The Utility of Combinatory Categorial Grammar in Designing a Pedagogical Tool for Teaching Languages

Simon Delamarre
TELECOM Bretagne
CS 83818 F29238 Brest cedex
simon.delamarre@telecom-bretagne.eu

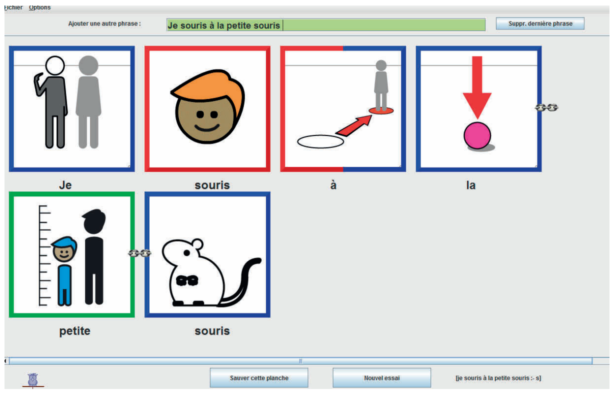
Abstract

This paper intends to demonstrate how Applicative and Combinatory Categorial Grammar (ACCG) can be drawn on to design powerful software applications for the teaching of languages. To this end, we present some modules from our "pictographic translator", a software that performs syntactical analysis of sentences in natural language directly written by the user, and then dynamically displays series of pictograms that illustrate the words and structure of the user's sentences. After a short presentation of our application and an introduction to ACCG, we will examine how this formalism enables the building of several high-level functions in our system, such as disambiguation, structure exhibition and grammatical correction/validation. We finally open a short discussion concerning the potential (and limits) of this architecture with regards to multilingualism.

Introduction

A pictographic translator

According to certain studies concerning learning and retention, we usually memorize about 90% of what we do, versus only 10% of what we read. And indeed, however forced this explicit quantification may seem, it is a fact that any learning activity, to be efficient, has to be active. It is in this spirit that, while the huge majority of current pedagogical software programs mainly rely on cross-the-correct-answer/fill-the-gap exercises, we conceived this interactive pictographic translator. It consists of a graphic interface, in which the user is invited to build a sentence, by typing his own words. When the programs detects that a new word has been entered (using a tokenizer), it computes a syntactical analysis of the currently built sentence, and then, retrieves (by performing a lemmatization) and displays a pictogram that represents this word (see Figure 1). As often as it remains relevant, we use the rule of correspondence: one word, one pictogram. Situations in which an only pictogram for several words would be preferable (e.g. *washing machine*) are treated in a post-processing of the tokenisation step, by merging tokens that form a wider component present in the lexicon (*[washing,machine]* $\rightarrow$ *[washing machine]*).



Je souris à la petite souris

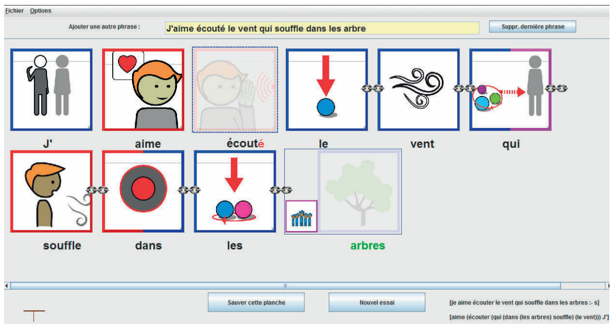
[je souris à la petite souris :- s]

Figure 1: The pictographic representation for "Je souris à la petite souris" (I'm smiling at the little mouse). The pictograms appear dynamically as the words are entered.

At the same time, by using several graphical effects, such as colored borders, subtitles, transparencies, small pictograms to indicate functions (plural, tenses...), the program gives to the user direct visual control of the sentence he is writing, enabling him to detect problems if there are any (see figure 2). If however the user is not able to correct one or several mistakes by himself, he may ask the program to indicate these for him.

Thus, little by little, through this process of writing/visual control/correction, the learner is led to build by himself correct sentences, without needing any other exterior help, which will bring him, beside the satisfaction of autonomy, a better understanding of the structures and mechanisms of language. Furthermore, in the case of pupils learning reading, this ludic activity of "making pictures to appear" may conduce them to grasp the expressive power of words.

The software program has been initially developed for the French language, and already supplies operational linguistic coverage. Our latest version also implements experimental extensions to English and Spanish, with reduced vocabulary and grammar. It should be noted that our implementation uses some elements of the excellent Michael White's Java API for categorial grammars, **OPEN CCG**, and also Jason Baldridge's associated format **DotCCG** for grammar definition (see `http://openccg.sourceforge.net/`).



J'aime écouté le vent qui souffle dans les arbre

[je aime écouter le vent qui souffle dans les arbres :- s]

[aime (écouter (qui (dans (les arbres) souffle) (le vent))) J']

Figure 2: Another example, "J'aime écouter le vent souffler dans les arbres" (I like listening the wind blowing in the trees). One can see here the errors highlighted and the correction function. Here the user fails in writing the infinitive verb "écouter" (to listen) -confusion with past participle "écouté"- and forgot the final s in the plural word "arbres" (trees)

The ACCG formalism

The grammatical core of our software relies on the Applicative and Combinatory Categorial Grammar formalism (Biskri and Descles 1997). We only aim here to informally present some elements to make the understanding of the following sections easier, for a complete introduction to categorial grammars, the reader should refer, for example, to (Biskri 1995) or (Baldridge 2002) (for CCG).

We follow the levelled architecture for natural language analysis proposed in (Desclés 1990), in this paper we will use the first two levels: the **morphosyntactic level** and the **predicative level**. At the first level, we observe the linguistic units concatenated in the order imposed by the syntax of the natural language considered, whereas the predicative level handles the logical structure hidden behind the linear order of the former level.

Now we present how the connection between the two first levels can be performed. As a Categorial Grammar Model, ACCG relies on the assignation of **orientated types** to linguistic units. The oriented types are defined by the data of:

- A set T_0 of atomic types. In this paper, we use $T_0 = \{S, N, N^\star\}$ (S : sentence, N : noun, $N^\star$: noun phrase)
- Two laws of composition / and \between types. Thus, if X and Y are oriented type, X/Y and X\Y are also (functional) orientated types. X/Y (resp. X\Y) represents an operator that takes an argument of type Y on its right (resp. left) side, to form an expression of type X.

These types permit the concatenated linguistic units to combine, according to rules that are presented below, and to form logical dependencies structure (according the operator-operand scheme), that is to say, the predicative structure of the sentence ; provided we dispose of a language manipulate these structures and introduce them in the rules. This work could be done using lambda-expressions, nevertheless we chose (in this paper as well as in our software project) to use here **Combinatory Logic** instead (as advocates ACCG, as a matter of fact). Indeed, as they allow us to avoid having to handle linked variables, the combinators of Combinatory Logic constitute entities far easier to manipulate in the context of a computational implementation.

We can thus formulate some of the ACCG rules (we only present four of them here, to see the complete rule system the reader should refer to (Biskri and Descles 1997)). *Notation : A linguistic unit with type X is designed by [X : u]*

- ⋆ Applications rules
  $>$: [X/Y : u_1] - [Y : u_2] $\rightarrow$ [X : $(u_1 u_2)$]
  $<$: [Y : u_1] - [X\Y : u_2] $\rightarrow$ [X : $(u_2 u_1)$]
- ⋆ Composition rules
  B$>$: [X/Y : u_1] - [Y/Z : u_2] $\rightarrow$ X/Z : (**B** u_1 u_2)]
- ⋆ Type-raising rules
  T$>$: [X : u] $\rightarrow$ [Y/(Y\X): ($\mathbf{C}_\star$ u)]

These rules are completed by a system of **metarules** (Desclés and Biskri 1995), whose role is to control the triggering of type-raising rules (and also rules implicated in distributive coordination processing - not handled by our current version). We also use the concept of **decomposition/structural reorganisation** (again, see (Desclés and Biskri 1995)), which permits to decompose a combinatorial structure within the quasi-incremental strategy in order to deal with backward modifiers.

Finally, in order to permit a direct processing of some grammar errors like subject-verbs agreement, determiner-noun agreement(etc.), we use lexical features associated to the atomic types, such as case, person, number markers (for instance). For example, in English, we could give to the verbal form "has" the category :

$$S\backslash N^\star_{pers=3rd}/N^\star$$

The $N^\star_{pers=3rd}$ permits to specialize the $N^\star$ operand waited by the verb "has" on its right side, restricting it to be in the *3rd person*. That way, we will be able to detect that, for example, "*You has" is an incorrect form, and also to retrieve the correct flexion.

ACCG and incremental strategy analysis

Since the goal of such a system is to back the dynamical process of building sentences, the functions of disambiguation, grammatical correction must be performed as soon as possible, we cannot wait for the sentence to be entirely complete to trigger them (e.g. we need to have a way to combine a subject with a transitive verb before the later has consumed its object, in order to be able for example to check subject-verb agreement). To analyse partial sentences, we must then fully exploit CCG's flexibility, in particular the associativity given by the conjugated action of type-raising rules and composition ones (see for example (Baldridge 2002) to go further on this).

This flexibility, on one hand, is known to introduce indeterminism, through the phenomenon of *spurious ambiguity* (generating several analysis that lead to a same predicative structure). On the other hand, we can take advantage of it to enable an **incremental** parsing strategy (that is to say, which systematically applies combinatory rules as soon as they are available, from left to right, c.f. (McConville) and Steedman's psychologically motivated parser in (Steedman 2000)), which precisely solves the problem of spurious ambiguity, with the trade, as we have said, of introducing additional control mechanisms, here the ACCG metarules and decomposition rule (Biskri 1995), (Desclés and Biskri 1995). We chose indeed to retain this later alternative, because, in addition to the arguments listed above, the incremental parsing strategy fits naturally the interactive nature of our program, the user's typing of words and the dynamic pictogram displaying being themselves incremental. [1]

The (simplified) mechanism of analysis we use is the following (we note i the current step, by what we mean that we are waiting for the i-th word to be entered) :

1. We have a list that contains the possible analysis for the part of the sentence previously analysed (we note it L_{i-1}, initialized with $L_0 = [\ ]$) L_{i-1} is a list of typed applicative expressions : $L_{i-1} = [[t_{i,1} : u_{i,1}], ..., [t_{i,N_i} : u_{i,N_i}]]$ (where N_i is the number of analysis the list contains).
2. When a new string s_i identified by the tokenizer as representing a word is entered, list all the possible "words" (that is to say, lexicon entries, including informations such as stem, features etc.) with their possible categorial types. Doing so, we build a list $W_{s_i} = [[ti : w_i]_{i=1...M_i}]$ (where M_i is the number of typed words possible for the string s_i).
3. Then, for each couple (typed analysis,typed word) (a, z) in $L_{i-1} \times W_{s_i}$, do :

```
{
  ⋆ If a metarule can be applied to $(a, z)$ :
    Apply the metarule. Obtain : $(a', z')$
    $(a, z) \leftarrow (a', z')$
  ⋆ If $a$ rule in the rule system can be applied to $(a, z)$ (with regard to the type of $a$ and $z$):
    Combine $z$ with $a$ according to this rule. This forms a new typed expression $(t_{(a,z)} : u_{(a,z)})$
    Store $(t_{(a,z)} : u_{(a,z)})$ in $L_i$
}
▷ As the end of the step, only $(a, z)$ that have been able to combine have been stored into $L_i$, the others have been discarded.
```

[1]Note that non-incremental algorithms would nevertheless be applicable as well, by triggering them after each word entered, with some adaptations to avoid redundant computations between steps (in a CYK chart, for example, we should conserve the previously calculated cells). However, we conjecture that incremental strategies can perform a better share of the algorithmic load between the different words entered than non-incremental one. Indeed, again with a CYK, entering the n^{th} word will always add in the chart a diagonal of n new cells, inducing $\frac{n(n-1)}{2} = O(n^2)$ new span partitions to test, whereas there is no such a systematic dependency between the step and the number of computations with incremental parsing (and no spurious ambiguity). It would be, however, interesting to have these two type of alternative empirically compared in term of mean complexity.

In the worst case (if all possible analysis in L_{i-1} combined with all possible type word W_{s_i} for the current step), the size of L_i could be $M_i N_i$, which means that this algorithm has an exponential worst-case complexity. Fortunately, as a direct consequence of the incremental analysis strategy, we can decide an immediate discarding of analysis which could not combine with any possible typed word (as presented in the algorithm above), which drastically reduces the number of analysis still in the race. As a consequence, the size of L_i remains under control, and so the number of computations needed to perform the analysis does not overflow

One should note, however, that to be viable, this strategy need a efficient metarule system, able to ensure entirely the incremental analysis. If it's not the case, one may have to permit to not-totally-combined analysis to survive for a while, for example by allowing a maximum number of not combined elements greater than 1, of by saving the analysis with a minimum number of not combined elements.

Let us see two steps of the algorithm for the sentence that appears in the Figure 1.

Analysis of the sentence *Je souris à la petite souris* (-I'm smiling at the little mouse)

- **Step 1**
  $W_{Je} = [N^{\star}_{pers=1st} : je]$ $L_1 = [[N^{\star}_{pers=1st} : je]]$
- **Step 2**
  $W_{souris} = [[N^{\star} : souris], [S\backslash N^{\star}_{pers=1st} : souris], [S\backslash N^{\star}_{pers=2nd} : souris]]$
  (the character string *souris* is ambiguous in French, it may refer to the substantive "mouse", or to the verb "to smile" (present, singular, 1st or 2nd person))
  - $([N^{\star}_{pers=1st} : je], [N^{\star} : souris])$ No metarule or rule can be activated : the combination is discarded.
  - $([N^{\star}_{pers=1st} : je], [S\backslash N^{\star}_{pers=1ere} : souris])$
    ⋆ The backward application rule (<) can be applied, we obtain the predicative structure: $([S : souris\ \ Je])$
  - $([N^{\star}_{pers=1st} : je], [N^{\star} : souris])$ No metarule or rule can be activated (due to the bad feature agreement pers=1st/pers=2nd) : the combination is discarded.

  $L_2 = [[S : souris\ \ Je]]$
- ○ **Step 3-5** (...)

- **Step 6** (the word entered is the second "souris")
  $L_5 = [[S/N^{\star} : \mathbf{B}(\mathbf{B}(\mathbf{B}(\mathbf{C}_{\star}Je)(\mathbf{B}(\mathbf{C}_{\star}souris)\grave{a}))la)petite]]$
  $W_{souris} = [[N^{\star} : souris], [S\backslash N^{\star}_{pers=1ere} : souris], [S\backslash N^{\star}_{pers=2nde} : souris]]$
  - $([S/N^{\star} : \mathbf{B}(\mathbf{B}(\mathbf{B}(\mathbf{C}_{\star}Je)(\mathbf{B}(\mathbf{C}_{\star}souris)...], [N^{\star} : souris])$
    ⋆ The forward application rule (>) can be applied, we obtain (after combinator reducing) the applicative expression: $([S : (\grave{a}(la(petite\ \ souris)))souris\ \ Je])$
  - $([S/N^{\star} : \mathbf{B}(\mathbf{B}(\mathbf{B}(\mathbf{C}_{\star}Je)(\mathbf{B}(\mathbf{C}_{\star}souris)...], [S\backslash N^{\star}_{pers=1ere} : souris])$ No metarule or rule can be activated : the combination is discarded.

– $([S/N^\star : \mathbf{B}(\mathbf{B}(\mathbf{B}(\mathbf{C}_\star Je)(\mathbf{B}(\mathbf{C}_\star souris)...], [N^\star : souris])$ No metarule or rule can be activated : the combination is discarded.

$L_6 = [[S : (\grave{a}(la(petite\ \ souris)))souris\ \ Je]]$

Wee see that these steps have permitted to discard all the hypothesis of analysis which were not syntactically coherent, that is to say, to perform a disambiguation over the character string entry (in our example, the first character string has been interpreted rightly as the verb for "to smile", and the second as the substantive for "mouse").

Exploiting the extracted combinatorial structure

Now that the combinatorial structure underlying the sentence has been retrieved, we are able, on that basis, to perform advanced operations and inferences. We can list at least five advanced functions it is able to offer, and which would not be reachable (or with serious difficulties) with a shallower analysis strategy :

- Efficient disambiguation mechanism.
- Accurate and powerful grammatical processing
- Validation of the syntactical structure of the entered sentences
- Computation on the fly, as the words are entered
- Multilingual flexibility (see the Discussion at the end of this paper)

We have already seen in the former paragraph how the incremental analysis permits to perform disambiguation with choosing the sequences of types that can combine, and to discard definitively others, permitting us to reach a very reasonable computing charge.

Concerning the grammatical processing, it is possible, for simple problems such as subject-verbs agreement, to take advantage of the feature mechanism presented in the introduction of this paper. Otherwise, subtler analysis can be realised on the basis of the combinatorial structure. As an illustration we give a method to deal with the French discontinuous negative operator *ne...pas*. The French negation is two-parted : firstly the verb to be negate must (except in colloquial style) be preceded by the word *ne*, secondly it must combine with a *negative block*. The difficulty came from the fact these "negative blocks" are multi-faced. It is generally the word "pas", as adverb, but depending on semantic nuances, other adverbs are also possible *ne...jamais*, *ne...point*, or even other negative elements that are not adverbs : *rien* (nothing), *aucun+N* (no +N)... that combine with the verb in a position of object or subject.

As the analysis step with combinatory logic precisely permits us to deal with the issue to know which word *combine* with which other, all these rules concerning French negation can be capture efficiently on the basis of the applicative expression built.

In the following lines, we only consider expressions in normal form, without combinators. We call *verbal applicative expression* an expression with the form : $W = (U_1(\ldots(U_N \quad V)\ldots))U_{N+1}\ldots U_{N+M}$, where V is an atomic expression of type verb, N the number of modifiers applied to the verb, M the number of its arguments (M=1 : intransitive verb, M=2 : transitive verb...). More rigorously, we can define the type verbal_expression as follows :

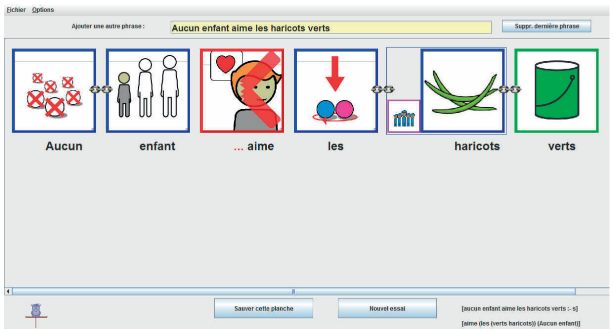
```
  type verbal_expression0
| atomic_verb;
| (expression verbal_expression0)
  type verbal_expression
| verbal_expression0;
| (verbal_expression expression)
```

We suppose also that the negative words has defined in the lexicon as having a "negative marker". Then, we can define the following functions :

```
is_negative(U)
| U atomic $\rightarrow$ has_negative_marker(U)
| U verbal expression $\rightarrow$ false
| U : U1 U2 $\rightarrow$ is_negative(U1)$\vee$ is_negative(U2)
is_correctly_negated($W$)
| W verbal expression : $\rightarrow$
  $\neg(\exists i \in [1,N]; U_i = \mathbf{ne} \oplus \exists i \in [1,M]; is\_negative(U_i)))$
| W $\rightarrow$ true
```

(using the above-mentioned notations)

Then, with these two simple functions we are able to handle with accuracy the subtleties of the French negation we described. For example, the sentence **Aucun enfant aime les haricots verts* (No child likes French beans), whose associated applicative expression is : aime (les (verts haricots)) (aucun enfant), will be detected as incorrect (see Figure 3). Indeed, has_negative_marker(aucun)=true, so is_negative(aucun enfant)=true, so *aime* has a negative element in its arguments whereas it is not combined with the adverb *ne*.



[aucun enfant aime les haricots verts :- s]
[aime (les (verts haricots)) (Aucun enfant)]

Figure 3: *Example of advanced grammatical processing: the French negation.* The sentence (*Aucun enfant aime les haricots verts - No child likes French beans) is incorrect, one would have to add the word "ne" before the verb to make the negation complete (which is graphically represented by an incomplete negation cross on the pictogramm for *aimer*).

Through this example, wee see that applicative expression can be a powerful way to express elegantly many grammar rules, which would be nearly unreachable with a simple analysis at the string level. In this view, it could be useful to develop a practical human-friendly syntax (in the spirit of **DotCCG**) to build external files defining such rules with the language of predicative expressions, in order to exploit them in natural language processing programs.

Another very useful consequence of the syntactical analysis is that we are able to determine if a sentence is syntactically valid (only by checking the type of the whole sentence is $-S$), and then to "congratulate" the user when he succeeded to building a syntactically correct sentence, and inform him when he has succeeded. Using the same principle, we are also able to highlight some elements of the phrase structure, such as the noun phrase (see 1, 2, 3, where the noun phrase "blocks" are linked by little chains, permitting to identify them at the first view).

Results

The implementation we made of this architecture has demonstrated its algorithmic viability, and then, reaches the final functionalities we had targeted: right pictograms appear nearly instantaneously as the sentences are typed, the disambiguation mechanism works well, as well as functions of structure representation, error highlighting and correction. This permit to display the right pictograms corresponding to the user's idea, provided they exist in the image bank (otherwise our program simply substitute the character string to the missing pictogram, still with the category color if the word is in the lexicon). An interesting possible future work would be to use -carefully- synonymic relations to extend the coverage of word-pictogram association. The French version (that counts approximately 2000 word stems) is ready to be tested in schools. We are currently in a pre-test phase in collaboration with primary school teachers, in order to determine what correction and new functionalities should be added to make easier and more fruitful the first tests in real conditions with pupils. Note also that this work was integrated in the PALLICOM project (`http://recherche.telecom-bretagne.eu/palliacom/`), which aims to support disabled persons without speak by building a pictographic communicator (Abraham 2005), and deals with the reverse operation, i.e. building text from pictograms. Future works may consist in combining, on the ACCG basis, the two directions (text to/from pictograms) in a same architecture, which could be very fruitful for both applications (palliative communication and language teaching).

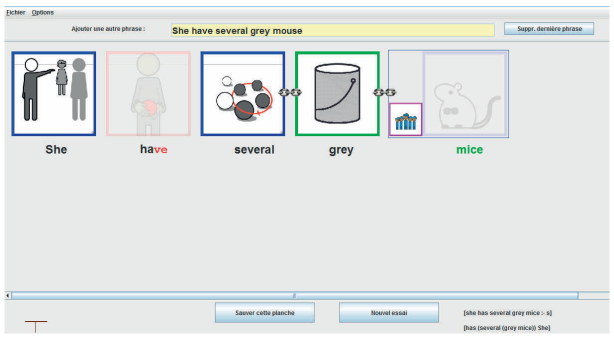
Discussion: how to deal with multilingualism?

At this point raises a very interesting question: if at first place this program has been designed to be a tool for the teaching of *French*, to what extent the principles it relies on could be extended to other languages? Which modules of the architecture can be reused without any adaptation in all (or at least in many) languages, and which ones should induce a specific processing from one to another?

In this view, one of the most challenging issues may seem to be the adaptation of the syntactic processing, in particular the disambiguation mechanism and the building of underlying applicative expressions... Fortunately, here can interfere another asset of Categorial Grammars : the suppleness induced by the fact they rely on only a few very general concepts (a small set of atomic type, and the two laws of composition) make them able to pretend to a certain universality.

Let us take an example with adjectives. The semantic concept of adjective is quite universal, however many differences raise between languages concerning the associated syntactic process: if in English the adjective always precede the noun it qualifies (fact expressed in Categorial Grammars by giving to the adjective the type N/N), in other languages, such as many Romance languages, it can also follow the noun (adding the possible category $N\backslash N$), whereas in Chinese adjectives can have a certain verbal value, as they can combine with a noun to form a correct sentence (which is expressed by adding $S\backslash N^{\star}$ to its list of categorial types).

Thus, we see that categorical types, unlike classical categories, offer a general language that permits to define with accuracy (and conciseness) the syntactic structure of any language, on the basis of the operator-operand scheme and some elemental types (noun-N, sentence-S), that are assumed to be universal. In the case of our program, this means that *one would only need to supply a lexicon with associated categorial categories* to be able to reuse all the syntactical and combinatorial processing (including the disambiguation mechanism) from one language to another. See the examples with English and Spanish in figures 4, 5.



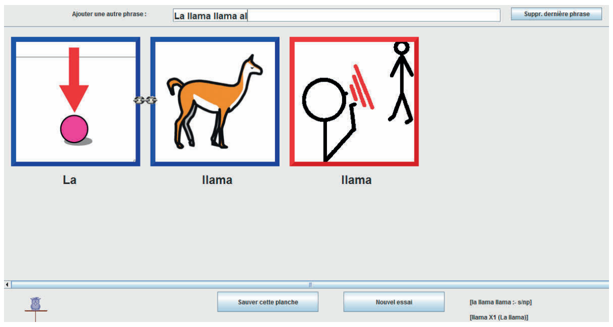
She have several grey mouse

[she has several grey mice :- s]

[has (several (grey mice)) She]

Figure 4: Reusing of the grammatical correction mechanism (by only supplying a different lexicon to the program, defined in a **DotCGG** file), with English. The user typed the sentence **She have several grey mouse* (bad subject-verb agreement, and plural forgotten for mouse).

This, however, only concerns the fundamental concepts the syntactical analysis is based on. But the architecture we propose for our pictographic translator introduce several other steps, which may need to introduce language-dependant rules, or even algorithms. In particular, the system of **metarules**, essential to make possible the incremental analysis while conserving a reasonable computational charge, is not a language invariant, which means that one would need for each language to define which metarules are needed (and which are not), and to check the consistency of the set chosen (which may be complex). Nevertheless, one should keep in mind that the number metarules is typically quite reduced ((Desclés and Biskri 1995) pro-



```
La llama llama al
```

```
[la llama llama :- s/np]
[llama X1 (La llama)]
```

Figure 5: Disambiguation at works, with Spanish. The user began to enter the sentence *La llama llama...* (the llama calls...), and we see that the program has already performed the desambiguation between *llama* (llama), and *llama* the third-person form of the verb *llamar*, to call

pose ten metarules for French), and that some metarules are commons to many languages (for example, the fact to type-raise a nominal phrase that precedes a transitive verb, in languages in which these terms are defined). More critical may be the step of the tokenizer, for which there is, at first glance, no other way than to have an specific algorithm for each language, and separating a string entry into pertinent morphemes in view to represent them with pictograms can indeed be a quite tricky issue (for instance in languages like German).

Another decisive question is the principle of the association word-pictogram itself. Of course, to begin, one has to define a mapping word→pictogram for each language. But the real difficulty lies above all in the possibility to reuse (or not) some sets of pictograms from one language to another.

It seems rather easy for certain class of words, such as nouns (despite possible cultural point of view towards aestheticism, we can assume that a nice pictogram of a lion will be as acceptable as a lion in any linguistic context), besides nothing prevents the user from adding its own nicer pictograms to the base. But as any language does perform a particular partition of the reality and have its own grammatical system, for more conceptual and grammatical pictograms (such as prepositions, articles, tenses...) reuse might be somewhat more delicate, and the user may have no idea (nor any will) for making or finding more adapted ones by himself. So, we should supply these grammatical pictograms to him, which implies, for each language, a reflection and a work with graphic designers and linguists, to design pertinent pictograms. Nevertheless, one should note that pictogram should not be considered as an “interlingua”, but only as supports, clues that offer to the user a direct visual control og the syntactic and semantic validity of its sentence, and also give him mnemonic keys to memorize new words. Thus, when it appears impossible to give through a pictogram a clear and intuitive understanding of a too abstract concept or grammatical word, it does not matter so much, firstly because the usefulness of the tool would not be undermined by a word without pictogram in a given sentence, secondly because it always remains possible to give a more indirect or symbolic graphical representation of difficult words, which would still assume a mere (but undeniably very useful anyway) mnemonic role.

Conclusion

The context of language teaching is an interesting field of application for Combinatory Categorial Grammar formalism for several reasons. On the one hand, we need not cover the whole language, but only the part one want to teach through the program. We can thereby assume we are in a specific context (by discarding in our lexicon the words we judge irrelevant to an introduction to one particular language), making the use of statistical methods unnecessary, at least at the first levels. To the other hand, for obvious pedagogical reasons, this part of the language chosen must be handled with linguistic exactness. This fully justifies the usefulness of Categorial Grammar formalisms, and in particular ACCG, for concrete application in the field of language teaching: their sharpness permits to develop several advanced linguistic functions and, in our case, to reach an innovative interactivity with the user, while the particular linguistic area that we are limited to by the context of teaching language learners ensures its implementation will remain computationally viable.

Acknowledgement: this work has been supported by French National Research Agency (ANR) through TecSan program (project/ /PALLIACOM No. ANR-08-TECS-014).

References

Abraham, M. 2005. *Dans le contexte des technologies de l'information et de la communication : une écriture pictographique fondée sur des principes sémantiques*. Ph.D. Dissertation, Institut Télécom-Télécom Bretagne, Université Paris Sorbonne Paris IV.

Baldridge, J. 2002. Lexically specified derivational control in Combinatory Categorial Grammar.

Biskri, I., and Descles, J. 1997. Applicative and Combinatory Categorial Grammar (from syntax to functional semantics). *Recent Advances in Natural Language Processing (selected Papers of RANLP 95)* 71–84.

Biskri, I. 1995. *La Grammaire categorielle combinatoire applicative dans le cadre de la grammaire applicative et cognitive*. Ph.D. Dissertation, Thèse de Doctorat, EHESS, Paris.

Desclés, J., and Biskri, I. 1995. Logique combinatoire et linguistique: grammaire categorielle combinatoire applicative. *Mathematiques Informatique et Sciences Humaines* 39–68.

Desclés, J. 1990. Langages applicatifs, langues naturelles et cognition.

McConville, M. Incremental natural language understanding with combinatory categorial grammar.

Steedman, M. 2000. *The syntactic process*, volume 131. MIT Press.