# A COMPUTATIONAL IMPLEMENTATION OF PERIPHRASTIC VERB CONSTRUCTIONS IN FRENCH

Leonel Figueiredo de ALENCAR*

▪ ABSTRACT: This paper describes the treatment of passive and compound past tense in FrGramm, a computational grammar of French, implemented within Lexical-Functional Grammar (LFG) using the XLE software. Due to the dual auxiliary system and past participle agreement, the latter periphrasis manifests greater structural complexity and therefore presents a greater challenge to computational implementation in French than in languages such as English and Portuguese. An additional difficulty is modeling the morphological and syntactic-semantic regularities of the passive construction. In FrGramm, this problem is solved by means of a productive lexical rule. FrGramm also implements the constraints governing the building of both verbal periphrases, excepting participle object agreement. The implementation was evaluated by applying a parser to a set of 157 grammatical sentences and a set of 279 ungrammatical sentences. All sentences from the former set were correctly parsed. Only two constructions from the latter set that violate the linear precedence of the compound past auxiliary over the passive auxiliary were analyzed as grammatical. FrGramm is the only LFG grammar of French with similar coverage that is freely available on-line. A future version will handle participle object agreement and avoid the mentioned overgeneration.

▪ KEYWORDS: Computational linguistics. Deep syntactic parsing. Lexical-Functional grammar. LFG/XLE. Finite-state morphology. French verbal periphrases. Passive voice.

## Introduction

Lexical-Functional Grammar (LFG) is a framework within generative theory. It is widely disseminated in theoretical, descriptive, and typological studies as well as in computational linguistics. This model underlies analyses of a great number of languages, from many different linguistic families (BRESNAN, 2001). Many of these analyses were computationally implemented, in part within deep parsing systems development projects, aimed at the semantic processing of natural language texts.[1]

* Federal University of Ceará (UFC), Fortaleza – CE – Brazil. Associate Professor of the Department of Foreign Languages. leonel.de.alencar@ufc.br.

[1] The most up-to-date and comprehensive survey of languages with computational grammars (or fragments of grammar) implemented in the LFG formalism contains 27 languages (MÜLLER, 2016, p. 213-214).

As far back as in the 1990s, French was one of the first languages whose syntactic structures were mathematically described in the LFG formalism and implemented in parsing systems (ZWEIGENBAUM, 1991; FRANK, 1996; SCHWARZE, 1998; BUTT et al., 1999). Due to the technological advancement, some of these approaches, such as Frank (1996) and Schwarze (1998), have become obsolete because of the impracticability of running the respective parsing systems on current platforms. Another problem is that the source codes of these implementations or the corresponding parsers are not freely available. This is also the case with the more recent approaches, namely the parsers SxLFG and XLFG as well as the French grammar built to test them (CLÉMENT; KINYON, 2001; BOULLIER; SAGOT; CLÉMENT, 2005; CLÉMENT, 2014; SAGOT, [2015?]).

This paper focuses on the treatment of periphrastic verbal constructions in FrGramm, a French computational grammar we have recently implemented in the Xerox Linguistic Environment (XLE).[2] This software represents the current state of the art in the development and parsing of grammars in the LFG formalism (CROUCH et al., 2011). Thanks to its user-friendliness and efficiency, this system has been used for more than a decade in both teaching and research as well as in industrial scale applications. In addition, it is distributed free of charge, under a non-commercial use license.[3]

An alternative to XLE is XLFG, which is more focused on teaching and research in LFG (CLÉMENT; KINYON, 2001). However, this system is not available for download. Instead, it must be used online (CLÉMENT, 2014). On the other hand, the most complete French grammar implemented in this system, and made available on its site, has very limited coverage. It analyzes periphrases with the auxiliary *avoir* 'to have', but overgenerates broadly, since it implements only a small part of the constraints involved in these constructions.

Following a common practice in the literature, the notational variant of the LFG formalism implemented in XLE is designated in this paper as LFG/XLE. The main motivation for the development of FrGramm in LFG/XLE was to make freely available a French grammar with medium syntactic coverage for use in teaching and research in areas such as formal grammatical theory, computational linguistics, or natural language processing. Before FrGramm, the only widely available grammar for use with XLE in a pedagogical context was the English grammar in the system's documentation (KING, 2004). However, as it is well known, French has several syntactic peculiarities in relation to English. Consequently, an adaptation of this English grammar to French is far from trivial. On the other hand, translating the

[2] This paper deepens some aspects of Schwarze and Alencar (2016), which is an introductory textbook in German about LFG theory and the development of computational grammars in XLE using French examples. FrGramm is a significantly improved version of the grammar fragment from this book's chapter 8. In the division of labor for the elaboration of this work, it was up to the author of this article to implement the different grammar fragments and to write the respective chapter sections. These grammars reflect intense dialogue between the two authors. For any errors, however, this author assumes full responsibility.

[3] To obtain XLE, see <http://www2.parc.com/isl/groups/nltt/xle/>.

mentioned French grammar of Clément (2014) from the XLFG formalism into the XLE notation would not be so difficult for an experienced user. The problem, however, is that this grammar, as we have pointed out, has very limited coverage and treats as grammatical simple examples that violate the regularities in the formation of compound tenses in French.

FrGramm, which has much wider coverage, does not suffer from this type of overgeneration. Among other approaches, it incorporates elements of Frank (1996) and Schwarze (1998), the two computational grammars of French whose implementations are sufficiently detailed in the literature. It is not, however, a reimplementation. On the contrary, it has been developed from scratch and fills gaps in these two proposals. It is the only French grammar in LFG/XLE that is freely distributed on the Internet, under a license allowing modifications by the users and redistribution of the modified versions.[4] It can thus be extended to cover other phenomena, adapted to different grammatical approaches or to other languages.

Various syntactic phenomena were implemented in FrGramm 1.0, the current version of the grammar. Among these, the periphrastic verbal constructions in (1)-(5) stand out, due to greater complexity and greater contrast with the analogous facts in English. These periphrases consist of a finite form (henceforth VFIN) of *être* 'to be' or *avoir* 'to have' and a participle (henceforth PTCP).

(1) La fée est arrivée.
[the:F;SG fairy(F)[SG] be;PRS;3s arrive:PTCP:F;SG]
'The fairy arrived.'

(2) La fée a dansé.
[the:F;SG fairy(F)[SG] have;PRS;3s dance:PTCP]
'The fairy danced.'

(3) La fée est annoncée.
[the:F;SG fairy(F)[SG] be;PRS;3s announce:PTCP:F;SG]
'The fairy is announced.'

(4) La reine a forcé les chevaliers
[the:F;SG queen(F)[SG] have;PRS;3s force:PTCP the:PL knight(M):PL]
'à achever la tâche.'
[COMP complete:INF the task]
'The queen forced the knights to complete the task.'

(5) Les chevaliers ont été forcés à achever la tâche.
[the:PL knight(M):PL have;PRS;3p be:PTCP force:PTCP:M;PL COMP complete:INF the task]
'The knights were forced to complete the task.'

[4] The conditions of use are detailed at <http://creativecommons.org/licenses/by-nc-sa/4.0/>. Source code, test sets as well as grammar evaluation results are available at https://github.com/lfg-french-grammar.

Sentences (1), (2), and (4) exemplify the compound past (*passé composé*). Unlike English and Portuguese, French, in this periphrastic tense, like Italian and German, exhibits a split in the intransitives: unaccusative verbs such as *arriver* 'to arrive' select the auxiliary *être*, while unergative verbs like *danser* 'to dance' select *avoir* (FRANK, 1996). An additional contrast, adding an extra complexity factor to a computational implementation, is the agreement manifested by the past participle (henceforth PTPST). In the case of verbs of the first group of intransitives, PTPST manifests agreement with the subject (see (1)). This agreement pattern, however, is blocked in the verbs of the second group (see (2)). On the other hand, the PTPST of transitive verbs, in constructions with the canonical SVO order, as in (4), is not inflected. Gender and number inflection, however, is mandatory in these verbs in constructions with an anteposed object, as in the relative sentence in (6):

(6) Ils mangent les pêches que la reine a pelées.
[they eat:PRS;3p the:PL peache(F):PL that the queen(F)[SG] have;PRS;3s peel:PTCP:F;PL]
'They eat the peaches the queen has peeled.'

Sentence (3) exemplifies the passive voice in a simple tense (in this case, the present indicative), while (5) exemplifies the passive in the compound past, bringing together the complexities of the two periphrastic constructions.

Sentences (1) and (3) share the same surface form. Like other Romance languages such as Portuguese, the passive participle (henceforth PTPASS) exhibits gender and number inflection in agreement with the subject. This agreement pattern does not occur in languages such as English. There is an apparent analogy of (1) and (3) with adjectival predicative constructions such as (7):

(7) La dame est vaillante.
[the:F;SG lady(F)[SG] be;PRS;3s brave:F;SG]
'The lady is brave.'

In LFG, the passive results from the application of a lexical rule to the entries of verbs that govern a direct object (henceforth OBJ) (KAPLAN; BRESNAN, 1982). This rule models the systematic relations between active and passive verbal forms. Thus, the latter do not need to be listed in the lexicon. This simplifies the encoding of this component and represents a large saving of storage space. The computational implementation of this approach, however, is not trivial. In fact, the changes both in the verbal morphology and in the subcategorization frames as well as the semantic relations between the two diatheses must be accounted for.[5] One complicating factor are object control verbs like *forcer* 'to force'. In these verbs, in the active form, the

[5] Diatheses are regular verbal valence alternations. They comprise both voice phenomena, as in the active-passive opposition, and alternations not expressed by verbal voice (BUSSMANN, 2002).

subject of the infinitive is controlled by the OBJ of the main sentence (see (4)). In the passive variant, however, the controller becomes the subject of the main sentence (see (5)). One of the main advantages of XLE is to provide an efficient mechanism for implementing lexical rules. The use of this mechanism in a given grammar, however, requires the fulfillment of two conditions: (i) formulating appropriate constraints to account for the grammatical examples, while preventing ungrammatical constructions that violate these constraints; (ii) implementing a morphological analyzer. In this article, we show how FrGramm satisfies these two requirements.

Because of the challenges they pose, these two periphrastic verbal constructions are very interesting from the point of view of the development of computational grammars. This is one of the reasons why we chose them as the focus of the present article, which presents FrGramm for the first time to the English-speaking public. We will show how FrGramm implements these phenomena in order to correctly analyze examples such as (1)-(5) and the analogous construction in (7), while not generating ungrammatical examples.

An implementation of these periphrases is also relevant from a theoretical point of view, given the discrepant approaches they have been subject of in LFG. What are the computational properties of each competing proposal? This article represents a contribution to this line of research, since it implements one of these approaches in XLE. As FrGramm is freely available, competing approaches could be more easily implemented in the same system using FrGramm as a basis and compared in complexity in terms of the computational resources of time or space consumed (PRATT-HARTMANN, 2010).

Before concluding this introduction, let us see the main points of divergence in the analysis of the constructions (1)-(5) and (7) in the recent LFG literature. Patejuk and Przepiórkowski (2014), for example, argue that in Polish passive constructions, analogously to adjectival predicative constructions, the verb *być* 'to be' is a raising verb, whose sole semantic argument they suggest is an XCOMP.[6] In LFG, this is a grammatical function with an open argument position, to be filled via functional control (BRESNAN, 2001). Thus, in sentences like (3) or (7), the sentential subject realizes a semantic argument not of the copula but of the XCOMP. According to this approach, (3) and (7) have a bipredicational structure: the first predication is expressed by the copula, the second one by the XCOMP.

This approach, however, is not consensual, as can be seen in the ParGramBank, a parallel treebank of 10 languages, generated by LFG/XLE grammars (SULGER et al., 2013). In this corpus, the analyses of examples from languages such as Norwegian, English, German, and Polish diverge in terms of the status assigned to the VFLEX and the PTCP, on the one hand, and to the adjective, on the other. The question regarding the adjective is whether or not this category instantiates an XCOMP. In the analyses

[6] The term **raising verb** is used in the LFG literature following the tradition of transformational generative grammar. However, in the analysis of these verbs in LFG, there is no constituent movement, given the non-transformational character of this theory.

of Polish examples, the PTCP, similarly to the adjective in constructions like (7), functions as the head of an AP. This AP, in turn, realizes the XCOMP of the VFLEX. In the analyses of English examples, on the other hand, the PTCP is the main verb of the passive construction, forming a monopredicational structure, whereas the predicative AP realizes the closed grammatical function PREDLINK.

In respect to these two points, three of the first computational grammars of French disagree. Frank (1996), for example, adopts the bipredicational analysis for the passive construction and the compound tenses. In this approach, the adjective predicates realize the ACOMP function, which is an adjectival XCOMP. Schwarze (1998) and Butt *et al.* (1999), in turn, implement a monopredicational analysis, in which the VFLEX is an auxiliary without argument structure. According to Schwarze (1998), the adjective predicate realizes an ACOMP. For Butt et al. (1999), however, it realizes the closed function PREDLINK. FrGramm implements the monopredicational analysis for the verbal constructions (1)-(5). It assigns the XCOMP function to the AP of (7).

The remainder of this paper is divided into four sections. To begin with, we address the theoretical framework and the computational system used to implement FrGramm. Following this, the next section explains the methodology, describing the datasets and procedures used in the implementation of the verbal periphrases. The penultimate section then outlines the general architecture of FrGramm and the role of its different modules, focusing on the modeling of the constraints involved in the periphrastic verbal constructions. This section also presents the grammar testing results. Finally, in the last section, we compare our grammar to previous counterparts and point out directions for further research.

## The LFG generative model and the XLE system

LFG is a branch of generative grammar (BRESNAN, 2001; FALK, 2001). Thanks to their mathematically explicit formalization, natural language grammars elaborated in the LFG formalism are directly implementable on the computer. The computational implementation of grammatical phenomena offers two main advantages over descriptions formulated in a natural language and/or not completely formalized. The first one is the possibility of using it in natural language technology applications, e.g., machine translation, corpora annotation, information extraction, and question answering systems. The most notable example of this latter type of application is the IBM Watson. In 2011, it defeated two human champions on the North-American television quiz show *Jeopardy* (BEST, 2013). Its system is based on deep syntactic parsing by means of a formalism analogous to LFG (MCCORD; MURDOCK; BOGURAEV, 2012). The second advantage is the ability to automatically test the internal consistency and empirical adequacy of the analyses in large datasets, such as lists of grammatical and ungrammatical sentences, treebanks, etc.

One appeal of LFG to the academic community is the free-of-cost availability of XLE. This is a very efficient and friendly grammar development and testing environment, which automatically constructs a parser for a given grammar elaborated in LFG/XLE, an LFG formalism notational variant. An important difference of this system compared to alternatives such as XLFG is the possibility of plugging in a lexical transducer for morphological analysis, which significantly reduces the lexicon encoding effort. Another advantage of XLE is the support for the implementation of automatic generators and translators.

In LFG/XLE, a grammar consists of at least two components, namely the annotated phrase structure rules and the lexicon. The latter may consist of (i) full forms and/or (ii) lemmas. In format (i), there is a lexical entry for each inflected form. In small grammars, this format is easier to implement. However, it is not feasible unless there already exists a full-form lexicon that can be adapted. Format (ii) is the most simple and has the lowest development cost. It requires, however, a morphological component implemented as a lexical transducer, a kind of finite-state automaton that associates inflected forms with lexical representations (BEESLEY; KARTTUNEN, 2003). Later, we will see how the lexical transducer we developed for FrGramm 1.0 greatly simplifies the computational implementation of passive voice and compound past.

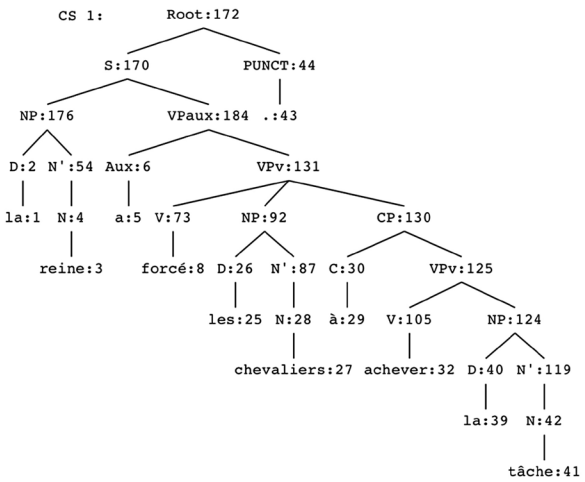
The parser generated by XLE for a given grammar can be automatically applied to an individual construction or to a corpus. For each grammatical construction, the system automatically generates the different syntactic representations that the grammar assigns to it. Figure 1 and Figure 2 show syntactic representations produced by XLE.

Unlike the Chomskyan models, such as Government and Binding Theory and Minimalism, LFG denies the existence in human language of syntactic transformations (BRESNAN, 2001; FALK, 2001). In LFG, phrase structure trees, once generated, do not suffer further modifications. Transformations are only allowed in the lexicon. Thus, in the case of (4), just one tree is generated, as shown in Figure 1. In this model, a constituent structure (henceforth c-structure) maps to a further level of representation, namely functional structure (f-structure). In Figure 2, we have the f-structure corresponding to the c-structure of Figure 1. In Figure 1, *CS 1* in the upper left corner indicates that this is the first c-structure assigned to the sentence by the parser (in this case, there is only one, since the sentence is unambiguous). The mapping from c-structure nodes of Figure 1 into the f-structure of Figure 2 is represented by means of the numerical indices in these structures. For example, the highest node in Figure 1, the root category, which represents the matrix sentence, carries index 172. Node S, which represents the sentence, is node 170, while VPaux (VP with an auxiliary) is represented by index 184. Indices 172, 170, and 184 in Figure 2 designate the f-structure of the entire sentence.

F-structures of phrases like the NPs or the VPaux of Figure 1 result from the unification of the f-structures of their constituents. Unification is the fundamental mathematical operation of LFG and similar models, such as HPSG (MÜLLER, 2016).

This operation collapses the information of two or more f-structures into a single structure, provided that the values of the different attributes do not conflict (FRANCEZ; WINTNER, 2012, p. 85).

**Figure 1** – C-structure of (4) generated by XLE from FrGramm 1.0

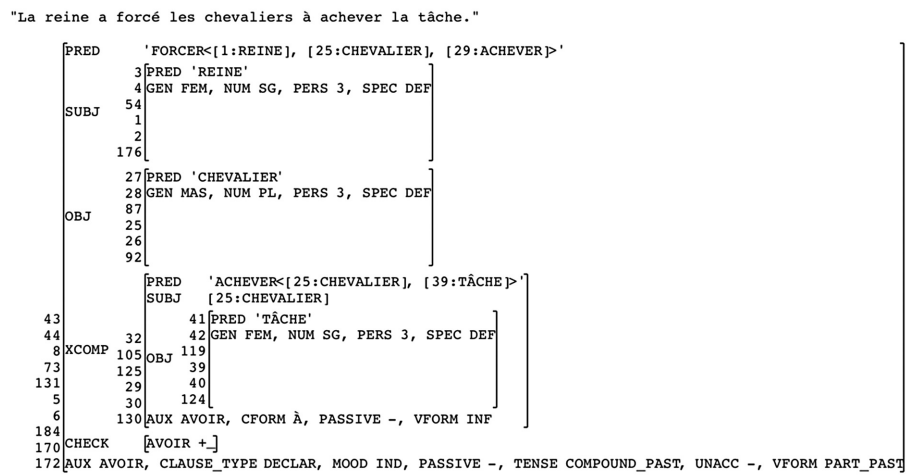



**Source:** Author's elaboration.

F-structures are attribute-value matrices (AVMs). They formalize the notion of feature, recurrent in several linguistic theories. In this context, a feature is an attribute (such as GEN 'gender' and NUM 'number' in Figure 2) with a value (FEM 'female', SG 'singular', etc.). For example, according to Figure 2, *la reine* has the features GEN=FEM, NUM=SG, PERS=3, and SPEC=DEF. The latter two respectively specify the grammatical person and the specification of the noun phrase, which, in this case, is definite. The f-structure of the sentence has, among others, the features CLAUSE_TYPE=DECLARE, MOOD=IND, and PASSIVE=-, indicating that it is a declarative, indicative, and active sentence.

Attributes of the NUM or PERS type of Figure 2 have atomic values, which can be of three types: (i) a string, as in NUM = SG, (ii) a natural number, as in PERS = 3, or (iii) a truth value ("+" or "-"), as in PASSIVE = -. In addition, attributes may have non-atomic values. The descriptive power of AVMs as a formalism for the description of natural language structures stems precisely from the possibility of one attribute having another AVM as its value. Thus, this formalism can account for the recursion of syntactic structures in natural languages. Examples of attributes with a complex value in Figure 2 are the grammatical functions SUBJ (subject), OBJ, and XCOMP.

**Figure 2** – F-structure of (4) generated by XLE from FrGramm 1.0




**Source:** Author's elaboration.

F-structures not only encode the grammatical properties of sentences, but also constitute input to semantic processing (MÜLLER, 2016). The semantic contribution of each individual lexical head to the construction of the sentence meaning is represented by the PRED attribute. Functional heads (determiners, auxiliaries, etc.) do not have a PRED attribute, since their contribution to the sentence's f-structure is merely grammatical. In the case of avalent lexical heads, the PRED value, called semantic form, is conventionally represented by the lemma enclosed in single quotation marks, for example PRED = 'REINE'. In the case of valence-bearing lexical heads, as the verb *forcer* in (4), the semantic form is called lexical form (FALK, 2001, p. 13). It specifies the valence in angle brackets. Thus, the lexical entry for an active form of this verb contains equation (8):

(8) PRED = 'FORCER <(^ SUBJ) (^ OBJ) (^ XCOMP)>'

Formulas like (8) are called functional schemata. These schemata not only form part of lexical entries, but are also used as annotations in the phrase structure rules to constrain the mapping between c-structure and f-structure. In (8), the verb *forcer* is a predicate with three arguments, to be realized, respectively, by the f-structures of the SUBJ, OBJ, and XCOMP. The realization of the arguments of a predicate is governed by two principles of well-formedness. The Completeness Condition determines that all arguments are realized, whereas the Coherence Condition excludes governable grammatical functions that are not specified in the predicate's valence.

In Figure 2, general formula (8) is instantiated as (9):

(9) PRED = ‘FORCER <[1: REINE], [25: CHEVALIER], [29: ACHEVER]>’

In (9), the empty slots of the predicate in (8) are filled by the f-structures of the SUBJ, OBJ, and XCOMP, referenced respectively by indices 1, 25, and 29. Argument slots in semantic forms can only be filled by f-structures with a PRED attribute. For mnemonic convenience, XLE also inserts, into the argument positions of semantic forms with saturated valence, the orthographic representations of the predicates of the grammatical functions that realize these arguments. Thus, in the case of (9), for example, the lemma REINE was inserted into the predicate’s first argument position.

For sentence (3), the parsing algorithm derives formula (9) from (8) by means of the functional annotations.[7] The symbol “^” corresponds to the metavariable “↑” in the traditional LFG notation. This metavariable is instantiated in the f-structure of a constituent by a variable that designates the parent node’s f-structure. In the example in question, “^” refers to the parent node of *forcé*, that is, category V. Because it functions as the head of VPaux, the information associated with V maps into the f-structure of the sentence. Thus, an expression of the form (^ GF) in the lexical form of a verb, where GF designates a grammatical function, is equivalent to ‘GF of the sentence’, for example, (^ OBJ) is equivalent to ‘direct object of the sentence’.

To conclude this section, we deal with lexical rules, which play a fundamental role in the treatment of diatheses in LFG. These rules, along with the remaining formal apparatus of the theory, dispense with the postulation of syntactic transformations. They are equivalent to functions that, applied to lexical entries, generate new entries. In XLE, these operations only manipulate the functional schemata of the entries. They cannot, therefore, manipulate the form of the lexemes, deriving a passive form like *forced* from the suffixation of the active form. XLE, however, makes it possible to overcome this deficiency by plugging in a morphological parser. This solution was adopted by FrGramm 1.0, as we will see later.

For now, we limit ourselves to a simplified formalization of the passive rule in French-type languages. The task of this rule is to model the systematic relationship between the main verbs of examples like (3) and (10), on the one hand, and (11), on the other. The main facts to be modeled are the following: (i) every passive participle corresponds to an active form governing an OBJ; (ii) the OBJ of the active form is realized as SUBJ in the passive form; (iii) the active SUBJ is realized optionally as an oblique in the passive form (SCHWARZE; ALENCAR, 2016, p. 149).

(10) La fée est annoncée par le chevalier.
[the:F;SG fairy(F)[SG] be;PRS;3s announce:PTCP:F;SG by the knight]
‘The fairy is announced by the knight.’

[7] A detailed explanation of the LFG parsing algorithm is beyond the scope of this paper. See, for example, Bresnan (2001, p. 56-60).

(11) Le chevalier annonce la fée.
[the:M;SG knight(M)[SG] announce:PRS;3s the fairy.]
'The knight announces the fairy.'

| | | | | |
|---|---|---|---|---|
| (12) | (i) | active form: | 'ANNONCER < | (^ SUBJ) (^ OBJ)>' |
| | (ii) | 1st passive form: | 'ANNONCER < | (^ OBL) (^ SUBJ)>' |
| | (iii) | 2nd passive form: | 'ANNONCER < | NULL (^ SUBJ)>' |
| | (iv) | thematic grid: | | AGENT THEME |

These generalizations are summarized in (12) (SCHWARZE; ALENCAR, 2016). In (12) (ii), the grammatical function OBL, in languages like French, is a prepositional verbal complement that cannot be pronominalised by a dative clitic. It thus differs from the OBJ2 function (indirect object or secondary object), which licenses this cliticization type. In this case, OBL expresses the passive agent. In (12) (iii), NULL represents the non-realization of this argument. Level (iv) is modeled in LFG as argument structure (a-structure), playing an important role in the theory's architecture (FALK, 2001, p. 105 *et passim*). This structure level, however, is not implemented in XLE.

Considering only the properties of (12), passive can be modeled as an operation that manipulates the grammatical functions of the active form's lexical entry, deriving, through the transformations of (13), two lexical entries for the passive participle. The first entry underlies examples as (10), the second one, examples as (3).

(13) {SUBJ → OBL | SUBJ → NULL}
OBJ → SUBJ

In the first line of (13) we have a logical disjunction, expressed by the "|" connector. The first part of the rule comprises two alternatives. By the first disjunct, the SUBJ is converted into OBL; by the second one, the SUBJ is converted into NULL, resulting in its deletion. The second line of the rule encodes the transformation of the OBJ into SUBJ.

## Data and procedures

This section discusses the two datasets used in the implementation of passive and compound past in FrGramm 1.0. The positive test set defines the range of grammatical phenomena to be modeled. The negative test set allows us to verify if the constraints that characterize the phenomena in question were correctly implemented in a way to avoid overgeneration. The section also explains the notions of fragment and spiral development.

As we have seen, LFG is a mathematically explicit model. From this it follows that the modeling of a grammatical phenomenon must be restricted to a language fragment, i.e., a definite set of sentences. Working with fragments is a common practice in computational syntax (FRANCEZ; WINTNER, 2012).

Closely related to this practice is the adoption of a spiral development. According to this software design technique, a simpler version of a program (a prototype), which covers only part of the problem that the software aims to solve, is first developed. Then, in successive stages, this prototype is progressively expanded, in order to account for more and more facets of the problem (ZELLE, 2004). The application of this technique to the elaboration of a grammar consists of starting with the implementation of a reduced fragment. The coverage of this initial prototype is then expanded through the implementation of successively more complex fragments.

In order to be tested on the computer, a computational language model must constitute a fragment of grammar capable of analyzing constructions that exemplify the different facets of the phenomenon in question. This implies implementing other phenomena present in these constructions. For example, a grammar fragment capable of parsing passive sentences must also deal with agreement, word order, prepositional phrase structure, etc.

LFG conceives a grammatical phenomenon as a series of constraints that define a set of grammatical constructions as opposed to a set of ungrammatical constructions. This approach has two immediate consequences for the computational implementation of an analysis. The first is that it must be tested against two test sets: the positive test set, with the grammatical sentences, and the negative test set, with constructions that violate the postulated constraints. The second consequence is that the implementation should cover superficially analogous but fundamentally different constructions in terms of constraints, as for example in (1), (3), and (7).

In these examples, we have the same surface structure, which we can schematize as *SUBJ est X* 'SUBJ is X', where X is a constituent that agrees in gender and number with the SUBJ. This schema corresponds, however, to three distinct constructions. Example (7) is an adjectival predicative construction. Sentence (1) exemplifies the compound past, while (3) is a passive sentence. What constraints characterize the passive, distinguishing it from the other two constructions? It is evident that only a joint implementation of the three constructions allows for establishing the sets of constraints that distinguish them from each other.

The positive test set only contains constructed sentences. The reason not to use, in the construction of a grammar fragment, examples extracted from real texts is to avoid a series of difficulties. First, in order to test the fragment in real examples, it would be necessary to implement a broad lexicon. In the initial development phase of a grammar fragment, this would mean diverting efforts from the complex task of formally modeling and computationally implementing the syntax. Second, real examples of a particular phenomenon usually instantiate syntactic complexities that do not relate specifically to this phenomenon, as in the passive construction occurrence in (14).

(14) […] *cette date ou l'indication de l'endroit où elle se trouve est annoncée par la mention " à utiliser de préférence avant fin …" ou par le symbole d'un sablier*.[8]
[This date or the indication of the place where it is situated is indicated by the words "to be used preferably before the end of..." or by the symbol of an hourglass.]

Given the complexity of the passive and the compound past in French, we restricted ourselves in implementing these phenomena in FrGramm 1.0 to the grammar fragment exemplified in (1)-(5), (7), (10), (11), and (15)-(21). The fragment thus also includes the adjectival predicative construction.

(15) La reine prie la dame de chanter dans
[the queen asks the lady COMP sing:INF in
les anciens châteaux blancs habités par des fées.']
[the:PL old:M;PL castles(M):PL white:M;PL inhabit:PTCP:M;PL by ART;INDF;PL fairies]
'The queen asks the lady to sing in the old white castles inhabited by fairies.'

(16) Les dames ont été priées de danser.
[the:PL ladies(F):PL have;PRS;3p be:PTCP ask:PTCP:F;PL COMP dance:INF]
'The ladies were asked to dance.'

(17) Le chevalier a été forcé
[the:M;SG knight(M):SG have;PRS;3s be:PTCP force:PTCP:M;SG]
[à inviter les dames à danser.]
[COMP invite:INF the ladies COMP dance:INF]
'The knight was forced to invite the ladies to dance.'

(18) La reine a ordonné aux chevaliers de danser.
[the:F;SG queen(F):SG have;PRS;3s order:PTCP to;the;PL knight(M):PL COMP dance:INF]
'The queen ordered the knights to dance.'

(19) La dame a été aimable.
[ the:F;SG lady(F):SG have;PRS;3s be:PTCP kind:SG]
'The lady was kind.'

(20) La fée demande à être invitée à danser.
[the:F;SG fairy(F):SG ask:PRS;3s COMP be:INF invite:PTCP:F;SG COMP dance:INF]
'The fairy asks to be invited to dance.'

(21) Les chevaliers ont été forcés
[the:PL knight(M):PL have;PRS;3p be:PTCP force:PTCP:M;PL]
par la reine à achever la tâche.
[by the queen COMP complete:INF the task]
'The knights were forced by the queen to complete the task.'

[8] This example was extracted on 01/25/2016 via Google from <http://ansm.sante.fr/Activites/Surveillance-du-marche-des-produits-cosmetiques/Periode-apres-ouverture-PAO/(offset )/1>.

In (22), we list the range of phenomena modeled in FrGramm 1.0. that are directly related to the passive and the compound past. Agreement of the PTPST with the OBJ was not included in this version (see (6)).

(22) (i) active voice and passive voice of different valence classes, including OBJ control verbs such as *forcer* 'to force', *prier* 'to ask', etc.
(ii) simple tenses and compound past in the active
(iii) passive in present and compound past
(iv) passive participle as main verb and as adnominal adjunct
(v) nominal and verbal agreement
(vi) auxiliary selection in the compound past

The positive test set that served as the basis for the implementation of FrGramm 1.0 consists of 157 grammatical sentences. The negative test set, which contains 279 ungrammatical sentences, was manually constructed from the first one by means of systematically transforming grammatical sentences into ungrammatical ones. For example, from (1), (16), (18), and (19), ungrammatical sentences such as (23)-(33) were generated by violating one or more of the constraints related to agreement, verbal form, auxiliary selection, passivization, etc.

(23) *La fée a arrivé.
[the:F;SG fairy(F):SG have;PRS;3s arrive:PTCP]
(24) *La fée a arrivée.
[the:F;SG fairy(F):SG have;PRS;3s arrive:PTCP:F;SG]
(25) *La fée est arrivé.
[the:F;SG fairy(F):SG be;PRS;3s arrive:PTCP]
(26) *La fée arrivée est.
[the:F;SG fairy(F):SG arrive:PTCP:F;SG be;PRS;3s]
(27) *La fée est est arrivée.
[the:F;SG fairy(F):SG be;PRS;3s be;PRS;3s arrive:PTCP:F;SG]
(28) *La fée est arriver.
[the:F;SG fairy(F):SG be;PRS;3s arrive:INF]
(29) *La dame été a aimable.
[the:F;SG lady(F):SG be:PTCP have;PRS;3s kind:SG]
(30) *La reine a ordonnée aux chevaliers de danser.
[the:F;SG queen(F):SG have;PRS;3s order:PTCP:F;SG to;the;PL knight(M):PL COMP dance:INF]
(31) *Les chevaliers sont ordonnés de danser.
[the:PL knight(M):PL be;PRS;3p order:PTCP:M;PL COMP dance:INF]

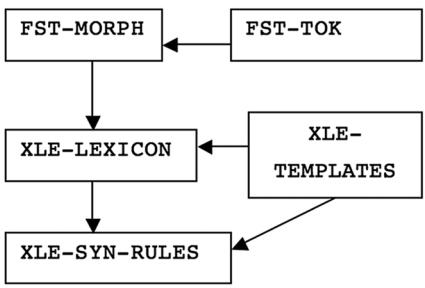
(32) *La reine est ordonnée aux chevaliers de danser.
[the:F;SG queen(F):SG be;PRS;3s order:PTCP:F;SG to;the;PL knight(M):PL COMP dance:INF]

(33) *Les dames été ont priées de danser.
[the:PL ladies(F):PL be:PTCP have;PRS;3p ask:PTCP:F;PL COMP dance:INF]

**Aspects of the implementation**

FrGramm 1.0 implements a standard architecture for LFG/XLE grammars (BUTT et al., 1999; KING, 2004; CROUCH et al., 2011). As schematized in Figure 3, it is made up of five modules: (i) FST-TOK, a tokenizer; (ii) FST-MORPH, a morphological analyzer; (iii) XLE-LEXICON, a set of lexical entries; (iv) XLE-TEMPLATES, a set of templates, analogous to parameterized macros of certain programming languages; (v) XLE-SYN-RULES, a set of context-free rules annotated with functional schemata. The modules with the FST prefix are finite-state transducers implemented in XFST (BEESLEY; KARTTUNEN, 2003). The XLE prefix indicates the implementation of the component in LFG/XLE.

**Figure 3** – FrGramm 1.0's architecture




**Source:** Author's elaboration.

From these components, XLE compiles a parser, which can be applied to the analysis of whole sentences or individual phrases. For each grammatical construction (according to the underlying grammar, FrGramm in the case at hand), XLE generates the respective set of c-structures. A sentence treated as structurally ambiguous by FrGramm, as (15), is assigned more than one c-structure. For this example, two c-structures are generated by XLE, compare (34a) and (34b). Each valid c-structure, in turn, is mapped to one or more f-structures, representing the different readings of the sentence in functional terms.

(34) a. La reine [$_{VP}$ [$_{V}$ prie] [$_{NP}$ la dame] [$_{CP}$ de chanter] [$_{PP}$ dans les … châteaux …]].
b. La reine [$_{VP}$ [$_{V}$ prie] [$_{NP}$ la dame] [$_{CP}$ [$_{C}$ de] [$_{VP}$[$_{V}$ chanter] [$_{PP}$ dans les … châteaux …]]]].

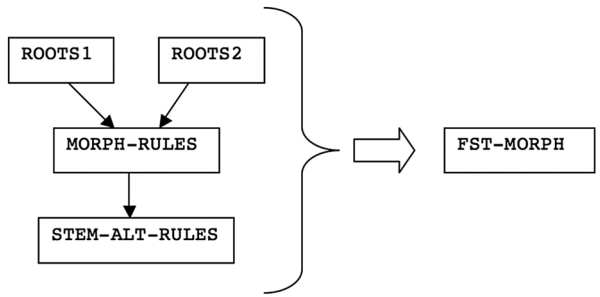
Let us detail each of the components of Figure 3, starting with the most basic, the FST-TOK tokenizer. The function of this module is to segment a string, given as parser input, into a sequence of tokens, i.e., words and punctuation marks. The tokens are delimited by the "@" symbol, as in the tokenization of sentence (11) in (35). In addition, it performs another important task in the pre-processing of sentences, which is normalization (PALMER, 2010). This task consists of converting the variant forms of a token into a standard form, as in example (35), where determiner *Le* 'the' is lowercased.

(35) le@chevalier@annonce@la@fée@.@

The morphological component FST-MORPH is a lexical transducer (BEESLEY; KARTTUNEN, 2003). In the current version of FrGramm, this analyzer is restricted to 39 verbs of the 1st conjugation, totaling 1794 forms. For example, for the inflected form *annonce*, which instantiates the 1st and 3rd person of the present indicative and subjunctive of *annoncer* 'to announce', the transducer produces the four analyses in (36). These representations consist of the lemma followed by a sequence of labels that respectively encode lexical category (V=verb), tense and mood (PRS=present indicative, SBJP= present subjunctive), and person and number (SG = singular).

(36) annoncer+V+SBJP+3+SG
annoncer+V+SBJP+1+SG
annoncer+V+PRS+3+SG
annoncer+V+PRS+1+SG

**Figure 4** – Architecture of FrGramm 1.0's morphological component




**Source:** Author's elaboration.

The morphological analyzer FST-MORPH was implemented according to the architecture outlined in Figure 4. The four components at the left-hand side were compiled into transducers and combined through finite state operations to produce the right-hand side component. The ROOTS1 and ROOTS2 modules contain stems.[9] The former consists of stems of regular verbs such as *regarder* 'to see' that do not suffer alterations in conjugation, while the latter consists of stems of verbs such as *annoncer* that undergo some kind of systematic orthographic change. In the case of *annoncer*, final <c> is replaced by <ç> before a posterior vowel, as in *annonçons* 'we announce'. Four other types of systematic stem alterations in the 1st verbal conjugation were handled. The verbs *acheter* 'to buy', *répéter* 'to repeat', *jeter* 'to throw', and *demander* 'to ask' exemplify these four types.[10]

The MORPH-RULES component is a grammar in the LEXC formalism (BEESLEY; KARTTUNEN, 2003). This grammar models the concatenation of stems and verbal inflections. It is compiled into a transducer that encodes a relation *(p, w)*, where *p* is a lexical representation of the type of (36) and *w*, an intermediate inflected form such as *mang^ons* from the paradigm of *manger* 'to eat'. The last component of the morphology is STEM-ALT-RULES. It consists of rewriting rules that model the orthographic alternations of the five subclasses of verbs referred to above. These rules apply to intermediate forms such as *mang^ons*, deriving final forms like *mangeons* 'we eat'.

Since our focus is the implementation of periphrastic verbal constructions whose kernel is a participle, let us see how the morphological analyzer handles this category. In (37), we transcribe part of a XFST command-line session. By means of the `load` command, we load the analyzer (stored in the `fst-morph` binary file) and then, by means of the `up` command, we apply it to the analysis of some French participles.

(37)
```
xfst[0]: load fst-morph
Opening input file 'fst-morph'
June 04, 2015 14:43:26 GMT
Closing input file 'fst-morph'
xfst[1]: up arrivé
arriver+V+PTPST+UNERG
arriver+V+PTPST+UNACC+M+SG
arriver+V+PTPASS+M+SG
xfst[1]: up dansé
danser+V+PTPST+UNERG
danser+V+PTPST+UNACC+M+SG
danser+V+PTPASS+M+SG
xfst[1]: up dansée
danser+V+PTPST+UNACC+F+SG
danser+V+PTPASS+F+SG
```

[9] The current version of the morphological component of FrGramm does not handle derivational morphology, so the ROOTS1 and ROOTS2 components contain only verbal roots.

[10] Because of lack of space, we cannot elaborate on this aspect. The construction of the morphological analyzer will be a subject of future work.

In FST-MORPH, French participles are classified according to Table 1. In this classification, the first division is between active participles (PTPST) and passive participles (PTPASS). The second criterion is agreement, which restricts itself to the former subcategory. While PTPASS always agrees with its SUBJ, PTPST does so only with verbs selecting the *être* auxiliary, such as *arriver*. Typically, they are unaccusatives (UNACC), while intransitives that select *avoir* (as *danser*) are unergatives (UNERG).[11]

**Table 1** – Classification of participles in FST-MORPH

| **DIATHESIS** | **AGREEMENT** | |
|---|---|---|
| | **YES** | **NO** |
| **ACTIVE** | PTPST+UNACC | PTPST+UNERG |
| **PASSIVE** | PTPASS | ∅ |

**Source:** Author's elaboration.

The examples in (37) show that FST-MORPH overgenerates. In fact, for each verb, the analyzer constructs all three participles, regardless of their valential properties. For example, for non-transitive verbs such as *danser*, *arriver*, and *ordonner*, passive participles are generated.[12] On the other hand, forms like *dansé* and *arrivé* are characterized by the analyzer as ambiguous between PTPST+UNERG and PTPST+UNACC, although only the former and the latter are, respectively, valid.

What is the reason for this overgeneration and what are its consequences for syntactic parsing with FrGramm 1.0? The overgeneration stems from a design choice about the grammar. Of course, it would have been possible to restrict the generation of the three types of participles on the basis of the two syntactic properties at play, namely verbal valence (i.e., governing an OBJ) and auxiliary selection. In fact, finite-state morphology provides a means of elegantly expressing these constraints.[13] Given the architecture of the grammar as shown in Figure 3, however, encoding valence classes in morphology would lead to redundancy in the grammar, since in LFG/XLE this information needs to be encoded in the semantic forms of the verbs in the respective entries in the lexical component, as we saw in (8). On the other hand, the fact that the morphology overgenerates does not necessarily imply that the syntax does so. This can be avoided by means of appropriate constraints in the syntax that filter out ungrammatical forms from the morphology. We will see later that FrGramm implements these constraints, preventing the generation of the negative test set constructions of the type exemplified in (23)-(32).

[11] The distinction between unaccusatives and unergatives does not exactly correspond to the distinction between verbs that select *être* and verbs that select *avoir* in the compound past. There are several important exceptions (SCHWARZE, ALENCAR, 2016, p. 160).

[12] Verbs usually have more than one valence. Non-passivizable are the variants of the verbs in question without an OBJ in their subcategorization frame.

[13] These two constraints can be encoded in the LEXC grammar, for example, by means of flag diacritics (BEESLEY; KARTTUNEN, 2003).

The XLE-LEXICON module has three types of lexical entries. The first type are full-form entries, which encode the morphosyntactic properties of items not handled by the morphology. As we have seen, in FrGramm 1.0, only 1st conjugation verbs are encoded in the lexical transducer. In this way, the remaining items are encoded as full-form entries. In (38), we have the entry for the form *est*, 3rd person singular indicative of the full verb and auxiliary *être*.

(38)
i. est V * (^ PRED)='ÊTRE<(^ XCOMP)>(^ SUBJ)'
ii. (^ SUBJ)=(^ XCOMP SUBJ) @(CAT (^ XCOMP) {AP PP})
iii. (^ TENSE)=PRES @IND @(V-AGR 3 SG);
iv. Aux * { (^ CHECK PASS) = +_ (^ PASSIVE) =c + (^ TENSE)=PRES |
v. (^ VFORM) =c PART_PAST (^ AUX) =c ÊTRE (^ CHECK ETRE) = +_
vi. (^ UNACC) = + (^ TENSE)=COMPOUND_PAST }
vii. @(V-AGR 3 SG) @IND.

In LFG/XLE, lexical entries for homonymous items such as *être* follow the general scheme in (39):

(39) *FORM CATEGORY1 SEPARATOR (FUNCTIONAL SCHEMATA); CATEGORY2 SEPARATOR (FUNCTIONAL SCHEMATA).*

In this scheme, the expressions in uppercase and italic represent the different types of constituent elements of a lexical entry. In the case of (38), the form is *est*, the categories are V (verb) and Aux (auxiliary), and the separator is "*". These three elements are obligatory. The functional schemata are enclosed in parentheses to indicate that they are optional.

In (38), three uses of *être* are encoded. In (i)-(iii), we have the variant that functions as copula in the adjectival predicative construction, which we analyze as a raising verb. Line (i) specifies the valence, as the value of the PRED attribute. This is a verb that requires a SUBJ and an XCOMP. Notice that the SUBJ in (38) is outside the angle brackets. This indicates that it is a grammatical function subcategorized by the verb, but does not realize a semantic argument of the verbal predicate. As we have seen, the XCOMP function represents a class of verbal complements with an open argument position to be filled via control by another grammatical function of the same predicate. Line (ii) initially characterizes this variant as a subject control verb. Next, XLE's CAT predicate determines that XCOMP is realized as AP or PP.[14] Line (iii) specifies the inflectional features: tense, mood, and agreement. In this line, we have the invocation of two templates, defined in the XLE-TEMPLATES module (Figure 3). The first is

[14] Due to lack of space, we cannot explain all the details of the XLE notation here; instead, we refer to Crouch et al. (2011).

the IND template, which assigns IND (indicative) to the MOOD attribute. Next, the invocation of the V-AGR template with arguments 3 and SG establishes 3rd person singular agreement.

In lines (iv)-(vi), in a logical disjunction enclosed by braces, we have the second and third variants, that is, respectively, the passive auxiliary (line (iv)) and the compound past auxiliary (lines (v)-(vi)). The invocation of the V-AGR and IND templates in line (vii) is outside the disjunction because these properties are common to the two auxiliaries. As functional categories, both do not have a PRED attribute, which, as we have seen, encodes the semantic information of lexical categories. In this way, the auxiliaries only contribute morphosyntactic features to the sentence's f-structure. The equations with the CHECK attribute in (iv) and (v) avoid overgeneration in examples of the type of (27), where ungrammatical repetition of an auxiliary occurs. This repetition is licensed by the recursive character of the VP structuring rules (see below). The CHECK attribute was proposed by King (2004) only to ensure syntactic well-formedness. It contributes nothing to the description of the grammatical properties of a sentence. The equation with the PASSIVE attribute restricts the use of this auxiliary to passive structures. The last equation of the passive auxiliary specifies that the verb tense is the present.

The compound past auxiliary requires the past participle of a verb that selects the *être* auxiliary (line (v)). Equation (^ UNACC)=+ in (vi) forces agreement of the participle with the sentential subject, in examples like (1). The second equation specifies that the verbal tense is the compound past (SCHWARZE, 2001, p. 5).

Analogously to (38), (40) encodes the 3rd person singular of the present tense of *avoir*, both in the full verb and auxiliary usage. Differently from the compound past *être* auxiliary, specified with the feature (^ UNACC)=+ in (38), the *avoir* auxiliary is unspecified for the UNACC attribute. The reason for this is that the participle, in this case, may or may not manifest agreement, depending on the type of structure (cf. (4) and (6)).

```
(40)  a V * (^ PRED)='AVOIR<(^ SUBJ)(^ OBJ)>'
          (^ TENSE)=PRES      @(V-AGR 3 SG)        @IND;
      Aux * (^ AUX) =c AVOIR (^ VFORM) =c          PART_PAST
        (^ TENSE)=COMPOUND_PAST
          (^ CHECK AVOIR) = +_      @(V-AGR 3 SG)          @IND.
```

The second type of entries in the XLE-LEXICON module are morphology tags. The analyses generated by the lexical transducer are not directly interpretable by XLE. It is necessary to translate these representations into functional schemata. In (41), we reproduce the entries of this type that relate directly to the implementation of the passive and the compound past. The first two entries invoke the PPAST and PASS templates, which are defined in (42). The definition of the PPAST template, in turn, invokes the ACT template, defined in (43). The lexicon includes entries for all tags produced by the

transducer, allowing, for example, +F and +SG to be mapped to the features GEN=FEM and NUM=SG, respectively.

(41) +PTPST V_SFX XLE @PPAST.
+PTPASS V_SFX XLE @PASS.
+UNACC V_SFX XLE (^ UNACC) =c +.
+UNERG V_SFX XLE (^ UNACC) = -.
(42) PASS = (^ PASSIVE) =c +.
PPAST = (^ VFORM)=PART_PAST @ACT.
(43) ACT = (^ PASSIVE) = -

In entries such as (41), the separator is not the asterisk "*", reserved for full forms as in (38), but the keyword "XLE". The tags +PTPST, +PTPASS, etc., generated by the morphological analyzer, are treated as verbal suffixes by XLE. This is why the category of these elements in (41) is V_SFX (verbal suffix). The functional equations assigned to these suffixes are inherited by the verbs that incorporate them. Thus, the passive participle requires a positive value for the PASSIVE attribute, whereas the past participle is specified as VFORM=PART_PAST and PASSIVE=-. The two types of past participle, in turn, are differentiated by the value of the UNACC attribute. If a positive value is required (3rd line of (41)), agreement must be made; if a negative value is set (4th line of (41)), agreement is blocked. The information thus assigned to the three types of participle, in interaction with the auxiliary entries (see (38) and (40)), the passive lexical rule, and the annotated phrase-structure rules allow the grammar to correctly analyze the positive test set sentences and recognize as ungrammatical the negative test set constructions of the type of (23)-(32).[15]

The third type of entry in the XLE-LEXICON component is devoted to the lemmas of the inflected forms from the morphological component. Examples are shown in (44)-(46). These entries underlie the verbal variants of (1), (2), and (18), respectively.

(44) arriver V XLE @(UNACC_V ARRIVER).
(45) danser V XLE @(UNERG_V DANSER).
(46) ordonner V XLE @(DIRECTIVE ORDONNER OBJ2 DE).

This type of entry encodes such lemma properties that are not encoded by the morphological tags. In the case of FrGramm, these additional properties are the lexical form of the verb (which includes valence), auxiliary selection, and past participle agreement, among others. The invocation of templates such as those in Table 2 allows this information to be specified in a very compact way. Each of these templates encodes the properties that are common to all members of the class. On the other hand, the specific

[15] As we will see later, the current version of FrGramm does not model the linear precedence of the compound past auxiliary relative to the passive auxiliary, analyzing examples of the (33) type, in which the order of these auxiliaries is inverted, as grammatical.

properties of a particular member of the class are specified by means of parameters. For example, in (44) and (45), the UNACC_V and UNERG_V templates are invoked with only one argument, which is the verb's lemma. By contrast, the DIRECTIVE template in (46) is invoked with three arguments: the lemma (i.e., ORDONNER), the controlling grammatical function (i.e., OBJ2) and the complementizer form (i.e., DE).

**Table 2** – Examples of valence-class templates in FrGramm 1.0

| **Template** | **Parameters** | **Valence class** | **Compound past auxiliary** | **Participle agreement** |
|---|---|---|---|---|
| UNACC_V | lemma | intransitive unaccusative verbs | ÊTRE | + |
| UNERG_V | lemma | intransitive unergative verbs[16] | AVOIR | - |
| TRANS | lemma | transitive verbs | AVOIR | |
| DIRECTIVE | lemma, controller, complementizer form | directive verbs | AVOIR | |

**Source:** Author's elaboration.

Passivization is an important lexical property. How does FrGramm specify which verbs are passivizable? The entries in (47) answer this question.

(47) peler V XLE @(PASSIVE @(TRANS PELER)).
forcer V XLE @(PASSIVE @(DIRECTIVE FORCER OBJ À)).

Following the standard implementation of passivization in LFG/XLE (KING, 2004), these entries invoke the PASSIVE template, which has a single argument: the invocation of a valence-class template (Table 2). It is therefore the application of one operation to the output of another. Let us exemplify this process. The application of the TRANS template to its argument generates the functional schemata that are proper of transitive verbs. Applied to these schemata, the PASSIVE template performs the transformations of (13), generating, in interaction with the information encoded in the verbal suffix entries (see (41)), active and passive lexical entries.

In (48) we have FrGramm's PASSIVE template definition. It is an adaptation of an analogous template proposed by King (2004) in her English LFG/XLE grammar.

(48) PASSIVE(_SCHEMATA) =
{ _SCHEMATA (^ PASSIVE) = - |
_SCHEMATA (^ PASSIVE) = + (^ OBJ) --> (^ SUBJ)
{ (^ SUBJ) --> NULL | (^ SUBJ) --> (^ OBL) (^ OBL CASE) =c PAR}}.

[16] In the context of FrGramm, the *unaccusative* and *unergative* labels have a purely mnemonic character, since they refer not to verbal semantics, but to compound past auxiliary selection. See note 11.

This template has as its sole parameter a set of functional schemata (_SCHEMATA variable). The template definition comprises a disjunction: the first alternative refers to the active diathesis, the second, to the passive diathesis. The latter, in turn, comprises another disjunction between two alternatives, depending on the transformation of SUBJ into NULL or into OBL. In this latter case, the value of the OBL's CASE attribute is required to be PAR. Common to the two passive variants is the transformation of OBJ into SUBJ. In the case of OBJ control verbs like *forcer* in (5), this transformation occurs both in the lexical form of the verb and in the functional control equation.

**Figure 5** – F-structure of (5) generated by XLE from FrGramm 1.0

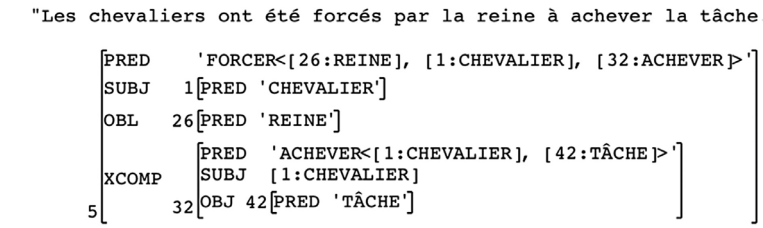
```
"Les chevaliers ont été forcés à achever la tâche."

      [PRED     'FORCER<NULL, [1:CHEVALIER], [26:ACHEVER]>'                    ]
      |        3[PRED 'CHEVALIER'                          ]                   |
      |        4[GEN MAS, NUM PL, PERS 3, SPEC DEF         ]                   |
      |SUBJ   51|                                          |                   |
      |        1|                                          |                   |
      |        2|                                          |                   |
      |      177[                                          ]                   |
   40 |         [PRED    'ACHEVER<[1:CHEVALIER], [36:TÂCHE]>'                 ]|
   41 |         |SUBJ    [1:CHEVALIER]                                        ||
   11 |         |       38[PRED 'TÂCHE'                          ]             ||
   73 |         |       39[GEN FEM, NUM SG, PERS 3, SPEC DEF     ]             ||
  117 |      29 |OBJ   105|                                      |             ||
    8 |XCOMP  91|       36|                                      |             ||
    9 |     111 |       37|                                      |             ||
  119 |      26 |      110[                                      ]             ||
    5 |      27 |                                                              ||
    6 |     116[AUX AVOIR, CFORM À, PASSIVE -, VFORM INF                       ]|
  185 |                                                                        |
  167 |CHECK    [AVOIR +_, ETRE +_, PASS +_]                                   |
  169 [AUX AVOIR, CLAUSE_TYPE DECLAR, MOOD IND, PASSIVE +, TENSE COMPOUND_PAST, VFORM PART_PAST]
```

**Source:** Author's elaboration.

In this respect, compare the f-structures of Figure 2, Figure 5, and Figure 6. In the f-structure of the active sentence as well as in the f-structures of the passive counterparts, the parser inserts the f-structure of *les chevaliers* into the second argument position of FORCER, at the main predication level, and into the first argument position of ACHEVER, at the secondary predication level. On the other hand, the f-structure of *la reine* is inserted into the first argument position of FORCER regardless of the realization of this argument as SUBJ in Figure 2 or as OBL in Figure 6.

As we have seen, in LFG's architecture, the f-structure of a sentence constitutes the input for building its meaning representation. In this respect, passives with *forcer*-type OBJ control verbs and their active counterparts constitute a greater challenge to computational treatment than simpler constructions like (3) and (11). The analyses of Figure 2, Figure 5, and Figure 6 show that FrGramm produces the expected f-structures for these sentences. These structures make it possible to calculate the semantic relations of entailment between (4) and (5) and equivalence between (4) and (21) (CRUSE, 2000, pp. 28-30), by converting the values of the PRED attributes into logical representations.

**Figure 6** – Simplified f-structure of (21) generated by XLE from FrGramm 1.0




**Source:** Author's elaboration.

Let us now turn to the question of morphological overgeneration. In the syntax, the proposed constraints act as a filter of the overgenerated forms. For example, the morphological analysis of *arrivé* as a non-inflected past participle (i.e., *arriver+V+PTPST+UNERG*) is blocked in the syntax because, on the one hand, the suffix +UNERG is mapped to UNACC=- (see (41)). On the other hand, the lexical entry of *arriver* in (44) projects UNACC=+. However, these two specifications are incompatible, because the values of the UNACC attribute, being different, do not unify.

In (48), instead of equation (^ PASSIVE)=c +, which requires a positive value for the PASSIVE attribute, as King (2004) proposes, we have equation (^ PASSIVE)=+, which defines this value as positive. This definition satisfies the requirement imposed by the passive participles generated in the morphology (see (42)). Let us explain, by means of the ungrammatical construction (32), how the passive template filters out passive participles of non-transitive verbs, such as *ordonner* in the (46) variant. For the *ordonnée* form, the morphological analyzer generates the representations in (49) and (50). Through the application of the DIRECTIVE template (Table 2) in the entry in (46), the AUX attribute of this verb receives the value AVOIR, which excludes the first representation, because the +UNACC tag requires an auxiliary marked with UNACC=+ (see (41)). According to (38), the compound past auxiliary is the only variant of *être* with this specification, however, it requires a verb with AUX=ÊTRE. The analysis in (50), in turn, is excluded because +PTPASS requires PASSIVE=+ (see (41)). But the only way for a verb to receive this feature is through the passive template in (48), which, according to (46), is not applied to the verb in question.

(49) ordonner+V+PTPST+UNACC+F+SG
(50) ordonner+V+PTPASS+F+SG

The last module of the architecture of Figure 3 is XLE-SYN-RULES. It is made up of annotated phrase structure rules. We limit ourselves here to the verbal phrase. Following Butt et al. (1999) and King (2004), but differing from Schwarze (1998) and Schwarze and Alencar (2016), we distinguish, based on the type of head, between VPv and VPaux, as formalized in (51). In this definition, VP is a metacategory, an

XLE feature that allows for both expressing linguistic generalizations and simplifying phrase structure rules and c-structures, since this type of category does not project a node in c-structure. Figure 1 exemplifies the two types of VP. While VPv is headed only by V, according to (53), VPaux, defined in (52), is co-headed by Aux and V.[17] The motivation for this distinction is to exclude examples such as (26), where the main verb erroneously precedes the auxiliary. However, in other rules, as in rule (52) itself, VPv and VPaux are interchangeable, a fact that is captured through the VP metacategory.

(51) VP = { VPv | VPaux}.
(52) VPaux --> Aux VP.

Example (53) transcribes part of the VPv rule, "[...]" indicates the suppressed information. This verb phrase expands into a V, optionally followed by a disjunction whose members represent the different possibilities of verbal complementation (not exhaustively listed here), referred to by the metacategories IO, DO, OBL-PP, and IC, defined in (54).

(53) VPv --> V { DO IO | DO | IO | IO OBL-PP [...] | OBL-PP | IC DO | IC [...] }#0#1 [...].
(54) IO = PP: (^ OBJ2)=! (! CASE)=c À.
DO = NP: (^ OBJ)=!.
OBL-PP = PP: (^ OBL)=!.
IC = VP: (^ XCOMP)=!.

The definitions in (54) consist of phrase categories annotated with functional schemata that specify the type of grammatical function of each category, namely OBJ2, OBJ, OBL, and XCOMP, respectively. In the case of the IO metacategory, it is required that the CASE attribute have the value À. The VP metacategory also occurs on the right side of (53), to account for the infinitival complements of control verbs. Given the recursive character of this expansion, fairly complex constructions with several embedded complements and with more than one auxiliary, such as (17) or (20), can be analyzed by the grammar.

We conclude this section with the evaluation of FrGramm. Applied to the positive test set, the parser generated by XLE assigned all grammatical sentences the expected c-structures and f-structures. Only 8 sentences received two analyses, due to the attachment ambiguity of a locative PP, as exemplified in (34). The application of the parser to the negative test set, by contrast, revealed the need for adjustments in the XLE-SYN-RULES module in the next version of the grammar. In fact, two of the 279 sentences of this set were classified as grammatical by the parser. One of these is (33), the other is a structurally analogous example. In these examples, the order of the

[17] On the notion of co-head in LFG, see Falk (2001, p. 39). In Bresnan's theory (2001, p. 132), functional co-heads are **extended heads** of a lexical category.

*avoir* and *être* auxiliaries is reversed; compare (33) with the grammatical construction in (16). This shows that FrGramm 1.0 overgenerates in this respect, not modeling the precedence relation between these two auxiliaries, since *avoir* must precede *être* when both function as auxiliary of a given main verb.

## Final remarks

In this article, we have described the treatment of passive and compound past in FrGramm 1.0, a computational grammar of French of medium syntactic coverage that we have recently implemented in LFG/XLE. Due to the duplicity of auxiliaries and the agreement of the participle, the compound past presents greater complexity in French than in languages such as Portuguese and English. On the other hand, the analysis of these constructions as well as of the adjectival predicative construction, superficially analogous to the passive, has been the object of controversies in LFG theory.

FrGramm is the only French grammar of this size implemented in LFG that is accessible on-line in unrestricted form, under a license allowing the redistribution of modifications. Thus, it constitutes a basis for testing the computational properties of the different theoretical approaches to these constructions in the LFG/XLE formalism and can also be adapted to other systems.

Frank (1996) and Schwarze (1998) are the two previous grammar fragments of French directly comparable to our approach because they are documented in a sufficiently detailed way that a reimplementation in XLE is possible. How does FrGramm stand in relation to these two proposals? For one thing, FrGramm has much wider coverage than the fragment of Schwarze (1998), which does not include the compound past, nor does it explain whether the passive rule, in its morphological or lexical dimension, was actually implemented.

The fragment of Frank (1996) is much broader than the grammar fragment modeled in FrGramm. While our fragment is restricted to declarative sentences with constituents in their canonical order, Frank's (1996) includes interrogative sentences, relative sentences, and several other constructions with displacement of constituents. This allows the grammar of Frank (1996) to handle agreement of the PTPST with the OBJ. Following the spiral development technique, this phenomenon was not included in the first version of FrGramm, given the complexity of the treatment of these constructions.

An important shortcoming of Frank's (1996) grammar is not to integrate a morphological analyzer, confining itself to a full-form lexicon. In this way, each passive participle is individually encoded in the lexicon, by means of a specific template for each valence class. In this approach, consequently, there is not a unique passive rule.

FrGramm fills these gaps in Frank's approach (1996). It incorporates a lexical transducer that analyzes the forms of a group of verbs of the 1st conjugation. This

allows integration between morphology and lexicon in the computational modeling of the passive rule. A single passivization rule handles all passivizable valence classes. Several valence classes were implemented, including copulas and object control verbs. The latter class represents an extra difficulty for the computational treatment, since the controller, in passive, becomes the subject. FrGramm handles the morphological, syntactic, and semantic aspects of the passive as a productive lexical process, producing suitable f-structures for both simple constructions and control structures.

Given these characteristics, FrGramm, in spite of lower syntactic coverage, is superior to Frank's (1996) proposal in modularity and scalability. FrGramm has a unique passive rule, valid for all valence classes, while Frank's approach assumes a separate rule for each class. This difference is decisive when expanding the lexicon. In FrGramm, to account for sentences with a verb like *avancer* 'to advance', for example, it suffices to include the *avanc-* root in the ROOTS1 morphology component and to specify the valence frames in the lexicon of lemmas, as exemplified in (44)-(47). The inclusion of new lexical items in the grammar of Frank (1996) is much more laborious.

The evaluation of FrGramm produced quite satisfactory results. The parser assigned the expected structures to the 157 grammatical sentences of the positive test set, which includes examples in the active and passive voice both in the present indicative and in the compound past. Of the 279 ungrammatical sentences of the negative test set, only two were not analyzed correctly. The grammar assigns them a valid f-structure. These two sentences are in the compound past passive voice. The reason for this deficiency is that the current version of FrGramm does not model the linear precedence relationship between *avoir* and *être* auxiliaries in this type of example. This problem will be remedied in the next version of the grammar. Besides, its coverage will be extended to account for PTPST agreement with the OBJ.

With this, we hope to contribute to the debate within LFG regarding the predicational structure of the two periphrastic verbal constructions, analyzed in this article as mono-predicational. In fact, this new version of the grammar can be adapted to implement the bipredicational analysis, enabling one to compare the computational complexity of the two competing approaches in XLE.

**Acknowledgments**

We would like to express our gratitude to Valeria de Paiva for intermediating with John Maxwell and Daniel Bobrow, both from Xerox's Palo Alto Research Center (PARC), so that we could obtain a free non-commercial XLE license. Our special thanks are extended to this company as well as all researchers involved in the creation of this software. We are also indebted to Christoph Schwarze, Jessé de Sousa Mourão, and the anonymous reviewers for helpful comments and suggestions on earlier drafts of this paper, although any remaining errors are our own.

ALENCAR, L. Uma implementação computacional de construções verbais perifrásticas em francês. **Alfa**, São Paulo, v.61, n.2, p.437-466, 2017.

- *RESUMO: Este artigo descreve o tratamento da passiva e do passado composto na FrGramm, uma gramática computacional do francês implementada na Gramática Léxico-Funcional (LFG) usando o software XLE. Devido à dualidade de auxiliares e concordância do particípio passado (PTPST), a segunda perífrase exibe uma maior complexidade estrutural em francês do que em línguas como inglês e português, representando, consequentemente, um maior desafio à implementação computacional. Uma dificuldade adicional é a modelação das regularidades morfológicas e sintático-semânticas da passiva. A FrGramm resolve esse problema por meio de uma regra lexical produtiva. Também implementa as restrições que governam a formação das duas perífrases verbais, exceto a concordância do PTPST com o objeto direto. A implementação foi avaliada pela aplicação de um analisador sintático automático (parser) a 157 sentenças gramaticais e 279 construções agramaticais. Todas as sentenças do primeiro conjunto foram analisadas corretamente. Apenas duas construções do segundo que violam a precedência do auxiliar do passado composto sobre o da passiva foram analisadas como gramaticais. A FrGramm é a única gramática LFG do francês com essa cobertura atualmente disponibilizada livremente. Uma versão futura dará conta da concordância do PTPST com o objeto direto e evitará a hipergeração referida.*

- *PALAVRAS-CHAVE: Linguística computacional. Análise sintática automática profunda. Gramática léxico-funcional. LFG/XLE. Morfologia de estados finitos. Perífrases verbais em francês. Voz passiva.*

## REFERENCES

BEESLEY, K. R.; KARTTUNEN, L. **Finite state morphology**. Stanford: CSLI, 2003.

BEST, J. IBM Watson: the inside story of how the Jeopardy-winning supercomputer was born, and what it wants to do next. **TechRepublic**, 2013. Retrieved from: <https://www.techrepublic.com/article/ibm-watson-the-inside-story-of-how-the-jeopardy-winning-supercomputer-was-born-and-what-it-wants-to-do-next/>. Access on: 16 Apr. 2016.

BOULLIER, P.; SAGOT, B.; CLÉMENT, L. Un analyseur LFG efficace pour le français: SxLfg. In: TRAITEMENT AUTOMATIQUE DES LANGUES NATURELLES, 12., 2005. **Actes**... Dourdan, 2005. p.403-408. Retrieved from: < http://www.atala.org/taln_archives/TALN/TALN-2005/taln-2005-court-004>. Access on: 9 Feb. 2016.

BRESNAN, J. **Lexical-functional syntax**. Malden: Blackwell, 2001.

BUSSMANN, H. (Ed.). **Lexikon der Sprachwissenschaft**. 3.ed. Stuttgart: Kröner, 2002.

BUTT, M. et al. **A grammar writer's cookbook**. Stanford: CSLI, 1999.

CLÉMENT, L. **XLFG**. Bordeaux: University Bordeaux, 2014. Retrieved from: <http://www.xlfg.org/>. Access on: 22 Feb. 2016.

CLÉMENT, L.; KINYON, A. XLFG – an LFG Parsing Scheme for French. INTERNATIONAL LEXICAL-FUNCTIONAL GRAMMAR CONFERENCE, 6., 2001. **Proceedings**... Stanford: CSLI, 2001. p.47-65.

CROUCH, D. et al. **XLE documentation**. Palo Alto: Palo Alto Research Center, 2011. Retrieved from: <http://www2.parc.com/isl/groups/nltt/xle/doc/xle_toc.html>. Access on: 22 Feb. 2016.

CRUSE, D. A. **Meaning in language**: an introduction to semantics and pragmatics. Oxford: Oxford University Press, 2000.

FALK, Y. N. **Lexical-functional grammar**: an introduction to parallel constraint-based syntax. Stanford: CSLI, 2001.

FRANCEZ, N.; WINTNER, S. **Unification grammars**. Cambridge: CUP, 2012.

FRANK, A. Eine LFG-Grammatik des Französischen. In: BERMAN, J.; FRANK, A. **Deutsche und französische Syntax im Formalismus der LFG**. Tübingen: Niemeyer, 1996. p. 97-244.

KAPLAN, R. M.; BRESNAN, J. Lexical-Functional Grammar: a formal system for grammatical representation. In: BRESNAN, J. (Ed.). **The mental representation of grammatical relations**. Cambridge: MIT Press, 1982. p.173-281.

KING, T. H. **Starting a ParGram grammar**. 2004. Retrieved from: <http://www2.parc.com/isl/groups/nltt/xle/doc/PargramStarterGrammar/starternotes.html>. Access on: 10 Nov. 2012.

MCCORD, M. C.; MURDOCK, J. W.; BOGURAEV, B. K. Deep parsing in Watson. **IBM Journal of Research and Development**, Armonk, v. 56, n. 3/4, p. 1-15, 2012.

MÜLLER, S. **Grammatical theory**: from transformational grammar to constraint-based approaches. Berlin: Language Science Press, 2016. Retrieved from: <http://langsci-press.org/catalog/book/25> Access on: 22 Mar. 2016.

PALMER, D. D. Text preprocessing. In: INDURKHYA, N.; DAMERAU, F. J. (Ed.). **Handbook of natural language processing**. 2.ed. Boca Raton, Florida: Chapman & Hall/CRC, 2010. p. 9-30.

PATEJUK, A.; PRZEPIÓRKOWSKI, A. In favour of the raising analysis of passivisation. INTERNATIONAL LEXICAL-FUNCTIONAL GRAMMAR CONFERENCE, 19., 2014. **Proceedings**... Stanford: CSLI, 2014. p. 461-481.

PRATT-HARTMANN, I. Computational complexity in natural language. In: CLARK, A; FOX, C.; LAPPIN, S. (Ed.). **The handbook of computational linguistics and natural language processing**. Malden: Wiley & Blackwell, 2010. p.43-73.

SAGOT, B. **Page web de Benoît Sagot – équipe Alpage (INRIA/Paris 7)**: SxLFG. Paris: Université Paris Diderot, [2015?]. Retrieved from: <http://alpage.inria.fr/~sagot/sxlfg.html>. Access on: 22 Feb. 2016.

SCHWARZE, C. Do sentences have tense? In: INTERNATIONAL LEXICAL-FUNCTIONAL GRAMMAR CONFERENCE, 6., 2001. **Proceedings**... Stanford: CSLI, 2001. p. 449-463.

SCHWARZE, C. **Lexikalisch-funktionale Grammatik**: eine Einführung in 10 Lektionen mit französischen Beispielen. 2.ed. Konstanz: Fachgruppe Sprachwissenschaft der Universität Konstanz, 1998.

SCHWARZE, C.; ALENCAR, L. F. de. **Lexikalisch-funktionale Grammatik**: eine Einführung am Beispiel des Französischen mit computerlinguistischer Implementierung. Tübingen: Stauffenburg, 2016.

SULGER, S. et al. ParGramBank: the ParGram paralell treebank. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, 51., 2013. **Proceedings**... Sofia: Association for Computational Linguistics, 2013. p. 550-560.

ZELLE, J. M. **Python programming:** an introduction to computer science. Wilsonville: Franklin, Beedle & Associates, 2004.

ZWEIGENBAUM, P. Un analyseur syntaxique pour grammaires lexicales-fonctionnelles. **T.A. Informations**, Paris, v. 32, n. 2, p. 19-34, 1991. Retrieved from: <https://perso.limsi.fr/pz/FTPapiers/ZweigenbaumTAI91.pdf>. Access on: 10 Feb. 2016.

Received in april 2016

Approved in january 2017