UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

PROTÓTIPO DE UM ANALISADOR SINTÁTICO-SEMÂNTICO PARCIAL DE ORAÇÕES EM LÍNGUA PORTUGUESA - SIMPLES

TRABALHO DE GRADUAÇÃO

Édison Josias Quaiatto

Santa Maria, RS, Brasil

2011

PROJETO DE UM ANALISADOR SINTÁTICO-SEMÂNTICO PARCIAL DE ORAÇÕES EM LÍNGUA PORTUGUESA – SIMPLES

por

Édison Josias Quaiatto

Trabalho de Graduação apresentado ao Curso de Ciência da Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para a obtenção do grau de
Bacharel em Ciência da Computação

Orientador: Prof. Dr. Giovani Rubert Librelotto

Trabalho de Graduação N. 303

Santa Maria, RS, Brasil

2011

Universidade Federal de Santa Maria
Centro de Tecnologia
Curso de Graduação em Ciência da Computação

A Comissão Examinadora, abaixo assinada,
aprova o Trabalho de Graduação

PROTÓTIPO DE UM ANALISADOR SINTÁTICO E SEMÂNTICO PARCIAL DE ORAÇÕES EM LÍNGUA PORTUGUESA – SIMPLES

elaborado por
Édison Josias Quaiatto

como requisito parcial para obtenção do grau de
Bacharel em Ciência da Computação

COMISSÃO EXAMINADORA:

Prof. Dr. Giovani Rubert Librelotto
(Presidente/Orientador)

Profª. Dra. Juliana Kaizer Vizzotto (UFSM)

Prof. Msc. Marcos Luis Cassal (UFSM)

Santa Maria, 30 de agosto de 2011.

Ao meu pai (in memoriam) e à minha mãe,
que um dia deixaram o campo para “dar estudo aos filhos”.

Dedico também a cada um dos meus irmãos.

AGRADECIMENTOS

Agradeço a atenção e a paciência da Janice, secretária do curso, pela excelente pessoa e profissional que é, sempre atendendo bem os discentes.

Agradeço a minha banca, professor Cassal e professora Vizzotto, por aceitarem o convite e ao meu orientador Librelotto, que teve a paciência e a confiança em mim necessárias para que esse projeto pudesse ser concluido.

Agradeço também a todos os professores do curso, não só pelos ensinamentos, mas também pela paciência que tiveram comigo, principalmente Iara, Benhur e Ceretta. Obrigado a todos mesmo.

Agradeço muitíssimo a todos os meus amigos e a minha família, especialmente minha esposa, Iranir, e meus três filhos: Rittieli, Josiéli e Jonathan. Vocês sempre serão minha prioridade. Amo vocês demais.

E agradeço, sobretudo, a Deus, por me proporcionar a convivência com todas essas pessoas magníficas citadas, direta ou indiretamente, nos parágrafos acima.

“O valor de um trabalho científico se avalia não apenas em termos dos problemas que resolve, mas também em termos de questões que levanta.”

Mário A. Perini

RESUMO

Trabalho de Graduação
Curso de Ciência da Computação
Universidade Federal de Santa Maria

PROTÓTIPO DE UM ANALISADOR SINTÁTICO-SEMÂNTICO PARCIAL DE ORAÇÕES EM LÍNGUA PORTUGUESA - SIMPLES

Autor: Édison Josias Quaiatto
Orientador: Prof. Dr. Giovani Rubert Librelotto

O presente projeto apresenta um protótipo de um analisador parcial de orações em Língua Portuguesa. A implementação do projeto é em Prolog e o protótipo ganhou o nome de SIMPLES por reconhecer as orações com somente um núcleo verbal – as orações simples.

Para reconhecer as orações como válidas, o *parser* considera não só os aspectos sintáticos das sentenças analisadas, como também alguns traços semânticos. Quanto à validação de aspectos sintáticos, a base teórica principal utilizada foi a Gramática Sintagmática (GS). Já a teoria aplicada na modelagem dos aspectos semânticos foi basicamente a Teoria dos Papéis Temáticos (TPT).

De um modo geral, esse projeto propõe uma linha de pesquisa investigativa para a área de Processamento de Linguagem Natural (PLN): a identificação de teorias linguísticas com potencial de contribuir no aprimoramento da modelagem computacional das linguagens naturais.

De um modo específico, o SIMPLES explora a aplicação prática computacional dos conceitos teóricos de papel temático, construção e estrutura sintagmática, com a finalidade de implementar um *parser* com fragmentos de inteligência semântica no reconhecimento de linguagem natural.

Palavras-chave: Processamento de Linguagem Natural. Analisador Sintático. Linguística Computacional. Papéis Temáticos. Construções Sentenciais.

ABSTRACT

Undergraduate Final Work
Undergraduate Program in Computer Science
Universidade Federal de Santa Maria

PROTOTYPE OF A SEMANTIC PARSING PARTIAL PHRASES OF PORTUGUESE - SIMPLES

Author: Édison Josias Quaiatto
Advisor: Giovani Rubert Librelotto, Dr.

This project presents a prototype of a partial analysis of phrases in Portuguese. The project implementation is in Prolog and the prototype was named SIMPLES to recognize the phrases with one, and only one, verbal core - the simple phrases.

In order to recognize as valid the phrases, the parser considers not only the syntactic sentences analyzed, as well as some semantic features. For syntactic validation, the main theoretical basis used was the Grammar Syntagmatic (GS). Yet the theory applied in the modeling of semantic aspects was basically a theory of thematic roles (TTR).

Overall, this project proposes a line of investigative research for the area of Natural Language Processing (NLP): the identification of linguistic theories with the potential to contribute in improving the computational modeling of natural languages.

In a specific way, the SIMPLES explores the computacional practical application of theoretical concepts of thematic role, construction and syntagmatic structure, in order to implement a parser with fragments of semantic intelligence in the recognition of natural language.

Keywords: Natural Language Processing. Parser. Computational Linguistics.Thematic Roles. Phrasal Constructions.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

CS	Construções Sentenciais
DP	Determinantes Primários
DCG	Definite Clause Grammar
Det	Determinante
ES	Estrutura Sintagmática
GGT	Gramática Gerativa Transformacional
GS	Gramática Sintagmática
GT	Gramática Tradicional
ISO	*International Organization for Standard*
LC	Linguística Computacional
LM	Lingüística Moderna
LN	Linguagem Natural
LP	Língua Portuguesa
Mod	Modificador
N	Nominal
NGB	Norma Gramatical Brasileira
NSA	Núcleo do Sintagma Adjetival
NSN	Núcleo do Sintagma Nominal
PB	Português Brasileiro
PLN	Processamento de Linguagem Natural
PPCR	Pronome Pessoal do Caso Reto
PN	Pré-Núcleo
SAdj	Sintagma Adjetival
SAdv	Sintagma Adverbial
SN	Sintagma Nominal
SP	Sintagma Preposicional
SV	Sintagma Verbal
TPT	Teoria dos Papéis Temáticos
V	Verbo

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

Os estudos teóricos sobre a linguagem tiveram origem ainda na Grécia Antiga (Azeredo, 2007). Até meados do século XIX, esses estudos estavam fortemente relacionados às áreas de Filosofia e Literatura (Lyons,1970). Tinham um caráter prescricional da linguagem literária culta padrão[1]. Apesar de pouco científicas, as teorias linguísticas dessa época influenciam até os dias atuais e formam uma base teórica sólida hoje conhecida como Gramática Tradicional (GT)[2].

A partir da segunda metade do século XIX, começaram a ocorrer pesquisas linguísticas visando o estudo do fenômeno da linguagem natural (LN) com um caráter mais descritivo e científico[3]. Seguindo essa nova diretriz, surgiram novas teorias e descrições da linguagem, muitas confrontando diretamente os dogmas da GT. Essa nova fase do estudo das linguagens é conhecida como Lingüística Moderna (LM) [4].

Foi a partir da década de 50 que a modelagem do conhecimento das línguas naturais começou a despertar interesse também entre pesquisadores ligados à Computação (Dias-da-Silva, 1996). Com a finalidade de construir um modelo matemático das estruturas linguísticas, de modo a tornar possível a automação do processamento de linguagem através do computador, surgia a Linguística Computacional (LC), um subconjunto de duas grandes áreas, a Linguística e a Ciência da Computação (Nugues, 2006, p.1).

Processamento de Linguagem Natural (PLN) é um termo muitas vezes utilizado como sinônimo de LC, embora, em sua origem, seja um termo genérico que se refere a

---

1 Segundo Lyons (1970), os primeiros gramáticos ocidentais estavam preocupados quase que exclusivamente com a preservação e a interpretação de textos de escritores gregos clássicos (Lyons, 1970, p.20).

2 Azeredo (2007).

3 Nessa fase, como consequência do seu novo *status* científico, a Linguística atingiu sua *autonomia*, ou independência, de outras disciplinas (Lyons, 1970, p.19).

4 O interesse da LM é modelar o que já existe (caráter descritivo). Já a GT visa resguardar as escritas antigas, considerando-as como parâmetro do correto, sendo incorreto tudo que as contraria (caráter prescritivo). Reside-se aí o fato de os conceitos de LM serem mais aplicáveis computacionalmente, pois, de alguma forma, ela já se preocupa com a modelagem da língua.

um amplo espectro de aplicações englobando o tratamento computacional das línguas naturais (Nugues, 2006, p.1); em outras palavras, dentro dessa concepção a LC seria uma subárea do PLN.

Os primeiros cientistas da computação interessados em PLN acreditavam que poderiam prescindir das teorias linguísticas em suas implementações; no entanto, os péssimos resultados das primeiras implementações em PLN obrigaram os pesquisadores a buscar algum embasamento teórico em Linguística para seus trabalhos (Dias-da-Silva, 1996).

O problema é que a má vontade dos pesquisadores de uma área em relação à outra (Linguística e Computação) era recíproca: se os projetistas de sistemas voltados ao PLN desprezavam o estudo de teorias linguísticas, os pesquisadores de Linguística não estavam nem um pouco interessados em aplicações computacionais dos estudos da linguagem (Dias-da-Silva, 2006)[5]. Nesse contexto, os projetistas de LC utilizavam as teorias linguísticas existentes para suas aplicações o mínimo possível, mantendo o foco principal das suas pesquisas em lógica e cálculos matemáticos (ibid).

Nesse projeto, a proposta é superar as dificuldades históricas de relacionamento entre essas duas áreas, ao menos sob o ponto de vista da Computação em relação à área de Linguística.

Com esse pensamento, decidiu-se investigar as potencialidades ainda pouco exploradas para aplicações em LC de algumas teorias linguísticas.

Esse trabalho investigativo resultou num protótipo de *parser*, doravante chamado de SIMPLES, que é basicamente o resultado da aplicação de três teorias linguísticas, entre outras, na modelagem da LN para fins computacionais: Gramática Sintagmática (GS), Teoria dos Papéis Temáticos (TPT) e Construções.

---

[5] Dias-da-Silva (2006), frisando o hiato ainda existente entre as áreas de Ciência da Computação e Linguística, escreve em seu artigo:

> ...O agravante é constatar que, por razões diversas, os complexos fenômenos da linguagem, cuja compreensão é, sem dúvida, condição essencial para o sucesso de qualquer empreendimento em PLN, têm sido subestimados e, consequentemente, subdimensionados no processo de desenvolvimento dos mais variados tipos de sistemas computacionais que, de alguma forma, envolvem o tratamento automático de entidades e processos linguísticos (Dias-da-Silva 2006, p. 105).

1.1 Objetivos Gerais e Específicos

Com esse projeto, busca-se investigar a aplicabilidade computacional de pesquisas teóricas do campo da Linguística.

Como objetivo geral, esse projeto ambiciona contribuir com a área de LC procurando nas teorias linguísticas aquelas que possam auxiliar na criação de um modelo computacional de linguagem natural mais completo, que contemple não só aspectos léxicos, como também aspectos sintáticos, lógicos e semânticos da LN.

Esse trabalho tem, como objetivo específico, implementar em linguagem Prolog, um protótipo de um analisador sintático-semântico parcial – o SIMPLES, que explora computacionalmente os conceitos teóricos de construção, de estrutura sintagmática e de papel temático, entre outros.

1.2 Contribuição

A principal contribuição desse projeto é oferecer aos pesquisadores de PLN um protótipo de um analisador sintático-semântico parcial de LN resultado da aplicação de conceitos modernos de Linguística, como papel temático e construção. A ampliação da exploração desses conceitos, especialmente o de construção, pode resultar num analisador que reconheça todas as construções de sentenças de uma língua.

1.3 Estrutura do Texto

O texto está estruturado da seguinte forma:

No capítulo 2, é traçado um panorama dos principais desafios enfrentados em PLN a fim de contextualizar o projeto. Em seguida, é definido o escopo do projeto, demarcando a área do espaço-problema abrangida no presente trabalho.

Já no capítulo 3, é apresentada a revisão bibliográfica. Primeiro, faz-se uma análise de algumas implementações relacionadas ao projeto em LC. Depois, apresentam-se os conceitos teóricos linguísticos que serviram de base para a implementação do SIMPLES.

No capítulo 4, é destacado o poder de expressividade para programação em PLN da linguagem Prolog, escolhida para o desenvolvimento do protótipo desse projeto.

No capítulo 5, encontra-se a descrição da gramática desenvolvida para o SIMPLES: o léxico, os sintagmas, as regras de produções e as soluções específicas adotadas para determinados *tradeoffs*. Excertos do código em Prolog são mostrados, pontualmente, para exemplificar conceitos.

O capítulo 6 especifica o protótipo e os testes realizados.

Finalmente, o capítulo 7 trata da conclusão e aponta caminhos para trabalhos futuros.

2 DELINEAMENTO DO ESCOPO DO PROTÓTIPO

Tratar de todos os problemas de uma área do conhecimento abrangente como a área de PLN em um único projeto é uma luta inglória. Por outro lado, pesquisar soluções específicas para uma área delimitada dentro de um espaço-problema, sem ter uma visão geral do todo, dificulta a integração do resultado dessas pesquisas com outras soluções encontradas dentro de outras áreas envolvendo o mesmo espaço-problema.

Por esse motivo, é apresentada nessa seção uma visão geral de PLN, com uma análise superficial de alguns pontos que serão enfrentados, de alguma forma, nesse projeto; depois dessa análise, é apresentado o escopo do SIMPLES[6].

2.1 Os diferentes domínios do PLN – visão geral

Historicamente, a Linguística se subdivide em diferentes disciplinas ou níveis, a saber[7]:

- Fonética:

  É a área de estudo dos aspectos da linguagem relacionados aos sons das palavras.

- Morfologia (Léxico):

  É o estudo relacionado com a formação da palavra, a sua composição e o modo como os segmentos (prefixo, radical, sufixo, etc.) se juntam para formar uma palavra.

- Sintaxe

  É o estudo do conjunto de regras que regem a combinação de palavras para formar frases.

---

6 Essa idéia de contextualizar primeiramente é bem exposta por Minsky (2006) em seu livro The Emotion Machine. Nele, o cientista, considerado o Pai da Inteligência Artificial, defende a ideia de que, num primeiro momento, deve-se elevar ao extremo a complexidade de um problema, sem procurar simplificações, para, só então, num segundo momento, dividi-lo em minúsculas partes com um grau de dificuldade menor o suficiente para que se possa resolvê-las. Foi esse o princípio adotado nesse projeto.

7 Azeredo (2007) e Nugues (2006).

- Semântica

  É o estudo do significado das palavras e das frases, quando se combinam.

  Relacionada à Semântica há pelo menos três áreas importantes:

  - Discurso
    - a Semântica dentro do contexto linguístico, considerando um texto como um todo
  - Diálogo
    - a Semântica dentro do contexto linguístico, considerando o contexto limitado de uma conversação
  - Pragmática
    - a Semântica geral, dentro do contexto linguístico e extra-linguístico

Em se tratando de processamento computacional, cada um desses níveis envolve a utilização de tecnologias diferentes, tais como processamento de sinais (Fonética) ou *parsing* (Sintaxe).

2.2 Escopo

Considerando a subdivisão de áreas apresentadas na seção anterior, pode-se situar o escopo principal desse projeto como sendo o Léxico, a Sintaxe e parcialmente a Semântica. A parte da Semântica será restrita ao Léxico – semântica das palavras - e à Sintaxe – semântica de sentenças.

Especificando melhor, o SIMPLES é um protótipo de um analisador parcial de estruturas sintáticas capaz também de validar algumas questões semânticas relacionadas à Sintaxe. Seu escopo sentencial são as orações simples[8]. O conjunto léxico reconhecido é de aproximadamente quatro mil vocábulos da Língua Portuguesa (LP). As regras de produção utilizadas foram

[8] Esse é o motivo de o projeto receber o nome de SIMPLES. Uma oração simples é aquela que tem a presença de apenas um núcleo verbal. *Latu sensu*, orações que tem um verbo principal e um auxiliar também são consideradas orações simples, mas, por simplificação, nessa primeira versão somente serão validadas orações com um, e apenas um, verbo fazendo parte da sua composição.

baseadas na bibliografia pesquisada e também em inferências empíricas, visando o reconhecimento de um número expressivo de estruturas sintáticas[9].

Quanto ao tempo verbal, são reconhecidas sentenças com verbos no futuro do presente, presente e pretérito perfeito, todos do modo indicativo.

[9] Desde que siga os padrões da norma culta – padrões sugeridos pela Norma Gramatical Brasileira (NGB).

3 REVISÃO BIBLIOGRÁFICA

A revisão bibliográfica foi dividida em duas partes: na primeira, faz-se a análise, sob o enfoque dos objetivos propostos nesse projeto, de alguns *softwares* computacionais que implementam, ao menos de forma parcial, analisadores sintáticos ou semânticos de LP; na segunda parte, define-se a gramática sintagmática, a teoria dos papéis temáticos e o conceito de construções, bases teóricas linguísticas utilizadas na implementação do SIMPLES.

3.1 *Parsers* de Língua Portuguesa

Essa subseção concentra a análise da concepção e implementação de *softwares* de LC que se enquadram ou tangenciam o escopo desse projeto e que, de alguma forma, influenciaram na implementação do protótipo.

3.1.1 Corretores gramaticais

Uma das possíveis utilizações de um *parser* é fazer parte de um corretor ortográfico e gramatical incorporado a um processador de textos. Existem vários, não só em Português Brasileiro (PB), mas em outras línguas também. Aqui se fará uma brevíssima explanação sobre dois corretores muito utilizados por usuários brasileiros de processadores de textos, a saber:

ReGra (DIAS-DA-SILVA et al., 2007)[10] ,

acoplado ao Word, do pacote Microsoft OFFICE, *software* proprietário da Microsoft,

COGRoo (KINOSHITA, 2007) [11]

acoplado ao Writer, do pacote OPENOFFICE.ORG, *software* livre.

Essas ferramentas auxiliam na produção de textos, pois identificam onde estão os possíveis erros ortográficos e gramaticais no texto, destacando-os e sugerindo alterações conforme a gramática e o dicionário incorporados. Os dois *softwares* reconhecem um amplo

10 *Software* desenvolvido por uma equipe interdisciplinar do Núcleo Interdisciplinar de Linguística Computacional – NILC – cujas instituições participantes são: a Universidade de São Paulo USP), a Universidade Federal de São Carlos (UFSCar) e a Universidade Estadual Paulista (UNESP).

11 Projeto originalmente desenvolvido por professores e alunos da Universidade de São Paulo (USP), em projeto financiado pela FINEP - Financiadora de Estudos e Projetos (Silva et al., 2006).

número de palavras e têm uma margem de acerto considerável nas correções propostas (Uliano et al., 2010).

O paradigma empregado na modelagem da gramática dessas ferramentas é baseado nos conceitos da GT, mas considerando os dados estatísticos em cima de um corpus (conjunto de textos compilados para uso em PLN) ou corpora (conjunto de corpus) (Dias da Silva et al., 2007).

A classificação dos *tokens* é feita conforme o método híbrido por um processo conhecido como etiquetagem ou anotação, no qual cada palavra é classificada conforme suas ocorrências em dados estatísticos levantados do corpora (Dias-da-Silva et al, 2007; Silva et al., 2006).

Com esse processo, essas ferramentas conseguem reconhecer a grande maioria das sentenças gramaticalmente corretas. Há, no entanto, algumas ocorrências de falsos positivos, como são chamadas as indicações de erros inexistentes (Silva et al., 2006).

Os dois projetos têm méritos pela sua aplicabilidade computacional alcançada. Ambos são ferramentas funcionais e gozam de prestígio entre usuários dos processadores de textos mais utilizados no Brasil, prestando-se, ao menos parcialmente, ao fim a que se destinam.

O SIMPLES tem o escopo menor que esses produtos, sob o ponto de vista de aplicabilidade prática como corretor ortográfico e gramatical; no entanto, possui fragmentos de reconhecimento semântico.

É exatamente nessa questão de semântica frasal que o SIMPLES apresenta um diferencial em relação aos dois softwares apresentados. Por exemplo, considerando a frase 1[12]

(1) * A estante adora o pai dela.[13]

Os dois corretores aceitariam como válida tal sentença.

Já o SIMPLES faz críticas semânticas a respeito do papel temático do sujeito. Como se verá mais adiante, “adorar” é um verbo do tipo PSICO1, que só pode ser experienciado por um ser animado. A palavra “estante” se refere a um ser inanimado, portanto a frase é julgada incorreta pelo SIMPLES.

---

[12] Para evitar muita repetição de palavras, nesse projeto os termos frase, oração e sentença serão utilizadas indistintamente como se fossem sinônimas, embora teoricamente não as sejam.

[13] O asterisco é utilizado para indicar que a sentença está incorreta.

3.1.2 *Parsers* fortemente baseados em teorias da LM

Utilizando outros paradigmas, na linha de pesquisa mais específica da Linguística, foram analisados especialmente os trabalhos de Pagani (2004) e Othero (2006).

O primeiro desenvolveu uma gramática exemplificativa das facilidades em escrever uma gramática estruturada em sintagmas na linguagem Prolog. O trabalho de Pagani (2004) implementa uma gramática simples, mostrada na Figura 1.

(a) S →SN SV
(b) SN →N
(c) SN →Det N
(d) SV →V
(e) SV →V SN
(f) SV → V SP
(g) SV → V SN SP
(h) SP → P SN
(i) Det → {o, a, os, as, …}
(j) N → {João, Maria, homem, menino, …}
(k) V →{corre, ama, gosta, coloca, …}
(l) P →{de, em, para, …}

Figura 1 - Gramática de Pagani (Pagani, 2004, p.13)

A gramática de Pagani foi ampliada e implementada em Prolog. O objetivo de Pagani era mostrar as potencialidades da ferramenta DCG[14] para linguistas leigos em Computação.

Mais tarde, seu trabalho serviu de base e inspiração para Othero desenvolver seu *parser* Grammar Play (Othero, 2006). Nele, não só o número de regras sintáticas aceitas pelo *parser* foi aumentado, como também o conjunto léxico foi bastante ampliado (chegando a cerca de treze mil palavras).

O que diferenciou o trabalho de Othero da implementação de Pagani, além da ampliação do escopo, foi o fato de que Othero desenvolveu um analisador baseado fortemente na Teoria X-Barra. Essa teoria é derivada da Gramática Gerativa de Chomsky. De acordo com Mioto (2007):

> A Teoria X-Barra é o módulo da gramática que permite representar um constituinte. Ela é necessária para explicitar a natureza do constituinte, as relações que se estabelecem dentro dele e o modo como os constituintes se hierarquizam pra formar a sentença. (MIOTO,2007, p.46)

Contudo, cabe destacar que há limitações no Grammar Play que, sob o ponto de vista de abrangência léxica, não se compara com aqueles incorporados a editores de textos, vistos na

[14] *Definite Clause Grammar*.

subseção anterior. Isso se deve ao próprio objetivo proposto, já que o Grammar Play foi um trabalho acadêmico a nível de mestrado, destinado a explorar especificamente a aplicação da Teoria X-Barra no campo da Linguística.

Em termos de escopo, o Grammar Play valida frases declarativas, em ordem direta e com apenas um verbo, ou seja, orações simples. A maioria das sentenças validadas tem que estar expressas na ordem natural direta (em termos de GT, primeiro o sujeito, depois o predicado).

O Grammar Play, nos moldes de Pagani, foi implementado utilizando destacadamente o recurso DCG (Pereira e Schieber,1987) da linguagem Prolog. Tal recurso, presente na maioria das versões dessa linguagem, facilita a implementação de uma gramática que se utilize do conceito de estrutura de constituintes, como conclui Pagani (2004).

Mais tarde, em recente trabalho (Othero, 2009), Othero admitiu as limitações de se implementar um *parser* mais abrangente utilizando-se somente a Teoria X-Barra, nos moldes do seu Grammar Play.

Como o objetivo precípuo desses analisadores é auxiliar no ensino acadêmico das Teorias Linguísticas, especificamente a Teoria X-Barra no caso de Othero (2006), pode-se dizer que, ao fim a que se destinam, foram bem sucedidos.

O projeto SIMPLES inspira-se nos trabalhos de Pagani e Othero, tanto no aspecto de valorização das pesquisas teóricas da LM em aplicações de LC, como na exploração do recurso DCG da linguagem Prolog.

Uma limitação do conjunto léxico do aplicativo Grammar Play é que só estavam previstos verbos no presente do indicativo e na terceira pessoa. Nesse quesito, o SIMPLES representa um avanço do modelo de Othero (2006), pois aceita verbos conjugados em todas as pessoas verbais (e faz a análise da concordância verbal também), bem como aceita verbos nos seguintes tempos verbais: presente, pretérito perfeito e futuro do presente, todos do indicativo.

3.1.3 Extratores de Sintagmas Nominais (SN)

Foram encontrados inúmeros trabalhos que varrem o texto visando à extração dos sintagmas nominais das sentenças. A motivação dessas pesquisas é a utilização dos SN como indexadores de banco de dados, a fim de agilizar as consultas a esses bancos (Miorelli, 2001; Morellato, 2007). A implementação do SIMPLES visa a delimitar não só os SN, como também os demais sintagmas existentes nas orações. Esses trabalhos têm relação com o SIMPLES à

medida que serviram de base para se procurar especificar os elementos constituintes dos sintagmas, especialmente os SN.

3.2 Bases Teóricas

Nessa subseção se encontra um resumo das principais teorias utilizadas como fundamentação teórica para a implementação do protótipo do projeto.

3.2.1 Gramática Sintagmática (GS)

Os avanços no estudo das gramáticas formais, sobretudo nos moldes em que é aplicada computacionalmente, se devem muito às pesquisas de Noam Chomsky (1957, 1965). O linguista preocupou-se em construir um modelo matemático para descrever alguns dos traços mais notáveis da linguagem (Lyons, 1970). Seus estudos de modelagem das linguagens naturais deram origem à Gramática Gerativa Transformacional (GGT). Essa teoria gramatical acabou sendo um marco não só para a Linguística, como também para a Ciência da Computação, pois os resultados de suas pesquisas influenciaram de maneira definitiva a área de linguagens de programação, servindo de base para alguns importantes componentes de *software*, incluindo algumas partes de compiladores[15].

Uma das teorias de Chomsky percola todas as suas demais teorias: é a de que toda a frase em linguagem natural é segmentada em microestruturas. Cada uma dessas partes ficou conhecida na literatura como sintagma. Daí o fato de que, segundo o modelo de Chomsky, qualquer gramática de linguagem natural é, inerentemente, sintagmática.

Para exemplificar as idéias de Chomsky (1965), ainda que de maneira breve, segue um fragmento de gramática na Figura 2[16].

---

[15] Hopcroft, Ullman e Motwani (2003).

[16] Na seção concernente à gramática do SIMPLES voltar-se-á a esse assunto, quando será apresentado um exemplo prático do funcionamento da GS.

O → SN SV

SN → det n

SV → v SN

Onde:

O – oração

SN – sintagma nominal

SV – sintagma verbal

n – nome

v – verbo

det – determinante

Figura 2. Esboço da gramática de Chomsky

3.2.2 Teoria dos Papéis Temáticos (TPT)

Perini (2010) descreve papel temático dessa forma: "papel temático é a relação semântica que existe entre o verbo e os diversos sintagmas que co-ocorrem com ele na oração" (Perini, 2010, p.51).

Basicamente, essa teoria afirma que a Semântica tem grande relevância na validação de estruturas sintáticas (Moreira,2000).

O conceito de papéis temáticos tem sido objeto de bastante estudo e alvo de muitos trabalhos acadêmicos, sobretudo de um grupo de pesquisas da Universidade Federal de Minas Gerais (UFMG). Na modelagem do SIMPLES, valeu-se muito do conteúdo de alguns desses trabalhos[17].

A utilização que se faz dessa teoria no SIMPLES é no estabelecimento de restrições semânticas aos sintagmas. Um exemplo:

(2) * A mesa gosta de crianças.

A TPT trabalha a idéia de que os verbos selecionam seus argumentos[18], ou seja, são eles que determinam, na oração em que participam, quais são os demais elementos que devem (ou

[17] Especialmente Cançado (1995, 1996, 2000, 2002, 2005), Moreira (2000), Ciríaco (2007), Berg (1996) e Godoy (2008).

[18] Com a ressalva de que determinados verbos possui mais de uma significação; nesse caso, a construção a ser utilizada e os papéis temáticos exigidos na sentença dependerão do emprego semântico do verbo na frase, cabendo ao *parser*, para ser efetivo, admitir todas as hipóteses semânticas de cada verbo modelado.

podem, dependendo do caso) ocorrer na oração e de que tipo eles tem que ser (ou podem ser, também dependendo do caso). No caso do exemplo, o verbo *gostar* exige que seu argumento externo (sujeito) necessariamente seja um ser animado e um argumento interno (objeto) como podendo ser qualquer SN, desde que esteja anteposto pela preposição *de*. Isso ocorre porque em verbos como *gostar*, que é um verbo psicológico (Cançado, 2000, 2002), somente seres animados podem ser agentes – sujeitos. Na TPT, o papel do agente é chamado de experienciador[19].

Mais detalhes sobre a utilização da TPT e suas implicações para o SIMPLES serão deixados para a seção 5.4, que tratará das construções sentenciais.

3.2.3 Construções

O objetivo geral ao se iniciar esse projeto, como se destacou anteriormente na introdução desse trabalho, era pesquisar teorias linguísticas ainda não exploradas em LC que pudessem contribuir efetivamente para a criação de um modelo computacional da linguagem. Nesse caminho, o conceito de "construção" pode ser considerado como a grande descoberta.

O conceito adotado nesse projeto para construção segue a definição de Perini (2010, p.50)[20]:

> Uma construção se define por seus constituintes sintáticos e pela relação semântica que cada um deles tem com o verbo da oração. Para usar os termos técnicos, a construção se define por:
> (a) os tipos de constituintes de que é formada (SN, V,…);
> (b) a função sintática de cada um deles (sujeito, objeto…);
> (c) os papéis temáticos de cada um deles (Agente, Paciente…).

A pressuposição desse conceito é a possibilidade de se fazer um mapeamento de todas as estruturas sentenciais – ou construções - possíveis numa determinada língua. A potencial aplicação em LC desse pressuposto é enorme, haja vista a objetividade de que é impregnada.

---

[19] Na TPT, há uma nomenclatura específica para cada papel selecionado; além do papel de experienciador, há outros como agente, paciente, objetivo, entre vários outros; como o enfoque nesse projeto é a parte prática, não se deterá sobre esses conceitos.

[20] As pesquisas sobre construções é um tema atual e efervescente em Linguística com muitas pesquisas sendo desenvolvidas a respeito desse assunto; uma temática em particular vem ganhando espaços no meio acadêmico - a gramática de construções (Santos, 2007). Essa teoria generaliza o conceito de construção como abrigando qualquer unidade constituinte. No momento decidiu-se que o melhor enfoque para esse projeto é mesmo a adoção da definição dada po Perini (2010), que restringe como unidade constituinte de construção a sentença.

Mas é preciso admitir que o trabalho de inventariar todas as construções possíveis de uma língua não é tarefa fácil, fato admitido também por Perini (2010, p. 97):

> É importante apontar que a análise das construções da língua está ainda na infância; são conhecidas algumas dezenas, mas certamente existem muitas mais, até hoje não descritas.

No SIMPLES se implementam algumas dessas construções.

4 PROLOG

Essa seção está subdividida em 3 partes: na primeira se fará uma abordagem sucinta de Prolog, explicando os conceitos básicos e a sintaxe básica da linguagem; a segunda subseção exemplifica um recurso especial da linguagem muito útil para a implementação do SIMPLES, o DCG (*Definite Clause Grammar*), e a última subseção especifica o compilador de Prolog utilizado nesse projeto.

4.1 Conceitos Básicos

Prolog é um acrônimo de PROgramming in LOGic[21]. Como o próprio nome sugere, essa é uma linguagem de programação que utiliza paradigma de programação lógica. Isso implica dizer que ela é declarativa, ou seja, seus programas consistem em declarações de fatos e regras, ao invés de instruções e fluxos de controle[22]. Nesse tipo de linguagem, os programas são escritos na forma de lógica simbólica e usa um processo de inferência lógica para produzir resultados.

Desenvolvida em 1972 por Alain Colmerauer e Phillipe Roussel, entre outros colaboradores da Universidade de Marselha (França) [23], Prolog ganhou notoriedade para aplicações em PLN[24]. A escolha por Prolog para a implementação do protótipo se deve exatamente pelo fato de essa ser uma linguagem que está intimamente relacionada com processamento de Linguagem Natural desde sua origem[25]. Alguns autores chegam a afirmar que ela é a linguagem de programação ideal para PLN[26].

---

21 Favero (2006).

22 Sebesta (2000).

23 Tate (2010, p.96) e Covington, Nute e Vellino (1997, p. 1).

24 Tate (2010, p.96).

25 Palazzo (1997), Favero (2006) e Nugues (2010, p.433). Sebesta (2000, p. 579) cita que o interesse de Colmerauer e Roussel era especificamente PLN.

26 Covington, Nute e Vellino (1997, p. 407).

Um programa Prolog consiste em uma coleção de fatos e de regras que definem relações entre as variáveis envolvidas no espaço-problema. A idéia desse paradigma é oferecer ao programador uma linguagem que simule o modo como o ser humano resolve os problemas utilizando a lógica.

Sua filosofia é bem simples: há basicamente 3 ideias em um programa Prolog[27]:

- fatos
  - observações diretas do mundo. Ex. Mickey é um rato. Rato gosta de queijo.
- regras
  - inferências lógicas sobre o mundo. Ex. Se alguém é um rato, então gosta de queijo.
- consultas
  - Perguntas sobre os fatos ou as regras. Ex. Mickey gosta de queijo?

Fatos e regras formam a base de conhecimento sobre a qual são feitos os questionamentos ou consultas. Essas consultas são também conhecidas como metas.

Fatos e regras são dois tipos de cláusulas. Por esse motivo, pode-se afirmar que um programa em Prolog é um conjunto de cláusulas, sendo que cada cláusula corresponde a uma fórmula lógica (Favero, 2006, p.6).

Nos primeiros tempos, cada implementação de Prolog praticamente criava a sua própria sintaxe. A partir de 1995, a *International Organization for Standard* (ISO) publicou um padrão sobre Prolog, que ficou conhecido como Prolog ISO. Tal padrão vem sendo seguido pela maioria dos fabricantes de interpretadores da linguagem, o que veio a facilitar a vida dos seus programadores[28].

Prolog não oferece dificuldades em termos de sintaxe. A seguir um exemplo de como se representam fatos e regras.

- fatos

---

[27] O exemplo é bem grotesco e tem o objetivo de facilitar a compreensão dos conceitos. Por oportuno lembra-se que um fato para o Prolog obviamente não precisa ser verdade para o mundo real; o conceito de verdade para o Prolog é manipulado pelo programador quando ele define quais são os fatos e quais são as regras do seu programa.

[28] Nugues (2010).

- forma geral: relação(objeto 1, objeto 2, ..., objeto n).
  - rato(mickey).
  - gostar(rato, queijo).

- regras – padrão
  - forma geral: *cabeça*[29] :- (meta1, meta2, ..., meta n).
    - gostar(X, queijo):-rato(X).

Qualquer forma de dados em Prolog é chamada de termo. Há 3 tipos de termos, basicamente, em Prolog:

- termos atômicos
  - valores são imutáveis
  - podem ser:
    - átomos
      - minúsculas (começam com)
    - números
      - podem ser
        - inteiros ou
        - ponto flutuante
- termos variáveis
  - valores podem mudar
  - maiúsculas (começam com)
- termos compostos
  - estruturas – formados por termos atômicos ou variáveis
    - formado por
      - functor(nome da relação)
      - argumento(s)[30]

[29] A fórmula completa seria cabeça :- cauda.

[30] De certa forma, um átomo é um caso especial de estrutura, pois poder-se-ia compreendê-lo como uma estrutura de aridade zero (Covington, Nute e Vellino, 1997, p.14).

- aridade – diz-se da quantidade de argumentos da estrutura.
  - Exemplo: a(b,c). A estrutura *a* tem aridade 2.

Um predicado é definido pelo conjunto de cláusulas com o mesmo functor principal e a mesma aridade (Nugues, 2006, p.441).

Executar um programa em Prolog, de forma muito resumida, consiste em fazer uma pergunta à base de conhecimento - o programa (conjunto de fatos e regras definidos pelo programador).

Para chegar à resposta, o interpretador Prolog utiliza basicamente dois mecanismos: *matching* e *backtracking*.

Matching, unificação ou casamento, é o processo pelo qual o interpretador busca casar a consulta com os fatos e regras que ele tem em sua base de conhecimento.

Suponha-se um programa bem básico como o do exemplo já mostrado do rato:

```
rato(mickey).
gostar(rato,queijo).
gostar(X,queijo):- rato(X).
```

Quadro 01 – Exemplo básico de código em Prolog

Ao fazer a consulta

```
gostar(mickey, queijo).
```

Quadro 02 – Outro exemplo básico de código em Prolog

O interpretador casará “gostar(mickey, queijo).” com “gostar(X,queijo).” e a partir daí tentará deduzir e processar as informações. Substituirá a variável X por “mickey” e seguirá derivando estruturas e verificando a validade do predicado consultado.

Portanto, a partir da consulta do usuário do programa, é disparado um processo interno que só cessará quando o interpretador souber que o predicado consultado é válido e/ou quando todos os caminhos possíveis já estiverem encerrados, ou seja, aquele predicado é falso à luz dos fatos e regras descritos naquele programa. Essa busca interna pela validação do predicado

consultado, na qual o interpretador tentará todas as alternativas de casamento possíveis, tendo que muitas vezes retroceder para avançar numa busca, até esgotar todas as possibilidades é que se chama *backtracking* ou retrocesso.

4.2 Gramática de Cláusulas Definidas

A Gramática de Cláusula Definida, ou DCG (*Definite Clause Grammar*), é um mecanismo excelente para processar gramáticas, embutido na sua notação de cláusulas, o que facilita bastante a tarefa do programador de PLN.

A sintaxe é básica é:

```
s --> sn sv.
```

Esse recurso facilita ao projetista porque possibilita a passagem das regras de produção diretamente da modelagem de dados para a codificação em Prolog.

Mas o DCG permite mais: possibilita uma solução para concordância tanto verbal quanto nominal, conforme visto em Pereira e Schieber (1987), Pagani (2004) e Othero (2006). Os elementos que permitem fazer a concordância são os atributos e aparecem como argumento dos termos.

A título de exemplo, segue a solução encontrada por Pagani (2004) para validar o fato de o verbo *correr* ser intransitivo[31] utilizando a DCG de Prolog.

```
            % ...
sv(Num, V) --> v(Num, i, V).
            % ...
 v(Num, Val, V) --> [V], {v(Num, Val, V)}.
              %...
v(sing,  i, corre).
v(plur,  i, correm).
```

Quadro 03 – Código em Prolog do verbo “correr”

[31] Pagani implementou somente a terceira pessoa do singular e plural em sua gramática. No SIMPLES, como se verá na próxima seção, são implementadas também as demais pessoas verbais.

A aplicação prática da DCG ficará mais evidente na próxima seção, quando será explicitada a gramática do SIMPLES.

4.3 SWI – Prolog

Há várias implementações de interpretadores disponíveis de Prolog. Para a programação do SIMPLES, o escolhido foi o SWI-Prolog. Ele é desenvolvido pela Universidade de Amsterdã e é um *software* com distribuição livre para ensino e pesquisa. Seu desempenho justifica sua escolha para esse projeto, pois utiliza poucos recursos da máquina.

O SWI utiliza o Prolog padrão, ou seja, o Prolog ISO, que disponibiliza ao programador o mecanismo do DCG. Dessa forma, para escrever uma gramática com DCG em SWI, basta escrever as regras de produção seguindo a sintaxe da linguagem.

A versão SWI utilizada na implementação do SIMPLES foi a 5.10.5 para Windows XP[32] e não apresentou qualquer tipo de problema em momento algum dos testes. Não se pode inferir que com uma quantidade maior de dados ele manterá o mesmo desempenho. O que se pode afirmar é que, ao menos com o quantitativo de léxico (cerca de quatro mil palavras) e de regras (cerca de 200) utilizados no projeto, não se teve qualquer tipo de problema de desempenho com essa versão.

[32] Essa versão pode ser obtida gratuitamente no seguinte endereço eletrônico: http://www.swi-prolog.org/download/stable

5 DESCRIÇÃO DA GRAMÁTICA DO SIMPLES

Essa seção explicita a modelagem da gramática utilizada no SIMPLES. Para ilustrar, são mostrados fragmentos do código exemplificando como foi feita a implementação em Prolog. Num primeiro momento, antes da exposição da modelagem propriamente dita, é feita uma revisão do conceito de gramática para fins computacionais. Depois, a gramática é apresentada, com seus elementos constituintes aparecendo de forma gradual, nessa ordem:

- Léxico,
- Estruturas Sintagmáticas e
- Construções Sentenciais.

As Construções Sentenciais (CS) são formadas por Estruturas Sintagmáticas (ES) e essas, por sua vez, são compostas por elementos léxicos.[33]

5.1 Revisão conceitual do termo gramática como aplicado em Ciência da Computação[34]

Para esclarecer o conceito de gramática, segue a apresentação do conceito, segundo Price e Toscani (2001, p.18):

> Uma gramática G é um mecanismo para gerar as sentenças e é definida pela quádrupla:
>
> (N, T, P, S)
>
> onde
>
> N é um conjunto de símbolos não terminais (variáveis)
>
> T é um conjunto de símbolos terminais (constantes), e $T \cap N = \varnothing$
>
> P é um conjunto de regras de produção (regras sintáticas)
>
> S é o símbolo inicial da gramática ($S \in N$).

Com base no conceito de Price e Toscani, algumas considerações podem ser tecidas para se entender melhor como funciona uma gramática sob o ponto de vista computacional:

- há somente dois tipos de símbolos: os terminais e os não terminais;

---

[33] Apesar de ES e CS formarem conjuntamente a Sintaxe, deixou-se esses conteúdos em subseções diferentes para uma maior clareza na exposição do assunto.

[34] Há várias definições para o termo "gramática" em Linguística; considerou-se que discuti-los aqui seria improdutivo, uma vez que o interesse é a aplicação prática do conceito em Computação.

- nas sentenças só aparecem os símbolos terminais[35];
- as regras de produção possuem do lado esquerdo sempre um, e apenas um, não terminal e do lado direito uma combinação de terminais e não terminais[36];
- as regras de produção determinam, direta ou indiretamente, a ordem permitida dos terminais na sentença;
- os não terminais são variáveis que o programador utiliza como recurso para codificar (ou decodificar) a gramática;
- os não terminais sempre derivam em algum outro símbolo (terminal ou não terminal) [37];

A título de exemplificação, na gramática da Figura 3 os terminais são os símbolos **det**, **n** e **v** e os não-terminais são os símbolos **O** (que também é o símbolo inicial), **SN** e **SV**.

Há duas estratégias básicas para se fazer uma análise sintática (Price e Toscani, 2001): *top-down* ou *bottom-up*[38].

Na análise *top-down*, constrói-se uma árvore de derivação a partir do símbolo inicial da gramática até chegar aos *tokens* de entrada (a sentença), enquanto que na estratégia *bottom-up* faz-se o caminho inverso, ou seja, constrói-se uma árvore de derivação a partir dos *tokens* até chegar ao símbolo inicial[39].

Portanto, pode-se fazer uma análise de uma sentença dada, partindo-se do símbolo inicial ou dos *tokens* de entrada; após sucessivas derivações, chega-se à conclusão de que a sentença analisada é válida ou não à luz de uma determinada gramática.[40]

Observa-se, então, a oração *Alice comprou um carro*, expressa na Figura 3. Supondo que as palavras *Alice*, *comprou*, *um* e *carro* sejam símbolos terminais válidos, ou seja, façam parte do

---

35 São os *tokens*.

36 O não terminal da esquerda pode derivar em vazio (o símbolo de vazio é ϵ).

37 Esse símbolo pode, eventualmente, ser o vazio ou uma combinação de terminais e não terminais.

38 Análise *top-down* é também conhecida como descendente e análise *bottom-up* como ascendente ou redutiva (Price e Toscani, 2001)

39 Em outras palavras, na *top-down* o analisador percorre a árvore das folhas à raiz e na *bottom-up* da raiz às folhas.

40 Se for válida, chega-se, por um processo de encadeamento de derivação, da raiz às folhas, ou vice-versa; se for inválida, após esgotar o teste de todas as alternativas de achar esse caminho que valida a sentença, o *parser* deve retornar a mensagem correspondente ao final das tentativas.

conjunto do Léxico da linguagem e que, portanto, esteja tudo certo com a frase sob o ponto de vista lexical. Para concluir se sintaticamente ela também é válida, o analisador utiliza sucessivas derivações para verificar se existe algum caminho derivacional que leve da raiz às folhas ou das folhas à raiz. Isso pode ser observado visualmente na Figura 3, seguindo o caminho das setas.

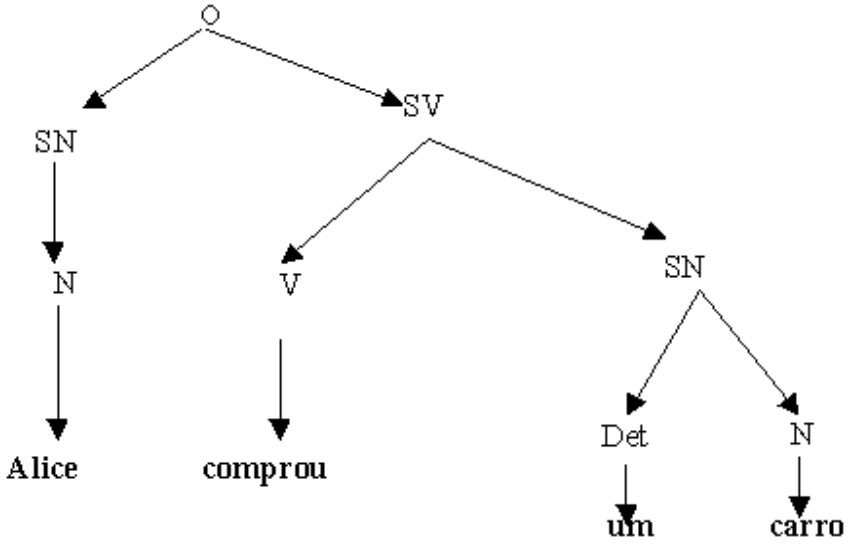

Figura 3– exemplo de gramática utilizando a estrutura de sintagmas.

Como se pode ver, a frase do exemplo também é válida sintaticamente, pois é possível percorrer o caminho das folhas pra raiz, ou vice-versa, sem nenhum problema. Isso, enfim, quer dizer que a sentença foi gerada[41] pelo mecanismo identificado pela quádrupla (N, T, P, S), onde:

- N = O, SN, SV
- T = N, V, Det[42]
- P = O $\rightarrow$ SN SV
  SN $\rightarrow$ Det N
  SV $\rightarrow$ v SN
  Det $\rightarrow$ um

[41] A expressão “foi gerada” é utilizada aqui mais no sentido de se encaixar ao conceito de Price e Toscani. Na prática, quer dizer que essa sentença se encaixa nas regras da gramática G, pois uma sentença igual a ela “poderia ter sido gerada” pela gramática G.

[42] No exemplo, os termos *Alice*, *comprou*, *um* e *carro* são considerados como terminais; mas eles poderiam naturalmente serem modelados como atributos, respectivamente, dos termos *n*, *v*, *det e n* , que, nesse caso, seriam os terminais.

$$N \rightarrow \text{carro} \mid \text{Beto} \mid \text{Alice}$$

$$V \rightarrow \text{comprou}$$

- S = O

As linguagens naturais estão longe de serem tão simples como a gramática do exemplo dado. Ora, há potencialmente um número infinito de combinações de estruturas que são utilizadas, de maneira empírica, pelos falantes nativos de uma linguagem. Além disso, o Léxico de uma língua natural é muito mais amplo.

A gramática aplicada nesse projeto, que será descrita nas próximas subseções, está longe de ser definitiva, mas pretende ser uma contribuição à área de LC, dentro do escopo a que se propõe.

5.2 O Léxico do SIMPLES

Diante da inadequação da aplicação para fins computacionais da divisão do Léxico em classes gramaticais conforme consagrada pela GT, optou-se por uma reclassificação das palavras[43] embasada em trabalhos recentes de linguistas brasileiros, sobretudo Perini (2009, 2010), Castilho (2010) e Ilari (2010).

Nesse processo, agrupou-se as palavras, de modo a formar grupos homogêneos, sobretudo quanto ao posicionamento sintático aceito em frases corretas[44] da LP. A escolha por essa

[43] Com a inadequação da classificação do Léxico da GT para aplicações de LC, optou-se pela ruptura de alguns desses padrões. Ainda assim, essa reclassificação não implica uma desconformidade com as regras oficiais devido a alguns fatores, a saber:

- Todo o trabalho de reclassificação é embasado em recentes pesquisas acadêmicas de linguistas renomados, como Perini (2009, 2010), Castilho (2010) e Ilari (2010), todos reconhecidos pela comunidade acadêmica de Letras;
- Esses ajustes não implicam uma alteração das normas da GT sob o ponto de vista da Sintaxe, pois não implicarão alterações sintáticas nas frases; por exemplo, o fato de a palavra *casa* ser classificada lexicalmente como nominal não importa ao usuário de uma língua, que a utilizará da mesma forma e na mesma posição sintática que a utilizaria se a considerasse um substantivo ao invés de nominal (na realidade, muitos usuários da Língua, mesmo os que escrevem textos dentro da norma culta, não sabem a classificação de todas as palavras que utilizam em seus textos, fato que não prejudica seu desempenho como escritor);
- a classificação do Léxico em si não é objeto de análise em textos, mesmo em redações de trabalhos acadêmicos. O que se observa é a Ortografia, a Sintaxe, a Semântica e o teor do discurso.

[44] O conceito de *correto* seguido no SIMPLES quanto à escrita do Léxico e à Sintaxe é o da GT, devido ao fato de essa ser a Gramática Oficial do Brasil, sendo essa a cobrada em concursos públicos e em trabalhos acadêmicos.

metodologia implicou, obviamente, uma classificação diferente da tradicional; porém, foi seguida, sempre que possível, a terminologia padrão da GT.

No tocante à quantidade de vocábulos, optou-se por um vocabulário enxuto, com poucos *tokens*, mas escolhidos estrategicamente, de modo a ser abrangente o suficiente para se testar a validade das várias construções sintáticas modeladas.

A seguir é feito um detalhamento dessa classificação[45].

5.2.1. Nominal

Os nominais abrangem os tradicionais substantivos e adjetivos. Esses dois grupos de palavras são tratados separadamente pela GT[46]. Essa abordagem agrupando-os em uma classe foi retomada recentemente por Perini (2010), que vê muitos traços em comum entre essas duas classes[47]. O SIMPLES baseou-se fortemente nessas idéias de Perini (2010)[48].

Na modelagem dos nominais optou-se por guardar em atributos distintos tanto o potencial comportamento do *token* como substantivo, quanto como adjetivo. Também se registra em atributos a função precípua do *token* segundo a GT[49], ou seja, se originalmente ele é substantivo ou adjetivo.

Assim, há três grupos de nominais modelados:

- aqueles que funcionam só como substantivos;

---

45 Algumas subclassificações são iguais para grupos léxicos diferentes; por exemplo, gênero e número são recorrentes em quase todos os grupos. Nesses casos, a explicação é dada somente na primeira vez em que aparecer o termo – para ilustrar, no exemplo de gênero e número a explicação é dada quando se apresentam os nominais.

46 Nem sempre foi assim; segundo Azeredo (2007), a classificação de substantivos e adjetivos sob um rótulo comum tem uma longa tradição, e só passaram a ser tratados separadamente a partir do século XIV (Azeredo, 2007, p.37).

47 Perini (2010) cita o fato de que a maioria dos substantivos pode ser utilizada como adjetivo e vice-versa.

48 Novamente, reforça-se a idéia reiterada ao longo dessa monografia, qual seja de que, se por um lado a modelagem da linguagem proposta nesse projeto é fortemente baseada em várias Teorias Linguísticas, por outro lado não se prende dogmaticamente a nenhuma delas. Assim, tem-se total liberdade de adaptá-las conforme as necessidades computacionais da modelagem.

49 O que se quer é a melhor modelagem possível para fins computacionais, utilizando-se da teoria que for necessária, seja ela taxada de tradicional ou moderna.

- aqueles que funcionam só como adjetivos e
- aqueles que funcionam tanto como substantivos quanto como adjetivos.

Segue a explicação mais detalhada dos atributos dos nominais.

5.2.1.1 Classificação precípua

Esse atributo guarda a classificação da GT para a palavra, ou seja, se ela é substantivo ou adjetivo.

Valores possíveis: *subst* ou *adj*.

5.2.1.2 [gênero, número][50]

Há dois gêneros (gn) possíveis: masculino (mas) e feminino (fem).

Já o atributo *número* (nr) classifica o vocábulo em singular (sg) ou plural (pl).

5.2.1.3 Classificação quanto à Semântica

Cada palavra tem valores semânticos associados. Pesquisou-se por trabalhos de Linguística que fornecessem esses dados, pois, dessa forma, bastaria ao modelador computacional somente registrar em variáveis distintas esses valores. No entanto, não foram encontrados trabalhos linguísticos significativos nesse sentido[51].

Devido à relevância detectada em se fazer tais distinções, fez-se de igual forma, uma modelagem, ainda que bem parcial e empírica, de algumas semânticas associadas a cada nominal.

Segue a classificação semântica[52] dos nominais.

- Lugar
  i. objeto
  ii. ser
  iii. pátrio

---

[50] Muitas classes do Léxico são classificadas em gênero e número; a título de resumo e apresentação das siglas:
gn → mas| fem nr → sg| pl

[51] Foram encontrados inúmeros trabalhos linguísticos sobre semântica de léxicos, mas todos com um estudo aprofundado por vezes de um vocábulo apenas. São estudos muito interessantes, mas que requerem um tempo maior de análise para se verificar a viabilidade de sua aplicação computacional e de que forma.

[52] Com toda a ressalva feita em relação à parcialidade e ao empirismo de tal classificação.

iv. advérbio de lugar

- Tempo
  i. dia de semana
  ii. hora
  iii. estação
  iv. advérbio de tempo
  v. período do dia
- Advérbio de Modo

Muitos adjetivos podem ser utilizados também com valor de advérbio.

(3) A cerveja desceu redondo.

Setando esse campo, abre possibilidades para, nas regras de produção, permitir que adjetivos de origem, como *redondo*, sejam utilizadas como advérbio de modo.

- Deverbal

Palavras como *construção* herdam de sua origem verbal características de transitividade que precisam ser observadas. No caso de *construção,* é preciso marcar no léxico que a origem é um verbo transitivo direto.

Substantivo que, posposto à preposição *de*, forma adjetivo

Exemplo: comunicação – substantivo
de comunicação = comunicacional - adjetivo

Essas distinções servem de base para algumas validações sintático-semânticas. Como em:

(4) João gosta de churrasco nos domingos.

Pode-se fazer uma restrição, na regra de produção que valida o sintagma adverbial oracional de tempo, para que seja somente aceito como núcleo desse sintagma um nominal se, e somente se, ele puder ter valor semântico de tempo. Pode-se ainda ir mais longe e permitir mais de um sintagma adverbial oracional de tempo na mesma frase, como em

(5) João gosta de futebol nos domingos de tarde

e, ao mesmo tempo, com a distinção do atributo tempo entre dia de semana e horas, não validar frases como essa:

(6) *João gosta de futebol nos domingos nos sábados.

Como é possível vislumbrar que existam inúmeras outras possibilidades de distinções semânticas que ainda não foram modeladas (num primeiro momento somente cinco foram trabalhadas), foi reservado no SIMPLES três “lugares” para possível ampliação do projeto. Essa estratégia visa diminuir o impacto da inclusão de até mais três distinções semânticas no código.

5.2.1.4 Possibilidades Sintáticas

Os nominais podem ocupar três funções diferentes em um Sintagma Nominal: pré-núcleo (PN), núcleo de sintagma nominal (NSN) e núcleo de sintagma adjetival (NSA) [53]. Quando não é possível classificar determinado vocábulo em algum dos quesitos, diz-se que a classificação não é aplicável (a sigla utilizada para isso é *n/a*).

Resumindo, quanto a possibilidades sintáticas, tem-se a seguinte classificação:

[pn, nsn, nsa]

5.2.1.5 Classificação como substantivo

Seguiu-se a classificação da GT (Bechara, 2009), com algumas adaptações.

- Animado/Inanimado
    i. racional
        a. humano
        b. figurado[54]
    ii. irracional
- Contável/ Incontável
- Concreto/ Abstrato
- Causa

    Esse atributo marca a possibilidade do substantivo ser utilizado como causa de determinado evento verbal[55].

---

[53] Essas terminologias devem-se a Perini (2009).

[54] Esse é um artifício para admitir a possibilidade de frases em que uma entidade pratica uma ação própria de um ser humano. Um exemplo: A torcida adorou a contratação do jogador.
No caso, *torcida*, mesmo não sendo um ser animado, através desse artifício pode ser equiparado a tal.

[55] Exemplo:
A inflação provocou a alta da gasolina.

5.2.1.6 Classificação como Adjetivo

Baseou -se fortemente na classificação de Castilho (2010) [56].

- classificatório
    i. pátrio;
    ii. cor;
    iii. dimensões;
    iv. derivado de substantivo

        Exemplos: naval = de navio
        comunicacional = de comunicação

- avaliativo
    i. típico;
    ii. modal1;
    iii. modal2;
    iv. geral.

5.2.1.7 Signo[57]

É a palavra propriamente dita.

Assim, o signo de *livro* é *livro*.

O nominal *inflação* é o núcleo do SN que é o sujeito do verbo *provocar*. Para validar sintaticamente a frase do exemplo, o termo *inflação* terá que constar no Léxico com o atributo "causa" setado.

[56] Praticamente há uma classificação de adjetivos para cada livro de gramática. No entanto, a maioria dessas classificações ainda estão baseadas na tradição, tendo pouca ou nenhuma aplicabilidade computacional. A de Castilho (2010) foi a mais útil encontrada.

[57] Signo é o próprio *token.* Preservou-se a nomenclatura com que é conhecida na GT.

5.2.1.8 Um exemplo da codificação em Prolog

```
n([mas, sg],                                                    %cfe classif. 5.2.1.1
  adj,                                                          %cfe classif. 5.2.1.2
 [n/a , nsn, nsa ],                                             %cfe classif. 5.2.1.3
 [s, anim(rac,_), _, _,causa],                                  %cfe classif. 5.2.14
 [s,classif,tipico],                                            %cfe classif. 5.2.1.5
 [(n,n/a), (n,n/a), (n, n/a), deverb(td), n/a, n/a, n/a , n/a],%cfe classif  5.2.1.6
          trabalhador                                           %cfe classif. 5.2.1.7 ).
```

Quadro 04 – Código em Prolog do nominal "trabalhador"

5.2.2 Verbo

O verbo é fundamental na modelagem do SIMPLES, uma vez que seleciona quais outros elementos serão permitidos (ou proibidos) na estrutura das sentenças. Diante desse fato, há a necessidade de haver um mapeamento de todas as possibilidades estruturais sintáticas e semânticas que o verbo permite (ou proíbe) nas estruturas oracionais modeladas.

O que cada verbo selecionará é se ele admite alguma construção com complementos e que tipo de complementos são admitidos. Adicionalmente, o verbo seleciona também o tipo semântico de SN que pode ser seu sujeito.

Por exemplo: se o verbo da oração for "amar", há a necessidade de que haja um complemento objeto sem preposição, como em

(7) Maria ama João.

Na classificação dos verbos modelada pelo SIMPLES, há um atributo que especifica em qual tipo de construção[58] cada verbo pode participar e outro que especifica as valências[59] que o verbo aceita na construção em que ele está sendo classificado[60].

A seguir a classificação completa do verbo após a modelagem.

---

[58] Na subseção 5.4 há mais detalhes sobre os tipos de construção e quais implicações sintáticas de cada tipificação.

[59] De forma sucinta, a valência define se o verbo é transitivo ou intransitivo e suas subclassificações.

[60] Há verbos que podem participar de mais de uma construção; nessa modelagem, apesar de homônimos, serão considerados como verbos diferentes.

5.2.2.1 Número

Resumidamente, pode ser *sg* ou *pl*, como os nominais.

5.2.2.2 Pessoa

É a classificação quanto à pessoa verbal (quem pratica a ação descrita pelo verbo) e pode ser primeira, segunda ou terceira pessoa (sendo os valores dos atributos 1, 2 e 3, respectivamente).

5.2.2.3 Tipo de Construção

Esse é o atributo que especifica em qual tipo de construção cada verbo pode participar. A especificação das configurações de cada construção será feita na subseção 5.4; por ora, far-se-á apenas a enumeração dos tipos modelados e as respectivas siglas.

- Construções Psicológicas tipo 1 (PSICO1);[61]
- Construções Psicológicas tipo 2 (PSICO2);[62]
- Construções Agentivas (AGENT);[63]
    i. Eventos internamente causados (eic)
        a. denotam atividade física (ativfis)
    ii. Eventos externamente causados (eec)
        a. denotação de mudança de estado (mudaest)
        b. denotação de contato (contato)
        c. denotação de criação (criat)
- Construções com o verbo ser (SER)[64]
- Construções estativas (ESTAT)[65]
    i. Posse (posse)
    ii. Complemento (compl)

---

61 Cançado (1995, 2002).

62 Cançado (1995, 2002).

63 Cançado e Godoy (2009).

64 Oliveira (2011).

65 Moreira (2000).

- Construções Locativas (LOCAT)[66]
- Construções Processuais (PROCES)[67]

5.2.2.4 Valências

Quanto à valência, na modelagem do SIMPLES optou-se por seguir o entendimento de Perini (2009), que a compreende de uma maneira um pouco diferente da tradicional.

Na GT, os verbos são classificados em:

- intransitivos (não necessitam de complemento)
- transitivos (necessitam de complemento)
    i. direto (complemento sem preposição)
    ii. indireto (complemento com preposição)
    iii. direto e indireto (2 complementos: 1 com preposição e outro sem)
- copulativos (os conhecidos como verbos de ligação)

Perini (2009) questiona a obrigatoriedade de haver complementos em certos verbos. Na modelagem do SIMPLES baseou-se na classificação verbal quanto às valências desse autor (Perini, 2009, p.161-168). Assim, chegou-se a um atributo verbal que admite vários valores.

A seguir são apresentados os valores possíveis desse atributo:

[i, td, ti, tdi, tdd, tii, vl, reserva, reserva, reserva],

onde

i – sem complemento

td – complemento direto, sem preposição

tdi – complemento direto e outro com preposição

tdd – dois complementos sem preposição

tii – dois complementos preposicionados

lig – verbo de ligação

reserva – deixou-se espaço para facilitar adaptação em eventual atualização da modelagem

---

66 Moreira (2000).

67 Ciriaco (2007).

Cada atributo possui ainda atributos internos para subclassificação. Assim, um verbo como “amar” tem esse atributo preenchido da seguinte forma:

[i(anaf), td(full), nhp, nhp, nhp, nhp, nhp, nhp, nhp, nhp],

onde

anaf – admite ocorrência anafórica (omissão de objeto aceita);

full – completo;

nhp – sigla de “não há possibilidade”.

5.2.2.5 [Modo, Tempo]

O modo nesse protótipo é o Indicativo. Novamente, o fato de deixar um atributo com esse valor é para facilitar futuras atualizações, quando se poderá modelar também o subjuntivo e o imperativo.

Os tempos modelados foram:

- presente (pres)
- futuro do presente (futu)
- pretérito perfeito (pass)

5.2.2.6 Signo

Vide nota 57.

5.2.2.7 Um exemplo da codificação em Prolog

```
v(      pl,                                                    %cfe classif. 5.2.2.1
        3  ,                                                   %cfe classif. 5.2.2.2
        [psico1,nhp ] ,                                        %cfe classif. 5.2.2.3
        [i(anaf), td(full), nhp, nhp, nhp, nhp, nhp, nhp, nhp, nhp],  %cfe classif. 5.2.2.4
        [indic, pres],                                         %cfe classif. 5.2.2.5
        amam                                                   %cfe classif. 5.2.2.6).
```

Quadro 05 – Código em Prolog do verbo "amam"

5.2.3 Preposição

A dificuldade maior na modelagem das preposições são as contrações. Estendeu-se a solução adotada por Pagani (2004) e Othero (2006). Ambos classificam a preposição em flexíveis (gênero e número), representadas pelas contrações de preposição com artigos, e inflexíveis, comportando as preposições sem contrações. Na modelagem do protótipo estende-se esse conceito, aceitando-se contrações de preposição também com demonstrativos e pronomes pessoais.

5.2.3.1 Preposição de origem

É a forma original da preposição, sem considerar se ela é uma manifestação com contração ou sem contração.

5.2.3.2 Tipo

Essa é uma peculiaridade da modelagem do SIMPLES[68]. Classifica-se a preposição conforme o elemento com o qual ela se encontra contraída.

Assim, temos, quanto ao tipo:

- art – contração com artigo. Ex. dos (de+os).
- dem – contração com demonstrativo. Ex. nas (em+as).
- ppcr – contração com pronomes pessoais do caso reto. Ex. deles (de+eles).
- pura – preposição sem contração. Ex. de.

[68] Na pesquisa feita, o mais semelhante ao tratamento dado aqui são os citados trabalhos de Pagani (2004) e Othero (2006).

Ressalta-se que Pagani (2004) e Othero (2006) já tratavam a forma contraída da preposição com artigo como preposição. A originalidade no SIMPLES é a extensão desse tipo de tratamento para classes demonstrativo e pronome pessoal do caso reto.

5.2.3.3 [gn, nr]

Os valores possíveis são:

- inf

  São os inflexíveis, ou seja, têm uma única forma. É o caso da preposição pura "com". Nesse caso, atribui-se o valor "inf" para esse campo, conforme já fizeram Pagani (2004) e Othero (2006).

- [_, _]

  Admitem qualquer concordância de gênero e número, como a preposição pura "de".

- [mas/fem, sg/pl]

  Esses variam conforme as contrações. Por exemplo, "dos" terá nesse atributo o valor de [mas, pl].

5.2.3.4 Signo

Como nas demais, mas a atenção especial no caso das contrações é que, nesse caso, ter-se-á nesse atributo o valor contraído (a preposição mesmo estará armazenada no campo preposição original).

5.2.3.5 Exemplo de Implementação em Prolog.

```
p(  de,
    art,
    [mas, pl],
    dos).
```

Quadro 06 – Código em Prolog da preposição "dos"

5.2.4 Artigo

Nesse caso apenas se destaca o fato de serem considerados como artigo somente os que a GT chama de artigos definidos – o, a, os, as.

A classificação só inclui dois atributos:

- [gn, nr]
- Signo

Ambos não requerem maiores explicações, haja vista já terem sido bem explicados anteriormente.

Para completar, um exemplo de implementação em Prolog:

```
ar([mas, pl], os).
```

Quadro 07 – Código em Prolog do artigo “os”

5.2.5 Demonstrativo

Como todos os seus atributos coincidem com outros já apresentados, apenas se mostra aqui um exemplo de implementação em Prolog.

```
dem([mas, sg],
        aquele).
```

Quadro 08 – Código em Prolog do demonstrativo “aquele”

5.2.6 Possessivo

Além dos dois atributos apresentados pelas classes anteriores, [gn,nr] e signo, o possessivo contém também um atributo semântico que guarda o valor da pessoa verbal que possui algo. Esses valores vão de s1 (primeira pessoa do singular) a p3 (terceira pessoa do plural).

Um exemplo de implementação em Prolog:

```
pos([mas, sg],
          p1,
        nosso).
```

Quadro 09 – Código em Prolog do possessivo “nosso”

5.2.7 Pronome Pessoal do Caso Reto

No caso dos pronomes pessoais, é preciso guardar o valor semântico de animado. Para isso mantém-se o formato do atributo de substantivo dos nominais.

Além desse valor, também se guarda o comum [gn,nr], a pessoa verbal (1, 2 ou 3) e o signo. Segue um exemplo de implementação.

```
ppcr(            [ _, sg],
                          1,
[_, anim(rac,hum),_ ,_,_],
                        eu).
```

Quadro 10 - Código em Prolog do pronome pessoal do caso reto “eu”

5.2.8 Pronome Pessoal do Caso Oblíquo

Acrescenta-se, no caso dos oblíquos, em relação aos pronomes do caso reto, um atributo para destacar se ele pode ser utilizado como objeto direto, objeto indireto e/ou reflexivo, dessa forma:

[od,oi,reflex]

onde

od – objeto direto

oi – objeto indireto

reflex – reflexivo

Um exemplo de implementação:

```
ppco(               [   _, sg],
                          1,
      [_, anim(rac,hum),_ ,_,_],
                [od,oi,reflex],
                          me).
```

Quadro 11 - Código em Prolog do pronome pessoal do caso oblíquo “me”

5.2.9 Nome Próprio

Utilizam-se os mesmos atributos que pronome pessoal do caso reto.

Um exemplo de implementação em Prolog é mostrado no Quadro 12.

```
npro(                [ fem , sg],
      [_, anim(rac,hum),_ ,_,_],
                           maria).
```

Quadro 12 - Código em Prolog do nome próprio “maria”

5.2.10 Intensificador

Não se aprofundou muito esse tópico e apenas limitou-se a adaptar a modelagem de Perini (2009, p.113-118) dos intensificadores de sintagmas adjetivais. Na prática, o parâmetro assumido para a classificação dos intensificadores foi somente a posição que cada um ocupa, ou pode ocupar, dentro do sintagma adjetival, conforme Perini (2009).

A seguir um exemplo de implementação em Prolog.

```
intens(     i4,
          muito).
```

Quadro 13 - Código em Prolog do intensificador “muito”

5.2.11 Numeral Cardinal

A classificação dos cardinais é essencialmente quanto ao posicionamento sintático, além de [gn,nr].

Segue um exemplo da implementação em Prolog.

```
ncard([mas, pl],
           c2,
         dois).
```

Quadro 14 - Código em Prolog do cardinal “dois”

5.2.12 Numeral Ordinal

A classificação dos ordinais, semelhantemente aos cardinais, também é essencialmente quanto ao posicionamento sintático, além de [gn,nr].

Segue um exemplo da implementação em Prolog.

```
nord([mas, sg],
           o2,
         segundo).
```

Quadro 15 - Código em Prolog do ordinal “segundo”

5.2.13 Adverbial

A categoria tradicional “advérbios” encobre uma série de classes de comportamento sintático por vezes radicalmente diferentes (Perini, 2009, p. 338). Diante desse fato, num modelo computacional da linguagem não há como considerá-la nos moldes da GT.

O que se chama nesse tópico de adverbial é um subconjunto dos tradicionais “advérbios”, especialmente os terminados em “mente”, cujo comportamento sintático é bem semelhante.

Há, na implementação em Prolog, três atributos específicos para guardar valores semânticos, como no exemplo do Quadro 16:

```
adv([tempo, durad, rápida],
    abreviadamente).
```

Quadro 16 - Código em Prolog do adverbial “abreviadamente”

5.2.14 Negação

Tradicionalmente advérbios, as negações, devido as suas idiossincrasias sintáticas, merecem uma classe própria nessa modelagem.

A seguir pode se verificar como ficou fácil a implementação do “não” em Prolog.

```
neg(não).
```

Quadro 17 - Código em Prolog da negação “não”

5.2.15 Quantificador

Não existem muitas pesquisas conclusivas sobre quantificadores, sobretudo a respeito de sua sintaxe[69]. Dada essa situação, o que se obteve nessa modelagem foi fruto de um levantamento empírico de tipos e classificação dos quantificadores. Devido a essas dificuldades, um aprofundamento no assunto será deixado para futuras versões.

Os tipos, obtidos de forma empírica, são os seguintes[70]:

- Indefinidos (I)

[69] Sintaxe no sentido de posicionamente na frase.

[70] Para cada tipo, há a necessidade também de diferenciação em subtipos, deivdo a diversidade sintática existente entre eles.

São os tradicionais indefinidos.

Exemplo: uns, umas, poucos, outro, diversas.

- Reforço (R)

  Esse termo aparece em Perini (2009).

  Exemplos: próprio, determinado, certo

- Universais (U)

  São os quantificadores universais, muito utilizados em lógica.

  Exemplo: todo, tudo, nada.

Um exemplo de implementação em Prolog de quantificador.

```
quantif(      [mas, sg],
           [indef, i2],
                    o2,
                  todo).
```

Quadro 18 - Código em Prolog do quantificador “todo”

5.3 Estruturas Sintagmáticas

Os não-terminais de uma gramática de língua natural são também chamados de estruturas constituintes sintagmáticas ou simplesmente sintagmas. Esse é um conceito herdado da GS de Chomsky, já descrita na revisão bibliográfica.

Os sintagmas tradicionalmente são classificados conforme a classe gramatical de seu núcleo em:

- Sintagma Nominal (SN)
- Sintagma Verbal (SV)
- Sintagma Adjetival (SAdj)
- Sintagma Adverbial (SAdv)
- Sintagma Preposicional (SP)

Os sintagmas são as estruturas macros da oração. Na implementação do protótipo, qualquer palavra que fizer parte de uma sentença de entrada do protótipo estará fazendo parte, estruturalmente, de um dos sintagmas descritos acima.

Outras estruturas constituintes são utilizadas como artifícios de programação para uma melhor organização dos dados. Na prática, esses não terminais intermediários são derivados de um dos cinco sintagmas.

5.3.1 Sintagma Nominal (SN)

O sintagma nominal exerce basicamente as funções tradicionais chamadas de sujeito e objeto[71] pela GT, podendo ainda ocupar as funções tradicionais de predicativo do sujeito ou de objeto.

O núcleo de um SN é geralmente um nominal com o campo NSN[72] setado, mas há outros tipos léxicos capazes de exercer essa função de núcleo do SN. Por exemplo, os pronomes pessoais do caso reto (PPCR) frequentemente aparecem sozinhos na função de SN (e, consequentemente, de núcleo), como na frase 7:

(8) Ela gosta de mim.

---

[71] Objeto indireto quando precedido de preposição e objeto direto quando sem preposição.

[72] Núcleo do Sintagma Nominal

A mesma situação pode ocorrer em alguns casos com os demonstrativos, nomes próprios e quantificadores, como nas frases 9, 10, 11:

(9) Isso foi legal.

(10) Paulo gosta de Joana.

(11) Tudo estava muito bom.

Porém, grande parte dos SN é composta por mais de um termo léxico e, nesse caso, a identificação do núcleo é fundamental para uma compreensão estrutural do SN. Então, além de reconhecer SN com apenas um elemento, como os mostrados nas sentenças 7 a 10, também foram modeladas as construções de SN compostas de mais de um vocábulo. Esses SN são formados potencialmente por três partes, nessa ordem[73]:

- Determinante (Det);
- Núcleo (NSN) e
- Modificadores (Mod).

A seguir, discorre-se sobre cada uma dessas partes e discute-se como essas partes se interligam para formar o SN.

5.3.1.1 Determinante (Det)

O Det precede o NSN na estrutura do SN.

Os demonstrativos, artigos e possessivos são considerados os Determinantes Primários (DP), pois ocorrem com maior frequência nessa posição[74]. Outros que podem ser Det são os numerais e os quantificadores, além de uma série de combinações feitas entre esses elementos, que geram determinantes compostos.

5.3.1.2 NSN

O núcleo é a referência principal da estrutura do SN.

O SN pode apresentar três partes, dessa forma:

SN → Determinante NSN Modificador

---

[73] Perini (2009).

[74] *Strictu sensu*, o termo determinante é utilizado somente para designar o artigo, o demonstrativo e o possessivo, pois esses determinam o gênero e o número do sintagma; no entanto, no SIMPLES, por questões práticas, ele será utilizado para designar, *latu sensu*, a parte anterior ao NSN de um modo geral, não restrito portanto aos DP.

Nessa estrutura, o núcleo do SN será exercido sempre por um nominal com o atributo NSN setado. São dos traços desse nominal que o SN herdará a maioria de seus traços.

Eventualmente o NSN pode não estar manifestado só por um nominal. É o caso quando ocorre um nominal imediatamente anterior a ele; nesse caso, sem estar com o atributo NSN setado; esse nominal com função adjetiva anteposto ao NSN é denominado de Pré-Núcleo (PN).

5.3.1.3. Modificadores[75]

Modificadores são elementos sintáticos que se agregam ao NSN para qualificá-lo.

Na prática, essa posição é ocupada pelo SAdj, que será visto na subseção 5.3.4.

5.3.1.4. O Sintagma Nominal – conectando as partes

Há várias combinações estruturais possíveis de ocorrência do SN. Para começar, muitos Det, eles próprios, podem formar sozinhos um SN. Exemplo:

(12) Essa é minha mãe.

Na frase acima, "essa" desempenha tanto a função de Det quanto a de núcleo do SN. Esse demonstrativo é uma referência a algo que não está explícito no SN. Ou seja, não há como saber exatamente se a expressão "essa" se refere a algo concreto, contável ou animado. Nem há subsídios no texto[76] para enquadrá-la semanticamente. Isso, às vezes, pode ficar claro para quem está envolvido numa interação devido a outros aspectos extra-linguísticos, como o contexto em que é feita a comunicação[77] ou as ilações que possam ser feitas com termos presentes em outras orações dentro de um texto. Esses dois fenômenos, interacional e discursivo, são conhecidos em Linguística, respectivamente, como aspectos dêiticos e aspectos anafóricos da linguagem. Esses fenômenos precisam necessariamente serem considerados e tratados de alguma forma.

No caso do SIMPLES, a opção que se fez foi marcar essas situações como anafóricas, já que somente o anaforismo pode ser tratado linguisticamente[78]. No exemplo dado, o SN formado

---

75 Guardou-se a nomenclatura de Perini (2009), mas, na prática, segundo o próprio Perini (Ibid), essa função é exercida exclusivamente pelo SAdj (Perini, 2009, p.113).

76 Até porque, no exemplo, é uma frase isolada.

77 Por exemplo, o contexto pode ser alguém apresentando a mãe para outra pessoa.

78 Os aspectos dêiticos são inerentemente extra-linguísticos e só podem ser, linguisticamente, no máximo inferidos. Esse tipo de inferência semântica foge ao escopo desse projeto.

pelo demonstrativo “essa” ficará marcado como anafórico. Muitas vezes isso pode gerar valores inconsistentes sob o ponto de vista semântico[79]. Porém, deixa-se aberta a possibilidade para, em versões posteriores, se fazer ligações semânticas de origem anafórica entre orações.

5.3.1.5 Implementação do SN em Prolog

A fim de possibilitar a implementação de SN compostos também por SP, foi criado uma estrutura auxiliar de SN que foi chamada de sn_main. Há uma regra para o SN acompanhado de SP e outra regra para o caso comum de o SN aparecer sozinho na sentença. Detalhes da implementação em Prolog são apresentados no Quadro 19.

```
%construção do SN geral, sem outros sintagmas aninhados
sn_main ( _ , DetPrim, Especif, Conc, Pes , Anim, Seman,SN) --> sn( _ , DetPrim, Especif, Conc, Pes , Anim, Seman , SN).

%construção do SN geral, com outros sintagmas aninhados
%no presente projeto serão validadas somente aquelas cujo sintagma preposicional aninhado é formado por de+SN
sn_main( _, DetPrim, Especif, Conc, Pes, Anim, Seman, [SN,SP]) -->
            sn (_, DetPrim, Especif, Conc, Pes, Anim, Seman, SN), sp(_, de,_,_, SP).
```

Quadro 19 – Código em Prolog do sn_main

Na prática, esse artifício auxiliar (sn_main) permite que se tenham SN ligados por preposições, como em:

... os amigos de infância...

No trecho destacado, o SN *os amigos de infância* é o sintagma nominal principal (sn_main) e é formado por dois SNs, *os amigos* e *infância*, que se ligam através da preposição *de*.

No Quadro 20 as regras dos SN propriamente ditos (derivados a partir de sn_main):

```
sn( _  , semdetprim , [especif], Conc, 3  , Anim, Seman        ,  NN     ) --> nn(Conc, Anim, Seman, NN).
sn( _  , semdetprim , [especif], Conc, 3  , Anim, Seman        , [NN,Modif]) --> nn(Conc, Anim, Seman, NN), sadj(Conc, Modif).
sn(suj ,  n/a    , [anaf], Conc, Pes, Anim, _           , PPCR    ) --> ppcr(Conc, Pes, Anim, PPCR).
sn( _  , DetPrim   , [especif], Conc, 3  , Anim, Seman        , [Det,NN]  ) --> det(DetPrim, Conc,Det), nn(Conc, Anim, Seman, NN).
sn( _, DetPrim, [especif], Conc, 3, Anim, Seman, [Det,NN,Modif]) -->
          det(DetPrim, Conc,Det), nn(Conc, Anim, Seman, NN), sadj(Conc, Modif).
sn( _  , comdetprim , [anaf   ], Conc, 3  ,  _ , _     , Det     )     --> det2(Conc, Det).
sn( _  , comdetprim , [anaf], Conc, 3  ,  _ , _        , Det     )     --> det3(Conc, Det).
sn( _  , comdetprim , [anaf   ], Conc, 3  ,  _ , _        , Det     )  --> det4(Conc, Det).
sn(obj , semdetprim , [especif ], Conc, 3  ,  _ , _        , Det     )  --> det5(Conc, Det).
```

Quadro 20 – exemplo de Código em Prolog do SN

[79] Nesse caso fica bem clara a decisão de projeto ter como foco principal a análise sintática. Ao mesmo tempo, deixa-se evidente também que os aspectos semânticos da linguagem não foram renegados e nem omitidos; pelo contrário, à medida do possível e ao menos tangencialmente, são também tratados.

Há oito atributos modelados para o SN. Eles definem o perfil do SN que servirá de base às regras de produção na validação das sentenças. A seguir, é mostrada a função de cada atributo e seus possíveis valores :

- Primeiro Atributo: Esse SN pode ser sujeito, objeto ou ambos?
    i. suj: somente sujeito;
    ii. obj: somente objeto;
    iii. _ : ambos
- Segundo Atributo: há determinante primário?
    i. comdetprim : sim, há DP;
    ii. semdetprim : não há DP;
    iii. DetPrim : há DP e essa variável segurará o valor de DP;
    iv. nhp: não há possibilidade.
- Terceiro Atributo: o SN é anafórico ou é específico?
    i. anaf
    ii. especif
- Quarto Atributo: guarda os valores de concordância, [gn,nr][80]
    i. Conc
- Quinto Atributo: Pessoa – primeira, segunda ou terceira pessoa[81]
    i. 1
    ii. 2
    iii. 3
    iv. Pes – variável
- Sexto Atributo: atributos do SN enquanto substantivo[82]
    i. Anim – variável
    ii. _ - tanto faz para o SN em questão
- Sétimo Atributo: atributos do valor semântico da expressão[83]

---

[80] Essa parte se deve a Pereira & Schieber (1987). Em trabalhos de LP já foram utilizados por Pagani (2004) e Othero (2006).

[81] Em geral, será terceira pessoa.

[82] Na prática, herda o atributo *como_subst* do nominal (rever classificação do nominal) que é núcleo do SN para fins de validação semântica.

- Oitavo Atributo: constituintes que formam o SN
    i. varia muito, conforme o lado direito das regras de produção

Um SN, a partir desses valores, passa a ter uma identidade, que ajudará a verificar as concordâncias da frase, conforme as regras de produção.

Para ilustrar melhor o processo de derivação das regras de produção, nesse projeto serão utilizadas figuras didáticas. Algumas considerações sobre a simbologia utilizada nessas figuras precisam ser explicadas a priori para um melhor entendimento.

Nesses gráficos utilizou-se o recurso da legenda para diferenciar 3 formatos diferentes que representam elementos diferentes, conforme a Figura 4:

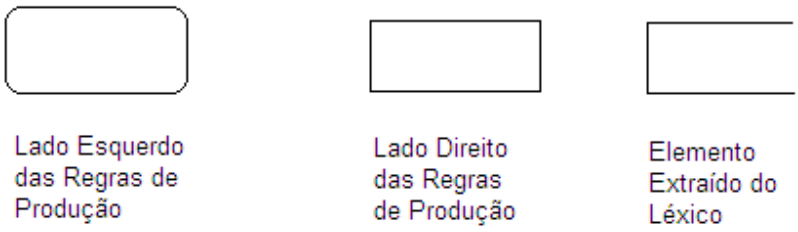

Figura 4 -Legenda didática

Através das cores também é possível diferenciar as regras de produção, pois cada cor representa uma regra de produção diferente.

A Figura 5 exemplifica a forma de derivação utilizada pelo SIMPLES para o SN *essa bela construção*.

Nesse exemplo, a primeira regra de produção (em azul) ilustra o código em Prolog transcrito no Quadro 21:

```
sn(_,DetPrim, especif, Conc, 3, Anim, Seman, [Det,NN])-->det(DetPrim,Conc,Det), nn(Conc,Anim,Seman,NN).
```

Quadro 21 – Exemplo de código em Prolog do SN da Figura 5

Na Figura 5 é mostrada a cadeia de derivação que torna a expressão *essa bela construção* válida à luz da gramática do SIMPLES; porém, é possível que até chegar a essas regras

---

[83] Na prática, herda o atributo *seman* do nominal (rever classificação do nominal) que é núcleo do SN para fins de validação semântica.

específicas da Figura 5 (até achar esse caminho) muitas outras[84] derivações tenham sido testadas pelo Prolog.

O atributo *Conc*, em destaque com cor diferente e presente tanto em *sn*, como em *det* e em *nn*, ilustra bem o que o Prolog procura "casar" em sua busca persistente pelo "caminho perfeito". Todos os atributos *Conc* presentes no SN em questão são [fem,sg].

Outros casamentos destacados na Figura 5 é que *bela* é um nominal com o campo PN setado e *construção* é um nominal com NSN setado, exatamente como requer a regra de produção "em verde" na Figura 5[85].

[84] O Prolog testará todas as regras, se for necessário, até achar às que se encaixam com a frase de entrada; enquanto não achar esse caminho, ele vai procurando; se cessarem todas as possibilidades, o Prolog conclui que a sentença de entrada não é válida à luz daquela gramática.

[85] Por simplificação, na Figura está abstraída a forma como a expressão "essa bela construção" foi submetida ao processo de validação à luz da gramática em questão.

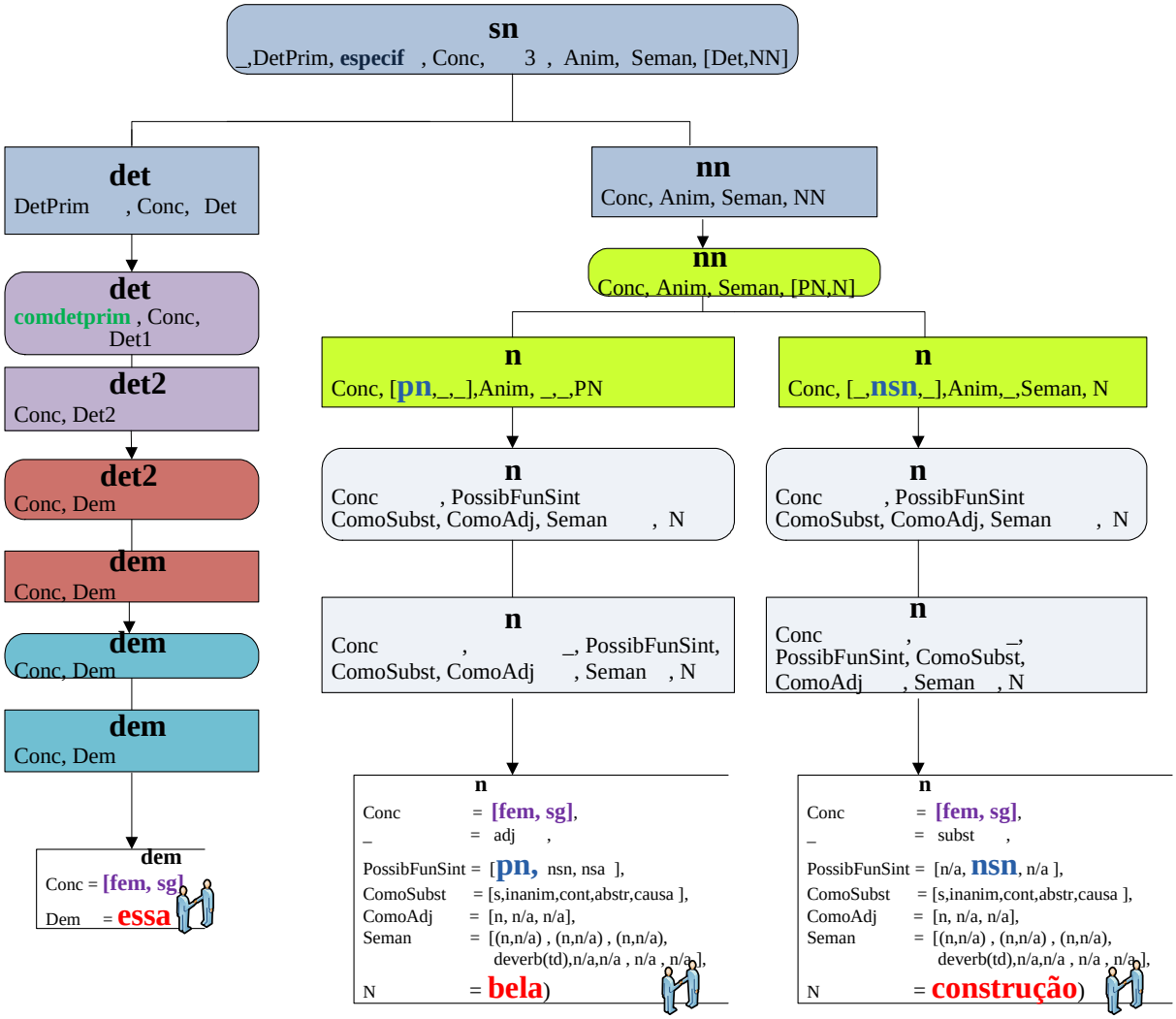

Figura 5 – Árvore de derivação do SN “essa bela construção”

No Quadro 22 estão as regras de produção do SIMPLES utilizadas na derivação esboçada Figura 5 que obteve sucesso[86].

[86] Reitera-se que, até achar uma regra que seja válida para uma determinada oração, o Prolog fará várias buscas, utilizando o mecanismo de *backtracking*, testando todas as possibilidades de derivações para aquela sentença.

```
sn( _  , DetPrim    , [especif], Conc, 3  , Anim, Seman , [Det,NN]  ) -->
        det(DetPrim, Conc,Det), nn(Conc, Anim, Seman, NN).
det(comdetprim, Conc,Det2) -->
        det2(Conc, Det2).
nn(Conc, Anim, Seman, [PN,N]) -->
        n(Conc, [pn,_,_],Anim, _,_,PN), n(Conc, [_,nsn,_],Anim, _,Seman, N).
det2( Conc, Dem) -->
        dem(Conc,Dem).
dem(Conc, Dem) -->
        [Dem], {dem(Conc, Dem)}.
n(Conc, PossibFunSint,ComoSubst, ComoAdj,Seman,N) -->
        [N], { n(Conc, _, PossibFunSint, ComoSubst, ComoAdj,Seman, N)}.
dem([fem, sg], essa).
n([fem, sg], subst   ,  [pn,  nsn, nsa  ], [s, _   , _   , _ , _  ],
        [s, aval, tipico] , [(n,n/a) , (n,n/a) , (n,n/a) , n/a, n/a,  n/a ,  n/a , n/a],    bela).
n([fem, sg], subst, [n/a, nsn, n/a ], [s, inanim, cont,abstr,  causa ], [n,  n/a, n/a],
        [(n,n/a) , (n,n/a), (n,n/a), deverb(td), n/a  ,  n/a,  n/a , n/a ], construção).
```

Quadro 22 – regras de produção do SIMPLES referente à derivação do SN “essa bela construção” esboçada na Figura 5, codificada em Prolog.

5.3.2. Sintagma Verbal (SV)

Em orações simples, por definição, esse é o único sintagma obrigatório. Nele está contido o elemento principal de qualquer oração: o verbo.

Os atributos do SV são os seguintes:

- Primeiro Atributo: o verbo está no sg ou no pl?
- Segundo Atributo: qual pessoa verbal?
- Terceiro Atributo: quais construções são admitidas?
- Quarto atributo: positivo ou negativo (frases negativas)
- Quinto Atributo: constituintes que formam o SV
    i. varia conforme o lado direito das regras de produção

5.3.2.1 Implementação do SV em Prolog

Segue um exemplo de derivação, através da Figura 6, onde é percorrida a árvore de derivação do SV “correu demais”.

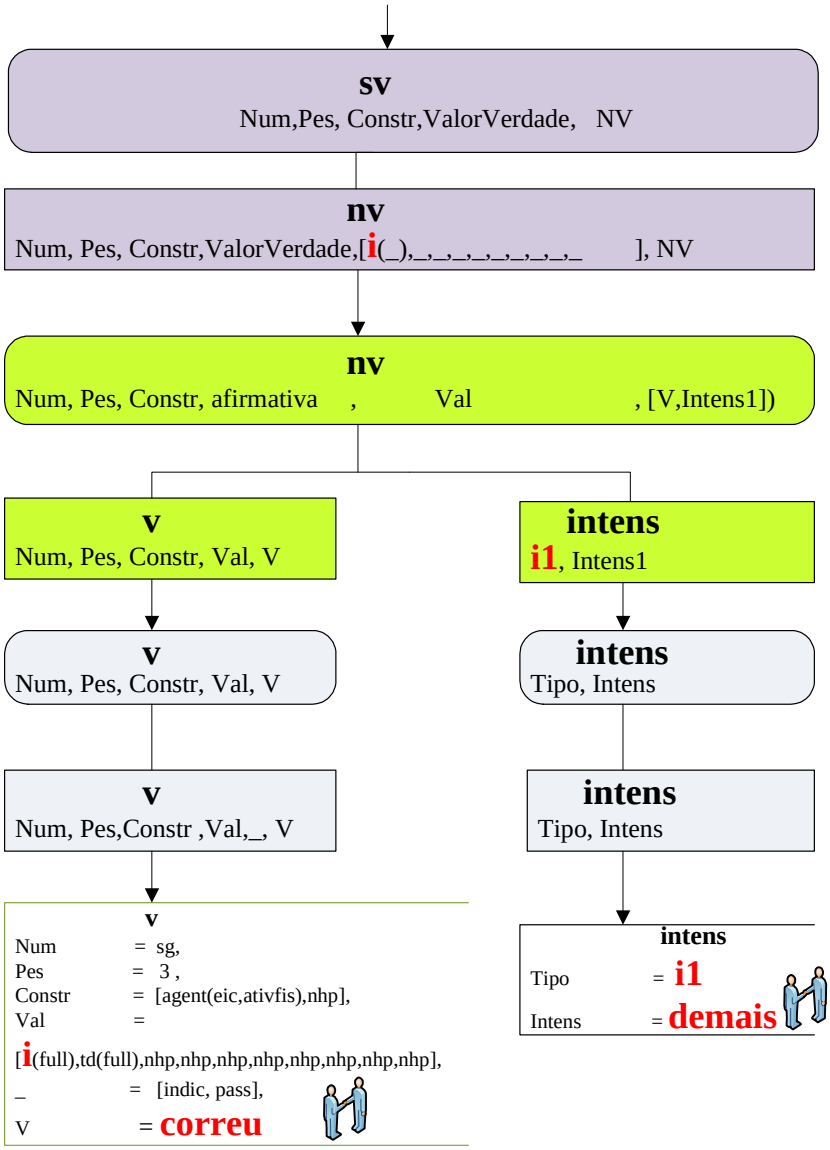

Figura 6 - árvore de derivação do SV “correu demais”

No Quadro 23 estão as regras de produção do SIMPLES utilizadas com sucesso na derivação do gráfico na Figura 6. O código está em Prolog.

```
sv(Num, Pes, Constr , ValorVerdade,   NV       )
 --> nv(Num, Pes, Constr, ValorVerdade, [i(_),_,_,_,_,_,_,_,_,_        ], NV).

nv(Num, Pes, Constr, afirmativa, Val , [V,Intens1])
--> v(Num, Pes, Constr, Val, V), intens(i1, Intens1).

v(Num, Pes, Constr, Val, V)
--> [V], {v(Num, Pes,Constr ,Val,_, V)}.

intens(Tipo, Intens) --> [Intens], {intens(Tipo, Intens)}.

intens(i1, demais).

v(sg,3, [agent(eic,ativfis),nhp],[i(full),td(full),nhp,nhp,nhp,nhp,nhp,nhp,nhp,nhp],
[indic, pass], correu).
```

Quadro 23 – código do SIMPLES em Prolog referente ao SV “correu demais”

5.3.3 Sintagma Adverbial (SAdv)

Ainda há muitas dúvidas teóricas a respeito de como organizar os dados das funções ditas adverbiais[87]. O que não há dúvida é que os SAdv têm a função semântica de esclarecer as circunstâncias de situações descritas na oração. Nas frases 13 e 14 são dados exemplos de quais são essas circunstâncias.

(13) Hoje o aluno fez as tarefas lentamente.

(14) Amanhã, na prova, ele resolverá as questões mais rápido.

Nas duas frases há os constituintes *lentamente* e *mais rápido,* que conferem semântica de velocidade; há também as palavras *hoje* e *amanhã*, que fazem contextualização temporal; e especificamente na frase 14 há a expressão *na prova*, que faz a contextualização espacial. Todos os elementos sublinhados nos exemplos acima são considerados como SAdv.

Além dos três tipos exemplificados nas frases 13 e 14, há vários outros elementos circunstanciais possíveis de serem descritos através do SAdv.

O fato de não haver um posicionamento sintático fixo desses sintagmas dentro da estrutura sintática, aliado à falta de um estudo teórico aprofundado sobre a sua tipificação, dificulta muito a modelagem desses sintagmas[88].

Por esses motivos, o SIMPLES também não se aprofunda no tema e se limita a validar algumas ocorrências mais comuns dos SAdv, quais sejam os referentes a Modo, Tempo e Localização.

---

87 Alguns teóricos chegam inclusive a ter restrições de considerar o adverbial como uma categoria única. Perini assim escreve sobre o assunto:

> Como não se fez ainda um estudo das funções possíveis de tais sintagmas com base em um conjunto confiável de funções sintáticas, torna-se difícil no momento definir não só o "sintagma adverbial"(que,de qualquer modo, não deve ser uma categoria única), mas ainda um conjunto de funções que ocupem o lugar do "sintagma adverbial". (PERINI, 2009, p.119)

Já Castilho (2010) escreve, em sua recente gramática, um capítulo inteiro (cap.13, p. 542-582) somente sobre esse sintagma.

Não será objeto de análise desse projeto qual a melhor adequação ao tema. Mas, mais uma vez, reforça-se o posicionamento pragmático do SIMPLES, ou seja, o que nos interessa, realmente, é o quão aplicável computacionalmente é determinado conceito.

88 No SIMPLES, alguns vocábulos, classificados como advérbios segundo a maioria das gramáticas pesquisadas, exceção feita a de Perini (2009, 2010), não são considerados nem como advérbio, nem como sintagma adverbial. Esse é o caso dos intensificadores, que se encontram normalmente ligados a outros sintagmas como elementos intensificadores desses.

5.3.3.1 Implementação do SAdv em Prolog

Na Figura 7 é mostrado um exemplo de uma dessas estruturas adverbiais mais frequentes que foi objeto de modelagem. Trata-se do SAdv de tempo. A primeira estrutura pontilhada quer dizer que ela estará no lado direito de alguma regra de produção que, no exemplo, é desconsiderado, pois o foco é mostrar o SAdv.

No Quadro 24 estão trechos de código do SIMPLES em Prolog exemplificando alguns SAdv modelados.

```
% exemplo da figura 7
sadv(Tem, SN) --> sn_main(_,_,_,_,_,_, [ _,_,(adv,Tem),_,_,_,_,_], SN).


% alguns outros sintagmas adverbais
sadv(Tem, SP) --> sp(_,em,_,[ _,(Tem,_),_,_,_,_,_,_], SP).
sadv(Loc, SP) --> sp(_,em_,_,[(Loc,_),_,_,_,_,_,_,_],SP).
sadv(Loc, SN) --> sn_main(_,_,_,_,_,_, [_,_,(adv,Loc),_,_,_,_,_], SN).
```

Quadro 24 – Exemplo de código em Prolog referente ao SAdv

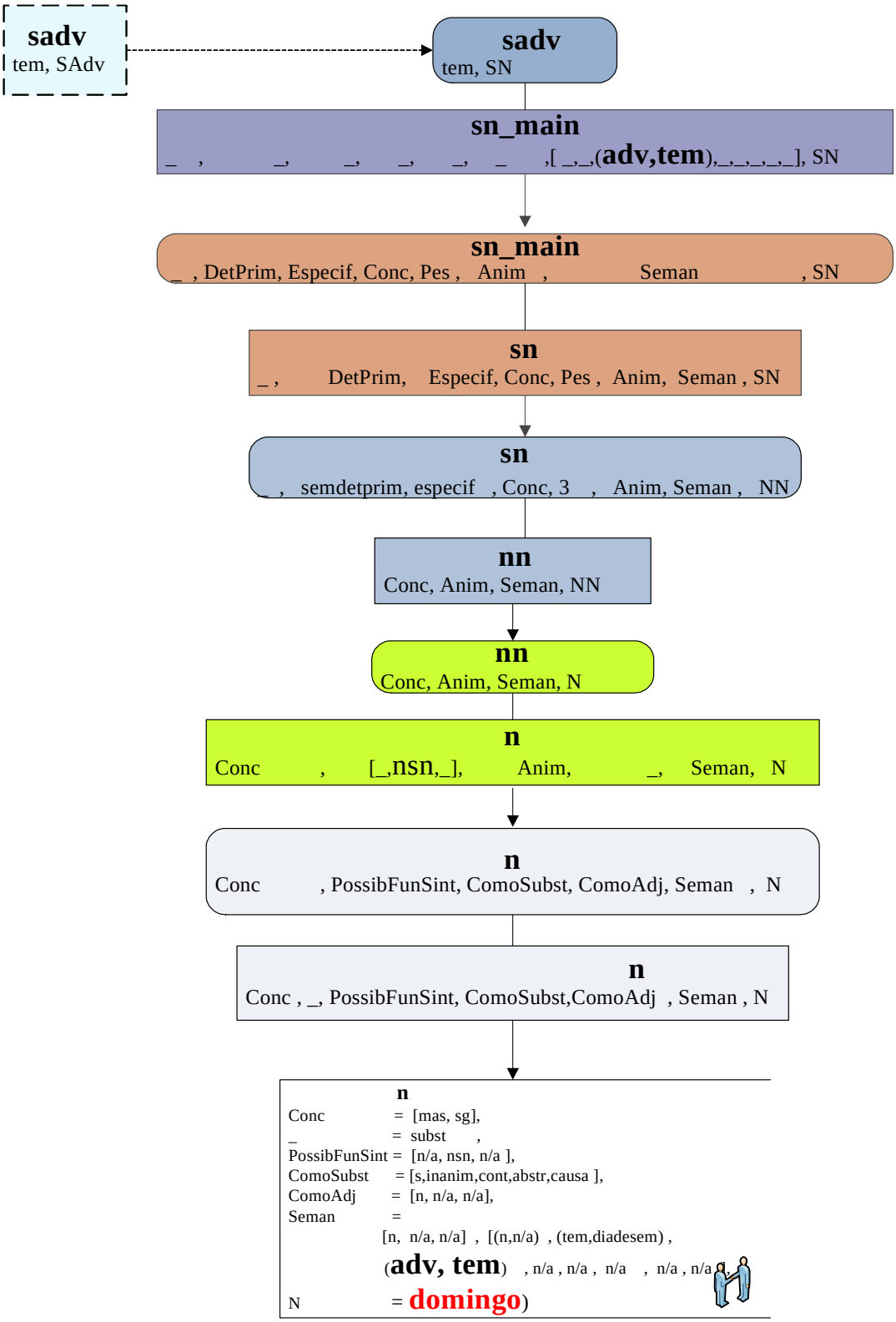

Figura 7 - Árvore de derivação do SAdv “domingo”

5.3.4 Sintagma Adjetival (SAdj)

O SAdj é prototipicamente formado por um adjetivo (um nominal com o traço núcleo do sintagma adjetival (NSA) setado), cercado ou não de intensificadores.

A modelagem do SAdj foi fortemente influenciada por Perini (2009, 2010).

Em Perini (2009, p.115), é apresentado uma ordem específica de combinações possíveis de intensificadores que podem servir como complementos ao núcleo do SAdj; o SIMPLES faz uma adaptação das idéias de Perini (Ibid), com alguns ajustes *ad hoc*, para formar algumas combinações possíveis[89].

5.3.4.1 Implementação do SAdj em Prolog

No Quadro 25 se encontra um exemplo do código em Prolog do SAdj.

```
%Sintagmas Adjetivais
sadj(Conc, Adj)-->
            n(Conc, [_,_,nsa],_, _, _,Adj).

sadj(Conc,[Intensific, Adj] ) -->
    intensific_adj(preN_adjqq,Intensific),
                n(Conc,[_,_,nsa],_, _,_ ,Adj).
```

Quadro 25 – Exemplo de código em Prolog referente ao SAdj

Há dois tipos de SAdj exemplificado: o primeiro é quando o SAdj é representado pelo adjetivo (um nominal com atributo nsa setado) e o segundo remete para uma estrutura auxiliar que comporta várias combinações de pré-núcleo adjetival, representado por possíveis combinações de intensificadores, como mostra o Quadro 27.

O Quadro 26 lembra alguns dos intensificadores mais utilizados.

```
intens(i6,      realmente).
intens(i5,      sempre).
intens(i4,      muito).
intens(i3,      bem).
intens(i2,      mais).
intens(i1,      demais).
```

Quadro 26 – Exemplo de código em Prolog referente aos intensificadores

[89] É justo que se diga que nem Perini (2009) é exaustivo em todas as possibilidades de ocorrências de SAdj, nem o SIMPLES o é. Por pertinente ressalte-se que, ao se dizer que o SIMPLES se baseou fortemente na classificação de Perini não há nessa afirmação um comprometimento em segui-la como um dogma; fez-se tantas adaptações quanto se achou necessárias.

```
%Combinações de intensificadores pré-nucleos adjetivais para qq adjetivo
intensific_adj(preN_adjqq, Intens6)                    -->
        intens(i6, Intens6).
intensific_adj(preN_adjqq, Intens5)                    -->
        intens(i5, Intens5).
intensific_adj(preN_adjqq, Intens4)                    -->
        intens(i4, Intens4).
intensific_adj(preN_adjqq, Intens3)                    -->
        intens(i3, Intens3).
intensific_adj(preN_adjqq, Intens2)                    -->
        intens(i2, Intens2).
intensific_adj(preN_adjqq, [Intens6, Intens5])              -->
        intens(i6, Intens6) ,  intens(i5, Intens5).
intensific_adj(preN_adjqq, [Intens6, Intens4])              -->
        intens(i6, Intens6) ,  intens(i4, Intens4).
intensific_adj(preN_adjqq, [Intens6, Intens3])              -->
        intens(i6, Intens6) ,  intens(i3, Intens3).
intensific_adj(preN_adjqq, [Intens6, Intens2])              -->
        intens(i6, Intens6) ,  intens(i2, Intens2).
intensific_adj(preN_adjqq, [Intens5, Intens4])              -->
        intens(i5, Intens5) ,  intens(i4, Intens4).
intensific_adj(preN_adjqq, [Intens5, Intens3])              -->
        intens(i5, Intens5) ,  intens(i3, Intens3).
intensific_adj(preN_adjqq, [Intens5, Intens2])              -->
        intens(i5, Intens5) ,  intens(i2, Intens2).
intensific_adj(preN_adjqq, [Intens4, Intens2])              -->
        intens(i4, Intens4) ,  intens(i2, Intens2).
intensific_adj(preN_adjqq, [Intens3, Intens2])              -->
        intens(i3, Intens3) ,  intens(i2, Intens2).
intensific_adj(preN_adjqq, [Intens6, Intens5,Intens4])      -->
        intens(i6, Intens6) ,  intens(i5, Intens5), intens(i4, Intens4).
intensific_adj(preN_adjqq, [Intens6, Intens5,Intens3])      -->
        intens(i6, Intens6) ,  intens(i5, Intens5), intens(i3, Intens3).
intensific_adj(preN_adjqq, [Intens6, Intens5,Intens2])      -->
        intens(i6, Intens6) ,  intens(i5, Intens5), intens(i2, Intens2).
intensific_adj(preN_adjqq, [Intens6, Intens4,Intens2])      -->
        intens(i6, Intens6) ,  intens(i4, Intens4), intens(i2, Intens2).
intensific_adj(preN_adjqq, [Intens6, Intens3,Intens2])      -->
        intens(i6, Intens6) ,  intens(i3, Intens3), intens(i2, Intens2).
intensific_adj(preN_adjqq, [Intens5, Intens4,Intens2])      -->
        intens(i5, Intens5) ,  intens(i4, Intens4), intens(i2, Intens2).
intensific_adj(preN_adjqq, [Intens5, Intens3,Intens2])      -->
        intens(i5, Intens5) ,  intens(i3, Intens3), intens(i2, Intens2).
intensific_adj(preN_adjqq, [Intens6, Intens5,Intens4,Intens2])  -->
        intens(i6, Intens6) ,  intens(i5, Intens5), intens(i4, Intens4),   intens(i2, Intens2).
intensific_adj(preN_adjqq, [Intens6, Intens5,Intens3,Intens2])  -->
        intens(i6, Intens6) ,  intens(i5, Intens5), intens(i3, Intens3),   intens(i2, Intens2).
```

Quadro 27 – Exemplo de código em Prolog referente a combinações de intensificadores

5.3.5. Sintagma Preposicional (SP)

Essa é uma composição que tem uma enorme flexibilidade de utilização. Ele pode tanto aparecer complementando verbos e nominais, como exercer, diretamente, as funções de SAdv ou SAdj.

Uma diferenciação precisa ser feita para os casos em que a preposição está na forma contraída. Esse cuidado é implementado com a inserção de um atributo para segurar esse valor, como se verá através do exemplo.

Os atributos do SP são os seguintes:

- Primeiro Atributo: reservado para potencial utilização[90]
- Segundo Atributo: guarda qual preposição foi instanciada
- Terceiro Atributo: classificação do elemento secundário que o forma[91]
- Quarto Atributo: a semântica do elemento secundário que o forma
- Quinto Atributo: varia conforme as derivações do lado direito da regra de produção

5.3.5.1 Implementação do SP em Prolog

Para ilustrar o SP, a Figura 8 traz a implementação em Prolog de um SP.

[90] Esse atributo “livre” foi inserido para facilitar a inclusão de novas funcionalidades em versões posteriores.

[91] Geralmente um SN.

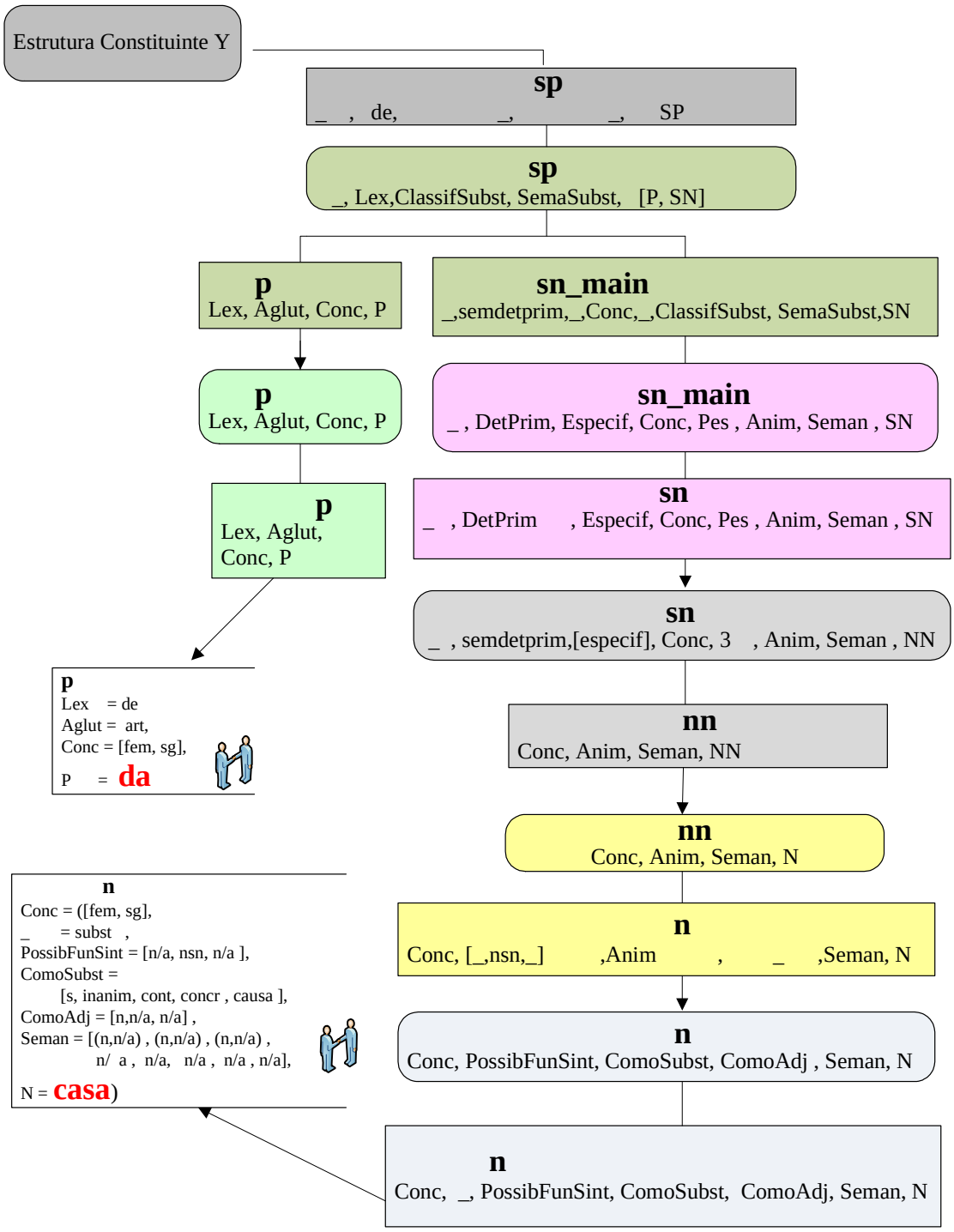

Figura 8 - Árvore de derivação do SP “da casa”

5.4. Construções

São espécies de *templates* das orações. Num conceito mais formal, seguindo Perini (2010), a construção se define por:

- os tipos de constituintes de que é formada uma oração (SN, SV e SAdv, no caso do SIMPLES);
- a função de cada um deles na oração (sujeito, objeto...);
- os papéis temáticos de cada um deles (Agente, Paciente, Experienciador..).

O mais importante a destacar é que o verbo define qual o tipo de construção nas orações em que ele é o núcleo verbal. O verbo seleciona quais são (ou quais podem ser) os demais constituintes da oração em que ele é o núcleo verbal, bem como os papéis temáticos que esses constituintes terão (ou poderão ter) naquela oração. Por esses motivos, uma classificação adequada dos verbos é pressuposto básico de uma modelagem linguística bem feita das construções. A classificação ora apresentada na Figura 9 é fortemente influenciada pela pesquisa em trabalhos científicos de Linguística[92], mas também é resultado de inferências, o que lhe confere uma boa dose empírica[93].

[92] Notadamente Cançado (1995,1996, 2000,2002,2005), Moreira (2000), Ciríaco (2007), Berg (1996) e Godoy (2008).

[93] Nesse ponto faz-se necessário lembrar que não foi encontrado, em nenhuma bibliografia pesquisada, um rol exaustivo de todas as construções possíveis, nem mesmo uma relação que esgote todos os papéis temáticos possíveis. Ao contrário, segundo os autores pesquisados esse é um caminho ainda em aberto. Então, o que fez nesse projeto pode ser considerado no máximo uma tentativa de sistematização da classificação verbal. Portanto, deixa-se bem claro que a classificação dos verbos ora apresentada quanto às construções em que podem ocorrer foi resultado de muitas inferências. Embora essas tenham sido baseadas em pesquisas bibliográficas realizadas, há um um componente empírico irrefutável. Ainda assim, os resultados foram bons o suficiente para serem implementados.

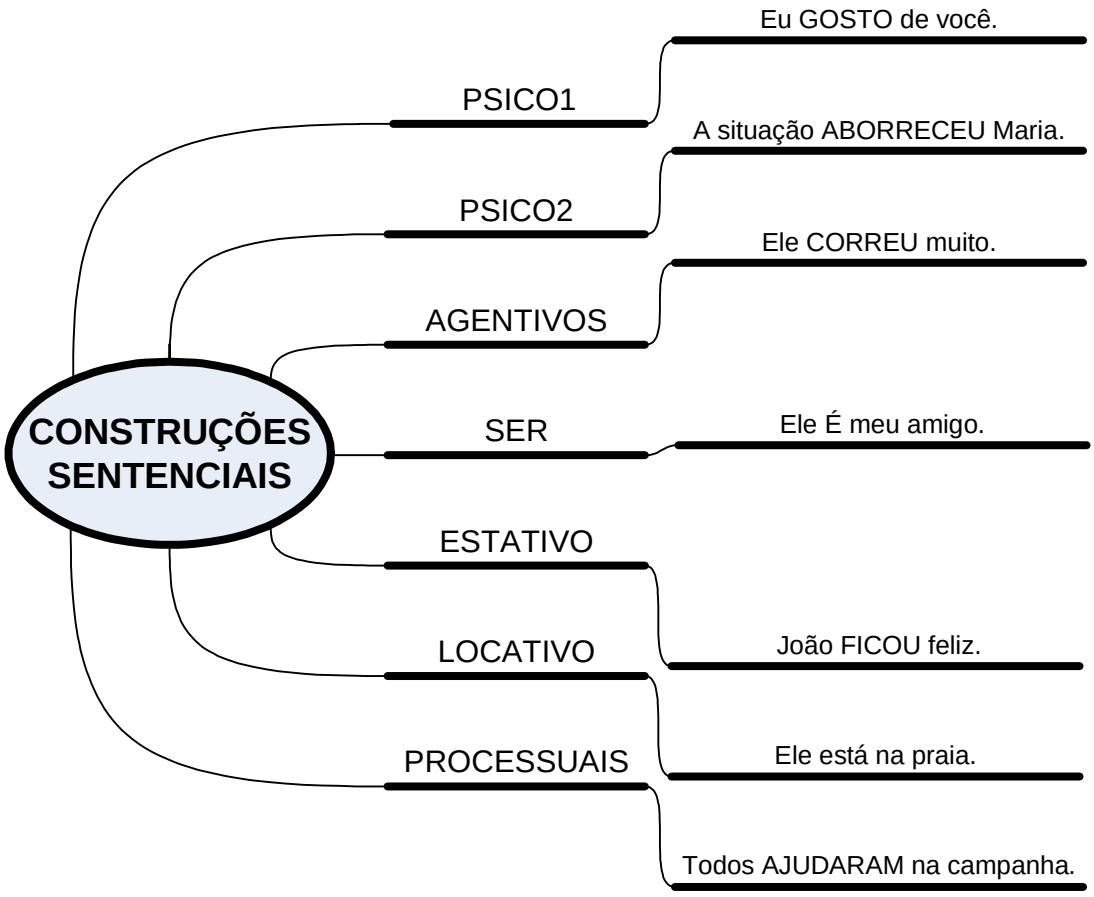

Figura 9 - Construções Sentenciais

5.4.1 Construções sentenciais prototípicas

Para uma maior clareza, é apresentado nessa subseção um exemplo de implementação em Prolog de cada tipo de construção modelada. Será mostrada a ocorrência prototípica, ou seja, aquela em que os termos estão na sentença de entrada na ordem direta padrão (em termos da GT, na sequência Sujeito-Predicado) e sem a presença de SAdv oracionais. Em outras palavras, elas estão na sequência SN – SV.

5.4.1.1 PSICO1

```
s([SN, SV, Ponto])      -->
                    sn_main(suj,_, _,[_,Num], Pes, [_, anim(rac,_),_ ,_,_], _, SN),
                    sv(Num, Pes, [psico1,_],_, SV),
                    ponto(Ponto).
```

Quadro 28 – Exemplo de código em Prolog referente a PSICO1

5.4.1.2 PSICO2

```
s([SN, SV_OD_Exper, Ponto]) -->
                    sn_main(suj,_, _,[_,Num], Pes, _, _, SN),
                    sv_OD_Exper(Num, Pes, [psico2,_],_, SV_OD_Exper),
                    ponto(Ponto).
```

Quadro 29 – Exemplo de código em Prolog referente a construções com verbos do tipo PSICO2

5.4.1.3 AGENTIVO

```
s([SN, SV, Ponto]) -->
          sn_main(suj,_, _,[_,Num], Pes, [_, _,_ ,_,causa], _, SN),
          sv(Num, Pes, [agent(eec,mudaest),_],_, SV),
          ponto(Ponto).
```

Quadro 30 – Exemplo de código em Prolog referente a construções com verbos do tipo agentivo

5.4.1.4 SER

```
s([SN, SV, Ponto]) -->
        sn_main(suj,_, _,[_,Num], Pes, [_, _,_ ,_,causa], _, SN),
        sv(Num, Pes, ser,_, SV),
        ponto(Ponto).
```

Quadro 31 – Exemplo de código em Prolog referente a construções com verbos do tipo ser

5.4.1.5 ESTATIVO

```
s([SN, SV, Ponto]) -->
        sn_main(_,_, _,[_,Num], Pes, [_, _,_ ,_,_], _, SN),
        sv(Num, Pes, estat(posse),_, SV),
        ponto(Ponto).
```

Quadro 32 – Exemplo de código em Prolog referente a construções com verbos do tipo estativo

5.4.1.6 LOCATIVO

```
s([SN, SV, Ponto])      -->
                sn_main(suj,_, _,[_,Num], Pes, [_, _,_ ,_,_], _, SN),
                sv(Num, Pes, [locat,_],_, SV),
                ponto(Ponto).
```

Quadro 33 – Exemplo de código em Prolog referente a construções com verbos do tipo locativo

5.4.1.6 PROCESSUAL

```
s([SN, SV, Ponto])      -->
        sn_main(suj,_, _,[_,Num], Pes, [_, anim(_),_ ,_,_], _, SN),
        sv(Num, Pes, [proces(_),_],_, SV),
        ponto(Ponto).
```

Quadro 34 – Exemplo de código em Prolog referente a construções com verbos do tipo processual

6 O PROTÓTIPO

Nessa seção faz-se uma definição do protótipo e apresenta-se um resumo do modo como se deu o processo de modelagem, implementação e testes do programa. Adicionalmente, faz-se uma análise dos resultados obtidos com os testes.

6.1 Definição e apresentação

Pode-se resumir o protótipo como sendo um conjunto de fatos e regras implementados em Prolog, onde os fatos representam um fragmento modelado do Léxico da LP e as regras são constituídas por um segmento modelado das construções sintáticas possíveis da Língua. Há alguma validação semântica no protótipo à medida que determinadas construções só são reconhecidas se determinados sintagmas tiverem certos atributos semânticos. Abstraindo essa parte da Semântica, poder-se-ia resumir o protótipo como segue:

Prolog	**SIMPLES**
fatos	Léxico
regras	Sintaxe

Quadro 35 – Prolog e a gramática do SIMPLES

6.2 Implementação, modelagem e testes

No fluxograma mostrado na Figura 10 está sintetizada a forma como se deu o processo de fabricação do *software*.

Como se pode depreender da Figura, a implementação do *parser* se deu em um processo de constante avaliação/depuração do código, em que o prazo acabou sendo o limitador inerente. Mesmo dentro do escopo determinado a priori, a presença do grupo de testes, formado por cinco pessoas que tem no ato de escrever diariamente uma de suas atribuições laborais[94], acabou sendo fundamental na depuração de alguns *bugs* e comprovar a eficácia da modelagem em outros pontos.

[94] Agradecimento especial aos amigos e colegas de trabalho Márcio, Leandro, Heber, Razera e Gilberto, que gentilmente colaboraram com esse projeto, fazendo parte do grupo de testes.

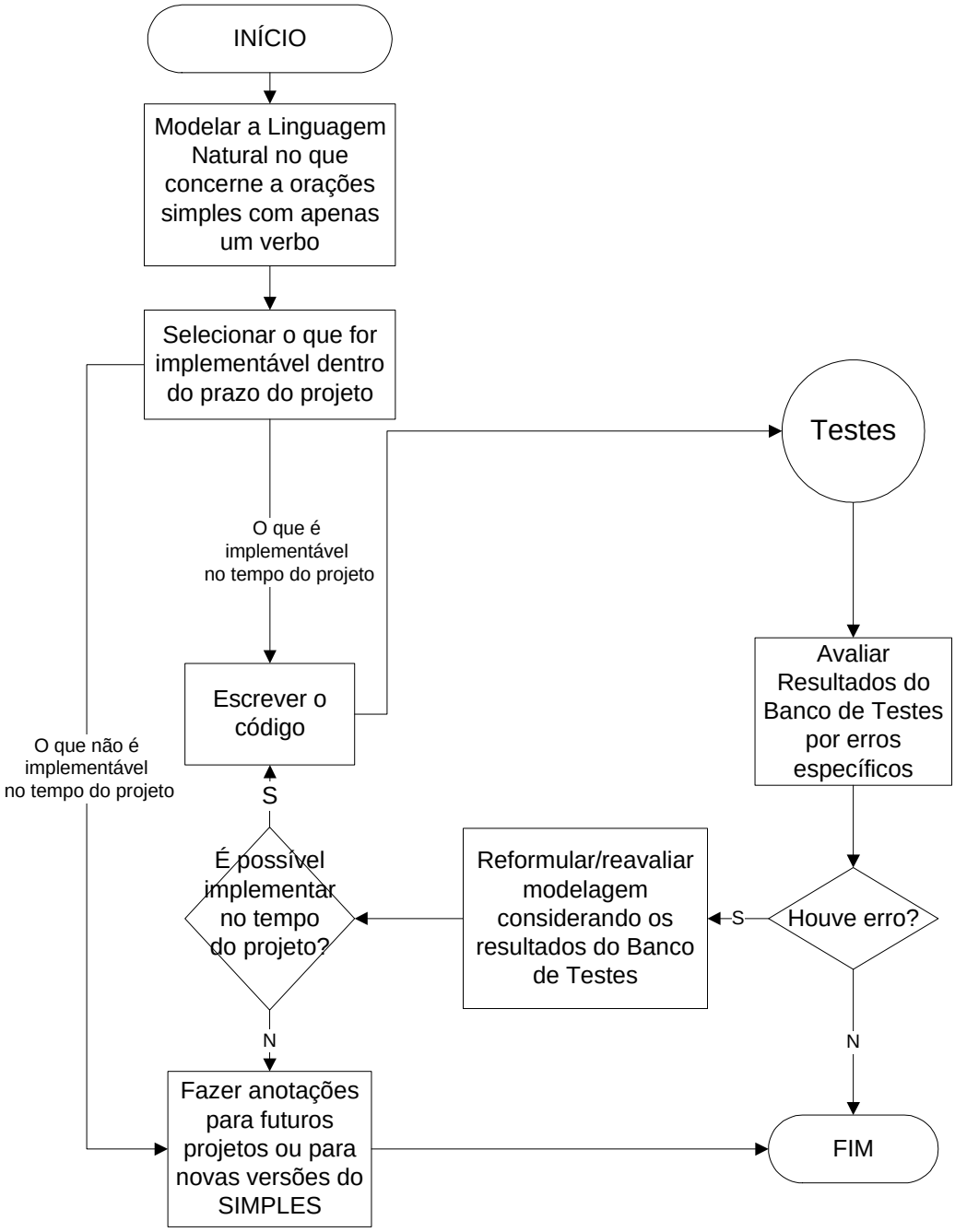

Figura 10 – Fluxograma do projeto

A forma de colaboração do grupo de testes nesse projeto pode ser resumida no fluxograma da Figura 11.

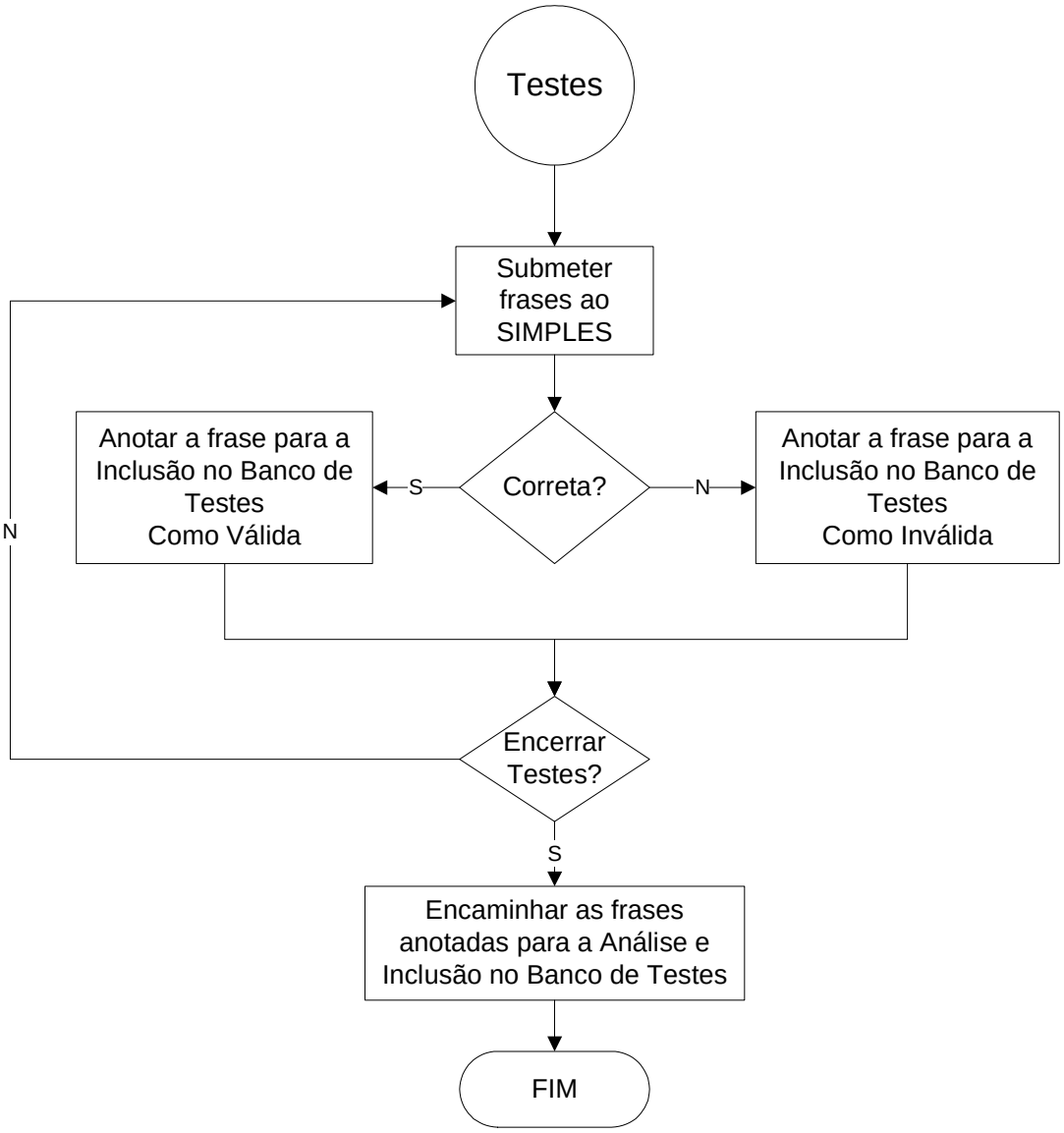

Figura 11 – Fluxograma da fase de testes

6.3 Análise dos resultados dos testes

A seguir a enumeração dos principais problemas e das principais virtudes observadas pelo grupo de testes durante a testagem do protótipo. Apresenta-se, quando for o caso, uma frase-exemplo para ilustrar qual aspecto do protótipo foi criticado ou elogiado nos testes.

As principais críticas, algumas já esperadas, foram as seguintes:

- vocabulário limitado, já que o usuário precisa verificar na lista de léxico fornecida quais palavras poderiam ser testadas.

(15) o futebol estava bem bom.

(incorretamente invalidada)

- ausência de *layout* amigável
- diferenciação de maiúscula no início da frase.
- falta de mensagens de erro que apontem qual o tipo de erro na sentença

O que o grupo de testes mais gostou e exaltou do protótipo foram:

- tratamento semântico, especialmente quanto aos verbos psicológicos

(16) A estante gosta de Maria.

(corretamente validada)

- concordância tanto verbal quanto nominal

(17) estas minhas novas amigas gostam de livros novos.

(corretamente validada)

(18) estas minhas novas amigas gostam de livros novas.

(corretamente invalidada)

- reconhecimento das estruturas com preposições, especialmente quanto ao aninhamento de SP

(19) a construção da casa nova do João empolga os homens.

(corretamente validada)

Nas figuras seguintes são apresentadas duas telas de execução do programa: a Figura 12, que mostra o comportamento do programa quando a frase de consulta é bem sucedida, e a Figura 13, que mostra quando uma frase de consulta não é aceita como correta, quando então o SIMPLES mostra uma mensagem padrão dando o *feedback* ao usuário que, por algum motivo, a frase não foi aceita.

Figura 12 – Funcionamento do SIMPLES com frases aceitas

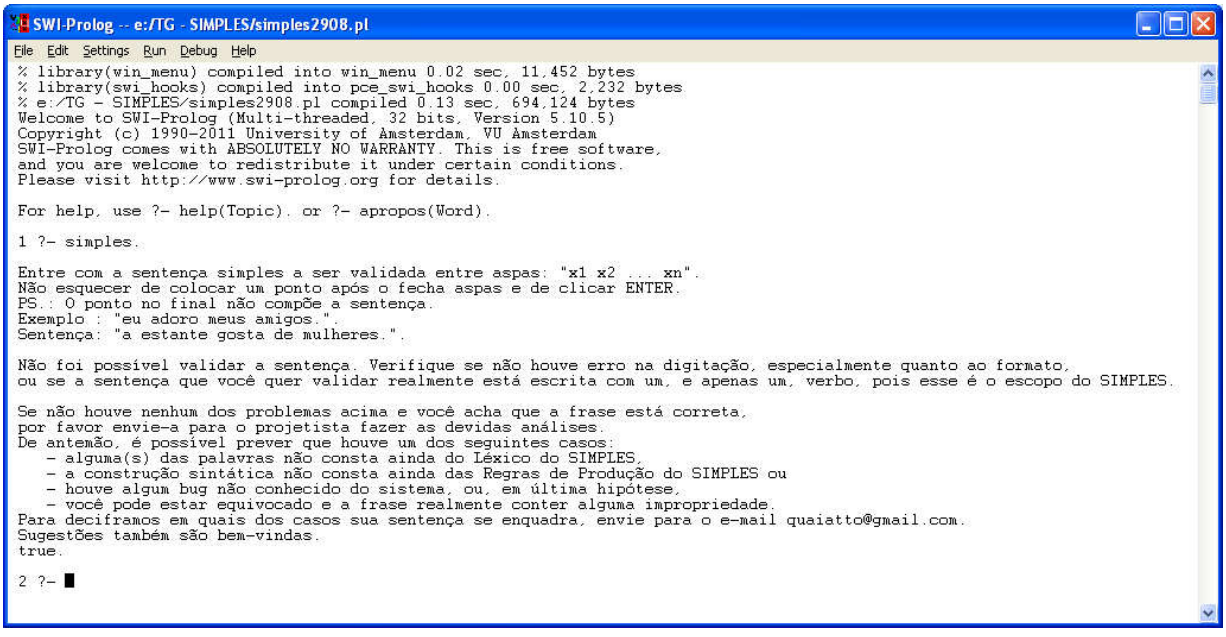

Figura 13 - Funcionamento do SIMPLES com frases não aceitas

7 CONCLUSÕES E TRABALHOS FUTUROS

Os objetivos desse projeto foram plenamente alcançados, à medida que se descobriram teorias lingüísticas realmente capazes de serem bastante úteis para a modelagem computacional das linguagens naturais. Com a implementação do SIMPLES foi possível experimentar na prática o potencial de algumas delas, sobretudo os conceitos de estruturas sintagmáticas, papéis temáticos e construções. A utilização desses conceitos permitiu que se validasse, além de uma boa gama de aspectos sintáticos, alguns aspectos semânticos de sentenças simples em LP.

Cabe destacar também que, ao se escolher Prolog como a linguagem de programação do protótipo, foi possível perceber porque ela é considerada uma das melhores para projetos de LC. E uma das razões para isso é o fato de ela possuir a ferramenta DCG, que é extremamente prática e funcional para se escrever gramáticas.

Voltando à questão das pesquisas em Lingüística, realmente foi possível constatar que há uma miríade de teorias lingüísticas com potencial para aplicações em LC. Um conceito em especial, utilizado no SIMPLES, parece ter ainda um potencial enorme a ser explorado: trata-se das construções. Essa teoria traz consigo uma pressuposição implícita que empolga a todo projetista de LC: a de que é possível se fazer um mapeamento de todas as estruturas sentenciais de uma língua. Por essa razão, a continuidade do projeto ora apresentado poderá se dar naturalmente, com a pesquisa para reconhecer novas construções sintáticas e ampliar o léxico reconhecido pelo protótipo.

Por certo, em uma continuidade de projeto, deve se melhorar alguns aspectos apontados pelo grupo de testes do SIMPLES. Uma dessas melhorias é fazer uma interface amigável com o usuário e quiçá considerar a idéia de integração com algum software de processamento de textos.

Os bons resultados obtidos com o protótipo implementado demonstram também que as idéias subjacentes a esse projeto podem servir de base ou inspiração ao aperfeiçoamento, especialmente quanto ao aspecto semântico, de outros projetos da área de LC, como tradutores automáticos de linguagens, sistemas de buscas e corretores gramaticais.

REFERÊNCIAS BIBLIOGRÁFICAS

AZEREDO, José Carlos de. **Iniciação à Sintaxe do Português**. 9ª Ed. Jorge Zahar Editor. Rio de Janeiro, 2007.

BECHARA, Evanildo. **Moderna Gramática da Língua Portuguesa**. Editora Lucerna. Rio de Janeiro, 2009.

BERG, Márcia Barreto. **A Natureza Categorial da Preposição.** Belo Horizonte. Dissertação de Mestrado, UFMG, 1996.

CANÇADO, M. **O papel do léxico em uma teoria de papéis temáticos.** *DELTA*. v.16.2, p. 297-321. Belo Horizonte, 2000.

CANÇADO, M. **Posições argumentais e propriedades semânticas.** *DELTA*. v. 21, n.1, p. 23-56. Belo Horizonte, 2005.

CANÇADO, Márcia e GODOY, Luisa. **Relacionando as estruturas semântico-lexical e sintático-lexical.** Anais Do Encontro do GT de Teoria da Gramática da ANPOLL. UNB. Brasília, 2009.

CANÇADO, M.**. Uma Aplicação da Teoria Generalizada dos Papéis Temáticos: Verbos Psicológicos.** *Revista do GEL. Número Especial: Em Memória de Carlos Franchi.* Eds.Altman C., M. Hackerott e E. Viotti.Humanitas/Contexto. São Paulo, 2002.

CANÇADO, M.. **Verbos Psicológicos: A Relevância dos Papéis Temáticos vistos sob a Ótica de uma Semântica Representacional.** Tese de Doutorado. IEL. UNICAMP.Campinas, 1995

CANÇADO, M.. **Verbos Psicológicos: Análise Descritiva dos Dados do Português Brasileiro.** *Revista de Estudos da Linguagem* 4. 1: 89-114. Campinas, 1996.

CASTILHO, Ataliba T. de. **Gramática do português brasileiro**. Editora Contexto. São Paulo, 2010.

CHOMSKY, Noam, RUWET, N. . **A Gramática Generativa**. Edições 70. Lisboa,1979.

CHOMSKY, Noam. **Aspects of the theory of sintax**. The MIT Press. Massachusetts, 1965.

CHOMSKY, Noam. **Syntactic Structures**. Monton. Haya,1957.

CIRÍACO, L**. A alternância causativo-ergativa no PB: restrições e propriedades semânticas.** Dissertação de mestrado. B. Horizonte, UFMG. 2007.

COVINGTON, Michael A.; NUTE, Donald; VELLINO, Andre. **Prolog programming in depth**. Prentice Hall. USA, 1997.

DIAS-DA-SILVA, B.C. **A face tecnológica dos estudos da linguagem: o processamento automático das línguas naturais**. Araraquara, 1996. 272p. Tese (Doutorado em Letras) - Faculdade de Ciências e Letras, Universidade Estadual Paulista. 1996.

DIAS-DA-SILVA, B.C. **O estudo linguístico-computacional da linguagem**. *Letras de Hoje*, v. 41, n. 2, p. 103-138. Porto Alegre, 2006.

DIAS-DA-SILVA, B.C.; MONTILHA, G.; RINO, L.H.M.; SPECIA, L.; NUNES, M.G.V.; OLIVEIRA Jr., O.N.; MARTINS, R.T.; PARDO, T.A.S. **Introdução ao Processamento das Línguas Naturais e algumas aplicações. Série de Relatórios Técnicos do NILC, NILC-TR-07-10**. ICMC. São Carlos, 2007.

FAVERO, Eloi Luiz. **Programação em Prolog – uma abordagem prática**.
Disponível em: http://www3.ufpa.br/favero/
Acesso em: Julho de 2011

GODOY, Luisa. **Os verbos recíprocos no PB.** Dissertacao de Mestrado. UFMG. 2008. Disponivel em < http://www.letras.ufmg.br/nucleos/nupes >

HOPCROFT, J. E, ULLMAN, J.D., MOTWANI, R.. **Introdução à Teoria de Autômatos, Linguagens e Computação** - tradução em português da segunda edição americana.Editora Campus. Rio de Janeiro, 2003.

HOUAISS, Antonio. **Dicionário Houaiss da Língua Portuguesa**. Editora Objetiva. Rio de Janeiro, 2009.

ILARI, Rodolfo. **A categoria advérbio na gramática do português falado.**
Disponível em: < http://seer.fclar.unesp.br/alfa/article/view/1430 >
Acesso em: 15 nov. 2010.

KINOSHITA, Jorge; SALVADOR, Laís N.; MENEZES, Carlos E. D. **COGROO – um corretor gramatical acoplável ao OPENOFFICE.** V Workshop em Tecnologia da Informação e da Linguagem Humana: IME - Rio de Janeiro, 2007.

LYONS, John. **As ideias de Chomsky**. Editora Cultrix. São Paulo, 1970.

LYONS, John. **Linguagem e Linguística: uma introdução**. LTC. Rio de Janeiro, 1987.

MINSKY, Marvin. **The Emotion Machine**. Simon e Schuster. New York, 2006.

MIORELLI, Vieira. **ED – CER: Extração do Sintagma Nominal em Sentenças em Português.** 2001. Monografia (Mestrado em Ciência da Computação) – Pontifícia Universidade Católica do Rio Grande do Sul. Porto Alegre, 2001.

MIOTO, Carlos et al. **Novo Manual de Sintaxe**. Editora Insular. Florianópolis, 2007.

MOREIRA, C. B. **Princípio de ligação entre sintaxe e semântica: construções estativas.** Belo Horizonte: Dissertação de Mestrado. UFMG, 2000.

MORELLATO, Luana Vieira. **SIDSN – Sistema Identificador de Sintagmas Nominais.** 2007. Monografia (Graduação em Ciência da Computação) – Universidade Federal do Espírito Santo.Vitória, 2007.

NILC - NÚCLEO INTERINSTITUCIONAL DE LINGUISTICA COMPUTACIONAL . **TEP v2.0**, 2010. Disponível em: <http://www.nilc.icmc.usp.br/tep2/download.htm> Acesso em: Agosto de 2010.

NUGUES, Pierre M. **An Introduction to language processing with Perl and PROLOG**. Springer. USA, 2006.

OLIVEIRA, Hugo Gonçalo. **PAPEL v2.0**, 2010. Disponível em: <http://www.linguateca.pt/PAPEL/PAPELv2.0.zip> Acesso em: Agosto de 2010.

OlIVEIRA, Maria do Carmo Pereira. **As frases copulativas com ser : natureza e estrutura.** Dissertação de Mestrado – Universidade do Porto (Portugal). Porto, Portugal, 2001.
Disponível em <http://hdl.handle.net/10216/13058> Acessado em Agosto/2011.

OTHERO, Gabriel de Ávila. **Teoria X-Barra: descrição do português e aplicação computacional**. Contexto. São Paulo,2006.

OTHERO, Gabriel de Ávila. **A gramática da frase em português**. ediPUCRS. Porto Alegre,2009.

PAGANI, Luiz Arthur. (2004). **Analisador gramatical em Prolog para gramáticas de estrutura sintagmática.** Revista Virtual de Estudos da Linguagem – ReVEL. Ano 2, n. 3. [www.revelhp.cjb.net]

PALAZZO, Luiz A. **Introdução a Programação Prolog.** Editora Universidade Católica de Pelotas(RS). Pelotas, 1997.

PEREIRA, F.; SHIEBER, S. M. **Prolog and natural-language analysis.** CSLI. Stanford,1987.

PERINI, Mário A.. **Gramática descritiva do português**. Editora Ática. São Paulo,2009.

PERINI, Mário A.. **Gramática do português brasileiro**. Parábola Editorial. São Paulo,2010.

PRICE, Ana M. A.; TOSCANI, Simão S. **Implementação de linguagens de programação: compiladores**. **2. ed.** Sagra Luzzatto. Porto Alegre, 2001.

RINO, L.H.M. et al.. **Aspectos da construção de um revisor gramatical automático para o português. Estudos Linguísticos, v. 31, Maio.** São Paulo, 2002.

SANTOS, Pedro Perini Frizzera da Mota. **Epistemologia cognitiva para uso de preposições – o caso da preposição de.** Doutorado em estudos Lingüísticos. Universidade Federal de Minas Gerais, Belo Horizonte, 2007.

SEBESTA, R. W. **Conceitos de linguagens de programação**. 4ªEd. Editora Bookman. Porto Alegre, 2000.

SILVA, W.D.C.M.; SUZUMURA, M.; GUSUKUMA, F.W.; PIRES, D.A.M. **CorretorGramatical Acoplável ao OpenOffice.org CoGrOO 2.0**. Monografia de Conclusão do Curso de Engenharia Elétrica e Engenharia de Computação. Escola Politécnica, Universidade de São Paulo, 2006.

SOUZA E SILVA, Maria Cecília Pérez de; KOCH, Ingedore Villaça. **Linguística aplicada ao português: sintaxe**. Editora Cortez. São Paulo, 2002.

TATE, Bruce A. **Seven Languages in Seven Weeks: A Pragmatic Guide to Learning Programming Languages.** Pragmatic Bookshelf. USA, 2010.

ULIANO, Sueli Caramelo et al. NÚCLEO INTERINSTITUCIONAL DE LINGUISTICA COMPUTACIONAL . **Uma análise do CoGrOO, um Corretor Gramatical acoplável ao OpenOffice**. Disponível em:
< http://www.pcs.usp.br/~cogroo/papers/analise-cogroo-corpus-metro.html >
Acesso em: Novembro de 2010.