When Categorial Grammars meet Regular Grammatical Inference

Isabelle Tellier

GRAppA & Inria Futurs, Lille
MOSTRARE project**
Université Lille 3
59653 Villeneuve d'Ascq
France
isabelle.tellier@univ-lille3.fr

Abstract. In this paper, we first study the connections between subclasses of AB-categorial grammars and finite state automata. Using this, we explain how learnability results for categorial grammars in Gold's model from structured positive examples translate into regular grammatical inference results from strings. A closer analysis of the generalization operator used in categorial grammar inference shows that it is strictly more powerful than the one used in usual regular grammatical inference, as it can lead outside the class of regular languages. Yet, we show that the result can still be represented by a new kind of finite-state generative model called a *recursive automaton*. We prove that every unidirectional categorial grammar, and thus every context-free language, can be represented by such a recursive automaton. We finally identify a new subclass of unidirectional categorial grammars for which learning from strings is not more expensive than learning from structures. A drastic simplification of Kanazawa's learning algorithm from strings for this class follows.

1 Introduction

Grammatical inference deals with the problem of how to infer a grammar from examples of sentences it generates - and from sentences it does not generate, if negative examples are available. In the grammatical inference community (see the ICGI conference), many efforts have concerned the learnability of subclasses of regular grammars, represented by finite state automata [1, 13, 8, 7].

The inference of context-free grammars is more difficult and has received less attention. The most achieved work in this domain is Kanazawa's [12], who proved learnability results of large subclasses of AB-categorial grammars in Gold's model from positive examples [9]. These results concern two kinds of input data: strings and structural examples, i.e. syntactic analysis structures

** This research was partially supported by: "CPER 2000-2006, Contrat de Plan état - région Nord/Pas-de-Calais: axe TACT, projet TIC"; fonds européens FEDER "TIC - Fouille Intelligente de données - Traitement Intelligent des Connaissances" OBJ 2-phasing out - 2001/3 - 4.1 - n 3. And by "ACI masse de données ACIMDD"

where intermediate categories are deleted. But these results are hardly useful, because (except in very restricted cases) the corresponding algorithms have a high complexity [5, 6]. Surprisingly enough, nobody seems to have ever tried to translate these results into the more restricted class of regular grammars. The main characteristic of this class is that it produces only flat trees. We can prove easily that, in this context, structural examples are available for free when strings are available. So, it is worth considering how learnability results for subclasses of AB-categorial grammars from structural examples translate into learnability results for subclasses of regular grammars from strings.

In this paper, we first study how finite state automata translate into categorial grammars, and conversely. We compare learning strategies used in both domains. Doing so, we compare the relative power of the usual generalization operator of "state fusion", used in traditional regular grammatical inference and the generalization operator of "unifying substitutions on variables" used in categorial grammar learning. We prove that the latter is strictly more powerful than the former, as it sometimes allows to transform a categorial grammar generating a regular language into a categorial grammar generating a context-free grammar. Yet, in this case, it is still possible to represent the result of the operator as a generalized automaton. This leads to the definition of a new class of automata, called *recursive automata*. This class, which is a natural extension of finite state automata, has at least the expressive power of unidirectional categorial grammars, and can thus generate every context-free language.

This article thus proposes to unify grammatical inference results coming from two different traditions: the one focusing on finite state automata, and the one focusing on AB-categorial grammars, showing that they can both benefit from ideas coming from the other one.

2 Finite State Automata and Categorial Grammars

In this section, we first recall basic definitions concerning finite state automata and AB-categorial grammars, and show that there are easy correspondences between them.

2.1 Finite State Automata and Regular Languages

We recall here the classical notations for automata and regular languages.

Definition 1 (Finite State Automaton (FSA) and their Language). *A **finite state automaton** (FSA in the following) A is a 5-tuple $A = \langle Q, \Sigma, \delta, q_0, F \rangle$ such that Q is the finite set of* states *of A, Σ its finite* vocabulary*, $q_0 \in Q$ is the* initial state *of A (we restrict ourselves to automata with a unique initial state) and $F \subseteq Q$ is the set of its* final states*. Finally, δ is the* transition function *of A, defined from $Q \times \Sigma$ to 2^Q.*

The language $L(A)$ recognized by A is defined as: $L(A) = \{w \in \Sigma^* | \delta^*(q_0, w) \cap F \neq \emptyset\}$, where δ^* is the natural extension of δ on $Q \times \Sigma^*$ such that: for any

$a \in \Sigma$, *any* $u \in \Sigma^*$ *and any* $q \in Q$, $\delta^*(q, au) = \{\delta^*(q', u) | q' \in \delta(q, a)\}$.
The set of languages recognized by FSA is called the set of ***regular languages***.

Example 1. Figure 1 displays a simple FSA A such that $L(A) = a^+b^+$.

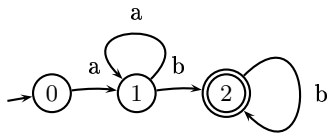

Fig. 1. A Simple Finite State Automaton

2.2 Grammars and Categorial Grammars

We recall the classical definitions of a generative grammar, and of categorial grammars. Here, we restrict ourselves to AB-(or classical) categorial grammars.

Definition 2 (Generative Grammars and their Language). *A* ***generative grammar*** *(or simply a grammar in the following) is a 4-tuple* $G = \langle \Sigma, N, P, S \rangle$ *with* Σ *the finite terminal vocabulary of* G, N *its finite non terminal vocabulary,* $P \subset (\Sigma \cup N)^+ \times (\Sigma \cup N)^*$ *its finite set of rules and* $S \in N$ *the axiom.*

The language generated by a grammar G *is* $L(G) = \{w \in \Sigma^* | S \longrightarrow^* w\}$ *where* $\longrightarrow^*$ *is the reflexive and transitive closure of the relation defined by* P.

Definition 3 (Categories, AB-Categorial Grammars and their Language). *Let* $\mathcal{B}$ *be a set (at most countably infinite) of basic categories containing a distinguished category* $S \in \mathcal{B}$, *called the axiom. We note* $Cat(\mathcal{B})$ *the smallest set such that* $\mathcal{B} \subset Cat(\mathcal{B})$ *and for any* $A, B \in Cat(\mathcal{B})$ *we have:* $A/B \in Cat(\mathcal{B})$ *and* $B\backslash A \in Cat(\mathcal{B})$.

For every finite vocabulary Σ *and for every set of basic categories* $\mathcal{B}$ *(*$S \in \mathcal{B}$*), a* ***categorial grammar*** *is a finite relation* G *over* $\Sigma \times Cat(\mathcal{B})$. *We note* $\langle v, A \rangle \in G$ *the assignment of the category* $A \in Cat(\mathcal{B})$ *to the element of the vocabulary* $v \in \Sigma$. *AB-categorial grammars (CGs in the following) are categorial grammars where the syntactic rules take the form of two rewriting schemes:* $\forall A, B \in Cat(\mathcal{B})$

- *FA (Forward Application) :* $A/B \quad B \rightarrow A$
- *BA (Backward Application) :* $B \quad B\backslash A \rightarrow A$

The language generated (or recognized) by a CG G *is:*
$L(G) = \{w = v_1 \dots v_n \in \Sigma^+ \mid \forall i \in \{1, \dots, n\}, \exists A_i \in Cat(\mathcal{B})$ *such that* $\langle v_i, A_i \rangle \in G$ *and* $A_1 \ \dots \ A_n \rightarrow^* S\}$, *where* $\rightarrow^*$ *is the reflexive and transitive closure of* $\rightarrow$.

Example 2. For example, let $\mathcal{B} = \{S, T, CN\}$ (where T stands for "term" and CN for "common noun"), $\Sigma = \{John, runs, loves, a, cat\}$ and G be defined by:
$G = \{\langle John, T \rangle, \langle runs, T\backslash S \rangle, \langle loves, (T\backslash S)/T \rangle, \langle loves, T\backslash(S/T) \rangle, \langle cat, CN \rangle,$
$\langle a, (S/(T\backslash S))/CN \rangle, \langle a, ((S/T)\backslash S)/CN \rangle\}$.
This CG generates sentences like "John runs", "John loves a cat", etc.

Definition 4 (FA-Structures, Structural Examples, Structured Language). *A **functor-argument (or FA-) structure** over an alphabet Σ is a binary-branching tree whose leaf nodes are labeled by elements of Σ and whose internal nodes are labeled either by BA or FA. The set of FA-structures over Σ is denoted Σ^F.*
For any AB-categorial grammar $G \subset \Sigma \times Cat(\mathcal{B})$, a structural example for G is an element of Σ^F which can be obtained from the analysis tree resulting from a syntactic parsing of a string $w \in L(G)$ in G by deleting each of its categories. For any CG $G \subset \Sigma \times Cat(\mathcal{B})$, the structured language $FL(G)$ associated with G is the set of structural examples for G.

$\mathcal{G}$ denotes the class of CGs. For every integer $k \geq 1$, the set of CGs assigning at most k distinct categories to each member of the vocabulary is the class of k-valued CGs denoted by $\mathcal{G}_k$. When $k = 1$ the grammars are also called rigid.

2.3 From Automata to Categorial Grammars and Back

The expressive power of CGs is the one of ϵ-free (ϵ is the empty string) context-free languages [2]. So, of course, they can also generate ϵ-free regular languages. Correspondences between FSA and CGs are easy to define.

Definition 5 (Regular CGs). *We call a **regular CG** a CG $G \subset \Sigma \times Cat(\mathcal{B})$ that only contains assignments of the form $\langle v, A\rangle$ or $\langle v, A/B\rangle$ with $v \in \Sigma$ and $A, B \in \mathcal{B}$. The set of regular CGs is noted $\mathcal{G}_r$.*

Property 1 (Transformation of an Automaton). Let $A = \langle Q, \Sigma, \delta, q_0, F\rangle$ be a FSA. Let $\mathcal{B} = (Q \backslash \{q_0\}) \cup \{S\}$ (where $S \notin Q$). It is possible to define a regular CG $G \subset \Sigma \times Cat(\mathcal{B})$ such that $L(G) = L(A) \backslash \{\epsilon\}$.

Proof (Proof of Property 1). This property is the consequence of sequentially applying the classical transformation of a FSA A into a left-linear regular grammar $G_1 = \langle \Sigma, Q, P_1, S\rangle$, then applying the transformation of G_1 into a CG: $\forall a \in \Sigma$ and $\forall q, q' \in Q$ such that $q' \in \delta(q, a)$ do:

- if $q' \in F$ then
  - $q \longrightarrow a$ is a rule of G_1 (element of P_1) and $\langle a, q\rangle \in G$ (replace q_0 by S);
  - if $\exists u \in \Sigma$, $\exists q'' \in \delta(q', u)$ then $q \longrightarrow aq'$ is a rule of G_1 and $\langle a, q/q'\rangle \in G$ (replace q_0 by S);
- else $q \longrightarrow aq'$ is a rule of G_1 and $\langle a, q/q'\rangle \in G$ (replace q_0 by S).

$\epsilon \in L(A)$ if and only if $q_0 \in F$. This situation gives rise to a new rule $S \longrightarrow \epsilon$ in P_1. But in CGs, it is impossible to assign a category to ϵ, so this rule has no counterpart in G. So, we have: $L(A) \backslash \{\epsilon\} = L(G_1) \backslash \{\epsilon\} = L(G)$.□

Example 3. Let us apply the previous process to the FSA given in Example 1. The rules of the corresponding left-linear regular grammar are the following (where the state of number i is associated with a non terminal symbol noted Q_i, with $Q_0 = S$): $S \longrightarrow aQ_1$, $Q_1 \longrightarrow aQ_1$, $Q_1 \longrightarrow b$, $Q_1 \longrightarrow bQ_2$, $Q_2 \longrightarrow b$,

$Q_2 \longrightarrow bQ_2$. And the CG G is:
$G = \{\langle a, S/Q_1\rangle, \langle a, Q_1/Q_1\rangle, \langle b, Q_1\rangle, \langle b, Q_1/Q_2\rangle, \langle b, Q_2\rangle, \langle b, Q_2/Q_2\rangle\}$.

So, FSA can easily be *lexicalized*. Note that the only operator used in categories of a regular CG is / and the only useful rule is FA (it would have been \ and BA if we had first transformed the automaton into a right-linear regular grammar). In fact, *the previous transformation preserves not only the string language, but also the structured language*. A crucial consequence is that structural examples in the sense of Definition 4 are available for free from the corresponding strings: underlying structures produced by automata are always flat, and the only label for internal nodes is FA.

Example 4. Figure 2 displays two analysis trees produced by the CG obtained in Example 3, and (on the right) the corresponding structural examples.

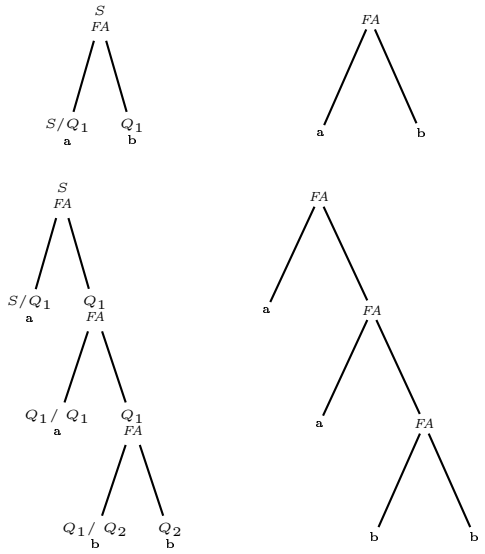

Fig. 2. Syntactic Analyses and the corresponding Structural Examples

Property 2 (Transformation of a Regular AB-Categorial Grammar). Every regular CG $G \subset \Sigma \times Cat(\mathcal{B})$ can be transformed into a FSA $A = \langle Q, \Sigma, \delta, q_0, F\rangle$ recognizing the same (ϵ-free) language.

Proof (sketch of the proof of Property 2). This transformation is the reverse of the one described in Property 1: the only thing to pay attention to is to add a unique final state F_A in A. Let $Q = \mathcal{B} \cup \{F_A\}$ with $F_A \notin \mathcal{B}$, $q_0 = S$ and $F = \{F_A\}$. Each assignment $\langle a, U/V\rangle \in G$ corresponds to a transition labelled

by a between the states U and V in A (so: $\delta(U,a) = V$) and each assignment $\langle a, U\rangle \in G$ to a transition labeled by a between U and F_A ($\delta(U,a) = F_A$).□

Example 5. The automaton built by applying this operation to the CG obtained in Example 3 is given in Figure 3. It is not exactly the same as the initial one (in Example 1), because of the final state added to the states coming from basic categories. The result is, usually, nondeterministic.

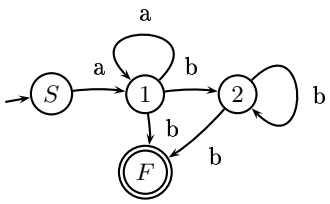

Fig. 3. The Automaton Obtained from the AB-Categorial Grammar

Remark 1. Properties 1 and 2 do not mean that only the CGs that are regular generate a regular language. The CG of Example 2 is not regular in the sense of Definition 5 but it generates a finite (and thus regular) language. But the structures it produces are not flat.

Property 3 (Language Associated with a State). Let G be a regular CG and A be the automaton obtained from it. Then for any basic category Q of G (corresponding to a non-final state Q of A, $S = q_0$), we have two equivalent ways to define the language $L(Q)$ associated with Q:
$L(Q) = \{w = v_1 \dots v_n \in \Sigma^+ \mid \forall i \in \{1, \dots, n\}\ \exists A_i \in Cat(\mathcal{B})$ such that $\langle v_i, A_i\rangle \in G$ and $A_1\ \dots\ A_n \rightarrow^* Q\}$ and $L(Q) = \{w \in \Sigma^+ | F_A \in \delta^+(Q, w)\}$.

Proof (sketch of proof of Property 3). The proof is an easy consequence of Properties 1 and 2, where Q replaces S.□

The second definition of state language slightly differs from the classical one, because it excludes ϵ (we know that $Q \neq F_A$ so $\epsilon \notin L(Q)$) and uses the fact that there is only one terminal state F_A in A. So, strings in $L(Q)$, which can be associated with the category Q in G, also correspond in A to strings produced by following a path starting from the state Q and reaching the final state F_A. Of course, if $Q = q_0 = S$, we have: $L(S) = L(A) = L(G)$.

Example 6. In the automaton of Figure 3, $L(Q_1) = a^*b^+$ and $L(Q_2) = b^+$.

3 Inference of CGs from Positive Examples

The learnability of CGs in Gold's model from positive examples has received great attention in the last years. Now that we have an easy translation of a subclass of such grammars (the subclass of the regular ones) into automata, it is natural to see how these results translate into regular language learning, to compare them with known results in this domain.

3.1 Gold's model

To deal with questions of learnability, Kanazawa [12] introduced the notion of grammar system, allowing a reformulation of Gold's model of *identification in the limit from positive examples* [9]. We recall this notion here.

Definition 6 (Grammar System). *A **grammar system** is a triple $\langle \Omega, \Lambda, L \rangle$ made of a hypothesis space Ω (here, Ω will be a set of grammars), a sample space Λ, which is a recursive subset of A^*, for some fixed alphabet A (elements of Λ are sentences and subsets of Λ are languages) and $L : \Omega \rightarrow pow(\Lambda)$ is a naming function. The question of whether $w \in L(G)$ which holds between $w \in \Lambda$ and $G \in \Omega$, is supposed to be computable.*

The main grammar systems we deal with in the following of this paper are $\langle \mathcal{G}, \Sigma^*, L \rangle$ and $\langle \mathcal{G}, \Sigma^F, FL \rangle$. The first one concerns the learnability of CGs from strings and the second one the learnability of CGs from structural examples.

Definition 7 (Learnability Criterion). *Let $\langle \Omega, \Lambda, L \rangle$ be a grammar system and $\phi : \cup_{k \geq 1} \Lambda^k \rightarrow \Omega$ be a computable function. We say that ϕ **converges** to $G \in \Omega$ on a sequence $\langle s_i \rangle_{i \in \mathbb{N}}$ of elements of Λ if $G_i = \phi(\langle s_0, ..., s_i \rangle)$ is defined and equal to G for all but finitely many $i \in \mathbb{N}$ - or equivalently if there exists $n_0 \in \mathbb{N}$ such that for all $i \geq n_0$, G_i is defined and equal to G. Such a function ϕ is said to **learn** $\mathcal{G} \subseteq \Omega$ if for every language L in $L(\mathcal{G}) = \{L(G) | G \in \mathcal{G}\}$ and for every infinite sequence $\langle s_i \rangle_{i \in \mathbb{N}}$ that enumerates the elements of L (i.e. such that $\{s_i | i \in \mathbb{N}\} = L$), there exists some G in $\mathcal{G}$ such that $L(G) = L$ and ϕ converges to G on $\langle s_i \rangle_{i \in \mathbb{N}}$.*

Theorem 1 (Learnability of $\mathcal{G}_k$ [12]). *For any $k \geq 1$, the class $\mathcal{G}_k$ of k-valued CGs is learnable in the grammar systems $\langle \mathcal{G}, \Sigma^*, L \rangle$ (i.e. from strings) and $\langle \mathcal{G}, \Sigma^F, FL \rangle$ (i.e. from structural examples).*

3.2 Categorial Grammars and Regular Grammatical Inference

Of course, Theorem 1 holds for k-valued regular CGs. In this section, we first translate Kanazawa's results into regular grammatical inference. We then show the equivalence between two other known results in the case of regular CGs.

Theorem 2 (Learnability of k-valued Regular CGs). *For every $k \geq 1$, the class of k-valued regular CGs $\mathcal{G}_k \cap \mathcal{G}_r$ is learnable from structural examples and from strings.*

Proof (Sketch of Proof of Theorem 2). This is a direct consequence of Theorem 1, restricted to the class $\mathcal{G}_r$. To learn regular k-valued CGs from strings, you just have to apply the BP learning strategy from structural examples that are flat trees with FA internal nodes, then to select among the output the CGs that are isomorph with a regular CG (which is decidable) before performing the inclusion tests.□

It is interesting to notice that, in the case of regular CGs, unlike in the case of general CGs, strings and structures are equivalent - that is, structures are available for free from strings. This is the idea we will try to generalize for larger classes of CGs in the following section.

Unfortunately, this class of grammars is not very interesting. As a matter of fact, if k is given, it is also a bound on the maximal number of transitions labeled by the same element of vocabulary in the corresponding FSA: there exists a finite number of distinct automata satisfying this condition, so the learnability result is in fact trivial.

Nevertheless, the notion of k-valued automaton, i.e. FSA which are the result of applying the process of Property 2 to k-valued regular CGs is original. As a matter of fact, it focuses on the total number of transitions labeled by the same symbol in an automaton, instead of focusing on the total number of states in this automaton. k-valued automata seem well adapted to large alphabets Σ (especially when k is small), which contrasts with usual classes of automata (and usual learning algorithms) considered in the field of regular inference.

Other interesting results worth being compared: the one concerning the class of 0-reversible FSA [1] and the one concerning the class of reversible CGs [3].

Definition 8 (0-Reversibility of a FSA [1]). *A FSA is said to be* 0*-reversible if and only if it is deterministic and the automaton obtained by reversing the transitions backwards is also deterministic.*

Definition 9 (Reversibility of a CG [3]). *A CG is said to be reversible if it does not contain two assignments of categories for the same element of vocabulary, which are distinct by only one basic category.*

Theorem 3 (Equivalence of these Reversibility Notions). *Let G be a regular CG and A be the FSA obtained from it. A is* 0*-reversible in the sense of Definition 8 if and only if G is reversible in the sense of Definition 9.*

Proof (Sketch of Proof of Theorem 3). This property is a direct consequence of Theorem 2. As we only considered automata with a unique initial state and no ϵ transition, the conditions for A to be deterministic only concern transitions starting from the same state and labeled by the same symbol. They are equivalent for G with the following ones: $\forall a \in \Sigma$

- $\forall Q_1, Q_2, Q_3 \in \mathcal{B}$: $\langle a, Q_1/Q_2\rangle \in G$ and $\langle a, Q_1/Q_3\rangle \in G \Leftrightarrow Q_2 = Q_3$.
- $\forall Q_1, Q_2 \in \mathcal{B}$: $\langle a, Q_1\rangle \in G$ and $\langle a, Q_1/Q_2\rangle \in G \Leftrightarrow Q_2 = F_A$ (in fact, $\langle a, Q_1\rangle$ stands for $\langle a, Q_1/F_A\rangle$);

Similarly, as A has only one final state, the conditions for the reversed automaton to be deterministic are equivalent with the following ones: $\forall a \in \Sigma$

- $\forall Q_1, Q_2 \in \mathcal{B}$: $\langle a, Q_1\rangle \in G$ and $\langle a, Q_2\rangle \in G \Leftrightarrow Q_1 = Q_2$
- $\forall Q_1, Q_2, Q_3 \in \mathcal{B}$: $\langle a, Q_1/Q_2\rangle \in G$ and $\langle a, Q_3/Q_2\rangle \in G \Leftrightarrow Q_1 = Q_3$.

For regular CGs, these conditions coincide with the one of Definition 9.□

So, the learnability of the class of 0-reversible FSA from strings [1] is equivalent with the one concerning the class of reversible CGs [3] from structures, in the special case of regular CGs. The corresponding learning algorithms should be more carefully compared, but they also seem equivalent.

3.3 Learning Algorithm

The learnability results of Theorem 1 are not only theoretical ones. The original algorithm BP able to identify the set of k-valued CGs without useless category compatible with a set of structural examples is due to Buszkowki & Penn [4]. We briefly recall it here, exemplifying it on a set of flat structural examples.

To identify every k-valued CG compatible with a given sample D of structural examples, the first steps are the following for each element of D:

1. introduce a label S at the root of each structural example;
2. introduce a distinct variable x_i at each argument node (i.e. at the left daughter of each BA node and at the right daughter of each FA node);
3. introduce a fractional category at every other node, respecting the labels of the functional application (FA or BA) to be applied.

The result of collecting all the categories associated with each distinct member of the vocabulary after these steps is a CG called the **general form** of D and noted $GF(D)$.

Example 7. Let D be the couple of structural examples given in Example 4. The previous process gives the result of Figure 4, and $GF(D)$ is defined as follows:

- a: $S/x_1,\ S/x_4,\ x_4/x_3$;
- b: $x_1,\ x_3/x_2,\ x_2$.

The corresponding FSA is given in Figure 5: it very much looks like what is known in regular grammatical inference as the **maximal canonical automaton** $MCA(D)$: the only difference is that the FSA corresponding to $GF(D)$ has a unique initial state and a unique final state. It is generally not deterministic.

Then, substitutions that unify category assignments are searched for.

Definition 10 (Variable Categories and Substitutions). *Let χ be an enumerably infinite set of variables and $\mathcal{B} = \chi \cup \{S\}$. A substitution is a function $\sigma : \chi \longrightarrow Cat(\mathcal{B})$ that maps variables to categories (it is initialized by the Identity function on χ). A substitution is extended to a function from categories to categories as follows: (i) $\sigma(S) = S$, (ii) $\sigma(A/B) = \sigma(A)/\sigma(B)$ and (iii) $\sigma(A\backslash B) = \sigma(A)\backslash\sigma(B)$ for any $A, B \in Cat(\mathcal{B})$. Similarly, a substitution is naturally extended to apply to a CG: $\forall G \in \mathcal{G}$, $\sigma(G) = \{\langle v, \sigma(A)\rangle | \langle v, A\rangle \in G\}$. For any CG G, a unifying substitution for G is a substitution that unifies categories assigned to the same element of the vocabulary in G.*

Property 4 (Fundamental Property [4]). For any CG G, the following two properties are equivalent: (i) $D \subseteq FL(G)$ and (ii) $\exists\sigma$ such that $\sigma[GF(D)] \subseteq G$.

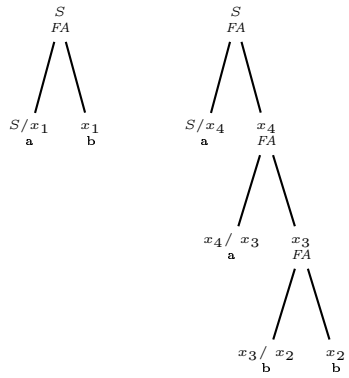

Fig. 4. First Step of the Learning Algorithm

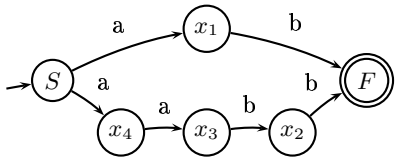

Fig. 5. The Automaton Corresponding with $GF(D)$

In other words, CGs whose structured language is compatible with the input D are those which contain a substitution of $GF(D)$. If the target grammar is k-valued, the learning strategy BP thus only consists in finding every possible unifying substitution σ such that the grammar $\sigma[GF(D)]$ is k-valued. If $k = 1$, the learning process is incremental and the target grammar, if it exists, is unique and available in the limit. If $k > 1$, a remaining problem is to select one grammar among the set. This is performed by inclusion tests on the structured languages of the candidates (or by bounded inclusion tests for string languages). We will not go into further details about this final step.

Applying a substitution on a CG is a generalization operation, because we have the following property [4]: $\sigma(G_1) \subseteq G_2 \Longrightarrow FL(G_1) \subseteq FL(G_2)$. So we always have: $FL(G) \subseteq FL(\sigma(G))$ and, similarly, $L(G) \subseteq L(\sigma(G))$. But in usual regular grammatical inference, the most often used generalization operator is the one of state merging [1, 13, 8]. What is the link between these two operators ? In the context of a set D containing only flat structures with internal nodes FA, $GF(D)$ is necessarily a regular CG (see Example 7). So, only two cases can occur to unify two categories:

- conditions of the form $\sigma(x_i) = \sigma(x_j) = x_j$ for any $x_i \in \chi$ and any $x_j \in \chi \cup \{S\}$ specify a state merging: states x_i and x_j are merged.
- conditions of the form $\sigma(x_i) = x_j/x_k$, for any $x_i \in \chi$ and any $x_j, x_k \in \chi \cup \{S\}$ are more difficult to understand. Such a condition means two things:

- the state x_i must be renamed as a state called x_j/x_k;
- every string that could be associated with the category x_i in the grammar $GF(D)$ can now be used as a "transition" between the states x_j and x_k.

Example 8. For example, let us define a substitution σ that unifies some of the categories assigned to a and b in the grammar $GF(D)$ of Example 7 as follows: $\sigma(x_4) = \sigma(x_1) = x_3/x_2$ (and σ is the identity elsewhere). The resulting CG $\sigma(GF(D))$ is the following:

- a: $S/(x_3/x_2),\ (x_3/x_2)/x_3$;
- b: $x_3/x_2,\ x_2$.

This CG is no longer regular. Nevertheless, it can be represented in a generalized automaton by adding a "recursive transition", that is a transition labeled by a state (here, the state x_3/x_2). The corresponding automaton is given in Figure 6.

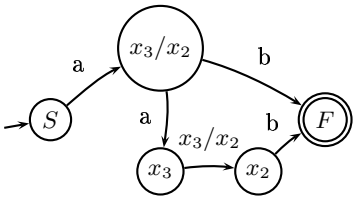

Fig. 6. The Generalized Automaton Corresponding with $\sigma(GF(D))$

In this automaton, previous states x_1 and x_4 were merged and renamed as x_3/x_2, and a new "recursive" transition labeled by x_3/x_2 replaces the one that was labeled by b between the states x_3 and x_2. To use this new transition, you need to produce a string of category x_3/x_2, that is a string that belongs to the state language of x_3/x_2. According to Property 3, another way to characterize such a string is that it would be obtained by following a path from the state x_3/x_2 to the final state F. To follow such a path, the first possible choice is, of course, the direct transition labeled by b. But another possible choice is to reach x_3 by a where the previous choice is posed again. Of course, a stack is necessary to remember the list of recursive transitions used. The language recognized by this generalized automaton is $a^n b^n$, which is not regular. This generalized automaton can be considered as a special case of Recursive Transition Network.

To understand how such a generalization could occur, let us look at the analysis tree produced by the grammar for the string *aaabbb*, given in Figure 7.

This tree is no longer flat. To understand how it was built, look back at the second tree of Figure 4. The specification of $\sigma(x_4) = x_3/x_2$ provides an equality between labels of this tree: the label of an internal node (x_4) becomes equal to the label of a leaf (x_3/x_2). This equality opens the possibility to insert the subtree rooted by x_4 in the place of the leaf labeled by x_3/x_2, as is done in Figure 7. This operation is exactly what is called an *adjunction*, in the formalism of Tree Adjoining Grammars [11].

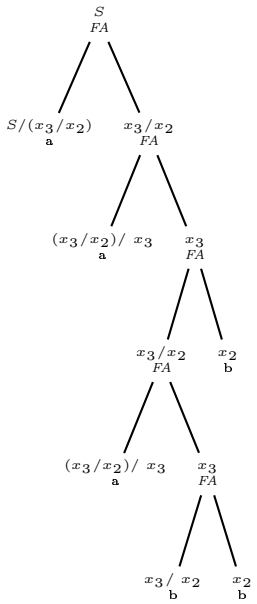

Fig. 7. Syntactic Analyses Tree for *aaabbb*

4 Learning CF-Languages from Flat Structures

Example 8 suggested that it is possible to generalize real trees from flat trees and to represent context-free languages by a recursive automaton. This section is dedicated to the formalization of these ideas, and to the study of their consequences to improve Kanazawa's learning algorithm from strings, when it is possible.

4.1 Recursive Automata an their Expressivity

Definition 11 (Recursive Automaton). *A **recursive automaton** (RA in the following) R is a 5-tuple $R = \langle Q, \Sigma, \gamma, q_0, F\rangle$ such that Q is the finite set of* states *of R, Σ is its finite* vocabulary*, $q_0 \in Q$ its (unique)* initial state *and $F \in Q$ its (unique)* final state*. γ is the* transition function *of R, defined from $Q \times (\Sigma \cup Q)$ to 2^Q.*

The only important difference between FSA and RA is that, in a RA, it is possible to label a transition by an element of Q. We call such a transition a **recursive transition**. To use a recursive transition labelled by $q \in Q$, you have to produce an element of $L(q)$. We restrict ourselves to RA with a unique initial state and a unique final state, but it is not a crucial choice. As we consider ϵ-free languages, it is supposed that $F \neq q_0$.

Definition 12 (Language Recognized by a RA). *Let $R = \langle Q, \Sigma, \gamma, q_0, F\rangle$ be a recursive automaton. The language $L(R)$ recognized by R is defined as: $L(R) = \{w \in \Sigma^+ | F \in \gamma^+(q_0, w)\}$, where γ^+ is the natural extension of γ on $Q \times \Sigma^+$, i.e. for any $u \in \Sigma^+$ and $v \in \Sigma^*$ any $q \in Q$, $\gamma^+(q, uv)$ is the smallest subset containing $\{\gamma^*(q', v) | q' \in \gamma(q, u)\}$ if $u \in \Sigma$ and $\{\gamma^*(q', v) | \exists t \in Q$ such that $q' \in \gamma(q, t)$ and $u \in L(t)\}$ else, where $L(t)$ is the state language of t.*

RA also produce *structures*. These structures are not necessarily flat, because recursive transitions allow a real branching. A real recursivity occurs when there exists a path starting from a state $q \in Q$, using a recursive transition labelled by q and reaching the final state (as it was the case for x_3/x_2 in Figure 6).

Theorem 4 (From Unidirectional GCs to RA). *A unidirectional CG only assigns categories that are either basic or built from the operator $/$. The set of unidirectional CGs is noted $\mathcal{G}_U$. Every $G \in \mathcal{G}_U$ can be transformed into a strongly equivalent RA, i.e. a RA generating the same structured language.*

Proof (Proof of Theorem 4). It is well known [10] that any CG $G \subset \Sigma \times Cat(\mathcal{B})$ can be transformed into a strongly equivalent context-free grammar in Chomsky Normal Form $H = \langle \Sigma, N, P, S\rangle$ in the following way: N is the set of every subcategory of a category assigned to a member of the vocabulary in G (a category is a subcategory of itself). Then, for every $\langle v, A\rangle \in G$, let $A \longrightarrow v \in P$ and for all A, B in N, let $A \longrightarrow A/B\ B \in P$ (for unidirectional CGs, this is enough). The set of states of our RA R is the set $N \cup \{F\}$, with $F \notin N$. Rules of the form $A \longrightarrow v$ correspond to a transition labelled by v between the state A and the final state F. Rules of the form $A \longrightarrow A/B\ B$ correspond to a recursive transition labelled by A/B between states A and B. The use of a rule of this form in a derivation in G means that a subtree rooted by A can be decomposed into two subtrees: one rooted by A/B and one rooted by B. This is exactly what is also expressed by the corresponding recursive transition in R.□

Corollary 1. *Unidirectional CGs can generate every ϵ-free context-free language [2]. So, it immediately follows from Theorem 4 that every ϵ-free context-free language can also be generated by a RA.*

Example 9. The classical unidirectional CG recognizing the language $a^n b^n$, $n \geq 1$, is the following: $G = \{\langle a, S/B\rangle, \langle a, (S/B)/S\rangle, \langle b, B\rangle\}$. The corresponding RA (distinct from the one of Example 8) is given in Figure 8. This RA can be simplified: the recursive transitions that are not *really* recursive can be lexicalized. Here, you can delete the state $(S/B)/S$ and replace the label of the recursive transition between S/B and S by a. But it is not possible for the state S/B.

The main interest of RA is that they produce the same structures as the ones produced by unidirectional CGs, i.e. using only FA rules. So, we hope to infer them from flat trees, like in Example 8. The problem is that the RA obtained from the process exemplified in Example 9 does not belong to the search space of any set of flat trees, because of the states that are not reachable from the initial state. So, we will need a more constraint form for RA (or for unidirectional CGs).

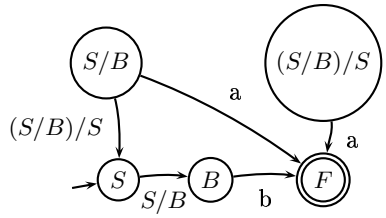

Fig. 8. Another RA recognizing $a^n b^n$

4.2 Learning Unidirectional CGs from Flat Structure

When only strings are available, Kanazawa's learning strategy consists in generating every possible structural example compatible with the input data, then applying the process described in section 3.3. It is a very expensive strategy, not tractable in practice. On the basis of the previous examples, we propose to refine Kanazawa's learning strategy to learn CF-languages from strings.

Definition 13. *A CG G is said to have no useless category if every assignment of a category to a member of the vocabulary is used in at least one syntactic analysis of an element of $L(G)$ (G is also said to be in reduced form [12]). Let: $\mathcal{G}_{k,\sigma} = \{\sigma(G) | G \in \mathcal{G}_k \cap \mathcal{G}_r$ and G has no useless category and σ is a unifying substitution for $G\}$.*

Of course, for every $k \geq 1$, $\mathcal{G}_{k,\sigma} \subseteq \mathcal{G}_k \cap \mathcal{G}_U$ and $\bigcup_{k\geq 1}\{L(G) | G \in \mathcal{G}_{k,\sigma}\}$ has a non null intersection with the set of non-regular CF-languages (see Example 8). But we do not know whether it can generate every CF-language.

The problem is that the notion of accessibility in a RA is different from the notion of accessibility in a FSA, because of recursive transitions. A state which is not accessible in a FSA (corresponding with a useless category in the corresponding CG) can become accessible after a substitution has been applied.

For every k, the main interest of the class $\mathcal{G}_{k,\sigma}$ is that it naturally extends the class $\mathcal{G}_k \cap \mathcal{G}_r$, whose fundamental property is that its members produce structural examples that are flat trees with FA internal nodes only. This property can be used to adapt Kanazawa's standard learning algorithm, as explained below.

Theorem 5. *For every $k \geq 1$, for every $G \in \mathcal{G}_{k,\sigma}$, there exists a set D of flat structures with FA internal nodes and there exists τ, a unifying substitution for $GF(D)$ such that $G = \tau(GF(D))$.*

Proof (proof of Theorem 5). For every $G \in \mathcal{G}_{k,\sigma}$, by definition there exists $G' \in \mathcal{G}_k \cap \mathcal{G}_r$ without useless category and σ a unifying substitution for G' such that $G = \sigma(G')$. We know by Theorem 2 that $\mathcal{G}_k \cap \mathcal{G}_r$ is learnable from structural examples and from strings. Let D be a characteristic set of structural examples for G' (see [12]). G' is regular, so D is only made of flat structures with FA internal nodes. G' has no useless category, so it belongs to the result of the BP algorithm [12] whose inputs are k and D. This means that there exists a

unifying substitution ρ for $GF(D)$ such that $G' = \rho(GF(D))$. So $G = \sigma(G') = \sigma(\rho(GF(D))$. Let $\tau = \sigma \circ \rho$, which is a unifying substitution for $GF(D)$.□

Theorem 5 means that the members of $\mathcal{G}_{k,\sigma}$ admit a characteristic sample made of only flat structures, and that they belong to the result of the BP algorithm whose input is this characteristic sample. This suggests an adaptation of Kanazawa's standard algorithm to learn the class $\mathcal{G}_{k,\sigma}$ from strings (see Algorithm 1). The main difference between this algorithm and Kanazawa's is that not all binary branching structures need to be associated to each string: it is enough to associate flat structures with FA internal nodes. But, to ensure the convergence of this algorithm, flat structures will be associated only to input strings not already recognized (associated with any structure) by the current hypothesis grammars (remember that a^3b^3 is recognized by the CG of Example 8 but not with a flat structure).

Algorithm 1 algorithm to compute elements of $\mathcal{G}_{k,\sigma}$ generating $\langle s_0, ..., s_i \rangle$

Require: $\langle s_0, ..., s_i \rangle$ where $\forall i,\ s_i \in \Sigma^+$ and k
1: $j \longleftarrow 0$
2: **repeat**
3: $C_j = \{s_0, ..., s_j\}$ \\ try C_j as a characteristic sample
4: associate a flat structure with FA internal nodes to every element of C_j
5: apply BP algorithm to find the set $R_{j,k} \subset \mathcal{G}_k$ of CGs compatible with this set
6: discard all elements of $R_{j,k}$ whose string language does not include $\{s_{j+1}, ..., s_i\}$
7: $j \longleftarrow j + 1$
8: **until** $j = i + 1$ OR $R_{j,k} \neq \emptyset$
Ensure: $R_{j,k}$: a set of CGs in $\mathcal{G}_{k,\sigma}$ whose string language includes $\langle s_0, ..., s_i \rangle$

Elements of $\mathcal{G}_{k,\sigma}$ are obtained by applying unifying substitutions to k-valued CG, so they are also at most k-valued. But the k needed by Algorithm 1 may be greater than the one needed by the BP algorithm. Grammars in $\mathcal{G}_{k,\sigma}$ can be considered as having a special normal form: they produce only flat trees or trees that are the result of adjunctions on flat trees (cf Example 8). Theorem 5 ensures that as soon as the input includes a set of strings corresponding to a characteristic sample of flat structures with FA nodes (which will always occur in the limit), then for some j, the set $R_{j,k}$ contains at least one CG generating the target language. Inclusion tests are still necessary to select one grammar.

5 Conclusion

To conclude, this study shows that the domain of regular grammatical inference and the domain of CGs learning can be integrated into a unified framework. The first benefits of this unification is the translation of results from one domain to the other one with very few efforts, and a better understanding of the nature of the generalization operator used in the BP algorithm and Kanazawa's work.

Another less expected result is the introduction of a new class of automata: the class of RA, which naturally extends the class of FSA and allows to represent unidirectional CGs. Unidirectional CGs are better adapted to represent CF-languages for inference from strings than general CGs, because they are more constrained but generate the same class of languages. The class $\mathcal{G}_{k,\sigma}$ is even more interesting because, for each element of this class, there exists a characteristic set of flat structural examples with FA internal nodes only. So, learning this class from strings is not more expansive than learning it from structures. Of course, the expressivity of this class, which is still not known, should be characterized more precisely.

This study shows that the inference of CF-grammars may not be so different from the inference of regular grammars as it first seemed. But a lot remains to be done. A natural perspective is the adaptation of algorithms learning regular languages from positive and negative examples (such as RPNI [13], or Delete [7]) to CF-languages. This requires to define a canonical target. Another possible perspective concerns the adaptation of this work to Lambek grammars.

References

1. D. Angluin. Inference of Reversible Languages *Journal of the ACM* 3: 741–765, 1982.
2. Y. Bar Hillel and C. Gaifman and E. Shamir. On Categorial and Phrase Structure Grammars. *Bulletin of the Research Council of Israel*, 9F, 1960.
3. J. Besombes and J-Y. Marion. newblock Learning Reversible Categorial Grammars from Structures newblock proceedings of *Categorial Grammars* 148–163, 2004.
4. W. Buszkowki and G. Penn. Categorial grammars determined from linguistic data by unification, newblock *Studia Logica*, p. 431–454, 1990.
5. C. Costa-Florencio. Consistent Identification in the Limit of Any of the Classes k-valued Is NP-hard. proceedings of *LACL*, 125-134, LNAI 2099, 2001.
6. C. Costa-Florencio. Consistent Identification in the Limit of Rigid Grammars from Strings Is NP-hard. proceedings of the *IGGI: Algorithms and Applications*, 49–62, LNAI 2484, 2002.
7. F. Denis and A. Lemay and A. Terlutte. Some language classes identifiable in the limit from positive data. proceedings of the *ICGI: Algorithms and Applications*, 63-76, LNAI 2484, 2002.
8. P. Dupont and L. Miclet and E. Vidal. What is the search space of the regular inference. proceedings of *ICGI*, 25–37, LNCS 862, 1994.
9. E.M. Gold. Language identification in the limit. *Information and Control*, 10: 447–474, 1967.
10. G. Huet and C. Retore Survey of a few fundamental representation structures for computational linguistics *ESSLI 2002 lecture.*
11. A. Joshi and Y. Schabes. Tree-Adjoining Grammars. *Handbook of Formal Languages*, 3:69-120. Springer, 1997.
12. M. Kanazawa. *Learnable Classes of Categorial Grammars*, CSLI Publications. 1998.
13. J. Oncina and P. Garcia. *Identifying regular languages in polynomial time*, In *Advances in Structural and Syntactic Pattern Recognition*, vol.5: 99-108, World Scientific, 1992.