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] → [washing machine]).

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/).

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.
Combinatory Logic

we chose (in this paper as well as in our software project) to work could be done using lambda-expressions, nevertheless these structures and introduce them in the rules. This provided we dispose of a language manipulative and Combinatory Categorial Grammar formalism (generating several analysis that lead to a same predictivity given by the conjugated action of type-raising rules and distribution of atomic types. In this paper, 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: 

\[ S \rightarrow N^{*}_{\text{pers}=3rd}/N^{*} \]

The \( N^{*}_{\text{pers}=3rd} \) permits to specialize the \( N^{*} \) 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 indetermination, 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 to handle linked variables, the combiners 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)).

\[ \star \text{Applications rules} \]

\[ \begin{align*}
\rightarrow & : [X/Y : u_1] \cdot [Y : u_2] \rightarrow [X : (u_1 u_2)] \\
\leftarrow & : [Y : u_1] \cdot [X \backslash Y : u_2] \rightarrow [X : (u_2 u_1)]
\end{align*} \]

\[ \star \text{Composition rules} \]

\[ B \rightarrow : [X/Y : u_1] \cdot [Y/Z : u_2] \rightarrow X/Z : (B \ u_1 \ u_2) \]

\[ \star \text{Type-raising rules} \]

\[ T \rightarrow : [X : u] \rightarrow [Y/(Y/X) : (C, u)] \]

These rules are completed by a system of **metarules** (Delcles 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 (Descles 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, “*You has*” is an incorrect form, and also to retrieve the correct flexion.

**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 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 **oriented 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 \( T_0 = \{ S, N, N^* \} \) (\( S \) : sentence, \( N \) : noun, \( N^* \) : noun phrase)

- Two laws of composition / and \( \backslash \) between types. Thus, if \( X \) and \( Y \) are oriented type, \( X/Y \) and \( X \backslash Y \) are also (functional) oriented types. \( X/Y \) (resp. \( X \backslash Y \)) 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
as they are available, from left to right, c.f. (McConville) and Steedman’s psychologically motivated parser (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), (Descle`es 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.\footnote{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.}

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 : $L_{i-1} = [[t_1 : u_1], \ldots, [t_i : u_i, N_i]]$ (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 $W_{s_i} = [[t : u_i]_{i=1\ldots M_i}]$ (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 :
   \[
   \begin{align*}
   &\text{If a metarule can be applied to } (a, z) : \\
   &\quad \text{Apply the metarule. Obtain : } (a', z') \\
   &\text{if a rule in the rule system can be applied to } (a, z) \text{ (with regard to the type of } a \text{ and } z) : \\
   &\quad \text{Combine } a \text{ with } a' \text{ according to this rule. This forms a new typed expression } (t_{a(z)} : u_{a(z)}) \\
   &\quad \text{Store } (t_{a(z)} : u_{a(z)}) \text{ in } L_i
   \end{align*}
   \]
\[\triangleright\] 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.

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 $M_iN_i$, 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**
  \[W_{Je} = [N^*_{pers=1st} : je] \]
  \[L_1 = [[N^*_{pers=1st} : je]]\]

- **Step 2**
  \[W_{souris} = [N^* : souris], [S\setminus N^*_{pers=1st} : souris], [S\setminus N^*_{pers=2nd} : souris]\]
  \[(\text{the character string } \text{souris} \text{ is ambiguous in French, it may refer to the substantive “mouse”, or to the verb “to smile” (present, singular, 1st or 2nd person)})\]
  \[- ((N^*_{pers=1st} : je), [N^* : souris]) \text{ No metarule or rule can be activated : the combination is discarded.}\]
  \[- ((N^*_{pers=1st} : je), [S\setminus N^*_{pers=1ere : souris})\]
  * The backward application rule (<) can be applied, we obtain the predicative structure: ([S : souris Je])
  \[- ((N^*_{pers=1ere : je}), [N^* : souris]) \text{ No metarule or rule can be activated (due to the bad feature agreement pers=1st/pers=2nd) : the combination is discarded.}\]
  \[L_2 = [[S : souris Je]]\]

- **Step 3-5**

- **Step 6** (the word entered is the second “souris”)
  \[L_3 = \]
  \[([[S/N^* : B(B(B(C_souris)Je)(B(C_souris)à)la)petite]\]
  \[W_{souris} = [N^* : souris], [S\setminus N^*_{pers=1ere : souris}, [S\setminus N^*_{pers=2nd} : souris] ]\]
  \[- ((S/N^* : B(B(B(C_souris)Je)(B(C_souris)…), [N^* : souris])\]
  * The forward application rule (>\text{)} can be applied, we obtain (after combinator reducing) the applicative expression: ([S : ile(souris souris)Je])
  \[- ((S/N^* : B(B(B(C_souris)…), [S\setminus N^*_{pers=1ere : souris]}\]
  No metarule or rule can be activated : the combination is discarded.

152
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 realized 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: \( W = (U_1(\ldots(U_N \ V)\ldots))U_{N+1}\ldots U_{N+M} \), 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:

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

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").

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*).
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 underly-

![Figure 4: Reusing of the grammatical correction mechanism](image)

Figure 4: Reusing of the grammatical correction mechanism (by only supplying a different lexicon to the program, defined in a DotCCG 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 ((Descles and Biskri 1995) pro-
pose ten metarules for French), and that some metarules are common 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 a 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 of 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


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