ISSN 1870-4069

Proposal for Modeling Brazilian Portuguese with Adaptive Grammars

Djalma Padovani, João José Neto

University of São Paulo,
School of Engineering of the University of São Paulo,
Brazil

djalma.padovani@usp.br, joao.jose@poli.usp.br

Abstract. Natural Language Processing uses different techniques for identifying elements of the language and the syntactic and semantic roles they carry out in the text under analysis. Traditionally, NLP systems are built with modules that divide the text, identify its elements, verify whether the syntactic trees are in accordance with grammar rules, and apply specific formalisms to validate the semantics. However, it is noticed that there are few formalisms that represent semantics in a syntactic way and such formalisms are either very complex or incomplete. Adaptive Grammars is a formalism in which a grammar can modify itself based on the character chain parsing and the application of rules associated to the context. It brings several advantages over similar techniques because it allows the representation of both syntactic and semantics together in a single model. This work presents a method for modeling Natural Languages using Adaptive Grammars and illustrates the proposal with an application to Brazilian Portuguese.

Keywords: Computational, terminology, formalisms and knowledge representation ·linguistics, grammars, automata, natural, language processing.

1 Introduction

Natural Language Processing uses several techniques for identifying elements of the language and the syntactic and semantic roles they carry out in the text in which they are inserted. Traditionally, NLP systems are built with modules that divide the text, identify its elements, verify whether the syntactic trees are in accordance with grammar rules, and apply specific formalisms to validate the semantics. However, it is noticed that there are few formalisms that represent semantics in a syntactic way and such formalisms are either very complex or incomplete. Adaptive technology presents a very practical formalism with great potential for application in all stages of natural language processing, including semantic validation. Adaptive Grammars is a formalism proposed by Iwai [1] in her doctoral thesis, in which grammatical rules are created dynamically from the processing of the input chain and from information about the context in which they are found.

That formalism presents several advantages over similar techniques, because with a single model it is possible to represent both syntactic and semantic characteristics. Iwai also demonstrates the computational equivalence between adaptive grammars and adaptive automata, which allows phrases of the language generated by an adaptive grammar to be recognized by adaptive automata. This work presents a method for modeling Natural Languages by using Adaptive Grammars and illustrates the proposal with an application to Brazilian Portuguese.

1.1 Concepts and Related Works

Natural Language Processing requires the development of programs that are capable of determining and interpreting the sentence structure at many levels of detail. Natural languages exhibit intricate structural behavior since the particular cases to be considered are profuse. Since natural languages are never formally designed, their syntactic rules are neither simple nor obvious and thus their computational processing is complex. Many methods are employed in NLP systems, adopting different paradigms, such as exact, approximate, pre-defined or interactive, intelligent or algorithmic methods [2].

Regardless of the method used, natural language processing involves the operations of lexical-morphological analysis, syntactic analysis, semantic analysis and pragmatic analysis [3]. The lexical-morphological analysis seeks to assign a morphological classification to each sentence word from the information stored in the lexicon [4]. The lexicon or dictionary is the data structure containing the lexical items and information corresponding to these items. Among the information associated with lexical items are the grammatical category of the item, such as noun, verb and adjective, and morphosyntactic-semantic values such as gender, number, grade, person, time, mode, verbal or nominal regency.

In the parsing step, the parser checks whether a sequence of words is a valid sentence in the language, recognizing it or not. The syntax analyzer makes use of a lexicon and a grammar that defines the rules of combining the items in sentence formation. In cases where there is a need to interpret the meaning of a text, the lexical-morphological analysis and the syntactic analysis are not enough, and it is necessary to perform a new type of operation, called semantic analysis [4]. The semantic analysis looks for mapping the syntactic structure to the domain of the application, making the structure gain a meaning.

The mapping is done by identifying the semantic properties of the lexicon and the semantic relationship between the items that compose it [5]. The pragmatic analysis seeks to reinterpret the structure that represents what was said to determine what was really meant. This category includes anaphoric relations, relations, determinations, focuses or themes, deictics and ellipses [6].

Adaptive grammar model is defined in [7] as a grammatical formalism that allows sets of production rules to be explicitly manipulated within a grammar. Types of manipulation include rule addition, deletion, and modification. The first description of grammar adaptivity (though not under that name) in the literature is generally [8, 9, 10] taken to be in a paper by Alfonso Caracciolo di Forino published in 1963 [11]. The next generally accepted reference to an adaptive formalism (extensible context-free grammars) came from Wegbreit in 1970 [12] in the study of extensible programming languages, followed by the dynamic syntax of Hanford and Jones in 1973[13]. The work of Iwai in 2000 [1] takes the adaptive automata of Neto [14, 15] further by applying adaptive automata to context-sensitive grammars.

2 Adaptive Grammars

According to Iwai [1], adaptive grammar is a generative formalism capable of representing context-sensitive languages. What distinguishes this grammar from the conventional ones is its ability to self-modify as the sentences of language are derived. The modifications take place during the generation of the sentence, when applying production rules to which are associated adaptive actions, whose execution causes changes in the set of production rules and, possibly, in the set of non-terminal symbols.

A sentence ω belonging to the language represented by an adaptive grammar is generated from an original grammar G^0 and from a succession of intermediate G^1 ... G^{n-1} grammars, created whenever some adaptive action is activated during sentence generation, and finishes using G^n as the final grammar.

The author defines an adaptive grammar G as being a ordered triple (G^0, T, R^0), where: T is a finite, possibly empty, set of adaptive functions; $G^0 = (V_N^0, V_T, V_C, P_L^0, P_D^0, S)$ is an *initial grammar*, where $V_N{}^0$ is a finite non-empty set of *non-terminal symbols*, V_T is a finite non-empty set of *terminal symbols*, $V_N^0 \cap V_T = \varnothing$, V_C is a finite set of *context symbols*.

$V^0 = V_N^0 \cup V_T \cup V_C$, where V_N^0, V_T and V_C are disjoint sets two to two, $S \in V_N^0$ Is the initial symbol of grammar, P_L^0 is the set of rules of production applicable to situations free of context and P_D^0 is the set of rules of production applicable to situations dependent on context. The production rules consist of expressions with the following formats, i being an indicator of the number of adaptive changes already applied to the initial grammar:

Type1 or belonging to the set P_L^i, where i is an integer, greater than or equal to zero:

$N \rightarrow \{ A \} \alpha$, where $\alpha \in (V_T \cup V_N)^*$, $N \in V_N$ and A is an optional adaptive action associated with the production rule.

Type 2 or belonging to the set P_L^i, where i is an integer, greater than or equal to zero:

$N \rightarrow \varnothing$, where $\varnothing$ is a meta-symbol indicating the empty set.

This production indicates that although the non-terminal symbol N is defined, an empty set is derived, that is, there is no intended substitution for that non-terminal. This means that if this rule is applied in some derivation, the grammar will not generate any sentence. This rule is used for the case where there are rules that refer to non-terminals that should be dynamically defined, as a result of the application of some adaptive action.

Type 3 or belonging to the set $P_D{}^i$, where i is an integer, greater than or equal to zero:

$\alpha N \leftarrow \{ A \} \beta M$ where $\alpha \in V_C \cup \{\varepsilon\}$ e $\beta \in V_C$, or $\alpha N \rightarrow \{ A \} \beta M$, where $\alpha \in V_C$, $\beta \in V_T \cup \{\varepsilon\}$, N and $M \in V_N{}^i$, and A being an optional adaptive action. The first production has the arrow in reverse, indicating that β is being injected into the input chain. This production exchanges αN by βM, inserting context information. The second output has the arrow in the right direction, but has on its left side a context symbol followed by a non-terminal symbol, which indicates that αN is being replaced by βM and generating β in the output chain. R^0 is a relationship of type (r, A), where r ∈ ($P_L{}^0$ ∪ $P_D{}^0$) and A ∈ T, $R^0 \subseteq P^0 \times (T \cup \{\varepsilon\})$. For each production rule r there is a relation R^0 that associates it to an adaptive action A.

A production rule to which an associated adaptive action is associated is called the adaptive production rule. The expression defining a production rule of type 1 of the adaptive grammar is of the context-free type less than the adaptive action, which, together with the productions in $P_D{}^0$, accounts for the representation of the context dependencies in this grammar.

According to Neto [14], adaptive actions are called adaptive functions. Iwai [1] uses the concept, originally designed for automata, and modifies it to be used in adaptive production rules. The notation proposed by the author is as follows:

Action name (list of formal parameters)
{ list of variables, list of generators:
optional adaptive action
elementary action 1,
elementary action 2,
.....
elementary action n,
optional adaptive action.
}

In the adaptive function header there is the name of the adaptive function, followed by a list of formal parameters, separated by commas, in parentheses. When the adaptive function is called, this list will be filled with the corresponding values of the arguments passed in the particular function call, which values will be preserved intact throughout the function execution. The body of the adaptive function is represented by braces. It consists of a list of variables (separated by commas), a list of generators (separated by commas), followed by colon.

Next, the adaptive function may optionally contain the call of some adaptive function. This adaptive action is processed prior to the execution of the adaptive function being declared. Then, several elementary adaptive actions are listed, and in the end, there may be another adaptive function call to be processed after the execution of all the elementary actions of the adaptive function being declared.

Variables are symbols that give names to elements whose values are unknown at the time of the call of the adaptive function and which must be filled during the execution of the adaptive function as a result of adaptive query actions.

Generators are elements similar to variables, but are automatically filled at the beginning of the execution of the adaptive function, with values that do not repeat, preserving this value during the whole execution of the function.

According to the author's proposal, there are three types of elementary adaptive actions: query, addition and removal, which can be represented as follows, respectively:

? [N → { optional_adaptive_action } M],

+ [N → { optional_adaptive_action } M],

- [N → { optional_adaptive_action } M],

where $N \in V_N{}^i$, $M \in V^{i*}$, for some $i \in I$, where i represents the step of the evolution of grammar, and for some adaptive action, if it exists.

Elementary Consultation

This action verifies the existence of some production rule that presents the indicated format in the current set of productions of the grammar. When one or more symbols of the expression are represented by some variable, this action will fill those variables with the corresponding values that are found. Consequently, variables are filled as a result of performing the basic query actions. It should be noted that the variables, initially indefinite, are filled at most only once during the execution of an adaptive function.

Elementary Addition Action

This elementary action includes in the grammar a new production rule with the format indicated, being able to use variables and also generators for the definition of symbols that did not exist until the moment of its application. The addition of an existing rule is innocuous. The addition of rules that refer to undefined variables is also innocuous.

Elementary Removal Action

The execution of this elementary action is done in two steps. The first is to check the existence of the rule if you want to exclude with the consequent completion of the variables. If so, the removal is done, otherwise nothing is removed. It should be noted that, since there is more than one elementary adaptive action to be performed regardless of the order in which they were declared, the query rules take precedence. Between removals and additions, removals take precedence. The additions are always performed last. Elementary actions that refer to undefined variables will not be performed.

Adaptive Actions for Grammar Change

In the execution phase, adaptive actions are responsible for grammatical changes. When an adaptive action is performed, the grammar evolves, changing its non-terminal symbol sets and production rules.

Example of an adaptive function call:

$$
\begin{aligned}
A\,(A1, B1, C1) = \{A2^*, B2^*, C2^*:\\
&+ [A1 \rightarrow \{A\,(A2, B2, C2)\}\ aA2]\\
&+ [A1 \rightarrow \{B\,(B2, C2)\}\ \varepsilon]\\
&+ [B \rightarrow b\ B2\ ]\\
&+ [C \rightarrow c\ C2]\}.
\end{aligned}
$$

Derivation Sequence

The generation of the sentences of a language represented by an adaptive grammar occurs through successive substitutions of the non-terminal symbols, according to the rules of grammar, starting from an initial symbol S. The primordial difference of this procedure is the presence of the adaptive actions that, when activated, alter the grammar that was being used until the moment the adaptive rule was applied. Another important difference with respect to conventional grammars refers to contextual productions, which can be used for context symbols in the derivation, so that the substitutions that

Table 1. Example of Celso Luft's phrasal patterns.

1	SS1	Vlig	SS2
2	SS	Vlig	Sadj
3	SS	Vlig	Sadv| Sadj

depend on them are performed only when said context symbol is explicitly present in the Sentential form when substitution of the non-terminal affected by it.

Normal form of adaptive grammars

The author also presents the concept of normal form of adaptive grammars and a standardization technique. Considering $G = (G^0, T, R^0)$ an adaptive grammar, where $G^0 = (V_N^0, V_T, V_C, P_L^0, P_D^0, S)$ the initial grammar that implements G, T is the set of adaptive actions, and R^0 the set of relationships that associates productions rules with adaptive actions.

The normalized grammar $G^N = (G^{0'}, T', R^{0'})$ is defined by $G^{0'} = (V_N^{0'}, V_T', V_C', P_L^{0'}, P_D^{0'}, S)$, the new grammar that implements G^N, T', its set of adaptive actions, and $R^{0'}$, the set of relationships that associates their production rules with the adaptive actions. If the grammar root S is recursive, a production $S' \rightarrow S$ is created, ensuring that all grammar roots are not self-recursive. Let's take a production rule with the following form:

$$A \rightarrow \{A\}\ a_1\ a_2 \ldots a_n.$$

a) if n = 0, then $A \rightarrow \varepsilon$, and the corresponding normal form is $A \rightarrow \{A\}\ \varepsilon$,

b) if n > 0, then the corresponding normal form is:

1. $A \rightarrow \{A'\}\ a_1\ A_1,$
2. $\alpha_1\ A_1 \rightarrow a_2\ A_2$

:

n. $\alpha_{n-1}\ A_{n-1} \rightarrow a_n\ A_n$

n+1. $\alpha_n\ A_n \leftarrow \alpha A_{n+1}$

n+2. $A_{n+1} \rightarrow \varepsilon$

with $\alpha \in V_C$, $\alpha_i \in V_C$ if $a_i \in V_N$, and $\alpha_i = \varepsilon$ if $\alpha_i \in V_T.$

A' is the normalized adaptive action equivalent to A. If $A = \varepsilon$, then $A' = \varepsilon$. If $a_1 \in V_N$, then the first rule of the set of normalized rules is changed to:

0. $A \rightarrow \{A'\}\ A_0.$

And the rule 1 becomes:

1. $A_0 \rightarrow a_1\ A_1$

If A is the initial symbol of the grammar, then there is not a rule n+1, because it is not necessary to insert a context symbol for the initial one. The latest normalized rules are now as follows:

n. $\alpha_{n-1}\ A_{n-1} \rightarrow a_n\ A_n,$

n+1. $\alpha_n\ A_n \rightarrow \varepsilon.$

The normalization of productions of type X→ ({A1} A1 | {A2} A2) is done as follows:

$X \rightarrow \{A_1'\}\ A_1X_1 \qquad \alpha_1X_1 \rightarrow X_2,$
$X_2 \rightarrow \{A_2'\}\ A_2X_3 \qquad \alpha_2X_3 \rightarrow X,$
$X_2 \rightarrow \varepsilon.$
with $\alpha_i \in V_C$ if $A_i \in V_N$, and $\alpha_i = \varepsilon$ if $A_i \in V_T$.

Ai' is the normalized adaptive action equivalent to Ai. If Ai = ε, then Ai'= ε. If each Ai is an expression (VN ∪ VT) * then apply the rules a and b before mentioned. The normalization of productions of the type X→ ({A1} A1 | {A2} A2 |... |{An} An) is done as follows:

$X \rightarrow \{\ A_1'\}\ A_1\ X_1 \qquad \alpha_1X_1 \rightarrow X_2,$
$X \rightarrow \{\ A_2'\}\ A_2\ X_1 \qquad \alpha_2X_1 \rightarrow X_2,$
$\vdots \qquad \vdots$
$X \rightarrow \{A_n'\}\ \ A_nX_1 \qquad \alpha_nX_1 \rightarrow X_2,$
$X_2 \rightarrow \varepsilon.$
with $\alpha_i \in V_C$ if $A_i \in V_N$, and $\alpha_i = \varepsilon$ if $A_i \in V_T$.

Ai' is the normalized adaptive action equivalent to Ai. If Ai = ε, then Ai'= ε. If each Ai is an expression (VN ∪ VT) * then apply the rules a and b before mentioned. The normalization of adaptive actions is done in a similar way to the previously presented production rules. But is these case, intermediary non-terminals are variables that will be filled as arguments are passed. For example, an adaptive action that contains elementary actions of consultation, removal and addition, and represented as follows:

A (x, y) = { ? [x→ a b c]
- [y→ d e]
+ [z → f] }.

In the normalized form becomes:

$B\ (x, y) = \{\ u_1, u_2, u_3, u_4, v_1, v_2, v_3, t_1, t_2:$
$?\ [\ x \rightarrow a\ u_1],\ ?[u_4 \rightarrow \varepsilon],\ -[v_3 \rightarrow \varepsilon\ ],$
$?\ [u_1 \rightarrow b\ u_2],\ -[\ y\rightarrow d\ v_1],\ +[\ z \rightarrow f\ t_1],$
$?\ [u_2 \rightarrow c\ u_3],\ -[v_1 \rightarrow e\ v_2],\ +[\ t_1\leftarrow z'\ t_2],$
$?\ [u_3 \leftarrow x'\ u_4],\ -[v_2 \leftarrow y'\ v_3],\ +[\ t_2 \rightarrow \varepsilon\ ]\ \}.$

Context productions of the forms A ←αB, αA←βB, αA→B , and αA→ bB do not change, since they are defined only with the right side of the production, presenting at most only two elements.

3 Modern Brazilian Grammar of Celso Luft

The Modern Brazilian Grammar of Celso Luft [16] categorizes in a clear and precise way the different types of sentences of Portuguese Language, differing from the other grammars, which prioritize the description of the language to the detriment of the structural analysis of the same. Luft says that sentences are shaped by phrasal patterns, composed of elements called phrases. An example is given in Table 1.

Phrasal pattern is any immediate constituent of a sentence, and can play the role of subject, complement (direct and indirect object), predicative and adjunct adverbial. It is composed of one or more words, one being classified as a nucleus and the other as dependent. Dependent words may be located to the left or right of the nucleus. Luft uses the following names and abbreviations:

1. SS: Noun phrase - nucleus is a noun,
2. SV: Verb phrase - nucleus is a verb,
3. Sadj: Adjective phrase - nucleus is an adjective,
4. Sadv: Adverbial phrase - nucleus is an adverb,
5. SP: Prepositional phrase - formed by a preposition (Prep) plus one SS,
6. Vlig: linking verb,
7. Vi: intransitive verb,
8. Vtd: direct transitive verb,
9. Vti: indirect transitive verb,
10. Vtdi: direct and indirect transitive verb,
11. Vtpred: transitive verb predicative.

Luft also presents a comprehensive formula for sentence patterns:

[SS] V [SS] [SS | Sadj | Sadv | SP] [SP] [SP]

The symbol V represents all types of verbs: Vlig, Vi, Vtd, Vti, Vtdi, and Vtpred. Luft's formula allows us to generate any type of sentence pattern by simply using the productions defined by the grammar.

4 Introduction of Adaptive Mechanisms in Celso Luft's Grammar

The introduction of adaptive mechanisms in Celso Luft's Grammar has as main objective the use of a single model of representation that is able to generate sentences free and dependent on context.

Specifically, the adaptive characteristic of the Iwai Grammar will be used to generate the productions corresponding to the Luft sentence patterns in function of the lexical characteristics of the analyzed text. In the case of simple periods, verbs will be used to determine the pattern of previous and subsequent terms.

In the case of compound periods, conjunctions and relative pronouns will be used as defining elements of sentence break patterns. Therefore, the context dependence of this work is limited to the use of semantics to define syntactic sentence patterns and their respective rules of production. The work was divided into the following steps:

1. Consolidation of Celso Luft's Sentence Patterns.
2. Standardization of Production Rules.
3. Specification of Adaptive Grammar.

Table 2. Consolidated sentence patterns of Celso Luft's grammar.

Nominal Personal Patterns				
SS	Vlig	SS		
SS	Vlig	Sadj		
SS	Vlig	Sadv		
SS	Vlig	SP		
Verbal Personal Patterns				
SS	Vtd	SS		
SS	Vti	SP		
SS	Vti	Sadv		
SS	Vti	SP	SP	
SS	Vtdi	SS	SP	
SS	Vtdi	SS	Sadv	
SS	Vtdi	SS	SP	SP
SS	Vi			
Verbal Nominal Personal Patterns				
SS	Vtpred	SS	SS	
SS	Vtpred	SS	Sadj	
SS	Vtpred	SS	SP	

Table 2. Consolidated sentence patterns of Celso Luft's grammar – Continuation.

Verbal Nominal Personal Patterns			
SS	Vtpred	SS	Sadv
SS	Vtpred	SS	
SS	Vtpred	Sadj	
SS	Vtpred	SP	
Impersonal Nominal Patterns			
	Vlig	SS	
	Vlig	Sadj	
	Vlig	Sadv	
	Vlig	SP	
Impersonal Verbal Patterns			
	Vtd	SS	
	Vti	SP	
	Vi		

4. Specification of Grammar Changes.

5. Specification of Semantics.

The first two steps are to prepare the Luft's Grammar so that it can be processed computationally. The third step corresponds to the presentation of the adaptive model itself. The fourth step presents the notation used to make changes in the grammar rules and the last step exemplifies the use of adaptive grammar in semantic analysis.

4.1 Consolidation of Celso Luft's Sentence Patterns

This proposal begins with the consolidation of the standards presented by Luft, unifying the types of patterns, in order to simplify computational processing. For example, instead of using SS1 and SS2 to indicate noun phrases, only a single SS symbol will be used, which can be repeated according to the sentence pattern to which it belongs. The consolidated patterns are presented in Table 2.

4.2 Standardization of Production Rules

The production rules of the Celso Luft's Grammar have to be standardized to be used computationally. This work uses a model proposed Neto [17] with some modifications to refine the phrase structures. The productions are as follows:

S → [con] [PrRel] [SS] SV pont [S]

– S: Sentence

– PrRel: Relative pronoun

– con: Conjunction or relative pronoun,

– SS: Noun phrase,

– SV: Verb frase,

– pont: punctuation.

SS → [Sadj] SS [Sadj | SP | S] | S

– SS: Noun frase,

– Sadv: Adverbial frase,

– Sadj: Adjective frase,

– SP: Prepositional frase,

– S: Sentence.

SS → (([num | PrA] Sc) | Sp | PrPes)

– SS: Noun frase,

– num: numeral,

– PrA: Adjective pronoun,

– Sc: Common noun,

– Sp: Proper noun,

– PrPes: Personal Pronoun.

SV → [Neg] [Aux | PreV] V [SS | Sadj | Sadv | SP] | ((SS SP) | (SS Sadj) | (SP SP) | (SS SS) | (SS Sadv) | (SS SP SP))

– Neg: negation particle,

– Aux: Auxiliar passive voice particle,

– PreV: Pre-verbal,

– V: Vlig|Vtda|Vtdna|Vtdi, |Vtpred |Vti

– SS: Noun phrase,

– Sadv: Adverbial phrase,

– Sadj: Adjective phrase,

– SV: Verb frase,

– SP: Preposicional frase,

– SS: Noun phrase.

V → (Vlig | Vtda | Vtdna | Vtdi | Vtpred |Vti |Vi)

– V: Verb,

– Vlig: Linking verb,

– Vtda: Transitive verb convertible to passive voice,

– Vtdna: direct transitive verb not convertible to passive voice,

– Vtdi: Direct and indirect transitive verb,

– Vtpred: Predicative verb,

– Vti: Indirect transitive verb,

– Vi: Intransitive verb.

SP → Prep (SS | Sadj) | S

– SP: Preposicional phrase,

– Prep: preposition,

– SS: Noun phrase,

– Sadj: Adjective frase,

– S: Sentence.

Sadj → Sadj [SP] [S] | S

– Sadj: Sintagma adjetivo,

– SP: Sintagma preposicional,

– S: Sentença.

Sadj → [Adv] Adj [S] | S

– Sadj: Adjective frase,

– Adv: Adverb,

– Adj: Adjecive,

– S: Sentence.

Sadv → Sadv [SP] [S] | S

– Sadv: Adverbial frase,

– SP: Preposicional frase,

– S: Sentence.

Sadv → [Adv] Adv [S] | S

– Sadv: Adverbial frase,

– Adv: Adverb,

– S: Sentence.

PrA → [PrA] (Ind | ArtDef | ArtInd| Dem| Pos)

– PrA: Adjective pronoun,

– Ind: Indefinite pronoun,

– ArtDef: definite article,

– ArtInd: indefinite article,

– Dem: Indefinite demonstrative pronoun.

4.3 Specification of the Adaptive Grammar

The grammar is specified according to the production rules defined in item 4.2 and based on the Portuguese Language lexicon, being as follows:

$G = (G^0, T, R^0)$, where:
$G^0 = (V_N^0, V_T, V_C, P_L^0, P_D^0, S)$,
V_N^0 = { S, con, SS, SV, SS, Sadj, SP,num, PrA, Sc, Sp, PrPes, Neg, Aux, PreV,V, Vlig ,Vtda,Vtdna , Vtdi, Vtpred ,Vti ,Vi, Sadv, Prep, Ind, ArtDef, ArtInd, Dem, Pos, PrRel, Adv, Adj, pont },
V_T = {{lexicon},{,}, ; , . ,:,!,?,1, 2, 3, 4, 5, 6, 7,8, 9, 0},
P_L^0 = { S → [con] [PrRel] [SS] SV pont [S]
SS → [Sadj] SS [Sadj | SP] [S] | S
SS → (([num | PrA] Sc) | Sp | PrPes)
SV→ [Neg] [Aux | PreV] V [SS | Sadj | Sadv | SP] | ((SS SP) | (SS Sadj) | (SP SP) | (SS SS) | (SS Sadv) | (SS SP SP))
V → (Vlig | Vtda | Vtdna | Vtdi | Vtpred |Vti |Vi)
SP → Prep (SS | Sadj)
Sadj → Sadj [SP] [S] | S
Sadj → [Adv] Adj [S]

Sadv → Sadv [SP] [S] | S
Sadv → [Adv] Adv [S]
PrA → [PrA] (Ind | ArtDef | ArtInd| Dem| Pos)
con → (aditiva | adversativa | alternativa | conclusiva| explicativa | integrante | causal | comparativa | concessiva | condicional | conformativa | consecutiva | final | proporcional | temporal)[1]
num→ (0 |1 |2 | 3 |4 |5 |6 |7 |8 |9)| (cardinal| ordinal| multiplicativo| fracionario)[2]
pont → (,| ; | . |: |! |?)
},

$V_C = \varnothing$,
$P_{D0} = \varnothing$.

With the normalization of the grammar G, the new grammar G'= (G^{0}', T', R^{0}') is defined as follows:

G^{0}' = (V_N^{0}', V_T', V_C', P_L^{0}', P_D^{0}', S),
V_N^{0}' = V_N^{0}, V_T' = V_T, V_C'= {δ , χ}.

P_L^{0}'∪ P_D^{0}' formed by the productions obtained with the normalization.

The normalization of the production S → [con|PrRel][SS] SV pont [S] generates:

$1.^0$ SV → SV pont	$5.^0$ S → SS S_1
$2.^0$ S → SV	$6.^0$ S1 → SV S
$3.^0$ S → SV S	$7.^0$ S → con S
$4.^0$ S → SS SV	$8.^0$ S → PrRel S

In this case, prior to the application of the Iwai standardization rules, is added the rule SV → SV pont, that incorporates the punctuation to the verbal phrase, since in the Grammar of Luft there is no production rules involving the punctuation.

The normalization of the production SS → [Sadj] SS [Sadj | SP | S] | S, generates:

$9.^0$ SS → SS	$13.^0$ SS → Sadj SS	$17.^0$ SS2 → SS SP
$10.^0$ SS → SS Sadj	$14.^0$ SS → Sadj SS_1	$18.^0$ SS→ Sadj SS_3
$11.^0$ SS → SS SP	$15.^0$ SS_1 → SS Sadj	$19.^0$ SS3 → SS S
$12.^0$ SS→ SS S	$16.^0$ SS → Sadj SS_2	$20.^0$ SS → S

The normalization of the production SS → (([num | PrA] Sc) | Sp | PrPes), generates:

$21.^0$ SS → Sc
$22.^0$ SS → Sp
$23.^0$ SS → PrPes
$24.^0$ SS→ num Sc
$25.^0$ SS → PrA Sc

The normalization of the production SV→ [Neg] [Aux | PreV] V [SS | Sadj | Sadv | SP] | ((SS SP) | (SS Sadj) | (SP SP) | (SS SS) | (SS Sadv) | (SS SP SP)), generates:

[1] set of conjunction types of Portuguese Language
[2] set of numerical types of Portuguese Language

Simple Complements:

26.^{0} SV → V	40.^{0} SV → Aux V	53.^{0} SV → Neg SV_7
27.^{0} SV → V SS	41.^{0} SV → Aux SV_1	54.^{0} SV_7 → Aux SV_3
28.^{0} SV → V Sadj	42.^{0} SV → Aux SV_2	55.^{0} SV → Neg
29.^{0} SV → V Sadv	43.^{0} SV → Aux SV_3	56.^{0} SV_8 → Aux SV_4
30.^{0} SV → V SP	44.^{0} SV → Aux SV_4	57.^{0} SV → Neg SV_9
31.^{0} SV → Neg V	45.^{0} SV → PreV V	58.^{0} SV_9 → PreV V
32.^{0} SV → Neg SV_1	46.^{0} SV → PreV SV_1	59.^{0} SV → Neg SV_{10}
33.^{0} SV_1 → V SS	47.^{0} SV → PreV SV_2	60.^{0} SV_{10} → PreV SV_1
34.^{0} SV → Neg SV_2	48.^{0} SV → PreV SV_3	61.^{0} SV → Neg SV_{11}
35.^{0} SV_2 → V Sadj	49.^{0} SV → PreV SV_4	62.^{0} SV_{11} → PreV SV_2
36.^{0} SV → Neg SV_3	50.^{0} SV → Neg SV_5	63.^{0} SV → Neg SV_{12}
37.^{0} SV_3 → V Sadv	51.^{0} SV_5 → Aux V	64.^{0} SV_{12} → PreV SV_3
38.^{0} SV → Neg SV_4	51.^{0} SV → Neg SV_6	65.^{0} SV → Neg SV_{13}
39.^{0} SV_4 → V SP	52.^{0} SV_6 → Aux SV_1	66.^{0} SV_{13} → PreV SV_4

Compound Complements:

67.^{0} SV → V (SS SP)	85.^{0} SV → Aux SV_{14}	103.^{0} SV → Neg SV_{23}
68.^{0} SV → V (SS Sadj)	86.^{0} SV → Aux SV_{15}	104.^{0} SV23 → Aux SV_{17}
69.^{0} SV → V (SP SP)	87.^{0} SV → Aux SV_{16}	105.^{0} SV → Neg SV_{24}
70.^{0} SV → V (SS SS)	88.^{0} SV → Aux SV_{17}	106.^{0} SV_{24} → Aux SV_{18}
71.^{0} SV → V (SS Sadv)	89.^{0} SV → Aux SV_{18}	107.^{0} SV → Neg SV_{25}
72.^{0} SV → V (SS SP SP)	90.^{0} SV → Aux SV_{19}	108.^{0} SV_{25} → Aux SV_{19}
73.^{0} SV → Neg SV_{14}	91.^{0} SV → PreV SV_{14}	109.^{0} SV → Neg SV_{26}
74.^{0} SV_{14} → V (SS SP)	92.^{0} SV → PreV SV_{15}	110.^{0} SV_{26} → PreV SV_{14}
75.^{0} SV → Neg SV_{15}	93.^{0} SV → PreV SV_{16}	111.^{0} SV → Neg SV_{27}
76.^{0} SV_{15} → V (SS Sadj)	94.^{0} SV → PreV SV_{17}	112.^{0} SV_{27} → PreV SV_{15}
77.^{0} SV → Neg SV_{16}	95.^{0} SV → PreV SV_{18}	113.^{0} SV → Neg SV_{28}
78.^{0} SV_{16} → V (SP SP)	96.^{0} SV → PreV SV_{19}	114.^{0} SV_{28} → PreV SV_{16}
79.^{0} SV → Neg SV_{17}	97.^{0} SV → Neg SV_{20}	115.^{0} SV → Neg SV_{29}
80.^{0} SV_{17} → V (SS SS)	98.^{0} SV_{20} → Aux SV_{14}	116.^{0} SV_{29} → PreV SV_{17}
81.^{0} SV → Neg SV_{18}	99.^{0} SV → Neg SV_{21}	117.^{0} SV → Neg SV_{30}
82.^{0} SV_{18} → V (SS Sadv)	100.^{0} SV_{21} → Aux SV_{15}	118.^{0} SV_{30} → PreV SV_{18}
83.^{0} SV → Neg SV_{19}	101.^{0} SV → Neg SV_{22}	119.^{0} SV → Neg SV_{31}
84.^{0} SV_{19} → V (SS SP SP)	102.^{0} SV_{22} → Aux SV_{16}	120.^{0} SV_{31} → PreV SV_{19}

The normalization of the production SP → Prep (SS | Sadj), generates:

121.^{0} SP → Prep SS
122.^{0} SP → Prep Sadj

The normalization of the production Sadj → Sadj [SP] [S] \| S, generates:

123.^{0} Sadj → Sadj
124.^{0} Sadj → Sadj SP
125.^{0} Sadj → Sadj S
126.^{0} Sadj → Sadj $Sadj_1$
127.^{0} $Sadj_1$ → SP S
128.^{0} Sadj → S

The normalization of the production Sadj → [Adv] Adj [S] , generates:

129.^{0} Sadj → Adj
130.^{0} Sadj → Adv Adj
131.^{0} Sadj → Adv $Sadj_2$
132.^{0} $Sadj_2$ → Adj S
133.^{0} Sadj → Adj S

The normalization of the production Sadv → Sadv [SP] [S] | S, generates:

134.^{0} Sadv → Sadv
135.^{0} Sadv → Sadv SP
136.^{0} Sadv → Sadv $Sadv_1$
137.^{0} $Sadv_1$ → SP S
138.^{0} Sadv → Sadv S
139.^{0} Sadv → S

The normalization of the production Sadv → [Adv] Adv [S], generates:

140.^{0} Sadv → Adv
141.^{0} Sadv → Adv Adv
142.^{0} Sadv → Adv $Sadv_2$
143.^{0} $Sadv_2$ → Adv S

The normalization of the production V → ({A(t)}Vlig | {A(t)}Vtda | {A(t)}Vtdna | {A(t)}Vtdi | {A(t)}Vtpred |{A(t)}Vti|{A(t)}Vi), generates:

44.^{0} V →{A(Vlig)}V	148.^{0} V → {A(Vtpred)}V
145.^{0} V → {A(Vtda)}V	149.^{0} V →{A(Vti)}V
146.^{0} V → {A(Vtdna)}V	150.^{0} V → {A(Vi)}V
147.^{0} V → {A(Vtdi)}V	

The normalization of the production PrA → [PrA] (Ind | ArtDef | ArtInd| Dem| Pos), generates:

151.^{0} PrA → Ind	156.^{0} PrA → PrA Ind
152.^{0} PrA → ArtDef	157.^{0} PrA → PrA ArtDef
153.^{0} PrA → ArtInd	158.^{0} PrA → PrA ArtInd
154.^{0} PrA → Dem	159.^{0} PrA → PrA Dem
155.^{0} PrA → Pos	160.^{0} PrA → PrA Pos

The normalization of the production con → (aditiva | adversativa | alternativa | conclusiva| explicativa|integrante | causal | comparativa | concessiva | condicional | conformativa | consecutiva | final | proporcional | temporal), generates:

con →{B(cAditiva)}con	168.^{0} con → {B(sConcessiva)} con
162.^{0} con → {B(cAdversativa)} con	169.^{0} con → {B(sCondicional)} con
163.^{0} con →{B(cAalternativa)} con	170.^{0} con → {B(sConformativa)} con
164.^{0} con → {B(cExplicativa)} con	171.^{0} con → {B(sConsecutiva)} con
165.^{0} con → {B(sIntegrante)} con	172.^{0} con → {B(sFinal)} con
166.^{0} con → {B(sCausal)} con	173.^{0} con → {B(sProporcional)} con
167.^{0} con → {B(sComparativa)} con	174.^{0} con → {B(sTemporal)} con

The normalization of the production num→ (0 |1 |2 | 3 |4 |5 |6 |7 |8 |9)| (cardinal | ordinal | multiplicativo | fracionario), generates:

175.^{0} num → 0	182.^{0} num → 5
176.^{0} num → 1	183.^{0} num → 6
177.^{0} num → 2	184.^{0} num → 7
178.^{0} num → 3	185.^{0} num → 8
179.^{0} num → 4	186.^{0} num → 9
180.^{0} num → cardinal	187.^{0} num → ordinal
181.^{0} num → multiplicativo	188.^{0} num → fracionario

The normalization of the production pont → (,| ; | . |: |! |?), generates:

189.^{0} pont → ,	192.^{0} pont → :
190.^{0} pont → ;	193.^{0} pont → !
191.^{0} pont → .	194.^{0} pont → ?

The normalization of the production PrRel → (substantivo | adjetivo| adverbio), generates:

195.^{0} PrRel → {C(PrRelSubs)} PrRel
196.^{0} PrRel → {C(PrRelAdj)} PrRel
197.^{0} PrRel → {C(PrRelAdv)} PrRel

The other non-terminals generate production rules based on the classification of the Portuguese Language lexicon.

4.4 Specification of Grammar Changes

Adaptive actions are used to modify the rules of production according to the context defined by the type of verb found in the text and by the presence of conjunctions or relative pronouns. However, for these modifications to be made, it is necessary to exclude and include a very large number of production rules. In view of this need, Iwai's notation has been extended with the instruction R, which means replace, i.e. exchange of a set of rules from the previous grammar by a new set of rules.
The syntax of the R statement is as follows:

$$R: <_{G}{}^{n}> <P_{I-F}> : <_{G}{}^{n+1}> P',$$

where:

R: replace

$_{G}{}^{n}$: Grammar prior to the update

P_{I-F}: Production rules for updating, where I = Interval start and F = interval end. It is allowed to add more than one rule set by using the comma for separating

$_{G}{}^{n+1}$: Grammar posterior to the update

P': new production rule

∅ : Symbol indicating that the rules set out in P_{I-F} will be removed

The following examples show the use of R instruction in an adaptive action A.

$$T = \{ \quad \mathtt{A}\ (t{=}Vlig) = \mathtt{F}\ (t) = \{ \quad + [R: {}_{G}{}^{n} P_{26\text{-}66:}\ {}_{G}{}^{n+1}V \leftarrow \chi V] \\ + [{}_{G\ n+1}\chi V \rightarrow Vlig] \\ +[R: {}_{G}{}^{n} P_{67\text{-}120:}\ {}_{G}{}^{n+1}\varnothing],$$

$$T = \{ \quad B\ (p = cAditiva) = B\ (p) = \{ \quad + [R: {}_{G}{}^{n} P_{7}: {}_{G}{}^{n+1}\ con \leftarrow \delta con] \\ + [G\ n{+}1\ \delta con \rightarrow cAditiva] \\ + [R: {}_{G}{}^{n} P_{7}: {}_{G}{}^{n+1}\ \varnothing].$$

4.5 Semantics

The formalism presented by Iwai provides support to include additional features to solve problems where context is important for selection of production rules. For example, the notation allows you to check whether the analyzed sentence is correct or to resolve possible ambiguities if you have information about the type of verb, time, mode and person of the analyzed verb. In the example below, the adaptive action A uses as input parameters the type identification of the verb, mode and person to define the adaptive functions:

$$T = \{ \quad \mathtt{A}\ (\text{verb=Vlig, time = Present, mode = Singular, person = First}) = \\ \mathtt{F}\ (\text{verb, time, mode, person}) = \{ \quad + [R: {}_{G}{}^{n} P_{i\text{-}f:}\ {}_{G}{}^{n+1}V \leftarrow \chi V \\ + [{}_{G}{}^{n+1}\chi V \rightarrow V_{lig,\ Present,\ Singular,\ First}] \\ + [R: {}_{G}{}^{n} P_{i\text{-}f:\ G}{}^{n+1}\varnothing] \qquad \}.$$

Adaptive formalism was used to add contextual features to grammar, which would not be possible using only context-free grammar. Analogously, it is possible to include rules for analysis of nominal agreement or any other type of semantic content, without the need to use any element other than formalism. Adaptive grammars can also be used to represent the use of probabilistic information in the selection of production rules. Such a technique is used when there is more than one applicable production rule and there is insufficient syntactic and semantic information to disambiguate them.

In this case, it is possible to use the probability of occurrence of the rules as a choice factor and the formalism of Iwai can be used to represent this type of contextual information. In the example below, the adaptive action A uses as input parameter probability of occurrence of the evaluated production rules and an indication to use the rule of maximum probability:

$$SV_{prob} \rightarrow V\ SP\ 10\%, \\ SV_{prob} \rightarrow V\ (SS\ SP)\ 90\%, \\ T = \{ \quad \mathtt{A}\ (t{=}SV,\ u = prob,\ v{=}max) = \\ \mathtt{F}\ (SV, prob, max) = \{ + [R: {}_{G}{}^{n} P_{inicio\text{-}fim}: {}_{G}{}^{n+1}SV \leftarrow \chi SV_{prob}] \\ + [{}_{G}{}^{n+1}\chi SV_{prob} \rightarrow \chi SV_{max}] \\ ?\ [{}_{G}{}^{n+1}\chi SV_{max}] \\ + [{}_{G}{}^{n+1}\chi SV_{max} \rightarrow V\ (SS\ SP)] \\ + [R: {}_{G}{}^{n} P_{inicio\text{-}fim}: {}_{G}{}^{n+1}\varnothing]. \qquad \}.$$

5 Conclusions

Adaptivity extends the capabilities of conventional grammars, providing syntactic and semantic representation power in a single device. The formality presented by Margarete Iwai structures this knowledge and allows it to be applied to natural language grammars. This work presented a method for modeling natural languages using Adaptive Grammars and illustrates the proposal with an application to Brazilian Portuguese, based on Modern Brazilian Grammar of Celso Luft. Luft's production patterns were formatted in the Iwai model and adaptive actions were created for the generation of coordinated and subordinate sentences, with validation of Luft's proposed sentence patterns as a function of the context of the analysis.

Due to the amount of production rules used by the Luft grammar, a new adaptive action, the R instruction, was introduced, replacing a set of rules from the previous grammar with a new set of rules, allowing a large set of production rules to be updated with a single instruction. The semantics representation was exemplified with an ambiguity problem, in which context was important to choose the most adequate production. Finally, the robustness of the model was proven with an example in which it was included probabilistic information in the selection of production rules.

6 Future Works

The intention is to continue this research through the development of several aspects not considered in the scope of this work. An example is the syntactic patterns analyzed before the verb of the sentence, which can be changed the moment the type of verb is identified. Another example is the verification of the possibility of inversion of the sentence patterns, also depending on the type of the verb. Finally, it is intended to evaluate the application of nominal and verbal regency criteria to verify the adequacy of the sentence analyzed to the cult pattern of the Portuguese Language.

References

1. Iwai, M. K.: A gramatical formulation for context-dependent languages. Escola Politécnica da Universidade de São Paulo. Tese de Doutorado (2000)
2. Neto, J. J., Moraes, M. D.: Using adaptive formalisms to describe context-dependencies in natural language. In: International Workshop on Computational Processing of the Portuguese Language, Springer, vol. 2721, pp. 94–97 (2003) doi: 10.1007/3-540-45011-4_14
3. Rich, E., Knight, K., Shivashankar, B. N.: Artificial intelligence (chapter natural language processing), 3rd Edition, Tata McGraw-Hill, (2009)
4. Vieira, R., Lima, V. L. S.: Linguística computacional: princípios e aplicações. IX Escola de Informática da SBC–Sul (2001)
5. Fuchs, C., Le-Goffic, P.: Initiation aux problèmes des linguistiques contemporaines. Hachette Université (1975)
6. Nunes, M. D. G. V., Pardo, T. A.: Introdução ao Processamento das Línguas Naturais. Notas didáticas do ICMC, Universidade São Paulo, vol. 180 (1999)
7. Shutt, J. N.: What is an Adaptive Grammar? (2001)
8. Christiansen, H.: A survey of adaptable grammars. ACM SIGPLAN Notices, vol. 25, no. 11, pp. 35–44 (1990)

9. Shutt, J. N.: Recursive adaptable grammars. Master's Thesis, Worcester Polytechnic Institute (1993)
10. Jackson, Q. T.: Adapting to babel: adaptivity and context-sensitivity in parsing. Ibis Publications (2006)
11. Di-Forino, A. C.: Some remarks on the syntax of symbolic programming languages. Communications of the ACM, vol. 6, no. 8, pp. 456–460 (1963)
12. Wegbreit, B.: Studies in extensible programming languages, ESD-TR-70-297, Harvard University Cambridge, Massachusetts (1970)
13. Hanford, K. V., Jones, C. B.: Dynamic syntax: A concept for the definition of the syntax of programming languages. Annual Review in Automatic Programming, Pergamon Press, pp. 115–142 (1973)
14. Neto, J. J.: Adaptive automata for context-sensitive languages. ACM SIGPLAN Notices, vol. 29, no. 9, pp. 115–124 (1994)
15. Neto, J. J.: Adaptive rule-driven devices - general formulation and case study. Lecture Notes in Computer Science. In: Implementation and Application of Automata 6th International Conference, CIAA'01, Springer, vol. 2494, pp. 234–250 (2001)
16. Luft, C.: Moderna gramática brasileira. 2ª. Edição Revista e Atualizada, Editora Globo (2002)
17. Neto, J. J.: Adaptive handling of some linguistic phenomena. Available at: https://sites.google.com/site/2015pcs5004/material-para-download