Mairal-Usón, Ricardo and Cortés-Rodríguez, Francisco (2017) Automatically Representing TExt Meaning via an Interlingua-based System (ARTEMIS). A further step towards the computational representation of RRG. Journal of Computer-Assisted Linguistic Research 1: 61-87.

Automatically Representing TExt Meaning via an Interlingua-based System (ARTEMIS). A further step towards the computational representation of RRG

Ricardo Mairal-Usón*[1], Francisco Cortés-Rodríguez[2]
[1]Universidad Nacional de Educación a Distancia, Spain
[2]Universidad de La Laguna, Spain
*Corresponding author: rmairal@flog.uned.es

Received: 9 March 2017 / Accepted: 25 April 2017 / Published: 19 June 2017

Abstract
Within the framework of FUNK Lab – a virtual laboratory for natural language processing inspired on a functionally-oriented linguistic theory like Role and Reference Grammar-, a number of computational resources have been built dealing with different aspects of language and with an application in different scientific domains, i.e. terminology, lexicography, sentiment analysis, document classification, text analysis, data mining etc. One of these resources is ARTEMIS (Automatically Representing TExt Meaning via an Interlingua-Based System), which departs from the pioneering work of Periñán-Pascual (2013) and Periñán-Pascual & Arcas (2014). This computational tool is a proof of concept prototype which allows the automatic generation of a conceptual logical structure (CLS) (cf. Mairal-Usón, Periñán-Pascual and Pérez 2012; Van Valin and Mairal-Usón 2014), that is, a fully specified semantic representation of an input text on the basis of a reduced sample of sentences. The primary aim of this paper is to develop the syntactic rules that form part of the computational grammar for the representation of simple clauses in English. More specifically, this work focuses on the format of those syntactic rules that account for the upper levels of the RRG Layered Structure of the Clause (LSC), that is, the *core* (and the level-1 construction associated with it), the *clause* and the *sentence* (Van Valin 2005). In essence, this analysis, together with that in Cortés-Rodríguez and Mairal-Usón (2016), offers an almost complete description of the computational grammar behind the LSC for simple clauses.

Keywords: Role and Reference Grammar, computational grammars, the layered structure of the clause, syntactic rules, lexical rules, constructional rules, attribute-value matrix, conceptual modeling.

DOI: https://doi.org/10.4995/jclr.2017.7788
PUBLISHER: UNIVERSITAT POLITÈCNICA DE VALÈNCIA.

1. INTRODUCTION

Our latest research has been mainly concerned with the development of a natural language processing laboratory using tools developed within the framework of Role and Reference Grammar (RRG), a functional linguistic paradigm with a strong typological adequacy (Van Valin 2005; 2008; Mairal-Usón et al. 2012; etc.) [1]. As a result, a number of computational resources have been built with a focus on different domains:

1. Navigator: this tool allows the user to retrieve data from the lexical entries in the English Lexicon (e.g. morphosyntactic, pragmatic and collocational information) and from the conceptual entries in the Core Ontology (e.g. thematic frame, meaning postulate, etc), as developed within the framework of the FunGramKB Project. This resource allows the user to nativage through the linguistic (the English lexicon) and the conceptual levels (the ontology).
2. **R**ec**O**gnizi**N**g **D**omains with I**A**TE (RONDA): this tool is used to categorize a text or a collection of documents in different specialized domains as specified in the IATE database.
3. **CA**tegory- and **S**entiment-based **P**roblem Find**ER** (CASPER): this resource analyses micro-texts (e.g. tweets) for the automatic detection of user-defined problems by following a symbolic approach to topic categorization and sentiment analysis.
4. **DA**ta **MI**ning **EN**countered (DAMIEN): it is a workbench that allows researchers to do text analytics by integrating corpus-based processing with statistical analysis and machine-learning models for data mining tasks.
5. **D**iscovering and **Ex**tracting **TER**minology (DEXTER): this tool has been developed as an online multilingual workbench which is provided with a suite of tools for (a) the compilation and management of small -and medium- sized corpora, (b) the indexation and retrieval of documents, (c) the elaboration of queries by means of regular expressions, (d) the exploration of the corpus, and (e) the identification and extraction of term candidates (i.e. unigrams, bigrams and trigrams) (Periñán-Pascual 2015).
6. **A**utomatically **R**epresenting **Te**xt **M**eaning via an **I**nterlingua-based **S**ystem (ARTEMIS): this computational resource is currently a proof-of-concept laboratory which allows the automatic generation of a conceptual logical structure (CLS), that is, a fully specified semantic representation of an input text, on the basis of a reduced sample of sentences (cf. Periñán 2013; Cortés-Rodríguez and Mairal-Usón 2016; Fumero Pérez and Díaz Galán 2017; Martín Díaz 2017).

This paper is concerned with this latter resource and is thought to offer a further step in the computational implementation of the RRG Layered Structure of the Clause (LSC). Although the computational adequacy of RRG has not been a major focus, recently there have been some

[1] For further references, see the RRG website: http://www.acsu.buffalo.edu/~rrgpage/rrg.html

papers providing different venues: Guest (2009) worked on the syntactic parser and provides a tool to visualize RRG structures; Nolan and Salem (2011), Salem et al. (2008) developed a first computational translation engine (Arabic-English); Nolan and Periñán (2014) edited a volume containing a number of different proposals on the computational adequacy of RRG. Following this line, within the context of ARTEMIS a number of relevant works have been published: Cortés (2016) deals with the formal representation of RPs; Galán and Fumero (2016) reinterpet *do auxiliary* in terms of computational rules; Cortés-Rodríguez and Mairal-Usón (2016) provide the first formal representation of the RRG LSC by offering a computational representation of the following layers: predicate and nucleus.

Within this context, our primary aim in this paper is to complete the computational representation for the LSC by designing the computational syntactic rules that represent the upper layers in the LSC: the core node, the level-1 construction node[2], the clause and the sentence. Hence, the organization of this papers goes as follows: Section 2 deals with the architecture of the ARTEMIS engine and zooms in on the Grammar Development Environment (GDE), where lexical, constructional and syntactic rules are stored. Sections 3 discusses the methodological repercussions that a computational move like ARTEMIS brings to the actual configuration of RRG and the LSC in particular. Section 4 presents the format of the syntactic rules for the *level-1 construction* node, the core, the clause and the sentence. Finally, section 5 offers a few concluding remarks.

2. ARTEMIS

As part of the FUNK Lab project, Periñán (2014) and Periñán and Arcas (2013) first developed the architecture of ARTEMIS with the primary aim of having a resource with the potential to convert a natural language fragment (a text) into its morphosyntactic form and subsequently to its underlying semantic structure. This semantic output, called CLS, constitutes a major step for a number of different projects dealing with information retrieval, question-answering systems, automatic textual annotation, etc. where access to a comprehensive semantic decription is vital. As a matter of fact, this is a leading topic in today's most challenging research in natural language processing. However, it is also certain that such a linguistically-oriented approach like ARTEMIS is excepctional provided the prominent role of stochastic approaches based on very powerful algorithms which are, in turn, the result of controlled machine learning procedures. Hence, ARTEMIS is a linguistically grounded computational resource and makes use of those linguistic analytical tools which offer robust explanations to complex issues (e.g. the syntax-semantics interface, the non-proposional dimension of meaning -or non-coded meaning-, constructional meaning, cognitive operations, etc). There are two linguistic models which are particularly influential: RRG and the Lexical Constructional Model (LCM), a usage-based, comprehensive theory of meaning construction that aims to give explanations of how all aspects of meaning, including those that go beyond so-called core-grammar (e.g. traditional implicature, illocutionary force, and discourse coherence) interact among one another (Mairal-Usón and Ruiz de Mendoza 2008; see also Ruiz de Mendoza and Mairal-Usón 2008; 2011; Mairal-Usón

[2] Note that this node does not form part of the original proposal of the LSC but was first proposed in Periñán (2013) (cf. below).

and Ruiz de Mendoza 2008; Ruiz de Mendoza and Galera 2014) (cf. section 3 for the specific linguistic tools that ARTEMIS makes use of).

There are three main modules in ARTEMIS: the Grammar Development Environment (GDE), the CLS constructor, and the COREL-Scheme Builder. While the primary focus of the GDE is to provide a morphosyntactic representation of an input text, that is, a parsed tree, the last two modules are concerned with deriving the semantic representation of the same input text. In so doing, these two modules retrieve information from FunGramKB, the multilanguage knowledge base that supports the application, and the reasoner engine which is being developed. Thus, the GDE is a computational grammar which includes those grammatical rules necessary for the morphosyntactic parsing of a text. Here is a UML diagram representing the interaction of the following three modules (cf. Periñán-Pascual & Arcas 2014, 178):

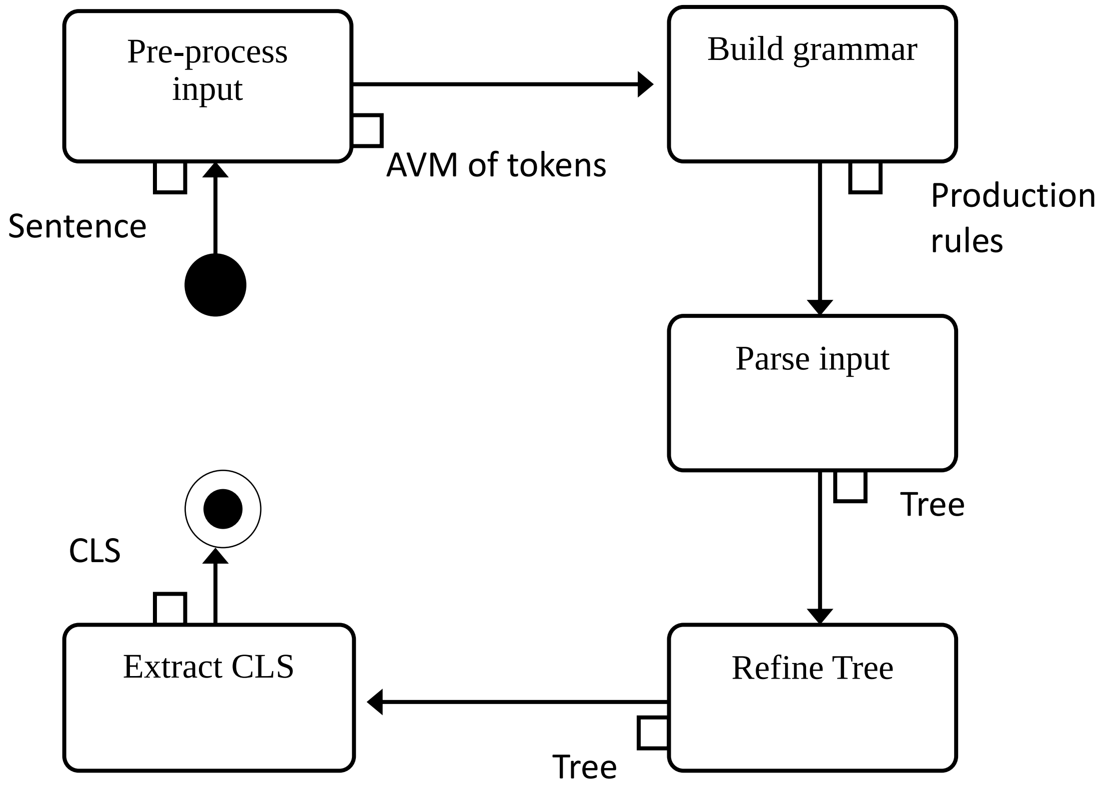

FIGURE 1. THE ARTEMIS ENGINE

The process begins with an initial phase of tokenization which splits the text into sentences and the sentences into word tokens, which are, in turn, stored in terms of Attribute-Value Matrices (AVMs). The output of this first phase serves as input for the Build Grammar Module, the GDE, where syntactic, constructional and lexical rules operate with the aim of generating a morphosyntactic representation. Next, a tree refinement process is intended to relocate, if necessary, some tree nodes and to filter out some node attributes. Finally, the last phase is concerned with the extraction of those semantic properties that are relevant for the generation of the CLS.

The focus of this paper, and the work of a linguist, rests upon the GDE. It is fundamental to design the rules for the computational grammar which will diagnose the morphosyntactic encoding of an input text. In connection with this, there are three types of rules:

1. Lexical rules describe the morphosyntactic properties of function words while content words are assigned the linguistic information represented in the FungramKB lexicon which, as explained in Mairal-Usón et al. (2012), is connected to ontology. Although this is beyond the scope of this paper, it is important to note that each lexical entry is connected to a conceptual unit in the ontology and consequently the information coded in the entry goes beyond those aspects that are grammatically relevant.
2. Constructional rules account for the morphosemantic properties of constructional templates as discussed in Construction Grammar (cf. Goldberg 2006) and the FunGramKB grammaticon (Van Valin and Mairal-Usón 2014; Mairal-Usón and Periñán-Pascual 2016; Mairal-Usón 2017). Recall that the grammaticon stores four types of constructional templates following the distinctions in the LCM: (i) level 1 deals with argument-structure constructions (e.g. *He hammered the metal flat*), (ii) level 2 is concerned with implicational constructions (e.g. *Do I look like I'm happy?*); (iii) level 3 addresses illocutionary configurations (e.g. *Can you pay attention to what I'm saying?*); (iv) and level 4 focuses on discourse constructions (e.g. *He is one of the most attractive, not to mention intelligent, men I know*).
3. Syntactic rules consists of a set of rules that represent the internal constituency of each node in the LSC and a library of AVMs for grammatical units.

While lexical and constructional rules are constructed automatically, syntactic rules are built manually. Given that Cortés-Rodríguez and Mairal-Usón (2016) have already described the syntactic rules for PRED and NUC, we are now concerned with the format of the syntactic rules for the three upper nodes in the LSC: the core (plus the level-1 construction node), the clause and the sentence. In so doing, we will deal with the actual design of the syntactic rules expressing the combinatorial properties of a grammatical category together with the set of AVMs necessary to represent the grammatical features associated to each category.

It is certain that the format of syntactic rules might result too complex and awkard since operators, as conceived in RRG, form part of the GDE and must be accounted for inside a syntactic rule (cf. below). This means that, unlike RRG, ARTEMIS includes only one projection and therefore the GDE consists of feature-based production rules subject to the linearity of constituents, since parsing – following Earley's algorithm - proceeds in a bottom-up fashion complemented with top-down predictions. In connection with this, Periñán-Pascual and Arcas (2014, 182) affirm:

> …the psychologically-plausible behavior of the parser lies in the fact that it is: a. an incremental left-corner parser, where <u>each successive word being encountered is incorporated into a larger structure</u> by combining bottom-up processing with top-down predictions, and b. a parallel parser, since multiple parse structures

can be generated locally, so there is no need to re-analyze the input if one parse structure proves incorrect (i.e. no backtracking)[3].

From this passage a number of important methodological corollaries follow. First, every feature – both grammatical units and nodes in the LSC – are represented in terms of AVMs. Unification is a key operation in the generation of the parsed tree. As a matter of fact, ARTEMIS is an object-oriented paradigm using unification approaches to grammar[4] and feature-oriented structures. The ultimate aim in the morphosyntactic generation of a parsed tree is to satisfy the principle of unification, that is, the structural and semantic constraints encoded in the AVMs must be compatible. This process of feature-unification is a bottom-to-top process running up through the whole structure of the clause. For example, let us consider how the attribute 'illocutionary force', which is present in the first constituent of the nucleus, percolates up to the sentence node.

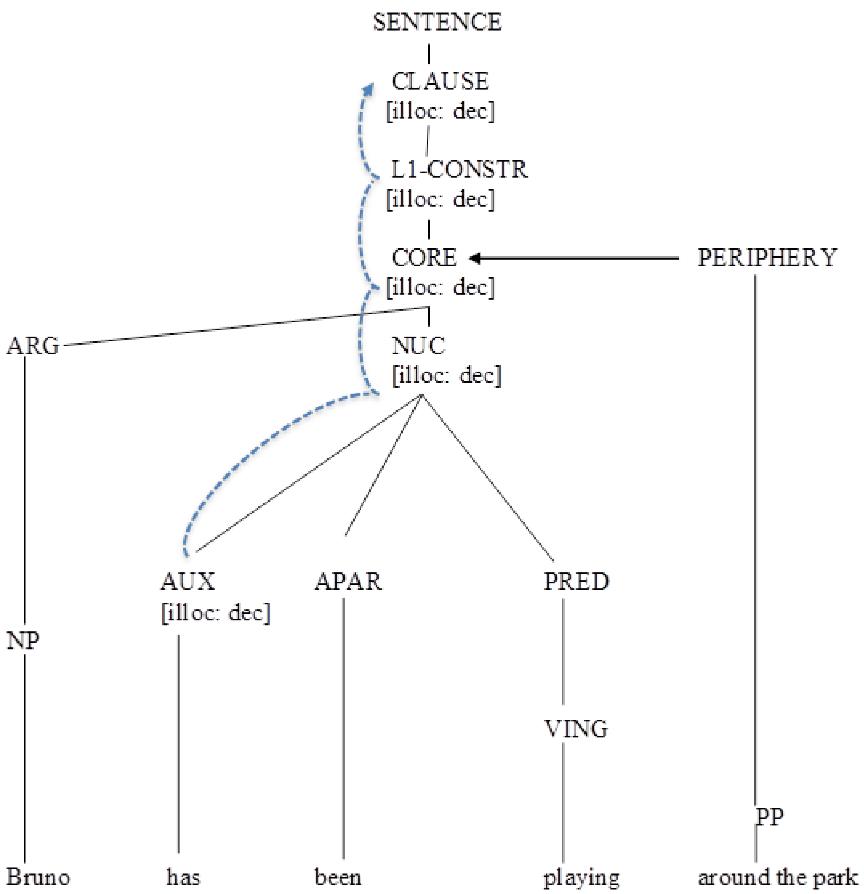

FIGURE 2. FEATURE UNIFICATION PATH OF ILLOCUTIONARY FORCE
(Taken from Cortés-Rodríguez and Mairal-Usón 2016)

[3] Underlining is ours.
[4] It is interesting to note that some of the most recent works on the computational implementation of Construction Grammar, i.e. Sign Based Construction Grammar, follow the same type of unificational approach (Boas and Sag 2012; Sag et al. 2003).

This representation shows how the 'Illoc(utionary force)' Attribute, which appears encoded in the AVMs of the first constituent of NUC (the first auxiliary verb or the lexical predicate if there are no auxiliaries), percolates up to the clause level where unification will finally take place.

In summary, ARTEMIS consists of a set of production rules (cf. section 4) and a number of feature unification operations intented to satisfy the structural and semantic constraints of the items represented in each AVM. As briefly discussed in the following section, this object-oriented paradigm introduces a number of important changes in the linguistic model.

3. Looking at the layered structure of the clause in computational terms

As discussed somewhere else (Periñán 2013; Cortés-Rodríguez and Mairal-Usón 2016; Mairal-Usón and Periñán-Pascual 2016), ARTEMIS has brought a number of significant modifications in the orthodox version of the RRG linking algorithm. We briefly summarize the most significant.

First, the standard RRG lexical representation in terms of *logical structures* has now been replaced by the notion of Conceptual Logical Structure (CLS; cf. Van Valin and Mairal-Usón 2014; Mairal-Usón, Periñán-Pascual and Pérez 2012). Instead of using primitives (state and activity predicates), lexical entires are defined by means of conceptual units that come from the ontology. Hence, we favour an ontological approach to RRG semantics. This ontological formalism is more robust and provides each lexical entry with information which goes beyond those aspects that are grammatically relevant Each lexical entry is linked to a conceptual unit in the ontology and each concept is described in terms of a thematic frame specification of event participants and a meaning postulate, which provides a meaning definition. In so doing, the resulting RRG lexical entries are provided with a further semantic decomposition, which was one of the challenges of RRG semantics (cf. Van Valin and Wilkings 1993; Van Valin and LaPolla 1997; Mairal-Usón and Faber 2002; 2017).

Second, ARTEMIS is neither a projectionist nor a constructivist theory in that both lexical meaning and constructional meaning are fundamental for the semantics-to-syntax interface. Hence, a preliminary distinction between kernel and non-kernel structures is made in order to account for the fact between the unmarked case argument structure of a predicate and that where constructional meaning intervenes. Within kernel structures, there are three types: Kernel-1, Kernel-2 and Kernel-3 representing intransitive, transitive and ditransitive predicates. In contrat, non-kernel structures are stored in the grammaticon. Recall that this latter component houses an array of constructional schemata arranged in four different levels (cf. Mairal-Usón and Ruiz de Mendoza 2008):

1. Level-1 or argument constructions.
2. Level-2 or implicational constructions.
3. Level-3 or illocutionary constructions.
4. Level-4 or discourse constructions.

The assumption that it is not always possible to predict the syntactic structure of a predicate from its argument structure is an argument in favour of recognizing the value of constructional meaning in the computational implementation of RRG. If we assume that constructional meaning is influential in the syntactic expression of a predicate, we will need a new node in the parsed tree that accounts for the occurrence of those arguments which are contributed by the meaning of a given construction. This explains why a new node - the CONSTR-L1 node - in the LSC is proposed:

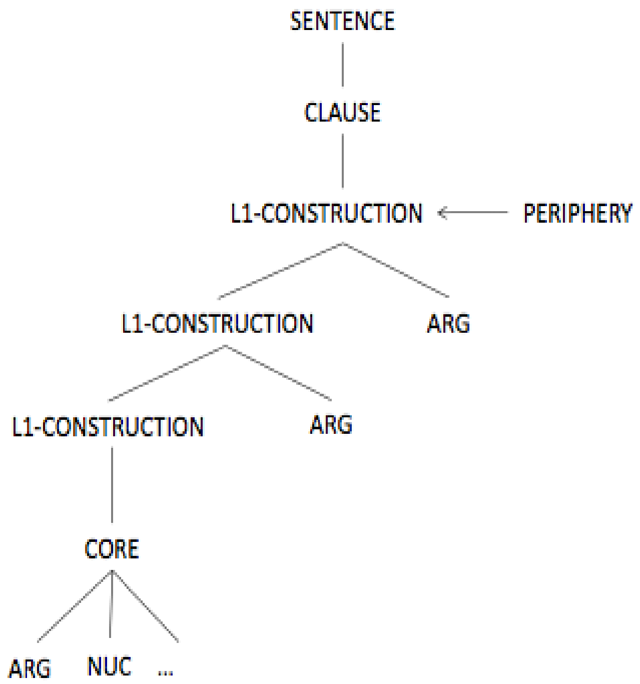

FIGURE 3. A NEW LOOK AT THE LSC
(Taken from Periñán 2013)

Under this L1-Construction node, the argument constructions (AAJs and NUC-S) will be represented (cf. section 3.1.). Let us consider the following example: the predicate *kick* in the unmarked case is a transitive predicate with two arguments in the core (a Kernel-2 structure). However, the argument structure of this predicate might be enriched by the meaning of the resultative construction and, even more, by the caused motion construction (example taken from Mairal-Usón and Periñán-Pascual 2016):

(1) a. John kicked the ball flat out of the stadium.

b. [[[John kicked the ball]$_{\text{Kernel-2}}$ flat]$_{\text{Transitive-Resultative}}$ out of the stadium]$_{\text{Caused-Motion}}$

It is not possible to account for this syntactic realization without resorting to the grammaticon and searching for the linguistic description of both the resultative and the caused motion construction. This means that the AVM for the resultative construction will unify with the lexical entry, the output of which will then unify with the AVM for the caused motion construction. Hence, ARTEMIS needs a label to identify each of these constructs that are functional in the semantic representation of an input text. Therefore, the clause is configured now as one or more argumental constructions (L1-CONSTRUCTION) which are recursively arranged as shown in Figure 3.

Finally, unlike the three level projection schema operating in the RRG LSC, ARTEMIS uses only one projection which means that operators are interpreted as AVMs. As advanced above, ARTEMIS uses a set of production rules together with a number of feature unification operations which guarantee that the semantic and structural constraints encoded in two AVMs are satisfied. This means that operators like tense, aspect, modality, negation, illocutionary force etc. together with function words (e.g. articles, modals, etc.) are represented in terms of AVMs with their corresponding values. This theoretical move provides a different picture of the RRG LSC in that the operator projection is substituted by feature-bearing nodes:

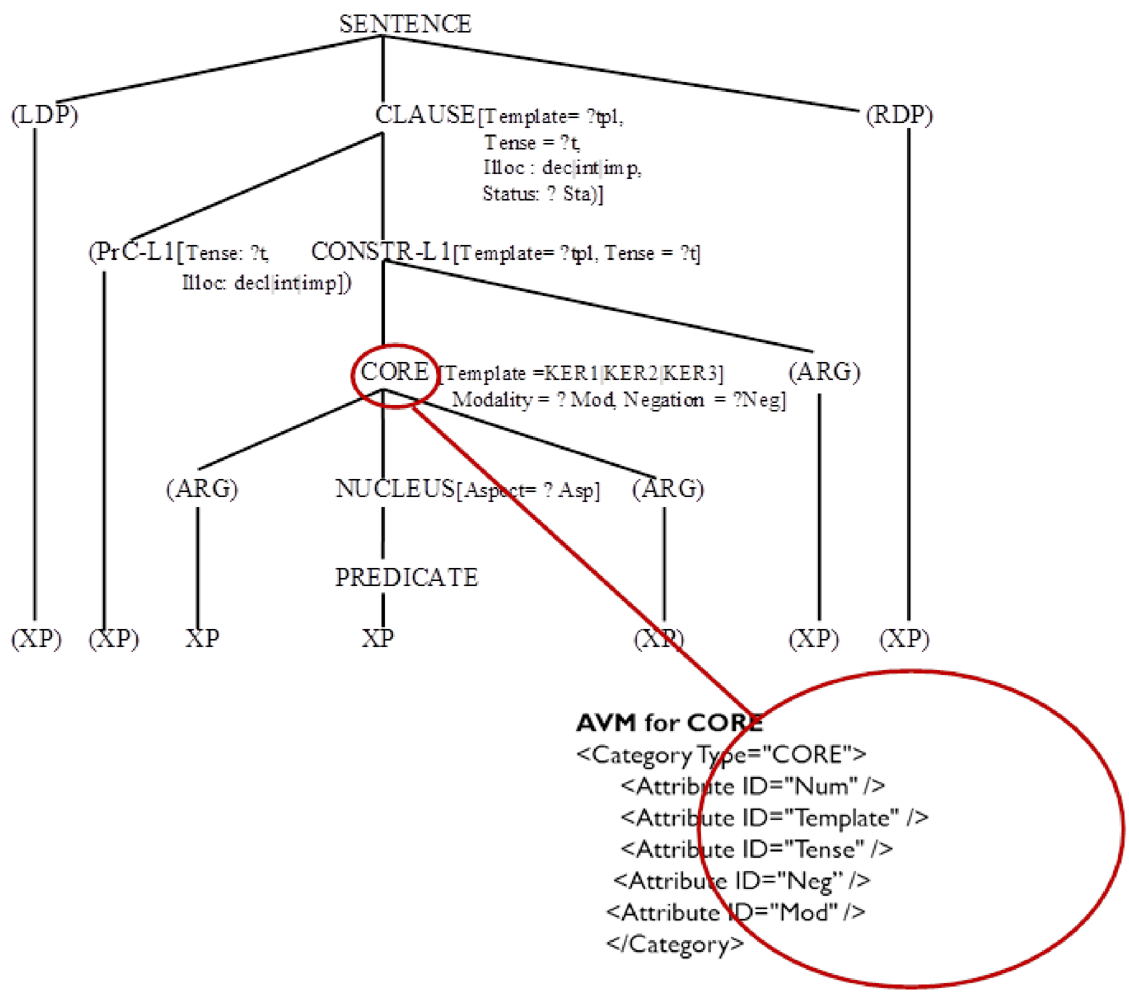

FIGURE 4. AVMs in the LSC
(A partial representation taken from Cortés-Rodríguez and Mairal-Usón 2016)

This representation shows that the node CORE is interpreted as a feature complex including different types of morphosyntactic parameters which are described as attributes. In this specific case, the CORE includes attributes for number ("Num"), Template (type of argument structure), negation, modality (see appendix 3 for the full list of AVMs); this is only a partial description of the AVM corresponding to the CORE. In the next section we offer a complete lisitng of all the attributes encoded at this layer.

3.1. The Core and L1 Construction

ARTEMIS uses two sources to motivate both semantically and structurally the basic semantic structure underlying a sentence as encoded in the CORE:

1. The inventory of kernel constructions, which account for the unmarked argument structure of a predicate, include the following frames: Kernel-1, Kernel-2 and Kernel-3 Constructions (corresponding to intransitive, monotransitive and ditransitive structures, respectively) (cf. Periñán-Pascual, 2014). Note that, in general, kernel constructions would account for the configuration of the CORE in unmarked cases;
2. The inventory of Level 1 constructions (argument structure constructions5) stored in the Grammaticon, and the information about the basic subcategorization frames of predicates, as encoded in their corresponding lexical entries in the Lexica.

Thus the rule for CORE allows for these three possible patterns as discussed in a): intransitive (ARG-NUC), monotransitive (ARG-NUC-ARG) and ditransitive (ARG-NUC-ARG-ARG):

CORE

CORE [concept=?, illoc=?, mod=?, neg=?, num=?, per=?, recip=?, reflex=?, sta: ?, tpl=?, t=?] -> ARG[concept=?, macro= a | u | n, num=?, per=?, phrase=?, role: agent | attribute | goal | instrument | location | manner | origin | referent | result | theme, tpl=?, var= x | y | w | z] NUC[asp: ?, concept: ?, illoc:?, mod: ?, num:?, per:?, recip:?, reflex:?, sta: ?, tpl:?, t: ?] || ARG[concept=?, macro= A | U | n, num=?, per=?, phrase=?, role: agent | attribute | goal | instrument | location | manner | origin | referent | result | theme, tpl=?, var= x | y | w | z] NUC[asp: ?, concept: ?, illoc:?, mod: ?, num:?, per:?, recip:?, reflex:?, sta: ?, tpl:?, t: ?] ARG[concept=?, macro= A | U | n, num=?, per=?, phrase=?, role: agent | attribute | goal | instrument | location | manner | origin | referent | result | theme, tpl=?, var= x | y | w | z] || ARG[concept=?, macro= A | U | n, num=?, per=?, phrase=?, role: agent | attribute | goal | instrument | location | manner | origin | referent | result | theme, tpl=?, var= x | y | w | z] NUC[Asp: ?, concept: ?, illoc:?, mod: ?, num:?, per:?, recip:?, reflex:?, sta: ?, tpl:?, t: ?] ARG[concept=?, macro= A | U | n, num=?, per=?, phrase=?, role: agent | attribute | goal | instrument | location | manner | origin | referent | result | theme, tpl=?, var= x | y | w | z] ARG[concept=?, macro= A | U | n, num=?, per=?, phrase=?, role: agent | attribute | goal | instrument | location | manner | origin | referent | result | theme, tpl=?, var= x | y | w | z]

Despite its apparent complexity, this rule combines only two types of constituents, one Nucleus (NUC) and the Argument(s) (ARG) encoded in the lexical entry of the Predicate of the clause. The AVMs for CORE and ARG are as follows:

```
<Category Type="CORE">
    <Attribute ID="Concept" />
    <Attribute ID="Illoc" />
    <Attribute ID="Mod" />
    <Attribute ID="Neg” />
    <Attribute ID="Num"/>
```

[5] Level 2, 3 and 4 constructions represent the non-coded dimension of meaning, that is, these refer to the implicational, illocutionary and the discursive aspects of meaning. At the present stage of our research, there is a very tentative proposal for the representation of these three levels (see Mairal-Usón and Periñán-Pascual, 2016), although it is fair to note that such a proposal has not been evaluated.

```
        <Attribute ID="Per"/>
        <Attribute ID="Recip" />
        <Attribute ID="Reflex" />
        <Attribute ID="Sta" />
        <Attribute ID="Template"/>
        <Attribute ID="Tense" />
</Category>
<Category Type="ARG">
        <Attribute ID="Concept"/>
        <Attribute ID="Macro"/>
        <Attribute ID="Num"/>
        <Attribute ID="Per"/>
        <Attribute ID="Phrase"/>
        <Attribute ID="Prep" />
        <Attribute ID="Role"/>
        <Attribute ID="Template" />
        <Attribute ID="Var"/>
</Category>
```

The following examples illustrate one of the 3 realizational variants encoded in this rule; the specific values for the attributes activated in this structure are encoded in the information between brackets:

(2) The dog is chasing the cat (Kernel 2, monotransitive): ARG[concept=+DOG_00, macro= A, num= s, per=3, phrase=RP, role: theme, var= x] NUC[asp: prog, concept: +CHASE_00, illoc: dec, num: s, per:3, tpl: k-2, t: pres] ARG[concept=+CAT _00, macro= U, num= s, per=3, phrase=RP, role: theme, var= x].

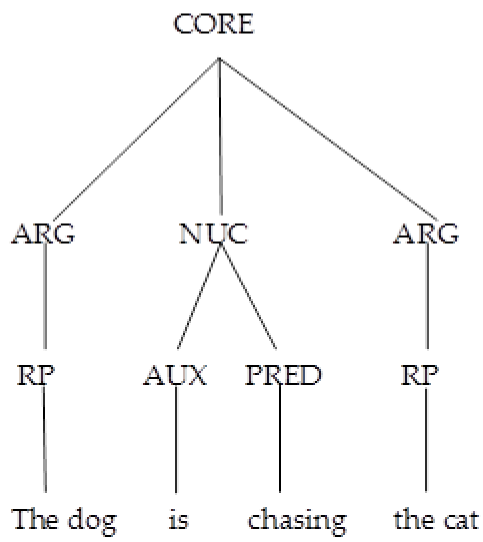

FIGURE 5. CONSTITUENT PROJECTION OF A KERNEL-2 CORE
(A partial representation)

Although Periñán-Pascual and Arcas (2014) contemplate the introduction of a NUC-S in the CORE, we do not think that this should be part of this syntactic rule for the following two reasons:

1. There is no kernel construction that introduces such a node. All kernel constructions respond to ARGs as encoded in the core grammar section of lexical entries. A verb like *elect* will encode three Args in its entry for sentences like *They elected him President*; or *consider* and *find* in *They consider him intelligent* or *I found the novel rather boring*; the MPs *intelligent* and *rather boring* are ARGs with an *attribute* semantic function;
2. NUC-S therefore must be the output of some constructional rules (as happens with resultative constructions).

Then, such non-optional constituents are then contributed by L1-Constructions, and they can be of two types, namely Secondary Nucleii (NUC-S) and Argument-Adjuncts (AAJ). The following rule represents these two combinations:

L1 CONSTRUCTION

CONSTR-L1[Akt=?, concept=?, illoc=?, mod=?, neg=?, sta: ?, tpl=?, t=?, weight= 0 | 1 | 2 | 3 | 4 | 5 | 6] -> CORE[concept=?, illoc=?, mod=?, neg=?, num=?, per=?, recip=?, reflex=?, sta: ?, tpl=?, t=?] || CORE[concept=?, illoc=?, mod=?, neg=?, num=?, per=?, recip=?, reflex=?, sta: ?, tpl=?, t=?] AAJ[concept=?, macro= A | U | n, phrase=?, prep=?, role: agent | attribute | goal | instrument | location | manner | origin | referent | result | theme, tpl=?, var= x | y | w | z] || CORE[concept=?, illoc=?, mod=?, neg=?, num=?, per=?, recip=?, reflex=?, sta: ?, tpl=?, t=?] NUC-S [concept=?, macro= A | U | n, phrase=?, prep=?, role: agent | attribute | goal | instrument | location | manner | origin | referent | result | theme, tpl=?, var= x | y | w | z] || CONSTR-L1[Akt=?, concept=?, illoc=?, mod=?, neg=?, sta: ?, tpl=?, t=?, weight= 0 | 1 | 2 | 3 | 4 | 5 | 6] AAJ[concept=?, macro= A | U | n, phrase=?, prep=?, role: agent | attribute | goal | instrument | location | manner | origin | referent | result | theme, tpl=?, var= x | y | w | z] || CONSTR-

L1[Akt=?, concept=?, illoc=?, mod=?, neg=?, sta: ?, tpl=?, t=?, weight= 0 | 1 | 2 | 3 | 4 | 5 | 6] NUC-S[concept=?, macro= A | U | n, phrase=?, prep=?, role: agent | attribute | goal | instrument | location | manner | origin | referent | result | theme, tpl=?, var= x | y | w | z] || CONSTR-L1[Akt=?, concept=?, illoc=?, mod=?, neg=?, tpl=?, t=?, weight= 0 | 1 | 2 | 3 | 4 | 5 | 6] AAJ[concept=?, macro= A | U | n, phrase=?, prep=?, role= agent | attribute | goal | instrument | location | manner | origin | referent | result | theme, tpl=?, var= w] NUC-S [concept=?, macro= A | U | n, phrase=?, prep=?, role= agent | attribute | goal | instrument | location | manner | origin | referent | result | theme, tpl=?, var= v]

This rule contemplates the cases in which the L1 Grammaticon contributes to the syntactic make up of the clause. There are different realizational variants for the CONSTR-L1 node:

1. The simplest one refers to the cases in which the CONSTR-L1 is saturated only by the CORE:

    CONSTR-L1[Akt=?, concept=?, illoc=?, mod=?, neg=?, sta: ?, tpl=?, t=?, weight= 0 | 1 | 2 | 3 | 4 | 5 | 6] -> CORE[concept=?, illoc=?, mod=?, neg=?, num=?, per=?, recip=?, reflex=?, sta: ?, tpl=?, t=?]

2. The construction adds an AAJ node or a NUC-S to the Kernel structure of the original Predicate; e.g. *The breeze rolled the curtain open* (where *open* is a NUC-S) or *I'll cook a chocolate pudding for my colleagues in the Thermomix* (where *for my colleagues* is a Beneficiary AAJ introduced at this level); the following sections of the CONSTR-L1 rule above encode these possibilities:

    2.1. For AAJs (see analysis in figure 6 below):

    CONSTR-L1[Akt=?, concept=?, illoc=?, mod=?, neg=?, sta: ?, tpl=?, t=?, weight= 0 | 1 | 2 | 3 | 4 | 5 | 6] ->…|| CORE[concept=?, illoc=?, mod=?, neg=?, num=?, per=?, recip=?, reflex=?, sta: ?, tpl=?, t=?] AAJ[concept=?, macro= A | U | n, phrase=?, prep=?, role: agent | attribute | goal | instrument | location | manner | origin | referent | result | theme, tpl=?, var= x | y | w | z]

    2.2. for NUC-S:

    CONSTR-L1[Akt=?, concept=?, illoc=?, mod=?, neg=?, sta: ?, tpl=?, t=?, weight= 0 | 1 | 2 | 3 | 4 | 5 | 6] ->…|| CORE[concept=?, illoc=?, mod=?, neg=?, num=?, per=?, recip=?, reflex=?, sta: ?, tpl=?, t=?] NUC-S [concept=?, macro= A | U | n, phrase=?, prep=?, role: agent | attribute | goal | instrument | location | manner | origin | referent | result | theme, tpl=?, var= x | y | w | z]

3. A bit more complicated are the cases in which there is constructional recursion; i.e. when the AAJ is attached to a CONST-L1 node; that is, when there are at least two constructional patterns overlaying the original kernel structure, as in the example from Mairal-Usón and Periñán-Pascual (2016): *John kicked the ball flat out of the stadium.*

    CONSTR-L1[Akt=?, concept=?, illoc=?, mod=?, neg=?, sta: ?, tpl=?, t=?, weight= 0 | 1 | 2 | 3 | 4 | 5 | 6] ->… CONSTR-L1[Akt=?, concept=?, illoc=?, mod=?, neg=?, sta: ?, tpl=?, t=?, weight= 0 | 1 | 2 | 3 | 4 | 5 | 6] AAJ[concept=?, macro= A | U | n, phrase=?, prep=?, role: agent | attribute | goal | instrument | location | manner | origin | referent | result | theme, tpl=?, var= x | y | w | z] || CONSTR-L1[Akt=?, concept=?, illoc=?, mod=?, neg=?, sta: ?, tpl=?, t=?, weight= 0 | 1 | 2 | 3 | 4 | 5 | 6] NUC-S[concept=?, macro= A | U | n, phrase=?, prep=?, role: agent | attribute | goal | instrument | location | manner | origin | referent | result | theme, tpl=?, var= x | y | w | z]

4. The last set is:

CONSTR-L1[Akt=?, concept=?, illoc=?, mod=?, neg=?, sta: ?, tpl=?, t=?, weight= 0 | 1 | 2 | 3 | 4 | 5 | 6] ->... || CONSTR-L1[Akt=?, concept=?, illoc=?, mod=?, neg=?, tpl=?, t=?, weight= 0 | 1 | 2 | 3 | 4 | 5 | 6] AAJ[concept=?, macro= A | U | n, phrase=?, prep=?, role= agent | attribute | goal | instrument | location | manner | origin | referent | result | theme, tpl=?, var= w] NUC-S [concept=?, macro= A | U | n, phrase=?, prep=?, role= agent | attribute | goal | instrument | location | manner | origin | referent | result | theme, tpl=?, var= v]

and accounts for the cases in which a same pattern introduces simultaneously an AAJ and a NUC-S, as in: *Bruno barks doors*(AAJ) *open*(NUC-S) or *It's OK to let your baby cry himself* (AAJ) *to sleep* (NUC-S).

It is also interesting to consider the set of attributes that should be encoded in the AVM for CONSTR-L1, as we believe that the RRG core operators (Negation and Modality) would now have to 'raise' their scope up to this node. The AVM would also include those attributes that will be unified at the clause level and whose feature path unification started in lower nodes, as happens in the case of illocutionary force, number, status and tense. Therefore, the CONSTR-L1 AVM would have the following format:

<Category Type="CONSTR-L1">

<Attribute ID="Akt" />

<Attribute ID="Concept" />

<Attribute ID="Illoc" />

<Attribute ID="Mod" />

<Attribute ID="Neg" />

<Attribute ID="Sta" />

<Attribute ID="Template" />

<Attribute ID="Tense" />

<Attribute ID="Weight" />

</Category>

We have also devised the AVM for AAJs and completed the one for NUC-S:

<Category Type="AAJ">

<Attribute ID="Concept"/>

<Attribute ID="Macro"/>

<Attribute ID="Num"/>

```
            <Attribute ID="Per"/>
            <Attribute ID="Phrase"/>
            <Attribute ID="Prep" />
            <Attribute ID="Role"/>
            <Attribute ID="Template" />
            <Attribute ID="Var"/>
    </Category>
    <Category Type="NUC-S">
            <Attribute ID="Concept"/>
            <Attribute ID="Macro"/>
            <Attribute ID="Phrase"/>
            <Attribute ID="Prep" />
            <Attribute ID="Role"/>
            <Attribute ID="Template" />
            <Attribute ID="Var"/>
    </Category>
```

The following figure shows how the rule for CONSTR-L1 would provide an effective parsing of the structure described as the realizational variant (b) above:

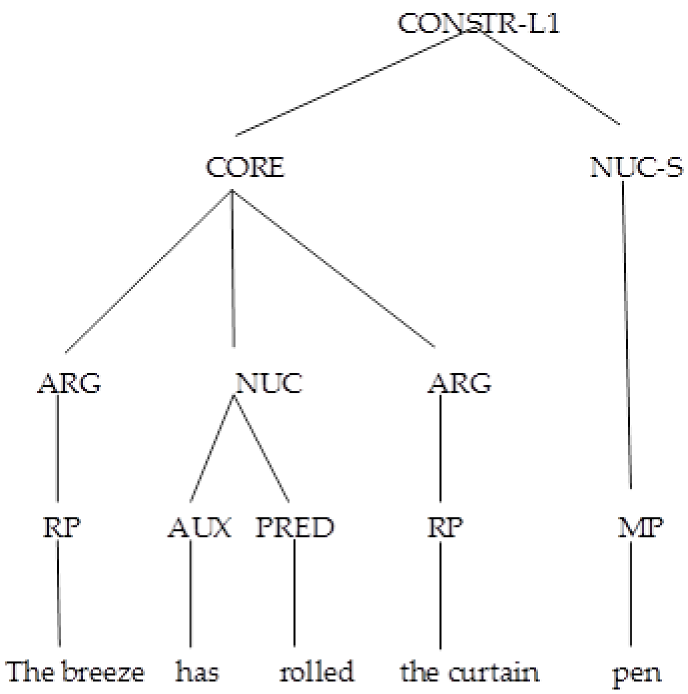

FIGURE 6. CONSTITUENT PROJECTION OF A CONSTR-L1
(a partial representation)

As explained before, this section of our rule accounts for the cases in which the CORE node is followed by a NUC-S, in the case shown in Figure 6 this constituent is contributed into the clause structure by the Template for Resultative Constructions encoded in the Grammaticon in FunGramKB.

3.2. The Clause

In Cortés-Rodríguez (2016) it is argued that, once the Level-1 construction layer is accepted, it seems more sensible to redefine the original Pre-Core slot position as PreC-L1 positions; the PreCore Slot is described in RRG as the place typically occupied by question words in languages in which they do not appear in situ (*What can you buy in such an expensive shop?*), and also by fronted constituents as in *A fortune they must have cost.* However, in sentences like:

(3) For whom did you knit the sweater? (Beneficiary L1-Construction)

(4) What did you wash these clothes with? (Instrumental Construction)

(5) Inside a vacuum chamber we blew a huge soap bubble, and it fell to the ground! (Caused-Motion L1-Construction)

the fronted constituent is not an argument of the CORE but an Argument Adjunt contributed by the corresponding L1 Construction; therefore, it is more plausible for our analysis to consider that the fronted position takes place before the CONSTR-L1 node. This possibility will be housed in the rewrite rule for the CLAUSE node, which is the next higher layer in the Layered Structure of Clauses.

CL [Akt:?, concept=?, Illoc : ?, status: ?, tpl=?, t=?] -> CONSTR-L1[Akt=?, concept=?, illoc=?, mod=?, neg=?, sta: ?, tpl=?, t=?, weight= 0 | 1 | 2 | 3 | 4 | 5 | 6] || PreC-L1 CONSTR-L1[Akt=?, concept=?, illoc=?, mod=?, neg=?, sta: ?, tpl=?, t=?, weight= 0 | 1 | 2 | 3 | 4 | 5 | 6] .

The rule that expands the node CL will basically include the CONSTR-L1 constituent plus the possibility of having any other constituent in a fronted clause initial position (PreC-L1). As mentioned before, such a constituent can be obligatory (ARG or AAJ) or an optional Adjunct. This is captured in its corresponding rule:

PreC-L1[concept=?, macro= A | U | n, num=?, per=?, phrase=?, role: ?, tpl=?, var=?] -> ARG[concept=?, macro= A | U | n, num=?, per=?, phrase=?, role: agent | attribute | goal | instrument | location | manner | origin | referent | result | theme, tpl=?, var= x | y | w | z] || AAJ[concept=?, macro= A | U | n, phrase=?, prep=?, role: agent | attribute | goal | instrument | location | manner | origin | referent | result | theme, tpl=?, var= x | y | w | z] || ADJUNCT[concept=?,, phrase=?, prep=?, role: Duration | Frequency | Goal | Instrument | Location | Manner | Means | Origin | Position | Purpose | Quantity | Reason | Result | Scene | Time]

Thus, the CL node will have the following AVM (cf. Cortés-Rodríguez 2016):

```
<Category Type="CL">
        <Attribute ID="Akt"/>
        <Attribute ID="Concept"/>
        <Attribute ID="Illoc" />
        <Attribute ID="Sta" />
        <Attribute ID="Template" />
```

```
            <Attribute ID="Tense" />
    </Category>
```

And the attributes, *Status*, *Tense*, and *Illocutionary Force*, which correspond to the original clause operators in RRG, would have the following AVMs:

```
     <Attribute ID="Status " obl="*" num="1">
            <Value>?sta</Value>
            <Value>inference>inf</Value>
            <Value>necessity>nec</Value>
            <Value>possibility>poss</Value>
            <Value>subjunctive>subj</Value>
     </Attribute>
    <Attribute ID="Tense" obl="*" num="1">
            <Value>?t</Value>
            <Value>past</Value>
            <Value>pres</Value>
    </Attribute>
    <Attribute ID="Illoc " obl="+" num="1">
            <Value>?illoc</Value>
            <Value Tag="declarative" >dec< /Value>
            <Value Tag="interrogative" >int< /Value>
            <Value Tag="imperative" >imp< /Value>
    </Attribute>
```

3.3. The Sentence

The rule for Sentence nodes introduces the node tags LDP and RDP, to account for the positions of detached constituents, which are clause external but sentence internal elements. Both LDP and RDP can be saturated by different types of constituents: ARGs, AAJs and ADJUNCTs. This is captured in their respective rewrite rules:

S—> CL[Akt:?, concept=?, Illoc : ?, status: ?, tpl=?, t=?] || LDP CL[Akt:?, concept=?, Illoc : ?, status: ?, tpl=?, t=?] || CL[Akt:?, concept=?, Illoc : ?, status: ?, tpl=?, t=?] RDP

LDP[concept=?, macro= A | U | n, num=?, per=?, phrase=?, role: ?, tpl=?, var=?] -> ARG[concept=?, macro= A | U | n, num=?, per=?, phrase=?, role: agent | attribute | goal | instrument | location | manner | origin | referent | result | theme, tpl=?, var= x | y | w | z] || AAJ[concept=?, macro= A | U | n, phrase=?, prep=?, role: agent | attribute | goal | instrument | location | manner | origin | referent | result | theme, tpl=?, var= x | y | w | z] || ADJUNCT[concept=?,, phrase=?, prep=?, role: Duration |Frequency |Goal |Instrument| Location|Manner|Means|Origin |Position |Purpose |Quantity |Reason |Result |Scene |Time]

RDP[concept=?, macro= A | U | n, num=?, per=?, phrase=?, role: ?, tpl=?, var=?] -> ARG[concept=?, macro= A | U | n, num=?, per=?, phrase=?, role: agent | attribute | goal | instrument | location | manner | origin | referent | result | theme, tpl=?, var= x | y | w | z] || AAJ[concept=?, macro= A | U | n, phrase=?, prep=?, role: agent | attribute | goal | instrument | location | manner | origin | referent | result | theme, tpl=?, var= x | y | w | z] || ADJUNCT[concept=?,, phrase=?, prep=?, role: Duration |Frequency |Goal |Instrument| Location|Manner|Means|Origin |Position |Purpose |Quantity |Reason |Result |Scene |Time]

The following figure shows the simplified constituent projection analysis obtained by these rules for the sentence *This toy bone, Marita has bought it for Bruno*:

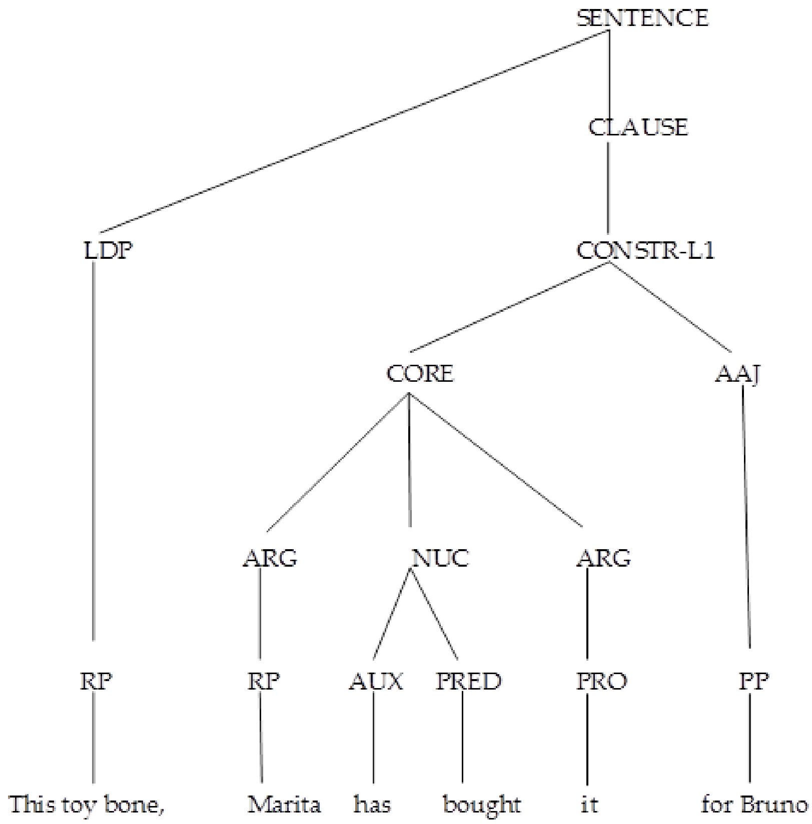

FIGURE 7. CONSTITUENT PROJECTION OF A SENTENCE
(a partial representation)

4. Conclusions

One of the computational resources that form part of the research project FUNK Lab is ARTEMIS, which, unlike most trending research in natural language processing, uses linguistic theories, i.e RRG, and the LSC, and the LCM four-level constructional schemata. This tool has the capacity to provide a morphosyntactic representation of an input text which subsequently transforms into a CLS, a fully-fledged semantic representation. ARTEMIS is composed of a number of different modules, the GDE, the CLS constructor and the COREL-Scheme Builder. The focus of this paper is on the GDE, which stores the production rules that form part of what we can call computational grammar. Three types of rules are used: lexical and constructional, which are created automatically, and syntactic rules, which are manually constructed.

Moreover, each grammatical unit is represented in terms of AVMs.

Within this framework, this paper discusses the format of the syntactic rules that account for the upper nodes of the LSC: the core (and the construction level-1 node associated to it), the clause and the sentences. In this process, we show that ARTEMIS, unlike the LSC, only uses one projection and operators are described in terms of AVMs. A number of important distinctions are discussed, i.e. the role of constructional schemata and their computational implementation by means of a new node, Level-1 Construction, that is part of the CORE.

The set of rules presented in this paper, together with those in Cortés-Rodríguez and Mairal-Usón (2016), provide an almost complete computational implementation of the LSC. There is, though, an issue which is still unresolved and this concerns the pervasive location of peripheral adjuncts. In closing, a further step is to evaluate this prototype within a simplified controlled language domain and observe the success rate when compared to other tools.

Acknowledgements

Financial support for this research has been provided by the Spanish Ministry of Economy, Competitiveness and Science, grant FFI2014-53788-C3-1-P.

References

Boas, Hans C. 2005. "From theory to practice: Frame Semantics and the design of FrameNet" In *Semantik im Lexikom* , edited by Stefan Langer and Daniel Schnorbusch, 129-160. Tübingen: Narr.

Boas, Hans C., and Ivan Sag, eds. 2012. *Sign-Based Construction Grammar*. Stanford: CSLI.

Cortés-Rodríguez, Francisco J. 2016. "Towards the computational implementation of Role and Reference Grammar: Rules for the syntactic parsing of RRG Phrasal constituents" *Círculo de lingüística aplicada a la comunicación* 65, 76-108. [http://www.ucm.es/info/circulo/no65/cortes.pdf]. https://doi.org/10.5209/rev_CLAC.2016.v65.51982

Cortés-Rodríguez, Francisco J., and Ricardo Mairal-Usón. 2016. "Building an RRG computational grammar" *Onomazein* (34), 86-117. DOI: 10.7764/onomazein.34.22 https://doi.org/10.7764/onomazein.34.22

Díaz Galán, Ana C., and M. Carmen Fumero Pérez. 2015. "Developing parsing rules within ARTEMIS: the case of DO auxiliary insertion" In *Understanding Meaning and Knowledge Representation: From Theoretical and Cognitive Linguistics to Natural Language Processing*, edited by Carlos Periñán-Pascual and Eva Mestre i Mestre, 283-302. Newcastle: Cambridge Scholars Press.

Diedrichsen, Elke. 2011. "The theoretical importance of constructional schemas in RRG" In *New Perspectives in Role and Reference Grammar*, edited by Wataru Nakamura, 168-197. Newcastle upon Tyne: Cambridge Scholars Publishing.

Diedrichsen, Elke. 2014: "A Role and Reference Grammar Parser for German" In *Language Processing and Grammars* , edited by Brian Nolan and Carlos Periñán-Pascual, 105-142. Amsterdam/Philadelphia: John Benjamins.

Foley, William A., and Robert D. Jr. Van Valin. 1984. *Functional Syntax and Universal Grammar*. Cambridge: Cambridge University Press.

Fumero Pérez, M. Carmen, and Ana C. Díaz Galán. 2017. "ARTEMIS: State of the Art and Future Horizons". *Revista de Lenguas para Fines Específicos 23/1* (2017). *In press.*

Goldberg, Adele E. 1995. *Constructions: A Construction Grammar approach to argument structure*. Chicago: University of Chicago Press.

Goldberg, Adele E. 2006. *Constructions at Work. The Nature of Generalization in Language*. Oxford: Oxford University Press.

Guest, Elisabeth. 2009. "Parsing using the Role and Reference Grammar paradigm". [http://eprints.leedsbeckett.ac.uk/778/6/Parsing%20Using%20the%20Role%20and%20Reference%20Grammar%20Paradigm.pdf, accessed 19 February 2016].

Mairal-Usón, Ricardo. 2015. "Constructional Meaning representation within a knowledge engineering framework" *Review of Cognitive Linguistics* 13:1 (2015), 1-27. https://doi.org/10.1075/rcl.13.1.01mai

Mairal-Usón, Ricardo. 2017. "A computational implementation of idiomatic and non-idiomatic constructions" *Signos* 50 (94), 241-264. https://doi.org/10.4067/S0718-09342017000200241

Mairal-Usón, Ricardo, and Pamela Faber. 2002. "Functional Grammar and Lexical Templates". In *New perspectives on predicate argument structure in Functional Grammar,* edited by R. Mairal-Usón and M.J. Pérez Quintero, 41-98. Berlin and New York: Mouton de Gruyter.

Mairal-Usón, Ricardo, and Pamela Faber. 2007. "Lexical templates within a functional cognitive theory of meaning". *Annual Review of Cognitive Linguistics* (5), 137-172. Amsterdam: John Benjamins.

Mairal-Usón, Ricardo, and Pamela Faber. 2017. "A conceptually-oriented approach to semantic composition in RRG". In *The Cambridge Handbook of Role and Reference Grammar*, edited by Robert D. Van Valin Jr. Cambridge: Cambridge University Press.

Mairal-Usón, Ricardo, and Carlos Periñán-Pascual. 2016. "Representing Constructional Schemata in FunGramKB Grammaticon" In *Exploring the Syntax-Semantics Interface*, edited by Jens Fleischhauer, Anja. Latrouite and Rainer Osswald, 77-108. Language and Cognition Series, Volume 3. Düsseldorf: Düsseldorf University Press.

Mairal-Usón, Ricardo, Carlos Periñán-Pascual, and María Beatriz Pérez Cabello de Alba. 2012. "La representación léxica. Hacia un enfoque ontológico". In *El Funcionalismo en la Teoría Lingüística. La Gramática del Papel y la Referencia. Introducción, Avances y Aplicaciones*, edited by Ricardo Mairal-Usón, Lilián Guerrero, and Carlos González, 85-102. Akal: Madrid.

Mairal-Usón, Ricardo, and Francisco J. Ruiz de Mendoza. 2009. "Levels of description and explanation in meaning construction" In *Deconstructing Constructions,* edited by Christopher Butler and J. Martín Arista, 53-98. Amsterdam: John Benjamins. https://doi.org/10.1075/slcs.107.08lev

Martín Díaz, M. Auxiliadora. 2017. "An Account of English YES/NO Interrogative Sentences within ARTEMIS". *Revista de Lenguas para Fines Específicos 23/1* (2017). *In press.*

Nolan, Brian, and Carlos Periñán-Pascual, eds. 2014. *Language processing and grammars*. Amsterdam: John Benjamins

Nolan, Brian, and Yasser Salem. 2011. "UniArab;: RRG Arabic-to-English machine translation" In *New Perspectives in Role and Reference Grammar*, edited by Wataru Nakamura, 312-346. Newcastle upon Tyne: Cambridge Scholars.

Pavey, Emma. 2010. *The Structure of Language. An Introduction to Grammatical Analysis.* Cambridge: Cambridge University Press. https://doi.org/10.1017/CBO9780511777929

Periñán-Pascual, Carlos. 2013. "Towards a model of constructional meaning for natural language understanding" In *Linking Constructions into Functional Linguistics: The Role of Constructions in a Functional Grammar*, edited by Brian Nolan and Elke Diedrichsen, 205-230. Amsterdam: John Benjamins.

Periñán-Pascual, Carlos, and Francisco Arcas Túnez. 2014. "The implementation of the FunGramKB CLS Constructor in ARTEMIS" In *Language Processing and Grammars,* edited by Carlos Periñán-Pascual and Brian Nolan, 165-196. Amsterdam: John Benjamins.

Periñán-Pascual, Carlos, and Ricardo Mairal-Usón. 2009. "Bringing Role and Reference Grammar to natural language understanding". *Procesamiento del Lenguaje Natural* 43, 265-273.

Ruiz de Mendoza, Francisco José, and Alicia Galera. 2014. *Cognitive modeling. A linguistic perspective.* Amsterdam: John Benjamins.

Ruiz de Mendoza, Francisco José, and Ricardo Mairal-Usón. 2008. "Levels of description and constraining factors in meaning construction: an introduction to the Lexical Constructional Model". *Folia Linguistica* 42 (2), 355-400.

Ruiz de Mendoza, Francisco José, and Ricardo Mairal-Usón**.** 2011. "Constraints on syntactic alternation: lexical-constructional subsumption in the Lexical-Constructional Model". In P.

Guerrero (ed.) *Morphosyntactic Alternations in English. Functional and Cognitive Perspectives*. Equinox; 62-82.

Salem, Yasser, A. Hensman, and Brian Nolan. 2008. "Towards Arabic to English machine translation." *ITB Journal* 17, 20–31.

Sag, Ivan. A., Thomas Wasow, and Emily. M. Bender. 2003. *Syntactic Theory: Formal Introduction*. Stanford: CSLI Publications.

Van Valin, Robert D. Jr. 2005. *Exploring the Syntax-Semantics Interface*. Cambridge: Cambridge University Press. https://doi.org/10.1017/CBO9780511610578

Van Valin, Robert D. Jr. 2008. "RPs and the nature of lexical and syntactic categories in Role an Reference Grammar". In *Investigations of the Syntax-Semantics-Pragmatics Interface,* edited by Robert D.Jr. Van Valin, 161-178. Amsterdam/Philadelphia: John Benjamins. https://doi.org/10.1075/slcs.105.14van

Van Valin, Robert D. Jr., and Randy LaPolla. 1997. *Syntax: Structure, Meaning and Function*, Cambridge: Cambridge University Press. https://doi.org/10.1017/CBO9781139166799

Van Valin, Robert D. Jr., and Ricardo Mairal-Usón. 2014. "Interfacing the Lexicon and an Ontology in a Linking Algorithm" In *Theory and Practice in Functional-Cognitive Space,* edited by María Ángeles Gómez, Francisco Ruiz de Mendoza and Francisco Gonzálvez-García, 205-228. Amsterdam: John Benjamins, pp. 205-228.

Van Valin, Robert D. Jr., and D. Wilkins. 1993. "Predicting syntax from semantics". In *Advances in Role and Reference Grammar,* edited by R. D. Jr. Van Valin, 499–534. Amsterdam: John Benjamins.

APPENDIX 1: LIST OF ABBREVIATIONS

AAJ	Argument-adjunct
ADV	Adverb
ADJ	Adjunct
APAR	Auxiliary (participle)
ARG	Argument
AUX	Auxiliary verb
AVM	Attribute-Value Matrix
CL	Clause
CLS	Conceptual Logical Structure
CONSTR-L1	Level 1 Construction
GDE	Grammar Development Environment
LCM	Lexical Constructional Model
LDP	Left detached Position
LSC	Layered Structure of the Clause
MODD	Modal verb (deontic)
MODST	Modal verb (epistemic)
N	Noun
NUC	Nucleus
NUC-S	Secondary Nucleus
XP	Phrase
PER	Periphery
PoCS	Post-Core Slot
POS	Part of speech
PP	Prepositional Phrase
PrCS	PreCore Slot
PreC-L1	Pre L1 Construction Slot
PRED	Predicate
RDP	Right Detached Position
RP	Referential Phrase
RRG	Role and Reference Grammar
S	Sentence
VING	Verb (gerund form)
VPAR	Verb (participle)

Appendix 2: Syntactic rules

CORE

CORE [concept=?, illoc=?, mod=?, neg=?, num=?, per=?, recip=?, reflex=?, sta: ?, tpl=?, t=?] -> ARG[concept=?, macro= a | u | n, num=?, per=?, phrase=?, role: agent | attribute | goal | instrument | location | manner | origin | referent | result | theme, tpl=?, var= x | y | w | z] NUC[asp: ?, concept: ?, illoc:?, mod: ?, num:?, per:?, recip:?, reflex:?, sta: ?, tpl:?, t: ?] || ARG[concept=?, macro= A | U | n, num=?, per=?, phrase=?, role: agent | attribute | goal | instrument | location | manner | origin | referent | result | theme, tpl=?, var= x | y | w | z] NUC[asp: ?, concept: ?, illoc:?, mod: ?, num:?, per:?, recip:?, reflex:?, sta: ?, tpl:?, t: ?] ARG[concept=?, macro= A | U | n, num=?, per=?, phrase=?, role: agent | attribute | goal | instrument | location | manner | origin | referent | result | theme, tpl=?, var= x | y | w | z] || ARG[concept=?, macro= A | U | n, num=?, per=?, phrase=?, role: agent | attribute | goal | instrument | location | manner | origin | referent | result | theme, tpl=?, var= x | y | w | z] NUC[asp: ?, concept: ?, illoc:?, mod: ?, num:?, per:?, recip:?, reflex:?, sta: ?, tpl:?, t: ?] ARG[concept=?, macro= A | U | n, num=?, per=?, phrase=?, role: agent | attribute | goal | instrument | location | manner | origin | referent | result | theme, tpl=?, var= x | y | w | z] ARG[concept=?, macro= A | U | n, num=?, per=?, phrase=?, role: agent | attribute | goal | instrument | location | manner | origin | referent | result | theme, tpl=?, var= x | y | w | z]

L1 CONSTRUCTION

CONSTR-L1[Akt=?, concept=?, illoc=?, mod=?, neg=?, sta: ?, tpl=?, t=?, weight= 0 | 1 | 2 | 3 | 4 | 5 | 6] -> CORE[concept=?, illoc=?, mod=?, neg=?, num=?, per=?, recip=?, reflex=?, sta: ?, tpl=?, t=?] || CORE[concept=?, illoc=?, mod=?, neg=?, num=?, per=?, recip=?, reflex=?, sta: ?, tpl=?, t=?] AAJ[concept=?, macro= A | U | n, phrase=?, prep=?, role: agent | attribute | goal | instrument | location | manner | origin | referent | result | theme, tpl=?, var= x | y | w | z] || CORE[concept=?, illoc=?, mod=?, neg=?, num=?, per=?, recip=?, reflex=?, sta: ?, tpl=?, t=?] NUC-S [concept=?, macro= A | U | n, phrase=?, prep=?, role: agent | attribute | goal | instrument | location | manner | origin | referent | result | theme, tpl=?, var= x | y | w | z] || CONSTR-L1[Akt=?, concept=?, illoc=?, mod=?, neg=?, sta: ?, tpl=?, t=?, weight= 0 | 1 | 2 | 3 | 4 | 5 | 6] AAJ[concept=?, macro= A | U | n, phrase=?, prep=?, role: agent | attribute | goal | instrument | location | manner | origin | referent | result | theme, tpl=?, var= x | y | w | z] || CONSTR-L1[Akt=?, concept=?, illoc=?, mod=?, neg=?, sta: ?, tpl=?, t=?, weight= 0 | 1 | 2 | 3 | 4 | 5 | 6] NUC-S[concept=?, macro= A | U | n, phrase=?, prep=?, role: agent | attribute | goal | instrument | location | manner | origin | referent | result | theme, tpl=?, var= x | y | w | z] || CONSTR-L1[Akt=?, concept=?, illoc=?, mod=?, neg=?, tpl=?, t=?, weight= 0 | 1 | 2 | 3 | 4 | 5 | 6] AAJ[concept=?, macro= A | U | n, phrase=?, prep=?, role= agent | attribute | goal | instrument | location | manner | origin | referent | result | theme, tpl=?, var= w] NUC-S [concept=?, macro= A | U | n, phrase=?, prep=?, role= agent | attribute | goal | instrument | location | manner | origin | referent | result | theme, tpl=?, var= v]

CLAUSE

CL [Akt:?, concept=?, Illoc : ?, status: ?, tpl=?, t=?] -> CONSTR-L1[Akt=?, concept=?, illoc=?, mod=?, neg=?, sta: ?, tpl=?, t=?, weight= 0 | 1 | 2 | 3 | 4 | 5 | 6] || PreC-L1 CONSTR-L1[Akt=?, concept=?, illoc=?, mod=?, neg=?, sta: ?, tpl=?, t=?, weight= 0 | 1 | 2 | 3 | 4 | 5 | 6] .

PreC-L1[concept=?, macro= A | U | n, num=?, per=?, phrase=?, role: ?, tpl=?, var=?] -> ARG[concept=?, macro= A | U | n, num=?, per=?, phrase=?, role: agent | attribute | goal | instrument | location | manner |

origin | referent | result | theme, tpl=?, var= x | y | w | z] || AAJ[concept=?, macro= A | U | n, phrase=?, prep=?, role: agent | attribute | goal | instrument | location | manner | origin | referent | result | theme, tpl=?, var= x | y | w | z] || ADJUNCT[concept=?,, phrase=?, prep=?, role: Duration | Frequency | Goal | Instrument | Location | Manner | Means | Origin | Position | Purpose | Quantity | Reason | Result | Scene | Time]

SENTENCE

S—> CL[Akt:?, concept=?, Illoc : ?, status: ?, tpl=?, t=?] || LDP CL[Akt:?, concept=?, Illoc : ?, status: ?, tpl=?, t=?] || CL[Akt:?, concept=?, Illoc : ?, status: ?, tpl=?, t=?] RDP

LDP[concept=?, macro= A | U | n, num=?, per=?, phrase=?, role: ?, tpl=?, var=?] -> ARG[concept=?, macro= A | U | n, num=?, per=?, phrase=?, role: agent | attribute | goal | instrument | location | manner | origin | referent | result | theme, tpl=?, var= x | y | w | z] || AAJ[concept=?, macro= A | U | n, phrase=?, prep=?, role: agent | attribute | goal | instrument | location | manner | origin | referent | result | theme, tpl=?, var= x | y | w | z] || ADJUNCT[concept=?,, phrase=?, prep=?, role: Duration |Frequency |Goal |Instrument| Location|Manner|Means|Origin |Position |Purpose |Quantity |Reason |Result |Scene |Time]

RDP[concept=?, macro= A | U | n, num=?, per=?, phrase=?, role: ?, tpl=?, var=?] -> ARG[concept=?, macro= A | U | n, num=?, per=?, phrase=?, role: agent | attribute | goal | instrument | location | manner | origin | referent | result | theme, tpl=?, var= x | y | w | z] || AAJ[concept=?, macro= A | U | n, phrase=?, prep=?, role: agent | attribute | goal | instrument | location | manner | origin | referent | result | theme, tpl=?, var= x | y | w | z] || ADJUNCT[concept=?,, phrase=?, prep=?, role: Duration |Frequency |Goal |Instrument| Location|Manner|Means|Origin |Position |Purpose |Quantity |Reason |Result |Scene |Time]

APPENDIX 3: ATTRIBUTE VALUE MATRICES

CATEGORIES

```
<Category Type="CORE">
        <Attribute ID="Concept" />
        <Attribute ID="Illoc" />
        <Attribute ID="Mod" />
        <Attribute ID="Neg” />
        <Attribute ID="Num"/>
        <Attribute ID="Per"/>
        <Attribute ID="Recip" />
        <Attribute ID="Reflex" />
        <Attribute ID="Sta" />
        <Attribute ID="Template"/>
        <Attribute ID="Tense" />
 </Category>


<Category Type="CONSTR-L1">
        <Attribute ID="Akt" />
        <Attribute ID="Concept" />
        <Attribute ID="Illoc" />
        <Attribute ID="Mod" />
        <Attribute ID="Neg” />
        <Attribute ID="Sta" />
        <Attribute ID="Template" />
        <Attribute ID="Tense" />
        <Attribute ID="Weight" />
 </Category>

<Category Type="CL">
         <Attribute ID="Akt"/>
         <Attribute ID="Concept"/>
         <Attribute ID="Illoc" />
         <Attribute ID="Sta" />
         <Attribute ID="Template" />
         <Attribute ID="Tense" />
 </Category>

 <Category Type="ARG">
           <Attribute ID="Concept"/>
           <Attribute ID="Macro"/>
           <Attribute ID="Num"/>
           <Attribute ID="Per"/>
           <Attribute ID="Phrase"/>
           <Attribute ID="Prep" />
           <Attribute ID="Role"/>
           <Attribute ID="Template" />
           <Attribute ID="Var"/>
  </Category>
```

```
<Category Type="AAJ">
        <Attribute ID="Concept"/>
        <Attribute ID="Macro"/>
        <Attribute ID="Num"/>
        <Attribute ID="Per"/>
        <Attribute ID="Phrase"/>
        <Attribute ID="Prep" />
        <Attribute ID="Role"/>
        <Attribute ID="Template" />
        <Attribute ID="Var"/>
</Category>

<Category Type="NUC-S">
        <Attribute ID="Concept"/>
        <Attribute ID="Macro"/>
        <Attribute ID="Phrase"/>
        <Attribute ID="Prep" />
        <Attribute ID="Role"/>
        <Attribute ID="Template" />
        <Attribute ID="Var"/>
</Category>
```


ATTRIBUTES


```
<Attribute ID="Status " obl="*" num="1">
        <Value>?sta</Value>
        <Value>inference>inf</Value>
        <Value>necessity>nec</Value>
        <Value>possibility>poss</Value>
        <Value>subjunctive>subj</Value>
</Attribute>

<Attribute ID="Tense" obl="*" num="1">
        <Value>?t</Value>
        <Value>past</Value>
        <Value>pres</Value>
</Attribute>

<Attribute ID="Illoc" obl="+" num="1">
        <Value>?illoc</Value>
        <Value Tag="declarative" >dec< /Value>
        <Value Tag="interrogative" >int< /Value>
        <Value Tag="imperative" >imp< /Value>
</Attribute>
```