Towards a Principled Computational System of Syntactic Ambiguity Detection and Representation

Hilton Alers-Valentín[1], Carlos G. Rivera-Velázquez[2], J. Fernando Vega-Riveros[2] and Nayda G. Santiago[2]

[1]*Linguistics and Cognitive Science Program, Department of Hispanic Studies, University of Puerto Rico-Mayagüez, P.O. Box 6000, Mayagüez, 00681-6000, Puerto Rico*

[2]*Department of Electrical and Computer Engineering, University of Puerto Rico-Mayagüez, P.O. Box 6000, Mayagüez, 00681-6000, Puerto Rico*

Keywords: Syntax, Parser, Lexicon, Structural Ambiguity, Computational Linguistics, Natural Language Processing.

Abstract: This paper presents the current status of a research project in computational linguistics/natural language processing whose main objective is to develop a symbolic, principle-based, bottom-up system in order to process and parse sequences of lexical items as declarative sentences in English. For each input sequence, the parser should produce (maximally) binary trees as generated by the Merge operation on lexical items. Due to parametric variations in the algorithm, the parser should be able to output (up to four) grammatically feasible structural representations accounted by alternative constituent analyses because of structural ambiguities in the parsing of the input string. Finally, the system should be able to state whether a particular string of lexical items is a possible sentence in account of its parsability. The system has a scalable software framework that may be suitable for the analysis of typologically-diverse natural languages.

1 INTRODUCTION

Natural language parsing is a computational process that takes as input a sequence of words and yields a syntactic structure for said sequence according to some sort of procedure. The production of a syntactic structure from a sequence determines whether it legally belongs to a language. There are two main types of parsers used to analyse word sequences. On one hand, there are probabilistic parsers which, given a statistical model of the syntactic structure of a language, will produce the most likely parse of a sentence, even if the word sequence is actually judged as ungrammatical by native speakers. Probabilistic parsers are widely used in natural language processing applications. However, they require a manually annotated corpus, a statistical learning algorithm, as well as training. Although these parsers are particularly good in identifying syntactic categories or parts of speech and have a desirable cost-benefit relation between accuracy and speed, they have been found rather ineffective in the representation of sentences containing relative clauses or long distance relations among constituents.

Deterministic parsers, on the other hand, use a system of syntactic rules to produce a structural representation. Deterministic parsers take input strings of natural languages and analyse them using production rules of a context free grammar. If, for a given sequence of lexical items, the rules of a language grammar cannot produce a structural representation, the sequence is considered ungrammatical for that language.

A single grammatical sequence, however, may have multiple representations if it is syntactically ambiguous. The (generally assumed) Principle of Compositionality states that the meaning of an expressions is a function of the meaning of its parts and of the way they are syntactically combined (Partee, 2004). As a consequence, syntactically ambiguous sequences may also be semantically ambiguous. There are two main causes of syntactic ambiguity: referential and structural. Referential ambiguity is due to possible valuations and interpretations of noun phrases and pronouns, as it happens with the three possible assignments of the possessive pronoun in *the woman said that she kicked her lover*. Structural ambiguity occurs when there exist multiple structural relations between the lexical

Alers-Valentín, H., Rivera-Velázquez, C., Vega-Riveros, J. and Santiago, N.
Towards a Principled Computational System of Syntactic Ambiguity Detection and Representation.
DOI: 10.5220/0007698709800987
In *Proceedings of the 11th International Conference on Agents and Artificial Intelligence (ICAART 2019)*, pages 980-987
ISBN: 978-989-758-350-6

items and constituents of a sentence, as in the classical example *the boy saw a man with a telescope*. Sometimes, structural ambiguity is caused by multiple syntactic category assignment to a lexical item, as can be seen in *visiting relatives can be a nuisance,* in which the first word can be tagged either as a transitive verb or as an adjective. In this paper, a computational system is described that detects syntactic ambiguity in a string and yields the correspondent structural representations.

For most probabilistic parsers, syntactic ambiguity, even ungrammaticality, remains undetected. To deal with structural ambiguity, we propose a deterministic (symbolic) parser that produces X-bar structural representations based on Principle-and-Parameters Theory modules to generate multiple syntactic parses for syntactically ambiguous sentences. Deterministic parsers in the form of minimalist grammars have been already formalized (Stabler, 1997, 2011; Collins and Stabler, 2016). Other symbolic parsers have been developed as computational models of syntactic competence (Berwick, 1985; Fong, 1991, 2005; Chesi, 2004, 2012); however, the parser we propose implements variation parameters that may account for structural ambiguity.

2 THEORETICAL BACKGROUND

Principles and Parameters Theory (Chomsky, 1981, 1995) is a generative-derivational theory of the human Faculty of Language. According to this theory, a natural (I-)language is an internal, individual, intensional cognitive state of the human mind (hence a mental organ) whose initial state, known as Universal Grammar (UG), contains a set of invariable principles and constraints that apply to all languages, as well as a set of variable parameters (possibly binary-valued) that children set during language acquisition from the primary linguistic data to which they are exposed. Among the fundamental principles of UG are the Structural Dependency Principle (syntactic structures show hierarchical structure and non-linear dependencies) and the Projection Principle (every minimal category projects its features to a maximal or phrase-level projection). Some of the best studied syntactic parameters are the Null Subject Parameter (languages may allow or disallow null subjects in finite clauses) and the Head Parameter (syntactic heads can be linearized before or after their complements). Languages, as steady states in the development of the Faculty of Language, are computational cognitive systems consisting of a lexicon, that contains representations of all primitives of linguistic computation (along with their features), and a grammar, a combinatorial system of operations on these representations. Sequences that satisfy all grammatical constraints of a language are mapped to (at least) one tuple of syntactic levels of structural representations: the theory-internal levels of what has been known as Deep and Surface Structure (DS, SS), and the interface levels of Logical Form (LF) and Phonetic Form (PF). In this theory, constraints are highly modularized and apply to syntactic structures from a certain level of representation onwards.

X-bar is a powerful and compact module of Principles and Parameters Theory (Adger, 2003; Carnie, 2013; Sportiche, Koopman and Stabler, 2014) for the representation of syntactic category formation in natural language, as it yields hierarchical structures in binary trees that encode the constituency of a sentence. The syntactic category or part-of-speech of a lexical item in a sentence is determined according to the item's morphology, grammatical features and syntactic distribution. Syntactic categories with referential meaning or content are classified as lexical (nouns, verbs, adjective, adverbs, prepositions), while those that strictly serve grammatical purposes and are required for well-formedness are called functional (determiners, complementisers, coordinators, tense, auxiliaries, negation). Heads are lexical items from which full phrases are formed and they project themselves into different levels. X-bar theory (where the variable X stands for a syntactic category) assumes three syntactic projection levels: minimal, intermediate and maximal. In the X-bar binary tree structure, minimal projections or heads (denoted sometimes as X°) are nuclear categories and do not dominate any other category, in other words, the terminal nodes of a syntactic tree. Intermediate projections (denoted as X′ and read as "X-bar") are typically generated from the merge of a minimal projection and a subcategorized complement. Maximal projections or phrases (denoted as XP) are the highest level of a nuclear category which has satisfied all its subcategorization requirements and may dominate another phrase-level constituent (a specifier) merged with the intermediate projection. The X-bar module has only three general rules that apply to all lexical categories, i.e. the specifier rule, the complement rule, and the adjunct rule. The context-free X-bar rules may be combined in any order so it allows the production of different structures from the same array of words or lexical items. As a recursive rule, Adjunction is the most

unconstrained syntactic operation and is related to most instances of structural ambiguity. X-bar by itself is overgenerative, which is problematic for a descriptively adequate model of linguistic competence. For this reason, other syntactic modules, like the Thematic Criterion, Case Filter, Binding Principles and Bounding Theory, among others, are needed to impose conditions on the legal combinations in a language.

More recent proposals call for a Minimalist Program (Chomsky, 1995, 2000) in the revision of linguistic theory. Under closer analysis, X-bar may not be a primitive, independently motivated principle of Universal Grammar, but the result of *Merge,* a more fundamental, recursive, binary operation on syntactic structures. Also, LF and PF may be the only required levels of syntactic representation as they are the interface between the computational system (the grammar) and both the conceptual-intentional and sensory-motor external systems.

3 SYSTEM ARCHITECTURE

As a computational model of syntactic analysis, the system we describe has two main components, i.e., the parser and the lexicon. The parsing process requires an input in the form of a word sequence. The input sequence is tokenized during the pre-processing stage. Tokenization involves the segmentation of a string into lexical items or word elements called tokens. The syntactic category or part-of-speech tagging process takes place right after tokenization. This process will access each tokenized element's grammatical features. For the tagging process to be successful, the tokenized element must be found in the lexicon. The parser will take the tagged sequence and will establish a relationship between each tagged element by generating the syntactic structures of the input word sequence in the form of binary trees that correspond to the SS level of syntactic representation.

3.1 The Parser

Parsing can be performed by means of bottom-up or top-down approaches. In the system described, a bottom-up, bidirectional algorithm is implemented that produces multiple binary trees when a sequence has syntactic ambiguity. The algorithm starts by segmenting the sequence into tokens that correspond to lexical items; this is followed by the categorization of each lexical item and the application of the syntactic rules in X-bar to produce the parse tree. The absolute upper bound of structural representations for a given sequence of *n* lexical items corresponds to the *n-1* Catalan number, a sequence of natural numbers that are used in combinatorics to determine, among other things, the number of distinct binary trees that can be created from a number of nodes.

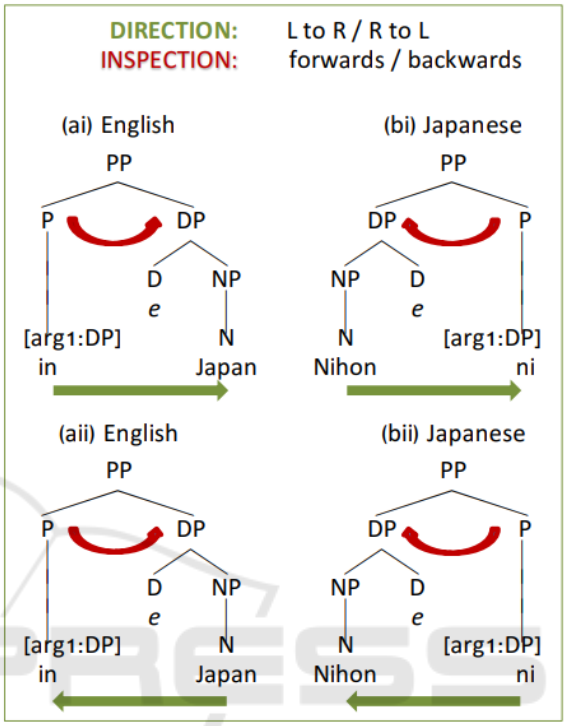

Figure 1: Interaction of direction and inspection.

The parser currently takes into account three parameters in order to be able to produce multiple representations in case of structural ambiguity. These parameters are direction, inspection and delay. The direction parameter determines if the input word sequence is to be analysed from left to right or from right to left. The implemented algorithm is bidirectional (LR and RR) because it analyses word sequences from left to right and from right to left, producing in both cases a rightmost derivation in reverse (Grune and Jacobs, 2008). The inspection parameter determines if the current lexical element looks forwards or backwards to compare itself with another lexical item to check for selection features. If the analysis is from left to right, the inspection parameter must be forwards (look one token ahead); if the analysis is from right to left then the inspection must be backwards (look one token back).

As it can be seen in (1), the setting of the inspection parameter is dependent on the direction parameter. Syntactic heads are categories with c(ategorial)-selection features or requirements

(subcategorization). In the case of head-first languages as English, the c-selection features of a head can be checked by a constituent to its right. In the case of subject-object-verb (SOV) or head-last languages, the left to right analysis would have backward inspection (1bi) and the right to left analysis would have forward inspection (1bii). Since this parser currently deals with declarative sentences in English, which show the subject-verb-object (SVO) constituent order typical of head-first languages, the inspection parameter is set to forward if the direction is left to right (1ai), but it is set to backwards if the direction is right to left (1aii).

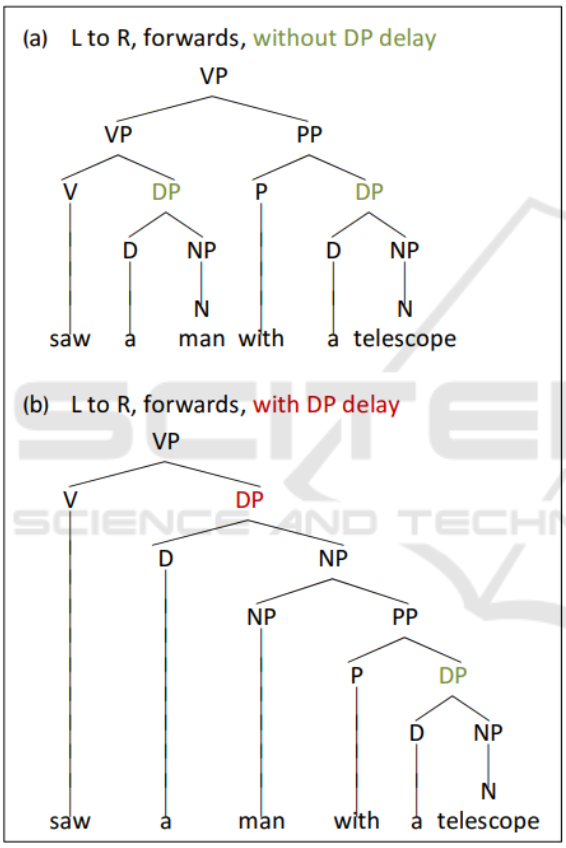

Figure 2: Interaction of direction, inspection, and delay.

The delay parameter, which is either true or false, determines if the formation of determiner phrases (DPs) not governed by a preposition are delayed until other phrase formations take place. The delay in the formation of these DPs, i.e., the merge of a determiner (D) with a noun phrase (NP), allows the formation of more complex DPs in certain contexts, particularly within verbal phrases (VPs). For example, prepositional phrases (PPs) can sometimes be parsed as adjuncts (modifiers) of VPs or as adjuncts of NPs, but not as adjuncts of DPs (for syntactic and semantic reasons out of the scope of this paper). While an NP is not merged with a D, the NP may adjoin a PP; otherwise, the PP can only be merged to the structure via VP adjunction. For example, the sequence *saw a man with a telescope* has two structural descriptions as shown in (2). Without DP delay (2a), the DPs *a man* and *a telescope* are formed within the same iteration. In the next iteration, the VP *saw a man* and the PP *with a telescope* are formed; thus, the PP can only adjoin the VP. However, with DP delay (2b), the merge of the D *a* with an NP is delayed, allowing the formation of a more complex NP with PP adjunction. As can be observed, the formation of the DP *a telescope* is not delayed as this DP is governed by the P *with*. Since the direction and inspection parameters seem to be mutually dependent, the combination of these three parameters will produce a total of four variations of the same basic algorithm. This allows for the possible production of up to four syntactic representations for structurally ambiguous word sequences.

3.2 The Lexicon

The lexicon is the language module that contains the grammatical information about the lexical items in the sentence that is to be analysed by the parser. Since it is necessary to determine if a certain combination of words is licensed or grammatical in the language, this system also requires the construction of a robust lexicon that at least contains the syntactic category, subcategorization frames and relevant grammatical features (such as case, c-selectional and phi- [or agreement] features) for each lexical item. As Fong, (2005:313) defines it, the lexicon is "the heart of the implemented system."

For this system, the lexicon was manually tagged by a team of linguists. To facilitate pre-processing, the lexicon contains every fully inflected word-form appearing in a corpus of 200 sentences that were constructed for validation purposes. Lexical items are entered as a string of literals, and features are indicated by means of different data types. All lexical items are labelled with a syntactic category; additionally, each category requires a specific subset of valued features.

Verb. The main predicative category, verbs are labelled according to their argument structure with the subclasses transitive, ditransitive and intransitive, and by their tense, aspect and mood (TAM) features as ±pret, perf, prog, pas, base. Transitive and ditransitive verbs are assigned a case feature, acc(usative), and are given a subcategorization frame for arg1; ditransitive verbs subcategorize also for

arg2. Optional c-selection features in subcategorization frames are indicated within parentheses. Intransitive verbs, on the other hand, which include both unergative and unaccusative verbs, are not assigned case or c-selection features. As for TAM features, verb forms tagged as ±pret, perf, prog, and pas are also labelled as finite with a 1 bit, and those tagged as base were labelled as non-finite with a 0 bit. In the case of passive participles, they are tagged as pas, no case feature is assigned, their arg1 frame is replaced with their arg2 frame, leaving the arg2 frame subsequently as an empty list. In this way, the parser could be able to analyse passive declarative clauses without accounting for syntactic movement or thematic roles.

Auxiliary. Items of this category are tagged with a subclass: Perf, Prog, Pas. Each auxiliary c-selects for a specific subclass of AuxP or VP with a particular TAM feature. For example, the perfect auxiliary *has* c-selects a VP (or another AuxP) with TAM perf: *He has interrogated the witness; He has been interrogated.*

T(ense). This category includes true modals and tense/finiteness markers. Items of this category always precede the negation particle *not*, are assigned a nom(inative) case feature, and c-select either a VP, an AuxP or a NegP as arg1 and (with the exception of infinitival *to*) a DP or CP as arg0. As a consequence, T will always merge with a verbal (functional) projection as complement and with a nominal or a clause-level projection in their specifier as a subject. Thus, the universal requirement that every clause must have a subject (known in the syntactic literature as the Extended Projection Principle or EPP) is satisfied to check the c-selection features in arg0.

Complementiser. With the exception of small clauses and raising structures, complementisers (Cs) are the maximal functional category of clauses and sentences. At this stage of the implemented system, complementisers are only labelled by their force as ±Q, although the parser is not yet handling interrogatives. Cs c-select a TP complement, and it is so specified in the arg1 frame.

Noun. Words of this category are classified as common or proper, which is syntactically relevant as the latter subclass does not generally admit determiners (at least definite and indefinite articles) in English. Nouns can also have argument structure, specially deverbal nouns, which they inherit from the verb they are derived from. But, unlike verbs, the c-selection features of nouns need not be checked in order to produce a well-formed structure: *The destruction was imminent; the destruction of Carthage was imminent.*

Determiner. This nominal functional category includes articles, demonstratives, possessives, as well as quantifiers and personal pronouns. The distribution of DPs is highly constrained: all (referential) DPs must be syntactically licensed by certain heads in order to appear in well-formed structures. Thus, the syntactic module of Case Theory rules the licensing and distribution of DPs. In order to implement this constraint, all Ds have to check a formal case feature with a licensing head: finite T, which checks (nom); transitive V, (acc); prepositions (Ps), obl(ique). Most determiners (with the exception of personal pronouns) may check any case in order to be licensed. On the other hand, since case in personal pronouns is morphological and not abstract, they must check a specific case according to their morphology: *he* must check nom; *him* must check either acc or obl. Non-pronominal determiners also c-select a complement NP, which is specified in the arg1 frame.

Other categories, such as adjective, adverb and preposition, are also labelled in the lexicon with the applicable features.

4 IMPLEMENTATION, TESTING AND RESULTS

The system was implemented in Python 3.6.1. The lexicon was provided and stored in a MySQL database system. The tagging process needed the *pymysql* external library in order to communicate with the database. The parsing algorithm was purely implemented in python but the representation of the binary trees was represented through the use of the external libraries *Plotly* and *igraph.*

As for the database system, a decision was made between different technologies that best fitted the system. MySQL was chosen mainly because of its compatibility with all operating systems and horizontal partitioning. As soon as the lexicon was completed, the design of the database started by defining its entity relationship diagram, how the tables should look, and the queries to be utilized for the algorithm. The database uses four tables: Lexicon, Arguments, Case, and Contractions. As each lexical item has a maximum of three arguments, a separate table was created to avoid mistakes on the lexicon. It should also be noted that, since arguments are represented as a string separated with commas, there is no intermediate table to assign an argument to each lexical item, since that is exactly what the tagging component will be expecting to receive. The final table, Contractions, was created to manage

contractions in a dynamic way. The tokenizer communicates with the server and, if a contraction is found on the sentence, it is separated in the two lexical items that form the contraction so the parser can analyse them. This Contractions table bears no relationship with the other tables created so far.

The system has a pipeline structure shown in (3):

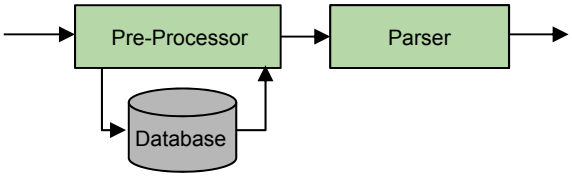

Figure 3: Pipeline structure of the system.

The system efficiency was tested with a set of 200 declarative sentences that was constructed to validate the algorithm. The two major aspects of testing involved grammaticality judgement and ambiguity detection. Ambiguous sequences have first to be considered as grammatical before alternate representations are produced. For the system to pass the grammaticality judgement test for a particular sequence, at least one structural representation must be produced for the sequence to be identified as grammatical. On the other hand, to pass the ambiguity detection test, the analyser had to produce at least two syntactic representations for a structurally ambiguous sequence. The parser correctly identified a sequence as grammatical for 78% of the validation sentences. The ambiguity detection test resulted in an 82.76% success rate for the grammatically identified sequences. However, some structures were particularly difficult for the parser to judge as grammatical, like for example, subordinate finite clauses with null complementisers, as in *I think he will not leave the house*. Although grammatically correct, this sequence failed to parse due to the fact that the verb *think* assigns accusative case to determiners and pronouns, yet the case feature of *he* is nominative, which was interpreted by the parser as a case feature mismatch. As expected, the sequence is judged correctly when the complementiser is overt (*I think that he will not leave the house*). Non-finite clauses were also challenging, like infinitival subjects (*To err is human*), which the current algorithm does not parse, and adjunct gerundive clauses (*I will meet John eating waffles*), since the parser expected present participles to be licensed by a progressive auxiliary. Double object constructions (*Poirot promised Maigret the job last week*) and Saxon genitive constructions (*The army's destruction of the city was imminent*) were typically misjudged by the parser as ungrammatical, on account of a case checking failure. Finally, for some sequences, although correctly judged as grammatical, the parser did not detect ambiguity (*I ate the macaroni that my mother cooked yesterday*).

5 CURRENT LIMITATIONS AND FUTURE WORK

The validation test results were promising, yet there are still some foreseen limitations in the system. Currently, the parser cannot produce all possible adjunctions in cases of four or more consecutive maximum projections on which adjunction can be performed (such as APs, PPs, and CPs for NPs; AdvPs, PPs and CPs for VPs). Two mechanisms have been identified to overcome this limitation. First, a new parameter, adjunction implementation, may be added to the algorithm. This parameter may signal either of two methods: sequential adjunction, in which only the original nodes of the current iteration would be available objects for adjunction, or nested adjunction, in which the newly formed nodes by previous adjunction within an iteration would be available as well for this operation. Each method produces distinct results in sequences of four or more consecutive maximal projections that can be adjoined. A second, perhaps simpler and more elegant mechanism, involves a recursive method that generates all possible adjunction patterns of this binary operation.

Due to time constraints in the project development, the current implementation deals with CPs in a somewhat different way than other phrases, by using a top-down rule instead of the typical bottom-up strategy used everywhere else. This fact may be the cause of some failures in infinitival subject analysis and other non-finite clauses, as well as occasional flaws in ambiguity detection in subordinate clauses. This present limitation is expected to be relatively easy to overcome by adding extra rules or functionalities to the implementation.

Noun phrases allow for empty determiners (nude NPs) for certain kinds of noun heads. Not only the distinction between common and proper nouns is relevant in this regard, but also the distinction between count and mass nouns is necessary for the correct grammaticality judgement of nude NPs in sentences. Some polysemic nouns have a meaning associated with the mass noun class and another with the count noun class (as with *resistance*). This requires additional tagging in the lexicon for the

distinction between these two subclasses, along with a method in the parsing implementation to handle the distinction appropriately. Also within the nominal domain, the Saxon genitive construction was challenging. A solution to the problem requires a treatment similar to the contractions and labelling 's as a D that c-selects an arg0.

Double object constructions, or even ditransitives in general, may present a challenge to our current system. For similar reason, complex NPs with more than one argument may not be correctly parsed. Among possible solutions to this problem is the inclusion of other functional or light categories to allow for richer structural representations.

Currently, all intransitive verbs, either unaccusatives or unergatives, are handled in the same way by the system. This may be problematic for the analysis of passive construction or for auxiliary selection in languages where the choice of certain verbal auxiliaries is dependent on whether the verb is unaccusative or not. Again, a richer VP-internal structure representation may be needed, as well as some implementation of a Theta-Theory module for both the parser and the lexicon.

Since the current parser does not account for syntactic movement, structures that require overt transformations such as interrogatives and relative clauses are not analysed. A single syntactic object may not comply with all required conditions, but a chain structure consisting of a moved object (such as a DP) and its trace in its base position would comply simultaneously with, for example, Case Filter and Theta Theory, respectively. Different mechanisms are being considered for its implementation.

Along with the inclusion of mechanisms to account for movement, it would be necessary for the parser to recognize locality of dependencies and violations thereof, for which a Bounding Theory module must be implemented.

Referential ambiguity is not detected by the present system, as it requires additional data structures, as indices for coreferentiality and the implementation of Binding Conditions on the interpretation of referential expressions and pronouns. Various mechanisms should be considered to overcome this limitation.

The structural representations generated by this system mostly correspond to the SS level of syntactic representation. At this level, certain operator scope ambiguities may not be detected. These scope differences are encoded in the LF interface level, where it is argued that operators such as quantifiers or interrogative expressions covertly move, obeying the same movement restrictions and locality conditions of overt movements. To achieve this, the system would have to generate LF structural representations instead. Operating on logical forms would also facilitate the integration of this system with semantic functionalities such as semantic composition and valuation, and textual entailment.

Fortunately, this computational system has been purposely designed to have a scalable software framework, so that more functionalities may be added with minimum impact on current methods and data structures. Although at this stage the system has been implemented exclusively to analyse English sentences, it may well suit typologically-diverse natural languages. The lexicon database may be easily augmented and the language-specific methods to handle English sequences are minimal, an advantage inherent to principled-based over rule-based systems.

ACKNOWLEDGEMENTS

This project was born from the fruitful discussions within the CompLing/NLP Research Group at the University of Puerto Rico-Mayagüez (UPRM) The authors would like to acknowledge the contributions of the other members of Team Forest at UPRM: Orlando Alverio, who was responsible for the system database and user interface, and the Linguistics team who were in charge of lexical tagging: Maday Cartagena, Jerry Cruz, Luis Irizarry, Joshua Mercado, Alejandra Santiago, and Ammerys Vázquez. David Riquelme and Victor Lugo reviewed the parser's output and helped with the validation process. The authors are also indebted to two anonymous reviewers for their valuable observations and suggestions.

REFERENCES

Adger, D., 2003. *Core Syntax: A Minimalist Approach,* Oxford University Press. Oxford.

Berwick, R.,1985. *The acquisition of syntactic knowledge,* MIT Press. Cambridge, Mass.

Carnie, A., 2013. *Syntax: A Generative Introduction,* Blackwell. Oxford, 3rd edition.

Chesi, C., 2004. *Phases and Cartography in Linguistic Computation.* Ph.D. Thesis. University of Siena.

Chesi, C., 2012. *Competence and Computation: Towards a Processing Friendly Minimalist Grammar,* Unipress. Padova.

Chomsky, N., 1981. *Lectures on Government and Binding,* Mouton de Gruyter. Berlin.

Chomsky, N., 1995. *The Minimalist Program,* MIT Press. Cambridge, Mass.

Chomsky, N., 2000. Minimalist Inquiries: The Framework. In Martin, R. et al., 2000. *Step by Step. Essays on Minimalist Syntax in Honor of Howard Lasnik*. Cambridge, Mass: MIT Press.

Collins, C., Stabler, E., 2016. A Formalization of Minimalist Syntax. *Syntax* 19,1.

Fong, S., 1991. *Computational Properties of Principle-Based Grammatical Theories.* Ph.D. Thesis. MIT.

Fong, S., 2004. *Computation with probes and goals.* In Di Sciullo, 2004. Amsterdam: John Benjamins.

Gomez-Marco, O., 2015. *Towards an X-Bar parser: A model of English syntactic performance.* M.S. Thesis. University of Puerto Rico-Mayagüez.

Grune, J., Jacobs, C.J.H., 2008. *Parsing Techniques. A Practical Guide,* Springer. Amsterdam, 2nd edition.

Joshi, A., Schabes, Y., 1997. *Tree-Adjoining Grammars.* In Rozenberg, G. and Salomaa, A., 1997. Berlin: Springer.

Partee, B., 2004. *Compositionality in Formal Semantics*, Blackwell. Oxford.

Partee, B., ter Meulen, A., Wall, R., 1990. *Mathematical methods in linguistics,* Kluwer Academic. Dordrecht.

Phillips, C., 1996. *Order and Structure.* Ph.D. Thesis. MIT.

Sportiche, D., Koopman, H., Stabler, E., 2013. *An introduction to syntactic analysis and theory*, John Wiley and Sons. Oxford.

Stabler, E., 1997. *Derivational Minimalism.* In Retoré, C., 1997. *Logical Aspects of Computational Linguistics.* Berlin: Springer.

Stabler, E., 2011. *Computational Perspectives on Minimalism.* In Boeckx, C., 2011. *The Oxford Handbook of Linguistic Minimalism.* Oxford: OUP.