): ensemble frontière de f_{OK} (resp. f_{NOK}).

Plus formellement, on a :

$$f_{OK}(A_0) = \{fusion\ f\ /L(A_0/(f)) \cap I_- = \emptyset\}$$

$$f_{NOK}(A_0) = \{fusion\ f\ /L(A_0/(f)) \cap I_- \neq \emptyset\}$$

$$F_{OK}(A_0) = \{f \in f_{OK}(A_0)\ /\ \nexists f' \in f_{OK}(A_0), f \subset f'\}$$

$$F_{NOK}(A_0) = \{f \in f_{NOK}(A_0)\ /\ \nexists f' \in f_{NOK}(A_0), f' \subset f\}$$

F_{OK} est l'ensemble des plus grandes fusions telles que l'automate dérivé soit compatible avec I_+ et I_- alors que F_{NOK} est l'ensemble des plus petites fusions telles que l'automate dérivé soit incompatible avec I_+ et I_-.

3.2 Caractérisation des ensembles solutions

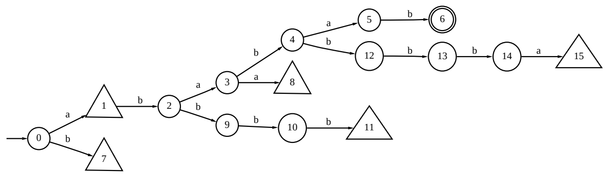
Pour analyser les relations entre le problème de l'inférence régulière et les fusions, nous introduisons la notion de PTA augmenté §3.2.1, suivant la description de la structure de données proposée en [HOV96], pour pouvoir traiter simultanément les informations fournies par les échantillons positif I_+ et négatif I_-. Une caractérisation générale des fusions dans le PTA augmenté est présentée §3.2.2. Le cas de la recherche d'automates canonique est étudié §3.2.3, un algorithme de caractérisation des fusions interdites étant proposé §3.2.4.

3.2.1 PTA augmenté

Définition 10 *Nous appelons* **PTA augmenté**, *noté* $APTA(I_+, I_-)$, *le sextuplet* $(Q, \Sigma, \delta, q_0, F^+, F^-)$ *avec :*

- $PTA(I_+ \cup I_-) = (Q, \Sigma, \delta, q_0, F^+ \cup F^-)$
- F^+, F^- *sont les sous-ensembles de* Q *identifiant respectivement les états d'acceptation de* I_+ *et* I_-, *les seuls états finals étant ceux de* F^+.

Nous donnons en illustration le PTA augmenté de l'échantillon 1 du langage 2 (cf §4), où $I_+ = \{ababab\}$ et $I_- = \{a, b, abaa, abbbb, ababbbba\}$



(les états de F^+ sont représentés par des double-cercles, ceux de F^- par des triangles).

Intuitivement, le PTA augmenté peut être considéré comme la superposition du $PTA(I_+)$ et du $PTA(I_-)$, chaque état étant étiqueté comme membre de l'une des 3 classes : état d'acceptation d'une instance positive, état d'acceptation d'une instance négative ou état intermédiaire, n'acceptant aucune instance.

On a par construction la propriété suivante :

$$L(APTA(I_+, I_-)) = I_+ \tag{1}$$

De plus, pour une partition π sur les états de $PTA(I_+)$:

$$L(PTA(I_+)/\pi) = L(APTA(I_+, I_-)/\pi) \tag{2}$$

3.2.2 Fusions dans le PTA augmenté

Une étude des différents types de fusions et de leur conséquences pour RPNI a été effectuée dans [Tho96]. Nous considérons ici la notion de *fusion incompatible.*

Définition 11 *Nous appelons* **fusion incompatible***, une fusion contenant un état de* F^+ *et un état de* F^-*.*

L'intérêt de cette appellation est que les automates obtenus par de telles fusions de l'automate $APTA(I_+, I_-)$ contiennent les automates les plus spécifiques non solutions puisqu'ils acceptent au moins une instance de I_-. Nous pouvons alors caractériser l'ensemble $f_{NOK}(APTA(I_+, I_-))$ des fusions sur l'automate $APTA(I_+, I_-)$ menant à l'acceptation d'une phrase de I_- : les fusions incompatibles et les fusions introduisant un non-déterminisme qui, pour être résolu, implique de réunir des états dont la fusion est incompatible.

3.2.3 Caractérisation de l'ensemble des automates canoniques

Pour la recherche de l'ensemble des automates canoniques définis par le théorème 3, on peut utiliser comme opération de déterminisation, la *fusion pour déterminisation.* La caractérisation de l'ensemble $f_{NOK}(APTA(I_+, I_-))$ est alors la suivante :

Pour la recherche d'automates canoniques, une fusion f appartient à $f_{NOK}(APTA(I_+, I_-))$ ssi f vérifie au moins l'une des deux conditions suivantes :

1. f est une fusion incompatible ;
2. il existe une fusion $f' \in f_{NOK}$ telle que $f \to^* f'$.

L'automate $APTA(I_+, I_-)$ n'étant autre que l'automate $PTA(I_+)$ auquel ont été rajoutées des branches "inutiles"(ne conduisant pas à des reconnaissances de mots), l'ensemble $f_{NOK}(PTA(I_+))$ est la restriction de l'ensemble $f_{NOK}(APTA(I_+, I_-))$ aux fusions impliquant uniquement des états de $PTA(I_+)$.

En tenant compte de la structure particulière du PTA augmenté, on peut établir que $F_{NOK}(APTA(I_+, I_-))$ est composé exclusivement de fusions à deux états qui sont soit des fusions incompatibles, soit des fusions forçant des fusions incompatibles, ce que nous formalisons dans la relation suivante :

Propriété 2

$$\begin{aligned} F_{NOK}(APTA(I_+, I_-)) = \{\ &\{q_1, q_2\}/(q_1 \in F^+ \wedge q_2 \in F^-) \\ &ou \quad (\exists f \in F_{NOK}(APTA(I_+, I_-))/\{q_1, q_2\} \rightarrow^* f)\} \end{aligned}$$

La démonstration de cette relation s'effectue en remarquant que les plus petites fusions incompatibles sont celles contenant uniquement un état de F^+ et un état de F^- et en tenant compte de la structure de l'automate $APTA(I_+, I_-)$ qui est un arbre par construction. Ainsi, chaque fusion $f' = \{q'_1, q'_2\}$ dans F_{NOK} a pour une lettre a donnée au maximum un père $f = \{q_1, q_2\}$ dans F_{NOK}, avec $\delta(q_1, a) = q'_1$ et $\delta(q_2, a) = q'_2$.

3.2.4 Algorithme de construction de F_{NOK}

L'utilisation de la propriété 2 permet de concevoir un algorithme efficace de construction de $F_{NOK}(APTA(I_+, I_-))$ - et donc de $F_{NOK}(PTA(I_+))$ -.

algorithme 1 Algorithme de génération de F_{NOK}

$F_{NOK} \leftarrow \emptyset$
pour tout q^+ de F^+ **faire**
  pour tout q^- de F^- **faire**
    propager_incompatibilité(q^+,q^-)
  fin pour
fin pour
Fonction propager_incompatibilité(q_1,q_2)
si $\{q_1, q_2\} \notin F_{NOK}$ **alors**
  $F_{NOK} \leftarrow F_{NOK} \cup \{q_1, q_2\}$ { On sait, du fait de la structure d'arbre, que tout état q a un prédécesseur unique, pour un caractère de transition unique Symbole_vers(q)}
  si Symbole_vers(q1)=Symbole_vers(q2) **alors**
    propager_incompatibilitéé(père(q_1),père(q_2))
  fin si
fin si
retourner (F_{NOK})

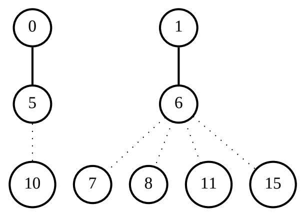
Si p est le nombre d'instances positives, n le nombre d'instances négatives et m la longueur moyenne des instances, l'algorithme est dans le pire des cas en o(pnm). On sait également que le nombre d'opérations est limité par la moitié du carré de la taille du PTA augmenté, puisque à chaque étape dans l'algorithme, on considère une paire d'états différents de l'automate.

3.3 Génération des automates solution avec F_{NOK}

$F_{NOK}(PTA(I_+))$ est une représentation dans l'espace des fusions qui peut être utilisée en tant que telle. Elle peut être utile pour une approche dans le cadre de l'espace des versions ou pour le choix des fusions à effectuer dans un algorithme du type RPNI procédant à l'exploration de l'espace par fusions successives.

On peut aussi vouloir revenir dans l'espace classique des automates. $f_{OK}(PTA(I_+))$ étant l'ensemble de toutes les fusions ne contenant pas de fusion de $F_{NOK}(PTA(I_+))$, il est possible de construire $f_{OK}(PTA(I_+))$ et $F_{OK}(PTA(I_+))$ à partir de $F_{NOK}(PTA(I_+))$. Nous avons donc deux représentations duales de l'ensemble des solutions dans l'espace des fusions. Si l'on ne s'intéresse qu'au critère de compatibilité, les automates solutions correspondants sont les automates A_0/π tels que chaque bloc B_i de la partition $\pi = (B_0, \ldots, B_r)$ n'inclut aucune fusion de $F_{NOK}(PTA(I_+))$.

3.3.1 Automates canoniques et k-coloriage

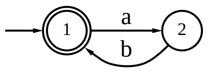


Si l'on considère la relation 2 qui spécifie que pour les automates canoniques, $F_{NOK}(APTA(I_+, I_-))$ n'est composé que de couples d'états (k, l), l'ensemble $F_{NOK}(PTA(I_+))$ peut être représenté par un graphe, un arc joignant les états k et l ssi (k, l) appartient à $F_{NOK}(PTA(I_+))$ (ceux dont la fusion provoque l'obtention d'un automate non compatible avec I_-). Nous donnons ainsi le graphe de F_{NOK} pour l'exemple traité précédemment (§3.2.1).

La relation entre le graphe $F_{NOK}(APTA(I_+, I_-))$ et l'ensemble des automates solutions peut alors être formulée en terme de coloriage. Nous rappelons la définition d'un coloriage :

Définition 12 *[BCL92] Un graphe $G = (S, A)$, où S désigne l'ensemble des sommets et A l'ensemble des arcs, est k-coloriable ssi il existe une application γ de S dans l'ensemble $\{1, 2, \ldots, k\}$ telle que $(u, v) \in A$ entraîne $\gamma(u) \neq \gamma(v)$ (une telle fonction est appelée k-coloration)*

Les automates A_0/π solution, $\pi = (f_0, \ldots, f_r)$, sont tels que chaque fusion f_i contient des états compatibles, c'est à dire de la même couleur. De plus, si l'on recherche les automates canoniques donc déterministes, l'ensemble des fusions doit vérifier la condition $\forall f_i \in \pi, f_i \rightarrow^* f \Rightarrow \exists f_j \in \pi, f \subseteq f_j$. Construire un automate canonique solution, c'est trouver une k-coloration vérifiant ces critères.



Si l'on désire satisfaire en plus le critère de minimalité sur les automates, la k-coloration doit être minimale La figure ci-contre donne ainsi le seul automate canonique minimal solution pour l'exemple du §3.2.1.

Bien entendu, le problème du coloriage demeure NP-complet dans sa généralité. La forme du graphe étant particulière, la complexité de notre algorithme reste néanmoins à étudier. Le but est ici de déplacer la limite des problèmes pratiquement traitables et de permettre des approches heuristiques plus performantes. On réduit au minimum les calculs par rapport à une approche où l'on doit tester à chaque étape que les contre-exemples sont bien rejetés. On espère également que la forme particulière du graphe et la recherche de solutions minimales déterministes contraindra suffisamment la recherche des coloriages.

Nous présentons dans la prochaine section une première expérimentation montrant la faisabilité de l'approche proposée.

4 Expérimentation

Nous avons réalisé une implémentation naïve (en C, non optimisée) des idées décrites dans cet article pour la recherche d'automates canoniques compatibles. Les étapes de l'algorithme sont données. On utilise pour le coloriage une adaptation du programme DSATUR développé par D. Brélaz [Bré79].

algorithme 2 Algorithme de génération des automates canoniques solution

- Construction du PTA augmenté
- Construction de $F_{NOK}(APTA(I_+, I_-))$
- Recherche de tous les coloriages minimaux sur $F_{NOK}(PTA(I_+))$ avec inclusion de la fusion pour déterminisation (pour obtention des automates déterministes).
- Pour chaque k-coloriage minimal, production de l'automate solution correspondant : l'automate dérivé du $PTA(I_+)$ par fusion des états de même couleur.

Nous résumons les résultats des exécutions sur les 20 paires d'échantillons de la base de test de [Dup94] dans le tableau suivant :

Langage	#états/#trans	#sols	taille APTA	#paires	temps
L1	1/1	1	3.2	0	0.01
L2	2/2	1	25.3	43.0	0.01
L3	4/7	1.75	180.9	998.2	6.35
L4	3/5	8.5	102.2	192.2	8.64
L5	4/8	2.1	65.1	217.8	0.70
L6	3/6	2.45	40.7	108.5	0.07
L7	4/7	4.75	85.2	545	2.87
L8	2/2	1.25	19.0	43.0	0.01
L9	4/7	1.65	60.8	357.9	0.05
L10	5/6	2.4	81.2	431.7	0.12
L11	4/8	2.85	64.8	214.85	6.11
L12	3/3	1.5	37.6	120.2	0.01
L13	2/4	1.25	32.4	70.7	0.04
L14	3/4	1.4	37.6	100.5	0.04
L15	4/6	2.1	70.7	353.3	0.04

#états/#trans caractérise la taille des automates cibles en nombre d'états et de transitions ; #sols est le nombre moyen d'automates solution les plus généraux trouvés ; taille APTA est la taille moyenne du PTA augmenté ; #paires est le nombre moyen de paires incompatibles. Le temps de production de tous les k-coloriages minimaux moyen est donné en secondes pour un sparc ultra cadencé à 200MHz.

La production des solutions est dans l'ensemble très rapide. Il faudrait bien entendu valider la méthode sur des vocabulaires et des ensembles d'instances plus grands, mais d'ores et déjà, la méthode conduit à des résultats qui étaient restés inaccessibles aux autres méthodes. En particulier, il n'existait à notre connaissance aucune méthode permettant d'obtenir l'ensemble des solutions en un temps raisonnable.

Pour les problèmes complexes, poser le problème en termes de coloriage permettra d'introduire facilement d'autres types de contraintes dans la sélection des automates pertinents.

5 Conclusion

Nous avons proposé une caractérisation de l'ensemble des solutions canoniques du problème de l'inférence régulière à partir d'exemples et de contre-exemples. Nous avons également ramené le problème de la production effective des automates solution à celui du coloriage de graphe. Les premiers résultats pratiques, obtenus en appliquant un algorithme standard de coloriage sont très encourageants. Bien que nous n'ayons pas tiré parti dans cette étude de la structure particulière du graphe des états non fusionnables, la plupart des problèmes ont pu être traités de façon rapide.

Les perspectives de ce travail sont nombreuses et peuvent être résumées selon deux axes principaux.

Nous souhaitons tout d'abord nous attaquer au problème du coloriage. La rapidité de celui-ci dépend fortement de l'estimation du nombre de couleurs nécessaires, i.e. de l'estimation de la taille des automates solution. La définition d'heuristiques adaptées et l'intégration de la déterminisation dans le coloriage devrait nous permettre de proposer des avancées dans ce domaine. Sur le plan théorique, le problème reste ouvert de savoir si la structure de F_{NOK} permet de se réduire à un problème polynomialement traitable.

La seconde partie de nos recherches portera sur l'intégration de ces travaux dans le cadre de l'espace des versions. En particulier, la définition d'une version incrémentale de l'algorithme semble un but accessible.

References

[AS83] Angluin (D.) et Smith (C.H.). – Inductive inference : Theory and methods. *Computing Surveys*, vol. 15, 1983, pp. 237–269.

[AU72] Aho (A.) et Ullman (J.). – *The Theory of Parsing, Translation and compiling, Vol 1 : Parsing.* – Englewood Cliffs, Prentice-Hall, 1972.

[BCL92] Barthélémy (J.-P.), Cohen (G.) et Lobstein (A.). – *Complexité algorithmique.* – Masson, 1992.

[Bré79] Brélaz (D.). – New methods to color the vertices of a graph. *Communications of the ACM*, vol. 22, 1979, pp. 251–256.

[DME94] Dupont (P.), Miclet (L.) et E.Vidal. – What is the search space of the regular inference ? *ICGI'94, Grammatical inference and Applications*, 1994, pp. 25–37. – Springer Verlag.

[Dup94] Dupont (P.). – Regular grammatical inference from positive and negative samples by genetic search : the gig method. *ICGI'94, Grammatical inference and Applications*, 1994, pp. 236–245. – Springer Verlag.

[Dup96] Dupont (P.). – *Utilisation et apprentissage de modèles de langages pour la reconnaissance de la parole continue.* – Thèse de PhD, Ecole Nationale Supérieure des Télécommunications, 1996.

[FB75] Fu (K. S.) et Booth (T. L.). – Grammatical inference : introduction and survey. *IEEE Transactions on SMC*, vol. SMC-5, n° 5, 1975, pp. 85–111 409–423.

[Gio93] Giordano (J.Y.). – Espace des versions et inférence de grammaires algébriques. *In : Journées Francophones d'Apprentissage.*

[Gio95] Giordano (J.Y.). – *Inférence de grammaires algébriques.* – Thèse de PhD, Université de Rennes1, jan 1995.

[Hir92] Hirsh (H.). – Polynomial-time learning with version spaces. *In : National Conference on Artificial Intelligence*, pp. 117–122. – San Jose, CA, juillet 1992.

[HOV96] Higuera (de la) (C.), Oncina (J.) et Vidal (E.). – Identification of dfa : data-dependent versus data-independant algorithms. *Grammatical inference Learning Syntax from Sentences, ICGI'96*, 1996, pp. 311–325. – Springer Verlag.

[MdG94] Miclet (L.) et de Gentille (C.). – Inférence grammaticale à partir d'exemples et de contre-exemples : deux algorithmes optimaux : (big et rig) et une version heuristique (brig). *JAVA94, Journées Acquisition, Validation, Apprentissage*, 1994, pp. F1–F13. – Strasbourg France.

[Mic79] Miclet (L.). – *Inférence de Grammaires Régulières.* – Thèse de PhD, Ecole Nationale Supérieure des Télécommunications, 1979.

[Mit82] Mitchell (T.). – Generalization as search. *Artificial Intelligence*, no18, 1982, pp. 203–226.

[Nic93] Nicolas (J.). – Une représentation efficace pour les espaces de versions. *In : 8ièmes JFA93.* – Saint-Raphael, mars 1993.

[OG92] Oncina (J.) et Garcia (P.). – Inferring regular languages in polynomial update time. *Pattern Recognition and Image Analysis*, 1992, pp. 49 – 61.

[Seb94] Sebag (M.). – Une approche par contraintes de l'espace des versions. *In : RFIA'94*, pp. 17–25.

[Tho96] Thollard (F.). – Etude algorithmique de l'inférence grammaticale. *Mémoire de DEA, Université de Montpellier II*, 1996.

[VB87] Vanlehn (K.) et Ball (W.). – A version space approach to learning context-free grammars. *Machine Learning*, no2, 1987, pp. 39–74.