MINISTÉRIO DA DEFESA
EXÉRCITO BRASILEIRO
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
CURSO DE MESTRADO EM SISTEMAS E COMPUTAÇÃO

RICARDO QUEIROZ DE ARAUJO FERNANDES

UMA PLATAFORMA PARA ANÁLISE SINTÁTICA E SUA APLICAÇÃO AO PORTUGUÊS

Rio de Janeiro
2009

INSTITUTO MILITAR DE ENGENHARIA

RICARDO QUEIROZ DE ARAUJO FERNANDES

UMA PLATAFORMA PARA ANÁLISE SINTÁTICA E SUA APLICAÇÃO AO PORTUGUÊS

Dissertação de Mestrado apresentada ao Curso de Mestrado em Sistemas e Computação do Instituto Militar de Engenharia, como requisito parcial para obtenção do título de Mestre em Sistemas e Computação.

Orientador: Prof. Alex de Vasconcellos Garcia - D.C.

Rio de Janeiro
2009

INSTITUTO MILITAR DE ENGENHARIA
Praça General Tibúrcio, 80-Praia Vermelha
Rio de Janeiro-RJ CEP 22290-270

F363P Fernandes, Ricardo Queiroz de Araujo
Uma Plataforma para Análise Sintática e sua Aplicação ao Português/ Ricardo Queiroz de Araujo Fernandes.
– Rio de Janeiro: Instituto Militar de Engenharia, 2009.
128 p.: il., tab.

Dissertação (mestrado) – Instituto Militar de Engenharia – Rio de Janeiro, 2009.

1. Linguística Computacional. 2. Linguagens de Programação. I. Título. II. Instituto Militar de Engenharia.

CDD 41.285

INSTITUTO MILITAR DE ENGENHARIA

RICARDO QUEIROZ DE ARAUJO FERNANDES

UMA PLATAFORMA PARA ANÁLISE SINTÁTICA E SUA APLICAÇÃO AO PORTUGUÊS

Dissertação de Mestrado apresentada ao Curso de Mestrado em Sistemas e Computação do Instituto Militar de Engenharia, como requisito parcial para obtenção do título de Mestre em Sistemas e Computação.

Orientador: Prof. Alex de Vasconcellos Garcia - D.C.

Aprovada em 15 de janeiro de 2009 pela seguinte Banca Examinadora:

Prof. Alex de Vasconcellos Garcia - D.C. do IME - Presidente

Prof. Violeta de San Tiago Dantas Barbosa Quental - D.C. da PUC-Rio

---

Prof. Edward Hermann Haeusler - D.C. da PUC-Rio

Prof. José Antônio Moreira Xexéo - D.C. do IME

Rio de Janeiro
2009

Dedico esta dissertação à minha avó materna, vovó Odêmia, que perdi durante este curso de mestrado.

AGRADECIMENTOS

Agradeço ao Exército Brasileiro por acreditar que a formação acadêmica é um investimento. Agradeço ainda ao Exército a oportunidade ímpar que me ofereceu de estudar minha própria língua, estudo esse muito mais do que uma obrigação profissional, no meu caso, um prazer.

Agradeço a meus pais a oportunidade de estar aqui e poder viver estes momentos. Agradeço a eles ainda o grande apreço aos livros, ao estudo e ao debate das ideias.

Agradeço à minha esposa por todo o carinho e toda a dedicação durante o curso de mestrado. Agradeço ainda por me ajudar a corrigir os erros da dissertação, tendo lido esta dissertação toda em um dia.

Agradeço aos meus familiares, pelo prazeroso convívio enquanto estive no Rio de Janeiro, minha cidade natal. Eles me proporcionaram um acolhimento que em outra cidade eu não teria.

Agradeço muito ao meu orientador, o professor Alex de Vasconcellos Garcia, pelo zelo com que conduziu os rumos do meu curso de mestrado. Sem dúvidas, devo agradecer a ele ainda o tema da dissertação, pois aprendi muito com o presente trabalho. Além disso, sem suas palavras de amizade ou sua objetividade de cobrança, meu curso teria perdido muito de sua motivação.

Agradecimento especial devo aos professores doutores que aceitaram com entusiamo participar da Banca Examinadora de minha dissertação.

Agradeço ainda aos colegas de curso que souberam aproveitar essa oportunidade para criar relações de empatia e amizade mútuas.

Agradeço a todas as pessoas que contribuíram com o desenvolvimento desta dissertação de mestrado, tenha sido por meio de críticas, ideias, apoio, incentivo ou qualquer outra forma de auxílio.

Por fim, a todos os professores e funcionários do Departamento de Engenharia de Sistemas (SE/8) do Instituto Militar de Engenharia.

Ricardo Queiroz de Araujo Fernandes

"Porque na muita sabedoria há enfado;
e o que aumenta em ciência, aumenta em trabalho."
Eclesiastes

Ricardo Queiroz de Araujo Fernandes

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

ABREVIATURAS

NILC	-	*Núcleo Interinstitucional de Lingüística Computacional*
CETEMPúblico	-	*Corpus de Extractos de Textos Electrónicos MCT/Público*
GLR	-	*Generalized Left-to-right Rightmost derivation parser*
PLN	-	*Processamento de Linguagem Natural*
LP	-	*Linguagens de Programação*
GN	-	*Gramática Normativa*

RESUMO

Esta dissertação apresenta o desenvolvimento de uma plataforma de análise sintática, bem como seu uso para o desenvolvimento de um analisador sintático para o Português. Esse analisador sintático é ferramenta necessária para o desenvolvimento de diversas aplicações de processamento de texto escrito, em especial aplicações de Recuperação de Informações. Foram aplicadas técnicas de processamento de linguagens de programação para tratar problemas de explosão combinatória do número de árvores de derivação geradas e para tratar a manutenibilidade da Gramática Livre de Contexto. Entre essas técnicas, destacamos a codificação de fenômenos linguísticos, como concordância e regência, em Gramática de Atributos.

ABSTRACT

This dissertation presents the development of an infrastructure for syntactic analysis and its application into the development of an syntactic analyser for Portuguese Language. That syntactic analyser is a necessary tool for the development of many other tools for written text processing, specially tools for Information Retrieval. Techniques from programming languages processing were employed to treat problems of Combinatory Explosion of the number of derivation trees generated and to treat the manutenibility of the Context Free Grammar. Among those techniques, we emphasize the encoding of linguistic phenomena, such as Agreement, through an Attribute Grammar description.

1 INTRODUÇÃO

Dentro da estrutura organizacional do Exército Brasileiro, o órgão responsável por coordenar a estrutura de Ciência e Tecnologia é o Departamento de Ciência e Tecnologia (DCT). Esse órgão produz o documento mais importante para a definição das prioridades dos esforços em Ciência e Tecnologia do Exército: o Plano Básico de Ciência e Tecnologia (PBCT). Nesse documento, as principais áreas do conhecimento de interesse estratégico para o Exército estão organizadas em Grupos Finalísticos. O DCT incluiu o presente trabalho no Grupo Finalístico de Segurança da Informação, determinando, além disso, o tema Recuperação de Informações (RI).

1.1 CONTEXTO E MOTIVAÇÃO

Se levarmos em consideração que significativa parcela de todas as informações disponíveis para a tomada de decisões, para o desenvolvimento de pesquisas e produtos e para a documentação está codificada em linguagem natural e que esse conjunto constitui muitas vezes gigantescos repositórios, entenderemos que o aproveitamento eficiente de toda informação semântica passa cada vez mais a depender de maneira crucial do processamento automático do português. De outra maneira, as informações terão sua acessibilidade restrita a algum processo manual de busca e verificação, que, em muitos casos, se torna inviável.

Segundo (MITKOV, 2005), o termo recuperação de informações (RI) diz respeito à ciência de encontrar objetos que sejam relevantes para uma pesquisa. Esses objetos podem estar em qualquer meio de armazenamento. No nosso caso, estamos interessados em sentenças e textos em português escrito. No entanto, embora os textos estejam codificados em linguagem natural, pouco se fez uso de Linguística Computacional em benefício da Recuperação de Informações.

Por outro lado, como temos interesse em informações semânticas, a perspectiva da Ciência da Computação nos oferece um histórico de desenvolvimento teórico do estudo da semântica que remonta ao trabalho de Frege, passando por Tarki e Carnap até chegar ao trabalho de Richard Montague (BLACKBURN, 2005).

Sabemos a partir de Frege que "A atribuição de significado às construções sintáticas (...) é um tema central da Ciência da Computação." (MENEZES, 2006). Mas foi Montague o primeiro a propor e a defender a tese de que a relação entre sintaxe e semântica em linguagens naturais não era essencialmente diferente da relação que existe entre sintaxe e semântica em linguagens formais (PARTEE, 1990).

Montague propõe métodos formais para o estudo da semântica de linguagens naturais a partir do *princípio da funcionalidade* (MENEZES, 2006; PARTEE, 1990) de Frege, conseguindo representar funções através de *lambda calculus* (PARTEE, 1990; CHIEHSHAN, 2002).

O desenvolvimento de compiladores (AHO, 1986), por sua vez, nos oferece métodos extremamente eficientes de análise sintática. Para aplicá-los a linguagens naturais é necessário o desenvolvimento de gramáticas que correspondam aos métodos escolhidos. Nesse ponto a Linguística nos oferece o suporte de séculos de desenvolvimento teórico em torno dos constituintes de frases. Por essa razão, entendemos que a sintaxe é o ponto de confluência de diferentes áreas do conhecimento humano e uma oportunidade muita rica de interação entre elas.

O desenvolvimento relativamente recente da Linguística Computacional tira proveito dessa interação entre Linguística e Ciência da Computação, produzindo diversas ferramentas de *Processamento de Linguagens Naturais* (PLN) para diversas línguas ao redor do mundo. Apesar da maturidade deste campo de estudo, ainda há, no caso do Português, carência de ferramentas específicas (SILVA, 2008).

A única ferramenta de marcação morfossintática disponível para a pesquisa em Português é o sistema PALAVRAS (BICK, 2000). Esse sistema é muito robusto, com desempenho excelente para a marcação (*tagging*) morfossintática. No entanto, as árvores sintáticas que produz tendem a ser relativamente enxutas, omitindo alguns relacionamentos entre os constituintes da frase que podem ter significado semântico relevante.

Outro aspecto importante é o fato de não termos o domínio do código de um analisador sintático para o Português, o que nos coloca em situação desfavorável para o desenvolvimento de novas aplicações de PLN que tenham a análise sintática como insumo fundamental, como por exemplo a Recuperação de Informações.

1.2 OBJETIVO

Este trabalho é resultado da pesquisa em Linguística Computacional desenvolvida no Instituto Militar de Engenharia, no Departamento de Engenharia de Sistemas. Ele se propõe a desenvolver uma plataforma para análise sintática e aplicá-la ao português.

Optamos por trabalhar com Recuperação de Informações sobre o português escrito pela razões mencionadas na seção anterior. Em nosso caso, uma plataforma de RI sobre a linguagem natural depende fundamentalmente da capacidade de associação de significado semântico ao texto escrito.

Dessa maneira, o desenvolvimento de um analisador sintático sobre o português é ferramenta indispensável para o estudo de RI sobre o Português. Como, no começo de nosso trabalho, não tínhamos qualquer ferramenta de análise sintática que fundamentasse nosso interesse sobre RI, nosso objetivo passou a ser o desenvolvimento de um analisador sintático próprio para o Português.

Uma vez definido nosso objetivo de curto prazo, decidimos estudar os formalismos de análise sintática que o estudo de linguagens de programação e compiladores nos oferecem. A razão disso reside em nossa preocupação em realizar a análise de maneira eficiente em computadores. Sendo assim, nossa abordagem tem por fundamento as Gramáticas Livres de Contexto (GLC).

Utilizamos, assim, técnicas oferecidas pelo estudo de compiladores para a análise sintática. Empregamos, em especial, a infraestrutura de análise do método $LR(1)$ (AHO, 1986) para a análise sintática e adotamos regras semânticas descritas em Gramáticas de Atributos (AHO, 1973a; RANGEL, 2000) para a especificação de restrições.

Como podemos observar, nosso trabalho exigiu um diálogo constante entre os formalismos e algoritmos necessários à computação e os fenômenos próprios de nossa própria linguagem natural: o Português. Aplicar e adaptar métodos de processamento de linguagens de programação a esse objeto tão rico e complexo que é a nossa lingua, revelou-se um processo desafiador, enriquecendo a nossa experiência em linguagens de programação.

Como o trabalho de análise sintática embasa diversas aplicações de PLN, temos ainda a convicção de que nosso trabalho poderá ser usado para construir diferentes sistemas de PLN.

1.3 APRESENTAÇÃO

O restante desta dissertação encontra-se dividida nos seguintes capítulos:

O capítulo 2 - Trabalhos Relacionados - apresenta os trabalhos relacionados ao tema de nosso trabalho, incluindo suas características e seus formalismos.

O capítulo 3 - Conceitos de Análise Sintática - apresenta conceitos de análise sintática, cuja compreensão é fundamental para o entendimento do presente trabalho.

O capítulo 4 - Análise Sintática *GLR* do Português - explica como foi feita a aplicação dos conceitos apresentados no capítulo 3 para o tratamento da língua portuguesa, incluindo as dificuldades encontradas.

O capítulo 5 - Análise sintática com restrições - apresenta as soluções propostas e implementadas para tratar as dificuldades apresentadas no capítulo 4.

O capítulo 6 - A Ferramenta de Análise Sintática - apresenta as tarefas de criação e de funcionamento do analisador sintático.

O capítulo 7 - Resultados - apresenta o resultado dos testes de calibração e de avaliação de nossa plataforma, bem como a análise desses resultados.

O capítulo 8 - Considerações finais - ficou reservado para as conclusões acerca do presente trabalho e para a explanação dos trabalhos futuros que podem servir para a continuidade do desenvolvimento de nossa plataforma.

2 TRABALHOS RELACIONADOS

De acordo com Bick(BICK, 2007), Othero(OTHERO, 2008) e o catálogo de recursos do sítio Linguateca[1], apresentamos aqui os analisadores sintáticos disponíveis para o português:

- PALAVRAS(BICK, 2000): Analisador robusto desenvolvido para a abordagem de *dependency grammar* (DG) e *phrase structure grammar* (PSG) sobre o formalismo das *constraint grammars* (CG);

- Curupira(MARTINS, 2003): Analisador robusto desenvolvido a partir de regras PSG do sistema Regra(MARTINS, 1998) com prioridades e restrições;

- GojolParser[2]: Analisador morfo-sintático comercial que, segundo (BICK, 2007, 2006; OTHERO, 2008), é baseado nos formalismos DG e PSG. Bick(BICK, 2007, 2006) ainda ressalta que esse sistema se autoproclama o melhor e que possuiria taxas de erro abaixo de 1%.

- LX-Suite[3]: Sistema desenvolvido na Universidade de Lisboa. A análise sintática é feita pela gramática computacional LXGram(COSTA, 2007). A LXGram é desenvolvida sobre o formalismo *Head-Driven Phrase Structure Grammar* (HPSG) e é capaz de realizar análise semântica formal sobre lógica de primeira ordem através de cálculo lambda.

- Grammar Play(OTHERO, 2004): Analisador sintático desenvolvido sobre a linguagem Prolog. O Grammar Play analisa sentenças declarativas simples (com apenas um verbo) do português brasileiro e retorna sua estrutura de constituintes. Foi desenvolvido sobre os formalismos PSG e o modelo de descrição sintática *X-barra* padrão.

Nas próximas seções veremos em detalhe dois analisadores sintáticos importantes: na seção 2.1 o sistema PALAVRAS e na seção 2.2 o sistema Curupira.

[1] http://www.linguateca.pt

[2] http://www.linguateca.pt/Repositorio/GojolParser.txt

[3] http://lxsuite.di.fc.ul.pt/

2.1 PALAVRAS

O analisador sintático PALAVRAS é, de longe, o mais robusto e o mais conhecido pelo meio acadêmico. O trabalho de Edward Bick remonta à sua dissertação de mestrado, tendo seguido por sua tese de doutorado, permanecendo em contínuo desenvolvimento e refinamento. Esses são os motivos pelos quais tomaremos o sistema PALAVRAS como nosso referencial de comparação.

O sistema de análise PALAVRAS é descrito pela tese de doutorado de Edward Bick(BICK, 2000). Bick teria ficado impressionado pela robustez e pelo uso de marcas textuais (*tags*) que integram morfologia e sintaxe na *English Constraint Grammar*(KARLSSON, 1994). Decide, então, aplicar o mesmo formalismo para uma língua românica: o Português.

O sistema é organizado segundo o conceito de *Progressive Level Parsing*, que Bick não define formalmente, mas constantemente usa ao longo de sua tese. Este conceito é entendido como o processo que é feito a partir da análise léxico-morfológica e segue com a resolução de ambiguidades e considerações morfológicas, sintáticas, semânticas e de aplicações. No entanto, este conceito é muito similar ao processo apresentado por Karlsson(KARLSSON, 1990) na descrição das *constraint grammars* (CG), que veremos no próximo capítulo.

Segundo Bick(BICK, 2000), qualquer sistema CG é fortemente dependente da qualidade e da abrangência de sua base léxico-morfológica. Antes mesmo de sua tese ficar pronta, Bick(BICK, 1996) já afirmava que seu sistema de análise trabalhava com um léxico de grande abrangência produzido manualmente, alcançando taxa de acerto de 99% para análise léxico-morfológica.

No sistema PALAVRAS, o analisador léxico-morfológico se chama PALMORF. Ele é responsável por atribuir ao texto bruto as possíveis leituras para cada palavra. Para isso, ele é composto por dois módulos: o pré-processador e o analisador morfológico.

O pré-processador é responsável por identificar e resolver os seguintes fenômenos da linguagem: palavras compostas, letras maiúsculas, ênclise, abreviações. O analisador morfológico é responsável por produzir os conjuntos de leituras, realizando: identificação de lexemas, flexões, derivações, incorporação de verbos, hifenização, aspas, heurísticas de nomes próprios, heurísticas de acentuação, bimorfismo luso-brasileiro[4], além de outras funções.

O resultado do módulo PALMORF é, então, submetido a aplicações iterativas de regras CG com base em informações de classes de palavras, forma da palavra, forma do lema da palavra, marcas de valência e de classes semânticas. Depois disso, o resultado é submetido ao pós-processador. Todo este processo, incluindo o PALMORF, caracteriza o módulo PALTAG, responsável por atribuir ao texto bruto todas as informações léxico-morfológicas.

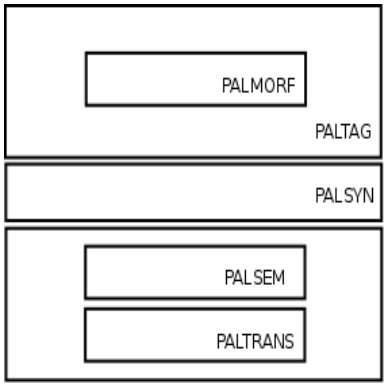

FIG. 2.1: Organização do sistema PALAVRAS

No próximo módulo, o PALSYN, é realizado o mapeamento sintático, que atribui todas as possíveis funções sintáticas às classes de palavras ou aos lemas no contexto de uma determinada regra CG. A resolução de ambiguidades é feita neste mesmo módulo, com a aplicação iterativa de regras CG para tratar: estruturas de argumentos e adjunções, associações núcleo-complemento e diversas funções de oração subordinadas.

Além desses módulos, o sistema PALAVRAS inclui os módulos PALSEM, para re-

[4]O bimorfismo é a relação entre palavras que possuem ortografia diferente nos dois países. Por exemplo, a palavra “ato” está relacionada com a palavra “acto”.

solução de ambiguidades de valência e de classes semânticas, e o PALTRANS, responsável por processos de tradução para outras línguas.

2.1.1 *CONSTRAINT GRAMMARS*

O formalismo das *Constraint Grammars* (CG) primeiramente apresentado por Karlsson (KARLSSON, 1990) e largamente utilizado por Bick em seu conhecido analisador sintático PALAVRAS (BICK, 2000) surgiu com o objetivo claro de atacar um dos problemas centrais do processamento de linguagens naturais: a ambiguidade inerente aos processos de análise sintática.

Karlsson (KARLSSON, 1990) e Bick (BICK, 1996) argumentam que as abordagens gramaticais estão sujeitas aos limites da expressividade e da ambiguidade, em que uma tentativa de ganho por um lado resulta em perda do outro. Este fato nós sentimos com a abordagem livre de contexto sobre o português. Cada aumento de expressividade teve por resultado explosão combinatória.

A essência das *constraint grammars* é a definição de um método de análise sensível ao contexto iterativa, ou seja, não recursiva, que trabalha com poucos níveis de hierarquia para a resolução de ambiguidades.

Karlsson divide em subproblemas o trabalho de análise sintática de textos irrestritos:

- pré-processamento;
- análise morfológica;
- resolução local de ambiguidades morfológicas;
- mapeamento morfossintático;
- resolução dependente do contexto de ambiguidades morfológicas;
- determinação dos limites de orações dentro dos períodos;
- resolução de ambiguidades de funções sintáticas de superfície.

revista						
	”revista”	N	F	S		
	”revestir”	V	PR	1/3S	SUBJ	VFIN
	”revistar”	V	IMP	2S	VFIN	
	”revistar”	V	PR	3S	IND	VFIN
	”rever”	V	PCP	F	S	

TAB. 2.1: Exemplo de *cohort*

Uma *constraint grammar* para uma linguagem é um conjunto de regras de restrições que serão aplicadas a textos que já foram morfologicamente analisados. A análise morfológica deverá preservar todas as leituras possíveis de cada palavra. Ao conjunto de todas as leituras de uma palavra dá-se o nome de *cohort*, como no exemplo:

Cada leitura é composta pela forma base da palavra, sua classe gramatical e demais características morfológicas. As regras de uma CG especificam quais leituras são impossíveis e quais leituras são obrigatórias no contexto do período.

A especificação segue uma notação formal que define o domínio da regra (em geral, a palavra), o operador (aceitar, rejeitar) o alvo (de qual leitura a regra trata) e as condições de contexto (definidas a partir das posições relativas das palavras). Apresentamos abaixo um exemplo de regra:

@w =0 (VFIN) (-1 PRP)

A regra acima significa descartar (=0) qualquer leitura (@w) que seja verbo em modo finito (VFIN) se a palavra anterior (-1) for uma preposição (PRP). A mesma ideia se aplica à análise sintática, excluindo tudo que não pode pertencer a alguma estrutura sintática.

Ao aplicarmos o conjunto de regras de maneira iterada, conseguiremos fazer com que o número de alternativas para cada palavra diminua. Contudo, a última leitura remanescente nunca pode ser descartada. Esta leitura é, sem dúvida, aquela que atende ao maior número de restrições. Desta maneira, mesmo frases não gramaticais produzem algum tipo de análise.

Esta condição de parada é também uma garantia de robustez ao paradigma da CG,

pois a adição de novas regras e novas condições só tendem a tornar os sistemas cada vez mais refinados.

2.2 CURUPIRA

O analisador sintático Curupira(MARTINS, 2003) surgiu como uma das partes do revisor gramatical Regra(MARTINS, 1998) desenvolvido pelo Núcleo Interinstitucional de Lingüística Computacional (NILC). Criado em 1993 por cientistas da Universidade de São Paulo (USP), o NILC atualmente abriga também o trabalho cooperativo de cientistas e desenvolvedores da Universidade Federal de São Carlos (UFSCar) e da Universidade Estadual Paulista (UNESP) de Araraquara.

O analisador Curupira recebe sentenças ou textos, realizando análise *top-down* da esquerda para a direita através de uma gramática livre de contexto e de um extenso léxico do Português do Brasil desenvolvido pelo próprio NILC.

O Curupira toma por hipótese que uma dada sentença está correta e procura oferecer, para essa sentença, todas as alternativas de árvores sintáticas com base nas classes de palavras da sentença. As árvores são construídas com não-terminais que representam os conceitos propostos pela Nomenclatura Gramatical Brasileira (NGB - Portaria MEC 36, de 28/01/59). Ao fazê-lo, o analisador procura informar as categorias dos sintagmas (nominal, verbal, etc.) bem como sua função sintática na sentença (sujeito, objeto, etc.).

Segundo (MARTINS, 2003), o Curupira sofre com a explosão combinatória no caso de sentenças muito longas. Além disso, ele não oferece qualquer gerenciamento de memória consumida pelo sistema.

O Curupira utiliza o mesmo léxico que o sistema Regra. Este léxico conta com 1,5 milhão de palavras, incluindo derivações e flexões e excluindo palavras compostas (locuções, por exemplo). Cada entrada no léxico armazena a classe das palavras (pronome, por exemplo), as subclasses (pronome pessoal, por exemplo), o gênero, o número, o tempo, o modo, a pessoa, o grau, a transitividade, a regência, o tipo (para advérbios) e a forma canônica. O tratamento das palavras compostas é feito no próprio algoritmo de análise sintática, caracterizando antes parte da gramática. O léxico é representado por

um autômato finito, que ocupa 1,3Mb de memória e que permite compartilhamento de prefixos.

A gramática do Curupira conta com aproximadamente 600 regras de re-escrita categorial com prioridades. O Curupira foi desenvolvido tendo em mente a variedade culta do português do Brasil, especialmente estruturas diretas e simples. Dessa maneira, o Curupira não prevê tratamento adequado a inversões e rupturas sintáticas que a língua aceita. Contudo, algumas manifestações agramaticais muito comuns foram incluídas.

O Curupira processa uma sentença de maneira *top-down*, recursiva da esquerda para a direita, tendo por base as informações disponibilizadas pelo léxico e as prioridades das regras da gramática. O Curupira segue, então, a seguinte sequência de análise:

a) Quebra da sentença em símbolos (tokens) de acordo com as possibilidades listadas pelo sistema (token pode ser palavra, pontuação ou lixo, por exemplo);

b) Recuperação das informações do léxico e identificação de mesóclises, ênclises e palavras compostas;

c) Resolução de ambiguidades léxicas: aplicação de regras locais com base em classes de palavras e relações de vizinhança (semelhante a *constraint grammars*) ;

d) Análise sintática;

e) As árvores sobreviventes são representadas em um arquivo texto;

f) O arquivo de saída é utilizado pela interface para que o resultado final possa ser visualizado pelo usuário.

3 CONCEITOS DE ANÁLISE SINTÁTICA

Neste capítulo apresentamos os conceitos que são necessários para a compreensão do trabalho de análise sintática desenvolvido. Na seção 3.1, apresentamos o conceito da análise *GLR*. Na seção 3.2, falaremos rapidamente sobre o gerador de analisadores sintáticos chamado BISON (DONNELLY, 2006). Na seção 3.3 apresentamos a Gramática de Atributos.

3.1 ANÁLISE *GLR*

O problema da análise sintática é descobrir se uma sequência de símbolos pertence à linguagem descrita por uma gramática e, caso pertença, determinar como a gramática gera a sequência. Quando essa relação de pertinência se verifica, dizemos que a gramática reconhece a sequência de símbolos.

Podemos entender que a técnica de análise sintática depende da gramática. Contudo, para facilitar o estudo das gramáticas, temos à disposição a Hierarquia de Chomsky (HOPCROFT, 1969). Essa hierarquia classifica as classes de gramáticas a partir de propriedades que cada classe deve conter. Essa classificação se reflete nas linguagens, classificando-as de acordo com a classe de gramáticas que as descreve. O interessante dessa classificação é o estabelecimento de inclusões que nos permitem compreender o trabalho de análise sintática de maneira gradual. Além disso, a partir dessa classificação, diversos métodos genéricos de análise sintática foram desenvolvidos, tornando a técnica de análise não mais dependente da gramática, mas sim da classe em que a gramática se insere.

Nosso trabalho lida com uma classe ampla de gramáticas: a classe das gramáticas livres de contexto (GLC). Exigimos dessa classe de gramáticas somente que o lado esquerdo de qualquer regra seja composto por somente um não-terminal. A classe das linguagens reconhecidas por alguma GLC se chama classe das linguagens livre de contexto (LLC).

O estudo de compiladores e de linguagens de programação depende de maneira fundamental da capacidade de se realizar a análise sintática eficientemente. Por essa razão, a classe das linguagens que admitem algum método determinístico de análise sintática é amplamente aplicada e estudada em computação. Essa classe de linguagens é chamada de classe das *linguagens determinísticas* (BAUER, 1976), que é um subconjunto próprio das LLC.

A classe das linguagens determinísticas possui diversos métodos de análise sintática. Muitos desses métodos tratam somente parte das linguagens determinísticas, como por exemplo os métodos da família LL (***L**eft to right **L**eftmost-derivation*). O método $LL(k)$ possui pilha e analisa a sequência da esquerda para a direita, produzindo uma árvore de derivação mais à esquerda. Isso quer dizer que as regras sempre se aplicam ao primeiro não-terminal da lista de símbolos ao longo da sequência de derivações. O número k indica quantos símbolos à frente o método investiga para tomar decisões. Esse método inicia a análise sintática a partir do símbolo inicial, produzindo derivações que produzem a sequência de entrada, se ela pertencer à linguagem descrita pela gramática. Métodos que seguem essa direção de análise são chamados de *top-down* (AHO, 1973b).

No entanto, a classe das linguagens determinísticas coincide com a classe das linguagens que admitem uma gramática para algum método da família LR (***L**eft to right **R**ightmost-derivation*) (BAUER, 1976). Um método LR analisa a sequência da esquerda para a direita e produz uma árvore de derivação mais à direita. Isso quer dizer que as regras sempre se aplicam ao último não-terminal da lista de símbolos ao longo da sequência de derivações. Esses métodos iniciam a análise sintática a partir da sequência de entrada, produzindo derivações que produzem o símbolo inicial, se a sequência pertencer à linguagem descrita pela gramática. Métodos que seguem essa direção de análise são chamados de *bottom-up* (AHO, 1973b).

Para entendermos o método de análise *bottom-up*, observaremos a análise da sentença "**1 + 2 * 3**" para a gramática abaixo (RANGEL, 2000).

Gramática $\mathcal{G}_0$

E	→	E	'+'	T
	|	T		
T	→	T	'*'	F
	|	F		
F	→	'(' E ')'		
	|	'1'		
	|	'2'		
	|	'3'		

TAB. 3.1: Exemplo de gramática Livre de Contexto

Começamos com a pilha vazia e terminamos com o símbolo inicial "E".

	Pilha	**Decisão**
1)	$\emptyset$	*nenhuma regra, empilhar próximo terminal*
2)	**1**	***1*** *aparece à direita de uma regra de F*
3)	F	*F aparece à direita de uma regra de T*
4)	T	*T aparece à direita de uma regra de E*
5)	E	*nenhuma regra, empilhar próximo terminal*
6)	E +	*nenhuma regra, empilhar próximo terminal*
7)	E + **2**	***2*** *aparece à direita de uma regra de F*
8)	E + F	*F aparece à direita de uma regra de T*
9)	E + T	*empilhar próximo terminal*
10)	E + T *	*empilhar próximo terminal*
11)	E + T * **3**	***3*** *aparece à direita de uma regra de F*
12)	E + T * F	*"T * F" aparece à direita de uma regra de T*
13)	E + T	*"E + T" aparece à direita de uma regra de E*
14)	E	*símbolo inicial e final da sequência : aceitar sentença*

TAB. 3.2: Sequência de análise *bottom-up*

Este método é inerentemente não-determinístico. Em diversos passos no exemplo acima, não sabemos qual regra da gramática usar, ou ainda se devemos usar uma regra ou empilhar um novo símbolo. Portanto, o método não é eficiente, pois todas as alternativas devem ser exploradas. O método $LR(k)$ é um refinamento deste método, que permite determinar a ação a tomar, olhando os próximos k símbolos a serem lidos.

Em comparação com as demais técnicas de análise sintática, os métodos LR podem ser tão eficientes quanto os demais, consumindo espaço e tempo linear em função do tamanho da entrada. Por essas razões, é fácil compreender o largo emprego dessa técnica para a análise sintática.

A árvore sintática correspondente à análise feita na tabela 3.2 é:

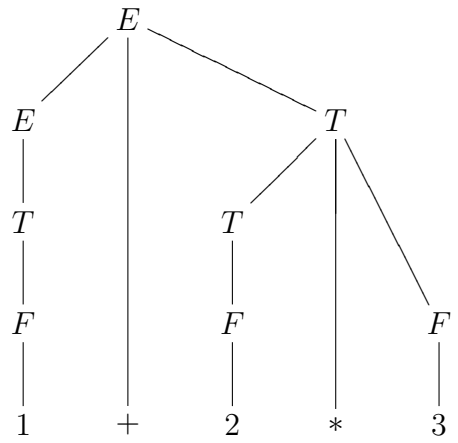

FIG. 3.1: Árvore sintática

Como estamos olhando a análise sintática a partir da perspectiva do estudo de compiladores e linguagens de programação, todas essas técnicas determinísticas exigem que as gramáticas sejam livres de ambiguidades. Essa restrição faz sentido se pensarmos em linguagens de programação, em que a ambiguidade não pode estar presente por limitações do que podemos efetivamente processar com nossos computadores. O ser humano, por sua vez, é capaz de lidar com ambiguidades de interpretação durante a comunicação.

Assim, ao criarmos uma gramática livre de contexto para o português, estaremos diante da ambiguidade sintática. Esse fenômeno acontece, pois a língua nos oferece diferentes construções (sintagmas), que podem ser formados por sequências iguais de símbolos terminais (classes gramaticais).

- *Nós gostamos [de estudo].* (Objeto indireto : Preposição + Substantivo)
- *Nós temos direito [a escolhas].* (Complemento nominal : Preposição + Substantivo)
- *A fazenda [de arroz] prospera.* (Adjunto adnominal : Preposição + Substantivo)

Desejamos verificar todas as possibilidades de construção de sintagmas. Assim, em um determinado estado da análise, estamos interessados em testar todos os empilhamentos e todas as reduções possíveis. Isso quer dizer que não poderíamos mais trabalhar com a análise LR, pois ela exige que as tabelas de ação e de transição tenham, no máximo, uma alternativa de ação ou de transição possível. Passamos, então, a necessitar de um método de análise sintática que pudesse lidar com quaisquer linguagens livres de contexto.

A alternativa foi passar para algoritmos que pudessem tratar gramáticas livre de contexto quaisquer. Esses algoritmos são conhecidos como algoritmos para análise livre de contexto genérica (*general context-free parser algorithms*) (TOMITA, 1986). Tais algoritmos têm de lidar com todos os fenômenos próprios das linguagens livres de contexto, incluindo ambiguidades. Em termos de expressividade, as gramáticas livres de contexto conseguem descrever muitos fenômenos das linguagens naturais. E, além disso, muitas das alternativas teóricas para PLN são construções livres de contexto (TOMITA, 1986).

No entanto, deixar de usar a análise LR significa deixar de usar um algoritmo extremamente eficiente de análise. O desafio passa a ser o desenvolvimento de um algoritmo que possa tirar proveito dos fenômenos da língua. O termo *Generalized LR Parsing* (GLR) está associado ao conjunto dos algoritmos para lidar com as linguagens livres de contexto e nasceu motivado por problemas de linguagens naturais (TOMITA, 1991).

O nosso algoritmo buscou ser simples e próximo ao algoritmo de análise $LR(1)$. Ele possui, no entanto, complexidade exponencial. Apesar disso, existem otimizações em (TOMITA, 1987, 1986) que, uma vez implementadas, tornam nosso algoritmo polinomial. A descrição detalhada do funcionamento de nosso algoritmo de análise sintática será feita na seção 4.4.

3.2 BISON

O desenvolvimento de nosso algoritmo GLR dependia da transformação da gramática livre de contexto em um autômato com pilha. Esse autômato é descrito em forma de tabelas de ação e transição na análise LR. Pela familiaridade com a análise LR, preferimos adotar uma sistemática parecida.

Para realizar essa transformação da GLC em tabelas, poderíamos partir para o desenvolvimento próprio de um gerador de analisadores sintáticos. No entanto, como o foco do presente trabalho é a análise sintática propriamente dita, optamos por buscar geradores de analisadores sintáticos disponíveis para que pudéssemos aproveitar a geração das tabelas. O conjunto de geradores, no entanto, foi reduzido àqueles que gerassem analisadores sintáticos *GLR*. Dentre aqueles que oferecem a funcionalidade *GLR*, a documentação e a facilidade de recuperação das tabelas de decisão foram decisivos em favor do BISON (DONNELLY, 2006).

O Bison é o gerador de analisadores sintáticos oferecido pela *GNU*[5]. Ele é um software livre muito conhecido na área de compiladores e linguagens de programação. Ele transforma uma gramática livre de contexto em um analisador sintático $LaLR(1)$ ou *GLR* para essa gramática. A gramática que o Bison recebe como entrada deve seguir uma formatação própria (DONNELLY, 2006), como no exemplo da tabela 3.3

```
%token   NUM
%left     '+'    '-'
%left     '*'
%%
exp        :     exp     '+'   exp
           |     exp     '-'   exp
           |     exp     '*'   exp
           |     exp     '/'   exp
           |     NUM
           ;
%%
```

TAB. 3.3: Exemplo de gramática do BISON

No entanto, como nosso interesse era desenvolver um analisador sintático próprio, optamos por não aproveitar qualquer código gerado pelo Bison. A partir da descrição textual do analisador *GLR* feita pelo BISON, construímos todas as tabelas necessárias para nossa análise *GLR*: uma tabela para reduções, uma tabela para empilhamentos e uma tabela para transições. Para que o Bison produza a descrição de uma analisador *GLR*, devemos utilizar a diretiva *%glr-parser*.

[5] www.gnu.org/software/bison/

Um detalhe importante: o Bison acrescenta sempre uma regra à gramática para que o símbolo *$accept* seja sempre o inicial. Isso nos obrigou a acrescentar ao final de cada representação de sentença duas ocorrências da representação do *token $end*.

3.3 GRAMÁTICA DE ATRIBUTOS

Na seção 3.1, falamos sobre o trabalho de análise sintática e como podemos obter a árvore sintática de uma lista de símbolos de entrada. Sabemos, por exemplo, construir sintaticamente a sentença "1 + 2 * 3". Contudo, não sabemos o que isso significa, o que isso quer comunicar na linguagem descrita pela gramática $\mathcal{G}_0$. O uso dos símbolos + e * sugerem que a linguagem foi desenvolvida para realizar manipulação numérica de valores. Mas sintaticamente não sabemos o que significa somar ou multiplicar, olhando somente regras sintáticas.

Para sabermos somar, devemos atribuir significado semântico à regra que descreve sintaticamente o operador "+". Essa atribuição de significado não faz parte da análise sintática, mas dela faz uso para uma descrição formal da semântica da linguagem. Nesse exemplo, precisamos atribuir um valor para cada aplicação da regra $E \rightarrow E + T$. Mas, se quisermos fazer com que a soma dependa dos valores dos não-terminais que a constroem, devemos atribuir valores também à demais regras da gramática $\mathcal{G}_0$ e, além disso, atribuir valores aos terminais que aparecem na sequência de entrada. Seguindo a notação de (RANGEL, 2000), reapresentamos a gramática $\mathcal{G}_0$ com atribuições de valores.

Gramática $\mathcal{G}_0$

E^0	$\rightarrow$	E^1	$+$	T^0	E^0.valor := E^1.valor + T^0.valor
	|	T			E^0.valor := T^0.valor
T^0	$\rightarrow$	T^1	$*$	F^0	T^0.valor := T^1.valor * F^0.valor
	|	F^0			T^0.valor := F^0.valor
F^0	$\rightarrow$	'(' E^0 ')'			F^0.valor := E^0.valor
	|	'1'			F^0.valor := 1
	|	'2'			F^0.valor := 2
	|	'3'			F^0.valor := 3

TAB. 3.4: Exemplo de Gramática de Atributos

Chamamos de *Gramática de Atributos* o par formado por uma gramática livre de contexto e suas regras de cálculo de atributos. Observe-se que, na definição das regras de cálculo, utilizamos as definições de + como operador soma e * como operador multiplicação e o próprio valor dos números. Além disso, as regras de cálculo só podem dizer respeito a atributos de não-terminais que aparecem na própria regra.

Para calcularmos o valor de "1 + 2 * 3", aproveitamos a árvore sintática, que reapresentamos aqui com seu atributo valor em subscrito.

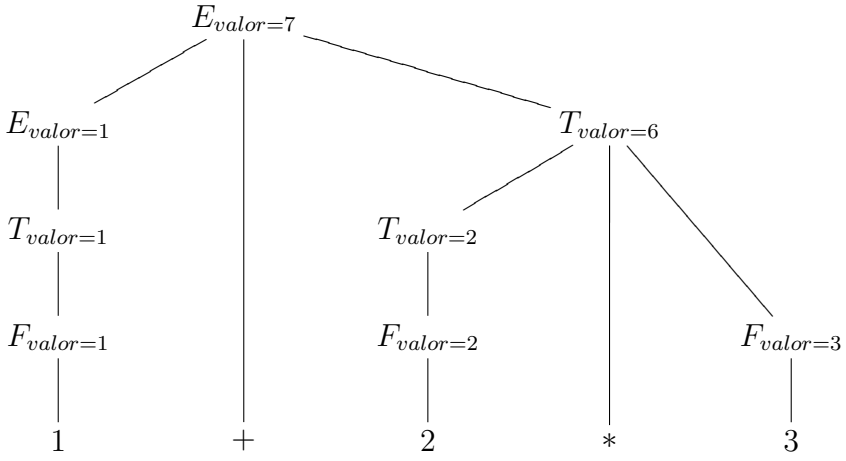

FIG. 3.2: Árvore sintática com o atributo valor

Devemos ressaltar que as regras de cálculo definem relações de dependência entre os atributos. Ou seja, uma regra de cálculo pode ser aplicada em qualquer momento desde que todos os atributos do lado direito já tenham sido calculados. Nesse sentido E^0.valor depende de E^1.valor e T^0.valor. Essa relação de dependência pode ser expressa em forma de grafo (Grafo de Dependências (RANGEL, 2000)). O principal problema a ser identificado sobre este grafo é o problema da ciclicidade. Se um atributo está em um ciclo ele depende de sua própria definição para poder ser avaliado.

Para tratar essas dependências e definir uma metodologia de cálculo de atributos, (AHO, 1973a) primeiro classifica os atributos em *sintetizados* e *herdados*. Os atributos sintetizados são atributos de um não-terminal que são definidos em uma regra sintática que tem o mesmo não-terminal como lado esquerdo da regra. Se um atributo é definido em uma regra sintática que não tem o seu não-terminal como lado esquerdo da regra, então esse atributo é herdado.

Para tornar o problema mais simples de ser tratado, (RANGEL, 2000) faz duas exigências sobre as regras de cálculo:

a) Os atributos só podem ter um tipo. Ou seja, cada atributo será sintetizado ou herdado, mas não os dois;

b) Para cada regra da gramática livre de contexto, devem ser apresentadas as regras semânticas para calcular todos os atributos sintetizados no não-terminal do lado esquerdo da regra sintática e regras para todos os atributos herdados de todos os não-terminais do lado direito da regra sintática.

Uma gramática de atributos que atende às duas restrições é chamada de *Gramática Normal*. Nossa gramática de atributos é normal e todos os atributos são sintetizados. Isso significa que em cada regra toda a informação para o cálculo de atributos já está disponível, sem a necessidade de percurso na árvore sintática completa. Em outras gramáticas de atributos isso não é possível, sendo necessário o estabelecimento de percursos. Sobre os métodos de avaliação de atributos ver (RANGEL, 2000).

Até o momento apresentamos os atributos como a representação semântica da estrutura sintática. Todavia, em nosso trabalho, encontramos um outro uso para os atributos: a possibilidade de expressar que determinada estrutura sintática deve ser rejeitada. Ao fazê-lo, estamos aptos a retirar do resultado da análise sintática as árvores sintáticas marcadas para serem rejeitadas. Mais do que isso, **durante** a análise sintática podemos avaliar localmente os atributos segundo as regras de cálculo e, na ocorrência de inconsistência (valor *falso*), podemos interromper a análise sintática. E interromper uma alternativa de análise não interfere nas demais alternativas de análise sintática.

Nossas regras de cálculo de atributos se dividem em dois grupos: restrições de compatibilidade e regras de construção de atributos. São exemplos do primeiro grupo as regras que dizem respeito à concordância (nominal e verbal) e à regência (nominal e verbal). Nós decidimos expressar essas restrições através de atributos, pois a representação sintática de todas as informações morfológicas tornaria a gramática muito mais complexa.

Para a análise sintática deixamos somente a principal informação morfossintática, a classe gramatical da palavra. Essa informação somente contribui para a estrutura de uma árvore sintática, como podemos ver na figura 3.3

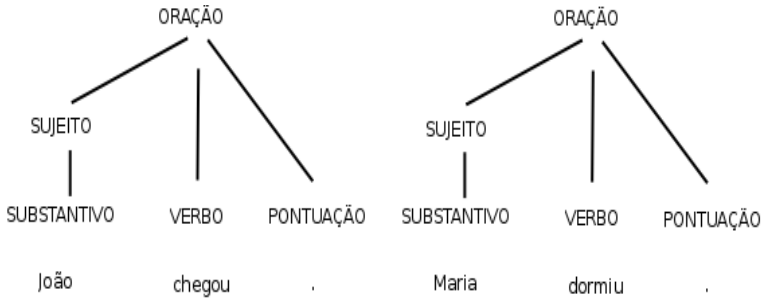

FIG. 3.3: Exemplos de árvores sintáticas

Dessa maneira, cada classe gramatical é representada por um terminal da gramática. As demais informações morfológicas contribuirão somente para a aceitação ou rejeição de uma dada estrutura sintática.

- *O carro chegou.*
- **O carro chegaram.*

A primeira frase deverá ser aceita, enquanto a segunda não. A única informação que utilizamos para rejeitar a segunda frase é o número do sujeito ("O carro") e o número do verbo "chegaram". Como essa análise de aceitação não é estrutural, ela não será feita pelo mecanismo de análise sintática, mas nele se insere através da análise de atributos. Por essa razão, de agora em diante, chamaremos de atributos as informações morfológicas. Durante nosso trabalho, criamos um conjunto de atributos que podem ser associados aos terminais (classes de palavras) e aos não-terminais (elementos da sintaxe). Dentre eles podemos citar: Gênero, Número, Pessoa, Tempo e Modo.

4 ANÁLISE SINTÁTICA *GLR* DO PORTUGUÊS

4.1 INTRODUÇÃO

Neste capítulo apresentaremos como foram aplicados os conceitos do capítulo anterior ao processamento do português.

Inicialmente apresentamos, na seção 4.2, o trabalho envolvendo a criação e a manipulação da gramática. Nessa seção, colocamos os desafios ao se escrever a gramática livre de contexto para o português. Os exemplos de regras da gramática estão escritos em itálico e seguem as normas definidas em (DONNELLY, 2006).

Na seção 4.3, colocamos as tarefas de recuperação, armazenamento e representação de um conjunto extenso de informações léxicas. O tratamento dessas informações é o insumo necessário à tarefa principal deste trabalho: a análise sintática.

Nossa percepção é a de que a análise sintática concentra os maiores desafios e oportunidades de contribuição para o PLN. Procuramos esclarecer, na seção 4.4, como funciona a análise sintática em nossa plataforma.

Em cada seção são relatados os problemas encontrados, muitos dos quais já apresentados na literatura sobre PLN. Eles nos forneceram a convicção de que a linguagem natural é um objeto de estudo que possui comportamentos de organismo vivo, que segue algumas regras mas desrespeita muitas delas quando quer. Mais do que isso, os problemas colocaram à prova os conceitos escolhidos para abordar o tema.

4.2 GRAMÁTICA

A gramática é o principal formalismo de descrição de nossa abordagem sintática. Ela é responsável pela descrição das estruturas sintáticas que serão construídas. Dessa maneira, as regras da gramática irão diretamente implicar a aceitação ou a rejeição de uma determinada sentença.

Assim, verificar se uma sentença é ou não gramatical é tarefa realizada de forma positiva, ou seja, só serão aceitas as estruturas que forem explicitamente descritas na gramática.

A abordagem negativa equivale a aceitar tudo aquilo que não é rejeitado. Essa abordagem descreve as proibições sintáticas sobre as palavras de uma sentença. O exemplo típico é abordagem por *constraint grammars* que foi descrita na seção 2.1.1.

Pudemos observar que a abordagem positiva exige um conhecimento profundo das variações posicionais e estruturais dos sintagmas e das próprias construções sintáticas a serem aceitas como sentença. Se levarmos em consideração que o Português é uma língua bastante flexível no que que se refere às estruturas, a abordagem positiva pode ser extremamente trabalhosa.

No entanto, mesmo na abordagem negativa, a preocupação em estabelecer os limites à flexibilidade sintática pode demandar bastante trabalho descritivo. O que significa dizer que as duas abordagens podem e devem ser investigadas.

A escolha que fizemos pelo gerador de analisadores sintáticos chamado BISON foi descrita no capítulo 3. Nossa gramática, portanto, segue as exigências prescritas para o funcionamento do BISON. A gramática segue a formatação e codificação específicas descritas em (DONNELLY, 2006).

A primeira parte da gramática contém as definições de todos os símbolos terminais a serem utilizados nas regras que produzirão os não-terminais de nosso interesse. Nesse momento podemos observar a integração entre a abordagem de linguagem de programação e o PLN. Inicialmente buscamos fazer com que cada terminal correspondesse a uma classe gramatical definida pela Gramática Normativa. Analogamente, os não-terminais procuravam corresponder aos sintagmas e às estruturas sintáticas oferecidos pela Gramática Normativa.

Posteriormente incluímos alguns terminais para palavras com comportamento sintático específico. O terminal para a pontuação final foi um deles. No entanto, a possibilidade de aumentar indiscriminadamente o número de terminais para tratar comportamentos específicos de algumas palavras em estruturas específicas nos pareceu uma alternativa que dificultaria a leitura da gramática. Procuramos nos limitar aos terminais extremamente necessários. Atualmente contamos com 28 terminais, dentre os quais estão Verbo, Substantivo e Adjetivo, por exemplo.

Na segunda parte da gramática descreve-se o conjunto das regras. Desenvolvemos as regras de maneira incremental a partir da estrutura mais simples que pudemos imaginar:

Frase : Sujeito Verbo Complemento_Verbal
;

Obviamente, mesmo para essa regra simples, tivemos que incluir as permutações aceitáveis.

Posteriormente nos demos conta de que algumas construções e algumas regras de posicionamento dependiam unicamente da transitividade do verbo. O complemento verbal, por exemplo, deve ser um objeto indireto, exigindo preposição no caso de o verbo ser transitivo indireto e um sintagma nominal para verbos nominais (DENICOLA, 1993). Além disso, verbos intransitivos dispensam objeto, enquanto que os verbos transitivos não podem fazer o mesmo. E assim nos demos conta de que o divisor de águas da construção de orações principais e subordinadas seria a transitividade dos verbos.

Extrato de regras escritas, sem a verificação de atributos, para alguns sintagmas verbais:

```
Sintagma_Verbal_Intransitivo
    :   Verbo_Intransitivo
    |   Verbo_Intransitivo Locucao_Adverbial_Composta
    |   Locucao_Adverbial_Composta Pontuacao Verbo_Intransitivo
    ;
Sintagma_Verbal_Transitivo_Direto
    :   Verbos_Transitivos_Diretos Objetos_Diretos
    |   Verbos_Transitivos_Diretos Objetos_Diretos Locucao_Adverbial_Composta
    |   Locucao_Adverbial_Composta Pontuacao Verbos_Transitivos_Diretos Objetos_Diretos
    ;
Sintagma_Verbal_Ligacao
    :   Verbos_Ligacao Predicativos
    |   Verbos_Ligacao Predicativos Locucao_Adverbial_Composta
    |   Locucao_Adverbial_Composta Pontuacao Verbos_Ligacao Predicativos
    ;
```

Obviamente essa abordagem gerou a repetição de muitas estruturas semelhantes em contextos diferentes. Porém, ela nos permitiu maior controle das dependências internas e algumas dependências externas das orações. Um exemplo prático disso é a facilidade de se fazer a verificação de regência se sabemos que o verbo exige objeto indireto.

No caso de dependências externas, o exemplo mais prático é a construção de uma oração que usa o verbo de outra oração. O caso típico é o da elipse do verbo como na sentença abaixo:

"Eu gosto de estudar e ele, de viajar."

Demos o nome de extensão a esse fenômeno, em que uma estrutura depende da estrutura de outra semelhante (mesma transitividade). Para tratar esse fenômeno, tivemos de adicionar regras como abaixo:

Sintagma_Oracional_Transitivo_Indireto_Completo:
Sintagma_Oracional_Transitivo_Indireto_Composto Conjuncao_Coordenativa Sujeitos Pontuacao Objetos_Indiretos;

Merece menção também a riqueza de possibilidades de construção dos sintagmas nominais. Elas começam no complexo dos qualificadores possíveis dos substantivos. Mas se multiplicam com a transitividade nominal. Ou seja, quando um substantivo, adjetivo ou advérbio é transitivo, ele exige um complemento nominal. Dessa forma, ele exige a descrição explícita de uma nova estrutura sintática, produzindo novas regras. Separamos os casos de substantivos transitivos dos casos não transitivos.

Poderíamos ter separado os verbos em diferentes terminais de acordo com a transitividade. Mas isso significaria multiplicar entradas no léxico, uma para cada transitividade do mesmo verbo. Em vez disso, realizamos a verificação da transitividade através de listas que associam o lema de um verbo às suas possíveis transitividades.

4.2.1 MANUTENIBILIDADE DA GRAMÁTICA

A escrita da gramática foi uma tarefa que exigiu bastante atenção e trabalho. A preocupação em escrever um conjunto de regras que pudesse fazer frente à expressividade da língua portuguesa foi um desafio.

Outro ponto importante diz respeito ao conjunto dos terminais. Poderíamos definir terminais para lidar com construções específicas. No entanto, isso poderia resultar em uma explosão de terminais da gramática, fenômeno que não encontramos na literatura, nem encontramos motivação linguística para essa abordagem. Por essa razão a limitação do conjunto de classes gramaticais se tornou uma opção de projeto.

A riqueza do conjunto de estruturas, a variabilidade das estruturas do ponto de vista posicional, bem como as exceções e proibições em contextos restritos foram as propriedades que impuseram um número sempre crescente de regras.

Por essa razão, nossa gramática conta atualmente com mais de 920 regras e mais de 1400 estados, gerando mais de 9200 conflitos de empilhamento/redução e mais de 6350 conflitos de redução/redução. O número por si só pode não significar muito, mas, para um ser humano, administrar um conjunto de regras com esse tamanho é tarefa nada trivial. A começar pela dificuldade de criar uma sequência de construções que seja inteligível. Ou seja, questões do tipo "o não-terminal que representa o sujeito deve ser definido perto do não-terminal que representa o complemento verbal ou daquele que representa o sintagma nominal?" começam a impor restrições cada vez mais difíceis de atender.

Além disso, a preocupação de não escrever diferentes regras para a mesma construção linguística começa a ser algo mais sutil de observar e, consequentemente, de reparar. Por essa razão, é necessária a compreensão profunda do encadeamento das regras, o que vai se tornando humanamente impossível com o aumento do conjunto de regras.

Esse fenômeno é significativo na abordagem livre de contexto por causa da semelhança de construções em diferentes contextos e pela natural interdependência das construções linguísticas. Isso significa que as próprias decisões de projeto na escrita da gramática têm impacto nas decisões posteriores, podendo até inviabilizá-las.

4.3 ANÁLISE LÉXICA

Conforme dito anteriormente, a informação produzida pela análise léxica constitui insumo necessário para análise sintática. Isso se deve ao fato de o analisador sintático depender do fornecimento da lista de terminais para uma dada sentença. Esse trabalho é feito pelo analisador léxico. A partir dessa lista de terminais, o analisador sintático verifica a pertinência da sentença de entrada a uma linguagem, ou, dito de outra maneira, a aceitação de uma sentença.

Dessa maneira, podemos entender que a dependência entre o analisador sintático e o analisador léxico é intrínseca ao processo de análise livre de contexto. Isso se aplica às linguagens de programação, que são eminentemente formais, e à linguagem natural, no nosso caso, o Português.

Como observado na seção 4.2, para recuperarmos os terminais de uma sentença, devemos ser capazes de obter a classe gramatical de cada palavra e de todos os seus homônimos. Mais do que isso, qualquer palavra do português deveria ter suas informações léxicas reconhecidas. Para que isso seja possível devemos dispor de algum mecanismo que reconheça todas as palavras e todas as flexões e derivações possíveis de todas as palavras variáveis do português.

Essa tarefa pode ser realizada de duas maneiras: através do desenvolvimento de uma análise léxica própria ou através do armazenamento de um conjunto significativo de palavras lexicalmente analisadas.

Um mecanismo de análise léxica foi desenvolvido, por exemplo, no sistema PALAVRAS. Para realizá-lo é necessário o conhecimento de um conjunto grande de palavras, de prefixos, de sufixos, de regras de formação de palavras, de mapeamentos luso-brasileiros e de algoritmos de limpeza, de busca e de análise. A tese de Edward Bick é bem instrutiva nesse ponto. Contudo, essa abordagem não é o objetivo desta dissertação, pois nosso interesse reside na análise sintática propriamente dita. Além disso, a exigência, mesmo nessa abordagem, de conhecermos um conjunto grande de palavras nos levou a escolher a outra abordagem.

Conjuntos de palavras lexicalmente analisadas são facilmente encontrados na forma de *corpus* anotados. Um corpus é um conjunto extenso de textos ou fragmentos de textos que segue alguma metodologia de seleção. Um corpus anotado é um corpus que teve cada sentença analisada léxica e/ou sintaticamente.

No nosso caso, temos o interesse de processar sentenças de jornais e revistas. Por essa razão, aproveitamos o corpus conhecido como CETEMPúblico(SANTOS, 2001b). Esse corpus teve por metodologia a seleção de textos do jornal português Público. O CETEMPúblico contém aproximadamente 180 milhões de palavras. No sítio Linguateca[6] nos foi disponibilizada uma versão anotada desse corpus. Esse corpus foi anotado com informações léxicas e sintáticas por uma versão antiga do sistema PALAVRAS. As informações sintáticas foram desprezadas.

Outro corpus que conseguimos, no mesmo sítio, foi o CETENFolha, que contém aproximadamente 24 milhões de palavras do jornal Folha de São Paulo. A anotação foi feita por uma versão mais recente do sistema PALAVRAS. Como o CETEMPúblico possui mais palavras e a formatação era mais simples de tratar, optamos por trabalhar somente com o CETEMPúblico.

O termo vocábulo será usado para representar uma sequência de caracteres que caracteriza uma palavra. Um vocábulo não inclui, portanto, espaços em branco. O termo sentença será usado para representar uma sequência de caracteres a ser processada pela plataforma. A sentença eventualmente poderá ser vista como uma sequência de vocábulos.

[6]www.linguateca.pt

Para cada vocábulo da sentença, a análise léxica produz os seguintes dados:

- **Vocábulo**: é a expressão escrita da palavra;
- **Conjunto de homônimos**: palavras diferentes com mesma ortografia.

Cada homônimo possui as seguintes informações:

- **Lema**: é o radical da palavra. Pode ser diferente do vocábulo no caso de palavras variáveis;
- **Classe Gramatical**: é o terminal que definimos na gramática para representar a classe da palavra segundo a Gramática Normativa;
- **Conjunto de atributos**: contém a representação das informações morfológicas da palavra. Por exemplo: Gênero, Número e Pessoa.

Como exemplo, a análise léxica do vocábulo "é" produz as seguintes informações:

- **Vocábulo**: é
- **Lema**: ser
- **Classe Gramatical**: Verbo
- **Gênero**: Uniforme
- **Número**: Singular
- **Pessoa**: Terceira
- **Modo**: Indicativo
- **Tempo**: Presente

Para facilitar a leitura dessas informações em sentenças, adotaremos a convenção de apresentar em forma de texto formatado como se segue:

$$\text{é} = [\text{Verbo}\{\text{ser}\} : \text{Uniforme - Singular - Terceira - Indicativo - Presente}]$$

A análise léxica da sentença “Alegria é vida.” produz os seguintes dados:

a) alegria = [Substantivo{alegria} : Feminino - Singular -];

b) é = [Verbo{ser} : Uniforme - Singular - Terceira - Indicativo - Presente];

c) vida = [Substantivo{vida} : Feminino - Singular -];

d) . = [Pontuacao_Final{.} :].

Podemos observar que a palavra “alegria” aparece com todas as letras minúsculas. Isso acontece pois a plataforma só está trabalhando com as letras minúsculas por questões de simplificação.

No exemplo acima, cada vocábulo possui uma única palavra associada. Porém, antes mesmo de começarmos a abordar a análise sintática da linguagem natural, já sabíamos que um tipo fundamental de ambiguidade precisaria ser tratado: a ambiguidade léxica. A presença de palavras diferentes com a mesma ortografia é um fenômeno conhecido da língua portuguesa. Basta pensarmos em vocábulos como: revista, guarda ou corrente. Cada um deles possui dois ou mais homônimos associados.

Cada homônimo pode representar uma classe gramatical diferente ou um conjunto diferente de atributos. Isso significa que, para a análise sintática, um vocábulo pode oferecer mais de uma alternativa de terminal da linguagem ou de restrições de contexto diferentes. Dessa maneira, nossa análise sintática utiliza cada homônimo de um vocábulo para uma análise independente.

A frase “Eu vejo o amor.” produz as seguintes informações léxicas:

a) eu

- eu = [Pronome_Pessoal_Reto{eu} : Uniforme - Singular - Primeira -];
- eu = [Substantivo{eu} : Masculino - Singular -].

b) vejo

- vejo = [Verbo{ver} : Uniforme - Singular - Primeira - Indicativo - Presente].

c) o

- o = [Artigo{o} : Masculino - Singular -];
- o = [Pronome_Pessoal_Atono{o} : Masculino - Singular - Terceira -].

d) amor

- amor = [Substantivo{amor} : Masculino - Singular -].

e) .

- . = [Pontuacao_Final{.} :].

Além da ocorrência de homônimos, a língua portuguesa permite que diferentes palavras compartilhem o mesmo vocábulo. As contrações são o caso típico dessa situação. Elas normalmente acontecem entre preposição e pronome (“àquele” = “a” + “aquele”, por exemplo) ou entre preposição e artigo (“do” = “de” + “o”, por exemplo). Como o conjunto das contrações não é muito grande, obtemos todas as contrações por meio de listas textuais que editamos manualmente.

A frase “Viemos doutro lugar.” produz as seguintes informações léxicas:

a) viemos = [Verbo{vir} : Uniforme - Plural - Primeira - Indicativo - Preterito_Perfeito].

b) doutro = [Preposicao_Essencial{de} : , Pronome_Indefinido{outro} : Masculino - Singular - Terceira -].

c) lugar = [Substantivo{lugar} : Masculino - Singular -].

d) . = [Pontuacao_Final{.} :].

Como a análise sintática não leva em consideração os vocábulos, a contração armazena todas as informações necessárias. Observe que as contrações são tratadas como palavras separadas, mas que devem ser processadas como uma única alternativa. Isso significa que, por exemplo, para a contração

$$\text{da} = [\text{Preposicao\_Essencial}\{\text{de}\} : , \text{Artigo}\{\text{o}\} : \text{Feminino - Singular - }]$$

a preposição deve ser processada e somente o artigo deve ser processado na sequência. Não pode acontecer, após a preposição “de”, ser feita qualquer outra escolha entre os homônimos do vocábulo “a”.

Outro fenômeno da língua portuguesa é a ocorrência de expressões multivocabulares (SILVA, 2008). As expressões multivocabulares são palavras cuja expressão é composta por uma sequência de vocábulos. Dessa maneira, uma expressão multivocabular possui uma classe gramatical, bem como as demais informações léxicas.

As entidades mencionadas são fonte inesgotável de exemplos de expressões multivocabulares: “Ministério da Defesa”, “Exército Brasileiro”, “Instituto Militar de Engenharia”. Além delas, podemos também pensar em expressões com função sintática específica, tais como preposições acidentais (“em caso de”, “na hora de”) (DENICOLA, 1993) ou conjunções (“bem como”).

A sentença “Por isso durmo aqui.” produz duas sentenças distintas:

- a) por = [Preposicao_Essencial{por} :].
  b) isso = [Pronome_Demonstrativo_Invariavel{isso} : Masculino - Singular - Terceira -].
  c) durmo = [Verbo{dormir} : Uniforme - Singular - Primeira - Indicativo - Presente].
  d) aqui = [Adverbio{aqui} :].
  e) . = [Pontuacao_Final{.} :].

- a) por isso = [Conjuncao_Coordenativa{por isso} :].
  b) durmo = [Verbo{dormir} : Uniforme - Singular - Primeira - Indicativo - Presente].
  c) aqui = [Adverbio{aqui} :].
  d) . = [Pontuacao_Final{.} :].

A identificação das expressões multivocabulares começa em listas de expressões multivocabulares conhecidas. Nestas listas são incluídas as informações léxicas da palavra associada à expressão multivocabular. Quando uma nova sentença será processada, é verificada a ocorrência de alguma expressão multivocabular conhecida. Uma vez identificada uma expressão, são criadas duas sentenças alternativas: uma com a expressão e outra sem a expressão. Dessa maneira podemos ter várias sentenças alternativas para uma mesma sentença de entrada. Na sentença com a expressão, são retirados os vocábulos dessa expressão e já se sabe qual palavra entrará nessa posição da sentença. Depois disso a verificação de expressões continua nos vocábulos remanescentes da sentença.

Nessa busca por expressões existe uma sutileza: supomos que as expressões multivocabulares são compostas por vocábulos contíguos. Existe, no entanto, um conjunto especial de expressões multivocabulares cujos vocábulos não estão em sequência na sentença. Chamamos essas expressões de expressões multivocabulares descontínuas. O caso típico é a conjunção coordenativa “não só..., mas também” cujos vocábulos se encontram separados em dois grupos.

Essa situação só pode ser tratada por gramática livre de contexto se juntarmos os grupos de vocábulos. Por essa razão, de maneira similar ao tratamento das expressões multivocabulares comuns, retiramos os vocábulos da expressão multivocabular e colocamos a palavra associada na posição do segundo grupo de vocábulos. A razão dessa escolha se deve ao fato de os exemplos mais conhecidos de expressão multivocabular descontínuas serem as conjunções coordenativas.

A sentença "Ora quero, ora detesto." produz duas sentenças:

- a) ora
  - ora = [Preposicao_Essencial{por} :].
  - ora = [Interjeicao{ora} :].
  - ora = [Verbo{orar} : Uniforme - Singular - Terceira - Indicativo - Presente]

  b) quero = [Verbo{querer} : Uniforme - Singular - Primeira - Indicativo - Presente]

  c) ora
  - ora = [Preposicao_Essencial{por} :].
  - ora = [Interjeicao{ora} :].
  - ora = [Verbo{orar} : Uniforme - Singular - Terceira - Indicativo - Presente]

  d) detesto = [Verbo{detestar} : Uniforme - Singular - Primeira - Indicativo - Presente]

  e) . = [Pontuacao_Final{.} :].

- a) quero = [Verbo{querer} : Uniforme - Singular - Primeira - Indicativo - Presente]

  b) ora*, ora[7] = [Conjuncao_Coordenativa{ora*, ora} :]

  c) detesto = [Verbo{detestar} : Uniforme - Singular - Primeira - Indicativo - Presente]

  d) . = [Pontuacao_Final{.} :].

[7] O asterisco é uma representação interna da plataforma para a palavra associada a uma expressão multivocabular descontínua.

Além dos fenômenos da língua portuguesa, tivemos de lidar com um alguns problemas frutos da escolha por realizar a importação das informações léxicas do corpus anotado. Podemos observar a falta de certos vocábulos no corpus e a importação de marcações erradas.

A falta de vocábulos pode ser entendida se levarmos em consideração que o corpus tem somente algo em torno de meio milhão de palavras distintas. Além disso, como a fonte é um jornal português, existem palavras cuja ortografia é diferente da ortografia brasileira.

Os erros de marcação foram produzidos primeiramente por erros na confecção do próprio corpus e, posteriormente, por erros de classificação do analisador PALAVRAS. Considerando-se que o número de vocábulos é muito grande, percentuais pequenos de erro podem produzir ainda assim muitos erros.

Para lidar com mais esse problema, desenvolvemos uma ferramenta de edição do léxico obtido a partir do corpus. Essa ferramenta foi utilizada para a correção manual de todos os vocábulos presentes nos testes apresentados no capítulo 7. Ou seja, os resultados da análise sintática apresentados supõem que esta foi feita a partir de um analisador léxico perfeito.

4.4 ANÁLISE SINTÁTICA *GLR*

No capítulo 3, vimos que a linguística nos oferece estruturas sintáticas que são ambíguas. Vimos também que a análise *GLR* foi desenvolvida para tratar problemas de ambiguidade sintática em PLN. Nosso desafio foi desenvolver um mecanismo que realizasse a aplicação de todas as alternativas sintáticas sobre cada alternativa léxica. Isso significa aplicar todas as ações e transições possíveis para o processamento de todas as sentenças constituídas de todos os homônimos de todos os seus vocábulos.

Desde o início se tornou claro que a primeira tarefa seria a definição de estruturas de dados que permitissem a organização de todas essas informações para cada processamento distinto. Por essa razão, essa definição sofreu diversas alterações ao longo do desenvolvimento da plataforma. Muitas dessas alterações foram motivadas por necessidades de processamento de informações específicas que as estruturas ainda não comportavam.

As estruturas de dados para as informações léxicas nos sugeriam, desde o início, uma organização geral em formato de listas. Primeiramente essa convicção se formou pela ideia de organização de dicionários e léxicos e, depois, pela necessidade de verificação sequencial das alternativas léxicas. Seguindo o mesmo princípio, organizamos as sentenças possíveis em listas de palavras e todas as sentenças em listas.

As informações das tabelas de decisão foram organizadas em listas para a aplicação sequencial de todas as decisões de ação e de transição. As informações sintáticas dos processamentos parciais foram armazenadas em pilhas de estados, como normalmente acontece na análise LR. Os atributos, por sua vez, foram organizados em forma de conjunto de mapeamentos.

A verificação de contexto nos exigiu o conhecimento da topologia das estruturas sintáticas durante o processamento parcial. Para representar essa topologia, tivemos de armazenar cada árvore que compõe cada símbolo da pilha de símbolos. Cada árvore contém todos os símbolos que a compõem, sejam terminais ou não-terminais. Armazenamos essas árvores em uma pilha, que chamamos de pilha de subárvores.

Dessa maneira, cada processamento parcial deve armazenar uma pilha de subárvores e uma pilha de estados. A esse conjunto de informações de um processamento parcial damos o nome de configuração.

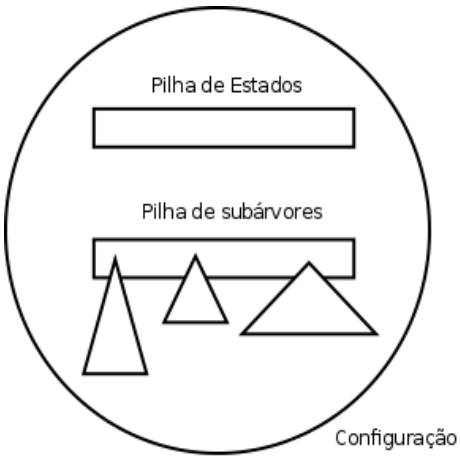

FIG. 4.1: Esquema de uma configuração

A partir do estado inicial da análise sintática LR(HOPCROFT, 2001; BAUER, 1976), construímos a configuração inicial. Dada uma configuração, criamos uma cópia sua para cada ação ou transição disponível no estado em que ela se encontra. Como as sentenças e a gramática são finitas, o algoritmo, ainda que crie muitas configurações, pára.

Se pensarmos a análise GLR como uma árvore de busca, cada configuração corresponde a um caminho distinto a partir da raiz, o estado inicial. Como cada sentença dá início a um processamento independente, diferentes estados iniciais podem ser criados, um para cada sentença. Além disso, como cada processamento pode produzir uma árvore distinta, o resultado da plataforma são florestas sintáticas. Uma floresta para cada sentença possível.

4.4.1 SEQUÊNCIA DE ANÁLISE SINTÁTICA

Para que se torne mais claro o funcionamento da plataforma, incluímos aqui a sequência de análise de uma sentença:

a) Ao iniciar o programa, é solicitada a digitação da sentença.

b) Informada a sentença, o programa tentará identificar todas as possíveis expressões e estruturas da sentença.

c) Para cada possível interpretação da sentença, o Dicionário do programa irá atribuir para cada vocábulo seu conjunto de homônimos. Cada homônimo contém suas próprias informações morfológicas (conjunto de atributos) que o caracterizam.

d) Para cada interpretação da sentença, a análise *GLR* é realizada.

e) Ao final da análise aparece uma componente visual como na figura 4.2.

f) Para sair do programa, o usuário digita “quit”.

"Ele gosta dela."

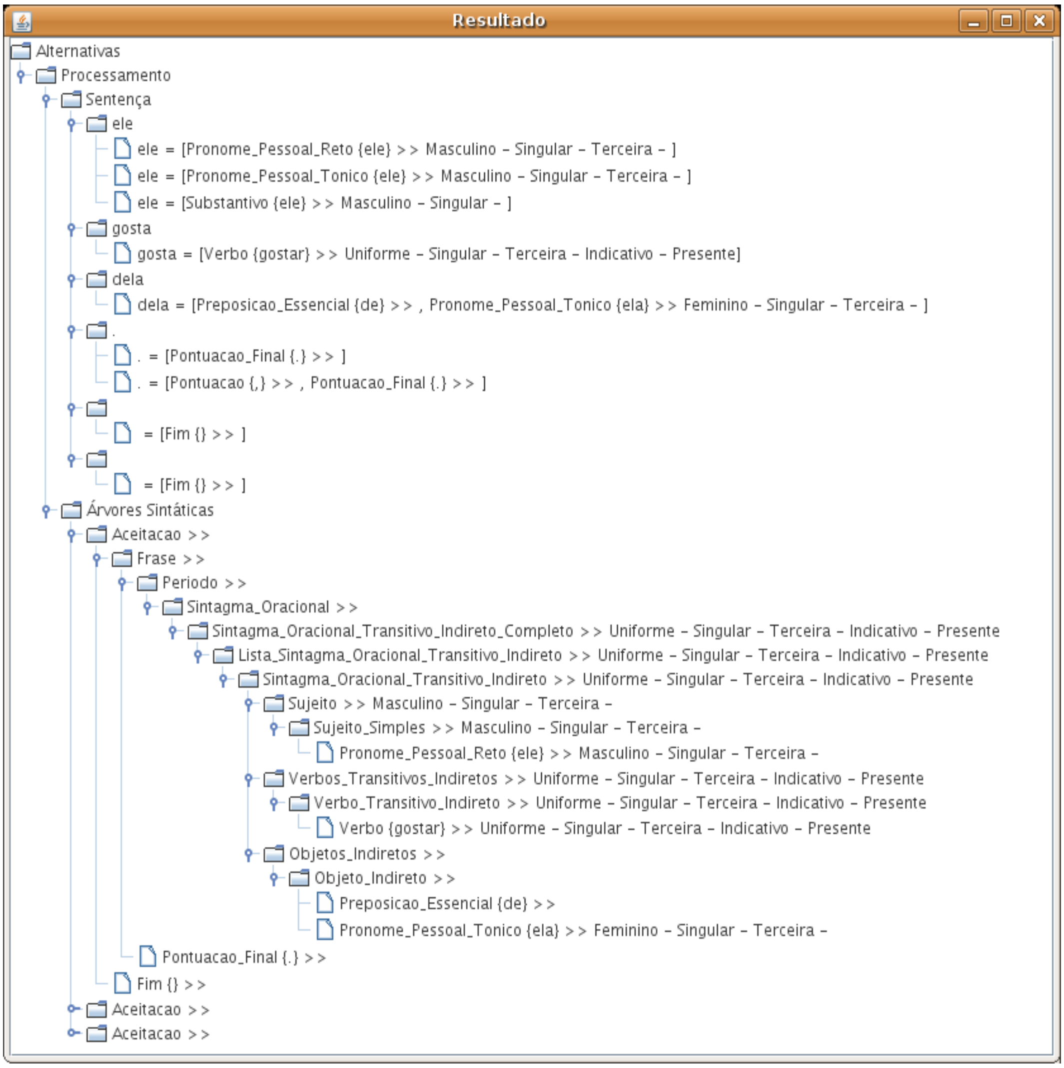

FIG. 4.2: Resultado de análise sintática

O resultado da análise sintática é organizado em forma de árvore, cujo nó raiz se chama "Alternativas". Cada alternativa de sentença é analisada independentemente e o resultado de sua análise é armazenado em uma subárvore de raiz "Processamento". Cada subárvore de cada sentença possui as subárvores "Sentença" e "Árvores Sintáticas".

A subárvore "Sentença" apresenta o conjunto de homônimos de cada palavra da sentença. A subárvore "Árvores Sintáticas" apresenta todas as árvores sintáticas que foram construídas para essa sentença. Cada árvore sintática distinta de uma sentença é representada por uma subárvore cuja raiz sempre recebe o rótulo "Aceitação".

4.4.2 EXPLOSÃO COMBINATÓRIA

Na realização da análise GLR para o Português, a verificação pura de todas as possibilidades apresenta o problema de explosão combinatória. Chegamos a observar a criação, durante o processamento, de algo em torno de 2 milhões de árvores para uma única frase. Além disso, já realizamos verificações de frases que consumiram horas e tiveram de ser interrompidas.

Observamos que o processamento consumia toda memória principal disponível, no computador de testes, algo em torno de 750 MB. A partir do momento em que o processamento começava a exigir a paginação da memória, o processo deveria, invariavelmente, ser interrompido.

Observamos que a explosão combinatória não é determinada pelas possibilidades de associação léxica. O número de possibilidades de associação léxica para um vocábulo é, em geral, menor que 10 e, numa frase, o número de vocábulos ambíguos também fica abaixo de 10.

As expressões multivocabulares, por sua vez, multiplicam o número de sentenças. No entanto, elas diminuem o número de símbolos a serem processados para uma sentença. Assim, o maior tempo de processamento de uma alternativa de sentença tende a ocorrer quando os vocábulos não compõem expressões.

Observamos, por outro lado, através do acompanhamento do processamento de frases, que a explosão combinatória é determinada sim pela redundância da gramática, ou seja, pelas possibilidades de redução nos estados da análise GLR.

5 ANÁLISE SINTÁTICA COM RESTRIÇÕES

Neste capítulo apresentamos as soluções desenvolvidas para o tratamento dos problemas mencionados no capítulo 4. A seção 5.1 apresenta as soluções para tratar a explosão combinatória e seus efeitos. A seção 5.2 apresenta as soluções para tratar a manutenibilidade da gramática.

5.1 TRATAMENTO DO PROBLEMA DE EXPLOSÃO COMBINATÓRIA

Conforme vimos na subseção 4.4.2, a criação, pela análise GLR, de muitas configurações para uma mesma sentença pode consumir todos os recursos de memória disponíveis. O resultado desse fenômeno é a demora no processamento devido à paginação de memória. Isso faz com que o tempo de resposta não dependa exclusivamente do número de combinações de ambiguidades léxicas e sintáticas. Além disso, o computador que realiza o processamento fica indisponível para outras atividades. Nessas condições, a possibilidade de uso prático da plataforma fica comprometida.

Apresentamos nesta seção cada um dos mecanismos empregados no controle da explosão combinatória. Na subseção 5.1.1, apresentamos um mecanismo para conter a criação de configurações. Nas subseções 5.1.2 e 5.1.3, apresentamos mecanismos para eliminar configurações. Na subseção 5.1.4, apresentamos uma solução para tratar exclusivamente o efeito da explosão combinatória sobre a memória.

5.1.1 REFINAMENTO DA GRAMÁTICA

A causa da explosão combinatória reside na ambiguidade da gramática. Logo, tornar a gramática menos ambígua é um objetivo a ser perseguido sempre. Procuramos transformar algumas estruturas sintáticas que geravam muitas alternativas equivalentes de análise sintática. Colocamos abaixo um exemplo de regra que procuramos retirar da gramática:

Sintagma_Nominal : *Substantivo*
| *Sintagma_Nominal Adjetivo*
| *Adjetivo Sintagma_Nominal*
;

Esse tipo de mecanismo de construção de não-terminais é redundante por natureza. Diferentes árvores podem ser criadas somente devido a diferenças na estrutura interna do sintagma nominal. Essas diferenças são fruto da ambiguidade na escolha da ordem dos empilhamentos e das reduções que a descrição acima oferece. Assim, o sintagma "bonito carro amarelo" poderia ter pelo menos duas estruturas diferentes, como mostra a figura abaixo:

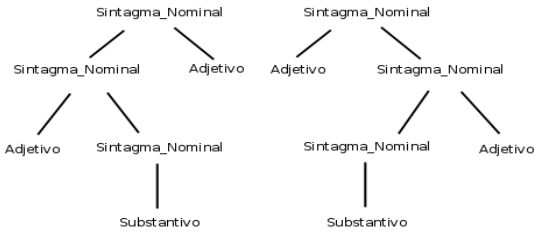

FIG. 5.1: Ambiguidade estrutural

As diferenças estruturais nesse caso não contribuem com informação nova interessante. Por essa razão, admitir sua ocorrência significa desperdício de recursos. Procuramos eliminar todos os tipos de ambiguidades irrelevantes para diminuir o esforço de processamento.

No entanto, o maior desafio para o uso de uma gramática livre de contexto é descrever todas as possibilidades sintáticas. Ou seja, alcançar a expressividade da língua natural exige o acréscimo de novas regras e estruturas quando alguma variante linguística nova aparece. Assim, para fazer frente aos testes iniciais de preparação da plataforma, tivemos de incluir muitas regras às primeiras versões da gramática.

Nossa preocupação foi escrever regras suficientemente expressivas, que não gerassem muitas ambiguidades. Sabemos, no entanto, que o controle da ambiguidade na gramática exige o profundo conhecimento das decisões anteriormente tomadas para a escrita da gramática. Nossa limitação de tempo não nos permitiu continuar a rescrever e a refinar a gramática até o ponto que gostaríamos.

5.1.2 VERIFICAÇÃO DE ATRIBUTOS

Nossa primeira tentativa automatizada de controle efetivo da produção de configurações foi a aplicação de gramática de atributos durante a análise sintática. Ela nos permitiu diminuir drasticamente o número de configurações aceitas, pois suas restrições se aplicam a todos os níveis da análise sintática, desde as palavras isoladas até às orações.

As principais restrições que desejávamos aplicar eram aquelas concernentes à concordância. Não queríamos que nossa plataforma aceitasse formas não gramaticais que ferissem os princípios de concordância, seja ela verbal ou nominal.

- *O carro chegou.*
- **O carro chegaram.*[8] (Problema de concordância verbal)
- **O luz chegou.* (Problema de concordância nominal)

No entanto, a gramática de atributos nos ofereceu a possibilidade de desenvolvimento de muitas outras restrições. Como por exemplo a determinação de regência de verbos e nomes (substantivos, adjetivos e advérbios) transitivos.

- *Gosto de você.*
- **Gosto em você.* (Problema de regência verbal)
- *Todos devem respeito aos mais velhos.*
- **Todos devem respeito sobre os mais velhos.* (Problema de regência nominal)

Conseguimos ainda descrever outro aspecto: a seleção de terminais que podem participar de uma regra. No exemplo abaixo, queremos selecionar quais verbos podem ser transitivos indiretos.

- *Gosto de você.*
- **Influenciamos de você.* (O verbo influenciar é sempre transitivo direto)

[8] É comum entre os linguistas o emprego do asterisco para indicar uma sentença não gramatical. Veja por exemplo (CHOMSKY, 2006)

A gramática de atributos nos permitiu descrever e aplicar essas restrições durante a análise sintática. Essas restrições têm o poder de simplificar a escrita da GLC, evitando a descrição explícita de terminais e de estruturas específicas que, de outra maneira, deveriam ser repetidas para cada combinação de atributos. Como exemplo, a estrutura sintática denominada **Sujeito** deveria ser desdobrada em estruturas semelhantes, tais como **Sujeito_Masculino_Singular_Primeira** ou **Sujeito_Feminino_Plural_Terceira**. O mesmo raciocínio se aplica a todas as outras estruturas que carregam informações morfológicas. Ou seja, essas restrições têm caráter puramente combinatório, nos permitindo reduzir significativamente o número de terminais e não-terminais da GLC.

A verificação de atributos é o nome que damos ao trabalho de verificar se as condições impostas pela gramática de atributos estão sendo atendidas em um dado contexto. O contexto a que nos referimos é sempre o contexto de uma regra da GLC. Para implementar a verificação de atributos, precisávamos associar a cada regra da GLC uma descrição de suas restrições. A maneira mais simples que encontramos foi descrever as restrições na mesma linha da regra a que elas se associam. Isso significa que tivemos de escrever no texto da GLC as descrições da gramática de atributos. Para tanto, usamos comentários na gramática do BISON, os quais processamos e transformamos em restrições associadas a cada regra.

Porém, como vimos no capítulo 3, a gramática de atributos pode ser usada para verificar restrições e para definir os atributos do não-terminal que uma regra constrói. Por essa razão, dentro do espaço criado para o comentário, criamos dois campos distintos: o campo das restrições e o campo dos novos atributos.

O campo das restrições é delimitado por "%!" e "!%". Nesse campo são escritas todas as restrições que podem rejeitar a aplicação da regra. Apresentaremos agora alguns exemplos esquemáticos de regras da GLC com as descrições de gramática de atributos.

```
Sintagma_Oracional_Intransitivo
   :   Sujeitos Verbo_Intransitivo /* %! concordancia[<1,2>] !% */
   ;
```

Nesse exemplo, a restrição implementada pela função booleana **concordancia** será aplicada no contexto dessa regra. A definição de como serão as aplicações da função é feita dentro dos colchetes Os colchetes devem sempre ser colocados imediatamente após o nome da função. Cada aplicação da função corresponde a uma tupla definida pelos símbolos "<" e ">". As tuplas são separadas por vírgulas e os argumentos dentro das tuplas também.

Os elementos na tupla podem ser de três tipos: inteiros, *strings* sem espaço e referências a classes de atributos. Os inteiros são referências aos símbolos dentro da própria regra. Um número inteiro indica a posição que o símbolo ocupa dentro da regra, a começar pela posição 1 que fica mais à esquerda. Os *strings* são interpretados diretamente como argumentos de funções, não sofrendo qualquer manipulação. O segundo exemplo abaixo mostra o uso de *strings*. As referências a classes de atributos é identificada no texto pela presença do ponto no meio de uma *string*. Dessa maneira, se quisermos utilizar um atributo constante dentro das restrições, fazemos uso dessas referências. Por exemplo, para usarmos o atributo que representa o número plural, escrevemos **Numero.PLURAL**. Essa referência é uma chamada a uma variável estática da classe **Numero** que representa a informação morfológica de número. O terceiro exemplo abaixo mostra o uso de chamadas dessa natureza.

Assim, o exemplo acima quer dizer que, para que os símbolos **Sujeitos** e **Verbo_Intransitivo** formem um **Sintagma_Oracional_Intransitivo**, é necessário que eles concordem. A função **concordancia** implementa a concordância verbal e nominal.

```
Sintagma_Oracional_Transitivo_Indireto
   :   Sujeitos Verbos_Transitivos_Indiretos Objetos_Indiretos
       /* %! concordancia[<1,2>] regencias[<2,3>] !% */
   ;
```

Nesse exemplo, vemos a aplicação de mais de uma regra de verificação. A regra só poderá ser aplicada se todas as suas funções booleanas associadas retornarem o valor verdadeiro. As funções são separadas por espaço.

O exemplo mostra o uso da verificação de regências. Todas as funções que lidam com regências tem por base listas de associações entre lemas de palavras e listas de preposições. Cada associação representa o conjunto de preposições que pertencem à regência de uma palavra. As listas foram elaboradas manualmente e precisam ser atualizadas toda vez que um novo verbo que admite objeto indireto ou um novo nome transitivo deve ser processado pela plataforma.

```
Verbo_Transitivo_Direto_Passiva_Sintetica
   :  Verbo Pronome_Pessoal_Atono
      /* %! funcao[<verbo_transitivo_direto,1>] funcao[<pronome_apassivador,2>] !% */;
```

Quando desejamos selecionar quais representantes de um determinado terminal podem participar de uma regra, estamos atribuindo uma função específica para os representantes selecionados. Para implementar estas funções, utilizamos listas de lemas para cada função específica. O exemplo acima mostra o uso das funções **verbo_transitivo_direto** e da função **pronome_apassivador** para selecionar as palavras.

O campo dos novos atributos é delimitado por "#!" e "!#". Nesse campo são descritas as maneiras de se obter os atributos do novo não-terminal a partir dos atributos disponíveis pelos símbolos da regra ou a partir de constantes. No exemplo abaixo veremos diversos aspectos da utilização do campo de novos atributos. Numeramos as regras para facilitar a leitura.

```
1)          Sujeito   :  Sujeito_Simples
                         /* #! nominal[<1>] !# */
2)                    |  Lista_Sujeito Conjuncao_Coordenativa Sujeito_Simples
                         /* %! funcao[<conjuncao_aditiva,2>] !%
                         #! nominal[<1,3>] mapear[<Numero.PLURAL>] !# */;
3)  Lista_Sujeito     :  Sujeito_Simples
                         /* #! nominal[<1>] !# */
4)                    |  Lista_Sujeito Pontuacao Sujeito
                         /* #! nominal[<1,3>] mapear[<Numero.PLURAL>] !# */;
```

Nesse exemplo, o não-terminal **Sujeito** pode ser construído de duas maneiras: a partir de um não-terminal **Sujeito_Simples** ou a partir da conjunção entre um não-terminal **Lista_Sujeito** e um não-terminal **Sujeito_Simples**. A motivação para essa descrição recursiva é devida à possibilidade da construção de sujeitos compostos.

"O banco, a indústria e o comércio cresceram."

A lista dos componentes do não-terminal **Lista_Sujeito** é descrita pela presença de vírgula (terminal **Pontuacao**). A lista é sempre concluída com a conjunção. Esse mecanismo de criação de listas é utilizado em diversas outras situações ao longo da gramática. Estudaremos melhor esse mecanismo na subseção 5.2.2. Observemos, no entanto, que para evitar que essa recursividade se tornasse uma ambiguidade estrutural, somente admitimos a composição em todos os casos à direita.

Nas regras 1 e 3, os atributos do novo não-terminal são obtidos pela extração dos atributos nominais do primeiro símbolo da regra. Nas regras 2 e 4, os atributos são obtidos pela extração dos atributos nominais dos símbolos nas posições 1 e 3. Depois é feita a atribuição de uma constante: **Numero.PLURAL**. Observe-se que a ordem é importante, pois no campo de novos atributos um método pode sobrescrever a atribuição feita por um método anterior. A atribuição dessa constante deve ser feita pois os atributos **Numero** dos símbolos 1 e 3 podem ser ambos **Numero.SINGULAR**. Isso acontece na frase acima em que cada componente do sujeito está no singular. Portanto, o sujeito composto deve receber o atributo **Numero.PLURAL** quando a conjunção é aditiva.

Para observar o funcionamento do mecanismo de redução de configurações, apresentamos duas sentenças muito semelhantes:

a) *Ele chamou a meninos de rua.*

b) *Ele chamou a menina de rua.*

A primeira sentença admite somente uma árvore sintática em que o verbo "chamar" é transitivo indireto. A segunda sentença, por sua vez, admite uma árvore sintática em que o verbo "chamar" é transitivo indireto e outra árvore em que o mesmo verbo é transitivo direto. A exclusão de uma das árvores para a primeira sentença ocorre devido à restrição de concordância entre o artigo "a" e o substantivo "meninos".

A segunda sentença é aceita pela plataforma de duas formas, pois o verbo chamar está na lista dos verbos de transitividade indireta e na lista dos verbos de transitividade direta. Dessa maneira, também aqui fazemos uso da seleção do terminal que vai participar das construções. Além disso, a regência também foi verificada. A regência do verbo "chamar" está descrita para aceitar as preposições "a", "por" e "de".

Os resultados com a verificação de atributos foram impressionantes, com ganhos significativos de desempenho, principalmente para as frases complexas. A frase abaixo não pode ser processada sem a verificação de atributos:

"O principal deles é avaliar se a companhia seguradora possui patrimônio suficientemente elevado para garantir a indenização em caso de sinistro, bem como a segurança que o seu nome inspira no mercado."(CAMARGO, 1996) (1)

5.1.3 AMBIGUIDADE DE POSICIONAMENTO DE SUBESTRUTURAS SINTÁTICAS

Estudando melhor as árvores resultantes após o processamento da frase (1), observamos que muitas árvores eram diferentes somente devido a diferenças no posicionamento da locução adverbial "no mercado" na árvore sintática. Entendemos esse fenômeno como um problema, pois nos pareceu artificial aceitar que a locução adverbial não pertença à oração mais próxima,ou seja, à ultima à direita. Dessa maneira, passamos a denominá-lo como problema de posicionamento.

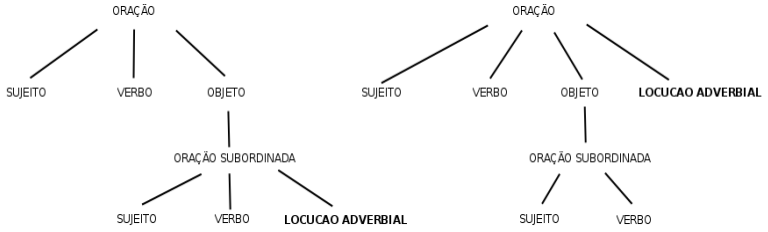

FIG. 5.2: Exemplo de problema de posicionamento

Trata-se de uma ambiguidade da gramática do português. Mas entendemos que, em função da semântica usual, a locução adverbial deveria ser associada somente à oração mais profunda na árvore, que é a oração que apareceu por último. No entanto, em nossa gramática, todos os não-terminais que representam orações possuem regras em que a locução adverbial participa na última posição da regra. Ou seja, essa é uma regra geral que ocasiona o problema de posicionamento.

O tratamento desse problema através de alteração da GLC ocasiona multiplicação de terminais e não-terminais de forma análoga ao problema do *dangling else* em linguagens de programação (AHO, 1986).

Resolver o problema de posicionamento implica excluir, em determinados contextos, algumas regras. Em outras palavras, a validade das regras de construção de não-terminais deixa de ser absoluta e passa a ser relativa ao contexto em que a regra pode ser aplicada.

No caso da frase (1) a oração principal "O principal deles é..." é uma oração com verbo de ligação. Em situações normais, a oração com verbo de ligação admite uma locução adverbial na última posição de uma regra de construção. Por exemplo: "Ele fica alegre de manhã.".

Contudo, a verificação desse tipo de restrição deve ser feito a partir do contexto da regra que se pode aplicar em um determinado estado. O objetivo é fazer com que sejam verificadas as incompatibilidades para que a regra não seja aplicada e o número de configurações não cresça. Mas, para verificar essa restrição a partir do contexto de uma regra, deveríamos ter conhecimento da estrutura interna das subárvores de cada não-terminal na pilha de símbolos da configuração atual e, alem disso, deveríamos ter a capacidade de verificar todas as regras de construção disponíveis para cada não-terminal dessas subárvores.

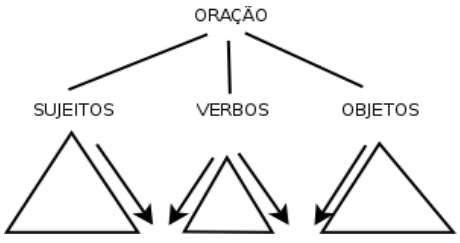

FIG. 5.3: Verificações de borda

Tratamos apenas o caso em que a verificação acontece somente para os não-terminais nas bordas, ou fronteiras, das subárvores. Por essa razão demos o nome de *verificação de borda* ao mecanismo desenvolvido e implementado em nossa plataforma para tratar problemas de posicionamento.

Existem outras possibilidades de restrições de posicionamento que podem ser estudadas e implementadas. A mesma frase (1) oferece a possibilidade de se estudar o fenômeno de posicionamento do objeto direto em diferentes orações transitivas diretas que estejam aninhadas. Chegamos a identificar este fenômeno, mas não implementamos as restrições para tratá-lo. Sua implementação é mais complexa devido à possibilidade de composição de objetos diretos com conjunção, o que nos obriga a fazer verificações além da borda das subárvores, pois a conjunção ocupa essa posição especial.

A verificação consiste em descobrir se algum não-terminal da borda à esquerda poderia ser construído utilizando-se algum não-terminal à direita. Para que isso fosse possível era necessário algum mecanismo que permitisse a navegação entre diferentes contextos. Isso só foi conseguido quando passamos a representar a própria gramática em um objeto a ser utilizado durante a própria análise sintática. Esse objeto encapsula um mapeamento entre cada não-terminal e a lista de regras que o derivam.

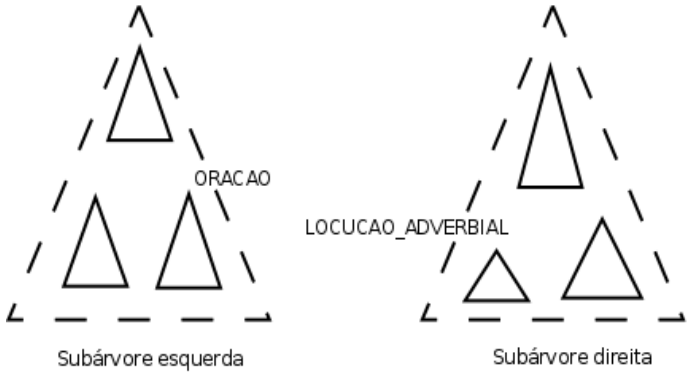

FIG. 5.4: Problema de posicionamento

Tomamos como borda à esquerda todos os não-terminais à esquerda abaixo da raiz e à direita todos os não-terminais à direita. Evitamos a raiz à esquerda para não rejeitarmos reduções válidas, por exemplo para uma regra do tipo:

Oracao : Oracao Locucao_Adverbial
;

A verificação ocorre em duas etapas: verificação direta e a verificação com os filhos. A verificação direta tenta acrescentar ao não-terminal à esquerda um não-terminal à direita para ver se existe alguma regra que derive o primeiro não-terminal a partir dos dois não-terminais.

A verificação com os filhos é semelhante, contudo, o não-terminal à direita é acrescentado aos filhos na subárvore do não-terminal à esquerda. A ideia é verificar que nenhum não-terminal na borda direita poderia ter participado da construção de algum não-terminal da borda esquerda, considerando-se os símbolos que foram efetivamente usados na regra que construiu o não-terminal à esquerda.

```
Sintagma_Oracional_Intransitivo
    :   Sujeitos Verbo_Intransitivo /* %! concordancia[<1,2>] !% */
    |   Sujeitos Verbo_Intransitivo Locucao_Adverbial_Composta /* %! concordancia[<1,2>] !% */;
```

No exemplo acima, se a verificação de problema de posicionamento encontrasse o não-terminal **Sintagma_Oracional_Intransitivo** à esquerda e o não-terminal **Locucao_Adverbial_Composta** à direita, não seria identificado problema, pois não existe regra como abaixo:

Sintagma_Oracional_Intransitivo:
Sintagma_Oracional_Intransitivo Locucao_Adverbial_Composta;

Contudo, se o não-terminal foi construído usando a primeira regra, o não-terminal **Locucao_Adverbial_Composta** à direita poderia ter sido usado para construir o não-terminal **Sintagma_Oracional_Intransitivo**. Por essa razão, a verificação de problema de posicionamento também leva em conta os símbolos que compõem os não-terminais à esquerda.

A verificação de problema de posicionamento busca uma regra que poderia ter sido aplicada entre dois não-terminais já construídos. Mas essa verificação possui um detalhe sutil. Ao encontrar uma regra que pode ativar a restrição, é necessário ainda verificar se a regra encontrada atende a suas próprias restrições de atributos com os símbolos que foram reunidos. Isso por causa do entendimento de que a validade de uma regra depende do atendimento às restrições que se aplicam ao seu contexto.

Além disso, a verificação dos atributos nos resguarda de construções da língua em que ambiguidades são totalmente resolvidas através de informações morfológicas.

5.1.4 LIMITAÇÃO DO CONSUMO DE MEMÓRIA

A primeira tentativa de controle dos efeitos da explosão combinatória foi distribuir o processamento das configurações pelo tempo. Este mecanismo não se propõe a diminuir a produção de configurações, mas se propõe a garantir que o consumo de memória seja controlado. A motivação para tal abordagem foi a necessidade de não tornar indisponível a própria máquina de testes, que é um computador pessoal.

Para garantir que a plataforma ocuparia a memória principal com um número limitado de configurações, impusemos um limite de controle do tamanho do conjunto de configurações em memória principal. Este limite é um parâmetro ajustável do sistema, que é aplicado após o processamento de cada vocábulo, ou dito de outra maneira, conjunto de homônimos.

Quando o conjunto de configurações em memória ultrapassa esse limite, permanece na memória principal somente um conjunto de configurações que tem o tamanho do limite ajustado. As demais configurações são armazenadas em memória secundária em conjuntos de tamanho igual ao do limite ou inferior.

Para que mais tarde fosse possível continuar de maneira independente o processamento dos conjuntos em memória secundária, foi necessário o armazenamento de uma cópia da sentença a partir da posição em que o processamento dessas configurações parou. E, assim, este mecanismo possibilita inclusive a distribuição do processamento em diferentes computadores.

Se pensarmos em nossa análise GLR do Português como uma floresta de busca, esse mecanismo caracteriza uma busca híbrida entre a busca em largura e a busca em profundidade. Esse mecanismo realiza busca em largura em conjuntos de configurações e busca em profundidade dentro de cada conjunto.

Conseguimos assim evitar a paginação, sem controlar a explosão combinatória.

5.2 TRATAMENTO DO PROBLEMA DE MANUTENIBILIDADE DA GRAMÁTICA

Na subseção 4.2.1, mencionamos o problema da manutenibilidade da gramática. Pôde ser observado que o problema não se restringe ao número de regras, mas se agrava com as necessidades de legibilidade, de controle de ambiguidades e de expressividade.

Apresentamos nesta seção as abordagens que puderam nos auxiliar a administrar a gramática.

5.2.1 DIRETRIZES

Para que se mantenha um conjunto inteligível e coerente da disposição das definições da gramática é necessário o estabelecimento de diretrizes de escrita. Diretrizes estas cada vez mais difíceis de administrar sem ferramentas automatizadas de verificação.

Procuramos definir algumas regras para a disposição das mais de 920 regras que escrevemos:

a) definições de alto nível (frase e período, por exemplo) devem vir primeiro;

b) as definições que dependem da transitividade devem seguir sempre a mesma ordem (definições intransitivas são sempre seguidas de definições transitivas diretas similares, por exemplo);

c) as definições que podem compor o sujeito devem ficar próximas;

d) em seguida as definições que podem compor o objeto devem ficar próximas;

e) em seguida as definições que podem compor verbos de determinada transitividade devem ficar próximas;

f) em seguida as definições que podem compor as orações subordinadas devem ficar próximas;

g) as definições subordinadas devem procurar seguir as mesmas diretrizes que foram aplicadas às orações principais;

h) por último aparecem as definições relativas às locuções adverbiais.

5.2.2 PADRÕES SINTÁTICOS

Durante o desenvolvimento da gramática pudemos observar que alguns padrões de escrita se repetiam. O mais simples e comum é o padrão de construção de listas. As listas estão presentes na formação de sujeitos ou de objetos compostos.

"Ele quer alegria, saúde e dinheiro."

Para descrever todas as possibilidades de objeto direto, precisamos das seguintes regras:

```
Objetos_Diretos   :  Objeto_Direto
                  |  Lista_Objeto_Direto Conjuncao_Coordenativa Objeto_Direto
                  ;
Lista_Objeto_Direto  :  Objeto_Direto
                     |  Lista_Objeto_Direto Pontuacao Objeto_Direto
                     ;
```

O não-terminal **Objetos_Diretos** é formado por uma lista de **Objeto_Direto**.

"Ele é, está e ficou feliz."

Podemos comparar com as regras de formação do não-terminal **Verbos_Ligacao**

```
Verbos_Ligacao   :  Verbo_Ligacao
                 |  Lista_Verbo_Ligacao Conjuncao_Coordenativa Verbo_Ligacao
                 ;
Lista_Verbo_Ligacao  :  Verbo_Ligacao
                     |  Lista_Verbo_Ligacao Pontuacao Verbo_Ligacao
                     ;
```

A maneira de descrever a lista de não-terminais é idêntica a menos dos nomes. Ou seja, só precisamos do nome do não-terminal que representa a lista e o nome do símbolo que compõe a lista.

A partir dessas informações, desenvolvemos um construtor de padrões sintáticos para a gramática. Seu objetivo é facilitar a escrita da gramática, automatizando parte do trabalho mecânico de descrição. Para que isso seja possível, é criado um arquivo com ordens de construção de regras automáticas, como no exemplo abaixo:

Periodo_Composto := *Lista (Sintagma_Oracional)*

Esse é um exemplo de uso do construtor de listas. O não-terminal **Periodo_Composto** é construído como uma lista de **Sintagma_Oracional**. Essa linha não é entendida pelo BISON. É necessário o processamento do arquivo criado para transformá-lo em uma gramática aceita pelo BISON. O resultado fica:

```
Periodo_Composto : Sintagma_Oracional
                 | Lista_Periodo Conjuncao_Coordenativa Sintagma_Oracional
                 ;
Lista_Periodo    : Sintagma_Oracional
                 | Lista_Periodo Pontuacao Sintagma_Oracional
                 ;
```

Se for necessária alguma inclusão de regras ou de restrições de atributos, deverá ser feita diretamente sobre o resultado do processamento dos padrões sintáticos.

Identificamos ainda um outro padrão, muito mais sutil. Encontramos a motivação para esse novo padrão na frase (1) acima. Nela, observamos que era necessário incluir na gramática a possibilidade de construção de sintagmas cujas estruturas fossem parciais e dependentes de uma estrutura completa congênere que as precede. Basta pensarmos na frase com elipse:

"Eu gosto de estudar, Pedro, de dormir."

Na frase acima temos uma oração ("Pedro, de dormir") que depende da estrutura de uma oração anterior que tem estrutura completa ("Eu gosto de estudar").

Na frase (1), tal fenômeno ocorre no trecho: "garantir a indenização em caso de sinistro, bem como a segurança que o seu nome inspira no mercado". A locução adverbial "em caso de sinistro" completa uma oração subordinada e o segundo objeto direto de "garantir" aparece em uma estrutura dependente.

A inclusão desse tipo de construção trouxe mudanças extensas na gramática, a tal ponto que tivemos de desenvolver mecanismos de construção automática de padrões. O construtor de extensões é extremamente útil por automatizar processos demorados de escrita. Esse padrão sintático depende das regras do não-terminal que está sendo estendido.

Sintagma_Oracional_Intransitivo_Completo
:= *Extensão (Sintagma_Oracional_Intransitivo , Verbo_Intransitivo)*

Sintagma_Oracional_Intransitivo
: *Sintagma_Verbal_Intransitivo*
| *Sujeito Verbo_Intransitivo*
| *Sujeito Verbo_Intransitivo Locucao_Adverbial_Composta*
| *Locucao_Adverbial_Composta Pontuacao Sujeito Verbo_Intransitivo*
| *Verbo_Intransitivo Sujeito*
| *Verbo_Intransitivo Sujeito Locucao_Adverbial_Composta*
| *Locucao_Adverbial_Composta Pontuacao Verbo_Intransitivo Sujeito*
| *Locucao_Adverbial_Composta Verbo_Intransitivo Sujeito*
;

Desejamos obter o não-terminal **Sintagma_Oracional_Intransitivo_Completo** como uma extensão que admite estruturas dependentes sobre a estrutura do não-terminal **Sintagma_Oracional_Intransitivo**. Para isso, selecionamos as regras do não-terminal **Sintagma_Oracional_Intransitivo** que tenham o símbolo **Verbo_Intransitivo**. Esse símbolo é o núcleo sobre o qual a estrutura dependente se apoia. Ele será a base de construção das regras dependentes. O resultado fica como abaixo:

```
Sintagma_Oracional_Intransitivo_Completo
    :   Lista_Sintagma_Oracional_Intransitivo
    |   Lista_Sintagma_Oracional_Intransitivo Conjuncao_Coordenativa Sujeito Pontuacao Locucao_Adverbial_Composta
    |   Lista_Sintagma_Oracional_Intransitivo Conjuncao_Coordenativa Locucao_Adverbial_Composta Pontuacao Sujeito
    |   Lista_Sintagma_Oracional_Intransitivo Conjuncao_Coordenativa Sujeito Locucao_Adverbial_Composta
    |   Lista_Sintagma_Oracional_Intransitivo Conjuncao_Coordenativa Locucao_Adverbial_Composta Pontuacao Pontuacao Sujeito
    |   Lista_Sintagma_Oracional_Intransitivo Conjuncao_Coordenativa Locucao_Adverbial_Composta Pontuacao Sujeito
    ;
Lista_Sintagma_Oracional_Intransitivo
    :   Sintagma_Oracional_Intransitivo
    |   Lista_Sintagma_Oracional_Intransitivo Pontuacao Sujeito Pontuacao Locucao_Adverbial_Composta
    |   Lista_Sintagma_Oracional_Intransitivo Pontuacao Locucao_Adverbial_Composta Pontuacao Sujeito
    |   Lista_Sintagma_Oracional_Intransitivo Pontuacao Sujeito Locucao_Adverbial_Composta
    |   Lista_Sintagma_Oracional_Intransitivo Pontuacao Locucao_Adverbial_Composta Pontuacao Pontuacao Sujeito
    |   Lista_Sintagma_Oracional_Intransitivo Pontuacao Locucao_Adverbial_Composta Pontuacao Sujeito
    ;
Sintagma_Oracional_Intransitivo
    :   Sintagma_Verbal_Intransitivo
    |   Sujeito Verbo_Intransitivo
    |   Sujeito Verbo_Intransitivo Locucao_Adverbial_Composta
    |   Locucao_Adverbial_Composta Pontuacao Sujeito Verbo_Intransitivo
    |   Verbo_Intransitivo Sujeito
    |   Verbo_Intransitivo Sujeito Locucao_Adverbial_Composta
    |   Locucao_Adverbial_Composta Pontuacao Verbo_Intransitivo Sujeito
    |   Locucao_Adverbial_Composta Pontuacao Verbo_Intransitivo Sujeito Hiato
    |   Locucao_Adverbial_Composta Verbo_Intransitivo Sujeito
    ;
```

As regras são criadas retirando-se a ocorrência do símbolo base e substituindo-se pelo terminal **Pontuacao**. No entanto, as regras que serão usadas devem ter mais de 2 símbolos para serem consideradas. Além disso, a regra resultante não pode ter o terminal **Pontuacao** na última posição da regra criada. É interessante observar também que o padrão prevê a existência de uma lista de estruturas dependentes.

6 A FERRAMENTA DE ANÁLISE SINTÁTICA

Neste breve capítulo fazemos uma descrição de alto nível da ferramenta de análise sintática, uma vez que uma descrição detalhada só interessa a quem for continuar o trabalho. Com o propósito de auxiliar a continuidade da plataforma, estão disponíveis os anexos.

Duas sequências de atividades são necessárias para o funcionamento do analisador sintático: a sequência de geração do analisador sintático, ilustrada pela figura 6.1, e a sequência de funcionamento, ilustrada pela figura 6.2.

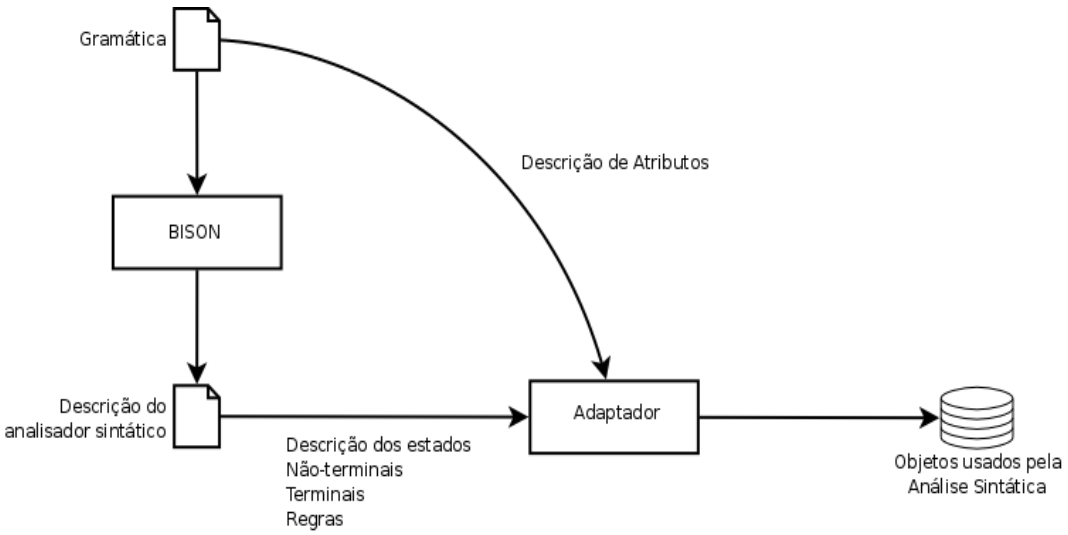

FIG. 6.1: Geração do analisador sintático

O sistema BISON produz a descrição textual do analisador sintático e produz também o código de um analisador sintático para a gramática fornecida. Nós aproveitamos somente a descrição textual, não usando o código que foi gerado. A razão disso é que implementamos nosso próprio analisador sintático.

As descrições codificadas como comentários na gramática são descartadas pelo BISON. Essas descrições são processadas pelo **Adaptador**, um dos projetos da plataforma[9]. Ele é responsável por organizar e codificar as informações textuais em código, produzindo os objetos necessários ao funcionamento do analisador sintático. Esses objetos são responsáveis por definir as alternativas de análise e quais restrições de contexto serão aplicadas.

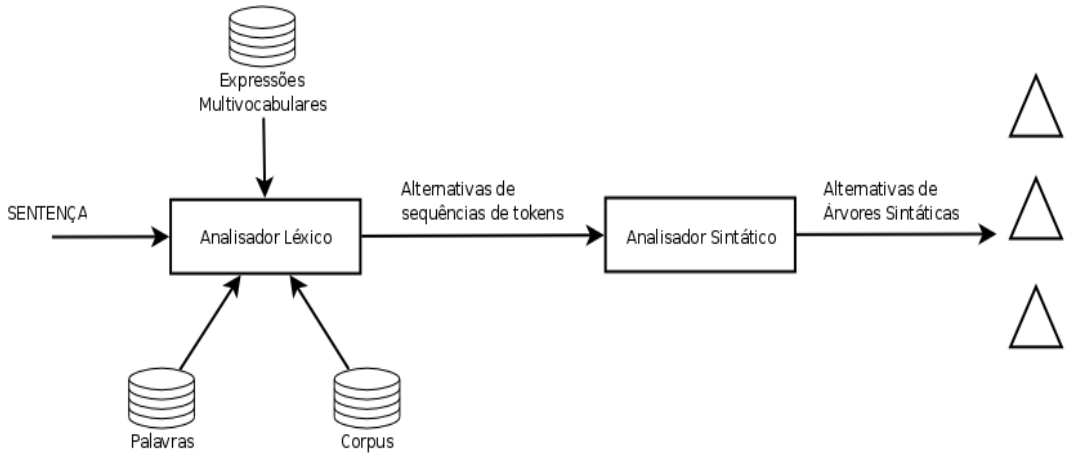

FIG. 6.2: Funcionamento do analisador sintático

O **Analisador Léxico** recebe uma sentença informada pelo usuário da plataforma. Para essa sentença de entrada, é verificada a presença de expressões multivocabulares conhecidas. Cada expressão presente produz uma nova sequência de *tokens*. A identificação dos *tokens* é feita a partir dos léxicos de palavras e do léxico de *corpora* anotados. Exemplificamos abaixo esse mecanismo:

O Plano Real evolui.

Como transformamos todas as letras em minúsculas, a sentença se torna:

o plano real evolui.

Entre as expressões multivocabulares conhecidas está a expressão "Plano Real", que, convertida para minúsculas, fica "plano real". Essa expressão tem valor de nome próprio masculino singular. Podemos observar que a expressão aparece efetivamente no texto acima. E, dessa maneira, passamos a ter duas sequências de *tokens* possíveis:

[9]Para conhecer todos os projetos da plataforma, ver anexos.

a) *o plano real evolui.*

b) *o [plano real] evolui.*

A primeira sequência possui 4 *tokens*, enquanto a segunda possui 3 *tokens* somente.

O analisador sintático trata independentemente cada sequência de *tokens*. Durante a análise de cada sequência de *tokens*, cria uma nova configuração de análise sintática para cada alternativa de ação ou de transição disponíveis em cada estado. O resultado é um conjunto de árvores sintáticas para cada sequência de *tokens* analisada. No exemplo acima, a primeira sequência produz duas árvores sintáticas (figuras 6.3 e 6.4) e a segunda produz uma única árvore sintática (figura 6.5).

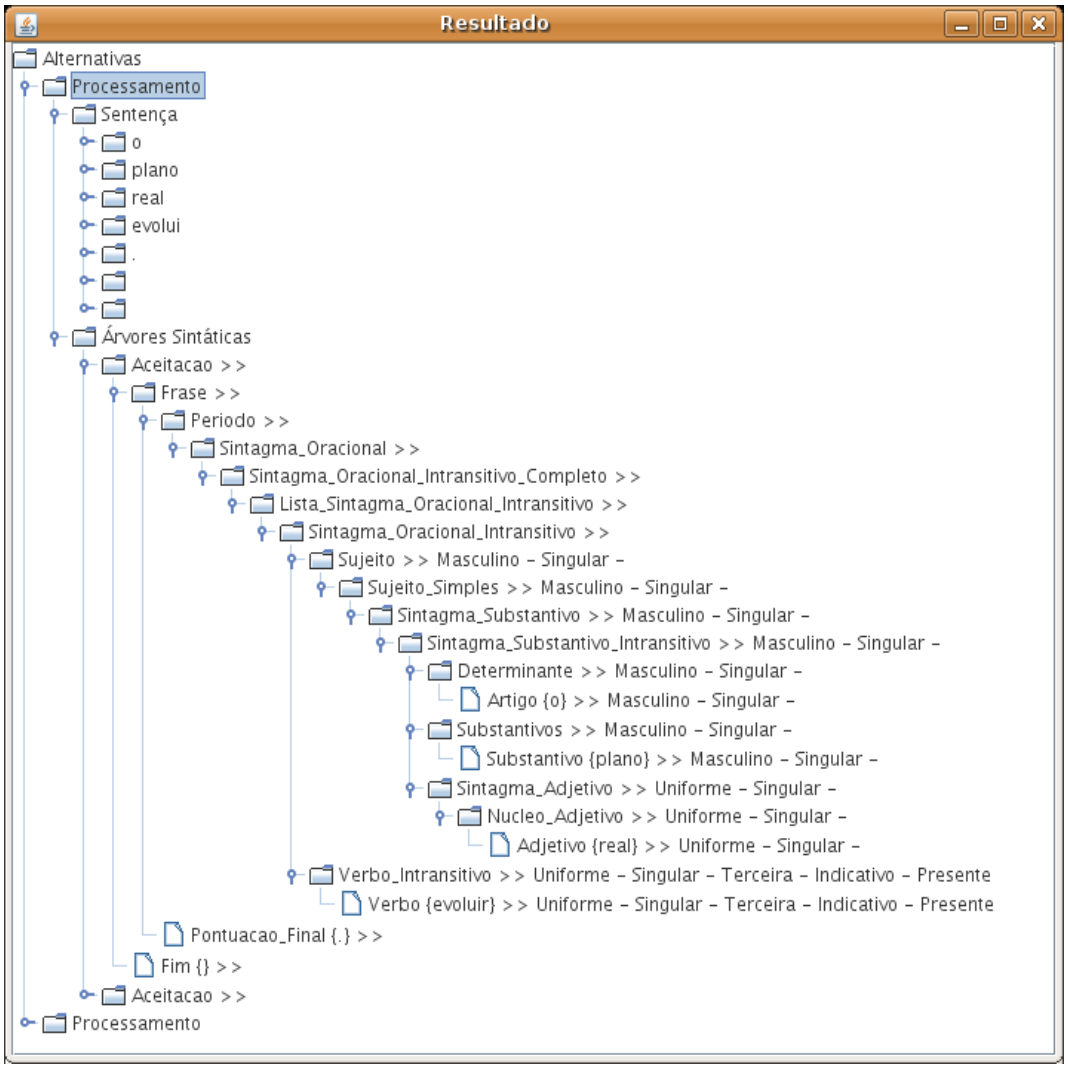

FIG. 6.3: Primeira árvore sintática da primeira sequência de *tokens*.

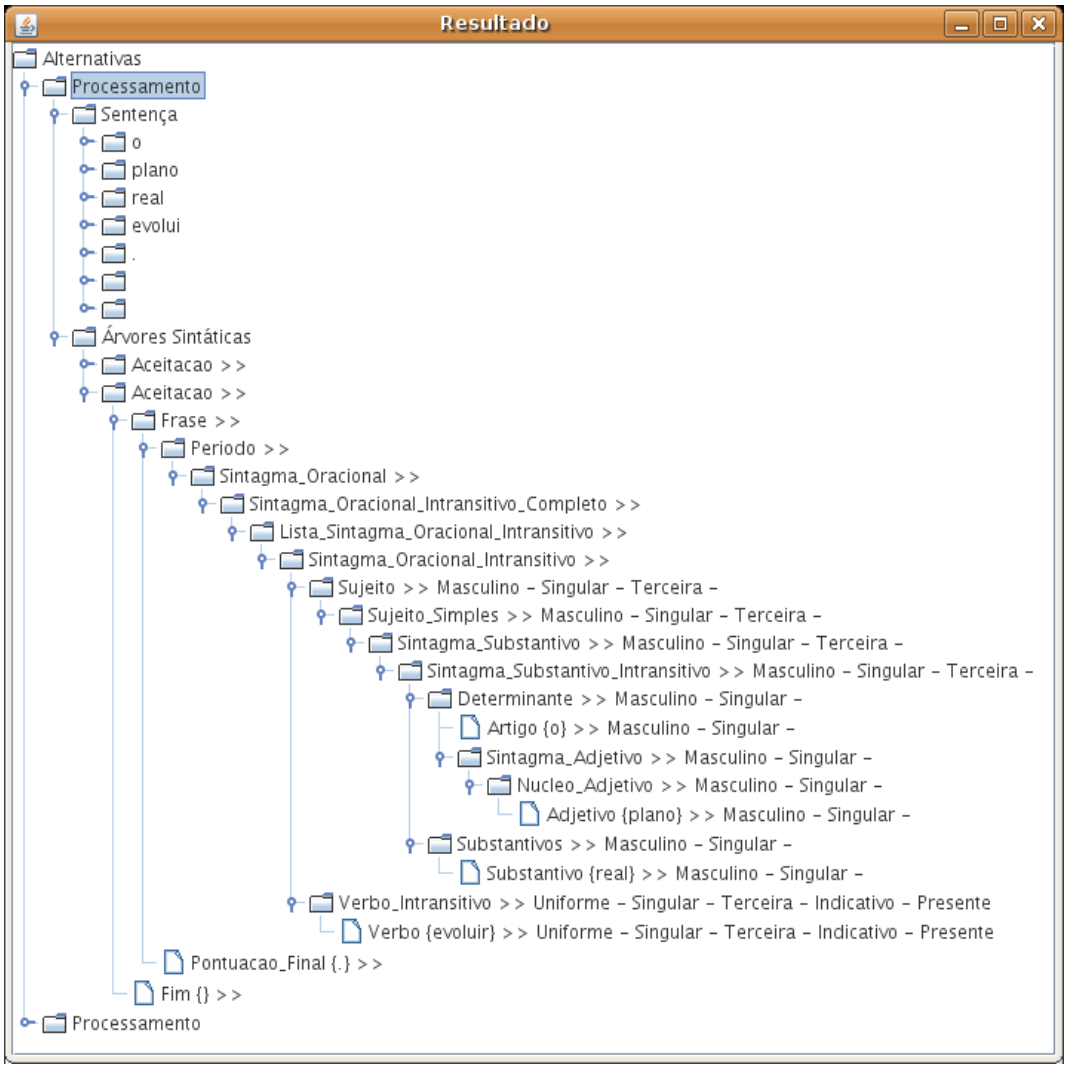

FIG. 6.4: Segunda árvore sintática da primeira sequência de *tokens*.

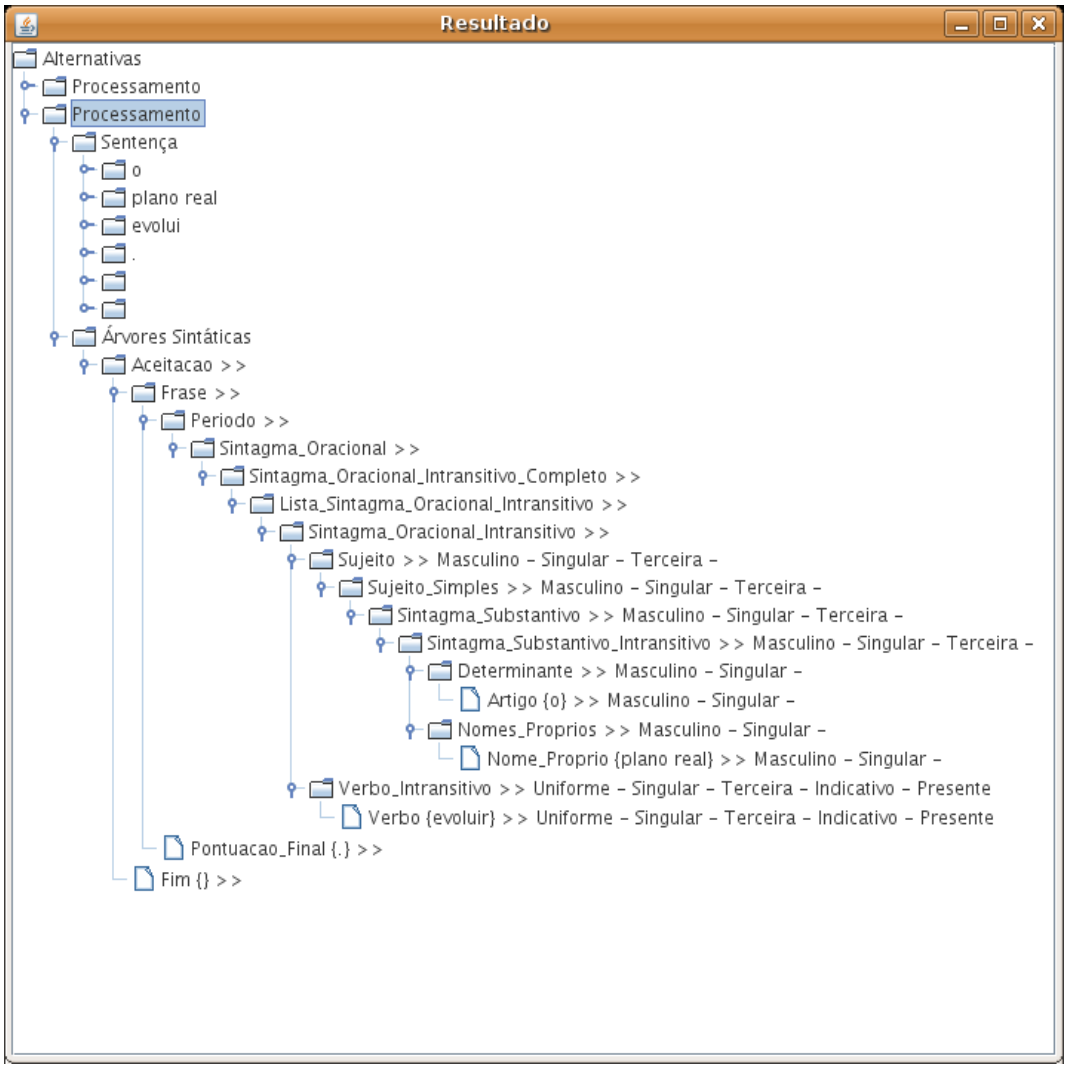

FIG. 6.5: Única árvore sintática da segunda sequência de *tokens*.

7 RESULTADOS

Neste capítulo descrevemos os testes calibração e os testes de avaliação da plataforma enquanto ferramenta para processamento de Linguagem Natural. Falaremos sobre os testes de calibração na seção 7.1 e sobre os testes de avaliação na seção 7.2.

7.1 TESTES DE CALIBRAÇÃO

Para a realização dos testes de calibração, tivemos de realizar testes anteriores de adequações de regras, de restrições e de mecanismos, até o ponto de considerarmos a gramática e a plataforma prontas em estágio bruto, em que suas bases estavam estáveis o suficiente para que um refinamento maior pudesse ser buscado. Além disso, precisávamos definir que tipos de frase seriam submetidas aos testes de avaliação.

Para a realização dos testes de calibração, selecionamos 100 frases em sequência de uma revista Exame (CAMARGO, 1996). Incluímos uma restrição para o tamanho das frases. Cada frase poderia ter no máximo 12 palavras, contada cada contração como 2 palavras e cada expressão conhecida como 1 palavra. A lista das frases está no anexo 10.7.1

Para esse conjunto de frases, obtemos os resultados apresentados na tabela 7.1:

Frase	Resultado	Árvores	Frase	Resultado	Árvores	Frase	Resultado	Árvores
1	falhou	0	36	aceitou	8	71	falhou	0
2	falhou	0	37	aceitou	2	72	aceitou	84
3	falhou	0	38	falhou	0	73	aceitou	42
4	falhou	0	39	aceitou	6	74	falhou	0
5	falhou	0	40	aceitou	9	75	falhou	0
6	falhou	0	41	falhou	0	76	falhou	0
7	aceitou	9	42	falhou	0	77	falhou	0
8	falhou	0	43	aceitou	6	78	falhou	0
9	falhou	0	44	falhou	0	79	falhou	0
10	aceitou	10	45	falhou	0	80	aceitou	6
11	aceitou	16	46	aceitou	7	81	falhou	0
12	aceitou	3	47	aceitou	14	82	falhou	0
13	falhou	0	48	aceitou	4	83	falhou	0
14	falhou	0	49	falhou	0	84	falhou	0
15	falhou	0	50	falhou	0	85	falhou	0
16	falhou	0	51	falhou	0	86	falhou	0
17	falhou	0	52	falhou	0	87	falhou	0
18	falhou	0	53	falhou	0	88	falhou	0
19	falhou	0	54	falhou	0	89	falhou	0
20	falhou	0	55	aceitou	2	90	falhou	0
21	falhou	0	56	falhou	0	91	falhou	0
22	falhou	0	57	falhou	0	92	falhou	0
23	falhou	0	58	falhou	0	93	falhou	0
24	falhou	0	59	aceitou	1	94	aceitou	8
25	falhou	0	60	falhou	0	95	falhou	0
26	falhou	0	61	aceitou	5	96	falhou	0
27	falhou	0	62	aceitou	2	97	falhou	0
28	falhou	0	63	falhou	0	98	falhou	0
29	aceitou	4	64	falhou	0	99	falhou	0
30	falhou	0	65	falhou	0	100	falhou	0
31	falhou	0	66	falhou	0			
32	falhou	0	67	falhou	0			
33	falhou	0	68	falhou	0			
34	aceitou	16	69	falhou	0			
35	falhou	0	70	falhou	0			

TAB. 7.1: Tabela de resultados da calibração

Após os testes de calibração, realizamos ainda alguns ajustes na plataforma.

7.2 TESTES DE AVALIAÇÃO

A avaliação ocorreu sobre um conjunto de 50 frases de uma revista Época (MANSUR, 2008). A lista das frases está no anexo 10.7.2. Os critérios definidos a partir dos testes de calibração para a seleção das frases dos testes de avaliação foram:

a) As frases foram buscadas de maneira sequencial a partir de um ponto aleatório do meio da revista;

b) As frases poderiam ter no máximo 10 palavras. As contrações contam como duas palavras e as expressões como uma palavra só;

c) As frases deveriam ter oração principal;

d) As frases não poderiam ter vírgula;

e) As frases não poderiam ter características léxicas não tratadas, tais como números ou aspas;

f) As frases só poderiam ter um verbo;

g) As frases não poderiam estar na voz passiva;

h) As frases não poderiam ter verbos pronominais[10].

Acreditamos que todas as restrições acima podem ser removidas pelo refinamento da gramática. A exceção, contudo, é a regra que restringe o tamanho da sentença. Para que ela seja removida é necessária a alteração do algoritmo de realização da análise sintática. E, para isso, mudanças no código devem ser realizadas.

Para estabelecermos nosso padrão de comparação, aplicamos o mesmo conjunto de frases ao sistema PALAVRAS. Apresentamos, na tabela 7.2, os resultados dos testes de avaliação.

[10] As estruturas sintáticas que envolvem verbos pronominais ainda não foram descritas na gramática.

	Plataforma de Análise Sintática		Sistema PALAVRAS	
Frase	**Árvores**	**Corretas**	**Árvores**	**Corretas**
1	2	2	1	1
1	1	1	1	1
2	0	0	1	1
3	6	1	1	1
4	3	1	1	1
5	2	2	1	1
6	2	1	1	1
7	6	4	1	1
8	6	0	1	1
9	3	0	1	0
10	1	1	1	1
11	1	1	1	1
12	14	2	1	0
13	0	0	1	1
14	3	1	1	1
15	1	0	1	1
16	13	1	1	1
17	4	2	1	0
18	2	1	1	1
19	11	1	1	1
20	0	0	1	1
21	3	1	1	1
22	4	1	1	1
23	1	1	1	1
24	0	0	1	1
25	2	1	1	1
26	4	1	1	1
27	1	1	1	1
28	2	1	1	1
29	0	0	1	1
30	26	1	1	1
31	0	0	1	1
32	1	1	1	1
33	5	2	1	1
34	3	0	1	1
35	5	2	1	1
36	2	1	1	0
37	2	1	1	1
38	3	1	1	1
39	0	0	1	1
40	3	1	1	1
41	16	0	1	0
42	1	1	1	1
43	4	2	1	1
44	4	0	1	1
45	1	1	1	1
46	1	1	1	1
47	3	1	1	1
48	2	2	1	1
49	3	1	1	1
50	2	1	1	1

TAB. 7.2: Tabela de resultados de avaliação

A determinação das árvores que estavam corretas foi feita através da observação das subestruturas de cada árvore. A presença de alguma estrutura muito artificial foi rejeitada.

Por exemplo, para a frase abaixo, nossa plataforma produz, dentre outras, a árvore que aparece na figura 7.1:

O apocalipse não vai acontecer agora.

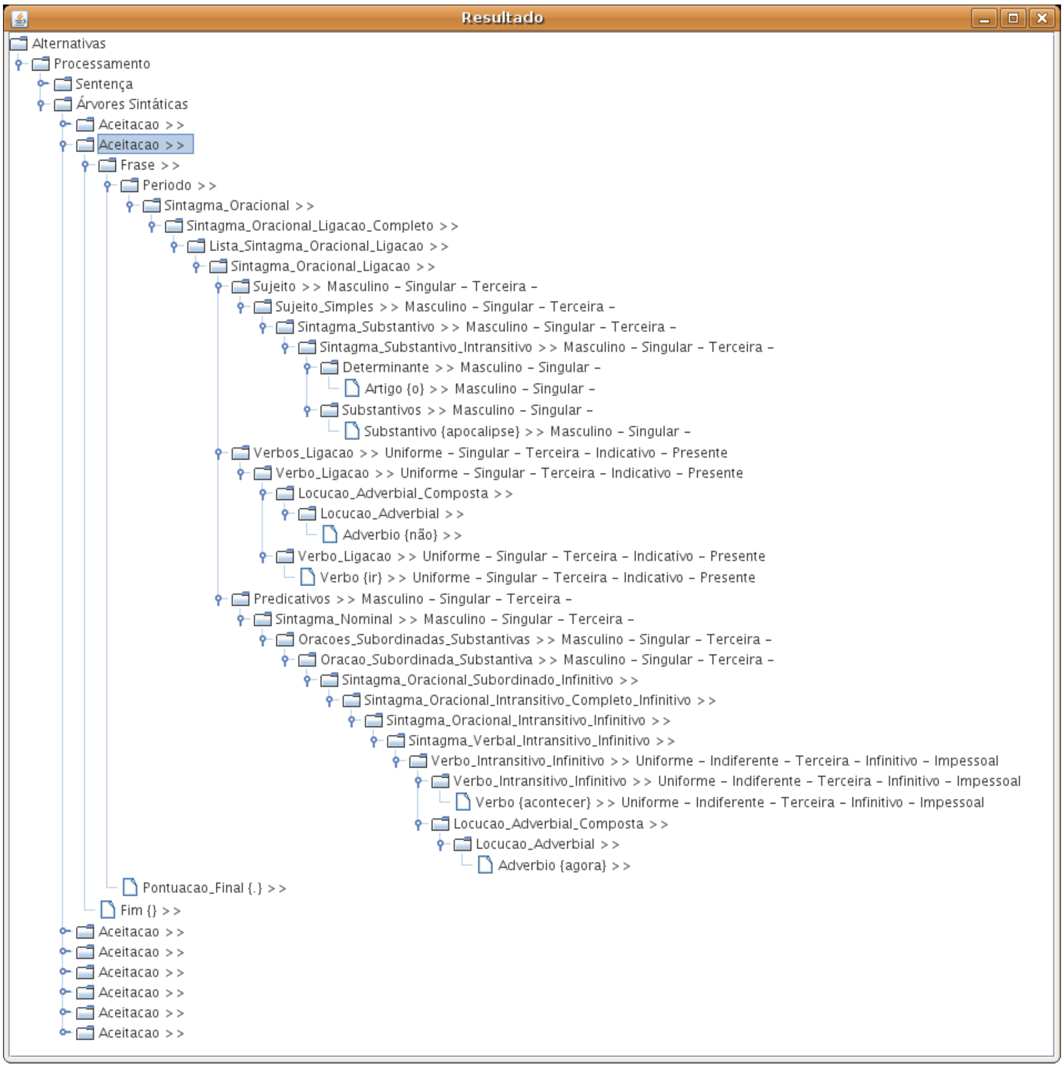

FIG. 7.1: Exemplo de árvore rejeitada

Sabemos que o verbo "ir" pode ser verbo de ligação e que uma oração subordinada de infinitivo cumpre uma função nominal. No entanto, para esta frase, não faz sentido admitir essa combinação de estruturas.

7.3 ANÁLISE DOS RESULTADOS

A partir da tabela 7.2 podemos observar que muito trabalho ainda deve ser feito para que a plataforma comece a apresentar resultados próximos ao sistema PALAVRAS. Apesar disso, podemos observar alguns fenômenos que dizem respeito às modelagens de nossa plataforma de análise sintática e do sistema PALAVRAS.

O primeiro fenômeno é comum aos dois sistemas: a presença de alguma árvore não é sinônimo de que alguma análise esteja correta. Podemos ver que os dois sistemas, para determinadas frases, apresentaram uma ou mais árvores sem que alguma delas fosse correta. Na nossa plataforma isso é o resultado de alguma análise que produziu uma árvore sintática inaceitável. No caso do sistema PALAVRAS, isso é uma opção de projeto, pois o formalismo das *constraint grammars* exige que não se descarte a última alternativa de análise. Isso faz com que o sistema PALAVRAS sempre apresente algum resultado, ainda que tenha identificado alguma restrição que descartaria a árvore sintática apresentada.

Outro fenômeno diz respeito à expressividade das árvores geradas. O sistema PALAVRAS tende a produzir árvores com poucos níveis. Isso simplifica a estrutura, mas pode esconder informações de relacionamento entre as subestruturas. As árvores que nossa plataforma produz, por sua vez, tendem a produzir muitos níveis. Por exemplo, para a frase:

O comércio também é intenso.

as árvores geradas por nossa plataforma e pelo sistema PALAVRAS aparecem respectivamente nas figuras 7.2 e 7.3.

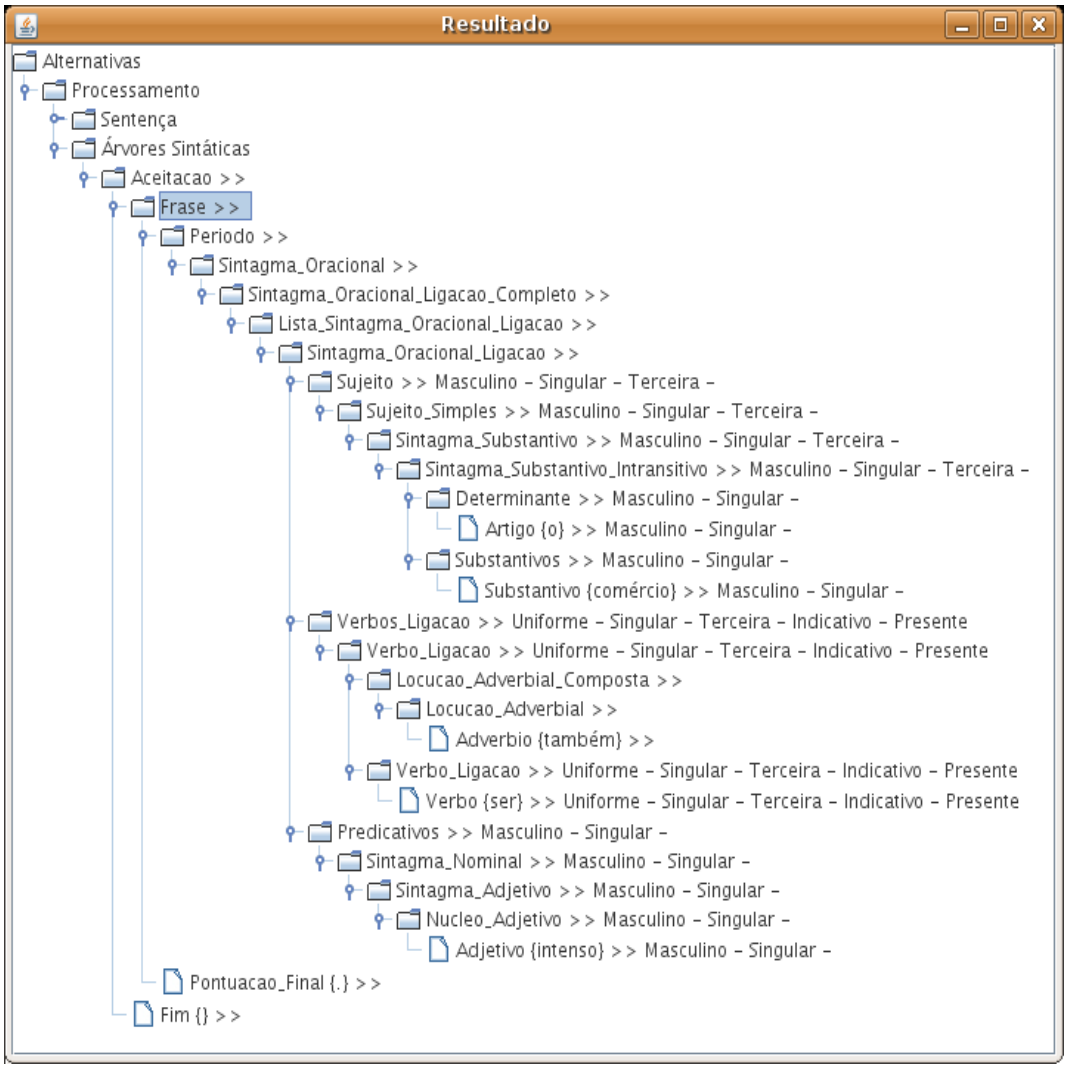

FIG. 7.2: Árvore sintática em nossa plataforma

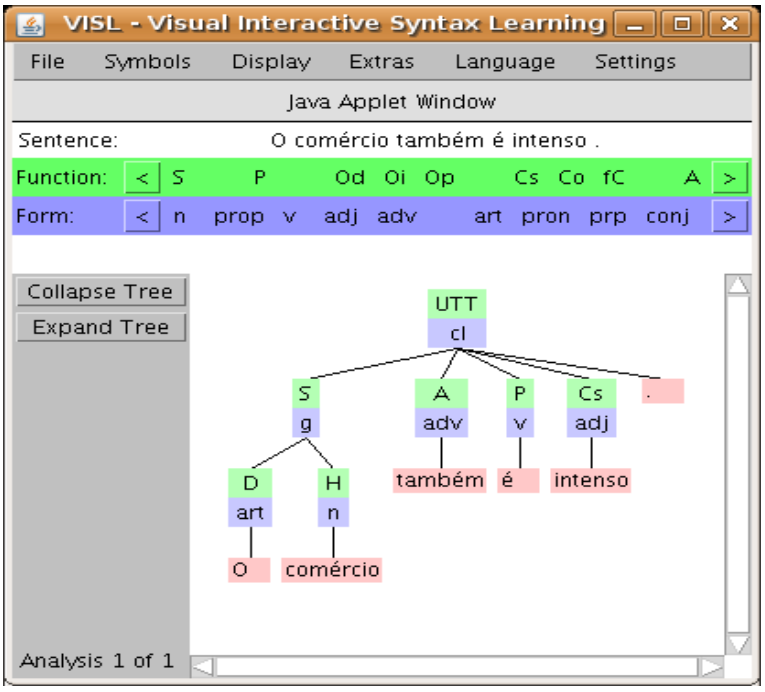

FIG. 7.3: Árvore sintática no sistema PALAVRAS

Podemos observar que o advérbio "também" altera o verbo "é". Na árvore que produzimos, essa relação fica clara pela construção de um não-terminal a partir do verbo e do advérbio. Na árvore produzida pelo sistema PALAVRAS, essa relação não fica tão clara. Ou seja, pode existir algum proveito para análise semântica em uma árvore com muitos níveis.

Outro aspecto que diz respeito às opções de projeto é o fato de o sistema PALAVRAS somente apresentar a árvore que ele entende ser a mais provável. Isso pode significar o desprezo de alternativas viáveis de análise sintática.

Os algoritmos e heurísticas do sistema PALAVRAS são muito eficientes para a determinação de uma única análise sintática. Para que isso seja possível, ele realiza uma análise léxica muito bem feita. Se comparado à nossa plataforma, ele consegue diferenciar, por exemplo, quando uma palavra tem maiúsculas. Para as frases abaixo, o sistema PALAVRAS produz respectivamente as árvores nas figuras 7.4 e 7.5.

O Plano Real evolui.

O plano real evolui.

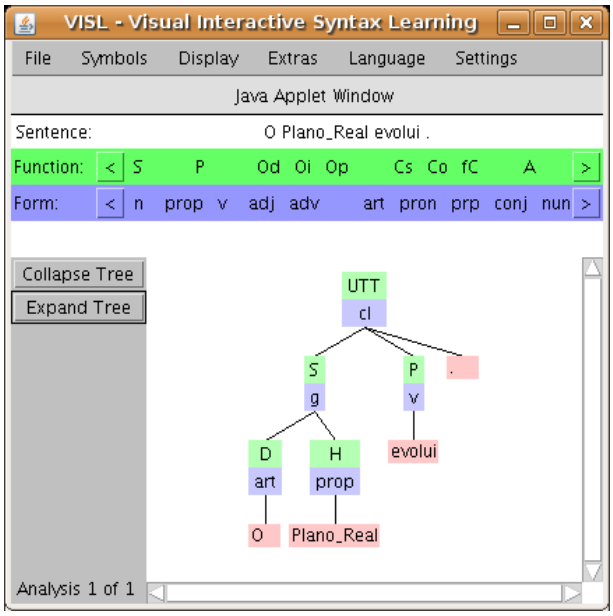

FIG. 7.4: Árvore sintática no sistema PALAVRAS

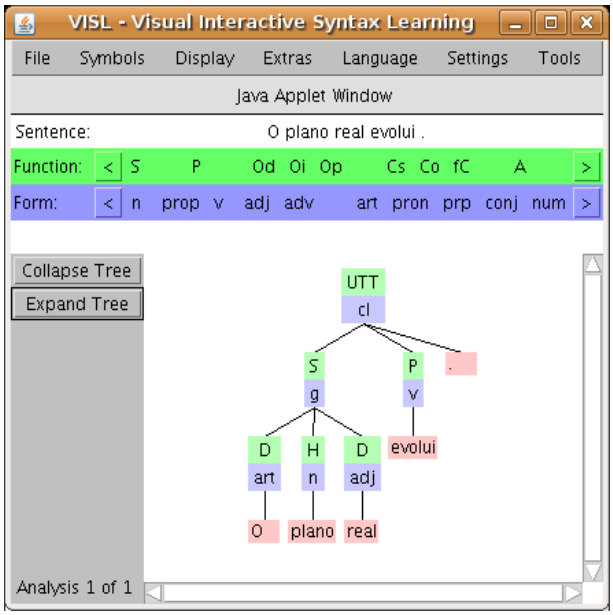

FIG. 7.5: Árvore sintática no sistema PALAVRAS

Por outro lado, o emprego desses mecanismos para a obtenção de uma única alternativa de análise podem provocar a escolha de uma alternativa mais artificial em detrimento de uma alternativa mais aceitável que tenha passado pelo mesmo número de restrições, mas que tenha ficado com prioridade ligeiramente menor.

Por exemplo, para a frase abaixo, o sistema PALAVRAS produz a árvore que está na figura 7.6

O gordo pensativo chegou.

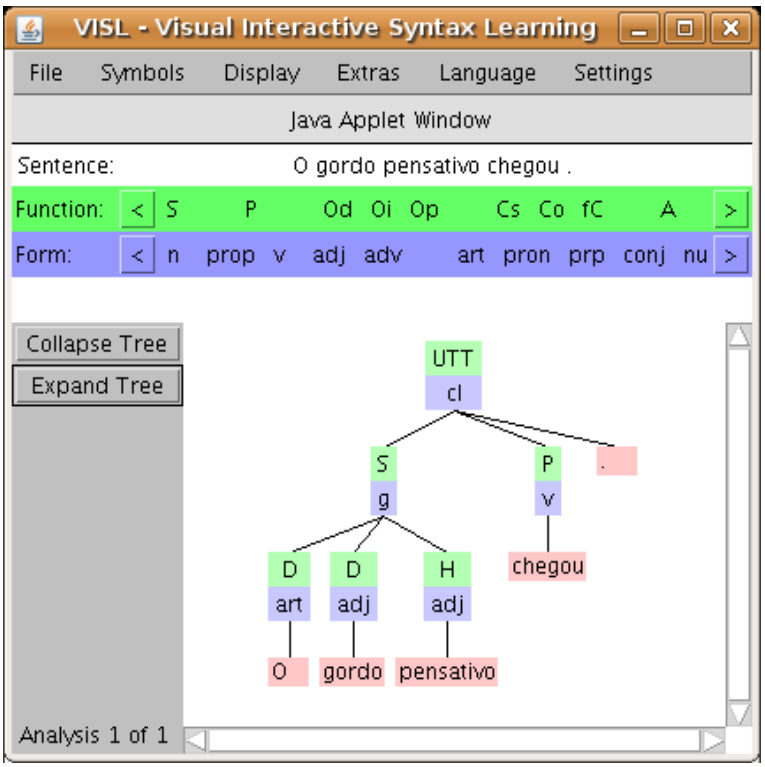

FIG. 7.6: Árvore sintática no sistema PALAVRAS

Podemos observar que o núcleo[11] do sintagma nominal foi escolhido para ser o adjetivo "pensativo", enquanto a escolha mais natural era atribuir o núcleo à derivação imprópria formada pelo artigo e pelo adjetivo "gordo". Como o sistema PALAVRAS foi desenvolvido para o uso sobre texto irrestrito, corrigir tais distorções para poder lidar com todas as situações da língua natural exige um esforço constante sobre os algoritmos de análise.

Em nossa plataforma, o substantivo "gordo" é utilizado para construir o núcleo do sujeito da frase. Essa frase produz a árvore que está na figura 7.7.

[11] A definição do termo principal de um sintagma é indicado pela letra "H", letra inicial de *head* em inglês.

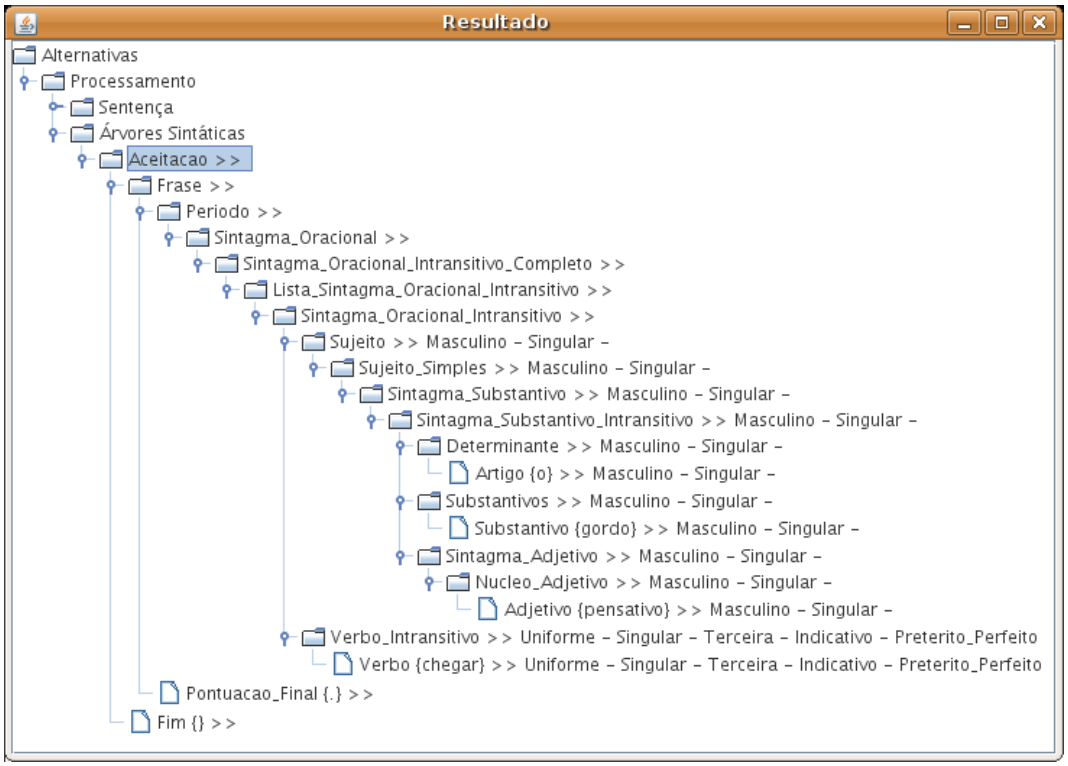

FIG. 7.7: Árvore sintática em nossa plataforma

Para que possamos melhorar o desempenho de nossa plataforma, deveremos ser capazes de reduzir as múltiplas árvores que são produzidas, mas que não contribuem com informação semântica nova. Sabemos que muitas informações semânticas podem ser obtidas a partir das funções sintáticas dos sintagmas na frase. Se duas árvores apresentam relacionamentos e dependências muito semelhantes entre sintagmas a informações semânticas tendem a ser muito similares. Por exemplo, para a frase abaixo, nossa plataforma produziu árvores como na figura 7.8.

Não foi uma nacionalização dos bancos.

FIG. 7.8: Árvore sintática em nossa plataforma

Nossa plataforma criou duas alternativas de posicionamento da locução adverbial. Uma alternativa em que o advérbio "não" altera o verbo e outra em que o mesmo advérbio altera a oração inteira. No caso do Português, esta diferenciação nem sempre é significativa. Logo, pode ser desejável não produzir essa combinação de resultados, pois podemos eliminar metade do número de árvores produzidas.

Esse fenômeno acontece, pois definimos na gramática duas posições para a locução adverbial. Ela pode estar contígua a um verbo ou nas extremidades de uma oração. No caso de uma oração apresentar verbos na extremidade, existe uma sobreposição de posições habilitadas para a locução adverbial na frase[12]. Podemos, portanto, implementar a correção dessa sobreposição através do refinamento da gramática, deixando somente as regras que definem a posição da locução adverbial em extremidades de orações, se o não terminal contíguo não for um verbo. Devemos eliminar, por exemplo, a regra:

```
Sintagma_Oracional_Intransitivo   :   Locucao_Adverbial_Composta Verbo_Intransitivo Sujeito
                                  ;
```

e deixar, por exemplo

```
Sintagma_Oracional_Intransitivo   :   Verbo_Intransitivo Sujeito
                                  |   Locucao_Adverbial_Composta Pontuacao Verbo_Intransitivo Sujeito
                                  ;
```

Uma outra maneira de controlar o número de árvores redundantes é a utilização de mais restrições, através da gramática de atributos. Para a frase abaixo, nossa plataforma constrói duas alternativas de árvores como na figura 7.9.

A técnica ainda tem limitações.

[12]Para a representação em árvore, não existe sobreposição, mas, para a representação linear das palavras na frase, ocorre a sobreposição, pois a locução adverbial pode ser construída a partir de um único advérbio.

FIG. 7.9: Árvore sintática em nossa plataforma

Podemos observar que a construção da derivação imprópria (DENICOLA, 1993) não faz sentido quando existe um substantivo homônimo ao adjetivo. Para corrigirmos isso, podemos descrever uma restrição que se aplica à regra que define a derivação imprópria. A restrição deverá exigir que não exista substantivo homônimo para o adjetivo da regra.

Por outro lado, o problema que salta aos olhos a partir da tabela 7.2 é a inexistência de árvores para muitas frases. Por exemplo, a frase número 2 não produziu qualquer árvore.

Dois exemplos recentes seguiram essa linha de atuação.

Mas, ao incluirmos a regra abaixo, obtivemos a árvore que aparece na figura 7.10.

Determinante : Pronome_Demonstrativo;

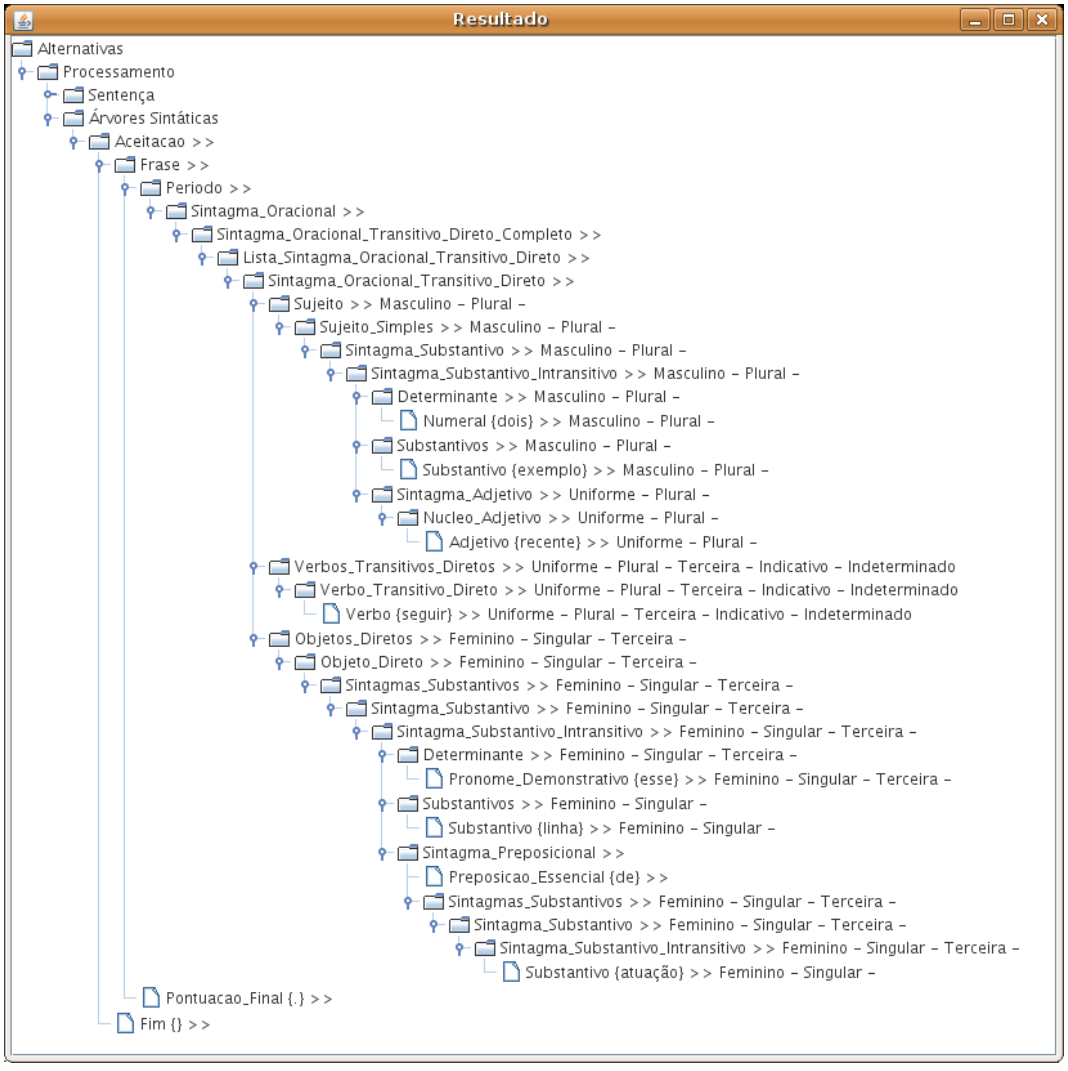

FIG. 7.10: Árvore sintática em nossa plataforma

Nossa plataforma foi desenvolvida para admitir constantes refinamentos da gramática, sem que qualquer alteração em algoritmos precise ser feita. Além disso, ela não é dependente da língua portuguesa, podendo inclusive ser empregada em diferentes línguas.

Em suma, apesar da clara superioridade do sistema PALAVRAS, acreditamos que podemos atingir uma qualidade comparável, aperfeiçoando-se a gramática e as restrições. Quanto à qualidade das árvores, acreditamos que aquelas produzidas por nossa plataforma explicitam relacionamentos entre os componentes sintáticos que podem torná-las mais ricas para uma futuro processamento semântico.

8 CONSIDERAÇÕES FINAIS

Neste trabalho foi desenvolvida uma plataforma para análise sintática. Ela foi aplicada para o desenvolvimento de um analisador sintático para o Português escrito.

Fizemos uso dos formalismos de análise sintática que o estudo de linguagens de programação e compiladores nos oferecem. Desenvolvemos a plataforma para o tratamento de fenômenos livres de contexto, por essa razão, somente utilizamos as gramáticas livres de contexto. Empregamos, em especial, a infraestrutura de análise do método $LR(1)$ (AHO, 1986) para a análise sintática e descrevemos em Gramáticas de Atributos (AHO, 1973a; RANGEL, 2000) regras semânticas para a especificação de restrições.

Em primeiro lugar implementamos uma solução que, a partir de uma gramática livre de contexto, pudesse tratar todas as alternativas de ambiguidades léxicas e sintáticas.

Ao realizarmos testes durante o desenvolvimento da plataforma, encontramos problemas, tais como, a explosão combinatória e a manutenibilidade da gramática. Esses problemas foram apresentados no capítulo 4.

Posteriormente atacamos os problemas criando restrições com base em Gramáticas de Atributos, verificações estruturais, limitação do consumo de memória e padrões sintáticos. Essas soluções foram apresentadas no capítulo 5.

Portanto, acreditamos que o objetivo de desenvolver uma plataforma de análise sintática e aplicá-la ao Português foi satisfatoriamente atingido.

Dessa maneira, a partir de nossa plataforma de análise sintática, podemos pensar no desenvolvimento de diversas aplicações para a língua portuguesa, em especial aplicações de Recuperação de Informações, contribuindo assim para a pesquisa em Linguística Computacional.

Dentre as contribuições do presente trabalho ao assunto, destacamos o desenvolvimento de mecanismos de representação estrutural, permitindo a implementação de métodos de verificação estrutural durante a própria análise sintática.

Do ponto de vista computacional, o analisador sintático que lida com ambiguidades léxicas e sintáticas é o núcleo do funcionamento de nossa plataforma. Além dele, ressaltamos o mecanismo de controle do uso de memória, que nos permitiu evitar a paginação de memória e abriu caminho para o uso da plataforma em redes de computação distribuída.

Do ponto de vista linguístico, a gramática livre de contexto que escrevemos é uma proposta inicial abrangente para a descrição da Sintaxe do Português.

Do ponto de vista da interação entre Ciência da Computação e Linguística, a descrição em Gramática de Atributos de fenômenos linguísticos, como concordância e regência, com ganhos reais de desempenho, foi uma demonstração clara da contribuição mútua para o Processamento de Linguagens Naturais.

Conforme observado no capítulo anterior, nossa plataforma foi desenvolvida para admitir constantes refinamentos da gramática, sem que qualquer alteração em algoritmos precise ser feita. Além disso, ela não é dependente da língua portuguesa, podendo inclusive ser empregada em diferentes línguas.

8.1 TRABALHOS FUTUROS

Como podemos observar do resultado dos testes, existe ainda muito trabalho a ser desenvolvido para que nossa plataforma apresente desempenho comparável ao do sistema PALAVRAS. No entanto, a plataforma foi desenvolvida para admitir o refinamento de sua gramática, sem que seja necessária alteração manual de código. Para que a gramática livre de contexto seja aprimorada, é necessário o trabalho conjunto de Informática e Linguística.

Sabemos que ainda persistem regras que produzem ambiguidades na gramática, por exemplo:

Locucao_Adverbial : Locucao_Adverbial Locucao_Adverbial;

Com relação à manutenção da gramática, existe trabalho a ser feito no desenvolvimento de diretrizes de organização e no desenvolvimento de processos de automação de controle de diretrizes, de verificação de ambiguidades, de relatórios, de análise de impacto de regras sobre a gramática e, principalmente, de ferramentas de escritas de regras e de atributos.

Com relação às próprias regras, existem muitos fenômenos da língua portuguesa que ainda não foram tratados, tais como, por exemplo: Verbos Bitransitivos Indiretos, Verbos Predicativos Circunstanciais, Verbos Pronominais, Verbos Transobjetivos, Preposição Elíptica e Verbo Impessoal.

Para aprimorar a análise léxica, existe espaço para o desenvolvimento de um analisador léxico que leve em consideração regras de formação de palavras. Ainda para a análise léxica, há que se aprimorar os léxicos e as listas de descrições morfológicas.

Com relação à geração de analisadores sintáticos, hoje dependente da ferramenta BISON, pode-se pensar no desenvolvimento de um gerador próprio. Ainda para a análise sintática, uma outra abordagem, que pode ser experimentada, é o aproveitamento de ideias das *constraint grammar*, em especial a definição de restrições explicitamente descritas. O objetivo claro é diminuir a explosão combinatória e permitir que a plataforma sempre possa produzir alguma análise sobre as sentenças.

Contudo, o trabalho mais interessante fica por conta de tornar nosso autômato de análise sintática GLR comparável em complexidade ao algoritmo de Tomita (TOMITA, 1987), que tem complexidade polinomial. Atualmente nosso algoritmo tem complexidade exponencial. Por essa razão, para sentenças grandes, seu tempo de resposta é muito grande.

Pudemos observar algumas fontes de retrabalho em nosso algoritmo, que devem sofrer otimizações:

- *Ele assistiu a menina de rua.* (“a” como artigo)
- *Ele assistiu a menina de rua.* (“a” como preposição)

Nos dois casos, nosso algoritmo de análise chegará à conclusão de que o trecho “Ele” é um sujeito e que “de rua” é um sintagma preposicional. No entanto, ele teve de fazer a análise duas vezes para cada um desses trechos, não compartilhando a análise entre as diferentes configurações.

A estratégia de análise sintática, por sua vez, também pode ser alvo de alterações. Talvez seja interessante investigar a ordem entre as tarefas de criação das árvores, verificações de estrutura (problemas de posicionamento) e verificação de atributos.

Com relação à interação com o usuário, pode-se ainda pensar em implementações de mensagens de erro de análise sintática que possam ser úteis na determinação das partes não gramaticais de uma sentença. Nesse sentido, talvez seja interessante fazer como o paradigma das *constraint grammars*, que impedem a eliminação da última alternativa remanescente.

Finalmente, acreditamos que nossa plataforma pode ser utilizada em diversas aplicações de processamento de linguagem natural, tais como (SANTOS, 2001a; VEIGA, 2001):

- Recuperação de informação para grandes quantidades de textos;
- Construção automática de ontologias para representação de conhecimento (GOTTSCHALG-DUQUE, 2004);
- Categorização de documentos;
- Levantamento automático de requisitos;
- Corretores ortográficos, sintáticos, gramaticais e estilísticos.
- Identificação de autor;
- Indexação inteligente;
- Tradução automática;
- Sistemas de ensino;
- Criação automática de sumários;

- Tradução entre as modalidades texto e fala;
- Criação automática de legendas.

9 REFERÊNCIAS BIBLIOGRÁFICAS

AHO, A. V., SETHI, R. e ULLMAN, J. D. ***Compilers: Principles, Techniques, and Tools***. Addison-Wesley Pub. Co., 1986.

AHO, A. V. e ULLMAN, J. D. ***The Theory of Parsing, Translation, and Compiling***, volume II: Compiling. Prentice Hall, 1973a.

AHO, A. V. e ULLMAN, J. D. ***The Theory of Parsing, Translation, and Compiling***, volume I: Parsing. Prentice Hall, 1973b.

BAUER, F. L., REMER, F. L. D., GRIFFITHS, M., HILL, U., HORNING, J. J., KOSTER, C. H. A., MCKEEMAN, W. M., POOLE, P. C. e WAITE, W. M. ***Compiler Construction***. Springer-Verlag, 1976.

BICK, E. **Automatic parsing of Portuguese**. Em *II Encontro para o processamento de português escrito e Falado*, págs. 91–100, 1996.

BICK, E. ***The Parsing System Palavras, Automatic Grammatical Analysis of Portuguese in a Constraint Grammar Framework***. Aarhus University Press, 2000.

BICK, E. **Automatic Syntatical Annotation**. Em *Primeira Escola de Verão da Linguateca*, 2006.

BICK, E. **Computational Syntax**. Em *Encontro Um passeio pela Floresta Sintá(c)tica*, 2007.

BLACKBURN, P. e BOS, J. ***Representation and Inference for Natural Language. A First Course in Computational Semantics***. CSLI, 2005.

CAMARGO, G., MARTINS, I., FUCS, J., CASTANHEIRA, J., BERNARDI, M. A., GALUPPO, R. E ROSSETTO, R., editores. ***Revista Exame***. Editora Abril, Agosto 1996.

CHIEH SHAN, C. **Monads for natural language semantics**. *CoRR*, cs.CL/0205026, 2002.

CHOMSKY, N. ***Sobre natureza e linguagem***. Martins Fontes, 2006.

COSTA, F. **Deep Linguistic Processing of Portuguese Noun Phrases**. Dissertação de Mestrado, Department of Informatics, University of Lisbon, November 2007. DI/FCUL TR-07-34.

DE NICOLA, J. e INFANTE, U. ***Gramática Comtemporânea da Língua Portuguesa***. Editora Scipione, 1993.

DONNELLY, C. e STALLMAN, R. ***Bison: The Yacc-compatible Parser Generator***. Free Software Foundation, bison version 2.3 edition, May 2006.

GOTTSCHALG-DUQUE, C. e LOBIN, H. **Ontology extraction for index generation**. Em *Proceedings of the 8th ICCC/IFIP International Conference on Electronic Publishing*, 2004.

HOPCROFT, J. E., MOTWANI, R. e ULLMAN, J. D. ***Introduction to Automata Theory, Languages, and Computation***. Addison Wesley, 2001.

HOPCROFT, J. E. e ULLMAN, J. D. ***Formal Languages and their relation to Automata***. Addison Wesley, 1969.

KARLSSON, F. **CONSTRAINT GRAMMAR AS A FRAMEWORK FOR PARSING RUNNING TEXT**. Em *13th International Conference on Computational Linguistics*, págs. 168–173, 1990.

KARLSSON, F., VOUTILAINEN, A., HEIKKILA, J. e ANTTILA, A. ***Constraint Grammar: A Language-independent System for Parsing Unrestricted Text***. Walter de Gruyter, 1994.

MANSUR, A., MASSON, C., EVELIN, G., MELLO, K. E GIRON, L. A., editores. ***Revista Época***. Editora Globo, 532 edition, Julho 2008.

MARTINS, R., NUNES, G. e HASEGAWA, R. **Curupira: A Functional Parser for Brazilian Portuguese**. Em MAMEDE, N. J., BAPTISTA, J., TRANCOSO, I. E DAS GRAÇAS VOLPE NUNES, M., editores, *PROPOR*, volume 2721 of *Lecture Notes in Computer Science*, págs. 179–183. Springer, 2003. ISBN 3-540-40436-8.

MARTINS, R. T., HASEGAWA, R., DAS GRAÇAS VOLPE NUNES, M., MONTILHA, G. e DE OLIVEIRA JR., O. N. **Linguistic issues in the development of ReGra: a Grammar Checker for Brazilian Portuguese**. *Natural Language Engineering*, 4:287–307, 1998.

MENEZES, P. B. e HAEUSLER, E. H. ***Teoria das Categorias para Ciência da Computação***. Editora Sagra Luzzatto, 2006.

MITKOV, R., editor. ***The Oxford Handbook of Computational Linguistics***. Oxford University Press, 2005.

OTHERO, G. A. **Grammar Play: um parser sintático em Prolog para a língua portuguesa**. Dissertação de Mestrado, Pontifícia Universidade Católica do Rio Grande do Sul, 2004.

OTHERO, G. A. e MENUZZI, S. M. **Sintaxe X-barra: uma aplicação computacional**. Working Papers em Lingüística, UFSC, 2008.

PARTEE, B. H., TER MEULEN, A. G. B. E WALL, R. E., editores. ***Mathematical Methods in Linguistics***. Springer, 1990.

RANGEL, J. L. **Apostila de Compiladores**. http://www-di.inf.puc-rio.br/ rangel/comp.html, 2000.

SANTOS, D. ***Tratamento das Línguas por Computador. Uma introdução à linguística computacional e suas aplicações***, chapter Processamento de linguagem natural através das aplicações. Caminho, 2001a.

SANTOS, D. e ROCHA, P. **Evaluating CETEMPúblico, a free resource for Portuguese**. Em *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, págs. 442–449, Toulouse, July 2001b.

SILVA, N. C. F. D. O. E. **Identificação automática de expressões cristalizadas preposicionais em corpora da língua portuguesa**. Dissertação de Mestrado, Instituto Militar de Engenharia, 2008.

TOMITA, M. ***Efficient Parsing For Natural Language***. Kluwer Academic Press, 1986.

TOMITA, M. **An Efficient Augmented-Context-Free Parsing Algorithm**. *Computational Linguistics*, 13:31–46, 1987.

TOMITA, M., editor. ***Generalized LR Parsing***. Springer, 1991.

VEIGA, P. e SANTOS, D. *Mais Línguas, Mais Europa: celebrar a diversidade linguística e cultural da Europa*, chapter **Contributo para o processamento computacional do português: o CRdLP**. Colibri, 2001.

10 APÊNDICES

10.1 INSTALAÇÃO

Para começar a usar o programa é necessário:

a) Obter uma cópia da plataforma com todos os seus diretórios;

b) Colocar essa cópia em um computador com máquina virtual *Java SDK 6.0* ou superior. Pode ser cópia em disco rígido ou qualquer outra mídia em que se possa atribuir permissão de escrita;

c) Executar, com permissão de escrita, o *script* "AnaliseSintatica.bat" que se encontra na raiz da estrutura de diretórios da plataforma. O *script* deve ser executado no diretório raiz da plataforma, pois os endereços de todos os arquivos usados durante a análise sintática dependem do endereço do diretório raiz.

10.1.1 PARÂMETROS

Existem alguns parâmetros que podem acrescentar funcionalidades à plataforma. Para alterar esses parâmetros é necessário manipular o arquivo "Recursos.properties" de cada projeto da plataforma. Falaremos aqui somente dos parâmetros do projeto "Análise Sintática".

- **Janela**: Define o número limite de alternativas de análise sintática em memória principal. Serve para controlar o uso de memória principal. Deve ser um número;
- **Escolha**: Serve para ativar a funcionalidade de escolha dos homônimos que irão participar da análise sintática. A escolha dos homônimos é feita em um componente visual. Só ativa com o valor "sim";
- **Caminhos**: Serve para ativar a funcionalidade de armazenamento de todos os caminhos de análise percorridos. Deve ser usado com cuidado, pois mesmo com sentenças pequenas o consumo de memória principal é grande. Só ativa com o valor "sim";
- **GuardaVisual**: Serve para ativar a funcionalidade de armazenamento dos componentes visuais que mostram os resultados de análise. Esses resultados ficam armazenados no diretório "Visualização" e podem ser recuperados através uma ferramenta administrativa da plataforma. Só ativa com o valor "sim".

FIG. 10.1: Componente visual de escolha de homônimos

10.2 DIAGRAMAS DE ESPECIFICAÇÃO

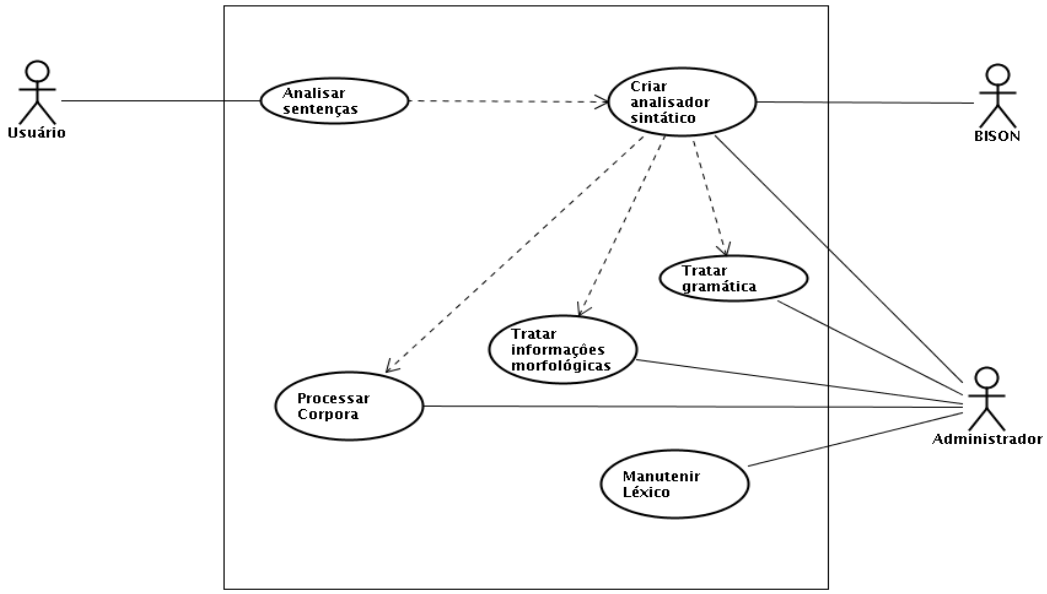

FIG. 10.2: Diagrama de caso de uso

Descrevemos abaixo cada caso de uso.

- – **Caso de Uso:** Processar Corpora

  - **Atores:** Administrador
  - **Tipo:** Primário
  - **Descrição:** Para que sejam criados léxicos extensos, é necessário processar corpora morfologicamente anotados. O Administrador coloca cópias dos corpora no diretório correto e aciona o início do processamento.

- **Caso de Uso:** Tratar informações morfológicas
  - **Atores:** Administrador
  - **Tipo:** Primário
  - **Descrição:** A descrição de diversas informações morfológicas é feita por meio de listas textuais. O Administrador manipula essas listas quando necessário.

- – **Caso de Uso:** Tratar gramática
  - **Atores:** Administrador
  - **Tipo:** Primário
  - **Descrição:** A gramática é um arquivo textual que descreve o funcionamento do analisador sintático. Administrador cria e corrige a gramática.
- – **Caso de Uso:** Manutenir léxico
  - **Atores:** Administrador
  - **Tipo:** Primário
  - **Descrição:** O processamento de corpora pode incorporar erros de anotação já presentes em algum corpus anotado. O Administrador seleciona um conjunto de palavras e altera o léxico para esse conjunto de palavras.
- – **Caso de Uso:** Criar analisador sintático
  - **Atores:** Administrador, Bison
  - **Tipo:** Primário
  - **Descrição:** A gramática é processada pela ferramenta Bison, que produz uma descrição textual do analisador sintático. O Administrador deve acionar o processamento dessa descrição textual para que o analisador seja efetivamente criado.
- – **Caso de Uso:** Analisar sentenças
  - **Atores:** Usuário
  - **Tipo:** Primário
  - **Descrição:** O Usuário deseja analisar sintaticamente sentenças de sua escolha. Para isso ele informa as sentenças a serem analisadas e a plataforma retorna as florestas sintáticas produzidas.

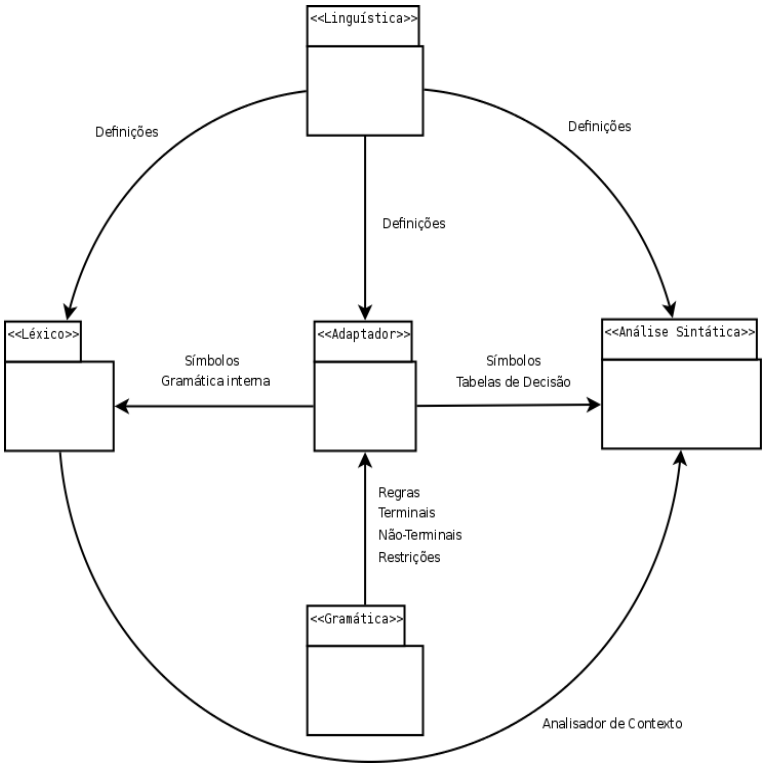

FIG. 10.3: Projetos da plataforma e suas dependências

A plataforma foi organizada a partir de diferentes projetos. Cada projeto teve sua motivação em um conjunto de atividades ou tarefas afins. O nome de cada projeto e as respectivas tarefas são:

- **Linguística**: Definição das classes que representam os conceitos linguísticos e de análise sintática que serão utilizados pela plataforma;
- **Gramática**: Escrita e manutenção da gramática;
- **Adaptador**: Obtenção das tabelas de redução, de empilhamento e de transição a partir do resultado do processamento da ferramenta BISON;
- **Léxico**: Manipulação de corpus e listas de definições morfológicas;
- **Análise Sintática**: Análise sintática de sentenças.

Dessa maneira, a plataforma é o resultado da integração de diferentes projetos. Todos os projetos foram totalmente desenvolvidos em Java no ambiente de programação Eclipse.

A comunicação entre os diferentes projetos é feita através de arquivos em disco. Essa foi uma escolha que teve por objetivo garantir que os produtos de cada projeto pudessem ser obtidos em diferentes computadores em diferentes momentos. Para disponibilizar classes, são utilizados arquivo do tipo JAR. Para disponibilizar objetos, são utilizados arquivos que armazenam a *serialização* dos mesmos objetos. A figura 10.3 mostra uma visão ampla das dependências entre os projetos.

Para melhor compreensão das dependências entre os projetos, apresentamos os produtos que cada projeto oferece, bem como os insumos de que precisa.

10.3.1 PROJETO **LINGUÍSTICA**

O projeto divide as definições entre seus pacotes: **conceito**, **estrutura**, **processual** e **sequencia**.

- **conceito**: contém as definições de classes abstratas **Atributo**, **Gramatical** e **Sintaxe**. A primeira serve para construir as classes que representam atributos morfológicos, tais como a classe **Genero**. Todas as classes que representam atributos são definidas nesse pacote. A segunda classe abstrata serve para a construção de classes que representam os terminais da gramática. A última classe abstrata serve para construir todos os não-terminais da gramática;

- **estrutura**: contém a definição das duas estruturas de dados mais amplamente utilizadas para representar as informações morfológicas e sintáticas: a classe **Mapa** e a classe **Arvore** respectivamente;

- **processo**: contém as definições associadas ao processo de análise sintática, tais como **Regra**, **Estado** e **Decisao**. Essa última encapsula as tabelas de redução, de empilhamento e de transição da análise GLR. Nesse pacote também ficam a definição da representação da gramática, na classe **Gramatica**, a definição da classe **Morfologia**, responsável pela implementação dos métodos associados a atributos e a definição da classe responsável pela análise de contextos: a classe **Contexto**;

- **sequencia**: armazena as definições associadas à representação sequencial de palavras: **Palavra**, **Homonimia**, que representa um conjunto de homônimos, **Lexico** e **Dicionario** que lida com os dois léxicos: um obtido a partir de corpora anotados e o outro a partir de listas.

10.3.2 PROJETO **GRAMÁTICA**

O projeto **Gramática** é responsável por produzir o arquivo texto no formato que o BISON aceita. Isso pode ser feito diretamente ou em cooperação com os construtores de padrões.

Primeiramente a gramática define o conjunto de todos os terminais que serão usados por toda a plataforma. Para isso é utilizada a diretiva *%token* como no extrato abaixo:

```
%token  Adjetivo
%token  Adverbio
%token  Artigo
```

Ao final da definição de todos os terminais, é incluída a diretiva *%debug* para que o BISON produza o arquivo de descrição textual do analisador sintático. Logo em seguida é incluída a diretiva *%glr-parser* para que o BISON produza um analisador GLR. Isso se reflete na descrição textual que o BISON produz. Para marcar o início da parte onde são definas as regras de criação de todos os não-terminais da gramática, é inserida a diretiva *%%*. Os não terminais são definidos conforme a documentação do BISON (DONNELLY, 2006).

```
Frase   :  Interjeicao Pontuacao_Final
        |  Periodo Pontuacao_Final
        ;
```

Além disso, para cada regra podem ser definidas restrições de atributos e métodos de construção de novos atributos conforme dito anteriormente.

```
Objeto_Direto   :  Pronome_Pessoal_Atono
                   /* %! nao_funcao[<objeto_indireto,1>] !% #! nominal[<1>] !# */
                |  Sintagma_Substantivo
                   /* #! nominal[<1>] mapear[<Pessoa.TERCEIRA>] !# */
                |  Oracoes_Subordinadas_Substantivas
                   /* #! nominal[<1>] !# */
                ;
```

10.3.3 PADRÕES DA GRAMÁTICA

Os padrões da gramática são descritos por uma formatação específica. Os construtores são identificados pela presença da sequência ":=" em uma linha da gramática. O tipo de construtor é identificado pela sequência de caracteres após ":=". Os argumentos são obtidos entre os parênteses da linha.

```
                Sujeitos     :=    Lista ( Sujeito )
Sintagma_Oracional_Intransitivo_Completo    :=    Extensão ( Sintagma_Oracional_Intransitivo , Verbo_Intransitivo )
```

O construtor **Lista** não depende de outras regras da gramática. A classe **Padroes** somente aproveita os nomes que estão na linha, produzindo o resultado:

```
Sujeitos          :  Sujeito
                  |  Sujeito_Composto Conjuncao_Coordenativa Sujeito
                  ;
Sujeito_Composto  :  Sujeito
                  |  Sujeito_Composto Pontuacao Sujeito
                  ;
```

O construtor **Extensão** é mais complexo, dependendo das regras já escritas na gramática e incluindo um construtor de listas. Para facilitar a compreensão, chamaremos de *extensão* ao não-terminal que aparece na começo da linha, de *base* ao primeiro argumento e de *núcleo* ao segundo do construtor.

A classe **Padroes** identifica pela posição a extensão, a base e o núcleo. Depois disso, o Legislador procura em toda a gramática o começo de todas as regras que definem o não-terminal base. Ele faz isso, procurando pela linha em que a sequência de caracteres da base aparece bem no começo da linha. No exemplo acima, temos respectivamente **Sintagma_Oracional_Intransitivo_Completo**, **Sintagma_Oracional_Intransitivo** e **Verbo_Intransitivo**.

No exemplo acima, o **Padroes** encontra as regras abaixo. Numeramos as regras para facilitar a explicação.

```
    Sintagma_Oracional_Intransitivo
1)      :   Sintagma_Verbal_Intransitivo
2)      |   Sujeitos Verbo_Intransitivo
3)      |   Sujeitos Verbo_Intransitivo Locucao_Adverbial_Composta
4)      |   Locucao_Adverbial_Composta Pontuacao Sujeitos Verbo_Intransitivo
        ;
```

A partir dessa regra o **Padroes** procura verificar quais regras podem ser usadas para definir a extensão. Para isso, ele primeiro verifica se a regra possui mais de 2 símbolos. No exemplo acima, as regras 1 e 2 são descartadas por esse critério. Depois disso, é verificada a presença do núcleo na regra. Se a regra não contém o núcleo, ela também é eliminada. Por esse critério a regra 1 é excluída.

Passados esses dois critérios, a regra é aceita para construir a extensão. A nova regra é criada e inserida dentro de um construtor de listas, nesse caso, de listas de estruturas dependentes. A nova regra é obtida retirando-se o núcleo da regra aceita e colocando um terminal **Pontuacao** no lugar, exceto no caso de o núcleo estar no começo ou no fim da regra. Nessa situação, não é colocado o terminal **Pontuacao**.

```
    Sintagma_Oracional_Intransitivo_Completo
5)      :   Sintagma_Oracional_Intransitivo_Composto
6)      |   Sintagma_Oracional_Intransitivo_Composto Conjuncao_Coordenativa Sujeitos Pontuacao Locucao_Adverbial_Composta
7)      |   Sintagma_Oracional_Intransitivo_Composto Conjuncao_Coordenativa Locucao_Adverbial_Composta Pontuacao Sujeitos
        ;
    Sintagma_Oracional_Intransitivo_Composto
8)      :   Sintagma_Oracional_Intransitivo_Composto
9)      |   Sintagma_Oracional_Intransitivo_Composto Pontuacao Sujeitos Pontuacao Locucao_Adverbial_Composta
10)     |   Sintagma_Oracional_Intransitivo_Composto Pontuacao Locucao_Adverbial_Composta Pontuacao Sujeitos
        ;
```

Observamos que as regras 5 e 8 fazem parte do construtor de listas, que as regras 6, 9 foram obtidas a partir da regra 3 e que as regras 7 e 10, a partir da regra 4.

10.3.4 PROJETO **ADAPTADOR**

O projeto **Adaptador** é responsável por construir as tabelas de decisão da análise GLR a partir da descrição textual dos estados gerada pelo BISON no processamento da gramática. Para isso, todas as possíveis ações e transições de cada estado são aproveitadas.

Além disso, como a gramática guarda a definição de todos os terminais e não-terminais, esse projeto também realiza a criação e compilação das classes que representam todos os símbolos a serem usados pela plataforma.

As classes que representam os terminais são criadas como classes que especializam a classe abstrata **Gramatical**. No caso dos não-terminais, as classes especializam a classe **Sintaxe**. Essas duas classes abstratas implementam a interface **Linguistica**. As classes depois de compiladas são armazenadas em um arquivo do tipo *jar* e disponibilizadas para os demais projetos.

O projeto, por lidar diretamente com a gramática, também é responsável pela criação da representação interna da gramática.

10.3.5 PROJETO **LÉXICO**

O projeto **Morfeu** possui dois pacotes: **morfossintatico** e **corpus**. O primeiro é responsável pela manipulação de corpora e das diversas listas que integram o projeto, o segundo é responsável por facilitar o processo de limpeza e manipulação do léxico obtido a partir de corpora anotados, abrindo uma janela de edição.

Dentro do pacote **morfossintatico** temos a classe **CETEMPublico**, uma clara referência ao corpus escolhido para a construção de um dos léxicos da plataforma. A classe **Extrator** cria extratos de arquivos muito grandes, tais como os corpora anotados.

A classe **Primitivas** realiza a construção de um léxico a partir de listas de palavras conhecidas. As demais classes do pacote são responsáveis por criar léxicos a partir de listas de regências, de funções, de expressões multivocabulares e de expressões multivocabulares descontínuas.

A classe **Morfolex** é responsável por integrar os produtos das demais classes e tornar persistente o segundo léxico da plataforma. Além disso, ela é responsável por produzir um objeto da classe **Morfologia** e, através da recuperação da representação interna da gramática, tornar persistente um objeto da classe **Contexto**.

10.3.6 PROJETO **ANÁLISE SINTÁTICA**

Nesse projeto é realizado o objetivo principal deste trabalho: a análise sintática de sentenças do português. O projeto **Sinapse** é dividido em dois pacotes: **comunicacao**, responsável pela comunicação com o usuário, e **semantica**, responsável pela identificação das alternativas para as sentenças e pela análise sintática propriamente dita.

O resultado do processamento é apresentado em um componente visual em estrutura de árvore. Nele se encontram todos os processamentos realizados para uma dada sentença. Cada processamento corresponde a uma alternativa de interpretação da sentença. Isso não quer dizer interpretação das palavras em razão dos homônimos, mas sim as possibilidades de uma sequência de caracteres corresponder a um conjunto de palavras distintas ou a uma única palavra.

A classe **Sentenciador** do pacote **semantica** tem por responsabilidade verificar essas possibilidades em sua lista de expressões conhecidas. Ele identifica uma expressão e cria duas sentenças, que correspondem a duas interpretações: uma em que a expressão participa e outra em que ela não participa. E assim uma mesma sentença pode produzir várias interpretações. Existe ainda a situação em que somente a interpretação com a expressão sobrevive, pois existe a presença de termos desconhecidos pelo dicionário. Isso acontece muito com nomes próprios compostos.

Mecanismo similar funciona para as estruturas. O Sentenciador identifica a possível presença de uma estrutura e cria duas interpretações. Na interpretação com a estrutura, os fragmentos são retirados e a palavra correspondente é inserida na construção da interpretação na segunda posição dos fragmentos.

A classe **Automato** realiza todas as instruções inseridas nas tabelas de redução, de empilhamento e de transição. Para cada ação ou transição da lista fornecida pelas tabelas de decisão é criada uma cópia da configuração atual e aplicada essa ação ou transição.

Mas a classe que coordena todas as atividades em torno da análise sintática é a classe **Processamento**. Essa classe recebe o texto bruto, chama a classe **Sentenciador**, cria a primeira configuração, chama a classe **Automato** para iniciar o processamento, verifica o tamanho do conjunto de configurações em memória principal e organiza os resultados a serem apresentados ao usuário.

10.4 TRATAMENTO DO PROBLEMA DE POSICIONAMENTO

O problema de posicionamento é verificado pela classe **Contexto** do pacote **processual** do projeto **Linguistica**. Os objetos dessa classe são responsáveis pela verificação de dependências de posicionamento e pela verificação das restrições de atributos das regras da gramática.

Para realizar essas tarefas, os objetos da classe **Contexto** armazenam um objeto da classe **Gramatica** e um objeto da classe **Morfologia**. Como o próprio nome sugere, um objeto da classe **Gramatica** é a representação em objeto das regras escritas em formato texto. Nele ficam armazenados mapeamentos entre objetos que implementam a interface **Conceito** e listas de objetos da classe **Regra**. Isso permite investigações de posicionamento em todas as regras possíveis.

A classe **Morfologia** contém todos os métodos de verificação de atributos e de criação de novos atributos. Os objetos dessa classe armazenam dois mapeamentos: o de regências e o de funções. O mapeamento de regências é feito entre objetos da classe **Lema** e listas de objetos da mesma classe. O mapeamento de funções é feito entre objetos da classe **String** e listas de objetos da classe **Lema**. Dessa maneira, as verificações de regência e de funções não são estáticas, dependendo de objetos que são criados pelo projeto **Léxico**.

Um objeto da classe **Contexto** recebe uma regra e uma lista de símbolos e deve responder se a regra pode ser aplicada sobre a lista. Para isso ele percorre a lista de símbolos verificando se existe algum problema de posicionamento entre as subárvores de dois símbolos consecutivos. Se não houver problema de posicionamento, são verificadas as restrições de atributos da regra sobre os símbolos da lista. Queremos ressaltar que a lista é sempre do tamanho da regra, pois ela é obtida como a sublista que está no topo da pilha de subárvores e que tem tamanho igual ao da regra.

A verificação de problemas de posicionamento acontece sempre entre duas subárvores: a **esquerda** e a **direita**. Se esquerda é terminal a verificação termina e retorna falso para a presença de problema. Se esquerda não é terminal, criamos um iterador chamado **cauda** para todos os símbolos que ficam na fronteira direita de **esquerda** e uma lista chamada **cabeca** para todos os símbolos na fronteira esquerda de **direita**.

Para cada elemento de **cauda**, percorremos com um iterador a lista **cabeca**. A possibilidade de problema de posicionamento é verificada perguntando para o objeto da classe **Gramatica** se o elemento de **cauda** possui alguma regra em que o elemento de **cabeca** esteja na última posição. Se isso não acontecer, não existe problema.

Em cada situação em que existe a possibilidade, são construídas duas listas de símbolos: uma chamada **virtual** com o elemento de **cauda** e o elemento de **cabeca** nessa ordem e outra chamada **conceitos** com todos os filhos do elemento de **cauda** e o elemento de **cabeca** nessa ordem. No entanto, as verificações ocorrem sobre listas de subárvores, por essa razão, criamos também as listas **acoplamento**, associada a **virtual**, e a lista **filhos**, associada a **conceitos**.

Depois disso, o objeto da classe **Gramatica** fornece a lista das regras em que há possibilidade de problemas de posicionamento. Percorremos essa lista de regras perguntando se sua lista de conceitos é igual a **virtual** ou igual a **conceitos**. Em caso afirmativo, verificamos se a regra aplicada à lista de subárvores associada (**acoplamento** ou **filhos**) atende a suas restrições de atributos. Se não atende, não há problema de posicionamento. Caso contrário, a função retorna imediatamente verdadeiro para a existência de problema.

```
private boolean dependencia ( Arvore esquerda , Arvore direita ){

if ( ! esquerda.isLeaf() ){

Iterator<Arvore> cauda = esquerda.cacula().cauda().iterator();
List<Arvore> cabeca = direita.cabeca();

while ( cauda.hasNext() ){

Arvore  contexto  = cauda.next();

Conceito  naoterminal = ( ( Mapa ) contexto.objeto() ).linguistica();

Iterator<Arvore> cabecas = cabeca.iterator();

while ( cabecas.hasNext() ){
```

```
Conceito  conceito = ( ( Mapa ) cabecas.next().objeto() ).linguistica();

// fica configurada dependência quando
// o conceito da direita poderia ter participado
// da regra que formou um dos conceitos da cauda.

if ( gramatica.ultimo ( naoterminal , conceito ) ){

// para verificar diretamente com o conceito
List<Conceito>virtual = new ArrayList<Conceito>();

virtual.add ( naoterminal );
virtual.add ( conceito );

List<Arvore> acoplamento = new ArrayList<Arvore>();

acoplamento.add ( contexto );
acoplamento.add ( direita );

// para verificar com filhos na árvore
List<Arvore>  filhos = contexto.filhos();

filhos.add ( direita );

List<Conceito>conceitos = projecao ( filhos );

Iterator<Regra> regras  = gramatica.regrasCauda ( naoterminal , conceito ).iterator();

while ( regras.hasNext() ){

Regra regra = regras.next();

// verifica com o próprio conceito
if ( regra.conceitos().equals( virtual ) ){

// verifiquei aqui a dependencia posicional
// falta verificar as restrições da própria regra sobre a lista de árvores
if ( germinar ( regra , acoplamento ) ){

//System.out.println ( "Acoplamento interno encontrado. " + virtual );

return true;

} }

// verifica com filhos
if ( regra.conceitos().equals( conceitos ) ){

// verifiquei aqui a dependencia posicional
// falta verificar as restrições da própria regra sobre a lista de árvores
if ( germinar ( regra , filhos ) ){

//System.out.println ( "Dependência interna encontrada. " + conceitos );

return true;

} } } } } }

return false;

}
```

10.5 TRATAMENTO DE ATRIBUTOS

A verificação de restrições de atributos também é feita por objetos classe **Contexto**. Para isso é necessário informar o objeto da classe **Regra** e a lista de subárvores sobre a qual se deseja aplicar a regra. O objeto da classe **Regra** fornece o nome de todos os métodos de verificação e a lista com as aplicações de todos os métodos. Cada aplicação de um método corresponde a uma lista de listas de argumentos.

Percorremos a lista de nomes de métodos e a lista de aplicações simultaneamente, pois cada nome de método corresponde a um elemento da lista de aplicações na ordem em que aparecem. Sendo assim, percorremos a lista de listas de argumentos que corresponde ao nome do método. Para cada lista de argumentos devemos fazer a invocação do método para verificar se o seu resultado é verdadeiro ou falso. Se alguma invocação de algum método de verificação retornar falso, a regra é rejeitada. Caso contrário a regra é aceita.

A invocação do método é feita através da classe **Method** por reflexão. Essa foi a maneira encontrada de não tornar o código da classe **Contexto** dependente do código da classe **Morfologia**.

Um detalhe com relação aos argumentos: se a classe de um argumento for **Integer**, o argumento é substituído pela subárvore do símbolo que ocupa a posição na regra que o número representa.

```
public boolean germinar ( Regra regra , List<Arvore> floresta ){

//System.out.println ( " ** regra deriva: " + regra.deriva() + " restricoes: " + restricoes.keySet() );

Iterator<String> metodos = regra.restricoes();
Iterator<List<List<Object>>> aplicacoes = regra.restricoes_aplicacoes();

while ( metodos.hasNext() ){

String nome = metodos.next();

Iterator<List<Object>> listas = aplicacoes.next().iterator();

while ( listas.hasNext() ){

List<Object> argumentos = listas.next();

try{

Method metodo = Morfologia.class.getDeclaredMethod( nome , tipos ( nome , argumentos ) );

int tamanho = argumentos.size();
```

```
Object[] projecao = new Object [ tamanho ];

for ( int i = 0 ; i < tamanho ; i++ ){

if ( argumentos.get ( i ).getClass().equals( Integer.class ) ){

projecao [ i ] = floresta.get ( ( Integer ) argumentos.get ( i ) );

}
else{

projecao [ i ] = argumentos.get ( i );

} }

if ( Boolean.FALSE.equals ( metodo.invoke( morfologia , projecao ) ) ){

return false;

} }
catch ( Exception e ){

System.out.println ( "Problema germinar." );
System.out.println ( e.getMessage() );
e.printStackTrace( System.out );

} } }

return true;

}
```

10.6 OBJETOS NECESSÁRIOS AO ANALISADOR SINTÁTICO

O **Adaptador** é responsável por organizar e codificar as informações textuais em código, produzindo a **Gramática Interna** e as **Tabelas de Decisão**. A **Gramática Interna** encapsula todas as regras da gramática, permitindo a verificação de restrições em todos contextos. As **Tabelas de Decisão**, por sua vez, definem o funcionamento do analisador sintático, determinando as alternativas de ação ou de transição para o autômato que realiza a análise sintática. No entanto, o funcionamento desse projeto pressupõe que a gramática e a descrição do analisador sintático estejam sincronizadas, ou seja, é necessário que a descrição seja produto da gramática a ser lida pelo projeto.

O projeto **Léxico** contribui para a geração do analisador sintático fornecendo o **Analisador de Contexto**. Ele é construído a partir de um objeto da classe a **Gramatica** e de um objeto da classe **Morfologia**. Um objeto da classe **Morfologia** é construído a partir de listas de informações morfológicas, especificamente as listas de regências e as listas de funções. Essas listas são arquivos textuais que guardam o lema de cada vocábulo que participa da lista.

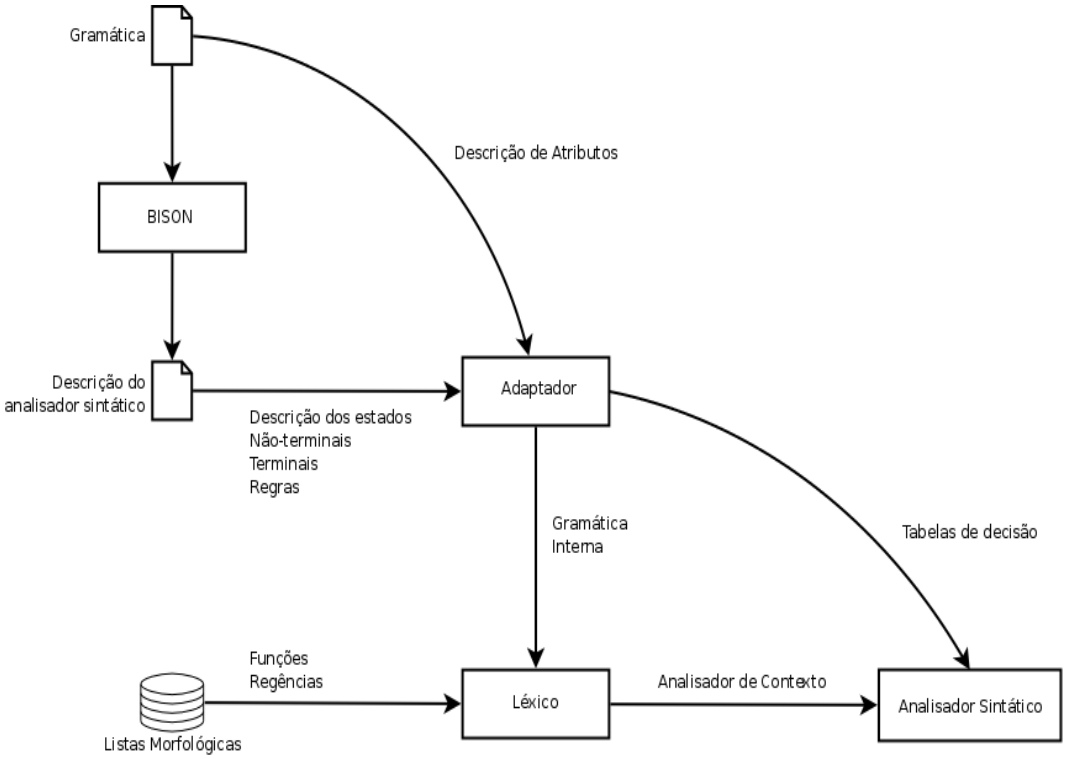

FIG. 10.4: Geração do analisador sintático

O analisador sintático pressupõe que as **Tabelas de Decisão** e o **Analisador de Contexto** correspondem, ou seja, que o **Analisador de Contexto** tenha sido produzido pela **Gramática Interna** que foi construída na mesma execução do **Adaptador**. Sobre as listas, não existem quaisquer exigências de sincronismo.

10.7 LISTAS DE FRASES

Aqui estão as listas de frases empregadas nos testes.

10.7.1 TESTES DE CALIBRAÇÃO

1) *"Só me resta agradecer", disse Galuppo.*

2) *Ambos são grisalhos, usam barba e dispensam gravatas.*

3) *O Pão de Açúcar era a quinta maior empresa do Brasil em 1986.*

4) *Desde então, vem recuperando espaço.*

5) *E das Indústrias Reunidas Francisco Matarazzo, quem se lembra?*

6) *Era a 16ª empresa na primeira lista de MELHORES E MAIORES, publicada em 1974.*

7) *Hoje virou um conjunto de ruínas.*

8) *Enquanto as vendas se reduziam, as do concorrente Carrefour aumentavam.*

9) *Sua antiga parceira, a Ford, ocupa a 16ª posição.*

10) *É um resultado superior aos faturamentos individuais de 373 empresas da lista.*

11) *Ocupa a sétima posição, com faturamento de 3,8 bilhões de dólares.*

12) *Deu certo.*

13) *A Shell propriamente dita agora só cuida da distribuição de combustíveis.*

14) *"Mas a demanda cresceu além de nossas expectativas."*

15) *Essas, a rigor, foram idênticas para todos.*

16) *Significa vantagens muito mais concretas.*

17) *Em 1995, foi o 11º colocado entre as que mais faturaram.*

18) *Seu principal concorrente, o Carrefour, manteve-se na sexta posição.*

19) *Essa norma se aplica a qualquer tipo de empresa.*

20) *É a sétima colocada, com um faturamento de 3,8 bilhões de dólares.*

21) *Observe-se a Ford.*

22) *A Volks, por seu lado, deseja tudo.*

23) *Menos ceder a liderança.*

24) *No ano passado, faturou 6,3 bilhões de dólares.*

25) *Não é pouco dinheiro, convenhamos.*

26) *No passado, ser uma grande empresa no Brasil poderia significar muitas coisas.*

27) *A lista publicada nesta edição é muito mais diversificada.*

28) *A primeira empreiteira que aparece, a Andrade Gutierrez, está em 82º lugar.*

29) *A presença da dupla de produtores de aço é importante.*

30) *Antes, ela jorrava na Praça dos Três Poderes.*

31) *Agora, mina onde o consumidor estiver.*

32) *Agradá-lo pode ser mais trabalhoso do que marcar presença nos gabinetes oficiais.*

33) *É isso que as empresas brasileiras estão aprendendo numa velocidade impressionante.*

34) *A concorrente tampouco quer ficar atrás.*

35) *Uma conseqüência possível é que voltem a galgar posições na lista.*

36) *Ano passado não foi um ano fácil.*

37) *O ramo de maior rentabilidade foi a indústria farmacêutica.*

38) *A menos endividada de todas as 500 é a destilaria Heublein.*

39) *Deve o equivalente a 2,9% de seus ativos.*

40) *A Heublein é também a empresa com maior liquidez.*

41) *Tem 11,41 reais disponíveis para cada real em dívidas.*

42) *A mais endividada é a Transbrasil, seguida de perto pela Vasp.*

43) *A Transbrasil deve o equivalente a 132% de seus ativos.*

44) *As quatro empresas privadas com mais capital de giro próprio são empreiteiras.*

45) *A Andrade Gutierrez lidera com 1,9 bilhão de dólares.*

46) *Em seguida, vem a Camargo Corrêa, com 1,1 bilhão de dólares.*

47) *Seu lucro foi equivalente a 950% de seu patrimônio líquido.*

48) *O pagamento de dividendos pela Cemig é um marco histórico.*

49) *Isso sim é um investimento de longo prazo.*

50) *Esse resultado equivale a 3% do patrimônio da estatal.*

51) *Mostra uma das dezenas de fábricas de micros que havia no Brasil.*

52) *Dos 20 maiores prejuízos registrados em 1995, 16 são de empresas estatais.*

53) *Em 1994, as estatais registraram um lucro de 1,5 bilhão de dólares.*

54) *Isso equivale ao endividamento das 500 maiores empresas privadas.*

55) *É baixo.*

56) *A recordista é a paulista Fepasa, de transporte ferroviário.*

57) *Perdeu 1,3 bilhão de dólares em 1995.*

58) *O prejuízo do Dersa foi de 753 milhões de dólares.*

59) *Hoje é a 17ª.*

60) *A Usiminas seria a 16ª colocada na lista de 1990.*

61) *Hoje está na 19ª posição.*

62) *Esse é um dos lados da moeda.*

63) *Todas as outras aumentaram suas vendas.*

64) *A empresa tinha uma receita de 496 milhões de dólares em 1991.*

65) *Suas vendas no ano passado foram de 820 milhões de dólares.*

66) *"Não basta ter caixa elevado", diz seu presidente, o ex-ministro Luiz Fernando Cirne Lima.*

67) *"É preciso saber aproveitar a liquidez."*

68) *O mesmo receituário foi aplicado na antes moribunda Cosipa.*

69) *A rentabilidade da siderúrgica paulista, hoje controlada pela Usiminas, foi de 3,6%.*

70) *Parece pouco.*

71) *Quem já tinha as contas em ordem, então, melhorou ainda mais.*

72) *Não é sempre que a resposta vem tão rápido.*

73) *Chegou a 300 milhões de dólares – ainda em conseqüência dos ajustes.*

74) *"Não existe milagre."*

75) *Segundo ele, a prioridade é adequar a empresa ao mercado.*

76) *"Só em 1997 devemos sair do prejuízo", diz.*

77) *Petrobrás e BR Distribuidora também fazem parte dessa relação.*

78) *Dos 20 maiores empregadores do Brasil, sete são estatais.*

79) *A Petrobrás lidera o ranking.*

80) *Gerava 46226 contracheques em dezembro do ano passado.*

81) *A cada ano, fica mais difícil para as estatais justificarem seu prejuízo.*

82) *As defasagens de tarifas que existiam no passado foram eliminadas pela estabilidade.*

83) *Mesmo assim, algumas delas continuam andando em ritmo lento.*

84) *Seu prejuízo foi de 463 milhões de dólares.*

85) *Sempre que se comparam duas empresas, é preciso tomar alguns cuidados.*

86) *Uma firma de telecomunicações é, obviamente, distinta de uma siderúrgica.*

87) *O governo está prometendo acelerar a privatização a partir do próximo ano.*

88) *O ramo de telecomunicações é visto hoje como um dos mais promissores.*

89) *Em 1995, as empresas de telefonia cresceram 8,3%.*

90) *Devem crescer ainda mais com a privatização das telecomunicações.*

91) *Hoje, a frota brasileira é muito mais diversificada.*

92) *Seria alguma coisa abaixo de 10 ou teria desaparecido.*

93) *Como soube evoluir, a Brahma cresceu.*

94) *No ano passado, foram 35%.*

95) *Você pode não acreditar, mas as máquinas que está vendo são computadores.*

96) *E foram mesmo.*

97) *No ano passado, foi vendido no Brasil 1 milhão de computadores pessoais.*

98) *A venda deve aumentar 50% este ano.*

99) *Houve uma época em que Estrela era sinônimo de brinquedo no Brasil.*

100) *Este ano, foi para o 484º posto.*

10.7.2 TESTES DE AVALIAÇÃO

1) *Também são controversas as disputas familiares de Agripino.*

2) *Dois exemplos recentes seguiram essa linha de atuação.*

3) *O primeiro é a Operação Satiagraha.*

4) *Tecnologia faz o impossível.*

5) *Esse recorde criou uma situação inusitada.*

6) *É patrimônio da humanidade.*

7) *Surpresas são uma constante na indústria do petróleo.*

8) *O Brasil está distante desse cenário belicoso.*

9) *São quase mil.*

10) *Lagos desaparecem.*

11) *O comércio também é intenso.*

12) *O vôo dura apenas três horas e meia.*

13) *Cerca de 35 mil são bolivianos.*

14) *A proximidade geográfica dos países facilita o trânsito humano.*

15) *Mas o nome do capitão é o mesmo.*

16) *É um tremendo desafio de engenharia aeronáutica.*

17) *Um vôo cruzará os Estados Unidos de costa a costa.*

18) *Piccard levará 120.*

19) *É uma grande epopéia aérea.*

20) *Meu pai e meu avô me transmitiram essa paixão.*

21) *Tinham existências extraordinárias.*

22) *Eram os heróis da minha infância.*

23) *Era surpreendente.*

24) *Sinto falta daqueles momentos.*

25) *É verdade.*

26) *A demanda parece ilimitada.*

27) *Os relatórios eram desanimadores.*

28) *O potencial é enorme.*

29) *Uma dessas áreas é a engenharia de poços.*

30) *É um grande laboratório em escala real.*

31) *No começo eles eram grandes.*

32) *Encolheram.*

33) *Depois vieram as câmeras e os tocadores de música.*

34) *Estão mais largos e maiores.*

35) *Lígia está em seu sexto aparelho.*

36) *Os fabricantes estão otimistas com o crescimento das vendas.*

37) *Você é normal.*

38) *Trabalho no setor de produção.*

39) *De médico e louco todo mundo tem um pouco.*

40) *Prexige é a dica.*

41) *Os consumidores já viram esse filme outras vezes.*

42) *Muitas investem em estudos de longo prazo.*

43) *Está liberado.*

44) *Apenas a versão injetável continua no mercado.*

45) *Ela é uma das principais causas de inflamação.*

46) *Elas são eficientes e mais baratas.*

47) *O combate à dor não depende apenas de remédios.*

48) *Nós pensamos financiamento empresarial.*

49) *Não pouparam nem as crianças.*

50) *A Croácia e a Eslovênia declararam independência.*

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo