Inférence régulière

Rigoux Lionel

2007

Table des matières

Introduction

Nous allons nous intéresser à l'apprentissage de grammaires régulières, c'est à dire la construction dynamique d'un automate fini à partir d'un ensemble de mots. Ce problème, connu sous le nom d'*inférence grammaticale*, fut d'abord posé par Gold en 1967 et a donné naissance à de nombreuses recherches théoriques au cours de ces dernières décennies. Si de nombreux algorithmes – utilisant généralement la fusion d'états – ont été découverts, la majorité des résultats sont négatifs dans le cas général et nous ont fourni de nouveaux problèmes NP-complets, souvent en correspondance avec la cryptographie.

Nous présenterons tout d'abord le paradigme de l'inférence grammaticale, en exposant les principaux modèles d'apprentissage. Ensuite, après avoir donné un cadre formel à notre problème, nous définirons un espace de recherche et un algorithme permettant l'induction d'automates déterministes.

1 Modèles d'identification

Exposons tout d'abord les principaux paradigmes d'inférence qui existent. Nous nous restreindrons aux cas de l'apprentissage d'automates déterministes, laissant volontairement de côté d'autres approches comme l'inférence d'automates à états résiduels ou l'apprentissage d'automates probabilistes. Le lecteur pourra aisément trouver une littérature abondante sur ces derniers modèles.

1.1 Quelques définitions préliminaires

Nous commencerons par introduire le vocabulaire propre au problème de l'inférence grammaticale. Les définitions suivantes, très simples, sont communes aux divers modèles que nous verrons.

1.1.1 Exemple

Un *exemple* d'un langage L est la donné d'une paire (w, e) où w est un mot de Σ^* et e un booléen qui vaut 1 si $w \in L$ et 0 sinon. Il est dit *positif* dans le premier cas, *négatif* sinon. Les exemples négatifs sont aussi appelés *contre-exemples*.

1.1.2 Présentation

Une *présentation* σ d'un langage L est une séquence infinie d'exemples de L. Une présentation est dite *complète* si elle contient tous les mots de Σ^*.

Une présentation est dite *positive* (res. *négative*) si elle ne contient que des exemples positifs (res. négatifs). Nous noterons alors respectivement σ_+ et σ_-. Une présentation positive est dite *complète* si elle contient tous les

mots de L. Une présentation négative est dite complète si elle contient tous les mots de $\Sigma^* \setminus L$).

1.1.3 Échantillon

Un *échantillon* I est un ensemble fini d'exemples. De manière similaire aux présentation on définit les notions d'*échantillon positif*, noté I_+, et d'*échantillon négatif*, noté I_- .

1.1.4 Consistance

Un automate est dit *consistant* avec une présentation positive ou échantillon positif s'il reconnaît tous les exemples (positifs) de la séquence.

1.1.5 Compatibilité

Un automate $\mathcal{A}$ est dit *compatible* avec une présentation négative ou un échantillon négatif s'il rejette tout les exemples (négatifs) de la séquence.

1.2 Identification à la limite

Le premier modèle d'inférence régulière a été défini par Gold en 1967. Dans ce paradime, dit d'*identification à la limite*, l'apprentissage est considéré comme un processus infini.

1.2.1 Modèle de Gold

Voici la première formulation du problème de l'apprentissage de langages formels :

Identification à la limite On dira qu'un algorithme M identifie à la limite une classe de langage $\mathcal{L}$ si pour tout langage L de $\mathcal{L}$ et pour toute présentation complète $\sigma = (e_i)_{i\epsilon\mathbb{N}}$ de L il existe un indice T tel que si M prend en entrée les $\sigma_{t\geqslant T}$ alors il renvoie une représentation de L (*i.e.* un automate reconnaissant exactement L). La classe $\mathcal{L}$ est dite *identifiable* ou *apprenable à la limite* .

Nous voyons que ce modèle ne donne aucun contrainte sur le temps d'apprentissage de l'automate cible. En effet, si le temps d'exécution de l'algorithme M est fini, il peut être arbitrairement long. En particulier, si l'existence de l'indice T est assurée, nous ne pouvons savoir à quel moment celui-ci est atteint, et donc à quel moment arrêter l'algorithme. Ainsi nous ne pouvons être certain que l'automate retourné est le bon qu'au bout d'un temps infini, ce qui explique le terme d'apprentissage "à la limite". Nous avons néanmoins le résultat suivant, plutôt encourageant :

Théorème (Gold 67) Toute classe récursivement énumérable est apprenable à la limite par présentation complète.

Preuve En effet, on peut trouver un algorithme extrêmement simple permettant de trouver l'automate deterministe minimal consistant avec un échantillon I : il suffit d'énumérer les automates déterministes par ordre croissant et de retourner le premier compatible avec I. Il est alors facile de se convaincre qu'il existe un index T fini tel que si cet algorithme prend les T premiers exemples d'une présentation complète σ d'un langage L il renvoie l'automate minimal compatbile avec σ.

La classe des langages réguliers est donc apprenable dans ce modèle. Mais il est évident qu'un tel résultat est difficilement exploitable dans la pratique. Nous avons d'ailleurs le constat suivant en complexité :

Théorème (Gold 78) Le problème de trouver, pour un échantillon I et un entier t, le plus plus petit automate d'au plus t états consistant avec I est *NP-Complet*.

Nous obtenons là un premier résultat négatif. Diverses variantes du modèle de Gold ont par la suite vu le jour, et tentent d'introduire de nouvelles contraintes pour restreindre la complexité du problème. Voyons maintenant quelques un de ces paradigmes.

1.2.2 Identification avec oracle

Une solution au problème de non-terminaison à été proposée par Angluin en 1987 : le recours à un oracle. Ce dernier est une entité qui répond aux questions du "disciple" (*i.e.* celui qui essaye d'apprendre le langage). L'élève se voit proposer deux options :

- demander un exemple (positif) à l'oracle, ce qui lui permettra d'utiliser un algorithme d'inférence – sur tous les exemples déjà demandés – pour déduire un automate solution
- faire une "demande d'équivalence" pour que l'oracle confirme que l'automate trouvé est bien le bon ou bien donne un contre-exemple dans le cas contraire

Angluin a même donné en 87 un algorithme (appelé L^*) dont la complexité a été étudiée par Pitt en 89. Nous n'étudierons pas plus en détail ce modèle vu la difficulté de trouver un oracle dans la pratique.

1.2.3 Identification à la limite par exemples positifs

Une autre façon de restreindre le modèle de l'identification à la limite est de n'autoriser que les présentations positives. On parlera alors de classes de langages identifiables à la limite par exemples positifs seuls.

Voici un résultat qui montre que cette contrainte entraîne un énorme rétrécissement de l'ensemble des classes apprenables. On rappelle qu'une classe superfinie est une classe de langages qui contient tous les langages de cardinalité finie et au moins un langage de cardinalité infinie.

Théorème (Gold 67) Aucune classe superfinie n'est identifiable à la limite à partir de présentations positives.

Ceci implique que même la classe des langages réguliers n'est plus apprenable à la limite si l'on accepte la contrainte de la présentation positive !

On a l'extension de ce résultat :

Théorème (Kapur 91) Si une classe de langages $\mathcal{L}$ contient une suite $(L_n)_{n\in\mathbb{N}}$ et s'il existe une suite $(S_n)_{n\in\mathbb{N}}$ telle que $\forall n$, $S_n \subsetneq S_{n+1}$, $S_n \subseteq L_n$ et $\cup_{n\in\mathbb{N}} S_n \in \mathcal{L}$ alors $\mathcal{L}$ n'est pas identifiable à la limite par exemples positifs.

Il existe malgré tout des classes de langages non triviales qui sont apprenable à la limite par exemples positifs : la classe des langages k-testables, celle des langages k-réversibles, entre autres. Nous ne développerons pas plus en détails les définitions et algorithmes correspondant, par souci de généralité.

1.3 Identification par données fixées

1.3.1 Le modèle

En revenant sur la définition du modèle d'identification à la limite, on remarque que l'inférence exacte sur une présentation $\sigma = \{e_i\}_{0\leqslant i}$ est possible dès qu'un certain indice T est atteint. Autrement dit, l'échantillon $\{e_i\}_{0\leqslant i\leqslant T} \subset \sigma$ est suffisant pour l'inférence. Il est alors pertinent de se demander quelles propriétés doit avoir cet échantillon pour assurer le succès de l'algorithme utilisé. En supposant que ces propriétés ne dépendent que du langage cible, nous pouvons donner une définition non plus extensive – et donc dépendante de la présentation – mais compréhensive de l'échantillon suffisant pour l'inférence. Ceci est formalisé par le modèle de l'identification par données fixées :

Définition Une méthode d'inférence M identifie un langage L pour une présentation à données fixées s'il existe un échantillon $I^c = (I^c_+, I^c_-)$ tel que pour tout échantillon $I = (I_+, I_-)$ tel que $I^c_+ \subseteq I_+$ et $I^c_- \subseteq I_-$ la méthode M identifie L exactement en prenant I en entrée. I^c est alors appelé *échantillon caractéristique* de L pour M.

Ainsi le modèle d'*identification par données fixées* utilisera les mêmes algorithmes que le modèle d'identification à la limite, mais sur un échantillon précis – incluant l'échantillon caractéristique – plutôt que sur une présentation infinie.

Ce modèle pose aussi une contrainte sur le mode de représentation d'un langage. En effet il est exigé que l'échantillon caractéristique puisse être construit en temps polynomial du plus petit automate représentant le langage (dans le mode choisi), hors l'automate canonique (déterministe) est parfois beaucoup plus grand qu'un automate non déterministe équivalent. Dans le cas non déterministe, c'est l'*automate fini à états résiduels* qui est utilisé

pour la détermination de l'échantillon et l'inférence. Dans le cas déterministe on peut construire l'échantillon caractéristique directement comme nous le verrons ci dessous.

C. de la Higuiera l'a résumé ainsi :

Définition Une classe de langage $\mathcal{L}$ est apprenable polynomialemaent à partir de données fixées en utilisant un mode de représentation Rep s'il existe deux algorithmes $\mathcal{T}$ et $\mathcal{M}$ tels que pour chaque langage cible $L \in \mathcal{L}$ et pour n'importe quelle représentation $r \in Rep(L)$:
- $\mathcal{T}$ prenant r en entrée produit un échantillon d'apprentissage I^c dont la taille est polynomiale en la taile de r
- à partir de n'importe quel échantillon I de L avec $I^c \subseteq I$, $\mathcal{M}$ produit une représentation de L en temps polynomial en la taille de I

Notons tout de même que nous obtenons le résultat recherché :

Théorème (Gold 78) Les langages réguliers représentés par des automates finis déterministes sont polynomialment apprenables par données fixées.

Nous voyons maintenant pourquoi les problème de l'apprentissage à la limite appartient à NP : on peut "choisir" dans une présentation l'échantillon caractéristique, et appliquer ensuite un algorithme qui sera polynomial en la taille de l'échantillon, qui est lui-même polynomial en la taille de l'automate cible.

1.3.2 Spécification de l'échantillon caractéristique

Comme nous l'avons dit plus haut, nous ne nous intéresserons ici qu'au cas déterministe en donnant une définition de l'échantillon caractéristique pour les langages réguliers et l'algorithme RPNI qui sera développé en 3$^{\text{ème}}$ partie. Le cas non-déterministe, s'appuyant sur la notion d'automate fini à états résiduels, nécessite l'introduction de notions qui bien que simples sont très nombreuses, et qui ne seront donc pas abordées ici.

Etudions maintenant plus en détail le concept d'échantillon caractéristique. Nous illustrerons les différentes définitions qui suivent d'exemples basé sur l'automate $\mathcal{M}$ ci-dessous.

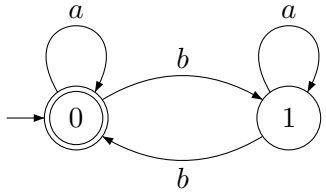

FIG. 1 – L'automate $\mathcal{M}$

Préfixes court L'ensemble des *préfixes courts* $Sp(L)$ d'un langage L est défini par :

$$Sp(L) = \{x \in Pr(L) \,|\, \nexists u \in \Sigma^* \; avec \; u^{-1}L = x^{-1}L \; et \; u < x\}$$

où $Pr(L)$ représente l'ensemble des préfixes de L. Nous savons que dans l'automate canonique $\mathcal{A}(L)$ du langage L, il y a autant d'états que de quotients à gauche $x^{-1}L$ des chaînes x appartenant aux préfixes de L. Ainsi l'ensemble des préfixes court est l'ensemble des premières chaînes dans l'ordre lexicographique qui chacune mène à un état de l'automate canonique $\mathcal{A}(L)$.

Dans notre exemple, nous avons $Sp(L(\mathcal{M})) = \{\varepsilon, b\}$

Noyeau d'un langage Le noyeau $N(L)$ d'un langage L est défini par :

$$N(L) = \{\lambda\} \cup \{xa \,|\, x \in Sp(L), a \in \Sigma, xa \in Pr(L)\}$$

C'est donc l'ensemble des préfixes courts rallongés d'une lettre qui sont toujours des préfixes de L.

Dans notre exemple, nous avons $N(L(\mathcal{M})) = \{\varepsilon, a, b, ba, bb\}$

Nous pouvons maintenant définir l'échantillon caractéristique.

Échantillon caractéristique Un échantillon $I^c = (I^c_+, I^c_-)$ est caractéristique relativement à un langage L et pour l'algorithme RPNI s'il vérifie les conditions suivantes :

1. $\forall x \in N(L)$, si $x \in L$ alors $x \in I^c_+$ sinon $\exists u \in \Sigma^*$ tel que $xu \in I^c_+$
2. $\forall x \in Sp(L), \forall y \in N(L)$ si $x^{-1}L \neq y^{-1}L$ alors $\exists u \in \Sigma^*$ tel que ($xu \in I^c_+$ et $yu \in I^c_-$) ou ($xu \in I^c_-$ et $yu \in I^c_+$).

La première partie de la définition implique en particulier la complétude structurelle de I^c_+ avec l'automate canonique (voir définition ci dessous). La seconde propriété permet de distinguer des préfixes menant à des états distincts de l'automate canonique (ayant des résiduels différents) en intégrant dans l'échantillon caractéristique des contre-exemples pertinents, ce qui, nous le verrons plus loin, empêchera une surgénéralisation dans les algorithmes à fusion d'états. Il est également assez facile de se convaincre qu'il existe un algorithme permettant de trouver un tel échantillon en temps polynomial à partir de l'automate canonique.

Nous laisserons le soin au lecteur de vérifier que dans notre exemple nous avons $I^c_+ = \{\varepsilon, a, bb, bba, baab, baaaba\}$ et $I^c_- = \{b, ab, aba\}$.

2 Espace de recherche

Nous allons maintenant poser un cadre un peu plus formel à ce problème. L'ensemble des définitions suivantes nous pourvoiront d'un espace de

recherche précis dans lequel nous pourrons appliquer les divers algorithmes d'inférence régulière. En particulier, nous allons voir que l'on peut toujours retrouver l'automate cible en fusionnant les états de l'automate accepteur des préfixes.

2.1 Complétude structurelle

Un échantillon I_+ est *structurellement complet* relativement à un automate $\mathcal{A}$, s'il existe une acceptation de I_+ par $\mathcal{A}$ telle que :

- toute transition de $\mathcal{A}$ soit exercée
- tout état final de $\mathcal{A}$ soit utilisé comme état d'acceptation

2.2 Automate maximal canonique

On construit à partir d'un échantillon positif $I_+ = \{u_1, \ldots, u_M\}$ l'*automate maximal canonique* $MCA(I_+) = (Q, \Sigma, \delta, q_0, F)$ comme suit :

- Σ est l'alphabet sur lequel I_+ est défini
- $Q = \{v_{i,j} | 1 \leq i \leq M, 1 \leq j \leq |u_i|, v_{i,j} = a_{i,1} \ldots a_{i,j}\} \cup \{\varepsilon\}$
- $q_0 = \varepsilon$
- $F = I_+$
- $\delta(\varepsilon, a) = \{a_{i,1} | a_{i,1} = a, 1 \leq i \leq M\}$
- $\delta(v_{i,j}, a) = \{v_{i,j+1} | v_{i,j+1} = v_{i,j}a, 1 \leq i \leq M, 1 \leq j \leq |u_i|\}$

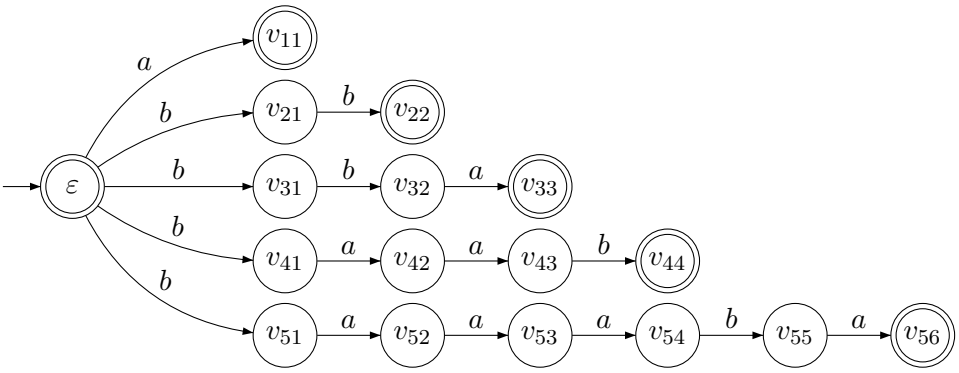

FIG. 2 – $MCA(\{\varepsilon, a, bb, bba, baab, baaaba\})$

Ainsi $MCA(I_+)$ est le plus grand (ayant le plus grand nombre d'états) automate déterministe émondé pour lequel I_+ est structurellement complet. Plus simplement, c'est l'automate arborescent qui aura à partir de l'état initial autant de "branches" qu'il y a de mots dans I_+, chaque branche étant composée de la succession des transitions nécessaires à la reconnaissance d'un mot de I_+. $MCA(I_+)$ est donc clairement non-déterministe.

2.3 Arbre accepteur des préfixes

On appelle *arbre accepteur des préfixes*, d'un échantillon I_+, noté $PTA(I_+)$ l'automate quotient $MCA(I_+)/\pi_{I_+}$ où π_{I_+} est la partition définie sur l'ensemble des mots de I_+ telle que $q \sim q'$ ssi q et q' ont même préfixe.

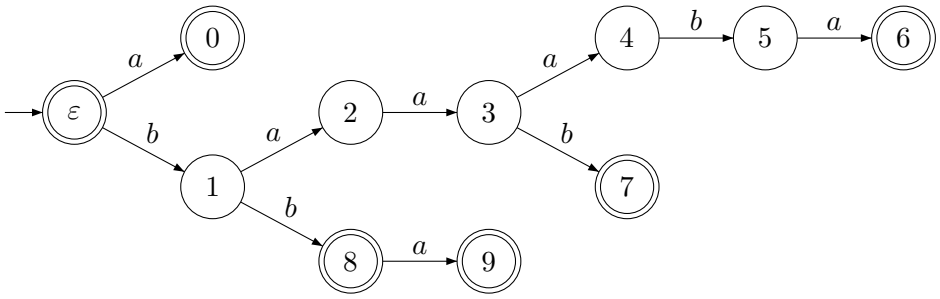

FIG. 3 – $PTA(\{\varepsilon, a, bb, bba, baab, baaaba\})$

Ainsi, on peut voir $PTA(I_+)$ comme une version déterministe de $MCA(I_+)$ dans lequel nous avons fusionné tous les états ayant le même préfixe.

2.4 Partitions et ordre

On dit qu'une partition π_2 dérive d'une autre partition $\pi_1 = \{B_1, \ldots, B_n\}$ s'il existe j et k, $(j \neq k)$, compris entre 1 et n, tels que :

$$\pi_2 = \{B_j \cup B_k\} \cup (\pi_1 \setminus \{B_j, B_k\})$$

C'est à dire si π_1 est plus fine que π_2. On peut alors noter $\pi_1 \preceq \pi_2$.

Par extension, on dira que $\mathcal{A}/\pi_1$ est plus fin $\mathcal{A}/\pi_2$ que $\mathcal{A}/\pi_2$ ou que dérive de $\mathcal{A}/\pi_1$, noté $\mathcal{A}/\pi_2 \preceq \mathcal{A}/\pi_2$.

Il est important de remarquer que l'on a alors $L(\mathcal{A}/\pi_1) \subset L(\mathcal{A}/\pi_2)$.

2.5 Treillis d'automates

L'ensemble des automates quotients de $\mathcal{A}$ partiellement ordonné par $\preceq$ forme alors un treillis complet et complémenté. On notera par la suite ce treillis $Lat(\mathcal{A})$.

Remarque : plus on "monte" dans le treillis, moins l'automate a d'états ; le maximum du treillis est l'automate universel, qui ne possède qu'un seul état.

2.6 Théorèmes fondamentaux

Nous allons voir maintenant quelques théorèmes qui nous permettront d'utiliser les définitions précédentes dans des algorithmes de recherche d'automates.

Théorème Soit $\mathcal{A}_{I_+}$ l'ensemble des automates auxquels un échantillon positif I_+ est structurellement complet. Cet ensemble est $Lat(MCA(I_+))$.

Théorème Soit I_+ un échantillon positif d'un quelconque langage régulier L et soit $\mathcal{A}(L)$ l'automate canonique acceptant L. Si I_+ est structurellement complet relativement à $\mathcal{A}(L)$ alors $\mathcal{A}(L)$ appartient à $Lat(PTA(I_+))$.

Nous avons de plus la propriété suivante :

Propriété $Lat(PTA(I_+)) \subseteq Lat(MCA(I_+))$

Notre problème de recherche de l'automate canonique peut se ramener à une exploration du treillis $Lat(PTA(I_+)$ qui contient notre automate cible si I_+ est structurellement complet.

Il est à noter que nous rechercherons par la suite des automates cibles déterministes, hors $Lat(PTA(I_+)$ contient des automates non déterministes. Le théorème suivant nous assure que l'exploration du treillis peut se faire uniquement sur les automates déterministes qu'il contient.

Théorème Soit I_+ un échantillon positif d'un quelconque langage régulier L et soit $\mathcal{A}(L)$ l'automate canonique acceptant L. Si I_+ est structurellement complet relativement à $\mathcal{A}(L)$ alors il existe une séquence d'automates (finis et déterministes) telle que :

$$PTA(I_+) = PTA(I_+)/\pi_0 \preceq PTA(I_+)/\pi_1 \preceq \cdots \preceq PTA(I_+)/\pi_n = \mathcal{A}(L)$$

Ceci explique que les algorithmes que nous rencontrerons seront basés sur la fusion d'états, c'est à dire sur la recherche de partitions ordonnées qui de proche en proche mènent à un automate quotient qui est l'automate cible.

Ceci est formalisé par un corolaire du théorème précédent :

Corolaire Tout automate fini déterministe $\mathcal{A} \in Lat(PTA(I_+))$ peut être obtenu par fusions successives en suivant une certaine séquence d'automates finis déterministes quotients du $PTA(I_+)$.

Nous pouvons déjà apercevoir comment réaliser un algorithme d'inférence : à partir de l'automate accepteur des préfixes, nous allons fusionner des états petit à petit jusqu'à trouver l'automate minimal. Dans le cas d'inférence sur exemple positifs seuls, on peut facilement "aller trop loin" dans le treillis, c'est à dire surgénéraliser l'échantillon et donc retourner un automate qui reconnaîtra un langage incluant strictement le langage cible. La présence dans l'échantillon caractéristique de contre-exemples précis nous évitera ce problème.

3 Algorithme RPNI

Nous allons enfin présenter un algorithme célèbre d'inférence régulière. Il en existe une multitude, largement documentée dans la littérature. Celui

présenté ici, très utilisé, doit plutôt être vus comme un exemple introductif. Il permet de construire le l'automate minimal déterministe à partir d'exemples positifs et négatifs.

Nous devons tout d'abord introduire une fonction permettant de déterminiser un automate par fusion d'états (attention, ce n'est pas une déterminisation classique !).

Algorithm 1 Fusion pour déterminisation

Require: π l'automate $\mathcal{A}$ et une partition de ses états
Ensure: π' une partition telle que $\mathcal{A}/\pi'$ soit déterministe
  $\pi' \leftarrow \pi$
  while $\exists B_i, B_j \in \pi', B_i < B_j$ tels que
  $\exists B, B < B_j$ et $\exists a \in \Sigma$ avec $B_i \in \delta(B, a)$ et $B_i \in \delta(B, a)$ **do**
    $\pi' = \{B_i \cup B_j\} \cup (\pi \setminus \{B_i, B_j\})$
  end while
  retourner π'

Les blocs de π sont indexés par le rang de sont état minimal dans l'ordre standard.

Nous pouvons maintenant écrire l'algorithme complet.

Algorithm 2 RPNI

Require: I_+, I_-
Ensure: un automate fini déterministe consistant avec I_+ et compatible avec I_-
  $\mathcal{A} \leftarrow PTA(I_+)$
  $\pi \leftarrow$ la partition triviale sur $\mathcal{A}$
  for $i = 1$ to $|\pi| - 1$ **do**
    for $j = 0$ to $i - 1$ **do**
      $\pi' \leftarrow \pi \setminus \{B_j, B_u\} \cup \{B_i \cup B_j\}$
      $\mathcal{A} \leftarrow \mathcal{A}/\pi'$
      $\pi'' \leftarrow$ fusion-pour-determinisation$(\mathcal{A}, \pi')$
      if compatible$(\mathcal{A}/\pi'', I_-)$ **then**
        $\mathcal{A} \leftarrow \mathcal{A}/\pi''$
        $\pi \leftarrow \pi''$
        sortir boucle
      end if
    end for
  end for
  retrouner $\mathcal{A}$

Théorème (Oncina et García 92) L'algorithme RPNI identifie à la limite la classe des langages réguliers.

En particulier, si $I = (I_+, I_-)$ contient un échantillon caractéristique, alors l'automate retourné par RPNI est l'automate déterministe canonique qui reconnaît le langage dont est issu l'échantillon.

Une façon de voir cet algorithme est de le remarquer qu'il fusionne les états correspondant à un même langage résiduel, en "repliant" l'automate accepteur des préfixes. Le fait que I contienne l'échantillon caractéristique nous assure d'une part que l'automate cible appartienne à $LAT(PTA(I_+))$ et soit accessible par fusion d'états à partir de $PTA(I_+)$, et d'autre part que deux états correspondant à des états résiduels distincts ne soient pas fusionnés grâce aux mots contenus dans I_-. Ce dernier point découle du point (2) de la définition de l'échantillon caractéristique. L'algorithme effectue ainsi une recherche en profondeur dans le treillis des automates quotients du $PTA(I_+)$, essayant d'aller le plus loin possible (pour trouver l'automate contenant le moins d'états) tout en respectant la frontière délimitée par I_-.

Pour une preuve plus complète de cet algorithme nous renvoyons le lecteur à l'article de Oncina et García paru en 1992. Le lecteur trouvera un exemple de l'exécution de cet algorithme sur l'automate donné dans la section 1.3.2 en [4, p. 54-58].

Il est à remarquer qu'il existe une version incrémentale de cet algorithme, beaucoup plus économique en temps lorsque les exemples ne sont pas donnés simultanément.

Conclusion

Le problème de l'inférence régulière est domaine vaste, qui contient beaucoup de résultats intéressants, mais souvent au prix de contraintes et de restrictions lourdes. Nous nous sommes ici intéressé au cas simple de l'apprentissage des langages réguliers représentés par des automates déterministes. Les modèles de l'identification à la limite et de l'identification sur données fixées ont montrés beaucoup de similitudes dans leurs résultats. En particulier nous avons présenté un algorithme polynomial en la taille de l'échantillon utilisé, dont la réussite est assurée pour tout échantillon non dégénéré, ou pour toute présentation complète dans le cas limite.

D'autres paradigmes existent, permettant l'inférence d'automates non déterministes ou bien l'apprentissage d'automates probabilistes. De nombreux algorithmes viennent enrichir ce champ de recherche encore très actif.

Références

[1] Carme J., Lemay A., Terlutte A., *Identification à la limite de langages réguliers d'arbres à résidules premiers disjoints*, 2003

[2] Costes F., Nicolas J., *Inférence grammaticale régulière : caractérisation de l'ensemble des solutions canoniques*

[3] Costes F., Nicolas J., *L'inférence grammaticale régulière vue comme un problème de coloriage et propagation de contraintes sur un graphe*

[4] Dupont P. et Miclet L., *Inférence grammaticale régulière : fondements théoriques et principaux algorithmes*, juillet 1998

[5] de la Higuiera C., *A bibliographical study of grammatical inference*, avril 2004

[6] Lemay A., *De l'apport des langages résiduels en inférence grammaticale de langages réguliers*, novembre 2002

[7] Oliveira A.L., Silva J.P., *Efficient search techniques for the inference of minimum sized finite state machines*

[8] Parekh R., Honavar V., *Learning DFA from simple examples*

[9] Parekh R., Honavar V., *Grammar inference, automata induction and language acquisition*