FRIGRAM: a French Interaction Grammar

Guy Perrier, Bruno Guillaume

FRIGRAM: a French Interaction Grammar

Guy Perrier and Bruno Guillaume

LORIA, Université de Lorraine, Nancy, France

Abstract. We present FRIGRAM, a French grammar with a large coverage, written in the formalism of Interaction Grammars. The originality of the formalism lies in its system of polarities, which expresses the resource sensitivity of natural languages and which is used to guide syntactic composition. We focus the presentation on the principles of the grammar, its modular architecture, the link with a lexicon independent of the formalism and the companion property, which helps to guarantee the consistency of the whole grammar.

Keywords: Formal Grammar, Model Theoretic Syntax, Polarity, Interaction Grammar.

1 Introduction

The aim of our work is to show that it is possible to build a realistic computational grammar of French, which integrates fine linguistic knowledge with a large coverage. As a framework, we have chosen the formalism of Interaction Grammar (IG) [7]. IG combines a flexible view of grammars as constraint systems with the use of a polarity system to control syntactic composition. The system of polarities expresses the saturation state of partial syntactic structures and their ability to combine together.

The main challenge is to guarantee and to maintain the consistency of the grammar while aiming at the largest coverage. We resort to several means:

- a modular organization of the grammar in a hierarchy of classes, which is able to capture the generalizations of the language,
- principles of well-formedness for the elementary structures of the grammar,
- a separation of the grammar itself from the lexicon, which is independent of any grammatical formalism,
- the use of the *companion property* to help the checking of the grammar consistency.

Starting with a brief presentation of IG, we continue with an explanation of the different points mentioned above and with a comparison with other French grammars and a discussion about the evaluation of the grammar.

2 Interaction Grammars

IG is a grammatical formalism which is devoted to the syntax of natural languages using two notions: *tree description* and *polarity*. For a complete presentation of the formalism, the reader can refer to [7].

2.1 Tree Descriptions

The notion of a tree description [11] is related to a model theoretic view of the syntax of natural languages [10]. In this view, the basic objects of the grammar are not trees but properties that are used to describe them, in other words *tree descriptions*. This approach is very flexible allowing the expression of elementary properties in a totally independent way, and their combination in a free manner. A tree description can be viewed either as an underspecified tree, or as the specification of a tree family, each tree being a model of the specification.

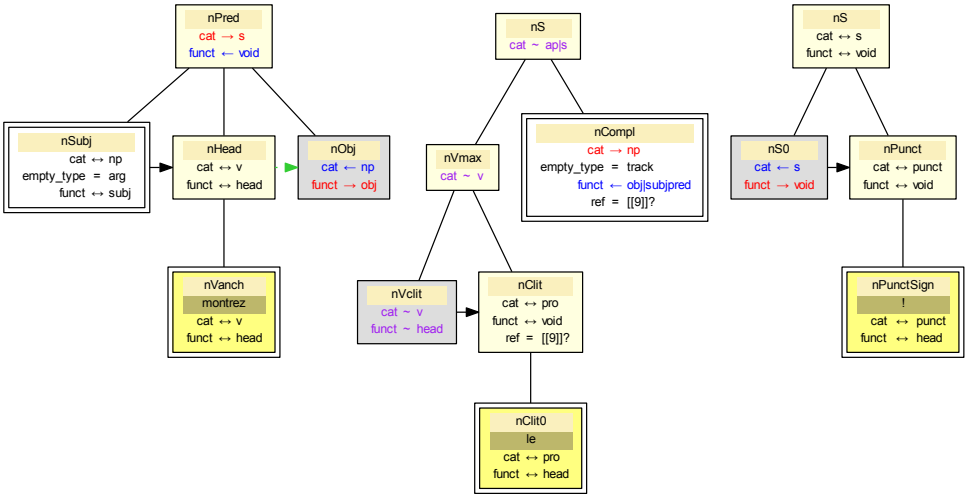

Fig. 1. PTD associated with the sentence *"montrez-le !"* by the grammar FRIGRAM.

Figure 1 gives an example of the tree description, which is associated with the sentence *"montrez-le !"* [*"show it !"*]. Even if the description is composed of three parts associated with the three words of the sentence (punctuation signs are considered as words), it must be considered as a unique tree description.

A tree description is a finite set of nodes structured by two kinds of relations: *dominance* and *precedence*. Dominance relations can be immediate or large. In the example, there are only immediate dominance relations represented with solid lines. Precedence relations can also be immediate or large. They are represented with arrows in Figure 1; these arrow are solid and black or dashed and green, depending on whether the dependencies are immediate or large.

Nodes, which represent constituents, are labelled with features describing their morpho-syntactic properties. Feature values are atoms or atom disjunctions. When a feature value is the disjunction of all elements of a domain, this

value is denoted with "?". A mechanism of co-indexation between feature values (a common index $[[n]]$ is put before their values) allows for sharing.

It is possible to add constraints on the phonological form and on the daughters of nodes: a node is declared to be *empty* if its phonological form is empty and it is graphically represented with a white rectangle; at the opposite, it is declared to be *full*, and it is represented with a light-yellow rectangle; it is declared to be *closed*, if the set of its daughters is fixed and it is represented with a double rectangle; finally, a node is declared to be an *anchor*, if it is a full leaf, and it is used to anchor a word of the language. An anchor is represented with a canary yellow rectangle.

IG uses three kinds of empty nodes:

- when an argument has moved from its canonical position, this position is marked with a *trace*, an empty node with the feature `empty_type = track`; this covers all cases of extraction, subject inversion and cliticization of arguments; in Figure 1, node *nCompl* is the empty trace of the object represented with the clitic pronoun *"le"*;
- when an argument is not expressed with a phonological form, it is represented with an empty node carrying the feature `empty_type = arg`; this is the case for subjects of adjectives, infinitives and imperatives, as well as some objects of infinitives (tough movement); in Figure 1, node *nSubj* represents the non-expressed subject of the imperative verb *"montrez"*;
- in presence of an ellipsis, the head of the elided expression may be represented with an empty node carrying the feature `empty_type = ellipsis`.

2.2 Polarities

Polarities are used to express the saturation state of syntactic trees. They are attached to features that label description nodes with the following meaning:

- a *positive* feature `f` $\rightarrow$ `v` expresses an available resource, which must be consumed;
- a *negative* feature `f` $\leftarrow$ `v` expresses an expected resource, which must be provided; it is the dual of a positive feature; one negative feature must match exactly one corresponding positive feature to be saturated and conversely;
- a *saturated* feature `f` $\leftrightarrow$ `v` expresses a linguistic property that needs no combination to be saturated;
- a *virtual* feature `f` $\sim$ `v` expresses a linguistic property that needs to be realized by combining with an actual feature (an actual feature is a positive or saturated feature).

In Figure 1, node *nObj* carries a negative feature `cat` $\leftarrow$ `np` and a positive feature `funct` $\rightarrow$ `obj`, which represents the expected object noun phrase for the transitive verb *"montrez"*.

The virtual features of the second part of the tree description represent the syntactic context required by the clitic pronoun *"le"*: a verb *nVclit* put immediately before the pronoun to build the node *nVmax* with it.

Only resource sensitive features are polarized. Other features are called *neutral* features and denoted `f = v`. For instance, agreement properties are expressed with neutral features.

The descriptions labelled with polarized features are called *polarized tree descriptions (PTDs)* in the rest of the article.

2.3 Grammars as constraint systems

An interaction grammar is defined by a finite set of Elementary PTDs, named EPTDs in the following, and it generates a tree language. A tree belongs to the language if it is a model of a finite set of EPTDs in the sense given by [7]. Each node of the EPTDs is mapped to a node of the model through an interpretation function. Properties of models include:

- A model is *saturated*: every positive feature `f` $\rightarrow$ `v` is matched with its dual feature `f` $\leftarrow$ `v` in the model and vice versa. Moreover, every virtual feature has to find an actual corresponding feature in the model.
- A model is *minimal*: it has to add a minimum of information to the initial descriptions (it cannot add immediate dominance relations or features that do not exist in the initial descriptions).

Parsing a sentence with a grammar G consists first in selecting an appropriate set of EPTDs from G. The selection step is facilitated if G is lexicalized: each EPTD has an anchor associated with a word of the language. It strongly reduces the search space for the EPTDs. Then, the parsing process itself reduces to the resolution of a constraint system. It consists in building all models of the selected set of EPTDs.

Figure 1 represented a possible selection of EPTDs from FRIGRAM to parse the sentence *"montrez-le !"*. The selection includes three EPTDs[1], which are gathered in a unique PTD. Figure 2 shows the unique minimal and saturated model of the PTD. It is an ordered tree where nodes are labelled with non polarized features in the form `f : v`, where `v` is an atomic value. In the head of each node, a list gives the nodes of the PTD that are interpreted in the node of the model.

In an operational view of parsing, the building of a saturated and minimal model is performed step by step by refining the initial PTD with a merging operation between nodes, guided by one of the following constraints:

- neutralise a positive feature with a negative feature having the same name and carrying a value unifiable with the value of the first feature;
- realize a virtual feature by combining it with an actual feature (a positive or saturated feature) having the same name and carrying a value unifiable with the value of the first feature.

[1] The EPTDs are labelled by the name of the class of the grammar generating them followed by a number. In the order of the sentence, we have NP0_V_NP1_134, PROCLIT_COMPL_IMPERS_POS_38 and PUNCTSTOP_S1INTER_IMPER_1.

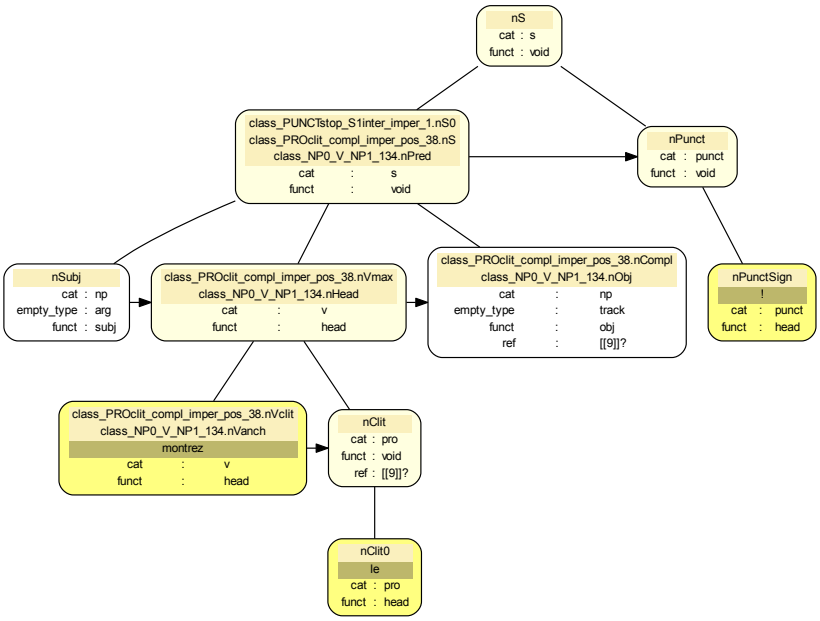

Fig. 2. Model of the PTD shown in Figure 1 representing the syntax of the sentence *"montrez-le !"*.

The constraints of the description interact with node merging to entail a partial superposition of their contexts represented by the tree fragments in which they are situated. So the model of Figure 2 can be obtained from the PTD of Figure 1 with a sequence of three node merging operations: *nVanch* with *nVclit*, *nObj* with *nCompl* and *nPred* with *nS0*.

To summarize, IG combine the strong points of two families of formalisms: the flexibility of *Unification Grammars* and the saturation control of *Categorial Grammars*.

3 The Principles of the Grammar FRIGRAM

FRIGRAM is a IG for the French language; it contains 3 794 EPTD templates. FRIGRAM follows a set of principles which express formally the chosen linguistic modeling. These principles are also used to automatically check the consistency of the grammar and its conformity to linguistic principle. The constituant to dependency transformation used with FRIGRAM also strongly relies on this set of principles.

Definition 1. *A node with a positive or saturated* `cat` *feature is called a* concrete node*.*

Principle 1 *(***cat-funct***) In an EPTD, any node has a* `cat` *feature and if it is concrete, it has also a* `funct` *feature.*

The consequence is that any node of a model has a `cat` feature and a `funct` feature. Another consequence is that any node of a model has a unique concrete antecedent in the original PTD, because two concrete nodes of a PTD cannot merge in the model, according to the composition rules of polarities.

Principle 2 *(***strict lexicalisation***) Any EPTD has exactly one anchor node. This anchor node has a saturated* `cat` *feature with an atomic feature value.*

Definition 2. *A* spine *in an EPTD is a list of nodes $N_1, N_2, \ldots, N_p$ such that:*

- *for any i such that $1 < i \leq p$, node N_i is a daughter node of N_{i-1};*
- *for any i such that $1 < i \leq p$, node N_i has a saturated feature* `cat` *and a feature* `funct` $\leftrightarrow$ `head`*;*
- *node N_1 is a concrete node and its feature* `funct` *has a value different from* `head`*; it is called the* maximal projection *of all nodes belonging to the spine;*
- *node N_p is either an anchor or an empty leaf; in the first case, the spine is called a* main spine*; in the second case, it is called an* empty spine*; in both cases, node N_p is called the* lexical head *of all nodes belonging to the spine.*

Principle 3 *(***spine***) Any concrete node of an EPTD belongs to exactly one spine.*

An important corollary of the spine principle is that every node N of a PTD model has exactly one lexical head in this model, denoted $head(N)$ and defined as follows: the concrete antecedent of N in the initial PTD belongs to exactly one spine and $head(N)$ is the interpretation in the model of the leaf ending the spine.

A second important corollary is that every node in a PTD model which is not a leaf has exactly one daughter node with the feature `funct` : `head`. By following all nodes with this feature, we have a more direct way of finding the lexical head of every node in a PTD model.

A third corollary is that each model node with a positive feature `cat` is the maximal projection of some spine.

From the strict lexicalisation and spine principles, we can also deduce that every EPTD has exactly one main spine.

To illustrate the concept of a spine, let us consider the EPTDs of Figure 1. The EPTD associated with the verb *"montrez"* has two spines: the main spine *nPred*, *nHead*, *nVanch* with its lexical head *nVanch*, and an empty spine reduced to a single node *nSubj*. The formalism of IG is situated in the constituency approach to syntax, as opposed to the dependency approach but the principles of FRIGRAM allow for an automatic transformation of any parse made with IG from a constituency setting into a dependency setting. Our purpose here is not to describe the transformation in detail but to give an outline of it.

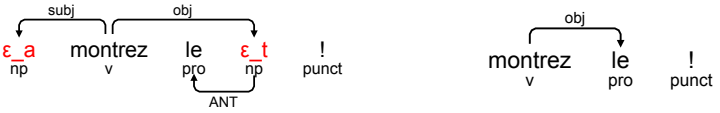

Fig. 3. The dependency graphs representing the syntax of the sentence *"montrez-le !"*.

The dependencies are generated by the interactions between the polarized features `cat` and `funct` of the different nodes of the initial PTD and they are projected on the full and empty words of the sentence through the notion of lexical head.

For the sentence *"montrez-le !"*, from the PTD of Figure 1 and its model from Figure 2, we compute the dependency graph on the left of Figure 3. It has two empty words, the subject of *"montrez"*, named ϵ_a (which corresponds to the node with feature `empty_type = arg`), and its object ϵ_t (which corresponds to the node with feature `empty_type = track`), which is the trace of the clitic pronoun *"le"*. Traces are linked to their antecedent with the relation ANT.

In a second step, the empty words are removed and their incident dependencies are transferred to their full antecedent, when it exists. In our very simple example, the resulting dependency graph reduces to the tree on the right of Figure 3, but in more complex sentences, the dependency graph includes cycles and nodes with several governors. When there is more than one solution, handcrafted rules are used to compute a weight for each solution in order to rank them.

4 The Architecture of the Grammar

4.1 The Modular Organisation of the Grammar

It is unthinkable to build a grammar with about 4000 EPTD templates manually, considering each one individually. Even if it were possible, to maintain the consistency of such a grammar would be intractable.

Now, the EPTD templates of FRIGRAM share a lot of fragments and it is possible to organize the grammar as a class hierarchy. A tool, XMG [4], was specially designed to build such kind of grammars. XMG provides a language to define a grammar as a set of classes. A class can be defined directly but also from other classes by mean of two composition operations: *conjunction* and *disjunction*.

Each class is structured according to several dimensions. FRIGRAM uses two dimensions: the first one is the syntactic dimension, where objects are EPTD templates, and the second one is the dimension of the interface with the lexicon, where objects are feature structures.

The terminal classes of the hierarchy define the EPTD templates of the grammar that are computed by the XMG compiler. Figure 4 gives the example of a

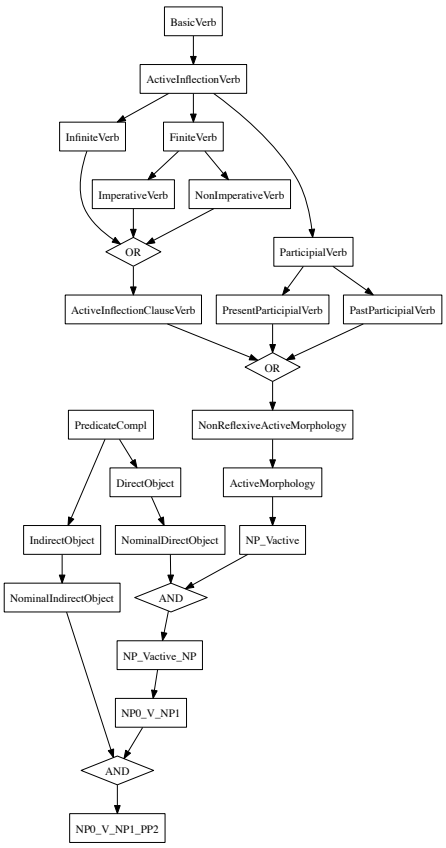

Fig. 4. The hierarchy of classes used to define the NP0_V_NP1_PP2 class of transitive verbs with an indirect complement

terminal class, the NP0_V_NP1_PP2 class of transitive verbs with an indirect complement, with the hierarchy of classes used to define it.

The current grammar FRIGRAM is composed of 428 classes, including 179 terminal ones, which are compiled into 3 794 EPTD templates. Of course, some general classes can be used in several different contexts. For instance, adjectives, nouns and verbs description all inherit from the same subclasses related to complements of predicative structures. The set of classes is organized in a module hierarchy 5.

There is another hierarchy related to the different forms of extraction, in relative, interrogative and cleft clauses. The root module of the hierarchy is the EXTRACTGRAMWORD module. Three modules depend on it: COMPLEMENTIZER, INTERROGATIVE and RELATIVE. Moreover, there are isolated modules related to specific categories.

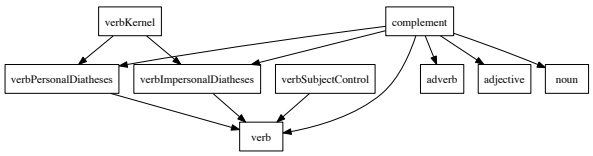

Fig. 5. The main hierarchy of modules

4.2 The link with a lexicon independent of the formalism

The full grammar is produced from the set of EPTD templates and a lexicon. Each EPTD template is associated to a feature structure (called its interface) which describes a syntactic frame corresponding to lexical units able to anchor it; lexicon entries are also described through features structures. Unification between interface of the EPTD template and lexicon feature structure is used to control the combination of a lexical unit description and the template. Thanks to the strict lexicalisation principle, each EPTD of the grammar has a unique anchor node linked with a lexical unit of the language. Since there is a co-indexation between features of the EPTD template and features of the interface, a side effect of anchoring is the instantiation of some feature values in the EPTD.

In our system, the lexicon used is FRILEX[2] which combines morphological information taken in ABU[3] and in Morphalou [12] with syntactical information for verbs from Dicovalence [14]. FRILEX contains 530 000 entries. To avoid size explosion, the full grammar is built on the fly on each input sentence.

5 The Companion Property and the Consistency of the Grammar

Our ambition is to build a grammar with a coverage of all of the most frequent phenomena of French syntax. Even if the hierarchical structure of the grammar makes it more compact ans eases the maintenance of its consistency, the size of the grammar may be important and the grammar offers no global view of its contents.

To verify the consistency of the grammar, it is necessary to check the behavior of each EPTD in the composition process with other EPTDs. A way of doing it is to parse corpora with the grammar but this is a very partial checking. Now, the formalism of IG provides a mechanism to verify the consistency of a grammar in a static way based on the EPTDs of the grammar without using parsing. The mechanism uses the *Companion Property*.

Originally, this property was introduced by [1] to perform lexical disambiguation with IG. Let us consider an interaction grammar.

[2] `http://wikilligramme.loria.fr/doku.php?id=frilex`
[3] `http://abu.cnam.fr/DICO/mots-communs.html`

Definition 3. *A companion of a polarized feature in an EPTD E_1 of the grammar is a polarized feature of an EPTD E_2 such that the first feature is saturated by the second feature in a merging of their nodes leading to a consistent PTD.*

What we mean with "a consistent PTD" is that the PTD resulting from the node merging has at least one model, which is a tree but not necessarily minimal and saturated.

For instance, consider the EPTD associated with the verb *"montrez"* in Figure 1. A companion of the positive feature `funct` $\rightarrow$ `obj` is the negative feature `funct` $\leftarrow$ `obj|subjpred` of the EPTD associated with the clitic pronoun *"le"*. The notion of companion can be expressed a the template level: we compute systematically the companions of all polarized features of the EPTDs templates of the grammar. In this way, we limit the number of companions to compute.

So, for the EPTD template E_0 corresponding to the EPTD anchored with *"montrez"* in Figure 1 and for the positive feature `funct` $\rightarrow$ `obj` , we find 97 companions: 85 are right companions, that is, companions coming from EPTD templates for which the anchor is on the right of the anchor of E_0 after merging, and 12 are companions without order constraints on the anchor of their EPTD.

Among all the information given by the computation of all the companions, a particular part is immediately usable: the polarized features without companions. If they have no companion, their EPTDs cannot enter any parsing, which means that the EPTD template must be removed from the grammar or that there is some mistake in their definition.

6 Comparison with other French Grammars and Evaluation of the Grammar

There is very little work on the construction of French computational grammars from linguistic knowledge using semi-automatic tools. Historically, a very fruitful work was the PhD thesis of Candito [2] about the modular organization of TAGs, with an application to French and Italian. This thesis was a source of inspiration for the development of several French grammars.

A first grammar produced according to this approach and able to parse large corpora was FRMG [15]. FRMG falls within the TAG formalism and its originality lies in the use of specific operators on nodes to factorize trees: disjunction, guards, repetition and shuffling. As a consequence, the grammar is very compact with only 207 trees. Moreover, these trees are not written by hand but they are automatically produced from a multiple inheritance hierarchy of classes.

Another French grammar inspired by [2] is the French TAG developed by [3]. Like Frigram, this grammar was written with XMG. Contrary to FRMG, it is constituted of classical TAG elementary trees, hence its more extensive form: it includes 4200 trees and essentially covers verbs. It was a purely syntactic grammar and then it was extended in the semantic dimension by [5] for generation.

To evaluate the soundness of Frigram and to compare its coverage with other French grammars is problematic. The first difficulty is that there is no

robust parser able to deal with IG. The tool developed so far (LEOPAR [6]) was designed to experiment, to test and to help grammar development. It was latter enriched with filtering algorithms to improve the supertagging stages of the parsing process. Nevertheless, it does not have any robust mechanism to deal with sentences that are not completely covered by the grammar. After filtering steps, deep parsing relies on an exhaustive search of tree description models which is an NP-hard task. As a consequence, LEOPAR can be used to parse sentence of length up to 15 words. FTB contains 3398 sentences of length lower than 15. Moreover, all linguistic phenomena present in real corpora, like the FTB, cannot be modeled through a lexicalized grammar: dislocation, coordination of non constituents, parenthetical clauses ... These phenomena require an extra-grammatical treatment, which is not yet implemented in LEOPAR. Thus, we consider the subset of sentence without explicit extra-grammatical phenomenon (parenthesis, reported speech); there are 2166 such sentences. The parser LEOPAR with the FRIGRAM resource is able to parse 56.4% of the sentences considered.

Another way to evaluate a grammar coverage is to use test suites. Such suites must include not only positive examples but also negative examples to test the overgeneration of the grammar. There exists such a suite for French, the TSNLP[8], but unfortunately, it ignores a lot of phenomena that are very frequent in French. On the set of grammatical sentences of the TSNLP, LEOPAR and FRIGRAM is able to parse 88% of the sentences. This is equivalent to the number achieved in [9] but the remaining sentences correspond to sentences that should be covered by the robustness of the parser rather than by the detailled grammar (unusual kind of coordination, sentence with incomplete negations, ...)

To try to deal with TSNLP drawbacks, we have designed our own test suite which si complementary to the TSNLP; it contains 874 positive sentences and 180 negative ones. 93% of the grammatical sentences are parsed and the ratio is 21% for ungrammatical sentences. The reader can find the test suite on a web page[4]. For the positive sentences, there is also the result of parsing in the form of a dependency graph. The variety of the examples gives a good idea of the coverage of FRIGRAM and the richness of dependency graphs helps to understand the subtlety of the grammar.

7 Conclusion

The next step to go ahead with FRIGRAM is to solve the bottleneck of the parser LEOPAR in order to parse raw corpora. We need to improve the efficiency of the parser to contain the possible explosion resulting from the increase of the grammar size in combination with the increased sentence length. It is also necessary to take robustness into account in the parsing algorithm and to add extra-grammatical procedures to deal with phenomena that go beyond the lexicalized grammar. For English, [13] is a first attempt to build a IG grammar

[4] `http://wikilligramme.loria.fr/doku.php?id=hmge_2013`

that should be extended in order to have a coverage equivalent to the one of FRIGRAM.

References

1. G. Bonfante, B. Guillaume, and M. Morey. Polarization and abstraction of grammatical formalisms as methods for lexical disambiguation. In *11th International Conference on Parsing Technology, IWPT'09*, Paris, France, 2009.
2. M.-H. Candito. *Organisation modulaire et paramétrable de grammaires électroniques lexicalisées. Application au français et à l'italien,*. Thèse d'université, Université Paris 7, 1999.
3. B. Crabbé. *Représentation informatique de grammaires fortement lexicalises : application à la grammaire d'arbres adjoints.* thèse de doctorat, université Nancy2, 2005.
4. B. Crabbé, D. Duchier, C. Gardent, J. Le Roux, and Y. Parmentier. XMG : eXtensible MetaGrammar. *Computational Linguistics*, 39(3):1–66, 2013.
5. C. Gardent and Y. Parmentier. SemTAG: a platform for specifying Tree Adjoining Grammars and performing TAG-based Semantic Construction. In *45th Annual Meeting of the Association for Computational Linguistics*, pages 13–16, Prague, Tchèque, République, 2007.
6. B. Guillaume, J. Le Roux, J. Marchand, G. Perrier, K. Fort, and J. Planul. A Toolchain for Grammarians. In *Coling 2008*, pages 9–12, Manchester, Royaume-Uni, 2008.
7. B. Guillaume and G. Perrier. Interaction Grammars. *Research on Language and Computation*, 7:171–208, 2009.
8. S. Lehmann, S. Oepen, S. Regnier-Prost, K. Netter, V. Lux, J. Klein, K. Falkedal, F. Fouvry, D. Estival, E. Dauphin, H. Compagnion, J. Baur, L. Balkan, and D. Arnold. TSNLP — Test Suites for Natural Language Processing. In *Proceedings of COLING 1996, Kopenhagen*, 1996.
9. G. Perrier. A French interaction grammar. In *International Conference on Recent Advances in Natural Language Processing (RANLP 2007)*, pages 463–467, Borovets, Bulgaria, September 27-29 2007.
10. G. K. Pullum and B. C. Scholz. On the Distinction between Model-Theoretic and Generative-Enumerative Syntactic Frameworks. In *LACL 2001, Le Croisic, France*, volume 2099 of *Lecture Notes in Computer Science*, pages 17–43, 2001.
11. J. Rogers and K. Vijay-Shanker. Obtaining trees from their descriptions: an application to tree-adjoining grammars. *Computational Intelligence*, 10(4):401–421, 1994.
12. L. Romary, S. Salmon-Alt, and G. Francopoulo. Standards going concrete: from LMF to Morphalou. In M. Zock, editor, *COLING 2004 Enhancing and using electronic dictionaries*, pages 22–28, Geneva, Switzerland, August 29th 2004. COLING.
13. S. Tabatabayi Seifi. An interaction grammar for English verbs. In R. K. Rendsvig and S. Katrenko, editors, *Proceedings of the ESSLLI 2012 Student Session*, pages 160–169, Opole, Poland, August 6–17 2012.
14. K. Van den Eynde and P. Mertens. La valence : l'approche pronominale et son application au lexique verbal. *French Language Studies*, 13:63–104, 2003.
15. É. Villemonte De La Clergerie. Building factorized TAGs with meta-grammars. In *The 10th International Conference on Tree Adjoining Grammars and Related Formalisms - TAG+10*, pages 111–118, New Haven, CO, États-Unis, 2010.