L'inférence grammaticale régulière vue comme un problème de coloriage et propagation de contraintes sur un graphe

François Coste, Jacques Nicolas
IRISA Campus de Beaulieu
35042 RENNES Cedex
fcoste@irisa.fr, jnicolas@irisa.fr

1 Introduction

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

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_-. Dans ce cadre, nous essayons de résoudre le problème du *plus petit automate fini déterministe (AFD) compatible* qui peut être défini comme suit : étant donnés un échantillon positif I_+ et un échantillon négatif I_-, il s'agit de trouver un AFD A tel que

- $L(A) \supseteq I_+$
- $L(A) \cap I_- = \emptyset$
- A est un AFD comportant le nombre minimal d'états.

où $L(A)$ est le langage accepté par l'automate A.

Le problème du *plus petit AFD compatible* a été montré NP-complet [Ang78][Gol78]. Mais, 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 une propriété raisonnable (complétude structurelle), l'espace de recherche est un ensemble fini pouvant être partiellement ordonné. Dans ce cadre, avec I_- non vide, les principales méthodes actuelles[OG92] [MdG94] [Dup94] consistent à cheminer dans cet ensemble pour trouver la meilleure solution compatible avec I_- . Ces algorithmes ne produisent qu'un seul automate et nécessitent des conditions supplémentaires sur I_+ et I_- pour garantir que la solution fournie est effectivement l'automate recherché.

Notre approche aborde le problème de la caractérisation de *l'ensemble* des automates solutions. Un algorithme de construction de l'ensemble des solutions 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. Le problème du *plus petit AFD compatible* peut alors être reformulé comme une recherche de plus petit coloriage.

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 est développée en section 3. Nous terminons par une expérimentation sur un jeu de tests [Dup94] largement employé dans le domaine.

2 Inférence grammaticale régulière : rappels

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* $Q \times \Sigma \rightarrow 2^Q$ *(qu'on étend 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.*

Le langage accepté par un automate fini A est : $L(A) = \{\sigma \in \Sigma^*, \delta(q_0, \sigma) \cap F \neq \emptyset\}$. L'acceptation d'une phrase σ correspond à l'existence d'un chemin dans l'automate entre q_0 et un état de F.

Définition 2 *Si pour tout état $q \in Q$ et tout $a \in \Sigma$, $\delta(q, a)$ a au plus un élément, l'automate est* **déterministe**.
Pour tout langage régulier L, il existe un automate déterministe minimal (possédant le moins d'états) unique, appelé **automate canonique** *de L, noté $A(L)$.*

Définition 3 *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 :*

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

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.

Remarque 1 *Si l'on caractérise une fusion f par l'ensemble des états fusionnés $f = \{q_1, \ldots, q_r\}$, une partition π d'un ensemble d'états peut être définie comme un ensemble de fusions deux à deux disjointes : $\pi = (f_0, \ldots, f_p)$ avec $\forall(f_i \neq f_j), f_i \cap f_j = \emptyset$, où tout état n'appartenant pas à une fusion est un singleton dans la partition.*

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 4 *L'ensemble partiellement ordonné des automates dérivés d'un automate A est un treillis, que nous désignerons par $Lat(A)$.*

2.2 Espace de recherche

Définition 5 *[Dup94] 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.

Définition 6 *A partir d'un échantillon I, on peut construire l'***automate canonique maximal***, noté $MCA(I)$*[1] *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 7 *L'***arbre accepteur des préfixes** *[Mic79], noté $PTA(I)$*[2]*, est l'automate obtenu en fusionnant les états partageant les mêmes préfixes* [3] *du $MCA(I)$.*

Théorème 1 *[Dup94] Soit I_+ un échantillon positif d'un langage régulier L et soit $A(L)$ l'automate canonique de L. Si I_+ est structurellement complet relativement à $A(L)$ alors $A(L)$ peut être obtenu par dérivation du $PTA(I_+)$.*

3 Caractérisation des solutions dans l'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, le problème du *plus petit AFD compatible* peut être considéré comme un problème de recherche du plus petit automate compatible dans un treillis.

Les automates du treillis, obtenus par dérivation de l'automate nul du treillis $PTA(I_+)$, peuvent s'écrire en termes de partitions $PTA(I_+)/\pi$ ou en termes de fusions $PTA(I_+)/(f_0, \ldots, f_p)$ (cf remarque 1). Nous pouvons alors choisir de considérer comme espace de recherche l'ensemble des fusions possibles sur le $PTA(I_+)$ que nous appelons espace des fusions.

Définition 8 *Nous appelons* **PTA augmenté***, noté $APTA(I_+, I_-)$, le sextuplet $(Q, \Sigma, \delta, q_0, F_+, F_-)$ où*

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

Dans cet espace, les fusions menant à un automate dérivé incompatible (acceptant au moins une phrase de I_+ et de I_-) sont celles menant à la fusion d'un état de F_+ et un état de F_-. Nous proposons de caractériser ces fusions *incompatibles* par l'ensemble de contraintes suivant.

[1] Maximal Canonical Automaton.
[2] Prefix Tree Acceptor.
[3] u est un préfixe de l'état q si $q \in \delta(q_0, u)$.

Définition 9 *Soit $F_{NOK}(APTA(I_+, I_-))$ l'ensemble des contraintes sur les partitions π de l'ensemble des états du $APTA(I_+, I_-)$.*

$$q_\pi \neq q'_\pi \qquad \forall q \in F_+, \forall q' \in F_- \tag{1}$$

$$q_{1\pi} \neq q_{2\pi} \Rightarrow q'_{1\pi} \neq q'_{2\pi} \quad \forall q_1, q_2 \in APTA(I_+, I_-)/ \; (q'_1, q'_2) \rightarrow (q_1, q_2) \tag{2}$$

où q_π désigne le bloc de la partition (la fusion) auquel appartient q et la relation binaire entre couple d'états $\rightarrow$ est définie comme suit :

$$(q_1, q_2) \rightarrow (q'_1, q'_2) \text{ ssi } \exists a \in \Sigma \; \delta(q_1, a) = q'_1 \wedge \delta(q_2, a) = q'_2 \tag{3}$$

L'ensemble des fusions incompatibles peut être représenté par un graphe G dans lequel il existe un arc (q, q') pour chaque contrainte $q_\pi \neq q'_\pi$. Le problème de la recherche d'un AFD compatible peut alors être reformulé sous la forme de la recherche d'une *k-coloration* sur le graphe G avec propagation des contraintes selon la relation 2. L'automate solution sera alors obtenu en fusionnant les états de même couleur. Si l'on désire satisfaire en plus le critère de minimalité sur les automates, la k-coloration doit être minimale (et le nombre chromatique du graphe correspond au nombre d'états des automates solution).

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. Les étapes de l'algorithme sont données. On utilise pour le coloriage une adaptation du programme DSATUR développé par D. Brélaz [Bré79].

Nous résumons les résultats dans le tableau suivant. #états/#trans caractérise la

algorithme 1 Algorithme de génération des automates canoniques solution

construction du PTA augmente
construction de F_{NOK} (complexité quadratique)
recherche de tous les coloriages minimaux sur le graphe issu de F_{NOK} avec propagation des contraintes.

taille des automates cibles en nombre d'états et de transitions. #sols est le nombre moyen d'automates solution trouvés. taille PTA est la taille moyenne du PTA augmenté. #paires est le nombre moyen de paires inconsistantes

Les temps sont donnés en secondes sur un UltraSparc cadencé à 200 Mhz.

Langage	#états/#trans	#sols	taille PTA	#paires	temps
L1	1/1	1	3.2	0	0.01
L2	2/2	1	25.3	43.0	0.03
L3	4/7	1.75	180.9	998.2	18.3
L4	3/5	8.5	102.2	192.2	7.08
L5	4/8	2.1	65.1	217.8	1.81
L6	3/6	2.45	40.7	108.5	0.51
L7	4/7	4.75	85.2	545	3.15
L8	2/2	1.25	19.0	43.0	0.02
L9	4/7	1.65	60.8	357.9	0.45
L10	5/6	2.4	81.2	431.7	0.59
L11	4/8	2.85	64.8	214.85	4.17
L12	3/3	1.5	37.6	120.2	0.09
L13	2/4	1.25	32.4	70.7	0.20
L14	3/4	1.4	37.6	100.5	0.16
L15	4/6	2.1	70.7	353.3	0.34

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'ors et déjà, la méthode conduit à des résultats qui étaient restés inaccessibles aux autres méthodes.

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 avec propagation des contraintes. 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, les problèmes ont pu être traités de façon rapide.

Les perspectives de ce travail sont nombreuses mais 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 devrait nous permettre de proposer des avancées dans ce domaine. De même une gestion efficace de la propagation des contraintes est à étudier. 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.

References

[Ang78] Angluin (D.). – On the complexity of minimum inference of regular sets. *Information and Control*, vol. 29, n° 3, 1978, pp. 741 – 765.

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

[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.

[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.

[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.

[Gol78] Gold (E. M.). – Language identification in the limit. *Information and Control*, vol. 37, 1978, pp. 302 – 320.

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

[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.

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