UNIVERSIDADE FEDERAL DE SANTA CATARINA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

'UMA INTERFACE DE LINGUAGEM NATURAL PARA UM SISTEMA DE ADMINISTRAÇÃO DE CAPITAL DE GIRO'

DISSERTAÇÃO SUBMETIDA À UNIVERSIDADE FEDERAL DE SANTA CATARINA PARA A OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA

FERNANDO BORGES MONTENEGRO.

FLORIANÓPOLIS, FEVEREIRO DE 1991

'UMA INTERFACE DE UNGUAGEM NATURAL PARA UM SISTEMA DE ADMINISTRAÇÃO DE CAPITAL DE GIRO'

FERNANDO BORGES MONTENEGRO.

ESTA DISSERTAÇÃO FOI JULGADA ADEQUADA PARA A OBTENÇÃO DO TÍTULO DE

MESTRE EM ENGENHARIA

ESPECIALIDADE ENGENHARIA DE PRODUÇÃO E APROVADA EM SUA FORMA FINAL PELO PROGRAMA DE PÓS-GRADUAÇÃO

RICARDO MIRANDA BARCIA, Ph.D - ORIENTADOR

NERI DOS SANTOS, Dr.Ing - COORDENADOR DO CURSO

BANCA EXAMINADORA

RICARDO MIRANDA BARCIA, Ph.D - PRESIDENTE

NERI DOS SANTOS, Dr.Ing.

LEILA AMARAL GONTIJO, Dr.Eng.

Aos meus pais

AGRADECIMENTOS:

Ao professor Ricardo Barcia, pela orientação e apoio no preparo deste trabalho. E também por todo o seu esforço em conseguir meios para a manutenção do grupo de Inteligência Artificial.

Aos amigos Paulo Luna e Roberto Pacheco, companheiros na construção do Sistema de Administração de Capital de Giro, pelo incentivo e ajuda dispensada durante todo o periodo do curso.

Ao professor Álvaro Lezana, pelo seu apoio e trabalho prestado na função de especialista em Capital de Giro.

Aos integrantes do Grupo de Inteligência Artificial.

Aos meus familiares e amigos, pelo o incentivo, compreenssão e auxílios fornecidos no decorrer do curso de pós-graduação.

À CAPES pelo apoio financeiro através da bolsa de estudo que me foi concedida.

E a todos os outros que de alguma forma ajudaram durante o desenvolvimento do trabalho.

RESUMO:

Esta dissertação propõe uma Interface em Linguagem Natural, para a Língua Portuguesa, construída com uma estrutura flexível. A Interface foi desenvolvida para o Sistema Inteligente de Apoio à Decisão para a Administração do Capital de Giro (SECAGI).

Uma interface em linguagem natural propicia uma linguagem de comando para sistemas iterativos baseada em palavras e frases familiares ao usuário. Esta familiaridade incrementa significativamente a produtividade do usuário, facilitando a interação com o sistema.

Neste trabalho são apresentadas diversas técnicas para cada módulo do Processamento de Linguagem Natural. Em seguida, é apresentada a estrutura escolhida do sistema, composta pelos módulos: Léxico, Sintático (incluindo Gramática), Análise de Consistência e Semântico.

ABSTRACT

This work presents a Natural Language Interface for the Portuguese Language, with a flexible structure. This interface was developed for Working Capital Management Inteligent Decision Support System (SECAGI).

This Natural Language Interface given an iterative system comand language upon words and sentences commonly used by the user, substantialy increasing his (her) produtivity.

Various techniques usual for Natural Language are presented. The chosen structure, that is, the Lexical, Syntactic (grammars), Consistence and Semantic Analysis are presented too.

ÍNDICE

ÍNDICE DE FIGURAS

CAPÍTULO I - INTRODUÇÃO

1.1 - INTRODUÇÃO

O Sistema de Administração de Capital de Giro é um Sistema Inteligente de Apoio à Tomada de Decisão, no qual conhecimentos heurísticos de um especialista são utilizados em associação a modelos matemáticos existentes. No sistema existe uma base de dados e outra de conhecimento, com as quais o usuário irá interagir para poder chegar a solução de um problema ou fazer uma simples pesquisa no banco de dados.

Para que ocorra o diálogo entre o programa e o usuário é necessário uma (ou várias) interface(s), que podem ser menus, teclas de comando, linguagem natural, etc. Na maior parte das vezes, uma Interface em Linguagem Natural facilitá o trabalho para as questões dos diálogos com o usuário, pois este utiliza sua Linguagem Natural, sem ter que aprender algo novo.

O Sistema de Processamento de Linguagem Natural, proposto aqui, está classificado na área da Linguagem Natural que se preocupa com a elaboração de sistemas de cunho prático, para aplicações reais. Portanto utilizando técnicas já desenvolvidas, da melhor maneira possível para se fazer o sistema proposto.

As técnicas acima referidas são principalmente as gramáticas, os parsers (analisadores gramáticais) e a análise semântica. A gramática de uma linguagem é um esquema para especificar a posição de uma sentença em uma linguagem, indicando regras para combinar palavras dentro de frases e ciaúsulas. Os parsers servem para de acordo com a gramática escolhida (ou na mistura entre algumas delas) determinar as funções das palavras na sentença de entrada de modo que crie uma estrutura de dados. A parte semântica vai dizer qual o significado da sentença, fazendo a sua interpretação. Para tudo isto é necessário a criação de uma estrutura léxica contendo informações sintáticas e semânticas.

1.2 - OBJETIVO:

O objetivo principal do trabalho é o de construir um protótipo de uma Interface em Linguagem Natural para o Sistema de Administração de Capital de Giro, de modo que o seu usuário possa interagir com o sistema através de sua linguagem natural, ou seja, na Língua Portuguesa.

A contrução do sistema é toda feita utilizando uma estrutura o mais genérica possível, a qual foi desenvolvida para a construção (também na forma de protótipo) do Sistema de Administração de Capital de Giro.

1.3 - IMPORTÂNCIA:

Através da utilização da estrutura genérica, conseguiu-se construir um Processador de Linguagem Natural com um formato bem diferente do usualmente encontrado na Literatura conhecida, além do fato de ter sido construído para a Lingua Portuguesa.

Apesar de mencionarmos a interface com sendo para o Sistema de Administração de Capital de Giro, na realidade, através da utilização de uma estrutura bem flexível, pode-se com algumas simples alterações, servir para qualquer outro fim.

1.4 - ORGANIZAÇÃO DO TRABALHO

O capítulo II justifica a escolha da Linguagem Natural como Interface principal do Sistema de Administração de Capital de Giro, são mostrados alguns tipos de interfaces com as suas respectivas vantagens. Uma breve noção sobre Processamento de Linguagem Natural é introduzida aqui.

No capítulo III são feitas algumas definições que serão utilizadas no trabalho, além disto são apresentados alguns tipos de gramáticas e a definição de parsers.

No capítulo IV é apresentado a metodologia de pesquisa, contendo as técnicas escolhidas para a construção do protótipo. Neste capítulo estão as estrutras de todas as etapas do Processador de Linguagem Natural.

No capítulo V é apresentado um exemplo comentado da análise de uma sentença pela interface. E por fim no capítulo VI se encontram a conclusão e as recomendações. Nos anexos estão as listagens das etapas da interface.

CAPÍTULO II - INTERFACES HOMEM-MÁQUINA

2.1 - INTRODUÇÃO

Este capítulo trata da escolha de uma interface homem-computador para o Sistema de Administração de Capital de Giro, o qual é um sistema inteligente de apoio à tomada de decisão. Este sistema incorpora conhecimentos heurísticos de um especialista a modelos matemáticos existêntes, portanto tomando o comportamento real humano na inexistência de algorítmos para melhor cumprir a tarefa proposta. Ele é construído de forma flexível e de modo que o usuário possa conduzir o sistema através de perguntas e alterações, podendo encontrar a(s) solução(ões) desejadas.

Segundo [SPER89] "uma boa interface depende dos tipos de tarefas, da prática dos utilizadores e dos seus hábitos anteriores".

2.2 - MÉTODOS DE INTERAÇÃO HOMEM-COMPUTADOR

Para a escolha desta interface é necessário, primeiramente, conhecer as características, vantagens e desvantagens de algumas interfaces julgadas as mais viáveis a serem utilizadas como neste tipo de problema estarão apresentadas abaixo :

- Linguagem de Comando;
- Menus;
- Linguagem Natural;
- Questão/Resposta.

2.2.1 - Linguagem de Comando

Neste tipo de interação, o usuário aciona uma tecla (ou um conjunto de teclas) que representem o comando desejado ou digita um conjunto de letras que represente este comando, que pode ser acrescido por um argumento. A linguagem de comando deve ter uma sintaxe coerente, para facilitar sua utilização. Os comandos geralmente são palavras de forma abreviadas.

"Os utilizadores experimentados poderão preferir um estilo de diálogo por teclas de comando ou uma linguagem de comando com uma sintaxe mais estruturada." [SPER89]. Alguns exemplos, comumente utilizados em computação, seguem abaixo:

- CLS (clear screen) - serve para limpar a tela;
- WRITE "bom dia" - escreve a frase indicada na tela;
- F2 (salvar) - tecla de função que corresponde a salvar o arquivo corrente;
- PgDn (page down) - tecla que significa pular para a próxima página;

Algumas vantagens e desvantagens para este tipo de interação [SANT90], podem ser visto abaixo:

vantagens:

- facilita a utilização do software por pessoas experimentadas, em utilização contínua, muitas vezes agilizando o processo;
- grande utilidade para comandos de muita utilização.

desvantagens:

- comando muitas vezes sem significado para o usuário;
- precisa de algum tempo de aprendizado e usuários ocasionais em geral não podem dispor deste tempo.

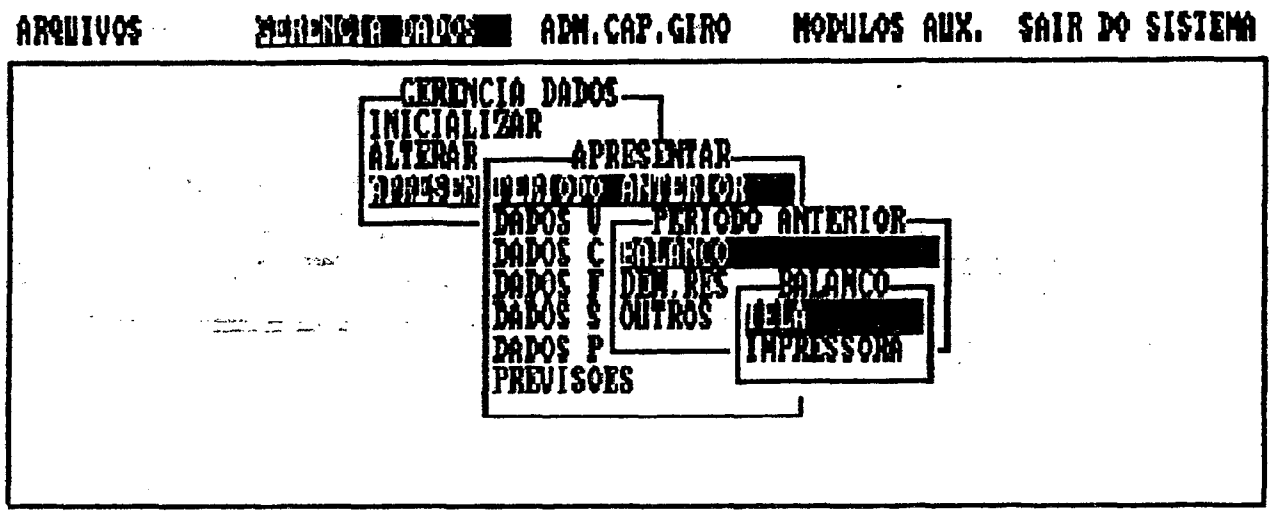


Figura 1 - Exemplo de Menu do SECAGI

2.2.2 - Menus

Neste tipo de diálogo, o usuário tem que escolher uma opção entre aquelas que lhe são propostas. A figura 1 mostra o exemplo de um menu para o Sistema de Administração de Capital de Giro.

Os menus podem ser simples (um único item é selecionado);binários (só apresenta duas alternativas, tipo sim/não); ou com seleções múltiplas (mais de um item pode ser selecionado em cada menu).

"Os diálogos por menus são muitas vezes lentos, mas fáceis de aprender. Exigem um esforço mínimo de memorização. Utilizados em excesso, enervam rapidamente e fazem perder muito tempo e paciência". [SPER89]

Segundo [SANT90], algumas vantagens e desvantagens são:

<u>vantagem:</u>

- necessita pouco treinamento, sendo indicado para usuários iniciantes ou ocasionais;
- padrão da comunicação homem-computador.

desvantagens:

- lento quando a hierarquia dos menus é complexa;
- pode ser incomoda para os usuários experts;
- estrutura pouco flexível.

2.2.3 - Linguagem Natural

Esta forma de interface, utiliza a linguagem utilizada normalmente pelas pessoas, para se comunicar com outras, como forma de diálogo com a máquina, o que proporciona muitas vantagens. Atualmente o principal meio de se comunicar com o computador é através de teclados. Sistemas com reconhecimento de voz, que seria a maneira ideal para a utilização de interfaces homem-máquina estão em um estágio ainda pouco avançado em termos de equipamentos comercializáveis.

Para o usuário, torna-se muitas vezes incomodo ter que digitar o que se deseja expressar, ao fazê-lo em Linguagem Natural através de um teclado, o usuário tende a simplificar a entrada, o que não aconteceria caso ele utiliza-se um sistema de reconhecimento de voz.

Uma variação da Linguagem Natural seria a Linguagem Natural Operativa, que é uma linguagem aplicada a área de trabalho do usuário, serve para minimizar o processo interativo, principalmente no caso da utilização de teclado. A Linguagem Operativa permite que o usuário digite somente um conjunto de palavras que possam ser entendida pelo computador ou por usuários da mesma área. O tratamento computacional para linguagem Operativa pode ser considerado o mesmo que o para linguagem Natural mais abrangente.

Algumas vantagens e desvantagens são as seguintes :

vantagens:

- o usuário não precisa aprender uma linguagem diferente, utilizando a linguagem natural do seu dia a dia;
- facilita a utilização por usuários ocasionais;
- com o desenvolvimento da tecnologia de reconhecimento e síntese de voz, muito provavelmente este tipo de diálogo será o mais utilizado em interfaces.

desvantagens:

- tecnologia pouco performante em reconhecimento de voz;
- dificuldades linguísticas;
- se o diálogo se efetua em linguagem escrita, rapidamente ele se torna cansativo;
- é difícil levar em conta o contexto e a sub-determinação do discurso;
- problemas de ambiguidade da linguagem natural.

2.2.4 - Questão/Resposta

Em geral, o software questiona e o usuário responde. A iniciativa do diálogo é do software. Este modo é utilizado principalmente em buscas de informações em banco de dados. Interfaces deste tipo funcionam de modo muito parecido com as de linguagem natural.

Na Inteligência Artificial, com a utilização de processamento em linguagem natural, é possível que o diálogo questão/resposta seja de iniciativa do usuário.

Segundo [SPER89] para os usuários iniciados ou para tarefas que um dado utilizador efetua raramente, os diálogos estandardizados e dirigidos pelo programa serão

muitas vezes mais rápidos e mais seguros do que os diálogos deixados à iniciativa do utilizador.

Além das vantagens e desvantagens da linguagem natural outras seriam as seguintes

vantagens:

- indicado para modos intermitentes, isto é, utilizado só poucas vezes, por exemplo, sistemas de informações

desvantagens:

- para utilização de modo contínuo torna-se muito cansativo.

2.3 - ESCOLHA DA INTERFACE

A escolha de uma boa interface depende dos tipos de tarefa e de usuário para o qual será voltado o sistema. Quanto ao usuário; ele é um utilizador freqüente da informática, ou um que quase não possui conhecimentos deste tipo ? É um profissional permanente ou ocasional da área ? Será que ele possui tempo e vontade para aprender a utilizar um sistema um pouco mais complexo ?.

Com relação ao problema, O Sistema de Administração de Capital de Giro, existe a necessidade de interface para as seguintes tarefas: Gerenciamento de Base de Dados, Interagir com o usuário para a Tomada de Decisões, e a de Pesquisa nas Bases de Dados e de Conhecimentos.

Pela características de Sistema de Apoio a Decisão, as entradas do sistema, na maior parte das vezes, deverá ser ou em forma de perguntas (por exemplo, "Qual o volume de vendas nos últimos 3 meses ?) ou afirmações que existam variáveis linguísticas (por exemplo, " A situação do pais está muito ruim"). Isto torna o sistema propício a utilização de Interface de Linguagem Natural.

Quanto ao tipo de usuário, como o sistema será voltado principalmente para as pequenas e médias empresas, que pelas suas características, raramente terão um utilizador frequente da informática, muitas vezes sendo o próprio dono da empresa o usuário do sistema. É necessário, portanto, a construção de uma interface dirigida para este tipo de usuário, que possui conhecimento na área do problema mas não tem tempo para aprender a utilizar um sistema mais complexo.

Devido as características do usuário e da tarefa, escolheu-se como interface principal o Processamento de Linguagem Natural, na qual inclui-se a Questão-Resposta. Deve-se ser salientado que outros tipos de interfaces estarão sendo utilizadas no sistema como Linguagem de Comandos e Menus, que podem ser vistas em [LUNA91] e nos relatórios do SECAGI. A utilização de várias interfaces é para que se possa abranger o maior contingente de usuários e tarefas possíveis, pois "o ideal (nem sempre possível) é oferecer uma gama de opções diferentes para diferentes categorias de situações ou de pessoas, como cada vez mais se faz." [SPER89]

2.4 - EVOLUÇÃO DO PROCESSAMENTO DE LINGUAGEM NATURAL

Antes de entrarmos no capítulo sobre processamento de Linguagem Natrural, propriamente dito, será dado um breve histórico da evolução do processamento de Linguagem Natural.

Segundo Feingenbuam [FEIN81], o processamento de Linguagem Natural pode ser dividido em quatro diferentes períodos históricos, está divisão foi baseada no surgimento de pesquisas e programas importantes. Em seguida veremos estes quatro períodos:

- Os programas de Linguagem Natural iniciais procuravam obter somente resultados limitados em domínios específicos e restritos. Estes programas, tais como BASEBALL de Green, SAD-SAM de Lindsay, STUDENT de Bobrow, e ELISA de Weizenbaum, utilizam estruturas de dados ad hoc para estocar fatos sobre o

domínio limitado. Sentenças de entrada são restringidas a formas declarativas e interrogativas simples e são examinadas cuidadosamente pelo programa por palavras-chaves e padrões pré-declarados que indicam objetos e relações conhecidas. Regras de domínio específico, chamadas de heurísticas, são usadas para deduzir respostas requeridas por palavras-chaves na sentença e o conhecimento na base de dados. Por causa de seus domínios sobre o assunto ser tão restrito, estes sistemas precursores são capazes de ignorar muitas das complexidades da linguagem e obter muitas vezes resultados imprecionantes na resposta da questão.

- Uma outra abordagem inicial para o processsamento de Linguagem Natural foi tentada no PROTO-SYNTHEX-I (Simmons, Burger, e Long, 1966) e Memórias Semânticas (Semantic Memory) (Quillian, 1968). Estes sistemas essencialmente estocava a representação do texto ele próprio dentro da base de dados, usando uma variedade de esquemas, indexação inteligentes para recuperar materiais contendo palavras e frases específicas. Nesta abordagem baseada em textos, os sistemas não são unidos pela sua construção de domínio específicos, desde que a base de dados textual possa cobrir vários assuntos. Contudo, eles são restritos severamente no sentido que eles possam somente responder com material que tem sido pré-estocado explicitamente. Embora mais geral que seu predecessor, estes programas tendem a falhar igualmente para implicações óbvias da sentença em base de dados, porque elas não lidam com o significado da linguagem entrada em Inglês, isto é, eles não tem nenhuma força dedutiva.
- Para abordar o problema de como caracterizar e usar o significado da sentença, um terceiro grupo de programas foi desenvolvido durante o meio dos anos 60. Nestes sistemas logicos limitados (limited-logic systems), que incluem o SIR (Raphael, 1968), TLC (Quillian, 1969), a informação na base de dados foi estocada em notações formais, e mecanismos provindos para tradução de sentenças de entrada dentro desta forma interna (análise semântica). A notação formal foi uma tentativa para liberar o conteúdo informacional da entrada para a

estrutura do inglês. A meta global destes sistemas eram para fazer inferências na base de dados para que se descubra respostas para as perguntas que não estão estocadas explicitamente na base de dados. Por exemplo, se o sistema sabe que Danger é um Doberman e os Dobermans são cachorros, então ele pode ser capaz de responder a pergunta, Danger é um cachorro? Este sistema é limitado no sentido de que deduções que eles podem fazer não são somente um subconjunto de ciasses completas de inferências lógicas usadas em conversações populares.

- O quarto grupo de programas para o entendimento de Linguagem Natural pode ser chamado de sistemas baseados em conhecimento; seu desenvolvimento é completamente entrelaçado com a pesquisa de Inteligência Artificial em representação de conhecimento. Estes programas usam uma grande quantidade de informações sobre o domínio sobre discussão para auxiliar o entendimento do conhecimento da sentença que é estocado dentro do programa usando vários esquemas de representação de conhecimento, tais como: lógica, semântica procedural, redes semânticas, ou frames (quadros).

Apesar da evolução do tratamento de linguagem Natural ter sido dividida em períodos de tempo, isto não quer dizer que as contribuições dos primeiros periodos foram esquecidas nas pesquisas atuais. Existem, hoje em dia, pesquisadores que utilizam e defendem alguns dos períodos precursores. Além da divisão, vista anteriormente, existe ainda uma vasta quantidade de métodos e técnicas, utilizados no processamento de linguagem natural, que poderia acarretar outras sub-divisões da área. No próximo capitulo serão vistos maiores detalhes sobre o processamento de linguagem Natural.

CAPÍTULO III - PROCESSAMENTO DE LINGUAGEM NATURAL

3.1 - INTRODUÇÃO

"A expressão PROCESSAMENTO DE LINGUAGEM NATURAL, ou PLN, oferece um significado global relativamente estável e significados específicos bastante variados. De uma maneira geral, PLN é o setor da Inteligência Artificial destinado a tratar textos, escritos ou falados, de natureza diversa, em qualquer língua. O tratamento pode referir-se à produção (ou sintetização) de enunciados em linguagem natural, à sua compreensão (ou interpretação), tradução, resumo, correção e classificação (ou indexação), para citar os rótulos mais freqüentes das aplicações científicas ou comerciais do PLN. De fato, tais rótulos se sobrepõem na medida em que, essencialmente, os processadores de linguagem natural são sistemas que interpretam e/ou produzem enunciados de uma língua".[SOUZ87]

Se computadores pudessem entender o que as pessoas querem dizer quando digitam (ou falam) sentenças em português, os sistemas poderiam ser mais fáceis de se usar e poderiam se adaptar mais naturalmente dentro da vida das pessoas. Além disto, pesquisadores tentam aprender como construir computadores que possam comunicar com as pessoas para extender nosso conhecimento da natureza da linguagem e da mente.

Até agora, os programas escritos tem tido bastante sucesso com processos um tanto restrito na entrada: O usuário é limitado tanto na variação estrutural de suas sentenças (sintaxe restringida pela gramática artificial) ou no número de coisas que ele pode expressar (em domínios com restrições semânticas). Vários destes sistemas são adequados para construir um ponto inicial do português para uma variedade de tarefas de processamento de dados e são comercialmente vantajosos. "Mas o uso fluente da linguagem típica humana é um tanto ilusória, e o entendimento da linguagem natural é uma área ativa de pesquisa na Inteligência Artificial."[FEIN81]

O PLN é uma atividade estreitamente ligada a posturas teóricas assumidas, por um lado, no campo da IA e, por outro, na lingüística. Em relação a Inteligência Artificial o PLN pode ser encarado, portanto, ou como um ambiente para o teste de propostas teóricas que sustentam o desenvolvimento de sistemas dedicados a interpretar e/ou produzir textos em alguma língua humana, não importando o modelo escolhido para a representação do conhecimento lingüístico ou o tipo de computação efetuada.[SOUZ87]

No que tange a lingüística, a situação é bastante similar, sofrendo apenas pequenas alterações na perspectiva. O PLN pode ser considerado uma ferramenta para o desenvolvimento da ciência da linguagem, na medida em que oferece um excelente ambiente de testes para hipóteses gerais (sobre geração e compreensão da linguagem) ou específicas (formalização de sistemas descritivos de fenômenos fonológicos, morfologicos, sintaticos, semânticos ou pragmáticos). Pode, por outro lado, ser um campo de aplicação (de provas) para descobertas realizadas na lingüística".[SOUZ87]

3.2 - PROCESSAMENTO DE LINGUAGEM NATURAL NA LÍNGUA PORTUGUESA

Ao tomar como base sistemas já desenvolvidos, surge o problema de que estes modelos são desenvolvidos em sua grande maioria para línguas diferentes da Língua Portuguesa do Brasil, a maior parte é para a Língua Inglesa. Um dos objetivos desta dissertação é o de apresentar as técnicas para o Processamneto da Linguagem Natural voltada totalmente para a Língua Portuguesa.

Segundo [OLIV87], "deve-se levar em conta que aplicar tais modelos para o processamento da Língua Portuguesa do Brasil, sem levar em conta o aspecto pragmático particular que nos é pertinente e as relações estruturais específicas da Língua Portuguesa do Brasil, dentre outros aspectos, pode e geralmente acrescenta aos trabalhos graus mais elevados de dificuldades e menores de eficiência."

"A língua portuguesa como outras línguas néo-latinas, apresenta uma rica variedade de formas e grande irregularidade morfológica em gênero, número, para

adjetivos, nomes e pronomes, bem como para formas verbais em pessoa, tempo e modo."[FRAG86] O que de certa forma, dificulta a construção de uma interface de Processamento de Linguagem Natural para a Língua Portuguesa.

3.3 - ESTRUTURA DE UMA FRASE

O passo inicial para a construção de uma interface de Processamento de Linguagem Natural, é a escolha de qual estrutura será utilizada, para isto alguns conceitos devem ser definidos previamente.

3.3.1 - Elementos Básicos da Linguagem

Alguns elementos básicos da linguagem podem ser definidos da seguinte forma:

A unidade básica da linguagem é a **palavra** ou **vocábulo**, cadeias de palavras podem formar frases.

Frase é uma unidade de linguagem que comunica um pensamento ou a intenção de uma pessoa.

A **sintaxe** da linguagem é o estudo das regras que determinam quais cadeias de palavras podem formar frases. Cadeias sintaticamente corretas são fortes candidatas a frases.

A **semântica** de uma linguagem é o estudo da atribuição de significados para as frases e palavras.

O processo de determinação de validade sintática é chamado de **análise sintática.**

O conjunto de regras que determinam as cadeias sintaticamente válidas é a **gramática** da linguagem.

Informalmente, uma gramática é um conjunto de regras que agrupam palavras em porções bem definidas de frase, que recebem nomes próprios, e as agrupam entre si para formar frases.

3.3.2 - A Frase

O que vai nos interessar, no fundo é o significado de uma frase, mas a análise sintática é essencial para o tratamento semântico no sentido de definir quais regras semânticas serão aplicadas e em que ordem.

O significado de uma frase é a lista de condições relevantes, concretas ou abstratas, que devem existir no mundo para que a frase seja verdadeira.

Existem vários tipos de frases que são eles os seguintes:

- Declarativas - *A inflação deste mês é de 22%.*
- Interrogativas - *Qual o preço do componente X?*
- Imperativa - *Apague o arquivo.*
- Exclamativas - *Ah!*
- Optativas - *Que tudo corra bem.*

obs: as frases exclamativas e optativas não têm uso para interfaces, mas as restantes são de vital importância.

3.3.3 - Fases da Análise

Para a análise da frase serão necessárias três fases: Léxica, Sintática e Semântica. A léxica é estrutura na qual estão armazenados as funções, características e propriedades das palavras, que serão utilizados nas demais fases. A análise sintática estará totalmente centrada na escolha de uma gramática apropriada. E por fim a análise semântica que dará na real o significado da sentença, necessita dos dados sintáticos da frase.

3.4 - GRAMÁTICAS

Um dos principais itens para a construção de um interpretador de linguagem natural, é a escolha de uma gramática adequada. Para isto serão apresentadas além de uma definição de gramática, algumas das gramáticas de utilização mais comum em Processadores de Linguagem Natural existentes.

Cada língua é um sistema altamente organizado e estruturado, em consequência, o saber interiorizado do falante é de uma enorme complexidade interna (problema que, em última instância, terá de ocupar a psicolingüística e a psicologia). De que modo, então, é que o falante possui este sistema interiorizado, e o põe em execução, sempre que fala? No que diz respeito a este problema, e de um modo muito simplificado, e possível colocar duas hipóteses:[RAPO79]

(a) O falante tem de memória todas as frases e estruturas possíveis da sua língua. Esta hipótese deve ser posta de lado por duas razões. Primeiro, porque o falante está sempre produzindo ou compreendendo frases novas, que nunca tinha tido contato anteriormente, e levando em conta que o número de frases e estruturas de uma língua poder ser considerado infinito, e como mesmo não acontecendo com a nossa capacidade de memória que apesar de ser enorme, é finita, tornando impossível memorizar todas as frases e estruturas existentes. Em segundo, a rapidez e homogeneidade com que uma criança adquiri o complexo sistema que é a sua língua, com base naquilo que ouvem, sugerem que se trata de um fenômeno mais complexo do que a memorização.

(b) Existe por trás da língua, de um modo não palpável, um corpo de generalizações, princípios e regras mais abstratos mas também mais simples e em número finíto, que determinam as frases de uma língua, a sua gramaticalidade, as suas propriedades e características. E considerando este corpo de regras, princípios e generalizações como sendo componentes importante do cérebro humano. A este corpo altamente organizado vamos chamar de gramática. Cada ser humano possui então uma gramática interiorizada adquirida enquanto criança num espaço de tempo relativamente

curto e possivelmente na base de alguns princípios inatos, próprios à espécie humana, "a faculdade da linguagem". Caracterizar com o máximo de rigor a gramática interiorizada que cada falante possui, é o objetivo da lingüística. A gramática deve ser formulada de um modo conciso, simples e totalmente explícito, o que não se consegue recorrendo a uma determinada (meta)linguagem formal, isto é, uma linguagem fazendo uso de símbolos devidamente interpretados e definindo operações rigorosas.

Uma gramática adequada deverá também atribuir a cada frase de uma língua uma representação semântica, isto é, uma formulação explícita, num vocabulário universal do sentido da frase, ou dos sentidos, se a frase for ambígua; uma representação fonética, isto é, representação, igualmente universal, do modo como ela é pronunciada; e uma representação estrutural sintática da frase, a sua relação com outras frases, a existência de ambiguidades que radiquem em mais do que uma possível estrutura sintática, etc. A parte sintática possui um papel central na gramática em relação tanto a parte semântica como a parte fonológica, principalmente a parte semântica não pode ser tratada independente de uma análise sintática.

Algumas informações que a representação estrutural sintática de uma frase são as seguintes:

(a) uma gramática deve dizer algo sobre a estrutura sintática das frases. Deve poder afirmar, por exemplo, que as duas frases seguintes têm a estrutura sintática principal idêntica:

cães latiram.

o João disse que a Maria o tinha beijado.

(b) uma gramática deve atribuir o mesmo sentido aos diferentes membros dos seguintes conjuntos de frases e também, por outro lado, mostrar de uma maneira explícita como é que estas frases se relacionam entre si, fato que é claramente sentido pelo falante.

que o João tenha lido esse autor surpreende-me

surprende-me que o João tenha lido esse autor

coisas interessantes ocorrem no Brasil.

ocorrem coisas interessantes no Brasil.

(c) uma gramática deve ser capaz de descrever e representar a ambiguidade de certas frases da língua:

o João trouxe tinta da China.

a velhinha ouviu o ruído da janela.

Uma gramática de uma linguagem é um esquema para especificar se a sentença é permitida em uma linguagem, indicando a regra de sintaxe para combinação de palavras dentro de frases e orações gramaticais (cláusulas) bem formadas. Em programas de processamento de Linguagem Natural, a gramática é usada em análise gramatical (parsing) para dividir em partes a sentença que foi entrada no programa para auxiliar a determinação do seu significado e por conseguinte uma resposta apropriada.

As gramáticas podem ter definições diferentes, de acordo com blocos sobre o estudo do Processamento de Linguagem Natural [SOUZ87]:

O **primeiro**, composto por investigadores comprometidos com teorias (psico)lingüísticas, que desenvolvem trabalhos no âmbito da ciência cognitiva, definem Gramáticas de maneira parecida com a das teorias lingüísticas. Os sistemas automáticos concebidos por este grupo são uma espécie de instrumento metodológicos para o teste de hipótese psicolingüísticas acerca do comportamento humano.

O **segundo** é o bloco daqueles que se concentram em formalismos e metodologias de análise sintático-semântica, por isto o significado de Gramática está mais ligado a formalismos ou recursos notacionais. A diferença entre esta noção de gramática e a anterior, é que aqui, a eficiência, perspicuidade, e consistência do formalismo ao representar uma realidade lingüística são o objetivo fundamental da pesquisa. A estes

conceitos estão ligadas as gramáticas livres de contexto, sensíveis de contexto, regulares, de estrutura frasal, redes de transiçao (aumentadas), estados finitos. Estão na área da lingüística matemática.

O terceiro grupo é o dos investigadores ocupados com a elaboração de sistemas de cunho prático, para aplicações reais, e vê a gramática como um significado próximo ao de base de conhecimento lingüístico: o acervo de regras de análise e interpretação que faculta a um sistema automático "compreender" enunciados de uma língua. Aqui se dá ênfase na viabilidade e na técnica de construção de sistemas automáticos destinados a tratar textos. O compromisso com uma corrente teórica lingüística ou psicológica não é fundamental; os formalismos são freqüentemente adaptações de modelos puros, através de heurísticas especiais.

Esta dissertação se encontra situada no terceiro grupo, pois o objetivo aqui é exatamente o de elaborar um sistema prático, utilizando uma teoria já bem definida e consolidada.

Uma das maiores dificuldades na construção de um sistema Processamento de Linguagem Natural é quanto a complexidade e flexibilidade da linguagem humana. Para facilitar o trabalho da linguagem natural é feita algumas restrições, utilizando-se um subconjunto da gramática. A seguir veremos alguns dos principais tipos de gramáticas, usualmente utilizadas no Tratamento de Linguagem Natural.

3.4.1 - Gramáticas Generativas

Uma das mais importantes contribuições para o estudo de linguagem foi a teoria das linguagens generativa introduzida por Noam Chomsky nos anos 50. A teoria foi desenvolvida em um estudo matemático não em um lingüístico, e influenciou radicalmente toda a pesquisa linguística e a ciência de computação, no desenvolvimento de linguagens de programação para computadores (linguagens artificiais). Entretanto, isto é útil na

conexão com sistemas de entendimento de Linguagem Natural, tanto em ferramentas teóricas quanto na prática.

Uma das preocupações fundamentais que presidiu à formulação da gramática generativa, foi a tentativa de descrever e, em última instância, explicar o fato de o falante de uma língua ser capaz de distinguir as sequências de palavras da sua língua que são gramaticais (que são portanto frases bem construidas da sua língua) das sequências que não são frases bem construidas (e isto independente de o falante conseguir associar um significado a algumas destas sequências). [RAPOS0]

O João ofereceu flores à Maria.

O João à Maria flores ofereceu.

A idéia básica da gramática generativa consiste na hipótese de que o falante possui um saber interiorizado sobre o seu sistema linguístico (a sua língua), o que lhe permite, entre outras coisas, reconhecer a) como pertencente a sua língua, e b) como não sendo frase da sua língua.

Uma gramática é dita generativa porque ela deverá ser capaz de enumerar (explicitamente) as frases gramaticais de uma língua, isso significa descrever os objetivos de um determinado conjunto, neste caso as frases (sequências gramaticais) de uma língua. Ao mesmo tempo a gramática deverá excluir as sequências de palavras que não são gramáticais numa língua.

Definições :

Uma linguagem formal é definida como um (possivelmente infinito) conjunto de strings de tamanho finito formado por um vocabulário de símbolos. (Ex, a string pode ser a sentença composta por um vocabulário de palavras.) A gramática de uma linguagem formal é especificada em termos dos seguintes conceitos:[FEIN81]

- As categorias sintáticas (tais como < SENTENÇA> e < FRASE NOMINAL>) são referidas como **simbolos não terminais**, ou variáveis. Notacionalmente, os não terminais de uma gramática são frequentemente indicados cercando os nomes de categorias por sinais de maior e menor, como acima.
- Os **simbolos terminais** de uma linguagem (ex, as palavras em português) são concatenados dentro de strings chamadas sentenças (se os terminais são palavras). Uma linguagem é então também um subconjunto dos conjuntos de todas as strings que podem ser formadas pela combinação de simbolos terminais em todos os caminhos possíveis. Exatamente o qual subconjunto é permitido na linguagem é especificada reescrevendo as regras, descritos a seguir.
- Um sistema de Reescrita é um sistema formal em que o alfabeto é finito e as regras de formação das expressões são pares ordenados de cadeias do alfabeto, consideradas em número finito, chamadas **regras de produções ou reescrita.**

Regras de reescrita, ou produções, especificam quais as relações entre strings de símbolos terminais e não terminais. Alguns exemplos de produção são:

< SENTENÇA> ⇒ < FRASE NOMINAL> < FRASE VERBAL>

< FRASE NOMINAL> ⇒ < DETERMINANTE> < NOME>

< DETERMINANTE> ⇒ o

< NOME> ⇒ cão

< NOME> ⇒ gato

< FRASE VERBAL> ⇒ corre

A primeira produção diz que o símbolo não terminal < SENTENÇA> pode ser reescrito como o simbolo < FRASE NOMINAL> seguido pelo simbolo < FRASE VERBAL>. O segundo permite que < FRASE NOMINAL> seja trocado por < DETERMINANTE> seguido pelo terminal < NOME>. O < DETERMINANTE> é uma string composta pela palavra o, o qual é um simbolo terminal. Os dois seguintes < NOME> permitidos podem ser trocados tanto por cão quanto por gato, desde que eles

sejam seqüências de produções permitidas na < FRASE NOMINAL>, este símbolo é dito então gerador destes dois terminais. Finalmente, < FRASE VERBAL> pode ser trocada pelo terminal corre.

- Um simbolo não terminal é distinguido e chamado de **simbolo "inicial"** ou "sentença" tipicamente denotado < SENTENÇA> ou **S**. O conjunto de strings de terminais que pode ser derivada deste simbolo distinguido, pela aplicação de sentenças de produção, é chamado a linguagem gerada pela gramática. Na gramática simples do nosso exemplo, exatamente duas sentenças são geradas:

  O gato corre

  O cão corre.

O aspecto importante de definir linguagens formalmente, do ponto de vista da lingüística computacional e processamento de linguagem natural, é que se a estrutura de sentenças do sistema é bem entendido, então um algoritmo para analisar as sentenças de entrada será relativamente fácil para escrever.

Dentro da estrutura desenhada acima, Chomsky delineou 4 tipos de gramáticas e numerou as de 0 até 3. A mais geral classe de gramática é a tipo 0, a qual não tem restrições na forma que regras reescritas possam pegar. Para sucessivos tipos de gramáticas, a forma de reescrever regras permitidas é crescentemente restrita, e as linguagens que são geradas são correspondentemente simples. As linguagens formais mais simples (tipos 2 e 3) são, como ela produz, inadequada para descrever a complexidade de linguagens humanas. Por outro lado, as linguagens formais mais gerais são dificeis de manusear computacionalmente. Existe uma íntima e interessante conexão entre as teorias de linguagens formais e complexidade computacional.

As seguintes definições dão uma breve consideração formal das diferentes restriçoes aplicadas em cada uma das gramáticas generativas.[FU82]

A estrutura frasal de uma **gramática G** é a quadrupla **G = (Vn,Vt,P,S)** no qual:

- **Vn** é um conjunto finito não vazio, chamado de vocabulário **não terminal**
- **Vt** é um conjunto finito não vazio, chamado de vocabulário **terminal**
- A união de **Vn** e **Vt** forma o vocabulário total **V** de **G**, e a interseção deles é igual a vazio.
- **P** é um conjunto finito de regras ou produções na forma

$$X \Rightarrow Y,$$

onde: **X** e **Y** são strings de **V** e em **X** existe pelo menos um simbolo de **Vn**.

- **S** pertence a **Vn** é o simbolo inicial de uma **sentença.**

3.4.1.1 - Tipo 0 (Gramáticas Sensíveis ao Contexto com Apagamento ou Recursivamente Enumerável).

Uma gramática do tipo 0 é definida como acima: um conjunto de produções sobre um dado vocabulário de simbolos com nenhuma restrição na forma de produções. "Pode ser mostrado que uma linguagem pode ser gerada por uma gramática do tipo 0 se e somente se ela possa ser reconhecida por uma máquina de Turing *, isto é, se nós pudermos construir uma máquina de Turing que detenha em um estado APROVADO para exatamente esta sentença de entrada que possa ser gerada pela linguagem."[FEIN81]

Entretanto uma gramática deste tipo é muito geral para ser usada.

* obs: Máquina de Turing é um teste clássico para determinar se uma máquina possui inteligência ao "nível humano". Neste teste são colocados um humano e um computador para serem interrogados, caso não se possa descobrir quem é quem com um mínimo de 50 % de precisão, então o computador passou pelo teste de Turing.

3.4.1.2 - Tipo 1 (Gramáticas Sensitivas ao Contexto).

Uma gramática é do tipo 1 se a forma de reescrever as regras é restrita, de modo que, para cada produção X ⇒ Y da gramática, o lado direito (Y), contenha pelo menos tanto simbolos quanto o lado esquerdo (X). Um exemplo de gramática sensitiva ao contexto com o simbolo inicial S e os terminais a, b, e c é o seguinte:

$$S \Rightarrow aSBC$$

$$S \Rightarrow aBC$$

$$CB \Rightarrow BC$$

$$aB \Rightarrow ab$$

$$bB \Rightarrow bb$$

$$bC \Rightarrow bc$$

$$cC \Rightarrow cc$$

A linguagem gerada por esta gramática é o conjunto de strings abc, aabbcc, aaabbbccc,... Esta linguagem, onde cada simbolo ocorre o mesmo número de vezes e precisa aparecer na própria posição na string, não podendo ser gerada por uma gramática de um tipo mais restrito (isto é, tipo 2 ou 3).

Uma definição alternativa (equivalente) para a gramática sensitiva ao contexto é que as produções podem ser da forma

$$uXv \Rightarrow uYv$$

onde X é um simbolo não terminal simples; u e v são strings arbritárias possivelmente vazias, dos elementos de V; e Y é uma string não vazia sobre V. Isto pode mostrar que esta restrição gera a mesma linguagem que a primeira restrição, mas esta última definição esclarece o termo sensitivo ao contexto: X pode ser reescrita como Y somente no contexto de u e v.[FEIN81]

3.4.1.3 - Tipo 2. (Gramáticas Livre de Contexto)

Gramática do tipo 2, são gramáticas na qual cada produção precisa ter somente um simbolo não terminal simples em cada lado esquerdo. Por exemplo uma gramática livre de contexto gerando as sentenças ab, aabb, aaabbb,... é:

$$S \Rightarrow aSb$$

$$S \Rightarrow ab$$

De novo, não é possível escrever uma gramática livre de contexto para uma linguagem composta pelas sentenças abc, aabbcc, aaabbbccc,...; tendo o mesmo número de cs no fim feito por linguagens mais complexas. A linguagem simples gerada aqui, por vez, não pode ser gerada por uma gramática mais restritas (tipo 3).

Um exemplo de gramática livre de contexto que pode ser usada para gerar algumas sentenças em linguagem natural é a seguinte:

< SENTENÇA> ⇒ < SINTAGMA NOMINAL> < SINTAGMA VERBAL>

< SINTAGMA NOMINAL> ⇒ < DETERMINANTE> < NOME>

< SINTAGMA NOMINAL> ⇒ < NOME>

< SINTAGMA NOMINAL> ⇒ < DETERMINANTE> < NOME> < SENT. PREPOSICIONAL>

< SINTAGMA NOMINAL> ⇒ < NOME> < SENTENÇA PREPOSICIONAL>

< SINTAGMA VERBAL> ⇒ < VERBO> < SINTAGMA NOMINAL>

< SENTENÇA PREPOSICIONAL> ⇒ < PREPOSIÇÃO> < SINTAGMA NOMINAL>

< DETERMINANTE> ⇒ a

< NOME> ⇒ João

< NOME> ⇒ porta

< NOME> ⇒ chave

< VERBO> ⇒ abrir

< PREPOSIÇÃO> ⇒ com

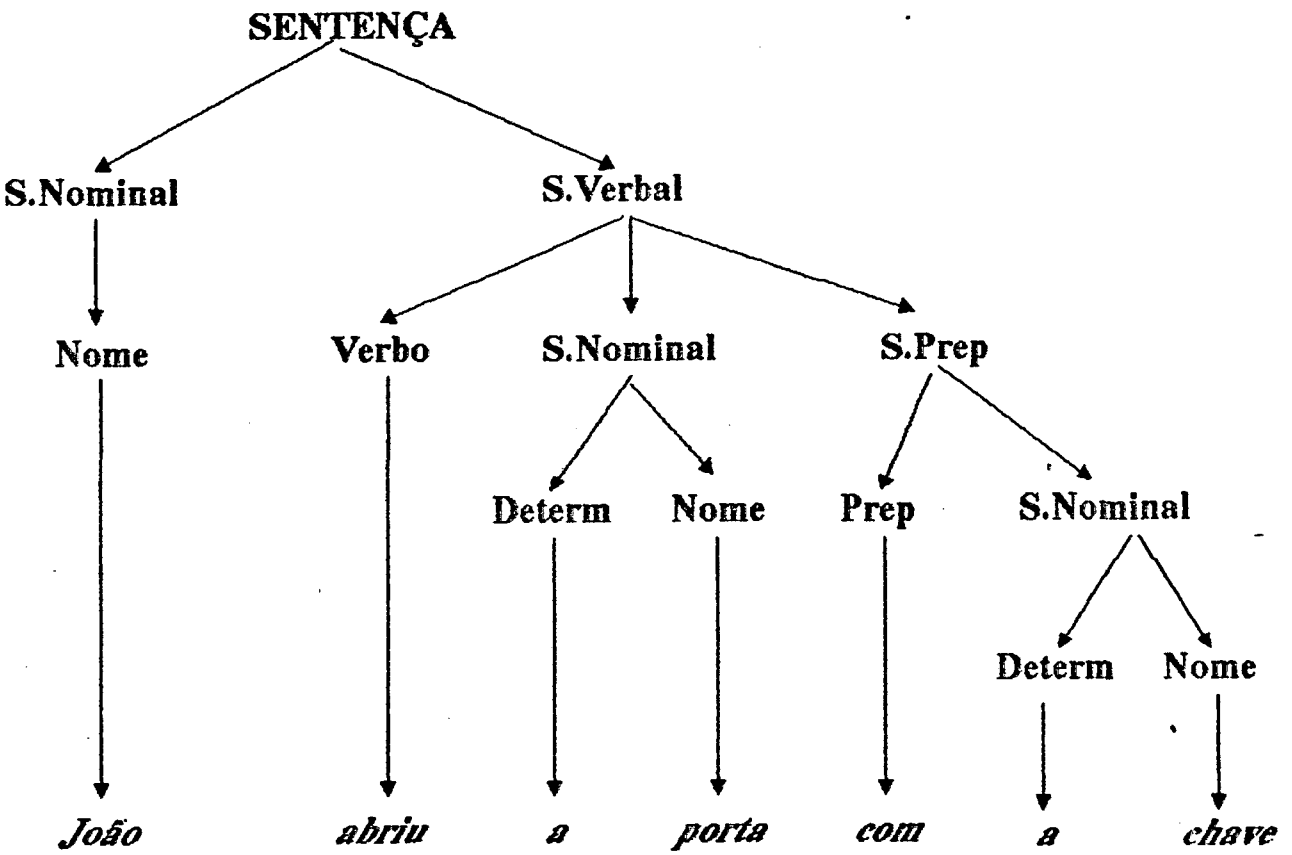


Figura 2 - Árvore de Derivação

Neste exemplo, a, João, porta, chave, abrir e com são os terminais na linguagem, e < SENTENÇA> é o simbolo inicial.

Uma importante propriedade das gramáticas livre de contexto no seu uso em programas de linguagem natural é que toda derivação pode ser representada como uma árvore, a qual pode ser pensada como uma figura da estrutura da sentença derivada. Usando a gramática anterior, a sentença "João abriu a porta com a chave" tem a Árvore de Derivação mostrada na figura 2.

É claro que, "A porta abriu o João com a chave" também é uma sentença legal nesta linguagem. Árvores de derivação podem também ser usadas com gramáticas sensitivas ao contexto (tipo 1), preparadas as produções tem-se a forma alternativa uXv⇒uYv descrita anteriormente. Gramáticas sensitivas ao contexto e livres de contexto são freqüentementes chamadas de gramáticas estruturadas em frases.

3.4.1.4 - Tipo 3 (Gramática Regular ou de Estado Finito).

Se toda produção é da forma:

$$X \Rightarrow aY \text{ ou } X \Rightarrow a,$$

onde X e Y são variáveis simples e a é um terminal simples, a gramática é do tipo 3 ou gramática regular. Por exemplo, uma gramática regular pode ser dada para gerar o conjunto de strings de um ou mais as seguidos de um ou mais bs (mas com nenhuma garantia de um número igual de as e bs):

S ⇒ aS

S ⇒ aT

T ⇒ b

T ⇒ bT

Na figura 3 tem-se a relação de cada tipo de gramática generativa entre si. A gramática do tipo 0 engloba a de tipo 1, que por sua vez engloba a de Tipo 2, que é mais genérica que a de tipo 3.

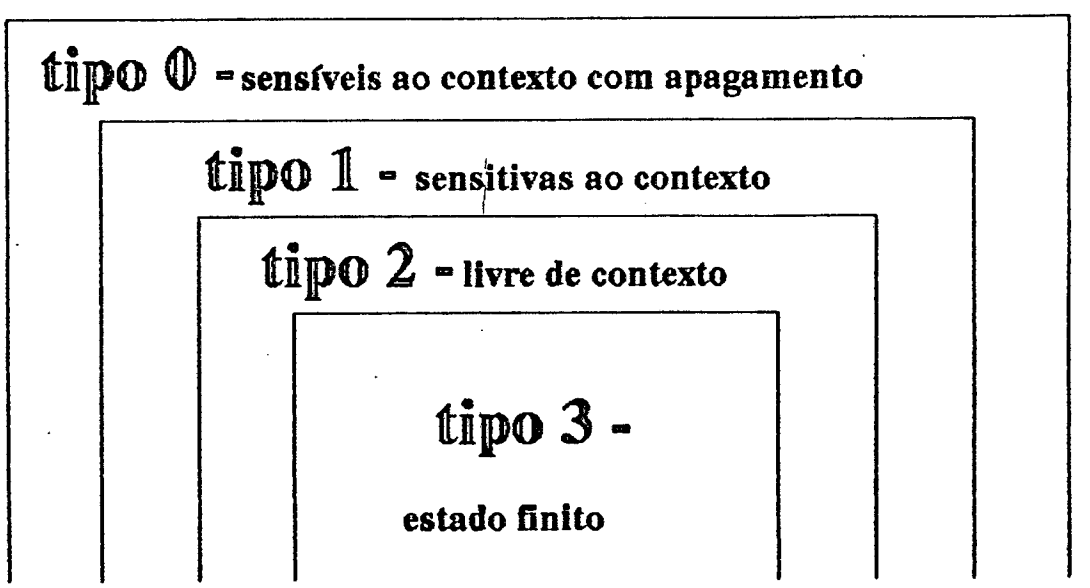


Figura 3 - Gramáticas Generativas

3.4.2 - Gramática Transformacional

Em seu trabalho publicado em 1957, Chomsky também apresenta uma classe de gramáticas chamadas "Transformacionais. Apesar de suas gramáticas de estrutura de frase (Tipo 2) refletirem aspectos fundamentais das estruturas lingüísticas, elas não conseguem reconhecer certas construções usuais das linguagens, que são fundamentais para uma compreensão global, como o uso da palavra respectivamente na frase:[MIRA86]

> *João, Maria, e Sandra são respectivamente, o pai, a mãe, e a irmã de Pedro.*

Ou ainda a simples concordância do sujeito com o verbo. Todas as regras de reescrita consistem em processos de substituição de cadeias de símbolos por outras cadeias.

Muitos fenômenos lingüísticos, no entanto, não podem ser tratados pelas gramáticas de estrutura de frase, pois não se pode expressar uma seqüência de constituintes em um único nó. Chomsky propôs uma mudança de ponto de vista, Uma árvore de derivação, ou outra representação, que somente descreve as regras que foram aplicadas para a análise ou geração de uma sentença, seria chamada estrutura superficial. Posteriormente, dois outros componentes, um transformacional e outro morfofonético, agiriam sobre esta estrutura superficial para produzir ou reconhecer sentenças da língua. Essa nova idéia deu origem as chamadas Gramáticas Transformacionais, que apresenta uma organização tripartida.

A primeira parte é uma gramática de estrutura de frase que gera cadeias de morfemas representando sentenças simples, declarativas e na voz ativa, cada qual com uma árvore de derivação correspondente ou outra representação similar. A segunda parte é uma seqüência de regras de transformação (componente transformacional) que reorganiza as cadeias e adiciona ou retira morfemas para formar a representação de um número maior de sentenças. Finalmente, uma sentença de regras morfofonéticas

(componente morfofonético) mapeia cada representação da sentença em uma cadeia de fonemas, chamada de estrutura profunda. [MIRA86]

As teorias de Chomsky foram amplamente aceitas e deram origem a um grande número de pesquisas lingüísticas. Como conseqüência principal, essas pesquisas revelaram alguns problemas com suas teorias, que por volta dos anos 60 foram seriamente questionadas. Por exemplo, se o nível sintático é mantido completamente separado do nível semântico, as gramáticas de estrutura de frase produzem sentenças desprovidas de um significado, pois uma mesma árvore de derivação pode representar mais de uma sentença com significados diferentes, como se pode ver nas duas sentenças a seguir:

- *o teste foi resolvido por um aluno aplicado.*

- *o teste foi resolvido por um método aplicado.*

Também a noção de transformações foi questionada, pois pode ocorrer que uma sentença e sua transformada não tenham necessariamente o mesmo significado, como é o caso de uma sentença declarativa e sua negação, que chegam ao extremo de espressarem idéias opostas, Outra dificuldade que pode ser encontrada diz respeito a ambigüidades. Uma sentença, tendo mais de um significado, pode produzir mais de uma sentença transformada. Por exemplo:

- *Pedro foi cortar o cabelo*

pode ser transformada nas duas sentenças a seguir:

- *Pedro é cabeleireiro e cortará o cabelo de alguém.*

- *Pedro achou seu cabelo grande e foi cortá-lo.*

Por essas e outras razões, Chomsky incorpora os resultados de outros pesquisadores em um modelo mais completo dessas gramáticas. Surge uma nova versão das gramáticas transformacionais. Esta nova versão consiste basicamente de três partes: Componente sintático, componente semântico e componente fonológico.

O componente sintático possui uma base composta de regras de estrutura de frase e um dicionário com regras de inserção léxica, e um componente transformacional. O componente sintático é chamado de gerativo, pois produz as estruturas superficial e profunda, que servem como entrada para os dois outros componentes.

O componente semântico e fonológico são referidos como interpretativos. O componente semântico interpreta a estrutura profunda gerada pelo componente sintático, fornecendo como saída a representação do significado da sentença. Já o componente fonológico interpreta a estrutura superficial gerada pelo componente sintático, resultando como saída a seqüência de sons da sentença.

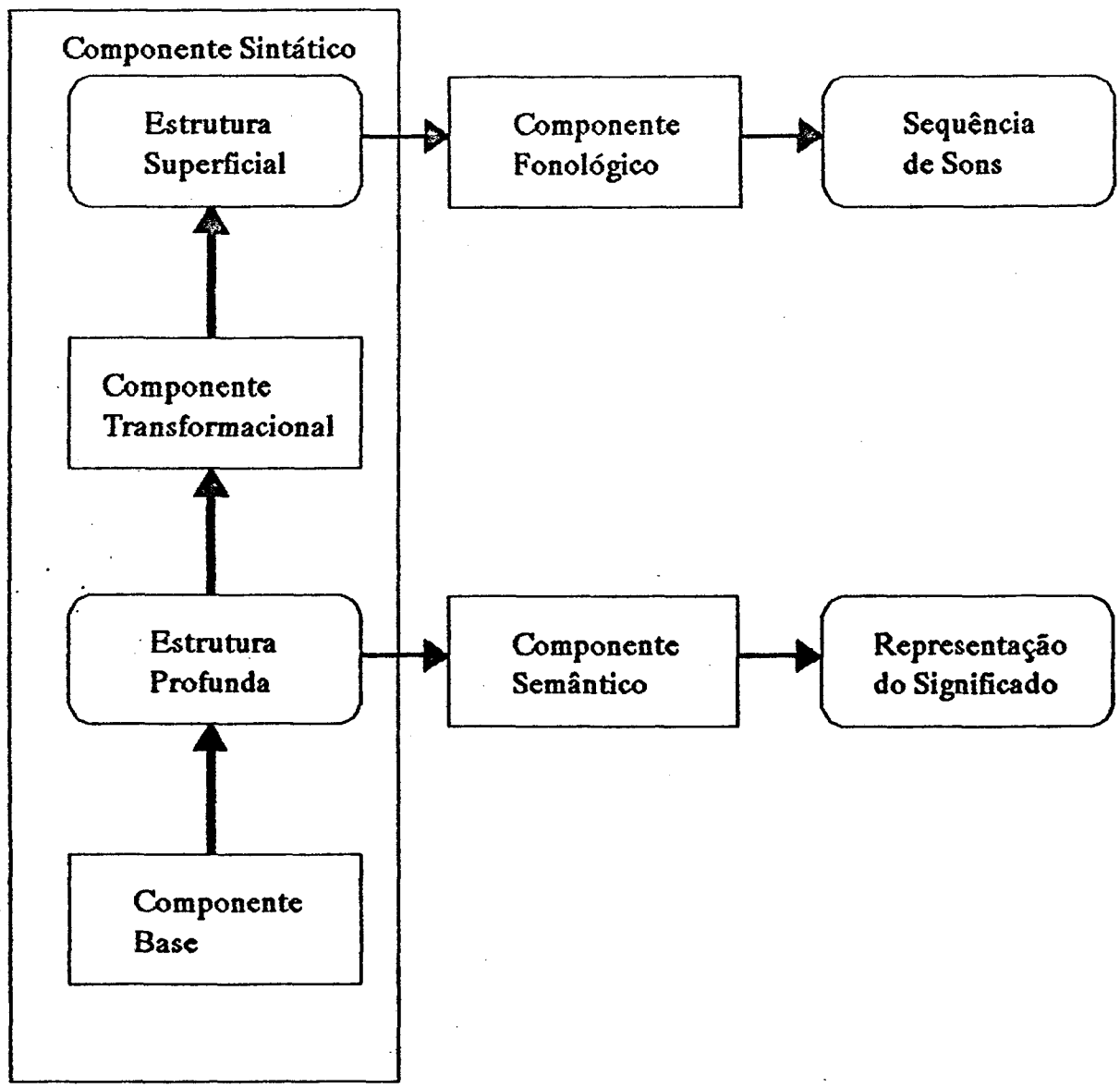


Figura 4 - Componentes das Gramáticas Transformacionais

A estrutura profunda gerada pelo componente base pode seguir dois caminhos distintos. O primeiro leva ao componente semântico, que realiza a interpretação semântica, e o segundo ao componente transformacional, que através de regras de transformação e estruturas de árvores, gera a estrutura superficial da sentença, onde podem ser então aplicadas as regras fonológicas que resultam na seqüência de sons da sentença gerada.

"A função das regras de reescrita é, no modelo Transformacional, completamente diversa da sua função no modelo sintagmático. Enquanto que neste modelo, as regras de reescrita se destinam a explicitar a organização em constituintes imediatos da frase, ou seja, a sua organização visível, aparente, superficial; pelo contrário, no modelo da gramática Transformacional as regras de reescrita têm como função gerar uma estrutura abstrata da frase, que não corresponde necessariamente à sua organização em constituintes imediatos: é a estrutura profunda. A estrutura correspondente à organização em constituintes imediatos - a estrutura de superfície -, essa, é derivada através da aplicação de regras transformacionais a partir da estrutura profunda."[RAPOSO]

3.4.3 - Gramática de Casos

As gramáticas de caso foram desenvolvidas por Fillmore em 1968, com o objetivo de solucionar alguns problemas evidentes nas gramáticas transformacionais. Tradicionalmente, as formas de casos são aplicadas a substantivos e pronomes, mostrando a relação de cada palavra com as outras palavras constituintes de uma sentença. Vê-se assim, que a noção de caso é relativa a análise morfológica e sintática das partes de uma sentença.[MIRA86]

A idéia central é que a proposição encorpada em uma sentença simples tem uma estrutura profunda consistida de um verbo (componente central) e uma ou mais frases nominais. Cada frase nominal é relacionada com um verbo em um relacionamento particular. Este relacionamento é chamado de Caso. Por exemplo na sentença

- João abriu a porta com a chave.

João pode ser o agente do verbo abriu, a porta pode ser o objeto, e a chave pode ser o instrumento. Para a sentença

- A porta foi aberta por João com a chave

a designação de caso pode ser a mesma, mesmo que a estrutura tenha sido mudada.

É importante na teoria de Fillmore que o número de relacionamento de casos seja pequeno e fixo. Fillmore (1971) propôs os seguintes casos:

- Agente: o instigador de um evento;
- Contra-agente: a força ou resistência contrária ao que a ação está executando;
- Objeto: a entidade que move, muda ou cuja posição ou existência está sendo considerada;
- Resultado: a entidade que surge como um resultado de uma ação;
- Instrumento: o estímulo ou causa física imediata de um evento;
- Origem: o local de onde alguma coisa se move;
- Destino: o local para onde alguma coisa se move;
- Receptor: a entidade que recebe, aceita, sofre ou experimenta os efeitos de uma ação.

Os verbos são classificados de acordo com os casos que podem ocorrer com cada um deles. O conjunto de casos que o verbo aceita forma uma estrutura ordenada chamada case frame. Por exemplo, o verbo abrir teria o seguinte case frame:

[objeto (instrumento) (agente)]

indicando que o objeto é obrigatório na estrutura profunda, mas é permitido omitir o instrumento (João abriu a porta), o agente (a chave abriu a porta) ou ambos (a porta

abriu). Assim, os verbos fornecem um padrão no qual o restante da sentença pode ser entendido, facilitando a construção de analizadores gramaticais dirigidos por espectativas.

3.4.4 - Gramáticas de Claúsulas Definidas

O formalismo das gramáticas de Claúsulas Definidas foi primeiramente descrito por Colmerauer, em 1975, a partir da idéia de generalizar o formalismo das Gramáticas Livre de Contexto, expressando-o em Claúsulas da Lógica de Predicados de Primeira Ordem.

"As gramáticas de Claúsulas Definidas são uma extensão natural das gramaticas Livre de Contexto. As gramáticas Livre de Contexto não são completamente adequadas para descrever as linguagens naturais, nem mesmo muitas artificiais. As extensões apresentadas pelas gramáticas de Claúsulas Definidas tornam-as um formalismo claro e poderoso tanto para escrever linguagens naturais quanto artificiais." [MIRA86]

Uma gramática de Claúsulas Definidas tem a mesma forma geral de uma gramatica Livre de Contexto, mas aqui os símbolos não terminais pode ter argumentos como em

SentNominal(X,Ct,Y) ⇒ Det(X1,Z),Nome(X2,Y)

com os argumentos representando constantes ou variáveis de uma metalinguagem auxiliar usada para falar sobre o processo de análise sintática. Note que o mesmo símbolo de variável em símbolos não terminais diferentes representa, ou unifica com, um mesmo valor.

Segundo [GEET86], com as gramáticas de Claúsulas Definidas pode-se:

- Prover a gramática de dependência contextual, de tal modo que a forma permitida para um sintagma pode depender do contexto no qual o sintagma ocorre na sentença.

- Construir estruturas de árvore durante o processo de análise da sentença, apesar das estruturas recursivas da gramática, com o auxílio de variáveis lógicas.
- Adicionar condições extras nas regras gramaticais, caso sejam necessárias computações auxiliares para a análise da sentença.

3.4.5 - Gramáticas de Redes de Transição Aumentadas

Foi desenvolvido, em 1970, por Woods e não se trata de uma gramática propriamente dita, mas um formalismo poderoso e claro, no qual pode-se expressar uma gramática para reconhecimento de linguagem.

"São automatos de pilha (máquina s abstratas que processam gramáticas livres de contexto) acrescidos do potencial de condicionamento (se determinada condição é verdadeira na passagem de um a outro estado/nódulo, podem-se efetuar computações sobre a estrutura)." [SOUZ87]

"Estas redes são uma extensão da **Rede de Transição Recursiva** (RTN), a qual consiste de nós e arcos rotulados. Estes arcos podem estar referidos a outras redes, não somente a nós. Pode-se usar **Redes de Transição Aumentadas** (ATN) para coletar informações sobre a estrutura gramatical com contrução afastada e para testar características (assim como número e gênero) das palavras em uma sentença para detectar estruturas não sintática. Esta rede também auxilia a análise semântica." [GEET90]

"Por suas características não determinísticas, as Redes de Transição Aumentadas podem consumir muito esforço para o reconhecimento de sentenças que exijam uma grande quantidade de "backtracking". A medida que a gramática representada pela Rede de Transição Aumentada se torna mais ampla para capturar um maior número de construções lingüísticas da linguagem, mais difícil se torna evitar o "backtracking" necessário para a análise das sentenças, especialmente daquelas que contêm ambiguidades."[MIRA86]

3.5 - SINTAXE X SEMÂNTICA

Assim como ocorre no caso da Gramática, estas duas noções nem sempre coincidem na visão de cientistas da linguagem e cientistas da computação. De maneira geral, as diferenças estão relacionadas às distinções entre a teoria da linguagem natural e a teoria das linguagens formais.

"Para a teoria lingüística, **Sintaxe** é a parte da gramática que descreve as regras pelas quais combinam-se as unidades significativas das sentenças. A **Semântica** trata da representação do significado dos enunciados. A teoria semântica deve explicar as regras gerais que condicionam a interpretação dos enunciados. Embora alguns teóricos como Montague, proponham que as linguagens formais e a linguagem natural podem ser tratadas pelo mesmo modelo, esta não é uma visão compartilhada pela maioria, sobretudo dentre os linguístas."[SOUZ87]

Nas linguagens artificiais, a sintaxe é totalmente independente da semântica. Porém, a linguagem natural oferece um sem-número de evidências de que semântica e sintaxe são interdependentes, embora não integralmente. Dentre tais indícios, citem-se as noções semânticas de "indivíduo" e "objeto", presentes em classes sintáticas como a de Substantivo. Vejam-se, também, fenômenos como a alteração do comportamento sintático de verbos como "morrer", devido a características semânticas, aceitando a construção "As plantas têm morrido", mas não aceitando "João tem morrido".[SOUZ87]

As observações anteriores reforçam o desafio em que se constitui o processamento automático de linguagem natural. Os algoritmos computacionais disponíveis, e utilizados necessariamente em algum nível da análise, são inspirados na teoria da compilação, para a qual, se a definição de sintaxe pode permanecer a mesma, ou pouco alterada, a definição de semântica é bastante diversa.

"Para as línguas humanas, o paralelismo dos processos de análise sintática e semântica motivaram várias revisões de propostas de modelos de inspiração formal. Tal

é o caso da hipótese lexicalista de Chomsky (1972), revendo suas posições anteriores (1957, 1965), onde a independência dos componentes sintático e semântico era resguardada. Entretanto, no desenvolvimento de sistemas reais destinados a tratar automaticamente a linguagem natural, o paralelismo ainda é uma aspiração, e não um fato. Assim é que os processadores de linguagem natural existentes, em sua maioria, repousam sobre princípios semelhantes ao da compilação. Isto é especialmente verdade no caso dos sistemas dedicados ao processamento de consultas em linguagem natural a bases de dados. Tais sistemas têm por função primordial traduzir perguntas formuladas em uma língua em comandos de uma linguagem artificial de consultas, objetivando um comportamento cooperativo com o usuário. Esta característica depende muito mais de recursos pragmáticos, que nitidamente extrapolam a função básica."[SOUZ87]

3.6 - ANÁLISE SINTÁTICA (Parsing)

Analisar sintaticamente uma sentença de um linguagem natural, tal como o português, é o processo de determinar se ela é gramaticalmente bem formada e, nesse caso, de descobrir uma ou mais estruturas que decodificam informações utilizáveis de algum tipo sobre isto. A palavra é derivada do latim pars orationis (parte da fala) e reflete um processo que tem sido cuidado pelos humanos dos tempos medievais até o presente. Esta atividade tradicionalmente pegava a forma de designar uma parte da fala para todas palavras em uma dada sentença, de determinação das categorias gramaticais de palavras e frases, e da enumeração das relações gramaticais entre palavras. Este suposto era pedagógico para ajudar estudantes de uma linguagem a aumentar seu domínio sobre ela.

"**Parsing** é a delinearização da entrada lingüística, isto é, o uso de regras gramáticais e outras fontes de conhecimento para determinar as funções das palavras na sentença de entrada (uma string linear de palavras) de modo a criar uma estrutura de dados mais complicada, por exemplo, uma árvore de derivaçao. Esta estrutura descreve várias das relações entre palavras na sentença ("este adjetivo modifica aquele nome, o qual é o objeto da frase preposicional...") e pode ser usado para pegar o "significado" da

sentença. Todo sistema computacional de processamento de linguagem natural contém um componente de análise gramatical (parsing) de algum tipo, mas os programas Linguagem Natural antecessores eram baseados em palavras-chaves esperadas na entrada ou eram restringidos pela grande limitação das estruturas das frases. A aplicação prática das gramáticas para todo o escopo da Linguagem Natural é comprovado difícil". [FEIN81]

O desenvolvimento de um analisador sintático é um problema, tanto na teoria quanto na implementação. A primeira parte do desenvolvimento diz respeito a especificação da gramática a ser utilizada. O resto do sistema de análise sintática é relacionado com o método do uso da gramática, isto é, a maneira na qual string de palavras é juntada de acordo com o padrão da gramática. Estas considerações chocam-se com muitas das questões gerais da ciência dos computadores e Inteligência Artificial pertencendo a controles de processos e manipulações de estruturas de dados representacionais.

Nos tempos modernos desenvolvimentos em lingüística e ciência dos computadores tem levado para um conjunto diferente de atividades sendo associadas com o termo "parsing". A disponibilidade de computadores fez com que, parte dos procedimentos específicos que eram cuidados por humanos, fossem trocados por algoritmos bem específicos que são cuidados por máquinas. Tambem, a mudança correspondente na natureza das descrições estruturais produzidas. Concepções pedagógicas foram recolocados por um requerimento, cujas descrições estruturais refletem o significado das sentenças. Esta é uma importância especial nas aplicações de IA, na qual a intenção da sentença entrada pode ser entendida e atacada dentro de uma maneira apropriada.

Existem outras mudanças retida pelo uso de sistemas formais para modelar aspectos da linguagem natural. Em particular, muitos tipos diferentes de gramáticas formais e geradora tem sido criadas para especificar as sentenças de uma linguagem e

juntar cada sentença com um conjunto correspondente de descrições estruturais. A natureza destas gramáticas, entretanto, usualmente não proporciona um algoritmo óbvio para descrições estruturais de computação para uma sentença. Neste aspecto eles são similares a sistemas de lógicas, o qual implicitamente especifica um conjunto de teoremas prováveis, mas não fala explicitamente como faz para um teorema particular ser provado. Somente procedimentos provados podem ser utilizados por sistemas lógicos, assim como procedimentos sintáticos por gramáticas formais da linguagem natural.[PETR87]

3.7 - REPRESENTAÇÃO DE CONHECIMENTO PARA PLN

"A idéia de Representação de Conhecimento em Processamento de Linguagem Natural, existe desde os primeiros conceitos de máquinas tradutoras, na década de 40. Isto ocorreu na idéia de Weaver para tradução de uma língua A para uma língua B, primeiramente a língua A seria traduzida para uma lingua universal intermediária, compartilhada por todos os homens, e depois desta língua universal para a língua B. Essa idéia de uma Representação intermediária é chamada, nos modernos sistemas de IA, como Representação de conhecimento."[FEIN81]

"Existem dois grupos de representação de conhecimento, o **declarativo** e o **procedural** ou **programático**. A opção entre uma técnica e outra está ligada à utilização de máquinas de inferência lógicas (semelhantes a provadores de teoremas), manipulando informações axiomatizadas, e à utilização de procedimentos heurísticos, trazendo informações programadas.

Representações procedurais são freqüentemente o caminho mais direto para implementar os passos de raciocínio específico necessário para um sistema de Linguagem Natural. A maioria dos sistemas de trabalho reais, que tem sido desenvolvido, tem se utilizado bastante de uma representação procedural especializada.

Os esquemas de Representação Declarativas mais influentes são Lógica e Redes Semânticas. A representação por Redes Semânticas foi primeiramente proposto por

Quillian (1968) como um modelo para memória associativa humana. Ele aplicou os conceitos de teoria de grafo, representando palavras e significados como um conjunto de nodos ligados implementados como estruturas de dados no programa de computador. Pelo uso de um conjunto sistemático de tipos ligados, Quillian foi capaz de operar programas simples (tais como as referidas cadeias de ligação) que correspondem a fazer inferências.

A vantagem da Redes Semânticas sobre a Lógica padrão, como esquema de representaçào, é que vários conjuntos selecionados de inferência, daqueles que são possíveis, podem ser feito em caminhos especializados e eficientes. Se estes correspondem a inferências que pessoas fazem naturalmente, então o sistema será capaz de fazer um tipo mais natural de raciocínio que pode ser facilmente alcançado por deduções lógicas formais. Redes Semânticas tem sido a base para representação de conhecimento em um grande número de sistemas. Recentemente tem sido muito divulgado trabalhos que formalizam noções de rede, assim como existe uma correspondência clara entre operações de grafos e as semânticas formais das declarações representadas.

Representação de Casos acrescentam às noções básicas de Redes Semânticas, a idéia de um Frame (Minsky, 1975), onde são agrupadas as propriedades de um objeto ou evento dentro de um conceito singular. Mas como em Redes Semânticas, a vantagem da Representação de Casos é de agrupar conjuntos relevantes do relacionamento dentro de estruturas de dados simples. Enquanto Representação de Caso lida primariamente com sentenças simples ou ações, em Frames são aplicados soluções integrais, objetos complexos, ou séries de eventos. Na análise de uma sentença, um sistema de entendimento de linguagem baseado em Frames tenta ligar a entrada com um conjunto de objetos ou eventos do seu domínio, que estão estocados na sua base de dados. [FEIN81]

CAPÍTULO IV - METODOLOGIA DA PESQUISA

4.1 - INTRODUÇÃO

O objetivo do sistema, desenvolvido nesta dissertação, é o de servir como uma parte intermediária entre o usuário e o sistema de Administração de Capital de Giro, fazendo com que o usuário obtenha a "resposta" desejada, como mostrado na figura 5.

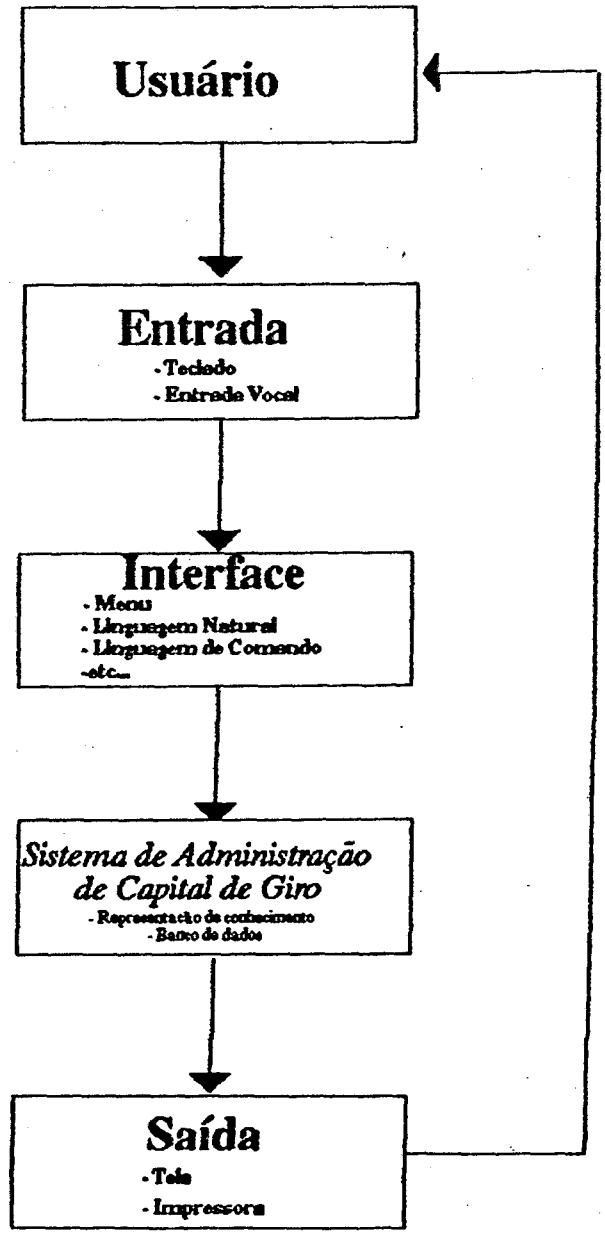


Figura 5 - Sistema de Administração de Capital de Giro

Para a construção da Interface de Processamento de Linguagem Natural proposta aqui, dividiu-se o trabalho em quatro fases distintas:

- **Análise léxica** - nesta parte é definida a estrutura léxica que o sistema proposto utiliza, contendo informações sintáticas e semânticas. Além disto, esta fase possui também a função de ler uma sentença e transforma-la em uma lista de palavras. As palavras tem que pertencer a base de dados que contem a estrutura léxica, ou ser sinônimo de alguma outra existente.

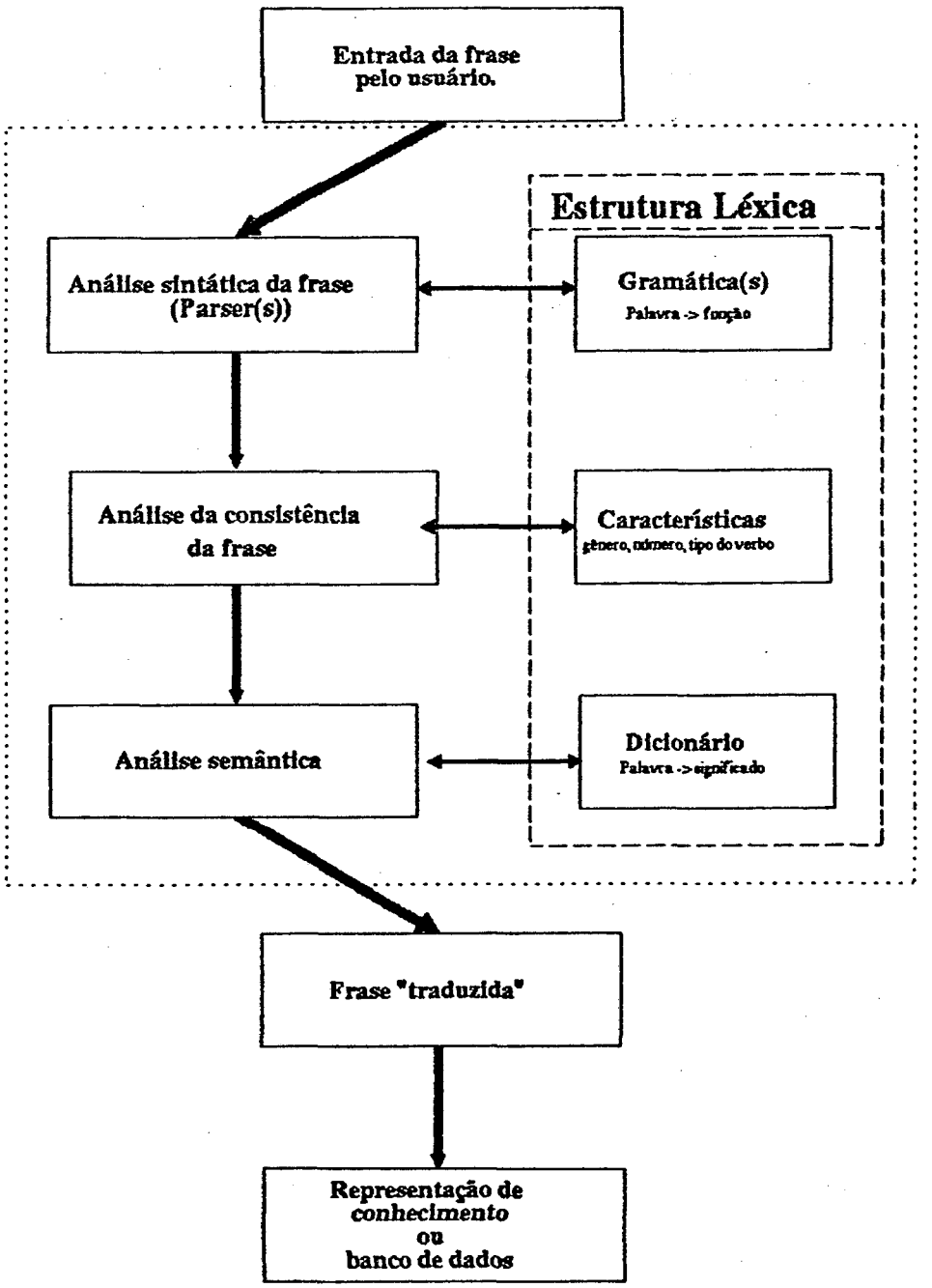


Figura 6 - Estrutura do Processador de Linguagem Natural

- **Análise Sintática** - é feita a escolha da gramática que se irá trabalhar. Apartir desta escolha é verificada a compatibilidade da estrutura da sentença com a gramática escolhida, isto é, se as palavras estão posicionadas em seu devido lugar. Para se fazer está análise, busca-se a função da palavra na base de dados léxica. Caso exista algum tipo de erro, é enviado uma mensagem para que o usuário possa corrigir a entrada da frase.
- **Consistência da frase** - nesta fase são investigados três tipos de concordância: de número, de gênero e do verbo. Na concordância de número verifica-se a consitência de número da sentença. No tipo seguinte, de modo semelhante, o gênero é verificado. Na ultima, a análise é quanto a transitividade do verbo.
- **Análise semântica** - a função desta fase é a de transformar a sentença em uma seqüência de códigos compatíveis ao sistema computacional em questão. É feita a escolha de uma estrutura para representação semântica, que trata-se de uma estrutura de frame centrada no verbo principal da sentença.

4.2 - SISTEMA DE ADMINISTRAÇÃO DE CAPITAL DE GIRO:

Como já foi dito anteriormente, o sistema que está sendo proposto aqui faz parte de um sistema maior, que é o Sistema Apoio a Decisão para Administração de Capital de Giro, que está sendo desenvolvido no Grupo de Inteligência Artificial do PPGEP da UFSC, da qual o autor desta dissertação faz parte.

No Sistema de Administração de Capital de Giro foi desenvolvida uma estrutura flexível, que visa contornar alguns problemas de modularidade, nesta estrutura qualquer novo módulo que entre no sistema pode utilizar de outros existentes. Mas para isto ele precisa estar em um formato pré-definido.

Por exemplo, banco de dados principal com a estrutura léxica está colocada em uma base de dados externa com um formato semelhante aos utilizados para os demais dados do Sistema de Apoio à Decisão para a Administração de Capital de Giro (ver maiores

detalhes em [LUNA91]). Além disto, esta estrutura utilizando base de dados externa tem a vantagem de não ocupar memória de trabalho do computador, cada dado necessário colocado nela é buscado externamente, por isto a quantidade de informações é limitada somente pelo tamanho do disco utilizado (rígido ou flexível).

Principalmente por ser um sistema de Inteligência Artificial, foi escolhida a linguagem de programação Prolog como linguagem de implementação da estrutura flexível do sistema protótipo. No PPGEPS - UFSC, tem se disponível o Turbo Prolog da Borland, que é portanto o interpretador escolhido para a construção do processador de linguagem natural. O interpretador poderá rodar em micro computadores IBM-PC com o mínimo de 640 Kbytes de memória RAM.

Apesar de ser recomendável, quando na construção e desenvolvimento de sistemas computacionais, que a escolha da linguagem de programação seja feita depois que a estrutura do problema esteja pronto, no nosso caso a escolha se encaixa perfeitamente. Pois, as linguagens de programação são projetadas para resolver problemas em áreas quase sempre bem definidas.

Existem linguagens que são voltadas para a manipulação numérica, utilizadas em problemas procedurais, tais como Pascal, C, Fortran, etc. E existem outras que são voltadas mais para a manipulação simbólica, utilizadas na Inteligência Artificial, tal como o Lisp e o Prolog.

O Prolog ainda tem a vantagem de que foi criado com o objetivo de servir de apoio a sistemas de processamento de linguagem natural. No Prolog os fatos e claúsulas são baseados na Lógica de Predicados de Primeira Ordem e tem um conjunto definido de Claúsulas Condicionais ou Incondicionais (chamada claúsula de Horn).

4.3 - ANÁLISE LÉXICA:

Esta fase do processamento de linguagem Natural tem a importância de separar a frase entrada pelo usuário, em seus vários vocábulos ou constituintes terminais. Junto com os vocábulos estarão as informações sintáticas e semânticas, que irão ser úteis para o reconhecimentos dos vocábulos nas próximas fases. Para se fazer o reconhecimento das palavras, é necessário primeiro definir uma estrutura léxica de base de dados, a qual contém todos os dados necessários para o decorrer da análise da sentença.

Uma fase inicial é referente a separação das conjunções que é de vital importância, para a análise sintática da frase. Um predicado (predicado, de uma forma simplificada, é como se é chamado uma variável de função no Prolog) chamado conjunção é da seguinte forma:

conjunçao (palavra-conjunção, primeira parte, segunda parte).

Exemplificando, fica da seguinte forma:

conjunção (no , em , o) ou,

conjunção (da , de , a)

Esta conversão é necessária, pois cada conjunção contem dois constituintes terminais.

As informações referentes a cada vocábulo ou constituinte terminal, que serão utilizadas nas fases posteriores de Análises Sintática e Semântica, são as seguintes:

Nome, Tipo, Origem, Número, Gênero, Lista de Propriedades, Lista de Sinônimos.

O Nome refere-se à própria palavra. O Tipo é a classe gramatical que a palavra pertence (artigo, preposição, nome, etc.). A Origem é uma raiz da palavra, tem uma utilidade maior quando o Tipo é verbo, de modo a representar diferentes conjugações verbais com uma única palavra. O Número indica se a palavra está no plural, singular ou neutro; enquanto que o Gênero refere-se a ser masculino, feminino ou neutro.

A Lista de Propriedades contém todas as características semânticas que se referem a palavra, tais como ser um objeto, lugar, humano, instrumento, etc. Associado a cada propriedade está também um número real, visando uma futura implementação de algum tipo de tratamento de incerteza.

E por ultimo a Lista de Sinônimos, a qual contém palavras que tenham o mesmo significado e também formas de escritas alternativas ou possuindo algum tipo de erro. Esta lista terá uma função extra de aprendizado do sistema, pois quando uma palavra não é encontrada durante a análise, um sinônimo da palavra é pedido ao usuário. A palavra que ainda não existia na estrutura léxica é acrescentada na Lista de Sinônimos da ultima palavra entrada pelo usuário. Da mesma forma que a Lista de Propriedades, também tem-se associado um número real, para ser utilizado em uma futura expansão do sistema.

Exemplos de alguns vocábulos podem ser visto abaixo:

PALAVRA : A TIPO = PREPOSICAO
ORIGEM = A NUMERO = SINGULAR
GENERO = COMUM

PALAVRA : A TIPO = DETERMINANTE
ORIGEM = A NUMERO = SINGULAR
GENERO = FEMININO

PALAVRA : LUCRO LIQUIDO TIPO = NOME
ORIGEM = LUCRO LIQUIDO NUMERO = SINGULAR
GENERO = COMUM
PROPRIEDADE = OBJETO GRAU = 0.9
PROPRIEDADE = PALAVRA DUPLA GRAU = 1.0
SINONIMO = LUCRO LIQIDO GRAU = 0.6

PALAVRA : E TIPO = CONJUNCAO ADITIVA

ORIGEM = E NUMERO = INDETERMINADO

GENERO = NEUTRO

PALAVRA : VENDAS TIPO = NOME

ORIGEM = VENDA NUMERO = PLURAL

GENERO = FEMININO

PROPRIEDADE = OBJETO GRAU = 0.9

Para verbos existem algumas pequenas modificações, é excluido o gênero, e acrescentado: Tempo, que pode ser presente, passado ou futuro; Transitivo (sim ou não). Como mostrado abaixo:

PALAVRA: APRESENTAR TIPO = VERBO

ORIGEM = APRESENTAR NUMERO = SINGULAR

TEMPO = PRESENTE TRANSITIVO = SIM

SINONIMO = ACONTECE GRAU = 0.60

SINONIMO = IMPRIMIR GRAU = 0.60

SINONIMO = ESCREVER GRAU = 0.80

SINONIMO = APESENTAR GRAU = 0.60

PALAVRA: AUMENTAR TIPO = VERBO

ORIGEM = AUMENTAR NUMERO = SINGULAR

TEMPO = PRESENTE TRANSITIVO = SIM

SINONIMO = SUBIR GRAU = 1.00

Como complemento destas informações para cada verbo, o léxico tem uma estrutura de frame que define as suas diferentes propriedades semânticas. O verbo agirá como um conceito central para a análise semântica. A estrutura de frame fornece uma relação de outras partes da sentença com o verbo.

A estrutura léxica aqui utilizada possui três tipos de informações associadas ao verbo. Um primeiro tipo de informação consiste em uma lista de funções com presença obrigatórias junto a determinado verbo. A outra, trata-se de uma lista com funções opcionais, a cada função está associada uma palavra default, isto é, palavra que virá a ser utilizada no caso da inexistência da função na sentença. E por ultimo, se o verbo é de entrada ou saída.

Estrutura léxica para o verbo apresentar:

frame_verbo ("APRESENTAR",["OBJETO"],[["LOCAL","TELA"]],"SAIDA")

Exemplo de sentenças utilizando o verbo apresentar:

"APRESENTAR O DEMONSTRATIVO DE RESULTADO NA IMPRESSORA"

ou;

"POR FAVOR APRESENTE O LUCRO LÍQUIDO"

Na última sentença o local de saída, pela estrutura do frame, será a tela.

4.4 - ANÁLISE SINTÁTICA

O primeiro passo para se fazer a análise sintática é escolher, entre as diversas existentes, qual gramática utilizar. Para em seguida montar uma estrutura compatível com a escolha, são apresentados dois tipos de estruturas, a primeira apresenta a gramática embutida dentro da programação, já na outra procurou-se fazer de uma forma bem flexível, de modo que a escolha da gramática não tenha uma influência decisiva, podendo ser alterada facilmente sem prejuízos muito grande ao sistema.

Com o uso do Prolog como linguagem de programação, a gramática de Claúsulas Definidas tornou-se uma ótima opção para a análise sintática. Esta gramática é um método de expressar regras livres de contexto. Poderia ter-se escolhido Redes de Transição Aumentada, que também processam gramáticas livre de contexto, e também tem sido

muito utilizada em sistemas já desenvolvidos. Várias referências bibliográficas, indicam como sendo a melhor escolha para trabalhar junto com o Prolog, a gramática de Claúsulas Definidas.

As vantagens da escolha da gramática de Claúsulas Definidas, são segundo [GEET90] e [MIRA86]: o suporte para dependência contextual, a capacidade de deixar estruturas serem construidas durante a analise gramatical, e a capacidade de deixar condições extras serem especificadas nas regras gramaticais. Além disto possui uma completa integração com o Prolog e facilitará a análise semântica posterior.

4.4.1 - Gramática:

A relação entre os constituintes terminais da sentença (que são aqueles - como definido em 3.4.1 - que representam uma palavra) para a gramática escolhida, está mostrado na figura 7, de modo simplificado (por exemplo: depois de um nome pode vir outro nome, um adjetivo, uma preposição, etc; depois de um adjetivo pode vir um verbo, uma preposição e assim por diante).

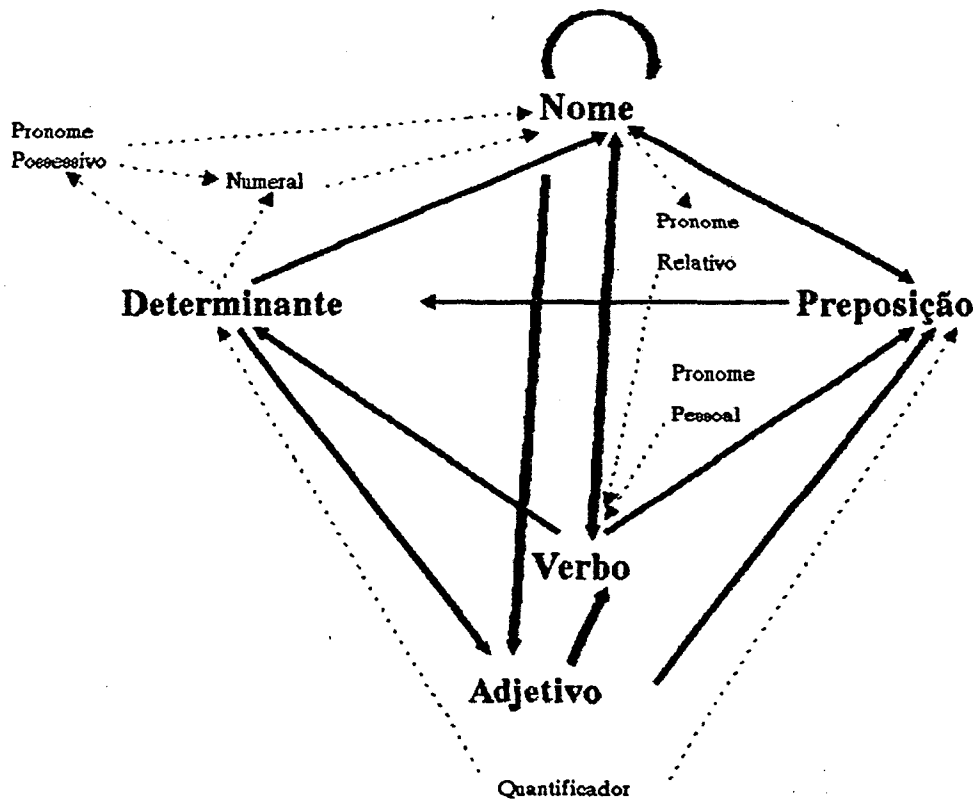


Figura 7 - Relação entre os Constituintes Terminais

As estruturas gerais para uma frase, começa pela definição do constituinte inicial, ou seja, a Sentença : um primeiro tipo composto por sintagma nominal seguido por sintagma verbal, que é a maneira normalmente utilizada na língua portuguesa, por exemplo "A inflação foi de 20%" , sendo a sentença nominal = "a inflação" e a sentença verbal = "foi de 20%". Uma outra forma é composta apenas pelo constituinte sintagma verbal, algo como "Faça isto". Um ultimo modo, serve para sentenças compostas de mais de uma oração, por exemplo: "se a inflação foi de 20 %, então faça isto".

SENTENÇA = SINTAGMA NOMINAL + SINTAGMA VERBAL;

SENTENÇA = SINTAGMA VERBAL;

SENTENÇA = SENTENÇA + SENTENÇA

Um outro exemplo seria o da estrutura sintagma nominal, que pode formar frases de quatro tipos: a primeira composta pelas que contêm determinante, nome e complemento ex: "os preços altos", sendo o determinante = "o", o nome = "preços" e o complemento (adjetivo) = "altos". O seguinte é composto por determinante mais nome, por exemplo: "todos os preços", onde sentença determinante = "todos os" e nome = "preços". Tem-se no terceiro somente nome e complemento, enquanto que no ultimo a sentença nominal é constituida somente de nome.

SN = DETERM + NOME + COMPLEMENTO DO NOME;

SN = NOME + COMPLEMENTO DO NOME;

SN = DETERM + NOME;

SN = NOME;

Os complementos do nome são de 4 origens diferentes, podendo ser um relativo (com pronome relativo), preposicional, adjetivo ou aditivo (com conjunção aditiva). O complemento relativo é da forma de um pronome relativo mais uma sentença verbal (ex: "fornecedor que possui o preço mais baixo", "que" é um pronome relativo enquanto "possui o preço mais baixo" é uma sentença verbal). O preposicional é uma preposição com uma sentença nominal (ex: "em dolar"). O adjetivo pode ser de dois modos, um composto de adjetivo e sentença preposicional (ex: "inflação grande para o Brasil", "grande" é um

adjetivo e "para o Brasil" é uma sentença preposicional) e outro somente com adjetivo. E por fim, o aditivo que é uma conjunção aditiva junto com uma sentença nominal (ex: "componente X e componente Y", conjunção aditiva = "e" e sentença nominal = "componente Y").

COMPLEMENTO DO NOME = SADJETIVA

COMPLEMENTO DO NOME = SADITIVA

COMPLEMENTO DO NOME = SPREPOS

COMPLEMENTO DO NOME = SRELATIVO

Outras estruturas são as seguintes:

SRELATIVO = PRONOME RELATIVO + SV;

SPREPOS = PREPOSICAO + SN;

SADJETIVA = ADJETIVO + SPREPOS;

SADJETIVA = ADJETIVO;

SADITIVA = CONJUNCAO ADITIVA + SN;

SV = VERBO + SN;

SV = VERBO;

4.4.2 - Estrutura Sintática:

Uma estrutura sintática pode ser construída totalmente centrada na escolha da gramática, existindo a necessidade da construção de um predicado para cada constituinte da gramática. Isto faz com que este tipo de estrutura seja muito rígida, mas é muito comum sua utilização em alguns Parsers existentes.

Os predicados de um programa referente a esta estrutura, simplificadamente, possuem três argumentos: em primeiro lugar uma lista de palavras representando a sentença entrada, o seguinte é uma lista com a sentença de saida, está deverá ser vazia ou igual a um ponto terminal. E por ultimo o argumento para a construção do objeto Prolog

para as análises posteriores, que corresponde a sentença com as respectivas sentenças nominal e verbal e suas respectivas divisões.

Um exemplo de como seriam estes argumentos, para a frase: "a inflação do mês foi de vinte porcentos.", está mostrado abaixo, e uma arvore representando a análise da frase (OBJETO PROLOG), está na figura 8:

LISTA DE PALAVRAS = [["a"], ["inflação"], ["de"], ["o"], ["mês"], ["foi"], ["de"], ["vinte"], ["porcentos"]];

RESTO = [["."]];

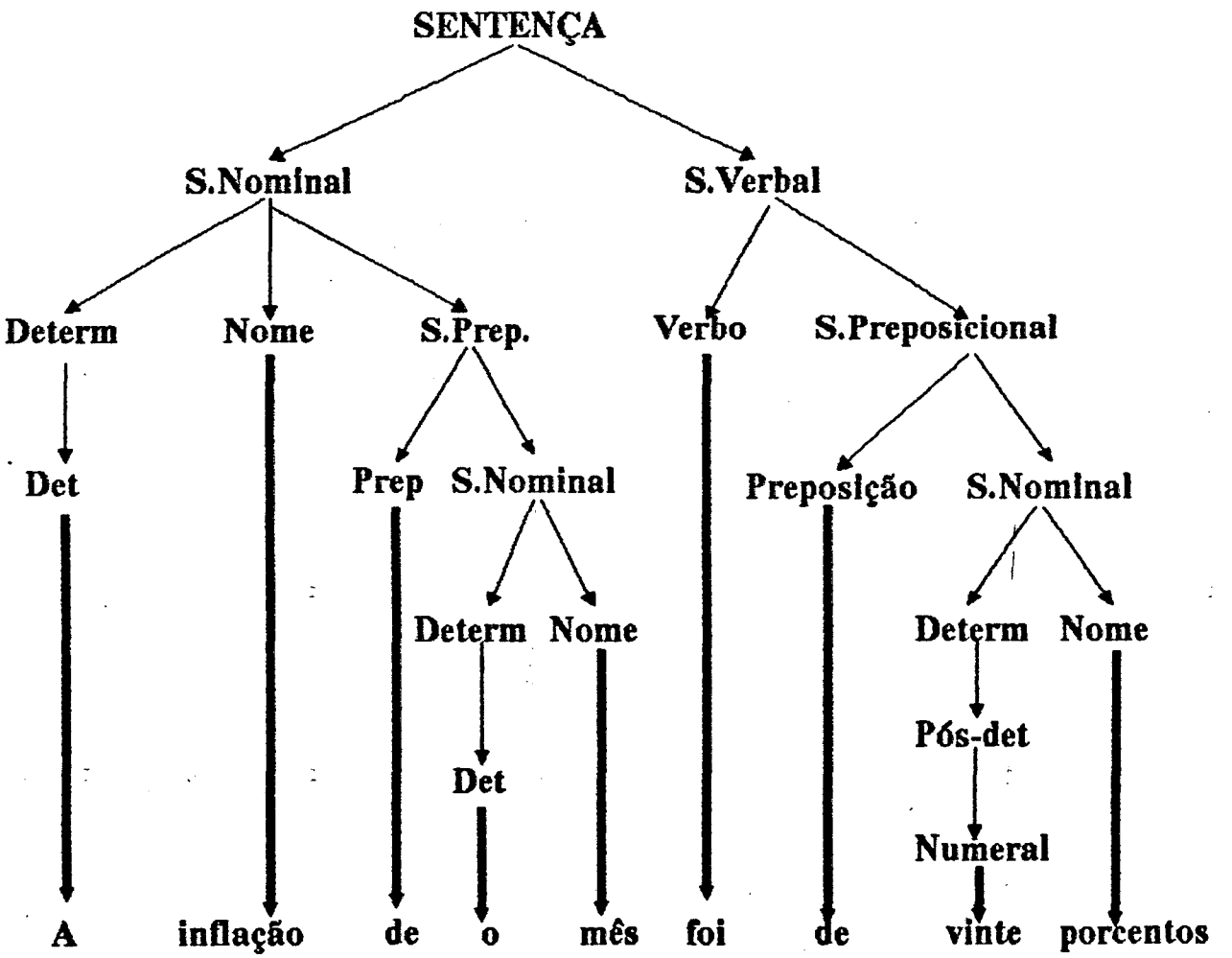


Figura 8 - Árvore Parser Sintática

```
OBJETO PROLOG =
  sent (                    sn (determ (nada, "a", nada),
                            "inflacao",
                            sp (sprep ("de",
                            sn (determ (nada, "o", nada),
                            "mes",
                            nada)))),
        sv1 (               "foi",
                            sprep ("de",
                            sn (determ (nada, nada, pos_det ("vinte")),
                            "porcentos",
                            nada))))
```

Um programa, em Turbo Prolog, referente a está estrutura se encontra, mais detalhadamente, incluído no Anexo1.

4.4.3 - Estrutura Sintática Flexível:

Foi utilizado aqui uma maneira de se fazer a análise sintatica de uma palavra, através de uma estrutura mais genérica. Este tipo de estrutura segue os conceitos de flexibilidade do Sistema de Apoio à Decisao para Administração de Capital de Giro (ver [LUNA91]).

A estrutura anterior tem a desvantagem de que cada modificação a ser feita nos constituintes da frase, ou seja, na gramática, tem de ser feita através de modificações no próprio programa, além de ter uma difícil visualização do funcionamento do programa.

Já uma estrutura recursiva mais geral pode ser feita de modo que as modificações a serem feitas, possam estar, por exemplo, em uma base de dados interna, o que não afetaria a compilação do programa. Através de simples modificações na base de dados,

pode-se colocar somente as partes da gramática que forem do interesse, dependendo da utilização.

Uma estrutura deste tipo é constituída de um predicado principal, com a função de obter os passos da gramática. A estrutura representará todas as fases da analise sintática. Cada fase é representada por uma lista de strings composto pelo nome do constituinte e pela frase (ou palavra) correspondente, o que é diferente da outra estrutura, cujo o resultado é um objeto único.

Além da lista composta pelo nome do constituinte e pela frase que o representa, existirá uma outra lista paralela, contendo as divisões do constituintes ou a palavra "terminal" caso o constituinte não tenha nenhuma divisão. Esta lista será muito útil nas fases seguintes, além de poder servir para uma possível explicação didática do funcionamento do programa e sobre a própria gramática. Um programa em Turbo Prolog detalhado está mostrado no ANEXO 2.

Um exemplo do resultado da análise sintática da frase "o dolar no paralelo subiu cinco cruzeiros" é mostrado abaixo. O primeiro termo é o constituinte, no seguinte tem-se a frase (ou palavra) e entre parenteses a divisão do constituinte, junto às palavras que são terminais está colocado também o seu gênero e número:

ex: sentenca = o dolar em o paralelo subiu cinco cruzeiros ⇒ (sintagma nominal + sintagma verbal),

sintagma nominal = o dolar em o paralelo ⇒ (determinante + nome + complemento nominal)

determinante = o ⇒ (terminal) - masculino, singular

nome = dolar ⇒ (terminal) - masculino, singular

complemento nominal = em o paralelo ⇒ (sentença preposicional)

sentença preposicional = em o paralelo ⇒ (preposição + sintagma nominal)

preposição = em ⇒ (terminal) - neutro, neutro

sintagma nominal = o paralelo ⇒ (determinante + nome)

determinante = o	⇒ (terminal) - masculino, singular
nome = paralelo	⇒ (terminal) - masculino, singular
sintagma verbal = subiu cinco cruzeiros	⇒ (verbo + sintagma nominal)
verbo = subiu	⇒ (terminal) - singular
sintagma nominal = cinco cruzeiros	⇒ (numeral + nome)
numeral = cinco	⇒ (terminal) - neutro, plural
nome = cruzeiros	⇒ (terminal) - neutro, plural

Os constituintes citados como terminais estão colocados no predicado de base de dados bd_terminal. E são eles:

NOME, ADJETIVO, PREPOSICAO, ARTIGO, CONJUNCAO ADITIVA, PRONOME RELATIVO, PRONOME PESSOAL, PRONOME POSSESSIVO, NUMERAL, QUANTIFICADOR, VERBO

4.5 - CONSISTÊNCIA DA FRASE

O objetivo da consistência da frase implementada aqui é a de verificar se as concordâncias em número e em gênero entre as palavras de determinados constituintes da frase está correta. Apartir de alterações do exemplo anterior podemos obter os seguintes erros de consistência: O dolar no paralelo subiram cinco cruzeiros, o sujeito O dolar no paralelo está no singular, portanto não concorda em número com o verbo subiram que está no plural. Um exemplo para erro em relação ao gênero seria: a paralelo, a (feminino) não concorda com paralelo (masculino).

Um outro teste de consistência da frase é em relação a transitividade do verbo, isto é, se o verbo necessita de complemento ou não, este teste estará embutido dentro da análise Semântica.

A maior parte destes erros podem até não influenciar no entendimento da linguagem natural pela máquina, entretanto não devemos esquecer que geralmente um sistema em linguagem natural estará direcionado a usuários leigos em computação. A

detecção de erros de concordâncias é importante para que o usuário tenha confiança no programa que está utilizando. Entretanto, existem erros que podem afetar significativamente o entendimento pelo computador de uma frase entrada pelo usuário.

4.5.1 - Concordância de Número:

Na concordância de números é verificada a incompatibilidade das palavras dentro da mesma frase sintática, na tentativa de obter o número correspondente a este constituinte, de modo geral, este tem que ser igual ao de cada divisão do constituinte, eliminando os neutros. Um exemplo para a frase "O dolar no paralelo", está mostrado abaixo:

em	o	paralelo	= complemento nominal,
neutro +	singular +	singular	= singular;
o	dolar	em o paralelo	= sintagma nominal,
singular +	singular +	singular	= singular.

Existem alguns casos especiais para alguns constituintes, que são os seguintes:

- O caso em que o sintagma nominal possui uma conjunção aditiva ("e" ou ","), ele é então transformado em plural, por exemplo:

  o componente X e o Y = singular + neutro + singular = plural;

- A SENTENÇA a concordância é verificada entre o sujeito (sintagma nominal) e o verbo, conforme o exemplo abaixo:

O dolar no paralelo	subiu	cinco cruzeiros	= SENTENÇA
sintagma nominal	verbo	sintagma nominal	
singular	singular	plural	= singular;

- Se a SENTENÇA for composta somente de sintagma verbal, o número que será computado é o do verbo

  ex:"aumente a produção" = singular.

A inicialização de um predicado que trate a análise de consistência da frase é feita com valores vindo da analise sintática da frase

4.5.2 - Concordância de Gênero

A análise do gênero funciona de forma quase idêntica a concordância de número. A diferença consiste de que nesta, não se verifica a concordância para todos os constituintes e sim para os formados por determinantes, nomes e adjetivos, indicando um erro nestes casos. Por exemplo: "A preço" ou " A inflação alto".

Um programa de analise da concordância da sentença está mostrado no ANEXO 3.

4.6 - ANÁLISE SEMÂNTICA

A meta principal desta fase do processamneto de linguagem natural é a de transformar a estrutura sintática, em um conjunto de informações que o programa principal "entenda". No programa principal existe um predicado chamado faz, cujo único argumento é uma lista contendo um conjunto de informações, colocados em uma determinada ordenação. Este mesmo predicado é usado para a interface de menu e a de linguagem de comando.

Após a fase de análise sintática, as palavras estarão agrupadas dentro das frases sintáticas definidas pela gramática. Cada frase sintática tem uma palavra principal na qual a frase gira, e será com está palavra que iremos trabalhar para se fazer a análise semântica da sentença.

Para a análise semântica resolveu-se utilizar uma estrutura de casos de Simmons [GEET90], que possui toda a estrutura da sentença centrada no verbo com as outras demais unidades ligadas a ele. As frases sintáticas operam como uma unidade produzindo funções reais que definem sua relação com o verbo.

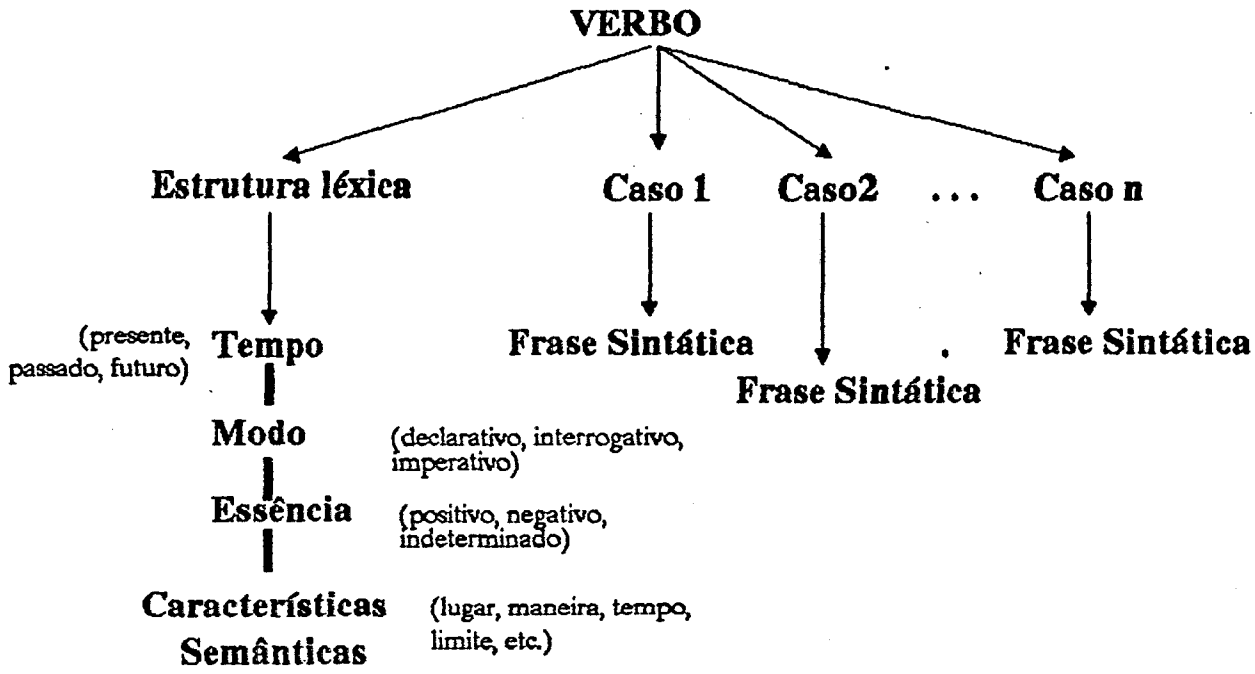


Figura 9 - Estrutura de Frame para um Verbo

Uma estrutura de frame geral para um verbo está mostrada na figura 9, e em seguida é apresentado na figura 10 um exemplo de como seria uma estrutura para a frase "a venda aumentou 10 %".

Nesta estrutura escolhida, combina-se a análise gramatical sintática com a interpretação semântica, para que se possa retirar o máximo de informações pertinentes para se representar o significado da sentença.

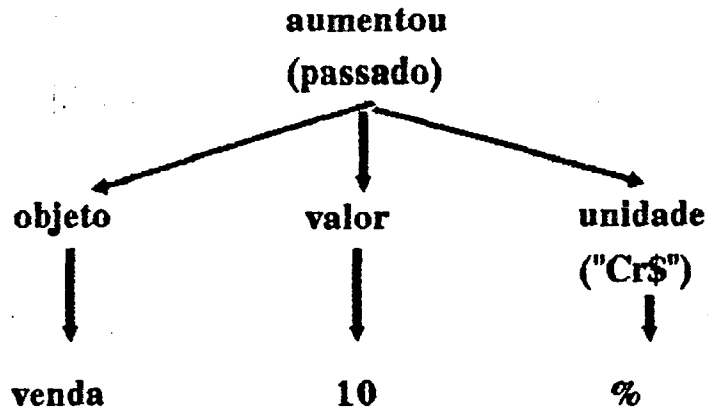


Figura 10 - Uma Estrutura em Frame para "aumentar"

Antes de se passar para o tratamento semântico, algumas adaptações na estrutura sintática devem ser feitas. Um primeiro passo, para simplificar o processo, seria a criação de uma estrutura para "limpar os ruídos", que consiste de retirar algumas palavras sem função para o entendimento da sentença, tais como artigos, pronomes, palavras que o léxico indicar como sem nenhuma função,etc. Caso seja preposição sua função deverá ser colocado em seu lugar, para auxiliar a palavra principal da frase.

O passo seguinte é o de dividir a sentença em verbo, palavras principais que estejam localizadas antes do verbo, e nas que estejam depois. Apartir dos dados da estrutura léxica, substitui-se as palavras pela sua Origem e pela sua(s) função(ões). Para as palavras que venham antes do verbo associa-se a palavra "AGENTE", por terem estas palavras uma função de agente em relação ao verbo.

Com a sentença colocada em um formato ideal para se fazer a análise do seu significado, recupera-se a estrutura de frame do verbo - para o mesmo verbo pode-se ter diferentes estruturas, que serão analizadas uma a uma até se achar a correta. Com os dados referentes ao verbo, compara-os aos da sentença buscando-se uma solução para que se possa entrar em contato com o programa principal.

A busca de uma solução é feita, tentando-se preencher o verbo com os seus complementos, em primeiro lugar é verificada a existência na sentença das funções que são obrigatórias na sentença, e em seguida são colocadas as funções opcional ou seu default, para o caso da função não existir na sentença.

A análise da frase "apresentar os componentes que estejam na parte A da curva ABC", considerando que a frase passou pelas fases de análise sintática, limpa ruído e a que divide o verbo; ficará da seguite forma:

Frase já tratada:

verbo = "APRESENTAR",

pré_verbo = [],

pós-verbo = [["OBJETO","COMPONENTE"],["LIMITE","A"],["OBJETO","CURVA ABC"]]

Um dos frames para "apresentar" (podem existir outros frames):

frame_verbo ("APRESENTAR", ["OBJETO", "CURVA ABC"], [["SINAL", "="], ["LIMITE", "TODOS"]], "SAIDA")

Primeiramente, se faz a procura dos complementos que são obrigatórios, a existência de dois "OBJETO"s não irá infleunciar, pois ele pegará apenas o primeiro, que está ligado ao verbo. Portanto, pega-se "COMPONENTE" como "OBJETO", em seguida pega-se a própria função "CURVA ABC". É então feita a procura pelas funções opcionais, não existe a função "SINAL" na sentença então coloca-se "=", e em limite coloca-se "A". Tendo portanto como resultado:

["APRESENTAR","COMPONENTE","CURVA ABC","=","A"]

Existem alguns casos especiais de presença de conjunção na construção da arquitetura da sentença, como alguns casos da presença de uma conjunção aditiva dentro de uma mesma oração, por exemplo "O componente X e o Y aumentaram 10 %" será dividida em duas orações "O componente X aumentou 10%" e "O componente Y aumentou 10 %", está separação será feita um pouco antes da fase limpa ruído.

A presença de conjunções ligando orações dentro de uma sentença, o que é muito comum neste sistema, frases tais como: "O QUE ACONTECE COM O LUCRO LIQUIDO , SE AUMENTAR AS VENDAS EM DEZ PORCENTOS", terão um tratamento distinto dentro da análise semântica, o mesmo que o do resultante da frase com a presença de sentenças aditivas. Trata-se apenas de separar as sentenças, o que dá para se fazer simplesmente através da estrutura sintática da frase, e envia-se primeiramente as sentenças que são de entrada (informações retiradas da estrutura léxica) e depois as de saída.

CAPÍTULO V - APRESENTAÇÃO DOS RESULTADOS

5.1 - INTRODUÇÃO

Todos os passos feitos pelo protótipo do sistema será mostrado neste capítulo através de um tipo de sentença muito comum no Sistema de Administração de Capital de Giro. O exemplo poderá ser acompanhado através das figuras que fazem parte deste capítulo.

Considera-se aqui que todas as palavras entradas fazem parte da estrutura léxica, caso uma delas não existisse, seria feito uma consulta ao usuário para a entrada de um sinônimo.

5.2 - ANÁLISE LÉXICA

Um exemplo de todos os passos do programa será feito aqui, a frase a ser avaliada será "O QUE ACONTECE COM O LUCRO LIQUIDO , SE AUMENTAR AS VENDAS EM DEZ PORCENTOS". O primeiro passo e verificar a presença de todas as palavras na estrutura léxica e coloca-las em forma de lista.

["O", "QUE", "ACONTECE", "COM", "O", "LUCRO", "LIQUIDO", ",", "SE", "AUMENTAR", "AS", "VENDAS", "EM", "DEZ", "PORCENTOS"]

5.3 - ANÁLISE SINTÁTICA E CONSISTÊNCIA DA FRASE

A fase seguinte é da análise sintática, que através da gramática utilizada, verificará se a sentença está em um formato correto e indicará todas as frases sintáticas existentes. O passo seguinte, que será mostrado junto, é verificar se os constituintes estão corretos em número e gênero.

A sentença entrada representada pelo constituinte SENTENÇA tem como divisão duas outras sentenças, cada uma representará uma oração. O número por se tratar da

ultima análise possível é colocado a palavra FIM, isto é um sinal que a sentença está correta e já foi analisada, o mesmo serve para o gênero.

SENTENCA = O QUE ACONTECE COM O LUCRO LIQUIDO , SE AUMENTAR AS VENDAS EM DEZ PORCENTOS

DIVISAO = SENTENCA + SENTENCA2

NUMERO = FIM **GENERO = FIM**

Em seguida a primeira oração da sentença original é analisada, também é um constituinte SENTENÇA, e tem como divisão um determinante seguido por uma sentença relativa. Em número e gênero está colocado a palavra FIM. Nota-se que está se fazendo é uma busca em árvore, em que o constituinte (SENTENCA) é um nó inicial, e as suas divisões representam o nó final de um caminho.

SENTENCA = O QUE ACONTECE COM O LUCRO LIQUIDO

DIVISAO = DETERMINANTE + SRELATIVA

NUMERO = FIM **GENERO = FIM**

Chegou-se agora em um constituinte terminal, que possui número SINGULAR e gênero MASCULINO. Este nó indica o fim de um caminho da árvore.

DETERMINANTE = O

terminal ⇒ SINGULAR , MASCULINO

Da SENTENÇA que representa a primeira oração, tem-se o constituinte SRELATIVA que é dividido em pronome relativo acrescido de sintagma verbal. O número aqui foi analizado e teve como resposta SINGULAR, significando que o constituinte está no singular. Já para o gênero tem-se como resposta "nada", isto é, não foi necessario se fazer a análise de consistência de gênero para este tipo de constituinte. O mesmo processo é feito para todos os demais constituintes até que se chegue a uma estrutura final.

SRELATIVA = QUE ACONTECE COM O LUCRO LIQUIDO

DIVISAO = PRONOME RELATIVO + SINTAGMA VERBAL

NUMERO = SINGULAR GENERO = nada

PRONOME RELATIVO = QUE

terminal ⇒ INDETERMINADO, COMUM

SINTAGMA VERBAL = ACONTECE COM O LUCRO LIQUIDO

DIVISAO = VERBO + COMPLEMENTO DO VERBO

NUMERO = SINGULAR GENERO = nada

VERBO = ACONTECE

terminal ⇒ SINGULAR, TRANSITIVO

COMPLEMENTO DO VERBO = COM O LUCRO LIQUIDO

DIVISAO = SPREPOS

NUMERO = SINGULAR GENERO = nada

SPREPOS = COM O LUCRO LIQUIDO

DIVISAO = PREPOSICAO + SINTAGMA NOMINAL

NUMERO = SINGULAR GENERO = nada

PREPOSICAO = COM

terminal ⇒ INDETERMINADO, NEUTRO

SINTAGMA NOMINAL = O LUCRO LIQUIDO

DIVISAO = NOME_AUX

NUMERO = SINGULAR GENERO = nada

NOME_AUX = O LUCRO LIQUIDO

DIVISAO = SDETERM + NOME

NUMERO = SINGULAR GENERO = COMUM

SDETERM = O

DIVISAO = DETERMINANTE

NUMERO = SINGULAR GENERO = nada

DETERMINANTE = O

terminal ⇒ SINGULAR, MASCULINO

NOME = LUCRO LIQUIDO

terminal ⇒ SINGULAR, COMUM

SENTENCA2 = SE AUMENTAR AS VENDAS EM DEZ PORCENTOS

DIVISAO = CONJUNCAO + SENTENCA

NUMERO = FIM GENERO = FIM

CONJUNCAO = SE

terminal ⇒ INDETERMINADO, COMUM

SENTENCA = AUMENTAR AS VENDAS EM DEZ PORCENTOS

DIVISAO = SINTAGMA VERBAL

NUMERO = FIM GENERO = FIM

SINTAGMA VERBAL = AUMENTAR AS VENDAS EM DEZ PORCENTOS

DIVISAO = VERBO + COMPLEMENTO DO VERBO

NUMERO = SINGULAR GENERO = nada

VERBO = AUMENTAR

terminal ⇒ SINGULAR, TRANSITIVO

COMPLEMENTO DO VERBO = AS VENDAS EM DEZ PORCENTOS

DIVISAO = SINTAGMA NOMINAL

NUMERO = PLURAL GENERO = nada

SINTAGMA NOMINAL = AS VENDAS EM'DEZ PORCENTOS

DIVISAO = NOME_AUX + COMPLEMENTO DO NOME

NUMERO = PLURAL GENERO = nada

NOME_AUX = AS VENDAS

DIVISAO = SDETERM + NOME

NUMERO = PLURAL GENERO = FEMININO

SDETERM = AS

DIVISAO = DETERMINANTE

NUMERO = PLURAL GENERO = nada

DETERMINANTE = AS

terminal ⇒ PLURAL, FEMININO

NOME = VENDAS

terminal ⇒ PLURAL, FEMININO

COMPLEMENTO DO NOME = EM DEZ PORCENTOS

DIVISAO = SPREPOS

NUMERO = PLURAL GENERO = nada

SPREPOS = EM DEZ PORCENTOS

DIVISAO = PREPOSICAO + SINTAGMA NOMINAL

NUMERO = PLURAL GENERO = nada

PREPOSICAO = EM

terminal ⇒ INDETERMINADO, COMUM

SINTAGMA NOMINAL = DEZ PORCENTOS

DIVISAO = NOME_AUX

NUMERO = PLURAL GENERO = nada

NOME_AUX = DEZ PORCENTOS

DIVISAO = SDETERM + NOME

NUMERO = PLURAL **GENERO = COMUM**

SDETERM = DEZ

DIVISAO = POSDET

NUMERO = PLURAL **GENERO = nada**

POSDET = DEZ

DIVISAO = NUMERAL

NUMERO = PLURAL **GENERO = nada**

NUMERAL = DEZ

terminal ⇒ PLURAL, COMUM

NOME = PORCENTOS

terminal ⇒ PLURAL, COMUM

Pode-se verificar os resultados da análise sintática (fig 11), análise da consistência de número (fig 12) e análise da consistência de gênero (fig 13) através da análise de suas respectivas árvores sintáticas.

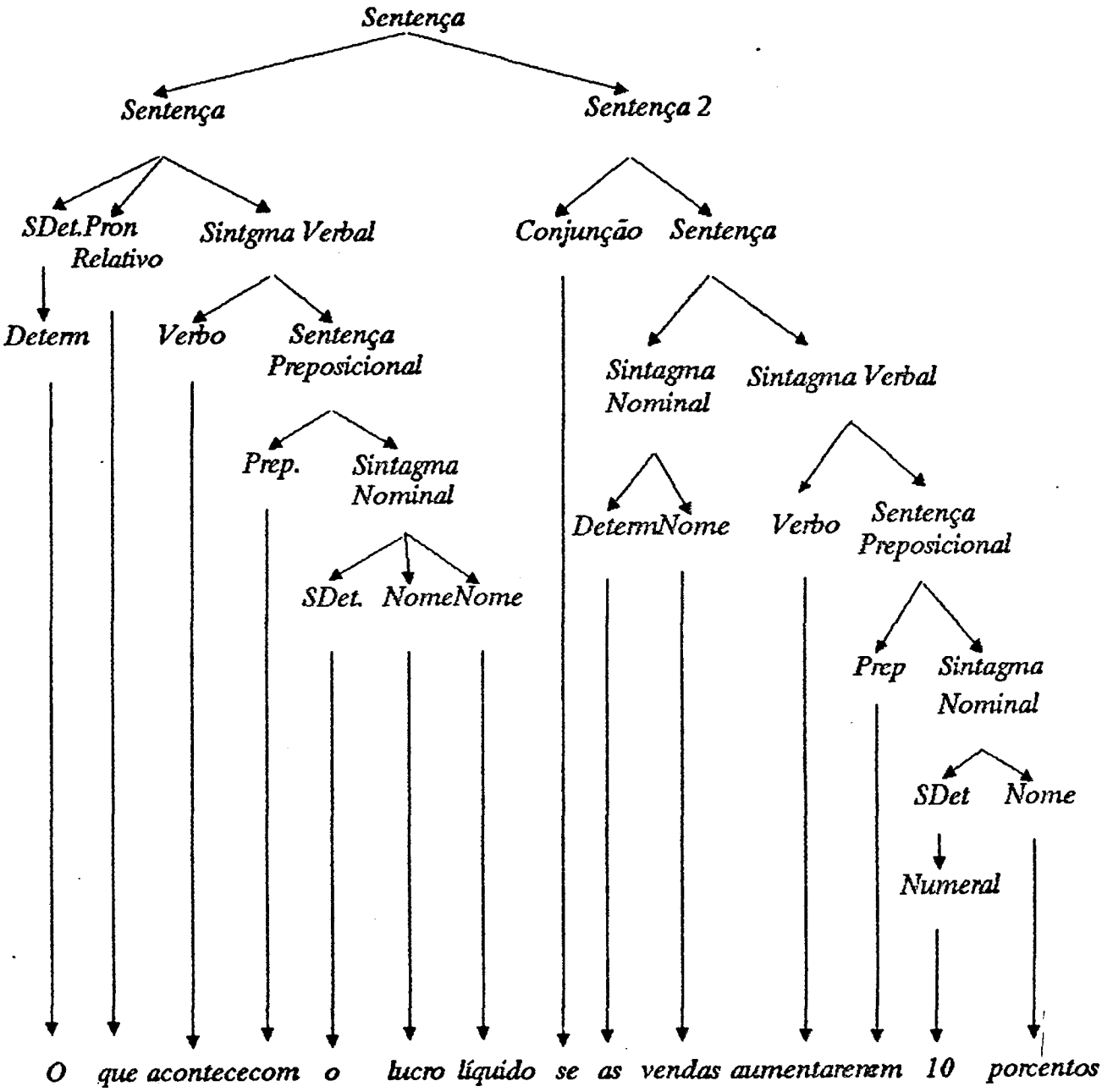


Figura 11 - Árvore Sintática da Sentença

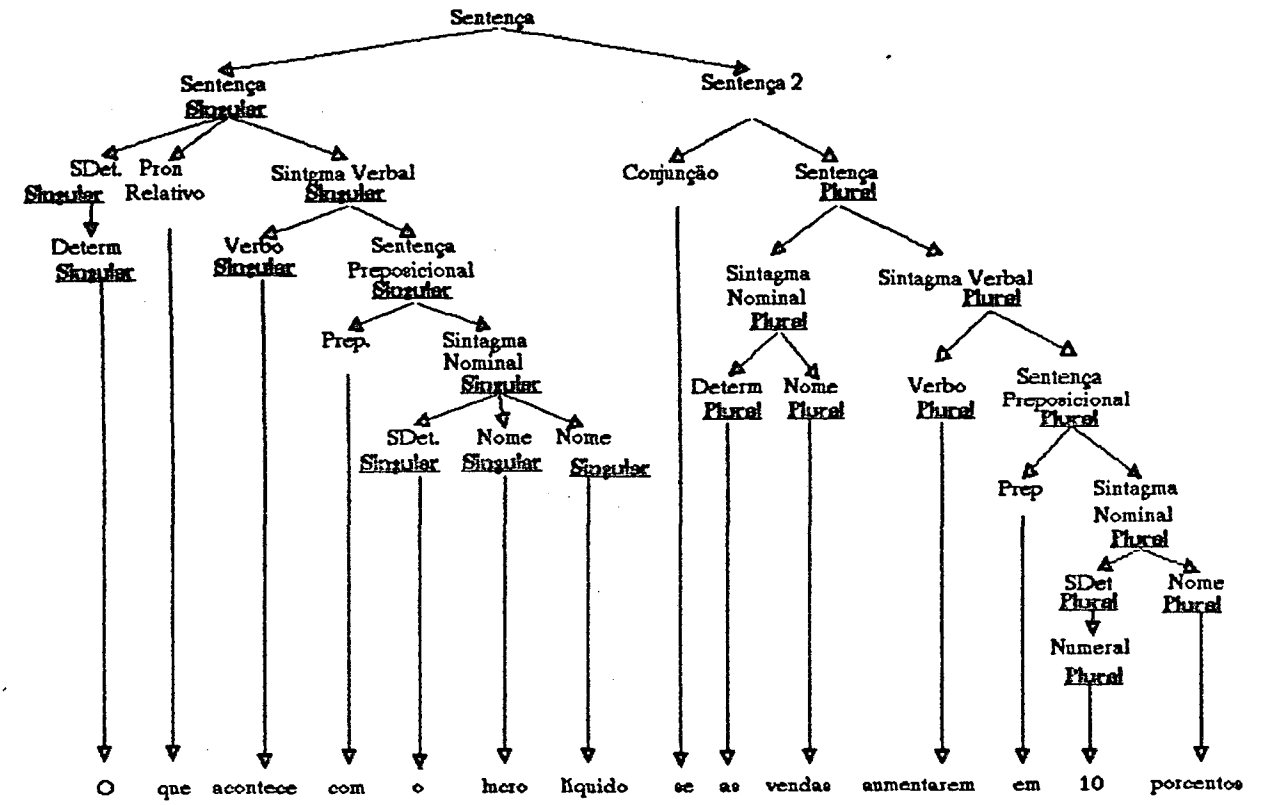


Figura 12 - Consistência de Número

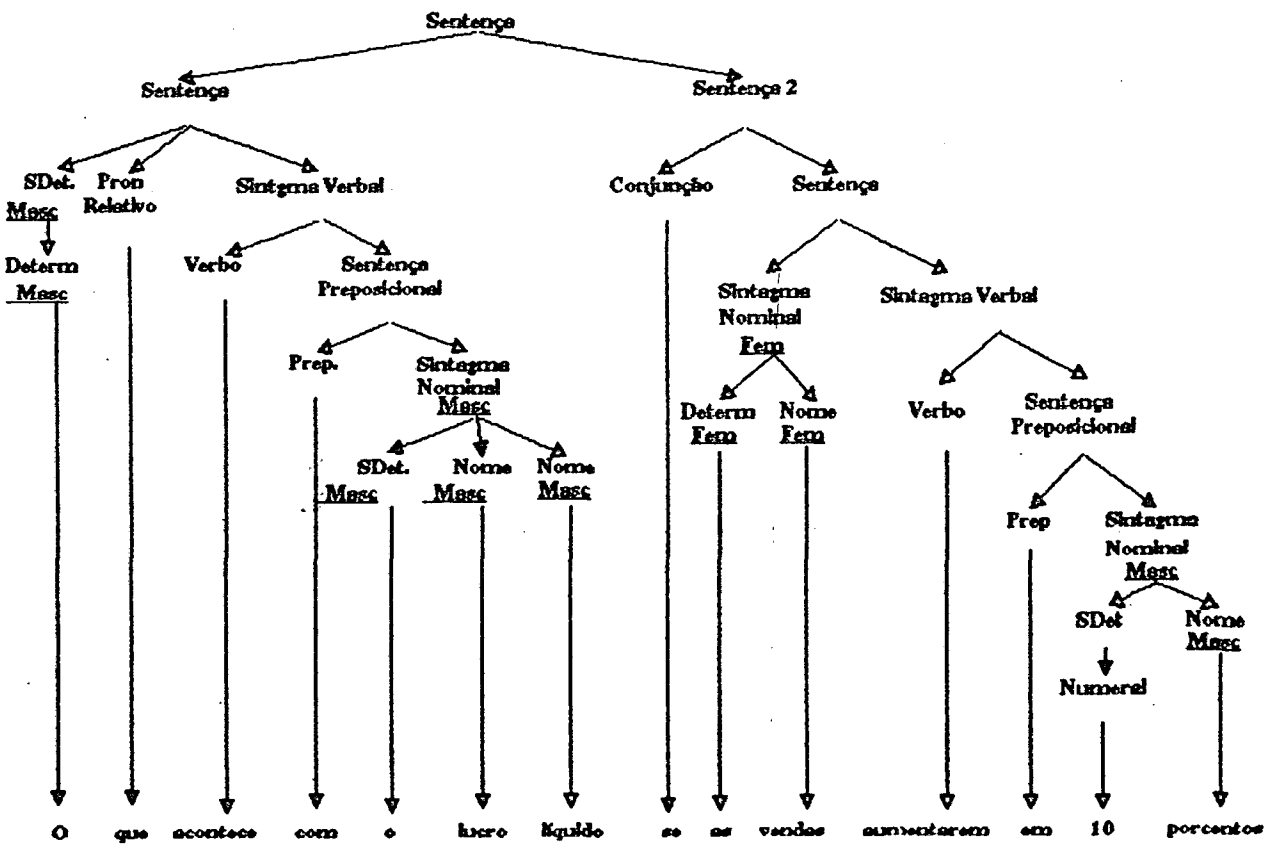


Figura 13 - Consistência de Gênero

5.4 - ANÁLISE SEMÂNTICA

No início da análise semântica, por ser este um caso especial em que a sentença é composta por duas orações, é necessário separa-las. Tendo como resultado, o seguinte

Primeira oração:

[["SENTENCA", "O QUE ACONTECE COM O LUCRO LIQUIDO "], ["DETERMINANTE", "O"], ["SRELATIVA", "QUE ACONTECE COM O LUCRO LIQUIDO "], ["PRONOME RELATIVO", "QUE"], ["SINTAGMA VERBAL", "ACONTECE COM O LUCRO LIQUIDO "], ["VERBO", "ACONTECE"], ["COMPLEMENTO DO VERBO", "COM O LUCRO LIQUIDO "], ["SPREPOS", "COM O LUCRO LIQUIDO "], ["PREPOSICAO", "COM"], ["SINTAGMA NOMINAL", "O LUCRO LIQUIDO "], ["NOME_AUX", "O LUCRO LIQUIDO "], ["SDETERM", "O "], ["DETERMINANTE", "O"], ["NOME", "LUCRO LIQUIDO"]]

Segunda oração:

[["SENTENCA", "SE AUMENTAR AS VENDAS EM DEZ PORCENTOS "], ["SCONJUNCAO", "SE AUMENTAR AS VENDAS EM DEZ PORCENTOS "], ["CONJUNCAO", "SE"], ["SINTAGMA VERBAL", "AUMENTAR AS VENDAS EM DEZ PORCENTOS "], ["VERBO", "AUMENTAR"], ["COMPLEMENTO DO VERBO", "AS VENDAS EM DEZ PORCENTOS "], ["SINTAGMA NOMINAL", "AS VENDAS EM DEZ PORCENTOS "], ["NOME_AUX", "AS VENDAS "], ["SDETERM", "AS "], ["DETERMINANTE", "AS"], ["NOME", "VENDAS"], ["COMPLEMENTO DO NOME", "EM DEZ PORCENTOS "], ["SPREPOS", "EM DEZ PORCENTOS "], ["PREPOSICAO", "EM"], ["SINTAGMA NOMINAL", "DEZ PORCENTOS "], ["NOME_AUX", "DEZ PORCENTOS "], ["SDETERM", "DEZ "], ["POSDET", "DEZ "], ["NUMERAL", "DEZ"], ["NOME", "PORCENTOS"]]

Em seguida verifica-se através do verbo se ela é de entrada ou saída. O verbo "**acontecer**" é de **saída** e "**aumentar**" é de **entrada**, portanto é feito primeiro a transformação da segunda oração.

Aqui em diante é feito da mesma forma caso fosse uma sentença com somente uma oração, que iria iniciar a análise semântica exatamente desta parte. De início retira-se as palavras sem funções (ver fig 14) e divide a sentença em três partes.

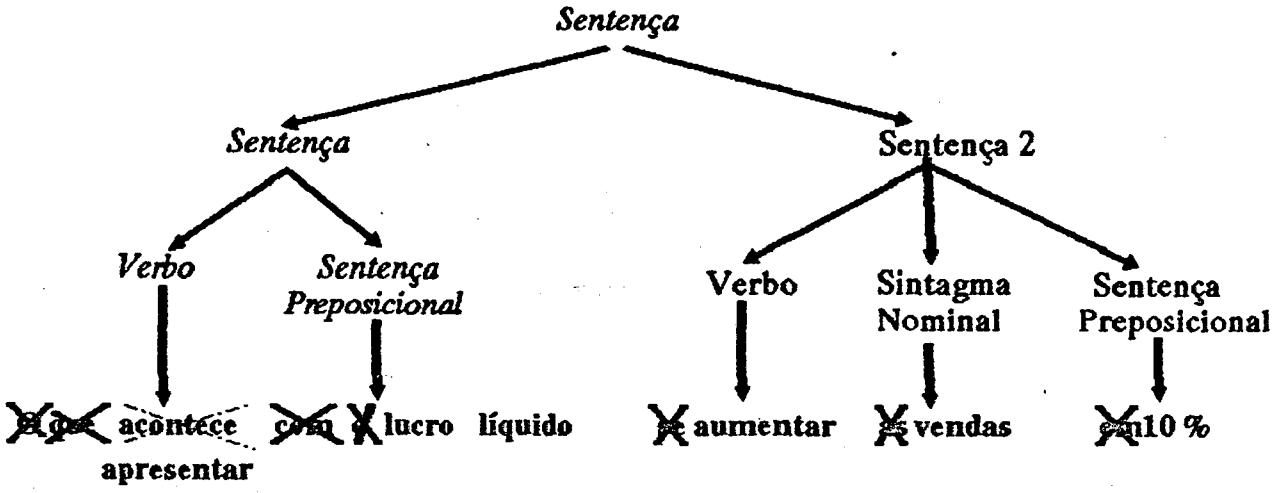


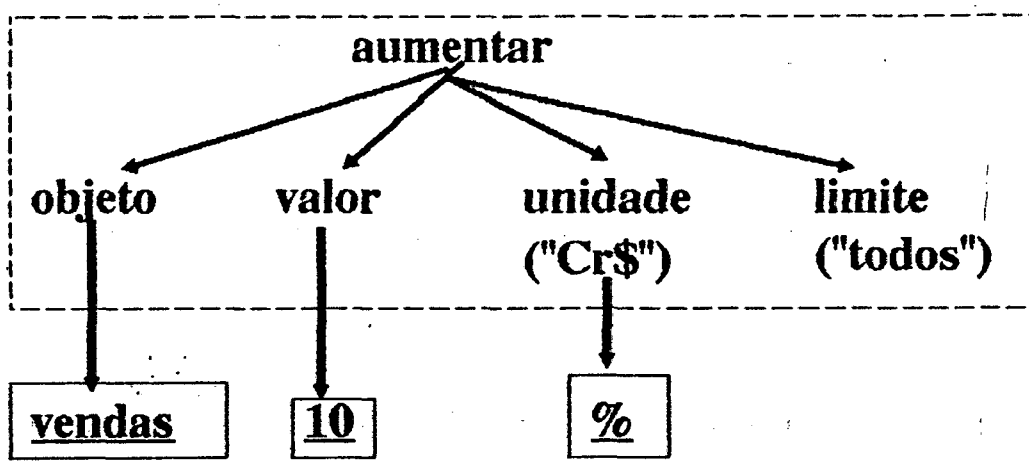
Figura 14 - Ajustes na Sentença, para a Análise Semântica

Verbo - AUMENTAR = ENTRADA,

Pré-Verbo - [],

Pós-Verbo - [["OBJETO","VENDA"],["QUANTIDADE","10"],["AUXILIAR QUANTIDADE","PORCENTOS"]])

Utiliza-se o frame do verbo "aumentar" para retirar da sentença os códigos códigos necessários ao predicado faz do sistema principal (fig 15).



Resultado = faz (aumentar, vendas, 10, %, todos)

Figura 15 - Estrutura para o Verbo Aumentar

frame_verbo:

Verbo = AUMENTAR

Lista obrigatorios = ["OBJETO","QUANTIDADE"]

Lista opcionais = [["AUXILIAR QUANTIDADE","MOEDA CORRENTE"]]

faz (["AUMENTAR","VENDA","10","PORCENTOS"])

O mesmo é feito para a sentença "O QUE ACONTECE COM O LUCRO LIQUIDO", onde o verbo "acontece" é substituído por "apresentar", que para o caso do sistema em questão tem o mesmo significado (ver fig 14). Com o frame de "apresentar" chega-se a um resultado final da análise da sentença. (fig 16)

Verbo - APRESENTAR = SAIDA

Pré-Verbo - []

Pós-Verbo - ["OBJETO","LUCRO LIQUIDO"]]

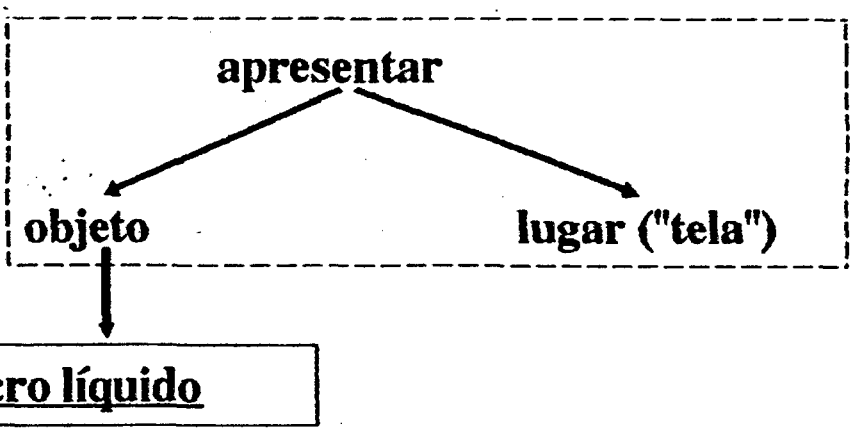
frame_verbo:

Verbo = APRESENTAR

Lista obrigatorios = ["OBJETO"]

Lista opcionais = [["LUGAR","TELA"]]

faz(["APRESENTAR","LUCRO LIQUIDO","TELA"])



➡ **Resultado = faz (apresentar, lucro líquido, tela)**

Figura 16 - Estrutura para o Verbo Apresentar

CAPÍTULO VI - CONCLUSÃO E RECOMENDAÇÕES

6.1 - CONCLUSÃO:

O objetivo do trabalho era o de construir na prática um Processador de Linguagem Natural, para a língua Portuguesa, aplicado a um universo restrito de conhecimento - uma interface para o Sistema de Administração de Capital de Giro, que estamos implementando no PPGEPS - UFSC. Este objetivo foi cumprido plenamente, apesar do Sistema de Administração de Capital de Giro estar ainda em fase de conclusão, o Protótipo da Interface demonstrou seu potencial com os módulos já em funcionamento.

Houve uma grande dificuldade na procura de bibliografia existente, que apesar de ser bem vasta sobre o assunto, se encontra concentrada quase totalmente em teorias, com muitas poucas referências práticas, relatando sistemas já em funcionamento. Este trabalho tenta dar uma boa contribuição neste sentido. Além disto existe um grande desacordo entre os especialistas da área.

Buscou-se a construção de uma interface que fosse o mais flexível possível para que possam ser feitos ajustes nela, dependendo do interesse do usuário, sem ter que fazer nenhuma programação extra. Preocupou-se a princípio em não construir algo que fosse amplo o suficiente sem ser demasiadamente rigoroso. Tornando a interface o mais amigável possível.

Os ajustes finais desta interface deverão ser feitos, durante o próprio teste e validação do Sistema, no qual o contato com vários usuários levará a incorporação de novas palavras e frames de verbo na estrutura Léxica da Interface de Linguagem Natural. O que já era esperado anteriormente, devido a construção quase em paralelo da Interface e do Sistema.

Através da estrutura utilizada, pela sua generalidade, torna-se simples a sua utilização em outro campo de conhecimento. Isto é válido tanto para a interface quanto para o Sistema de Administração de Capital de Giro.

6.2 - SUGESTÕES:

Uma sugestão seria a construção de um sistema didático, para o auxílio no ensino de gramática e análise sintática, assim como o de concordância de gênero e número.

Pode-se também aplicar o sistema a outro campo de conhecimento, para verificar sua versatilidade, de preferência algo que involva o público em geral, já que o nosso usuário era um certo tipo de especialista.

Outra sugestão interessante é a de incorporar regras de tratamento de incerteza para se fazer, principalmente a análise semântica. O que já estava previsto durante a criação da estrutura léxica.

Aproveitando que alterações da gramática podem ser feitas através de uma base de dados, para se testar outros tipos de gramática generativas. Para facilitar o trabalho sugerimos a criação de uma shell, para ajudar a manipulação da base de dados.

ANEXO 1

Este anexo representa a listagem comentada do programa 1 (indicado como sendo de uma estrutura rígida) para a análise sintática de uma sentença. A linguagem de programação é o Turbo Prolog.

O predicado roda1 serve para a entrada da sentença a ser anlisada, caso se tenha teclado ENTER ou ESC o predicado falha e volta para o menu principal, caso contrario a sentença entrada é mandada para o predicado roda2

```
roda1:-
 write("\ n Escreva uma sentenca:\ n "),
 readln(LIN), LIN "",
 roda2(LIN), !,
 roda1.
```

O predicado roda2 é que vai coordenar a utilização do analisador de sentença. Em primeiro lugar, o predicado lefrase vai separar a frase entrada em uma lista de palavras, esta lista de palavras vai então para o predicado s_sentença que irá definir qual a função de cada palavra.

```
roda2(LIN):-
 lefrase(LIN,LSTPAL),
 s_sentenca( LSTPAL, _, SENT ),
 write("ANALISE SINTATICA = ",SENT," "),!.

roda2(_).
```

O predicado lefrase vai separar a frase entrada em uma lista de palavras correspondentes. O predicado do Turbo Prolog fronttoken, tem a função de pegar a palavra TOK da string STR sobrando a string STR1. Na primeira claúsula existe o predicado

conjunção que verifica se a palavra TOK, retirada da frase, é uma conjunção, caso seja o predicado volta com as duas palavras correspondentes, e as coloca (durante o backtracking) na lista de palavras.

Caso não seja conjunção então verifica-se a claúsula lefrase seguinte, esta segunda claúsula faz o mesmo que a anterior, somente no lugar do predicado conjunção tem o predicado checar que verifica a existência ou não da palavra TOK procurada. E por ultimo a claúsula que volta com a lista vazia no final da operação, para que a lista seja preenchida no backtracking.

```
lefrase(STR,[TOK1 |[TOK2 |LSTPAL]]) :-
 fronttoken(STR,TOK,STR1),
 conjuncao( TOK, TOK1, TOK2 ),!,
 lefrase(STR1,LSTPAL).

lefrase(STR,[TOK|LSTPAL]) :-
 fronttoken(STR,TOK,STR1),
 checar(TOK),!,
 lefrase(STR1,LSTPAL).

lefrase(_,[]).
```

O predicado s_sentença possui três argumentos: em primeiro lugar o LSTPAL representa a sentença (na forma de uma lista de palavras) entrada, enquanto que LSTPAL2 é a sentença de saida, esta será sempre vazia ou igual a um ponto terminal. E por ultimo o argumento para a construção do objeto Prolog para as análises posteriores, que é a sentença (representado por sent) com as respectivas sentenças nominal e verbal.

A ultima cabeça é uma mensagem, para o caso de ter ocorrido alguma falha na constituição da sentença entrada.

```
/* sentenca = Sent. Nominal + Sent. Verbal */

/* sentenca = Sent. Verbal */

        s_sentenca(LSTPAL,LSTPAL2,sent(SN,SV)):-
          s_sn(LSTPAL,LSTPAL1,SN),
          s_sv(LSTPAL1,LSTPAL2,SV),
          tom(LSTPAL2),!.

        s_sentenca(_,_,_):-
          write("> Sentenca nao reconhecida (Use F8 para pegar a ultima linha)\ n"),
          fail.
```

O predicado tom verifica se a frase chegou em um ponto terminal.

```
        tom([]).

        tom(["."]).

        tom(["?"]).
```

O predicado s_sn tem, da mesma forma que s_sentença, três argumentos, que são respectivamente, a sentença de entrada a ser analisada, a sentença de saída (a lista que restou) e o objeto Prolog construído com o determinante (ou o símbolo nada), o nome e o complemento (ou o símbolo nada). No predicado s_sn chama-se o s_determ que vai achar a sentença determinante existente, o predicado e_nome que verifica se a palavra NOME tem a função de nome e o predicado s_outros verifica se o que segue na lista LSTPAL1 é algum dos complementos possíveis. A segunda cabeça é para o caso de falhar (a única possibilidade de falha é o predicado e_nome), e com isto o objeto Prolog para a sentença nominal volta com o símbolo nada.

```
/* Sent. Nominal = Sent Determinante + Nome + Complemento
Sent. Nominal = Sent Determinante + Nome
Sent. Nominal = Nome + Complemento
Sent. Nominal = Nome */

        s_sn(LSTPAL,LSTPAL2,sn(DETERM,NOME,OUTROS)):-
          s_determ(LSTPAL,[NOME|LSTPAL1],DETERM),
          e_nome(NOME),
          s_outros(LSTPAL1,LSTPAL2,OUTROS).

        s_sn(LSTPAL,LSTPAL,nada).
```

A estrutura para o constituinte de sentença determinante pode ser de sete tipos diferentes, que estão mostrados com exemplos a seguir

```
/* Sent Determinante = Pre-determinante + Determinante + Pos-determinante
exc algumas de essas minhas sete
Sent Determinante = Pre-determinante + Determinante
exc todos aqueles
Sent Determinante = Determinante + Pos-determinante
exc as oito
Sent Determinante = Determinante
exc os
Sent Determinante = Pre-determinante + Pos-determinante
exc algumas de tuas
Sent Determinante = Pos-determinante
exc todos
Sent Determinante = Pre-determinante
exc minhas */
```

O predicado s_determ possui três argumentos, a lista de palavras entradas, a lista de palavras saidas e o objeto Prolog correspondente a sentença determinante. Cada claúsula corresponde a um dos tipos de sentença determinante mostrada acima. Com o último representando com o símbolo nada, a não existência de algum dos tipos de sentença determinante.

O predicado e_det verifica se a palavra DETERM é considerada um determinante.

```
s_determ(LSTPAL,LSTPAL2,determ(PRE, DETERM, POS)):-
  s_predet(LSTPAL,[DETERM|LSTPAL1],PRE),
  e_det(DETERM),
  s_posdet(LSTPAL1,LSTPAL2,POS).

s_determ([DETERM|LSTPAL],LSTPAL2,determ(nada, DETERM, POS)):-
  e_det(DETERM),
  s_posdet(LSTPAL,LSTPAL2,POS).

s_determ(LSTPAL,LSTPAL2,determ(PRE,nada,POS)):-
  s_predet(LSTPAL,LSTPAL1,PRE),
  s_posdet(LSTPAL1,LSTPAL2,POS).

s_determ(LSTPAL,LSTPAL2,determ(nada,nada,POS)):-
  s_posdet(LSTPAL,LSTPAL2,POS),
  not(POS = nada).

s_determ(LSTPAL,LSTPAL,nada).
```

O predicado e_quant verifica se a palavra QUANT é considerada um quantificador. E e_preposição verifica se a palavra PREP é uma preposição.

```
/* Pre_determinante = Quantificador + Preposicao
```

```
/* Pre_determinante = Quantificador */

s_predet([QUANT|[PREP|LSTPAL]],LSTPAL,pre_det(QUANT,PREP)):-
  e_quant(QUANT),
  e_preposicao(PREP).

s_predet([QUANT|LSTPAL],LSTPAL,pre_det(QUANT,nada)):-
  e_quant(QUANT).
```

O predicado e_p_poss verifica se a palavra POSS é considerada um pronome possessivo. E e_numeral verifica se a palavra NUM é um numeral.

```
/* Pos-determinante = Pronome Possessivo + Numeral
Pos-determinante = Pronome Possessivo
Pos-determinante = Numeral */

s_posdet([POSS|[NUM|LSTPAL]],LSTPAL, pos_det(POSS,NUM)):-
  e_p_poss(POSS),
  e_numeral(NUM).

s_posdet([POSS|LSTPAL],LSTPAL,pos_det(POSS,nada)):-
  e_p_poss(POSS).

s_posdet([NUM|LSTPAL],LSTPAL,pos_det(nada,NUM)):-
  e_numeral(NUM).

s_posdet(LSTPAL,LSTPAL,nada).
```

Para o objeto prolog todos os complementos nominais pertencem ao domínio Outros. Nota-se que os predicados s_outros são praticamente iguais, as mudanças ocorrem basicamente no objeto Prolog para a construção da sentença e nos predicados específicos que averigüam se determinada palavra pertence a certo grupo gramatical ou se forma um tipo de sentença.

```
/* Complemento (relativo) = Pronome Relativo + Sent Verbal
Complemento (preposicional) = Preposicao + Sent. Nominal
Complemento (adjetivo) = Adjetivo + Sent. Preposicional
Complemento (adjetivo) = Adjetivo
Complemento (aditivo) = Conjuncao Aditiva + Sent. Nominal */

s_outros([REL|LSTPAL],LSTPAL1,relativo(REL,SV)):-
  e_pron_rel(REL),
  s_sv(LSTPAL,LSTPAL1,SV).

s_outros([PREP|LSTPAL],LSTPAL1, sp( sprep(PREP,SN))):-
  e_preposicao(PREP),
  s_sn(LSTPAL,LSTPAL1,SN).

s_outros([ADJETIVO|LSTPAL],LSTPAL1,sa(ADJETIVO,SP)):-
  e_adjetivo(ADJETIVO),
  s_sp(LSTPAL,LSTPAL1,SP).

s_outros([ADJETIVO|LSTPAL],LSTPAL,sa(ADJETIVO,nada)):-
  e_adjetivo(ADJETIVO).

s_outros([ADITIVA|LSTPAL],LSTPAL1,aditiva(ADITIVA,SN)):-
  e_aditiva(ADITIVA),
  s_sn(LSTPAL,LSTPAL1,SN).

s_outros(LSTPAL,LSTPAL,nada).
```

A sentença preposicional é formada por uma preposição acrescido de uma sentença nominal. O predicado s_sp coloca no objeto Prolog os valores correspondentes da frase.

```
/* Sent. Preposicional = Preposicao + Sent. Nominal */
```

ANEXO 2

Programa número 2 para a análise sintatica da frase, este programa foi construído de uma forma mais genérica que o anterior (anexo1), aqui estão incluídos, argumentos a serem utilizados pela análise da consistências de número e gênero da sentença. O predicado roda1 é o mesmo do programa anterior, no predicado roda2 ocorrem modificações apenas na chamada do predicado de análise sintática.

```
roda1:-
 write('\n Escreva uma sentenca:\n '),
 readln(LIN), LIN =,
 roda2(LIN), !,
 roda1.

roda2(LIN):-!,
 lefrase(LIN,LSTPAL),
 analise_sintatica ( ['SENTENCA'],LSTPAL, Resto, Frase, Sent, Lista_consti, Lista_num,
 Lista_gen),
 tom (Resto).

roda2(_).
```

O predicado analise_sintatica tem a função principal de obter a estrutura sintática da frase, contendo todos os passos da gramática (generativa),

O predicado analise_sintática posssui oito argumentos, com o seguinte formato:

analise_sintatica (Constituintes, Lista de Palavras, Resto de Palavras, Frase reconstruida, Estrutura Sintática, Lista de Divisão de Constituintes,Número dos Terminais, Gênero dos Terminais).

Onde a função de cada um deles é a seguinte:

```
s_sp([PREP|LSTPAL],LSTPAL1,sprep(PREP,SN)):-
  e_preposicao(PREP),
  s_sn(LSTPAL,LSTPAL1,SN).
/* Sent. Verbal = Verbo + Sent. Nominal
Sent. Verbal = Verbo + Sent. Preposicional
Sent. Verbal = Verbo */
s_sv([VERBO|LSTPAL],LSTPAL1,sv(VERBO,SN)):-
  e_verbo(VERBO),
  s_sn(LSTPAL,LSTPAL1,SN).
s_sv([VERBO|LSTPAL],LSTPAL1,sv1(VERBO,SP)):-
  e_verbo(VERBO),
  s_sp(LSTPAL,LSTPAL1,SP).
```

Constituinte - é uma lista contendo o(s) nome(s) do(s) a ser(em) testado(s), por exemplo: ["SENTENÇA"] ou ["SENTENÇA NOMINAL","SENTENÇA VERBAL"]. Lista de string.

Lista de Palavras - é a frase a ser analisada representada em forma de lista. É uma lista de string.

Resto de Palavras - argumento auxiliar, que possui a função de representar a parte da Lista de Palavras que ainda falta ser analisada. Lista de string

Frase Reconstruida - representa trechos reconstruidos da frase até que se chegue a frase original, isto é feito durante o backtracking e servirá para armazenar todos os passos feitos pelo analisador sintático. É uma string.

Estrutura Sintática - é o argumento principal do predicado, que irá representar todas as fases da analise sintática. Cada fase é representada por uma lista de strings composto pelo nome do constituinte e pela frase (ou palavra) correspondente. A estrutura sintática completa será uma lista de lista de string.

Lista de divisão de Constituintes - é utilizado junto com o argumento anterior, contém as divisões do constituinte ou a palavra "terminal" para representar um constituinte terminal. Tambem é uma lista de lista de string.

Número dos Terminais - serve para a posterior análise consistência numérica da frase, guarda o número de cada palavra, ou seja, dos terminais da frase, é uma lista de string.

Gênero dos Terminais - é uma lista de string onde o gênero dos terminais da frase são armazenados para uma posterior análise da frase.

Em relação a ser de entrada ou saída os argumentos são da seguinte forma:

analise_sintatica (entrada, entrada, saída, saída, saída, saída, saída, saída),

A inicialização do predicado é feito com o argumento Constituintes tendo o valor ["SENTENÇA"] e a Lista de Palavras da frase a ser analisada. O predicado é composto de três claúsulas, cujos funcionamentos é o seguinte:

A primeira claúsula serve para determinar o fim da lista de constituintes de determinada parte da sentença, isto é, caso o argumento Constituintes seja uma lista vazia ([]), será dado os valores os valores para os predicados de saída, segundo representado na claúsula abaixo.

```
analise_sintatica ([],RR",[],[],[],[]):-!.
```

A claúsula seguinte acha os constiuintes terminais da sentença, através do predicado de base de dados bd_terminal, verificando se o constituinte Tipo é um constituinte terminal.

Caso seja verdade descobre-se qual o número e o gênero da palavra, que durante o backtracking serão colocados nos argumentos Número dos Terminais e Gênero dos Terminais, respectivamente. Também durante o backtracking são acrescentados o Tipo e a Palavra no argumento Estrutura Sintática, assim como a palavra terminal no argumento Lista de Divisão de Constituintes.

Em seguida chama-se recursivamente o predicado analise sintática com o Resto da lista de constituintes e o Resto_pal (da Lista de Palavras). O predicado concat serve para a construção da Frase a ser utilizada na reconstrução dos passos do programa.

```
analise_sintatica ([Tipo|Resto], [Palavra|Resto_pal], Resto_pal2, Frase2,
[[Tipo,Palavra]|Lista_duplas], [["terminal"]|LISTA_CONST], [NUMERO|LISTA_NUM],
[GENERO|LISTA_GEN]):-
  bd_terminal (Lista_terminais),
  member (Tipo,Lista_terminais),
  busca_palavra (Palavra,Tipo,NUMERO,GENERO),
```

```
analise_sintatica (Resto, Resto_pai, Resto_pai2, Frase, Lista_duplas, LISTA_CONSTI,
LISTA_NUM, LISTA_GEN),
concat (Palavra," ",Palavra2),
concat (Palavra2,Frase,Frase2),!.
```

A ultima claúsula funciona para os constiuintes que não são terminais, pega os constituintes imediatos do "Tipo", através do predicado de base de dados bd_nao_terminal e fazendo recursivamente o processo com a chamada do predicado analise_sintatica para esta divisão encontrada. E da mesma forma que a claúsula anterior o predicado analise_sintatica é chamado outra vez para o Resto da lista de Constituintes.

O predicado junta_listas liga as listas encontradas nos dois prediicados analise_sintatica chamados para a criação de listas unicas para os argumentos do predicado, a mesma função tem o predicado concat para as frases.

```
analise_sintatica ([Tipo|Resto], Frase, Resto_pai2, Frase2,
[[Tipo,Frase_cab]|LISTA_DUPLAS3], [Constituintes|LISTA_CONSTI3],
LISTA_NUM3,LISTA_GEN3):-
  bd_nao_terminal (Tipo,Constituintes),
  analise_sintatica (Constituintes,Frase, Resto_pai, Frase_cab, LISTA_DUPLAS2,
  LISTA_CONSTI1,LISTA_NUM1,LISTA_GEN1),
  analise_sintatica (Resto,Resto_pai, Resto_pai2,Frase_anterior, LISTA_DUPLAS,
  LISTA_CONSTI2,LISTA_NUM2,LISTA_GEN2),
  junta_listas (LISTA_NUM1,LISTA_NUM2,LISTA_NUM3),
  junta_listas (LISTA_GEN1,LISTA_GEN2,LISTA_GEN3),
  junta_listas (LISTA_DUPLAS2,LISTA_DUPLAS,LISTA_DUPLAS3),
  junta_listas (LISTA_CONSTI1,LISTA_CONSTI2,LISTA_CONSTI3),
  concat (Frase_cab,Frase_anterior,Frase2).
```

Um exemplo de como ficaria uma análise sintática de uma frase é mostrado a seguir:

exc o dolar no paralelo subiu cinco cruzeiros

sentenca = o dolar em o paralelo subiu cinco cruzeiros = sintagma nominal + sintagma verbal,

sintagma nominal = o dolar em o paralelo = determinante + nome + complemento nominal,

determinante = o = terminal - masculino, singular,

nome = dolar = terminal - masculino, singular,

complemento nominal = em o paralelo = sentença preposicional,

sentença preposicional = em o paralelo = preposição + sintagma nominal,

preposição = em = terminal - neutro, neutro,

sintagma nominal = o paralelo = determinante + nome,

determinante = o = terminal - masculino, singular,

nome = paralelo = terminal - masculino, singular,

sintagma verbal = subiu cinco cruzeiros = verbo + sintagma nominal,

verbo = subiu = terminal - singular,

sintagma nominal = cinco cruzeiros = numeral + nome,

numeral = cinco = terminal - neutro, plural,

nome = cruzeiros = terminal - neutro, plural,

Os constituintes terminais estão colocados no predicado bd_terminal.

```
bd_terminal(["NOME", "ADJETIVO", "PREPOSICAO","ARTIGO", "CONJUNCAO",
"PRONOME RELATIVO", "PRONOME PESSOAL", "PRONOME POSSESSIVO",
"NUMERAL", "QUANTIFICADOR", "VERBO"]),
```

A base de dados bd_nao_terminal indica os componentes não terminais seguido de suas respectivas divisões.

```
bd_nao_terminal ("SENTENCA",["SN","SV"]),
bd_nao_terminal ("SENTENCA",["SV"]),
bd_nao_terminal ("SENTENCA",["SENTENCA","SENTENCA2"]),
bd_nao_terminal ("SENTENCA2",["CONJUNCAO","SENTENCA"]),
bd_nao_terminal ("SN",["NOME_AUX","COMPLEMENTO NOMINAL"]),
bd_nao_terminal ("SN",["NOME_AUX"]),
```

```
d_nao_terminal ("NOME_AUX",["SDETERM","NOME"]),
bd_nao_terminal ("NOME_AUX",["NOME"]),
bd_nao_terminal ("COMPLEMENTO NOMINAL",["SADJETIVA"]),
bd_nao_terminal ("COMPLEMENTO NOMINAL",["SADITIVA"]),
bd_nao_terminal ("COMPLEMENTO NOMINAL",["SPREPOS"]),
bd_nao_terminal ("COMPLEMENTO NOMINAL",["SRELATIVO"]),
bd_nao_terminal ("SDETERM",["PREDET","ARTIGO","POSDET"]),
bd_nao_terminal ("SDETERM",["ARTIGO","POSDET"]),
bd_nao_terminal ("SDETERM",["PREDET","ARTIGO"]),
bd_nao_terminal ("SDETERM",["ARTIGO"]),
bd_nao_terminal ("SDETERM",["PREDET","POSDET"]),
bd_nao_terminal ("SDETERM",["PREDET"]),
bd_nao_terminal ("SDETERM",["POSDET"]),
bd_nao_terminal ("PREDET",["QUANTIFICADOR","PREPOSICAO"]),
bd_nao_terminal ("PREDET",["QUANTIFICADOR"]),
bd_nao_terminal ("POSDET",["PRONOME POSSESSIVO","NUMERAL"]),
bd_nao_terminal ("POSDET",["PRONOME POSSESSIVO"]),
bd_nao_terminal ("POSDET",["NUMERAL"]),
bd_nao_terminal ("SRELATIVO",["PRONOME RELATIVO","SV"]),
bd_nao_terminal ("SPREPOS",["PREPOSICAO","SN"]),
bd_nao_terminal ("SADJETIVA",["ADJETIVO","SPREPOS"]),
bd_nao_terminal ("SADJETIVA",["ADJETIVO"]),
bd_nao_terminal ("SADITIVA",["CONJUNCAO","SN"]),
bd_nao_terminal ("SV",["VERBO","COMPLEMENTO VERBAL"]),
bd_nao_terminal ("SV",["VERBO"]),
bd_nao_terminal ("COMPLEMENTO VERBAL",["SN"]),
```

ANEXO 3

Este programa representa a análise consitência da sentença quanto ao número e ao gênero, é complementar ao programa do anexo 2 de análise sintática. O predicado roda2 está repetido aqui para mostrar como os predicados específicos são chamados.

```
roda2(LIN):-!,
  lefrase(LIN,LSTPAL),
  analise_sintatica ( ["SENTENCA"],LSTPAL, Resto, Frase, Sent, Lista_consti, Lista_num,
  Lista_gen),
  tom (Resto),
  consistencia_numero (Sent,Lista_consti,Lista_num,Lnum),
  consistencia_genero (Sent,Lista_consti,Lista_gen,Lgen).

roda2(_).
```

Em relação a programação utilizado o predicado consistencia_numero, que é composto de quatro argumentos, mostrados a seguir:

consistencia_numero (Estrutura Sintática, Lista de Divisão de Constituintes, Número dos Terminais, Número dos Constituntes)

Estrutura Sintática - Lista de duplas de string com o nome do constituinte e a frase correspondente, argumento oriundo do predicado analise_sintatica;

Lista de Divisão de Constituintes - Lista de lista com a divisao do constituinte da lista do argumento anterior, também do predicado analise_sintatica;

Número dos Terminais - Lista dos números das palavras terminais, encontrada no predicado analise_sintatica;

Número dos Constituntes - Lista com o numero resultante de cada frase dos constituintes, este argumento é que realmente dará o resultado da análise de consistência do constituinte em questão;

Em relação a saída e entrada o predicado tem a seguinte forma:

consistencia_numero (entrada, entrada, entrada, saída, saída)

O predicado é inicializado com valores vindo da analise sintática (predicado roda2) da frase, no total existem sete claúsulas, sendo uma para descobrir o termino da analise, uma para as palavras terminais, uma para os casos normais e o restante para as excessões, cada uma delas será mostrada detalhadamente a seguir:

Esta primeira claúsula serve para indicar o término das listas, inicialiando as outras para serem construidas no backtracking.

```
consistencia_numero ([],[],[],[]):-!.
```

Esta segunda coloca nas listas o número de palavras terminais. E o predicado é chamado recursivamente.

```
consistencia_numero ([_|Constituinte],[["terminal"]|Lista_const],
[Numero|Lista_num],[Numero|Lista_numeros]):-
  consistencia_numero (Constituinte,Lista_const,Lista_num,Lista_numeros).
```

Para o caso de ser uma SENTENCA composta por Sintagma Nominal + Sintagma Verbal, isto é feito através da Divisão da SENTENÇA, somente a existência de SN (sintagam nominal) é verificada, pois todas as sentenças obrigatoriamente possuem sintagma verbal. Chama-se o predicado recursivamente, para os demais componentes da sentença.

Depois de se verificar as partes mais elementares da sentença, a concordância é então comparada entre o sintagma nominal e o verbo, através do predicado verifica_numero_verbo. No Número dos Constituintes é colocada a palavra FIM, para indicar que não necessita de mais nenhuma comparação.

```
consistencia_numero ([["SENTENCA",Frase]|Constituinte], [Divisao|Lista_const],
Lista_num, ["FIM"|Lista_numeros]):-
```

```
Divisao = ["SN"|_],
consistencia_numero (Constituinte,Lista_consti,Lista_num,Lista_numeros),
verifica_numero_verbo (Lista_numeros,Frase).
```

Para o caso de ser uma SENTENCA, e ter falhado na claúsula anterior, apenas se coloca a palavra FIM na lista Numero dos constituintes, e chama-se o predicado recursivamente.

```
consistencia_numero ([["SENTENCA",_]|Constituinte], [_|Lista_consti], Lista_num,
["FIM"|Lista_numeros]):-
  consistencia_numero (Constituinte,Lista_consti,Lista_num, Lista_numeros).
```

Para o caso de ser uma Sentenca Verbal (SV), é colocado o número correspondente ao verbo na lista Número dos Constituintes, então a anállse de consistência é chamada para os constituintes restantes da frase.

```
consistencia_numero ([["SV",_]|Constituinte],[_|Lista_consti], Lista_num,
[Numero|Lista_numeros]):-
  consistencia_numero (Constituinte,Lista_consti,Lista_num,Lista_numeros),
  Lista_numeros = [Numero|_].
```

Claúsula utilizada para uma Sentenca Nominal (SN) que possua um COMPLEMENTO NOMINAL, que consequentemente é uma SENTENÇA ADITIVA (isto é, possui uma conjunção aditiva), então está sentença nominal terá PLURAL como número.

Os passos detalhados no programa é da seguinte forma: em primeiro lugar, através do predicado member a existência de complemento nominal na Divisão do sintagma nominal é verificada. Em seguida, pega-se a posição do elemento que possua complemento nominal na lista de constituintes, isto no predicado orderistist (fornece a posição, dado o elemento ou o elemento, quando é dada a posição em uma lista de lista de string). Com a posição descoberta, o predicado orderistist verifica se o complemento nominal é uma sentença aditiva.

```
consistencia_numero ([["SN",_]|ConstituInte],[Divisao|Lista_const],Lista_num,
["PLURAL"|Lista_numeros]):-
  member ("COMPLEMENTO NOMINAL",Divisao),
  orderlistlst (Num,["COMPLEMENTO NOMINAL",_],ConstituInte),
  orderlistlst (Num,["SADITIVA"],Lista_const),
  consistencia_numero (ConstituInte,Lista_const,Lista_num, Lista_numeros).
```

Para os casos restantes, é verificado se todas as palavras são do mesmo tipo, através do predicado verifica_lista_numero. O restante da claúsula é feito de forma similar aos anteriores. Primeiro chama-se recursivamente o predicado consistencia_numero para se chegar ao ultimo elemento da sentença. Quando se está fazendo o backtracking, o número do ultimo constituinte entrado será acrescentado na Lista de Palavras através do predicado correspondente_frase. E por fim com a lista de palavras já atualizada, chama-se o predicado verifica_lista_numero para ver se todas as palavras do constituintes possuem o mesmo número.

```
consistencia_numero ([[_,Frase]|ConstituInte],[Divisao|Lista_const],Lista_num,
[Numero|Lista_numeros]):-
  consistencia_numero (ConstituInte,Lista_const,Lista_num, Lista_numeros),
  correspondente_frase (Divisao,ConstituInte,Lista_numeros,Lista2),!,
  verifica_lista_numero (Lista2,Numero,Frase).
```

O predicado consistencia_genero funciona de forma identica ao consistencia_numero, a única diferença é que a concordancia em gênero é verificada somente entre determinantes, nomes e adjetivos, indicando um erro nestes casos. Por exemplo: "A preço" ou " A inflação alto".

```
consistencia_genero ([],[],[],[],[]):-!.

consistencia_genero ([_|ConstituInte],[["terminal"]|Lista_const],
[Genero|Lista_gen],[Genero|Lista_generos]):-
  consistencia_genero (ConstituInte,Lista_const,Lista_gen, Lista_generos).
```

```
consistencia_genero([["NOME_AUX",Frase]|Constituinte], [Divisao|Lista_const],
Lista_gen, [Genero|Lista_generos]):-
  consistencia_genero (Constituinte,Lista_const,Lista_gen,Lista_generos),
  correspondente_frase (Divisao,Constituinte,Lista_generos,Lista2),!,
  verifica_lista_genero (Lista2,Genero,Frase).

consistencia_genero ([_|Constituinte],[_|Lista_const], Lista_gen,
["nada"|Lista_generos]):-
  consistencia_genero (Constituinte,Lista_consti,Lista_gen,Lista_generos).
```

Predicado correspondente_frase serve para construir uma lista com o valor (numero ou genero) de cada divisão do constituinte para o qual está sendo feita a análise, é composta de quatro argumentos:

correspondente_frase (Lista de Divisão de Constituintes, Estrutura Sintática, Lista dos valores anteriores, Lista do valor resultante)

Como este predicado está colocado em uma posição recursiva em relação ao predicado consistencia-..., portanto os argumentos de entrada, por serem todos em forma de lista, representarão somente a cauda da lista principal referentes a elas.

Lista de Divisão de Constituintes - Lista de lista com a divisao do constituinte da lista do argumento anterior, também do predicado analise_sintatica;

Estrutura Sintática - Lista de duplas de string com o nome do constituinte e a frase correspondente, argumento oriundo do predicado analise_sintatica;

Lista dos valores anteriores - Lista com os valores correspondentes aos constituintes que ja passaram por este predicado;

Lista de valores resultante - Lista resultante com os valores correspondentes ao constituinte em questão.

A primeira claúsula indica o final da análise.

```
correspondente_frase ([],_,_,[]):-!.
```

Na calúsula principal, o predicado orderlistlst, pega a posicao Num referente ao constituinte. A posicao Num é utilizada para se achar a lista de valores do constituinte, correspondente a Divisão, na Lista dos valores anteriores. O predicado é chamado recursivamente, até que se acabe as divisões, e por fim as listas são ligadas pelo predicado junta_listas.

```
correspondente_frase ([Divisao|Resto],Constituinte,Lista,Lista3) :-
    orderlistlst (Num,[Divisao,_],Constituinte),
    orderlistlst (Num,Sublista,Lista),
    correspondente_frase (Resto,Constituinte,Lista,Lista2),
    junta_listas (Sublista,Lista2,Lista3).
```

O predicado verifica_lista_numero serve para verificar se todas as palavras de um contituinte pertencem ao mesmo tipo, eliminando-se os casos NEUTRO e INDETERMINADO, caso tenha-se um erro na concordância a palavra ERRO é retornada. É formado por:

obs: número aqui se refere a ser plural ou singular.

verifica_lista_numero (Lista dos números, Número resultante, Frase),

Lista dos números - Lista dos números dos contituintes terminais

Frase - frase correspondente ao constituinte que esta sendo verificada, serve para ser apresentada caso exista algum erro de concordância de número.

A primeira claúsula indica o fim da busca, na existência de somente um número na lista, e este número é retornado

```
verifica_lista_numero ([N],N,_) :-!.
```

Claúsula para o caso em que o número seja diferente de PLURAL ou SINGULAR, o qual nao é levado em conta. valendo então o número encontrado pelo backtracking

```
verifica_lista_numero ([Numero|Resto],Numero2,Frase) :-
  Numero <> "PLURAL",
  Numero <> "SINGULAR",
  verifica_lista_numero (Resto,Numero2,Frase).
```

Caso a claúsula anterior falhe então, esta claúsula irá verificar (durante o backtracking) se todos os números dos constituintes são iguais.

```
verifica_lista_numero ([Numero|Resto],Numero,Frase) :-
  verifica_lista_numero (Resto,Numero,Frase).
```

Caso não seja igual, pode ter ocorrido o caso da última palavra da lista ter o número igual a NEUTRO ou INDETERMINADO, então o troca-se pelo da palavra corrente

```
verifica_lista_numero ([Numero2|Resto],Numero2,Frase) :-
  verifica_lista_numero (Resto,Numero,Frase),
  Numero <> "PLURAL",
  Numero <> "ERRO",
  Numero <> "SINGULAR".
```

Caso tenha falhado em todos os casos anteriores é porque ocorreu alguma espécie de erro de concordância, este erro é indicado com a apresentação da frase errada, e retorna-se a palavra ERRO

```
verifica_lista_numero (_,"ERRO",Frase) :-
  write("Erro de concordancia de numero na frase: \n ",Frase),
  readchar(_),nl,nl.
```

O predicado verifica_lista_genero funciona de forma idêntica ao anterior a única modificação é que as comparações são feitas para MASCULINO e FEMININO.

```
verifica_lista_genero ([G],G,_) :-!.
```

```
verifica_lista_genero ([Genero|Resto],Genero2,Frase) :-
  Genero "MASCULINO",
  Genero "FEMININO",
  verifica_lista_genero (Resto,Genero2,Frase).

verifica_lista_genero ([Genero|Resto],Genero,Frase) :-
  verifica_lista_genero (Resto,Genero,Frase).

verifica_lista_genero ([Genero2|Resto],Genero2,Frase) :-
  verifica_lista_genero (Resto,Genero,Frase),
  Genero "MASCULINO",
  Genero "ERRO",
  Genero "FEMININO".

verifica_lista_genero (_,"ERRO",Frase) :-
  write("Erro de concordancia de genero na frase: \ n ",Frase),
  readchar(_),nl,nl.
```

ANEXO 4

O predicado analise_semantica será colocado no fim do predicado roda2, este predicado funcionará como uma fase intermediária entre o analisador da sentença e o sistema computacional propriamente dito.

```
roda2(LIN2):-!,
  lefrase(LIN,LSTPAL),
  analise_sintatica ( ["SENTENCA"],LSTPAL, Resto, Frase, Sent2, Lista_const!, Lista_num,
  Lista_gen),
  consistencia_numero (Sent2,Lista_const!,Lista_num,Lnum),!,
  consistencia_genero (Sent2,Lista_const!,Lista_gen,Lgen),!,
  analise_semantica (Sent2,Lista_const!),!.

roda2(_).
```

O predicado de análise semantica serve para iniciar a fase de análise semântica, onde a frase será transformada em instruções computacionais, para ser ligado ao programa para o qual a interface será utilizada

A primeira claúsula detecta a existência de mais de uma oração, separando-as, através do predicado separa_sentencas, em seguida um predicado auxiliar (analise_semantica2) é chamado, este predicado manda as instruções que são de entradas em primeiro lugar para depois mandar as de saida.

```
analise_semantica (SENTENCA, [DIVISAO|LISTA_CONST] ) :-
  order (NUM, "SENTENCA2", DIVISAO),
  remove (NUM, DIVISAO, RESULTADO),!,
  separa_sentencas ( SENTENCA, [RESULTADO|LISTA_CONST] ,SENT2, LSTCST2,
  SENT3, LSTCST3 ),
  analise_semantica2 ("ENTRADA",SENT2,LSTCST2 ),
```

```
analise_semantica2 ("ENTRADA",SENT3,LSTCST3 ),
analise_semantica2 ("SAIDA", SENT2,LSTCST2 ),
analise_semantica2 ("SAIDA", SENT3,LSTCST3 ).
```

Esta claúsula serve para sentenças com somente uma oração, tem a funçãoo de dar uma primeira "limpada" na frase, retirando palavras não necessárias, e chama a interface com o sistema

```
analise_semantica (SENTENCA, LISTA_CONST ) :-
  limpa_ruido (SENTENCA,LISTA_CONST,LIMPO),
  interface_sistema (LIMPO).
```

Auxiliar ao analise_semantica, o predicado analise semantica2 serve verificar se o verbo é do tipo (Saída ou Entrada) que está sendo procurado, caso seja verdade a sentenca é mandada outra vez para análise semântica, que terá duas opções ou subdividir a sentença, caso existam outras orações dentro dela, ou ajeita a oração para se chamar a interface com o sistema.

```
analise_semantica2 (Tipo, SENT, LSTCST ) :-
  orderlistlst (Num,["VERBO",_], SENT ),
  orderlistlst (Num,[_,Verbo], SENT ),
  frame_verbo (Verbo,_,_,Tipo),
  analise_semantica (SENT , LSTCST).

analise_semantica2 (_,_,_).
```

Predicado que para o caso em que a sentenca é composta por mais de uma oração, separa-as assim que surge a segunda oração, e recontroi as listas de constituintes e de divisões de constituintes.

```
separa_sentencas ([["SENTENCA2",_]|RESTO], [_|LSTCST], [], [], SENT,LSTCST2):-
  separa_sentencas (RESTO,LSTCST,[["LIVRE"]],[],SENT,LSTCST2).

separa_sentencas ( [TERMO|SENTENCA], [DIVISAO|LISTA_CONST1], [TERMO|SENT2],
[DIVISAO|LSTCST2], SENT3, LSTCST3 ):-
  separa_sentencas (SENTENCA, LISTA_CONST1, SENT2, LSTCST2, SENT3, LSTCST3).

separa_sentencas ([["SENTENCA",FRASE]|RESTO],LSTCST,[["LIVRE"]],[],
[["SENTENCA",FRASE]|RESTO],LSTCST):-!.

separa_sentencas ([_|SENTENCA],[_|LISTA_CONST1] ,[["LIVRE"]],[],SENT3, LSTCST3 ):-
  separa_sentencas (SENTENCA, LISTA_CONST1, [["LIVRE"]],[], SENT3, LSTCST3).
```

O predicado limpa_ruido serve para retirar as palavras e partes da sentença que não será necessária para a análise da sentença. Além disto, substitui as palavras por uma outra equivalente que o programa possa entender e coloca a função da palavra.

```
limpa_ruido ([],[],[]) :- !.

limpa_ruido ([["VERBO",VERBO]|RESTO1],[["terminal"]|RESTO2],
[["VERBO",ORIGEM]|Lista]):-
  tipo_palavra ("VERBO",VERBO,ORIGEM,_),
  limpa_ruido (RESTO1,RESTO2,Lista).

limpa_ruido ([[TIPO,_]|RESTO1],[["terminal"]|RESTO2],Lista) :-
  bd_ruido (RUIDOS),
  member (TIPO,RUIDOS),
  limpa_ruido (RESTO1,RESTO2,Lista).

limpa_ruido ([[Tipo,PALAVRA]|RESTO1],[["terminal"]|RESTO2],
[[Funcao,Origem]|Lista]):-
  tipo_palavra (Tipo,PALAVRA,Origem,Funcao),
  Funcao <> "NENHUMA",
```

```
limpa_ruido (RESTO1,RESTO2,Lista).

limpa_ruido ([_|RESTO1],[_|RESTO2],Lista):-
limpa_ruido (RESTO1,RESTO2,Lista).
```

Este predicado é que irá chamar o programa principal, no caso o Sistema de Administração de Capital de Giro, primeiro a oração é dividida em verbo, pré verbo e pós verbo. Através do predicado agente_preverbo em cada elemento da lista PreVerbo é colocada a palavra "AGENTE", a lista resultante é ligada com a lista de PosVerbo.

Em seguida são pego os dados relativos ao verbo, pelo predicado de base de dados frame_verbo.

O frame_verbo é da seguinte forma:

frame_verbo (Nome do verbo, Lista dos elementos obrigatorias, Lista dos opcionais, Tipo - entrada ou saída)

A lista de elementos obrigatorios contém as funções das palavras que são imprescindiveis para o verbo. Um exemplo seria o verbo apresentar, é obrigatório a presença de um objeto (resultado apresentar objeto). A lista de opcionais contém elementos que não são obrigatório e seu default, para o caso da inexistência. Para o mesmo verbo apresentar, uma opção seria o lugar (impressora ou tela). Tipo corresponde as características do verbo, se tem função de saída de entrada. Um exemplo está mostrado abaixo:

```
frame_verbo ("APRESENTAR",["OBJETO"],[["LUGAR","TELA"]],"SAIDA")
```

O predicado monta_lista colocara cada elemento da lista em seu lugar. O verbo é colocado como cabeça da lista e o resultado é enviado para o predicado faz, que já é um predicado pertencente ao Sistema de Administração de Capital de Giro.

```
interface_sistema (SENT):-
  divide_sentenca (SENT,PreVerbo,Verbo,PosVerbo),
  agente_preverbo (PreVerbo, PreVerbo2),
  junta_lista (PreVerbo2,PosVerbo,Resultado),
  frame_verbo (Verbo,L1,L2,_),
  monta_lista (L1,L2,Resultado,Lista2),
  Lista = [Verbo|Lista2],
  faz (Lista).
```

Predicado divide_sentenca, tem a funcao de separar a sentenca em tres partes, o verbo, o que vem antes do verbo e o que vem depois do verbo.

```
divide_sentenca ([],[],"",[]) :- !.

divide_sentenca ([["VERBO",VERBO]|RESTO1],[],VERBO,POS_VERBO) :-
  divide_sentenca (RESTO1,POS_VERBO,_,_).

divide_sentenca ([LISTA|RESTO1],[LISTA|PRE_VERBO],VERBO,POS_VERBO) :-
  divide_sentenca (RESTO1,PRE_VERBO,VERBO,POS_VERBO).
```

O predicado agente_preverbo apenas inclui a palavra "AGENTE" nos elementos que vem antes do verbo, indicando que a palavra mais uma função da palavra em relação ao verbo.

```
agente_preverbo([],[]) :-!.

agente_preverbo([Lista|Resto],["AGENTE"|Lista]|Retorno]):-

agente_preverbo (Resto,Retorno).
```

```
monta_lista([],[],_,[]):-!.

monta_lista ([],[[Funcao|_]|Resto],Lista,[Valor|Retorno]):-
  pega_funcao (Funcao,Lista,Valor),
  monta_lista ([],Resto,Lista,Retorno).

monta_lista ([],[Funcao|Resto],Lista,[Default|Retorno]):-
  size (Tam,Funcao),
  order (Tam,Default,Funcao),
  monta_lista ([],Resto,Lista,Retorno).

monta_lista ([Funcao1|Resto],Funcao2,Lista,[Valor|Retorno]):-
  pega_funcao (Funcao1, Lista,Valor),
  monta_lista (Resto,Funcao2,Lista,Retorno).

monta_lista ([Funcao|_],_,_,_) :-
  write ("Erro semantico. Falta um(a) ",Funcao),!,fail.

pega_funcao (Funcao,[Lista|_],Valor):-
  member(Funcao,Lista),
  size (Tam,Lista),
  order (Tam,Valor,Lista),!.

pega_funcao (Funcao,[_|Resto],Valor) :-
  pega_funcao (Funcao,Resto,Valor).
```

REFERÊNCIAS BIBLIOGRÁFICAS:

[ARAG87] - DORIS FERRAZ ARAGON & ANTONIO CARLOS SARAIVA BRANCO

UM SISTEMA LÓGICO SINTÁTICO PARA ANÁLISE DE ASPECTOS DA LINGUAGEM NATURAL

ANAIS 4o SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL - 1987 - UBERLANDIA

[ASSI86] - FIDELIS SIGMARINA GOMES DE ASSIS

UMA FERRAMENTA DE AUXÍLIO NO DESENVOLVIMENTO DE INTERFACES DE LINGUAGEM NATURAL

IME - RJ - 1986 - DISSERTAÇÃO DE MESTRADO

[ASSI86] - FIDELIS SIGMARINDA G. ASSIS & EMMANUEL LOPES PASSOS

TRADUTOR PARA ESTRUTURAS LÓGICAS E INTERPRETADOR SEMÂNTICO

ANAIS 3o SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL - 1986 - RJ - pp 13

[BATE87] - M. BATES

NATURAL LANGUAGE INTERFACES

ENCYCLOPEDYA OF ARTIFICIAL INTELLIGENCE - pp 655 - 660 - 1987

[BENN89] - RAYMOND W. BENNETT AT ALLI

SPEAKING TO, FROM AND THROUGH COMPUTERS: SPEECH TECHNOLOGIES AND USER INTERFACE

AT&T TECHNICAL JOURNAL - SEPTEMBER/OCTOBER - pp 17 - 30 - 1989

[BERW87] - R. BERWICK

GRAMMAR, TRANSFORMATIONAL

ENCYCLOPEDYA OF ARTIFICIAL INTELLIGENCE - pp 353 - 361 - 1987

[BOLC87] - LEONARD BOLC

NATURAL LANGUAGES - PARSING SYSTEMS

ED. SPRINGER - VERLAG - 1987

[BRUC87] - B. BRUCE & M. G. MOSER

GRAMMAR, CASE

ENCYCLOPEDYA OF ARTIFICIAL INTELLIGENCE - pp 333 - 339 - 1987

[BURT87] - R. BURTON

GRAMMAR, SEMANTIC

ENCYCLOPEDYA OF ARTIFICIAL INTELLIGENCE - pp 351 - 353 - 1987

[CARB87] - JAIME G. CARBONELL & PHILIP J. HAYES

NATURAL LANGUAGE UNDERSTANDING

ENCYCLOPEDYA OF ARTIFICIAL INTELLIGENCE - pp 660 - 677 - 1987

[CARR87] - DAVID W. CARRAHER, PAULO H. BORBA & ANDRE LUIS SANTOS

ACESSO A UM BANCO DE DADOS ATRAVÉS DE LINGUAGEM NATURAL

ANAIS 4o SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL - 1987 - UBERLANDIA

[CARR87] - DAVID W. CARRAHER, V. MUTCHINIK & CID C. ALBUQUERQUE

A ANÁLISE MORFOLÓGICA E SINTÁTICA DO PORTUGUÊS

ANAIS 4o SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL - 1987 - UBERLANDIA

[CARV86] - MAURICIO BRITO DE CARVALHO

UMA INTRODUÇÃO A GRAMÁTICA DE CASOS

UNIVERSIDADE FEDERAL DE VIÇOSA- 1986

[CERI89] - STEFANO CERI & GEORG GOTTLOB & GIO WIEDERHOLD

EFFICIENT DATABASE ACCESS FROM PROLOG

IEEE TRANSACTIONS ON SOFTWARE ENGINEERING - VOL 15 No 2 - FEBRUARY - 1989-

[COEL87] - H. COELHO

GRAMMAR, DEFINITE - CLAUSE

ENCYCLOPEDYA OF ARTIFICIAL INTELLIGENCE - pp 339 - 342 - 1987

[COEL89] - HELDER COELHO & GABRIEL LOPES & ROSA VICCARI

MODELO PARA A CONCEPÇÃO DE GRAMÁTICAS EVOLUTIVAS

ANAIS 6o SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL - 1989 - RJ - pp 20

[CORD85] - JOSE LUIZ CORDEIRO

ANATOMIA DO PROLOG

IME - RJ - 1985 - DISSERTAÇÃO DE MESTRADO

[CUNH85] - CELSO CUNHA & LINDLEY CINTRA

NOVA GRAMÁTICA DO PORTUGUÊS CONTEMPORANEO

ED NOVA FRONTEIRA- 1985

[EVAS88] - STEVEN E. EVASON

HOW TO TALK TO AN EXPERT

AI EXPERT - FEBRUARY - 1988 - pp 36 - 52

[FALZ89] - PIERRE FALZON

ERGONOMIE COGNITIVE DU DIALOGUE

PRESSES UNIVERSITAIRES DE GRENOBLE - 1989

[FEIG81] - AVRON BARR & EDWARD FEIGENBAUM

THE HANDBOOK OF ARTIFICIAL INTELIGENCE

HEURISTECH PRESS- 1981

[FRAG86] - PALTONIO DAUN FRAGA

TRATAMENTO COMPUTACIONAL DE LÍNGUAS NATURAIS

ANAIS 3o SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL - 1986 - RJ - pp 16

[FU82] - K. S. FU

SYNTACTIC PATTERN RECOGNITION AND APPLICATIONS - 1982

[GAL85] - A. GAL & J. MINKER

A NATURAL LANGUAGE DATABASE INTERFACE THAT PROVIDES COOPERATIVE ANSWERS

THE 2o CONFERENCE ON ARTIFICIAL INTELLIGENCE APPLICATIONS - THE ENGINEERING

[GARC86] - A.C.B. GARCIA & E.B.S. CARVALHO & J.A. FORTUNY NETO

ASPECTOS DA IMPLEMENTAÇÃO DE UM PROCESSADOR DE LINGUAGEM NATURAL EM PASCAL

ANAIS 3o SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL - 1986 - RJ- pp 10

[GAZD87] - G. GAZDAR

GRAMMAR, GENERALIZED PHRASE STRUCTURE

ENCYCLOPEDYA OF ARTIFICIAL INTELLIGENCE - pp 342 - 344 - 1987

[GEET 90] - T.V. GEETHA & R.K SUBRAMANIAN

REPRESENTING NATURAL LANGUAGE WITH PROLOG

IEEE SOFTWARE - MARCH 90 - pp85 - 92 - 1990

[GONC86] - CARLOS ALBERTO GONCALVEZ

AQUISIÇÃO E REPRESENTAÇÃO DE CONHECIMENTO PARA SISTEMAS ESPECIALISTAS

FEA - USP- 1986 - TESE DE DOUTORADO

[GROS86] - BARBARA J. GROSZ, KAREN S. JONES & BONNIE L. WEBBER

READINGS IN NATURAL LANGUAGE PROCESSING

MORGAN KAUFMANN PUBLISHERS INC- pp XI - 1 - 161 - 1986

[HART89] - H. REX HARTSON & DEBORAH HIX

TOWARD EMPIRICALLY DERIVED METHODOLOGIES AND TOOLS FOR HUMAN-COMPUTER INTERFACE

INT J MAN-MACHINE STUDIES (1989) 31 - pp 477 - 494

[HORN84] - NORBERT HORNSTEIN

LOGIC AS A GRAMMAR - AN APROACH TO MEANING IN NATURAL LANGUAGE

MIT PRESS- 1984

[JACO87] - R. JACOB

HUMAN - COMPUTER INTERACTION

ENCYCLOPEDYA OF ARTIFICIAL INTELLIGENCE - pp 383 - 388 - 1987

[JOSH87] - A. JOSHI

GRAMMAR, PHRASE STRUCTURE

ENCYCLOPEDYA OF ARTIFICIAL INTELLIGENCE - pp 344 - 351 - 1987

[LIMA89] - LIENE B. LIMA & EMMANUEL P. L. PASSOS

INTERFACE AMIGÁVEL PARA SISTEMAS BASEADOS EM REGRAS

ANAIS XXII CONGRESSO NACIONAL DE INFORMATICA 1989 - pp 448 -454

[LUNA 91] - PAULO DE TARSO MENDES LUNA

SISTEMA DE APOIO Ã DECISÃO: UMA APLICAÇÃOO A ANÁLISE DA INFLUÊNCIA DA INFLAÇÃO NA ADMINISTRAÇÃO DO CAPITAL DE GIRO

UFSC- 1991 - DISSERTAÇÃO DE MESTRADO

[MCDO87] - D. D. MCDONALD

NATURAL LANGUAGE GENERATION

ENCYCLOPEDYA OF ARTIFICIAL INTELLIGENCE - pp 650 - 655 - 1987

[MIRA86] - CLAUDIO SILVA MIRANDA

PROCESSAMENTO DE LINGUAGEM NATURAL

IME - RJ- 1986 - DISSERTAÇÃO DE MESTRADO

[MOSC86] - PAULO ROBERTO FERRARI MOSCA

UMA GRAMÁTICA LIVRE DE CONTEXTO PARA CONSTRUCOES COM AUXILIARES

ANAIS 3o SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL - 1986 - RJ- pp 18

[McSK79] - JAMES R. McSKIMIN & JACK MINKER

A PREDICATE CALCULUS BASED SEMANTIC NETWORK FOR DEDUCTIVE SEARCHING

ASSOCIATIVE NETWORKS REPRESENTATION AND USE OF KNOWLEDGE BY COMPUTERS - 1979

[OLIV87] - CARLOS ALBERTO DE OLIVEIRA

A MORFOLOGIA E A SINTAXE: UM ENFOQUE INTEGRADO BASEADO NO CONHECIMENTO LINGÜÍSTICO

ANAIS 4o SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL - 1987 - UBERLANDIA

[OLIV89] - CARLOS ALBERTO OLIVEIRA

A SINTAXE,SEMÂNTICA E A PRAGMÁTICA: UM ENFOQUE INTEGRADO BASEADO NO CONHECIMENTO

ANAIS 6o SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL - 1989 - RJ - pp 21

[PAPE63] - SEYMOUR PAPERT

INTELIGÊNCIA ARTIFICIAL E MECANISMOS GERAIS DE DESENVOLVIMENTO

TEORIAS DA LINGUAGEM - TEORIAS DA APRENIZAGEM POR MASSIMO PIATTELLI-PALMARIN - 1983

[PAU86] - L.F. PAU

ARTIFICIAL INTELLIGENCE IN ECONOMICS AND MANAGEMENT: WHY?

ELSEVIER SCIENCE PUBLISHER - 1986

[PETR87] - S. PETRICK

PARSING

ENCYCLOPEDYA OF ARTIFICIAL INTELLIGENCE - pp 687 - 696 - 1987

[RAPO79] - EDUARDO PAIVA RAPOSO

INTRODUÇÃO A GRAMÁTICA GENERATIVA

MORAES EDITORES - 1979

[REIT84] - WALTER REITMAN ET ALLI

ARTIFICIAL INTELLIGENCE APPLICATION FOR BUSINESS

ABLEX PUBLISHING CORPORATION - NEW JERSEY- 1984

[RICH85] - ELAINE RICH

NATURAL LANGUAGE UNDERSTANDING: HOW NATURAL CAN IT BE?

THE 2o CONFERENCE ON ARTIFICIAL INTELLIGENCE APPLICATIONS - THE ENGINEERING

[RICH88] - BRADLEY L. RICHARDS

WHEN FACTS GET FUZZY

BYTE - APRIL - 1988- pp 265 - 290

[RIEB87] - C. K. RIEBSBECK

PARSING, EXPECTATION - DRIVEN

ENCYCLOPEDYA OF ARTIFICIAL INTELLIGENCE - pp 696 -701 - 1987

[RINO86] - LUCIA HELENA MACHAEDO RINO

UM ENFOQUE DE UMA INTERFACE EM LINGUAGEM NATURAL

ANAIS 3o SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL - 1986 - RJ- pp 12

[SANT90] - NERI DOS SANTOS

APOSTILA DA DISCIPLINA ERGONOMIA DAS INTERFACES HOMEM-COMPUTADOR

UFSC - DEPARTAMENTO DE ENGENHARIA DE PRODUÇÃO E SISTEMAS - 1990

[SCHI87] - HERBERT SCHILDT

ADVANCED TURBO PROLOG

ED. MC GRAW HILL - 1987

[SIQU86] - I. S. P. SIQUEIRA, A. E. C. PEREIRA & R G. COHN

UMA GRAMÁTICA CONEXIONISTA: PROPRIEDADES E APLICAÇÕES

ANAIS 3o SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL - 1986 - RJ - pp 11

[SIQU87] - IDMEA SEMEGHINI SIQUEIRA & HELMER MARTIN VELLOSO

INTERAÇÃO HOMEM - MÁQUINA DIANTE DE INTERFACE EM UNGUAGEM NATURAL COM PORTUGUÊS

ANAIS 4o SIMPÓSIO BRASILEIRO DE INTELIGÊNCIA ARTIFICIAL - 1987 - UBERLANDIA