RAIRO-Theor. Inf. Appl. **42** (2008) 165–182
DOI: 10.1051/ita:2007055

Available online at:
www.rairo-ita.org

LEARNING DISCRETE CATEGORIAL GRAMMARS FROM STRUCTURES

Jérôme Besombes[1] and Jean-Yves Marion[2]

Abstract. We define the class of discrete classical categorial grammars, similar in the spirit to the notion of reversible class of languages introduced by Angluin and Sakakibara. We show that the class of discrete classical categorial grammars is identifiable from positive structured examples. For this, we provide an original algorithm, which runs in quadratic time in the size of the examples. This work extends the previous results of Kanazawa. Indeed, in our work, several types can be associated to a word and the class is still identifiable in polynomial time. We illustrate the relevance of the class of discrete classical categorial grammars with linguistic examples.

Mathematics Subject Classification. 68Q32, 68T50, 03B47.

In 1988, I was a student in "Maîtrise de Mathématiques Discrètes" at Lyon University, and Serge Grigorieff was one of my professors. I was fascinating by the course on computability that he gave. I followed him when he moved to the university Paris 7 and I made a master thesis on Kolmogorov complexity. Then I made a Ph.D. thesis under his supervision. I got a lot out of our discussions and our works in his small office in Jussieu, and in particular the ability to ask questions and to study relationships between information, complexity and computability. We wrote together a paper on Kolmogorov complexity several years later, and I studied computational linguistic and grammatical inference, with the same spirit. This paper with Jérôme, which is my first *Ph.D.* student, follows this line and is dedicated to him. I owe a lot to him and not only from a scientific point of view...

Keywords and phrases. Classical categorial grammar, grammatical inference, gold's identification in the limit, types, positive examples.

[1] ONERA/DTIM (Traitement de l'Information et Modélisation); jerome.besombes@onera.fr
[2] Nancy-Université, Loria-INPL-ENSMN; Jean-Yves.Marion@loria.fr

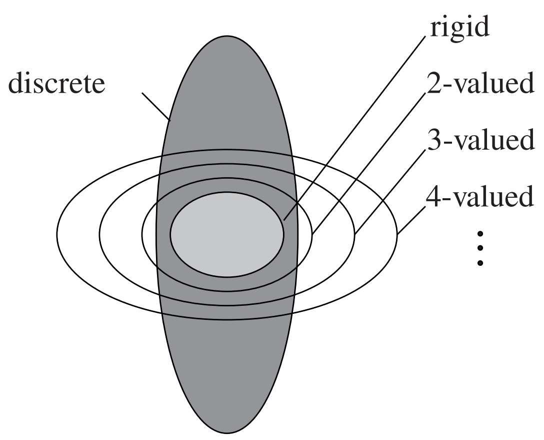


Figure 1. The k-valued hierarchy compare with discrete grammars.

1. Introduction

Classical categorial grammars are widely used in linguistics. Kanazawa [8] established that the class of k-valued categorial grammars is learnable from structured examples, but, for $k > 1$, Florêncio [5] showed that identification of k-valued categorial grammars is NP-hard. This result constitutes a strong limitation in the interest of these classes for the formalization of natural language acquisition which would be feasible. On other hand, categorial rigid grammars ($k = 1$), which are learnable in polynomial time, allow to associate only a single type to each word. There is a strong limitation for considering realistic linguistic phenomena with rigid categorial grammar. In this article, we define a new class of classical categorial grammars, the discrete categorial grammars, which strictly contains rigid grammars and is learnable in quadratic time. This class is completely independent to the hierarchy of k-valued grammars since an arbitrary number of types may correspond to a same word (Fig. 1). Since a pre-defined limit k seems not to be realistic for the formalization of natural language acquisition, the motivation of our work is to define a learnable class of categorial grammars independently from such a limit.

We construct an algorithm which identifies this class of discrete categorial grammars in quadratic time, we give a proof of its correctness and illustrate it on three short examples. The first one is taken from Kanazawa [8] for a comparison with Kanazawa's algorithm. The second and third show how we learn non-rigid grammars for which non-rigidity expresses two kinds of common linguistic ambiguities: homonymy and transitivity and non-transitivity of the same verb.

2. Categorial grammars

2.1. Trees

We consider ordered labeled trees like terms, and inversely. Throughout, we shall write terms linearly or we shall draw them. We think that the use of both notations should help the reader to have a better understanding and to visualize what is going on. In particular, we shall see types, partial parse trees, and FA-structures either linearly as terms or as ordered labeled trees. These notions are of course define in details below.

We now give several definitions about trees that are useful in the rest of the paper. A *context* $t[\diamond]$ is a tree in which the symbol $\diamond$ has at most one occurrence. The symbol $\diamond$ is not in the vocabulary, it marks a node on the border of a tree. For any tree t', we note $t[t']$ the result of the substitution of $\diamond$ by t' in t. The notation $t[t']$ is a convenient way to say that t' is a subtree of t. The size $|t|$ of a tree t is defined by $|a| = 1$ and $|f(t_1, \ldots, t_n)| = 1 + \sum_{i=1,n} |t_i|$.

2.2. Types

Given a set *Var* of primitive types and a special type $\{s\}$ which is not in *Var*, *Tp* is the smallest set of types generated by

- if $x \in Var \cup \{s\}$ then $x \in Tp$,
- if $A \in Tp$ and $B \in Tp$, then $A\backslash B \in Tp$,
- if $A \in Tp$ and $B \in Tp$, then $A/B \in Tp$.

A type A of *Tp* is a *subtype* of a type C of *Tp* if and only if:

- $C = A$,
- or $C = B\backslash B'$ and A is a subtype of B or B',
- or $C = B/B'$ and A is a subtype of B or B'.

A subtype x of a type C is a primitive *subtype* if x is a primitive type of *Var*. A subtype A of a type C is an *argument-subtype* if there is a type B such that either $A\backslash B$ or B/A is a subtype of C. A subtype A of a type C is a *functor-subtype* if there is a type B such that either A/B or $B\backslash A$ is a subtype of C. A *type-context* $A[\diamond]$ is a context of a type.

2.3. Categorial grammars

Bar-Hillel *et al.* [2] introduced classical categorial grammars or AB-grammars. The reader may consult the nice survey of Rétoré [13].

There are several references on categorial grammars, Morrill [12], Moortgat [10]. We present briefly the pre-requisite.

A categorial grammar G is defined from a quadruplet $(\Sigma, Var, \{s\}, \mapsto)$ where

- Σ is a finite vocabulary. Elements of Σ are named letters.
- *Var* is a set of primitive types which plays the role of type variables.
- s is a special type which expresses that a word is recognized by G.

- $\mapsto$ is a finite binary relation over $\Sigma \times Tp$ which defines the lexicon. A lexical entry is written $\alpha \mapsto A$ and means that the letter α is of type A. We shall write $\alpha \mapsto_G A$ when it is necessary to mention G to avoid confusion.

Given a letter α in Σ, $Cat_G(\alpha)$ is the set of types associated to α.

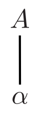
$$Cat_G(\alpha) = \{A \in Tp,\ \alpha \mapsto A\}.$$

We put $Cat(G) = \cup_{\alpha \in \Sigma} Cat_G(\alpha)$ which is the set of all types in the lexicon.

2.4. Partial parse trees

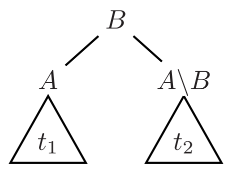
Given a categorial grammar G, a *partial parse tree* for G is any ordered binary tree of the following form.

First, for any $\alpha \in \Sigma$ and any $A \in Tp$ such that $\alpha \mapsto A$, we have

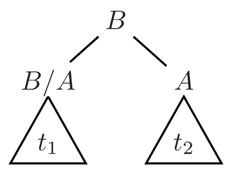


As we have previously explained, we shall note the tree above $A(\alpha)$.

Second, for any partial parse tree t_1 (called argument subtree) and t_2 (called functor subtree) where the root node of t_1 and the root node t_2 are respectively labeled by the types A and $A\backslash B$.



Third, for any partial parse tree t_1 (called functor subtree) and t_2 (called argument subtree) where root node of t_1 and root node t_2 are respectively labeled by B/A and A.



When no ambiguity can arise, we shall simply write $B(t1_1, t_2)$ to abbreviate both previous partial parse tree shape.

The set of partial parse trees of G is $Parse(G)$. As previously, a *parse-context* $t[\diamond]$ is a partial parse tree t in which the symbol $\diamond$ has at most one occurrence.

2.5. Parse trees and languages

A *parse tree* is a partial parse tree which root node is labeled by the type s. The set of parse trees of a grammar G is noted $PT(G)$. We have $PT(G) \subseteq Parse(G)$. Tiede [16] explains that both $PT(G)$ and $Parse(G)$ are regular tree languages.

A parse tree t yields a word of Σ^*. For this, we define a mapping w from $PT(G)$ to Σ^* which extracts a word from a partial parse tree, as follows.

$$\begin{aligned} w(A(\alpha)) &= \alpha \\ w(B(t_1, t_2)) &= w(t_1) \cdot w(t_2) \qquad\qquad \text{where } (\Sigma, \cdot) \text{ is a free monoid.} \end{aligned}$$

Here $A(\alpha)$ is the linear notation of the first kind of partial trees drawn above. And $B(t_1, t_2)$ corresponds to the second and third kind of partial parse trees.

The language produced by a grammar G is the set

$$L(G) = \{u \ : \ w(t) = u \text{ where } t \in PT(G)\}.$$

A type A is *useless* for a grammar G if there is no parse tree with a node labeled by A. Throughout the following discussion, we assume wlog that we are talking about categorial grammar without useless types.

3. FA-structures

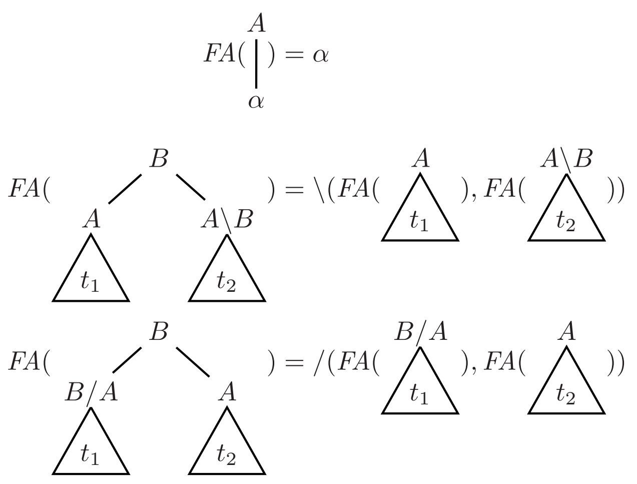
3.1. Structural examples

A consequence of Gold's study [7] is that we can not infer word languages generated by categorial grammars. To cope with this problem, several authors have suggested to learn from a class of grammars, like Sakakibara [14] for context free grammars or Kanazawa [8] for classical categorial grammars, rather than from a class of languages. Like this, positive examples are annoted by additional informations which are related to grammars. We call them *structural examples.* We consider functor-argument structures, used by Kanazawa [8], which we call FA-structures all along the paper.

3.2. Partial FA-structures

A *partial FA-structure* is a tree $FA(t)$ obtained from a partial parse tree t by replacing the label of each node by / if the argument node is the right son, by \ if the argument node is its left son and by the leaf label if the son is a leaf.

Formally, *FA* is a mapping defined inductively as follows:



$FA(\ ) = \alpha$
$FA(\ ) = \backslash(FA(\ ), FA(\ ))$
$FA(\ ) = /(FA(\ ), FA(\ ))$
A
α
B
$A\backslash B$
B/A
t_1
t_2

3.3. FA-structures

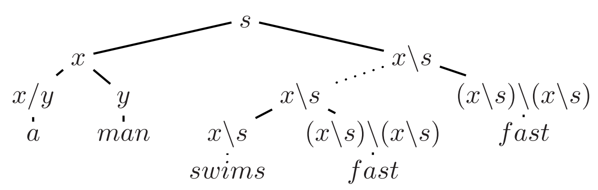
A FA-structure is a partial FA-structure obtained from a parse tree. The set of FA-structures is the functor-argument tree language $FA(G)$. A *FA-context* $f[\diamond]$ is a FA-structure in which there is a hole marked by the special variable $\diamond$.

Remark 3.1. Actually, a way of understanding FA-structure is to see them as semantic informations among the word of a sentence. Thus, we can see FA-structures as dependency tree languages where the functor indicates the head. It is worth mentioning the work of Moortgat [11] which shed some light on the relationships between dependencies and categorial grammars. Another related work is the one of Dudau-Sofronie [6] and of Tellier [15] in which they suggest to label sentences by partial Montague semantics informations.

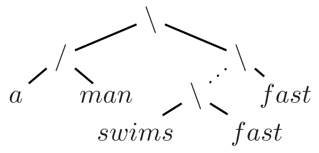
Example 3.2. The first example comes from Kanazawa [8]. The categorial grammar G above will be our running example.

$$\begin{array}{rclrcl} a & \mapsto & x/y & man & \mapsto & y \\ swims & \mapsto & x\backslash s & fast & \mapsto & (x\backslash s)\backslash(x\backslash s). \end{array}$$

We obtain $PT(G)$ the set of trees of the form:



FA(G) the set of trees of the form:



and

$$L(G) = \{\text{a man swims } \underbrace{\text{fast} \ldots \text{fast}}_{\text{n} \geq 0}\}.$$

3.4. Substitutions and FA-equivalence

A *substitution* σ is a mapping from *Var* to *Tp* that we extend canonically over *Tp* as follows.

$$\begin{aligned} \sigma(s) &= s \\ \sigma(A\backslash B) &= \sigma(A)\backslash\sigma(B) \\ \sigma(B/A) &= \sigma(B)/\sigma(A). \end{aligned}$$

Next, we define the image $\sigma(G)$ of a categorial grammar G by applying σ to each type of G. In other words, $\sigma(G)$ is defined by the relation $\alpha \mapsto_{\sigma(G)} \sigma(A)$ iff $\alpha \mapsto_G A$.

Define $G \subseteq G'$ if there is a substitution σ such that for each lexical entry $\alpha \mapsto_G A$, we have $\alpha \mapsto_{G'} \sigma(A)$. Of course, G' has possibly more lexical entries.

Lemma 3.3. [4] *$G \subseteq G'$ implies $FA(G) \subseteq FA(G')$.*

Proof. By the previous lemma, there is a substitution σ such that $\sigma(\mathit{Parse}(G)) \subseteq \mathit{Parse}(G')$. Take $f \in FA(G)$. There is $t \in \mathit{Parse}(G)$ such that $f = FA(t)$. Then, we have $f = FA(\sigma(t))$ because the *FA* mapping erases types, but the shape remains identical. We conclude that $f \in FA(G')$. □

Two grammars G and G' are *FA-equivalent* if and only if $FA(G) = FA(G')$. An important point is that if $FA(G) \subseteq FA(G')$ then $L(G) \subseteq L(G')$. So, if G and G' are FA-equivalent then they recognize the same word language, that is $L(G) = L(G')$.

4. Flat categorial grammars

We associate to each type A a rank $rk(A)$ which gives its level of functionality.

$$\begin{aligned} rk(A) &= 0 && A \in \mathit{Var} \cup \{s\} \\ rk(B\backslash C) &= \max\{rk(B)+1, rk(C)\} \\ rk(C/B) &= \max\{rk(B)+1, rk(C)\}. \end{aligned}$$

A grammar G is *flat* if for each lexical entry $\alpha \mapsto A$, the rank of A is at most 1. In this way, each argument-subtype is of rank 0 and the label of the root of each argument subtree of a partial parse tree is a type of rank 0.

Lemma 4.1. *Let G be a categorial grammar. There is a flat categorial grammar $Level(G)$ and an injective substitution σ such that $G = \sigma(Level(G))$.*

Proof. The transformation $Level(G)$ is described by algorithm 1. The algorithm terminates because an argument-type of rank > 0 in G is deleted at each loop and the number of argument-types is bounded by the number of subtypes in G. The grammar $Level(G)$ is flat because each argument-type of rank >0 is replaced by a new primitive type. It is not difficult to see that the output substitution σ is injective and satisfies $G = \sigma(Level(G))$. ☐

Algorithm 1 The transformation $Level(G)$

```
Input: a categorial grammar $G$
Output: a flat categorial grammar $Level(G)$ and a injective substitution $\sigma$ such
that $G = \sigma(Level(G))$
**while** there is an argument-subtype $A$ s.t. $rk(A) > 0$ **do**
  Choose a new primitive type $a \notin Var$,
  Set $Var = Var \cup \{a\}$
  Set $\sigma(a) = A$
  For each $\alpha \mapsto B[A]$,
      if $A$ is an argument subtype of $B$ then replace $\alpha \mapsto B[A]$ by $\alpha \mapsto B[a]$
      if $A$ is a functor subtype of $B$ or $B = A$ then add $\alpha \mapsto B[a]$ and keep
  also $\alpha \mapsto B[A]$.
**end while**
```

Theorem 4.2. *Let G be a categorial grammar. Then, $Level(G)$ and G are FA-equivalent.*

Moreover, the size of $Level(G)$ is linearly bounded in the size of G. The size of a grammar is $\sum_{\alpha \mapsto A} |A|$.

Proof. First, $FA(Level(G)) \subseteq FA(G)$ is a consequence of Lemma 4.1 because $G = \sigma(Level(G))$.

Conversely, take a lexical entry $\alpha \mapsto_G B/A$ (or $\alpha \mapsto_G A\backslash B$). There is a lexical entry $\alpha \mapsto_{Level(G)} B'/a$ which corresponds to it such that $\sigma(a) = A$ and $\sigma(B') = B$. Indeed, if the rank of A is 0, then A is unchanged and $a = A$. Otherwise, the rank of A is > 0, and A is replaced by a primitive type a and $\alpha \mapsto_{Level(G)} B'/a$ is added to the lexicon of $Level(G)$ with $\sigma(a) = A$. The type B is transformed into B' in a similar way. As a consequence, we see that the shape of each type of G is retained by the transformation which leads to $Level(G)$.

Now, take a partial parse tree t of G. Replace each leaf type C by the corresponding type c of $Level(G)$ of the same shape satisfying $\sigma(c) = C$. We obtain a partial parse tree t' of $Level(G)$ such that $\sigma(t') = t$. We have $Parse(G) \subseteq \sigma(Parse(Level(G)))$. So we conclude that $FA(G) \subseteq FA(Level(G))$. $\square$

Example 4.3. The grammar G given in example 3.2 is not flat since the type $(x\backslash s)$ is an argument subtype of $(x\backslash s)\backslash(x\backslash s)$. The FA-equivalent flat grammar $G' = Level(G)$ is defined by:

$$G' : \begin{array}{rcll} a & \mapsto & x/y & \\ man & \mapsto & y & \\ swims & \mapsto & z, & x\backslash s \\ fast & \mapsto & z\backslash(x\backslash s), & z\backslash z \end{array}$$

where z is the new primitive type introduced by the algorithm.

In conclusion, each categorial grammar G can be translated into a flat categorial grammar $Level(G)$ whose size is the same than G up to a linear constant. The fact that G and $Level(G)$ are FA-equivalent is crucial. Indeed from the point of view of grammatical inference from FA-structure, it implies that we can not distinguish between G and $Level(G)$. We lose nothing by working on $Level(G)$ because it has the same FA-structures than G.

Remark 4.4. We shall henceforth consider only flat categorial grammars. About grammar translation, the reader may consult Le Nir [9] which embedded fragments of Lambek calculus into categorial grammars.

5. Discrete categorial grammars

A set Γ of types is *discrete* if no two types of Γ differ in only one primitive subtype. In other words, a set Γ of types is *discrete* if for each type-context $A[\diamond]$ of Γ, there is at most one primitive type a such that $A[a]$ is in Γ.

A grammar G is *discrete* if G is flat and if for each $\alpha \in \Sigma$, the set $Cat_G(\alpha)$ is discrete.

Example 5.1. The grammar G' of example 4.3 is discrete. The following flat categorial grammar G'' is not discrete, since $fast \mapsto A[z_1]$ and $fast \mapsto A[z_2]$, where $A[\diamond]$ is the type-context $\diamond\backslash(x\backslash s)$.

$$G'' : \begin{array}{rclll} a & \mapsto & x/y & & \\ man & \mapsto & y & & \\ swims & \mapsto & z_1, & x\backslash s & \\ fast & \mapsto & z_1\backslash(x\backslash s), & z_2\backslash(x\backslash s), & z_1\backslash z_2 \end{array}$$

$L(G'') = \{a\ man\ swims,\ a\ man\ swims\ fast,\ a\ man\ swims\ fast\ fast\}$.

Lemma 5.2. *Assume that G is discrete categorial grammar. Let f be a partial FA-Structure in $FA(G)$ and $A[\diamond]$ be a type-context. Then, there is at most one partial parse tree t of G which satisfies $f = FA(t)$ and whose root is labeled by $A[a]$ where a is a primitive type.*

Note that the existence of t implies that the primitive type a is unique.

Proof. The proof goes by induction on the size of the FA-structure f.

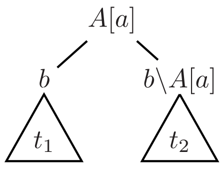
Suppose that f is a letter α. Since G is discrete, there is at most one lexical entry $\alpha \mapsto A[a]$ where a is a primitive type. In this case, $A[a](\alpha)$ is the unique partial parse tree such that $f = FA(A(\alpha)) = \alpha$.

Next, suppose that $|f| > 1$. By definition of FA-structures, f is of one of the following forms: (i) $f = \backslash(g', g)$ or (ii) $f = /(g, g')$. Consider the first case. The second case is treated symmetrically.

Suppose that there is a partial parse tree t_1 such that $FA(t_1) = g'$. Suppose that the root of t_1 is labeled by b. Since G is flat and since b is an argument type. The induction hypothesis on g' and $\diamond$ claims that t_1 and b are unique.

Now, consider the other branch g of f. Again by induction hypothesis on g and $A[\diamond]\backslash b$, we know that there is at most one partial tree t_2 such that $g = FA(t_2)$ and whose root is labeled by the type $A[a]\backslash b$ for some primitive type a.

We conclude that if t_1 and t_2 exist, then they both form a unique partial parse tree t of root $A[a]$ such that $f = FA(t)$.

$t =$


□

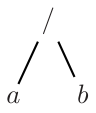
Theorem 5.3. *Assume that G is a discrete categorial grammar. Let f be a partial FA-Structure in $FA(G)$ and A be a type. Then, there is at most one partial parse tree t which satisfies $f = FA(t)$ and the root of t is labeled by A.*

Proof. It is just a consequence of the above Lemma by setting $A[\diamond] = A$, that is $\diamond$ does not occur in A. □

Remark 5.4. Take a FA-structure f of a discrete grammar. Theorem 5.3 claims that to any FA-structure f corresponds at most an unique parse-tree. Indeed, the type of such parse tree is necessarily s. This result does not hold for the case of partial parse-trees. For instance, if we consider the following discrete grammar:

$$G^{(3)} : \begin{array}{ccll} a & \mapsto & (s/x_2)/x_1, & x_4/x_1 \\ b & \mapsto & x_1 & \\ c & \mapsto & x_2, & s/x_4. \end{array}$$

To the FA-structure:



correspond the both partial parse-trees:

$$\begin{tikzcd}
 & s/x_2 \arrow[dl, no head] \arrow[dr, no head] & & & x_4 \arrow[dl, no head] \arrow[dr, no head] & \\
(s/x_2)/x_1 \arrow[d, no head] & & x_1 \arrow[d, no head] & x_4/x_3 \arrow[d, no head] & & x_3 \arrow[d, no head] \\
a & & b & a & & b
\end{tikzcd}$$

This property differs from rigid categorial grammars considered by Kanazawa [8] in the following respect. For each partial FA-structure f of a rigid categorial grammar, there is a unique parse tree t such that $f = \mathit{FA}(t)$. This statement does not hold when we consider discrete categorial grammars.

Given a FA-context $f[\diamond]$, we define $\mathit{Cat}_G(f[\diamond])$ as the set of types A such that there is a partial parse tree t whose root is labeled by A which satisfies $f[FA(t)]$ is a FA-structure of G.

Theorem 5.5. *Assume that G is a discrete categorial grammar. Let $f[\diamond]$ be a FA-context of G. Then, $\mathit{Cat}_G(f[\diamond])$ is a singleton.*

Proof. The proof goes by induction on the size of $f[\diamond]$. The base case is when $f = \diamond$. Then, we have $\mathit{Cat}_G(\diamond) = \{s\}$. We have to consider several cases.

Assume that $f[\diamond] = h[\backslash(\diamond, g)]$.

By induction hypothesis, the type B corresponding to the FA-context $h[\diamond]$ is unique, that is $\mathit{Cat}_G(h[\diamond]) = \{B\}$. Take a partial parse tree t such that $g = \mathit{FA}(t)$. The type of the root of t is necessarily $a\backslash B$ for some type a. Now, the type a is primitive because G is flat. So, we can apply Lemma 5.2 which implies that a is unique. We have $\mathit{Cat}_G(f[\diamond]) = \{a\}$.

Assume that $f[\diamond] = h[/(g, \diamond)]$. This case is similar to the previous one.

Assume that $f[\diamond] = h[\backslash(g, \diamond)]$.

By induction hypothesis, the type B corresponding to the context $h[\diamond]$ is unique. By Lemma 5.2, there is a unique partial parse tree t such that $g = \mathit{FA}(t)$. We name a the type of the root of t. It follows that $\mathit{Cat}_G(f[\diamond]) = \{a\backslash B\}$.

Assume that $f[\diamond] = h[/(\diamond, g)]$. This case is similar to the previous one. □

6. Rigid grammars

Before going further, there is a certain interest in discussing about k-valued categorial grammars introduced by Kanazawa [8] and in seeing how they differ from discrete categorial grammars. The discussion is summed up un Figure 1. Here, and throughout, we consider the set of parse trees generated by grammars in order to compare the generative capacity of classical categorial grammars.

A categorial grammar G is *k-valued* if for any word $\alpha \in \Sigma$, there are at most k lexical entries. A 1-valued categorial grammar is also called *rigid*.

In fact, the class of discrete categorial grammars strictly contains the class of rigid ones.

Theorem 6.1. *For any rigid categorial grammar G, there is a discrete grammar G' which is FA-equivalent.*

Proof. The flat categorial grammar $\mathit{Level}(G)$ is discrete. Lemma 4.1 implies that the substitution σ such that $G = \sigma(\mathit{Level}(G))$ is injective. If $\alpha \mapsto A[a]$ and $\alpha \mapsto A[b]$ are two lexical entries of $\mathit{Level}(G)$, it means that $a = b$ because G is rigid and so contains only one entry $\alpha \mapsto \sigma(A[a])$. Put $G' = \mathit{Level}(G)$ to conclude. □

Consider the following discrete categorial grammar.

$$\begin{aligned} \mathit{John} &\mapsto x \\ \mathit{is\_eating} &\mapsto x\backslash s, (x\backslash s)/y \\ \mathit{a\_chicken} &\mapsto y. \end{aligned}$$

The verb *is_eating* is a transitive verb, which corresponds to $(x\backslash s)/y)$ and non-transitive, which corresponds to $x\backslash s$. There is no rigid-grammar which can encode this grammatical phenomenon.

On the other hand, there are k-valued grammars which are not discrete, like the grammar G'' in Example 5.1.

It would be interesting to see the notion of discrete categorial grammar can be extended to Lambek calculus in the spirit of the ideas of Bonato and Retoré [3] which adapted rigidity to Lambek calculus.

7. Discrete grammars are learnable

7.1. Characteristic samples

Let G be a target categorial grammar, that is the grammar that we try to guess from a finite sequence of FA-structures of G. A characteristic sample is a set of FA-structures which are sufficient to infer G when G is discrete.

We start by some preliminary definitions on partial parse trees in order to define FA-structures which will be in a characteristic sample.

(1) We associate to each type A a partial parse tree $\mathit{root}(A)$ whose root is labeled by A.
(2) We associate to each type A a parse-context $\mathit{leaf}_A[\diamond]$ which is recursively defined as follows.
   - $\mathit{leaf}_s[\diamond] = \diamond$.
   - If A is a primitive type a, $\mathit{leaf}_a[\diamond]$ is a parse-context such that $\mathit{leaf}_a[\mathit{root}(a)]$ is parse tree.
   - If $A = B/a$ where a is a primitive type,

$$\mathit{leaf}_A[\diamond] = \mathit{leaf}_B\left[\begin{array}{ccc} & B/a & \\ \diagup & & \diagdown \\ \diamond & & \mathit{root}(a) \end{array}\right].$$

- If $A = a\backslash B$, where a is a primitive type,

$$\mathit{leaf}_A[\diamond] = \mathit{leaf}_B \left[\begin{array}{ccc} & \backslash & \\ / & & \diagdown \\ \mathit{root}(a) & & \diamond \end{array}\right].$$

A finite set $\mathcal{C}$ of FA-structures is said to be *a characteristic sample* for a flat grammar G if

- for each subtype A of a type in $\mathit{Cat}(G)$, $\mathit{FA}(\mathit{leaf}_A[\mathit{root}(A)]) \in \mathcal{C}$,
- and for each lexical entry $\alpha \mapsto A$ of G, $\mathit{FA}(\mathit{leaf}_A[\alpha]) \in \mathcal{C}$.

We may easily check that $\mathcal{C} \subseteq \mathit{FA}(G)$.

Lemma 7.1. *Assume that G and G' are two discrete categorial grammars. Let $\mathcal{C}$ be a characteristic sample of G. If $\mathcal{C} \subseteq \mathit{FA}(G')$, then $G \subseteq G'$.*

Proof. We define a substitution σ which will satisfy the fact that $\sigma(G)$ is embedded in G'.

For any primitive type a of G, $\mathit{FA}(\mathit{leaf}_a[\mathit{root}(a)])$ is in $\mathcal{C}$.

Since $\mathcal{C} \subseteq \mathit{FA}(G')$, $\mathit{FA}(\mathit{leaf}_a[\mathit{root}(a)])$ is a FA-structure of G'. Furthermore, $\mathit{FA}(\mathit{root}(a))$ is necessarily an argument subtree of $\mathit{FA}(\mathit{leaf}_a[\mathit{root}(a)])$.

So we can apply Lemma 5.2 which states that there is a unique partial parse tree t of G' such that $\mathit{FA}(\mathit{root}(a)) = \mathit{FA}(t)$. Let a be the primitive type which labels the root of t. We put $\sigma(a) = a'$.

We show by induction on A that $\mathit{Cat}_{G'}(\mathit{FA}(\mathit{leaf}_A[\diamond])) = \{\sigma(A)\}$. The unicity is a consequence of Lemma 7.1.

- if A is a primitive type, the result is obvious by the definition of σ
- if $A = a\backslash B$. By definition, we have

$$\mathit{leaf}_A[\diamond] = \mathit{leaf}_B[\diamond] \left[\begin{array}{ccc} & \backslash & \\ / & & \diagdown \\ \mathit{root}(a) & & \diamond \end{array}\right].$$

  By induction hypothesis, $\mathit{Cat}_{G'}(\mathit{leaf}_B[\diamond]) = \{\sigma(B)\}$. As we have previously said, the type associated to $\mathit{FA}(\mathit{root}(a))$ is $\sigma(a)$. So, we have

$$\mathit{Cat}_{G'}(\mathit{leaf}_{a\backslash B}[\diamond]) = \sigma(a)\backslash\sigma(B) = \sigma(a\backslash B)$$

- if $A = B/a$. This case is identical and so we skip it.

Lastly, consider $\alpha \mapsto_G A$. We have established that $\mathit{Cat}_{G'}(\mathit{FA}(\mathit{leaf}_A[\diamond])) = \{\sigma(A)\}$. So, $\alpha \mapsto_{G'} \sigma(A)$ is a lexical entry of G'. □

7.2. An inference algorithm

Following Gold [7] and Kanazawa [8], we present the definition of identification in the limit from positive FA-structure examples. A *positive presentation* of a categorial grammar G is a sequence $f_1, f_2 \ldots$ which enumerates each FA-structure of $\mathit{FA}(G)$. An inference algorithm I takes as input a finite set $S = \{f_1, \ldots, f_n\}$

of a positive presentation of G and guesses a grammar $I(S)$. Given a positive presentation, the inference algorithm I converges if there is a stage N such that for all $n > N$, $L(I(f_1, \dots, f_n)) = L(G)$. Gold [7] established that there is no such inference algorithm for a class of grammars as broad as the class of categorial grammars. For this reason, the success criterion is restricted to a particular class of categorial grammars. A class of categorial grammars is *identifiable* if and only if there is an inference algorithm I such that for each positive presentation, I converges.

We present an efficient inference algorithm *infer* for the class of discrete categorial grammars. This inference algorithm is described in 2.

Algorithm 2 The inference algorithm *infer*

```
Input: a finite set of FA-structures $S$
Output: a discrete categorial grammar $infer(S)$
for each $f \in S$ do
Construct a parse tree $t$ by decorating the root of $f$ with $s$ and each argument node by a new type variable.
Define $G_0$ by collecting all the lexical entries $\alpha \mapsto A$ obtained on each $t$.
end for
Let $\sigma_0$ be the identity substitution.
while there is $\alpha \mapsto A[a_1]$ and $\alpha \mapsto A[a_2]$ where $a_1$ and $a_2$ are two distinct primitive types do
  Set $\sigma_{i+1}(a_2) = a_1$ and $\forall a \neq a_2, \sigma_{i+1}(a) = a$
  Set $G_{i+1} = \sigma_i(G_i)$
end while
```

Lemma 7.2. *Algorithm 2 terminates on each finite set S of FA-structures and outputs a discrete categorial grammar infer(S).*

Proof. Starting from G_0, we compute a sequence of flat categorial grammars $G_1, \dots, G_n$ by merging two variable types at each step. The process stops after n steps because at each step we decrease by one the number of variable types and so we write *infer*$(S) = G_n$. The categorial grammar *infer*(S) is discrete because when we exit the while loop, there is no more lexical entries $\alpha \mapsto A[a_1]$ and $\alpha \mapsto A[a_2]$ such that $a_1 \neq a_2$. □

Lemma 7.3. *Assume that S is a set of FA-structures of a categorial grammar G. Let infer(S) be the discrete categorial grammar computed by algorithm 2. We have infer$(S) \subseteq G$.*

Proof. First, we have $G_0 \subseteq G$. Then, we also have $G_i \subseteq G_0$ because $G_i = \sigma_i(G_{i-1})$. So, we have $G_i \subseteq G$. We conclude that *infer*$(S) \subseteq G$. □

Theorem 7.4. *The class of discrete categorial grammars is identifiable from FA-structures.*

Proof. Assume that G is a discrete categorial grammar. We consider a positive presentation $f_1, \dots, f_n, \dots$ of G. There is a stage N such that for each $n \geq N$, the set $S = \{f_1, \dots, f_n\}$ contains a characteristic sample. We see that $S \subseteq FA(infer(S))$. Lemma 7.1 implies that $G \subseteq infer(S)$.

Conversely, Lemma 7.3 shows that $infer(S) \subseteq G$.

Lemma 3.3 implies $FA(G) = FA(infer(S))$ and so $L(G) = L(infer(S))$.

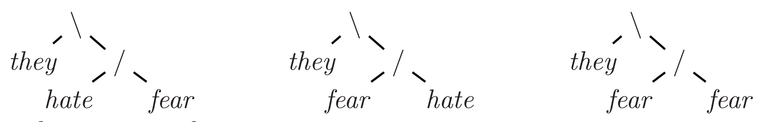
We conclude that *infer* converges. □

The inference algorithm is incremental and runs in quadratic time in the size of the input FA-structures.

Remark 7.5. Based on Angluin [1], characteristic samples are telltale sets for the FA-structure languages of discrete categorial grammars. Therefore, FA-structure languages are identifiable.

7.3. Examples

Example 7.6. Take the following FA-structures.

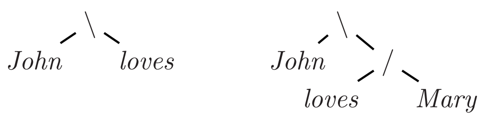


The algorithm outputs the grammar:

$$G: \begin{array}{lcll} they & \mapsto & x & \\ hate & \mapsto & (x\backslash s)/y, & y \\ fear & \mapsto & (x\backslash s)/y, & y. \end{array}$$

This example illustrate a case of homonymy. Words *hate* and *fear* are correctly associated to a verb type ($(x\backslash s)/y$) and a noun type (y). There is no corresponding rigid grammar and the algorithm of Kanazawa fails on this input.

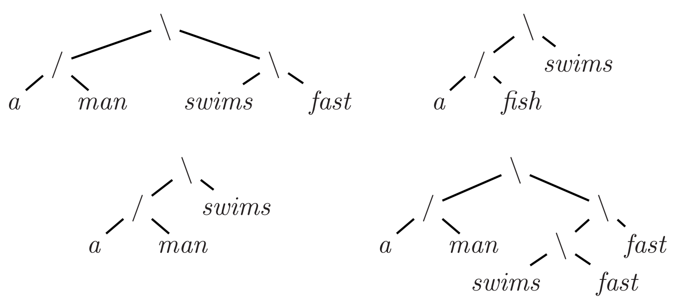
Example 7.7. We come back on the example which shows that the case of a verb with a transitive and a non-transitive form may be treated. For this, it suffices to consider the following FA-structures:



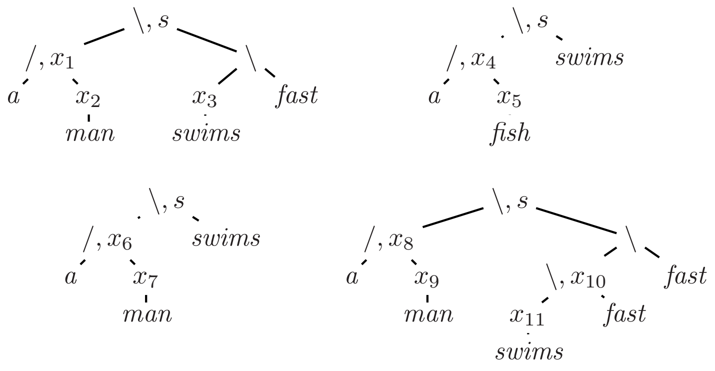
We get the discrete categorial grammar

$$\begin{aligned} John &\mapsto x \\ loves &\mapsto x\backslash s, (x\backslash s)/y \\ Mary &\mapsto y. \end{aligned}$$

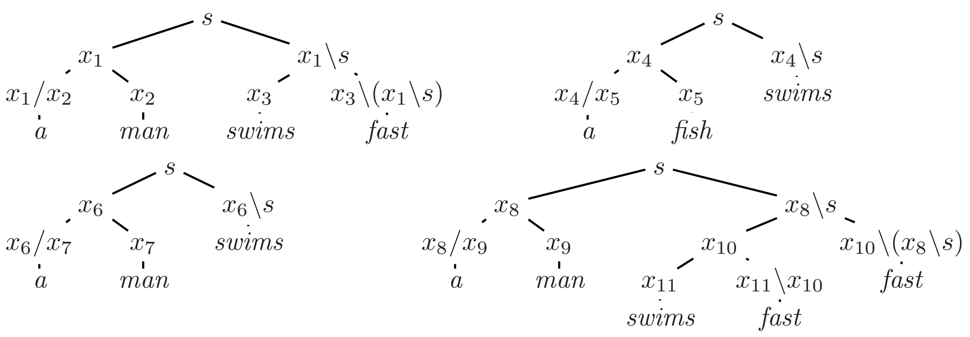
Example 7.8. This example shows how our algorithm identifies the example grammar of [8]. The inputs are the following four FA-structures which form a characteristic set.



First, we label the root node of each argument subtree with a new variable.



According to these labels, trees are completed.



This gives the first grammar G_0 which produces exactly the input FA-structures[1]. This grammar is flat by construction but not discrete. We successively merge

[1]This first step is equivalent to the first step of the RG algorithm described in [8] for the inference of rigid grammars from structures.

primitive types until we obtain a discrete grammar.

$$G_0 : \begin{array}{rclllll} a & \mapsto & x_1/x_2, & x_4/x_5, & x_6/x_7, & x_8/x_9 \\ man & \mapsto & \mathbf{x_2}, & \mathbf{x_7}, & \mathbf{x_9} & \\ fish & \mapsto & x_5, & & & \\ swims & \mapsto & \mathbf{x_3}, & \mathbf{x_4}\backslash s, & \mathbf{x_6}\backslash s, & \mathbf{x_{11}} \\ fast & \mapsto & x_3\backslash(x_1\backslash s), & x_{11}\backslash x_{10}, & x_{10}\backslash(x_8\backslash s) & \end{array}$$

$$\Rightarrow \begin{array}{c} x_2 = x_7 = x_9 \\ x_3 = x_{11} \\ x_4 = x_6 \end{array}$$

$$G_1 : \begin{array}{rclllll} a & \mapsto & \mathbf{x_1}/x_2, & x_4/\mathbf{x_5}, & \mathbf{x_4}/\mathbf{x_2}, & \mathbf{x_8}/x_2, \\ man & \mapsto & x_2 & & & \\ fish & \mapsto & x_5 & & & \\ swims & \mapsto & x_3, & x_4\backslash s & & \\ fast & \mapsto & x_3\backslash(x_1\backslash s), & x_3\backslash x_{10}, & x_{10}\backslash(x_8\backslash s) & \end{array}$$

$$\Rightarrow \begin{array}{c} x_2 = x_5 \\ x_1 = x_4 = x_8 \end{array}$$

$$G_2 : \begin{array}{rcllll} a & \mapsto & x_1/x_2 & & \\ man & \mapsto & x_2 & & \\ fish & \mapsto & x_2 & & \\ swims & \mapsto & x_3, & x_1\backslash s & \\ fast & \mapsto & \mathbf{x_3}\backslash(x_1\backslash s), & x_3\backslash x_{10}, & \mathbf{x_{10}}\backslash(x_1\backslash s) \end{array}$$

$$\Rightarrow x_3 = x_{10}$$

$$G_3 : \begin{array}{rcll} a & \mapsto & x_1/x_2 & \\ man & \mapsto & x_2 & \\ fish & \mapsto & x_2 & \\ swims & \mapsto & x_3, & x_1\backslash s \\ fast & \mapsto & x_3\backslash(x_1\backslash s), & x_3\backslash x_3 \end{array}$$

The grammar G_3 is discrete and the process stops. We remark that G_3 is a renaming of the grammar G' given in example 4.3.

References

[1] D. Angluin, Inference of reversible languages. *J. ACM* **29** (1982) 741–765.

[2] Y. Bar-Hillel, C. Gaifman, and E. Shamir, On categorial and phrase structure grammars. *Bulletin of Research Council of Israel* F(9) (1960) 1–16.

[3] R. Bonato and C. Retoré, Learning rigid lambek grammars and minimalist grammars from structured sentences, in *Third Learning Language in Logic Workshop (LLL2001)* (2001).

[4] W. Buszkowski and G. Penn, Categorial grammars determined from linguistic data by unification. *Studia Logica* **49** (1990) 431–454.

[5] C. Costa Florêncio, Consistent identification in the limit of any of the classes -valued is np-hard, in *Logical Aspects of Computational Linguistics*, edited by C. Retoré, P. de Groote, G. Morrill. *Lect. Notes Comput. Sci.* (2001) 125–138.

[6] D. Dudau-Sofronie, Apprentissage de Grammaires Catégorielles pour simuler l'acquisition du Langage Naturel à l'aide d'informations sémantiques. *Ph.D. thesis*, Lille I University (2004).

[7] M.E. Gold, Language identification in the limit. *Inform. Control* **10** (1967) 447–474.

[8] M. Kanazawa, *Learnable classes of Categorial Grammars*. CSLI (1998).

[9] Yannick Le Nir, *Structures des analyses syntaxiques catégorielles. Application à l'inférence grammaticale. Ph.D. thesis*, Rennes 1 University (2003).

[10] M. Moortgat, Categorial type logics, in *Handbook of Logic and Language*. North-Holland, J. van Benthem and A. ter Meulen edition (1996).

[11] M. Moortgat, Structural equations in language learning, in *Logical Aspects of Computational Linguistics*, edited by C. Retoré, P. de Groote, G. Morrill. *Lect. Notes Comput. Sci.* **2099** (2001) 1–16.

[12] G.V. Morril, *Type Logical Grammar: categorial logic of signs*. Kluwer (1994).

[13] C. Rétoré, The logic of categorial grammars. Technical Report 5703, INRIA (2005). `http://www.inria.fr/rrrt/rr-5703.html`

[14] Y. Sakakibara, Efficient learning of context free grammars from positive structural examples. *Inform. Comput.* **97** (1992) 23–60.

[15] I. Tellier, Modéliser l'acquisition de la syntaxe du langage naturel via l'hypothése de la primauté du sens. *Ph.D. thesis*, Lille I University (2005).

[16] H.J. Tiede, Lambek calculus proofs and tree automata. *Lect. Notes Comput. Sci.* **2014** (2001) 251–265.