Learning reversible categorial grammars from structures

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

[1] LORIA
615, rue du jardin botanique
54602 Villers-lès-Nancy, France
[2] INPL-Ecole Nationale Supérieure des Mines de Nancy
Parc de Saurupt,
54042 Nancy CEDEX, France
{Jerome.Besombes, Jean-Yves.Marion}@loria.fr

Abstract. We define the class of reversible classical categorial grammars, similar in the spirit to the notion of reversible class of languages introduced by Angluin and Sakkakibara. We show that the class of reversible 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 reversible classical categorial grammars with linguistic examples.

1 Introduction

Gold's identification from positive examples [5] is a fascinating model of learning. Its importance stems from the fact that linguists like Chomsky or Pinker [7] show that natural language acquisition is based on the analysis of the structures of correct phrases. Our objective is to contribute to the design and the understanding of the formal process of language acquisition. Classical categorial grammars are widely used in linguistics. In [6], Kanazawa established that the class of k-valued categorial grammars is learnable from structured examples, but, for $k > 1$, Costa Florêncio [4] showed that identification of k-valued categorial grammars is NP-hard. This result constitutes a strong limitation in the interest of these class for the formalization of natural language acquisition. In other hand, rigid grammars ($k = 1$), which are learnable in polynomial time, allow to associate only a single type to any given word of the vocabulary. This is an other strong limitation for considering realistic linguistic phenomena. In this article, we define a new class of classical categorial grammars, the reversible 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. Since a pre-defined limit

k seems not to be realistic for the formalisation of natural language acquisition, the motivation of our work is to define a learnable class of categorial grammars independently to such a limit.

We construct an algorithm which identifies this class of reversible 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 [6] 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: homonymity and transitivity and non-transitivity of the same verb.

2 Reversible categorial grammars

Bar-Hillel [9] defined classical categorial grammars which are based on a non-commutative logic (assumptions are totally ordered).

A classical grammar G is a relation between two sets Σ and Tp where:

- Σ is a finite vocabulary
- Tp is a set of types defined from a finite set of primitive types Var as the smallest set verifying:
  - Tp contains a special type s which is not an element of Var (we note Pr the set $Pr = Var \cup \{s\}$ of primitive types),
  - if $a \in Pr$ then $a \in Tp$,
  - if $A \in Pr$ and $B \in Pr$, then $A\backslash B \in Pr$,
  - if $A \in Pr$ and $B \in Pr$, then $A/B \in Pr$

We note $\alpha \mapsto_G A$ if $(\alpha, A) \in G$ and when there is no ambiguity, we simply note $\alpha \mapsto A$. A grammar G is said to be *k-valued* if for any word $\alpha \in \Sigma$, there are at most k distinct types $A_1, \ldots, A_k$ such that $\forall 1 \leq i \leq k,\ \alpha \mapsto A_i$. A grammar $1-$*valued* is also called *rigid*. If α is an element of Σ, the finite set $Cat_G(\alpha)$ is $\{A \in Tp,\ G : \alpha \mapsto A\}$.

A subtype a of a type C is said to be a *primitive-subtype* if a is a subtype of C and a is a primitive type.

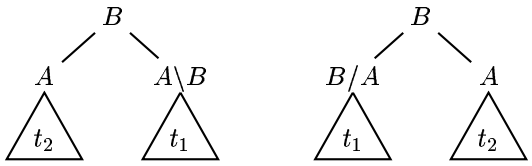
A subtype A of a type C is said to be an *argument-subtype* if there is a type B such that either $A\backslash B$ or B/A is a subtype of C. A subtype A of a type C is said to be an *functor-subtype* if there is a type B such that either A/B or $B\backslash A$ is a subtype of C.

A *type-context* $A[\sharp]$ is a type for which one primitive-subtype is replaced by the symbol $\sharp$. If B is any type, we note $A[B]$ the type obtained by the substitution of $\sharp$ in $A[\sharp]$ by B.

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

$$A(\alpha)$$

for any $\alpha \in \Sigma$ and any $A \in Tp$ such that $G : \alpha \mapsto A$



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 $A\backslash B$ and A for the first tree and B/A and A for the second .

A *parse tree* is a partial parse tree which root node is labelled by s. The set of parse trees of a grammar G is noted $PL(G)$. The language produced by a grammar G is the set $L(G) \subset \Sigma^+$ of strings $\alpha_1 \dots \alpha_n$ corresponding to the ordered leaf nodes of a parse tree. A type A of a set Tp is *useless* for a grammar G is there is no parse tree with a node labelled by A. We only consider grammars with no useless type.

A *FA-structure* is a tree $Structure(t)$ obtained from a partial parse tree t 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 (by definition a leaf son is unique). For a grammar G, the set of FA-structures obtained from the parse-trees is noted $FL(G)$.

Remark 1. If f is a FA-structure and if for any argument subtree of f, a type A_f is given, there exists a unique partial parse tree t such that $Structure(t) = f$ and for any argument subtree f, the label of the root node of f corresponds to A_f.

If f is a FA-structure, we note $Cat_G(f)$ the set of types A such that there exists a partial parse tree t with $Structure(t) = f$ and the root node of t is labelled by A.

Example 1. From [6], let consider the categorial grammar G defined by:

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

The two following trees belong repsectively to $PL(G)$ and $FL(G)$

$$s(x(x/y(a), y(man)), x\backslash s(x\backslash s(x\backslash s(swims), (x\backslash s)\backslash(x\backslash s)(fast)), (x\backslash s)\backslash(x\backslash s)(fast)))$$

$$\backslash(/(a, man), \backslash(\backslash(swims, fast), fast))$$

and

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

A *context* $t[\sharp]$ is a tree for which exactly one subtree is replaced by the symbol $\sharp$. For any tree t', we note $t[t']$ the result of the substitution of $\sharp$ by t' in t. A *parse-context* is a context of partial parse tree and a *FA-context* is a context of a FA-structure.

A *substitution* σ is a function from Var to Tp. A substitution is naturally extended to a function from Tp to Tp by:

$$\sigma(A\backslash B) = \sigma(A)\backslash\sigma(B)$$

and

$$\sigma(B/A) = \sigma(B)/\sigma(A)$$

and defines a grammar $G' = \sigma(G)$:

$$G' : \alpha \mapsto A' \Leftrightarrow \exists A \in Var,\ A' = \sigma(A) \text{ and } G : \alpha \mapsto A$$

Lemma 1 (Buszkowski and Penn [3]). *If G is a grammar and σ a substitution for G, then $FL(G) \subseteq FL(\sigma(G))$.*

A *projection* is a substitution from Var to Var and a *renaming* is a bijective projection.

A type C of one of the following forms is said to be *compact*:

- $C = a$, where a is a primitive type,
- $C = B/a$, where a is a primitive type and B is compact
- $C = a\backslash B$, where a is a primitive type and B is compact

A grammar G is compact if any element of $Cat(G) = \{Cat_G(\alpha) \ : \ \alpha \in \Sigma\}$ is compact.

Lemma 2. *For any categorial grammar G, there exists a compact categorial grammar G', noted $Compact(G)$, such that $FL(G) = FL(G')$.*

Proof (sketch). If G is a categorial grammar and G' the result of the transformation of G by the algorithm 1, then G' is compact and $FL(G) = FL(G')$.

Example 2. The grammar G given in example 1 is not compact since the type $(x\backslash s)$ is an argument subtype of $(x\backslash s)\backslash(x\backslash s)$. The compact grammar corresponding is the grammar G' defined by:

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

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

A set of type $\Gamma \subseteq Tp$ is said to be *reversible* if no two types of Γ differ in only one primitive subtype. More formally, a set of types $\Gamma \subseteq Tp$ is *reversible* if for any type-context $A[\sharp]$, there exists at most one primitive type a such that $A[a]$ is in Γ. A grammar G is *reversible* if G is compact and if for any $\alpha \in \Sigma$, the set $Cat_G(\alpha)$ is reversible.

Algorithm 1 Transformation of a grammar into an equivalent compact grammar

Input: a grammar G
Output: a compact categorial grammar G' such that $FL(G) = FL(G')$
while exists a type $C \in Cat(G)$ and an argument-subtype A of C with $A \in Tp \cap Var$ **do**
  Choose a new primitive type $a \notin Var$,
  For any B in $Cat(G)$, if A is an argument subtype of B then replace A with a in B
  For any B in $Cat(G)$, if A is a functor subtype of B then for all $\alpha \mapsto B$ in G, add $\alpha \mapsto B'$ to G with B' the type obtained by replacing A with a in B ($\alpha \mapsto B$ is kept)
end while

3 Properties of reversible grammars

In this section, we first show that the class of reversible categorial grammars strictly contains the class of rigid grammars. Then, we establish some classical results in grammatical inference; these results are constituting the basic toolkit for proving the learnability of a class of languages.

Theorem 1. *For any rigid grammar G, there is a reversible grammar G' such that $FL(G) = FL(G')$.*

Proof (sketch). Let G be any rigid grammar. Then the compact grammar $G' = Compact(G)$ obtained by processing the algorithm defined in the proof of lemma 2 is reversible and $FL(G) = FL(G')$.

Theorem 2. *Let G be a reversible grammar and f a FA-structure in $FL(G)$. There exist an unique parse tree t in $PL(G)$ for G such that $f = structure(t)$.*

We prove this theorem with help of the following Lemma.

Lemma 3. *Let G be a reversible grammar, f a FA-Structure in $FL(G)$ and $A[\sharp]$ a type-context. There exists at most one primitive type a such that $A[a]$ is in $Cat_G(f)$.*

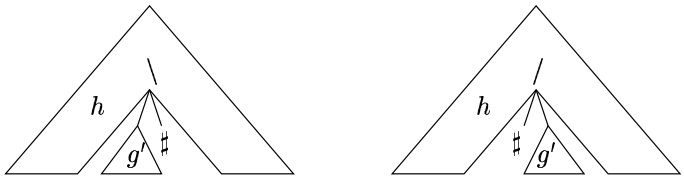
Proof (sketch). The proof goes by induction in the depth of the tree f.

Proof of theorem 2. By the previous lemma, if G is a reversible grammar, Cat_G is a function from the set of argument subtrees of $FL(G)$ to the set of primitive types Tp. Indeed, given an argument subtree g of $FL(G)$ and the empty type-context $\sharp$, there is an unique primitive type a in $Cat_G(g)$. By remark 1, this function induces a bijection between $FL(G)$ and $PL(G)$ (corresponding to the inverse $Structure^{-1}$ of the function $Structure$). □

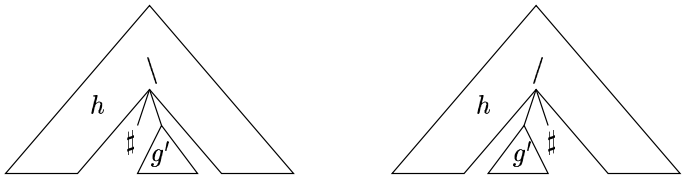
Theorem 3. *Let G be a reversible grammar and $f[\sharp]$ a FA-context for G. There exists an unique type A such that for a FA-structure g, $f[g]$ is in $FL(G)$ ($f[g]$ is a structure for a parse-tree of G) and $Cat_G(g) = A$.*

Proof. The existence of a type A is a direct consequence of the definition of a FA-context. We now have to prove that A is unique. The proof goes by induction on the depth d of the node $\sharp$ in the tree $f[\sharp]$.

- $d = 0$. $f = \sharp$ and if $f[g] = g$ is in $FL(G)$, the we have $Cat_G(g) = s$.
- $d > 0$. Let consider two cases.
  - First case. $f[\sharp]$ is of one of the two following forms:



    If g is such that $f[g]$ is in $FL(G)$, g is an argument subtree of $FL(G)$ and then $Cat_G(g) = a$ is a primitive type which is uniquely defined. By induction hypothesis, the type B corresponding to the context $h[\sharp]$ is unique, which implies that the type A is respectively defined by $A = a\backslash B$ for the left hand figure and by $A = B/a$ for the right hand one.
  - Second case. $f[\sharp]$ is of one of the two following forms:



    By induction hypothesis, the type B corresponding to the context $h[\sharp]$ is unique. By construction, the type corresponding to the functor subtree g' is respectively defined by $A = a\backslash B$ for the left hand figure and by $A = B/a$ for the right hand one. By lemma 3, a is unique (B is defined and $a\backslash B$ or B/a is in $Cat_G(g')$).

We note $Cat_G(f[\sharp])$ the type A as defined in the theorem 3. We may also see Cat_G as a function from the set of FA-contexts to Tp.

Let G be a reversible grammar. Following the classical works [1,8,6], we now define the notion of characteristic sample for a reversible grammar G. For this, we associate to each primitive type a of G a FA-context $Leaf_G(a)$ such that $Cat_G(Leaf_G(a)) = a$ and a FA-structure $Root_G(a)$ such that $a \in Cat_G(Root_G(a))$. For the special case of s, we choose $Leaf_G(s) = \sharp$. For each non-primitive type A, we associate a FA-structure $Root_G(A)$ such that $A \in Cat_G(Root_G(A))$. If A is in $Cat(G)$ ($\alpha \mapsto A$ for a word $\alpha \in \Sigma$), we choose $Root_G(A) = \alpha$. Now for each type A, we recursively define the FA-context $Leaf_G(A)$ by:

- if $A = B/a$, $Leaf_G(A) = Leaf_G(B)[/(\sharp, Root_G(a))]$

- if $A = a\backslash B$, $Leaf_G(A) = Leaf_G(B)[\backslash(Root_G(a), \sharp)]$

A finite set of structures is said to be *a characteristic sample* for a reversible grammar G if it contains, for all types A, the following structure: $Leaf_G(A)[Root_G(A)]$. Such a set is noted $CS(G)$ and we may easily check that $CS(G) \subseteq FL(G)$.

Lemma 4. *Let G and G' be two reversible grammars such that $CS(G) \subset FL(G')$, then $FL(G) \subset FL(G')$.*

Proof. To prove this inclusion, let exhibit a projection σ such that $\sigma(G) \subset G'$. Lemma 1 will provide the conclusion. From the definition of a characteristic sample, for any primitive type a of G there is a FA-structure $Leaf_G(a)[Root_G(a)]$ in $CS(G)$. Since $CS(G) \subset FL(G')$, the FA-context $Leaf_G(a)[\sharp]$ is also a FA-context for G'. Futhermore, $root_G(a)$ is an argument subtree of $Leaf_G(a)[\sharp]$ and G' is reversible, that implies that $Cat_{G'}(Leaf_G(a)[\sharp])$ is a primitive type a' of G'. We define $\sigma(a) = a'$ and σ is extended to non-primitives types by $\sigma(A\backslash b) = \sigma(A)\backslash\sigma(b)$ and $\sigma(b/A) = \sigma(b)/\sigma(A)$. We show by induction in the size of A that for any type A, $Cat_{G'}(Leaf_G(A)) = \sigma(A)$.

Let consider $\alpha \mapsto_G A$. We have $Root_G(A) = \alpha$ and $Cat_{G'}(Leaf_G(A)) = \sigma(A)$ that implies that $\alpha \mapsto_{G'} \sigma(A)$ and finally $\sigma(G) \subseteq G'$

4 The learning algorithm

Following [5], we define the identification in the limit from positive examples. A *positive presentation* of a language $\mathcal{L}$ is a sequence $s\mathtt{t}$ which enumerates each element of $\mathcal{L}$. Let Ω be a given class of grammars. An inference algorithm I takes as input a finite segment $\mathtt{t}_1, \ldots, \mathtt{t}_n$ of a positive presentation of $\mathcal{L}$ and guesses a grammar $I(\mathtt{t}_1, \ldots, \mathtt{t}_n)$. The inference algorithm I converges to $\mathcal{L}$ if there is a stage N such that for all $n > N$, the language provided by $I(\mathtt{t}_1, \ldots, \mathtt{t}_n)$ is exactly $\mathcal{L}$. A class of languages $\mathbb{L}$ is *identifiable* if and only if there is an inference algorithm I such that for each positive presentation of a language $\mathcal{L}$ in $\mathbb{L}$, I converges to $\mathcal{L}$.

A class of categorial grammars Π is said to be *learnable from structures* if the set $FL(\Pi) = \{FL(G) \ : \ G \in \Pi\}$ is identifiable.

Theorem 4. *The class of reversible categorial grammars is learnable from structures.*

Proof. Based on Angluin [1], the characteristic samples are telltale sets for the the structures languages of reversible categorial grammars. Angluin proved that a class such that any language has a telltale set is learnable in the Gold's identification in the limit model. identifiable from structures.

Algorithm 2 Learning algorithm

Input: a finite state of FA-structures $\{f_1, \ldots, f_p\}$
Output: a reversible categorial grammar
Construct G_0 by introducing a new primitive type the root node of each argument subtree of the structures $\{f_1, \ldots, f_p\}$ and completing the trees to obtain parse trees.
while exists a word $\alpha \in \Sigma$ such that $\alpha \mapsto A[a_1]$ and $\alpha \mapsto A[a_2]$ with a type-context A and a_1 and a_2 two distinct primitive types **do**
  $G_n = \sigma_n(G_{n-1})$ with σ the projection defined with $\sigma_n(a_2) = a_1$ and forall $a \neq a_2$, $\sigma_n(a) = a$
end while
Output G_n

The algorithm 2 defines an efficient algorithm for this identification.

Theorem 5. *The algorithm described above identifies the class of reversible grammars from structures.*

Proof. Let $f_1, f_2, \ldots$ be a positive presentation of grammar G. For any integer p, on input $\{f_1, \ldots, f_p\}$, the algorithm calculates a serie of grammars $G_0, G_1, \ldots$. Each G_n is obtained from G_{n-1} by merging variables. The number of different variables of G_0 is finite and so the algorithm stop on a grammar G_n which is trivially reversible. We have $FL(G_0) = \{f_1, \ldots, f_p\}$ and so $\{f_1, \ldots, f_p\} \subseteq FL(G_n)$. There exists a stage P such that $CS(G) \subseteq \{f_1, \ldots, f_p\}$ if $p \geq P$; from Lemma 4, we have $FL(G) \subseteq FL(G_n)$. By construction, G_0 associates a different primitive type to each argument subtype in $\{f_1, \ldots, f_p\}$, there so exists a projection σ such that $\sigma(G_0) \subseteq G$. Each time the algorithm enters the main loop, a projection σ_n is applied. G is reversible that implies $\forall 1 \leq i \leq n$, $\sigma_i(a) = \sigma_i(a') \Rightarrow \sigma(a) = \sigma(a')$. We so have $\sigma = \theta \circ \sigma_n \circ \ldots \sigma_1$, for a mapping θ and $\theta \circ \sigma_n \circ \ldots \sigma_1(G_0) = \theta(G_n) \subseteq G$ that implies $FL(G_n) \subseteq FL(G)$ and finally $FL(G_n) = FL(G)$.

The time complexity of the algorithm is quadratic in the size of the set $\{f_1, \ldots, f_p\}$ given in input. We now illustrate it on three examples.

5 Examples

Example 3. The first example shows how our algorithm identifies the example grammar of [6]. Let consider the four following trees as input.

$$\backslash(\backslash(a, man), \backslash(swims, fast)),\ \backslash(/(a, fish), swims),\ \backslash(/(a, man), swims)$$
$$\text{and } \backslash(/(a, man), \backslash(\backslash(swims, fast), fast))$$

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

$$s(x_1(a, x_2(man)), \backslash(x_3(swims), fast));\ s(x_4(a, x_5(fish)), swims),$$
$$s(x_6(a, x_7(man)), swims),$$
$$s(x_8(a, x_9(man)), \backslash(x_{10}(x_{11}(swims), fast), fast))$$

According to these labels, trees are completed.

$$s(x_1(x_1/x_2(a), x_2(man)), x_1\backslash s(x_3(swims), x_3\backslash(x_1\backslash s)(fast))),$$
$$s(x_4(x_4/x_5(a), x_5(fish)), x_4\backslash s(swims)),$$
$$s(x_6(x_6/x_7(a), x_7(man)), x_6\backslash s(swims)),\ s(x_8(x_8/$$
$$x_9(a), x_9(man)), x_8\backslash s(x_{10}(x_{11}(swims), x_{11}\backslash x_{10}(fast), x_{10}\backslash(x_8\backslash s)(fast))))$$

This gives the first grammar G_0 which products exactly the structures given in the input [1]. This grammar is compact by construction but not reversible. We so successively merge primitive types which are in contradiction with the property of reversiblility.

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

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

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

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

The grammar G_3 is reversible and the process stops. We remark that G_3 is a renaming of the grammar G' given in example 2.

The two following examples illustrate how we manage we some non-rigid grammars.

Example 4. We consider the trees $\backslash(they, /(hate, fear))$ and $\backslash(they, /(fear, hate))$.

The algorithm outputs the grammar:

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

[1] This first step is equivalent to the first step of the RG algorithm described in [6] for the inference of rigid grammars from structures.

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 5. This example shows that the case of a verb with a transitive and a non-transitive form may be treated. Consider the input $\backslash(John, loves)$ and $\backslash(John, /(loves, Mary))$

$$G : \begin{array}{rl} John & \mapsto x \\ loves & \mapsto x\backslash s, (x\backslash s)/y \\ Mary & \mapsto y \end{array}$$

To the verb *loves* corresponds a transitive verb type $((x\backslash s)/y)$ and a non-transitive verb type $(x\backslash s)$. Like in the former example, there exists no corresponding rigid grammar.

We defined an original class of classical categorial grammars which is identifiable from structures in the Gold's model and a quadratic algorithm for this identification. The expressivity of this class allows to consider linguistic ambiguities not yet considered by a polynomial learning algorithm. It's now interesting to see how the algorithm may be adapted to learn the class of reversible Lambek grammars from structures, following the idea of Bonato and Retoré [2] for the adaptation of Kanazawa's algorithm to learn rigid Lambek grammars.

References

1. D. Angluin. Inference of reversible languages. *Journal of the ACM*, 29:741–765, 1982.
2. R. Bonato and C. Retoré. Learning rigid lambek grammars and minimalist grammars from structured sentences. In *Third Learning Language in Logic Workshop (LLL2001)*, 2001.
3. W. Buszkowski and G. Penn. Categorial grammars determined from linguistic data by unification, 1990.
4. C. Costa Florêncio. Consistent identification in the limit of any of the classes -valued is np-hard. In C. Retoré P. de Groote, G. Morrill, editor, *Logical Aspects of Computational Linguistics*, Lecture Notes in Computer Science, pages 125–138. Springer-Verlag, 2001.
5. M.E. Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.
6. M. Kanazawa. *Learnable classes of Categorial Grammars.* CSLI, 1998.
7. S. Pinker. *The language instinct.* Harper, 1994.
8. Y. Sakakibara. Efficient learning of context free grammars from positive structural examples. *Information and Computation*, 97:23–60, 1992.
9. C. Gaifman Y. Bar-Hillel and E. Shamir. On categorial and phrase structure grammars. *Bulletin of Research Council of Israel*, F(9):1–16, 1960.