Finding Missing Categories in Incomplete Utterances *

Mehdi Mirzapour
LIRMM, Université de Montpellier, Montpellier, France
mehdi.mirzapour@lirmm.fr

RÉSUMÉ

Cet article propose un algorithme efficace (en $O(n^4)$) pour trouver la catégorie d'un mot manquant dans un énoncé incomplet. Notre travail fait appel à l'algorithme d'unification comme lors de l'apprentissage des grammaires catégorielles et à la programmation dynamique comme dans l'algorithme Cocke-Younger-Kasami. En utilisant l'interface syntaxique / sémantique des grammaires catégorielles, ce travail peut être utilisé pour dériver les lectures sémantiques possibles d'un énoncé incomplet. Des exemples suivis illustrent notre propos.

ABSTRACT

Finding Missing Categories in Incomplete Utterances

This paper introduces an efficient algorithm ($O(n^4)$) for finding a missing category in an incomplete utterance by using unification technique as when learning categorial grammars, and dynamic programming as in Cocke–Younger–Kasami algorithm. Using syntax/semantic interface of categorial grammar, this work can be used for deriving possible semantic readings of an incomplete utterance. The paper illustrates the problem with running examples.

MOTS-CLÉS : Syntaxe, Grammaires Catégorielles, Inférence grammaticale, Apprentissage, Unification, Programmation Dynamique..

KEYWORDS: Syntax, Categorial Grammars, Grammar Inference, Learning, Unification, Dynamic Programming..

1 Introduction

This paper is a part of the author's Ph.D. proposal regarding possible semantic readings of a given utterance with ranking. An utterance may yield several readings either from ambiguities (lexical, syntactical and semantical) or from syntactic incompleteness. The incomplete utterances can be linguistically considered as a type of non-canonical utterances (Pullum & Scholz, 2001) and they can be caused by different reasons such as missing categories, extra categories, swap categories, misused categories, or any possible combination of mentioned reasons. The focus in this paper is on incomplete utterances with one missing category. To make our research problem clear we should emphasize that by the missing category one can generally suppose two things : a missed word in an incomplete sentence or an unknown word in a sentence with no proper category assignment. In this paper we especially deal with the problem of a missing category for a missing word in an incomplete sentence. Moreover, we assume that the position of the word in the sentence is unknown. So obviously,

∗. This work is thought as a possible complement to the e-fran project AREN (ARgumentation Et Numérique) as a preprocessing technique for the automated analysis of online debates by high-school pupils.

our task is to find the possible position(s) of a missing word and its proper category assignment.

Syntactic analysis of incomplete utterances has been the subject of study in rule-based, data-driven and statistical approaches in computational linguistics (Leacock *et al.*, 2010). The scope of automatic error detection/fixation and robust parsing is so wide and different mechanisms are introduced by researchers such as [1] : over-generating and ranking parse trees (Dini & Malnati, 1993), imposing ranking constraints on grammatical rule Mellish (1989), introducing "mal-rules" (Schneider & McCoy, 1998), relaxing constraints in feature unification (Vogel & Cooper, 1995), modeling grammar as model-theoretic constraints (Blache, 2000; Dahl & Blache, 2004; Prost, 2009), and finally, sub-tree parse fitting for non-parsable utterances. (Jensen *et al.*, 1983). All the mentioned approaches have their own advantages, and address the same issue that we have defined. However, the formalisms that are introduced in them is not the same as what is used in this paper.

Basic Categorial Grammars or AB grammars, is a formalism which is used in this research. It is a family of the categorial grammars. This choice is based on the requirement to bridge the incomplete utterances with semantical readings (author's Ph.D. proposal). There are a number of reasons to explain why categorial grammars are suitable for our specific research topic : Firstly, categorial grammars have a very nice correspondence with semantics and they are very suitable for syntactic-semantic interface modeling ; they can represent semantic ambiguities such as quantifier scope and negation (Moot & Retoré, 2012, Chapter 3) which does not exist in shallow approaches ; secondly, they enjoy learn-ability property from positive sentences which converge after learning a fixed number of sentences by using unification techniques (Buszkowski & Penn, 1990; Kanazawa, 1998) ; thirdly, by applying a left-to-right incremental procedure and proof net construction for categorial grammar we can correctly predict a wide variety of human performance phenomena such as garden-pathing, preference for lower attachment, left-to-right quantifier scope preference (Morrill, 2000) ; finally, there is a wide-coverage syntactic and semantic parsing for French language, named Grail, that allows researchers to design and experiment with multimodal categorial grammars which includes AB grammars as well. (Moot, 2010). Grail is a modern, flexible and robust parser/automated theorem prover that has around 900 different category types for French language and works efficiently with the newspaper articles. Categorial grammar is extracted semi-automatically from the Paris 7 tree-bank and its semantic lexicon maps combinations of words, part-of-speech tags and formulas to Discourse Representation Structures (Kamp & Reyle, 1993). [2]

Our strategy for resolving incomplete utterances is based on using AB grammars as the basis of our syntactic formalism on the one hand, and adopting the notion of unification that exists in RG algorithm (learning Rigid AB grammars) (Buszkowski & Penn, 1990; Buszkowski, 1987) plus some dynamic programming technique on the other hand. This approach, to the best of our knowledge, is a new research technique that can efficiently deal with incomplete utterances with missing categories while benefiting from the good properties in categorial grammars. As we will see, the time complexity for the algorithm that we use to find one missing category with n number of words is $O(n^4)$. This research can be extended to more missing categories, and other kind of incomplete utterances as we will discuss in the conclusion section.

Since this paper is self-contained, we have introduced some basic notions and definitions in section (2) which is essential for understanding AB grammars and notion of categorial unification. Moreover,

1. See (Leacock *et al.*, 2010, Chapter 1) for detailed descriptions.

2. As for the behaviour of the Grail parser on the sentences containing an unknown word we should say that the wide-coverage version of the Grail parser uses a probabilistic model to propose one or more of the contextually most likely options. It is worth mentioning that our proposal can be used for finding missed words ; although it can be adopted for finding categories for unknown words as well.

There are a number of running examples and illustrations on other sections to make notions more clearer. Section (3) discusses unification technique of RG algorithm, and how this mechanism can be adopted for finding the missing category with some examples and the possible combinations of the structural tree. Section (4) illustrate informally how unification technique and the notion of dynamic programming can be tackled with using bottom-up parsing in a modified version of well-known dynamic CYK algorithm. Section (5) concludes our approach and explains possible direction in future studies. We have also included an appendix that contains the pseudo-code of our algorithm provided with discussions on the complexity of the algorithm.

2 Reminder of AB grammars and Unification

Here, AB grammars, and some essential notions are discussed very quickly. As highlighted, this formalism is very suitable for our purpose since it enjoys learn-ability from positive examples and it converges after learning from a fixed number of sentence structures. (Buszkowski & Penn, 1990; Kanazawa, 1998; Bonato, 2000)

Definition 1 : Categories

Taking Lambek's notation (Lambek, 1958), categories in AB grammars can be written as follows (Moot & Retoré, 2012, Chapter 1) :

$$L ::= P \mid (L\backslash L) \mid (L/L)$$

where P is the set of primitive category (such as np and S) while $B\backslash A$ and A/B can be interpreted as the functors that take B as their arguments on its left and right-hand side respectively ; in the all mentioned cases, the result would be A.

A function Lex is a lexicon which maps words to finite set of categories. An expression, that is a sequence of words or terminals $w_1, \cdots, w_n$ is of category u whenever there exists for each w_i a category u_i in $Lex(w_i)$ such that $u_1 \cdots u_n \rightarrow u$ with the left or right elimination rule patterns, namely $(A/B)B \rightarrow A$ and $B(B\backslash A) \rightarrow A$.

It is worth mentioning that variable categories are also used such as $x_1, x_2, \cdots, x_n$ (or $y_1, y_2, \cdots, y_n$) in addition to above fixed category–or what can be properly named fixed learned categories with syntactic labels– for our future purposes in unification phase. See following example a lexicon and the related process which ends up to S :

Example 1 : Lexicon

Jean : np	aime : $(np\backslash S)/np$	Marie : np

Jean aime Marie $\Longrightarrow np,\ (np\backslash S)/np,\ np \Longrightarrow np,\ (np\backslash S) \Longrightarrow S$

Definition 2 : Structural trees with variable categories

It is basically a tree-structured proof representation with variable categories instead of having fixed categories. The only exception is fixed category S in the root of the tree which is naturally expected. Arguments will have unique variable while the functors would get the argument category as its sub-formula in the right or left side. All the possible structural trees of a sequence with 3 words are

represented in figure 1. For convenient, when we use the term structural tree we mean the structural tree with variable categories.

It can be easily noticed that the number of possible trees for a sequence of string with n length would be $catalan(n-1) * 2^{n-1}$ in which $catalan(n-1)$ is the number of possible binary trees and 2^{n-1} is the number of possible operations for $\backslash$ and $/$ over each binary sub-trees. [3] In the case of a sentence with 3 unspecific words, the number of all possible structural trees is 8 ($= catalan(3-1) * 2^2$) as it can be observed in the figure 1.

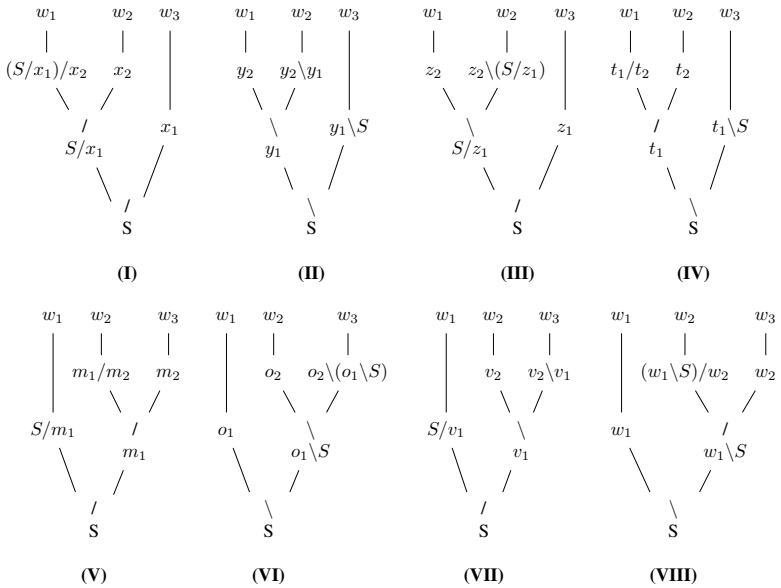

FIGURE 1 – Possible structural trees of a sequence $w_1 w_2 w_3$

Words	Fixed learned categories	I	II	III	IV	V	VI	VII	VIII
Jean	np	$(S/x_1)/x_2$	y_2	z_2	t_1/t_2	S/m_1	o_1	S/v_1	w_1
aime	$(np\backslash S)/np$	x_2	$y_2\backslash y_1$	$z_2\backslash(S/z_1)$	t_2	m_1/m_2	o_2	v_2	$(w_1\backslash S)/w_2$
Marie	np	x_1	$y_1\backslash S$	z_1	$t_1\backslash S$	m_2	$o_2\backslash(o_1\backslash S)$	$v_2\backslash v_1$	w_2

TABLE 1 – Unification of variable categories and fixed (learned) categories

Definition 3 : Pattern matching

Let t be a category which consists of at least one variable category and possibly some fixed ones, and let L be a set of fixed categories, we say t is matchable with set L, if there exist a substitution of t which is a category in L or a sub-category in L. For instance, if we define $L = \{np, S, S/(np\backslash S)\}$

3. The Catalan(n) is given in terms of binomial coefficients by following formula :

$$\mathrm{C}_n = \frac{1}{n+1}\binom{2n}{n} = \frac{(2n)!}{(n+1)!\,n!} = \prod_{k=2}^{n} \frac{n+k}{k} \qquad \text{for } n \geq 0.$$

and $t = np\backslash x_1$, then, since the substitution $x_1 := S$ exists and $np\backslash S$ is a sub-category of $S/(np\backslash S)$ in L, we can say that t is matchable with L [4].

Definition 4 : Derivation of variable categories

Let X, Y and Z be categories with at least one primitive variable and A a category with no primitive variable category (=fixed category). We can define, the derived result of X and A as $Y[X := (Y/A)]$; and the derived result of A and X as $Y[X := A\backslash Y]$; and finally the derived results of X and Y as $Z[Y := (X\backslash Z)]$ or $Z[X := (Z/Y)]$. The notation $t := t^{'}$ means substitution of t with $t^{'}$.

Table 1 illustrates a fixed learned categories (the second column from left), and eight columns showing the variable categories (labeled from I to VIII) which are the yields of derivation trees in figure 1. We want to find one column, let say n (between I to VIII), such that every line k of the fixed learned categories unifies with the k line of the n column. It can be clearly observed that there is only one column (among eight possible cases) which is unifiable with the fixed learned categories and it is (VIII). By simple substitution we can observe that $w_1 = np$, $w_2 = np$, and since $(w_1\backslash S)/w_2$ matches with $(np\backslash S)/np$. One can observe how the rest seven cases fail in the unification phase. So, we can claim that structural tree (VIII) is unifiable with the lexicon and more important, the substitutions of $w_1 = np$ and $w_2 = np$ in the structural tree (VIII) yields its final derived tree.

It is worth mentioning that the technique described above can distinguish the same word used with different categories, and we will also see how the technique we propose can potentially be generalized to k valued AB grammars in the conclusion section. Having these basic notations will allow us to start describing our techniques for dealing with uncompleted utterances.

3 Unification Technique of RG Algorithm

The unification technique which is used here is introduced for the first time in learning rigid AB grammars algorithm (Buszkowski & Penn, 1990) and it has been re-studied and extended by some researchers as well (Kanazawa, 1998; Bonato, 2000; Bonato & Retoré, 2014). We are using almost the same technique, but with a new purpose which is dealing with incomplete utterances with one missing category that to our knowledge is a new field to experience. Since we have explained the essential notions, we just focus on some examples to highlight the mechanism of unification in this section.

Let us consider the following lexicon which is adopted for our illustration purpose, and it is assumed that the lexicon is already learned from the positive sentences in corpus [5]. Now, we consider the following examples where a category is missing. One missing determiner in example (3) and one missing preposition in example (4) :

Example 2 : Lexicon

le , la : np/n	dans : $(s\backslash s)/np$	poisson , mer : n	nage : $np\backslash S$	vite : $(np\backslash S)\backslash(np\backslash S)$

Example 3 : * Poisson nage vite.

4. As might be noted, pattern matching here is very similar to unification. We have adopted this notation to make distinction between unification of two categories, and one category with some external references.

5. See (Moot & Retoré, 2012, Section 1.6.3) for technical details of learning AB rigid grammar; although, this is not essential for grasping the content of this section.

Example 4 : * Le poisson nage vite la mer.

Let us start with example (3). As it is illustrated in figure 2 we have illustrated only three cases. (Just note that without knowing the categories of words there would be 160 ($= catalan(3) * 2^3 * (3+1)$) possible functor-argument tree structures that can be associated to example 3).

We can unify the gained variable categories associated to the words in our structural trees with the one we have in our lexicon (figure 2).

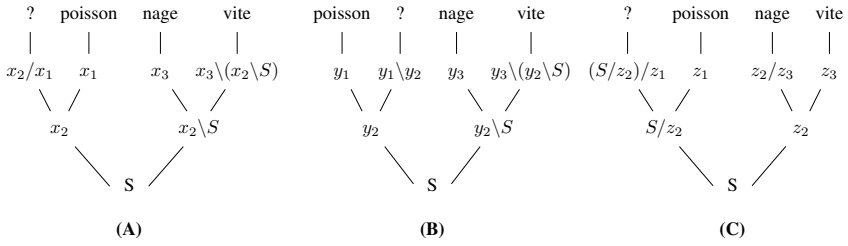

FIGURE 2 – Three possible structural trees for example 3

Words	Learned categories	A	B	C
poisson	n	x_1	y_1	z_1
nage	$np\backslash S$	x_3	y_3	z_2/z_3
vite	$(np\backslash S)\backslash(np\backslash S)$	$x_3\backslash(x_2\backslash S)$	$y_3\backslash(y_2\backslash S)$	z_3
?	–	x_2/x_1	$y_1\backslash y_2$	$(S/z_2)/z_1$

TABLE 2 – Unification of variable types and learned types

We can observe that structural tree (A) is unifiable and the solutions are as follows : $x_1 = n$, $x_2 = np$ and $x_3 = np\backslash S$; while the missed category can be gained by substitution of all occurrences of x_i in variable category x_2/x_1 by its relevant fixed category which yields np/n in our case. The suggested category with unification technique on this tree can only be licensed if it exists in our lexicon (or some valid external lexicon) which is the case. Notice that np/n corresponds to determiner category.

Structural tree (B) is unifiable as well, and the unification technique yields $n\backslash np$ as a potential solution for the missing category with the same analysis. Since this solution does not exist in our lexicon it can not be accepted. Structural tree (C) is not unifiable and it is rejected without further analysis. We can analyze example (4) with the same technique. The problem is large number of its possible combinations of its structural tree. We will see a treatment for this issue in the next section.

4 AB grammars, Unification and Dynamic Programming

As we saw in the previous section, for (unknown) n sequence of words the possible structural trees are ($= catalan(n-1) * 2^{n-1}$) and also the number of positions for a missing category is $n+1$. [6] The final number of different combinations of a n sequence of words with one missing category is

6. For n number of words with one missing category, we will have $n-1$ potential solutions between the words, one in the beginning and another one in the end.

$= catalan(n) * 2^n * (n+1)$ which grows faster than exponential rate (UTH, 1973). Actually, we do not need to explore all the possible structural trees since we know that some of our categories are fixed. In this section, we illustrate how the problem in example (4) can be tackled with dynamic bottom-up parsing.

The idea behind our solution is to try all the (n+1) positions for the missing category and avoiding searching all the possible structural trees by early pattern matching and unification technique of RG algorithm using a modified version of CYK dynamic programming technique [7]. With this strategy we can save each step results for making decision as we go further. As we will see, this leads us to an efficient computational complexity of $O(n^4)$ with n number of words. The table 3 is one of the possible cases (out of eight) for parsing example (4) assuming that x_1 (=missing category) is between 'vite' and 'la'. We start parsing step-by-step to illustrate how it works. [8]

1	np/n	n	$np\backslash S$	$(np\backslash S)\backslash(np\backslash S)$	x_1	np/n	n
	Le	**poisson**	**nage**	**vite**	**?**	**la**	**mer**
	1	**2**	**3**	**4**	**5**	**6**	**7**

TABLE 3 – Step (1) of parsing with unification for example (4)

The table 3 shows the initial phase of parsing, in which the fixed rigid learned categories are assigned plus x_1, a variable category, which is supposed to be specified. [9]

2	np	–	$np\backslash S$	? $x_2[x_1 := ((np\backslash S)\backslash(np\backslash S))\backslash x_2]$	? $x_3[x_1 := x_3/(np/n)]$	np	
1	np/n	n	$np\backslash S$	$(np\backslash S)\backslash(np\backslash S)$	x_1	np/n	n
	Le	**poisson**	**nage**	**vite**	**?**	**la**	**mer**
	1	**2**	**3**	**4**	**5**	**6**	**7**

TABLE 4 – Step (2) of parsing with unification for example (4)

The table 4 shows step 2, i.e. all possible derivations of two sequential categories. Since, we have the variable categories, we can have the flexibility of working with unification technique. For example, variables x_2 and x_3 would be a possible derivation if x_1 be substituted with $x_3/(np/n)$ and $((np\backslash S)\backslash(np\backslash S))\backslash x_2$ respectively.

Notice that we have included question mark(=?) in the cells of x_2 and x_3 to signify that we have not performed the pattern matching yet. Our algorithm benefits from early pattern matching of variable categories with the lexicon or any external reference that provides categories [10]. Note that all the possible substitutions for x_1, namely $((np\backslash S)\backslash(np\backslash S))\backslash x_2$ and $x_3/(np/n)$ fail in pattern matching phase ; since, there is no fixed type category (and even none subcategory) in our lexicon that can be matched with them. Early matching is an efficient technique since it prevents unnecessary computation. As depicted in step (3) in table 5 all possibilities for x_1 (in row 2) are ruled out.

7. See appendix for the pseudo-code of the algorithm

8. We have used bottom-up representation which is not very intuitive. This choice is made since it is in alignment with chart representation in dynamic programming.

9. Note that, in principle, we should assume all the possible positions of the unknown missing word which is 7 in this case. We have shown only one of the cases that leads to a solution. Although, our proposed algorithm would consider all the cases.

10. Until now, Moot's Grail (Moot, 2010) provides around 900 categories for French language. This can be used for pattern-matching in this phase as the external reference.

3	S	–	$?\, x_3[x_1 := (np\backslash S)/x_3]$	–	$x_2[x_1 := x_2/np]$		
2	np	–	$np\backslash S$	–	–	np	
1	np/n	n	$np\backslash S$	$(np\backslash S)\backslash(np\backslash S)$	x_1	np/n	n
	Le	**poisson**	**nage**	**vite**	**?**	**la**	**mer**
	1	**2**	**3**	**4**	**5**	**6**	**7**

TABLE 5 – Step (3) of parsing with unification for example (4)

Again, we will follow the same procedure that we practiced for step 3 in table 5. In this step, all the possible derivations of three possible sequential categories are shown in each cell. The variable category $(np\backslash S)/x_5$ does not match with any fixed category in our lexicon, so, it will be ruled out. Although, x_6/np matches with some categories in our lexicon so we will keep it as we go to the next step. We will continue the same procedure until we will reach to step 7 which is the last one. Table 6 shows the last step.

7	$x_5[x_3 := x_5/np]$ $x_4[x_2 := S\backslash x_4]$						
6	–	–					
5	$x_3[x_1 := S\backslash x_3]$	–	–				
4	S	–	–	–			
3	S	–	–	–	$x_2[x_1 := x_2/np]$		
2	np	–	$np\backslash S$	–	–	np	
1	np/n	n	$np\backslash S$	$(np\backslash S)\backslash(np\backslash S)$	x_1	np/n	n
	Le	**poisson**	**nage**	**vite**	**?**	**la**	**mer**
	1	**2**	**3**	**4**	**5**	**6**	**7**

TABLE 6 – Last step of parsing with unification for example (4)

In table 6, all the early pattern matching is finished, and we should perform the unification. We can find two following results :

— In row 7, what we expect to see is S, so, $x_5 = S$. By substitution we have $x_3 = S/np$, so, $x_1 = S\backslash(S/np)$, since, this category does not exist in our lexicon it is ruled out.
— The same as above, in row 7, what we expect to see is S. By substitution we have $x_4 = S\backslash S$, so, $x_1 = (S\backslash S)/np$, since, this category exist in our lexicon it accepted as a potential answer, and this is what we were initially looking for, a determiner.

In this section, we informally analyzed how our modification to CYK algorithm can be useful in finding a missing category using early pattern matching and unification in learning RG algorithm. The relevant algorithm and its computational complexity are discussed in the appendix.

5 Conclusion and Possible Extensions

We explained how we can find one missing category in an incomplete utterance. The method is basically designed for AREN project (ARgumentation Et Numérique) in which the average number of words in a sentence is twenty and the number of categorial types are around two hundred, so our proposal efficiently works on this specific domain. There are four possible extensions for our future

study and research.

The first extension is moving to non-rigid categories, namely fixed number of several categories assigned to each single word– instead of rigid ones. Since our approach benefits from chart parsing, this is, in principle, tractable. However, it will increase our computational complexity as it is expected. We can reduce our search space when we have lexical ambiguities (several categories for each word of the input sentence) by adopting super-tagging technique (Bangalore & Joshi, 2011). The one which is very suitable for our purpose is Moot's super-tagger (Moot, 2015) which is implemented in the wide-coverage version of the Grail parser.

The second extension is to focus on several numbers of missing categories, instead of just one. Of course, we need to limit the number of missing categories, since, comprehension of sentences with so many missing categories is so hard even for human users of language, nevertheless, it will be useful if we stay limited to fixed number of categories. One, primitive suggestion would be trying all the possible combinations of k missing words. This solution would have some drawbacks since we need to know the accurate number of missing categories, and more important, we should consider the increasing number of categories. It is difficult to rule out these categories by early pattern matching. One of the approaches would be using Jenson's approach (Jensen *et al.*, 1983) to build possible structural sub-trees of an incomplete utterance to find a limited number of suggestion for the position of missing categories. Further study is required for more clarification.

The third extension is to tackle other classes of incomplete utterances such as extra categories, misused categories, swap categories or any possible combinations. The main problem with this approach is the need of having a computational mechanism to know what is the exact problem of a sentence before solving it. For instance, if we know that in a given utterance, we have the fixed numbers of missing words, extra words, and swap categories, then we can tackle the problem more efficiently with a different module that we might build. The author has designed an algorithm that transforms a Context Free Grammar into Constraint Handling Rules (implemented in Prolog) for a fragment of natural language with relative clauses. Nevertheless, still, we need to have more research and study to analyses the efficiency and suitability of this kind of approaches comparing to the approaches that do not consider such distinctions.

The fourth extension is working on other families of categorial grammars such as Lambek's Calculus (Lambek, 1958) or some restricted version of Partial Proof Trees (Joshi & Kulick, 1997) for categorial grammar. The main problem –that the author is still dealing with– is that partial proof trees need to be restricted in order to work with Lambek's calculus. Moreover, designing a computationally decidable system with efficient complexity for incomplete utterances is still an on-going research. Generally speaking, this line of research can possibly be helpful in the domain of incoherence discourses. For instance, writing-assistant systems with questionnaires, for some specific targeted users such as patients suffering from specific mental disorders.

As discussed in the introduction, categorial grammar is chosen for its excellent correspondence with semantic. We drew the attention of readers to a special case, namely missing quantifier in the incomplete utterances. Once the missing quantifier is found out, we can make a complete sentence, and deduce different semantic readings of the utterance. Since each semantic proof corresponds to a cut-free proof net, we can make different complexity profiles having different proof nets gained from different readings. The low complexity profile will have the preferred reading among other readings(Morrill, 2000). This is the main role of categorial grammars in modeling preferences of semantic readings that naturally makes our syntax-semantic interface work.

6 Acknowledgements

I would like to thank Christian Retoré for inviting me to his lecture on learning algorithms for categorial grammars at Montpellier University and also for his advice on the possibility of using the learning algorithms for addressing the problem of sentence completion.

7 Appendix : Algorithm

The proposed technique for finding a missing category is not implemented yet. However, algorithm (1) written in high-level pseudo-code is provided. The underlying technique namely the unification is illustrated in details with running examples in the previous sections. One can observe that the complexity is basically similar to CYK standard algorithm which is $O(n^3)$ with the exception that it runs one extra loop with n steps, which makes the complexity of our algorithm $O(n^4)$ in the worst case scenario.

```
input  : An array 'words' and an array 'lexlist' of the categories
output : An array [(pos,cat)] of the pairs indicating position and categories of the missing words
result ← [ ];
n ← length(words);
for t ← 1 to n + 1 do
    k ← 1;
    ylist ← null;
    n_words = insert(words, x[k], t);
    // initializes the first row;
    for r ← 1 to n + 1 do
        p [1, r, 1] = lex(n_words[r], null, null)
    end
    for i ← 2 to n + 1 do
        for j ← 1 to n − i + 2 do
            for m ← 1 to i − 1 do
                // performs early pattern matching and unification;
                xlist = unify(p[m, j, 1], p[i − m, j + m, 1], lexlist, k) ;
                if xlist != fail then
                    ylist = ylist + xlist ;
                    assign(p[i, j], xlist) ;
                    else p[i,j,1]=null ;
                end
            end
        end
    end
    // substitutes the variables with its content;
    for v ← k to 1 step − 1 do
        t2 ← snd(ylist(v));
        t3 ← trd(ylist(v));
        if is_fixed(t3) then
            substitute_all(ylist, t2, t3)
        end
    end
    // selects all the appropriate candidates for missing categories;
    for w ← 1 to len(ylist) do
        t2 ← snd(ylist(v));
        t3 ← trd(ylist(v));
        if (is_fixed(t3) and t2=x[1] and is_in_lexicon(lexlist,t3)) then
            result = result + [(t, t3)]
        end
    end
    return (result)
end
```

Algorithm 1: An algorithm for the new proposal

Références

BANGALORE S. & JOSHI A. (2011). *Supertagging : Using Complex Lexical Descriptions in Natural Language Processing*. MIT Press.

BLACHE P. (2000). Constraints, linguistic theories, and natural language processing. In *International Conference on Natural Language Processing*, p. 221–232 : Springer.

BONATO R. (2000). *Uno studio sull'apprendibilità delle grammatiche di Lambek rigide—a study on learnability for rigid Lambek grammars*. PhD thesis, Tesi di Laurea & Mémoire de DEA, Università di Verona & Université Rennes 1.

BONATO R. & RETORÉ C. (2014). Learning lambek grammars from proof frames. In *Categories and Types in Logic, Language, and Physics*, p. 108–135. Springer.

BUSZKOWSKI W. (1987). Discovery procedures for categorial grammars. *Categories, Polymorphism and Unification*, p. 35–64.

BUSZKOWSKI W. & PENN G. (1990). Categorial grammars determined from linguistic data by unification. *Studia Logica*, **49**(4), 431–454.

DAHL V. & BLACHE P. (2004). Directly executable constraint based grammars. In *Proc. Journees Francophones de Programmation en Logique avec Contraintes, Angers, France*, p. 149–166.

DINI L. & MALNATI G. (1993). Weak constraints and preference rules. *Studies in Machine Translation and Natural Language Processing*, p. 75–90.

JENSEN K., HEIDORN G. E., MILLER L. A. & RAVIN Y. (1983). Parse fitting and prose fixing : getting a hold on ill-formedness. *Computational Linguistics*, **9**(3-4), 147–160.

JOSHI A. K. & KULICK S. (1997). Partial proof trees as building blocks for a categorial grammar. *Linguistics and Philosophy*, **20**(6), 637–667.

KAMP H. & REYLE U. (1993). From discourse to logic ; introduction to the modeltheoretic semantics of natural language.

KANAZAWA M. (1998). *Learnable Classes of Categorial Grammars.* ERIC.

LAMBEK J. (1958). The mathematics of sentence structure. *The American Mathematical Monthly*, **65**(3), 154–170.

LEACOCK C., CHODOROW M., GAMON M. & TETREAULT J. (2010). Automated grammatical error detection for language learners. *Synthesis lectures on human language technologies*, **3**(1), 1–134.

MELLISH C. S. (1989). Some chart-based techniques for parsing ill-formed input. In *Proceedings of the 27th annual meeting on Association for Computational Linguistics*, p. 102–109 : Association for Computational Linguistics.

MOOT R. (2010). Wide-coverage French syntax and semantics using Grail. In *Proceedings of Traitement Automatique des Langues Naturelles (TALN)*, Montreal. System Demo.

MOOT R. (2015). A type-logical treebank for french. *Journal of Language Modelling*, **3**(1), 229–264.

MOOT R. & RETORÉ C. (2012). *The logic of categorial grammars : a deductive account of natural language syntax and semantics*, volume 6850. Springer.

MORRILL G. (2000). Incremental processing and acceptability. *Computational linguistics*, **26**(3), 319–338.

PROST J.-P. (2009). *Modelling Syntactic Gradience with Loose Constraint-based Parsing 8 Modélisation de la gradience syntaxique par analyse relâchéea base de contraintes*. PhD thesis, Citeseer.

PULLUM G. K. & SCHOLZ B. C. (2001). On the distinction between model-theoretic and generative-enumerative syntactic frameworks. In *International Conference on Logical Aspects of Computational Linguistics*, p. 17–43 : Springer.

SCHNEIDER D. & MCCOY K. F. (1998). Recognizing syntactic errors in the writing of second language learners. In *Proceedings of the 17th international conference on Computational linguistics-Volume 2*, p. 1198–1204 : Association for Computational Linguistics.

UTH D. (1973). The art of computer programming, vol 1 : Fundamental algorithms.

VOGEL C. & COOPER R. (1995). Robust chart parsing with mildly inconsistent feature structures. *Nonclassical feature systems*, **10**, 197–216.