Apprentissage des langages réguliers d'arbres à l'aide d'un expert

Jérôme Besombes*, jean-Yves Marion*

LORIA-INRIA Lorraine
615, rue du jardin botanique
54602 Villers-lès-Nancy, France
et
INPL-Ecole des Mines de Nancy
Parc de Saurupt,
54042 Nancy Cedex, France
{Jerome.Besombes, Jean-Yves.Marion}@loria.fr

Résumé : Dans le but de modéliser l'apprentissage des langues naturelles, nous nous intéressons à l'apprentissage des langages réguliers d'arbres à partir d'exemples positifs et en interaction avec un expert. L'expert est un oracle qui connaît le langage appris et qui répond à des questions d'appartenance. Nous donnons un algorithme efficace d'apprentissage dans ce paradigme, nous montrons sa correction et l'illustrons sur un exemple détaillé.

Mots clef : langages réguliers d'arbres, apprentissage, questions d'appartenance

1 Introduction

1.1 Des motivations linguistiques

Depuis les années 60, Chomsky et son école ont mis en évidence l'existence d'une grammaire universelle innée dans laquelle chaque langue peut être décrite. Cette capacité à gérer les données permet de découvrir la structure des phrases et les catégories associées aux mots. Une conséquence est notre faculté à décider si une phrase est ou n'est pas grammaticalement correcte sans connaître nécessairement le sens de chaque mot qui la compose. L'information apportée par le sens d'un mot n'est pas directement utile, mais permet juste de désambiguïser plus rapidement une phrase. Nous voyons que le constituant de base de l'apprentissage est la phrase et non le mot. Le sens d'un mot pris comme suite de phonèmes est arbitraire. L'apprentissage d'une langue passe obligatoirement par l'apprentissage de son lexique. Cependant, une phrase n'est pas un amalgame de mots qui serait générée par un automate d'états finis piochant dans un lexique. Au contraire, nous ne connaissons pas toutes les phrases, par contre nous pouvons les générer. L'apprentissage du langage consiste à déterminer les structures des phrases.

Pour cela, il faut penser à une phrase comme un arbre dans lequel les mots dépendent les uns des autres. Cet arbre est construit à partir de la spécialisation de la grammaire universelle de la langue étudiée. Ainsi Christophe (Christophe, 2000) montre le rôle capital de l'intonation, ce que Pinker (Pinker, 1994) appelle les leçons de grammaire implicite infligées aux bébés en « Motherese ». L'intonation structure les mots de la phrase. Un bébé a accès non à une phrase linéaire mais bien à une information semi-structurée faite d'une suite de phonèmes séparés par des frontières marquées. A partir d'un vocabulaire, l'enfant commence à produire des phrases qu'il n'a jamais entendues. Cette généralisation est la conséquence de la construction de règles cohérentes avec les exemples. Ces règles mettent en relation des mots et permettent la production de structures similaires aux structures des phrases entendues. La phrase est comprise comme un objet semi-structuré, i.e. un arbre comme cela est illustré par la figure 1, et que les phrases sont construites par calcul.

Un autre aspect important doit être mentionné : l'environnement. Un enfant apprend une langue par interaction avec ses proches. Ce dialogue fournit des informations à l'enfant. Nous pourrions entamer une longue discussion sur la nature exacte de l'information apportée par l'environnement. Dans l'hypothèse la plus basse, celle-ci se réduit à savoir si oui ou non une phrase produite est correcte. Ainsi, il paraît raisonnable de penser que l'enfant peut conclure du fait qu'un phrase qu'il vient de prononcer n'est pas comprise, que celle-ci n'est pas correcte.

1.2 Inférence des langages réguliers d'arbres

D'un point de vue formel, nous nous plaçons dans le cadre général des grammaires régulières d'arbres. Bien que ces langages soient très utilisés en informatique, ils semblent délaissés pour le traitement automatique des langues (TAL). Pourtant, les grammaires pour le TAL manipulent très souvent à un certain niveau des langages réguliers d'arbres. Par exemple, les ensembles d'arbres d'analyse des grammaires algébriques et catégorielles sont réguliers. L'apprentissage des langages réguliers d'arbres a été étudiée d'un point de vue stochastique dans (Carrasco *et al.*, 1998). Notre cadre d'inférence grammaticale est celui de l'identification exacte à partir d'un ensemble fini d'exemples et de requêtes d'appartenance. Nous disposons d'un ensemble fini d'arbres d'un langage régulier L et nous cherchons un automate qui reconnaît L en posant des requêtes d'appartenance à un oracle. Pour l'apprentissage du langage, les exemples sont des phrases formées des mots et de leurs dépendances (i.e. des arbres de dépendances), et les requêtes d'appartenance sont obtenues par « l'oracle » formé par les parents de l'enfant.

1.3 Inférence à partir de requêtes d'appartenance

Le paradigme que nous considérons, est bien connu de la communauté puisqu'il s'agit d'apprentissage à partir de requêtes. Le lecteur pourra se référer à différents états de l'art comme celui de Angluin (Angluin, 1988). Différents types de requêtes sont pris en comptes, à savoir l'appartenance, l'équivalence, le test de sous-ensemble, ...

De nombreux travaux ont porté sur l'inférence grammaticale à partir de tests d'appartenance et d'équivalence. Angluin (Angluin, 1987) a démontré que les langages

La phrase « *Ertol est un chat* » peut être mise sous forme d'un arbre de dépendance

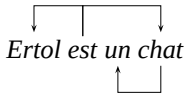

FIG. 1 – *Exemples semi-structurés*

réguliers de mots sont ainsi identifiables. Sakakibara (Sakakibara, 1988) a étudié les langages d'arbres de dérivation desétiqueté (*skeleton tree*) de grammaires algébriques.

A notre connaissance, le seul travail qui ne prend en compte que des requêtes d'appartenance pour l'inférence grammaticale est celui de Angluin (Angluin, 1981) au sujet des langages réguliers de mots. Dans un contexte linguistique, une requête d'équivalence n'a pas de sens. Un enfant ne peut pas demander à ses parents si la grammaire qu'il a élaboré est bien équivalente à celle du français.

Dans cet article, nous proposons un algorithme efficace, polynômial dans la taille des exemples, qui apprend les langages réguliers d'arbres. La convergence est assurée si l'ensemble d'exemples contient les exemples caractéristiques du langage cible. Typiquement, un exemple caractéristique est un exemple qui excite une transition de l'automate d'arbres. L'algorithme est proche de ceux décrits par Angluin dans (Angluin, 1981) et (Angluin, 1987). Mais il comporte aussi des différences notoires. Tout d'abord, la signature des arbres entraine, si l'on n'y prend garde, une explosion combinatoire. En effet, Angluin teste toutes les transitions possibles d'un automate de mots. Or, le coût de cette énumération serait exponentiel en l'arité des symboles de la signature dans le cas des arbres. De ce fait, nous remplaçons cette approche par la définition d'un ensemble suffisant de contextes pour démarrer avec une partition correcte. L'autre différence est la construction d'un contexte quand l'algorithme constate que l'automate est non-déterministe. Dans le cas des mots, il suffit de rajouter une lettre. Pour les arbres, il faut calculer un ensemble de contexte pour déterminer quel est le sous arbre coupable.

2 Langages réguliers d'arbres

Pour tout ensemble de symboles $\mathcal{F}$ et tout ensemble de variables[1] $\mathcal{X}$, les termes sont définis par : un symbole de $\mathcal{F}$ est un terme, une variable de $\mathcal{X}$ est un terme et si $\mathtt{f}$ est un symbole de $\mathcal{F}$ et $\mathtt{t}_1, \ldots, \mathtt{t}_n$ sont des termes, alors $\mathtt{f}(\mathtt{t}_1, \ldots, \mathtt{t}_n)$ est un terme. L'ensemble des termes est noté $\mathcal{T}(\mathcal{F}, \mathcal{X})$ et l'ensemble $\mathcal{T}(\mathcal{F}) = \mathcal{T}(\mathcal{F}, \emptyset)$ est l'ensembles des termes clos. Ainsi, les arbres ordonnés et étiquetés se représentent par des termes. Pour tout terme $\mathtt{t}$, l'ensemble des sous-termes de $\mathtt{t}$ est l'ensemble $\mathcal{S}(\mathtt{t})$ défini par : $\mathtt{t} \in \mathcal{S}(\mathtt{t})$ et si $\mathtt{f}(\mathtt{t}_1, \ldots, \mathtt{t}_n) \in \mathcal{S}(\mathtt{t})$ alors $\mathtt{t}_1, \ldots, \mathtt{t}_n \in \mathcal{S}(\mathtt{t})$. Pour tout ensemble de termes $\mathcal{E}$, $\mathcal{S}(\mathcal{E})$ est la réunion des ensembles des sous-termes des éléments de $\mathcal{E}$. Pour tout terme $\mathtt{t}$ et tout sous-terme $\mathtt{s}$ de $\mathtt{t}$, la profondeur $\mathtt{depth}(\mathtt{t}, \mathtt{s})$ de

1. Les variables sont considérées comme des symboles spéciaux d'arité 0.

s dans t est égale à 0 si $\mathsf{t} = \mathsf{s}$ et égale à $1 + \mathsf{depth}(\mathsf{t}, \mathsf{f}(\mathsf{t}_1, \ldots, \mathsf{t}_i, \mathsf{s}, \mathsf{t}_{i+1}, \ldots, \mathsf{t}_n))$ si $\mathsf{f}(\mathsf{t}_1, \ldots, \mathsf{t}_i, \mathsf{s}, \mathsf{t}_{i+1}, \ldots, \mathsf{t}_n)$ est un sous-terme de t.

Un *contexte* est un terme $\mathsf{c}[\diamond]$ qui contient une variable particulière $\diamond$ à une seule occurence. La variable $\diamond$ marque un emplacement vide dans le terme. La substitution de $\diamond$ par un terme s se note $\mathsf{c}[\mathsf{s}]$. On pourra utiliser cette notation pour indiquer que le terme s est un sous-terme de $\mathsf{c}[\mathsf{s}]$. Pour un ensemble donné de termes $\mathcal{E}$, $\mathsf{C}[\mathcal{E}]$ est l'ensemble des contextes obtenus à partir de tout élément t de $\mathcal{E}$, en remplaçant exactement un sous-terme de t par $\diamond$. En particulier pour tout ensemble de termes $\mathcal{E}$, $\mathsf{C}[\mathcal{E}]$ contient $\diamond$.

Un *automate d'arbres* d'états fini (NFTA) est un quadruplet $\mathcal{A} = \langle \mathcal{F}, \mathcal{Q}, \mathcal{Q}_F, \rightarrow_{\mathcal{A}} \rangle$ où $\mathcal{Q}$ est un ensemble fini d'états, $\mathcal{Q}_F \subseteq \mathcal{Q}$ est l'ensemble des états finaux et $\rightarrow_{\mathcal{A}}$ est l'ensemble des transitions. Une transition est une règle de la forme $\mathsf{f}(q_1, \ldots, q_n) \rightarrow_{\mathcal{A}} q$ où q et $q_1, \ldots, q_n$ sont des états de $\mathcal{Q}$ et f est un symbole de $\mathcal{F}$. En particulier, une transition peut être de la forme $\mathsf{a} \rightarrow_{\mathcal{A}} q$ où le symbole a a une arité nulle.

Un automate d'arbres d'états fini est *déterministe* (DFTA) s'il n'existe pas deux transitions distinctes possédant des parties gauches identiques. Pour ces automates, la relation $\rightarrow_{\mathcal{A}}$ est le graphe d'une fonction notée $\delta_{\mathcal{A}}$. Comme dans le cas des automates de mots, il a été montré que pour tout langage reconnu par un automate non-déterministe, il existe un automate déterministe qui le reconnaît (Comon *et al.*, 1997).

La dérivation simple $\rightarrow_{\mathcal{A}}$ est définie par $\mathsf{t} \rightarrow_{\mathcal{A}} \mathsf{s}$ si et seulement si il existe une transition $\mathsf{f}(q_1, \ldots, q_n) \rightarrow_{\mathcal{A}} q$ telle que $\mathsf{t} = \mathsf{u}[\mathsf{f}(q_1, \ldots, q_n)]$ et $\mathsf{s} = \mathsf{u}[q]$. On notera que $\rightarrow_{\mathcal{A}} \subseteq \mathcal{T}(\mathcal{F}, \mathcal{Q}) \times \mathcal{T}(\mathcal{F}, \mathcal{Q})$, et que les états de $\mathcal{Q}$ sont vus comme des variables.

La relation de dérivation $\rightarrow_{*,\mathcal{A}}$ est la clôture réflexive et transitive de $\rightarrow_{\mathcal{A}}$. Le langage d'arbres *reconnu* par $\mathcal{A}$ est l'ensemble $\mathcal{L}_{\mathcal{A}} = \{\mathsf{t} \in \mathcal{T}(\mathcal{F}) \ : \ \mathsf{t} \rightarrow_{*,\mathcal{A}} q_f \text{ et } q_f \in \mathcal{Q}_F\}$.

Soit un automate d'arbres $\mathcal{A} = \langle \mathcal{F}, \mathcal{Q}, \mathcal{Q}_F, \rightarrow_{\mathcal{A}} \rangle$ et une fonction surjective ϕ de $\mathcal{Q}$ dans $\mathcal{Q}'$, où $\mathcal{Q}' = \phi(\mathcal{Q})$ est un ensemble fini quelconque. ϕ définit un automate $\mathcal{A}' = \phi(\mathcal{A})$, avec $\mathcal{A}' = \langle \mathcal{F}, \phi(\mathcal{Q}), \phi(\mathcal{Q}_F), \phi(\rightarrow_{\mathcal{A}}) \rangle$ et $\phi(\rightarrow_{\mathcal{A}})$ est l'ensemble des règles de la forme $\mathsf{f}(\phi(q_1), \ldots, \phi(q_n)) \rightarrow_{\mathcal{A}'} \phi(q)$, pour toutes règles $\mathsf{f}(q_1, \ldots, q_n) \rightarrow_{\mathcal{A}} q$ dans $\rightarrow_{\mathcal{A}}$. Pour tout arbre t, si $\mathsf{t} \rightarrow_{*,\mathcal{A}} q$ alors $\mathsf{t} \rightarrow_{*,\mathcal{A}'} \phi(q)$, ce qui implique que $\mathcal{L}_{\mathcal{A}} \subseteq \mathcal{L}_{\mathcal{A}'}$. Si ϕ est bijective, on dit que ϕ est un renommage des états et l'on a : $\mathcal{L}_{\mathcal{A}} = \mathcal{L}_{\mathcal{A}'}$.

D'après le théorème de Myhill-Nerode (Comon *et al.*, 1997), tout langage $\mathcal{L}$, définit une relation d'équivalence $\equiv_{\mathcal{L}}$ par : pour tout t et tout s dans $\mathcal{S}(\mathcal{L})$, $\mathsf{t} \equiv_{\mathcal{L}} \mathsf{s}$ si et seulement si pour tout contexte $\mathsf{c}[\diamond]$ dans $\mathsf{C}[\mathcal{T}(\mathcal{F})], \mathsf{c}[\mathsf{t}] \in \mathcal{L} \Leftrightarrow \mathsf{c}[\mathsf{s}] \in \mathcal{L}$. Le théorème de Myhill-Nerode implique que pour tout langage $\mathcal{L}$, il existe un automate DFTA $\mathcal{A}$, nommé *automate minimal* de $\mathcal{L}$ tel que :

- $\mathcal{L} = \mathcal{L}_{\mathcal{A}}$,
- pour tout automate $\mathcal{A}'$ tel que $\mathcal{L} = \mathcal{L}_{\mathcal{A}'}$, le nombre d'états de $\mathcal{A}'$ est supérieur ou égal au nombre d'états de $\mathcal{A}$.

De plus, cet automate est unique à un renommage des états près, et pour tout arbre t et tout arbre t' on a : $\delta_{\mathcal{A}}(\mathsf{t}) = \delta_{\mathcal{A}}(\mathsf{t}')$ si et seulement si $\mathsf{t} \equiv_{\mathcal{L}} \mathsf{t}'$.

3 Ensemble caractéristique

Si $\mathcal{L}$ est un langage régulier d'arbres quelconque et $\mathcal{C}$ un sous-ensemble de $\mathcal{L}$, on dira que $\mathcal{C}$ est un *ensemble caractéristique* de $\mathcal{L}$ si pour un automate minimal $\mathcal{A}$ de $\mathcal{L}$, pour toute règle $\mathtt{f}(q_1, \ldots, q_n) \rightarrow_{\mathcal{A}} q$ de $\mathcal{A}$, il existe un arbre $\mathtt{t} = \mathtt{f}(\mathtt{t}_1, \ldots, \mathtt{t}_n)$ dans $\mathcal{S}(\mathcal{L})$ tel que $\forall 1 \leq i \leq n$, $\delta_{\mathcal{A}}(\mathtt{t}_i) = q_i$[2]. Puisque l'automate minimal d'un langage est unique à un renommage des états près, la définition d'ensemble caractéristique est indépendante du choix d'un automate minimal particulier.

De manière informelle, un ensemble caractéristique d'un langage $\mathcal{L}$ est un sous ensemble fini $\mathcal{C}$ tel que pour automate minimal de $\mathcal{L}$, chaque règle a été utilisée pour produire une fois au moins un élément de $\mathcal{C}$. Enfin tout ensemble fini contenant un ensemble caractéristique est aussi caractéristique.

Exemple 1 *Soit l'automate minimal $\mathcal{A}_1$ défini par les règles :*

$$\begin{array}{rcl} c & \rightarrow_{\mathcal{A}_1} & q \\ b & \rightarrow_{\mathcal{A}_1} & q_1 \\ a(q_1, q) & \rightarrow_{\mathcal{A}_1} & q_0 \\ a(q_0, q) & \rightarrow_{\mathcal{A}_1} & q_0 \end{array}$$

où q_0 est le seul état final. On a $\mathcal{L}_{\mathcal{A}_1} = \{\underbrace{a(\ldots(a}_{n}(b, c))\ldots, c) \ : \ n > 0\}$ et pour tout $n > 1$, le singleton $\{\underbrace{a(\ldots(a}_{n}(b, c))\ldots, c)\}$ est un ensemble caractéristique de $\mathcal{L}_{\mathcal{A}_1}$.

Soit pour tout $m > 0$, l'automate $\mathcal{A}_m$ suivant :

$$\begin{array}{rcl} c & \rightarrow_{\mathcal{A}_m} & q \\ b & \rightarrow_{\mathcal{A}_m} & q_m \\ a(q_m, q) & \rightarrow_{\mathcal{A}_m} & q_{m-1} \\ & \vdots & \\ a(q_1, q) & \rightarrow_{\mathcal{A}_m} & q_0 \\ a(q_0, q) & \rightarrow_{\mathcal{A}_m} & q_0 \end{array}$$

où q_0 est le seul état final. Alors $\mathcal{L}_{\mathcal{A}_m} = \{\underbrace{a(\ldots(a}_{n}(b, c))\ldots, c) \ : \ n > m\}$. Pour tout n, le singleton $\{\underbrace{a(\ldots(a}_{n}(b, c))\ldots, c)\}$ est un ensemble caractéristique de tout langage $\mathcal{L}_{\mathcal{A}_m}$, pour $0 < m < n$.

4 Apprentissage

Nous nous intéressons à l'identification de la classe des langages réguliers d'arbres à partir d'un ensemble caractéristique. On dira qu'un algorithme I est un algortihme

2. Puisque $\mathcal{A}$ est déterministe, on a $\delta_{\mathcal{A}}(\mathtt{t}) = q$.

d'apprentissage pour la classe des langages réguliers d'arbres si : pour tout langage $\mathcal{L}$, si I reçoit un ensemble caractéristique en entrée, alors A donne un automate minimal $\mathcal{A}$ en sortie avec $\mathcal{L} = \mathcal{L}_{\mathcal{A}}$. D'après l'exemple qui précède, il ne peut exister d'algorithme d'apprentissage ainsi défini pour la classe des langages réguliers d'arbres. En effet, si un tel algorithme existe, il doit être capable de trouver un unique automate pour l'entrée $\{\underbrace{a(\ldots(a}_{n}(b,c))\ldots,c)\}$, où n est un entier quelconque. Or pour tout $n > 1$, $\{\underbrace{a(\ldots(a}_{n}(b,c))\ldots,c)\}$ est un ensemble caractéristique de plusieurs langages (l'ensemble des $\mathcal{L}_{\mathcal{A}_m}$, pour $0 < m < n$). Pour résoudre le problème d'apprentissage, I doit pouvoir bénéficier d'information supplémentaire concernant le langage à identifier. Cette information supplémentaire lui est fournie grâce à une interaction avec un oracle O. L'oracle O connaît le langage cherché et il est capable de répondre à des question d'appartenance d'un terme proposé par I à ce langage. Ainsi I prend en entrée un ensemble caractéristique d'un langage inconnu $\mathcal{L}$, il pose une série des questions d'appartenance de termes t qu'il choisit à un oracle O et l'oracle lui répond $vrai$ ou $faux$ suivant que t appartient à $\mathcal{L}$ ou pas. Le processus s'arrête quand I n'a plus de question à soumetre à O et I fournit un automate $\mathcal{A}$ en sortie.

4.1 Réduction du problème

Classiquement, nous allons montrer que le problème d'identification d'un langage régulier à partir d'un ensemble caractéristique $\mathcal{C}$ peut se réduire à un problème de partitionnement de $\mathcal{C}$. Soit $\mathcal{L}$ un langage régulier d'arbres quelconque et $\mathcal{C}$ un ensemble caractéristique de $\mathcal{L}$. Si $\equiv$ est une relation d'équivalence sur $\mathcal{C}$ alors $\equiv$ définit un automate $\mathcal{A}_{\mathcal{C},\equiv}$ par : $\mathcal{A}_{\mathcal{C},\equiv} = \langle \mathcal{F}_{\mathcal{C},\equiv}, \mathcal{Q}_{\mathcal{C},\equiv}, \mathcal{Q}_{F,\mathcal{C},\equiv}, \rightarrow_{\mathcal{A}_{\mathcal{C},\equiv}} \rangle$, où

- $\mathcal{F}_{\mathcal{C},\equiv}$ est l'ensemble des symboles étiquetant les arbres de $\mathcal{C}$,
- $\mathcal{Q}_{\mathcal{C},\equiv} = \{[\mathtt{t}], \mathtt{t} \in \mathcal{S}(\mathcal{C})\}$, où $[\mathtt{t}]$ est la classe d'équivalence de $\mathtt{t}$,
- $\mathcal{Q}_{F,\mathcal{C},\equiv} = \{[\mathtt{t}], \mathtt{t} \in \mathcal{C}\}$,
- $\rightarrow_{\mathcal{A}_{\mathcal{C},\equiv}} = \{\mathtt{f}([\mathtt{t}_1],\ldots,[\mathtt{t}_n]) \rightarrow_{\mathcal{A}_{\mathcal{C},\equiv}} [\mathtt{f}(\mathtt{t}_1,\ldots,\mathtt{t}_n)], \mathtt{f}(\mathtt{t}_1,\ldots,\mathtt{t}_n) \in \mathcal{S}(\mathcal{C})\}$.

On notera que dans le cas général, $\mathcal{A}_{\mathcal{C},\equiv}$ n'a aucune raison particulière d'être déterministe.

Lemme 2
Pour tout langage $\mathcal{L}$, pour tout ensemble caractéristique $\mathcal{C}$ de $\mathcal{L}$, l'automate $\mathcal{A}_{\mathcal{C},\equiv_{\mathcal{L},\mathcal{C}}}$ est un automate minimal de $\mathcal{L}$, où $\equiv_{\mathcal{L},\mathcal{C}}$ est la restriction de $\equiv_{\mathcal{L}}$ sur $\mathcal{S}(\mathcal{C})$.

Démonstration. Considérons un automate minimal $\mathcal{A} = \langle \mathcal{F}, \mathcal{Q}, \mathcal{Q}_F, \rightarrow_{\mathcal{A}} \rangle$ de $\mathcal{L}$ et la fonction $\psi : \mathcal{Q} \mapsto \{[\mathtt{t}], \mathtt{t} \in \mathcal{S}(\mathcal{C})\}$ définie par $\psi(\delta_{\mathcal{A}}(\mathtt{t})) = [\mathtt{t}]$, pour tout $\mathtt{t}$ dans $\mathcal{S}(\mathcal{C})$. Puisque $\mathcal{C}$ est un ensemble caractéristique, à chaque état q de $\mathcal{Q}$, correspond un terme $\mathtt{t}$ dans $\mathcal{S}(\mathcal{C})$ tel que $\delta_{\mathcal{A}}(t) = q$ (chaque état q apparaît dans la partie droite d'une règle de $\rightarrow_{\mathcal{A}}$). Alors ψ est bien définie et on vérifie facilement que ψ est un renommage des états de $\mathcal{A}$.

D'après le lemme précédent, qui est une conséquence directe du théorème de Myhill-Nerode, le problème de construction d'un automate minimal pour un langage $\mathcal{L}$ à partir

d'un ensemble caractéristique $\mathcal{C}$ peut se réduire au problème de partitionnement de $\mathcal{C}$ suivant la relation $\equiv_{\mathcal{L},\mathcal{C}}$. En effet le théorème de Myhill-Nerode caractérise tout langage régulier d'arbres (résultat équivalent sur les mots) en terme d'ensemble fini de classes d'équivalence. Chaque classe correspondant à un état d'un automate minimal pour le langage, reconstruire cet automate revient à reconstruire la relation d'équivalence sur un ensemble caractéristique.

4.2 Tableau d'observation

Comme dans (Angluin, 1987), l'information traitée par l'algorithme d'apprentissage sera stockée dans un tableau nommé *tableau d'observation*.

Soit $\mathcal{L}$ un langage quelconque, $\mathcal{E}$ un ensemble quelconque de termes fini et F un ensemble fini quelconque de contextes. Le *tableau d'observation* $T_{\mathcal{L}}(\mathcal{E},\mathsf{F})$ relatif à $\mathcal{L}$, $\mathcal{E}$ and $\mathcal{F}$, est défini par :

- chaque ligne correspond à un élément de $\mathcal{E}$,
- chaque colonne correspond à un élément de F,
- pour chaque t dans $\mathcal{E}$ et chaque $\mathsf{c}[\diamond]$ dans F, $T_{\mathcal{L}}(\mathsf{t},\mathsf{c}[\diamond]) = vrai$ si $\mathsf{c}[\mathsf{t}]$ est dans $\mathcal{L}$ et $T_{\mathcal{L}}(\mathsf{t},\mathsf{c}[\diamond]) = faux$ sinon.

Un tableau d'observation $T_{\mathcal{L}}(\mathcal{E},\mathsf{F})$ définit une relation d'équivalence $\equiv_{T_{\mathcal{L}}(\mathcal{E},\mathsf{F})}$ sur $\mathcal{E}$ par : pour tout t et tout t' dans $\mathcal{E}$, $\mathsf{t} \equiv_{T_{\mathcal{L}}(\mathcal{E},\mathsf{F})} \mathsf{t}'$ si et seulement si les mots $ligne(\mathsf{t})$ et $ligne(\mathsf{t}')$ construits sur l'alphabet $\{vrai, faux\}^*$ et correspondant respectivement aux lignes de t et t' dans $T_{\mathcal{L}}(\mathcal{E},\mathsf{F})$ sont égaux.

Remarque 3 *Pour tous termes* t *et* t' *de* $\mathcal{E}$*, si* $\mathsf{t} \not\equiv_{T_{\mathcal{L}}(\mathcal{E},\mathsf{F})} \mathsf{t}$*, alors* $\mathsf{t} \not\equiv_{\mathcal{L},\mathsf{F}} \mathsf{t}'$*. En d'autres termes, la partition* $\equiv_{\mathcal{L},\mathsf{F}}$ *raffine toujours la partition* $\equiv_{T_{\mathcal{L}}(\mathcal{E},\mathsf{F})}$.

Un tableau d'observation $T_{\mathcal{L}}(\mathcal{E},\mathsf{F})$ est dit *cohérent* lorsque pour tous termes $\mathsf{f}(\mathsf{t}_1,\ldots,\mathsf{t}_n)$ et $\mathsf{f}(\mathsf{t}'_1,\ldots,\mathsf{t}'_n)$ de $\mathcal{S}(\mathcal{E})$, si pour tout $1 \leq j \leq n$ on a :

$$\mathsf{t}_j \equiv_{T_{\mathcal{L}}(\mathcal{E},\mathsf{F})} \mathsf{t}'_j$$

alors

$$\mathsf{f}(\mathsf{t}_1,\ldots,\mathsf{t}_n) \equiv_{T_{\mathcal{L}}(\mathcal{E},\mathsf{F})} \mathsf{f}(\mathsf{t}'_1,\ldots,\mathsf{t}'_n).$$

Nous allons maintenant définir notre algorithme d'apprentissage. Celui-ci prend un ensemble caractéristique d'un langage inconnu en entrée, puis construit une série de tableaux d'observation à l'aide de l'oracle. L'algorithme termine lorsque le dernier tableau construit est cohérent.

4.3 L'algorithme $IRTL$

Initialisation : $\mathsf{F} = \mathsf{C}[\mathcal{C}]$;
Construction du tableau $T_{\mathcal{L}}(\mathcal{S}(\mathcal{C}),\mathsf{F})$ à l'aide de questions d'appartenance à O;
TANTQUE
$T_{\mathcal{L}}(\mathcal{S}(\mathcal{C}),\mathsf{F})$ il existe $\mathsf{f}(\mathsf{t}_1,\ldots,\mathsf{t}_n)$ et $\mathsf{f}(\mathsf{t}'_1,\ldots,\mathsf{t}'_n)$ dans $\mathcal{S}(\mathcal{E})$ tels que $\mathsf{f}(\mathsf{t}_1,\ldots,\mathsf{t}_n) \not\equiv_{T_{\mathcal{L}}(\mathcal{S}(\mathcal{C}),\mathsf{F})} \mathsf{f}(\mathsf{t}'_1,\ldots,\mathsf{t}'_n)$ et pour tout $1 \leq i \leq n$,

$\mathsf{t}_i \equiv_{T_\mathcal{L}(\mathcal{S}(\mathcal{C}),\mathsf{F})} \mathsf{t}'_i$
FAIRE
Trouver un contexte $\mathsf{c}[\diamond]$ dans F tel que $\mathsf{c}[\mathsf{f}(\mathsf{t}_1,\ldots,\mathsf{t}_n)] \in \mathcal{L}$
et $\mathsf{c}[\mathsf{f}(\mathsf{t}'_1,\ldots,\mathsf{t}'_n)] \notin \mathcal{L}$;
$\mathsf{F} = \mathsf{F} \cup \{\mathsf{c}[\mathsf{f}(\mathsf{t}_1,\ldots,\mathsf{t}_{i-1},\diamond,\mathsf{t}_{i+1},\ldots,\mathsf{t}_n)], 1 \leq i \leq n\}$;
Construction de $T_\mathcal{L}(\mathcal{S}(\mathcal{C}),\mathsf{F})$ à l'aide de questions d'appartenance à O;
FINTANTQUE
Retourner l'automate $\mathcal{A}_{\equiv_{T_\mathcal{L}(\mathcal{S}(\mathcal{C}),\mathsf{F})}}$.

4.4 Preuve de correction

Nous allons montrer que la construction itérative du tableau et la verification de la cohérence à chaque étape permet de raffiner la partition jusqu'à l'obtention de $\equiv_{\mathcal{L},\mathcal{C}}$.

Lemme 4
Soient $\mathcal{L}$ un langage régulier d'arbres, $\mathcal{E}$ un ensemble de termes quelconque, $\mathcal{C}$ un ensemble caractéristique pour $\mathcal{L}$, F un ensemble de contextes contenant $\mathsf{C}[\mathcal{C}]$ et $\mathcal{A}$ un automate minimal pour $\mathcal{L}$. Pour toute paire de termes t et t' de $\mathcal{E}$, si $\mathsf{t} \equiv_{T_\mathcal{L}(\mathcal{E},\mathsf{F})} \mathsf{t}'$, alors pour toute règle de la forme $\mathsf{f}(q_1,\ldots,q_{i-1},\delta_\mathcal{A}(\mathsf{t}),q_{i+1},\ldots,q_n) \to_\mathcal{A} r$ dans $\mathcal{A}$, il existe également un règle de la forme $\mathsf{f}(q_1,\ldots,q_{i-1},\delta_\mathcal{A}(\mathsf{t}'),q_{i+1},\ldots,q_n) \to_\mathcal{A} r'$ dans $\mathcal{A}$.

Démonstration. Supposons $\mathsf{t} \equiv_{T_\mathcal{L}(\mathcal{E},\mathsf{F})} \mathsf{t}'$ dans $\mathcal{E}$ et $\mathsf{f}(q_1,\ldots,q_{i-1},\delta_\mathcal{A}(\mathsf{t}),q_{i+1},\ldots,q_n) \to_\mathcal{A} r$ une règle de $\mathcal{A}$. Puisque $\mathcal{C}$ est caractéristique et F contient $\mathsf{C}[\mathcal{C}]$, il existe un contexte de la forme $\mathsf{c}[\mathsf{f}(\mathsf{t}_1,\ldots,\mathsf{t}_{i-1},\diamond,\mathsf{t}_{i+1},\ldots,\mathsf{t}_n)]$ dans F, où $\delta_\mathcal{A}(\mathsf{t}_j) = q_j$, $\forall 1 \leq j \neq i \leq n$. D'après le défintion de $T_\mathcal{L}(\mathcal{E},\mathsf{F})$, on a $T_\mathcal{L}(\mathsf{t},\mathsf{c}[\mathsf{f}(\mathsf{t}_1,\ldots,\mathsf{t}_{i-1},\diamond,\mathsf{t}_{i+1},\ldots,\mathsf{t}_n)]) = vrai$. De plus, $\mathsf{t} \equiv_{T_\mathcal{L}(\mathcal{E},\mathsf{F})} \mathsf{t}'$ implique que $T_\mathcal{L}(\mathsf{t}',\mathsf{c}[\mathsf{f}(\mathsf{t}_1,\ldots,\mathsf{t}_{i-1},\diamond,\mathsf{t}_{i+1},\ldots,\mathsf{t}_n)]) = vrai$. Puisque $\mathsf{c}[\mathsf{f}(\mathsf{t}_1,\ldots,\mathsf{t}_{i-1},\mathsf{t}',\mathsf{t}_{i+1},\ldots,\mathsf{t}_n)]$ est dans $\mathcal{L}$, il doit exister une règle $\mathsf{f}(q_1,\ldots,q_{i-1},\delta_\mathcal{A}(\mathsf{t}),q_{i+1},\ldots,q_n) \to_\mathcal{A} r'$ dans $\mathcal{A}$.

Théorème 5
Soit $\mathcal{L}$ un langage régulier d'arbres, $\mathcal{C}$ un ensemble caractéristique de $\mathcal{L}$ et F un ensemble de contextes contenant $\mathsf{C}[\mathcal{C}]$. Si $\equiv_{T_\mathcal{L}(\mathcal{C},\mathsf{F})}$ n'est pas égale à $\equiv_{\mathcal{L},\mathcal{C}}$, il existe deux termes $\mathsf{f}(\mathsf{t}_1,\ldots,\mathsf{t}_{i-1},\mathsf{t},\mathsf{t}_{i+1},\ldots,\mathsf{t}_n)$ et $\mathsf{f}(\mathsf{t}'_1,\ldots,\mathsf{t}'_{i-1},\mathsf{t}',\mathsf{t}'_{i+1},\ldots,\mathsf{t}'_n)$ dans $\mathcal{S}(\mathcal{C})$ tels que :

- $\mathsf{f}(\mathsf{t}_1,\ldots,\mathsf{t}_{i-1},\mathsf{t},\mathsf{t}_{i+1},\ldots,\mathsf{t}_n) \not\equiv_{T_\mathcal{L}(\mathcal{C},\mathsf{F})} \mathsf{f}(\mathsf{t}'_1,\ldots,\mathsf{t}'_{i-1},\mathsf{t}',\mathsf{t}'_{i+1},\ldots,\mathsf{t}'_n)$,
- $\mathsf{t}_j \equiv_{\mathcal{L},\mathcal{C}} \mathsf{t}'_j$, *pour tout* $1 \leq j \neq i \leq n$,
- $\mathsf{t} \equiv_{T_\mathcal{L}(\mathcal{C},\mathsf{F})} \mathsf{t}'$.

Démonstration. Si $\equiv_{T_\mathcal{L}(\mathcal{C},\mathsf{F})}$ n'est pas égale à $\equiv_{\mathcal{L},\mathcal{C}}$, il existe deux termes t et t' dans $\mathcal{S}(\mathcal{C})$ tels que $\mathsf{t} \equiv_{T_\mathcal{L}(\mathcal{C},\mathsf{F})} \mathsf{t}'$ et $\mathsf{t} \not\equiv_{\mathcal{L},\mathcal{C}} \mathsf{t}'$. $\mathsf{t} \not\equiv_{\mathcal{L},\mathcal{C}} \mathsf{t}'$ implique l'existence d'un contexte $\mathsf{c}[\diamond]$ tel que $\mathsf{c}[\mathsf{t}] \in \mathcal{L}$ et $\mathsf{c}[\mathsf{t}'] \notin \mathcal{L}$.
Définissons le contexte $\mathsf{c}[\diamond]$ par :

- il existe deux termes s et s' dans $\mathcal{S}(\mathcal{C})$ tels que
  - $\mathsf{s} \equiv_{T_\mathcal{L}(\mathcal{C},\mathsf{F})} \mathsf{s}'$

- $\mathsf{c}[\mathsf{s}] \in \mathcal{L}$ et $\mathsf{c}[\mathsf{s}'] \notin \mathcal{L}$

– $\mathsf{depth}(\mathsf{c}[\diamond]) = \min\{\mathsf{depth}(\mathsf{c}'[\diamond]), \exists \mathsf{u}, \mathsf{u}' \in \mathcal{S}(\mathcal{C}), \mathsf{u} \equiv_{T_{\mathcal{L}}(\mathcal{C},\mathsf{F})} \mathsf{u}', \mathsf{c}'[\mathsf{u}] \in \mathcal{L}$ et $\mathsf{c}'[\mathsf{u}'] \notin \mathcal{L}\}$

Montrons tout d'abord que $\mathsf{depth}(\mathsf{c}[\diamond]) > 0$. Si $\mathsf{depth}(\mathsf{c}[\diamond]) = 0$, on a $\mathsf{c}[\diamond] = \diamond$. Or $\diamond$ est un contexte de F, donc si $\mathsf{s} \in \mathcal{L}$ et $\mathsf{s}' \notin \mathcal{L}$, alors $\mathsf{s} \not\equiv_{T_{\mathcal{L}}(\mathcal{C},\mathsf{F})} \mathsf{s}'$ ce qui montre une contradiction. Par conséquent $\mathsf{c}[\diamond]$ peut s'écrire $\mathsf{d}[\mathsf{f}(\mathsf{s}_1, \ldots, \mathsf{s}_{i-1}, \diamond, \mathsf{s}_i, \ldots, \mathsf{s}_n)]$, où $\mathsf{s}_1, \ldots, \mathsf{s}_{i-1}, \mathsf{s}_{i+1}, \ldots, \mathsf{s}_n$ sont des termes et $\mathsf{d}[\diamond]$ un contexte tel que $\mathsf{depth}(\mathsf{c}[\diamond]) = \mathsf{depth}(\mathsf{d}[\diamond]) + 1$. Puisque $\mathsf{d}[\mathsf{f}(\mathsf{s}_1, \ldots, \mathsf{s}_{i-1}, \mathsf{s}, \mathsf{s}_{i+1}, \ldots, \mathsf{s}_n)]$ est dans $\mathcal{L}$, dans tout automate $\mathcal{A}$ minimal pour $\mathcal{L}$, il existe une règle $\mathsf{f}(q_1, \ldots, q_{i-1}, \delta_{\mathcal{A}}(\mathsf{s}), q_{i+1}, \ldots, q_n) \to_{\mathcal{A}} r$, avec $q_j = \delta_{\mathcal{A}}(\mathsf{s}_j)$, pour tout $1 \leq j \neq i \leq n$. D'après le lemme 4, il existe aussi une règle $\mathsf{f}(q_1, \ldots, q_{i-1}, \delta_{\mathcal{A}}(\mathsf{s}'), q_{i+1}, \ldots, q_n) \to_{\mathcal{A}} r'$ dans $\mathcal{A}$, avec $q_j = \delta_{\mathcal{A}}(\mathsf{s}_j)$, pour tout $1 \leq j \neq i \leq n$. Si $r = r'$, alors on a $\delta_{\mathcal{A}}(\mathsf{f}(\mathsf{s}_1, \ldots, \mathsf{s}_{i-1}, \mathsf{s}, \mathsf{s}_{i+1}, \ldots, \mathsf{s}_n)) = \delta_{\mathcal{A}}(\mathsf{f}(\mathsf{s}_1, \ldots, \mathsf{s}_{i-1}, \mathsf{s}', \mathsf{s}_{i+1}, \ldots, \mathsf{s}_n))$ et $\mathsf{d}[\mathsf{f}(\mathsf{s}_1, \ldots, \mathsf{s}_{i-1}, \mathsf{s}, \mathsf{s}_{i+1}, \ldots, \mathsf{s}_n)]$ est dans $\mathcal{L}$ implique que $\mathsf{d}[\mathsf{f}(\mathsf{s}_1, \ldots, \mathsf{s}_{i-1}, \mathsf{s}, \mathsf{s}_{i+1}, \ldots, \mathsf{s}_n)]$ est dans $\mathcal{L}$ ce qui contredit l'hypothèse. $\mathcal{C}$ est un ensemble caractéristique, alors aux règles $\mathsf{f}(q_1, \ldots, q_{i-1}, \delta_{\mathcal{A}}(\mathsf{s}), q_{i+1}, \ldots, q_n) \to_{\mathcal{A}} r$ et $\mathsf{f}(q_1, \ldots, q_{i-1}, \delta_{\mathcal{A}}(\mathsf{s}'), q_{i+1}, \ldots, q_n) \to_{\mathcal{A}} r'$ correspondent deux termes $\mathsf{f}(\mathsf{t}_1, \ldots, \mathsf{t}_{i-1}, \mathsf{t}, \mathsf{t}_{i+1}, \ldots, \mathsf{t}_n)$, $\mathsf{f}(\mathsf{t}'_1, \ldots, \mathsf{t}'_{i-1}, \mathsf{t}', \mathsf{t}'_{i+1}, \ldots, \mathsf{t}'_n)$ dans $\mathcal{S}(\mathcal{C})$, avec $\delta_{\mathcal{A}}(\mathsf{t}_j) = \delta_{\mathcal{A}}(\mathsf{t}'_j) = q_j$, pour tout $1 \leq j \neq i \leq n$ et $\delta_{\mathcal{A}}(\mathsf{f}(\mathsf{t}_1, \ldots, \mathsf{t}_{i-1}, \mathsf{t}, \mathsf{t}_{i+1}, \ldots, \mathsf{t}_n)) = r$ et $\delta_{\mathcal{A}}(\mathsf{f}(\mathsf{t}'_1, \ldots, \mathsf{t}'_{i-1}, \mathsf{t}', \mathsf{t}'_{i+1}, \ldots, \mathsf{t}'_n)) = r'$. Finalement supposons que $\mathsf{f}(\mathsf{t}_1, \ldots, \mathsf{t}_{i-1}, \mathsf{t}, \mathsf{t}_{i+1}, \ldots, \mathsf{t}_n) \equiv_{T_{\mathcal{L}}(\mathcal{C},\mathsf{F})} \mathsf{f}(\mathsf{t}'_1, \ldots, \mathsf{t}'_{i-1}, \mathsf{t}', \mathsf{t}'_{i+1}, \ldots, \mathsf{t}'_n)$ et montrons que le contexte $\mathsf{d}[\diamond]$ contredit la minimalité de $\mathsf{depth}(\mathsf{c}[\diamond])$. En effet :

- $\mathsf{f}(\mathsf{t}_1, \ldots, \mathsf{t}_{i-1}, \mathsf{t}, \mathsf{t}_{i+1}, \ldots, \mathsf{t}_n) \equiv_{T_{\mathcal{L}}(\mathcal{C},\mathsf{F})} \mathsf{f}(\mathsf{t}'_1, \ldots, \mathsf{t}'_{i-1}, \mathsf{t}', \mathsf{t}'_{i+1}, \ldots, \mathsf{t}'_n)$
- $\mathsf{d}[\mathsf{f}(\mathsf{t}_1, \ldots, \mathsf{t}_{i-1}, \mathsf{t}, \mathsf{t}_{i+1}, \ldots, \mathsf{t}_n)] \in \mathcal{L}$ et $\mathsf{d}[\mathsf{f}(\mathsf{t}'_1, \ldots, \mathsf{t}'_{i-1}, \mathsf{t}', \mathsf{t}'_{i+1}, \ldots, \mathsf{t}'_n)] \notin \mathcal{L}$
- $\mathsf{depth}(\mathsf{c}[\diamond]) > \mathsf{depth}(\mathsf{d}[\diamond])$

On a donc bien $\mathsf{f}(\mathsf{t}_1, \ldots, \mathsf{t}_{i-1}, \mathsf{t}, \mathsf{t}_{i+1}, \ldots, \mathsf{t}_n) \not\equiv_{T_{\mathcal{L}}(\mathcal{C},\mathsf{F})} \mathsf{f}(\mathsf{t}'_1, \ldots, \mathsf{t}'_{i-1}, \mathsf{t}', \mathsf{t}'_{i+1}, \ldots, \mathsf{t}'_n)$ ce qui permet de conclure.

Remarque 6 *Si $\equiv_{T_{\mathcal{L}}(\mathcal{C},\mathsf{F})}$ n'est pas égale à $\equiv_{\mathcal{L},\mathcal{C}}$, le théorème précédent implique que $T_{\mathcal{L}}(\mathcal{C},\mathsf{F})$ n'est pas cohérent. En d'autres termes, si l'automate $T_{\mathcal{L}}(\mathcal{C},\mathsf{F})$ est cohérent, le langage est appris et l'algorithme s'arrête.*

Corollaire 7
L'algorithme $IRTL$ apprend la classe des langages réguliers d'arbres.

Démonstration. L'algorithme commence par la construction de $T_{\mathcal{L}}(\mathcal{E},\mathsf{F}) = T_{\mathcal{L}}(\mathcal{S}(\mathcal{C}), \mathsf{C}[\mathcal{C}])$ puis entre dans une boucle. D'après le théorème 5, si le programme sort de cette boucle, le tableau est cohérent et l'on a $\equiv_{T_{\mathcal{L}}(\mathcal{C},\mathsf{F})}$ est égale à $\equiv_{\mathcal{L},\mathcal{C}}$, et d'après le lemme 2, l'automate $\mathcal{A}_{\mathcal{C},\equiv_{\mathcal{L},\mathcal{C}}}$ est un automate minimal pour $\mathcal{L}$. Montrons alors que l'algorithme termine. La boucle collecte tous les couples de termes $\mathsf{f}(\mathsf{t}_1, \ldots, \mathsf{t}_n)$ et $\mathsf{f}(\mathsf{t}'_1, \ldots, \mathsf{t}'_n)$ dans $\mathcal{C}$, tels qu'il existe un contexte $\mathsf{c}[\diamond]$ dans F avec pour tout $1 \leq i \leq n$, $\mathsf{t}_i \equiv_{T_{\mathcal{L}}(\mathcal{C},\mathsf{F})} \mathsf{t}'_i$, $\mathsf{c}[\mathsf{f}(\mathsf{t}_1, \ldots, \mathsf{t}_n)] \in \mathcal{L}$ et $\mathsf{c}[\mathsf{f}(\mathsf{t}'_1, \ldots, \mathsf{t}'_n)] \notin \mathcal{L}$. D'après le théorème 5, parmi ces couples, il en est un $\mathsf{f}(\mathsf{t}_1, \ldots, \mathsf{t}_{i-1}, \mathsf{t}, \mathsf{t}_{i+1}, \ldots, \mathsf{t}_n)$ et $\mathsf{f}(\mathsf{t}'_1, \ldots, \mathsf{t}'_{i-1}, \mathsf{t}', \mathsf{t}'_{i+1}, \ldots, \mathsf{t}'_n)$, tel que $\mathsf{t}_j \equiv_{\mathcal{L},\mathcal{C}} \mathsf{t}'_j$, pour tout $1 \leq j \neq i \leq n$. $\mathsf{c}[\mathsf{f}(\mathsf{t}'_1, \ldots, \mathsf{t}'_{i-1}, \mathsf{t}', \mathsf{t}'_{i+1}, \ldots, \mathsf{t}'_n)] \notin \mathcal{L}$ est alors equivalent à $\mathsf{c}[\mathsf{f}(\mathsf{t}_1, \ldots, \mathsf{t}_{i-1}, \mathsf{t}, \mathsf{t}_{i+1}, \ldots, \mathsf{t}_n)] \notin \mathcal{L}$. Par définition d'$IRTL$, lorsque la boucle se termine, le contexte $\mathsf{c}[\mathsf{f}(\mathsf{t}_1, \ldots, \mathsf{t}_{i-1}, \diamond, \mathsf{t}_{i+1}, \ldots, \mathsf{t}_n)]$ a été ajouté à F, ce qui

différencie les colonnes de t et de t' dans $T_{\mathcal{L}}(\mathcal{C}, \mathsf{F} \cup \{\mathsf{c}[\mathsf{f}(\mathsf{t}_1, \ldots, \mathsf{t}_{i-1}, \diamond, \mathsf{t}_{i+1}, \ldots, \mathsf{t}_n)]\})$. Comme ces colonnes étaient précedemment semblables et comme deux colonnes différentes dans $T_{\mathcal{L}}(\mathcal{C}, \mathsf{F})$ le restent dans $T_{\mathcal{L}}(\mathcal{C}, \mathsf{F} \cup \{\mathsf{c}[\mathsf{f}(\mathsf{t}_1, \ldots, \mathsf{t}_{i-1}, \diamond, \mathsf{t}_{i+1}, \ldots, \mathsf{t}_n)]\})$, à chaque passage dans la boucle, la relation d'équivalence $\equiv_{T_{\mathcal{L}}(\mathcal{C},\mathsf{F})}$ se raffine (augmentation du nombre de classes). Or $\equiv_{\mathcal{L},\mathcal{C}}$ est toujours plus fine que $\equiv_{T_{\mathcal{L}}(\mathcal{C},\mathsf{F})}$, et comme le nombre de classes de $\equiv_{\mathcal{L},\mathcal{C}}$ est fini (égal au nombre d'état d'un automate minimal pour$\mathcal{L}$), l'algorithme $IRTL$ ne peut exécuter cette boucle qu'un nombre fini de fois.

4.5 Complexité

La complexité en temps de l'algorithme dépend de la taille des données d'entrée ainsi que de la taille de l'automate à déterminer. Le premier tableau construit est de taille n^2, où n est le nombre total de nœuds des arbres de l'ensemble caractéristique $\mathcal{C}$ donné. En effet, chaque ligne du tableau correspond à un élément de $\mathcal{S}(\mathcal{C})$ et chaque colonne à un élément de $\mathsf{C}[\mathcal{C}]$ et $|\mathcal{S}(\mathcal{C})| = |\mathsf{C}[\mathcal{C}]| = n$. Ensuite, le nombre de lignes n'évolue plus et c'est le nombre de colonnes qui augmente de la manière suivante : chaque fois qu'une itération ajoute k contextes à F, deux lignes du tableau initialement égales sont différenciées. Or le nombre de lignes différentes correspond au nombre de classes d'équivalence et est toujours plus petit que le nombre m d'états de l'automate minimal. De plus, le nombre k de contextes ajoutés à chaque itération est inférieur à n. Le nombre de contextes ajoutés est donc inférieur à $m \times n$ et le nombre final de colonnes est inférieur à $m \times n + n$, soit un tableau final de $n \times (m \times n + n)$ cases remplies par des questions d'appartenance. La complexité est donc de $n \times (m \times n + n)$ multiplié par une constante correspondant au temps de réponse de l'oracle. A cela nous devons ajouter le temps de vérification du tableau qui est lui aussi polynomial dans sa taille.

4.6 Comparaison

Il est ici intéressant de comparer notre algorithme à celui de Angluin dans (Angluin, 1981). Cet algorithme identifie la classe des langages réguliers (de mots) à partir d'un ensemble *live-complete* et de questions d'appartenance. Un ensemble live-complete est un ensemble fini d'exemples positifs vérifiant la propriété que, pour produire cet exemple, chaque état d'un automate minimal a été utilisé. Cette définition est moins restrictive que celle d'ensemble caractéristique puisque tout ensemble caractéristique est live-complete. Il est alors pertinent de voir si l'algorithme fonctionnant à partir d'un ensemble live-complete est adaptable à la classe des langages réguliers d'arbres. Il apparaît que l'utilisation de chaque transition dans les exemples est nécessaire à l'apprentissage. Pour obtenir la propriété d'un ensemble caractéristique à partir d'un ensemble live-complete, Angluin commence par considérer chaque mot $\omega\alpha$ formé d'un préfixe ω d'un mot appartenant à son ensemble live-complete et d'un suffixe α qui est une lettre de l'alphabet. Ainsi, puisqu'à chaque état d'un automate minimal correspond au moins un préfixe ω d'un mot de l'ensemble live-complete, alors l'ensemble $\{\omega\alpha\}$ obtenu est caractéristique. Pour appliquer cette technique sur les arbres, il nous faut trouver pour chaque sous-arbre d'un ensemble live-complete donné lc (avec une définition équiva-

lente à celle des langages de mots), tout contexte susceptible de correspondre à une transition de l'automate minimal cherché. Ainsi pour chaque sous-arbre t il nous faut construire tous les termes de la forme $f(\mathsf{t}_1, \ldots, \mathsf{t}_i, \mathsf{t}, \mathsf{t}_{i+1}, \ldots, \mathsf{t}_n)$ pour les ajouter aux lignes du tableau, f étant un élément de l'alphabet de taille k d'arité $n(k)$ choisi parmi tous les k possibles et les t_j pour $1 \leq j \leq n$ étant choisis parmi tous les sous-arbres de lc. Cette opération nécessiterait donc $\Sigma_k n(k)^{|lc|}$ étapes de calcul, ce qui en accroît trés nettement la complexité. Notre choix de considérer les ensembles caractéristiques permet donc d'obtenir un algorithme efficace avec une propriété un peu plus restrictive sur l'ensemble d'exemples positifs donné.

5 Un exemple d'exécution

Soit le langage $\mathcal{L}$ défini par l'automate minimal suivant $\mathcal{A}$[3]:

$a(q_1) \to_{\mathcal{A}} q_0$	$d(q_3, q_5) \to_{\mathcal{A}} q_1$	$e(q_7) \to_{\mathcal{A}} q_3$	$g \to_{\mathcal{A}} q_7$
$a(q_2) \to_{\mathcal{A}} q_0$	$d(q_4, q_5) \to_{\mathcal{A}} q_1$	$e(q_8) \to_{\mathcal{A}} q_4$	$h \to_{\mathcal{A}} q_8$
$b(q_1) \to_{\mathcal{A}} q_0$	$d(q_3, q_6) \to_{\mathcal{A}} q_1$	$f(q_9) \to_{\mathcal{A}} q_5$	$i \to_{\mathcal{A}} q_9$
$c \to_{\mathcal{A}} q_1$	$d(q_4, q_6) \to_{\mathcal{A}} q_2$	$f(q_{10}) \to_{\mathcal{A}} q_6$	$j \to_{\mathcal{A}} q_{10}$

où q_0 est l'unique état final. D'après cette définition, on a :

$$\mathcal{L} = \{a(c), b(c), a(d(e(g), f(i))), a(d(e(h), f(i))), a(d(e(g), f(j))),$$

$$a(d(e(h), f(j))), b(d(e(g), f(i))), b(d(e(h), f(i))), b(d(e(g), f(j)))\}.$$

Supposons alors donné l'ensemble caractéristique suivant :

$$\mathcal{C} = \{b(c), a(d(e(g), f(i))), a(d(e(h), f(i))), a(d(e(g), f(j))), a(d(e(h), f(j)))\}.$$

$IRTL$ commence par la construction de $\mathsf{F} = \mathsf{C}[\mathcal{C}]$ et de $T_{\mathcal{L}}(\mathcal{S}(\mathcal{C}), \mathsf{F})$ à l'aide de questions d'appartenance (tableau 1).

IRTL remarque les trois couples de termes :

$$d(e(h), f(i)) \text{ et } d(e(h), f(j))$$

$$d(e(g), f(i)) \text{ et } d(e(h), f(j))$$

$$d(e(g), f(j)) \text{ et } d(e(h), f(j))$$

En effet,

$$e(g) \equiv_{T_{\mathcal{L}}(\mathcal{S}(\mathcal{C}), \mathsf{F})} e(h)$$

et

$$f(i) \equiv_{T_{\mathcal{L}}(\mathcal{S}(\mathcal{C}), \mathsf{F})} f(j)$$

3. D'après le lemme 4, la partition définie par un tableau d'observation n'est pas correcte si l'automate minimal du langage appris contient des règles particulières : des règles identiques à un état près dans leur partie gauche. Si un automate ne contient pas de telles règles, le premier tableau contruit est cohérent et l'algorithme s'arrête. Afin d'illustrer au mieux une exécution, nous proposons donc un exemple de langage dont l'automate minimal possède ce type de règles.

TAB. 1 – $T_{\mathcal{L}}(\mathcal{S}(\mathcal{C}), \mathsf{C}[\mathcal{C}])$

$\mathcal{S}(\mathcal{C})$ \ F	$\diamond$	$a(d(e(\diamond), f(i)))$	$a(d(e(g), f(\diamond)))$	$a(d(\diamond, f(i)))$	$a(d(e(g), \diamond))$	$a(\diamond)$	$a(d(e(h), f(\diamond)))$	$a(d(e(h), \diamond))$	$a(d(e(\diamond), f(j)))$	$a(d(\diamond, f(j)))$	$b(\diamond)$
$a(d(e(g), f(i)))$	$vrai$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$
$d(e(g), f(i))$	$faux$	$faux$	$faux$	$faux$	$faux$	$vrai$	$faux$	$faux$	$faux$	$faux$	$vrai$
$e(g)$	$faux$	$faux$	$faux$	$vrai$	$faux$	$faux$	$faux$	$faux$	$faux$	$vrai$	$faux$
$f(i)$	$faux$	$faux$	$faux$	$faux$	$vrai$	$faux$	$faux$	$vrai$	$faux$	$faux$	$faux$
g	$faux$	$vrai$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$vrai$	$faux$	$faux$
i	$faux$	$faux$	$vrai$	$faux$	$faux$	$faux$	$vrai$	$faux$	$faux$	$faux$	$faux$
$a(d(e(h), f(i)))$	$vrai$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$
$d(e(h), f(i))$	$faux$	$faux$	$faux$	$faux$	$faux$	$vrai$	$faux$	$faux$	$faux$	$faux$	$vrai$
$e(h)$	$faux$	$faux$	$faux$	$vrai$	$faux$	$faux$	$faux$	$faux$	$faux$	$vrai$	$faux$
h	$faux$	$vrai$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$vrai$	$faux$	$faux$
$a(d(e(g), f(j)))$	$vrai$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$
$d(e(g), f(j))$	$faux$	$faux$	$faux$	$faux$	$faux$	$vrai$	$faux$	$faux$	$faux$	$faux$	$vrai$
$f(j)$	$faux$	$faux$	$faux$	$faux$	$vrai$	$faux$	$faux$	$vrai$	$faux$	$faux$	$faux$
j	$faux$	$faux$	$vrai$	$faux$	$faux$	$faux$	$vrai$	$faux$	$faux$	$faux$	$faux$
$a(d(e(h), f(j)))$	$vrai$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$
$d(e(h), f(j))$	$faux$	$faux$	$faux$	$faux$	$faux$	$vrai$	$faux$	$faux$	$faux$	$faux$	$faux$
$b(c)$	$vrai$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$
c	$faux$	$faux$	$faux$	$faux$	$faux$	$vrai$	$faux$	$faux$	$faux$	$faux$	$vrai$

mais

$$b(d(e(g), f(i))) \in \mathcal{L},$$

$$b(d(e(g), f(j))) \in \mathcal{L},$$

$$b(d(e(h), f(i))) \in \mathcal{L}$$

et

$$b(d(e(h), f(j))) \notin \mathcal{L}.$$

IRTL ajoute les contextes

$$b(d(e(g), \diamond)),$$

$$b(d(e(h), \diamond)),$$

$$b(d(\diamond, f(i)))$$

et

$$b(d(\diamond, f(j)))$$

à F et complète le tableau d'observation (tableau 2).

$IRTL$ remarque maintenant que

$$b(d(e(h), f(i))) \in \mathcal{L},$$

$$b(d(e(h), f(j))) \notin \mathcal{L}$$

et

$$i \equiv_{T_{\mathcal{L}}(\mathcal{S}(\mathcal{C}),\mathsf{F})} j$$

et que

$$b(d(e(g), f(j))) \in \mathcal{L},$$

$$b(d(e(h), f(j))) \notin \mathcal{L}$$

et

$$g \equiv_{T_{\mathcal{L}}(\mathcal{S}(\mathcal{C}),\mathsf{F})} h.$$

Il ajoute donc les nouveaux contextes

$$b(d(e(\diamond), f(j)))$$

et

$$b(d(e(h), f(\diamond)))$$

à F et complète un dernière fois le tableau (tableau 3).

Le tableau est maintenant cohérent et l'algorithme donne l'automate $\mathcal{A}_{\equiv_{T_{\mathcal{L}}(\mathcal{S}(\mathcal{C}),\mathsf{F})}}$ en sortie. On vérifie aisément que cet automate est un renommage de $\mathcal{A}$ et donc un automate minimal pour le langage appris.

TAB. 2 – $T_{\mathcal{L}}(\mathcal{S}(\mathcal{C}), \mathsf{C}[\mathcal{C}] \cup \{b(d(e(g), \diamond)), b(d(e(h), \diamond)), b(d(\diamond, f(i)))\}), b(d(\diamond, f(j)))\})$

$\mathcal{S}(\mathcal{C})$ \ F	$\diamond$	$a(d(e(\diamond), f(i)))$	$a(d(e(g), f(\diamond)))$	$a(d(\diamond, f(i)))$	$a(d(e(g), \diamond))$	$a(\diamond)$	$a(d(e(h), f(\diamond)))$	$a(d(e(h), \diamond))$	$a(d(e(\diamond), f(j)))$	$a(d(\diamond, f(j)))$	$b(\diamond)$
$a(d(e(g), f(i)))$	$vrai$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$
$d(e(g), f(i))$	$faux$	$faux$	$faux$	$faux$	$faux$	$vrai$	$faux$	$faux$	$faux$	$faux$	$vrai$
$e(g)$	$faux$	$faux$	$faux$	$vrai$	$faux$	$faux$	$faux$	$faux$	$faux$	$vrai$	$faux$
$f(i)$	$faux$	$faux$	$faux$	$faux$	$vrai$	$faux$	$faux$	$vrai$	$faux$	$faux$	$faux$
g	$faux$	$vrai$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$vrai$	$faux$	$faux$
i	$faux$	$faux$	$vrai$	$faux$	$faux$	$faux$	$vrai$	$faux$	$faux$	$faux$	$faux$
$a(d(e(h), f(i)))$	$vrai$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$
$d(e(h), f(i))$	$faux$	$faux$	$faux$	$faux$	$faux$	$vrai$	$faux$	$faux$	$faux$	$faux$	$vrai$
$e(h)$	$faux$	$faux$	$faux$	$vrai$	$faux$	$faux$	$faux$	$faux$	$faux$	$vrai$	$faux$
h	$faux$	$vrai$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$vrai$	$faux$	$faux$
$a(d(e(g), f(j)))$	$vrai$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$
$d(e(g), f(j))$	$faux$	$faux$	$faux$	$faux$	$faux$	$vrai$	$faux$	$faux$	$faux$	$faux$	$vrai$
$f(j)$	$faux$	$faux$	$faux$	$faux$	$vrai$	$faux$	$faux$	$vrai$	$faux$	$faux$	$faux$
j	$faux$	$faux$	$vrai$	$faux$	$faux$	$faux$	$vrai$	$faux$	$faux$	$faux$	$faux$
$a(d(e(h), f(j)))$	$vrai$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$
$d(e(h), f(j))$	$faux$	$faux$	$faux$	$faux$	$faux$	$vrai$	$faux$	$faux$	$faux$	$faux$	$faux$
$b(c)$	$vrai$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$
c	$faux$	$faux$	$faux$	$faux$	$faux$	$vrai$	$faux$	$faux$	$faux$	$faux$	$vrai$

$\mathcal{S}(\mathcal{C})$ \ F	$b(d(e(g), \diamond))$	$b(d(e(h), \diamond))$	$b(d(\diamond, f(i)))$	$b(d(\diamond, f(j)))$
$a(d(e(g), f(i)))$	$faux$	$faux$	$faux$	$faux$
$d(e(g), f(i))$	$faux$	$faux$	$faux$	$faux$
$e(g)$	$faux$	$faux$	$vrai$	$vrai$
$f(i)$	$vrai$	$vrai$	$faux$	$faux$
g	$faux$	$faux$	$faux$	$faux$
i	$faux$	$faux$	$faux$	$faux$
$a(d(e(h), f(i)))$	$faux$	$faux$	$faux$	$faux$
$d(e(h), f(i))$	$faux$	$faux$	$faux$	$faux$
$e(h)$	$faux$	$faux$	$vrai$	$faux$
h	$faux$	$faux$	$faux$	$faux$
$a(d(e(g), f(j)))$	$faux$	$faux$	$faux$	$faux$
$d(e(g), f(j))$	$faux$	$faux$	$faux$	$faux$
$f(j)$	$vrai$	$faux$	$faux$	$faux$
j	$faux$	$faux$	$faux$	$faux$
$a(d(e(h), f(j)))$	$faux$	$faux$	$faux$	$faux$
$d(e(h), f(j))$	$faux$	$faux$	$faux$	$faux$
$b(c)$	$faux$	$faux$	$faux$	$faux$
c	$faux$	$faux$	$faux$	$faux$

TAB. 3 – $T_{\mathcal{L}}(\mathcal{S}(\mathcal{C}), \mathsf{C}[\mathcal{C}] \cup \{b(d(e(g), \diamond)), b(d(e(h), \diamond)), b(d(\diamond, f(j))), b(d(e(\diamond), f(j))), b(d(e(h), f(\diamond)))\})$

$\mathcal{S}(\mathcal{C})$ \ F	$\diamond$	$a(d(e(\diamond), f(i)))$	$b(d(e(h), \diamond))$	$a(d(e(g), f(\diamond)))$	$b(d(\diamond, f(i)))$	$a(d(\diamond, f(i)))$	$a(d(e(g), \diamond))$	$a(\diamond)$	$a(d(e(h), \diamond))$	$a(d(e(\diamond), f(j)))$	$a(d(\diamond, f(j)))$	$b(\diamond)$
$a(d(e(g), f(i)))$	$vrai$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$
$d(e(g), f(i))$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$vrai$	$faux$	$faux$	$faux$	$vrai$
$e(g)$	$faux$	$faux$	$faux$	$vrai$	$vrai$	$faux$	$faux$	$faux$	$faux$	$faux$	$vrai$	$faux$
$f(i)$	$faux$	$faux$	$faux$	$faux$	$faux$	$vrai$	$vrai$	$faux$	$vrai$	$faux$	$faux$	$faux$
g	$faux$	$vrai$	$vrai$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$vrai$	$faux$	$faux$
i	$faux$	$faux$	$vrai$	$faux$	$faux$	$faux$	$faux$	$faux$	$vrai$	$faux$	$faux$	$faux$
$a(d(e(h), f(i)))$	$vrai$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$
$d(e(h), f(i))$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$vrai$	$faux$	$faux$	$faux$	$vrai$
$e(h)$	$faux$	$faux$	$faux$	$faux$	$vrai$	$faux$	$faux$	$faux$	$faux$	$faux$	$vrai$	$faux$
h	$faux$	$vrai$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$vrai$	$faux$	$faux$
$a(d(e(g), f(j)))$	$vrai$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$
$d(e(g), f(j))$	$vrai$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$vrai$
$f(j)$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$vrai$	$vrai$	$faux$	$faux$	$faux$	$faux$
j	$faux$	$faux$	$vrai$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$
$a(d(e(h), f(j)))$	$vrai$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$
$d(e(h), f(j))$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$vrai$	$faux$	$faux$	$faux$	$faux$
$b(e)$	$vrai$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$
c	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$	$vrai$	$faux$	$faux$	$faux$	$vrai$

$\mathcal{S}(\mathcal{C})$ \ F	$b(d(e(g), \diamond))$	$b(d(e(h), \diamond))$	$b(d(\diamond, f(i)))$	$b(d(\diamond, f(j)))$	$b(d(e(\diamond), f(j)))$	$b(d(e(h), f(\diamond)))$
$a(d(e(g), f(i)))$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$
$d(e(g), f(i))$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$
$e(g)$	$faux$	$faux$	$vrai$	$faux$	$faux$	$faux$
$f(i)$	$vrai$	$vrai$	$faux$	$faux$	$faux$	$faux$
g	$faux$	$faux$	$faux$	$faux$	$vrai$	$faux$
i	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$
$a(d(e(h), f(i)))$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$
$d(e(h), f(i))$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$
$e(h)$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$
h	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$
$a(d(e(g), f(j)))$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$
$d(e(g), f(j))$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$
$f(j)$	$vrai$	$faux$	$faux$	$faux$	$faux$	$faux$
j	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$
$a(d(e(h), f(j)))$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$
$d(e(h), f(j))$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$
$b(e)$	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$
c	$faux$	$faux$	$faux$	$faux$	$faux$	$faux$

Références

ANGLUIN D. (1981). A note on the number of queries needed to identify regular languages. *Information and Control*, **51**, 76–87.

ANGLUIN D. (1987). Learning regular sets from queries and counter examples. *Information and Control*, **75**, 87–106.

ANGLUIN D. (1988). Queries and concept learning. *Machine learning*, **2**, 319–342.

CARRASCO R., ONCINA J. & CALERA-RUBIO J. (1998). Stochastic inference of regular tree languages. In *Grammatical inference: Algorithms and Applications (ICGI)*, volume 1433 of *Lecture Notes in Computer Science*, p. 187–198.

CHRISTOPHE A. (2000). L'apprentissage du langage. In *Université de tous les savoirs*, volume 2, p. 41–51: Odile Jacob.

COMON H., DAUCHET M., GILLERON R., JACQUEMARD F., LUGIEZ D., TISON S. & TOMMASI M. (1997). Tree automata techniques and applications. Disponible à : http://www.grappa.univ-lille3.fr/tata.

PINKER S. (1994). *The language instinct*. Harper.

SAKAKIBARA Y. (1988). Learning context-free grammars from structural data in polynomial time. In *Proceedings of the first annual workshop on Computational learning theory*, p. 330–344: Morgan Kaufmann Publishers Inc.