Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica
Depto. de Semicondutores, Instrumentos e Fotônica

Este exemplar corresponde à redação final da tese defendida por GILBERTO ONOFRE TEDESCO e aprovada pela Comissão Julgadora em 14 / 12 / 90.

Orientador

UM ESTUDO SOBRE INTERFACES EM LINGUAGEM NATURAL, COM VISTAS À INTERAÇÃO ENTRE USUÁRIOS E BASES DE CONHECIMENTO

por

Gilberto Onofre Tedesco

Orientador: **Furio Damiani**

Dissertação apresentada à Faculdade de Engenharia Elétrica da Universidade Estadual de Campinas - UNICAMP, como parte dos requisitos exigidos para a obtenção do título de mestre em Engenharia Elétrica

Campinas (SP)
Dezembro de 1990

Este trabalho contou com o apoio financeiro
da Coordenadoria de Aperfeiçoamento de Pessoal de Ensino Superior (CAPES)
e do Banco do Brasil S.A.

Aos meus pais, irmãos e
à tia Ida

AGRADECIMENTOS

Ao prof. e amigo Furio Damiani por sua dedicação e pelo estímulo no trabalho de orientação, também pelo seu alto senso de respeito e compreensão, no acompanhamento das atividades realizadas.

À profª. Ariadne (IMECC- UNICAMP) pelas inúmeras contribuições e esclarecimentos dados no decorrer deste trabalho, além do material bibliográfico que forneceu.

Ao Paltônio Daun Fraga (USP), por sua ajuda, logo no início, com suas observações e material bibliográfico.

Aos colegas, professores e funcionários do DSIF, que proporcionaram um ambiente agradável e um relacionamento de alto nível, muito construtivo, em todos os aspectos.

Aos colegas e professores da pós-graduação, pela boa convivência dentro e fora da Universidade.

Aos amigos da república, que me suportaram pacientemente nestes anos, e que muito me ensinaram.

Aos funcionários do Banco do Brasil: do SUPORTE I - Campinas (SP) e do DESED, que foram muito prestativos e atenciosos no atendimento às minhas necessidades com relação ao Banco, durante o último ano do curso.

Ao João Manuel, ao Nogueira e demais amigos do movimento dos Focolari, pelo apoio constante e ajuda nos momentos mais difíceis.

Ao bom Pai, fonte de todo bem.

SUMÁRIO

A finalidade deste trabalho é o estudo e a aplicação de métodos de processamento de linguagem natural para que se possam construir interfaces entre a linguagem do usuário e a linguagem usada em sistemas de computação. Entende-se como linguagem natural a linguagem verbal, mais especificamente, sua representação escrita, com vistas unicamente ao seu processamento computacional.

Uma interface em linguagem natural, em grande parte, depende do domínio de conhecimento. Procuramos dar ênfase à sua portabilidade para diferentes domínios de aplicação.

Como exemplo, implementamos uma interface para que se possa acessar uma base de conhecimentos constituída de informações sobre os arquivos em discos de um microcomputador PC. A estrutura deste interpretador compõe-se de um conjunto de módulos interativos, tendo como entrada a solicitação do usuário e como saída a resposta a sua consulta ou a execução de algum comando.

Um outro aspecto que procuramos salientar é a possibilidade de expansão do dicionário e da base de dados pelo próprio usuário, durante sua interação com o sistema, dispensando a intermediação do profissional da computação para implementar tais modificações.

ÍNDICE

1

INTRODUÇÃO

Em um ambiente de software integrado de projetos, um fator importante para que se tenha um bom desempenho é a interação do especialista com o sistema computacional. A busca de meios para facilitar a construção de interfaces em linguagem natural que permitam esta interação é o objetivo deste trabalho.

Como ponto de partida, abordamos o processamento da linguagem natural, mais especificamente de sentenças escritas da língua portuguesa. Procuramos fazer uma tradução das sentenças do usuário para representações internas de uma linguagem artificial, que tenha uma sintaxe e semântica bem precisas.

O processamento de linguagem natural exige que se manipule uma variedade de conhecimentos, tais como: a morfologia e o significado das palavras, a estrutura das sentenças, as regras de conversação e ainda um grande apanhado de informações gerais sobre o universo do discurso.

Descrevemos as técnicas usadas no processamento de linguagem natural, procurando enfatizar o uso do formalismo lógico, seja para representar a gramática e executar a análise de uma sentença, seja para representar o significado desta sentença.

Neste capítulo introdutório procuramos expor, de maneira genérica, a motivação para o uso de interfaces em linguagem natural, assim como as técnicas empregadas para o seu processamento. Mostramos também os componentes envolvidos e descrevemos os módulos que geralmente compõem a arquitetura de um tal sistema. Comentamos os requisitos e as limitações inerentes a este processamento.

No segundo capítulo apresentamos o processamento sintático, fazendo inicialmente, um apanhado geral de vários formalismos gramaticais e, em seguida, descrevendo mais detalhadamente as gramáticas lógicas.

No capítulo três, onde o enfoque é o processamento semântico, mostramos primeiramente uma caracterização do que seja a semântica no contexto do processamento de linguagem natural. Partindo-se do princípio da composicionalidade e da associação do significado à atribuição de valores-verdade, apresentamos a abordagem de se "traduzir" o significado de uma sentença em uma expressão de alguma linguagem artificial.

No último capítulo descrevemos os módulos de uma interface, implementada a título de exemplificar sua factibilidade no acesso a bases de dados. Mais especificamente, trata-se de uma interface entre o usuário e o sistema operacional DOS, no que diz respeito à manipulação de arquivos em discos de um microcomputador PC.

1.1 - Interfaces com Sistemas de Conhecimento

Os sistemas de conhecimento interagem com os usuários da mesma forma que outros programas, porém eles fazem coisas diferentes e usam métodos diversos. Efetuam diálogos, explicam seus próprios raciocínios e geralmente procuram "compreender" o problema que lhes é apresentado, para depois resolvê-lo. Na solução de problemas, traduzem seus próprios objetivos para a linguagem natural; transformam suas linhas de raciocínio em estruturas que o usuário pode entender e mapeiam a descrição de um problema, fornecido pelo usuário, em formas próprias, entendidas pelo programa.

Estes sistemas oferecem serviços tais como:

- **Distribuição e acesso à tecnologia.**
- **Supervisão na aplicação de tecnologia.**
- **Consistência na aplicação de conhecimentos.**
- **Redução de dados, interpretação através de análise.**

- Raciocínio hipotético.
- Assistência na solução de problemas.
- Otimização no controle de processos complexos.

Por exemplo, necessita-se acessar informações tecnológicas onde há poucos recursos de instrução, onde há escassez de especialistas, onde o serviço profissional está abaixo do padrão. Uma tendência atual é a de se empregar Sistemas Especialistas em organizações comerciais, nas quais ocorra uma rápida mudança nos produtos, rotatividade de pessoal e variação no desempenho da mão-de-obra.

Cada área de aplicação será beneficiada se houver interfaces especializadas. As interfaces precisarão levar em conta a linguagem local; o estilo de comunicação dos especialistas e dos usuários; o nível de compreensão e sofisticação e muitos outros fatores.

Uma interface em Linguagem Natural elimina a necessidade de se aprender os detalhes de uma linguagem de programação. Surgem, no entanto, problemas de ambigüidade. Para superar estes problemas, desenvolvem-se ferramentas para a implementação destas interfaces.

Diferentes Usuários e Diferentes Interfaces

Os sistemas de conhecimento necessitam ser interfaceados com diferentes tipos de usuários: *usuário final, especialista, profissional de manutenção do programa e engenheiro do conhecimento.*

O ***usuário final*** precisa comunicar-se com um programa através de uma mistura de gráficos e textos; esclarecer dúvidas sobre as questões formuladas pelo sistema e pedir explicações sobre resultados não esperados. Ao se familiarizar com um sistema de conhecimento, as frases que usa para se comunicar tornam-se mais concisas, mas mesmo assim cada usuário final tem sua própria maneira de se expressar.

O ***especialista*** também necessita de uma interface apropriada com o sistema. Precisa dedicar cerca da metade de seu tempo para desenvolver e testar uma base de conhecimentos. Ele busca explicação para as conclusões do sistema; quer saber porque o sistema não chegou a determinada solução alternativa; ainda quer ver todas as formas possíveis com as quais o sistema pode raciocinar para chegar a uma conclusão particular.

Precisa editar regras e testá-las experimentalmente. Nesta fase, deve ser ajudado por módulos de correção na escrita e funções de testes parciais. Em resumo, ele necessita de um conjunto de construtores de programas e utilitários para base de conhecimentos, que não são oferecidos por linguagens como LISP e PROLOG.

O ***profissional de manutenção da base de conhecimentos*** normalmente tem um maior conhecimento sobre os componentes de programação do sistema. Necessita de uma interface que permita editar e modificar a biblioteca que contém o registro de casos. Ele pode querer ferramentas que o informem sobre exceções que surgem quando é confirmado o efeito das mudanças na base de conhecimentos, durante os testes por casos.

O ***engenheiro do conhecimento*** necessita acessar a base de conhecimentos de uma forma apropriada para o projeto e tarefas iniciais de aquisição de conhecimentos. Deve ser capaz de criar estruturas para os conceitos e variáveis do domínio, especificando as várias buscas de solução e estratégias de controle, e desenvolver formas para expressar fatos relevantes e heurísticas.

Ele pode querer reduzir o tempo e o esforço requeridos para entrar com os dados, conceitos do problema, regras heurísticas e teste do sistema. Quando o sistema já estiver funcionando, precisa seguir o raciocínio do sistema e localizar erros rapidamente.

1.2 - Processamento de Linguagem Natural (PLN)

Para enviar uma mensagem a um programa de computador a fim de que ele execute algo, precisa-se gerar uma sentença em qualquer linguagem que estiver sendo usada; o programa deve compreender aquela mensagem e traduzi-la em ação.

Visto que a compreensão é um processo de tradução, uma mensagem é compreendida somente com respeito a uma linguagem particular e um determinado conjunto de ações. Uma mensagem não tem nenhum significado se não se referir a ações. Esta é uma razão da dificuldade de se fazer interface em Linguagem Natural. A interface não pode ser genérica, mas específica para cada programa. A maioria dos sistemas necessita interfaces que têm como alvo ações com uma complexidade muito grande.

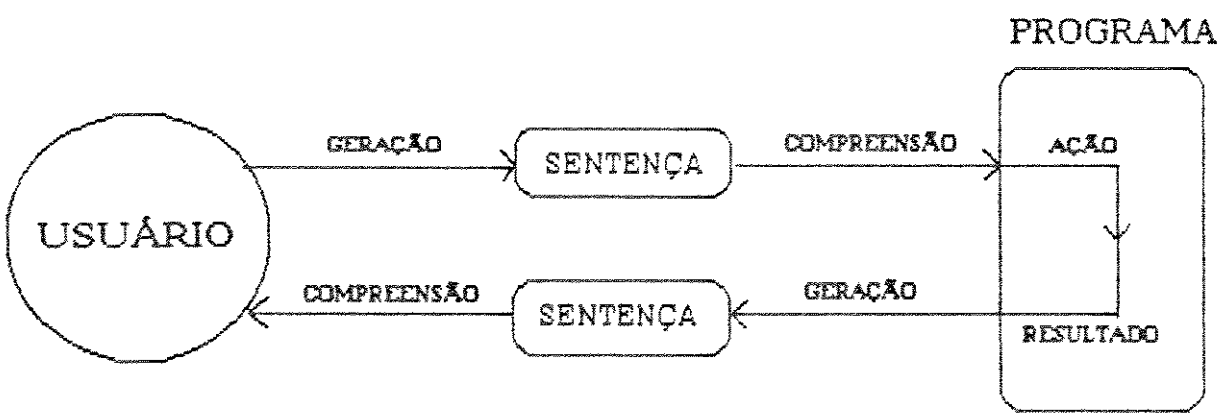


Fig. 1.1 - O processo de compreensão da linguagem no sistema usuário-programa

O envio da mensagem no sentido *usuário –> programa* (entrada) é mais difícil de se interfacear que no sentido oposto: *programa –> usuário* (saída). O motivo de tal dificuldade é que, no sentido de entrada, o controle está com o usuário e não com o programa. Este estudo trata das *interfaces de entrada*

1.2.1 - Linguagem Natural (LN)

A linguagem natural não é a melhor interface para todos os sistemas. É preciso saber quando ela se torna mais apropriada como ferramenta para construir sistemas de conhecimentos.

Ao se falar em LN, referimo-nos geralmente a um subconjunto da LN que será usado em determinada aplicação. Então é preciso também escolher um subconjunto apropriado da LN. Há vários fatores a serem considerados para se decidir no uso de uma interface em LN:

(1) Custo da Interface
Deve-se ter razões concretas para se usar uma interface para um programa particular.

(2) Facilidade de aprendizado
Deve-se permitir ao usuário uma considerável gama de opções para expressar uma determinada mensagem.

(3) Concisão
Ao escolher uma interface para um sistema particular é necessário considerar se o usuário é inexperiente ou não. Normalmente um usuário novato tende a ser menos conciso em suas frases.

(4) Necessidade de Precisão
Para programas onde a precisão é importante, uma interface em LN não é adequada. No entanto pode-se combinar linguagem natural e artificial para se obter as vantagens de ambas.

(5) Necessidade de Gráficos
A melhor maneira para se descrever conceitos, como formas e posições, não é com palavras, mas com figuras. Em sistemas que manipulam objetos gráficos é preciso integrar interface em LN com outras ferramentas de comunicação, baseadas em gráficos.

(6) **Complexidade Semântica**

LN é concisa e eficiente quando o universo de possíveis mensagens é grande. Programas como sistema de apoio "on-line" ou de gerência de banco de dados necessitam de interface em linguagem natural. Quando cresce a capacidade de "raciocínio" dos programas, diminui a distância entre o poder expressivo da LN e as capacidades de tais programas, podendo-se, com mais razão, justificar uma interface em LN.

1.2.2 - Componentes de uma LN em um sistema de Interfaceamento

O processo de se traduzir sentenças de uma linguagem, na qual elas são geradas, para a forma de programas específicos, que causam execução de ações apropriadas, pode ser dividido em três partes:

- *palavras e análise léxica,*
- *gramática e estrutura das sentenças,*
- *semântica e significado das sentenças.*

Palavras e Análise Léxica

A primeira coisa a ser feita para se compreender uma sentença é dividi-la em seus componentes. Em LN, estes componentes são as palavras. Chamamos de *análise léxica* a este tratamento lingüístico.

Na linguagem falada precisa-se de uma informação contextual de um nível mais alto. Na linguagem escrita é útil dividir a declaração em segmentos menores que palavras: *raiz, prefixo, sufixo.* Isto implica em fazer uma análise morfológica, que envolve uma lista de raízes, outra de afixos e mais uma com regras que descrevem como as partes são combinadas (incluindo as exceções).

Estas informações para a análixe léxica são armazenadas em um dicionário. Versões de vários dicionários estão disponíveis, mas a maioria dos programas de compreensão da linguagem não os utiliza por dois motivos:

(1) O vocabulário de um programa particular é normalmente um pequeno subconjunto do dicionário completo.

(2) A informação a ser associada com cada palavra e a maneira que a informação deveria ser representada no dicionário variam de um programa para outro, dependendo de como o restante do sistema de compreensão trabalha.

O dicionário, muitas vezes, não é guardado como uma lista (explicitamente) mas está contido implicitamente nas regras que descrevem as formas de sentenças aceitáveis. O projeto de uma interface em LN requer que um dicionário seja selecionado, em um processo de duas etapas:

(1) Observar o que o programa pode fazer e decidir que conceitos ele pode compreender.

(2) Selecionar palavras específicas para representar aqueles conceitos.

Isto facilita a escolha de uma palavra para cada conceito, mas traz dificuldades ao usuário, que precisa saber quais expressões são aceitáveis. Ao selecionar um dicionário para uma tarefa particular é útil fazer distinção entre categorias léxicas *abertas e fechadas*.

Categorias *abertas* (substantivo, verbo, adjetivo, advérbio) podem ter novas entradas adicionadas a elas, regularmente. Nas categorias *fechadas* (pronome, preposição e conjunção) isto ocorre raramente.

Em interfaces para programas, a distinção entre categorias abertas e fechadas depende da estrutura do programa. Toda vez que um subconjunto muito grande é usado, raramente se evita ambigüidade, que deve ser resolvida através do contexto em que ela se encontra.

Gramática e Estrutura das Sentenças

Uma primeira tentativa para encontrar o significado de uma declaração é associar a ela uma estrutura que provavelmente corresponderá, de alguma forma, à estrutura de seu significado. A associação de tal estrutura a um objeto não-estruturado é chamada de *Análise Sintática*[1]. Através desta análise, pode-se, por exemplo, testar se as possíveis entradas são sintaticamente corretas.

Não é interessante rejeitar uma declaração não-gramatical do usuário, portanto o analisador deve ter flexibilidade suficiente para aceitar uma determinada margem de erros gramaticais nas sentenças de entrada.

Freqüentemente um programa de análise sintática se refere a uma gramática que é representada explicitamente como um conjunto de regras. Há duas maneiras para se conduzir um processo de análise a partir da palavra na frase:

(1) **ascendente**[2]: Constrói-se uma árvore de análise a partir das palavras da frase. Vão se constituindo formas intermediárias, até que se chegue a uma forma completa, sintaticamente.

(2) **descendente**[3]: Parte-se da forma sintática completa (sintagma nominal + sintagma verbal). Nos níveis descendentes vão sendo aplicadas as regras de substituição, até que as palavras da frase de entrada sejam as folhas da árvore resultante.

Uma outra técnica seria codificar a gramática em uma rede de transição ampliada (ATN[4]). Caminha-se simultaneamente através da frase de entrada e da estrutura de rede.

---

[1]Em inglês: parsing.
[2]Em inglês: bottom-up
[3]Em inglês: top-down
[4]Em inglês: Augmented Transition Network (ATN)

A ambigüidade também deve ser considerada durante a análise sintática. Às vezes é resolvida durante um último processamento, ou ainda através de perguntas ao usuário.

Para uma determinada linguagem há muitas gramáticas diferentes. Embora diferentes gramáticas possam aceitar uma mesma linguagem, elas podem associar muitas estruturas diferentes a uma mesma sentença.

A análise sintática é somente uma etapa de um processo global de compreensão, mas da maneira como é feita pode simplificar ou complicar o próximo passo, que é a análise semântica.

Semântica e o Significado das Sentenças

A etapa seguinte do processo de compreensão é associar um significado à sentença, ou seja, determinar que ação no programa corresponde àquela sentença. Denomina-se a isto *processamento semântico*.

Na prática, combina-se dois processamentos que às vezes são vistos separadamente: processamento semântico e processamento pragmático.

1.2.3 - Combinando Análise Léxica, Sintática e Semântica em um Programa de Compreensão

Linguagem através de Janelas

Quando o número de sentenças usadas é pequeno, pode-se usar um sistema baseado em janelas [RICH 84]. O sistema mostra opções disponíveis ao usuário, que vai escolhendo-as e assim constrói uma frase que corresponde a uma ação que o programa pode executar. Neste caso a análise léxica é trivial, visto que a maioria das

palavras não são digitadas mas são apontadas diretamente nas próprias opções fornecidas na janela corrente.

A Análise Sintática vai sendo feita com a entrada das palavras da sentença que o usuário está compondo. Nas sucessivas janelas que são mostradas as opções vão sendo atualizadas para cada palavra nova a ser adicionada na frase, sendo que estas opções são somente aquelas sintaticamente possíveis.

O processamento semântico torna-se também fácil e já vai sendo formado durante a construção da frase, a qual nada mais é que um caminho em uma árvore. Esta facilidade é devida ao fato de que a estrutura de escolhas apresentadas ao usuário vêm de uma forma que corresponde à estrutura necessária à saída final.

Um sistema, de LN baseado em janelas é eficiente porém só é útil onde existe pouca complexidade semântica. Esta aproximação exige que seja possível codificar numa gramática todas as informações que são requeridas, para determinar se uma frase particular pode ser executada corretamente. Normalmente isto não é possível em domínios semanticamente ricos.

Gramáticas Semânticas

Trata-se de uma abordagem na qual os usuários compõem frases inteiras, cabendo ao sistema a tarefa de compreendê-las. A idéia básica é de que uma frase pode ser compreendida em somente duas etapas. Primeiramente a análise léxica separa a frase em palavras. Depois as palavras são analisadas sintaticamente e semanticamente em uma única etapa.

Não deixa de ser uma extensão do sistema de janelas, para permitir um grande número de opções ao usuário (historicamente as gramáticas semânticas vieram primeiro e serviram de base para se construir o sistema baseado em janelas). Quando uma frase é analisada pela gramática, ela é associada a uma estrutura que é determinada por regras.

As regras são construídas de forma que a estrutura produzida pelo analisador corresponda, tão restritamente quanto possível, às entradas do programa. Quando

uma gramática semântica é usada, como qualquer outra gramática, necessita-se de técnicas de análise sintática para que a gramática seja aplicada a cada frase de entrada. Gramáticas Semânticas são usadas somente quando um subconjunto relativamente pequeno da LN deve ser reconhecido.

Gramáticas Sintáticas

Quando um grande subconjunto da LN é usado como interface é preciso capturar tanto quanto for possível a regularidade da linguagem, nas regras usadas, para compreendê-la.

Ao se construir uma gramática semântica, olhamos tanto para a linguagem de entrada como para as ações do programa, e mapeamos as estruturas de um em outro. Mas quando a complexidade em ambos os lados é grande, torna-se difícil considerar os dois lados de uma só vez e completar a tradução em uma única etapa.

Neste caso usamos regras gramaticais que podem associar uma estrutura para cada sentença. Escolhemos estruturas que procuram generalidades na própria linguagem de entrada, independente do programa. Depois escrevemos outras regras (que podem ou não ser ligadas diretamente às regras gramaticais) para transformar a estrutura analisada em ações que o programa entende.

Comentários

Uma técnica freqüentemente usada em interfaces de LN é a *ATN*. Os sistemas *LUNAR* [WOODS 70] e *ROBOT* [HARRIS 77] são sistemas que operam desta maneira.

Infelizmente não é sempre possível saber, durante o processamento de uma frase, qual a classificação exata de um determinado constituinte. Precisa-se antes ter processado a maioria dos constituintes daquela frase. Para minimizar o número de constituintes rejeitados, são realizados testes, como o de concordância em número, ou ainda testes motivados semanticamente, como é o caso de um advérbio de companhia, onde o co-participante necessariamente é um ser animado.

O uso de regras semânticas não é trivial. Há duas formas de se aplicar regras semânticas.

Uma das técnicas é intercalar o processamento sintático e semântico e ligar a cada regra gramatical um conjunto de ações que podem ser executadas quando a regra for aplicada. Assim a ação executada torna-se uma tradução completa da frase de entrada.

Outra maneira é fazer primeiro um tipo de análise e depois o outro (lembrando-se que análise usualmente envolve busca –> retrocesso[5]. Esta separação torna possível construir uma gramática e um analisador para uma LN, e depois reutilizá-lo em muitas interfaces. Este é o caso do sistema *TEAM*[GROSZ 87].

1.3 - Requisitos e Limites de um sistema de PLN

Para resolver o problema de interfaceamento entre usuários e computador, não basta juntar algumas técnicas de computação, mas é preciso criar uma nova tecnologia.

Descrevemos, a seguir, alguns tópicos relativos aos requisitos necessários, em sistemas de PLN, que têm como uma de suas características principais o "*gerenciamento das expectativas do usuário*". Requisitos desta natureza foram levados em conta na implementação de sistemas como o *"LE COURTIER"* [GERSHMAN 85].

Aparecem dois tipos de problemas principais ao se projetar uma interface. O primeiro relaciona-se com a variedade de possíveis usuários e suas diferentes atitudes. O outro tipo tem a ver com as limitações do sistema de conhecimento e seu desempenho.

[5]Em inglês: backtracking

1.3.1 - Requisitos Relativos ao Usuário

Facilidades

- Processamento sintático completo.
- Manipulação de elipses e construções não-gramaticais.
- Correção de erros ortográficos.
- Entendimento da mensagem do usuário, dentro do contexto em que ela foi pensada.
- Entendimento dos objetivos do usuário.

Modelo do Domínio

- É preciso que o sistema e o usuário compartilhem o mesmo domínio do discurso.

Uso de Informação Incompleta

- O sistema deve oferecer gradualmente sua ajuda ao usuário, com o pouco ou muito de informação que for disponível em dado momento.

Uso de Informação Completa

- O sistema deve utilizar toda a sua capacidade de conhecimento (inclusive as informações obtidas do usuário) e de inferência, evitando perguntas redundantes.

Escolha de Tópicos pelo Usuário

- Reconhecimento de que o usuário está selecionando o tema.
- Decisão de aceitar ou não informações do usuário.
- Decisão de manter (salvar) ou não o tema anterior e o contexto da conversação.
- Identificação do novo contexto e determinação de como retornar ao anterior, caso ele tenha sido "salvo".

Seleção de Informação

- Não aproveitamento de informações desnecessárias.
- A pergunta ao usuário é refeita caso não seja obtida uma resposta, a menos que a resposta mude o fluxo de controle do programa.

Explicações

- Serviço de orientação do usuário caso ele esteja "perdido" no programa.
- Explicação dos termos que está usando.
- Explicação dos resultados obtidos.

Tratamento com Hipóteses

- Na ausência de uma solução ótima, o sistema apresenta possíveis soluções para que sejam avaliadas, escolhidas ou então corrigidas através de refinamentos iterativos.
- Ao fornecer uma solução ótima, deve permitir ao usuário que mude alguma das suposições anteriores para depois gerar uma nova solução, se for o caso.

Gerenciamento do Diálogo

- Resolução de referências pronominais,
- Elipses,
- Referências a dados mostrados,
- Descrições incompletas, mas contextualmente corretas.

Manutenção de Dados, Regras e Vocabulário

- Manutenção de consistência, após cada mudança ocorrida no sistema.
- A manutenção de regras envolve:
  - Inspeção e especificação de regras,
  - Deteção de inconsistências.
  - Possibilidade de inclusão de novas regras.

1.3.2 - Problemas com o Limite do Sistema

O sistema precisa estar ciente daquilo que não conhece para que possa direcionar o usuário para dentro da área de sua competência.

Limites de Abrangência conceitual

- Reconhecimento de que certos conceitos ainda não são conhecidos pelo sistema e a devida recondução do usuário para a área de competência.

Limites no Vocabulário do Sistema

- O tamanho do Vocabulário não diz muita coisa. Depende do limite de cobertura conceitual.
- O sistema deve comunicar ao usuário a extensão de seu vocabulário.

Limites do Conhecimento Real do Sistema

- *Exemplo:* Em um sistema usado na área comercial, deseja-se a média de preços nos últimos cinco dias. Porém o sistema só tem informação de dados dos últimos três dias. Então o sistema deve dizer que não sabe da média dos cinco dias, mas dá uma resposta do que sabe sobre a média dos últimos três dias.

Limites de Desempenho

- O sistema deve informar ao usuário o custo de sua pergunta, ou seja, o esforço computacional que seria gasto para que lhe seja fornecida uma resposta. E além disso, sugerir alternativas mais apropriadas, caso existam.

1.4 - Sistemas de Processamento de Linguagem Natural (PLN)

A finalidade de se processar computacionalmente a linguagem natural é facilitar a comunicação entre o usuário e o computador, permitindo que o usuário se expresse em linguagem natural. Várias são as aplicações dos sistemas de Processamento de Linguagem Natural, tais como:

- Sistemas de Perguntas e Respostas.
- Interfaces em Linguagem Natural: USUÁRIO <--> PROGRAMAS.
- Instrutores nos Processos Educativos.
- Reconhecedores de Texto e Discurso.
- Geradores de Texto.
- Ambientes para Desenvolvimento de Sistemas.

Baseando-se no modelo de representação do conhecimento e na utilização dos conhecimentos do domínio de aplicação, pode-se classificar os sistemas de PLN em três categorias:

Tipo A: *Não incorporam nenhum modelo do domínio.* Situam-se aqui os sistemas que utilizam estruturas de dados "ad hoc" para armazenar fatos sobre um domínio de aplicação limitado e usam palavras-chaves para analisar as sentenças de entrada. Estes sistemas muitas vezes levam a resultados indesejáveis por serem modelos muito simples da linguagem.

Também podem ser classificados aqui o que podemos chamar de **Sistemas de Lógica Limitada,** onde as informações na Base de Conhecimento são armazenadas em alguma notação formal. Técnicas de tratamento lingüístico são utilizadas para transformar as sentenças de entrada nessa forma interna (análise sintática e semântica). Estes sistemas são limitados porque cobrem somente parte das inferências lógicas usadas na conversação.

Tipo B: *Usam modelos explícitos do mundo.* Uma grande quantidade de informações sobre o domínio de aplicação é utilizada como auxílio no entendimento das sentenças. O conhecimento é representado através de modelos como os da *Lógica, Redes Semânticas* ou *"Frames".*

Tipo C: Incluem em seus modelos informações sobre objetivos e crenças de entidades inteligentes. Estes sistemas ainda estão em fase de pesquisa.

1.4.1 - Módulos Constituintes de um sistema de PLN

Um sistema de PLN processa as entradas em LN e gera saídas apropriadas, fundamentadas nos conhecimentos que possui. A Base de Conhecimentos para estes sistemas é construída anteriormente, no início da execução dos mesmos.

A arquitetura básica de tais sistemas (fig. 1.2) constitui-se geralmente dos módulos: **Analisador Gramatical, Dicionário, Interpretador, Base de Conhecimentos** e **Gerador de Respostas**, além do módulo de **Entrada** que recebe a sentença do usuário e a recondiciona num formato adequado para que seja processada pelos demais módulos.

O *Analisador Gramatical* recebe a sentença do módulo de *Entrada*, reconhece se está correta, e mapeia-a em uma estrutura interna que serve de entrada para o módulo *Interpretador*. A seguir o módulo *Interpretador* acessa a *Base de Conhecimentos* para obter conhecimentos, realizar inferências ou para adicionar novos conhecimentos à Base. A estrutura gerada pelo Interpretador alimenta o *Gerador de Respostas* que a transforma em sentença de saída. Temos ainda o *Dicionário*, que contém as palavras que o sistema reconhece e as informações sintáticas e semânticas.

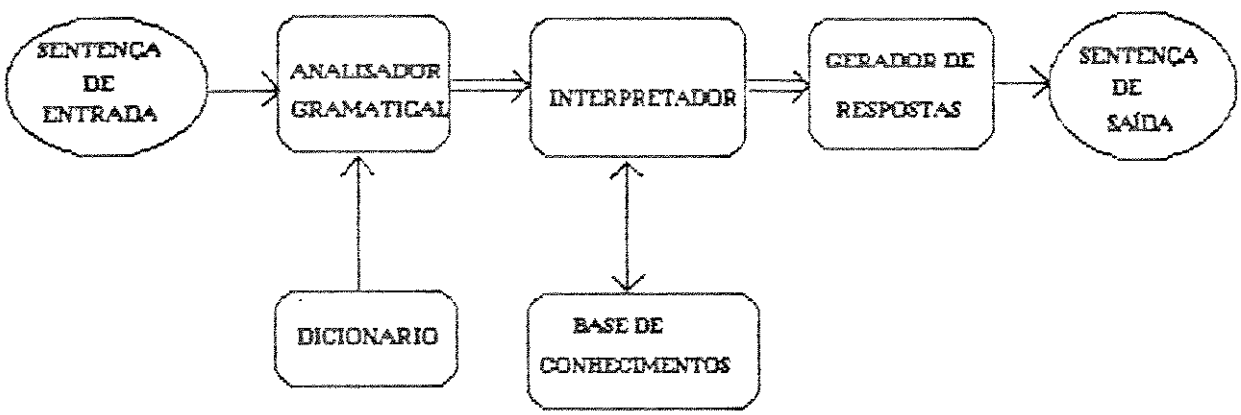


Fig. 1.2 - Arquitetura básica de um Sistema de PLN

Analisador Gramatical

Aceita sentenças como uma lista de palavras, transformando-a em itens léxicos. As palavras são morfologicamete verificadas no Dicionário, recuperando-se as informações sintáticas e semânticas a serem adicionadas a cada palavra. As estruturas sintáticas de entrada são checadas em um formalismo gramatical e, quando satisfeitas, são convertidas em uma representação interna.

Dicionário

Contém as palavras que o sistema compreende e as informações sintáticas e semânticas necessárias para que cada palavra possa ser processada pelo sistema. Auxilia o analisador gramatical na transformação da sentença de entrada em uma estrutura interna temporária.

Um dicionário pode ser otimizado através de considerações morfológicas das palavras, armazenando os ítens léxicos, através de seus constituintes básicos e acrescentando-se as informações sintáticas e semânticas que permitam identificar a estrutura sintática da sentença e facilitem a criação da estrutura interna do seu significado.

A construção de um dicionário otimizado permite um melhor desempenho em termos de tempo de execução e resposta do sistema, devido ao menor espaço de memória e métodos de busca mais velozes.

Interpretador

É responsável pelo acesso à Base de Conhecimento, incluindo as adições e modificações necessárias. Sua função depende do tipo de sentença de entrada. Se a sentença for declarativa, a estrutura interna será comparada ou adicionada à Base de Conhecimentos. Se a sentença for uma pergunta, a função do Interpretador será de recuperar alguma informação armazenada na Base de Conhecimentos. O interpretador também realiza inferências a partir do conhecimento armazenado.

Adicionar conhecimento é um processo complexo devido ao problema de consistência do novo conhecimento com relação ao que já está armazenado. Se a nova informação é conflitante, pode forçar a modificação da estrutura de conhecimento anterior como também eliminar da Base algum conhecimento, ou mesmo refutar o novo conhecimento.

Gerador de Resposta

Produz as sentenças de saída, como respostas às sentenças de entrada fornecidas pelo usuário. Para uma sentença declarativa, a saída deverá ser uma simples confirmação, indicando se o seu comando foi processado. Para uma sentença interrogativa, o Gerador de Respostas constrói uma sentença de saída a partir da estrutura interna que lhe foi fornecida pelo Interpretador.

Base de Conhecimentos

A Base de Conhecimentos é o conjunto de processos e estrutura de dados associada que fornece ao programa determinados tipos de informações e raciocínios. Estas informações dizem respeito ao conhecimento do domínio específico.

1.5 - Evolução dos Sistemas de PLN

Década de 40: os computadores passaram a manipular símbolos (processamento simbólico). Começaram os estudos da linguagem. Inicialmente procurava-se construir máquinas tradutoras. Porém a ineficiência destes sistemas mostrou logo a necessidade de um entendimento do significado das sentenças. Não bastava traduzir palavras de uma língua e reorganizá-las em sentenças de outra língua. (O interesse e o estudo pelas "máquinas tradutoras" foi retomado na década de 80.

Década de 50: os trabalhos de Chomsky abordaram a estrutura sintática das sentenças e modelos de análise gramatical [CHOMSKY 57] .

Década de 60: os pesquisadores começaram a ver a linguagem humana como um processo cognitivo complexo, que envolve conhecimentos de vários tipos: a estrutura das sentenças, o significado das palavras, as regras de conversação e um conjunto amplo de conhecimentos gerais sobre o mundo em que vivemos.

Os primeiros sistemas práticos de PLN eram muito limitados, e aplicados em um domínio muito restrito. Estes sistemas ignoravam as complexidades da linguagem, e não podem ser considerados como sistemas de processamento de linguagem natural. Como exemplos temos os sistemas BASEBALL (*Green, 1963*), ELIZA (*Weizenbaum, 1966*) e STUDENT (*Bobrow, 1968*).

A seguir foram desenvolvidos analisadores gramaticais mais eficientes, com regras mais precisas. Surgiram as teorias de processameto de informações semânticas e novos modelos de representação do conhecimento. Temos como exemplos a Gramática de Casos, desenvolvida por *Fillmore* [FILLMORE 68], que foi usada por *Carbonell* (1970) em seu sistema SCHOLAR; as Redes de Transição Ampliadas (ATNs), desenvolvidas por *Woods* [WOODS 70], usadas em seu próprio sistema LUNAR (1972); e ainda a Teoria da Dependência Conceitual, desenvolvida por *Schank* (1973), que serviu de base para a construção dos sistemas MARGIE (1975) e SAM (1975).

Na segunda metade da *década de 70* começaram a surgir sistemas de PLN que criavam um ambiente para o desenvolvimento de outros sistemas em linguagem natural. *Davis* (1976) criou o sistema TEIRESIAS que é uma ferramenta para a criação e atualização de bases de conhecimento para os sistemas especialistas. Em 1977, *Hendrix* apresenta seu sistema LIFFER, para desenvolvimento de sistemas de PLN (usando as ATNs).

No final da década de 70, através do uso da Lógica para representação do Conhecimento e da linguagem PROLOG [CLOCKSIN 81] na implementação de sistemas, abre-se uma nova linha de pesquisa. Como exemplo temos o sistema *TUGA* [COELHO 79] e o *CHAT-80* [PEREIRA 80].

O sistema *TUGA*, utilizado em consultas bibliográficas, permite que os usuários classifiquem novos documentos. Em certas situações, o programa toma a iniciativa do diálogo, fazendo perguntas ao usuário. Isto torna-se possível devido ao fato de que o sistema *TUGA* possui dois níveis de gramática: um para análise de sentenças individuais do Português e outro para os diálogos.

O sistema *CHAT-80* foi desenvolvido com o objetivo de mostrar que formalismos baseados em Lógica são poderosos para se construir interfaces em linguagem natural, para acesso a bases de conhecimentos. As perguntas dos usuários são analisadas por uma Gramática de Extraposição [PEREIRA 83], e transformadas em uma estruturta lógica final, executável pelo PROLOG.

A partir de meados da *década de 80* surgem sistemas de PLN mais sofisticados e flexíveis como o TEAM [GROSZ 86], que permite atualizações de seus módulos pelo próprio usuário ou pelo especialista em base de dados. O sistema LE COURTIER [GERSHMAN 85] é usado comercialmente em ambiente bancário e tem como uma de suas principais características orientar o usuário, tendo em vista as expectativas dele, no momento em que interage com o sistema.

1.6 - Tendências dos Sistemas de PLN

Nos sistemas de PLN implementados, observa-se que é dada uma importância maior à análise gramatical. As informações sintáticas obtidas nas sentenças de entrada, possibilitam a execução de certas tarefas como também respondem a perguntas do usuário. Já a semântica nem sempre é bem definida ou descrita formalmente, aparecendo, muitas vezes, embutida na própria gramática.

Torna-se necessária a determinação de uma análise semântica mais profunda, que extraia das sentenças seu significado completo e outras informações que normalmente vêm agregadas. Poderiam ser, assim, eliminados muitos problemas de ambigüidade, elipse e tantos outros que são comuns nas linguagens naturais.

Pesquisas mais recentes incluem mais uma fase de análise: a *Pragmática*. Na análise pragmática é tomado o valor prático das palavras e seu uso como critério de verdade. Os sistemas aceitam ou recusam uma interpretação baseando-se em conhecimentos que têm sobre objetos e ações.

Tratando-se da integração de Sintaxe, Semântica e Pragmática, surgem duas questões a serem discutidas: Os conhecimentos sintáticos e conceituais (semânticos e pragmáticos) são aplicados por processos de controle distintos ou

existe uma estrutura de controle integrada que aplica ambos? Uma mesma Base de Conhecimentos pode englobar conhecimentos sintáticos e conceituais ou existe uma separação nítida entre esses dois conjuntos?

Podem ser identificadas algumas situações possíveis:

Semântica e Sintaxe são completamente disjuntas: A análise sintática seria um processo independente e anterior ao processamento semântico. Torna-se ineficiente na captura do significado das sentenças de entrada, mesmo com domínios limitados.

Semântica e Sintaxe são fracamente interligadas: Nesta situação existe ainda um processo de análise sintática anterior cuja estrutura resultante alimenta o processamento semântico. Contudo o processo sintático pode interrogar um componente semântico ao executar uma decisão sintática. O controle deste processo fica por conta da Sintaxe.

Semântica e Sintaxe interagem reciprocamente: Os processos são ainda disjuntos, mas cooperativos. O controle é de ambos os processamentos.

Semântica e Sintaxe são aplicadas por uma estrutura de controle integrada: A decisão de se usar conhecimentos sintáticos ou conceituais é tomada por uma única estrutura de controle e o conhecimento disponível mais útil será aplicado na tentativa de analisar e entender a sentença de entrada.

Estas questões são ainda polêmicas, mas as representações declarativas, principalmente as da lógica, têm permitido representar conhecimentos sintáticos e conceituais em estruturas de dados semelhantes, e não têm forçado a necessidade de separações nítidas na base de conhecimentos.

Várias técnicas para construção de interfaces em LN têm sido desenvolvidas, mas a LN apresenta problemas de interpretação como as ambigüidades, que normalmente não aparecem em linguagens artificiais. É por isso que a compreensão da LN é um problema ligado à Inteligência Artificial. Uma boa parte do conhecimento a ser explorado não é sobre a própria linguagem, mas sobre o domínio em questão.

2

PROCESSAMENTO SINTÁTICO

No estudo de linguagens naturais para o processamento computacional, não se pode simplesmente transportar os métodos empregados na análise de uma língua para a análise de outra, visto que as construções lingüísticas são diferentes. Deve-se verificar a adequação dos métodos e realizar as transformações necessárias.

Existem diferenças entre linguagens naturais e formais. Numa gramática de linguagem natural as palavras são associadas aos seus significados e formas de pronúncia, enquanto que na formal os seus constituintes são um alfabeto e regras de formação das expressões.

Aparentemente seria desaconselhável usar considerações de gramáticas formais para o processamento de linguagem natural, porém essa não é a opinião de muitos lógicos, como *Montague* que afirma não existir diferença teórica entre linguagens naturais e artificiais e considera possível desenvolver uma teoria matemática precisa para a compreensão da sintaxe e semântica de ambas as espécies de linguagens [MONTAGUE 74].

Para se fazer a análise de uma sentença e atribuir-lhe uma estrutura sintática, pressupõe-se a existência de um *dicionário* e de uma *gramática*. Neste capítulo apresentamos vários formalismos gramaticais, procurando mostrar como se chegou às gramáticas livres de contexto e depois às gramáticas de cláusulas definidas, que consideramos mais apropriadas e são usadas na interface em linguagem natural que implementamos.

Gramáticas

Gramáticas são listas de procedimentos que permitem representações finitas das linguagens e a especificação das suas expressões bem formadas. Pode-se

classificar as gramáticas concebidas para o processamento de linguagem natural em três grupos:

1. **Gramáticas formais:** - Gerativas de Chomsky
   - Transformacionais
   - de Casos

2. **Gramáticas lógicas:** - de Metamorfose
   - de Cláusulas Definidas
   - de Extraposição

3. **Gramáticas procedurais:** - ATNs (redes de transição ampliadas)

2.1 - Gramáticas Gerativas de Chomsky

No final da década de 50, *Chomsky* introduziu a teoria de linguagens formais [CHOMSKY 57]. Desenvolvida como um estudo matemático, sua teoria tem influenciado fortemente no projeto de linguagem de programação e ainda se mostra como uma ferramenta teórica e prática para o desenvolvimento de sistemas de processamento de linguagem natural.

Chomsky afirma que uma linguagem natural é potencialmente infinita e não importa quantas sentenças uma pessoa possa falar ou ouvir, sempre outras podem ser produzidas ou compreendidas. Aprende-se uma língua, não por ouvir todas as sentenças da linguagem e memorizá-las, mas porque se possui um conhecimento inato da estrutura da linguagem, com que todo ser humano nasce. Para explicitar essas estruturas da linguagem, Chomsky propõe as *gramáticas gerativas*.

Para uma melhor compreensão dessas gramáticas, serão apresentados, de forma resumida, os *sistemas de reescrita*, que englobam tanto as gramáticas gerativas quanto as analíticas.

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 reescrita ou produções.

As regras de reescrita estabelecem que um certo símbolo pode ser expandido em uma estrutura de árvore, através de uma seqüência de outros símbolos.

Seja:

V um alfabeto ,

$x, y, z, w, x', y', z', w'$, cadeias de símbolos de V,

as produções são pares ordenados denotadas por $w ==> w'$.

Diz-se que

x gera diretamente y, simbolicamente , $x ==> y$,

se há palavras z, z', w, w' tal que x é zwz' e y é $zw'z'$

e $w ==> w'$ é uma regra de reescrita.

Uma *Gramática Gerativa de Chomsky* pode ser vista como um sistema de reescrita onde o vocabulário é dividido em dois alfabetos disjuntos. Um deles contém as categorias gramaticais e seus elementos serão referidos como *não-terminais* ou variáveis e o outro constituído dos símbolos do alfabeto da linguagem a ser gerada, chamados *terminais*. Esses símbolos serão concatenados para formar as palavras.

Entre os símbolos não terminais, um é distinto e corresponde ao axioma dos sistemas de reescrita. As regras são produções, em número finito, cujas restrições determinarão o tipo da gramática, conforme a Tabela 2.1 [MIRANDA 86].

TIPO	CLASSE DE GRAMÁTICAS	RESTRIÇÕES NAS PRODUÇÕES
0	Recursivamente Enumerável	Nenhuma
1	Sensível ao Contexto	Produções não-decrescentes
2	Livre de Contexto	O lado esquerdo deve ser um símbolo não-terminal
3	Regular	O lado esquerdo deve ser um único não-terminal e o lado direito deve ser um único terminal ou um único terminal seguido por um único símbolo não-terminal

TABELA 2.1 - Classificação das Gramáticas Gerativas segundo Chomsky

Constituintes das gramáticas de Chomsky

Usando-se uma notação que vem da representação de Backus, podemos descrever as gramáticas gerativas através de seus principais constituintes:

Símbolos não-terminais ou variáveis

Ex.: <SENTENÇA>
<SINTAGMA VERBAL>

Símbolos Terminais da Linguagem

Palavras da linguagem que, concatenadas, formam cadeias.
Ex: *programa, escreveu, Sérgio.*

Regras de Reescrita ou Produções

Especificam as relações entre certas cadeias de símbolos terminais e não-terminais.

Ex: <SENTENÇA> -> <SINTAGMA NOMINAL> <SINTAGMA VERBAL>
<SUBSTANTIVO> -> programa

Símbolo Inicial

É um símbolo não-terminal chamado <SENTENÇA>

A vantagem em se definir formalmente linguagens, do ponto de vista da Lingüística Computacional é que, se a estrutura das sentenças a serem processadas pelo sistema for bem determinada, então o algoritmo para análise gramatical das sentenças será fácil de ser escrito.

As Gramáticas Livres de Contexto têm sido mais utilizadas em sistemas de PLN. Suas produções têm apenas um símbolo não-terminal em seu lado esquerdo.

Por exemplo, seja a gramática:

<SENTENÇA> -> <SINTAGMA NOMINAL> <SINTAGMA VERBAL>
<SINTAGMA NOMINAL> -> <ARTIGO> <SUBSTANTIVO>
<SINTAGMA NOMINAL> -> <SUBSTANTIVO PRÓPRIO>
< SINTAGMA VERBAL> -> <VERBO> <SINTAGMA NOMINAL>
<ARTIGO> -> um
<SUBSTANTIVO> programa
<SUBSTANTIVO PRÓPRIO> -> Sérgio
<VERBO> -> escreveu

Todas as derivações estruturais das sentenças podem ser representadas por "árvores", facilitando a construção de algoritmos para seu tratamento. As regras de produção permitem definições recursivas.

Usando-se a gramática acima para analisar ou gerar a sentença *"Sérgio escreveu um programa"*, teríamos a seguinte árvore de derivação (fig. 2.1):

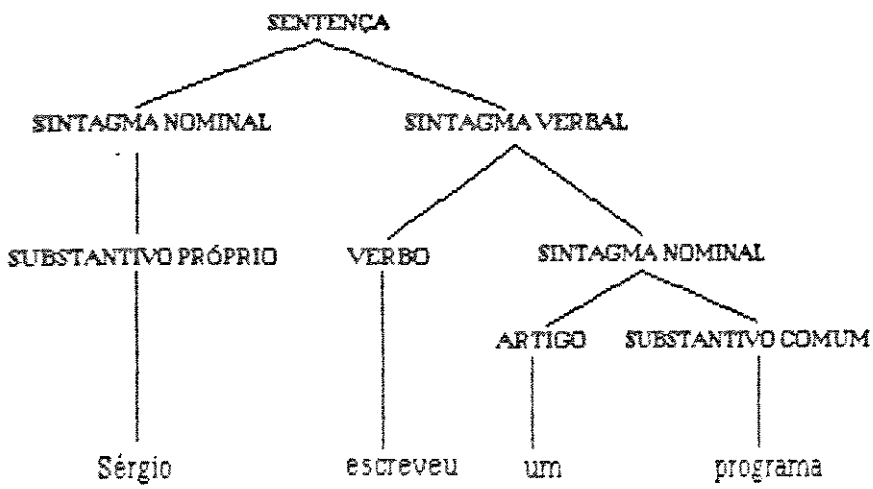


Figura 2.1 - Árvore de derivação da sentença *"Sérgio escreveu um programa"*

Qual seria a gramática mais apropriada para o processamento de linguagem natural?

As gramáticas livres de contexto (**tipo2**) são as mais adequadas para o tratamento computacional da linguagem natural, devido à facilidade de suas implementações. Estas gramáticas podem ser ampliadas, ou seja, passam a ter uma relativa sensibilidade ao contexto. A tarefa de encontrar propriedades formais que sejam universais (analogamente à tarefa de se encontrar fórmulas logicamente válidas) deve levar ainda um bom tempo.

As gramáticas de **tipo 1** seriam boas, porém a implementação destas gramáticas requer algoritmos muito complexos, e as de **tipo 0** (mais gerais) apresentam propriedades genéricas que, em um número significativo de casos, não são relevantes para a linguagem natural.

2.2 - Gramáticas Transformacionais

Apesar das gramáticas de *estrutura da 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, ou mesmo para a simples concordância do sujeito com o verbo.

Chomsky propõe uma outra representação, ou seja uma árvore de derivação que somente descreva as regras que foram aplicadas para a geração de uma sentença (estrutura superficial). Dois outros componentes, um *transformacional* e outro *morfofonético* agiriam sobre esta estrutura para reconhecer ou gerar sentenças da linguagem natural.

As gramáticas transformacionais apresentam uma organização tripartida:

1ª parte: Gramática de Estrutura da Frase - gera cadeias de morfemas representando sentenças simples.

2ª parte: Seqüência de regras de transformação - reorganiza as cadeias e adiciona ou retira morfemas para formar a representação.

3ª parte: Seqüência de regras morfofonéticas - mapeia cada representação da sentença em uma cadeia de fonemas, chamada de "estrutura profunda".

Exemplo:

Para transformar a sentença *"Marcelo projetou um circuito"*, em uma sentença correspondente, na voz passiva, devem ser aplicadas regras transformacionais que liguem um verbo auxiliar ao verbo principal e uma preposição ao sujeito:

Sentença na voz ativa (fig. 2.2):

Marcelo	projetou	um circuito	SN - sintagma nominal
SN_1	V	SN_2	V - verbo

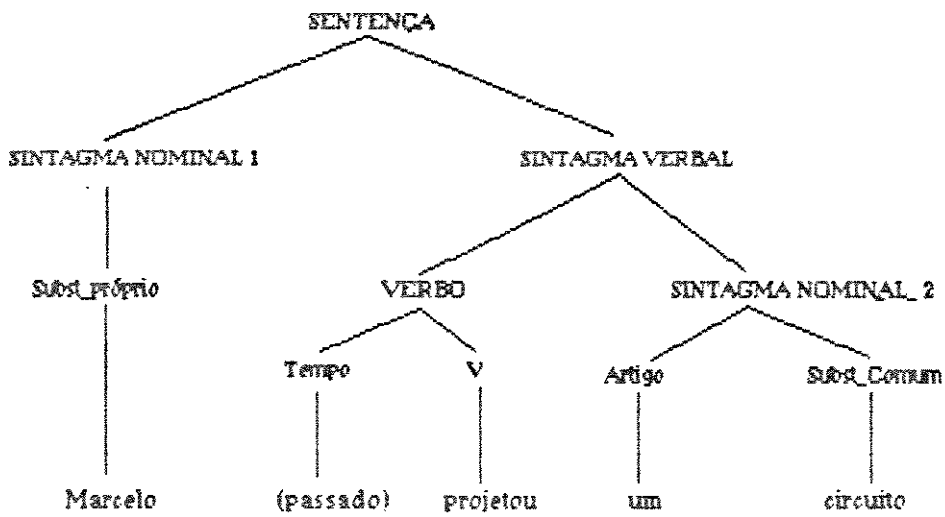


Fig. 2.2 - Árvore de derivação da sentença *"Marcelo projetou um circuito"*

Sentença na voz passiva (fig 2.3):

Um circuito	foi	projetado	por	Marcelo
SN_2	- Aux	+ V	- por	+ SN_1

Para concluir a transformação, são aplicadas as regras morfofonéticas para gerar o particípio passado do verbo *"projetar"*.

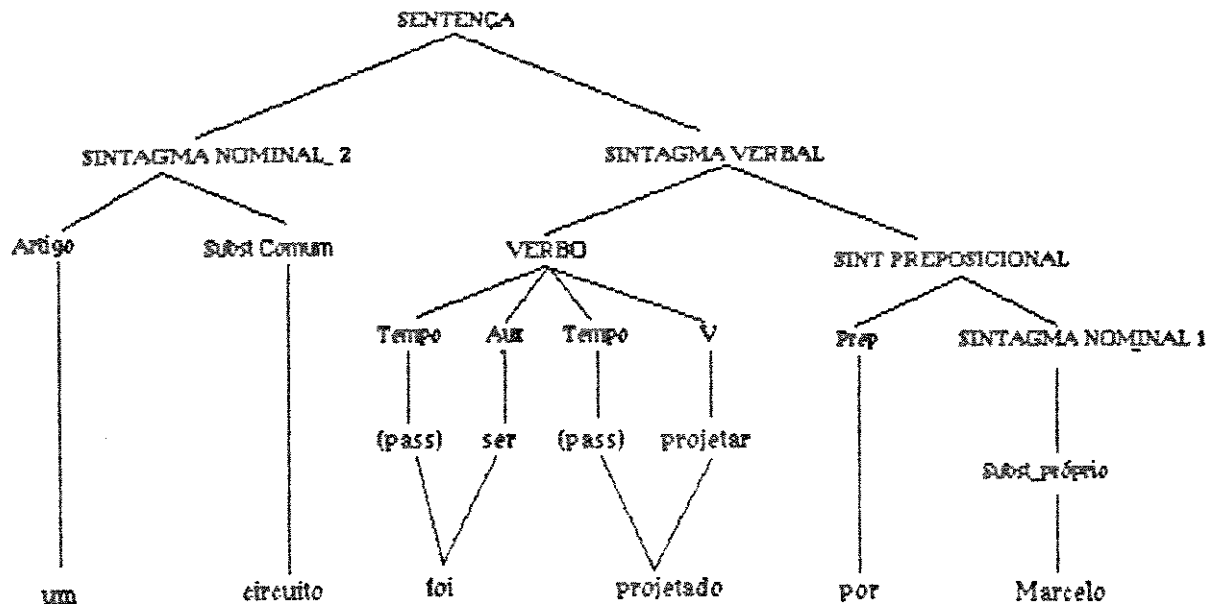


Fig. 2.3 - Árvore de derivação da sentença *"Um circuito foi projetado por Marcelo"*

As teorias de *Chomsky* foram amplamente aceitas e deram origem a um grande número de pesquisas lingüísticas. Como conseqüência, essas pesquisas revelaram alguns problemas com essas 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:

O teste foi resolvido por um aluno aplicado.
O teste foi resolvido por um método aplicado.

Também foi questionada a noção de transformações. Pode ocorrer que uma sentença e sua transformação não tenham necessariamente o mesmo significado, como, por exemplo, uma *sentença declarativa* e sua *negação*.

Chomsky incopora os resultados de outros pesquisadores em um modelo mais completo dessas gramáticas [CHOMSKY 65], compondo uma gramática de três partes:

1. Componente *sintático*.
2. Componente *semântico*.
3. Componente *fonológico*.

O componente *sintático* contém regras de estrutura da frase e um dicionário com regras de inserção léxica, produzindo uma estrutura de entrada para as outras duas partes.

O componente *semântico* interpreta a estrutura profunda gerada pelo componente sintático, fornecendo como saída a representação do significado da sentença.

O componente *fonológico* interpreta a estrutura superficial gerada pelo componente sintático, resultando como saída a seqüência de sons da sentença.

Acrescentou-se um *dicionário*, onde informações sintáticas, semânticas e fonológicas podem ser introduzidas junto às palavras já existentes, podendo-se também capturar dependências conceituais. Apesar das novas introduções, essas regras muitas vezes mudam o significado de algumas sentenças, deixando ainda vulnerável esta teoria.

Em 1970, *Chomsky* propôs uma extensão de sua teoria. As regras de transformação foram cuidadosamente estudadas e a estrutura superficial foi transformada em uma *estrutura lógica* na qual as regras do componente semântico são aplicadas.

As gramáticas transformacionais foram um marco importantíssimo, não só em lingüística mas também em suas aplicações na Ciência da Computação.

2.3 - Gramáticas de Metamorfose

Consistem na formalização lógica das gramáticas representáveis nos *sistemas-q* [COLMERAUER 77], que tinham sido escritos para análise e síntese

automática de partes da linguagem natural (estão relacionadas com a origem da linguagem de programação *PROLOG*).

Sistemas-q: Linguagem de programação, não-determinística, que permite a especificação de *gramáticas complexas*, que podem ser associadas a interpretadores.

Gramáticas complexas: Baseiam-se em regras de reescrita para *símbolos complexos*, que possuem estrutura interna, ao contrário das regras de reescrita das gramáticas gerativas.

Símbolos complexos: São árvores e listas de árvores (funtores arbitrários cujos argumentos podem ser terminais ou não-terminais). Uma árvore é dita parametrizada quando possui um argumento não-terminal, ou seja, um nó rotulado por uma variável. Desta forma pode-se transmitir a cada símbolo complexo uma variedade de informações adjacentes através de variáveis.

Devido à flexibilidade apresentada pelos *sistemas-q* sua implementação conduziu a grandes complexidades computacionais, pois não possuiam um ambiente de programação adequado. Este fato levou *Colmerauer* a desenvolver o *PROLOG*.

Em 1978 *Colmerauer* propôs as *gramáticas de metamorfose*, que consistem na explicitação em uma teoria de primeira ordem das idéias contidas nos *sistemas-q*, com vistas a sua realização no ambiente de programação *PROLOG*. *Estas* gramáticas são formalismos muito complexos, de difícil implementação, que produziram idéias relevantes para o desenvolvimento de outras gramáticas lógicas menos gerais e mais simples.

2.4 - Gramáticas de Cláusulas Definidas (GCD)

Foram descritas a partir da idéia de se generalizar o formalismo das gramáticas livres de contexto, expressando-o em cláusulas da Lógica de Predicados de

Primeira Ordem (LPO), e de se aplicar os fundamentos das gramáticas de metamorfose [PEREIRA 80].

As gramáticas de cláusulas definidas (GCD) são uma extensão natural das *gramáticas livres de contexto.* As GCD tornaram-se um formalismo claro e poderoso para descrever tanto linguagens naturais quanto artificiais e podem ser diretamente executadas como um programa *PROLOG*; comportando-se como um eficiente analisador *descendente* da linguagem que descreve.

Facilidades oferecidas pelas GCD:

a) Introduzem dependência conceitual na gramática, de tal modo que a forma permitida para um sintagma possa depender do contexto (dentro da sentença) no qual o sintagma ocorre na sentença.

b) Constroem estruturas de árvore durante o processo de análise da sentença, com o auxílio de *variáveis lógicas.*

c) Adicionam condições extras nas regras gramaticais, caso sejam necessárias computações auxiliares para a análise da sentença.

As Gramáticas de Cláusulas Definidas são casos particulares das gramáticas de metamorfose. O lado esquerdo de uma regra é sempre constituído de um único símbolo não-terminal (nas gramáticas de metamorfose, o lado esquerdo pode ser uma seqüência de símbolos não-terminais, sujeitos a certas restrições). Também podem ser vistas como *gramáticas livres de contexto,* com termos *lógicos* para os símbolos gramaticais.

Associa-se a cada regra (símbolo não-terminal) de uma GCD um predicado PROLOG com dois argumentos extras. Estes argumentos são os pontos inicial e final de um símbolo não-terminal na sentença (Apêndice A).

A sentença deve ser representada por uma lista de palavras. A gramática varre esta lista de palavras da esquerda para a direita, retirando as palavras à medida que

elas são reconhecidas como válidas, nos pontos em que aparecem. Caso a sentença esteja gramaticalmente correta, produz uma lista resultante vazia.

A construção destas estruturas se dá através do processo de unificação, à medida que os predicados do corpo das regras são avaliados com as regras correspondentes. Outros argumentos são acrescentados às regras, para que possam tratar as dependências conceituais (concordância , etc).

O dicionário também recebe informações de gênero e número, e para construir estruturas padronizadas, acrescenta-se a raiz das palavras.

O uso de argumentos extras nos símbolos não-terminais da gramática possibilita expressar na forma livre de contexto, gramáticas que, de outra forma, exigiriam uma representação explícita para as dependências contextuais (dentro da sentença).

O analisador gramatical, neste tipo de gramática, opera de maneira não-determinística e estritamente *descendente*, consumindo as palavras da esquerda para a direita.

As *Gramáticas de Cláusulas Definidas* apresentam grandes vantagens quanto à clareza e modularidade. Devido à independência das regras, torna-se mais fácil estender uma gramática para reconhecer outras construções lingüísticas ou modificar o tipo de estrutura sintática que ela produz.

2.5 - Outros Formalismos Gramaticais

2.5.1 - Gramáticas de Extraposição (XGs)

As *Gramáticas de Extraposição* podem ser consideradas como uma extensão das *Gramáticas de Cláusulas Definidas*. Foram desenvolvidas por Pereira [PEREIRA 83], motivado pela complexidade de expressar o fenômeno da "extraposição à

esquerda", tão freqüente nas linguagens naturais. Tanto as *GCD* quanto as *Gramáticas de Metamorfose* tratam desse fenômeno.

A "extraposição à esquerda" ocorre em sentenças da linguagem natural quando um subconstituinte de algum constituinte é omitido, e algum outro constituinte à esquerda o representa de algum modo.

Considere-se o exemplo abaixo:

Exemplo 1:

O circuito **que** [Kátia projetou t_1] era um inversor.

marcador	traço

Diz-se que o constituinte, cujo lugar o traço ocupa, foi extraposto à esquerda e sua nova posição é representada pelo marcador. O elemento ausente pode ser um objeto direto ou um sujeito, dizendo-se então que houve uma extraposição do objeto ou do sujeito.

A diferença entre as regras de uma gramática de extraposição (XG) e as regras de um GCD está no fato de que uma XG pode ter vários símbolos não-terminais em seu lado esquerdo.

De uma maneira geral, uma regra XG tem a seguinte forma:

$$S_1,\ S_2,\ \ldots,\ S_{k-1},\ S_k \ \text{--}> \ r$$

onde cada Si é uma seqüência de símbolos terminais e não-terminais, concatenados por uma vírgula, com a restrição de que S_i seja um símbolo não-terminal e que r tenha a mesma forma do lado direito de uma regra GCD.

2.5.2 - Gramáticas de Casos

Desenvolvidas por *Fillmore* e publicadas em seu artigo *"The case for case"* [FILLMORE 68], com o objetivo de solucionar alguns problemas que apareciam nas gramáticas transformacionais. Fez uma adaptação do estudo dos "casos" (latim) para a língua inglesa. *Fillmore* propôs novos casos: *"casos-funções"* para tratar a estrutura profunda das sentenças.

As gramáticas de casos, apesar de terem sido desenvolvidas baseando-se em colocações puramente lingüísticas, foram aplicadas em muitos sistemas de processamento de linguagem natural e serviram como fundamento para novas teorias lingüísticas, tais como a dependência conceitual [SCHANK 80].

2.5.3 - Gramáticas Procedurais

Nesta classe de gramáticas estão as Redes de Transição Ampliadas (ATNs). Foram desenvolvidas para fornecer uma estrutura prática para a compreensão de linguagem natural [WOODS 70]. Objetivando combinar as análises sintática e semântica, as ATNs possibilitam que se junte rotinas semânticas a partes específicas do mecanismo de análise sintática ou da gramática.

Uma ATN oferece vantagens, tais como:

(1) O esquema básico de análise sintática é fácil de ser compreendido; a informação gramatical é representada em uma rede de transição, o que, conseqüentemente, facilita o projeto de uma ATN.

(2) A análise semântica é realizada ao mesmo tempo que a análise sintática e, através de restrições, pode resolver ambigüidades.

Os arcos de uma ATN correspondem a palavras e frases (sintagmas). Cada arco é rotulado com uma especificação da condição sobre a qual pode ser percorrido. Tipicamente tais especificações se referem ou a uma palavra, ou a um sintagma, ou a um predicado a ser satisfeito por um prefixo de uma parte ainda não analisada, ou ainda a um nome de outra ATN (que pode ser, recursivamente, ela mesma).

Além das condições, há uma ação associada a cada arco. A ação pode envolver rotinas que fazem interpretações parciais, examinar registros já definidos por outras ações ou até mesmo não realizar nada.

Teoricamente as ATNs e GCDs têm o poder de uma máquina de Turing, e neste sentido são tão gerais quanto se queira [PEREIRA 80].

2.6 - Considerações

Diante da questão de se decidir qual destes formalismos seria mais apropriado para o processamento de linguagem natural, pode-se fazer algumas observações, sobretudo com relação às gramáticas de cláusulas definidas (GCD) e às gramáticas procedurais (ATNs).

As GCDs formam um mecanismo mais eficiente que as ATNs para se construir estruturas. Em uma ATN as estruturas devem se enquadrar na análise recursiva "PUSH/POP" (*chamadas/saídas* de arcos, respectivamente). Isto se deve ao fato de que o arco "POP" pode retornar somente uma estrutura simples e que todos os subconstituintes desta estrutura precisam ser conhecidos quando o "POP" é executado.

Em uma GCD, por outro lado, um símbolo não-terminal retorna mais que uma estrutura como seu resultado e essas estruturas contêm variáveis que somente mais tarde assumirão algum valor. Então, a estrutura gerada em uma GCD, como resultado da análise de uma frase, depende de itens da sentença que ainda não foram analisados.

As GCDs são mais gerais que as ATNs. Uma ATN é um formalismo específico para uma análise do tipo descendente e *no sentido da esquerda para a direita* [1] enquanto que uma GCD por ser antes de tudo uma descrição da linguagem, é neutra quanto à implementação. Em conseqüência, uma GCD pode ser executada de diferentes maneiras.

---

[1] Em inglês: left-to-right

3

ANÁLISE SEMÂNTICA

Dentro do contexto de processamento de linguagem natural, torna-se mais complicado formalizar a análise semântica do que a sintática. A semântica trata das condições de verdade de uma frase.

A palavra "semântica" é usada em áreas do conhecimento tais como: Lógica, Filosofia, Psicologia e Ciência da Computação, com acepções distintas ou análogas. Torna-se necessário explicitar qual é a acepção que está sendo usada.

Aqui, seu uso é para o processamento de linguagem natural, especificamente em sistemas de perguntas e respostas em português. Neste caso, considera-se a semântica como sendo parte da semiótica, e esta como a teoria dos signos - as menores unidades da linguagem (letras ou palavras). A semiótica divide-se em *sintaxe, semântica e pragmática.*

A importância da semântica para PLN é permitir a compreensão do significado e, conseqüentemente, facilitar a comunicação, a expressão de idéias, sentimentos, crenças, etc. O interesse aqui é a compreensão de perguntas para a obtenção das respostas desejadas, com relação a uma base de dados.

Pode-se organizar o processo de interpretação semântica, isto é, a derivação do significado de uma sentença, através de várias abordagens. Inicialmente apresentaremos o princípio da *composicionalidade*, que é comum a todas estas abordagens e procuraremos mostrar uma maneira de se associar o significado a uma frase, fazendo a atribuição de um valor-verdade.

Para nossos objetivos, entretanto, achamos mais conveniente representar o significado através de uma expressão em uma linguagem artificial. Descreveremos então a semântica via sistemas lógicos, onde se procurará "traduzir" a frase em uma expressão da Lógica de Primeira Ordem. Adotamos esta abordagem na interface que implementamos.

3.1- Composicionalidade

De acordo com o princípio da composicionalidade o significado de uma sentença pode ser expresso em termos dos significados das subfrases que estão contidas nesta sentença.

O significado das subfrases, por sua vez, dependem dos significados das subfrases que as compõem, e assim por diante, até que se chegue aos significados das palavras individualmente, ou mesmo aos significados dos morfemas que compõem as palavras.

O *princípio da composicionalidade* pode ser enunciado informalmente como: *"O significado do todo é uma função dos significados das partes"*.

Este princípio vem a auxiliar na organização das idéias e na decomposição do problema. Porém, resta ainda resolver uma série de questões para que se possa construir um processo eficiente de extração do significado de uma sentença, tais como:

- Quais são as subfrases apropriadas a se considerar?
- Como o significado de uma frase particular depende do significado de suas subfrases?
- Quais são os significados das unidades mínimas (frases, palavras, etc) ?
- Que espécie de coisas os significados seriam: símbolos, coisas no mundo, relações entre os dois?

As subfrases de uma frase poderiam ser precisamente aquelas fornecidas pela análise sintática.

Exemplo: **S -> SN SV**

O significado da sentença (S) é algo em função do significado do sintagma nominal (SN) e do significado do sintagma verbal (SV).

Existe alguma relação consistente entre sintaxe e semântica em LN. Uma gramática semanticamente adequada também captura generalizações sintáticas. Supondo que a análise sintática divida frases em subfrases semanticamente apropriadas, resta-nos especificar como o significado de uma expressão complexa depende dos significados das partes. Isto nos leva a uma abordagem onde exista uma correspondência entre regras sintáticas e semânticas.

Para cada regra sintática existe uma regra que diz como o significado de uma tal frase é composto dos significados das partes (hipótese *regra a regra*). Há vários métodos para se implementar computacionalmente as regras semânticas. Em um destes métodos usa-se a árvore de estrutura da frase para obter seu significado.

Supondo-se a seguinte regra em uma gramática:

A -> B C D

Uma regra semântica especifica como os significados das frases de tipos B, C e D, quando estas frases aparecem nesta ordem, podem ser combinados para dar o significado da frase do tipo A. Ou seja, as regras semânticas podem ser vistas como um mapeamento das árvores de estrutura das frases para os significados. Esta abordagem será melhor detalhada nos próximos itens.

3.2 - Significado como Referência

Supondo-se um sistema de perguntas-respostas sobre empresas nacionais, onde sejam consideradas questões do tipo *SIM/NÃO*, pode-se identificar o significado de uma frase com o seu valor-verdade (Verdadeiro ou Falso).

Numa interpretação de semântica computacional associamos a cada regra da gramática uma regra de como o significado de uma tal frase seria construido a partir dos significados das frases constituintes.

Significado de uma Entidade:

O significado de uma entidade poderia ser representado por uma foto, um quadro, um logotipo ou um símbolo associado somente com aquela entidade (usa-se o prefixo # para denotar tal representação).

Exemplo: a palavra *Petrobrás*

< cat > = SN
< significado > = #Petrobrás

O Significado de um Sintagma Verbal:

Exemplo: *" A Petrobrás é uma estatal "*

Temos que construir o significado de uma sentença a partir dos significados de seus constituintes: **SN** e **SV**. Procura-se produzir um valor (**V** ou **F**) a partir de um objeto representado por #Petrobrás, juntamente com todo o significado que se obtém do VP " é uma estatal ".

Para determinar, no mundo real, se *a Petrobrás* é uma estatal, podemos procurar em um catálogo de referências que tenha uma lista com todas as empresas estatais. Para saber se realmente é a *Petrobrás* na qual estamos interessados, possivelmente comprovaríamos através de uma descrição; ou então poderíamos fazer com que o significado de *"é uma estatal"* fosse associado a um conjunto de símbolos do tipo #Petrobrás.

Para se construir o significado de uma sentença contendo o sintagma verbal *"é uma estatal"*, precisa-se verificar se o significado do *sujeito* da sentença pertence ao conjunto que é o significado do sintagma verbal.

Representando um conjunto de elementos através da notação padrão, isto é, colocando os elementos entre chaves "{" e "}" , pode-se associar o significado do sintagma verbal *" é uma estatal"* ao conjunto:

{ #Petrobrás, #Embratel, #ECT, #Banco do Brasil, ... }.

Usando-se operações com conjuntos, tais como:

X ∈ S — X é um elemento do conjunto S
(isto tem o valor VERDADEIRO ou FALSO)

S1 ∩ S2 — o conjunto de todos os elementos que estão tanto em S1 como em S2

pode-se formular regras para sintagmas verbais e para sentenças, por exemplo:

Regra: **S -> SN SV**

<significado de S> = V se <significado de SN> ∈ <significado de SV>
F em caso contrário

Sintagma *" é uma estatal "*

<cat> = SV
<significado> = { #Petrobrás, #Embratel, #Banco do Brasil, ... }

Palavra *" Petrobrás"*

<cat> = SN
<significado> = #Petrobrás

Uma regra para a conjunção de duas sentenças seria:

Regra: **S0 -> S1 CONJ S2**

<significado de S0> = V se <significado de S1> = V
e <significado de S2> = V
F em caso contrário

Este sistema poderia ser espandido para manipular, por exemplo, a quantificação imposta pelos artigos. Assim seria possível distinguir semanticamente:

"uma estatal" e *"toda estatal"*

Uma forma seria em termos de propriedade. Podemos perceber a distinção identificando o significado de um sintagma nominal com o conjunto de propriedades que os referidos objetos têm. Estas propriedades também são vistas como conjuntos.

Exemplo:

O significado de um nome próprio, como *Petrobrás*, é tido como o conjunto de todos os conjuntos que têm #Petrobrás como membro.

	significado
"uma estatal"	Conjunto de todos os conjuntos de entidades que têm no mínimo uma estatal como membro.
"toda estatal"	Conjunto de todos os conjuntos de entidades que têm todas as estatais como subconjunto.

Limitações:

Torna-se complicado lidar com todos estes conjuntos. Podemos representar um conjunto implicitamente através de um **procedimento** que nos dirá se um dado objeto é um elemento deste conjunto. Também não está claro se os valores-verdade são os significados adequados às sentenças, quando colocamos juntos os significados, em uma forma composicional.

Por exemplo a frase "*creio que o céu está claro*" poderia ser falsa. O valor-verdade desta frase não pode ser determinado simplesmente pela referência ao valor-verdade de "*que o céu está claro*". Para se resolver isto pode-se modificar as

regras sintáticas ou, melhor ainda, introduzir uma noção mais sofisticada do significado de uma sentença que o seu valor-verdade. Uma forma de se fazer isto seria a tradução da sentença em uma linguagem de representação do significado.

3.3 - Tradução para uma Linguagem de Representação do Significado

Obter o valor-verdade de uma frase pode ter sua conveniência, mas é mais fácil imaginar outras maneiras pelas quais podemos querer manipular o significado de uma frase.

Por exemplo, podemos querer determinar se acreditamos em uma dada frase; que conclusões seguem de uma frase junto a nossas crenças anteriores; ou, ainda, podemos querer dizer se duas diferentes frases significam a mesma coisa. Em resumo, queremos mais informações sobre o significado de uma frase que, simplesmente, seu *valor-verdade*.

Tornou-se então, tradicional, "aproximar" o significado de uma frase, traduzindo-a numa expressão em uma linguagem artificial , própria para representar o significado. Optou-se por uma "Linguagem de Representação do Significado" (LRS), que de alguma forma, é mais bem comportada que a "Linguagem Natural". Isto trouxe algumas vantagens, tais como:

(1) Enquanto que a Linguagem Natural é ambígua, a LRS é precisa.

(2) Ao passo que é difícil especificar regras para determinar quando uma frase em LN é verdadeira, podemos querer que uma LRS venha completa com regras que nos permitam derivar as condições que o universo deveria ter para satisfazer uma declaração arbitrária (as condições para tornar a declaração verdadeira).

(3) Enquanto que é complicado determinar o que se segue a uma declaração em LN, podemos desejar que uma LRS venha com regras de inferência que nos

possibilitem derivar mecanicamente novas declarações que se seguem a uma dada declaração.

Regras Semânticas

Regras semânticas podem ser definidas como o mapeamento entre árvores de estrutura frasal e representação do significado. Se representações do significado são objetos estruturados, como o são as expressões em linguagens formais, então podemos considerar o mapeamento como aquele que produz árvores de estrutura frasal para as representações do significado. Temos então mapeamento de ávores em árvores.

Para se produzir uma tal representação, deve-se basear na representação do significado das subfrases. Então, uma regra de tradução semântica diz que uma árvore local de uma forma particular mapeia em uma árvore local de outra forma particular.

Subárvores da árvore original são mapeadas em subárvores de uma nova árvore em uma forma especificada. O mapeamento de subárvores, em geral, vai requerer o uso de outras regras de tradução semântica em um modo recursivo.

Vendo-se tradução semântica para uma linguagem de representação do significado como um mapeamento de árvore local, não é necessário discutir sobre quando as regras de tradução semântica devem ser chamadas: o significado de cada subfrase poderia ser computado logo que a árvore da análise sintática tivesse sido construída, ou então após a construção da árvore da sentença inteira.

Assumindo que ambas, LN e LRS, podem ser descritas pelo mesmo tipo de gramática, não há diferença entre os tipos de objetos que são as árvores de análise da LN e as árvores das expressões semânticas.

Uma técnica comum em geração de LN, que podemos chamar de geração orientada a mensagem, é usar regras composicionais que mapeiem de estruturas da semântica original (mensagem) em estruturas representando objetos lingüísticos.

Outro método de Interpretação Semântica

Logo que o analisador reconhece a presença de um sintagma consistindo de alguma sequência de subsintagmas (isto é, o analisador pode aplicar uma regra sintática completa), o significado do sintagma é imediatamente extraído dos significados das partes.

Em tal método o significado de uma simples palavra poderia ser construído logo que o analisador a encontrasse. Além disso, o analisador teria que memorizar qual a regra sintática que estaria usando correntemente e qual o significado dos subconstituintes encontrados até o momento. Nesta abordagem, há uma tradução direta entre a LN e a representação semântica, sem qualquer nível intermediário de representação.

Esta abordagem exige que se tenha um regime de representação *regra-a-regra*, enquanto que numa abordagem de mapeamento em árvore (que executa a interpretação da sentença completa), a interpretação composicional estaria livre para reconhecer e operar em árvores locais, originando-se da aplicação de várias regras gramaticais.

3.4 - Uma Classificação de Semântica para os Objetivos do Processamento de Linguagem Natural

Para nossos interesses podemos classificar a semântica em dois grupos: **lexical** e **formal** (composicional, aplicada e lógico-sintática).

Semântica Lexical

É baseada em primitivas semânticas [SCHANK 75]. Tem uma visão mais tradicional sobre a Semântica (e isso traz limitações). Dá mais importância ao conteúdo das palavras, descrevendo-as como enunciados complexos. As primitivas semânticas baseiam-se nos conceitos: *agente, objeto, instrumento e recipiente.*

Nesta abordagem a representação semântica é uma paráfrase, onde os conteúdos das palavras da sentença original foram substituídos por suas representações e combinados entre si, de modo a preencher adequadamente as lacunas existentes nos argumentos dessas representações.

Produz-se uma paráfrase de entrada e a partir dela se responde às perguntas formuladas. A semântica lexical não supre um mecanismo preciso e geral para derivar respostas de perguntas a partir de uma base de dados.

A única noção de conseqüência na semântica lexical é a de que *"um enunciado X é uma conseqüência do enunciado Y se o gráfico léxico de X é uma parte do gráfico léxico de Y".* Isto não é satisfatório, pois a mesma pergunta pode levar a gráficos lexicais distintos, o que dificultaria a resposta, já que a equivalência de significados é entendida como equivalência de gráficos lexicais (gráfico léxical é uma paráfrase de qualquer entrada em termos de primitivas).

Uma outra limitação é que o número de conseqüentes que podem ser extraídos de um gráfico léxical é finito, enquanto que, de determinados enunciados, pode-se tirar uma infinidade de conseqüências. Este método apresenta-se favorável, por exemplo, para resolver certos problemas de ambigüidades.

A partir da concepção tradicional de semântica, a Semântica Lexical postula que o significado de uma sentença deve ser descrito através das palavras. As idéias contidas na *Dependência Conceitual* são importantes mas não estão devidamente precisadas. Surgem problemas relacionados com a unicidade da paráfrase em um gráfico e com a interpretação para cada verbo da ação correspondente.

Semântica Formal (via Sistemas Lógicos)

Subdivide-se em: - *composicional*
- *formal aplicada*
- *lógico- sintática*

Semântica Composicional

É uma abordagem da linguagem natural por instrumentos lógicos que não diferem da abordagem das linguagens formais [MONTAGUE 74].

A análise de frase no nível mais profundo consiste na tradução em uma fórmula de cálculo lógico, por regras que descrevem como a tradução de uma frase é construída a partir das fórmulas de suas subfrases e do contexto de ocorrência da frase.

Através de uma lógica intensional, a semântica composicional procura mapear as regras sintáticas nas semânticas. Usa um formalismo complexo e suas regras são de difícil compreensão à primeira vista. Não se conhecem trabalhos que tenham implementado totalmente esta semântica.

Semântica Formal Aplicada

Baseia-se nos Sistemas Lógicos e consiste em considerar o significado da expressão como sua interpretação, atribuindo-se valores-verdade.

A interpretação para o cálculo de predicados consiste em considerar os constituintes das fórmulas da linguagem formal e interpretá-los segundo uma estrutura constituída de:

- *funtores e predicadores* que são interpretados como funções e relações da estrutura.
- *constantes* que são nomes de entidades do universo do discurso.
- *variáveis que* são avaliadas através de uma função de avaliação.
- *conectivos.*

Em semântica para processamento de linguagem natural, o fundamental é a possibilidade do processamento pela máquina. Precisa-se encontrar um análogo ao valor-verdade que caracterize a semântica de frases interrogativas. A linguagem PROLOG permite que se obtenha respostas corretas no que depende do mecanismo de inferência.

3.5 - Semântica via Sistemas Lógicos

Semântica para LN usando sistemas formais lógicos significa considerar a interpretação das sentenças como o seu valor-verdade. *Tarski* caracterizou "verdade" em sistemas formais [TARSKI 44] . Foi a partir desse trabalho que um estudo de semântica para a formalização do raciocínio lógico se tornou possível.

Sistema Lógico Formal

Caracteriza-se por uma linguagem formal. De acordo com a interpretação dos conectivos e operadores tem-se os diversos sistemas lógicos: **clássicos** (Lógica de Primeira Ordem), **trivalentes** [COLMERAUER 77] e **modais.** Aqui abordaremos os sistemas lógicos trivalentes.

3.6 - O Sistema Trivalente de Colmerauer

Colmerauer procurou definir um subconjunto mínimo da linguagem natural que pudesse ser usado, por exemplo, para criar e consultar bases de dados [COLMERAUER 77] . Para delimitar tal subconjunto são colocadas restrições iniciais, simplificando a sintaxe e definindo rigorosamente a semântica.

Na construção deste subconjunto, parte-se de enunciados atômicos, como nomes próprios, criando uma estrutura própria para o tratamento de artigos. Constrói-se estruturas mais complexas, capazes de lidar com frases relativas, conjunções e negações. Procura-se também criar uma forma sistemática de transformar sentenças da linguagem natural em fórmulas semânticas.

Colmerauer enunciou sete regras (hipóteses) metalingüísticas. Observa-se que, neste contexto, a palavra **interpretar** tem o sentido da *Teoria dos Modelos*, ou seja, o de se obter um *valor-verdade* numa determinada situação, em contrapartida com o sentido de interpretação na Lógica clássica.

Regras de Colmerauer:

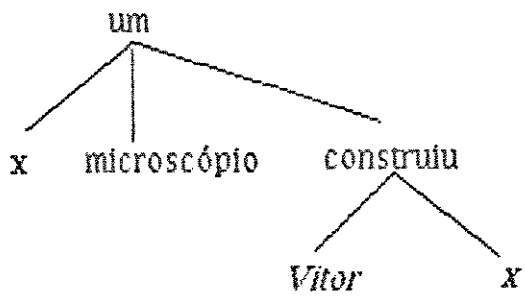
Regra 1: A cada verbo, a cada adjetivo e a cada substantivo corresponde uma propriedade com n argumentos (predicado de peso n), cada argumento sendo um nome próprio.

Exemplo: "Um professor da Unicamp visitou a USP"

verbo: visitou(usp)
substantivo: professor(unicamp)

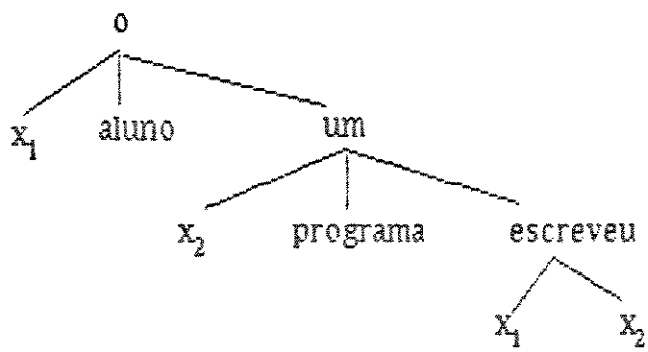
Regra 2: Em geral, a cada artigo d corresponde um "quantificador de três ramos" q, o qual, a partir de uma variável X e de duas fórmulas f_1 e f_2, cria a nova fórmula: $q(X,f_1,f_2)$.

Exemplo: *"Vitor construiu um microscópio"*



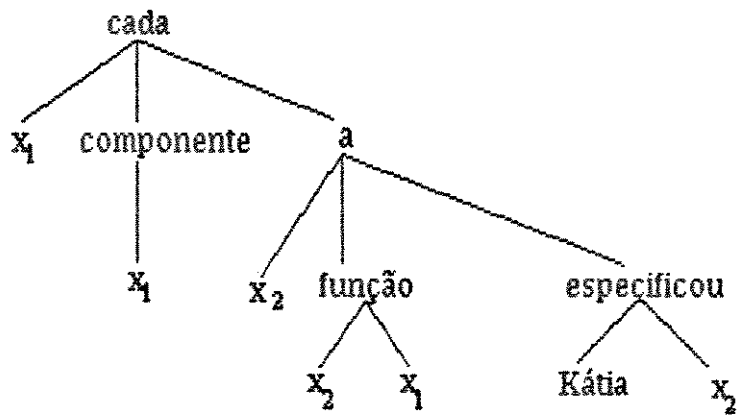
Regra 3: A quantificação introduzida pelo artigo do sujeito de um verbo domina a(s) quantificação(ões) introduzida(s) pelo(s) complementos(s) do verbo.

Exemplo: *"O aluno escreveu um programa"*



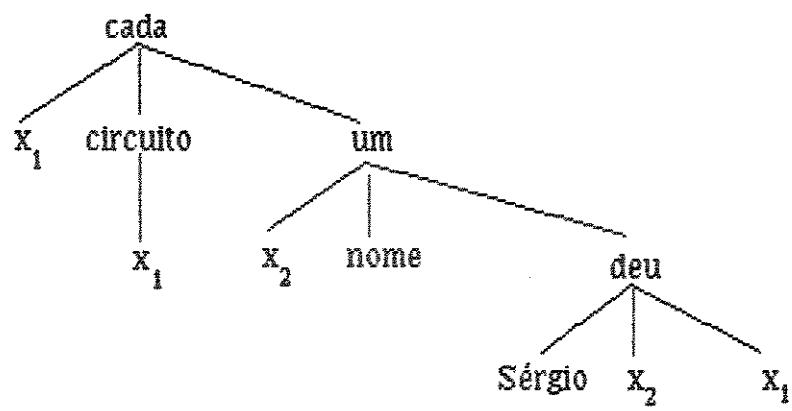
Regra 4: Numa construção envolvendo um substantivo e um complemento deste substantivo, a quantificação introduzida pelo artigo do complemento domina a quantificação introduzida pelo artigo do substantivo.

Exemplo: *"Kátia especificou a função de cada componente"*



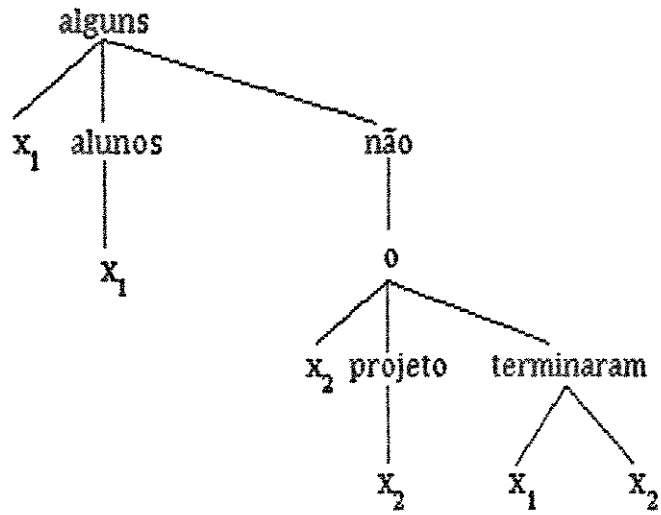
Regra 5: Quando um verbo, um adjetivo, ou um substantivo tem dois complementos, a quantificação é feita na ordem inversa da ordem natural da escrita, isto é, o complemento mais à direita dá uma quantificação que domina a quantificação gerada pelo outro complemento.

Exemplo: *"Sérgio deu um nome para cada circuito"*



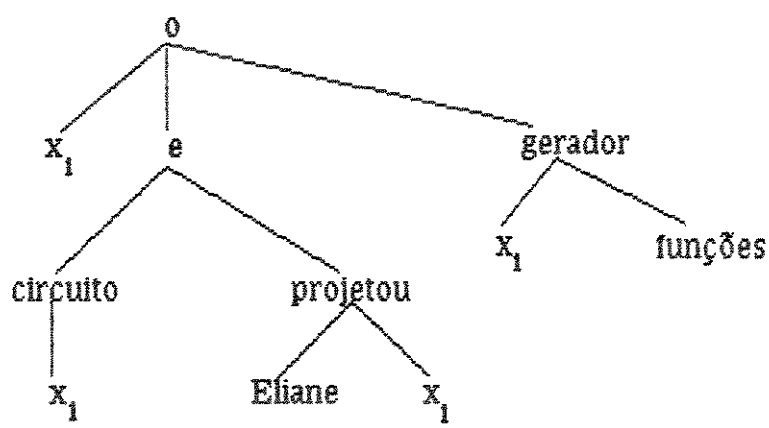
Regra 6: A negação introduzida pela partícula *"não"* é traduzida pelo operador *"não"* colocado imediatamente abaixo da quantificação introduzida pelo sujeito. Entretanto, se o artigo do sujeito é *"cada"*, *"todo"*, etc., o operador *"não"* se aplica ao enunciado todo.

Exemplo: *"Alguns alunos não terminaram o projeto"*



Regra 7: Toda cláusula relativa é tratada como um enunciado comum; o pronome relativo é substituído por variáveis apropriadas e o todo é ligado à substituição do substantivo pela conjunção *"e"*.

Exemplo: *"O circuito que Eliane projetou é gerador de funções"*



O Sistema Lógico

O sistema lógico proposto por *Colmerauer* é um sistema **trivalente**, onde surge a necessidade de um valor *"indefinido"*.

Por exemplo, na sentença *"XYZ é uma empresa nacional"*, se na base de dados, que contém nome de empresas, não constar a *empresa XYZ*, então a sentença não terá sentido neste sistema. Daí se dizer que tem um valor indefinido.

Construção do Sistema

Para a construção deste sistema lógico define-se:

- Um novo operador binário ***"se"***:

se **<p,q>** –> Se ***p*** = **verdadeiro**, então ***q***
caso contrário, indefinido.

- Um outro operador de negação ***"não-verdadeiro"***.

Para um melhor tratamento de plurais, as propriedades introduzidas por verbos, adjetivos e substantivos, se aplicarão sobre o conjunto de indivíduos e não sobre indivíduos [COELHO 79].

Seja:

K -> Conjunto de símbolos chamados nomes próprios.
R -> Conjunto de símbolos relacionais, tal que, para cada elemento r de R, corresponde um inteiro positivo, chamado *"grau de r"*.
X -> Conjunto infinito de variáveis.

fórmula enunciado -> Qualquer fórmula do tipo:

a) $r(s_{i1}, ..., s_{ip})$, com r elemento de R e grau <r>=p

b) $e(e_j, e_k)$

c) $se(e_j, e_k)$

d) $não(e_j, e_k)$

e) $não\text{-}verdadeiro(e_j)$

f) $igual(n_j, n_k)$

g) $menor(n_j, n_k)$

h) $para(x, e_j, e_k)$, com $x \in X$

fórmula conjunto -> Qualquer fórmula de um dos tipos:

a) c, $c \in K$

b) k, $k \in X$

c) aqueles(x, e_j), $x \in X$

fórmula inteira -> qualquer fórmula de um dos tipos:

a) j, onde j é um inteiro não negativo

b) card(s_1), onde s_1 é uma fórmula conjunto.

Diz-se que a ocorrência de uma variável *x* numa fórmula *f* é *livre*, se ela não aparece dentro de uma subfórmula da forma *aqueles(x,e)* ou *para(x,e,e')*. Uma fórmula que não contém variáveis livres é dita *fechada*.

Semântica do Sistema

Para a obtenção do valor verdade de uma fórmula precisa-se da noção de situação que é uma simplificação da noção de interpretação da Lógica de Primeira Ordem (LPO).

Definição de situação:

Uma "situação" S é um mapeamento que, a cada símbolo relacional r pertencente a R de grau n, associa uma relação n-ária S_n cujos argumentos E_i são os subconjuntos do conjunto K de nomes próprios, e cujos valores $(S_r)<E_1,...,E_n>$ são "verdadeiro" ou "falso" ou "indefinido".

Para se definir o valor de uma fórmula não fechada, é necessário que se defina uma *"situação estendida"*.

Uma **"situação estendida"** S é uma situação que foi estendida de tal forma que associa a cada variável x pertencente a X um subconjunto S_x do conjunto K de nomes próprios.

Para que se possa fazer a tradução de sentenças do Português para fórmulas lógicas, usa-se as equivalências (adaptadas por COELHO, para representação de artigos , através do quantificador "*para*" [COELHO 79]).

O trabalho de Colmerauer vale pelo seu pioneirismo no uso da lógica para representar o significado de sentenças em LN. Na lógica a noção de Semântica está bem clara, precisa e formalizada através da idéia de interpretação.

Colmerauer não apresenta qualquer operador de coletivização, necessário para trabalhar com conjuntos. Não fica claro qual o subconjunto da LN abrangido pelo sistema lógico. Não se distinguem outras formas do *"não"* quando a quantificação é universal. O tratamento com domínios infinitos se torna impossível e a computação do significado de uma fórmula é muito ineficiente.

3.7 - A Lógica de Primeira Ordem Estendida de Pereira

Partindo do estudo feito por *Colmerauer, Pereira* faz duas extensões para cobrir a ausência de negações, de quantificação universal e de conjuntos [PEREIRA 83] . Por causa destas extensões, o sistema perde algumas de suas propriedades metamatemáticas como, por exemplo, a *completude* e a *decidibilidade*.

A necessidade das extensões feitas podem ser observadas em análises de sentenças do tipo:

"Quais subcircuitos ocorrem mais de uma vez no circuito A?"

Para se representar o significado desta sentença é necessário um tratamento para a *cardinalidade* de conjuntos, de que a lógica de cláusulas definidas não dispõe. Pereira criou um sistema chamado de "*Cláusulas Definidas num Mundo Fechado*", que tem por objetivo representar o significado de perguntas em linguagem natural.

As fórmulas que podem ser mapeadas em cláusulas definidas, num "mundo fechado", são da forma:

literal <= condição

onde **condição** pode conter quantificadores e conectivos e ainda, em ambos os lados, as variáveis livres são quantificadas universalmente, no nível mais externo da cláusula.

4

IMPLEMENTAÇÃO DE UMA INTERFACE EM LN

Como exemplo de uma interface em LN para acesso a uma base de conhecimentos, implementamos o programa ARQUIVOS. Ele permite que o usuário, através de sentenças em linguagem natural, obtenha informações sobre arquivos que estão armazenados em discos de um microcomputador PC. Este programa também é uma interface entre o usuário e os comandos do sistema DOS, no que diz respeito à manipulação de arquivos.

Para sua implementação usamos o formalismo das gramáticas de cláusulas definidas (v. item 2.4 e Apêndice A) e o sistema lógico trivalente de Colmerauer (v. item 3.6), adaptado por Coelho [COELHO 79] para a língua portuguesa.

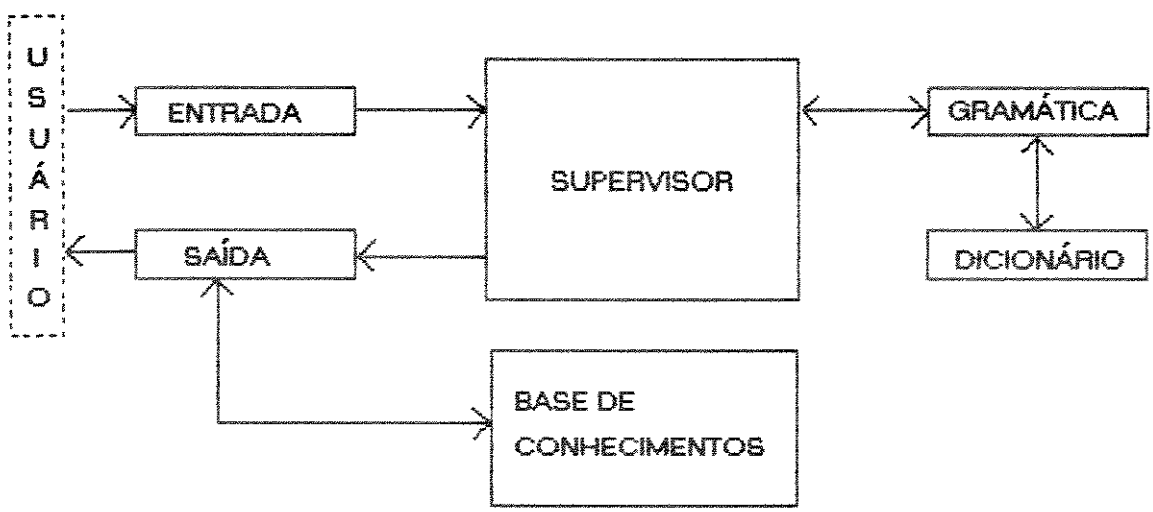


Fig. 4.1 - Organização do programa ARQUIVOS

ARQUIVOS é um programa organizado em módulos (fig. 4.1). Isto possibilita uma maior flexibilidade, permitindo que se faça modificações em um módulo, em particular, sem afetar os demais, ou mesmo o projeto como um todo. Neste capítulo descreveremos os módulos constituintes deste programa, procurando mostrar como são construídas as fórmulas lógicas, que correspondem às frases analisadas corretamente.

4.1 - Entrada

O módulo de entrada recebe a sentença, digitada pelo usuário. Em seguida transforma a seqüência de palavras e sinais de pontuação digitados em uma lista de palavras e sinais de pontuação. Esta lista será usada pelo módulo "*gramática*", como um argumento extra do predicado *"sent_principal"*, para que os elementos desta lista (as palavras) sejam "consumidos" através da aplicação das regras da gramática de cláusulas definidas.

Uma outra função deste módulo é a de verificar se todas as palavras da sentença do usuário são conhecidas.

4.2 - Supervisor

Este módulo é a estrutura de controle do programa ARQUIVOS. Através dele se inicia o programa e se processa a interação com o usuário, gerenciando o diálogo: solicitando ou fornecendo informações. Ele chama o módulo *"gramática"* para a análise da sentença, decidindo sobre a aceitação ou recusa do resultado da análise. Em caso de aceitação, chama o módudo *"saída"* para que seja fornecida a resposta ao usuário.

4.3 - Base de Conhecimentos

A base de conhecimentos de ARQUIVOS é constituída de um conjunto cujos objetos são os arquivos contidos nos discos de um microcomputador PC (no momento em que o usuário o estiver usando) e conhecimentos sobre este domínio. Inclui também os procedimentos de recuperação destas informações, a partir de estruturas lógicas. Por questão de simplicidade, incluiremos aqui apenas a unidade de disco rígido, que poderá estar particionada em mais de uma unidade lógica.

Os itens referentes aos arquivos foram organizados em um conjunto de tuplas, onde o acesso pode ser feito através de qualquer combinação de itens (como numa base de dados relacional). Os conhecimentos sobre o domínio de arquivos estão descritos através de predicados (regras) que definem as relações pertinentes a este domínio.

Uma regra define uma relação e exerce uma função de acesso à base de dados, fornecendo a possibilidade de respostas que não estão explicitas no conjunto destas tuplas. Sendo assim, poderíamos chamar de ***base de dados*** ao conjunto de tuplas e de **base de conhecimentos** ao conjunto mais abrangente, o qual contém, além da base de dados, aquelas regras que descrevem o domínio dos arquivos.

Usando-se a linguagem de programação PROLOG implementamos a base de dados, onde cada arquivo é representado por uma estrutura de nome *"arquivo"*, possuindo argumentos, que são os dados referentes àquele arquivo. Por exemplo, um arquivo de nome *"ARQ.EXT"* é armazenado da seguinte forma:

```
arquivo( $ARQ.EXT$,   32,   hora(9,25,50,0),   data(1990,10,18),   25730,   $C:\USR\$ )
           (1)        (2)        (3)               (4)              (5)        (6)
```

onde os argumentos representam os seguintes dados:

(1) nome do arquivo (e sua extensão)
(2) código referente aos atributos do arquivo
(3) hora, minuto e segundo da criação do arquivo
(4) ano, mês e dia da criação do arquivo

(5) tamanho do arquivo (em bytes)
(6) diretório onde se encontra o arquivo

Todos os arquivos presentes no disco, em um dado instante da consulta, estarão armazenados através deste termo. O processo da construção da base de dados ocorre logo no início, ao se chamar o programa ARQUIVOS. Este processo é realizado em três etapas: em primeiro lugar se faz o acesso ao sistema DOS, lendo-se os arquivos no disco. Em seguida, os dados são arranjados em um formato mais adequado (*termos* do PROLOG). Finalmente os termos são armazenados na base de dados.

4.4 - Processamento da Linguagem

A parte lingüística de ARQUIVOS é formalizada como uma gramática de cláusulas definidas (v. item 2.4 e Apêndice A), que usa um fragmento da sintaxe e do vocabulário da língua portuguesa que é relevante para o domínio de diálogos, neste ambiente de manipulação de arquivos.

Esta gramática satisfaz os dois principais critérios para que seja um modelo adequado de um subconjunto do português: primeiro é capaz de distinguir o conjunto de sentenças corretas e, em segundo lugar, associa uma estrutura lógica a cada sentença.

São considerados quatro aspectos ao se escrever a gramática, neste contexto:

(1) Os diálogos são analisados a fim de mostrar os tipos de construções lingüísticas envolvidas e as categorias sintáticas básicas.

(2) As palavras são classificada em dois grupos:

- **vocabulário principal** (artigos, preposições, advérbios, pronomes, e numerais): são independentes do domínio e podem ser usados em outros domínios de diálogos.

- **vocabulário periférico** (substantivos, verbos e adjetivos): são dependentes do domínio .

(3) As palavras são definidas levando-se em consideração sua participação (função, forma e significado) nos diálogos.

(4) As regularidades observadas nas construções lingüísticas - como são formadas e entendidas - são sintetizadas em regras gramaticais que reconhecem corretamente a estrutura das sentenças da língua portuguesa.

4.4.1 - Categorias Sintáticas Básicas

Categorias sintáticas são conceitos primitivos, a partir dos quais as regras sintáticas são construídas. Relacionaremos aqui as categorias básicas usadas para compor sentenças em português: *sintagmas nominais e verbais; artigos; nomes (substantivos), verbos, adjetivos e seus complementos e orações relativas.*

Sintagmas nominais e verbais

Um **sintagma nominal** tem a seguinte forma:

(1) Uma seqüência de nomes próprios ligados por um "e" , "ou", ou por uma vírgula.

(2) Um artigo (às vezes implícito), uma seqüência de adjetivos ou complementos de um nome e, eventualmente, uma seqüência de orações relativas

Um **sintagma verbal** consiste de uma negação (opcional), um verbo e seus complementos.

Artigos

Em nossa gramática, além dos artigos, comumente classificados como tal, incluiremos também nesta categoria algumas palavras, tais como: alguém, cada, nenhum(a)(s), ninguém, qualquer, quaisquer, todo(a)(s).

Nomes

Há dois tipos de nomes: **nomes comuns** (arquivo, discos) e **nomes próprios** (Unicamp, Campinas). Observamos que os nomes próprios compostos serão reconhecidos pelo programa, por serem escritos entre aspas, como por exemplo o nome próprio "Banco do Brasil".

Adjetivos

A noção de adjetivos é flexível: o particípio de verbos intransitivos será identificado como sendo um adjetivo, levando-se em conta que sua função é a de qualificar um nome.

Verbos

Os verbos serão definidos através de suas formas simples, nos tempos presente e passado (singular e plural).

Complementos de Nomes, Verbos e Adjetivos

Todos os complementos serão tratados da mesma maneira. Ao se definir cada nome, verbo ou adjetivo, adiciona-se informações sobre os tipos de complementos esperados, com uma indicação sobre cada preposição (opcional) que introduz o complemento. Por exemplo, o adjetivo *"apagados"* pode aparecer com a preposição *"em"* ou *"após"* ou *"antes"*, etc.

Os complementos dos verbos serão classificados conforme a classificação tradicional, tal como objeto direto ou indireto ou bitransitivos. Também observamos que existe um tipo de complemento que é exclusivo dos verbos *"ser"* e do verbo *"estar"*, que é o *predicativo do sujeito*.

Orações Relativas

Consideraremos aqui somente as orações relativas do tipo restritivo. Elas são iniciadas por pronomes relativos e são precedidas por um outro sintagma nominal, o qual está sendo restringido. A oração relativa funciona, assim, como um adjetivo.

4.4.2 - Dicionário

A composição de cada parte do dicionário (vocabulário principal e vocabulário periférico) leva em consideração a semântica das palavras. A semântica envolve a descrição da forma, função e significado das categorias sintáticas básicas.

Vocabulário principal

É a parte do dicionário independente do domínio. Abrange as seguintes palavras:

Artigos definidos: o(s), a(s).
Artigos e pronomes indefinidos: alguém, algum(s), alguma(s), cada, nenhum(a), ninguém, qualquer, quaisquer, todo(s) o(s), toda(s) a(s), um, uns, uma(s).
Pronomes interrogativos: quanto, quantos, quantas.
Pronomes relativos e interrogativos: onde, qual, quais, que, o que, quem, cujo(s), cuja(s), o(a) qual, os(as) quais.

Advérbios: bem, como, mais.
Preposições: a, após, até, com, de, desde, em , entre, para, por, sobre.
Conjunções: e, quando.
Contrações de preposições: à, ao, aos, do(s), da(s), no(s), na(s), num(a), nuns, numas, pelo(s), pelas(s).
Numerais: um, dois, ..., dez.

Vocabulário periférico

É a parte do dicionário dependente do domínio. Abrange as seguintes palavras:

Nomes comuns: arquivo(s), disco(a), unidade(s), diretório(s), extensão(ões), subdiretório(s), raiz, etc.
Nomes próprios: (Os nomes dos arquivos que estarão na base de dados).
Verbos: apagar, apague, deletar, delete, criar, crie, copiar, copie, transferir, transfira, é, são, foi, foram, tem, têm, existe, existem, etc.
Adjetivos: arquivado(s), apagado(s), deletado(s), criado(s), etc.

4.4.3 - Sintaxe e Semântica

O significado de palavras como **nomes, verbos** e **adjetivos** é descrito através da especificação da propriedade que eles introduzem e da caracterização dos argumentos associados. Uma propriedade é o predicado **"pr"**, cujo único argumento pertence ao conjunto das regras da base de conhecimentos. As regras permitem o acesso aos dados referentes aos arquivos.

Por exemplo o significado do verbo apagar é descrito pela propriedade **"pr(delete(Nome_do_arquivo))"**, que será interpretada pelo predicado **"delete"**, da base de conhecimentos, o qual irá, efetivamente, deletar o arquivo desejado, como também atualizar as informações da base de dados para incorporar esta alteração que está sendo realizada no conjunto dos arquivos.

A cada argumento de uma propriedade estão associados os seguintes parâmetros: **domínio** (D), **gênero** (G), **número** (N), **concordância** (Conc) e, opcionalmente, **caso** (Caso).

O **domínio** especifica um campo (subconjunto) da base de dados no qual se pode classificar a palavra que está sendo analisada semanticamente. Por exemplo, a propriedade que representa o nome de um arquivo está associada ao domínio *"arq"*. Isto permite que se faça restrições e facilite, por exemplo, a identificação de estruturas semanticamente incorretas, como também o acesso à base de dados.

O **gênero** pode ser masculino *(mas)* ou feminino *(fem)* e o **número** pode assumir os valores singular *(sin)* ou plural *(plu)*. O gênero e o número definem o parâmetro **concordância** *(Con)*.

O **caso** pode assumir um dos seguintes valores:

- *suj* (sujeito)
- *obj_dir* (objeto direto)
- *pred_suj* (predicativo do sujeito)
- *prep* (preposição)

Definições

No programa ARQUIVOS as definições para nomes comuns, nomes próprios, verbos e adjetivos apresentam as formas dadas abaixo. Estas formas estão escritas de acordo com o formalismo das gramáticas de cláusulas definidas, dentro do ambiente da linguagem PROLOG. As variáveis (nomes iniciados com letras maiúsculas) seguem, para efeito de clareza, a seguinte convenção:

X está associada com o primeiro argumento da "Propriedade",
D com o domínio,
G com o gênero,
N com o número,
Con com a concordância,
Caso com o tipo de caso (sujeito, objeto direto, predicativo do sujeito, etc).

L é uma lista de itens do tipo "**Caso-Con-D-X**", um para cada argumento extra da "Propriedade", isto é, para qualquer argumento além do primeiro. Se a "Propriedade" tiver apenas um argumento, então **L** será uma lista vazia.

Propriedade é um termo como, por exemplo, *"delete(Nome_do_arquivo)"*. Este termo (*"Propriedade"*) é o único argumento do predicado "**pr**", responsável por manipular cada propriedade com respeito à recuperação de dados e às relações entre seus argumentos, considerados como elementos ou conjunto de elementos da base de dados.

Tipo é um termo que indica o significado de um grupo de palavras. Por exemplo as palavras *"deletar"* e *"apagar"* têm o mesmo tipo *"del"*. O *"Tipo"* é responsável pela escolha da cláusula onde está definido corretamente o significado de uma palavra.

Observação: As variáveis sublinhadas (neste item) indicam que, nos lugares onde elas ocorrem, os argumentos aparecem como termos constantes.

Nome comum

nome([Con-D-X|L], pr(Propriedade)) --> **no**(Tipo, Con).
no(Tipo, G-N) --> [Nome_comum], {**no1**(Nome_comum, Tipo, G-N) }.
no1(Nome_comum, Tipo, G-N)

Nome próprio

np(Nome_próprio, G, D).

Verbos

verbo([N-D-X|L], pr(Propriedade)) --> **ve**(Tipo, N).
ve(Tipo, N) --> [Verbo], {**ve1**(Verbo,Tipo, N)}.
ve1(Verbo, Tipo, N).

Adjetivos

adj([A-D-X|L], **pr**(Propriedade)) --> **ad**(Tipo, A).
ad(Tipo,Con) --> [Adjetivo], {**ad1**(Adjetivo, Tipo, Con)}.
ad1(Adjetivo, Tipo, G-N).

Estas regras gramaticais são indexadas através das palavras de entrada, que estão armazenadas como cláusulas unitárias. Este fato permite, por exemplo que os nomes próprios sejam anexados ao dicionário à medida em que os nomes dos arquivos estiverem sendo armazenados na base de dados. Em outras palavras, este recurso permite a ampliação do dicionário durante o diálogo.

Para exemplificar as definições acima, apresentamos o exemplo da definição da forma verbal "**apague**".

verbo([N-dom(arq)-X], pr(delete(X)) --> **ve**(del, N).
ve(Tipo, N) --> [Verbo], {**ve1**(Verbo,Tipo, N)}.
ve1(apague, del, sin).

A relação de uma palavra com outras palavras é governada pela definição dessa palavra, através de duas regras de concordância: uma diz respeito ao domínio da palavra, a outra se refere à sua concordância de gênero e número.

A especificação de um domínio para cada argumento (representado como uma variável da propriedade) estabelece a relação da palavra a ser definida através de outras palavras da linguagem e detecta anomalias semânticas. O domínio especifica o universo de possíveis indivíduos, sobre os quais são feitas restrições definidas pelas relações.

Por exemplo, o programa não aceita a pergunta: *"Quais os diretórios cuja extensão é EXE ?"* porque o domínio no qual se classificam os nomes de diretórios não admite a existência de *extensões* para os mesmos. Ao se definir o nome comum *"extensão"*, o argumento relativo ao domínio do complemento desta palavra será *"arq"*, indicando que somente irá aceitar como complemento um nome de arquivo, ou seja, uma palavra que pertença ao domínio dos arquivos.

A especificação de concordância, através do gênero e número, também detecta erro, como no exemplo: *"As arquivos apagadas... "*. Esta sentença não é aceita pela falta de concordância de gênero, porque o artigo *"as"* e o adjetivo *"apagadas"* estão no feminino-plural, equanto que o nome comum *"arquivos"* está no masculino-plural.

O nome comum é a cabeça da sentença, porque sua forma governa a forma que as outras duas palavras podem tomar, como também o tipo de palavra ou seqüência de palavras que são possíveis no restante da sentença.

4.4.4 - Gramática

O módulo "Gramática" do programa ARQUIVOS é um conjunto de regras que formaliza a estrutura da lingua portuguesa; recusa sentenças não-gramaticais e compõe uma descrição lógica do significado de uma sentença.

Listamos a seguir um resumo das principais regras sintáticas usadas no programa ARQUIVOS para analisar o fragmento da lingua portuguesa, que o usuário normalmente emprega em suas sentenças, com o objetivo de obter as informações de que necessita. Por se tratar de um resumo, e para facilitar a leitura, apresentamos os termos não-terminais sem seus argumentos.

Sentença Principal

1. sent_principal --> pre_locuçao1, propos_principal, final.
2. sent_principal --> pron_interrog_relat1, propos_principal, final.
3. sent_principal --> propos_principal, final.

Sintagmas

4. propos_principal --> núcleo, compls, continuação.

5. núcleo --> argumento, neg, verbo.
6. núcleo --> argumento, neg, verbo, advg_g.
7. núcleo --> verbo, pron_pessoal.
8. núcleo --> verbo.

Negação

9. neg --> [não], {negativa}.
10. neg --> [neg].
11. neg --> [ ].

Colocação do sujeito no início da sentença

12. pron_interrog_relat1 --> pron_interrog_relat2.
13. pron_interrog_relat1, [núcleo], [mov_argumento] -->
pron_interrog_relat2, argumento, núcleo.

Geração do sujeito, artigo interrogativo e pronome relativo

14. pron_interrog_relat2, [argumento] --> pron.
15. pron_interrog_relat2, [caso], [art] --> caso, art_interrog.
16. pron_interrog_relat2, [caso], [núcleo_da_sentença], [mov_argumento] -->
caso, pron, marca_p_art_def, núcleo_da_sentença.

Complementos

17. compls --> [mov_argumento], comps.
18. compls --> [ ].
19. compls --> compls, argumento.

Inversão do sujeito e conjunções coordenativas

20. argumento, [neg], [ve] --> neg, ve, marca_de_inversão, argumento.
21. argumento, [neg], [verbo] --> neg, verbo, marca_de_inversão, argumento.

Argumentos

22. argumento --> caso, sint_nominal.

Sintagmas nominais

23. sint_nominal --> nomes_próprios.
24. sint_nominal --> art_def, nomes_próprios.
25. sint_nominal --> núcleo_da_sentença, compls, oraç_relativa.

26. núcleo_da_sentenca --> art, adjs, nome_comum, adj_g.

Orações relativas do tipo restritivo

27. oraç_relativa --> pron_interrog_relat1, propos_principal, oraç_relativa.
28. oraç_relativa --> [ ].

Disjunção ou conjunção de nomes próprios

29. nomes_próprios --> nps.
30. nps --> art_def, nps.
31. nps --> [P], {np}.
32. nps --> [ ]

Adjetivos e grupos de adjetivos

33. adjs --> [ ].
34. adjs --> adj, compls, adj_g.

35. adj_g --> [ ].
36. adj_g --> adj, compls, adj_g.

Grupos de advérbios

37. advg_g --> adv_g.
38. advg_g, [compls] --> compls, adv_g.
39. advg_g --> advérbio, núcleo_da_sentença, compls.

Elipses e geração de um artigo, conforme o caso

40. marca_de_inversão, [caso(_), nenhum] --> nenhum.
41. marca_de_inversão --> [ ].
42. marca_p_art_def, [marca_p_art_def] --> [ ].
43. caso(Caso) --> [caso(Caso)].
44. caso(suj), [nenhum] --> nenhum.
45. caso(suj), [nem_todo] --> nem_todo.
46. caso(suj) --> [ ].
47. caso(obj_dir), [art_indef] --> [ ].
48. caso(obj_dir) --> [ ].
49. caso(pred_suj), [art] --> [ ].
50. caso(pred_suj), [art] --> ind_art.
51. caso(pred_suj) --> [ ].
52. caso(prep(P)) --> contração, [P], {prep(P)}.

Pronomes relativos e interrogativos

53. pron --> prons_interrog.
54. pron --> pron_interrog.
55. pron --> art_interrog_cat.
56. pron --> pron_relat.

57. prons_interrog, [ve] --> (pron_3_r_i; pron1_r_i), ve.
58. prons_interrog, [ve] --> pron_3_r_i; pron1_r_i.

O fragmento da sintaxe, considerado aqui, mostra somente alguns aspectos essenciais do discurso do usuário. As observações que faremos, a seguir, referem-se a este fragmento. Faremos referências à numeração dada às regras. Explanaremos com maiores detalhes outras partes da gramática que ficaram fora deste subconjunto, tais como artigos e certos verbos especiais.

As regras, que são do tipo *formação,* caracterizam somente uma estrutura (diagrama de árvore), com somente um significado por vez e operam sobre símbolos simples. Algumas destas regras são mais complexas. Nestes casos serão chamadas de *regras de formação complexa.* Elas operam sobre objetos mais complexos, e executam funções similares àquelas realizadas por regras transformacionais, numa gramática transformacional (v. item 2.2).

As regras de formação complexa são de quatro tipos:

(1) Regras de *apagamento* - apagam certos elementos da entrada.
(2) Regras de *substituição* - substituem um elemento por outro.
(3) Regras de *inserção* - inserem um elemento adicional.
(4) Regras de *movimento* - trocam a ordem dos elementos.

A seguir, explicaremos as regras gramaticais fazendo um apanhado dos aspectos sintáticos e semânticos dos seguintes constituintes: *sentença principal, sintagmas, perguntas abertas, orações relativas e artigos*.

(1) Sentença principal

As regras de formação 1,2 e 3 (*sent_principal*) estabelecem que uma sentença pode ser inicializada por uma expressão introdutória (*pre_locução1*), como as sentenças iniciadas por *"Diga-me ..."*, ou então por uma classe de palavras interrogativas e relativas (*pron_interrog_relat1*) ou pela proposição principal (*propos_principal*).

Estes três casos correspondem a três tipos de sentenças:

1. Perguntas que não vêm sob forma interrogativa.
2. Peguntas abertas.
3. Perguntas fechadas, afirmações ou comandos.

Cada tipo de sentença está rotulado por uma **indicação** que aparece na estrutura lógica. Isto facilita o tratamento futuro das perguntas para gerar respostas, armazenar informações, etc. No programa ARQUIVOS aparecem as seguintes **indicações**:

"pergunta" - para perguntas que não estiverem escritas na forma interrogativa ou para perguntas fechadas (que exigem respostas do tipo **sim/não).**

"pergunta(conj)" - para conjunção de duas perguntas.

"quantos" - para perguntas fechadas iniciadas pelo "artigo" interrogativo "quantos", opcionalmente precedido por um pronome, seguido pelo verbo "ser" ou "estar".

"qual" - é usado para indicar perguntas iniciadas pelo pronome interrogativo "qual" ou "quem", seguido pelo verbo "ser" ou "estar, e por mais outro pronome interrogativo e verbo.

"fato" - para uma afirmação simples.

"fato(conj)" - para a conjunção de duas afirmações.

"**ordem**" - usado quando a sentença for um comando.

Uma sentença é sempre formada por uma proposição principal *(propos_principal)* e uma finalização *(final)*. O predicado *"final"* é responsável por detectar a pontuação final da sentença e ainda para anotar a correspondente **indicação** da estrutura da sentença. Cada indicação, como foi visto acima, refere-se a uma palavra chave, que aparece na estrutura lógica, para caracterizar o tipo de sentença do usuário.

(2) Proposição principal

As regras de números 4, 5, 6, 7 e 8 estabelecem que a proposição principal é constituída de dua partes: **núcleo** *(núcleo)* e **complementos** *(compls)*, e que pode ainda ser continuada por uma outra proposição, separada por uma vírgula ou por uma conjunção. Caso isto ocorra, será reconhecido pelo predicado *"continuação"*.

O núcleo da proposição principal é composto pelo sujeito *(argumento)*, opcionalmente seguido por uma negação *(neg)*, e por um verbo *(verbo)*. Entretanto o núcleo pode ser formado de outras maneiras.

A regra de número 6 define uma proposição particular, onde aparece um grupo adverbial, como nas seguinte sentenças:

"O diretório DIR1 tem como subdiretório SUBDIR1."
"O diretório DIR1 tem SUBDIR1 como subdiretório."

As regras 7 e 8 definem o núcleo de comandos típicos, como por exemplo:

"Dê-me os arquivos do diretório DIR1."

(3) Perguntas abertas e orações relativas

As perguntas abertas (exigem uma resposta mais elaborada que um simples SIM/NÃO) envolvem regras de movimento que são capazes de transferir palavras ou sintagmas de uma posição para outra, dentro da sentença. Elas são caracterizadas pela interação entre a inversão de argumentos e a presença de uma palavra que inicia a frase interrogativa.

As orações relativas têm muitos pontos em comum com as perguntas abertas. Portanto, é possível tirar conclusões que são válidas para ambos os tipos de orações. Por exemplo, os pronomes relativos (com exceção do pronome *"cujo"*), são os mesmos que os pronomes interrogativos. . O pronome interrogativo aparece na posição inicial, e há sempre um sintagma nominal que falta (exceto quando o pronome interrogativo é o sujeito da sentença). Também nas orações relativas normalmente falta um sintagma nominal.

A análise das perguntas abertas é realizada pela regra de número 2, associada com as regras 12 e 13 (referentes a pronomes interrogativos ou relativos). As regras 12 e 13 também são usadas para a análise das orações relativas, neste caso, vêm associadas com as regras 27 e 28.

(4) Artigos

A semântica de um artigo é especificada por uma regra gramatical como:

```
artigo(Conc-D-X, O1, O2 para([X,D], O3, O4 )) --> artigo.
```

No lado esquerdo da regra a variável X representa o sujeito, Conc e D, representam, respectivamente, concordância e domínio. O1 e O2 correspondem às propriedades referentes ao sujeito e ao predicado. O3 e O4 são as fórmulas lógicas do nosso sistema. Elas contém O1 e O2. A cardinalidade definida pelo artigo é representada pela variável lógica O4.

O lado direito da regra consiste de um termo não-terminal ou um terminal, opcionalmente seguido por uma seqüência de condições extras (chamadas a outros predicados do programa). Este termo não-terminal acessa as formas terminais.

Por exemplo, o artigo "**o**" é descrito pelas seguintes regras:

```
art((G-sin)-D-X, O1, O2, para([X,D], O1,
                        se(card(X,igual,1), O2))) - - > art_def(G-sin).
art_def(mas-sin) - - > [o].
```

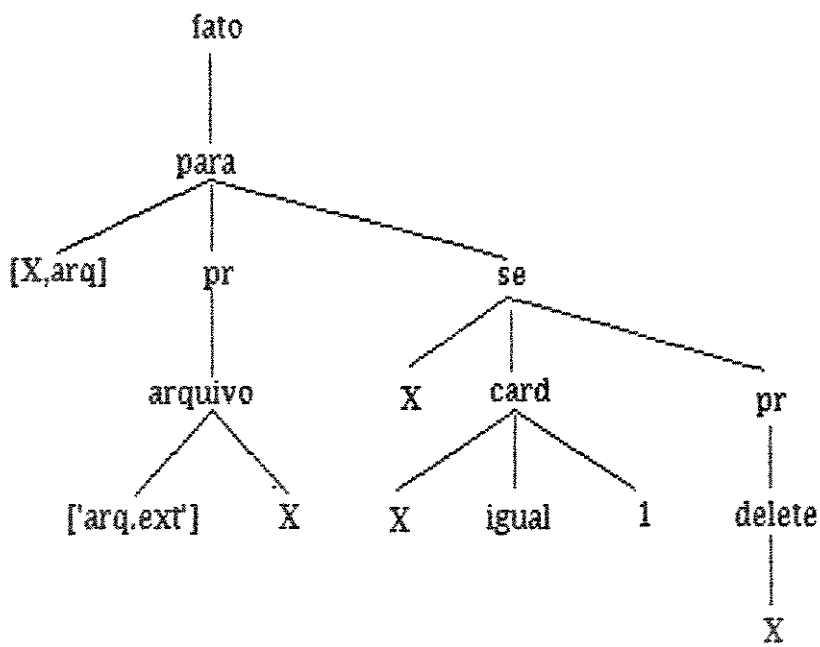


Fig. 4.2 - Estrutura lógica da sentença *"Apague o arquivo 'arq.ext' "*

Uma sentença contento o artigo "**o**", como por exemplo: *"Apague o arquivo 'arq,ext' "* terá a seguinte estrutura lógica (representada também pela árvore da fig. 4.2):

```
fato(
para( [X,arq],
        pr(arquivo(['arq.ext'],X)),
        se(card(X, igual, 1),
            pr(delete(X)))).
```

4.5 - Saída

O módulo *"saída"* manipula a formação de respostas, de observações ou mesmo de perguntas que devam ser mostradas ao usuário. Ele chama o módulo *"base de conhecimentos"* para avaliar a estrutura lógica e, de acordo com o que foi processado, produz a resposta em um formato adequado. A *tabela 4.1* mostra os tipos de respostas fornecidas pelo programa.

SENTENÇA	RESPOSTA
pergunta aberta	lista de itens de dados ou um simples dado, precedidos por uma preposição (se houver preposição na pergunta).
pergunta fechada ou afirmação	mensagem do tipo: SIM/NÃO.
comando	ação seguida por uma mensagem

Tabela 4.1 - Tipos de respostas fornecidas pelo programa

Os três aspectos principais do processamento das respostas, dentro dos objetivos do programa ARQUIVOS são os seguintes:

(1) Acesso à base de dados para encontrar e verificar os itens de dados relativos à pergunta do usuário, ou seja, o aspecto de *recuperação*. Este processo se encontra implementado no módulo *"base de conhecimentos"*.

(2) Avaliação da estrutura lógica correspondente à frase de entrada, sobre uma lógica trivalente (verdadeiro, falso e indefinido). Este aspecto também é tratado pelo módulo *"base de conhecimentos"*.

(3) Formação da resposta, escolhendo uma forma padrão adequada, e inserindo dentro dela os itens de dados selecionados. Este aspecto é tratado pelo módulo *"saída"*.

A figura 4.3 ilustra a interação do usuário **(USR>)** com o programa ARQUIVOS **(ARQ>)**, mostrando algumas das possíveis consultas do usuário com as respectivas respostas do programa.

```
USR>  Quantos arquivos existem no diretório DIR2?
ARQ>  12 arquivos.

USR>  Quais são os arquivos executáveis do diretório DIR3?
ARQ>  FAZ1.EXE
      FAZ2.COM

USR>  Apague o arquivo ARQ1.DOC.
ARQ>  O arquvio ARQ1.DOC foi apagado.

USR>  Quando foi realizada a última atualização do arquivo ARQ3.DOC?
ARQ>  Em 10.12.90 - às 15:00 hs.
```

Figura 4.3 - Perguntas do usuário (USR) e respostas do programa ARQUIVOS (ARQ)

4.6 - Avaliação da Estrutura Lógica

A compreensão de uma sentença é a computação de um valor-verdade através da avaliação de sua estrutura lógica sobre uma base de conhecimentos constituída de *fatos* e de *regras*, de acordo com um sistema lógico, baseado na Lógica de Predicados.

A compreensão de uma pergunta consiste em encontrar instâncias da frase declarativa correspondente, que sejam verdades. A estrutura lógica opera como um programa cujo significado é dado por regras de interpretação. A base de dados descreve uma certa situação em que o valor da estrutura lógica, combinado com os itens dos dados recuperados, fornece a resposta.

O sistema lógico adotado difere da Lógica de Primeira Ordem pelo fato de que seus quantificadores são mais ajustados à semântica da linguagem natural; as variáveis podem denotar indivíduos e conjuntos e há um terceiro valor-verdade, o valor *indefinido*, além dos valores *verdadeiro* e *falso*.

4.6.1 - Aspectos da Recuperação e Avaliação

A tarefa de recuperação dos dados e de avaliação é realizada pelo predicado *"encontra_todo"* do módulo *"base de conhecimentos"*. Este predicado encontra todos os elementos que satisfazem certas restrições e elimina as redundâncias. Requer uma descrição dos elementos que são os objetos da busca, e uma indicação da área a ser pesquisada.

A descrição dos elementos já está disponível na estrutura lógica. A área também já vem com restrições determinadas pelos argumentos referentes ao *domínio* e *à cardinalidade*.

A informação da cardinalidade dos artigos, presente em sua definição, permite a implementação dos testes de compatibilidade. Estes testes evitam o cálculo da cardinalidade e realizam comparações entre a cardinalicade da lista de elementos em construção e a cardinalidade esperada, toda vez que um novo elemento é encontrado.

CONCLUSÃO

O estudo sobre processamento de linguagem natural permitiu a aquisição de conhecimentos e a aprendizagem de técnicas que são básicas para nossos objetivos de projetar ferramentas, com as quais os usuários de um sistema de conhecimento possam construir interfaces, facilitando sua interação com tais sistemas.

Para chegar à abordagem que adotamos na construção do sistema ARQUIVOS, partimos de uma conceituação generalizada de interfaces homem-máquina, concluindo que, no projeto de interfaces deveríamos trabalhar em duas frentes: processamento de linguagem natural e representação do domínio de conhecimento.

Quanto ao processamento da linguagem do usuário, optamos pelo formalismo lógico, o qual se mostrou muito adequado, permitindo sua implementação através do *PROLOG* e das *gramáticas de cláusulas definidas*, que, historicamente foram criados para suportar o processamento lingüístico. Usamos também o *PROLOG* para a representação do domínio, ou seja, para construir a base de conhecimentos sobre os arquivos em discos de um microcomputador PC.

Ficou evidente a necessidade de se acoplar o processamento lingüístico com a representação do conhecimento, tendo-se em vista que os aspectos semântico e pragmático, inerentes à linguagem natural, tendem a ser resolvidos através de uma representação mais abrangente e completa do contexto onde se desenvolveria a interação usuário-computador.

O trabalho realizado nos levou a um discernimento maior sobre as questões envolvidas na construção de interfaces para sistemas especialistas. Isto permitiu que se adquirisse uma certa habilidade no que diz respeito à especificação e projeto de interfaces em linguagem natural, abrindo caminho para que futuramente se possa incluir tais interfaces em um ambiente integrado de projetos.

Uma continuação poderia ser feita em várias direções, como por exemplo, através da criação de recursos que permitissem ao usuário incrementar módulos como *o dicionário, a gramática e a base de conhecimentos,* de modo interativo, sem a intermediação do programador.

Seria também interessante fornecer ao sistema a possibilidade de tratar com os aspectos pragmáticos, ou seja, a capacidade de levar em conta o contexto no qual se realiza o diálogo entre o usuário e o sistema de conhecimento. Isto envolveria, não só o processamento da linguagem natural, mas também uma teoria de representação do conhecimento.

A interface em linguagem natural entre usuários e sistemas de conhecimentos, assim como foi desenvolvida aqui, parece ser muito adequada atualmente, pois várias inovações tecnológicas, como o aumento da velocidade e de capacidade de memória dos sistemas de computação, vêm a solucionar problemas que até então inviabilizavam o uso destas interfaces, pela necessidade de um grande espaço para armazenar, por exemplo, o dicionário e a base de conhecimentos e de acesso rápido a estas informações armazenadas.

Já existem placas de circuito, disponíveis comercialmente, que acopladas a um microcomputador, funcionam como transdutores de sinais de voz para cadeias de caracteres, substituindo o teclado, como periférico de entrada. Isto aponta para a possibilidade de se usar comandos de voz ao invés de comandos digitados. Neste caso as interfaces em linguagem natural serão necessárias, visto que aqueles comandos estarão muito mais próximos da linguagem natural que os digitados.

APÊNDICE A

A.1 - Gramáticas Livres de Contexto [PEREIRA 87]

Constituem um sistema para definir as expressões de uma linguagem em termos de regras, que são equações recursivas sobre tipos de expressões, chamadas de *não-terminais*, e expressões primitivas, chamadas de *terminais*.

A notação padrão para uma gramática livre de contexto é a seguinte:

$\mathbf{N_0} \rightarrow \mathbf{V_1} \ldots \mathbf{V_n}$ onde: $\mathbf{N_0}$ = não-terminal
$\mathbf{V_i}$ = terminais e não-terminais

Uma regra assim tem a seguinte interpretação:

Se expressões $\mathbf{w_1}, \ldots, \mathbf{w_n}$ unificam-se com expressões $\mathbf{V_1}, \ldots, \mathbf{V_n}$, respectivamente, então a expressão simples $\mathbf{w_1}, \ldots, \mathbf{w_n}$ (uma concatenação de **wi**) é própria do tipo de expressão $\mathbf{N_0}$.

Dizer que uma expressão $\mathbf{w_i}$ se unifica como uma do tipo $\mathbf{V_i}$, significa que, ou **Vi** é um terminal e é idêntico a $\mathbf{w_i}$, ou $\mathbf{V_i}$ é um não-terminal e $\mathbf{w_i}$ é daquele tipo (em virtude de alguma regra da gramática).

Consideremos o exemplo de uma gramática livre de contexto para um fragmento da língua portuguesa, onde um termo *não-terminal* inicia-se com letra maiúscula e um *terminal*, com letra minúscula:

S –> **SN SV**
SN –> **Art Subst_Comum Oração_Relativa**
SN –> **Art Subst_Próprio**
Oração_Relativa –> **[ ]**
Oração_Relativa –> *que* **SV**

SV --> VT SN
SV --> VI

Subst_Próprio --> *maria*
Subst_Próprio --> *joão*
Subst_Comum --> *homem*
Art --> *um*
Art --> *o*
VI --> *vive*
VT --> *ama*

Abreviações:

S	- sentença
SN	- sintagma nominal
SV	- sintagma verbal
VI	- verbo intransitivo
VT	- verbo transitivo
Art	- artigo

Esta classificação de uma expressão e suas subexpressões de acordo com uma gramática livre de contexto pode ser resumida em uma *árvore de estrutura de frase* ou *árvore de análise*. A árvore de análise correspondente à sentença: *"O joão vive"* é dada pela figura A.1.

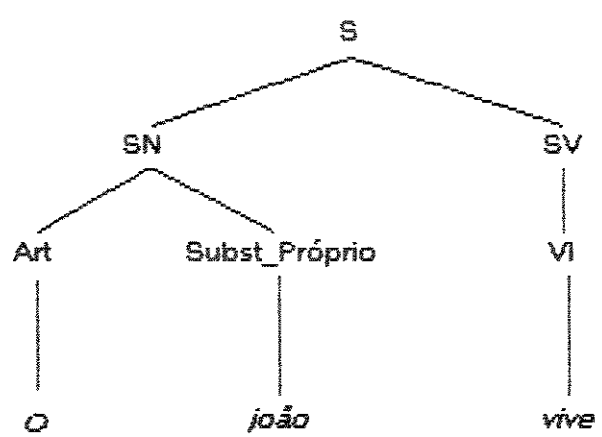


Fig. A.1 - Árvore de análise da sentença: *"O joão vive"*

Cada conjunto de nós, consistindo de um *pai* e de seus *filhos*, corresponde à aplicação de uma regra.

Por exemplo, o conjunto de nós, no topo da árvore, corresponde à aplicação da regra **S –> SN SV.**

As *folhas* da árvore, isto é, os nós sem filhos, correspondem às expressões primitivas (os símbolos terminais) e os nós no interior da árvore correspondem aos símbolos não-terminais.

Qualquer procedimento para determinar a árvores de análise correpondente a uma expressão deverá executar as aplicações de regras em uma dada ordem. Um tal ordenamento para as aplicações é denominado de *derivação*. Temos as derivações *descententes* e as derivações *ascendentes*.

É tradicional separar uma gramática em duas partes: uma contendo as regras gramaticais propriamente ditas e outra contendo as regras que possuem um simples terminal em seu lado direito. Esta última parte é denominada de ***dicionário*** da gramática. As regras do dicionário correspondem a linhas na árvores de análise que ligam um *não-terminal* (imediatamente antecedente a um *terminal*) ao terminal que faz parte desta mesma regra.

A.1.1 - Formalizando as Gramáticas Livres de Contexto

Em árvores de análise, como aquela da fig. A.2, os símbolos não-terminais podem ser interpretados não somente como uma classificação de expressões, mas também como uma relação binária sobre posições na expressão, onde uma posição divide uma expressão em duas subexpressões, que concatenadas formam a expressão original. Por exemplo, a posição 2 (fig. A.2) divide a expressão em duas subexpressões: *"um homem"* e *"vive"*.

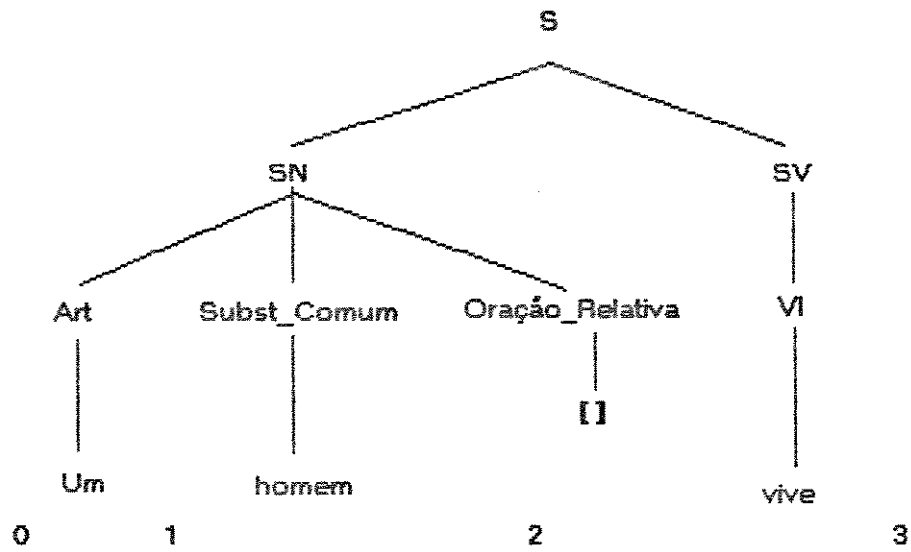


Fig. A.2 - Árvore de análise incluindo marcas de posição

Os símbolos *não-terminais* podem ser vistos como relações binárias sobre as posições. O par de posições < 0 , 2 > está na relação **SN** porque o *não-terminal* **SN** está sobre as subexpressões entre as posições 0 e 2. E, usando-se notação lógica, pode-se escrever esta relação como **SN(0,2).** Analogamente, **S(0,3)** faz sentido porque o *não-terminal* **S** está sobre a expressão entre as posições 0 e 3. A oração relativa (opcional) abrange o trecho entre a posição 2 e ela própria, isto é, **Oração_Relativa(2,2).**

A regra **S –> SN SV** pode ser resumida usando relações ou posições com a seguinte declaração lógica:

$$SN(p_0, p_1) \wedge SV(p_1, p) \Rightarrow S(p_0, p)$$

Quer dizer: " Se existe um **SN** entre as posições p_0 e p_1
e um **SV** entre as posições p_1 e p
então haverá uma **S** entre p_0 e p ".

Sendo assim, toda regra livre de contexto da forma:

$$N_0 \rightarrow V_1 \ldots V_n$$

pode ser representada como:

$$V_1(p_0,p_1) \wedge \ldots \wedge V_n(p_{n-1},p) \Rightarrow N_0(p_0,p)$$

A.1.2 - Gramáticas Livres de Contexto em Prolog

Esta forma de se representar as regras tem uma correspondência direta com *PROLOG*, visto que as regras estão na forma de *cláusulas definidas*. Por exemplo a regra **S –> SN SV** poderia ser escrita, em *PROLOG* da seguinte forma:

```
s(P0, P) - - >  sn(P0, P1),  sv(P1,P).
```

Uma completa formalização de um fragamento do português poderia ser escrita em *PROLOG* da seguinte maneira:

```
s(P0,P) :- sn(P0,P1),  sv(P1,P).
sn(P0,P) :- art(P0,P1),  subst_comum(P1,P2), oração_relativa(P2,P).
sn(P0,P) :- subst_próprio(P0,P).
sv(P0,P) :- vt(P0,P1), sn(P1,_).
sv(P0,P) :- vi(P0,P).
oração_relativa(P,P).
oração_relativa(P0,P) :- liga( que, P0,P1), sv(P1,P).
subst_comum(P0,P) :- liga( homem, P0,P).
subst_próprio(P0,P) :- liga( maria).
subst_próprio(P0,P) :- liga( joão,  P0, P).
vi(P0,P)  :- liga(  vive,  P0,P).
art(P0,P) :- liga( a,  P0,P).
subst_comum(P0,P) :- liga( homem, P0,P).
vt(P0,P) :- liga( ama, P0,P).
```

Usa-se o predicado **liga(Terminal, P1,P2)**, para dizer que a variável **Terminal** (que é instanciada por uma palavra da sentença) liga as posições **P1** e **P2.**

Esta formalização de uma gramática livre de contexto em PROLOG pode ser vista como a saída de uma tradução ou mapeamento de uma gramática livre de contexto em PROLOG. Este mapeamento toma qualquer regra livre de contexto e forma uma cláusula PROLOG da seguinte maneira:

(1) Para cada *não-terminal*, constrói-se um literal aplicando um predicado binário para aquele *não-terminal* com dois argumentos de posição. Por exemplo, o *não-terminal* NP torna-se **np(P1,P2).**

(2) Para cada *terminal*, constrói-se um literal aplicando o predicado "**liga**" com três argumentos: o símbolo terminal expresso como uma constante Prolog e dois argumentos de posição. Por exemplo, o *terminal* "**vive**" torna-se o literal **liga(*viva*, P1, P2).**

Este recurso de fazer o mapeamento em forma de algoritmo é a base para interpretadores neste formalismo gramatical.

A.2 - Gramáticas de Cláusulas Definidas (GCD) [PEREIRA 87]

O método de se traduzir gramáticas livres de contexto em cláusulas de Horn, descrito acima, pode ser usado como a base para uma extensão dessas gramáticas: as **gramáticas de cláusulas definidas.**

A fórmula geral da cláusula definida associada a uma regra de gramática livre de contexto é a seguinte:

$$N_0 \rightarrow V_1 \ldots V_n$$

que, na notação PROLOG, fica:

```
n0(P0,P) :- v1(P0,P1), ... , vn(Pn-1,P).
```

Podemos generalizar este axioma incluindo-se argumentos extras, além dos dois argumentos de posição. Por exemplo, supondo-se que desejamos distinguir o *número* (singular ou plural) dos sintagmas nominais e verbais, para que seja garantida a concordância em número. Para esta finalidade poderíamos estender a formalização da gramática e do dicionário com um argumento extra, para representar o *número*. A gramática ficaria assim:

```
s(P0,P) :-
        sn(Número, P0,P1),
        sv(Número, P1,P).
sn(Número, P0,P) :-
        subst_próprio(Número, P0,P).
sv(Número, P0,P) :-
        vt(Número, P0,P1),
        sn(_,P1,P).
sv(Número, P0,P) :-
        vi(Número, P0,P).
subst_próprio(singular, P0, P) :- liga(joão, P0,P).
vi(singular, P0, P) :- liga( vive, P0, P).
vi(plural, P0, P) :- liga( param, P0, P).
vt(singular, P0, P ) :- liga(ama, P0, P).
vt(plural, P0, P) :- liga( escrevem, P0, P).
```

A primeira regra nesta gramática diz que **s** (sentença) pode ser um **sn** (sintagma nominal), com *número* de valor **Número** seguido por um **sv** (sintagma verbal) com o mesmo *número* **Número**.

O uso de uma variável anônima (_) é um recurso utilizado para ignorar o número do objeto direto.

Esta gramática admite a sentença *"O joão vive"* mas não *"O joão param"*, mesmo que ambos os verbos sejam intransitivos.

Assim como as cláusula com predicado de dois argumentos podem servir para codificar uma das gramáticas livres de contexto, temos as cláusulas com predicados de múltiplos argumentos, representando uma generalização destas gramáticas.

As gramáticas livres de contexto diferem das GCD da mesma forma que a codificação estendida das cláusulas de *Horn* difere da codificação de predicados com dois argumentos.

O significado de uma regra das GCD é dado através da tradução da regra em uma cláusula definida, usando o mesmo mapeamento das regras livres de contexto, exceto que agora um termo *não-terminal* de **n** argumentos é traduzido para um literal de **n+2** argumentos, no qual os dois argumentos finais representam posições dentro da cadeia de palavras da sentença.

As GCD são tão úteis que no sistema PROLOG a tradução é feita automaticamente, sem que o programador tenha que executar todos os passos descritos acima.

A notação Prolog para se escrever uma regra GCD é a seguinte:

- Predicados e símbolos funcionais, variáveis e constantes obedecem à sintaxe normal do Prolog.
- Os símbolos adjacentes do lado direito de uma regra GCD são seguidos pelo operador " , " , como os literais em uma cláusula.
- Em uma regra GCD, a seta é representada pelo sinal " - - > ".
- Os símbolos terminais são escritos dentro de colchetes " [ ] ".
- A ausência de palavras é representada pela lista vazia.

A gramática que formalizamos acima, poderia ser codificada diretamente, usando-se a notação própria para as GCD:

```
s - - > sn(Número), sv(Número).
sn(Número,) - - > subst_próprio(Número).
sv(Número) - - > vt(Número), sn(_).
sv(Número) - - > vi(Número).
subst_próprio(singular) - - > liga(joão).
vi(singular) - - > liga( vive).
vi(plural) - - > liga( param).
vt(singular) - - > liga(ama).
vt(plural) - - > liga( escrevem).
```

O sistema Prolog realiza a tradução das GCD para cláusulas PROLOG no momento em que lê o programa, armazenando as cláusulas, internamente, na forma expandida. Conseqüentemente uma consulta feita à GCD terá a mesma resposta que uma consulta feita à versão expandida das regras.

A ligação entre GCD e PROLOG é fechada. No entanto é preciso não se esquecer que as GCD são uma linguagem formal, independente de sua implementação em PROLOG, assim como as cláusulas de *Horn* são independentes de sua instanciação em programas PROLOG.

É importante notar que, como o PROLOG é uma implementação incompleta da prova de teoremas das cláusulas de *Horn*, assim também a notação GCD do PROLOG é uma implementação incompleta das GCD, do ponto de vista teórico.

A.2.1 - Chamadas a predicados PROLOG, embutidas nas GCD

O formalismo abstrato das GCD expande as gramáticas livres de contexto, permitindo que os símbolos *não-terminais* tomem argumentos extras, que permitem a passagem de informações entre subfrases. A notação Prolog para as GCD vai além, fornecendo um mecanismo para se especificar arbitrariamente computações sobre variáveis lógicas através da execução direta de predicados

PROLOG. Estes predicados podem ser misturados com símbolos *terminais* e *não-terminais* do lado direito de uma regra GCD. Eles são distinguidos dos elementos gramaticais por virem escritos entre duas *chaves*.

Simplificando o Dicionário

Com o objetivo de simplificar a representação do dicionário, adicionam-se chamadas a predicados Prolog às GCD. Ao invés de representar o dicionário através de regras separadas para cada item, como:

```
substantivo --> [programa].
substantivo --> [professor].
substantivo --> [livro].
```

fica mais simples e menos redundante ter uma única regra

```
substantivo --> [Palavra], {nome(Palavra)}.
```

significando que o símbolo não-terminal **substantivo** pode abranger qualquer símbolo terminal que estiver em **nome**. Para acompanhar esta regra necessita-se de um dicionário do tipo:

```
nome(programa).
nome(professor).
nome(livro).
```

A utilidade desta técnica pode ser avaliada no contexto em que itens léxicos são associados a argumentos extras. Se um argumento é diretamente computável a partir da própria palavra, a entrada léxica pode realizar a computação e a entrada do dicionário não precisa dar um valor para o argumento. Este é o caso para as árvores de análise associadas a símbolos terminais.

Assim, para gramáticas de construção de árvores de análise, a entrada léxica pode ficar assim:

```
substantivo(nome(Palavra)) --> [Palavra], {nome(Palavra)}.
```

E para os argumentos extras, que são referidos às palavras, por exemplo, o número gramatical, a entrada vai conter essa informação na forma de tabela:

```
substantivo(Número) --> [Palavra], {nome(Palavra,Número)}.

nome(programa, singular).
nome(programas, plural).
nome(professor, singular).
nome(livros, plural).
```

BIBLIOGRAFIA

[ALLEN 87] Allen, J. *Natural Language Understanding*, Benjamin Cummings, Menlo Park, 1987.

[BRANCO 87] Branco, A. C. S. "Semântica para Processamento de Linguagem Natural", Tese de Mestrado, IME, 1987.

[CHOMSKY 57] Chomsky, N. *Syntactic Structures*, The Hague, Mouton, 1957.

[CHOMSKY 65] Chomsky, N. *Aspects of the Theory of Syntax*, MIT Press, 1965

[CLOCKSIN 81] Clocksin, W. F. and C. Mellish. *Programming in Prolog*, Germany: Spring-Verlag, Berlin, 1981.

[COELHO 79] Coelho, H. "A Program Conversing in Portuguese Providing a Library Service", Ph.D. Thesis, Univ. of Edinburgh, 1979.

[COLMERAUER 77] Colmerauer, A. *An Interesting Subset of Natural Language*, Universitè d'Aix-Marseille, G.I.A., 1977

[COLMERAUER 78] Colmerauer, A. "Metamorphosis Grammars", in L. Bolc (eds.), *Natural Language Communication with Computers*, pp 122-88, Springer Verlag, Berlin, 1978

[FILLMORE 68] Fillmore, C. J. "The Case for Case", in E. Bach and R. Harms (eds.), *Universals in Linguistic Theory* , pp. 1-88, Holt, Rinehart & Winston: New York, 1968.

[GADZAR 89] Gadzar, G. and C. Mellish. *Natural Language Processing in Prolog*, Addison-Wesley, Reading, MA, 1989.

[GERSHMAN 85] Gershman, A. and Wolf T.C. "Management of User Expectations in a Conversational Advisory System", *Proc. IEEE Fifth Conference on Artificial Intelligence Applications*, pp. 328-335, 1985.

[HARRIS 77] Harris, L. "User Oriented Database Query with the ROBOT Natural Language Query System", *Int'l J. Man-Machine Studies*, Vol. 9, pp.697-713, 1977.

[MIRANDA 86] Miranda, C. "Processamento de Linguagem Natural", Tese de Mestrado, IME, 1986.

[MONTAGUE 74] Montague, R. *"Formal Philosophy"*, in R. Thomason (Ed.), New Haven, Yale University, London, 1974.

[PEREIRA 80] Pereira, F. C. N. and Warren, D. H. D. "Definite Clause Grammars for Language Analysis - A Survey of the Formalism and a Comparasion with Augmented Transition Network", *Artificial Intelligence*, 13, pp. 231-278, 1980.

[PEREIRA 83] Pereira, F. C. N. "Logic for Natural Language Analysis", SRI Technical Note 275, Ph. D. Thesis, SRI International, Menlo Park, CA,1983.

[PEREIRA 87] Pereira, F. C. N. and Shieber, Stuart M. *Prolog and Natural-Language Analysis*, CSLI Lecture Notes, 10, Chicago University Press, Stanford, 1987.

[RICH 84] Rich, E. A. "Natural Language Interfaces", *Computer*, vol. 17, no. 9, pp. 35-50, 1984.

[SAVADOVSKY 88] Savadovsky, P. *A Construção de Interpretadores para Linguagem Natural*, EBAI, 1988.

[SCHANK 75] Schank R.C. *Conceptual Information Processing*, North-Holland, Amsterdam, 1975.

[SCHANK 80] Schank, R.C. and Birnbaum, L. "Memory, Meaning and Syntax", Research Report nº 189, Yale University, London, 1980.

[TARSKI 44] Tarski, A. " The Semantic Conception of Truth and the Foundations of Semantics", *Philosophy and Phenomenological Research* 4, 341-375, 1944.

[WINSTON 84] Winston, P. H. and Brown, R. H. *Artificial Intelligence*, Addison-Wesley, 1984.

[WOODS 70] Woods, W. A. "Transition Networks Grammars for Natural Language Analysis", *Communication of ACM*, 13, 591-606, 1970 .