A Toolchain for Grammarians

Bruno Guillaume, Joseph Le Roux, Jonathan Marchand, Guy Perrier, Karën Fort, Jennifer Planul

A Toolchain for Grammarians

Bruno Guillaume
LORIA
INRIA Nancy Grand-Est
Bruno.Guillaume@loria.fr

Joseph Le Roux
LORIA
Nancy Université
Joseph.Leroux@loria.fr

Jonathan Marchand
LORIA
Nancy Université
Jonathan.Marchand@loria.fr

Guy Perrier
LORIA
Nancy Université
Guy.Perrier@loria.fr

Karën Fort
LORIA
INRIA Nancy Grand-Est
Karen.Fort@loria.fr

Jennifer Planul
LORIA
Nancy Université
Jennifer.Planul@loria.fr

Abstract

We present a chain of tools used by grammarians and computer scientists to develop grammatical and lexical resources from linguistic knowledge, for various natural languages. The developed resources are intended to be used in Natural Language Processing (NLP) systems.

1 Introduction

We put ourselves from the point of view of researchers who aim at developing formal grammars and lexicons for NLP systems, starting from linguistic knowledge. Grammars have to represent all common linguistic phenomena and lexicons have to include the most frequent words with their most frequent uses. As everyone knows, building such resources is a very complex and time consuming task.

When one wants to formalize linguistic knowledge, a crucial question arises: which mathematical framework to choose? Currently, there is no agreement on the choice of a formalism in the scientific community. Each of the most popular formalisms has its own advantages and drawbacks. A good formalism must have three properties, hard to conciliate: it must be sufficiently expressive to represent linguistic generalizations, easily readable by linguists and computationally tractable. Guided by those principles, we advocate a recent formalism, Interaction Grammars (IGs) (Perrier, 2003), the goal of which is to synthesize two key ideas, expressed in two kinds of formalisms up to now: using the resource sensitivity of natural languages as a principle of syntactic composition, which is a characteristic feature of Categorial Grammars (CG) (Retoré, 2000), and viewing grammars as constraint systems, which is a feature of unification grammars such as LFG (Bresnan, 2001) or HPSG (Pollard and Sag, 1994).

Researchers who develop large lexicons and grammars from linguistic knowledge are confronted to the contradiction between the necessity to choose a specific grammatical framework and the cost of developing resources for this framework. One of the most advanced systems devoted to such a task is LKB (Copestake, 2001). LKB allows grammars and lexicons to be developed for different languages, but only inside the HPSG framework, or at most a typed feature structure framework. Therefore, all produced resources are hardly re-usable for other frameworks. Our goal is to design a toolchain that is as much as possible re-usable for other frameworks than IG.

Our toolchain follows the following architecture (see Figure 1):

- First, for building grammars, we use XMG (Section 3.1) which translates the *source grammar* into an *object grammar*.
- IGs that we have developed with XMG are all lexicalized. Therefore, the object grammar has to be anchored in a lexicon (Section 3.2) in order to produce the *anchored grammar*.
- Then, when analyzing a sentence, we start with a lexical disambiguation module (Section 3.3).
- The resulting lexical selections, presented in the compact form of an *automaton*, are finally sent to the LEOPAR parser (Section 3.4).

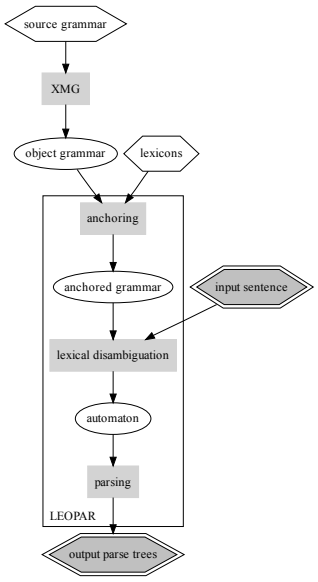

Figure 1: Toolchain architecture

2 Interaction Grammars

IGs (Perrier, 2003) are a grammatical formalism based on the notion of polarity. Polarities express the resource sensitivity of natural languages by modeling the distinction between saturated and unsaturated syntactic structures. Syntactic composition is represented as a chemical reaction guided by the saturation of polarities. In a more precise way, syntactic structures are underspecified trees equipped with polarities expressing their saturation state. They are superposed under the control of polarities in order to saturate them. In CG, Tree Adjoining Grammars (TAGs) and Dependency Grammars, syntactic composition can also be viewed as a mechanism for saturating polarities, but this mechanism is less expressive because node merging is localized at specific places (root nodes, substitution nodes, foot nodes, adjunction nodes ...). In IGs, tree superposition is a more flexible way of realizing syntactic composition. Therefore, it can express sophisticated constraints on the environment in which a polarity has to be saturated. From this angle, IGs are related to Unification Grammars, such as HPSG, because tree superposition is a kind of unification, but with an important difference: polarities play an essential role in the control of unification.

3 Description of the Toolchain

3.1 The XMG Grammar Compiler

The first piece of software in our toolchain is XMG[1] (Duchier et al., 2004), a tool used to develop grammars. XMG addresses the issue of designing wide-coverage grammars: it is based on a distinction between *source grammar*, written by a human, and *object grammar*, used in NLP systems. XMG provides a high level language for writing source grammars and a compiler which translates those grammars into operational object grammars.

XMG is particularly adapted to develop lexicalized grammars. In those grammars, parsing a sentence amounts to combining syntactical items attached to words. In order to have an accurate language model, it may be necessary to attach a huge number of syntactical items to some words (verbs and coordination words, in particular) that describe the various usages of those words. In this context, a grammar is a collection of items representing syntactical behaviors. Those items, although different from each other, often share substructures (for instance, almost all verbs have a substructure for subject verb agreement). That is to say, if a linguist wants to change the way subject-verb agreement is modeled, (s)he would have to modify all the items containing that substructure. This is why designing and maintaining strongly lexicalized grammars is a difficult task.

The idea behind the so-called metagrammatical approach is to write only substructures (called fragments) and then add rules that describe the combinations (expressed with conjunctions, disjunctions and unifications) of those fragments to obtain complete items.

Fragments may contain syntactic, morpho-syntactic and semantic pieces of information. An object grammar is a set of structures containing syntactic and semantic information, that can be anchored using morpho-syntactic information stored in the *interface* of the structure (see Section 3.2).

During development and debugging stages, portions of the grammar can be evaluated independently. The grammar can be split into various modules that can be shared amongst grammars. Finally, graphical tools let the users explore the in-

[1] XMG is freely available under the CeCILL license at `http://sourcesup.cru.fr/xmg`

heritance hierarchy and the partial structures before complete evaluation.

XMG is also used to develop TAGs (Crabbé, 2005) and it can be easily extended to other grammatical frameworks based on tree representations.

3.2 Anchoring the Object Grammar with a Lexicon

The tool described in the previous section builds the set of elementary trees of the grammar. The toolchain includes a generic anchoring mechanism which allows to use formalism independent linguistic data for the lexicon part.

Each structure produced by XMG comes with an interface (a two-level feature structure) which describes morphological and syntactical constraints used to select words from the lexicon. Dually, in the lexicon, each inflected form of the natural language is described by a set of two-level feature structures that contain morphological and syntactical information.

If the interface of an unanchored tree unifies with some feature structure associated with w in the lexicon, then an anchored tree is produced for the word w.

The toolchain also contains a modularized lexicon manager which aims at easing the integration of external and formalism independent resources. The lexicon manager provides several levels of linguistic description to factorize redundant data. It also contains a flexible compilation mechanism to improve anchoring efficiency and to ease lexicon debugging.

3.3 Lexical Disambiguation

Neutralization of polarities is the key mechanism in the parsing process as it is used to control syntactic composition. This principle can also be used to filter lexical selections. For a input sentence, a *lexical selection* is a choice of an elementary tree from the anchored grammar for each word of the sentence.

Indeed, the number of possible lexical selections may present an exponential complexity in the length of the sentence. A way of filtering them consists in abstracting some information from the initial formalism F to a new formalism F_{abs}. Then, parsing in F_{abs} allows to eliminate wrong lexical selections at a minimal cost (Boullier, 2003). (Bonfante et al., 2004) shows that polarities allow original methods of abstraction.

Following this idea, the lexical disambiguation module checks the global neutrality of every lexical selection for each polarized feature: a set of trees bearing negative and positive polarities can only be reduced to a neutral tree if the sum of the negative polarities for each feature equals the sum of its positive polarities.

Counting the sum of positive and negative features can be done in a compact way by using an automaton. This automaton structure allows to share all paths that have the same global polarity balance (Bonfante et al., 2004).

3.4 The LEOPAR Parser

The next piece of software in our toolchain is a parser based on the IGs formalism[2]. In addition to a command line interface, the parser provides an intuitive graphical user interface. Parsing can be highly customized in both modes. Besides, the processed data can be viewed at each stage of the analysis via the interface so one can easily check the behavior of the grammar and the lexicons in the parsing process.

The parsing can also be done manually: one first chooses a lexical selection of the sentence given by the lexer and then proceeds to the analysis by neutralizing nodes from the selection. This way, the syntactic composition can be controlled by the user.

4 Results

Our toolchain has been used first to produce a large coverage French IG. Most of the usual syntactical constructions of French are covered. Some non trivial constructions covered by the grammar are, for instance: coordination, negation (in French, negation is expressed with two words with complex placement rules), long distance dependencies (with island constraints). The object grammar contains 2,074 syntactic structures which are produced by 455 classes in the source grammar.

The French grammar has been tested on the French TSNLP (Test Suite for the Natural Language Processing) (Lehmann et al., 1996); this test suite contains around 1,300 grammatical sentences and 1,600 ungrammatical ones. The fact that our

[2] LEOPAR is freely available under the CeCILL license at `http://www.loria.fr/equipes/calligramme/leopar`

grammar is based on linguistic knowledge ensures a good coverage and greatly limits overgeneration: 88% of the grammatical sentences are correctly parsed and 85% of the ungrammatical sentences are rejected by our grammar.

A few months ago, we started to build an English IG. The modularity of the toolchain was an advantage to build this grammar by abstracting the initial grammar and then specifying the abstract kernel for English. The English TSNLP has been used to test the new grammar: 85% of the grammatical sentences are correctly parsed and 84% of the ungrammatical sentences are rejected. It is worth noting that those scores are obtained with a grammar that is still being developed .

5 Future work

The toolchain we have presented here aims at producing grammars and lexicons with large coverage from linguistic knowledge. This justifies the choice of discarding statistical methods in the first stage of the toolchain development: in the two steps of lexical disambiguation and parsing, we want to keep all possible solutions without discarding even the less probable ones. Now, in a next future, we have the ambition of using the toolchain for parsing large raw corpora in different languages.

For French, we have a large grammar and a large lexicon, which are essential for such a task. The introduction of statistics in the two modules of lexical disambiguation and parsing will contribute to computational efficiency. Moreover, we have to enrich our parsing strategies with robustness. We also ambition to integrate semantics into grammars and lexicons.

Our experience with English is a first step to take multi-linguality into account. The crucial point is to make our grammars evolve towards an even more multi-lingual architecture with an abstract kernel, common to different languages, and different specifications of this kernel for different languages, thus following the approach of the Grammatical Framework (Ranta, 2004).

Finally, to make the toolchain evolve towards multi-formalism, it is first necessary to extend XMG for more genericity; there is no fundamental obstacle to this task. Many widespread formalisms can then benefit from our original methods of lexical disambiguation and parsing, based on polarities. (Kahane, 2006) presents the polarization of several formalisms and (Kow, 2007) shows that this way is promising.

References

Bonfante, G., B. Guillaume, and G. Perrier. 2004. Polarization and abstraction of grammatical formalisms as methods for lexical disambiguation. In *CoLing'2004, 2004*, pages 303–309, Geneva, Switzerland.

Boullier, P. 2003. Supertagging: A non-statistical parsing-based approach. In *IWPT 03*, pages 55–65, Nancy, France.

Bresnan, J. 2001. *Lexical-Functional Syntax*. Blackwell Publishers, Oxford.

Copestake, A. 2001. *Implementing Typed Feature Structure Grammars*. CSLI Publications.

Crabbé, B. 2005. *Représentation informatique de grammaires fortement lexicalisées : application à la grammaire d'arbres adjoints*. Phd thesis, Université Nancy 2.

Duchier, D., J. Le Roux, and Y. Parmentier. 2004. The metagrammar compiler : A NLP Application with a Multi-paradigm Architecture. In *Second International Mozart/Oz Conference - MOZ 2004, Charleroi, Belgium*.

Kahane, S. 2006. Polarized unification grammars. In *21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 137–144, Sydney, Australia.

Kow, E. 2007. *Surface realisation: ambiguity and determinism*. Phd thesis, Université Nancy 2.

Lehmann, S., S. Oepen, S. Regnier-Pros, K. Netter, V. Lux, J. Klein, K. Falkedal, F. Fouvry, D. Estival, E. Dauphin, H. Compagnion, J. Baur, L. Balkan, and D. Arnold. 1996. TSNLP — Test Suites for Natural Language Processing. In *CoLing 1996, Kopenhagen*.

Perrier, G. 2003. *Les grammaires d'interaction*. Habilitation thesis, Université Nancy 2.

Pollard, C.J. and I.A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press.

Ranta, A. 2004. Grammatical Framework: A Type-Theoretical Grammar Formalism. *Journal of Functional Programming*, 14(2):145–189.

Retoré, C. 2000. The Logic of Categorial Grammars. ESSLI'2000, Birmingham.